www.elektor-magazine.de

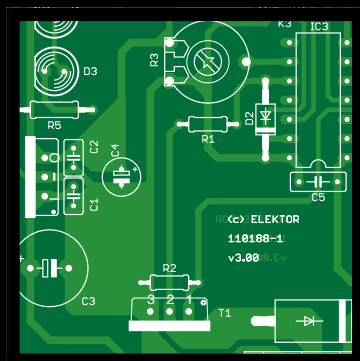
.magazine

Juli/August 2013 | Nr. 511/512

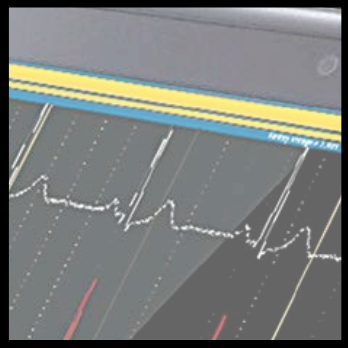
elektor

MIT EXTRA VIELEN PROJEKTEN!

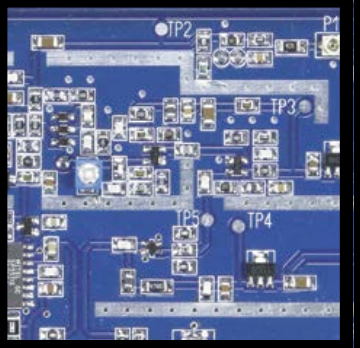
LIGHTSHOW CONTROLLER FÜR LED-STRIPS



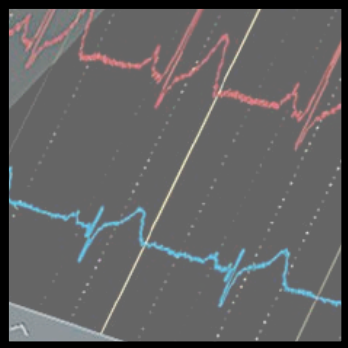
ELEKTORBUS SCHRITTMOTOR TREIBER

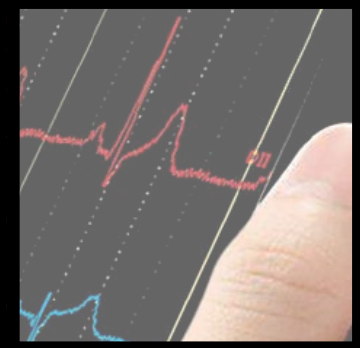


ELEKTOR ANDROID EKG

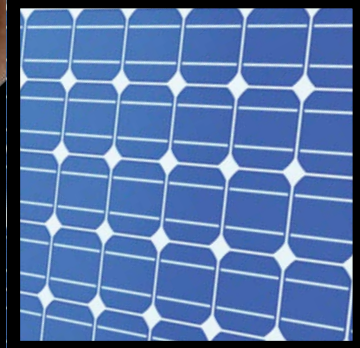


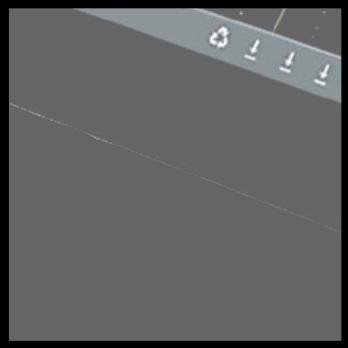
430 MHZ FM AUDIO-SENDER





132 SEITEN DOPPEL-AUSGABE





SMARTPHONE A/V REMOTE CONTROL

08

4 198630 314904

(D) € 14,90

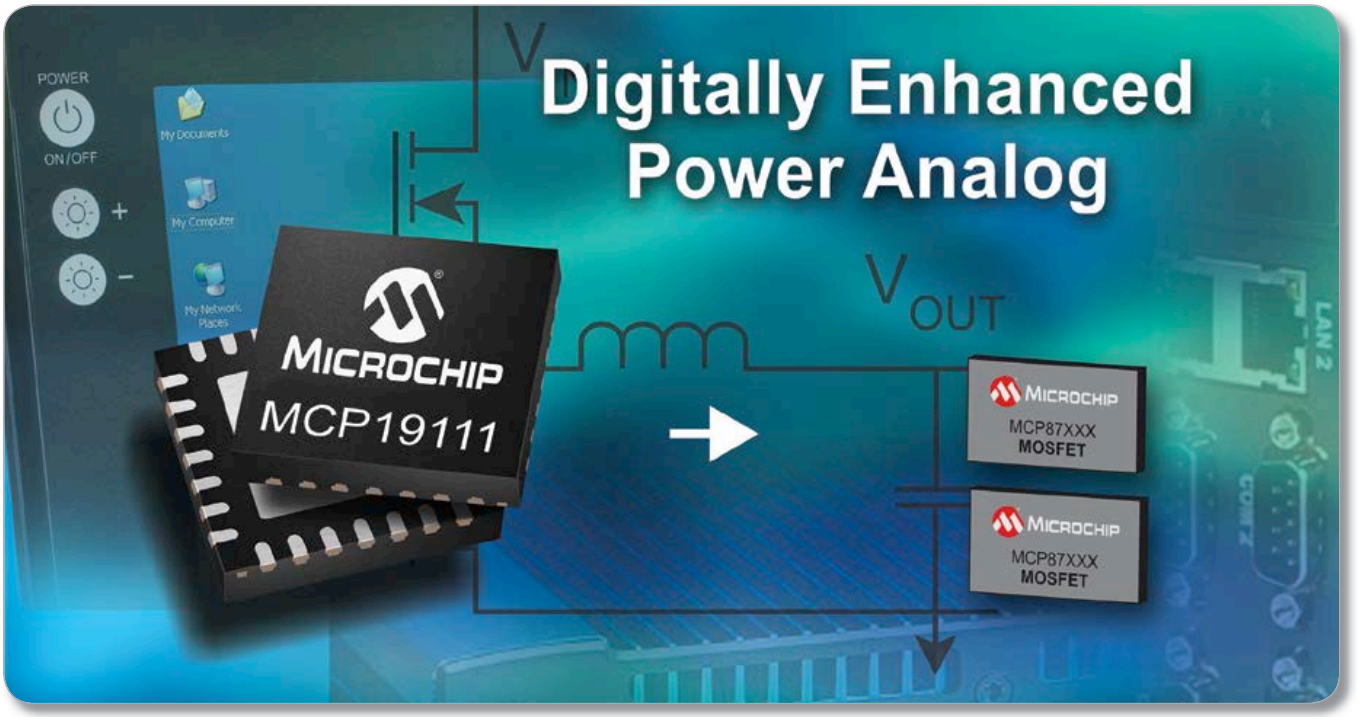
CHF 27,00

(A, B, L) € 16,00

G3078 E

Der weltweit erste digital erweiterte Analogcontroller

Neuer analog-basierter Power-Management-Controller mit integriertem Mikrocontroller



Kombinieren Sie die Flexibilität und I²C-Kommunikation digitaler DC/DC-Leistungswandlung mit der Geschwindigkeit, Leistungsfähigkeit und dem Auflösungsvermögen einer Analogregelung: mit dem neuen Power-Management-Controller MCP19111 von Microchip.

Der MCP19111 ist ein neuer hybrider Mixed-Signal-Controller, der analoges und digitales Power Management auf einem Chip vereint. Mit einem analogen PWM-Controller, einem 8-Bit PIC® Flash-Mikrocontroller und MOSFET-Treiber für synchrone Abwärtswandler, ermöglicht der MCP19111 eine konfigurierbare und hocheffiziente Leistungswandlung.

Mit dem Transientenverhalten der analogen Leistungswandlung erübrigt der MCP19111 leistungsfähige Mikrocontroller oder einen schnellen A/D-Wandler. Damit verringern sich die Kosten und der Stromverbrauch.

Um einen noch höheren Wirkungsgrad zu erzielen, kann der MCP19111 zur Ansteuerung von Microchips neuen schnellen SMPS-optimierten MOSFETs MCP87xxx verwendet werden. Die 25V-Logikpgel-MOSFETs bieten einen niedrigen FOM-Wert (Figure of Merit) mit einem Durchlasswiderstand von 1,8 bis 13 mΩ. Damit ergibt sich eine effizientere DC/DC-Leistungswandlung.

EINFACHER START IN 3 SCHRITTEN:

1. Kennenlernen des MCP19111 mit dem kostengünstigen ADM00397-Board
2. Kombination mit SMPS-optimierten MCP87xxx MOSFETs
3. Anpassung der DC/DC-Wandlung an die Anwendung

Weitere Informationen unter: www.microchip.com/get/eumcp19111

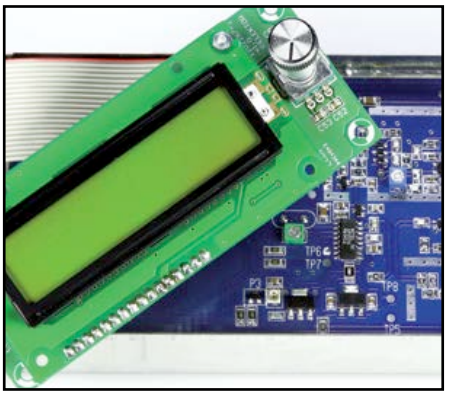
microchip DIRECT
www.microchipdirect.com

MICROCHIP

Microcontrollers • Digital Signal Controllers • Analog • Memory • Wireless

Inhalt

elektor magazine

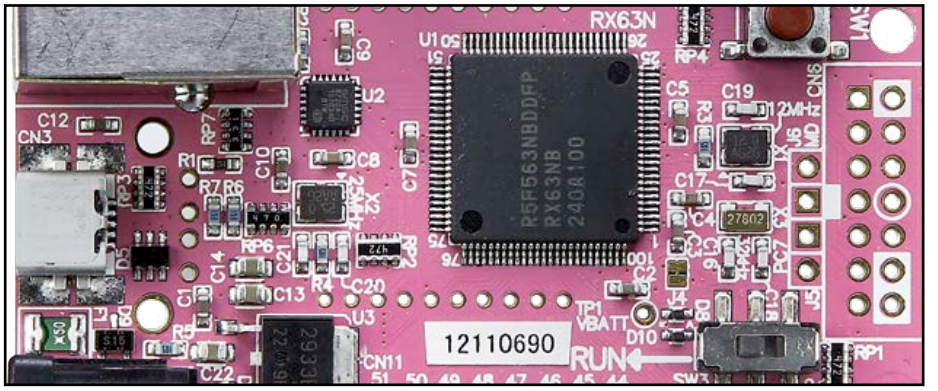


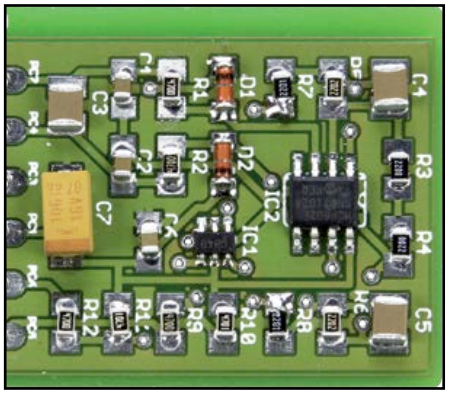
Community

Labs

Projects

44. Jahrgang – Nr. 511/512 Juli/August 2013





Impressum

44. Jahrgang, Nr. 511/512 Juli/August 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.

Hauptsitz des Verlags
Elektor International Media
Allee 1
NL-6141 AV Limbricht

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

Distribution:
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

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

EKG und Fernbedienung

Der menschliche Körper ist ja ein besonders faszinierendes Wissensgebiet. Ein Beweis hierfür ist das große Interesse unserer Leser für Projekte rund um Medizin, Gesundheit und Wellness. Vor ein paar Jahren haben wir zum Beispiel eine Schaltung veröffentlicht, mit der ein EKG aufgenommen und auf einem Gameboy dargestellt werden kann. Fast hätte ich geschrieben: „auf einem ‚Gameboy' (einer kleinen Spielkonsole von Nintendo)" – denn erinnert sich heute noch jeder daran, was ein Gameboy ist? Inzwischen gibt es ja für alles eine App. Daher finden Sie in diesem Heft das „Elektor Android EKG" – eine moderne Neuauflage des beliebten EKG-Themas.
Persönlich sehr gut gefallen – nicht nur weil ich selbst über einen stattlichen A/V-Gerätepark verfüge – hat mir auch das zweite Smartphone-Projekt in dieser Ausgabe. Per App und via Bluetooth lässt sich eine kleine Schaltung fernsteuern, die Kommandos über Infrarot an die Anlage weitergibt. Die dazu nötigen Befehlsfolgen kann man der Bridge-Schaltung über die bisher benutzten Fernbedienungsriegel beibringen; sie werden im EEPROM eines kleinen Mikrocontrollers gespeichert. Ein wirklich cleveres Projekt, bei dem wir unsere Leser wie gehabt nicht nur mit Schaltplan und Platine, sondern auch der Firmware und natürlich einer Android-App versorgen.
Endlich kommen wir auch dazu, unser Versprechen einzulösen, dass es weitere Hardware für den ElektorBus geben wird. Unser neues Board steuert bipolare Schrittmotoren bis 60 V, damit lässt sich im wahrsten Sinne des Wortes einiges bewegen. Die Schaltung und Platine wurde von Lehrkräften und Studenten am Management Center Innsbruck entwickelt. Es freut mich natürlich, dass der ElektorBus im dortigen Studiengang Mechatronik für praktische Übungen und inzwischen sogar für Bachelorarbeiten genutzt wird.
Das alles sind aber nur drei Projekte aus dem breit gefächerten Angebot in dieser extrastarken Ausgabe – schauen Sie gleich mal auf den nächsten Seiten!

Jens Nickel

Unser Team

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 · techthe future

Die Elektor-Community

Unsere Partner und Sponsoren

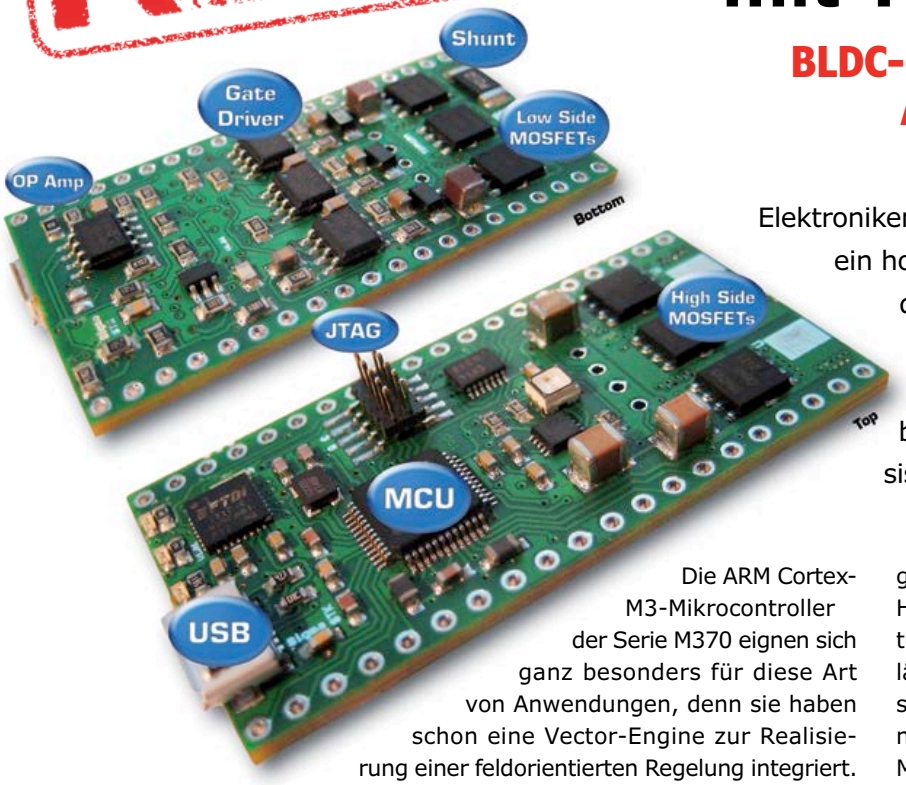
Sie möchten Partner werden?

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

NEWS

Design-Wettbewerb mit Toshibas M370!

BLDC-Motor-Controller mit M370
Attraktive Preise zu gewinnen!



Elektroniker aufgepasst: Noch bis zum 31. Juli läuft ein hochinteressanter Schaltungswettbewerb, der von Toshiba in Kooperation mit Elektor veranstaltet wird. Entwickelt werden sollen innovative Motorsteuerungen für bürstenlose Gleichstrommotoren auf Basis der M370-Mikrocontroller von Toshiba.

Die ARM Cortex-M3-Mikrocontroller der Serie M370 eignen sich ganz besonders für diese Art von Anwendungen, denn sie haben schon eine Vector-Engine zur Realisierung einer feldorientierten Regelung integriert. Damit lassen sich Motorsteuerungen konstruieren, die ganz besondere Eigenschaften aufweisen. Wir sind schon ganz gespannt auf Ihre Ideen!

Es gibt sehr interessante Sachpreise zu gewinnen:

1. **Preis: Toshiba Multimedia-Notebook der Spitzenklasse mit 15“-Display**
2. **Preis: Toshiba Tablet AT-300**
3. **Preis: 60 GB SSD PC Upgrade Kit – Interface SATA 6.0 Gbit/s, 2,5 Zoll (6,4 cm)**
4. **Preis: 8 GB Flash-Air-SD-Card**
5. **-10. Preis: 64 GB USB-Stick**

Damit die Sache eine Hardware-Basis hat, können interessierte Teilnehmer ein besonders günstiges Entwicklungsboard mit passendem Controller über Elektor beziehen. Das abgebildete „Sigma-Board“ enthält nicht nur einen 32-bit-Mikrocontroller vom Typ M373 mit 80 MHz Takt, 128 kB Flash-ROM, 6 kB RAM, 12-bit-ADC und jeder Menge 16-bit-Timern, sondern auch gleich einen FTDI-Chip für die Anbindung an einen PC via USB und genügend MOSFETs für drei Halbbrücken zur Motoransteuerung. Hinzu kommt eine hochmoderne Entwicklungsumgebung. Mit diesem Board kann man also gleich praktisch loslegen.

Hinzu kommt, dass auf dem Cortex-M3-Controller eine Firmware basierend auf FreeRTOS läuft, die in Kombination mit Toshibas kostenloser PC-Software MotorMind die Installation eines neuen BLDC-Motors extrem vereinfacht. In MotorMind werden die Motordaten (entsprechend dem Datenblatt) eingegeben und auf das Sigma-Board heruntergeladen - ohne eine Codezeile zu schreiben. Im Rahmen dieses Wettbewerbs kann man dieses Tool kostenlos nutzen und sich somit ganz auf die eigentliche Applikation konzentrieren. Der größte Teil der CPU-Power kann dann für zusätzliche Aufgaben genutzt werden.

Der Wettbewerb dauert noch bis zum 31. Juli 2013! Eingereicht werden soll eine genaue Beschreibung des Projekts nebst Dateien und Fotos. Auch ein kleines Video wäre schön.

Infos zu FreeRTOS:
www.freertos.org

Support gibt es im M370/Sigma-Forum:
http://forum.toshiba-components.com/forumdisplay.php?8-TMPM370

Anmeldung und weitere Infos:
ww.elektor-labs.com/m370-contest

elektorLive! – die 2-te

Der erste „ElektorLive!“ Seminar- und Ausstellungstag im Oktober 2012 hat den Lesern, Ausstellern und nicht zuletzt der Elektor-Crew eine Menge Spaß gemacht. Ein Grund, diese Veranstaltung am 12. Oktober 2013 zu wiederholen; in diesem Jahr haben wir uns Hanau bei Frankfurt als Veranstaltungsort ausgesucht. Wie immer gibt es eine Ausstellung, auf der man sich über neue Elektroniktrends informieren und mit Experten fachsimpeln kann. Herz des Ganzen ist wieder unser Seminarprogramm, das Profis genauso wie Hobbyisten interessieren dürfte. Hier eine Auswahl der geplanten Seminare:

- Elektronik-Apps mit Android
- Entwickeln mit dem Elektor FPGA-Board
- Embedded Linux
- Arduino – nicht nur für Einsteiger
- Menno van der Veen: Röhren
- Tipps und Tricks zu AVR-Controllern
- LabView
- Design mit Eagle

Der Eintrittspreis beträgt für Mitglieder 29,50 Euro (49,50 Euro für Nicht-Mitglieder), Studenten, Azubis und Schüler zahlen in jedem Fall nur 19,50 Euro. Im Eintrittspreis inbegriffen sind zwei Seminare. Insgesamt sind drei Seminarblöcke mit parallel stattfindenden Seminaren geplant. Einer der Zeitslots dauert in diesem Jahr zwei Stunden, was den Dozenten Gelegenheit gibt, das jeweilige Thema noch etwas weiter zu vertiefen. Ab August wird das Seminarprogramm feststehen und man kann sich unter folgender Internetadresse anmelden:

www.elektor-live.de

Wo gibt’s die neue Elektor?

Natürlich freuen wir uns über jeden Leser, der sich entschließt, der immer größer werdenden Elektor-Community beizutreten. Zusätzlich zu den Elektor-Ausgaben gibt es dann alle 14 Tage ein Extra-Projekt als PDF, unbeschränkten Zugang zu weiteren Online-Projekten und nicht zuletzt 10 % Rabatt auf alle Elektor-Produkte (*www.elektor.de/mitgliedschaft*).

Doch wer kein Mitglied ist, ist als Leser ebenso willkommen! In vielen Kiosken, Buchhandlungen und im Zeitschriftenhandel an Bahnhöfen können Sie die aktuelle Elektor-Ausgabe kaufen. Wo genau, das verrät Ihnen eine clevere Suchmaschine unter dem Link *www.pressekaufen.de*. Alle Verkaufsstellen bekommt man hier auf einer interaktiven Karte angezeigt.

Hier noch ein guter Tipp: Wem der Weg zur nächsten Verkaufsstelle zu weit ist, der kann das aktuelle Heft auch im Elektor-Shop bestellen – die Ausgabe wird frei Haus geliefert!

www.elektor.de

Seminar zu EAGLE

Elektor veranstaltet ein praxisorientiertes Seminar zum Thema Platinenlayout mit der Software EAGLE von CadSoft. Begonnen wird mit dem Zeichnen von Schaltplänen unter Verwendung von Standard-Eagle-Bibliotheken. Die Teilnehmer lernen, wie man Schaltpläne über mehrere Seiten hinweg zeichnet und eigene Bibliotheken und Bauteile erstellt. Es wird auch auf das Layouten von HF-Schaltungen eingegangen. Nach dem Layout werden Produktionsdaten für die Fertigung erzeugt. Zum Schluss gibt es Tipps und Tricks zum Umgang mit EAGLE.

Das erste Seminar findet am 12. September in München statt.

Elektor „Gold“- oder „Green“-Mitglieder erhalten 50 % Sonderrabatt auf den normalen Seminarpreis und zahlen somit nur 224,50 €. Die Teilnahmegebühr für Nicht-Mitglieder beträgt 449,00 €.

www.elektor.de/eagle-seminar

50 % Rabatt für Elektor-Mitglieder

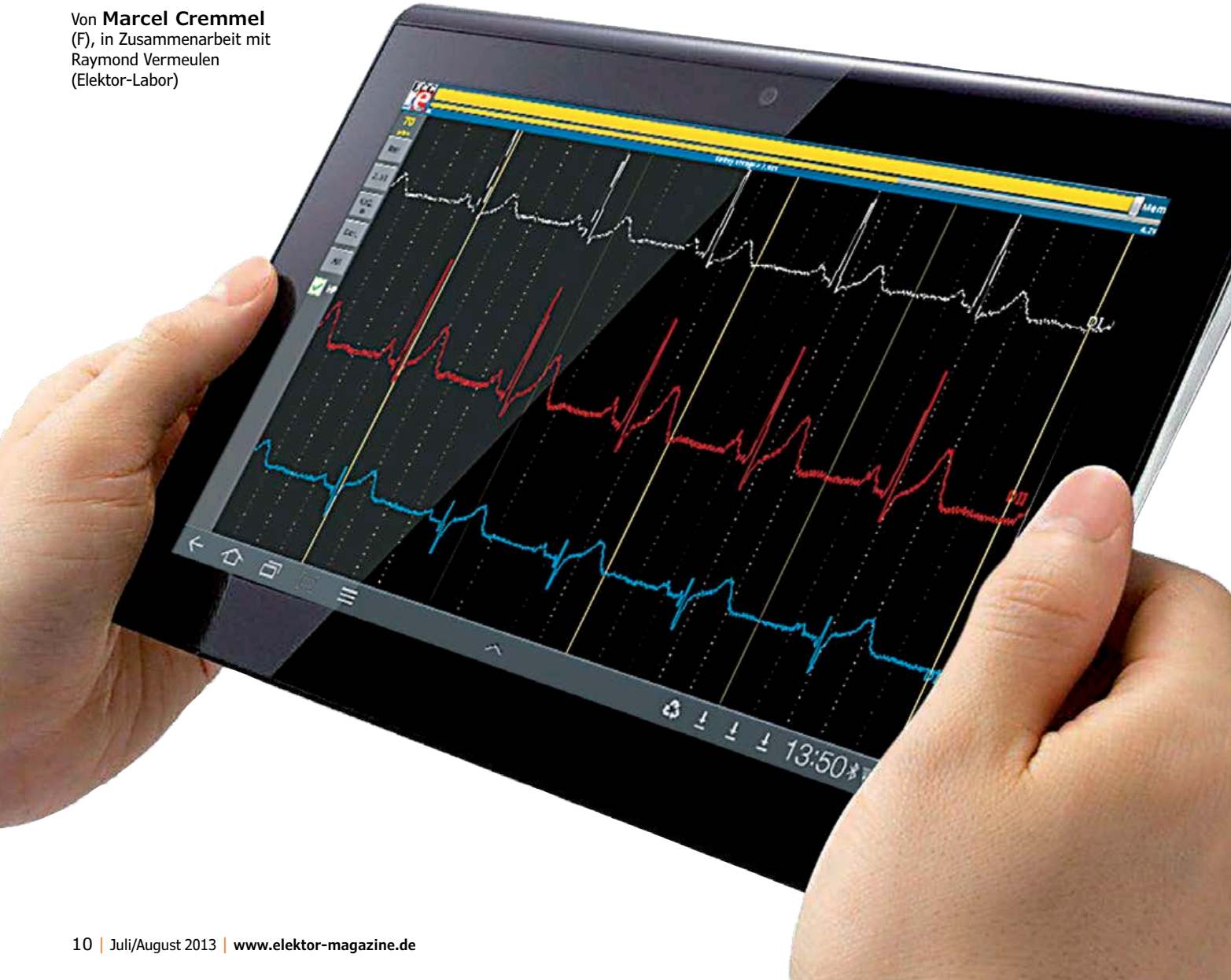
Elektor Android EKG

Draht- und knopflos mit Bluetooth und Touchscreen

Teil 1

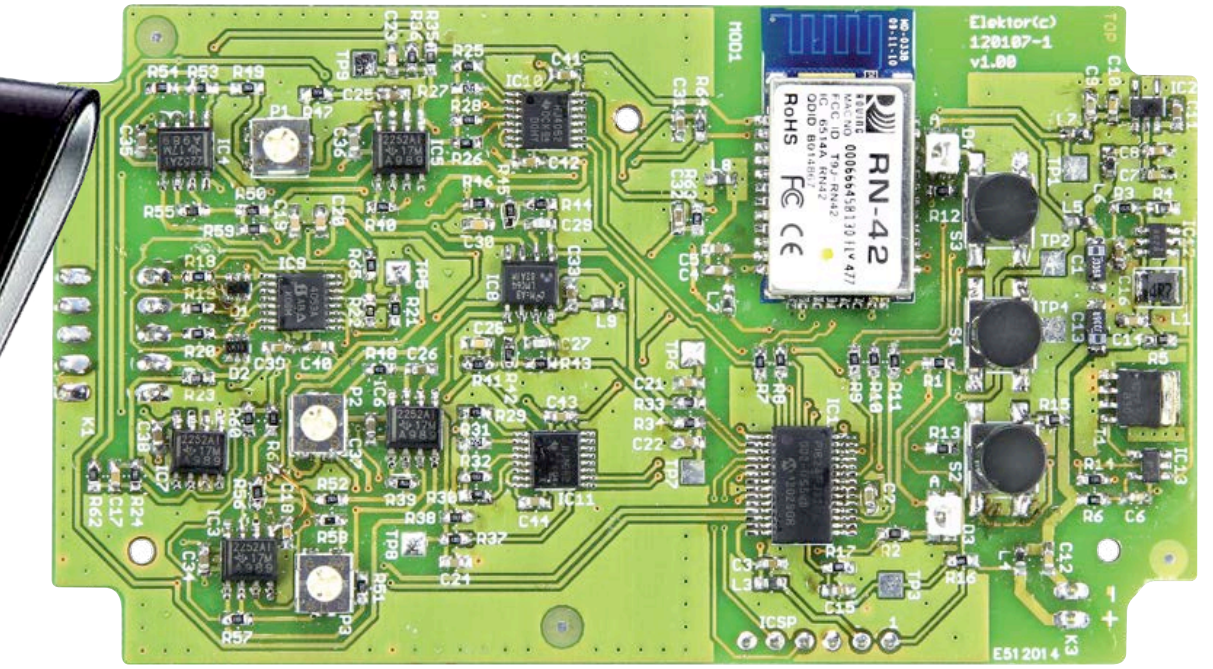
Ein Kardioskop ist ein Gerät, das elektrische Herzsignale visualisiert. Hier werden ein Android-Smartphone oder -Tablet für die Visualisierung drahtlos angebunden. Das Elektor-Kardioskop schafft es, die Herzsignale über eine analoge Vorstufe, einen PIC-Mikrocontroller und ein Bluetooth-Modul aufzunehmen und zu transportieren. Der Schwerpunkt liegt hier jedoch nicht auf der Hardware, die Software übernimmt die wesentlichen Aufgaben.

Von **Marcel Cremmel** (F), in Zusammenarbeit mit Raymond Vermeulen (Elektor-Labor)



Eigenschaften:

- Elektrokardiogramm (EKG) drahtlos über Bluetooth auf Android Smartphones und Tablets
- Kontinuierliches EKG, Standard-Ableitungen I, II und III, erweiterte Ableitungen aVR, aVL und aVF
- Automatisches Anpassen an die Display-Auflösung
- Messen und Anzeigen der Herzfrequenz
- Akustisches Signal im Rhythmus der Herzfrequenz
- Vorschubgeschwindigkeit 250, 125, 62,5 oder 31,25 Pixel in der Sekunde
- Erhöhen des Anzeigemaßstabs um die Faktoren 1, 1,2, 1,5, 2, 3 oder 10
- Empfindlichkeit für vollen Anzeigemaßstab: 3,2 mV, Konvertierung: 10 bit
- Abtastfrequenz der EKG-Signale: 2000 Hz
- Gleichtaktunterdrückung: > 100 dB
- Kompensation des Kontaktpotentials: ± 150 mV
- Kontinuierliche Auto-Zero-Funktion
- Kapazität des dynamischen Signalspeichers: 10 Minuten
- Länge eines EKG im Flash-Speicher: 10 Minuten
- Aufschaltbares Eichsignal 1 mV/2 Hz
- Stromversorgung mit zwei Akkus (1,2 V) oder Batterien (1,5 V)
- Kontinuierliche Anzeige der Akku-Restladung
- Stromaufnahme im Betrieb: 50 mA, im Standby: < 4 µA
- Betriebszeit mit 1-Ah-Zellen: 15 Stunden
- Niedrige Baukosten, hohe Funktionalität



Im Oktober 2006 veröffentlichte Elektor das „GBEKG“, das die Spielekonsole Game Boy zum Herzmonitor werden ließ. Das Betriebssystem Android gab es damals noch nicht, es wurde erst einige Jahre später aus der Taufe gehoben. Inzwischen sind sieben Jahre vergangen, und die Vielzahl der Applikationen für Android ist nicht mehr überschaubar. Nach Schätzungen sind schon heute über 900 Millionen Geräte mit dem Betriebssystem Android in Gebrauch. Trotzdem ist die Entwicklung der Plattform noch längst nicht abgeschlossen, denn tagtäglich erscheinen neue Applikationen am Horizont. Elektor freut sich, dieser Entwicklung mit einem ebenso anschaulichen wie lehrreichen Projekt förderlich zu sein: Das eigene EKG auf dem Smartphone oder Tablet, das dürfte manches Herz höher schlagen lassen!

Die Hardware des Elektor-Kardioskops besteht aus einer Platine mit den Abmessungen 5,5 · 10 cm,

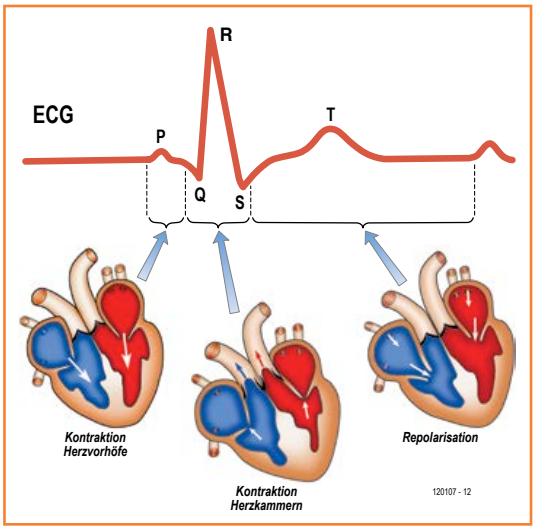


Bild 1.
Phasen elektrischer Aktivität beim Herzschlag.

darauf sind der analoge und digitale Teil untergebracht. Das Modul ist übrigens auch aufgebaut und getestet erhältlich. Was dann noch fehlt, sind vier Elektroden und natürlich die Android-App für Smartphone oder Tablet-Computer. Die Kommunikation der Systemkomponenten findet drahtlos über Bluetooth statt. Gemäß dem Stand der Technik sind nur wenige Elemente vorhanden, die bei der Inbetriebnahme auf der Platine eingestellt werden müssen. Ihre Anzahl ist ebenso wie die Anzahl der Bedienelemente auf drei beschränkt. Schon dies deutet darauf hin, dass die meisten Aufgaben der Software übertragen wurden.
Wir veröffentlichen hier den ersten Teil des Beitrags, mindestens ein weiterer Teil wird folgen. Der Inhalt ist in die folgenden Abschnitte gegliedert:

- Hardware in der von uns ausgewählten Konfiguration,
- Firmware des PICs, geschrieben in C unter MPLAB (dies ist eine kostenlose IDE mit Compiler von Microchip),
- App für Android, geschrieben in Java mit dem Software Development Kit (SDK) von Google,
- Bau der Hardware und Unterbringung in einem Gehäuse von der Größe eines Smartphones.

Wie schon angedeutet, hat die Software des Projekts einen stattlichen Umfang. Aus diesem Grund wollen wir nicht auf die Details des Quellcodes eingehen. Wir beschreiben nur die Funktionen der Software und hoffen, dass dies ausreicht, um vielleicht das Eine oder Andere anzupassen oder zu ergänzen. Erleichtert wird dies dadurch, dass im Quellcode die gleichen Bezeichnungen (*identifier*) wie in der Schaltung verwendet werden. Auf der Projektseite [3] steht sowohl der Teil für MPLAB als auch der Teil für Android zum Download bereit.

Abbild biologischer Vorgänge

Grundlage des Elektor-Kardioskops ist das Elektrokardiogramm (EKG), ein Verfahren, das im Wesentlichen auf den Niederländer Willem Einthoven zurückgeht. Im Beitrag „GBEKG“ vom Oktober 2006 [1] war darüber mehr zu erfahren. Ein EKG ist das grafische Abbild der elektrischen Spannungen, die durch die Tätigkeit des Herzens auf der Haut entstehen. An Hand solcher Abbilder können Kardiologen mögliche Herzerkrankungen aufspüren. Das Herz ist ein autonomer Muskel, was bedeutet, dass die Kontraktionen nicht vom Gehirn gesteuert werden. Im rechten Vorhof des Herzens befindet sich der so genannte Sinusknoten, er ist der primäre elektrische Taktgeber für die Herztätigkeit. Das Herz pumpt das Blut durch den Körper, indem sich der Herzmuskel rhythmisch zusammenzieht und entspannt. Die Kontraktionen entstehen, weil die Membranen umgebender Zellen elektrisch polarisiert werden. Das elektrische Signal ist räumlich nicht auf das Herzmuskelgewebe beschränkt, sondern breitet sich im Körper bis zu den Außenschichten der Haut aus. Elektroden, die auf die Haut geklebt werden, können das Signal auffangen und weiterleiten. Durch Variieren der Positionen auf der Haut entstehen Signalabbilder, die eine sichere Diagnostik des Herzens erlauben [2].
Jeder Herzschlag wird, wie **Bild 1** zeigt, von charakteristischen Phasen elektrischer Aktivität begleitet:

- *P-Phase*: Kontraktion der Vorkammern, die Kammern werden mit Blut gefüllt,
- *QRS-Phase*: Kontraktion der Herzkammern, das Blut wird in die Schlagadern gepresst,
- *T-Phase*: Repolarisierung der Herzkammern, die Kammermuskeln entspannen sich.

Das typische hörbare Herzklopfen entsteht während der Phasen P und QRS.
In **Bild 2a** und **Bild 2b** sind die an den Hand- und Fußgelenken angebrachten Elektroden sowie die dazugehörigen so genannten Ableitungen dargestellt, die das Kardioskop wiedergibt. Die Elektrode am rechten Fußgelenk ist die Erdung. „Ableitung“ nennt der Kardiologe eine Spannung, die aus den Spannungen einer oder mehrerer Elektroden, bezogen auf eine oder mehrere andere Elektroden, hergeleitet wird. Grundsätzlich

wird das Herz bei einem Vierdraht-EKG zweidimensional von vorn betrachtet. Die Elektroden haben die Bezeichnungen „Rechter Arm“, „Linker Arm“ und „Linkes Bein“, abgekürzt RA, LA und LL (L=leg=Bein). Ableitung I ist LA bezogen auf RA, notiert als LA - RA, Ableitung II ist RA - LL. Die Polaritäten sind so gewählt, dass sich die Ableitungen aus Additionen oder Subtraktionen ergeben. Ableitung III ist LA - LL, oder anders geschrieben II - I = LA - RA - LL = LA - LL. Das bedeutet, dass die von den Elektroden kommenden Signale vielfältig konfigurierbar sind. Dies wird vom Kardiologen bei der Diagnostik genutzt. Die Ableitungen I, II und III bilden das so genannte Einthovensche Dreieck, wobei die Mitte der Elektroden der Mittelpunkt des Dreiecks ist. Von diesem Mittelpunkt ergeben sich drei weitere Ableitungen, bezeichnet mit aVR, aVL und aVF (**Bild 2c**). Mit diesen sechs Ableitungen ist bereits eine genaue Herzdiagnose möglich.

Der analoge Teil des Kardioskops verstärkt die von den Elektroden kommenden Signale und leitet daraus die Signale der Ableitungen I und II des EKG ab. Diese Signale werden digitalisiert und über Bluetooth zum Android-Gerät gesendet. Die App, die dort läuft, berechnet aus I und II mit den Gleichungen in Bild 2b die übrigen Ableitungen III, aVR, aVL und aVF.

Die Hauptableitungen I, II und III werden auch mit DI, DII und DIII bezeichnet (D von *derivation*), dies ist jedoch nicht überall gebräuchlich. Die vom Display aufgenommenen Fotos lassen erkennen, dass das EKG klar und deutlich auf dem Smartphone oder Tablet-Computer erscheint. Der Anteil des überlagerten Rauschens und Netzbrummens ist kaum sichtbar.

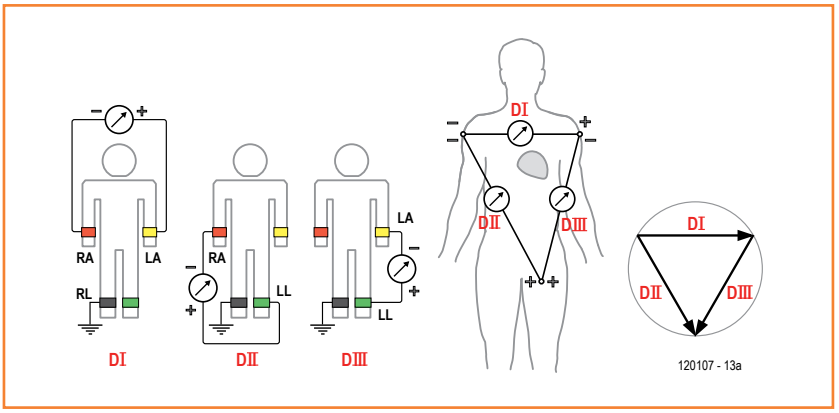


Bild 2a. Typische bipolare Ableitungen.

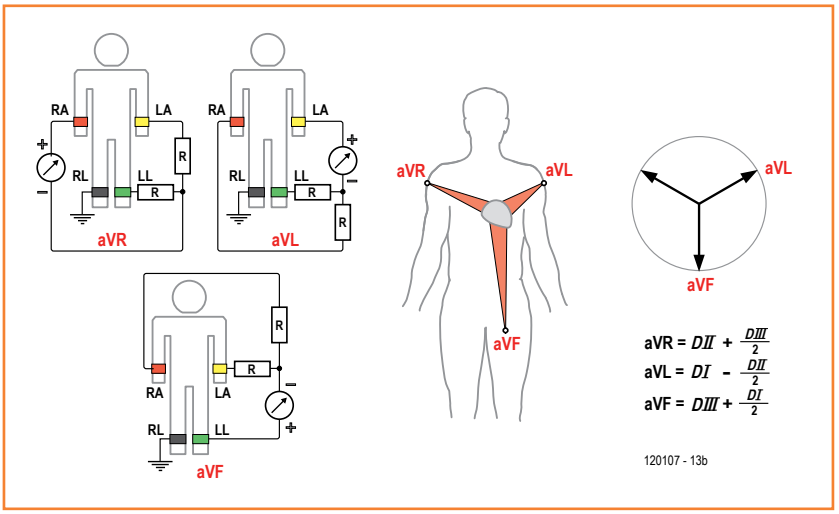


Bild 2b. Typische unipolare oder erweiterte Ableitungen.

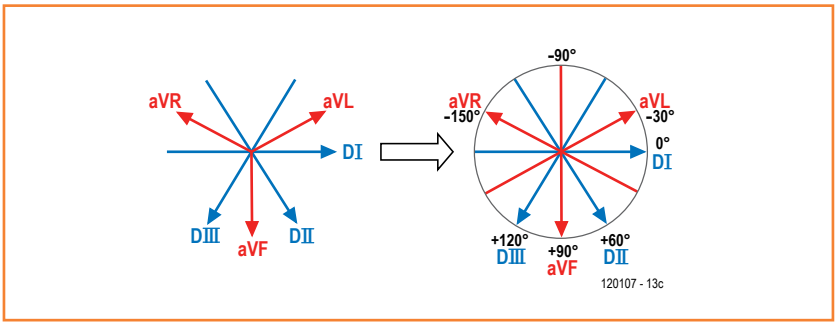


Bild 2c. Darstellung zum Berechnen der Ableitungen.

Der analoge Teil

Dieser Teil des Kardioskops verstärkt die extrem schwachen Signale der Ableitungen DI und DII. Das sind die Spannungen zwischen den Elektroden LA und RA sowie zwischen LL und RA. Wenn die Signale mit voller Dynamik als digitale 10-bit-Werte dargestellt werden sollen, müssen sie mit dem Faktor 1000, also um 60 dB verstärkt werden, was nicht ganz einfach ist. Eine weitere Hürde stellt das sogenannte Kontaktpotential dar, eine Gleichspannung, die beim Kontakt einer Elektrode mit der Haut entsteht. Die Spannung kann in der Größenordnung 100 mV liegen, sie beträgt damit das Hundertfache der Signale, die verstärkt werden müssen. Das Verstärken des Nutzsignals ist nur möglich, wenn das Kontaktpotential kompensiert wird. Ferner ist davon auszugehen, dass die Elektroden auch Einstreuungen vom Stromnetz mit der Frequenz 50 Hz auffangen.

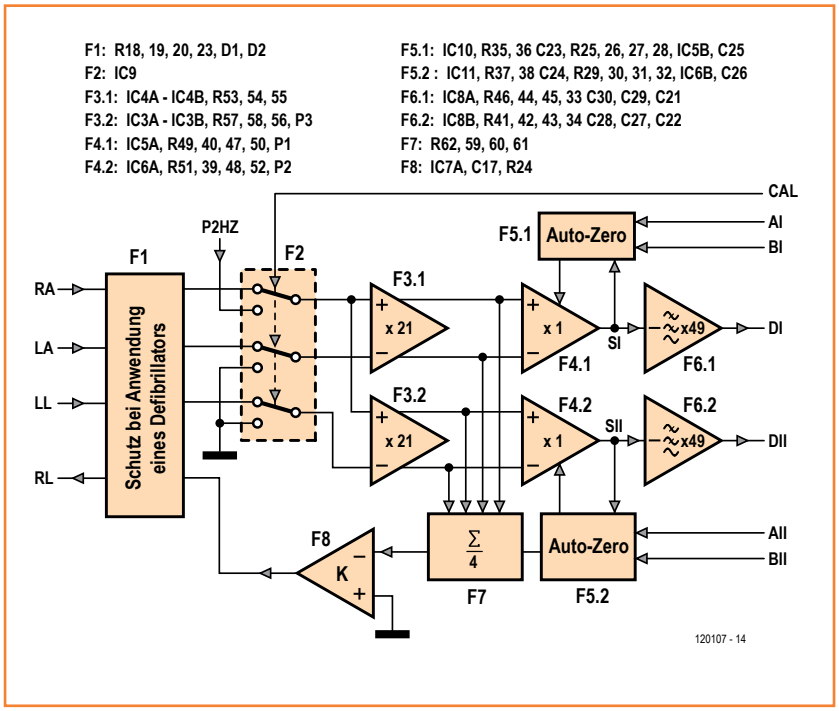


Bild 3.
Funktionsschema des analogen Teils.

Der menschliche Körper und die Zuleitungen der Elektroden stehen unter dem Einfluss von Spannungen und Potentialdifferenzen bezogen auf Erde. Die im Raum verlegten Leitungen des Stromnetzes haben daran einen nicht unbedeutenden Anteil. Die Kapazitäten gegen Erde sind zwar gering, trotzdem kann eine Spannung von mehr als 1 V an der Haut liegen, wobei die Netzfrequenz 50 Hz bestimmend ist.

Das Problem besteht darin, dass das schwache Nutzsignal in ein starkes Störsignal eingebettet ist, das Verhältnis beträgt 1 zu 1000. Die Lösung ist schwierig, zumal die Netzfrequenz 50 Hz in den Frequenzbereich des Nutzsignals fällt. Die Methode des frequenzselektiven Filterns ist folglich nicht anwendbar. Der Frequenz 50 Hz entspricht die Wellenlänge 6000 km, und die Haut ist ein vergleichsweise guter Leiter. Deshalb ist die Annahme begründet, dass die Störsignale an jedem Punkt des Körpers übereinstimmen. Bezogen auf die Elektroden ist das Störsignal ein gemeinsames Signal, auch *Common-Mode*-Signal genannt. Das Problem lässt sich mit einem Verstärker lösen, der ausschließlich Signaldifferenzen verstärkt: Ein differentieller Instrumentierungsverstärker mit der Gleichtaktunterdrückung CMRR (*Common Mode Rejection Ratio*):

$$CMRR \geq \left[\frac{S_p}{S_{ECG}}\right]_{dB} + \left[\frac{S}{N}\right]_{dB}$$

Darin ist S_p die Amplitude des störenden Signals, S_{ECG} ist die Amplitude des Nutzsignals (1 mV) und S/N ist das geforderte Signal-Rausch-Verhältnis (40 dB). Folglich muss **CMRR ≥ 60 db + 40 dB = 100 dB** sein.

Außerdem muss der Verstärker eine hohe Eingangsimpedanz (> 10 MΩ) und eine niedrige Offsetspannung aufweisen. Diese Vorgaben können dazu verleiten, einen darauf spezialisierten Chip in die engere Wahl zu ziehen, beispielsweise den ADS1294 von Texas Instruments. Ohne nennenswerte Konzessionen an das Ergebnis genügt aber auch ein klassischer Opamp des Typs TLC2252 vom gleichen Hersteller. Der TLC2252 ist ein Rail-to-Rail-Opamp, der sich durch einen weiten dynamischen Bereich, ein niedriges Rauschen und einen geringen Strombedarf auszeichnet. Zwar sind die Bandbreite und die Anstiegszeit (*slew rate*) vergleichsweise bescheiden, doch für das hier angestrebte Ziel hat dies kaum Bedeutung. Das einzige Zugeständnis ist ein Trimmpoti zum Einstellen des optimalen CMRR-Werts, dieses Trimmpoti ist unverzichtbar.

Funktionsschema

Vor den Schaltungsdetails soll zuerst das Funktionsschema in **Bild 3** betrachtet werden. Die vier Elektroden sind mit den Eingängen RA, LA, LL und RL verbunden. Elektrokardioskope werden manchmal zusammen mit so genannten Defibrillatoren eingesetzt, die hohe Spannungen erzeugen. Gegen Überspannungen werden die Kardioskop-Eingänge durch F1 geschützt. Die Schutzfunktion ist wirksam, sobald die Eingangssignale die Spannung ±3 V übersteigen.

Der Multiplexer **F2** schaltet zwischen den Signalen RA, LA, und LL auf das Kalibriersignal P2HZ um, die Amplitude des Kalibriersignals beträgt 1 mV, die Frequenz 2 Hz. Nach einem Kalibrierkommando legt der Multiplexer einmal in der Minute für zehn Sekunden das Kalibriersignal an die nachfolgende Stufe, so dass es mit den gemessenen Signalen verglichen werden kann.

Zum Instrumentierungsverstärker gehören die Blöcke **F3** und **F4**. Der Block **F6** ist ein Butterworth-Tiefpass 2. Ordnung mit der Eckfrequenz 170 Hz und der Verstärkung 0,71. Der Tiefpass unterdrückt Signalanteile mit höheren Frequenzen, für den nachfolgenden A/D-Wandler hat er die Funktion eines Anti-Aliasing-Filters.

Die Verstärkungen der einzelnen Stufen betragen AD3 = 21, AD4 = 1 und AD6 = 49, die Nummerierung entspricht den Funktionsblöcken. AD3

ist beispielsweise die differentielle Verstärkung von F3 (F3.1 oder F3.2). Die Gesamtverstärkung beträgt 1029, damit liegt sie über dem geforderten Wert 1000. Weil die Auto-Zero-Funktion bei niedrigen Verstärkungen besser arbeitet, liegen die Verstärkungen der ersten beiden Stufen relativ niedrig.

Die Funktionsblöcke F6, F7 und F8 sind Hilfsfunktionen des Instrumentierungsverstärkers. Die Opamps werden an der symmetrischen Betriebsspannung ±3,3 V betrieben. An den Opamp-Eingängen beträgt die ideale Ruhespannung genau 0 V.

Mit F7 und F8 legt Elektrode RL die mittlere Spannung der aktiven Elektroden als Nullpunkt fest. Am Ausgang von F7 liegt der Mittelwert der Spannungen RA, LA und LL. Funktion F8 vergleicht diese Spannung mit 0 V, die resultierende Fehlerspannung wird als Spannung RL zurückgegeben. Über die Elektroden fließt praktisch kein Strom, weil RA, LA und LL bis auf wenige Millivolt gleich RL sind. Mit dieser Steuerung über die Haut wird dafür gesorgt, dass die mittlere Spannung der aktiven Elektroden auf 0 V fixiert bleibt. Die Ruhespannung des Verstärkers beträgt 0 V, die hohe Eingangsimpedanz wird nicht beeinträchtigt. Leider sind damit noch nicht alle Probleme gelöst. Wie schon erwähnt, entsteht zwischen Haut und Elektroden ein Kontaktpotential, das einige Millivolt betragen kann. Das Kontaktpotential wird nicht unterdrückt, sondern vom Instrumentierungsverstärker ebenfalls verstärkt. Mit F7 und F8 kann

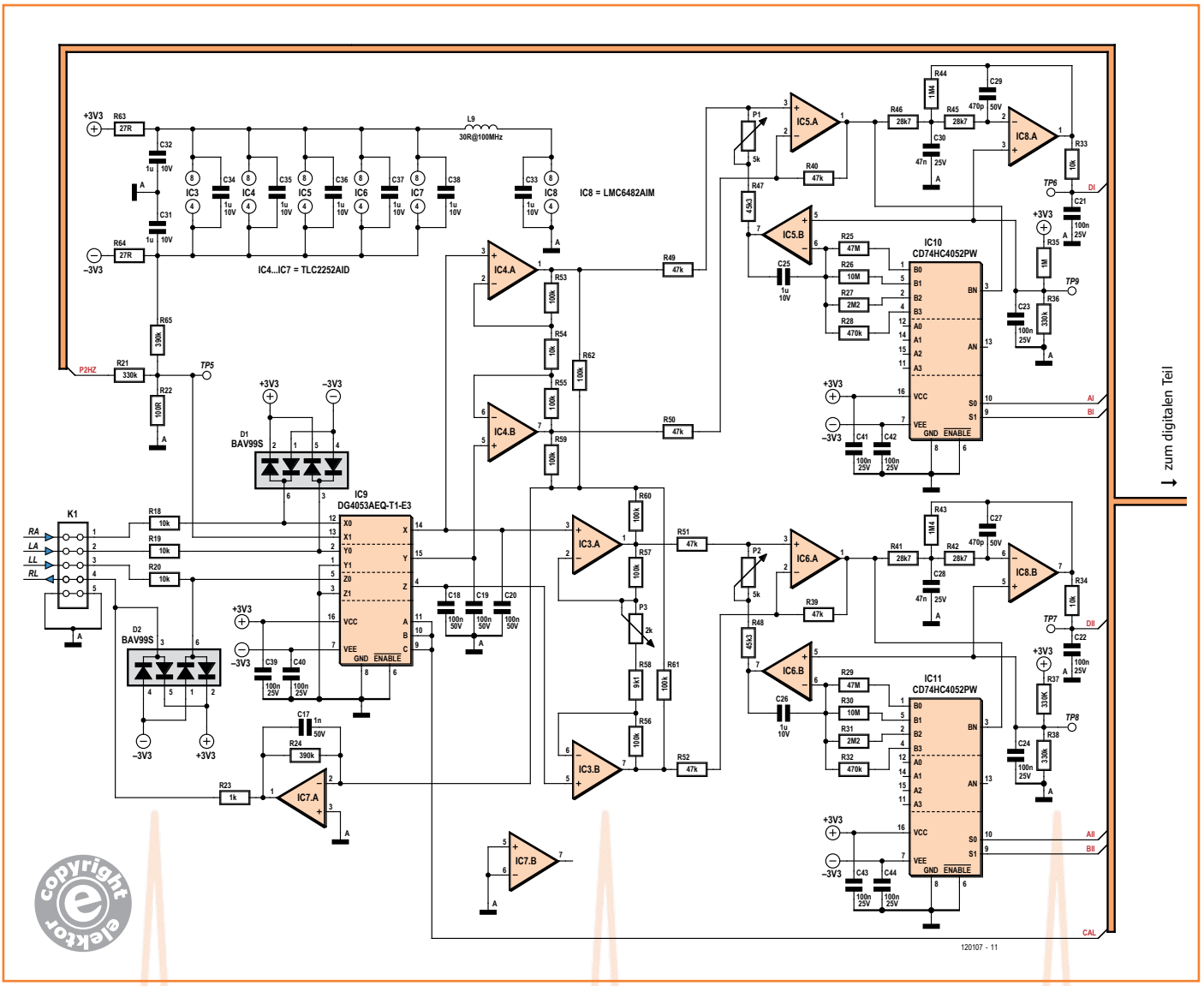


Bild 4a. Schaltung des analogen Teils.

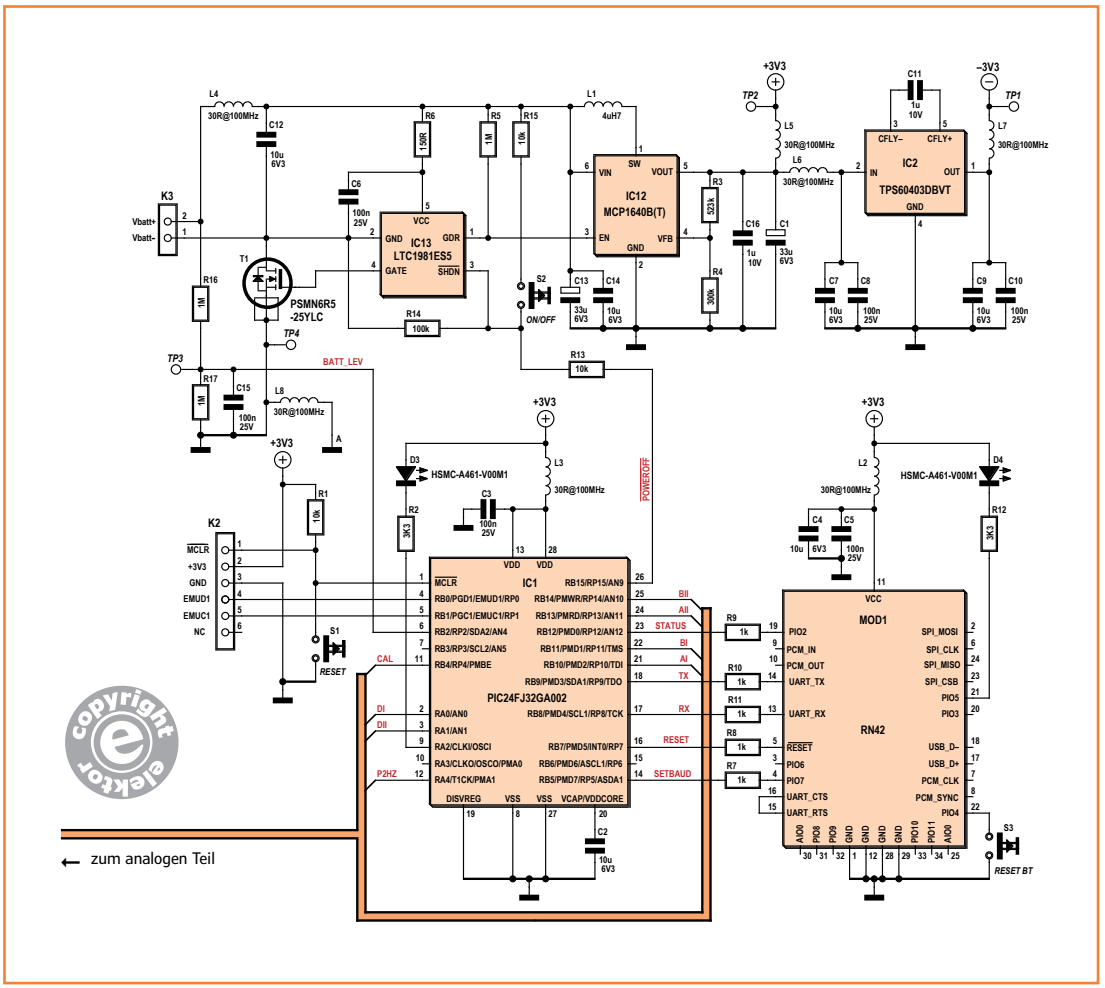


Bild 4b.
Der digitale Teil und die Stromversorgung.

diese Erscheinung nur teilweise kompensiert werden. Der Offset zwischen den nicht invertierenden und invertierenden Eingängen von F3.1 und F3.2 kann mehrere Volt betragen. Diese Offsets werden von F5.1 und F5.2 kompensiert, damit F4.1, F4.2, F6.1 und F6.2 nicht in die Sättigung geraten. Die Blöcke F5.1 und F5.2 vergleichen die Mittelwerte der Signale SI und SII mit einer Referenzspannung. Die Fehlerspannung wird mit einer von vier Zeitkonstanten integriert, sie können mit AI und BI für Kanal DI und mit AII und BII für Kanal DII eingestellt werden. Das Ergebnis ist eine kontinuierliche Offset-Spannung, sie wird F4.1 und F4.2 zugeführt. Die Offset-Spannung gelangt zurück nach SI und SII. Die entstandene Korrekturschleife bringt die Mittelwerte von SI und SII auf ihre Nullwerte zurück, die Kontaktpotentiale sind eliminiert.

Schaltung mit Herz

Die Blöcke aus dem Funktionsschema finden sich in der Schaltung des analogen Teils (**Bild 4a**) wieder. Der Anschluss der vier Elektroden ist K1. Die Widerstände R18, R19, R20, R23 und die Vierfach-Dioden D1 und D2 bilden den Überspannungsschutz F1, während IC9, ein CMOS-IC 4053, der analoge Multiplexer F2 ist.

Die Differenzverstärker beider Zweige F3 und F4 sind als klassische Instrumentierungsverstärker ausgeführt: F3.1 = IC4A + IC4B, F3.2 = IC3A + IC3B, F4.1 = IC5A und F4.2 = IC6A.

Die Verstärkung von DII ist mit P3 einstellbar, damit einer möglichen Asymmetrie bezogen auf den anderen Zweig entgegengewirkt werden kann. Schon geringe Unterschiede bei der Verstärkung würden die Genauigkeit der zu berechnenden Ableitung beeinträchtigen.

Mit den Potis P1 und P2 wird die Gleichtaktunterdrückung der Verstärker auf ihr Optimum eingestellt. Die zu F4 gehörenden Stufen bilden eine Gegenkoppelschleife. Ableitung DI durchläuft den Multiplexer IC10, einen der Widerstände R15... R28 und schließlich den mit IC5B aufgebauten

Integrator. Dieser Teil ist für die Funktion 5.1. zuständig. Die mittlere Spannung von DI wird über IC5A auf den Wert am Spannungsteiler R35/R36 gebracht, für DII ist dies R37/R38. Eine Software-Funktion steuert mit Multiplexer IC10 (IC11 im anderen Zweig) über AI und BI die Zeitkonstante dieser Schleife:

T1 = R28 · C25 = 0,47 s
T2 = R27 · C25 = 2,2 s
T3 = R26 · C25 = 10 s
T4 = R25 · C25 = 47 s

Wenn die Elektroden an ihrem Platz sind, wird zuerst die kürzeste Zeitkonstante eingestellt und das Nullniveau abgeschätzt. Danach wird die Zeitkonstante erhöht, um das tatsächliche Nullniveau möglichst genau zu treffen, dabei beträgt die Zeitkonstante bis zu 47 Sekunden. Der Wert der Zeitkonstanten wirkt sich noch nicht auf das EKG-Signal aus. Die Nullpunkt-Kompensationen der Signale DI und DII sind voneinander unabhängig. Damit ist die optimale Anpassung an das übergebene Signal möglich, der A/D-Wandler gerät nicht in die Sättigung, und gleichzeitig wird der volle dynamische Bereich genutzt.

Die letzte Stufe in beiden Zweigen ist der Butterworth-Tiefpass 2. Ordnung mit IC8A und IC8B. Die Eckfrequenz beträgt 170 Hz, die Verstärkung im Durchlassbereich ist 34 dB. Der Tiefpass ist als Anti-Aliasing-Filter für den nachfolgenden A/D-Wandler wirksam. Der A/D-Wandler tastet das Signal mit der Abtastfrequenz 2000 Hz ab. Dann folgt ein Tiefpass 1. Ordnung, Eckfrequenz 160 Hz, bestehend aus R33 und C21. Die Dämpfung beider Filter beträgt bei der halben Abtastfrequenz ungefähr 15 dB.

Vielleicht ist aufgefallen, dass der negative Wert der Betriebsspannung von IC8 nicht -3,3 V, sondern 0 V beträgt. Das ist Absicht, denn damit wird verhindert, dass zum Eingang des Mikrocontrollers negative Spannung gelangt.

Der analoge Teil des Kardioskops ist hoch empfindlich gegen Signaleinstreuungen, insbesondere gegen Störsignale aus dem digitalen Teil und der schaltenden Stromversorgung. Deshalb wurde das Platinenlayout konsequent so gestaltet, dass die drei Teile auch räumlich voneinander getrennt sind. Nachzutragen ist noch, dass die RC-Kombinationen R63/C32 und R64/C31 Welligkeiten der Betriebsspannung im analogen Teil mindern.

Der digitale Teil der Schaltung (**Bild 4b**) entspricht im Wesentlichen dem üblichen Standard, so dass er nur kurz betrachtet werden soll.

Bild 4c.
Das Bluetooth-Modul von Roving Networks.

Mikrocontroller

Der Mikrocontroller ist ein PIC24FJ32GA002 im 28-Pin-SOIC-Gehäuse aus der PIC24-Familie von Microchip. Das Beschalten mit externen Komponenten ist hier nicht erforderlich. Die Leistung bei der internen Taktfrequenz 8 MHz beträgt 2 MIPS, was für den Betrieb des Kardioskops völlig ausreicht. Bei dieser Taktfrequenz beträgt die Stromaufnahme nur etwa 5,4 mA, dies kommt dem für das Kardioskop notwendigen Akku- oder Batteriebetrieb entgegen.

Bluetooth-Modul

Das Bluetooth-Modul RN-42 von Roving Networks (**Bild 4c**) ist ein kompakter, energiesparender Baustein für den Einsatz von Bluetooth der Klasse 2. Mit der eingebauten Antenne lassen sich Distanzen bis etwa 20 m überbrücken. Das Modul unterstützt das RFCOMM-SPP-Protokoll, das Übertragungsgeschwindigkeiten bis 240 Kbit/s zulässt. Das Elektor-Kardioskop arbeitet mit der Geschwindigkeit 16000 bit/s.

Mit dem *Serial Port Profil* (SPP) ist unkompliziert eine asynchrone drahtlose Duplex-Verbindung realisierbar. Die Leitungen Rx und Tx des UARTs im Mikrocontroller werden mit den identisch bezeichneten Anschlüssen des Bluetooth-Moduls verbunden. Die asynchronen seriellen Datenwörter der Leitung Tx gelangen transparent zum angebundenen Android-Gerät. Über Leitung Rx werden die Antworten des Android-Geräts in gleichem Datenformat empfangen.

Für das Bluetooth-Modul gelten außerdem folgende Kommandos oder Signale:

STATUS, 1 = Verbindung besteht, 0 = keine Verbindung
RESET, Rücksetzen und Initialisieren des Bluetooth-Moduls
SETBAUD, Einstellen der Übertragungsgeschwindigkeit:

- 1 = 9600 baud
- 0 = 115,2 kbaud

Die Blinkfrequenz der LED D4 signalisiert die Aktivitäten des Bluetooth-Moduls:

- 10 Hz: Konfiguration
- 2 Hz: Initialisierung
- 1 Hz: Modul „sichtbar"
- kontinuierlich: Verbindung besteht

Stromversorgung

Obwohl die gesamte Schaltung an der gleichen symmetrischen Betriebsspannung arbeitet, fällt die Stromversorgung mit IC13, IC12 und IC2 vergleichsweise komplex aus. Das liegt daran, dass aus zwei AA-Akkus mit 2 · 1,2 V oder zwei Batterien mit 2 · 1,5 V eine stabile symmetrische Spannung ±3,3 V erzeugt werden muss und die Energie der Stromquelle fast vollständig genutzt wird.

Wegen der Sicherheit für Leib und Leben darf das Elektor-Kardioskop nicht über einen Adapter am Stromnetz betrieben werden!

MOSFET T1 schützt IC13 gegen eventuelle Verpolung der Energiequelle. Falls dies geschieht, begrenzt R6 den Strom für IC13, während T1 und die interne Diode sperren. Der negative Pol der Energiequelle ist nicht mit Erde verbunden. Damit ist die Schaltung vor Beschädigung gesichert.

Der „High Side Switch Controller" IC13 wird vom Signal /PowerOff des Mikrocontrollers gesteuert. Wenn die Spannung der Energiequelle ihren Grenzwert unterschreitet, ist der Mikrocontroller abgeschaltet, so dass das Signal /PowerOff nicht High werden kann. Der Eingang /SHDN liegt dann über R14 auf Low. Ausgang GATE von IC13 ist ebenfalls Low, und MOSFET T1 sperrt. Jetzt leitet die interne Freilaufdiode, weil Strom von GND zum negativen Pol der Energiequelle fließt, IC12 erhält nun Betriebsspannung. Ausgang GDR von IC13 bleibt jedoch Low, folglich fehlt die Betriebsspannung 3,3 V, die Spannungsumsetzer IC12 liefert. Damit ist auch Spannungsinverter IC2 funktionslos. Der Strombedarf beschränkt sich auf IC13 und IC12 im Standby-Betrieb, er beträgt weniger als 4 µA.

Durch Drücken des Tasters S2 geht Eingang /SHDN von IC13 auf High. MOSFET T1 wird über Ausgang GATE von IC13 durchgesteuert. In IC13 ist eine Ladungspumpe integriert, so dass die Spannung an GATE die Betriebsspannung V_{CC} um das 2,5-fache übersteigen kann. Der Aufwand ist nötig, damit der Durchlasswiderstand R_{ON} von MOSFET T1 einen möglichst niedrigen Wert annimmt. Der negative Pol der Energiequelle liegt nun an GND.

Auch Ausgang GDR von IC13 ist High, so dass IC12 freigegeben wird. Dieser DC/DC-Wandler erzeugt aus der Spannung der Energiequelle die stabile Betriebsspannung +3,3 V. Dann folgt IC2, ein Spannungsinverter mit integrierter Ladungspumpe, er setzt +3,3 V in −3,3 V um.

Wenn der Mikrocontroller startet, legt er sofort das Signal /PowerOff auf High. Andernfalls würde /SHDN Low werden, sobald der Taster S2 nicht mehr gedrückt wird.

Das Kardioskop wird über das Android-Gerät abgeschaltet, sobald der Anwender die App beendet. Dann gehen /PowerOff und anschließend / SHDN von IC13 auf Low, was auch GDR nach Low zieht und IC12 außer Betrieb setzt. Damit ist die Betriebsspannung nicht mehr vorhanden. Eine kurze Betätigung von S1 hat dieselbe Wirkung.

Der Wirkungsgrad der DC/DC-Wandler liegt nahe 90 %, sie sind funktionsfähig, bis die Spannung der Energiequelle auf 0,8 V oder sogar noch tiefer gesunken ist.

Der nächste Teil des Beitrags wird die Controller-Software, die Android-App, die Platine und das Einstellen des Kardioskops zum Inhalt haben. Das Einstellen ist, so viel sei schon vorweg genommen, überraschend einfach.

(120107)gd

Heartbeats auf Ihrem Android-Smartphone oder -Tablet

Weblinks

[1] GBEKG, Elektor Oktober 2006, S. 32
www.elektor.de/jahrgang/2006/oktober/gbekg.64480.lynkx

[2] Mehr über die Kardiologie (französischsprachig)
http://goo.gl/mSr20

[3] www.elektor.de/120107

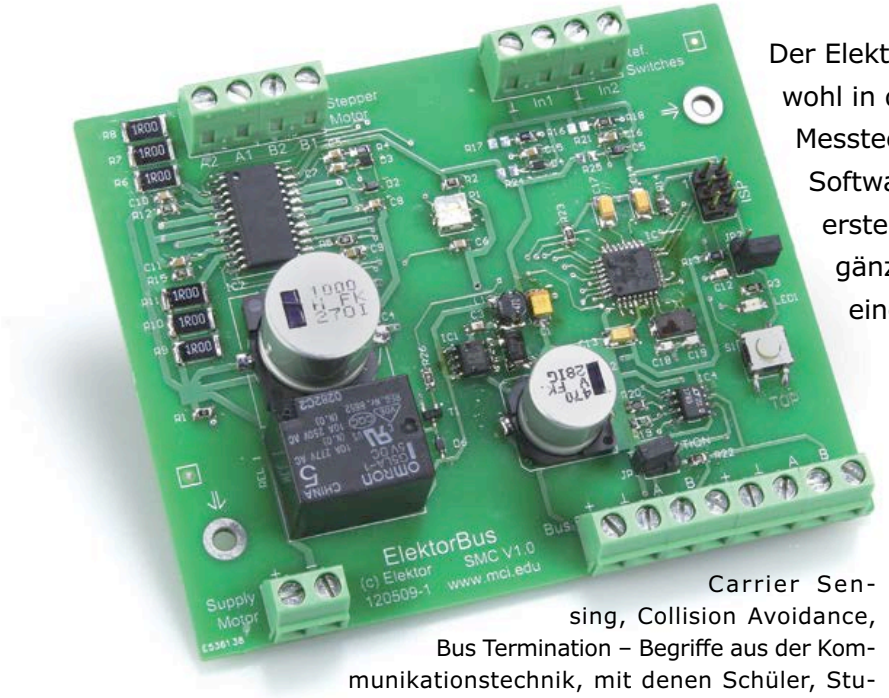
[4] Android Apps programmieren – Schritt für Schritt, von Stephan Schwark
www.elektor.de/products/books/programming/android.2169212.lynkx

[6] Website des Autors:
http://electronique.marcel.free.fr/

Schrittmacher

Schrittmotor-Treiber für den ElektorBus

Von **Ronald Stärz, Mathias Gfall** und **Jens Nickel**



Der ElektorBus eröffnet viele Möglichkeiten – sowohl in der Hausautomatisierung als auch in der Messtechnik. Durch die modulare Hard- und Software ist eine eigene Anwendung schnell erstellt. Unseren ElektorBus-„Baukasten“ ergänzen wir nun um ein Board zur Steuerung eines Schrittmotors.

Carrier Sensing, Collision Avoidance, Bus Termination – Begriffe aus der Kommunikationstechnik, mit denen Schüler, Studenten oder Hobbyelektroniker mitunter recht charmefrei konfrontiert werden. Spannender wird die Thematik, sobald man die Möglichkeit hat, sich mit eigenen Schaltungen und Projekten auf das Feld der Bussysteme zu stürzen. Der ElektorBus definiert die perfekte Grundlage, um – aufbauend auf dem Referenzdesign – eigene Anwendungen zu realisieren. So spielt der ElektorBus nun eine zentrale Rolle im Mechatronikstudium am MCI in Innsbruck [1]. Die Schaltung dieses Schrittmotor-Boards ist ein beispielhaftes Ergebnis einer studentischen Projektarbeit.

Das Board ist über Elektor sowohl als Platine als auch als bestücktes und getestetes Modul erhältlich; Software als Ausgangspunkt für eigene Anwendungen gibt es wie immer als kostenlosen Download [2].

Eigenschaften

- Schrittmotortreiber L6208 von STMicroelectronics
- Für bipolare Schrittmotoren bis 60 V
- Max. rund 2,5 A Motorstrom
- RS485-Interface
- 1x LED, 1x Taster für Test, Debugging und Inbetriebnahme
- 1x Sicherheitsrelais für Motor-Stromversorgung (per Software steuerbar)
- Controller- und Boardfile für Embedded Firmware Library (EFL) im Download
- Schrittmotorbibliothek für EFL im Download
- ElektorBus-kompatibel, Open-Source-Demoanwendung im Download

Let’s step

Die Schaltung gliedert sich wie in **Bild 1** dargestellt in mehrere Schaltungsblöcke. Im Mittelpunkt stehen die MCU und der Schrittmotortreiber. Flankiert werden diese Schaltungsteile vom Businterface, einer (getakteten) Stromversorgung sowie (universell anpassbaren) End-/Positionsschaltern.

Die Ansteuerung des Schrittmotors basiert auf dem IC L6208 von STMicroelectronics, **Bild 2** zeigt das Innere des Bausteins [3]. Man erkennt die integrierten Freilaufdioden, die zum Schutz der Leistungshalbleiter bei induktiven Lasten, wie sie die Spulenwicklungen darstellen, unerlässlich sind. Der Treiberbaustein wurde ausgewählt, weil er die Leistungsendstufen und eine State-Machine zur Erzeugung der dafür notwendigen Steuersignale beinhaltet, weiterhin kümmert er sich noch um die Stromregelung in den Motorspulen und

besitzt eine Überstromüberwachung. Somit ist einerseits eine einfache Ansteuerung möglich, die sich auf die Erzeugung von vier digitalen Steuersignalen für Drehrichtung (CW/CCW), Schritt (CLK), Betriebsmodus (HALF/FULL) sowie eine Freigabe (EN) reduziert. Andererseits bleibt durch die hohe Integrationsdichte die Anzahl externer Bauteile auf ein Minimum beschränkt, wie man im Schaltplan in **Bild 3** erkennt. Die Widerstände (R6..R11) an den jeweiligen Sense-Eingängen dienen der Strommessung, der Spannungsabfall an ihnen ist die Messgröße für den internen Stromregler. Die Führungsgröße wird an den Vref-Eingängen über das Trimmpotentiometer P1 vorgegeben. In der vorliegenden Dimensionierung ergibt sich ein Spulenstrom von 100 mA, wenn die Referenzspannung auf 33,3 mV eingestellt wird. Die RC-Netzwerke R12/C10 und R15/C11 definieren die Off-Zeit der Power-MOSFETs innerhalb der Brückenschaltungen [3]. Zur Erzeugung der Gatespannungen verwendet der L6208 eine Ladungspumpe, die durch D2, D3, R4, C5 und C7 vervollständigt wird. R5 und C9 sorgen für geordnete Spannungspegel am Enable-Eingang.

Als Controller wurde ein ATmega328 ausgewählt, er bietet mit 32 KB Flash genug Speicher für den Einsatz einer modularen Softwarebibliothek wie zum Beispiel der Embedded Firmware Library (siehe unten). Die Beschaltung des µC umfasst neben den Standardbauteilen für Spannungsstabilisierung und Takterzeugung auch eine Möglichkeit für einen manuellen Reset an JP2. Ein Taster an PD5 (S1) und eine LED an PD4 (LED1) leisten für die Inbetriebnahme und das Debuggen eigener Anwendungen wertvolle Dienste. Die hardwareseitige Anbindung an den Bus erfolgt über den RS485-Transceiver LT1785; der Bus kann mittels JP3 und R22 abgeschlossen werden, falls das Schrittmotor-Board der erste oder letzte Teilnehmer am Bus sein sollte. Die Schraubklemmen für die RS485-Signale und die 12-V-Versorgung sind wie beim bekannten ElektorBus-Experimentalknoten [4] jeweils zweifach vorhanden, sodass man die vier Leitungen des Bussystems bequem durchschleifen kann.

Spannungsversorgung

Die Spannungsversorgung für die Logik stellt der Schaltregler IC1 (ein LM2675M-5) zur Verfügung. In Kombination mit L1, D1, C1 und C3 erzeugt er nahezu verlustfrei die benötigten 5 V (aus einer Eingangsspannung zwischen 7 V und 24 V).

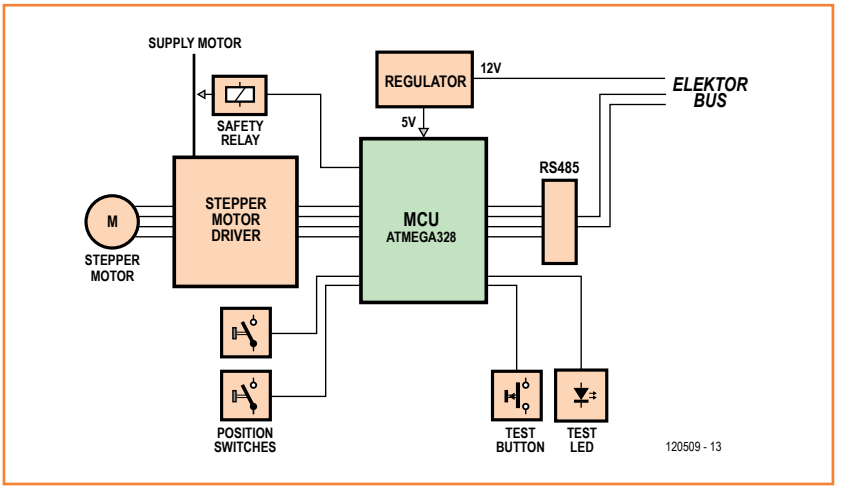
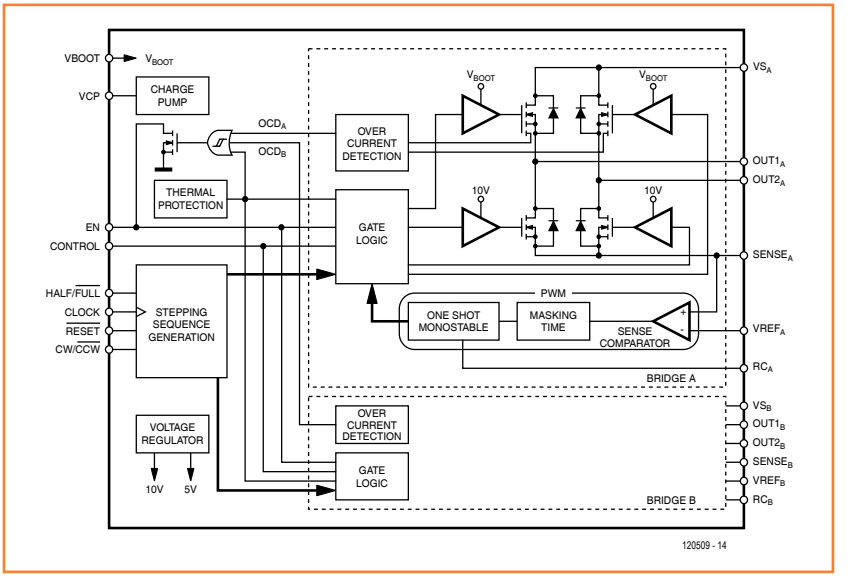


Bild 1. Blockdiagramm des Schrittmotor-Boards für den ElektorBus.

Die Stromversorgung der Leistungselektronik ist getrennt auf Anschlussklemmen geführt, um den Stromhunger des Schrittmotors nicht über die 12-V-Leitungen des ElektorBus decken zu müssen. Einen kleinen Nachteil weist der Schrittmotortreiber L6208 in diesem Zusammenhang jedoch auf: Ist die Versorgung des Leistungsteils bereits vorhanden und sind die 5 V für die Logik (noch) nicht präsent, dann quittiert der Treiberbaustein dies mit unmissverständlichen Rauchzeichen! In der Schaltung haben wir die Versorgungsspannung des L6208 daher über ein Relais (REL1) geführt, das softwareseitig über den Controllerpin PD6 steuerbar ist. Den gesamten Leistungsteil könnte man also gegebenenfalls über den ElektorBus stromlos schalten. Da es

Bild 2. Blockdiagramm des Schrittmotortreibers L6208 (aus [3], STMicroelectronics).



aufgrund der Ströme von einigen hundert Milliampere zu problematischen Spannungsabfällen auf den Masseleitungen kommen kann, haben wir einen sternförmigen Massepunkt angelegt, der bei R1 (einem 0-Ω-Widerstand) zu finden ist. R1 ermöglicht das Auftrennen der Schaltungsmassen von Logik- und Leistungsteil im Fehlerfall.

Wenn der Schrittmotor nicht nur als Antrieb mit variabler Drehzahl eingesetzt werden soll, sondern damit Positionieraufgaben abgedeckt werden sollen, erfordert dies die Einbindung von End- bzw. Positionsschaltern. Solche Schalter können mechanisch ausgeführt sein oder auch als Näherungsschalter auf induktiver oder kapazitiver Basis, ebenso als Gabellichtschranke (um nur einige zu nennen). Um möglichst viele dieser Möglichkeiten abzudecken, verfügt das Board über eine universell anpassbare Eingangsbeschaltung für zwei Kanäle, die aus R16, R17, R24, C15 und

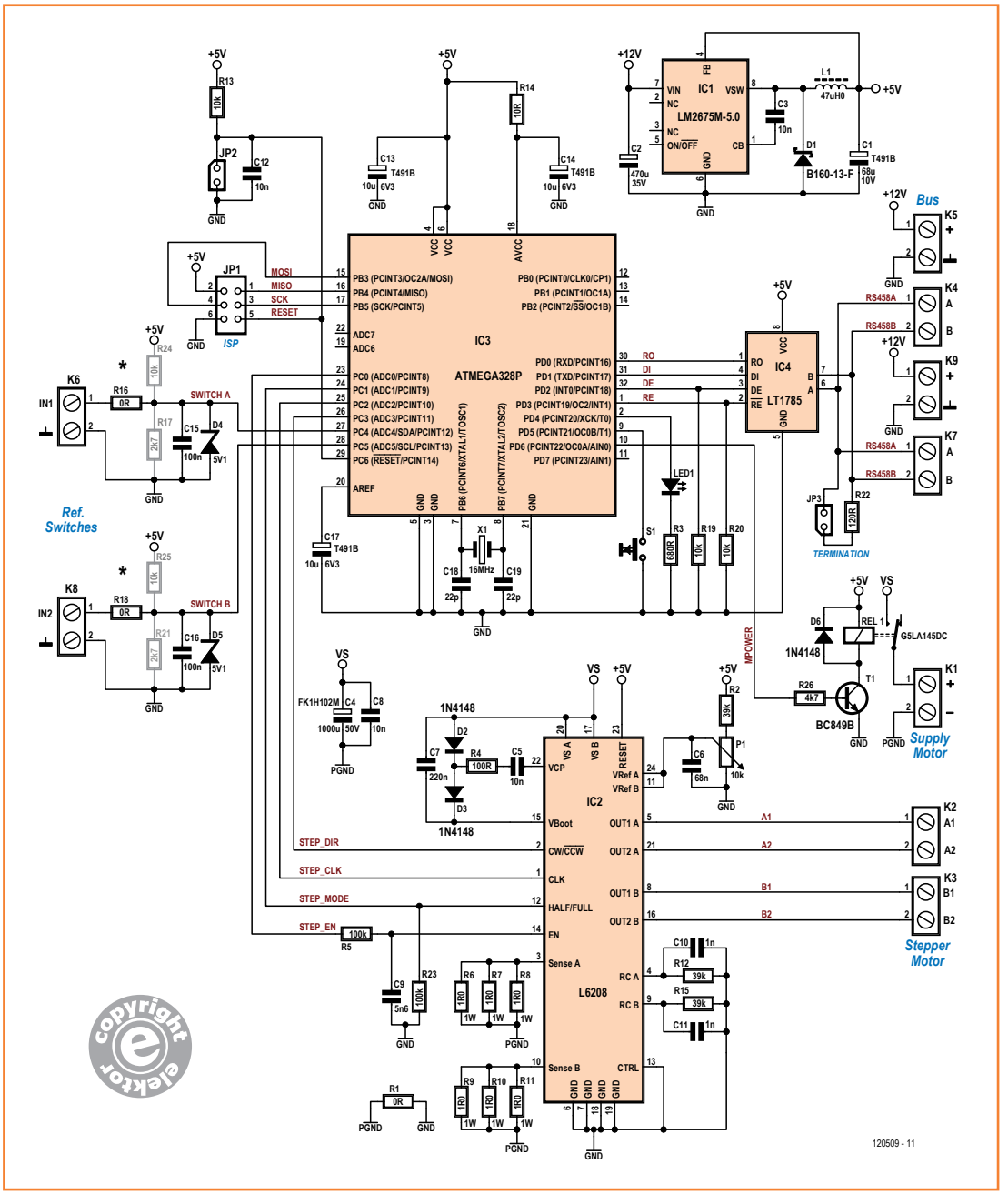
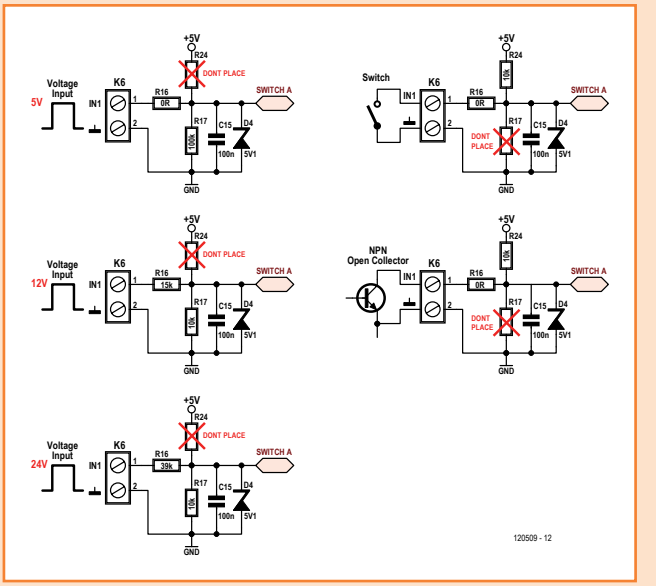


Bild 3.
Schaltplan des Schrittmotor-Boards. Zur Konfiguration der Positionsschalter siehe den Kasten.

Konfigurationen der Positionsschalter

Switch:
Soll ein Schalter angeschlossen werden, ist R16 mit 0 Ω zu bestücken, R17 wird nicht bestückt. Der Pullup–Widerstand R24 mit 10 k ist optional (falls man die Schaltung an einen Controller anschließt, der nicht über interne Pullup-Widerstände verfügt). Wird der Schalter geschlossen, dann liegt der entsprechende Mikrocontrollereingang auf Masse (active low). Diese Konfiguration findet sich auf dem bei Elektor bestellbaren Board als „Werkseinstellung“.



Voltage Input 5 V:
Wird R16 überbrückt und R24 nicht bestückt, liegt das 5-V-Ausgangssignal eines Referenzschalters gefiltert am Eingang des Mikrocontrollers an.

Voltage Input 12 V:
Ein Spannungsteiler definiert ein Verhältnis von 25k / 10k = 2,5, was einer Spannung von 4,8 V am Mikrocontroller entspricht.

Voltage Input 24V:
Wegen des Spannungsteilers R16 / R17 ergibt sich eine Ausgangsspannung von 4,9 V.

NPN Open Collector:
Schalter mit solchen Ausgängen besitzen ohne Beschaltung kein Ausgangssignal. Vielmehr kann der Pegel über einen Pullup-Widerstand an die schaltungstechnischen Bedürfnisse angepasst werden. Liegt an der Basis des Transistors eine Spannung an, ist die Kollektor-Emitter-Diode leitend, und das Potential des Mikrocontrollereingangs wird auf Masse gezogen.

D4 sowie R18, R21, R25, C16 und D5 gebildet wird. Die Schutz- und Filterschaltung aus C15/C16 und den Z-Dioden D4/D5 ist in jedem Fall erforderlich, die Widerstände werden anforderungsspezifisch bestückt. Im Kasten sind die unterschiedlichen Möglichkeiten dargestellt. Im Auslieferungszustand des Boards sind R17/R21 nicht und R16/R18 mit 0 Ω bestückt, so dass hier einfache mechanische Schalter oder Taster angeschlossen werden können. R24 und R25 sind optional zu bestückende Pullup-Widerstände; die per Software zuschaltbaren internen Pullups des AVR-Controllers reichen aber für eine korrekte Funktion aus.

Software

Nicht zufällig haben wir einen ATmega328 als Controller gewählt; dieser leistet uns ja schon im ElektorBus-Experimentalknoten gute Dienste. Und da der in der ElektorBus-Artikelserie [5] genutzte ATmega88 kompatibel ist, gibt es schon einiges an Software, die man leicht modifiziert auch hier nutzen könnte.
Leicht wiederverwendbaren und anpassbaren Code für das Board können wir mit der in der Maiausgabe vorgestellten „Embedded Firmware Library“ [6] erzeugen, die bereits eine Bibliothek für die ElektorBus-Kommunikation mitbringt (**Bild 4**). Außerdem deckt die EFL den ATmega328 bereits mit einem „Controllerfile“ ab. Peripherieblocks wie der RS485-Treiber sowie die LED und der Taster sind überdies an dieselben Controllerpins angeschlossen wie beim Experimentalknoten, für den in der EFL-Codebasis bereits ein „Boardfile“ existiert. Wir könnten in diesem Boardfile nun die Initialisierung der zweiten LED, des zweiten Tasters und des Extension-Connectors streichen (die sich auf der Schrittmotorplatine nicht finden). Wenn wir nun noch den Schrittmotortreiber als weiteren Peripherieblock ergänzen, haben wir bereits ein passendes Boardfile und könnten unsere eigentliche Anwendung hard-

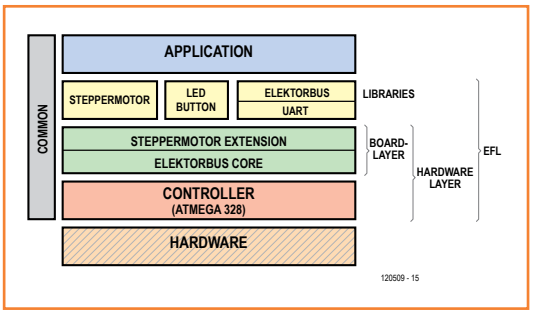


Bild 4.
Mit den Softwaremodulen der EFL ist die Firmware schnell erstellt.

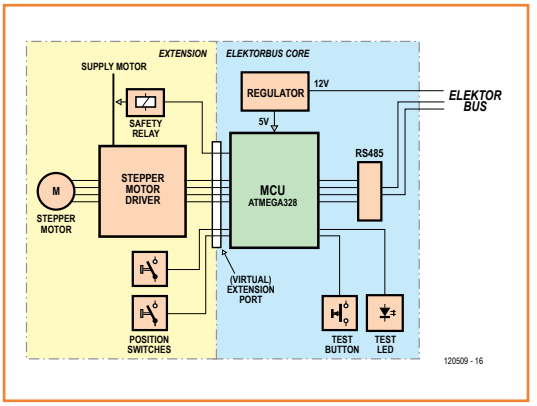


Bild 5.
Das Board wird (virtuell) in einen ElektorBus-Kern und einen Erweiterungsbereich eingeteilt. Derselbe Board-Code kann dann für alle ElektorBus-Platinen wiederverwendet werden.

wareunabhängig entwickeln.
Da wir aber noch weitere ElektorBus-Boards mit anderer Funktionalität geplant haben, beschreiten wir hier einen noch besseren Weg. Die auf allen ElektorBus-Platinen wiederkehrenden Peripheriefunktionen wie RS485-Treiber, Test-LED und Test-Button kapseln wir in einem Boardfile „BoardEFL.h/.c“, das wir in der Codebasis im Unterordner „ElektorBusCore“ unterbringen. Die Spezial-Funktionen auf dem Board – wie hier der Schrittmotortreiber – werden dagegen in einem Erweiterungsfile initialisiert („Extension-EFL.h/.c“ im Unterordner „ElektorBusStepperMotor“). Hierzu stellen wir uns einfach vor, dass der ATmega328 über einen virtuellen Erweiterungsport mit dem Schrittmotortreiber verbunden ist. In **Bild 5** ist das schematisch dargestellt.

Schrittmotor-Code

Unter [2] kann man den entsprechenden Quellcode downloaden. Das EFL-Projekt lautet in diesem Fall ElektorBusStepperMotor, ein Klick auf ElektorBusStepperMotor.atsuo öffnet es in Atmel Studio. Im Projekt-Ordner „Hardware“ erkennt man das Boardfile BoardEFL.h/.c, mit einer Initialisierung der ElektorBus-Kernfunktionen und des (virtuellen) Erweiterungssteckers. Im Extension-file findet sich die Funktion Extension_Init(), die ja immer beim Start der Anwendung aufgerufen wird. In der Funktion wird ein Peripherie-Block für den Schrittmotortreiber angelegt. In die Board-pin-Tabelle (alles darüber im EFL-Zusatzdokument [6]) werden hierzu drei Pins des Erweiterungsports eingetragen, die für die Signale EN, CLK und DIR zuständig sind (das MODE-Signal wird in der Software nicht verwendet). Gleichzeitig tragen wir noch einen Block für das Sicherheits-Relais und einen Block für die beiden Positionsschalter ein. All dies lässt sich nun schön hardwareunabhängig ansprechen. Man muss dazu nicht mehr die Controllerpins wissen, sondern nur noch den Typ und die Nummer des Peripherieblocks.
Zwar könnten wir die drei Signale EN, CLK und DIR nun bereits unabhängig von der Board-Verdrahtung bedienen, doch können wir nicht immer davon ausgehen, dass sich jeder Schrittmotortreiber gleichartig über drei Eingänge ansteuern lässt. Damit die Anwendungsentwickler doch nicht wieder ein Datenblatt studieren müssen und der Code hardwareunabhängig bleibt, wurden im Extensionfile noch einige Low-Level-Funktionen für Schrittmotoren implementiert. So zum Beispiel das Setzen der Richtung mit einem Direction-Bit 0 oder 1:

```
void StepperMotorDirection(uint8
StepperMotorBlockIndex, uint8 Direction)
```

oder eine Bewegung um einen Schritt:

```
void StepperMotorStep(uint8
StepperMotorBlockIndex, uint8
MillisecondsDelay)
```

Diese Step-Funktion blockiert den Programmablauf. Für professionellere Ansprüche haben wir noch eine Funktion implementiert, die den Schrittmotor timergesteuert verfahren lässt. Diese Funktion erwartet noch ein Array als Parameter, das eine Geschwindigkeits-Rampe abbildet. Der Motor beginnt langsam, wird dann schneller und erreicht eine gleichbleibende Geschwindigkeit; am Ende wird die Rampe in umgekehrter Richtung durchfahren (Anmerkung: in diesem Modus ist allerdings keine Abfrage der Positionsschalter möglich).
Zuerst initialisiert man den Timer mit…

```
void StepperMotorTimerSetup(uint8
```

```
StepperMotorBlockIndex, uint8
MillisecondsDelay)
```

…wobei die Angabe einer Verzögerung in Millisekunden pro Schritt eine gewisse Basisgeschwindigkeit des Motors bestimmt. Als Timer sucht sich die Funktion übrigens den ersten noch nicht verwendeten 16-bit-Timer des Controllers heraus. Mit der Funktion…

```
void StepperMotorTimerSteps(uint8
StepperMotorBlockIndex, uint16 Steps,
uint8* RampData, uint8 RampMax, uint8
StepsShiftForNextRampIndex)
```

…startet man nun das timergesteuerte Verfahren des Motors. `RampData` und `RampMax` beschreiben die Rampe, ein Wert von 128 steht in diesem Array für die Basisgeschwindigkeit, kleinere Werte für höhere Geschwindigkeiten. Über die Variable `StepsShiftForNextRampIndex` kann man bestimmen, nach wie vielen Schritten der nächste Geschwindigkeitswert an die Reihe kommt. Es handelt sich hier um eine Zweierpotenz, bei einer 2 macht der Schrittmotor also immer vier Schritte mit gleicher Geschwindigkeit. Sind alle Werte aus dem Array abgearbeitet, fährt der Motor mit gleichbleibender Geschwindigkeit weiter; am Ende der Bewegung wird das Array in umgekehrter Reihenfolge ausgelesen.

Die Low-Level-Funktionen im Hardware-Layer nehmen dem Entwickler schon eine Menge Arbeit ab. Doch müsste er jetzt noch eine Funktion zur Abfrage der Positionsschalter schreiben und sich außerdem mit einer Kalibrierung der Motorbewegung herumschlagen. Beides haben wir deswegen in eine kleine Schrittmotorbibliothek gepackt, die auf der eben beschriebenen Low-Level-Schnittstelle aufsetzt. Wie immer bei den EFL-Libraries lassen sich hiermit gleich mehrere Peripherieblocks desselben Typs steuern, in diesem Fall bis zu acht Schrittmotortreiber auf einem Board. Der Quellcode (StepperMotorEFL.c) findet sich im Projektordner „Libraries“. Eine übersichtliche Funktionsbeschreibung kann man sich in der Doxygen-Dokumentation im Download-Zip ansehen [2].

Eine kleine Demo

Zu Test- und Demozwecken haben wir eine einfache Anordnung mit einem 12-V-Schrittmotor (Nanotec SP2575M0206-A), einem Positionszeiger und zwei Endschaltern gebaut (siehe **Bild 6**).

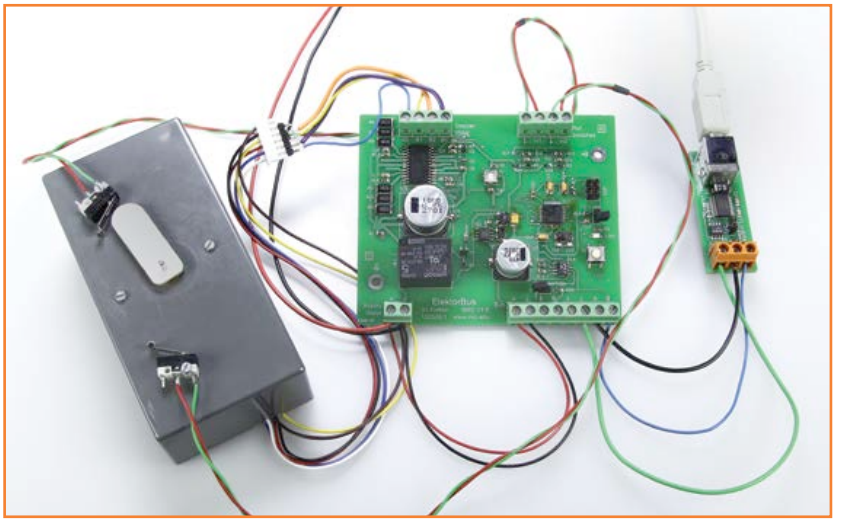


Bild 6. Demonstrator mit Positionsschaltern. Die Schrittmotorplatine ist über einen RS485/USB-Konverter mit dem PC verbunden.

Die Schrittmotorkarte wird über den ElektorBus mit dem altbekannten RS485/USB-Konverter und dieser wiederum mit einem PC verbunden.
Den eigentlichen Anwendungscode findet man im Hauptfile unseres Projekts (ElektorBusStepperMotor.c). Die immer gleich bleibende EFL-Main-Funktion und die Initialisierung der LEDButton-, UARTInterface- und ElektorBus-Bibliothek in der Funktion `ApplicationSetup()` wurden bereits beschrieben [6][7].

Mit…

```
StepperMotor_LibrarySetup(SwitchEventCall
back, 0, 0);
```

…initialisieren wir die Schrittmotortreiber-Bibliothek. Der erste Parameter benennt eine Funktion, die im Anwendungscode beim Erreichen eines der Endschalter aufgerufen wird. In unserem Fall haben wir diese Funktion (weiter unten im Hauptfile) so implementiert, dass die LED auf dem Board getoggelt wird. Die weiteren Parameter (0, 0) geben einen möglichen Versatz der Blocknummern zwischen dem Schrittmotortreiber, den Positionsschaltern und dem Sicherheits-Relais an. In unserem Fall ist der Schalterblock #0 und der Relaisblock #0 für den ersten Schrittmotortreiber #0 zuständig – doch das könnte ja auf anderen Boards anders sein.
Wenn der Anwender den Button auf dem Board betätigt, wird eine automatische Kalibrierung durchgeführt. Der Motor fährt dabei zuerst in die eine und dann in die andere Richtung, bis er jeweils einen Positionsschalter erreicht; dabei wird die Zahl der nötigen Schritte gezählt. Die

Library-Funktion `StepperMotorCalibration(…)` erkennt dabei auch automatisch, in welche Richtung der Motor bei einem Setzen des Direction-Bits 0 bzw. 1 tatsächlich fährt und merkt sich dies für spätere Zwecke.
Ab jetzt kann der Motor in eine bestimmte Position innerhalb des Verfahrbereichs gebracht werden, wobei 0 und 1023 die Endpositionen darstellen.

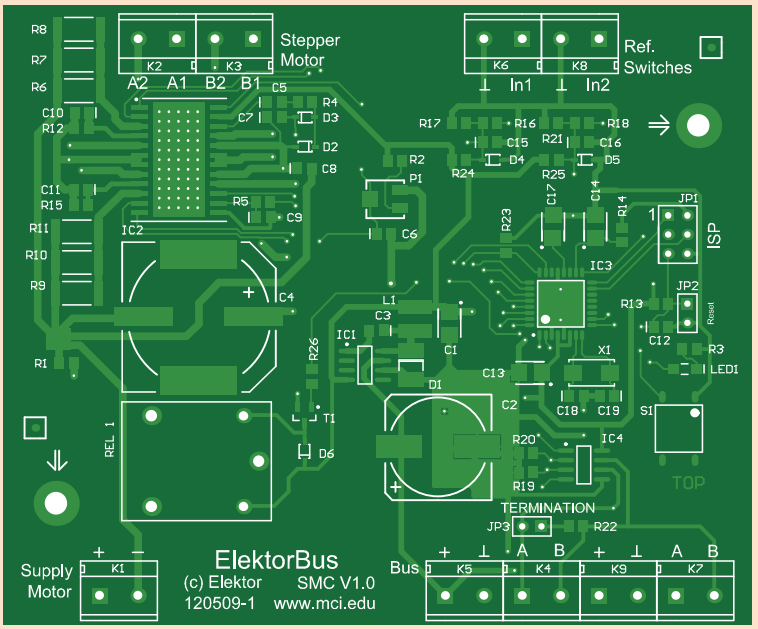
Ein Aufruf von…

```
StepperMotor_GotoMotorPosition(0, 512,
4);
```

…lässt den Schrittmotor in die Mittelstellung fahren, mit ungefähr 4 ms pro Schritt.

Wir wollen dies natürlich über den ElektorBus steuern. Schnell ist eine kleine HTML-Oberfläche programmiert, die im ElektorBusBrowser auf dem PC läuft (wie immer kann man dasselbe User-Interface dann auch auf einem Android-Smartphone oder -Tablet einsetzen [8][9]). Das HTML-File findet man im Ordner UIBus, den man sich einfach vom Download-Ordner aus auf den Desktop zieht. Der Rest ist ebenfalls wohlbekannt: ElektorBusBrowser.exe starten, COM-Port wählen und auf „Connect“ klicken, dann den Scheduler anwerfen. Die HTML-Buttons lösen nun Elektor-Bus-Messages vom PC zum Schrittmotor-Board aus, die gewünschte Position des Motors wird dabei auf Channel 0 als 10-bit-Wert übermittelt.

Stückliste



Widerstände:
(0805, wenn nicht näher spezifiziert)
R1 = 0 Ω
R2,R12,R15 = 39 k
R3 = 680 Ω
R4 = 100 Ω
R5,R23 = 100 k
R6..R11 = 1R0 (Vishay CRCW25121R00FKEG)
R13,R19,R20 = 10 k
R14 = 10 Ω
R16,R18 = 0 Ω (siehe Textkasten)
R17,R21 = nicht bestücken (siehe Textkasten)
R22 = 120 Ω
R24,R25 = 10 k optional (siehe Textkasten)
R26 = 4k7
P1 = Trimmer 10 k (POT4MM-2)

Kondensatoren:
(0805, wenn nicht näher spezifiziert)
C1 = 68 µ / 10 V Tantal (AVX TPSB686K010R0600)
C2 = 470 µ / 35 V Elko (Panasonic EEEFK1V471AQ)
C3,C5,C8,C12 = 10 n
C4 = 1000 µ / 50 V Elko (Panasonic EEVFK1H102M)
C6 = 68 n
C7 = 220 n
C9 = 5n6
C10,C11 = 1 n
C13,C14,C17 = 10 µ / 6V3 (AVX TCJA106M006R0300)
C15,C16 = 100 n
C18,C19 = 22 p

Induktivitäten:
L1 = 47 µH (744773147)

Halbleiter:
D1 = Schottky 1 A / 60 V (B160-13-F)
D2,D3,D6 = Diode 1N4148
D4,D5 = Zenerdiode 5V1 (BZX384-B5V1)
T1 = BC849B, SOT-23
LED1 = LED grün (5988270107F)
IC1 = LM2675M-5.0
IC2 = L6208D (SO24)
IC3 = ATmega328P-AU
IC4 = LT1785CS8

Außerdem:
JP1 = Stiftleiste 2x3, 2,54 mm
JP2,JP3 = Stiftleiste 1-reihig 2,54 mm
K1..K9 = Schraubklemme für Platinenmontage 5,08 mm
Rel1 = Relais SPDT (Omron G5LA145DC)
S1 = Taster (Omron B3S-1000)
X1 = 16 MHz Quarz, 50 ppm, 16 pF (Epson Toyocom FA-365)
Platine 120509-1

In der Firmware empfangen wir die Messages über die ElektorBus-Bibliothek, welche die Funktion `ProcessPart()` aufruft. Diese verarbeitet die eingehenden Nachrichten-Parts und lässt den Motor in die gewünschte Position fahren.

Wie immer stellt diese Demosoftware nur eine Anregung für weitere Experimente dar. Eine praktische Anwendung könnte die Steuerung eines Rollladens oder einer Lamellenverschattung sein, die ein Zimmer (teilweise) verdunkelt, wenn es zu hell wird. Hard- und Software für einen ElektorBus-Fotosensor haben wir ja bereits vorgestellt [6][10].

In der nächsten Ausgabe geht es weiter mit dem schon länger angekündigten Xmega-Webserver-Board, das ebenfalls einen RS485-Anschluss aufweist und sich für schöne ElektorBus-Applikationen eignet.

(120509)

Weblinks

[1] www.mci.edu

[2] www.elektor.de/120509

[3] www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/CD00002294.pdf

[4] www.elektor.de/110258

[5] www.elektor.com/elektorbus

[6] www.elektor.de/120668

[7] www.elektor.de/130154

[8] www.elektor.de/110405

[9] www.elektor.de/120097

[10] www.elektor.de/110428

Smartphone als Fernbedienung

Transmitter und App für Android-Geräte

Von **Peter Zirngibl**
(info@pezitec.com)

Die schönsten universellen Fernbedienungen sind zweifelsohne die mit einem großen Touchscreen. Genau wie bei einem Smartphone. Was liegt also näher, als dieses Multifunktionswunder auch als Remote Control für die A/V-Apparatur im Wohnzimmer einzusetzen?

Man kann mit ihnen im Internet surfen, mailen, simsen und chatten, Audio aufnehmen und hören, Fotos und Videos machen und betrachten, Radio hören und fernsehen, navigieren, daddeln, ja, sogar telefonieren. Zehntausende von Apps machen das Smartphone zur eierlegenden Wollmilchsau. Da nimmt es natürlich kein Wunder, dass es auch Applikationen gibt, mit denen sich modernes A/V-Equipment per Smartphone fernsteuern lässt – vorausgesetzt, dass diese Geräte auch mit einem WLAN in Kontakt stehen. Geräte, die nur eine althergebrachte Infrarot-Fernbedienung besitzen und an kein Netzwerk angeschlossen sind, kann man nicht so ohne weiteres mit dem Smartphone steuern, denn (noch) verfügen diese nicht über eine IR-Sendemöglichkeit.
Was fehlt, ist ein intelligenter Adapter, der auf der einen Seite mit dem Smartphone über Bluetooth,

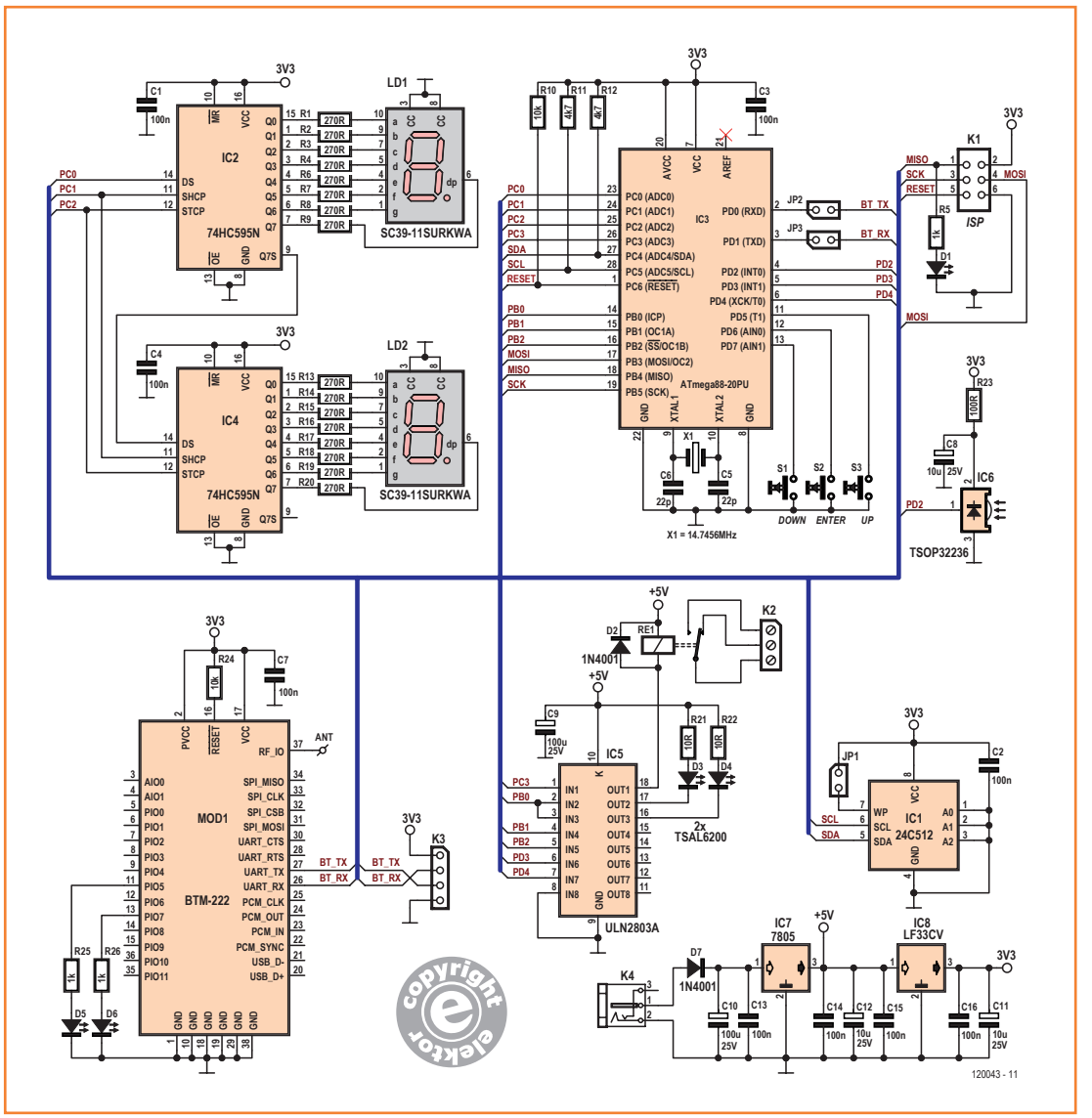


Bild 1.
Der Hardwareaufwand hält sich dank des Bluetooth-Moduls in Grenzen.

auf der anderen mit dem zu steuernden Gerät über Infrarot kommuniziert. Intelligent bedeutet, dass erst einmal ein Lernprozess notwendig ist. Denn wie soll der Adapter sonst wissen, um welche Fernbedienungsvariante es sich handelt und welcher Befehl welche Aktion auslösen soll? Einfache und billige käufliche Universal-Fernbedienungen sind zwar für tausende von Gerätetypen vorprogrammiert, kennen dabei aber nur einige wenige Standard-Befehle wie PLAY oder STOPP. Die Befehle sind nicht editierbar und eine Lernfunktion gibt es erst bei teureren Modellen. Manchmal, bei Luxus-Modellen, ist auch die Eingabe von Makros möglich, Ketten von oft nacheinander ausgeführten Einzelbefehlen (zum Beispiel: DVD-Film sehen = DVD-Player, TV, AV-Receiver einschalten, Video- und Audio-Kanäle korrekt einstellen). Dies soll auch mit unserem Bluetooth-IR-Adapter möglich sein.

Heutzutage lassen sich teure Luxus-Universal-Fernbedienungen komfortabel über den PC programmieren. Um den Programmieraufwand in Grenzen zu halten, haben wir uns aber dafür entschieden, auf eine Vor- und eine PC-Programmierung zu verzichten. Sämtliche Befehle, die der Adapter kennen muss, soll es von der originalen Fernbedienung lernen. Das heißt, dass die IR-Diode der Fernbedienung vor den IR-Empfänger des Adapters gehalten wird und dann die Befehle übermittelt. Wie das genau funktioniert, dazu später mehr, zunächst ein Blick auf die Hardware.

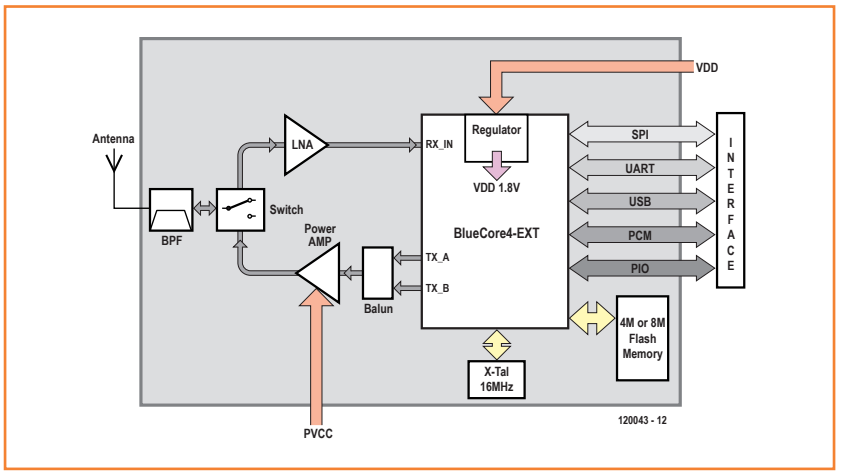


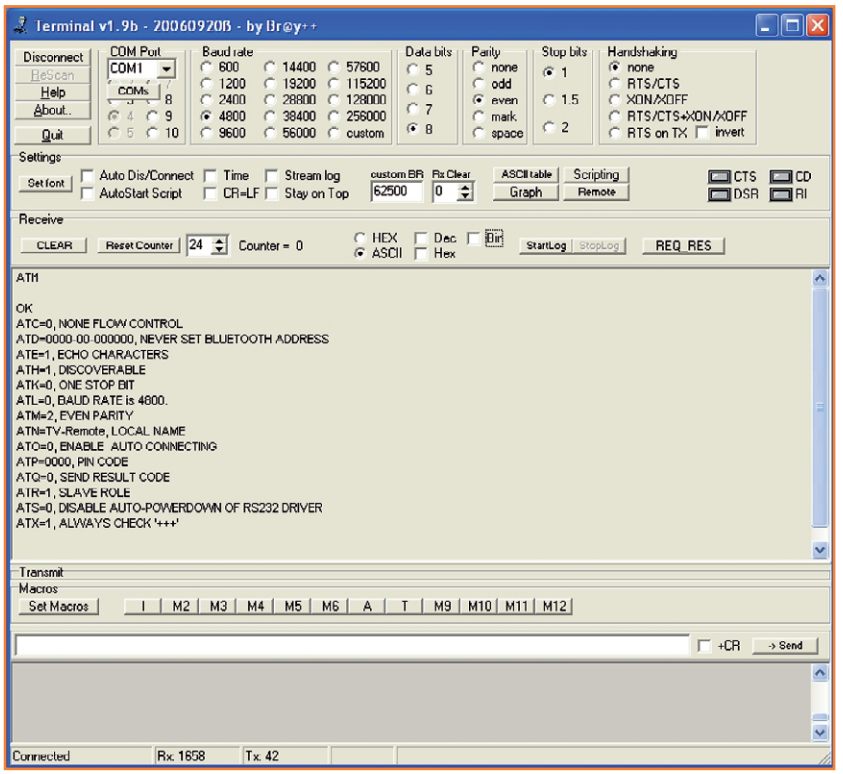
Bild 2.
Blockschaltung des Bluetooth-Moduls BTM-222.

Bild 3.
Das Bluetooth-Modul muss mit einem Terminal-Programm auf die gezeigten Parameter eingestellt werden.

Ein Controller mit Funkverbindung

Die Schaltung in **Bild 1** zeigt, dass die Elektronik für den Adapter nicht sehr aufwändig ist. Sie besteht aus einem kleinen Atmel-Mikrocontroller ATmega88, an den einige wenige Baugruppen angeschlossen sind, als da wären:

- Ein EEPROM 24C512 mit 512 kBit Speicher und I²C-Interface zur Ablage der programmierten Codes
- ein Infrarot-Empfangsmodul TSOP32236
- zwei IR-Sendedioden TSAL6200 über ein Leistungstreiber-IC ULN2803
- eine zweistellige 7-Segmentanzeige mit einem 8-bit-Schieberegister HC595 zur Anzeige der programmierten Codes und des Programmierstatus
- drei Taster UP, DOWN und ENTER, die beim Programmiervorgang von Bedeutung sind
- eine In-System-Programmierschnittstelle K1
- Ein Relais mit Wechselkontakt, das ebenfalls über den Leistungstreiber angesteuert wird und
- eine Spannungsstabilisierung für +5 V und 3,3 V



All dies ist nichts Besonderes und schon tausendfach erprobt, doch die eigentliche Sensation dieser Schaltung ist das Bluetooth-Modul BTM-222 von Rayson. Eine Beschreibung des Moduls sowie das Datenblatt finden Sie unter [1]. Es ist klein, handlich, relativ einfach zu bedienen und vor allem mit einem Preis von ungefähr 10 € bezahlbar. In Zeiten, in denen Computer gleich welcher Couleur immer weniger drahtgebundene Schnittstellen aufweisen, stellt ein solches Modul eine Möglichkeit dar, Peripheriegeräte anzusteuern. Das BTM-222 ist ein Klasse-1-Gerät und hat eine Reichweite von bis zu 100 m. Es weist, wie seine Blockschaltung in **Bild 2** zeigt, mehrere serielle Schnittstellen auf, von denen der UART für den Betrieb an Mikrocontrollern prädestiniert ist. Über den UART (und die USB-Schnittstelle) garantiert das BTM-222 die volle Datenrate von 921 kBit/s. Soll das BTM-222 lediglich über den UART mit dem Mikrocontroller kommunizieren, kann das Modul weitestgehend ohne externe Beschaltung bleiben. Es ist werksseitig auf die „klassischen" Parameter 8N1 eingestellt:

Baudrate 19.200 Baud
8 Datenbits
Parität: keine
1 Stoppbit

Der Mikrocontroller muss dazu lediglich den entsprechenden UART-Kanal öffnen. Bei Bedarf können die Parameter und andere Eigenschaften des Moduls über so genannte AT-Befehle angepasst werden. Auskünfte darüber erteilt das Datenblatt [1]. Die Einstellungen werden in einem internen Flash-Speicher abgelegt. Der „blaue" Kern des Moduls wird von einem internen 16-MHz-Oszillator getaktet. Das Ausgangssignal wird über ein

Symmetrierglied (Balun) und einen Leistungsverstärker mit +18 dBm zur Antenne geleitet. Zum Empfang schaltet das BTM-222 die Antenne auf den Low-noise-Verstärker LNA um. Ein Bandpassfilter BPF verbessert die Empfangseigenschaften. Eine externe Antenne besitzt das BTM-222 übrigens nicht, sie kann aber leicht durch ein geeignet langes Stückchen Leiterbahn oder – wie hier – einen kurzen Draht realisiert werden.
Das Modul verfügt über Anschlüsse, die den Datenstatus (LED D5 an Pin 11) und den Verbindungsstatus (LED D6 an Pin 13) anzeigen. Außerdem könnte (hier nicht verwendet) eine LED an Pin 14 als Betriebsspannungsanzeige angeschlossen werden.

BTM222 einrichten

Zuerst muss man das Bluetooth-Modul mit den richtigen Parametern einrichten. Dazu entfernt man die Jumper JP2 und JP3 und unterbricht damit die serielle Verbindung zum Controller. Verbinden Sie das BTM-222 und den PC zum Beispiel mit einem TTL-232R-Kabel von FTDI über K3. Der VDD-Anschluss dieses Kabels liefert allerdings 5 V, und diese Spannung darf nicht zum BTM-222 gelangen (dessen Betriebsspannung 3,3 V beträgt und 3,6 V auf keinen Fall überschreiten darf). Also lässt man einfach Pin 1 von K3 unverbunden und versorgt das Bluetooth-Modul über die Platine.
Starten Sie dann ein Terminal-Programm auf Ihrem PC wie Hyperterminal oder Hterm (**Bild 3**), wählen Sie den (virtuellen) COM-Port, an dem das Bluetooth-Modul angeschlossen ist und die oben genannten Werkseinstellungen des BTM-222 für die serielle Schnittstelle. Prüfen Sie, ob das Modul auf ATI1 reagiert. Dieser Befehl soll die Einstellungen des Moduls auf dem Monitor anzeigen. Wenn es überhaupt keine Antwort gibt, sind die 3,3 V-Versorgungsspannung, die COM-Port-Einstellungen und die Verbindung zu K3 (TxD/RxD vertauscht?) zu überprüfen. Wenn Sie sich über die aktuellen Einstellungen des Moduls nicht sicher sind oder wenn irgend etwas bei der Einrichtung des BTM-222 schief geht, kann das Modul auf die Werkseinstellungen zurückgesetzt werden, indem man PIO4 für mindestens drei Sekunden auf High legt.
Ändern Sie jetzt die UART-Einstellungen auf 4800 Baud (ATL = 0), gerade Parität (ATM = 2), keine Flusskontrolle (ATC = 0). Passen Sie die Einstellungen des Terminal-Programms nach jeder Parameteränderung an. Anschließend ändern Sie Parameter wie Modulname oder den PIN-Code für die Bluetooth-Verbindung nach Wunsch.
Schalten Sie nun das Smartphone (oder ein anderes Android-Gerät) ein, überzeugen Sie sich, dass das BTM-222 gefunden wurde und stellen Sie die Verbindung mit der PIN her. Wenn alles funktioniert, schalten Sie die Fernbedienungsschaltung aus, entfernen die Verbindung zum PC und stecken die beiden Jumper JP2/JP3.

Die Remote-App

Mit der Installation der App *Remote_Control.APK* wird das Android-Handy endgültig zur Fernbedienung. Starten Sie die App und führen ein Pairing mit dem BTM-222-Modul durch. Wählen Sie mit SELECT DEVICE das Bluetooth-Modul und stellen mit CONNECT eine Verbindung her. Die LED D6 leuchtet stetig und zeigt, dass das Modul mit dem Android-Gerät verbunden ist. D5 blinkt, wenn Daten vom Android-Gerät empfangen werden.

Bild 4. So präsentiert sich die Fernbedienungs-App auf dem Smartphone.

Dem Adapter wird durch Übertragung eines Wertes (0...99) plus eines Line-Feed-Zeichens (/n) mitgeteilt, welche Speicherstelle abgespielt werden soll. Im Abspielmodus können die vom Bluetooth-Modul empfangenen Befehle ausgewertet werden. Das Prinzip ist ganz einfach: „2/n“ wird empfangen, der Adapter springt zu Speicherstelle 2 und spielt den dort gespeicherten Befehl ab (und erhöht damit die Lautstärke bei Gerät 1). Eine Besonderheit stellen die Befehle 73 und 74 dar. Hier wird kein IR-Signal ausgegeben, sondern das Relais Re1 für 2 s (73) betätigt beziehungsweise umgeschaltet (74). An K2 kann man externe Geräte anschließen und ein- und ausschalten.
Bild 4 zeigt die App in ihrer ganzen Pracht auf dem Smartphone. Mit den oberen Pfeilen wählt man fünf verschiedene Geräte aus (1...5). Man

kann für jedes Gerät einen Namen in die Textbox eintragen und mit SAVE speichern. Der Name bleibt dann erhalten und wird beim nächsten Start der App wieder angezeigt. Zusätzlich kann man die Funktionen 1...10 beschriften. Dazu drückt man den EDIT-Button, wählt die gewünschte Taste aus und gibt einen Text ein. Mit dem Druck auf den SAVE-Button wird der Text gespeichert. Die Texte sind bei allen Geräten gleich.

Achtung Aufnahme!

Die Software des Controllers ist zweigeteilt in den Programmier- und den Betriebsmodus. Im Programmiermodus werden Infrarotbefehle einer Fernbedienung aufgezeichnet. So ist die Vorgehensweise:

Als erstes sollte man das serielle EEPROM löschen. Dazu müssen alle drei Taster gleichzeitig gedrückt und erst dann die Versorgungsspannung angelegt werden. Wenn in der 7-Segment-Anzeige `dE.` (delete) erscheint, kann man die Taster wieder loslassen. Der Löschvorgang dauert durchaus ein paar Minuten, dann erscheint `00` in der Anzeige.

Um zum Aufzeichnen der Befehle in den Programmiermodus zu gelangen, hält man die ENTER-Taste gedrückt (Port D6 auf GND), bevor man die Versorgungsspannung anlegt. Auf der 7-Segment-Anzeige blinkt drei Mal kurz `Pr.` (für Programmiermodus) auf. Jetzt muss ein Speicherplatz 0...99 über die UP/DOWN-Taster (Port D5 und Port D7) ausgewählt und mit ENTER bestätigt werden. Danach zeigt das Display mit – die Aufnahmebereitschaft der Schaltung. Die Fernbedienung muss in einem Abstand von 5...10 cm an den IR-Empfänger gehalten und die gewünschte Funktion einmal (einmal!) auf der originalen Fernbedienung betätigt werden. Um keine Störsignale aufzuzeichnen, sollte dieser Vorgang flott

Stückliste

Widerstände:

R1..R4,R6..R9,R13..R20 = 270 Ω
R5,R25,R26 = 1 k
R10,R24 = 10 k
R11,R12 = 4k7
R21,R22 = 10 Ω
R23 = 100 Ω

Kondensatoren:

C1,C2,C3,C4,C7,C13,C14,C15,C16 = 100 n
C5,C6 = 22 p keramisch
C8,C11,C12 = 10 µ/25 V radial
C9,C10 = 100 µ/25 V radial

Halbleiter:

D1,D5,D6 = LED rot, 3 mm, low current
D2,D7 = 1N4001
D3,D4 = IR-Sendediode 940 nm TSAL6200 (Vishay)
LD1,LD2 = 7-Segment-LED-Display 10 mm (Kingbright SC39-11SURKWA)
IC1 = CAT24C512LI-G (On Semiconductor)
IC2,IC4 = 74HC595N
IC3 = ATmega88-20PU (Atmel), programmiert: 120043-41 [2]
IC5 = ULN2803APG
IC6 = IR-Empfänger 36 kHz TSOP32236 (Vishay)
IC7 = 7805
IC8 = LF33CV (ST)
Mod1 = Bluetooth-Modul BTM-222 (Rayson)

Außerdem:

X1 = Quarz 14,7456 MHz
Re1 = 6V-Relais SPDT (1x um), Finder 43.41.7.006.2000
S1...S3 = Taktiler Schalter SPNO, rund

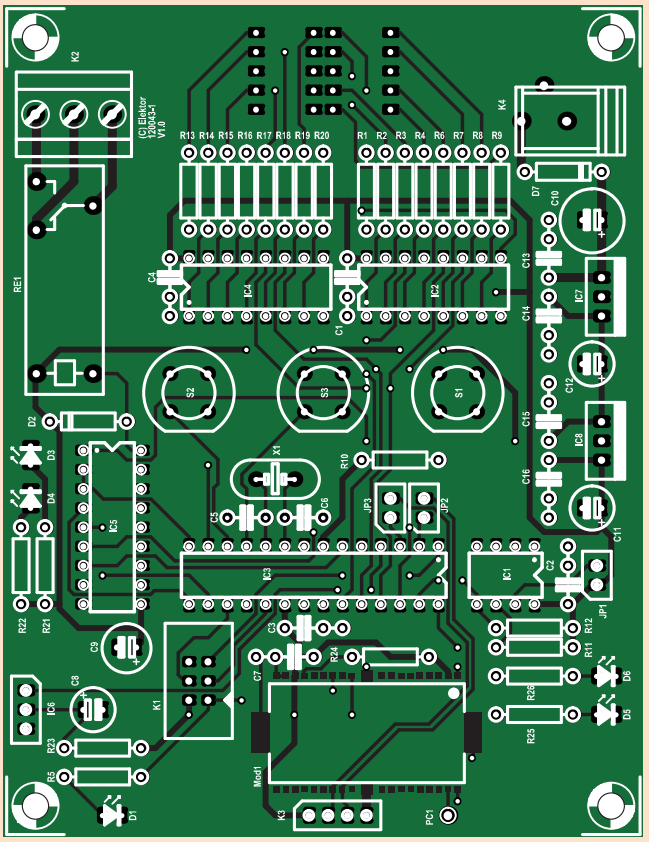


Bild 5.
Layout der Platine. Alle Bedienelemente sind auf der Platinenrückseite untergebracht.

vonstattengehen. Als Bestätigung für die Programmierung zeigt die Anzeige `io..` Pro Gerät ist die Eingabe eines Makros möglich, das bis zu sechs einzelne Befehle enthalten kann. Die Makro-Befehle werden gespeichert wie die anderen Befehle auch.

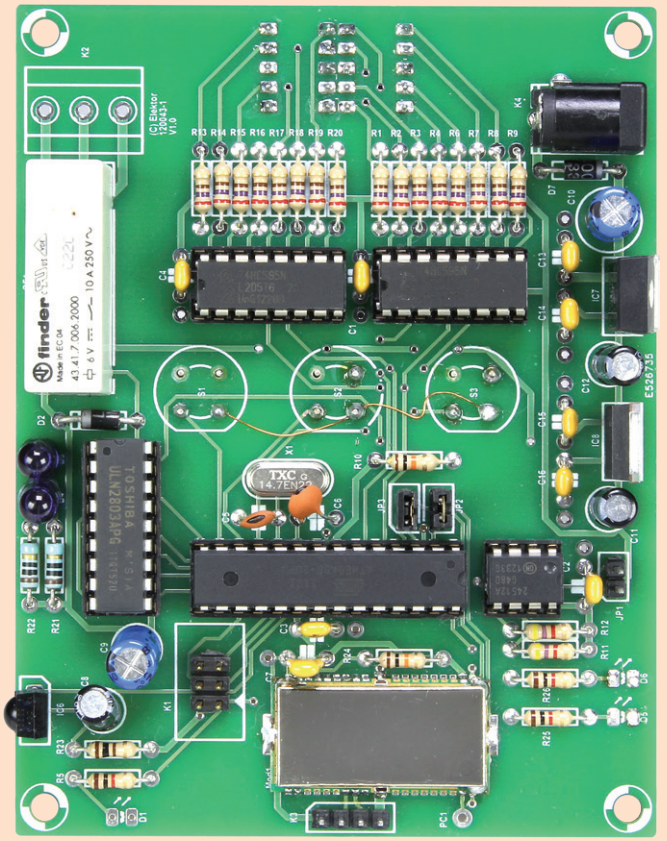
Bei der Aufnahme der IR-Fernbedienungsbefehle wird die Länge der Impulse gemessen und als *Unsigned integer Variablen* zwischengespeichert und dann im EEPROM abgelegt. Will man nach der Programmierung noch einen weiteren Befehl programmieren, muss man während der Bestätigungsmeldung (`io.` blinkt) die ENTER-Taste wieder gedrückt halten, da nach der Programmierung ein µC-Reset ausgeführt wird.

Wird beim Reset keine Taste gedrückt, springt das Programm automatisch in den Abspielmodus. Im Abspielmodus wählt man mit den UP/DOWN-Tasten den gerade aufgenommenen Befehl aus und gibt ihn mit der ENTER-Taste über die IR-Diode wieder aus. So kann man testen, ob der Befehl auch korrekt aufgenommen wurde.

Auf die Platine

Die Platine für den Fernbedienungsadapter ist selbstverständlich über Elektor [2] erhältlich, genauso wie ein programmierter Controller. Unter dem gleichen Link finden Sie auch die App compiliert und als Source-Code. Ein Blick auf das Platinenlayout in **Bild 5** beruhigt die Nerven ein wenig: Zwar ist das Bluetooth-Modul ein SMD, die Beinchen befinden sich aber glücklicherweise in einem für Lötkolben noch geeigneten Abstand von 1,25 mm. Somit dürfte das Modul mit gebotener Sorgfalt, aber ohne große Probleme auf die Platine gelötet werden können. Ein wenig problematisch ist aber die Orientierung des Moduls. Pin 1 ist dort, wo ein kleiner runder Punkt im Metall der

JP1...JP3 = 2-poliger Pfostenverbinder plus Jumper
K1 = 2x3-poliger Pfostenverbinder
K2 = 3-polige Platinenanschlussklemme RM 5
K3 = 4-poliger SIL-Verbinder
K4 = Kleinspannungsbuchse 2,1 mm
Platine 120043-1 [2]



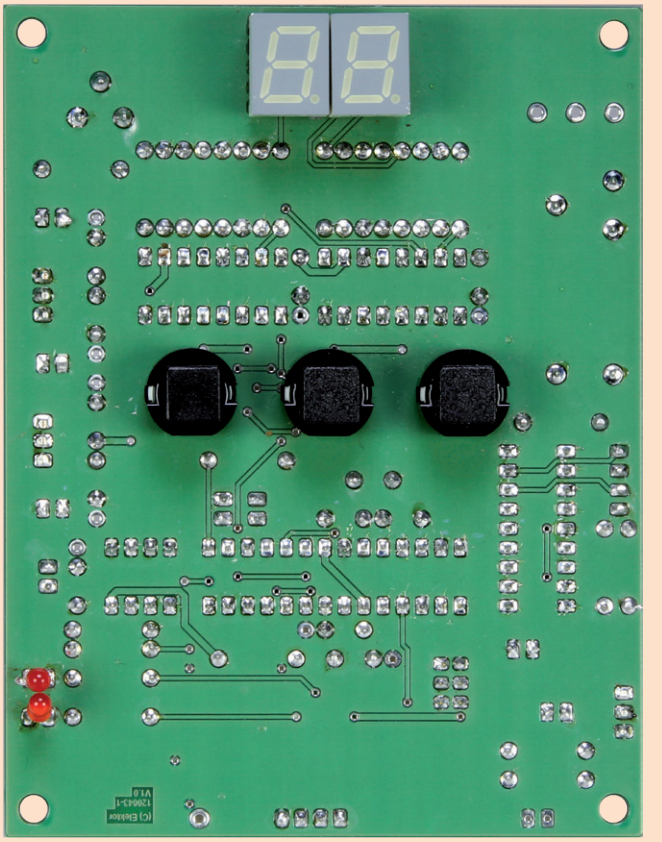


Tabelle 1. Belegung der Speicherstellen mit Funktionen

Funktionen	Gerät				
	1	2	3	4	5
MUTE	0	8	16	24	32
ON/OFF	1	9	17	25	33
VOL+	2	10	17	26	34
VOL-	3	11	19	27	35
PRG+	4	12	20	28	36
PRG-	5	13	21	29	37
AUX	6	14	22	30	38
MAKRO	7, 75-79	15, 80-84	23, 85-89	31,90-94	39, 95-99
Funktion 1	40	50	60		
Funktion 2	41	51	61		
Funktion 3	42	52	62		
Funktion 4	43	53	63		
Funktion 5	44	54	64		
Funktion 6	45	55	65		70
Funktion 7	46	56	66		71
Funktion 8	47	57	67		72
Funktion 9	48	58	68		73 [1]
Funktion 10	49	59	69		74 [2]

[1] Relais für 2 s ein [2] Relais toggle

Abschirmung (an der Seite, wo die Drahtantenne sitzt) erkennbar ist. Dieser Punkt sollte zu PC1 auf der Platine weisen. Die dicken Pads in der Mitte der langen Seiten und an den Stirnseiten führen Massepotential und werden mit der umlaufenden Massebahn des Moduls verbunden.

Wenn das Modul sicher verlötet ist, kann man die übrigen Bauteile bestücken, die sämtlichst für Durchsteckmontage geeignet sind. Dabei dürfen die ICs durchaus in Fassungen eingesetzt werden. Die beiden Displays, die drei Taster sowie D5 und D6 gehören auf die Platinenunterseite. Damit befinden sich alle „HIDs“ auf einer Seite der Platine und das ganze lässt sich schön in ein Gehäuse einbauen.

Als Antenne dient ein simpler Draht. Rein rechnerisch sollte dieser bei 2,4 GHz eine Länge von 3,1 mm (λ/4) haben, aber der Wert ist nicht so kritisch, zumal es bei dieser Anwendung ja nicht um große Reichweiten geht. Hauptsache, die Antenne ragt beim Einbau in ein Metallgehäuse aus selbigem heraus.

(120043)

Weblinks

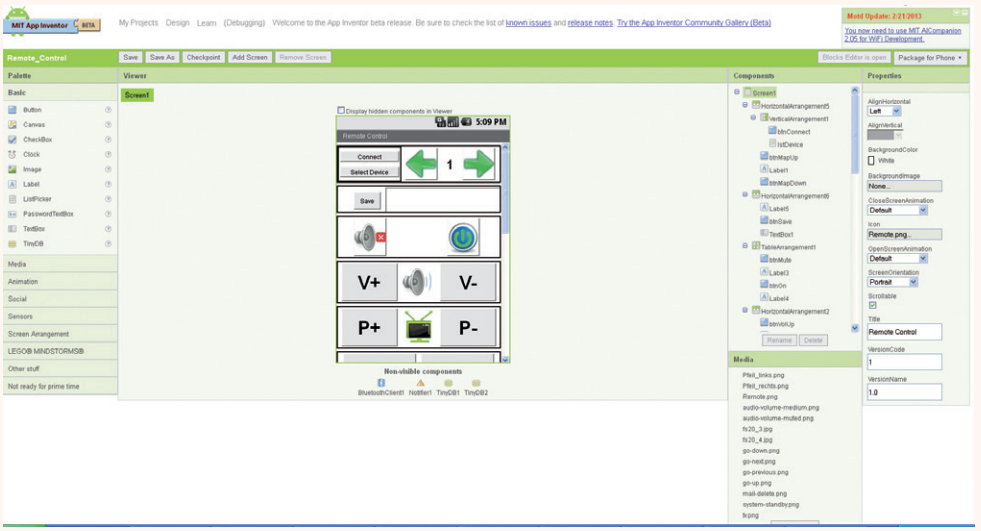
[1] www.mikrocontroller.net/wikifiles/f/fc/BTM222_DataSheet.pdf

[2] www.elektor.de/120043

[3] http://appinventor.mit.edu

App editieren

Die App wurde mit dem App-Inventor des MITs [3] entwickelt. Wenn Sie etwas ändern oder bearbeiten möchten, müssen Sie ein Gmail-Konto einrichten und auf der Webseite [3] einloggen. Laden Sie *Remote_Control.ZIP* unter „My Projects“ des App-Inventors in „More Options“ -> „Upload Source“ hoch. Die App kann dann bearbeitet, online compiliert und die daraus resultierende Datei *„Remote_Control.APK“* auf den PC heruntergeladen oder direkt auf dem Android-Gerät installiert werden.



Ambience Lighting Controller

Atmosphäre schaffen mit RGB-LED-Strips

Von **Goswin Visschers** (NL)

Regenbogenfarbige LED-Strips sind für stimmungsvolle Beleuchtungen die idealen Lichtquellen. Der Ambience Lighting Controller verhilft nicht nur dazu, Wunschfarben zu kreieren, er kann auch programmgesteuerte Lightshows ablaufen lassen. Durch die mobile Stromversorgung ist er überall einsatzfähig, auch dort, wo eine Netzsteckdose außer Reichweite ist.

Ursprünglich wurde der Ambience Lighting Controller zum Steuern der LED-Strips eines bekannten schwedischen Möbelhauses entworfen. Weil mit dem originalen Vorschaltgerät nur wenige Farben lediglich von Hand einstellbar waren, hatte der Autor die Idee, die in den Vielfarben-LED-Strips steckenden Möglichkeiten kreativer zu nutzen. Das Ergebnis wird hier vorgestellt: Der Ambience Lighting Controller kann alle RGB-LED-Strips steuern, die über eingebaute Strombegrenzungswiderstände an 12-V-Gleichspannung betrieben werden.

Am originalen Einsatzort der LED-Strips war kein Anschluss an das Stromnetz möglich. Deshalb ist die Schaltung so ausgelegt, dass sie an einem 12-V-Akku arbeitet. Die wichtigsten Eigenschaften sind nebenstehend zusammengefasst.

Schaltung

Möglicherweise sieht die Schaltung in Bild 1 kompliziert aus, sie ist aber überschaubar. Steuerzentrale ist ein Mikrocontroller PIC16F887 von Microchip. Die Hauptgründe für die Wahl dieses Typs waren das interne EEPROM zum Speichern der Lightshow-Programme, die Anzahl der Ein- und Ausgänge (I/O) sowie der integrierte A/D-Wandler (ADC). Ferner kann dieser Typ in einem speziellen ADC-Modus arbeiten, bei dem die ADC-Eingänge an RA0 und RA1 liegen. Anders als beim PIC16F877A sind Referenzspannungen an RA2 und RA3 nicht erforderlich. Mit Schalter S1, verbunden mit RC3, kann die Darstellung des Akku-Ladezustands auf dem LC-Display umgeschaltet werden. Kontaktleiste K6 ist der ICSP-Anschluss für die In-Circuit-Programmierung des Mikrocontrollers.

Quarz X1 ist dafür verantwortlich, dass der Mikrocontroller mit 20 MHz getaktet wird. Die für eine LED-Steuerung hoch erscheinende Taktfrequenz ist notwendig, weil der Mikrocontroller die Frequenz intern durch 4 teilt. Da die Pulsbreitensteuerung (PWM) der LED-Strip-Leuchtstärken von der Software übernommen wird, liegt die Taktfrequenz 5 MHz fast an der unteren Grenze. Das alphanumerische LC-Display ist ein Standard-Typ mit 2 · 16 Zeichen, es gehört zum ständigen Sortiment des Elektor-Shops. Für andere Typen lässt sich die Polarität der Spannung für die Hintergrundbeleuchtung mit den Jumpern J1 und J2 umschalten. Der Mikrocontroller steuert das Display über Port RB. Wenn die Bedienelemente

Eigenschaften

- Betriebsspannung 11...15 V
- Kompensation der Leuchtstärke bei sinkender Betriebsspannung
- LC-Display, zwei Zeilen mit 16 Zeichen
- Farbpalette mit 13 Farben durch Steuern der RGB-Werte
- Drei programmgesteuerte Lightshows, bestehend aus 20 Farbwechseln. Farbhaltezeit bis 255 s, Farbübergangszeit bis 255 s, beide einstellbar in Schritten von 1 s.
- Unendliches Wiederholen des gewählten Programms
- Abschalten der LED-Strips und akustisches Signal bei kritischem Sinken der Betriebsspannung
- LED-Anzeige für die Restladung des Akkus
- Integrierte Akkuladeschaltung mit Übergang auf Erhaltungsladung
- Sperre gegen unabsichtliche Änderungen der Farbeinstellungen und Programme
- Steuerung vom PC über USB/RS232-Konverter möglich

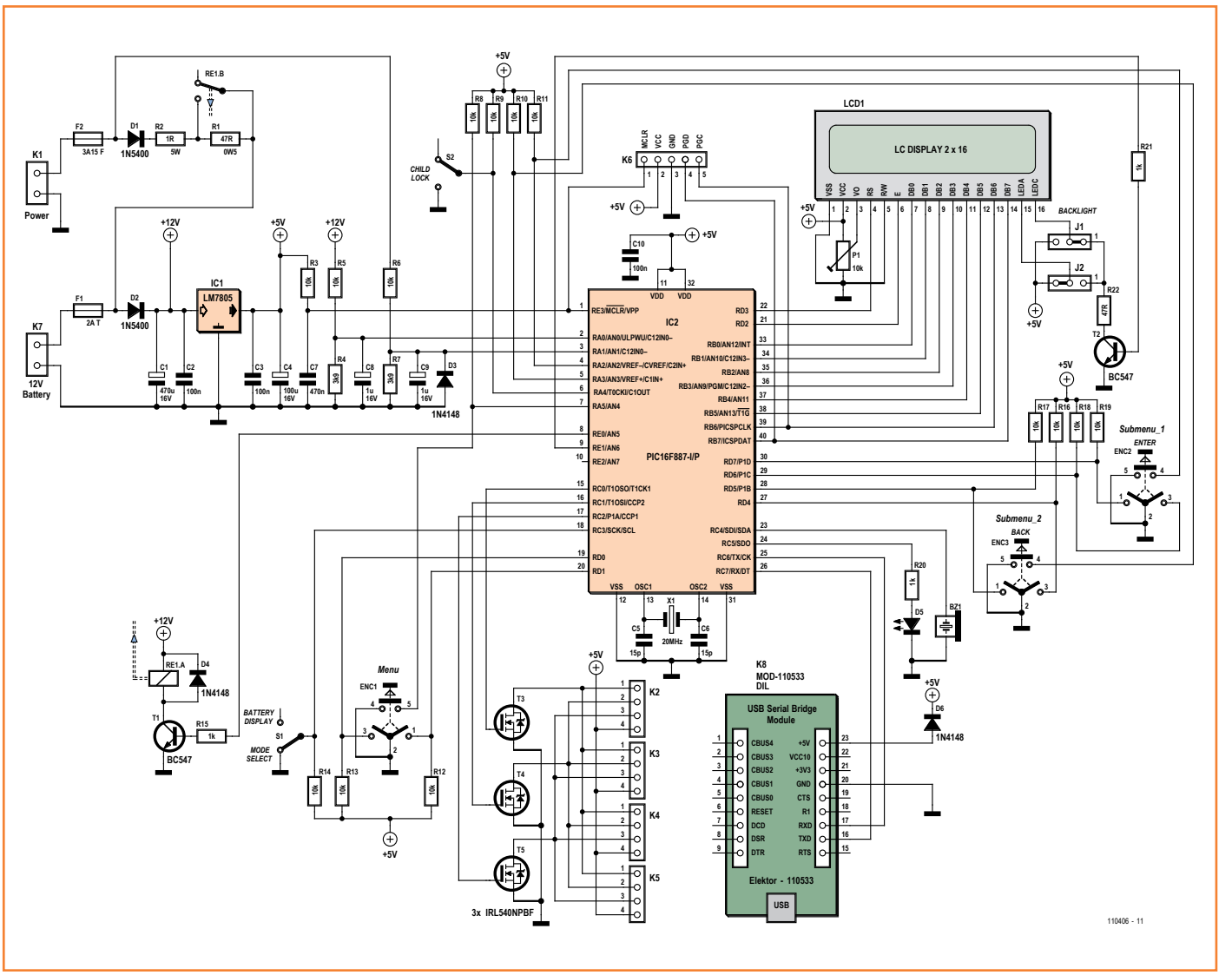


Bild 1.
Zentrale Schaltstelle ist ein Mikrocontroller des Typs PIC16F887.

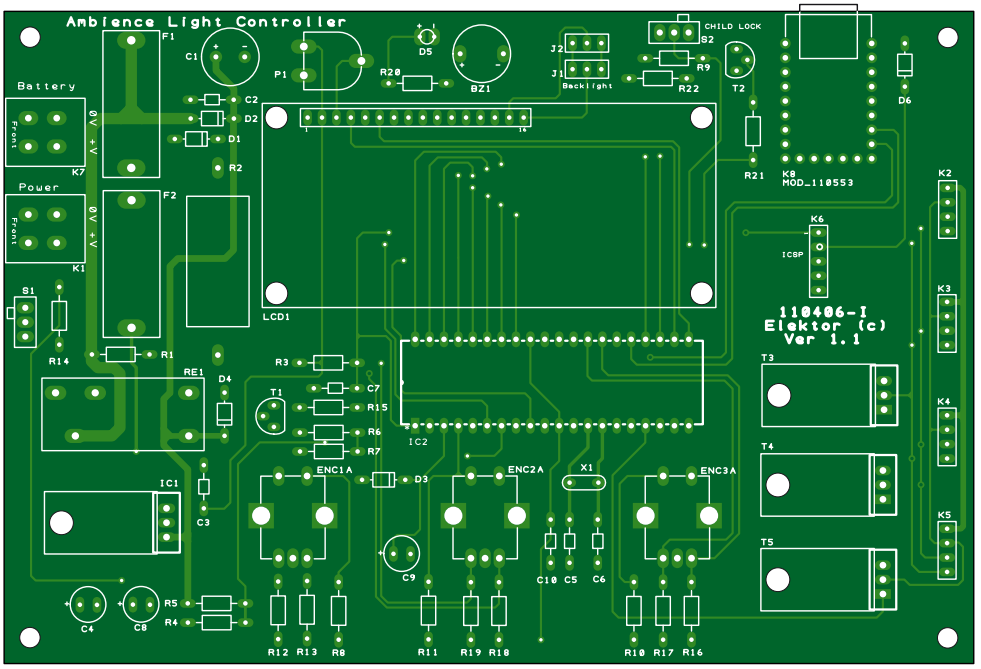


Bild 2.
Die Bedienelemente und das Display sind auf der Platine untergebracht.

länger als 10 s in Ruhe sind, schaltet der Mikrocontroller über T2 die Hintergrundbeleuchtung ab. Der Display-Kontrast ist mit Trimmpoti P1 einstellbar. Das LC-Display wird hier nicht, wie üblich, im 4-bit-Modus gesteuert. Weil der Mikrocontroller genügend I/O-Leitungen bereitstellt, konnte der vielseitigere 8-bit-Modus gewählt werden.
Wie schon erwähnt, werden die Helligkeiten der

Stückliste

Widerstände:
R1 = 47 Ω/0,5 W
R2 = 1 Ω/5 W
R3,R5,R6,R8...R14,R16...R19 = 10 k
R4,R7 = 3k9
R15,R20,R21 = 1 k
R22 = 47 Ω
P1 = 10 k Trimmpoti, liegend

Kondensatoren:
C1 = 470 µ/16 V stehend
C2,C3,C10 = 100 n
C4 = 100 µ/16 V stehend
C5,C6 = 15 p
C7 = 470 n
C8,C9 = 1 µ/16 V stehend

Halbleiter:
D1,D2 = 1N5400
D3,D4,D6 = 1N4148
T1,T2 = BC547B
T3,T4,T5 = IRL540 (International Rectifier, Farnell 8651078)
IC1 = LM7805
IC2 = PIC16F887 (programmiert, EPS 110406-41)
D5 = LED 3 mm, rot

Außerdem:
X1 = Quarz 20 MHz
F1 = Sicherung 2 AT mit Halter für Platinenmontage
F2 = Sicherung 3,15 AT mit Halter für Platinenmontage
BZ1 = Beeper (Gleichstrom, mit internem Tongeber)
RE1 = Relais 12 V mit Wechselkontakt, Schaltstrom mindestens 2 A (z. B. Finder 40.31.7.012.0000, Farnell 1169158)
MOD1 = Elektor USB-FT232R Breakout-Board [1]
S1,S2 = Schiebeschalter, gewinkelt für Platinenmontage (z. B. C&K OS102011MA1QN1, Farnell 1201431)
S3,S4,S5 = Drehencoder mit Drucktaster (z. B. Alps EC12E2424407, Farnell 1520813)
K1,K7 = Schraubklemmverbinder 2-polig, Raster 5 mm
K2,K3,K4,K5 = Stiftkontaktleiste 4-polig, Raster 2,54 mm
K6 = Stiftkontaktleiste 5-polig, Raster 2,54 mm
LCD1 = LC-Display 2 · 16 Zeichen, mit Hintergrundbeleuchtung (z. B. Elektor 120061-71)
J1,J2 = Stiftkontaktleiste 3-polig mit Jumper, Raster 2,54 mm
IC-Fassung DIL 40-polig, für IC2
Platine 110406-1, siehe [2]

regenbogenfarbigen LEDs durch Pulsbreitensteuerung (PWM) variiert. Im Mikrocontroller sind zwar PWM-Signalgeneratoren integriert, trotzdem hat sich der Autor entschieden, die PWM-Signale durch Software zu realisieren. Der Grund ist ausschließlich, dass das Platinenlayout einfacher gestaltet werden kann.

Die LEDs werden über die Power-MOSFETs T3, T4 und T5 gesteuert. Der IRL540 ist ein Typ, der bei TTL-Niveau schaltet, er verträgt ohne Kühlung Ströme in der Größenordnung einiger Ampere. Die Kontaktleisten K2...K5 stellen die Verbindungen zu den vier LED-Strips her. BZ1 ist ein akustischer Signalgeber mit eingebautem Tongenerator. Das Anlegen einer Spannung genügt, um einen Alarmton auszulösen. Bedient wird der Ambience Lighting Controller mit den Drehencodern ENC1A/B, ENC2A/B und ENC3A/B.

Die Leitungen RA0 und RA1 des Mikrocontrollers sind die Eingänge des A/D-Wandlers. Über RA0 misst der Mikrocontroller die Akkuspannung, über RA1 erkennt er, ob ein Akkulader angeschlossen ist und ob die Spannung 14 V übersteigt. Die Spannungsteiler R5/R4 und R6/R7 passen die Spannungen an den Messbereich des Mikrocontrollers an, C8 und C9 reduzieren überlagerte Störspannungen.

Das Seriell/USB-Modul (Elektor 110553-91, siehe [1]) an RC6 und RC7 stellt die Verbindung mit einem PC her. Wenn auf dem PC ein Terminalprogramm läuft, kann der Ambience Lighting Controller von dort ferngesteuert werden. Die Kommandos, die gesendet werden müssen, sind in einer von der Projektseite [2] herunterladbaren Anleitung zusammengefasst.

Die Betriebsspannung wird konventionell von einem Spannungsregler 7805 stabilisiert (IC1, C1...C4). Diode D2 schützt den Spannungsregler vor einer falsch gepolten Eingangsspannung. Die Sicherung F1 muss abhängig von der Stromaufnahme der LED-Strips dimensioniert werden. Der Wert 2 AT dürfte meistens passen, besser ist jedoch, die Stromaufnahme (bei voller LED-Leuchtstärke!) zu messen.

Kabelklemme K7 ist der Anschluss für den 12-V-Akku, der Akkulader wird mit Klemme K1 verbunden. Die Akkulader-Spannung muss ungefähr 15 V betragen, die Belastbarkeit darf nicht unter 2 A liegen. Wenn der Akku geladen werden muss, aktiviert der Mikrocontroller über T1 das Relais RE1. Der Relaiskontakt überbrückt R1, so dass der Akku mit vollem Strom geladen wird.

Ohne LED-Strips und Beleuchtung des LC-Displays beträgt der Strombedarf der Schaltung etwa 25 mA, mit Display-Beleuchtung steigt er auf ungefähr 50 mA an.

Platine

Bild 2 gibt das vom Elektor-Labor entworfene Platinenlayout wieder. Die Platine ist für bedrahtete Bauelemente ausgelegt, SMDs kommen nicht vor. Alle Komponenten haben ihren Platz auf der Oberseite. Die Beine der Power-MOSFETs und des Spannungsreglers werden vor der Montage rechtwinklig abgebogen, so dass diese Bauelemente flach auf der Platine liegen. Normalerweise ist eine zusätzliche Kühlung nicht erforderlich.

Die Montage einer Fassung für den (programmierten!) Mikrocontroller ist zweckmäßig. Das Seriell/USB-Modul für die Verbindung mit dem PC kann in zwei 9-polige SIL-Fassungsstege gesetzt werden. Ohne Fassung ist dieses Modul ebenfalls montierbar.

Software

Das Programm für den Mikrocontroller wurde in ANSI-C unter der Umgebung MPLAB geschrieben und mit einem professionellen C-Compiler compiliert. Der Compiler befand sich für 45 Tage in einem voll funktionsfähigen Evaluationsmodus. Die Lite-Version des Compilers genügte nicht, da die Code-Optimierung unzureichend war. Der ausführbare Code überstieg die Speicherkapazität 8 KB des Mikrocontrollers. Der Quell- und Hexcode des Programms kann ebenso wie das Platinenlayout kostenfrei von der Projektseite [2] heruntergeladen werden. Ein programmierter Mikrocontroller ist im Elektor-Shop erhältlich.

Der wichtigste Teil der Software ist die Interrupt Service Routine (ISR). Diese Routine wurde mit der Funktion Stopwatch von MPLAP auf kurze Ausführungsgeschwindigkeit getrimmt. Die ISR besteht aus Segmenten, die im zeitlichen Abstand von 100 µs, 5 ms, 100 ms oder 1 s ausgeführt werden müssen. Die Sprünge zur ISR finden in Intervallen von 100 µs statt. Das Ausführen der Segmente, die zu den anderen genannten Zeiten abzuarbeiten sind, wird von Zählern gesteuert.

Um Speicherplatz zu sparen, wurde das Timing des 1-ms- und 100-ms-Intervalls zunächst mit einem Zähler realisiert. Bei jedem ISR-Aufruf (Intervall 100 µs) wurde der Modulus (Rest) einer Division berechnet. Der Wert 0 als Ergebnis bedeutete, dass 1 ms verstrichen war. Das Debuggen mit Stopwatch ergab jedoch, dass die Berechnung des Modulus viel Zeit kostete. Es war

Listing 1

```
fade_step_red = current_red_value - next_red_value;
fade_step_green = current_green_value - next_green_value;
fade_step_blue = current_blue_value - next_blue_value;
fade_step_red = fade_step_red * 100;
fade_step_green =  fade_step_green * 100;
fade_step_blue = fade_step_blue * 100;
fade_step_red = fade_step_red / fade_time;
fade_step_green =  fade_step_green / fade_time;
fade_step_blue = fade_step_blue / fade_time;
```

Listing 2

```
tmp_red_value = fade_tmr * fade_step_red;
tmp_green_value = fade_tmr * fade_step_green;
tmp_blue_value = fade_tmr * fade_step_blue;
tmp_red_value = tmp_red_value / 100;
tmp_green_value = tmp_green_value / 100;
tmp_blue_value = tmp_blue_value / 100;
red_value = next_red_value + tmp_red_value;
green_value = next_green_value + tmp_green_value;
blue_value = next_blue_value + tmp_blue_value;
```

deshalb sinnvoller, das Timing des 1-ms-Intervalls mit einem eigenen Zähler durchzuführen.
Eine weitgehende Unabhängigkeit der LED-Strip-Leuchtstärke von der Akkuspannung wird über die PWM-Frequenz erreicht. Normalerweise beträgt die PWM-Frequenz 100 Hz, die Spannung beträgt 11 V. Bei höheren Akkuspannungen und unverändertem Duty-Cycle würden die LED-Strips heller leuchten. Das Anpassen des Duty-Cycles an die Akkuspannung macht für jede Farbe laufende Berechnungen des Duty-Cycles notwendig. Wesentlich einfacher lässt sich folgendes Verfahren realisieren: Die PWM-Einschaltzeit bleibt konstant, stattdessen wird bei höheren Akkuspannungen die PWM-Frequenz herabgesetzt. Jetzt muss die Berechnung nur noch einmal durchgeführt werden, das Endergebnis ist jedoch gleich. Das menschliche Auge nimmt das Absenken der PWM-Frequenz von 100 Hz auf 90 Hz nicht wahr.
Ein Programmteil, über den der Programmierer lange nachdachte, war die Realisierung der Farbübergänge. Eigentlich ist die Berechnung ganz einfach: Für jede Farbe wird vom aktuellen PWM-Wert der PWM-Wert des folgenden Schritts subtrahiert, das Ergebnis wird durch die Übergangszeit dividiert. Anschließend wird der PWM-Wert bei jedem Schritt während der Übergangszeit erhöht beziehungsweise herabgesetzt. Die Übergangszeit wird hier durch die Anzahl der 100-ms-Schritte ausgedrückt. Das Ergebnis der Division kann eine Zahl sein, die hinter dem Komma viele Stellen hat. Um mit der Zahl rechnen zu können, muss eine Fließkomma-Variable benutzt werden.
Die Mikrocontroller der PIC16Fxxx-Familie sind 8-bit-Typen, die bei solchen Operationen schnell an ihre Grenzen stoßen. Auch der Compiler tat sich mit den Fließkomma-Variablen schwer. Als Folge traten Timing-Probleme sowie Fehler im compilierten Programm auf. Die Lösung liegt nicht allzu fern: Wesentlich schneller als mit Fließkommazahlen ist das Multiplizieren und Dividieren mit Integern durchführbar, der Speicherbedarf ist deutlich geringer. Deshalb werden die Differenzen der PWM-Werte jeder Farbe mit 100 multipliziert, bevor sie durch die Übergangszeit dividiert werden (**Listing 1**).

Bei jedem Übergangsschritt (Intervall 100 ms) wird der aktuelle PWM-Wert berechnet. Dann wird er durch 100 geteilt, so dass das Ergebnis ein auf zwei Dezimalstellen gerundeter Integer ist, ohne Stellen hinter dem Komma (**Listing 2**).

Der angewendeten Methode ist zu verdanken, dass der 8-bit-Mikrocontroller die LED-Strips ohne Einschränkungen so steuern kann, dass die Farben tatsächlich ineinander fließen.

Praxis

Nach dem Einschalten gibt der Mikrocontroller auf dem LC-Display eine Startmeldung aus, anschließend erscheint das Startmenü. Die Bedienung ist eigentlich selbsterklärend, die Projektseite [2] hält aber auch eine ausführliche englischsprachige Anleitung zum Download bereit.
Mit dem Drehencoder ENC1 scrollt man durch die Menüs. Mit Run Program <x> wird das Lightshow-Programm gewählt (x=1...3), das ablaufen soll. Gestartet wird das Programm durch Drücken von ENTER (Taster von ENC2), die Rückkehr zum Menü ist mit BACK (Taster von ENC3) möglich.
Nachdem der Akkulader angeschlossen ist (15 V, ≥2 A), überwacht und lädt die Funktion Charge Battery den Akku. Das Relais zieht an und überbrückt Widerstand R1, so dass zum Akku der volle Ladestrom fließt. Während der Ladezeit sind die LED-Strips abgeschaltet, um sie vor zu hoher Spannung zu schützen. Wenn am Akku 13,8 V

liegen, fällt das Relais ab. Die LED-Strips werden eingeschaltet, und zum Akku fließt ein niedriger Strom, der im Ruhezustand die Selbstentladung kompensiert.
Mit Battery Charge wird die verbleibende Akkuladung in Stufen von 10 % angezeigt. Maßgebend ist die gemessene Akkuspannung, wobei 11 V dem Wert 0 % und 13,8 V dem Wert 100 % entsprechen.
LED D5 signalisiert den Ladezustand des Akkus. Die LED leuchtet kontinuierlich, solange die Spannung über 13,2 V liegt, sie blinkt im 1-Hz-Rhythmus, wenn die gespeicherte Energie zur Neige geht. Falls die LED blinkt, hängt das Hell-Dunkel-Verhältnis von der Restladung ab. Ist die Akkukapazität so weit erschöpft, dass die LED-Strips abgeschaltet werden müssen, gibt der akustische Alarmgeber einen Warnton ab.
Schalter S2 verhindert das versehentliche Ändern von Einstellungen. Bei geschlossenem Schalter sind die Menüpunkte Edit Program <x> und Edit <color> nicht verfügbar.
Zum Einstellen der Farben wird Edit <color> eingestellt und mit ENTER aktiviert. Danach sind mit den drei Encodern die Farben Rot, Grün und Blau in Stufen von 1 % programmierbar. Mit ENTER wird die neue Einstellung gespeichert, mit BACK kehrt das Menü ohne Speichern zur letzten Auswahl zurück.
Lightshow-Programme lassen sich mit Edit Program <x> gefolgt von ENTER erstellen. Jetzt dient Drehencoder ENC1 zur Auswahl der Farbe, ENC2 bestimmt die Hold-Zeit und mit ENC3 wird der Fade-Wert (Übergangszeit) eingestellt.

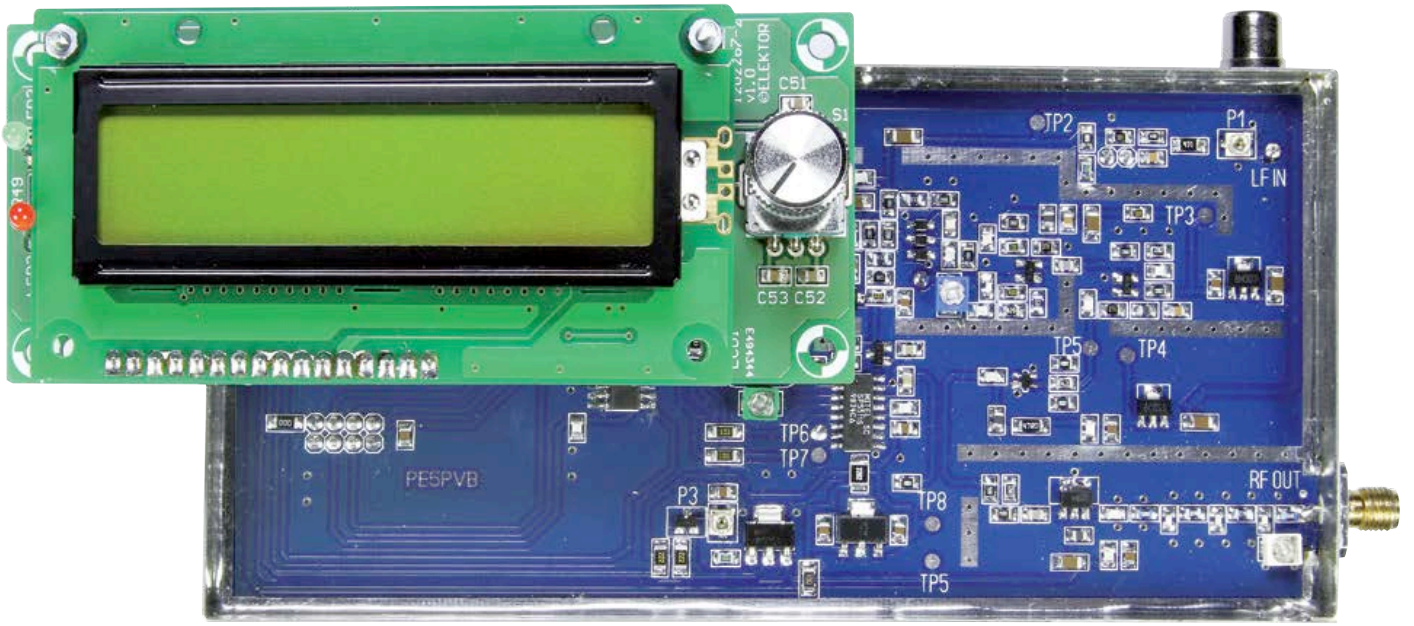
(110406)gd

Weblinks

[1] www.elektor.de/110553
[2] www.elektor.de/110406

FM-Audio-Sender für das 70-cm-Band

Mit 130 mW Ausgangsleistung



Von Sjef Verhoeven
(NL)

Die meisten FM-Sender, die Funkamateure im 70-cm-Band (430…440 MHz) einsetzen, sind nicht zum Übertragen hochqualitativer Audiosignale konstruiert. Diese Lücke soll hier geschlossen werden: Der Beitrag beschreibt den Bau eines breitbandigen FM-Senders, der Audiosignale im Bereich 20 Hz…100 kHz überträgt.

Eigenschaften

- Frequenzbereich: 430...440 MHz, Raster 25 kHz
- Übertragungsbereich: 20 Hz...100 kHz
- Einrastverzögerung der PLL: < 1 s
- Betriebsspannung: 12...15 V
- Stromaufnahme: 250 mA bei HF-Ausgangsleistung 130 mW

Die vielgenutzten „Walkie-Talkies“ sind kleine portable Geräte, die zur drahtlosen Sprachkommunikation über kurze Entfernungen dienen. Die Bandbreite der übertragenen Signale ist auf die Sprachfrequenzen beschränkt, und außerdem ist zwischen zwei Geräten nur das wechselweise Sprechen (Halbduplex) möglich. Diese Form der drahtlosen Sprachkommunikation wird bereits seit Anfang des letzten Jahrhunderts von Funkamateuren in aller Welt angewendet.

Mit dem nötigen Equipment ist auf den Amateurfunkbändern auch das Übertragen breitbandiger Audiosignale in HiFi-Qualität möglich. Den Beweis liefert der hier vorgestellte Sender für das 70-cm-Band, der hochqualitative Audiosignale in der Modulationsart FM überträgt. Die Übertragungsqualität steht dem vom FM-Rundfunk gewohnten Standard nicht nach, der übertragbare Audiobereich liegt zwischen rund 20 Hz und

100 kHz. Aufgrund der hohen Bandbreite ist das Einrichten eines verzerrungsarmen Audio-Chats mit mehreren beteiligten Sendern vergleichsweise einfach. Mit optimierten Antennen, hohen Antennenstandorten und eventueller Nachverstärkung lassen sich Distanzen von einigen 10 km überbrücken. Bei günstigen atmosphärischen Bedingungen können die Entfernungen sogar die Größenordnung 300 km erreichen.

Der Bau und die Inbetriebnahme des Senders ist nur Funkamateuren erlaubt, die eine Zulassung zur Teilnahme am Amateurfunkdienst der Klassen A oder E besitzen. Eine solche Zulassung erhält ein Kandidat, nachdem er seine Befähigung in einer persönlichen Prüfung bei der Bundesnetzagentur (BNetzA) unter Beweis gestellt hat. In der Prüfung geht es unter anderem um Kenntnisse der Elektrotechnik und Funktechnik, der Betriebstechnik von Sendeanlagen und der gesetzlichen Vorschriften. Nach erfolgreichem Abschluss stellt die Bundesnetzagentur dem Bewerber ein so genanntes Amateurfunkzeugnis aus. Über den Weg zum Funkamateur informieren in Deutschland beispielsweise der Deutsche Amateur-Radio-Club (DARC) und der Verband der Funkamateure in Telekommunikation und Post (VFDB).

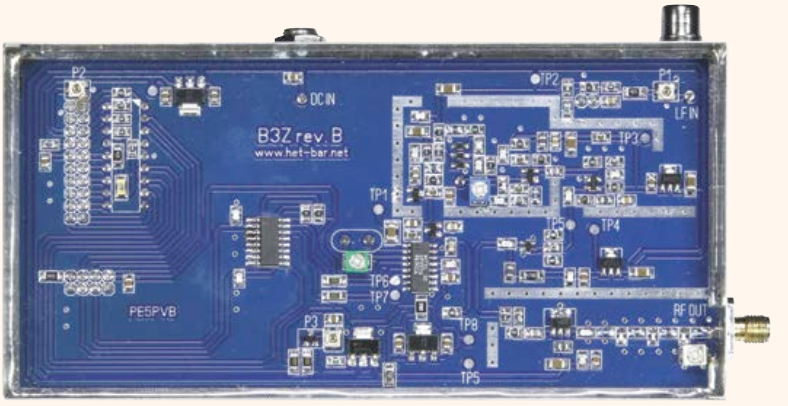
Die von diesem Sender ausgestrahlten Signale können mit Scannern oder anderen Geräten empfangen werden, die für breitbandige Frequenzmodulation (Wide band FM, WFM) ausgelegt sind. Der Empfang ist auch mit einem Frequenzumsetzer möglich, der den Bereich 430...440 MHz beispielsweise auf 90...100 MHz im FM-Rundfunkband konvertiert.

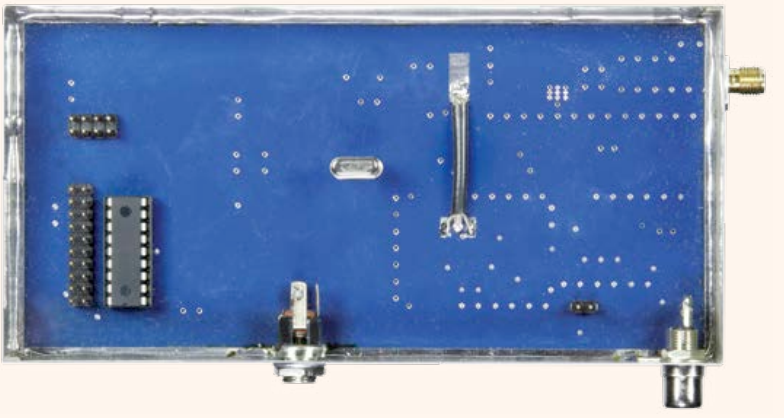
Senderschaltung

Wie Bild 1 zeigt, wird das hochfrequente Signal von einem Colpitts-Oszillator erzeugt, der mit einem Dual-Gate-MOSFET (T2) aufgebaut ist. Die Frequenz des Sender-Ausgangssignals ist gleich der Oszillatorfrequenz. Bei Colpitts-Oszillatoren hängt die Frequenz von einem Parallelschwingkreis ab, der aus einem kapazitiven Spannungsteiler (C24 und C25) und einer Induktivität (L4) besteht. Das kurze Stück Koaxkabel, das hier die Induktivität nachbildet, verhält sich ähnlich wie eine Streifenleitung. Ein flexibler Koaxkabel-Typ, beispielsweise RG174, reduziert die Einflüsse mechanischer Schwingungen auf die Modulation. Im Rückkopplungszweig liegt Widerstand R25, er trägt dazu bei, dass der Oszillator sauber schwingt. Die Abstimmung auf das Frequenzband wird mit Trimmer TR2 vorgenommen. Die Kapazitäten der Varicap-Dioden

Bautipps

- Verwenden Sie ausschließlich keramische Kondensatoren, auch bei Werten über 1 µF.
- Die meisten Widerstände und Kondensatoren haben die Bauform 0805, einige 1206.
- Wenn Sie die Platine anfertigen, müssen Sie für genügend Masse-Durchkontaktierungen sorgen. Dies gilt insbesondere für die Bereiche hochfrequenter Signale.
- In den Induktivitäten dürfen sich KEINE Ferritkerne befinden. Dies betrifft vor allem den Oszillator. Ferritkerne sind als Verursacher von Rauschen bekannt.
- Die aufgebaute Schaltung muss in ein Metallgehäuse eingebaut werden. Das Platinenformat ist so gewählt, dass ein gängiges Standardgehäuse passt.
- Bereiten Sie zuerst das Gehäuse vor, indem Sie die Durchbrüche bohren oder fräsen. Legen Sie die noch unbestückte Platine in





das Gehäuse und löten Sie die Platine beidseitig im vollen Umfang an die Gehäusewand. Erst wenn das getan ist, montieren Sie die Bauelemente.
- Betreiben Sie den Sender an einer sauberen Betriebsspannung. Oft sind den Spannungen von Schaltnetzteilen Störsignale überlagert, die sich im übertragenen Audiosignal als Rauschen oder Pfeifen bemerkbar machen. Ein konventionell aufgebautes Netzteil ist die beste Lösung.

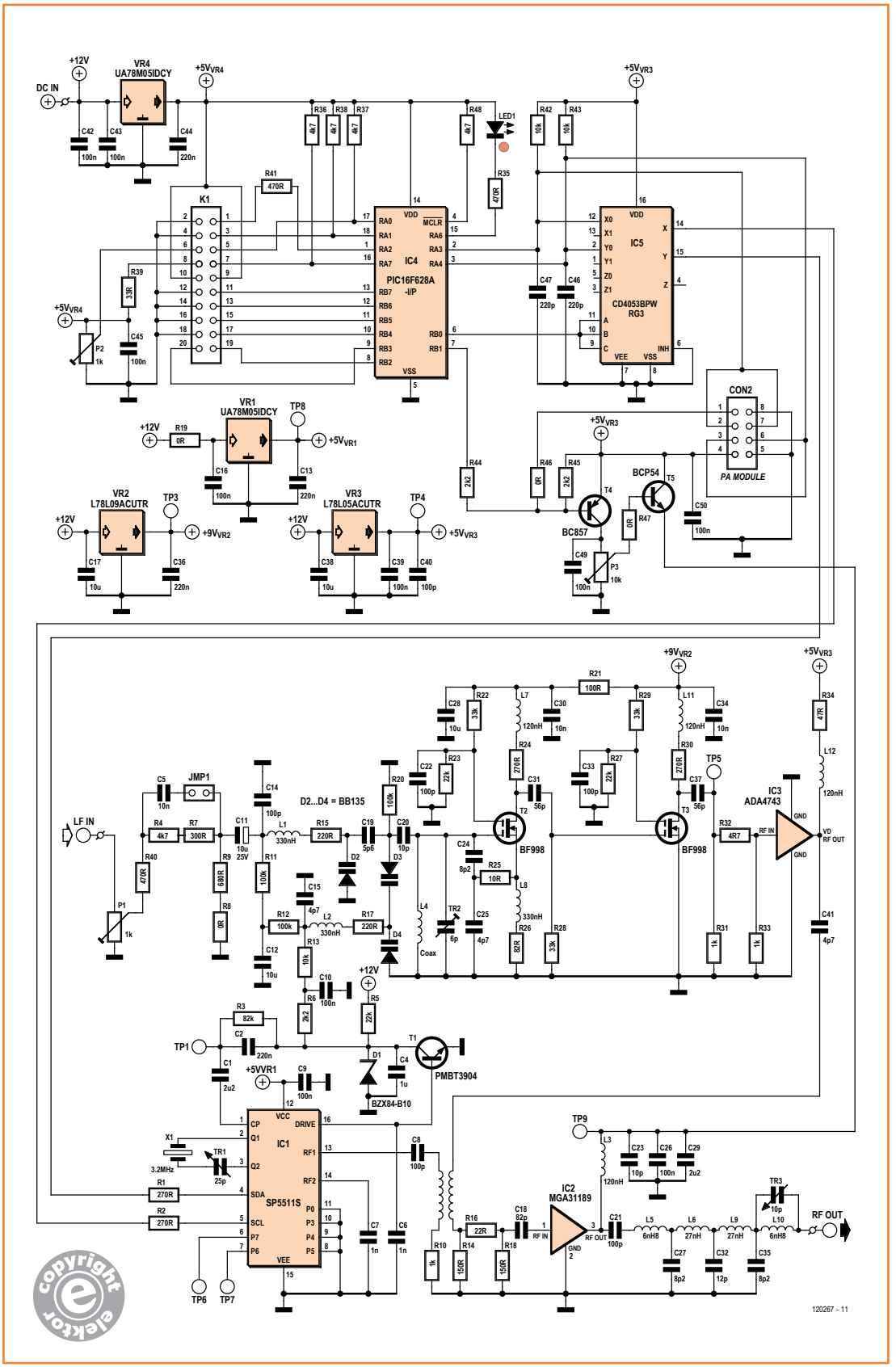


Bild 1.
Schaltung der HF-Stufen und der Steuerung auf der Hauptplatine.

D3 und D4 hängen von den angelegten Spannungen ab. Dem Variationsbereich der Spannungen entspricht ein Abstimmbereich von etwa 30 MHz. Frequenzmodulation bedeutet, dass die Frequenz der Trägerschwingung im Rhythmus des niederfrequenten Audiosignals verändert wird. Die Varicap-Diode D2 bewirkt diese Frequenzänderungen, indem sie die Kapazität im frequenzbestimmenden LC-Kreis beeinflusst. Für das Abstimmen der Trägerfrequenz und für die Modulation sind getrennte Varicap-Dioden vorhanden, was die Linearität der Modulation steigert. Da der Modulationsgrad bei niedrigen und hohen Trägerfrequenzen möglichst übereinstimmen soll, liegt Varicap-Diode D2 über den hochohmigen Spannungsteiler R11/R12 an einer Vorspannung. Kondensator C12 trägt zur Stabilität bei, und Induktivität L1 bewirkt, dass der Modulationszweig kapazitiv nur wenig belastet wird. Eine hohe kapazitive Last kann zum Abreißen der Oszillatorschwingungen führen. Kondensator C14 entkoppelt den Modulationseingang von hochfrequenten Signalen, gleichzeitig verhindert er, dass die nachfolgenden Komponenten das LC-Verhältnis des Oszillatorkreises verändern. Für die Preemphasis sind C5, R4, R7, R8 und R9 zuständig. Wenn die Preemphasis das zu übertragende Signal unerwünscht beeinflusst, kann sie durch Entfernen von Jumper JMP1 außer Funktion gesetzt werden. Die Dämpfung sinkt, wenn der Wert von R8 höher als angegeben gewählt wird. Mit den Werten in der Schaltung beträgt die Preemphasis annähernd 50 µs.

Die Pufferstufe hinter dem Oszillator, die ebenfalls mit einem Dual-Gate-MOSFET (T3) bestückt ist, sorgt für eine annähernd konstante Last. Für ein stabiles hochfrequentes Signal ist eine solche Pufferstufe unerlässlich. Um das hochfrequente Signal (etwa 1 mW) an die nachfolgende Stufe anzupassen, wird es zunächst abgeschwächt.

In den letzten Jahren kamen zahlreiche so genannte MMICs auf den Markt, die Abkürzung steht für Monolithic Microwave Integrated Circuit. Ein MMIC ist ein monolithisch integrierter Baustein, der zum Verarbeiten von Signalen ultrahoher Frequenzen entwickelt wurde. Eine höchst willkommene Eigenschaft der MMICs ist das Fehlen abgestimmter Resonanzkreise, was die Schaltungstechnik stark vereinfacht. Außerdem wird Platz auf der Platine eingespart, und ferner werden die Kosten des Endprodukts nicht unerheblich gesenkt. Der hier beschriebene breitbandige FM-Sender für das 70-cm-Band arbeitet mit MMICs des Herstellers Avago Technologies. Die verwendeten Typen von Avago passen nicht nur wie maßgeschneidert, darüber hinaus sind sie im Handel leicht beschaffbar. Das erste MMIC in der Schaltung, ein ADA4743 (IC3), verstärkt das hochfrequente Signal auf etwa 40 mW. Nach dem Abschwächen wird dieses Signal einem MGA31189 (IC2) zugeführt, ein MMIC, das Ausgangsleistungen bis 250 mW liefern kann. Da die Betriebsspannung mit T4 und T5 auf 4,2 V begrenzt wird, beträgt die Ausgangsleistung nur etwa 120...130 mW. Eine Leistung dieser Größenordnung ist gut geeignet, eine eventuell nachzuschaltende Endstufe zu steuern.

Zum Abstimmen und Konstanthalten der Trägerfrequenz macht die Schaltung vom Prinzip der PLL (Phase Locked Loop) Gebrauch. Die Frequenz des zu sendenden Signals wird von einem programmierbaren Frequenzteiler herabgeteilt und mit der Frequenz eines Referenzsignals verglichen. Wenn eine Abweichung erkannt wird, steuert eine Korrekturspannung die Frequenz so lange nach, bis das frequenzgeteilte Signal und das Referenzsignal in Phase sind. In der Schaltung gelangt die Korrekturspannung nach Durchlaufen eines Loop-Filters zu den Varicap-Dioden D3 und D4. Der Loop-Filter ist hier unverzichtbar, er ist so ausgelegt, dass die PLL mit einer dosierten Verzögerung reagiert. Das Verfahren der Frequenzmodulation bedingt, dass die Trägerfrequenz im Rhythmus des zu übertragenden NF-Signals (Audio) variiert. Würde die PLL die Frequenz zu schnell nachsteuern, könnte das

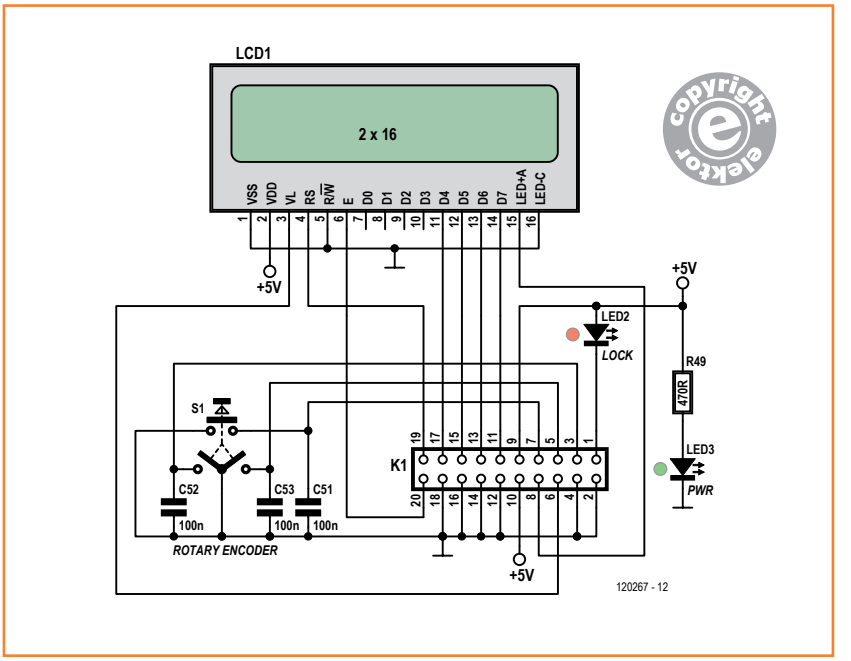


Bild 2.
Zum Bedienteil gehören ein Drehencoder und ein zweizeiliges LC-Display.

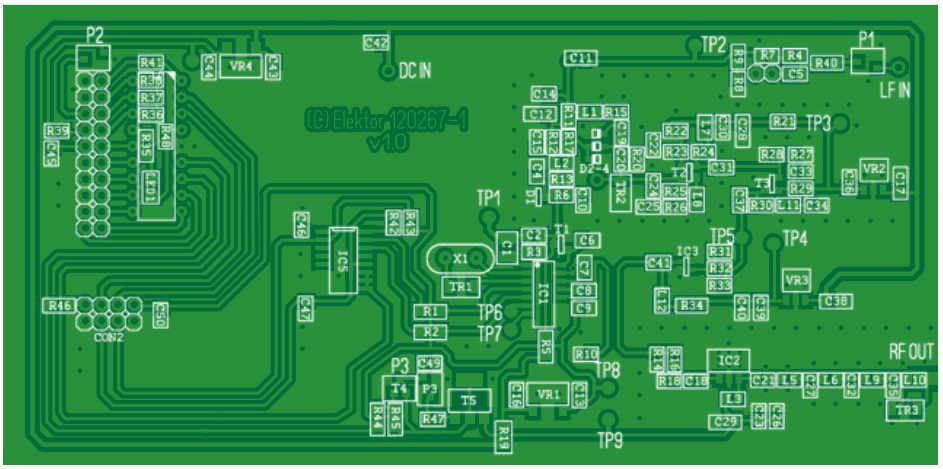


Bild 3.
Auf der Hauptplatine werden beidseitig Komponenten montiert.

NF-Signal nur stark verzerrt übertragen werden. Worauf es hier ankommt, ist ein geschickter Kompromiss zwischen der Bandbreite (etwa 180 kHz) und der Signalstabilität.

Die PLL arbeitet mit dem preiswerten und leicht beschaffbaren Baustein SP5511 (IC1), eine andere Typenbezeichnung lautet TSA5511. Normalerweise wird dieser Baustein in TV-Tunern für den Satellitenempfang eingesetzt, er ist in vielen Geräten der alten Generation zu finden. Mit einem 3,2-MHz-Quarz beträgt die Frequenzschrittweite 50 kHz, für den 70-cm-FM-Sender ist jedoch die Schrittweite 25 kHz gefordert. Das Problem lässt sich leicht lösen, indem die PLL mit der 2. Harmonischen des Sendersignals arbeitet. Dieses Signal wird zwischen IC3 und IC2 induktiv ausgekoppelt. Der PLL-Baustein wird über I2C von einem Mikrocontroller des Typs PIC 16F628A (IC4) gesteuert. Ein Handicap dieser Konfiguration besteht darin, dass die Signale des I2C-Busses auf das zu übertragende NF-Signal übersprechen. Im gesendeten Audiosignal sind Störgeräusche abhängig vom Datenfluss auf dem I2C-Bus hörbar. Der Autor ging noch einen Schritt weiter, indem er über den I2C-Bus auch eine Endstufe und andere Hardware steuerte. Die nicht für die PLL bestimmten I2C-Bus-Signale waren im NF-Signal zusätzlich zu hören. In der Schaltung des 70-cm-FM-Senders wurde das Problem durch Einfügen eines Mehrfach-Signalschalters HEF4053 (IC5) zwischen dem I2C-Bus und dem PLL-Baustein gelöst. Immer wenn die I2C-Daten nicht an die PLL adressiert sind, wird die I2C-Verbindung zur PLL unterbrochen. Dies ist eine ebenso unkomplizierte wie effiziente Methode.

Für die Bedienung des Senders ist ein Drehencoder mit Drucktaster vorhanden, er befindet sich zusammen mit zwei LEDs und dem Display auf einer eigenen Platine. Die Schaltung des Bedienteils, das mit K1 auf der Hauptplatine zu verbinden ist, geht aus Bild 2 hervor. Mit dem Drehencoder wird die Frequenz eingestellt, beim Drücken des Tasters wird sie in den Speicher übernommen, die gespeicherten Daten steuern die PLL. Wenn die PLL rückmeldet, dass die Phasendifferenz zwischen Sendersignal und Referenzsignal ausgeregelt ist, gelangt diese Rückmeldung zum Mikrocontroller. Als Zeichen dafür leuchtet LED2 auf (PLL Lock), gleichzeitig erhält das zweite MMIC Betriebsspannung. So wird sichergestellt, dass das hochfrequente Ausgangssignal nur zur Antenne gelangt, wenn die Frequenz tatsächlich eingestellt ist. Die Betriebsspannung des zweiten MMIC ist mit P3 einstellbar, so dass die Sendeleistung stufenlos bis etwa 1 mW reduziert werden kann. Eine nachgeschaltete empfindliche Endstufe wäre möglicherweise mit der vollen Leistung übersteuert. Über die Kontaktleiste CON2 kann der Mikrocontroller zusätzliche Systemkomponenten steuern, beispielsweise eine externe Endstufe.

Der Autor startete die Entwicklung des 70-cm-FM-Senders mit der „PIC Simulator IDE”. Dies ist eine relativ schlanke Programmierumgebung, die eine eigene Basic-Variante verwendet. Dort ist ein Simulator integriert, mit dem die Software vollständig simuliert werden kann, bevor sie in den Mikrocontroller geladen wird. Nachteilig wirken sich jedoch die begrenzten mathematischen Fähigkeiten aus. Damit das Programm

überschaubar bleibt, sind zwei Zähler in Funktion. Der erste Zähler ist für die Frequenzanzeige zuständig, während der zweite Zähler den Faktor der Frequenzteilung enthält. Immer wenn die Frequenz um 25 kHz verändert wird, ändert sich der Zählerstand um den Wert 1. An den Bandenden 430 MHz und 440 MHz bleibt der Zählerstand auf vorgegebenen Werten stehen.
Alle wichtigen Dokumente und Informationen zu diesem Projekt (Platinenlayouts, Stücklisten, Quell- und Hexcode für den Mikrocontroller) stehen auf der Elektor-Projektseite [1] zum Download bereit.

Bedienung

Das einzige Bedienelement des Senders ist ein Drehencoder mit integriertem Drucktaster. Wird der Drehknopf betätigt, erscheint im Display der Schriftzug „TUNE". Nachdem die Frequenz im Display mit dem Drehencoder eingestellt ist, muss der Drucktaster betätigt werden. Die Frequenz wird gespeichert und der Sender wird auf die gewählte Frequenz abgestimmt. War der Vorgang erfolgreich, leuchtet die LED „LOCK" auf, der Sender schaltet die Endstufe ein. Falls die PLL den Oszillator nicht auf die gewählte Frequenz abstimmen kann, bleibt die LED dunkel, im Display erscheint die Fehlermeldung „PLLERR". Die Fehlermeldung „I2CERR" wird ausgegeben, falls der Mikrocontroller keine PLL-Daten über den I2C-Bus erhält. Bei beiden Fehlern wird die Prozedur der Frequenzeinstellung so lange wiederholt, bis die PLL einrastet.

Aufbau

Für Schaltungen, in denen hochfrequente Signale verarbeitet werden, sind kurze Signalwege für den Erfolg mitentscheidend. Außerdem müssen die Masseleitungen und Masseflächen auf gemeinsamem, möglichst genau gleichem Potential liegen. Das Platinenlayout des Autors, verkleinert wiedergegeben in Bild 3, sieht deshalb diverse Durchkontaktierungen in den Masseflächen vor. Wenn die Platine in eigener Regie angefertigt wird, dürfen die Anzahl und Lage der Durchkontaktierungen nicht wesentlich modifiziert werden. Die Platine wurde für den Einbau in ein gängiges Metallgehäuse mit der Grundfläche 74 · 148 mm entworfen. Im Gehäuse wird der Platinenrand sowohl auf der Oberseite als auch auf der Unterseite in vollem Umfang an die Gehäusewand gelötet. Der Einbau in ein Metallgehäuse und die rundum verlaufenden Lötverbindungen sind für eine gute Masseverbindung entscheidend. Ein improvisiertes, aus Platinenmaterial zusammen gestückeltes Gehäuse ist für den 70-cm-FM-Sender nicht geeignet.

Bild 4.
Das kurze Stück Koaxkabel auf der Platinenunterseite ist die Induktivität des Oszillator-Schwingkreises.

Der Autor empfiehlt für den Aufbau folgende Vorgehensweise: Zuerst wird die noch leere Platine wie beschrieben im Gehäuse eingebaut. Dann werden die Buchsen für den HF-Ausgang, den Audioeingang und die Betriebsspannung montiert. Für den HF-Ausgang, an den die Antenne angeschlossen wird, ist eine SMA-Buchse die richtige Wahl. Eine flache Ausführung kann unmittelbar an das Gehäuse gelötet werden, Befestigungsschrauben sind nicht erforderlich. Die Audio-Eingangsbuchse und der Anschluss der Betriebsspannung sind beliebig, für das Audiosignal ist eine Cinch-Buchse eine zweckmäßige Lösung.
Alle SMDs haben ihren Platz auf der Platinenoberseite. Auf der Unterseite befinden sich drei Pfostenverbinder, das als Induktivität wirkende, flexible Stück Koaxkabel sowie der Mikrocontroller. Das Foto in Bild 4 zeigt das 4 cm lange, montierte Koaxkabel aus der Nähe. Auf einer Kabelseite werden Innenleiter und Mantel gemeinsam an die dafür vorgesehene Kupferfläche gelötet. Auf der anderen Seite wird der Mantel zweigeteilt angeschlossen. Der Anschluss des Kabelinnenleiters liegt, wie das Foto zeigt, zwischen den Mantellötpunkten.
Montieren Sie zuerst die zum Oszillator gehörenden Bauelemente. Prüfen Sie mit einem Frequenzzähler oder Spektrumanalyser, ob der Oszillator im Bereich 400...500 MHz schwingt. Die nächsten Etappen des Schaltungsaufbaus sind die Pufferstufe und die MMICs. Legen Sie zwischenzeitlich Betriebsspannung an den zweiten MMIC, so dass er

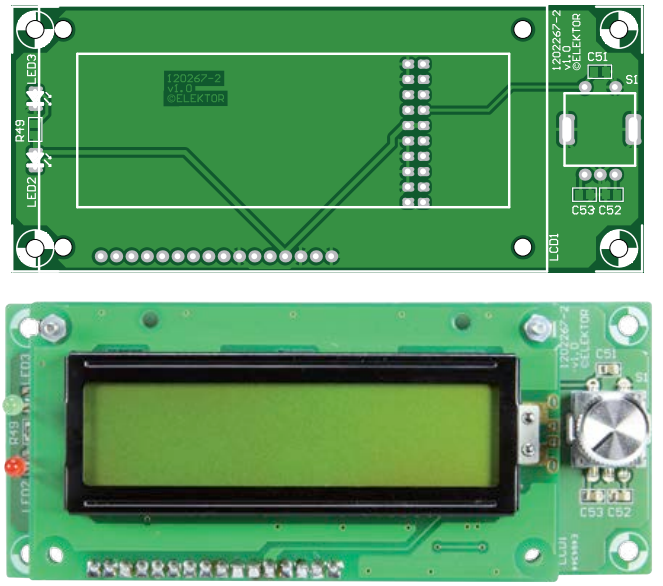


Bild 5.
Beim Bedienteil wird die 20-polige Stiftleiste für die Verbindung mit der Hauptplatine auf der Rückseite montiert.

in Funktion ist. Kontrollieren Sie das HF-Ausgangssignal, beispielsweise mit einem HF-Millivoltmeter. Wenn die HF-Stufen arbeiten, montieren Sie die Bauelemente der PLL und die Fassung für den Mikrocontroller. In die Fassung setzen Sie den Mikrocontroller erst ein, nachdem Sie ihn programmiert haben.

Die Bedien- und Anzeigeelemente sind auf der separaten Platine untergebracht, die Bild 5 zeigt. Der Aufbau ist unkritisch, die Bauelemente werden wie im Bestückungsaufdruck angegeben montiert. Für das LC-Display kann ein 16-poliger Steckverbinder vorgesehen werden. Die zweireihige Kontaktleiste K1 befindet sich auf der Platinenrückseite. Dort wird über eine Buchsensteckleiste ein 20-poliges Flachkabel angeschlossen, das die Verbindung mit Kontaktleiste K1 auf der Hauptplatine herstellt.

Einstellungen

Legen Sie die Betriebsspannung an den 70-cm-FM-Sender einschließlich Bedienteil und optimieren Sie mit P2 den Kontrast des LC-Displays.

Stellen Sie die Sendefrequenz grob auf 435 MHz ein, indem Sie Trimmer TR2 mit einem nichtmetallischen Werkzeug verdrehen. Messen Sie die Spannung an TP1 und drehen Sie TR2 so, dass die Spannung 6 V beträgt. Der genaue Abgleich auf 435 MHz ist nur mit einem Frequenzzähler oder Spektrumanalyser möglich. Nehmen Sie diese Feineinstellung mit Trimmer TR1 vor.

Nachdem Sie mit P3 die maximale Sendeleistung eingestellt haben, können Sie mit einem Spektrumanalyser den Notch-Filter abgleichen. Messen Sie die 2. Harmonische auf 870 MHz und stellen Sie mit Trimmer TR3 das Minimum ein. Wenn ein Spektrumanalyser nicht verfügbar ist, können Sie TR3 mit einem HF-Millivoltmeter auf maximale HF-Leistung einstellen. Beachten Sie, dass mit dieser Methode keine optimale Filtereinstellung möglich ist. Falls Sie eine HF-Endstufe nachschalten, sollten Sie dort die Unterdrückung der 2. Harmonischen optimieren.

Prüfen Sie, ob der 70-cm-FM-Sender auch an den Bandgrenzen 430 MHz und 440 MHz arbeitet. Setzen Sie einen Jumper auf JMP1 (Preemphasis) und schließen Sie eine Audioquelle, beispielsweise einen CD-Spieler an. Stellen Sie mit P1 die Lautstärke so ein, dass noch keine Verzerrungen auftreten.

Damit ist der 70-cm-FM-Sender einsatzbereit. Unsere Erfahrungen lassen vermuten, dass nicht jeder Funkamateur fit im Umgang mit SMD-Bauelementen ist. Der Autor hat sich bereit erklärt, bei genügendem Interesse eine begrenzte Stückzahl gegen ein angemessenes Entgelt aufzubauen und abzugleichen. Mit dem Autor können Sie über seine Email-Adresse [2] Kontakt aufnehmen.

(120267)gd

Weblinks

[1] www.elektor.de/120267

[2] pe5pvb@het-bar.net

Einstellelemente

P1	Modulationstiefe (Lautstärke)
P2	Kontrast LC-Display
P3	Ausgangsleistung
TR1	Feinabstimmung Referenz-Oszillator
TR2	Grobabstimmung Sender-Oszillator
TR3	Notchfilter für 2. Harmonische

Messpunkte

TP1	Regelspannung PLL
TP2	Modulation hinter Preemphasis
TP3	Betriebsspannung Oszillator (9 V)
TP4	Betriebsspannung Steuerstufen (5 V)
TP5	Betriebsspannung Endstufe (0...4,2 V)
TP6	Frequenz nach Teilung durch PLL
TP7	Frequenz der PLL-Referenz

8x Relais und vieles mehr

Erweiterungsmodule für Linux- und andere µC-Boards

Von
Benedikt Sauter [1]
und **Jens Nickel**

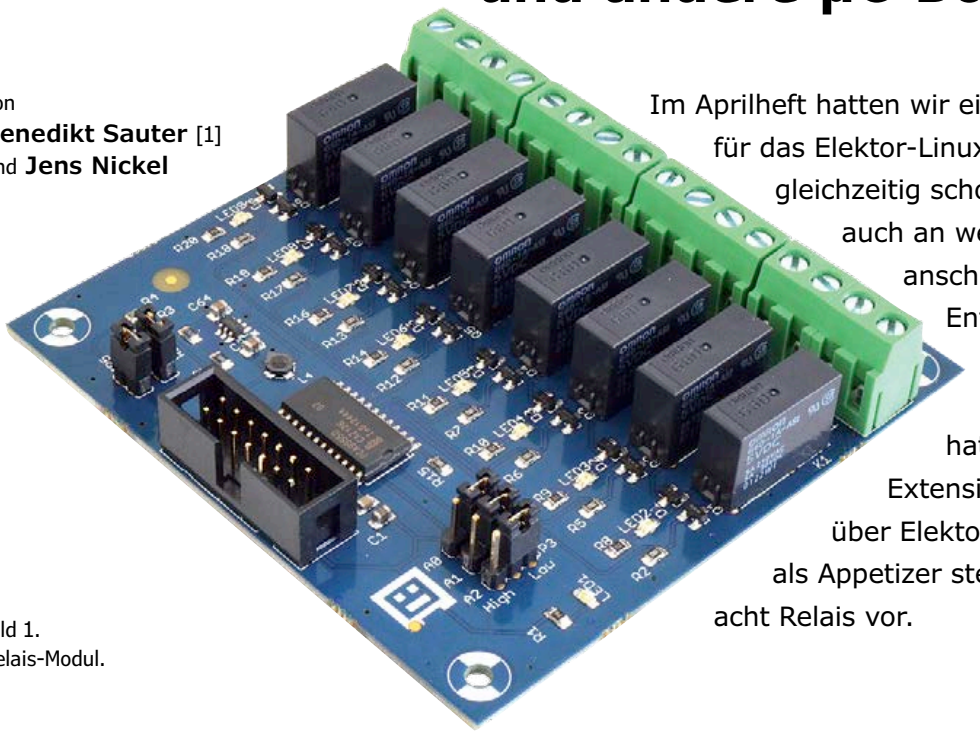


Bild 1.
Relais-Modul.

Im Aprilheft hatten wir eine Erweiterungsplatine für das Elektor-Linux-Board vorgestellt und gleichzeitig schon erwähnt, dass sich diese auch an weitere Controllerboards anschließen lässt. Die Entwicklerschmiede von Embedded Projects war inzwischen nicht untätig und hat eine Reihe von weiteren Extension-Boards konzipiert, die über Elektor erhältlich sind. Sozusagen als Appetizer stellen wir hier eine Karte mit acht Relais vor.

Das Relais-Modul (**Bild 1**) wird wie auch das im April vorgestellte „Linux Extension Board“ über den 14-poligen *Gnublin Connector* angesteuert. Als *Embedded Extension Connector* findet sich dieser auch auf dem Xmega-Webserver-Board von Elektor (siehe nächste Ausgabe), weitere damit ausgestattete Controllerboards sind in Planung. Somit eignet sich die Erweiterungskarte auch für Einsteiger, denen Linux (noch) zu kompliziert ist; genauso aber auch für Power-User, die ihre Programme lieber „bare metal“ (also ohne Zuhilfenahme eines Betriebssystems) entwickeln möchten.

Die Relais-Platine stammt wie das Elektor-Linux-Board aus dem Hause „Embedded Projects“ um Benedikt Sauter. Sie gehört zu einer Reihe von Erweiterungsboards (siehe Kasten), die alle an den erwähnten Stecker passen, denn dieser bringt Pins für SPI, I2C, PWM, analoge Eingänge und digitale Ein-/Ausgänge mit. Im Elektor-Labor wird dagegen an Modulen gearbeitet, die über einen 10-poligen Erweiterungsstecker (*Embedded Communication Connector*) für UART/TTL miteinander Kontakt aufnehmen (siehe Kasten). Es entsteht so also ein kleiner Zoo von Controllerboards und Erweiterungsplatinen, die flexibel kombiniert werden können – Mikrocontrollerfans dürfen sich auf ein interessantes zweites Halbjahr freuen!

Relais-Karte

Das Schaltbild der Relais-Karte sieht man in **Bild 2**. Wie auch beim Linux-Extension-Board erweitert ein über I2C angesprochener Portexpander PCA9555 (IC1) die Zahl der zur Verfügung stehenden Digital-Ausgänge auf 16, von denen hier acht genutzt werden. Die Adresse des I2C-Bausteins kann man über die Jumper K9 bis K11 einstellen. Über K13 und K14 lassen sich Pullup-Widerstände für den I2C-Bus zuschalten.

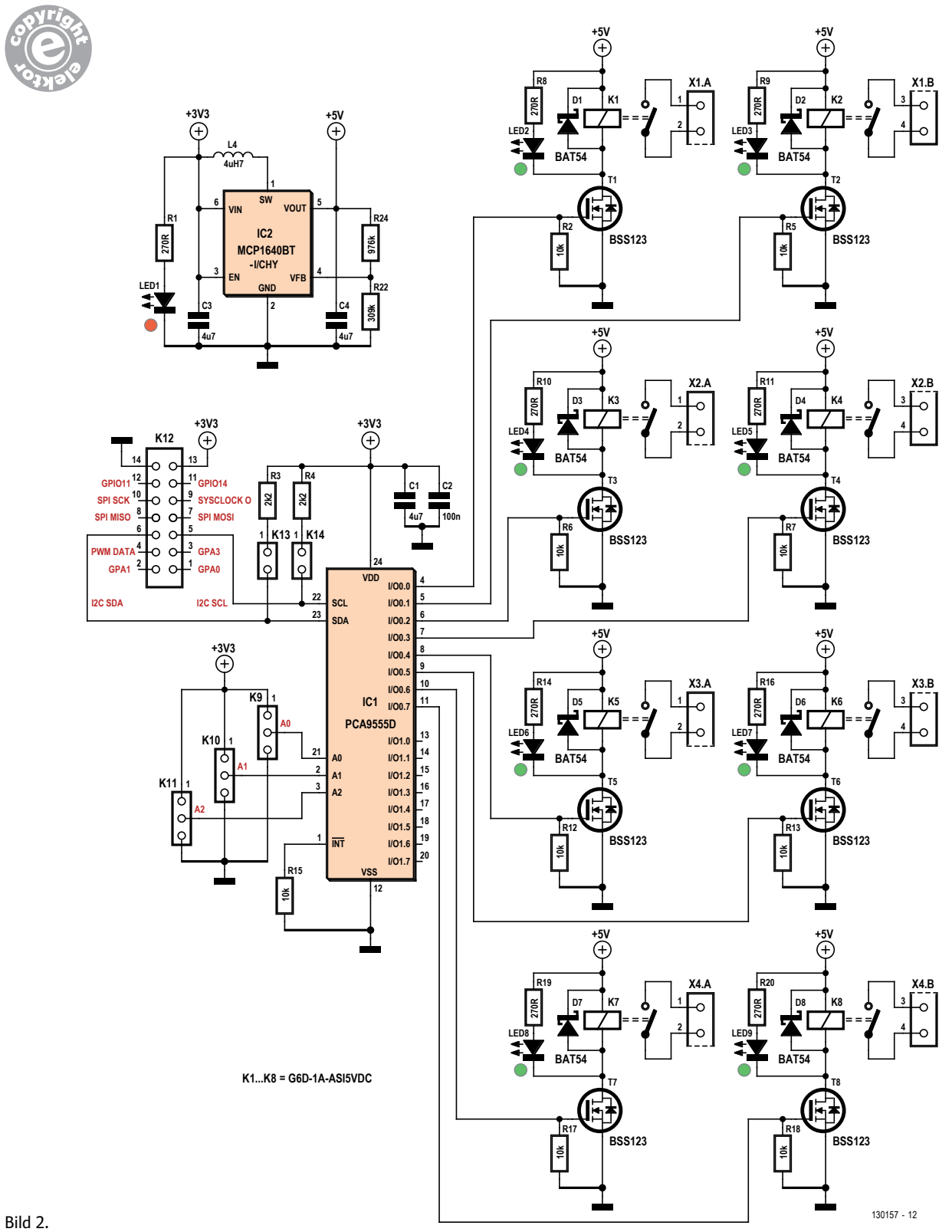


Bild 2.
Schaltplan des Relais-Moduls.

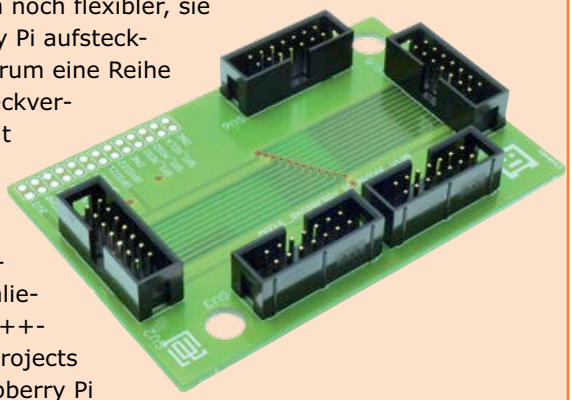
Erweiterungsmodule für den Gnublin/ Embedded Extension Connector (Auswahl)

8x Relais (130212-91)
4x20-Text-Display (130212-92)
Schrittmotortreiber (130212-93)
IO-Expander (130212-94)
Temperatursensor (130212-95)
Verteilerplatine „Bridge Module“ (130212-71)
Raspberry-Pi-Adapter „GnuPi“ (130212-72)

Diese und weitere Boards sind über Elektor erhältlich. Die SMD-Bauteile sind jeweils bereits bestückt, die bedrahteten Bauteile liegen zur Selbstmontage im Kit bei [2].

Raspberry-Pi-Adapter

Die Raspberry-Pi-Adapterplatine „GnuPI“ macht das System aus Erweiterungsplatinen noch flexibler, sie ist auf den Raspberry Pi aufsteckbar und bietet wiederum eine Reihe von Gnublin/EEC-Steckverbindern an [2]. Somit lassen sich alle gezeigten Erweiterungsboards auch an die trendige Computerplattform anschließen. Clever: Die C/C++-API von Embedded Projects ist auch für den Raspberry Pi nutzbar. Zum Umstellen einer Applikation vom Gnublin/Elektor-Linux-Board auf den Raspberry Pi genügt es, eine einzige Codezeile zu verändern:



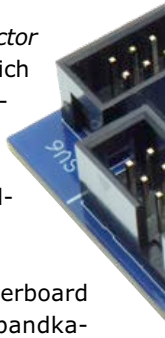
```
#define BOARD_GNUBLIN  →  #define BOARD_RASPBERRYPI
```

Erweiterungsmodule für den Embedded Communication Connector

RS485-Interface (in Entwicklung)
RS232-Interface (geplant)
433-MHz-Funkmodul (in Entwicklung)
Bluetooth per BTM-222 (geplant)
WLAN per WizFi220 (geplant)
USB per BOB (geplant)

Mehr darüber auf der Elektor .Labs Website [8].

Die Digitalausgänge IO 0.0 bis IO 0.7 des Portexpanders steuern jeweils einen FET an, der wiederum ein Relais treibt. Je eine LED dient als Statusanzeige.



Der *Gnublin/Embedded Extension Connector* verfügt über einen 3,3-V-Pin, mit der sich die Erweiterungskarten von der Controllerplatine aus versorgen lassen. Um die 5-V-Spulenspannung für die Relais zu erzeugen, ist deshalb ein Step-Up-Wandler (IC2) verbaut.

Zur Verbindung zwischen dem Controllerboard und der Relais-Platine dient ein Flachbandkabel. Die Entwicklerschmiede von Embedded Projects hat auch bereits an die Möglichkeit gedacht, gleichzeitig mehrere Erweiterungsboards anschließen zu können; auch die Verteilerplatine (**Bild 3**) ist bei Elektor erhältlich [2].

C/C++-API

Wie man die Ausgänge des Portexpander-ICs unter Linux schaltet, haben wir bereits in [3] und [4] gezeigt. Doch es gibt nun eine noch einfachere Möglichkeit. Benedikt Sauter und seine Mitstreiter haben eine komplette C/C++-API geschrieben, mit der sich die Erweiterungskarten einfach ansteuern lassen. Die Funktionen kann man in eigenen Programmen nutzen, es wird aber auch eine Reihe von kleinen Kommandozeilentools angeboten. Mehr darüber in der nächsten Ausgabe, in der wir auch die anderen Extension-Boards vorstellen.

Einen ersten Vorgeschmack auf die C/C++-API geben die Listings. In Listing 1 ist gezeigt, wie man einfach auf die digitalen Ein- und Ausgänge des Elektor-Linux-Boards zugreifen kann. Listing 2 demonstriert, wie man Werte über den analogen Eingang einliest. Und in Listing 3 ist gezeigt, wie man die Relaiskarte verwendet.

Die neue API [5] macht auch Einsteigern den Weg in die Embedded-Linux-Welt einfach; sie kommt ohne komplexere C-Features wie zum Beispiel Pointer aus. Bei den Funktionsnamen haben sich die Entwickler teilweise an die entsprechenden Arduino-Funktionen angelehnt. Neugierige können natürlich gerne einen Blick in den Quelltext werfen [6].

Debian für das Elektor-Linux-Board

Das Gnublin-Linux-System wurde nicht nur um

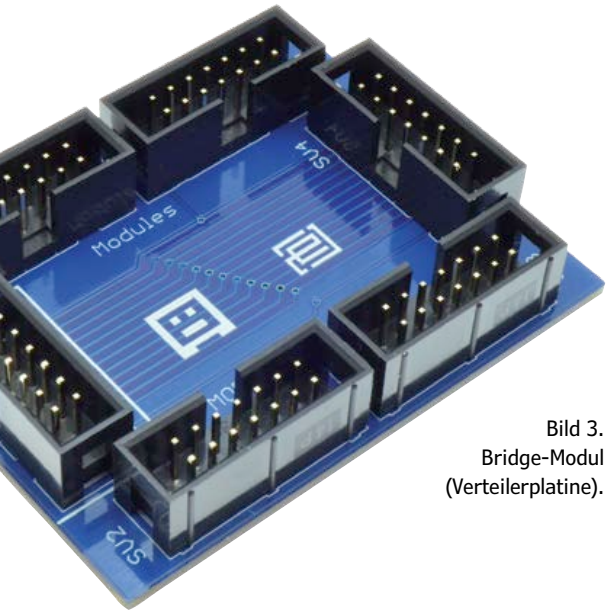


Bild 3.
Bridge-Modul
(Verteilerplatine).

Listing 1: Digitalen Ausgang auf dem Elektor-Linux-Board ansteuern.

```
#define BOARD_GNUBLIN
#include “gnublin.h”

int main()
{
   gnublin_gpio gpio;

   gpio.pinMode(3,OUTPUT);

   while(1){
     gpio.digitalWrite(3,HIGH);
     sleep(2);
     gpio.digitalWrite(3,LOW);
     sleep(2);
   }
}
```

neue Hardware erweitert, auch auf Seiten der Software gibt es ein Update. Wer mag, kann das Elektor-Linux-Board jetzt auch mit einem Debian-System ausstatten (anstelle des ELDK-Dateisystems). Unabhängig davon, ob es sich um die 8-MB- oder 32-MB-Version des Boards handelt, lässt sich Debian ganz einfach über die SD-Karte installieren; eine Anleitung findet man im Internet [7].

(130157)

Weblinks

[1] sauter@embedded-projects.net
[2] www.elektor.de/gnublin
[3] www.elektor.de/120596
[4] www.elektor.de/120518
[5] http://wiki.gnublin.org/index.php/API
[6] https://github.com/embeddedprojects/gnublin-api
[7] http://wiki.gnublin.org/index.php/GNUBLIN-Elektor
[8] www.elektor-labs.com/ECC

Listing 2: Wert am analogen Eingang einlesen.

```
#define BOARD_GNUBLIN
#include “gnublin.h”

int main()
{
   gnublin_adc ad;

   while(1){
        printf(“AD value %i \n”,ad.getValue(1));
   }
}
```

Listing 3: Ansteuerung der Relaiskarte.

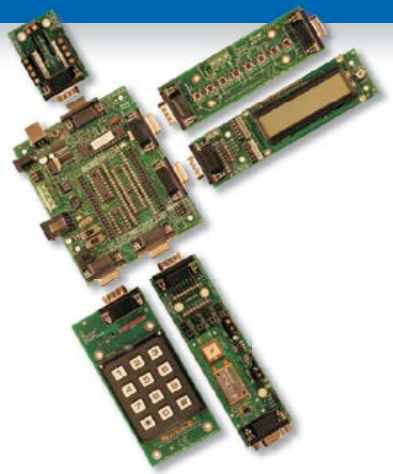
```
#define BOARD_GNUBLIN
#include “gnublin.h”

int main() {
  gnublin_module_relay relay;

  relay.setAddress(0x24);
  relay.switchPin(4, ON);
  sleep(2);
  relay.switchPin(4, OFF);
}
```

Entwickeln und Lernen

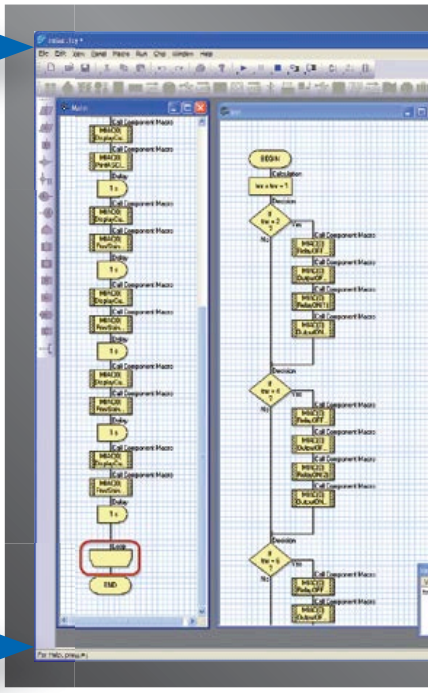
FLOWCODE5

Flowcode 5 ist eine der weltweit besten grafischen Programmiersprachen für Mikrocontroller (PIC, AVR, ARM und dsPIC/PIC24).
Der große Vorteil von Flowcode ist, dass man mit nur wenig (oder gar keiner) Programmiererfahrung in der Lage ist, komplexe elektronische Systeme in Minutenschnelle zu erstellen.

… für Elektronik



E-blocks sind kleine Schaltungen auf Platinen, die für sogenannte Embedded-Systeme typische Elektronik enthalten. Es gibt mittlerweile mehr als 50 unterschiedliche Platinen. Die Module reichen von einfachen LED-Boards bis zu komplexeren Einheiten wie Programmern, Bluetooth oder TCP/IP.
E-blocks können einfach zusammengesteckt werden, um damit eine große Bandbreite an Systemen zu Lernzwecken oder für die Ausbildung im Fach Elektronik zu realisieren. Außerdem ist Rapid Prototyping komplexer elektronischer Systeme möglich. Das Angebot wird ergänzt durch Sensoren, Software, Anwendungsinfos und Curricula.



… für Industrie-Steuerungen

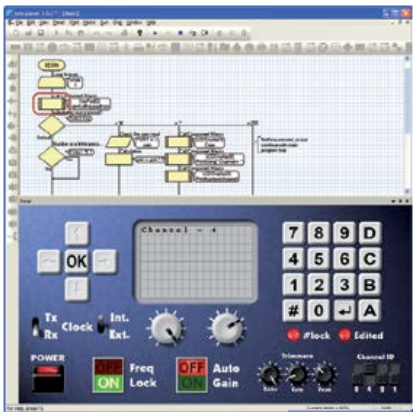
Ein MIAC (**M**atrix **I**ndustrial **A**utomotive **C**ontroller) ist eine Steuerungseinheit für den industriellen Bereich, der die Steuerung einer breiten Palette von elektronischen Systemen im Bereich Sensorik, Überwachung und Automotive erlaubt. Intern arbeitet ein MIAC mit leistungsfähigen Mikrocontrollern der PIC18-Serie und verfügt über USB. Das Modul kann mit Flowcode, C oder Assembler programmiert werden.
Flowcode ist zudem mit dem Industriestandard CAN-Bus ausgestattet, über den mehrere MIACs vernetzt werden können.

FlowKit

Das FlowKit-Modul ermöglicht In-Circuit-Debugging für Flowcode-Anwendungen in PIC- und AVR-Projekten:

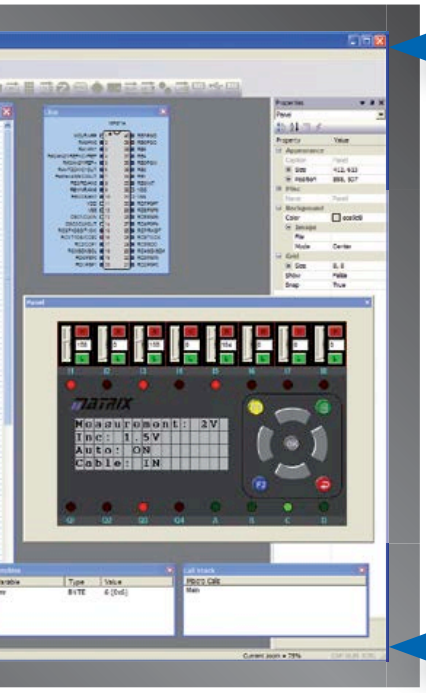
- Start, Stopp, Pause und Schritt für Flowcode-Programme in Echtzeit
- Anzeige der Programm-Variablen
- Ändern von Variablenwerten
- In-Circuit-Debugging für Formula Flowcode Buggy, ECIO- und MIAC-Projekte

mit Flowcode 5 ...

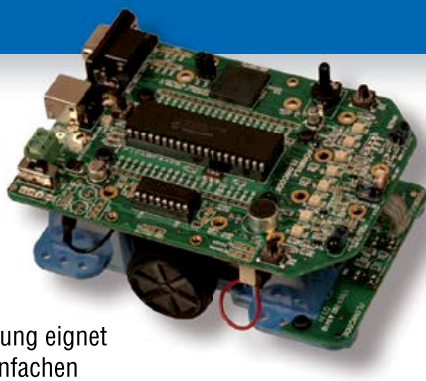


NEU in Flowcode 5:

- Neue C-Code-Ansicht und -Anpassung
- Verbesserte Simulation
- Funktion zum Suchen und Ersetzen
- Neue Variablen-Typen und Funktionen, Konstanten und Port-Variablen
- Automatische Projekt-Dokumentation
- Neuer Projekt-Explorer vereinfacht die Code-Erstellung
- Implementierung von Code-Bookmarks zur Programm-Navigation
- Zugriff auf mehr Chip-Funktionen durch komplettes Redesign des Interrupt-Systems
- Compilierungsfehler und Warnungen navigieren zu Icons
- Icon-Deaktivier-Funktion
- Verbesserte Annotationen
- Verbesserte Links zu Support-Medien



... für Roboter



Beim Formula Flowcode Buggy handelt es sich um ein preiswertes Roboter-Fahrzeug für Lernzwecke und zum Einsatz in der Aus- und Weiterbildung.
Entsprechend programmiert kann man damit auch auf Robotik-Events glänzen.
Das Vehikel lässt sich direkt via USB programmieren und ist mit Linien-Sensoren, Distanz-Sensoren, 8 LEDs, Mikrofon, Lautsprecher und einem E-blocks-Erweiterungs-Port ausgestattet. Die Lösung eignet sich für einen weiten Bereich an Robotik-Experimenten von der einfachen Linienverfolgung bis zum Entkommen aus einem Labyrinth. Via Erweiterungs-Port kann man Displays, Bluetooth- und Zigbee-Funk oder gar GPS anschließen.

... für USB-Projekte

ECIO-Module enthalten leistungsfähige via USB-programmierbare Mikrocontroller im Format von DIL-ICs mit 28 oder 40 Pins (0,6”). Technisch basieren sie auf Mikrocontrollern der PIC18- oder ARM7-Serien. ECIO-Module eignen sich perfekt für eigene Projekte wie auch für den Unterricht, da sich damit komplette Lösungen realisieren lassen. ECIO-Module können mit Flowcode, C oder Assembler programmiert werden.
Neue USB-Routinen in Flowcode bieten sich zum extrem schnellen Prototypenaufbau für USB-Projekte an und unterstützen USB-HID, USB-Slave und USB-Serial-Bus (nur PIC). Eigene Projekte können durch integrierte ECIO-Module um USB-Programmierbarkeit ergänzt werden.

Weitere Produkte und Infos zu E-blocks finden Sie unter

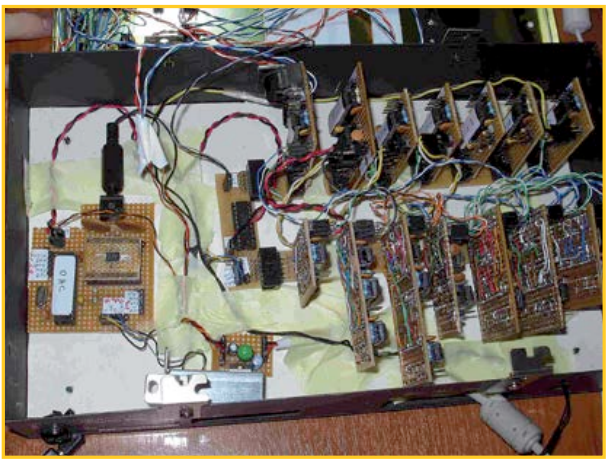
www.elektor.de/eblocks

Feierbiester...

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

Diesen Monat haben wir Grund zum Feiern, weil Elektor.LABS das 5.000. registrierte Mitglied begrüßen durfte. Danke! Wir feiern auch, dass wir nun für herausragende Beiträge auf der .LABS-Website Gold-Mitgliedschaften ausloben können. Ein weiterer guter Grund, sich .LABS anzuschließen ...

Die Wiedergeburt des Formants



Die meisten, die ein Projekt auf .LABS posten, beschränken sich auf ein paar Textzeilen und manchmal ein Foto oder einen Schaltplan. Nicht so der griechische Einsender „AChorevas“. Ohne bereits Mitglied der Elektor-Familie zu sein, kontaktierte er uns, um sein Projekt **D-Formant** auf Elektor.LABS zu posten, eine **d**igitale Version des legendären analogen modularen Musik-Synthesizers Formant aus den späten siebziger Jahren. Nach dem Lesen der Zusammenfassung des Projekts, die er uns freundlicherweise zur Verfügung stellte, waren wir mehr als glücklich, ihm freien Zugang zur .Labs-Website zu gewähren. Doch nicht einmal in unseren kühnsten Träumen haben wir erwartet, dass dies zu fünfzehn Beiträgen mit detaillierten Beschreibungen, Tonbeispielen, Schaltplänen und Quellcodes führen würde.

Der D-Formant des OPs ist ein komplett digitaler Sound-Synthesizer. Alle analogen Signale des ursprünglichen Formants wurden durch 16-bit-Digital-Samples ersetzt, während ein PIC24-Mikrocontroller die Rolle der Transistoren und Operationsverstärker eingenommen hat. Der Entwurf ist völlig modular und bietet die gleichen Konfigurationsmöglichkeiten wie der alte Formant, so dass der Benutzer neue Sounds on the fly erschaffen kann. Alle ursprünglichen Einstellungen wurden umgesetzt, obwohl der Prototyp eine viel einfachere und günstigere Benutzeroberfläche besitzt: ein LCD, ein paar Taster und einen Drehencoder (das allein ist Grund genug, das Projekt zu mögen ;-)). Zum Betrieb des Synthesizers kann ein Standard-MIDI-Keyboard oder ein anderes MIDI-Gerät benutzt werden, das die Befehle „note on“ und „note off“ senden kann.
Wir werden auf jeden Fall über dieses hervorragende Projekt in der Zeitschrift berichten. Inzwischen haben wir AChorevas eine kostenlose Gold-Mitgliedschaft verliehen. Herzlichen Glückwunsch!
www.elektor-labs.com/node/3124

(130098)

www.elektor-labs.com

OP steht für Original-Poster, die Person, die ein Online-Projekt oder eine Diskussion startet. OPs, die ihre Projekte in der Druckausgabe von Elektor sehen wollen, sollten (regelmäßig) ihre E-Mail-Adresse überprüfen, die sie für Elektor.Labs verwenden. Das ist unsere einzige Möglichkeit, Kontakt mit Ihnen aufzunehmen!

SMDs entlöten

Von **Thijs Beckers,**
Elektor-Redaktion

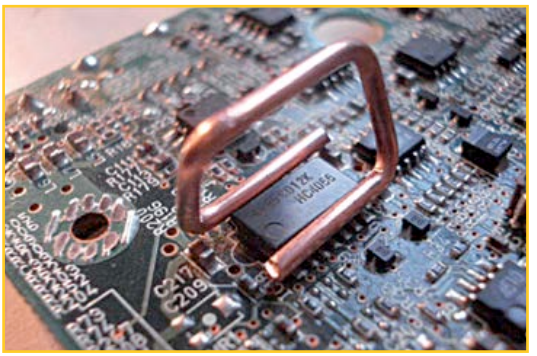
Wenn ein technisches Problem groß und kompliziert erscheint, dann muss eine geeignete Lösung nicht unbedingt teuer sein oder aus der High-Tech-Welt stammen. Luc Lemmens aus dem Elektor-Labor möchte hier gerne einen guten Tipp zum Entlöten von SMD-ICs weitergeben, den er im Internet gefunden hat. Man braucht dazu lediglich eine kleine, feine Zange, ein Stück Kupferdraht (keine Litze; gut geeignet ist der Klassiker mit 1,5 mm²), einen kräftigen Lötkolben, etwas Lötzinn und eine Pinzette.

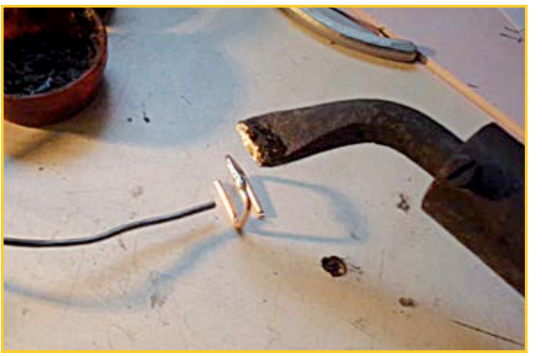
Wenn man ICs mit Anschlüssen auf zwei gegenüberliegenden Seiten auslöten möchte, dann biegt man sich ein Stück Kupferdraht so zurecht, wie auf dem ersten Foto abgebildet. Der Draht sollte auf den IC-Pins so gerade wie möglich aufliegen, damit so viele Pins wie möglich gleichzeitig berührt werden. Nun kommt an die Stelle des Drahtes, die erhitzt werden soll, etwas Lötzinn (siehe das zweite Bild). Das verbessert die Wärmeleitung zu den IC-Pins und den Pads auf der Platine beträchtlich. Danach setzt man das Drahtgebilde wie im dritten Foto auf die IC-Pins und erhitzt den Draht oben mit dem Lötkolben (mit leichtem Druck). Beide Pin-Reihen sollten guten Kontakt mit dem Draht haben. Wenn das Zinn geschmolzen ist, sollte man das IC schnell mit der Pinzette von der Platine heben.

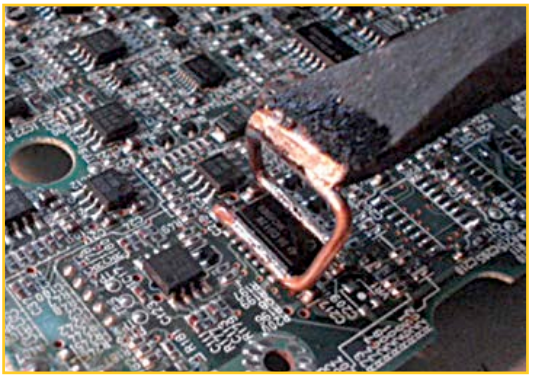
Mit einer passenden Biegung, wie im vierten Bild gezeigt, kann man auf die gleiche Weise sogar ICs entlöten, die Pins an allen vier Seiten haben. Man sollte darauf achten, IC und Platine nicht durch Überhitzung zu beschädigen. Von daher sollte man die Zeit der Hitzezufuhr soweit beschränken, dass gerade das Zinn schmilzt und man das IC entfernen kann. Ein IC, das zuviel Hitze abbekommen hat, indem es entweder zu lang oder auf eine zu hohe Temperatur erhitzt wurde, ist irreparabel beschädigt. Ähnliches gilt für die Platine, bei der sich dann Löt-Pads von der Platinenoberfläche lösen können. Wenn Sie aber den Dreh raus haben und nach dieser Methode richtig mit Zinn, Draht und Lötkolben umgehen können, dann wird weder das IC noch die Platine beschädigt.

(130099)

Gefunden auf http://youtu.be/dCUSwADP6DE.







Entwurf eines Schaltplans

Von **Neil Gruending**
(Kanada)

Letztes Mal habe ich berichtet, wie DesignSpark Technology-Dateien verwendet, um Konfigurationseinstellungen zu speichern. Jetzt wollen wir ein neues Projekt beginnen und starten natürlich mit dem Schaltplan. Zunächst konfigurieren wir die DesignSpark-Bibliotheken und richten dann ein Schriftfeld ein, so dass wir ein professionell anmutendes Schema erstellen können.

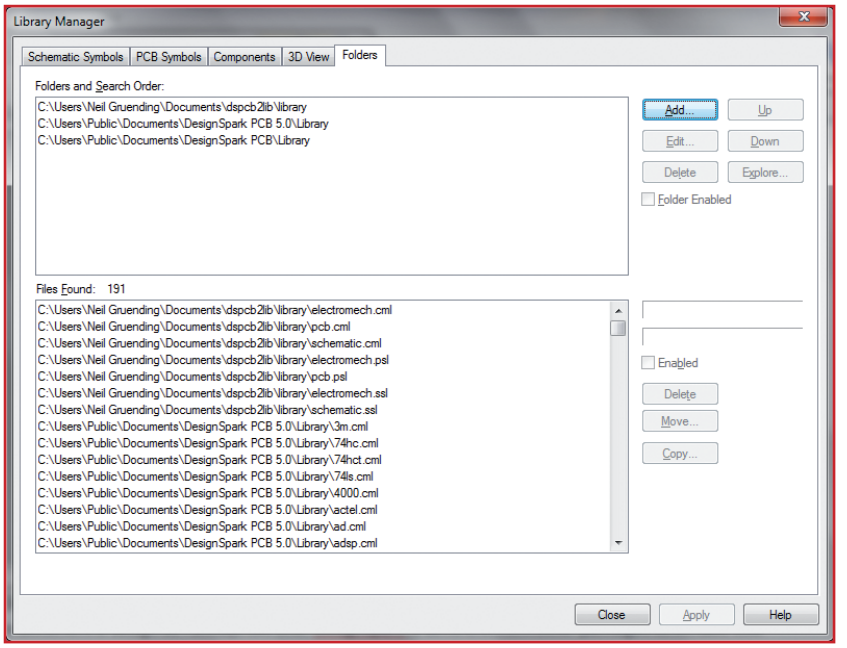


Bild 1.
Kontrolle der Bibliotheks-Dateipfade.

Konfiguration der Bibliotheken

DesignSpark verwendet Library-Dateien zur Organisation aller Design-Informationen. *Schematic symbols* ist ein Bibliothekstyp, *PCB Footprints* ein weiterer. Sie werden zu einer Bauteil-Bibliothek kombiniert, die man verwenden kann, um Bauteile und Dokumentationssymbole in das Design zu platzieren. Der einzige Unterschied zwischen einem Dokumentationssymbol und einem normalen Bauteil ist, dass das Dokumentationssymbol entweder ein Schaltplansymbol oder ein Platinen-Symbol darstellt, niemals aber beides enthält. Wenn Sie sich weiter über die Bedeutung und Anwendung des DesignSpark-Bibliothekssystems informieren möchten, finden Sie in [1] ein gutes Tutorial.

In diesem Artikel werden wir ein Schaltplan-Dokumentationssymbol für ein Schriftfeld (*title block*) im Schaltplan entwerfen, aber zuvor überprüfen wir die Dateipfade der DesignSpark-Bibliothek. Dazu wählen Sie im „Files -> Libaries ...“- Menü „Folders“ und erhalten ein Fenster wie in **Bild 1**.
Sie müssen nun sicherstellen, dass das Verzeichnis, in dem Sie die Library-Dateien speichern, an erster Stelle unter „Folders and Search Order“ genannt ist (bei mir „C:\Users\Neil Gruending\Documents\dspcb2lib\library“). Sie können das Verzeichnis mit den Up/Down-Tasten neu ordnen. Ich empfehle, dass Sie Änderungen oder neue Dateien nicht in den standardmäßigen Bibliotheksordnern des DesignSpark-Systems speichern, da ein zukünftiges Upgrade sie hier überschreiben würde.
Nun, da die Bibliothekspfade bestimmt sind, können wir eine neue Schaltplan-Symbolbibliothek einrichten, um unser Schaltplan-Schriftfeld zu speichern. Hierzu klicken Sie auf den „New Lib ...“-Knopf auf dem „Schematic Symbols“-Tab. Wählt man nun „New Lib ...“, so öffnet sich eine leere Schaltplansymbol-Seite. Für weitere Informationen konsultieren Sie bitte das schon genannte *symbol creation tutorial* der DesignSpark-Website [1].

Ein neues Schriftfeld

Ich persönlich verwende immer Schriftfelder im Schaltplan, da dies „professioneller“ aussieht und hilft, ein Design zu dokumentieren. DesignSpark unterscheidet sich von anderer PCB-Software, weil Schriftfelder als Schaltplan-Bauteile behandelt werden und nicht als Template- oder Technology-Datei. Das bedeutet, dass DesignSpark alle Zeichnungselemente in einer Schaltplan-Technology-Datei ignoriert. DesignSpark stellt einige Schriftfeld-Vorlagen in seiner Schaltplan-Bibliothek in verschiedenen Größen wie A4 und Letter zur Verfügung. Ich ziehe aber „Tabloid“ (11 inch × 17 inch) vor.
In meinem letzten Artikel habe ich Ihnen gezeigt, wie Sie proportionale TrueType-Schriftarten in einer Schaltplan-Technology-Datei verwenden, aber es gibt einen Haken, will man sie in Schaltplan-Schriftfeldern verwenden. DesignSpark ver-

schiebt nämlich proportionale Schriftarten beim Drucken in eine PDF-Datei leicht nach unten. Das spielt in der Regel etwa bei Bauteilbezeichnungen keine große Rolle, aber in Schriftfeldern, wo die Textausrichtung wichtig ist, werden Sie dies auf jeden Fall bemerken. Deshalb wähle ich für meine Schriftfelder stets nichtproportionale Schriftarten, wie in **Bild 2** zu sehen ist.
Ich empfehle, dass Sie die verschiedenen Textarten eindeutig benennen, um sie später leicht modifizieren zu können. Bei mir habe ich die Schriftarten wie in **Bild 3** organisiert.
Die Zahlen und Buchstaben rund um die Zeichenfläche benutzen die Zeichenart „Frame“, die Feldbezeichnungen den Stil „Titel-small“ und Elemente im Feld den Stil „Titel“. Da DesignSpark keine Projekt-Variablen unterstützt, müssen Sie die Text-Strings des Schriftfelds manuell hinzufügen, weshalb alle Schriftfelder zunächst leer erscheinen. Die Textstile müssen Sie übrigens nicht zu einer Schaltplan-Technology-Datei hinzufügen, da sie in den Schaltplan kopiert werden, wenn Sie das Schriftfeld hinzufügen.
Wenn das Schriftfeld fertig ist, sichern Sie es in der Schaltplan-Symbolbibliothek, die Sie zuvor erstellt haben, so dass wir nun ein Schaltplan-Symboldokument erstellen können. Der erste Schritt ist, den Library-Manager („File-> Libraries ...“) zu öffnen und zu Tab „Components“ zu gehen. Sie können eine neue Bauteilbibliothek erstellen, indem Sie den „New Lib ...“-Knopf anklicken und dann ein Dokumentensymbol durch Klicken auf die „New Item ...“-Schaltfläche anlegen. Ein neues „New component“-Fenster öffnet sich. Hier gibt man dem Schriftfeld einen Namen und wählt das Schriftfeld-Symbol. Deaktivieren Sie „PCB Symbol“, damit das Schriftfeld als Schaltplanbestandteil gekennzeichnet wird. Nach dem Speichern der Änderungen können wir ein neues Projekt in DesignSpark beginnen.

Ein neues Projekt!

Projekte in DesignSpark sammeln alle relevanten Informationen zu einem Design wie Schaltpläne und Platinendokumente an einem Ort. Der Hauptgrund für die Verwendung eines Projekts ist es, eine Reihe von Schaltplan-Seiten zu einem Platinenlayout zu verknüpfen. Die verbundenen Schaltplanseiten können als großes Projekt mit globalen Netzinformationen und gemeinsamen Bauteilkennzeichnungen angesehen werden.
Das Erstellen eines neuen Projekts in DesignSpark ist einfach. Gehen Sie in das „File -> New“-Menü, um ein „New Design“-Fenster zu öffnen, wählen Sie „Project“ und bestätigen Sie mit „OK“. Sie werden dann aufgefordert, das neue Projekt zu speichern. Jetzt können Sie bestehende Dateien mit „Project -> Add Files to Project ...“ zum Projekt hinzufügen. Die Eingabe neuer Elemente zu einem Projekt erfolgt im „New Design“-Fenster, bevor Sie aber auf OK drücken, überprüfen Sie die „Add to Open Project“-Box auf ihre Richtigkeit.

Was wir nun können:

Jetzt, wo wir Projekte und nett aussehende Schaltplanvorlagen erstellen können, ist es der nächste Schritt, Ihr Design mit Hilfe der Bauteile aus den Bibliotheken von DesignSpark zu zeichnen. Sie können auch eigene Bibliotheken mit zusätzlichen Bauteilen und ihren Attributen erstellen und nutzen. Dies macht es einfacher, später eine Stückliste (BOM, bill of material) zu generieren. Das hier von mir entworfene Schriftfeld (**Bild 4**) ist im dsppcb2lib-Projekt auf Bitbucket [3] gespeichert.

(130181)

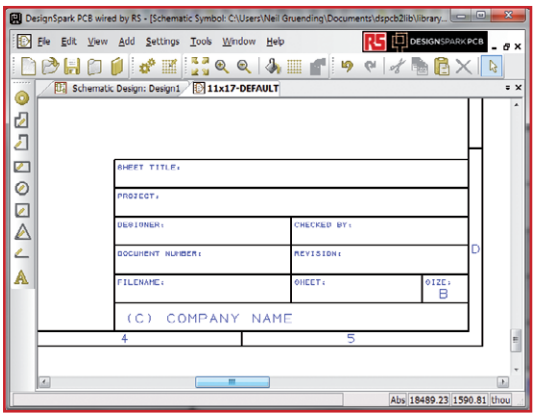


Bild 2. Struktur und Layout des Schaltplan-Schriftfelds.

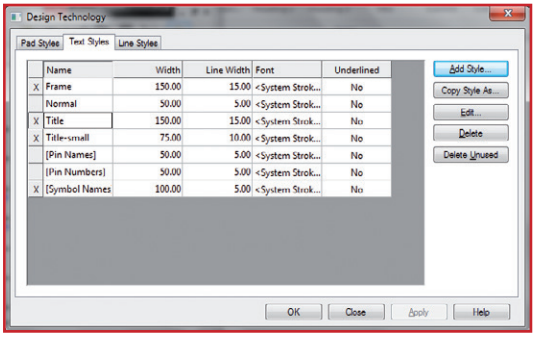


Bild 3. Textformatierung des Schriftfelds.

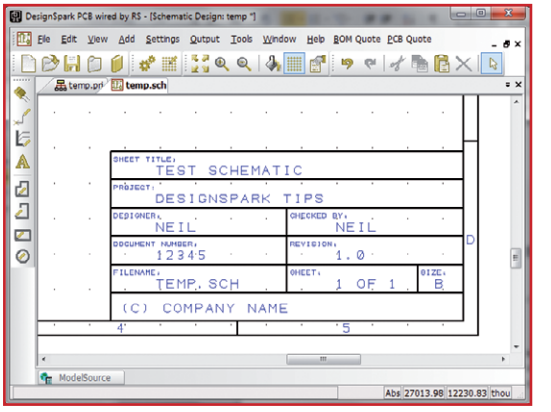


Bild 4. Ein vollständiges Schriftfeld für den Schaltplan.

Weblinks:

[1] www.designspark.com/tutorial/components-library-structure-library-manager

[2] www.designspark.com/tutorial/components-creation-with-symbol-footprint-wizards

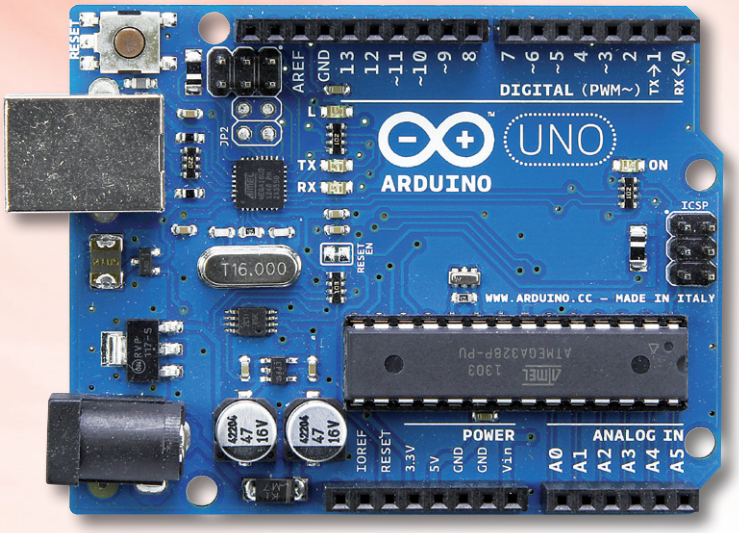
[3] https://bitbucket.org/neilg/dspcb2lib

Gleich oder verschieden?

Von **Wisse Hettinga**

Der Halbleiterhersteller Renesas legt besonderen Wert darauf, dass das neueste Board vom Typ GR Sakura arduinokompatibel ist.

Diese Übersicht der Eigenschaften erlaubt Ihnen selbst zu beurteilen, wie weit diese Kompatibilität geht.

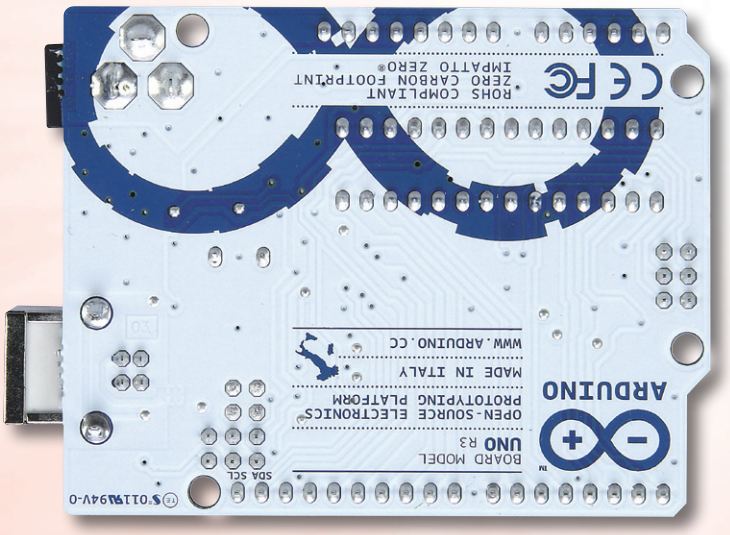


Unterschied 1

Ja, das GR Sakura Board ist pink! Aber schaut man auf die Daten gilt: Pink = Power!

Unterschied 2

Gegenüber dem AVR-Controller mit 8 bit, 16 MHz und wenig RAM und Flash kann das Renesas-Produkt mit vollen 32 bit bei 96 MHz und richtig viel Speicher aufwarten. Die eigentliche Frage ist, welche Anwendung das Potential von GR Sakura wirklich ausreizen kann.

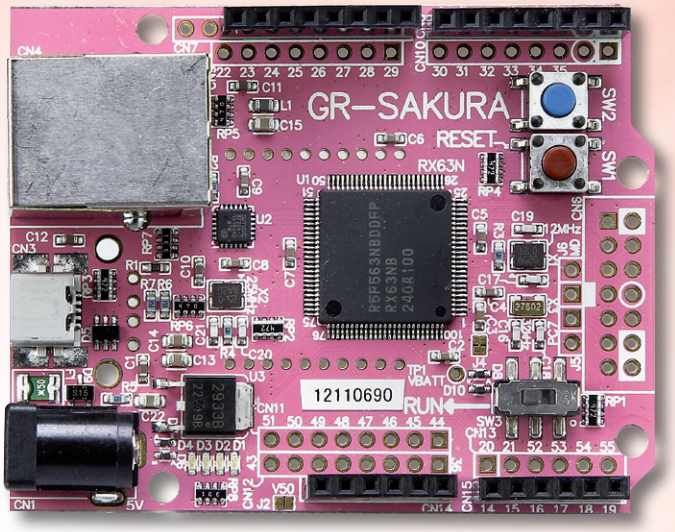


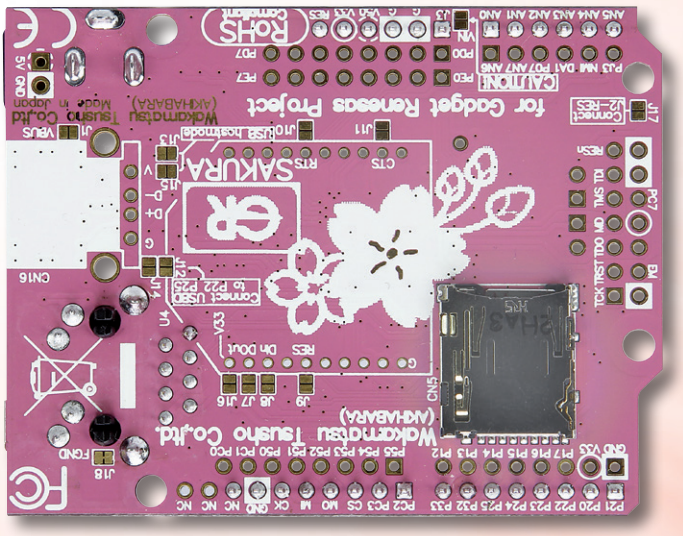
Unterschied 3

Das Sakura-Board fungiert auch als USB-Host. Hierfür ist eine Mini-B-Buchse vorgesehen. Eine Buchse vom Typ A kann noch auf der Rückseite der Platine bestückt werden.

	Arduino Uno	GR Sakura
Versorgungsspannung	5 V für den Mikrocontroller 5 V für das Board	3,3 V für den Mikrocontroller 5 V für das Board
USB	Buchse des Typs B Board wird direkt von USB versorgt	Mini-B-Buchse Host-Support Board wird direkt von USB versorgt
Netzwerk	keines	Ethernet mit RJ45-Buchse

ARDUINO UNO versus GR SAKURA





Unterschied 4

Es ist klar, dass die eigentliche Stärke des Arduino-Konzepts in der großen und leicht handhabbaren Programm-Library (unter www.arduino.cc) liegt – nicht so sehr in der Hardware. Doch die Programmier-Möglichkeiten des Sakura-Boards sind auch nicht zu verachten, was man schnell merkt, wenn man den cloud-basierten Compiler startet. Einfach das Board mit einem PC verbinden, die richtigen Tasten drücken und schon erscheint das Board als neues Laufwerk. Das alles ist sauber dokumentiert. Auf dem neuen Laufwerk befindet sich ein Link, der schnurstracks zur zugehörigen Website führt. Wenn Sie ein Android-Smartphone haben, dann schauen Sie sich einmal Gadget Director an, eine einfache „icon-basierte“ Programmiersprache.

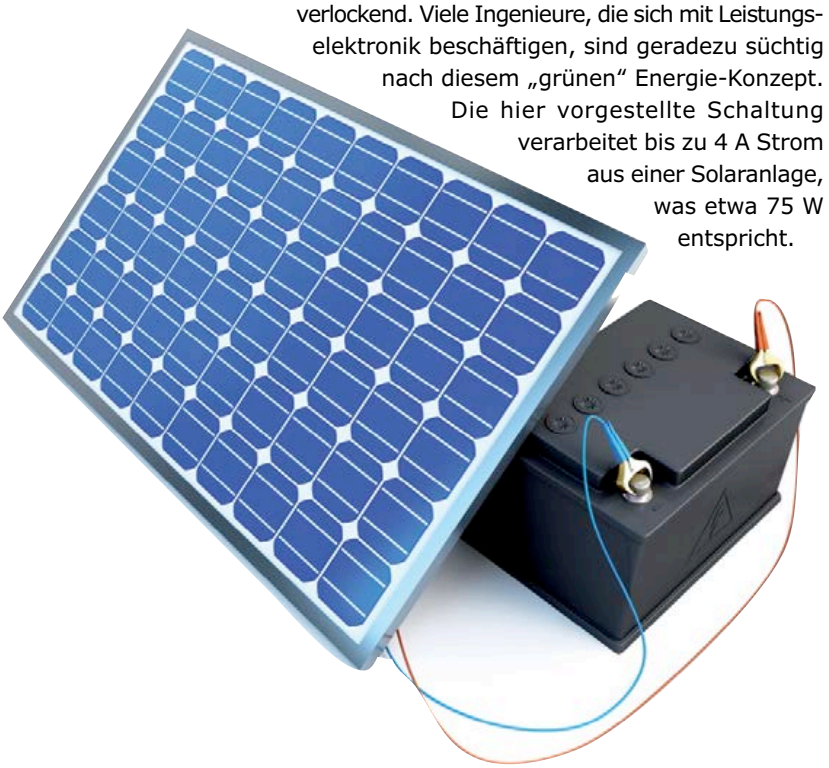
Weitere Infos unter www.designspark.com und dort bei „Design Centers“.

(130177)

	Arduino Uno	GR Sakura
Controller	ATmega328 8 bit 16 MHz Takt	RX63N 32 bit 96 MHz Takt
Speicher	32 KB Flash-Speicher (davon 0,5 KB vom Bootloader belegt) SRAM: 2 KB EEPROM: 1 KB	1 MB Flash-Speicher RAM: 128 KB Daten-Flash: 32 KB MicroSD-Slot

4-A-Solarlader

Der Einsatz von Solarenergie ist sowohl aus ökologischen wie auch aus Gründen der Kosteneffizienz verlockend. Viele Ingenieure, die sich mit Leistungselektronik beschäftigen, sind geradezu süchtig nach diesem „grünen" Energie-Konzept. Die hier vorgestellte Schaltung verarbeitet bis zu 4 A Strom aus einer Solaranlage, was etwa 75 W entspricht.



Von **T. A. Babu** (Indien)

Ein Lade-Algorithmus namens „Puls-Zeit-Modulation" wird in diesem Entwurf vorgestellt.

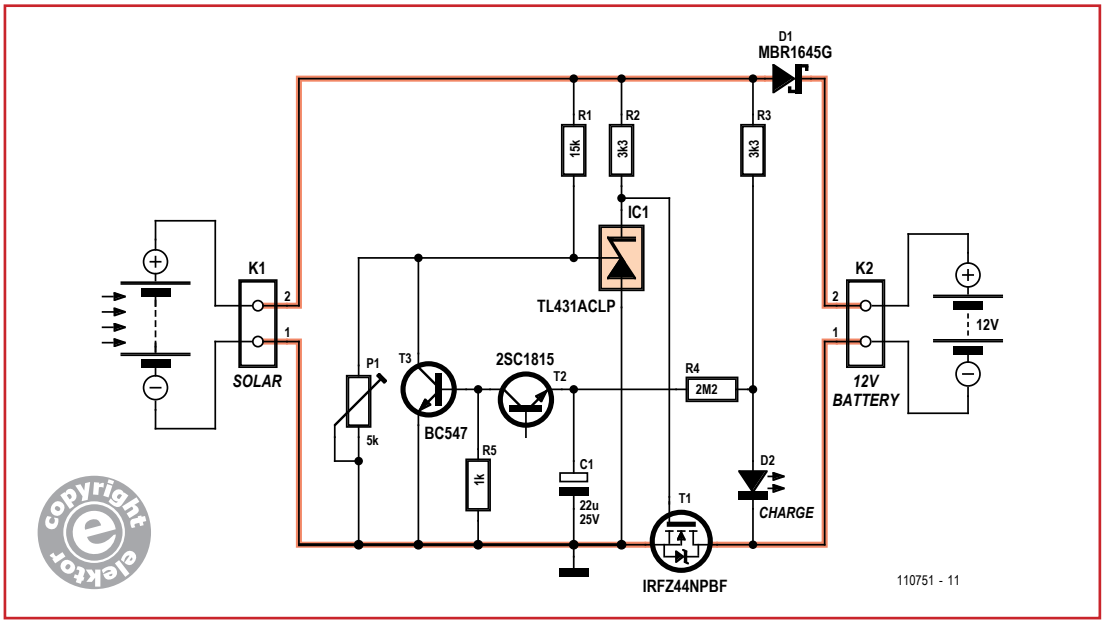
Der Strom aus dem Solarpanel zur Batterie wird vom N-Kanal-MOSFET T1 gesteuert. Dieser MOSFET benötigt keinen Kühlkörper, da sein niedriger RD-S(on) von nur 0,024 Ω eine nennenswerte Erwärmung von vornherein ausschließt. Die Schottky-Diode D1 verhindert die Entladung der Batterie über das Solar-Panel in der Nacht oder bei Abschattung. Außerdem bietet D1 auch Schutz vor Verpolung der Batterie. Im Schaltplan zeigen die roten Linien Pfade der potenziell höheren Ströme.

Der Laderegler bezieht niemals Strom von der Batterie, er wird völlig aus dem Solarpanel versorgt. In der Nacht geht der Laderegler schlafen, erst tagsüber bei Tageslicht, sobald das Solarpanel genug Strom und Spannung produziert, geht er an die Arbeit und startet das Laden der Batterie.

Die Klemmenspannung der Batterie wird von R1 und Trimmpoti P1 heruntergeteilt. Die resultierende Spannung bestimmt den Ladestatus des Controllers. Das Herz des Ladereglers ist IC1, ein Referenzspannungsregler TL431ACZ mit einem Open-Collector-Fehler-Verstärker. Die Batterie-spannung wird ständig mit der internen Referenz des TL431 verglichen. Solange der mit P1 eingestellte Wert unterhalb der internen Referenz-spannung liegt, sorgt IC1 dafür, dass der MOSFET leitet. Während die Batterie lädt, steigt ihre Klemmenspannung. Ist die Ladeschlussspannung erreicht, fällt das Ausgangssignal von IC1 auf Low (weniger als 2 V) und schaltet den MOSFET und dadurch den gesamten Stromfluss zur Batterie ab. Schaltet T1 aus, wird auch LED D2 dunkel.

Das Regler-IC besitzt keinen Hysterese-Pfad. Folglich bleibt der Ausgang des IC1 low, sobald kein Strom zur Batterie fließt und verhindert so, dass der MOSFET weiter leitet, wenn die Batteriespannung abfällt. Blei-Säure-Batterien fordern aber eine Erhaltungsladung. Ein sehr einfacher Oszillator wurde implementiert, um diese Funktion zu übernehmen. Der Oszillator nutzt den negativen Widerstand in Transistoren, ein Effekt, der von Leo Esaki im Rahmen seiner Forschung über das Tunnelphänomen in Festkörpern entdeckt wurde (eine Tat, für die er 1973 den Nobelpreis für Physik erhielt).

Hier wird dazu ein alltäglicher NPN-Transistor Typ 2SC1815 eingesetzt. Wenn die LED erlischt, lädt R4 einen 22-µF-Kondensator (C1), bis die Spannung hoch genug ist, um am Emitter-Basis-Übergang von T2 eine Stoßentladung zu verursachen. Zu diesem Zeitpunkt schaltet der Transistor durch und entlädt den Kondensator über R5. Der Spannungsabfall über R5 ist ausreichend, um T3 zu betätigen, was zu einer Änderung der Referenzspannungseinstellung führt. Jetzt versucht der MOSFET wieder, den Akku aufzuladen. Sobald die Batteriespannung die Ladespannung erreicht, wiederholt sich der Vorgang. Der 2SC1815-Transistor erwies sich als zuverlässig bei dieser Arbeit. Andere Transistortypen verhielten sich dagegen recht launisch. Wenn Sie wissen wollen, warum,



lesen Sie doch mal in Esakis ausgezeichneter Arbeit nach. Machen Sie sich aber auf ein wenig höhere Mathematik gefasst!

Wenn die Batterie voller und voller wird, verkürzt sich die Einschaltzeit des Oszillators, während die Aus-Zeit weiterhin durch die zeitmaßgeblichen Bauteile R4 und C1 bestimmt wird. So wird der Stromimpuls zur Batterie im Laufe der Zeit immer weiter verkürzt. Dieses Ladeverfahren kann man durchaus als Puls/Zeit-Modulation bezeichnen.

Um die Schaltung abzugleichen, benötigen Sie ein gutes digitales Voltmeter und eine variable Stromversorgung. Stellen Sie die Stromversorgung auf 14,9 V ein, das ist die Ladeschlussspannung von 14,3 V plus etwa 0,6 V über der Schottky-Diode. Drehen Sie das Trimmpoti, bis die LED erlischt. Dies ist der Umschaltpunkt, die LED beginnt zu flackern. Eventuell müssen Sie diese Einstellung mehrmals vornehmen. Je näher man den Komparator an genau 14,3 V heranführt, desto genauer wird auch die Ladeschaltung sein. Trennen Sie das Netzgerät vom Laderegler und schließen Sie das Solar-Panel an. Die Ladeschlussspannung von 14,3 V ist für die meisten Blei-Säure-Batterien geeignet, konsultieren Sie aber sicherheitshalber das Datenblatt des Herstellers. Achten Sie auch darauf, dass das Solarpanel nicht mehr Strom liefern kann, als die von Ihnen gewählte Batterie verträgt.

Die Schaltung und das Platinenlayout für dieses Projekt wurden mit DesignSpark entwickelt und stehen unter www.elektor.com/110751 zum Download zur Verfügung.

(110751)

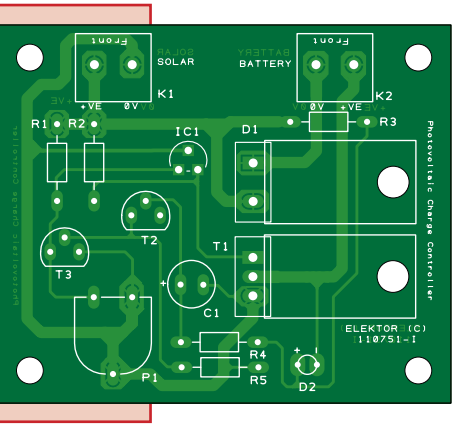
Stückliste

Widerstände
R1 = 15 k
R2,R3 = 3k3, 1%
R4 = 2M2
R5 = 1 k
P1 = 5 k Trimmpoti

Kondensator
C1 = 22 µ, 25V, radial

Halbleiter
D1 = MBR1645G (ON Semiconductor)
D2 = LED 5 mm
IC1 = TL431ACLP (Texas Instruments)
T1 = IRFZ44NPBF (International Rectifier)
T2 = 2SC1815 (Toshiba) (markiert mit: C1815)
T3 = BC547

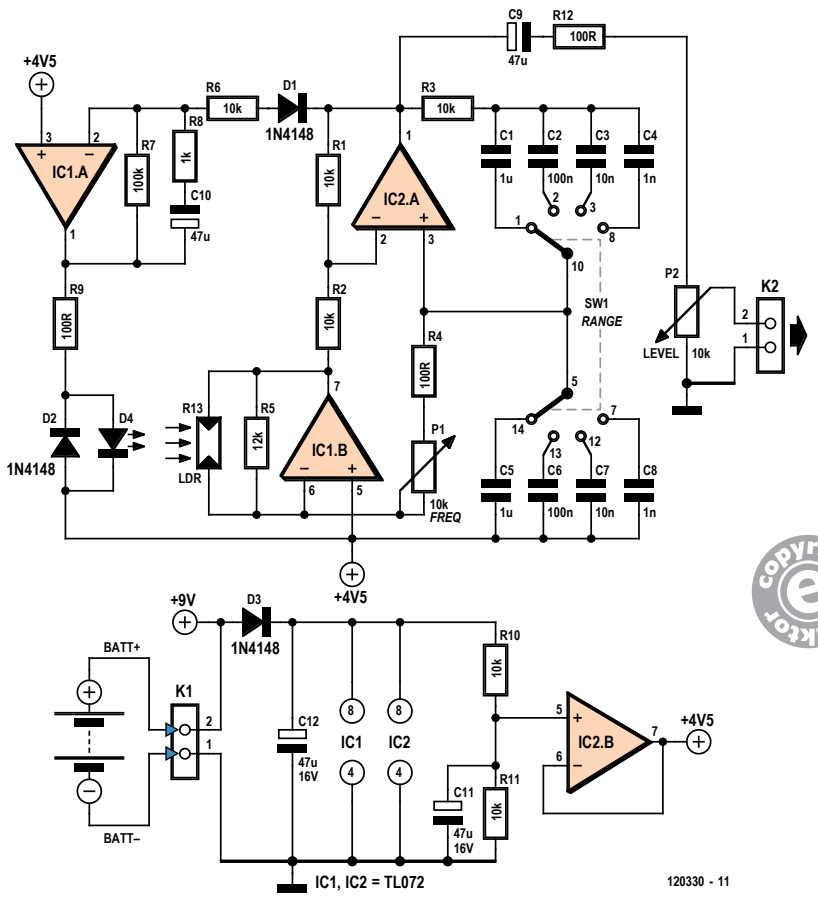
Außerdem
K1,K2 = 2-polige Platinenanschlussklemme, RM 5 mm
Platine 110751-1

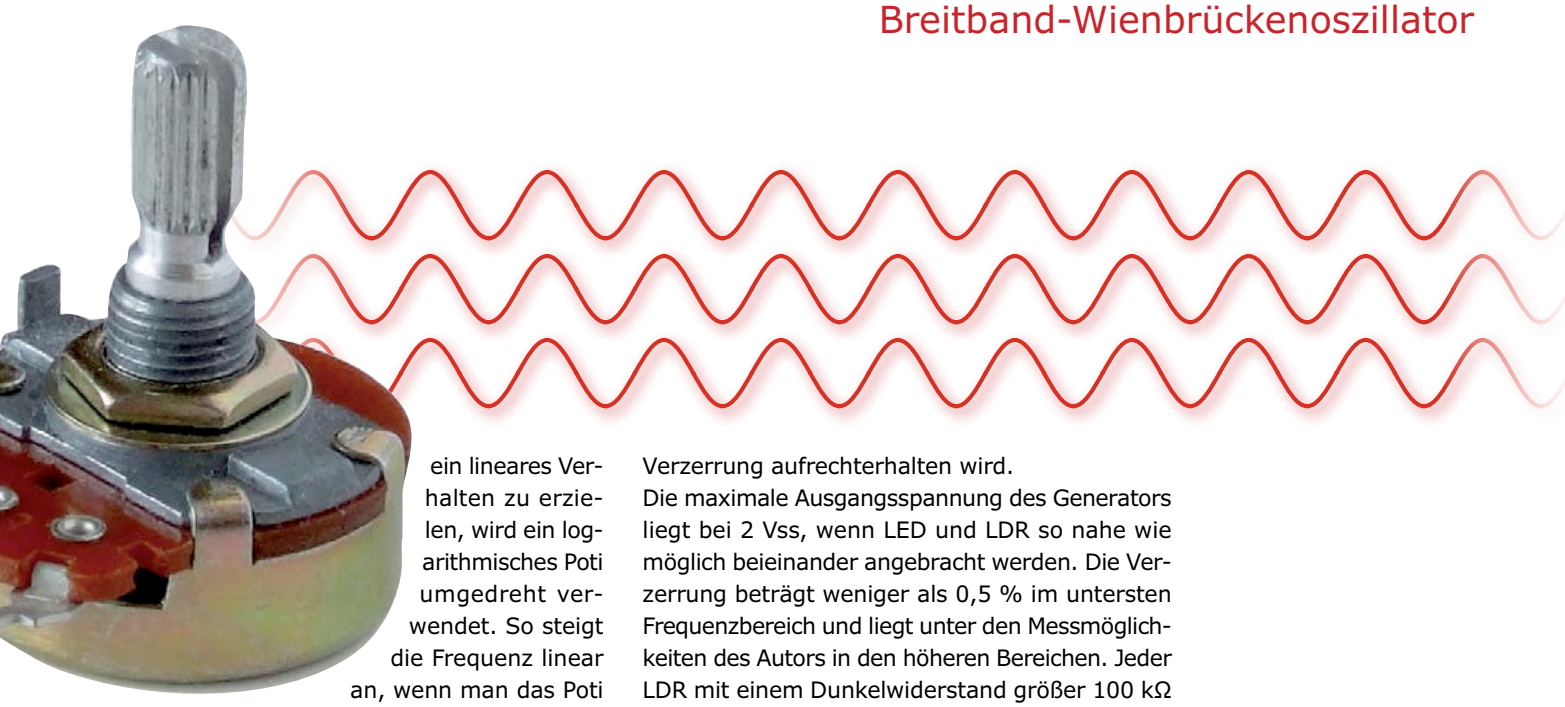


Breitband-Wienbrückenoszillator mit 1-Gang-Poti

Von **Merlin Blencowe**
(Großbritannien)

Der nach Max Wien (geb. 1866 in Königsberg, gest. 1938 in Jena) benannte Wien-Brücken-Oszillator liefert bei geringer Verzerrung ein Sinussignal konstanter Amplitude von etwa 15 Hz bis 150 kHz. Dazu sind nur vier Operationsverstärker erforderlich, die von einer 9-V-Batterie versorgt werden. Im Gegensatz zu den meisten Wien-Brücken-Oszillatoren wird bei dieser Schaltung kein teures Mehrgangpoti für den Abgleich benötigt.
Opamp IC2b sorgt für eine künstliche Masse, so dass die Schaltung aus einer unipolaren Spannungsquelle (Batterie oder Netzteil) betrieben werden kann. IC2a ist der Hauptverstärker des Oszillators. Der Frequenzbereich wird vom 2x4-Drehschalter SW1 in vier Dekaden unterteilt. Nur eine Seite des Wien-Netzwerks wird variiert, aber die dadurch bedingte Änderung der positiven Rückkopplung wird durch IC1b kompensiert, der R2 ansteuert. Dabei ist die negative Rückkopplung ausreichend, um die Oszillation aufrecht zu erhalten. Eine lineare Änderung des Widerstands des Abstimmungspotis bewirkt eine in etwa logarithmische Änderung der Frequenz. Um





ein lineares Verhalten zu erzielen, wird ein logarithmisches Poti umgedreht verwendet. So steigt die Frequenz linear an, wenn man das Poti nach links dreht. Man könnte auch ein „antilogarithmisches“ Poti einsetzen, aber diese Teile sind nur schwer erhältlich.

Integrator IC1a überwacht die Amplitude des Ausgangssignals und steuert LED D2 an. Die LED beleuchtet einen lichtempfindlichen Widerstand. Diese optische Verbindung muss von der Außenwelt, sprich dem Umgebungslicht abgeschottet werden, zum Beispiel durch einen Schrumpfschlauch. IC1a kann dann die Verstärkung so einstellen, dass eine Schwingung mit minimaler Verzerrung aufrechterhalten wird.

Die maximale Ausgangsspannung des Generators liegt bei 2 Vss, wenn LED und LDR so nahe wie möglich beieinander angebracht werden. Die Verzerrung beträgt weniger als 0,5 % im untersten Frequenzbereich und liegt unter den Messmöglichkeiten des Autors in den höheren Bereichen. Jeder LDR mit einem Dunkelwiderstand größer 100 kΩ sollte funktionieren. Wenn Sie nicht über einen LDR mit so hohem Widerstand verfügen, erhöhen Sie R5, bis die Schaltung zu schwingen beginnt.

Prototypen der Schaltung auf Experimentierplatinen wurden vom Autor mit Dual- und Quad-Operationsverstärkern bestückt; beide funktionierten gleichermaßen gut.

Die Schaltung und das Platinenlayout wurden in DesignSpark erstellt und stehen unter www.elektor.de/120330 zum Download zur Verfügung.

(120330)

Stückliste

Widerstände

R1,R2,R3,R6,R10,R11 = 10 k
R7 = 100 k
R4,R9,R12 = 100 Ω
R5 = 12 k
R8 = 1 k
P1,P2 = 10 k Poti logarithmisch
R13 = LDR, Dunkelwiderstand > 100 k, z.B Excelitas Tech VT90N1 (Newark/Farnell 2568243)

Kondensatoren

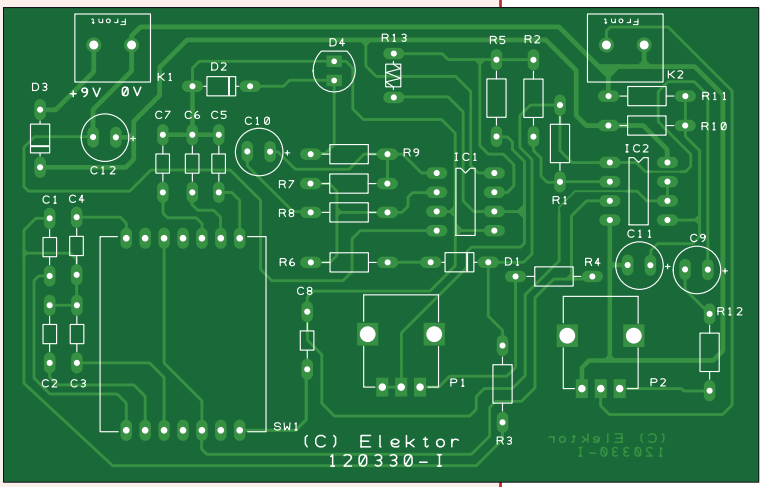
C1,C5 = 1 µ
C2,C6 = 100 n
C3,C7 = 10 n
C4,C8 = 1 n
C9...C12 = 47 µ 16 V, Elko radial

Halbleiter

D1,D2,D3 = 1N4148
D4 = LED rot, 5 mm
IC1,IC2 = TL072ACP

Außerdem

SW1 = Drehschalter 2x4, z.B. C&K Components RTAP42S04WFLSS
K1,K2 = 2-polige Platinenanschlussklemme, RM5
Platine 120330-1



X-treme Einschaltstrom-Begrenzung

Kontrollierter Start für Elko & Co.

Von
Raymond Vermeulen
(Elektor-Labor)

Dieses komplett analoge, mikrocontrollerlose (!) Projekt ist die Reaktion auf den verzweifelten Hilferuf eines Elektor-Mitarbeiters, der außerdem noch eingefleischter Modellflugzeug-Enthusiast ist und große Hochleistungs-Modelle liebt. Er hatte ein Problem mit Stromanschlüssen, die sich selbst zerstörten, wenn er den Akkupack an den Flieger, das heißt die Motorsteuerung anschloss. Der Schaden trat durch starke Funkenbildung aufgrund des hohen Einschaltstroms auf. Das waren recht teure Funken, denn es handelt sich um massive, 6 mm messende vergoldete Anschlüsse. Es war also eine Einschaltstrombegrenzung für einen kontrollierten, funkenlosen Stromfluss erforderlich. Solch ein Regler fällt aber nicht vom Himmel, sondern musste in mühsamer Kleinarbeit... – naja, es dauerte halt einige Zeit, bis das Elektor-Labor eine solche Schaltung entwickeln konnte. Hier ein paar Arbeitsblätter (sprich: Kritzeleien), die bei der Entwicklung des Projekts entstanden. Einen guten Start in das Projekt bot die Motorola-Applikationsschrift AN1542 [1]. Dort wird das Konzept einer Einschaltstrombegrenzung für 37 V Batteriepower und 200 A Last im Normalbetrieb grob skizziert (Bild 1). Um einen insgesamt niedrigen Rds(on) zu erreichen, verwendet man am besten ein paar MOSFETs parallel. Nach einer LTSpice-Simulation zeigte sich aber (schmerzhaft), dass wohl nicht im hohen Soll-Strom, sondern in der Lastkapazität der Grund für die Funkenbildung liegt. So wurde die Schaltung für den Worst-case-Fall entwickelt. Dennoch gab es Bedenken hinsichtlich der Safe-operation-area der MOSFETs. Um das einmal durchzuspielen, wurde eine Messung mit einem kleinen 10-A-3-Phasen-BLDC-Motor-Treiber durchgeführt, aber es erwies sich, dass er eine Eingangskapazität von „nur“ 120 µF aufwies. Dann haben wir eine größere Motorsteuerung, spezifiziert für 120 A, mit einer Eingangskapazität von 13.800 µF (13,8 mF) bei einem ESR von etwa 2,7 mΩ verwendet.

Für die praktische Schaltung haben wir den MOSFET IPB017N06N3 von Infineon ausgewählt, vor allem auf die Zusage des Datenblatts von einem

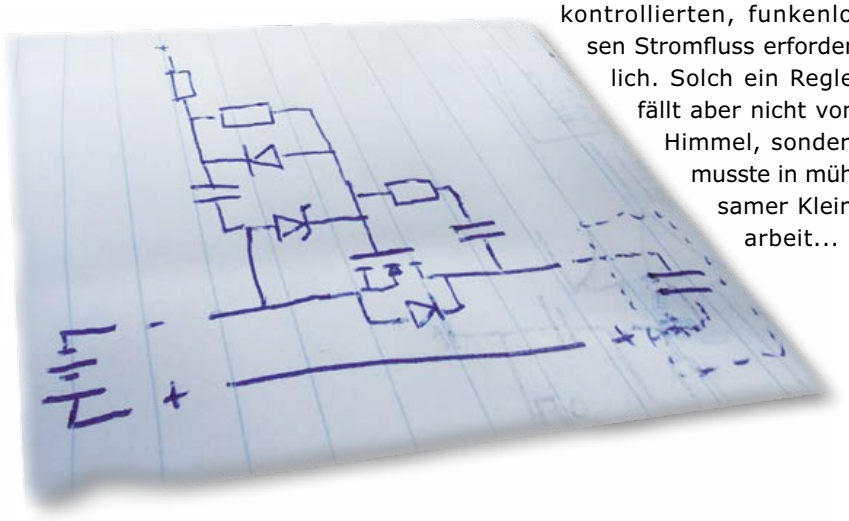


Bild 1.
Eine Skizze à la Bob Pease [3] für einen Einschaltstrombegrenzer.

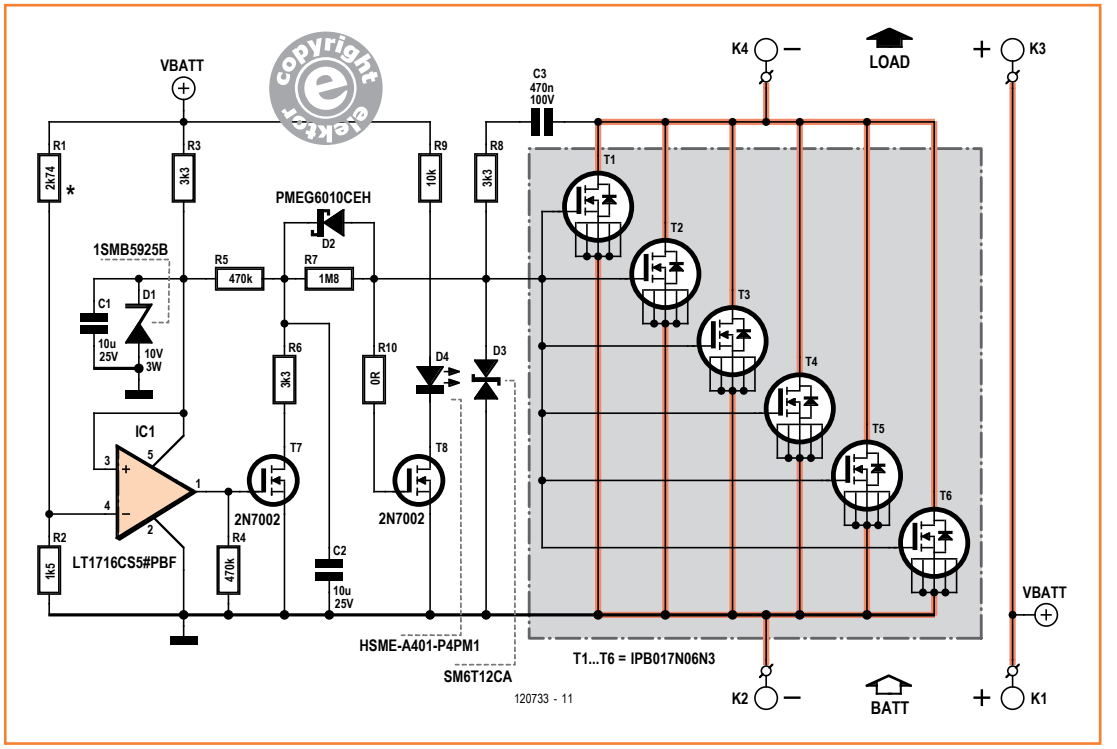


Bild 2.
Schaltung des X-treme-Einschaltstrombegrenzers. Die hohen Strom führenden Leiterbahnen sind hervorgehoben.

On-Widerstand von 1,7 mΩ pro MOSFET hin. Auch die geringen Kosten und die gute Erhältlichkeit bei Distributoren spielte natürlich eine Rolle. Bleibt nun die Frage: Wie viele MOSFETs brauchen wir?

Zurück zur LTSpice-Simulation. Mit dem IPB017N06N3-Modell wurden einige Bauteilwerte neu (fein-)berechnet. Wir haben auch einen passenden und preiswerten Kühlkörper mit Standard-Größe und vorgefertigten Bohrungen ausgemacht. Ein Blick auf den Schaltplan in Bild 2 zeigt einige deutliche Unterschiede zu dem Vorschlag der AN1542. Motorola stellt den Strom als Rechteck dar, was zu plötzlichen Strom- und Spannungsspitzen führt, die langsam abfallen. Im Gegensatz dazu erhöht die hier gezeigte Schaltung den Strom langsam, was zu einem sägezahnförmigen Stromfluss führt. Folglich sieht die Kurve der Verlustleistung (PFET) wie eine umgedrehte Parabel aus. Bild 3 zeigt die grundlegenden Wellenformen, die die SOA der MOSFETs weit besser respektieren als die AN1542.

Ein TVS-Diode (Transient Voltage Suppression) D3 schützt die MOSFETs im Falle einer versehentlichen Verpolung. Ein früher Versuchsaufbau der X-treme-Schaltung wurde mit einem Kondensator von 15.000 µF (15 mF) mit und ohne ohmsche Last getestet. Als Stromquelle diente ein 40-V-Netzteil. Alles schien wie erwartet zu funktionieren, obwohl die Unterspannungsabschaltung ohne ohmsche Last nicht korrekt bei der fallenden Flanke funktionierte.

Als letzter Test wurde die Schaltung an einem BLDC-Controller, der einen 10-kW-Motor trieb, ausprobiert. Er zog 8,5 A bei kontinuierlicher Geschwindigkeit und gedrosselt 20 A bis 30 A. Die Tests wurden mit 37 V und 48 V durchgeführt, „Kaltstarts“ mehrere Male simuliert. Obwohl Kabel und Steckverbinder merklich warm wurden, bewahrten MOSFETs und der Rest der Schaltung kühlen Kopf. Keine unüblichen Geräusche, keine explodierenden Kondensatoren, alles blieb schön ruhig...

Tabelle 1 Abhängigkeit des Schwellwerts von der Batteriespannung

Batterie-Typ (Lithium)	Batterie-spannung	U_{trip}	R1
4S	12 – 16,8 V	11 V	180 Ω
5S	15 – 21 V	14,2 V	620 Ω
6S	18 – 25,2 V	16,8 V	1 kΩ
8S	24 – 33,6 V	22,9 V	1,87 kΩ
10S	30 – 42 V	28,9 V	2,74 kΩ
11S	33 – 46,2 V	31,8 V	3,16 kΩ
12S	36 – 50,4 V	34,9 V	3,6 kΩ

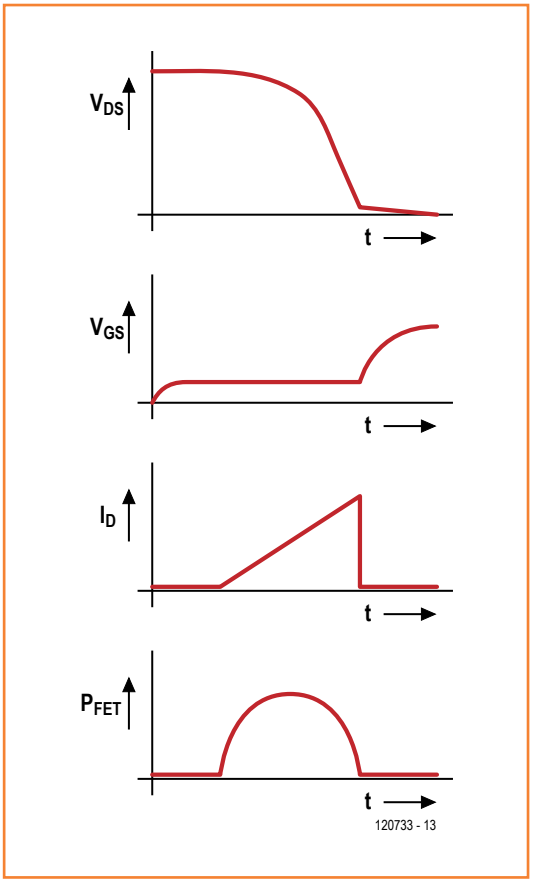


Bild 3.
Die Verlustleistung der MOSFETs hat einen umgedreht parabolischen Verlauf.

Dies war der Startschuss für Entwurf und Produktion einer einseitigen (!), gemischt mit bedrahteten und SMD-Bauteilen bestückten Platine. Das Bauteile-Layout ist in Bild 4 dargestellt. Der Wert von R1 legt die Auslösespannung fest und ist abhängig von der Batteriespannung (Tabelle 1).

In der Praxis sollte die Schaltung nicht bei Batterie-Spannungen kleiner 12 V verwendet werden. Glücklicherweise ist so etwas bei High-Power-(BLDC)-Anwendungen kaum erforderlich. Man sieht leicht ein, warum dies so ist.

Die Leiterplatte führt potenziell extrem hohe Ströme, sowohl als Spitze als auch kontinuierlich. Das bedeutet: Sie müssen alle Leiterbahnen von und zu Source und Drain der MOSFETs, über die gesamte Länge zu BATT- und BATT+ mit Stücken von 2,5 mm² Kupferdraht ausstatten, vorzugsweise zwei parallel. Die meisten dieser Klempnerarbeiten liegen im Bereich der Kühlkörper, der sie später gnädig abdeckt. Wenn Sie feststellen, dass Sie mit 1,5 mm² Kupferdraht leichter jonglieren können, in Ordnung, aber nehmen Sie drei oder sogar vier Stücke parallel. Seien Sie nicht sparsam mit dem Lötzinn und bringen Sie es in Mengen entlang der Leiterbahnen und Kupferdrähte an - es ist ein wenig wie „Sanitärarbeiten für Dummies“. Wenn Sie aus irgendeinem Grund eine Platine mit Lötstopplack erhalten, kratzen Sie ihn mit einem Cutter (oder eleganter mit einem Glasfaserstift) weg. Dann erst vor-verzinnen Sie die Leiterbahnen und installieren die Drähte.

Die Batterie- und Last-Anschlüsse K1-K2 und K3-K4 müssen hochwertig und vorzugsweise vergoldet sein. Besorgen Sie sich die besten, die sie finden können, egal ob rund oder flach, Hauptsache, man kann sie direkt auf die Platine löten. Um eine Verpolung zu verhindern, sollten Sie für die Anschlüsse je einen weiblichen und einen männlichen Verbinder verwenden. Denken Sie daran, jedes Milliohm zählt, und Sie möchten ja nicht, dass die Motorleistung oder das Drehmoment beim

Stückliste

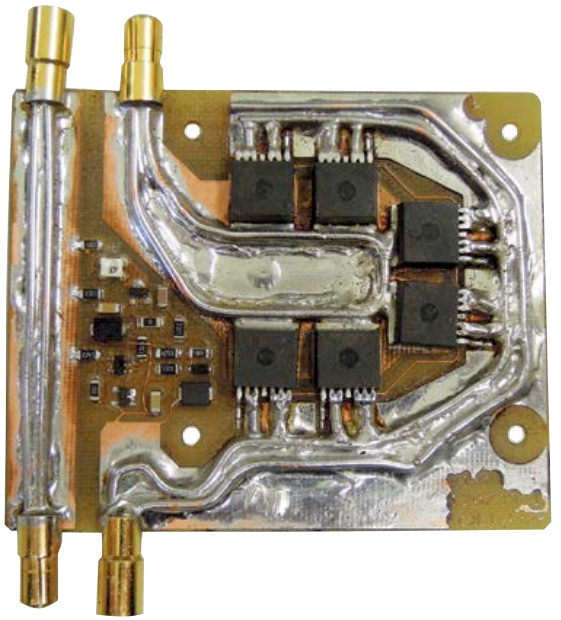
Widerstände
(Alle 0,25 W, 1%, SMD 1206)
R1 = 2k74 *
R2 = 1k5
R3,R6,R8 = 3k3
R4,R5 = 470 k
R7 = 1M8
R9 = 10 k
R10 = 0 Ω

Kondensatoren
C1,C2 = 10 µ, 10%, 25 V, X5R, 1206
C3 = 470 n, 10%, 100 V, X7R, 1206

Halbleiter
D1 = 1SMB5925B Z-Diode, SMB (Newark/Farnell 1894811)
D2 = PMEG6010CEH, Schottky-Diode, NXP, SOD-123F (Newark/Farnell 1510694)
D3 = SM6T12CA, TVS-Diode, STmicroElectronics, SMB (Newark/Farnell 9885870)
D4 = HSME-A401-P4PM1, LED, grün, Avago, PLCC-4 (Newark/Farnell 1058419)
IC1 = LT1716CS5#PBF, Komparator, Linear Technology, SOT-23-5 (Newark/Farnell 1417738)
T1,T2,T3,T4,T5,T6 = IPB017N06N3, N-MOSFET, Infineon, TO-263-7 (Newark/Farnell 1775519)
T7,T8 = 2N7002, N-MOSFET, Diodes Inc., SOT-23 (Newark/Farnell 1713823)

Außerdem
K1...K4 = Hochstrom-Verbinder, male & female, goldbeschichtet *
Kühlkörper, Aavid Thermalloy Typ 241204B92200G (60,96 mm x 57,91 mm x 11,4 mm bei Newark/Farnell 1703176)
Platine 120733-1

* nach Wunsch, siehe Text



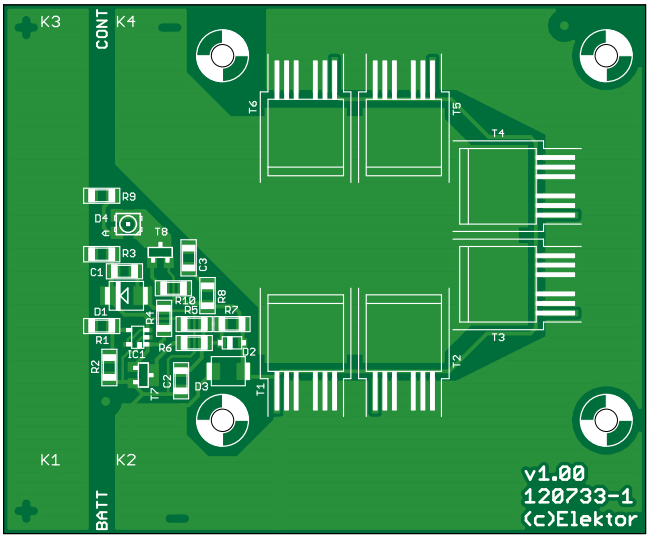


Bild 4.
Auf der kompakten Platine werden die MOSFETs vom Kühlkörper bedeckt. Das Layout ist nicht für den sofortigen Verzehr geeignet, erst müssen die hohen Strom führenden Leiterbahnen mit Kupferdrähten verstärkt werden.

Start leiden.
Die MOSFETs liegen flach auf der Platine und der Kühlkörper sitzt darüber. Ein wärmeleitendes Blättchen dazwischen nicht vergessen! Der Kühlkörper ist mit vier M3-Bolzen oder Schrauben befestigt, mit zwei M3-Muttern auf jeder Schraube, die als Abstandshalter zwischen der Plattenoberfläche und der flachen Seite des Kühlkörpers fungieren. Der gesamte Abstand beträgt ungefähr 5 mm. Die Schrauben sollten so leicht angezogen werden, dass sie das wärmeleitende Material nur ganz wenig komprimieren.
Obwohl wir hier meist über Motorsteuerungen für RC-Modelle gesprochen haben, ist die Schaltung für jede Last von 12...40 VDC geeignet, die beim Einschalten einen sehr geringen Widerstand besitzt, wie etwa große Elektrolyt-Kondensatoren und Glühlampen.

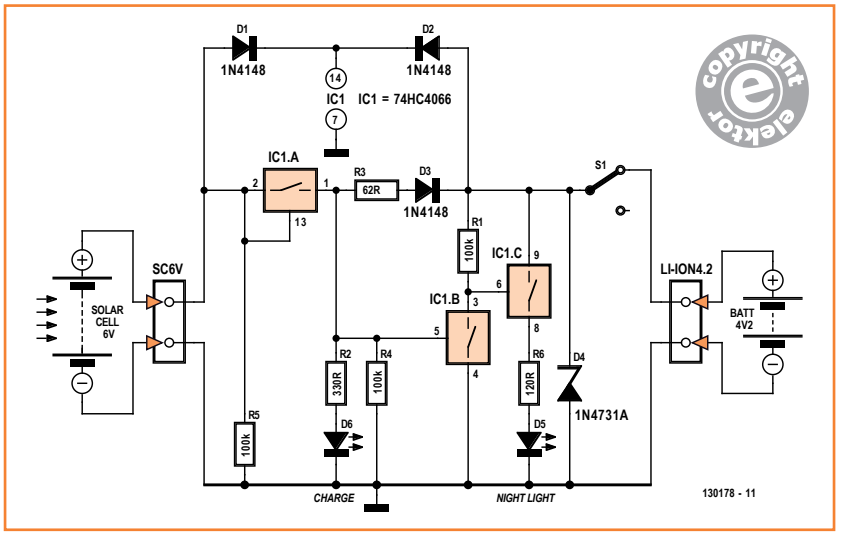
(120733)

[1] AN1542:
www.bonavolta.ch/hobby/files/MotorolaAN1542.pdf

[2] Datenblatt IPB017N06N3:
www.infineon.com/dgdl/IPB017N06N3_Rev2.2.pdf?folderId=db3a30431441fb5d01148ca9f1be0e77&fileId=db3a30431ddc9372011e264a7ab746ea

[3] Bob Pease:
http://en.wikipedia.org/wiki/Bob_Pease

Solar-Nachtlicht
mit Li-Ionen-Batterie



Von
Michael A. Shustov
(Russland)

Dieses Nachtlicht besitzt zwei Energiequellen: eine Solarzelle mit einer Spitzen-Ausgangsspannung von etwa 6 V und eine Li-Ionen-Zelle mit einer Spannung von 3,7...4,2 V. Drei der vier elektronischen Schalter eines 74HC4066N (IC1) steuern den Betrieb des Nachtlichts. IC1 erhält seine Versorgungsspannung entweder über die Diode D1 oder D2, je nachdem, welche Energiequelle die höhere Spannung liefert. Und das bestimmt dann auch die Höhe der Betriebsspannung, mit der IC1 arbeitet.

Tagsüber wird die Spannung von der Solarzelle mit einem Spitzenwert von rund 6 V geliefert. IC1a ist aufgrund des High-Pegels an seinem Steuereingang (Pin 13) geschlossen, so dass der Li-Ionen-Akku mit circa 10 mA durch Widerstand R3 und Diode D3 geladen wird. Gleichzeitig leuchtet LED D6, um anzuzeigen, dass die Batterie aufgeladen wird. Auch Schalter IC1b ist geschlossen, so dass Schalter IC1c offen und die LED D5 dunkel bleibt.

Wenn die Spannung der Solarzelle unter etwa 1/3 der minimalen Versorgungsspannung von IC1 (ungefähr 1,3 V) absinkt, öffnet der Schalter IC1a, so dass die Lade-LED verlischt. Die Spannung am Steuereingang des Schalters IC1b fällt auf Null, so dass der Schalter öffnet. Folglich wechselt IC1c seinen Zustand und verbindet die „Night Light“-LED über den Widerstand R6 mit der Batterie. Der LED-Strom ist auf 10...13 mA eingestellt. Die Farbe der LED können Sie ganz nach ihrem Gusto wählen. Beim Prototyp wurde eine grüne LED verwendet.

Der Ladestrom sowie die Helligkeit der LEDs kann durch Anpassung von R3, R2 und R6 eingestellt werden, allerdings darf der maximale Strom durch die 4066er-Schalter von 20 mA nicht überschritten werden. Die Zenerdiode D4 verhindert eine Überladung der Batterie. Wenn die Schaltung nicht in Betrieb ist, sorgt der geöffnete Schalter S1 dafür, dass sich die Batterie nicht entlädt.

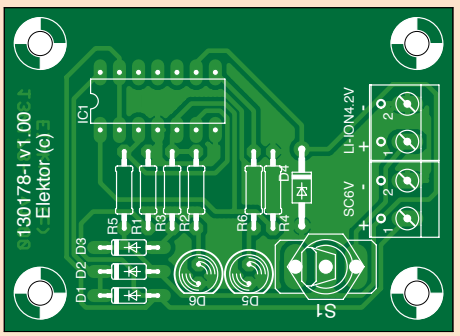
(130178)

Stückliste

Widerstände
R1,R4,R5 = 100 k, 1%, 0,25 W
R2 = 330 Ω, 1%, 0,25 W
R3 = 62 Ω, 1%, 0,25 W
R6 = 120 Ω, 1%, 0,25 W

Halbleiter
D1,D2,D3 = 1N4148
D4 = 1N4731A Zenerdiode (4,3 V)
D5 = LED, 5 mm, Farbe nach Wahl
D6 = LED, rot, 5 mm
IC1 = 74HC4066

Außerdem
S1 = Umschalter, Newark/Farnell 1310879
Platine 130178-1



Verbinder (SC6V und LI-ION4,2) = Platinenanschlussklemme, Rastermaß 5 mm

elektorLive! 2013

Seminar- & Ausstellungstag für die Elektronik-Entwicklung und -Anwendung

Am 12. Oktober 2013 findet im Congress Park Hanau die zweite Elektor*Live!*-Veranstaltung statt.

An diesem Tag halten kompetente Elektor-Autoren, -Entwickler und Experten aus der Elektronik-Branche verschiedene Seminare zu populären Elektronik-Themen ab. Geplant sind 3 Seminar-Runden mit bis zu 4 parallel stattfindenden Seminaren.

Begleitet werden die Seminare von einer Ausstellung renommierter Elektronik-Unternehmen, die ihre Produkte und Innovationen vor Ort präsentieren werden. Während der Pausen haben Sie die Möglichkeit, mit diesen Firmen in Kontakt zu treten.

Freuen Sie sich jetzt schon auf interessante Seminare und Workshops mit hochkarätigen Referenten!

Geplante Seminare (Änderungen vorbehalten)**:**
- Elektronik-Apps mit Android
- Entwickeln mit dem Elektor-FPGA-Board
- Embedded Linux
- Arduino – nicht nur für Einsteiger
- Röhren (mit Menno van der Veen)
- Tipps & Tricks zu AVR-Controllern
- LabVIEW
- Design mit EAGLE

Datum:
Samstag, 12. Oktober 2013

Zeit:
09:00 Uhr – 17:00 Uhr

Ort:
Congress Park Hanau

Eintritt:
29,50 € für Elektor-Mitglieder
49,50 € für Nicht-Mitglieder
19,50 € für Schüler/Studenten
Der Eintritt berechtigt zur Teilnahme an 2 Seminaren.

Tagesablauf:
09:00 Uhr – 17:00 Uhr: Ausstellung
09:30 Uhr – 11:30 Uhr: 1. Seminar-Runde
12:30 Uhr – 14:00 Uhr: 2. Seminar-Runde
15:00 Uhr – 16:30 Uhr: 3. Seminar-Runde

Weitere Infos & Anmeldung unter
www.elektor-live.de

Universelles Präzisions-Messinterface

Alles für die Auflösung!

Von
Michel Defrance (F)

Ein A/D-Wandler gehört heute zur internen Peripherie fast jedes gängigen Mikrocontrollers. Doch was tun, wenn die Auflösung für den geplanten Zweck nicht ausreicht? Grübeln Sie nicht länger, die Antwort kommt hier!

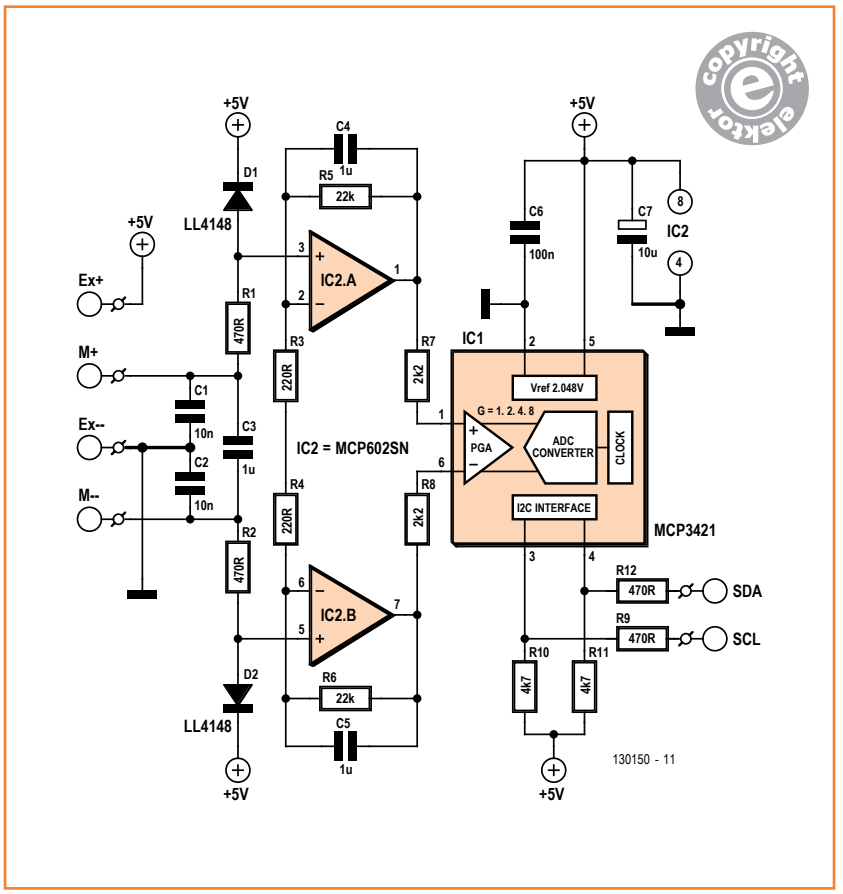


Bild 1.
Schaltung des Messinterfaces.

Tabelle 1. Eingangsspannung, abhängig von der Verstärkung.

Verstärkung MCP3421	Eingangsspannung
1	−20 mV ... +20 mV
2	−10 mV ... +10 mV
4	−5 mV ... +5 mV
8	−2,5 mV ... +2,5 mV

Nicht selten müssen in der Elektronik sehr niedrige Spannungen gemessen werden, beispielsweise die Signale von Temperatur- oder Drucksensoren. Auch die Spannungen, die sogenannte Wheatstonesche Brücken abgeben, gehören in diese Kategorie. Meistens liegen solche Spannungen in der Größenordnung von nur wenigen Millivolt. Für die Verarbeitung der Messwerte werden üblicherweise Mikrocontroller eingesetzt, denn sie sind vielseitig und flexibel, und die Programmierung ist vergleichsweise unkompliziert. Leider liegen die Auflösungen der A/D-Wandler, die in Mikrocontrollern integriert sind, mit 8, 10 oder 12 bit allenfalls im Mittelfeld. Außerdem müssen niedrige Messspannungen mit verstärkenden Vorstufen an die Eingangsbereiche der A/D-Wandler angepasst werden.

Ich hatte mir vorgenommen, ein Messinterface für A/D-Wandler von Mikrocontrollern zu entwickeln, das mit gebräuchlichen, kostengünstigen Bauelementen realisierbar ist und zu möglichst vielen Mikrocontrollern passt. In einem Katalog von Microchip fand ich zwei Chips, die mir für mein Vorhaben geeignet erschienen:

- Präzisions-Opamp MCP602,
- A/D-Wandler MCP3421, Auflösung 18 bit, mit Referenzspannungsquelle 2,048 V, I2C-Schnittstelle und programmierbarem Vorverstärker.

Instrumentierungsverstärker

Bevor niedrige Spannungen digitalisiert werden, ist häufig ein Instrumentierungsverstärker im Einsatz, der aus drei Opamps besteht. Wie Bild 1 zeigt, sind hier zwei Opamps des Typs MCP602 (IC2A und IC2B) als Differenzverstärker geschaltet, sie steuern den internen Verstärker des A/D-Wandlers MCP3421 (IC1) an. Die Spannung am

Ausgang des Differenzverstärkers hängt von der Spannungsdifferenz zwischen den Messeingängen M+ und M- ab, für sie gilt:

$G1_{+} = 1 + R5/R3$
$G1_{-} = 1 + R6/R4$

Da die Spannungsverstärkungen für die Eingänge M+ und M- identisch sein sollen, werden R5 = R6 und R3 = R4 gewählt. Aus den in der Schaltung angegebenen Werten folgt:

G1 = 1 + 100 = 101

Die Widerstände R3...R6 müssen eng toleriert sein (1 % oder besser), anderenfalls würden in der Eingangsstufe unzulässige Asymmetrien entstehen. Bei symmetrischem Betrieb des MCP3421 darf die Eingangsspannung des A/D-Wandlers im Bereich –2,048...+2,048 V liegen. Die Verstärkung G2 des internen Verstärkers lässt sich über die Software einstellen, so dass der Bereich der Eingangsspannung programmierbar ist. Die Gesamtverstärkung G = G1 · G2 ist zwischen G = 101 bei G2 = 1 und G = 808 bei G2 = 8 variierbar. Aus Tabelle 1 gehen die Eingangsspannungsbereiche abhängig von der Verstärkung des MCP3421 hervor. Abweichende Bereiche lassen sich durch Anpassen der Werte von R3...R6 realisieren.

Stromversorgung

Die Betriebsspannung des Messinterfaces beträgt 5 V. Den Spannungen üblicher Netzteile ist fast immer Rauschen überlagert, das sich beim Verarbeiten sehr kleiner Signale unliebsam bemerkbar macht. Auch dieses Messinterface ist dagegen empfindlich. Deshalb soll die Stromversorgung absolut stabil arbeiten, die Spannung soll frei von Überlagerungen sein. Bei der Auflösung 18 bit wirken sich schon kleinste Störsignale nachteilig auf die Messergebnisse aus. Ferner soll die Stromversorgung nach dem Einschalten „weich“ hochfahren, damit Drifterscheinungen infolge nicht völlig stabiler Eigenschaften der Bauelemente minimiert werden. Eine Einschaltverzögerung, realisiert durch Software, wäre bei dieser Konfiguration nur eine halbe Lösung. Besser geeignet ist eine Stromversorgung mit dem Low-Drop-Spannungsregler MIC 2941 von Micrel, wie sie unter [1] beschrieben ist. An anderer Stelle in dieser Elektor-Ausgabe finden Sie eine entsprechende Schaltung unter dem Titel „Slow Start Stabilisator“.

Eigenschaften

- Konvertierung mit 18 bit
- I2C-Schnittstelle
- Verstärkung über Software einstellbar

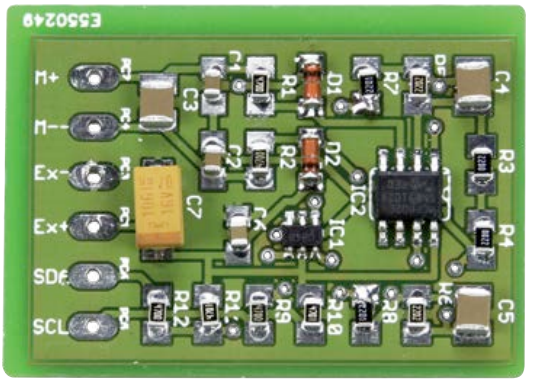


Bild 2.
Die Platine in der 3D-Simulation.

Aufbau und Einsatz

Mit etwas Fingerspitzengefühl ist die Montage der SMDs auf der 35 · 25 mm großen Platine (Bild 2) nicht schwierig. Für andere Platinenlayouts gelten die Empfehlungen, die im Datenblatt des MCP3421 stehen. Außerdem sind dort wichtige ergänzende Informationen zu diesem A/D-Wandler zu finden. Hier sei nur darauf hingewiesen, dass die I2C-Adresse des MCP3421 vom einzelnen Exemplar abhängt. Auch darüber gibt das Datenblatt genaue Auskunft.
Als Vorbild für den Einsatz des Messinterfaces habe ich eine Schaltung entworfen, in der ein Mikrocontroller PIC18F452 (oder alternativ PIC16F876A) über den MCP3421 und den I2C-Bus eine Spannung misst. Diese Schaltung gehört ebenfalls zu den Beiträgen in der vorliegenden Ausgabe. Messgröße ist die Spannung im Zweig einer Wheatstoneschen Brücke. Das Mikrocontroller-Programm, geschrieben in PICBASIC, rechnet den vom MCP3421 übergebenen Spannungswert in den Wert eines Drucks um und gibt beide Werte auf einem LC-Display aus. Das Programm lässt sich unschwer auf andere Plattformen portieren, Arduino ist ein Beispiel.
Das Präzisions-Messinterface ist in viele Messsysteme integrierbar. Eventuelle Klagen über die Eigenschaften der A/D-Wandler in Mikrocontrollern haben nun keine Berechtigung mehr.

(130150)gd

Weblink

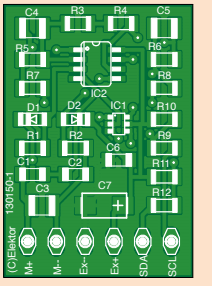
[1] www.elektor-labs.com/node/3053

Stückliste

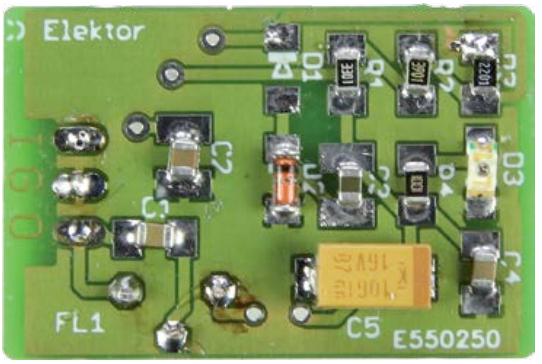
Widerstände:
R1,R2,R9,R12 = 470 Ω 5 %
R3,R4 = 220 Ω 5 %
R5,R6 = 22 k 1 %
R7,R8 = 2,2 k 5 %
R10,R11 = 4k7 5 %

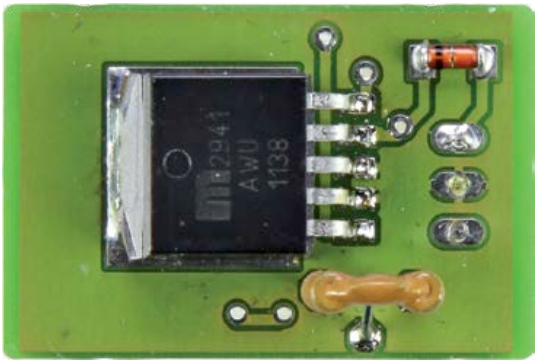
Kondensatoren:
C1,C2 = 10 n/50 V 10 %
C3,C4,C5 = 1 µ/50 V 10 %
C6 = 100 n/50 V 10 %
C7 = 10 µ/16 V 10 %, Elko

Halbleiter:
D1,D2 = LL4148
IC1 = MCP3421A1T
IC2 = MCP602SN



Slow Start Stabilisator



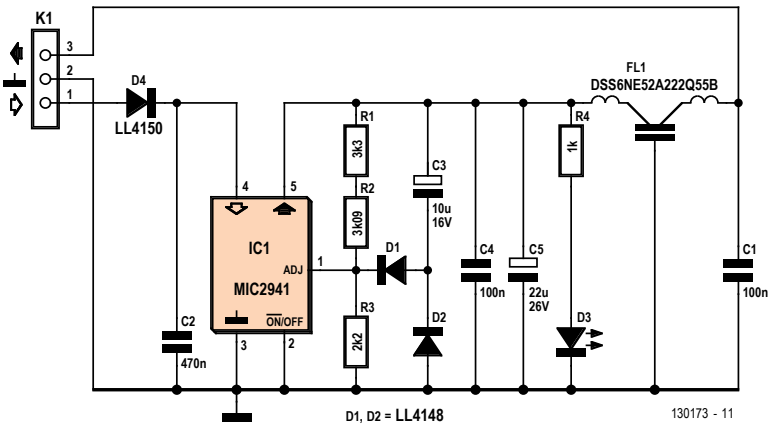


Von
Michel Defrance (F)

Für den Einsatz in den Stromversorgungen von hochpräzisen Messschaltungen und A/D-Wandlern ist der standardisierte Spannungsregler 7805 normalerweise nicht geeignet. Der Ausgangsspannung ist ein Rauschteppich überlagert, und das Einschaltverhalten ist unzureichend definiert. Ein Beispiel für ein empfindliches Messsystem ist das in dieser Ausgabe beschriebene Präzisions-Messinterface, das mit dem A/D-Wandler MCP3421 eine Auflösung von 18 bit erreicht. Damit diese hohe Auflösung tatsächlich genutzt werden kann, muss die Betriebsspannung möglichst absolut stabil und rauschfrei sein. Außerdem darf die stabilisierte Spannung beim Anlegen der Eingangsspannung nicht sprunghaft ansteigen, sie muss ihren Sollwert langsam erreichen. Den empfindlichen Messinterface-Komponenten muss Gelegenheit gegeben werden, ohne unerwünschte Einschalteffekte hochzufahren. Das zweite Problem ließe sich eigentlich auch mit einem Software-Timer lösen, doch die Wirkung würde sich dann auf bestimmte Komponenten beschränken.

Unser Slow Start Stabilisator erfüllt alle genannten Forderungen, er kann den 7805 in jeder Hinsicht ersetzen. Die Miniplatine hat fast gleiche Abmessungen, und auch die Anschlüsse sind mit dem 7805 identisch. Die Schaltung ist ausschließlich mit SMDs aufgebaut.

Der Spannungsregler ist der MIC2941 von Micrel, ein Low-dropout-Typ, dessen Ausgangsspannung ebenso wie beim bekannten LM317 mit einem Spannungsteiler festgelegt werden kann. Die Schaltung ist schnell erklärt: Für die Höhe der Ausgangsspannung ist das Verhältnis (R1+R2) / R3 maßgebend, mit den angegebenen Werten beträgt sie nahezu exakt 5 V. Die Diode D4 verhindert Schäden infolge einer Verpolung der Eingangsspannung, ferner sind diverse Kondensatoren vorhanden, die für die Entkopplung und Entstörung sorgen. Unterstützt werden die Bemühungen um Störfreiheit von dem dreibei-

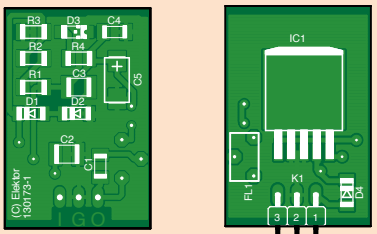


Stückliste

Widerstände (SMD 1206):
R1 = 3k3
R2 = 3k09
R3 = 2k2
R4 = 1k

Kondensatoren:
C1,C4 = 100 n (SMD 1206)
C2 = 470 n (SMD 1210)
C3 = 10 µ/16 V (SMD 1210)
C5 = 22 µ/10 V (SMD 2312)

Halbleiter:
D1,D2 = LL4148
D3 = LED Low-Current, SMD 1206
D4 = LL4150



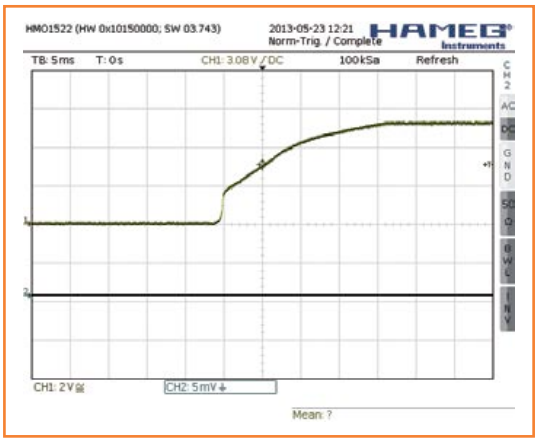
IC1 = MIC2941AWU TR (TO-263)

Außerdem:
EMI Suppression Filter
DSS6NE52A222Q55B (Murata)
Platine 130173-1, siehe [1]

nigen EMI-Filter FL1 am Ausgang. Der DSS6NE-52A222Q55B ist ein Typ von Murata, welcher aus zwei Induktivitäten und einem Kondensator zwischen dem Verbindungspunkt und Masse besteht. Den verzögerten Anstieg der Ausgangsspannung bewirkt Kondensator C3. Beim Anlegen der Eingangsspannung sorgt C3 zunächst dafür, dass am Verbindungspunkt R2, R3 eine Spannung nahe 0 V liegt. Dann wird C3 über R3 in ungefähr 20 ms geladen, so dass die Ausgangsspannung, wie die Grafik zeigt, langsam ansteigt. Die Dioden D1 und D2 verhindern, dass beim Abschalten negative Spannung zum Eingang des Spannungsreglers gelangt. Kondensator C3 kann sich über Diode D2 entladen.

Dieser 7805-Ersatz liefert gekühlt Ausgangsströme von mindestens 1 A. Ohne Kühlung können bei 12 V Eingangsspannung nur einige 10 mA entnommen werden. Das Platinenlayout kann auf der Projektseite [1] heruntergeladen werden.

(130173)gd



Weblink

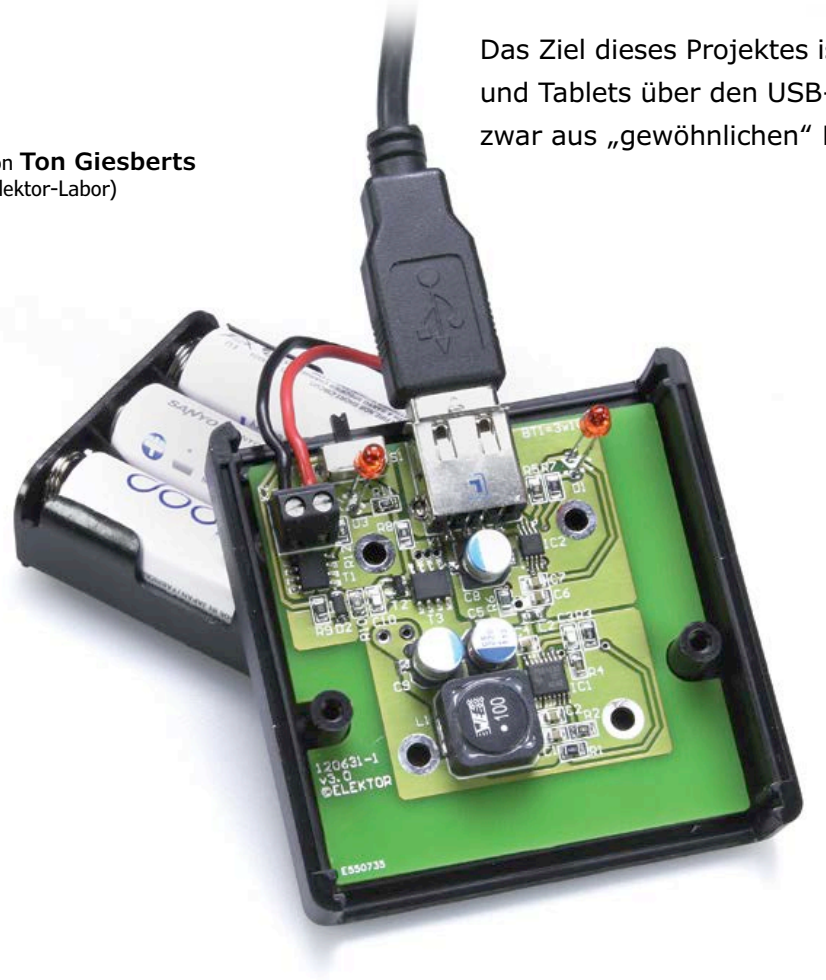
[1] www.elektor.de/130173

Charge-a-Phone

NiMH am USB-Power-Pack

Das Ziel dieses Projektes ist es, portable Geräte wie Smartphones und Tablets über den USB-Anschluss effektiv aufzuladen – und zwar aus „gewöhnlichen“ NiMH-Akkus.

Von **Ton Giesberts**
(Elektor-Labor)



Was auf den ersten Blick als eine gelinde gesagt merkwürdige Idee erscheint, entpuppt sich bei näherem Hinsehen als cleverer Schachzug: Es gibt nämlich viel mehr und einfachere Ladegeräte für NiMH-Akkus als für Lithium-Ionen- oder Lithium-Polymer-Batterien. Da wir die Batterien aber nicht in unser Gerät integriert haben, besteht zumindest die Möglichkeit, Li-Ion- oder LiPo-Akkus zu verwenden. Glücklicherweise ist ihre Spannung von 3,6...3,7 V fast genauso hoch wie die von drei in Reihe geschalteten NiMH-Batterien. Wegen des separaten Batteriehalters ist man auch in der Lage, entladene Batterien durch neue zu ersetzen, ohne sie zuerst aufladen oder das Gehäuse öffnen zu müssen. Das ist ein großes Plus, wenn Ihr Telefon, Tablet oder E-Gizmo nach Aufladung schreit und Sie sich irgendwo in der Walachei befinden.

Wie viele Batterien brauchen wir?

Die Schaltung soll 5 V produzieren und bis zu 1 A Ausgangsstrom liefern können. Vier frisch aufgeladene NiMH-Akkus in Reihe weisen aber eine Spannung deutlich über 5 V auf, so dass es ratsam erscheint, die Zahl der Batterien auf drei zu reduzieren. Beim USB sind zwar 5 V nur als „nominal“ anzusehen, wobei der tatsächliche Bereich 4,35 V bis 5,40 V betragen kann. Obwohl dies wieder für vier Batterien spricht, wollen wir doch eine 5,00-V-Versorgung entwerfen, die präzise ist, wenn auch nur, weil einige Entwickler die USB-Spannung als eine Referenz behandeln. Also: drei!

Boost Converter TPS61030

Drei Batterien erlauben zwar einen kleineren Batteriehalter, machen aber auch einen Spannungswandler erforderlich. Es gibt einen ausgezeichneten Synchron-Boost-Wandler von Texas Instruments, den TPS61030. Er besitzt einen internen 4-A-Schalter und einen Wirkungsgrad von 96 % (abhängig von Eingangsspannung und Ausgangsstrom). Der Wandler verfügt auch über einen Low-Battery-Komparator, der optional eingesetzt werden kann, um eine Tiefentladung der Akkus zu verhindern. Eine zusätzliche Unterspannungs-Lockout-Funktion, die bei 1,6 V wirksam wird, verhindert Fehlfunktionen. Die interne Referenzspannung von 0,5 V macht es leicht, den Spannungsteiler für die korrekte Ausgangsspannung zu berechnen. Hier sind es 1,8 MΩ für R3 und 200 kΩ für R4. Laut Datenblatt ist ein zusätzlicher Kondensator parallel zu R3 für die Stabilität erforderlich, wenn R4 deutlich niedriger ist als 200 kΩ. Um sicher zu gehen, haben wir einen kleinen 10-pF-Kondensator eingesetzt.
Der Widerstand R2 sollte niedrig genug für den Eingangsstrom des Komparators von ungefähr 10 nA sein. Ein Wert von 500 kΩ wird empfohlen. Der Level des Komparators liegt bei 500 mV mit einer Hysterese von 10 mV. Ein Schwellwert von 1,1 V

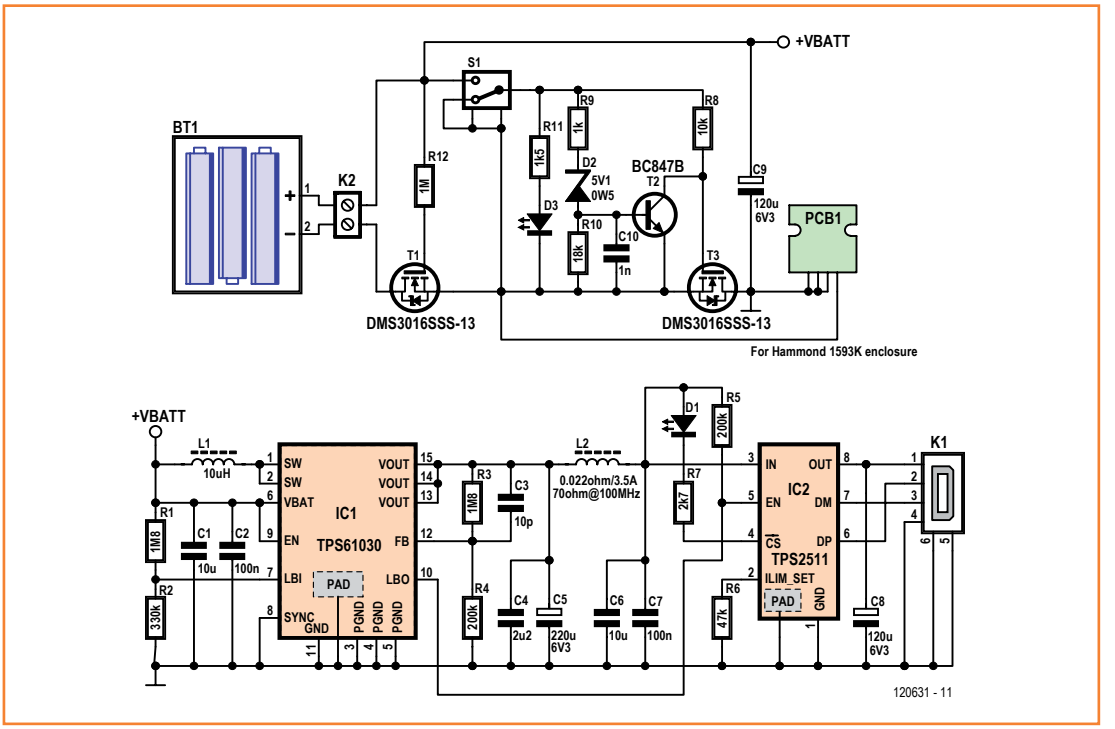
stellt eine komplett leere Batterie dar. Mit 1,8 MΩ für R1 und 330 kΩ für R2 liegt der theoretische Grenzwert bei 3,23 V. Wenn die Batteriespannung unter diesen Wert fällt, geht der Komparatorausgang LBO auf Low. Dieser Ausgang wird verwendet, um den Strom zur USB-Buchse zu unterbrechen. Die Entkopplung der Eingangsspannung mit C1 und C2 erfolgt in Übereinstimmung mit dem Datenblatt. Die Entkopplung der Ausgangsspannung hängt von der maximalen Ausgangswelligkeit ab. Ein paar Millivolt wären ideal, aber die ESR der Kondensatoren und das Platinenlayout verursachen in der Praxis einen höheren Wert. Theoretisch sollte mit einem Pufferkondensator von 220 µF die Welligkeit um 1 mV liegen, in der Praxis wurden aber etwa 60 mV über C5 gemessen (Eingangsspannung 3,50 V und 1 A Last). C5 hat laut Datenblatt einen ESR von 20 mΩ bei 100 kHz. Die tatsächliche Schaltfrequenz von 600 kHz ist viel höher und die höheren Schaltströme sorgen auch für eine höhere Brummspannung. Zur Unterdrückung der Schaltgeräusche wurde eine Ferritperle (L2) in Reihe mit dem Ausgangsstrom aufgenommen. Mit dem zusätzlichen abschließenden Ausgangskondensator C8 sollte die Welligkeit weiter deutlich reduziert werden.

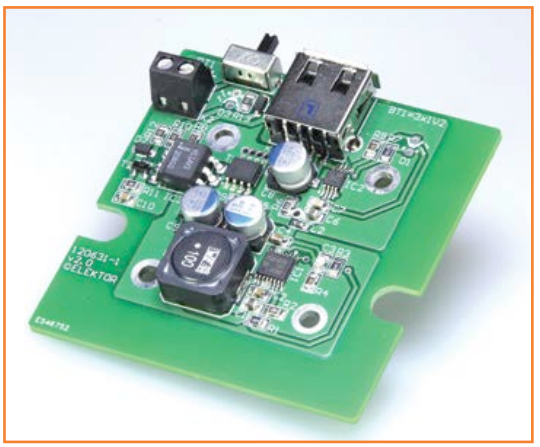
Bei der Berechnung der Induktivität wurde eine Abweichung von 10 % des maximalen gemittelten Spulenstroms berücksichtigt. Bei 3,20 V liegt der mittlere Spulenstrom nahe 2 A. Mit der Formel im Datenblatt (SLUS534E) ergibt dies eine Induktivität von etwa 10 µH.

Der Sync-Anschluss erlaubt es, den Wandler in verschiedenen Modi zu betreiben. Wir haben uns für Power Save entschieden (Sync auf Masse), was die Effizienz bei leichter Belastung verbessert. Der Wandler arbeitet dann diskontinuierlich und nur, wenn die Ausgangsspannung unter einen festge-

Messwerte und technische Daten

Eingangsspannungsbereich	3,3...4,1 V
Maximaler Eingangsstrom	1,7 A (V_{in} = 3,33 V)
Ausgangsspannung	4,93 V (ohne Last)
	4,92 V (0,3 A Last)
	4,82 V (1 A Last)
Low-Battery-Schwelle	3,25 V
Überspannungsschutz	4,30 V
Wirkungsgrad	95% (3,52 V_{in}, 0,3 A out)
	90% (3,52 V_{in}; 1 A out)
Stromaufnahme (ohne Last)	4,5 mA (Vin = 3,6 V)
Power-LED leuchtet bei	1,6 V
Verluste bei 1 A Ausgangsstrom gemessen:	
Über T1 und T3 (jeweils)	23 mV (1,7 A Eingangsstrom)
Von L2 zum USB-Anschluss auf der Platine	85,3 mV
Über IC2	80 mV
USB-Anschluss (jeweils)	13 mV





legten Schwellwert sinkt. Diesen Vorteil bezahlt man mit einem leicht höheren Brumm am Ausgang. Ohne Last wurde ein 80-mV-Sägezahn mit einer 150-ms-Periode gemessen. Mit steigender Belastung wurde es aber schnell besser.

TPS2511

Ein spezielles IC, ein TPS2511, übernimmt die Steuerung des Ausgangs. Texas Instruments nennt es einen USB Dedicated Charging Port Controller and Current Limiting Power Switch, trotz dieser sperrigen Bezeichnung haben wir es aus gutem Grund verwendet. Es reicht nämlich meist nicht, einfach 5 V an den USB-Anschluss zu legen und dann munter draufloszuladen. Es ist zwar schön, dass die Hersteller von Handys und anderen elektronischen Spielzeugen ihre Geräte zunehmend mit USB-Ladeanschlüssen ausstatten, aber beliebige Ladegeräte sind eben nicht mit beliebigen Plattformen kompatibel (außer man legt sich das sündhaft teure Original-Ladegerät des gleichen Herstellers zu). Zum Beispiel erwarten einige Geräte bestimmte Spannungen auf den Datenleitungen oder einfach nur eine Verbindung (Widerstand) zwischen den Datenleitungen, um das Ladegerät zu erkennen (Dedicated Charger Port oder DCP). Der TPS2511 unterstützt drei der gängigsten Protokolle:

- USB Battery Charging Specification, Revision 1.2 (BC1.2);
- Chinese Telecommunications Industry Standard YD/T 1591-2009;
- Divider-Modus.

Eine ausführliche Beschreibung aller Fähigkeiten des TPS2511 findet man im Datenblatt von TI (SLUSB18).

Der TPS61030 kann 2 A bei 3,3 V Batteriespannung liefern und der TPS2511 mit diesem Strom auch fertig werden. Aber bei 1 A Ausgangsstrom und 3,33 V Eingangsspannung zieht der Wandler bereits 1,7 A aus den Batterien. Bei 2 A Ausgangsstrom wird der Eingangsstrom wegen der höheren Verluste mehr als verdoppelt. Auch sinkt die Kapazität der Batterie bei höheren Ausgangsströmen. Deshalb arbeitet der TPS2511 wie ein 5-W-Ladegerät. Sein DP-Anschluss ist mit der Leitung D-, der DM-Anschluss mit der Leitung D+ des USB verbunden. Die Strombegrenzung ist geringfügig höher als nötig eingestellt (R6 = 47 kΩ), damit die Ausgangsspannung vom TPS2511 nicht vorzeitig begrenzt wird.

Der Anschluss Current Sensing Report wird hier anders als üblich verwendet. Da es bei maximal 1 A Ausgangsstrom nicht notwendig ist, die Rückkopplung des Wandlers zu verändern, um den Spannungsverlust zu kompensieren, wird der Anschluss verwendet, um LED D1 zu treiben. Wenn D1 leuchtet, bedeutet dies, dass mehr als die Hälfte des maximalen Ausgangsstroms geliefert wird. Der LED-Strom liegt bei etwas mehr als 1 mA. Wie bereits erwähnt, ist der Low-bat-Komparatorausgang an Enable (EN) des TPS2511 angeschlossen. Auf diese Weise wird die Ausgangsspannung in dem Fall, dass die Batterien entladen sind, gekappt. R5 ist erforderlich, da der nicht-aktive Ausgang des Komparators hochimpedant ist.

Polarität und Überspannungsschutz

Die Verbindung von der Platine zum Akkupack läuft über eine Platinenklemme mit einem Rastermaß von 0,15‘‘ (3,81 mm). Da es prinzipiell möglich ist, die Batterien falsch herum anzuschließen, haben wir den kleinen N-Kanal-MOSFET T1 (in umgedrehter Polarität) als Verpolschutz eingesetzt. Sind die Batterien richtig angeschlossen, befindet sich die Body-Diode in Durchlassrichtung und der MOSFET ist voll eingeschaltet (das Gate ist positiv in Bezug auf Source über R12). So sind geringste (fast keine) Verluste zu befürchten. Es ist auch kein Problem, dass der Strom von Source nach Drain fließt. Sind die Batterien jedoch falsch angeschlossen, ist das Gate negativ, bleibt der MOSFET ausgeschaltet und die Body-Diode blockiert. Die maximal zulässige Gate-Spannung des eingesetzten MOSFETs beträgt 12 V, was auch die maximale Spannung bestimmt, welche die Schaltung überlebt. Bei 1,7 A Strom über die MOSFETs fallen lächerliche 23 mV ab (gemessen am Prototyp).

Um einen „dicken“ und damit teuren An/Aus-Schalter zu vermeiden, ist der Überspannungsschutz mit einem kleineren und deshalb deutlich billigeren Schalter kombiniert. Der Überspannungsschutz ist einfach ausgeführt. Wenn die angelegte Spannung zu hoch ist, schaltet eine Z-Diode (D2) den NPN-Transistor (T2), der wiederum die Gate-Spannung des MOSFETs T3 kurzschließt. Die 5,1-V-Z-Diode leitet bereits unterhalb der nominellen Zenerspannung. Bei einer Batteriespannung von 3,60 V liegt der Strom durch D2 bei circa 12 µA, bei 4,25 V schon bei mehr als 30 µA. Dies kann leicht durch den Spannungsabfall über R9 bestätigt werden. Der Widerstand verhindert das lawinenartige Ansteigen des Stroms bei einer Eingangsspannung über etwa 5,70 V. Setzt der Überspannungsschutz wegen möglicher Toleranzen der Zenerdiode zu früh ein, können Sie R10 anpassen: Ein niedrigerer Wert ergibt eine höhere Schwelle.

Der Überspannungsschutz ist überhaupt nur erforderlich, wenn ein Steckernetzteil (eingestellt auf hoffentlich weniger als 12 V) oder eine 9 V-Batterie angeschlossen ist. Der TPS61030 könnte 7,00 V (absolutes Maximum, 5,50 V empfohlen) standhalten. Das Problem mit dem Boost-Wandler ist, dass die Ausgangsspannung steigt, wenn die Eingangsspannung höher wird als die geregelte Ausgangsspannung (hier 5,00 V nominal).

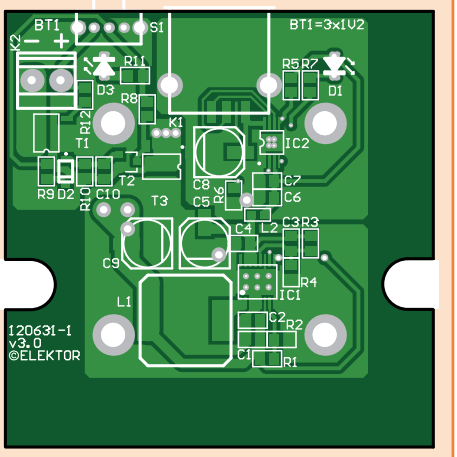
Konstruktion

Die Platine ist speziell für ein bestimmtes Hammond-Gehäuse (siehe Stückliste) ausgelegt. Es ist preiswert und einfach an unsere Anwendung anzupassen. Die Platine ist mit vier kurzen Blechschrauben und die obere und untere Gehäuseschale mit zwei längeren Schrauben fixiert. Die Vorder- und Rückseite des Gehäuses sind separate Platten. In eine Platte werden zunächst nur drei Löcher gebohrt. Die Löcher für USB-Anschluss und Schalter sollten an den Bauteilen auf der Platine ausgerichtet werden. Das gleiche gilt für die beiden Löcher für die LEDs in der oberen Schale. Die genaue Platzierung der Bohrung für die beiden Kabel zum externen Batteriehalter ist nicht so kritisch. Es gibt ausreichend Platz auf der anderen Platte, selbst der Einbau einer Niederspannungsbuchse ist möglich. Vermeiden Sie aber alle überflüssigen Übergangswiderstände, um die Effektivität der Schaltung zu erhalten.

Die Löcher für die Platinenbefestigung werden auch verwendet, um die „Power“-Flächen zu verbinden. Das Loch neben Verbinder K2 ist nicht mit Masse, sondern mit dem Netz zwischen T1 und T3 verbunden. Der Anschluss dieses Netzes an Masse zerstört die Schaltung zwar nicht, aber verhindert einfach ihre Funktion. Die anderen drei Löcher sind mit Masse verbunden, das Loch neben IC2 ist der Masse-Ausgang. Noch eine Kleinigkeit: Berühren Sie nicht die Verbindung R3/C3/R4, wenn die Schaltung in Betrieb ist. Dieser Punkt ist hochimpedant und jeder Eingriff hier kann IC1 zerstören.

(120631)

Stückliste



Widerstände

(0805, 125 mW)
R1,R3 = 1M8, 1%
R2 = 330 k, 1%
R4,R5 = 200 k, 1%
R6 = 47 k, 1%
R7 = 2k7, 5%
R8 = 10 k, 5%
R9 = 1 k, 5%
R10 = 18 k, 5%
R11 = 1k5, 5%
R12 = 1 M, 5%

Kondensatoren

C1,C6 = 10 µ, 10 V, 20%, X5R, 0805 (Taiyo Yuden LMK212 BJ106MG-T)
C2,C7 = 100 n, 50 V, 10%, X7R, 0805
C3 = 10 p, 50 V, ±0p5, C0G/NP0, 0805
C4 = 2µ2, 6,3 V, 10%, X5R, 0805
C5 = 220 µ, 6,3 V, 20%, SMD, Ir = 2,8 A (Nichicon PCS0J221MCL1GS)
C8,C9 = 120 µ, 6,3 V, 20%, SMD, Ir = 2,8 A (Nichicon PCS0J121MCL9GS)
C10 = 1 n, 50 V, 10%, X7R, 0805

Induktivitäten

L1 = 10 µH, 5 A, 0,025 Ω, 20% (Würth Electronics 74477110)
L2 = Ferritperle, 70 Ω@100 MHz, 3,5 A, 0,022 Ω, 0603 (Murata BLM18KG700TN1D)

Halbleiter

D1,D3 = LED, rot, 3 mm, bedrahtet (low current)
D2 = 5V1 Z-Diode, 0W5 (SOD123), Diodes Inc. MMSZ5231B-7-F
IC1 = TPS61030PWPG4 (Texas Instruments)
IC2 = TPS2511DGN (Texas Instruments)
T1,T3 = DMS3016SSS-13 (SO8)
T2 = BC847B

Außerdem

K1 = USB-Verbinder Typ A, Buchse für Platinenmontage, SMD
K2 = 2-polige vertikale Platinenanschlussklemme, Raster 0,15'' (3,81 mm) (Phoenix Contact MKDS 1/2-3.81)
S1 = Schiebeschalter, SPDT, rechtwinklig, 100 mA (C&K Components OS102011MA1QN1)
Gehäuse 66,22x67,22x28,00 mm³ (Hammond Manufacturing 1593KBK)
PC-Platinenschrauben (4 x 1/4“ Blechschraube, 6,4 mm, Hammond 1593ATS50)
BT1 = Halter für 3 AA-Zellen, Schnappkontakte (Keystone 2475) und Batterieclip (BUD Industries HH3449)
3 NiMH-Zellen AA
Platine 120631-1 v3.0

Von BASIC nach Python (3)
Kommunikation mit dem ElektorBus

Von
Jean-Claude Feltes
(LU)

In den ersten beiden Teilen ging es um die Unterschiede zwischen BASIC und Python sowie um die Erstellung von Diagrammen, mathematische Feinheiten wie die Fourier-Synthese und grafische Benutzerschnittstellen. Im dritten Teil wird die Anbindung an den ElektorBus beschrieben. Natürlich nicht nur grau und theoretisch, sondern praktisch und in Farbe.

Die Kombination von Elektronik + Python + PC ist sehr gut dazu geeignet, Daten von selbst gebauter oder kommerziell gefertigter externer Hardware zum PC zu übertragen und dort zu visualisieren. Und auch der umgekehrte Datenfluss macht Sinn, denn so lässt sich externe Hardware prima steuern. Zur Kommunikation mit externer Hardware kann man klassische serielle Schnittstellen verwenden oder, wenn es etwas mehr sein soll, auf einen Bus setzen. Und am ElektorBus kann man sehr gut zeigen, wie man so ein System in den Griff bekommt. Der ElektorBus wurde in einer elfteiligen Artikelserie [1] zwischen Januar 2011 und Januar 2012 in Elektor beschrieben. Als Hardware dient hier die in Teil 6 [2] beschriebene, mit Mikrocontroller, LEDs und Tastern bestückte Platine eines Experimentalknotens, die über den dort ebenfalls beschriebenen USB/RS485-Konverter an den PC angeschlossen wird (**Bild 1**).

Hilfsmodul Hexfunctions

Zunächst eine kleine Aufwärmübung: Ein Script stellt eine Hilfsfunktion zur Hex-Darstellung zur Verfügung und zeigt, wie man wiederverwendbare Funktionen in Module auslagert.
Über den ElektorBus werden die Daten binär übertragen. Es kommen also auch nichtdruckbare Zeichen vor, die in einem normalen Terminalfenster nicht optimal dargestellt werden. Daher soll eine Funktion die empfangenen Daten im Hex-Format anzeigen.
Hierzu wird der Code von **Listing 1** in der Datei *Hexfunctions.py* abgelegt. Die Funktion lässt sich so auch in anderen Programmen nutzen. Bei Bedarf kann das Modul noch um andere Funktionen erweitert werden. Ganz „pythonisch“ enthält das Modul gleich auch seine eigene Testfunktion. Wird es direkt gestartet, enthält die Variable *__name__* den Wert *„__main__“* und der untere Teil wird ausgeführt, so dass man gleich das Ergebnis der Umwandlung an einem Beispiel sieht. Als Ergebnis erscheint:

```
HELLO
48 45 4C 4C 4F 0A
```

Die Funktion lässt sich auch in einem anderen Programm einsetzen. Beispielsweise in *Test_hexfunctions.py*:

```
from hexfunctions import *
s="HELLO\n"
print s, translate2hex(s)
```

Das importierte Modul muss im gleichen Verzeichnis wie das Hauptprogramm oder im Python-Suchpfad liegen. Das Auslagern von Funktionen in Module ist praktisch, denn es erhöht die Code-Übersichtlichkeit. Außerdem können diese Module überall benutzt werden.

GUI mit wxPython

Viele Wege führen zu ansprechenden grafischen Oberflächen. Bei Python ist keiner so bequem wie der *Forms Designer* von Visual Basic. Dafür hat man allerdings die volle Kontrolle über das Ergebnis, und nach Einarbeitung klappt es dann doch recht gut. Meine ersten Versuche habe ich mit *Boa Constructor* gemacht. Das Debugging gestaltete sich ohne Verständnis der Hintergründe allerdings sehr mühsam. Etwas einfacher war dann *Python Card*. Wenn man nicht tief einsteigen will, ist dies sehr praktisch. Leider muss hierzu auf jedem Ziel-Rechner *Python Card* installiert sein. *Tkinter* ist die mit Python installierte GUI-Bibliothek. Sie ist recht einfach, enthält aber weniger Objekte als *wxPython*. Da mir eine Ansteuerung des Clipboards fehlte, habe ich mich für das umfangreichere *wxPython* entschieden.
Der Code von **Listing 2** enthält das Grundgerüst für eine grafische Oberfläche. Dieses Programm wird noch schrittweise erweitert. Damit es läuft, muss selbstverständlich *wxPython* installiert sein. Das Hauptfenster wird objektorientiert als Klasse *MyFrame* definiert, die von der Klasse *wx.Frame* alle Eigenschaften wie die Funktionen zum Vergrößern und Verkleinern, Verschieben etc. erbt. Die Eigenschaften des Fensters werden in der Funktion *__init__* festgelegt. Dort definieren wir die grafischen Elemente - hier eine Textbox und ein Knopf. Der Einfachheit halber werden sie mit fester Größe und Position erzeugt. Mit sogenannten Sizern wäre es auch möglich, ein sich auto-

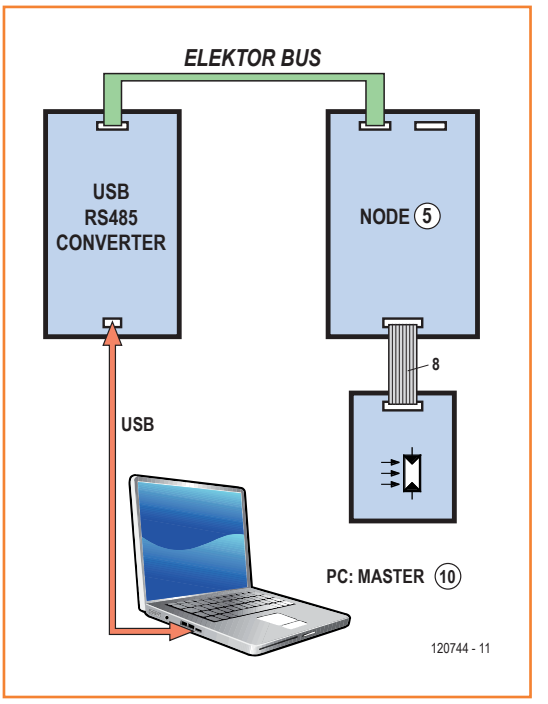


Bild 1.
Der PC empfängt Daten über den ElektorBus (RS485) von einer kleinen Controller-Platine, an die ein Fotosensor angeschlossen ist.

Listing 1: Hexfunctions.py

```
def translate2hex(c):
    """ translate character string c to hex representation string
    e.g ABC -> 41 42 43 """

    h=""
    for ch in c:                # iterate over all characters
        b=hex(ord(ch))          # get hex value
        b=b.replace("0x","")    # take away leading "0x for better overview"
        b=b.upper()             # all in upper characters
        if len(b)<=1:
            b="0"+b             # e.g. make "0A" out of "A"
        h=h+b+" "               # separate bytes by space

    return h

# test:
if __name__ == "__main__":
    s="HELLO\n"
    print s, translate2hex(s)
```

Listing 2: GUI_template.py

```
import wx

#  GUI
class MyFrame(wx.Frame):

    def __init__(self, **kwargs):
        # create frame
        wx.Frame.__init__(self, None, **kwargs)

        # text box with fixed width font for nice data representation
        self.textbox=wx.TextCtrl(self, style = wx.TE_MULTILINE,
            pos = (5,5),size=(300, 200))
        myfont = wx.Font(12, wx.MODERN, wx.NORMAL, wx.BOLD, False, u'Courier')
        self.textbox.SetFont(myfont)

        self.button=wx.Button(self, -1, "TEST", pos=(100,230))

        # Bindings
        self.Bind(wx.EVT_IDLE, self.OnIdle)
        self.Bind(wx.EVT_WINDOW_DESTROY, self.OnDestroy)
```

Listing 3: Klasse Serialthread

```
class Serialthread(serial.Serial):
    def __init__(self, port, baud, **kwargs):
        # Initialization of port + baudrate
        serial.Serial.__init__(self)
        self.sCOM =serial.Serial(port)
        self.sCOM.setBaudrate(baud)

        # open port if not already open
        if self.sCOM.isOpen()==False:
            self.sCOM.open()
        if self.sCOM.isOpen()==True:
            print „connected to“, self.sCOM.port
        else:
            print „Error opening port“

        # Counter for received data blocks
        self.ctr=0

        # Create stop event (to terminate endless receiving loop)
        # and message queue for thread (to transmit received text to TextCtrl)
        self.stopevent=threading.Event()
        self.msgQueue=Queue.Queue()

    def disconnect(self):
        # set stop event so endless receiving loop can be interrupted
        self.stopevent.set()

    def connect(self):
        # create a new thread object that runs serial thread
```

```
        self.Bind(wx.EVT_BUTTON, self.OnButton)

    def OnIdle( self, event):
        # if nothing else to do, update text from message queue
        pass

    def  OnDestroy(self, event):
        print “Exit”

    def OnButton(self, event):
        self.textbox.AppendText (“Button pressed\n”)
#---------------------------------------------------------------------

# Main program
if __name__ == “__main__”:
    app = wx.App(redirect = False)
    frame = MyFrame(title=”GUI”, size = (320,270))
    frame.Show(True)
    frame.Centre()
    app.MainLoop()
```

```
        # to read serial characters
        self.serialthread = threading.Thread(target=self.readSerial)

        # clear stopevent and Connect thread
        self.stopevent.clear()
        self.serialthread.start()

    def readSerial(self):
        # endless receiving loop
        while not self.stopevent.isSet():
            data=““

            # read from port
            c = self.sCOM.read(1)

            # synchronize
            if ord(c) == 0xAA:
                self.ctr += 1
                rest = self.sCOM.read(15)
                data=c+rest

            # format c to 16 bytes output
            datastring=str(self.ctr) + „\t“ + translate2hex(data) + „\n“

            # update message queue
            self.msgQueue.put(datastring)
            wx.WakeUpIdle()         # wake up to update text

        # end serial thread
        print „disconnected“
        self.sCOM.close()
```

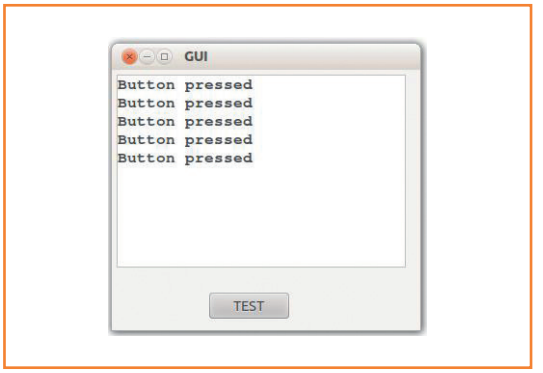


Bild 2.
Dieses Fenster wird vom Code aus Listing 2 erzeugt.

matisch anpassendes Layout zu generieren.
Im nächsten Schritt werden drei Event-Handler erzeugt, von denen wir zwei erst später benötigen. Traditionsgemäß beginnt der Name eines Event-Handlers immer mit „On“. Mit *Bind* werden diese Funktionen an bestimmte Ereignisse gebunden. *OnButton* reagiert z.B. auf einen Knopfdruck. Ist die Klasse MyFrame definiert, kann sie im Hauptprogramm benutzt werden. Dieses erzeugt zunächst ein wx.App-Objekt, welches sich hauptsächlich um die Verwaltung von Ereignissen kümmert. Dann wird eine Instanz unseres Fensters instanziiert, angezeigt und zentriert. In der Endlos-Schleife *app.MainLoop* werden die Ereignisse behandelt.
Nach Programmstart erscheint das in **Bild 2** gezeigte Fenster. Es reagiert schon auf Knopfdruck und verfügt sogar über weitgehende Editierfunktionen: Per rechter Maustaste erscheint ein Menü zum Selektieren, Kopieren, Löschen und Einfügen von Text.
Dieses Programmgerüst könnte leicht z.B. um eine Funktion zum Abspeichern von Text in eine Datei erweitert werden.

ElektorBus: Lesen

Der Experimentalknoten wird über den USB/RS485-Konverter an den PC angeschlossen. Die kleine Platine statten wir mit einem Mikrocontroller ATmega328 aus und programmieren das unter [4] downloadbare Hex-File in dessen Flash-Speicher. An den Erweiterungsstecker des Experimentalknotens (ADC0) schließt man zum Beispiel ein Poti oder einen Fotowiderstand an, so wie in [2a] und [2b] beschrieben.
Zunächst muss man nun den verwendeten seriellen Port des PCs identifizieren. In Windows schaut man per Device-Manager, welcher neue COM-Port auftaucht, wenn man den Konverter einsteckt. Unter Linux informiert man sich per Kommandozeile:

```
ls /dev/tt*U*
```

Als Resultat erhält man dann z.B.:

```
/dev/ttyUSB0
```

Auch das kleine Python-Script zum Scannen der Ports aus Teil 1 [3] funktioniert (der dort abgedruckte Code von ScanSerial.py enthielt allerdings einen Layout-Fehler, die Download-Version ist korrekt).
Sollte eines der folgenden Scripts nicht funktionieren, dann sollte man zuerst überprüfen, ob der korrekte Port eingestellt ist. Beim Experimentieren kann das Betriebssystem unter Umständen „heimlich“ die Port-Nummer ändern. Das kann z.B. passieren, wenn man bei mit ttyUSB0 laufendem Script den USB-Stecker abzieht und wieder einsteckt. In diesem Fall wird dem Konverter dann *ttyUSB1* zugewiesen. Das Script kann dann keine Daten mehr empfangen. Im normalen Betrieb kommt so etwas allerdings kaum vor.

Das GUI-Template von Listing 2 kann nun schrittweise erweitert werden. Zu Beginn müssen benötigte Module geladen und die Schnittstellenparameter festgelegt werden:

```
COMport = “/dev/ttyUSB0” # hier anpassen!
Baud = 9600

import threading, Queue
import serial
import time
```

Das Modul *threading* wird benötigt, da der serielle Empfang in einem separaten Prozess bzw. Thread laufen muss, denn zum Empfangen braucht es eine Endlos-Schleife, die ansonsten mit der Hauptschleife von *wx* in Konflikt geraten würde. Dies ist der komplexeste Teil der Programmierung.
Zum Zugriff auf die Schnittstelle erstellen wir eine eigene Klasse *Serialthread* (**Listing 3**). Ein Objekt dieser Klasse öffnet die Schnittstelle, liest die ankommenden Daten, formatiert sie und schickt sie über eine *Message Queue* an die anderen Teile des Programms. Dies läuft in einer unabhängigen Endlosschleife, bis der Thread gestoppt wird. Ein Serialthread-Objekt erbt mit Port-Name, Baudrate, Funktionen zum Schreiben und Lesen etc. alle Eigenschaften und Methoden der Basisklasse *Serial*. In der Prozedur *__init__* wird ein Seri-

al-Objekt erzeugt, das alle Port-Operationen ausführt. Wenn er nicht schon offen ist, wird damit auch der Port geöffnet und die Baudrate gesetzt. Der Zähler *self.ctr* ist nicht wichtig. Er dient später zum Nummerieren der Datenblöcke.
Es werden noch zwei wichtige, vom Thread benutzte Objekte erzeugt: Ein Stop-Event zum Verlassen des Threads und eine *Message Queue* zur Übertragung von Daten an das GUI.
Von außen startet und stoppt man den Thread mit den Methoden *connect* und *disconnect*. Dabei setzt *disconnect* nur das Stop-Event. Mit *connect* startet man einen neuen Thread, in dem die Funktion *readserial* ausgeführt wird. Diese liest bis zum Stop-Event in einer Endlosschleife Bytes von der Schnittstelle. Anschließend wird der Thread beendet und der Port geschlossen.
Innerhalb von *readSerial* gibt es noch einen Mechanismus zur Synchronisierung der Daten. Beim ElektorBus startet jedes Datenpaket mit 0xAA. Kommt ein solcher Wert vor, wird der Datensatzzähler erhöht und danach noch 15 Daten-Bytes eingelesen. Diese werden nun formatiert und als String in die Message Queue gesteckt. Mit *wx.WakeUpIdle()* wird dem GUI-Teil des Programms signalisiert, dass er die Message Queue auslesen kann, wenn sonst gerade nichts zu tun ist.
Soviel zu *Serialthread*. Mit dieser Klasse kann man fortlaufend Daten lesen, ohne den sonstigen Programmfluss zu stören. *Serialthread* läuft quasi parallel zu anderen Programmteilen. Damit das möglich ist, muss im Hauptprogramm ein Objekt der Klasse *Serialthread* erstellt und gestartet werden. Hierzu wird der Code von *MyFrame* (dem im GUI-Template-Script erstellten Hauptfenster) erweitert. In der Prozedur *__init__* wird folgendes (hinter den Bindings) hinzugefügt:

```
class MyFrame(wx.Frame):

    def __init__(self, **kwargs):
        ...

        # Bindings
        ...

        # serial thread
        self.serialreceive =
Serialthread(COMport, Baud)
        self.serialreceive.connect()
```

Nun gibt es ein Objekt *serialreceive*, das mit dem durch die Variable *COMPort* festgelegten Port verbunden ist und alle einkommenden Daten in die *Message Queue* schreibt. Damit man etwas davon bemerkt, muss der Text aus der *Message Queue* extrahiert und in die Textbox geschrieben werden. Hierzu wird die schon vorbereitete Funktion *OnIdle* benutzt:

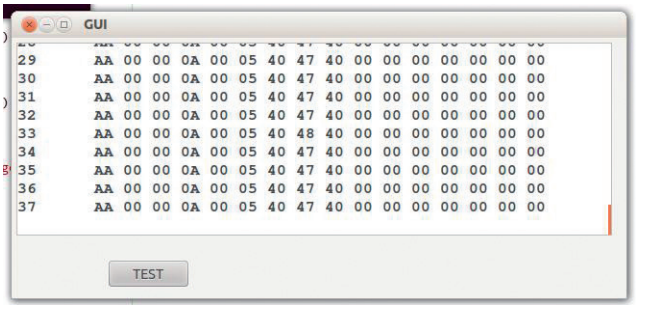


Bild 3. Empfangene Daten hexadezimal formatiert ausgegeben.

```
class MyFrame(wx.Frame):

    def __init__(self, **kwargs):
        ...

    def OnIdle( self, event):
        # if nothing else to do, update
text from message queue
        while not self.serialreceive.
msgQueue.empty():
            msg=self.serialreceive.
msgQueue.get()
            self.textbox.AppendText(msg)
```

Der Befehl *pass* dort war nur ein Platzhalter. Er wird entfernt. Mit diesen Änderungen sollten die ankommenden Daten wie in **Bild 3** gezeigt mit laufender Nummer in der Textbox sichtbar werden.
Damit beim Schließen des Fensters keine hässlichen Fehlermeldungen erscheinen, kommt noch eine Kleinigkeit bei der Prozedur *OnDestroy* hinzu:

```
def OnDestroy(self, event):
    self.serialreceive.disconnect()
    time.sleep(1)
```

Nun wird vor dem Schließen des Fensters zuerst der serielle Thread beendet. Dies kann etwas dauern, weswegen die kleine Pause mit *time.sleep(1)* eingebaut wurde. Die Änderungen dieses Kapitels ergeben zusammen das neue Programm *Serialreceive1.py*, das wie der andere Code auch kostenlos von der Elektor-Webseite zu diesem

Artikel [4] heruntergeladen werden kann.

Zur Modularisierung kann die Definition der Klasse *Serialthread* getrennt als Modul *Serialthread.py* abgespeichert werden, wenn nichts mehr daran geändert wird. Im Hauptprogramm wird sie dann importiert. So bleibt alles schön übersichtlich.
Hierzu ist noch eine kleine Anpassung nötig, denn die von *Serialthread* benötigten Module müssen hier und nicht im Hauptprogramm importiert werden. Die Datei *Serialthread.py* enthält auch noch diesen Code:

```
import threading, Queue
import serial
import time
from hexfunctions import *
import wx

class Serialthread(serial.Serial):
    def __init__(self, port, baud,
**kwargs):
        # Initialization od port +
baudrate
        serial.Serial.__init__(self)
        ....
        # end serial thread
        print “disconnected”
        self.sCOM.close()
```

Im resultierenden Hauptprogramm *Serialreceive2.py* kann nun der ganze Definitionsblock der Klasse *Serialthread* weggelassen werden. Auch die Import-Zeilen braucht man nicht, da die Importe in *Serialthread.py* durchgeführt werden.
Genau genommen ist diese Aufteilung noch nicht voll befriedigend, denn das Modul *Serialthread.py* ist recht projektbezogen ausgelegt und nicht allgemein anwendbar, doch für hier genügt es.

ElektorBus: Schreiben

Der Experimental-Knoten enthält eine rote LED. Diese soll nun vom PC aus mit zwei Knöpfen ein- und ausgeschaltet werden. Hierzu muss man eine bestimmte Byte-Sequenz über den ElektorBus schicken (siehe Bus-Spezifikation [1]):

- Einschalten: AA 00 00 05 00 0A 00 00 00 00 **60 01** 00 00 00 00
- Ausschalten: AA 00 00 05 00 0A 00 00 00 00 **60 00** 00 00 00 00

Im GUI-Programm muss also ein zweiter Knopf und ein zweiter Event-Handler hinzugefügt werden, und die Knöpfe sollen sinnvolle Bezeichnungen erhalten. Hierzu wird *__init__* der Klasse *MyFrame* editiert:

```
    def __init__(self, **kwargs):
        # create frame
        ....
        buttonOn=wx.Button(self, -1, “LED
ON”, pos=(100,230))
        buttonOff=wx.Button(self, -1,
“LED OFF”, pos=(200,230))

        # Bindings
        ....
        buttonOn.Bind(wx.EVT_BUTTON,
self.OnButtonOn)
        buttonOff.Bind(wx.EVT_BUTTON,
self.OnButtonOff)
```

Der Event-Handler ruft die Funktionen *self.OnButtonOn* und *self.OnButtonOff* auf. Diese müssen noch ergänzt werden:

```
    def OnButtonOn(self, event):
        self.textbox.AppendText (“LED
ON\n”)
        data=b”\xAA\x00\x00\x05\x00\x0A\
x00\x00\x00\x00\x60\x01\x00\x00\x00\x00”
        self.serial_thread.sCOM.
write(data)

    def OnButtonOff(self, event):
        self.textbox.AppendText (“LED
OFF\n”)
        data=b”\xAA\x00\x00\x05\x00\x0A\
x00\x00\x00\x00\x60\x00\x00\x00\x00\x00”
        self.serial_thread.sCOM.
write(data)
```

Wenn man sich den Quelltext der Klasse *Serialthread* anschaut, fragt man sich vielleicht, wo denn die Funktion *write* definiert ist. Sie muss nicht explizit definiert werden, denn sie wird von der als Basisklasse genutzten Klasse *serial.Serial* geerbt und steht somit automatisch zur Verfügung.
Mit diesen Änderungen ergibt sich das Programm *Receive_send.py*. Nun müsste sich die rote LED auf dem Experimentalknoten via PC ein- und ausschalten lassen. Die Software ist noch nicht perfekt, da die Synchronisation mit dem Knoten fehlt. Wenn dieser zufällig gerade gleichzeitig mit dem PC sendet, kommt es zu Kollisionen auf dem Bus und damit zu fehlerhaften Daten. Dieses kann vermieden werden, indem man die

Nachricht vom PC aus genau dann verschickt, wenn gerade zuvor Daten vom Knoten empfangen wurden (ElektorBus *DirectMode*, siehe dazu die ElektorBus-Spezifikation [1]). Hierzu könnte man z.B. ein Flag setzen und sofort nach dem Empfang der Daten bei Bedarf die Sequenz zum Schalten abschicken.

Diagramme

Nun soll zur Anzeige der Daten als Text noch ein Diagramm hinzukommen, das die vom A/D-Konverter des Knotens erzeugten Werte visualisiert. Die Standard-Bibliothek zur grafischen Darstellung ist *Matplotlib*. Diese Library muss daher installiert sein. In Teil 1 dieser Artikelserie wurde mit dem einfachen Interface *pyplot* gearbeitet, mit dem man sehr einfach und schnell Diagramme erstellen kann. Wenn man das Diagramm aber in eine grafische Oberfläche einbetten will, wird es etwas komplizierter: Man muss das objektorientierte Interface benutzen, das wesentlich mehr Möglichkeiten bietet. Die zugehörige Dokumentation dazu gibt es auf der Matplotlib-Homepage [5]. Die Vielfalt der gebotenen Möglichkeiten kann Anfänger fast überfordern. Ein Beispiel zum Einbetten in *wx* findet sich in [6].
Vor einer Änderung des Moduls Serialthread speichert man dieses zuvor unter anderem Namen, z.B. als *Serialthread_diagram.py* ab. Die darzustellenden Werte des A/D-Konverters befinden sich im 5. und 6. Byte des empfangenen Datenblocks. Sie müssen daraus extrahiert und an das Hauptprogramm übergeben werden. Dafür gibt es mehrere Möglichkeiten. Eine besteht darin, dem Objekt *Serialthread* die Arrays *x* und *y* als Attribute hinzuzufügen. Hiermit werden die x- und y-Werte des Diagramms übergeben. Hierzu wird *__init__* so ergänzt:

```
class Serialthread(serial.Serial):
    def __init__(self, port, baud,
**kwargs):
        ...

        ## init arrays and timer for data
        self.x=[]
        self.y=[]
        self.starttime=time.time()
        ...
```

Hier werden zwei leere Arrays für die x- und y-Werte angelegt. Das zusätzliche Attribut *starttime* dient zur Berechnung der abgelaufenen Zeit in Sekunden, damit statt der Datensatznummer die Zeit als x-Wert im Diagramm angegeben werden kann.
In der Empfangs-Funktion werden die ADC-Werte extrahiert, zusammengesetzt und in den xy-Arrays abgespeichert:

```
    def readSerial(self):

        # endless receiving loop
        while not self.stopevent.isSet():
            ...
            # synchronize
            if ord(c) == 0xAA:
                self.ctr += 1
                rest = self.sCOM.read(15)
                data=c+rest

                ## update x,y
                lbyte = ord(rest[6])
                hbyte = ord(rest[5]) & 7
                adc  =  lbyte + hbyte
*256
                t=time.time()-self.
starttime
                self.x.append(t)
                self.y.append( adc)
                print t,adc

                # format c to 16 bytes
output
                ....
                wx.WakeUpIdle()         #
wake up to update text
```

So umgebaut liefert das Modul zusätzlich zur hexadezimalen Darstellung die Werte für das Diagramm.

Nun zu den Änderungen am Hauptprogramm. Da der Name des Moduls *Serialthread* geändert wurde, muss die Änderung auch hier durchgeführt werden:

```
from serialthread_diagram import  *
```

Zum Zeichnen eines Diagramms müssen zuerst die benötigten Module importiert werden:

```
from matplotlib.backends.backend_wxagg
import FigureCanvasWxAgg as FCanvas
from matplotlib.figure import Figure
```

Matplotlib arbeitet mit sogenannten Backends, welche die eigentliche Zeichenaufgabe überneh-

Bitte bei allen Listings auf die Einrückungen achten! Stimmen sie nicht, kommt es zu Fehlermeldungen oder das Programm macht seltsame Dinge.

men. Das Backend unterscheidet sich je nachdem, ob ein Diagramm auf dem Bildschirm z.B. mit *wx*, einem anderen System, dem Drucker oder in eine Grafik-Datei ausgegeben wird. Die erste Zeile erstellt ein Objekt *FCanvas* für *wx*, auf das Matplotlib zeichnen kann. *FCanvas* erbt Methoden und Eigenschaften wie die Anpassung von Größe und Position von *wxPanel*.
Das Objekt *Figure* ist ein Container, in den gezeichnet wird. *Figure* ist aber noch nicht das eigentliche Diagramm. In ein Figure-Objekt können ein oder mehrere Axes-Objekte eingebettet werden. Diese sind die eigentlichen Diagramm-Flächen. Das ist zwar verwirrend, doch der Leistungsfähigkeit von Matplotlib geschuldet. Um alle Fenster einigermaßen gut unterzubringen wurden noch Anpassungen am Layout und an der Schriftgröße vorgenommen. Details dazu sieht man im Listing.
Das Fenster für das Diagramm (siehe **Bild 4**) wird in der Funktion *__init__* der Klasse *MyFrame* aufgebaut:

```
class MyFrame(wx.Frame):

    def __init__(self, **kwargs):
        # create frame
        wx.Frame.__init__(self, None,
**kwargs)

        # text box with fixed width font
for nice data representation
        self.textbox=wx.TextCtrl(self,
style = wx.TE_MULTILINE,
                pos = (5,5),size=(420,
200))
        myfont = wx.Font(10, wx.MODERN,
wx.NORMAL, wx.BOLD, False, u'Courier')
        self.textbox.SetFont(myfont)

        buttonOn=wx.Button(self, -1, "LED
ON", pos=(100,230))
        buttonOff=wx.Button(self, -1,
"LED OFF", pos=(200,230))
        ## diagram
        self.figure = Figure()
        self.axes = self.figure.
add_subplot(111)
        self.canvas = FCanvas(self, -1,
self.figure)
        self.canvas.SetPosition( (450,5))
        self.canvas.SetSize((300,250))
        ....
```

Zunächst wird in einer Figure-Instanz ein Subplot erzeugt. Ein Diagramm kann z.B. zur Darstellung mehrerer Datenreihen mehrere Subplots enthalten. Die Syntax ist dabei:

```
figure.add_subplot(numrows, numcols,
fignumber)
```

In unserem Fall gibt es nur ein Diagramm (*fignumber = 1*) und daher nur eine Zeile (*numrows = 1*) und eine Spalte (*numcols = 1*). In den letzten drei Zeilen wird mit dem FCanvas-Objekt die Zeichenfläche des wx-Backends erzeugt, positioniert und in der Größe festgelegt. Nun kann ein Diagramm gezeichnet werden.
Eine zweite Änderung betrifft die Funktion *OnIdle*. Sie wird immer dann aufgerufen, wenn das Programm sonst nichts zu tun hat - speziell dann, wenn neue Daten anstehen.

```
def OnIdle( self, event):
        # if nothing else to do, update
text from message queue
        while not self.serial_thread.
msgQueue.empty():
            msg=self.serial_thread.
msgQueue.get()
            self.textbox.AppendText(msg)

            ## display values in diagram
            self.axes.plot(self.serial_
thread.x, self.serial_thread.y)
            self.canvas.draw()
```

Hier werden die bisher empfangenen xy-Werte geplottet (das x-Array enthält die Zeit in Sekun-

den). Sichtbar wird das Diagramm erst mit dem Befehl *canvas.draw()*.
Die in diesem Kapitel vorgenommenen Änderungen ergeben das Programm *Diagram.py*. In diesem einfachen Beispiel wurde darauf verzichtet, das Diagramm fest zu skalieren. Die Skalierung passt sich den Daten fortwährend an. Ein weiterer Effekt der einfachen Realisierung ist ein ständiger Farbwechsel der Kurve, da jedes Mal ein neues Diagramm gezeichnet wird. Will man nach den eigenen Wünschen formatieren und anpassen, kommt man um die Matplotlib-Hilfe nicht herum.

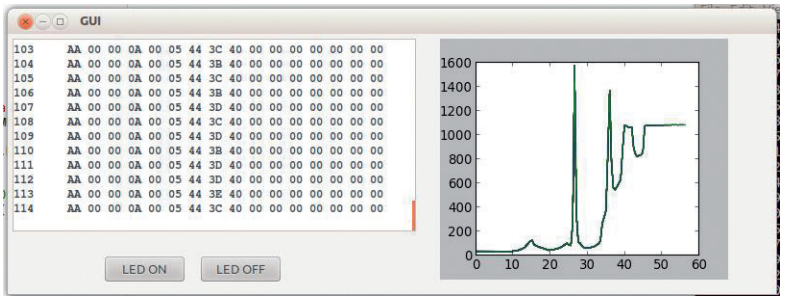


Bild 4.
Terminal-Fenster mit Zahlenwerten und grafischer XY-Darstellung.

Fazit und Ausblick

Nun sollten Sie einen ersten Eindruck von der grundlegenden Vorgehensweise haben. Die beschriebene Software ist sicher noch nicht perfekt. Dazu ist schon die Fehlerbehandlung zu rudimentär. Dafür gibt der Interpreter sofort Meldungen aus, anhand derer man sehen kann, was schief gegangen ist.
Nachteilig ist, dass das Programm nach einer bestimmten Zeit einschläft. Dies liegt daran, dass der Experimentalknoten zwei Mal pro Sekunde neue Werte liefert. Irgendwann sind die xy-Arrays mit so vielen Werten gefüllt, dass das Zeichnen des Diagramms zu lange dauert. Wenn während des Zeichnens schon wieder neue Werte ankommen, ist der GUI-Thread überfordert und das Fenster verfärbt sich grau. Interessanterweise sieht man aber im Terminal-Fenster immer noch die Aktivität des Empfangs-Threads, da dieser unabhängig davon weiter läuft.
Diesen Nachteil kann man zum Beispiel dadurch beheben, indem man das Programm so umschreibt, dass das Diagramm nur alle paar Sekunden einmal neu gezeichnet wird. Probieren Sie es aus!

(120744)

Weblinks & Literatur

[1] ElektorBus-Website:
www.elektor.com/elektorbus

[2a]ElektorBus, Teil 6:
www.elektor.de/110258

[2b]ElektorBus, Teil 8:
www.elektor.de/110428

[3] Von BASIC nach Python - Teil 1:
www.elektor.de/110483

[4] Von BASIC nach Python - Teil 3:
www.elektor.de/120744

[5] Doku zu Matplotlib:
http://matplotlib.org/contents.html

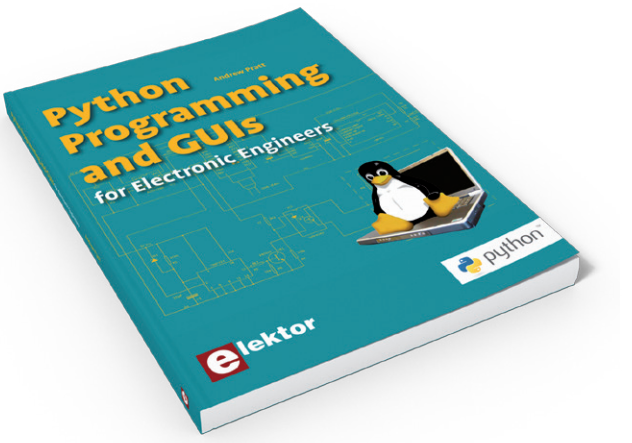
[6] Sandro Tosi: Matplotlib for Python Developers

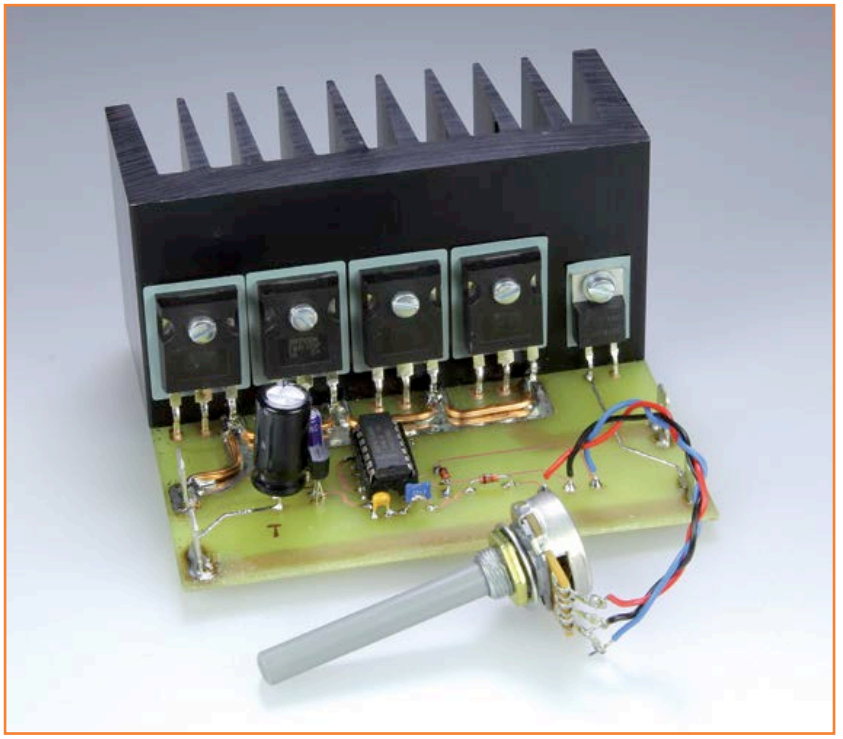
[7] Homepage des Autors:
http://staff.ltam.lu/feljc/home.html

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

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 [7]).





Treiber für dicke DC-Motoren

Von **Ton Giesberts**
(Elektor-Labor)

Diese einfache Schaltung ist für alle Arten von Gleichspannungsmotoren bis 40 A geeignet. Im Grunde handelt es sich nur um einen einfachen Oszillator, der eine Reihe von Leistungs-MOSFETs treibt. Der rudimentäre RC-Oszillator ist mit einem einzigen Schmitt-Trigger (IC1a) eines Sechsfach-Inverters 40106 aufgebaut. Wenn der Schleifer Richtung D2 gedreht wird, gibt Poti P1 maximale Spannung an den Ausgang. Die beiden Dioden verhindern einen Kurzschluss von Ein- und Ausgang des Inverters. An den Anschlägen des Potis sind die Lade- und Entladezeiten minimal. Beim Prototyp der Schaltung konnten wir einen negativen Impuls mit einer Länge von etwa 1 µs und einen positiven von rund 1,6 µs ermitteln.

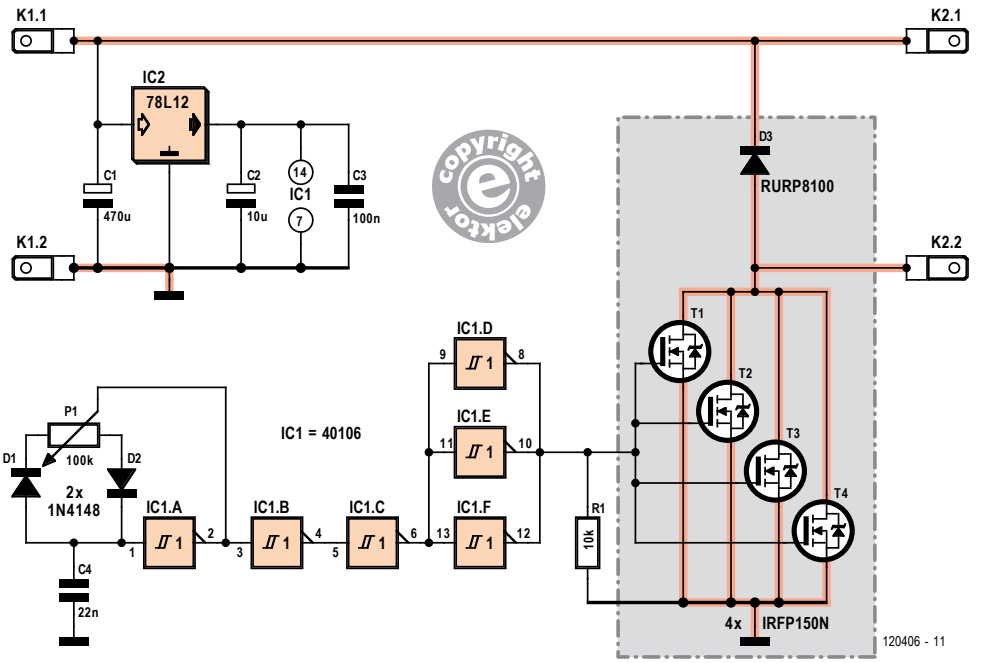
Die nächsten zwei Inverter, IC1b und IC1c, bereinigen das Oszillator-Signal und treiben einen Puffer von drei parallel geschalteten Invertern (IC1d, IC1e und IC1f). Der Widerstand R1 ist erforderlich, um den MOSFET zu sperren, wenn der 40106 nicht in der Schaltung ist. Die gesamte Eingangskapazität der vier MOSFETs beläuft sich auf fast 8 nF, deutlich zu viel für den Puffer, sie voll aufzuladen und zu entladen, wenn sich P1 an einem seiner Anschläge befindet. Das ist nützlich, weil es in der Praxis dem Motortreiber erlaubt, den vollen Bereich der Ausgangsspannung von 0 bis 100 % abzudecken. Die Betriebsfrequenz liegt grob bei 1 kHz, beim Prototyp wurde 1,07 kHz gemessen. Diode D3 am Ausgang unterdrückt die EMK induktiver Lasten, wie sie alle DC-Motoren darstellen.

Hohe Ausgangsströme und EMK sind hier durchaus wichtige Themen. Bei einem frühen Prototyp waren die Leiterbahnen zu D3 zu schmal und beim Testen der Schaltung mit einer der Motoren des Elektor-Wheelies abgebrannt. Unter Volllast zieht jeder der beiden Motoren des Wheelies bis zu 20 A bei 24 V. Die Schaltung wurde mit 40 A und 24 V an einer ohmschen Last getestet. Wie auch immer, die endgülige Platine ist nicht in der Lage, mit solchen Strömen zu arbeiten. Die Lösung besteht darin, die Leiterbahnen mit Stücken von massivem Kupferdraht (2,5 mm^2) hochstrombelastbar zu machen. Möglicherweise sind zwei parallele 1,5-mm^2-Stücke einfacher aufzulöten. Aus diesem Grund ist die Platine auch nicht mit einer Lötstopmaske versehen. Die dickeren Linien im Schaltplan zeigen, wo man hohe Ströme erwarten kann.

Die Versorgung des 40106 übernimmt ein 78L12-Spannungsregler (IC2) mit den üblichen Begleitumständen in Form von großen und kleinen Entkoppelkondensatoren.

Das Poti für die Drehzahlregelung kann abgesetzt von der Platine montiert und mit Schaltlitze angeschlossen werden. Der Kühlkörper wird am besten mit M3-Schrauben an der Platine befestigt. Vergewissern Sie sich, dass der Kühlkörper nicht die Lötpads der MOSFETs berührt. Um mechanische Spannungen an den Halbleiter-Füßchen zu vermeiden, biegen Sie diese leicht rund - es gibt spezielle Werkzeuge dafür – und erst dann legen Sie die Positionen für die Bohrungen der Befestigungsschrauben am Kühlkörper für die Transistoren und D3 fest. Drehen Sie M3-Gewinde in den Kühlkörper und vergessen Sie nicht, alle Halbleiter vom Kühlkörper zu isolieren. Aufgrund der geringen Schaltfrequenz haben Sie gute Chancen, ein Jaulen des Gleichspannungsmotors zu hören, doch keine Angst, das ist ziemlich normal und kein Grund zur Beunruhigung.

(120406)



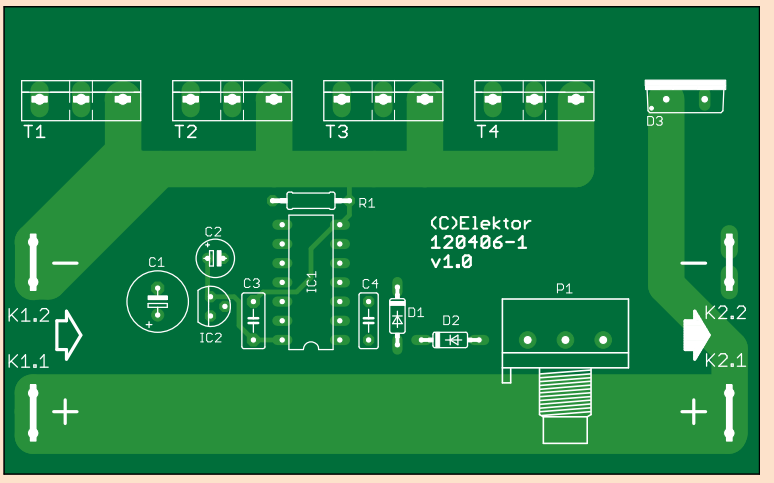
Stückliste

Widerstände
R1 = 10 k, 5%, 0,25 W
P1 = 100 k, 20%, lineares Poti, 0,2 W

Kondensatoren
C1 = 470 µ, 35 V, 20%, RM 3,5 mm
C2 = 10 µ, 25 V, 20%, RM 2 mm
C3 = 100 n, 50 V, 20%, keramisch, RM 5 mm
C4 = 22 n, 100 V, 20%, keramisch, RM 5 mm

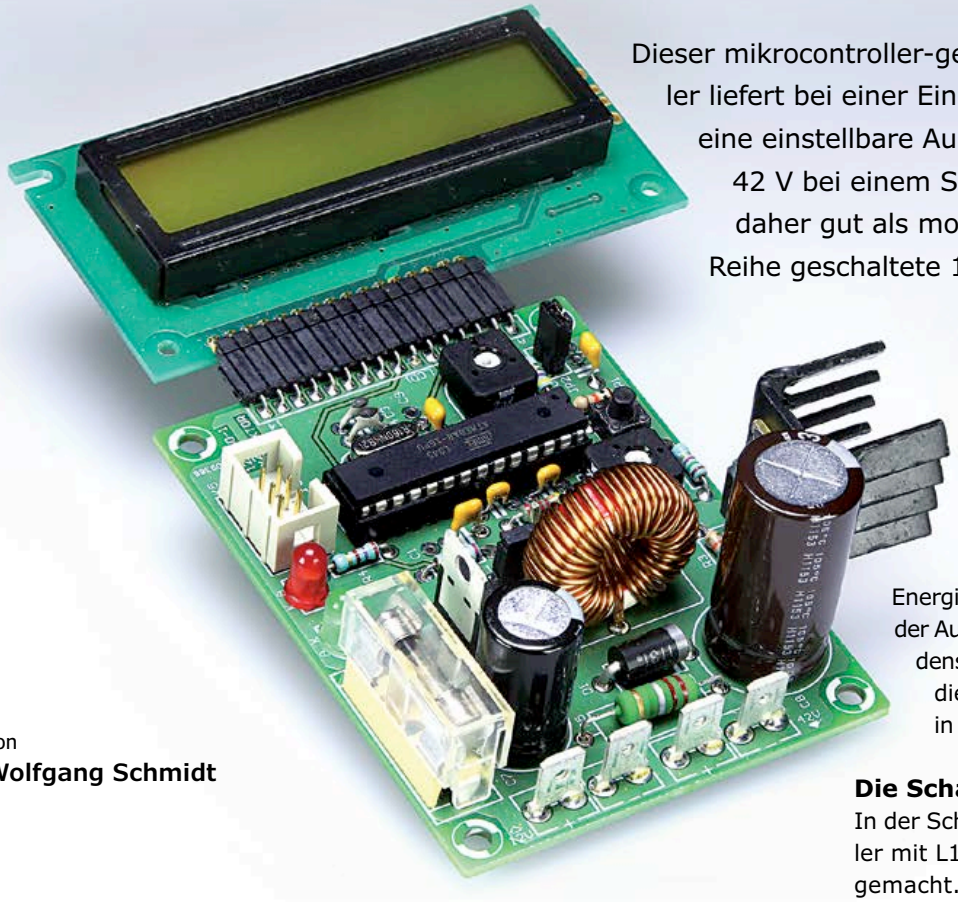
Halbleiter
D1,D2 = 1N4148
D3 = RURP8100
T1..T4 = IRFP150N
IC1 = 40106
IC2 = 78L12

Außerdem
4 Stück FastOn-Anschlüsse, gerade, für Platinenmontage, RM 0,2'' (5,1 mm)
Kühlkörper, 1,9 K/W, 100 x 40 x 50 mm, Fischer Elektronik SK 92/50 SA
TO-3P Silikon-Elastomer-Isolation (T1…T4)
TO-220 Isolationssatz; Mica-Blatt und Hülse (D3)
Platine 120406-1 v1.0



PWM-Aufwärtswandler

Get up, step up ...!



Dieser mikrocontroller-gesteuerte Step-Up-Schaltwandler liefert bei einer Eingangsspannung von 8...16 V eine einstellbare Ausgangsspannung von bis zu 42 V bei einem Strom von etwa 1 A. Er kann daher gut als mobiles Ladegerät für bis zu drei in Reihe geschaltete 12-V-Akkus einsetzt werden.

Von
Wolfgang Schmidt

Ein Aufwärtswandler (Step-Up- oder Boost-Converter) verwandelt eine niedrige Eingangsgleichspannung in eine hohe Ausgangsgleichspannung. Er besteht prinzipiell aus einer Induktivität, einem Kondensator, einer Diode und einem Schalter (Transistor), der mit einer pulsweitenmodulierten Spannung ein- und ausgeschaltet wird. Ein Schaltzyklus der Länge T setzt sich aus der Einschaltzeit t_1 und der Ausschaltzeit $T\text{-}t_1$ zusammen. Während der Einschaltzeit des PWM-Signals ist der Schalter geschlossen (**Bild 1** unten). Über der Induktivität liegt die Eingangsspannung U_e und es fließt in guter Näherung ein linear ansteigender Strom I_L durch die Spule. Dadurch steigt auch die in der Spule gespeicherte Energiemenge. Wird der Schalter geöffnet, so bricht das Magnetfeld der Spule zusammen, wobei in der Spule eine der angelegten Spannung entgegengesetzt gerichtete Spannung induziert wird. Diese Spannung addiert sich in diesem Fall zur angelegten Versorgungsspannung und führt zu einem Stromfluss über die Diode zum Kondensator. Man kann sagen, dass während der Einschaltphase die Induktivität im entstehenden Magnetfeld, welches sich im Wesentlichen im Ferritkern befindet, Energie aufnimmt und diese Energie während der Ausschaltphase über die Diode an den Kondensator übertragen wird. Wenn Sie tiefer in die Materie einsteigen wollen, finden Sie in [1] eine hervorragende Ausgangsbasis.

Die Schaltung

In der Schaltung (**Bild 2**) ist der Step-Up-Wandler mit L1, D1, C8 und MOSFET T1 schnell ausgemacht. Für die Erzeugung des PWM-Signals ist ein Mikrocontroller ATmega8-16PU von Atmel beziehungsweise die in ihm steckende Software verantwortlich. Das PWM-Signal, das an Pin PB1 erscheint, hat eine Frequenz von 66 kHz, da intern der Fast-PWM-Modus gewählt wurde. Weil die Ausgangsspannung über das Tastverhältnis nachgeregelt werden soll, muss der Controller über den aktuellen Wert dieser Spannung informiert sein. Dies geschieht über den Spannungsteiler R6, R7 und P2. Das Trimmpoti ist notwendig, da die vom Controller erzeugte interne Referenzspannung nicht präzise ist. Laut Datenblatt bewegt sie sich zwischen 2,3 V und 2,9 V, sollte aber mit P2 kalibriert werden können. Reicht der Verstellbereich nicht aus oder ist die Wahl von R7 = 43 k problematisch, so kann die Software angepasst werden. Zum Abgleich schließt man einfach ein DVM am Ausgang an und vergleicht die Werte mit denen auf dem LCD.

Der A/D-Wandler des Controllers besitzt eine Auflösung von 10 bit. Die Software berechnet die Spannung über den Spannungsteiler von 47 kΩ (R7+P2) und 2,7 kΩ (R6). Daraus ergibt sich eine Auflösung von 46 mV (((49,7 kΩ/2,7 kΩ)* 2,56 V)/1023). Der Wert, der im Display angezeigt wird, springt deswegen immer um 0,04 V oder 0,05 V.

Aufwärtswandler mit dieser Topologie verfügen über keine Strombegrenzung. Um eine Überlastung zu vermeiden, ist in die Ausgangsleitung ein Shunt R5 aufgenommen, der mit einem zweiten A/D-Eingang des Controllers verbunden ist. Die Software regelt das Tastverhältnis herunter, bevor der Wandler in den diskontinuierlichen Betrieb eintritt.

Um Rauschen an den beiden Wandlereingängen zu vermeiden, wurden C10, C11 und R8 hinzugefügt. An den Controller ist ein LCD angeschlossen, in dem alle Parameter, die aktuellen und (über Menüs) die eingestellten Werte für Ausgangsspannung und Ausgangsstrom dargestellt werden. Die Schaltung verfügt über drei Taster. S1 setzt den Mikrocontroller zurück, über S2 und S3 kann die Ausgangsspannung erhöht beziehungsweise vermindert werden. Drückt man beide Taster gleichzeitig, gelangt man in den Strombegrenzungsmodus. Dann kann man mit den Tastern einen neuen Wert für den maximalen Ausgangsstrom festlegen. Kurze Zeit nach dem letzten Tastendruck erscheint automatisch die normale Spannungsanzeige.

Zu Testzwecken sind zwei LEDs vorgesehen. LED D3 zeigt das Vorhandensein einer Eingangsspannung an. Leuchtet sie nicht, ist die Sicherung F1 durchgebrannt und mit der Schaltung läuft etwas gravierend falsch. D2 leuchtet, wenn die Strombegrenzung aktiv ist.

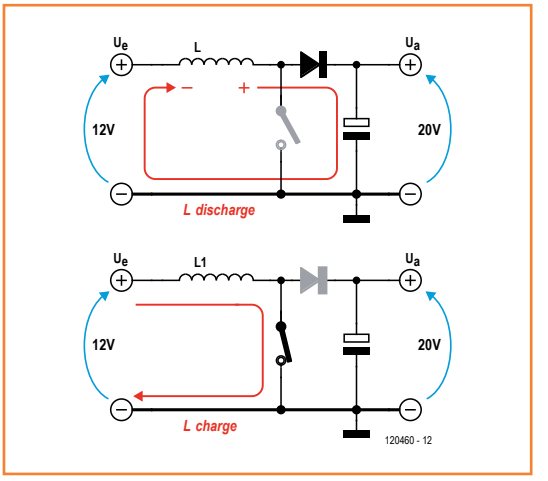


Bild 1.
Die zwei Phasen eines Aufwärtswandlers.

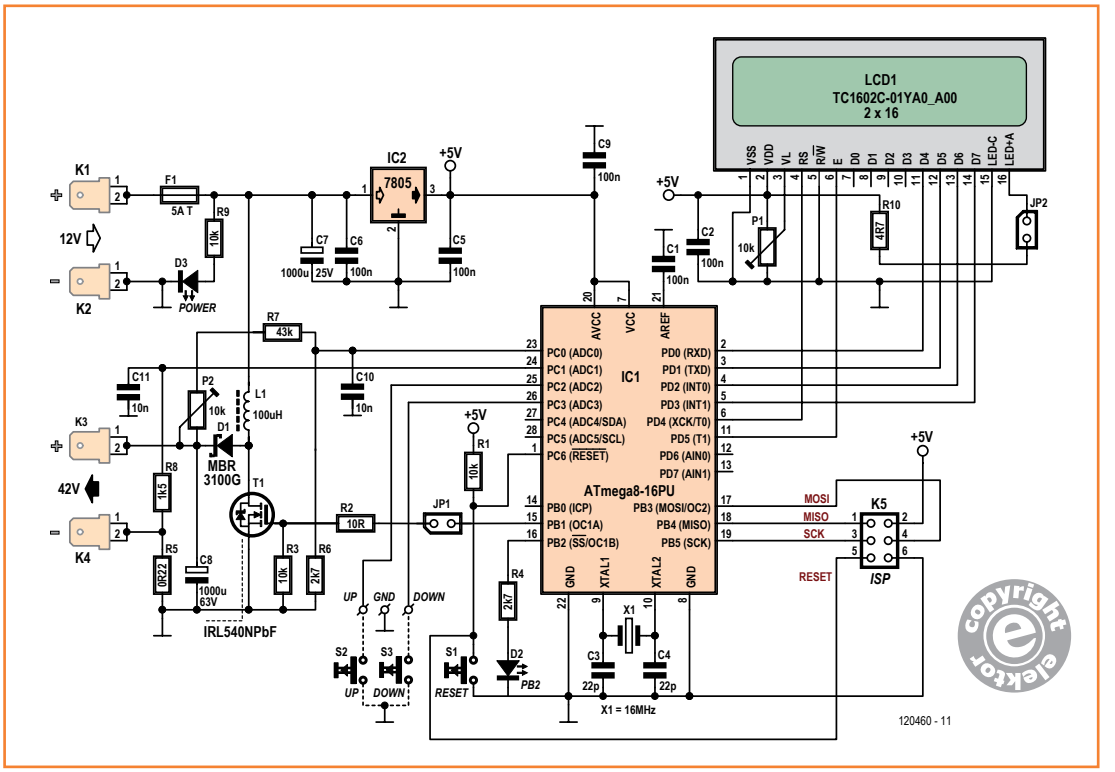


Bild 2.
Ein Step-Up-Wandler mit Mikrocontroller-Regelung.

Auf- und Einbau

Während sich der Reset-Taster auf der Platine (**Bild 3**) befindet (muss nur selten betätigt werden), werden die Tasten S2 und S3 mit etwas flexiblem Kabel über die drei Platinenanschlüsse Up, GND und Down angeschlossen. Je nach Konstruktion kann man zwei taktile Taster auf einem Stück Lochraster oder Typen für Chassismontage an eben einer Gehäusefront anbringen.

Die beiden LEDs machen natürlich nur Sinn, wenn sie gut sichtbar angebracht und nicht im Gehäuse versteckt sind. Mit P1 stellt man den Kontrast des LCD-Moduls ein. Mit JP2 lässt sich die Hintergrundbeleuchtung der Anzeige aktivieren. Hier kann natürlich auch ein Schalter eingesetzt werden.

Zur Kühlung des MOSFETs ist ein kleiner Fingerkühlkörper erforderlich. Bei Strömen bis 1 A ist ein thermischer Widerstand von 21 K/W ausreichend. Für die Diode D1 können sowohl radial bedrahtete Typen (DO201-AD) als auch solche im TO220-Gehäuse eingesetzt werden. Wenn Sie den zweiten Typ verwenden, orientieren Sie sich im Datenblatt, wie herum die Diode eingesetzt werden muss.

Die in Schaltwandlern eingesetzten Kondensatoren bedürfen besonderer Aufmerksamkeit. Dies betrifft hier C7 und C8, die der hohen Schaltfrequenz von 66 kHz ausgesetzt sind. Normale Kondensatoren können mit einer solchen Frequenz nicht umgehen, deshalb sollten Sie unbedingt die in der Stückliste genannten Typen verwenden!

Als wir die Schaltung im Labor testeten, haben wir das LCD mit geraden Pfostenverbindern ausgestattet und es in eine korrespondierende Buchsenreihe auf der Platine gesteckt. Das war sehr praktisch, wenn man den Wandler aber in ein Gehäuse einbauen will, sollte man das Display zum Beispiel an der Rückseite der Platine anbringen. Befestigen Sie ein stabiles, isolierendes Blatt (ich halte für solche Fälle stets einen Vorrat ganz dünner Pertinaxabschnitte bereit) zwischen Platine und Display, um Kurzschlüsse zu vermeiden.

Zur Programmierung des Controllers steht eine klassische ISP-Programmierschnittstelle an K5 zur Verfügung. Der Controller muss während des Programmiervorgangs über den 7805-Spannungsregler IC2 der Schaltung versorgt werden. Die Spannung an K5 informiert damit den Programmieradapter, etwa einen AVRISP mkII, über die Höhe der Versorgungsspannung (3,3 V oder 5 V). Beim Programmieren sind die Pegel an den Controllerausgängen undefiniert. Deshalb ist der Jumper JP1 in der Gateleitung des MOSFETs nötig. Wäre nämlich der Ausgang PB1 zu lange High, würde der MOSFET die Spannungsversorgung kurzschließen. R3 zieht das Gate auf Masse, wenn JP1 nicht eingesteckt ist. Also nicht vergessen, den Jumper während des Programmierens zu ziehen! R2 verhindert Instabilitäten im Umschaltzeitpunkt aufgrund der hohen Gatekapazität.

Die Gatekapazitäten der MOSFETs verursachen eine deutliche Umschaltzeit von EIN nach AUS und umgekehrt. Als Folge erwärmt sich ein Leistungs-MOSFET deutlich, da in diesen Umschaltzeiten beim relativ langsamen Übergang in den jeweils anderen Zustand der Eingangsstrom des Spannungswandlers bei einer Drain-Source-Spannung größer 0 V am MOSFET anliegt. Die daraus resultierende elektrische Leistung wird in Wärme umgesetzt. Verfügt man über hohe Schaltströme, um den Spannungszustand am Gate zu ändern, so werden die Umschaltzeiten verschwindend klein. Der ATmega bietet mit einem Ausgangsstrom von etwa 30 mA aber nur eine schwache Stromquelle.

Die maximale Höhe der Ausgangsspannung wird nur durch die Spannungsfestigkeit von D1 und T1 bestimmt. Dies zeigt erste Ansätze für Verbesserungen der Hardware, die sich hier lediglich auf die Realisierung eines Step-Up-Grundprinzips beschränkt und im Wesentlichen als Anregung dient.

Ausbaufähig: Die Software

Die Software, deren BASCOM-AVR-Quellcode Sie unter [2] finden, stellt einen recht einfachen Laderegler dar, der an diversen Punkten noch verbesserungswürdig ist. Dazu gehören einige kritische Punkte als auch die Implementierung eines „echten“ Bleiakku-Ladereglers mit mehreren Ladephasen.

Der Eingangsstrom ist übrigens bis zu 3,5 Mal so hoch wie der Ausgangsstrom, der Grund für die träge 5 A-Sicherung am Eingang.

Wir haben den Wandler an zwei verschiedenen Blei-Gel-Akkus getestet. Uns ist dabei ein merkwürdiges Verhalten des Ladegeräts aufgefallen. Bei der Einstellung des Stroms auf ein für Blei-Gel-Akkus relativ hohes Niveau von beispielsweise 0,2 C fiel die Batteriespannung rasch ab, nachdem die maximal eingestellte Spannung erreicht war und das Ladegerät abschaltete. In der Pra-

Stückliste

Widerstände:
R1,R3,R9 = 10 k, 5 %, 250 mW
R2 = 10 Ω, 5 %, 250 mW
R4,R6 = 2k7, 5 %, 250 mW
R5 = 0Ω22, 5 %, 1 W
R7 = 43 k, 1 %, 600 mW
R8 = 1k5, 5 %, 250 mW
R10 = 4Ω7, 5 %, 250 mW
P1,P2 = 10 k, 20 %, 0W15, Trimmpoti, liegend

Kondensatoren:
C1,C2,C5,C6,C9 = 100 n, 63 V, 5 %, RM5 oder RM7,5, keramisch
C3,C4 = 22 p, 50 V, 5 %, RM5
C7 = 1000 µ, 25 V, 20 %, radial Ø12,5 mm, RM5 (Panasonic EE-UTP1E102 bei Farnell 1890543)
C8 = 1000 µ, 63 V, 20 %, radial Ø16 mm, RM7,5 (Nichicon UPW1J102MHD bei Farnell 2112865)
C10,C11 = 10 n, 100 V, 10 %, RM5, keramisch

Induktivitäten:
L1 = 100 µH, 5 A, 20 %, radial 25 mm, RM8 (Würth Elektronik 7447070 bei Farnell 2082537)

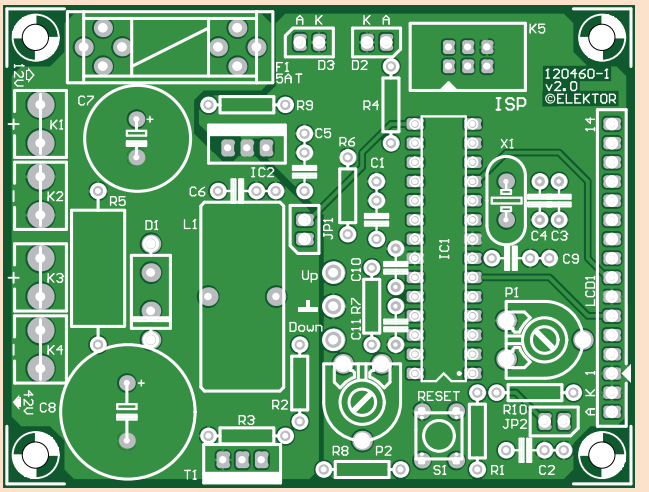


Bild 3. Platine des Aufwärtswandlers.

Halbleiter:
D1 = MBR3100G
D2 = LED rot, 3 mm
D3 = LED grün, 3 mm
T1 = IRL540NPbF

IC1 = ATmega8-16PU (programmiert: 120460-41)
IC2 = 7805

Außerdem:
F1 = Sicherungshalter, 20x5 mm mit Abdeckkappe
F1 = Sicherung 5 A (träge)
JP1,JP2 = 2-poliger Pfostenverbinder SIL, 0,1'' mit Jumper
K1...K4 = AMP-Stecker für Platinenmontage mit Lötanschluss, 5,08 mm
K5 = 2x3-poliger Pfostenverbinder, gerade, 0,1''
S1 = Taktiler Taster 6x6 mm SPST-NO
S2,S3 = Taster SPNO, für Platinen- oder Chassismontage*
PC1...PC3 = Lötnagel 1,3 mm für (S2,S3)
Kühlkörper FK230SAL1 von Fischer Elektronik
X1 = 16 MHz, HC49/US, 50 ppm, C_{load} 18 pF
LCD1 = LCD 2x16 (Elektor 120061-71)
Platine 120460-1

xis muss man den Ladevorgang also beobachten und wissen, wann das Laden von Hand beendet werden muss.
Das Ziel der Software ist freilich nicht, eine komfortable Benutzeroberfläche oder vielfältige Ladekontrollen zur Verfügung zu stellen. Sie zeigt vielmehr die Machbarkeit eines solchen Ladeprinzips und bietet dem interessierten Leser dank der enthaltenen Beschreibung und Übersichtlichkeit ein weites Betätigungsfeld für Verbesserungen.
Eine Anregung: Blei(-Gel)-Batterien sollen in zwei bis vier Phasen geladen werden [3]. Einen (nicht völlig) entleerten Akku lädt man zunächst in der Bulk-Phase mit maximalem Strom (0,1...0,2 C ist sinnvoll), bis eine Klemmenspannung von 2,4 V pro Zelle erreicht ist (so weit kommt auch die hier vorgestellte Software; die Batterie ist zu diesem Zeitpunkt allerdings erst zu etwa 80 % geladen). Dann begrenzt man die Spannung auf die Ladeschlussspannung und beobachtet den Ladestrom, bis er auf ein Zehntel des maximalen Werts gesunken ist. Diese zweite so genannte Absorptions-Phase lädt den Akku fast vollständig auf etwa 98 % auf. Der letzte kleine Rest gelangt in der Float-Phase in den Akku: Die Soll-Spannung am Ausgang wird auf 2,23 V pro Zelle zurückgedreht. In dieser Phase kann der Akku beliebig lang am Ladegerät verbleiben, ohne dass er zu gasen beginnt.
Falls Sie die vierte Phase vermissen, sie kommt dann zu tragen, wenn der Akku tiefentladen ist (unter 1,75 V pro Zelle). Dann muss er mit niedrigem Strom erst aufgepäppelt werden, bis er diese untere Spannungsgrenze erreicht.
Wenn Sie an solchen Verbesserungen der Software arbeiten und ein interessantes Ergebnis vorzuweisen haben, oder daran arbeiten und nicht weiter kommen: Besuchen Sie einmal unsere Project-Site [4]!

(120460)

Weblinks

[1] Schaltwandler-Grundlagen: http://schmidt-walter.eit.h-da.de/smps/smps.html, in Englisch: http://schmidt-walter.eit.h-da.de/smps_e/smps_e.html
[2] www.elektor.de/120460
[3] Laden von Bleiakkus: www.batterystuff.com/kb/articles/battery-articles/battery-basics.html#9
[4] www.elektor-projects.com

Akustische Wasserwaage

ATtiny45 mit Doppelnutzen

Von **Jörg Trautmann**

Dieses kleine Projekt wurde durch die Sensorplatine MMA7260 aus dem Sommerheft 2010 inspiriert. Die Idee war, den Neigungssensor möglichst multifunktional einzusetzen. Herausgekommen ist eine Schaltung mit zweifachem Zweck, die man entweder als akustische Wasserwaage oder zur Positionsüberwachung eines beliebigen Gegenstandes verwenden kann.

Sinn und Zweck des Gerätes soll es sein, einen Gegenstand wie etwa einen großen Tisch im Garten so auszurichten, dass er einigermaßen gerade steht. Da man gerade im Freien so gut wie keine Referenzpunkte hat und der Umgang mit einer Wasserwaage doch recht umständlich ist, kann dieses nützliche Gerät helfen, die Waagerechte zu ermitteln. Außerdem lässt sich das Gerät zur Positionsüberwachung einsetzen: Will jemand den Tisch klauen, ertönt ein Alarm und der vermeintliche Dieb sucht das Weite.

So funktioniert's

Die hier vorgestellte Schaltung in **Bild 1** basiert auf einem ATtiny45 Mikrocontroller und einem MMA7260QT. Der MMA7260 ist ein integrierter 3-Achsen-Beschleunigungssensor, den wir schon 2007 und im Sommerheft 2010 im Rahmen eines „größeren" Neigungsmessers mit LC-Display [1] eingesetzt haben. Die kleine integrierte Schaltung sitzt auf einer winzigen Platine (**Bild 2**) und verfügt über drei analoge Ausgänge, die eine der Beschleunigung proportionale Spannung liefern, bei einer Beschleunigung von +1 g sind es 2,45 V. Der Mikrocontroller ATtiny45 von Atmel [2] verfügt über mehrere A/D-Wandler, wovon drei zur Messung der Beschleunigung beziehungsweise Neigungsänderung verwendet werden. Da die interne Spannungsreferenz des Controllers auf 1,1 V festgelegt ist, müssen die zu messenden Spannungen über Spannungsteiler zugeführt werden. Anhand der bekannten Parameter ergibt sich für die Widerstände R1, R3 und R5 ein Wert von 470 kΩ, für die Fußwiderstände R2, R4 und R6 sind es 330 kΩ. Die maximale Ausgangsspannung von 2,45 V wird so auf etwa 1 V reduziert. Damit ist eine maximale Messauflösung gewährleistet. Der Mikrocontroller ist so programmiert, dass eine Veränderung der X-, Y- und Z-Parameter die Frequenz dreier Tongeneratoren beeinflusst. Befindet sich die Schaltung im ausgerichteten Zustand, schweigen die Tongeneratoren, liegt sie aber um mehr als etwa ± 2° in der Schräge, sind Töne zu hören, welche sich mit steigender Abweichung verändern. Zum Kalibrieren der Schaltung wird Taster S1 verwendet; hiermit wird auch der Betriebsmodus umgeschaltet. Wird der Taster länger als 5 s gedrückt, aktiviert man den Alarmmodus.

Die Spannungsversorgung erfolgt über einen Low-Drop-Spannungsregler des Typs LP2950CZ3.3, der eine sowohl für den Neigungssensor als auch für den Mikrocontroller optimale Spannung von 3,3 V erzeugt. Eine 9-V-Blockbatterie kann damit sehr lange genutzt werden. Beim Testen der Schaltung haben wir eine minimale Speisespannung von etwa 3,6 V ermittelt.

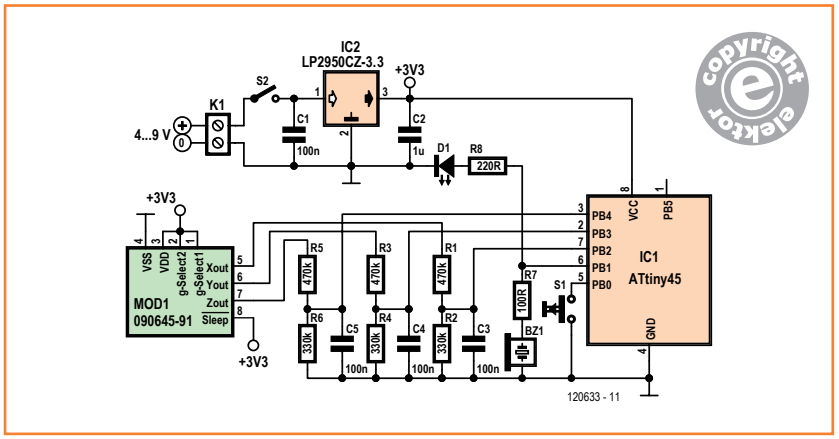
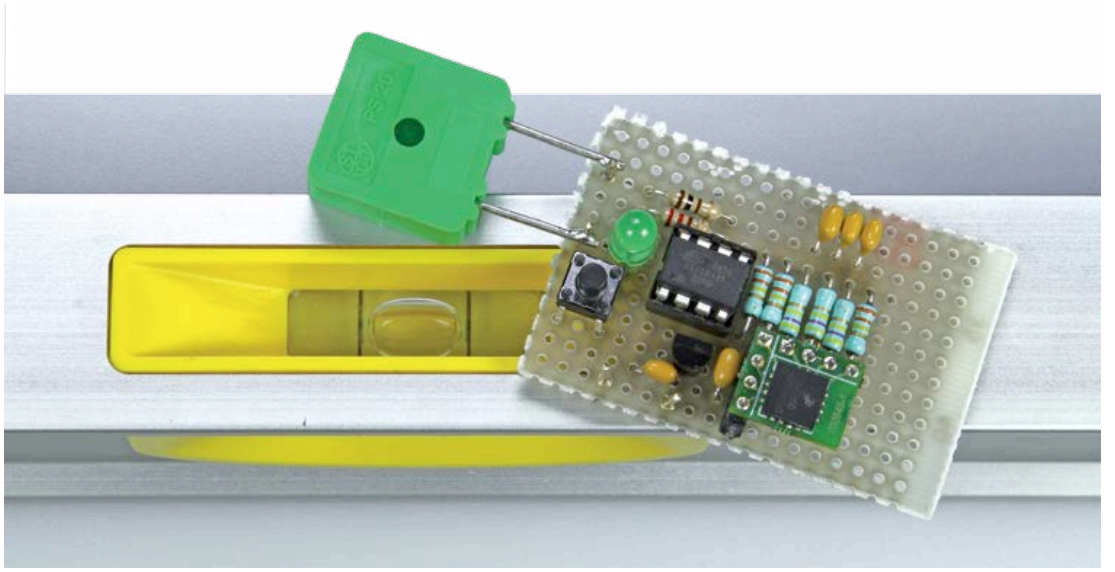


Bild 1.
Die Schaltung ist so winzig, dass sie sich gut auf einem Stück Lochraster aufbauen lässt.



Der maximale Strom beträgt etwa 4,56 mA, der mittlere ungefähr 3,06 mA, wenn die LED blinkt und der Piezo ertönt.

Aufbau und Inbetriebnahme

Der Aufbau ist recht einfach gehalten und kann auf einer Lochrasterplatine wie zum Beispiel der Experimentierplatine ELEX-1 erfolgen. Wird zum ersten Mal die Betriebsspannung angelegt, sollte die rote LED dauerhaft leuchten und nichts im Lautsprecher zu hören sein. Ist dies nicht der Fall, sollte der Schaltungsaufbau nochmals genau geprüft werden.

Nach der ersten Inbetriebnahme muss die Schaltung kalibriert werden. Hierzu legt man die Platine auf den hoffentlich geraden Boden und drückt Taster S1 für etwa eine Sekunde. Nach dem Loslassen des Tasters erlischt die LED und signalisiert die Speicherung der Werte und damit das Ende der Kalibrierung. Das Gerät sollte jetzt keinen Mucks mehr von sich geben. Wird die Platine etwas schräg gehalten, sollten ein, zwei, drei sich überlagernde Piepstöne zu hören sein. Die LED blinkt. Je nach Größe der Abweichung ändert sich die Frequenz dieser Töne. Wird die Platine wieder in die waagerechte Position gebracht, herrscht Ruhe und die LED erlischt. Eingebaut in ein flaches Gehäuse bekommt man selbst mit verbunden Augen ein schnelles Gefühl dafür, wann das Gerät korrekt ausgerichtet ist.

Soll die Schaltung zur Objektüberwachung eingesetzt werden, wird die Kalibriertaste erst dann gedrückt, nachdem das Gerät am zu überwachenden Gegenstand befestigt wurde (dieser Gegenstand muss also nicht zwangsläufig waagerecht ausgerichtet sein). Ist dies geschehen, wird Taster S1 nochmals gedrückt, diesmal allerdings mehrere Sekunden, bis die LED gleichmäßig blinkt. Nach dem Loslassen der Taste ist die Anlage „scharf“. Bei einer Abweichung von etwa 20° ertönt ein alarmanlagentypischer auf- und abschwellender Sirenenton. Um den Alarm abzuschalten, genügt es, den Taster S1 kurz zu drücken. Danach ist das Gerät wieder in Alarmbereitschaft. Um das Gerät wieder in den Wasserwaagenmodus zu versetzen, muss man die Stromversorgung kurz unterbrechen. Die zuletzt gespeicherten Kalibrierungswerte bleiben dabei übrigens gespeichert.

So arbeitet das Programm

Das Programm, das Sie von der Projektseite [3] herunterladen können, ist in BASCOM AVR verfasst. Portpin PB1 wird als Ausgang zum Betrieb des Piezolautsprechers konfiguriert, PB0 als Eingang mit Pullup-Widerstand. Die A/D-Wandler ADC0, ADC1 und ADC2 werden auf eine interne

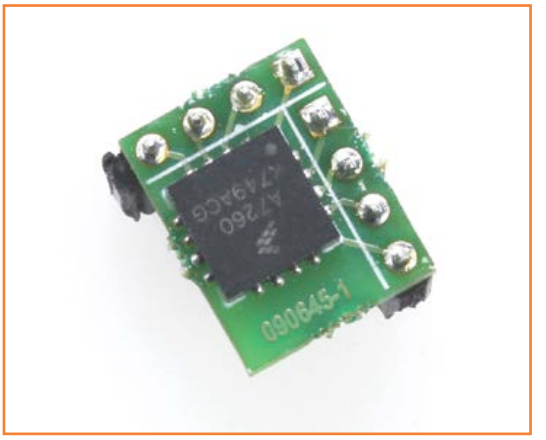


Bild 2.
Der Sensor auf seinem Adapterplatinchen.

Hinweis

Der MMA7260QT von Freescale wird nicht mehr hergestellt. Man kann noch welche im Handel finden, aber dies wird zunehmend schwierig. Im Elektor-Lager befinden sich zurzeit (13.04.2013) noch 67 Module 090645-91 auf Lager. Wer zuerst kommt, mahlt zuerst! Die Software (BASCOM-AVR) lässt sich aber leicht an einen anderen Sensor [4] anpassen.

Spannungsreferenz von 1,1 V gesetzt. Bei geschlossenem Taster S1 (PB0=Low) werden die Werte zur Kalibrierung im EEPROM gespeichert. Nach dem erneuten Anlegen der Versorgungsspannung werden diese Werte als Referenz wieder ausgelesen. Die Logik zur Auswertung des Schaltzustandes von PB0 ist so programmiert, dass eine Kalibrierung nur möglich ist, wenn sich das Programm nicht im Alarmmodus befindet. Ansonsten setzt das Drücken von S1 den Alarm zurück. Man hat also einen Taster für zwei Funktionen.
Ein Messzyklus umfasst sieben 3-Kanal-Messungen innerhalb von 210 ms und anschließender Berechnung des Durchschnittswertes. Dadurch ergibt sich eine ausreichende Messgenauigkeit und Stabilität, wie Experimente ergeben haben. Wer die Empfindlichkeit des jeweiligen Modus ändern möchte, kann den Wert der Variablen Trigger_value anpassen.
Wenn Sie mit den Vorgaben der Software einverstanden sind und sowieso keine Lust zum Programmieren verspüren, können Sie den Controller auch fertig programmiert im Elektor-Service erstehen.

(120633)

Weblinks

[1] www.elektor.de/070829

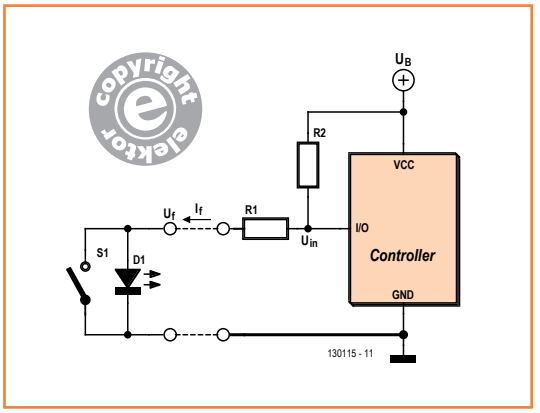
[2] www.atmel.com/devices/ATTINY45.aspx

[3] www.elektor.de/120633

[4] Low-g-Beschleunigungssensoren: www.freescale.com/webapp/sps/site/taxonomy.jsp?nodeId=01126911184209#2

Zweiadriges Interface 3.0

Von John Hind (UK)



Klaus Jürgen Thieslers „Zweiadriges Interface“ wurde als einfache [1] und als stromsparende [2] Variante in Elektor veröffentlicht. Beide Versionen verwenden zwei Transistoren und etliche andere Bauteile, um eine LED plus Taster an einen Mikrocontroller anzuschließen. Davon fühlte sich der Autor herausgefordert, dies nochmals zu vereinfachen und eine Lösung anzubieten, die mit nur zwei Widerständen und einem einzigen GPIO-Pin wohl kaum mehr weiter minimalisiert werden kann, oder etwa doch?

Weniger Bauteile bedeuten, dass der Mikrocontroller etwas mehr tun muss. Voraussetzung für diese Lösung ist, dass ein I/O-Pin zwischen Ein- und Ausgang umgeschaltet werden kann, was so gut wie immer gegeben ist. Als Ausgang mit „High“-Pegel kann er die LED leuchten lassen und als Eingang bei dunkler LED den Status des Tasters erfassen. Wenn man sich die Status-Ta-

Tabelle 1

Status	I/O-Pin	Pegel	Taster	LED	I_f	U_{in}
1	Eingang	high	offen	aus	$(U_B - U_f) / (R1 + R2)$	$U_f + I_f * R1$
2	Eingang	low	geschlossen	aus	$U_B / (R1 + R2)$	$I_f * R1$
3	Ausgang	high	offen	an	$(U_B - U_f) / R1$	U_B
4	Ausgang	high	geschlossen	aus	$U_B / R1$	U_B

Tabelle 2

LED-Farbe	U_f	I_f	@ 5 V: R1 | R2	@ 3,3 V: R1 | R2	@ 2,1 V: R1 | R2
Rot	1,7 V	10 mA	330 Ω | 470 kΩ	160 Ω | 220 kΩ	39 Ω | 56 kΩ
Orange, Gelb	2,1 V	10 mA	300 Ω | 430 kΩ	120 Ω | 180 kΩ	-
Grün	2,2 V	10 mA	270 Ω | 390 kΩ	110 Ω | 160 kΩ	-
Blau, Weiß	3,6 V	20 mA	68 Ω | 200 kΩ	-	-

belle anschaut, dann dürften die beiden ersten Zeilen mit den Nummern 1 und 2 klar sein. Der I/O-Pin ist als Eingang geschaltet und je nach Zustand des Tasters liegt eine Spannung U_{in} an, die als „low“ oder „high“ interpretiert wird - wenn die Widerstände R1 und R2 passend zur Versorgungsspannung U_B richtig gewählt sind, denn die meisten Mikrocontroller haben eine obere Schaltschwelle im Bereich der halben U_B.

Doch wie kann der Taster in den Zuständen 3 und 4 abgefragt werden, wenn der I/O-Pin als Ausgang fungiert? Ganz einfach: Mehrmals pro Sekunde wird der Pin einfach sehr, sehr kurz als Eingang geschaltet und man hat für weniger als einen Wimpernschlag die Zustände 1 bzw. 2, was durch die Trägheit des menschlichen Auges nicht sichtbar ist. Falls dabei festgestellt wird, dass der Taster gedrückt ist, bleibt der Pin solange in Zustand 2, bis er wieder losgelassen wird, denn der Status 4 würde ja keinen Unterschied (die LED bleibt dunkel) machen, außer dass unnütz Strom fließt. Anschließend aber wird sofort wieder in den Zustand 3 geschaltet, damit die LED schön weiter leuchtet.

In die Firmware des Mikrocontrollers könnte man neben der grundlegenden Entprellung dann auch noch Luxus-Funktionen wie eine variable LED-Helligkeit implementieren, indem einfach entsprechend schnell zwischen den Zuständen 1 und 3 umgeschaltet wird. Der Fantasie sind hier keine Grenzen gesetzt.

Der Autor hat seine Lösung auf Basis eines PIC16F883 entwickelt [3]. Dieser Typ verfügt genau wie die bekannten AVR-Controller über aktivierbare interne Pull-up-Widerstände. Prinzipiell könnte man damit vor allem bei niedriger UB den Widerstand R2 ersetzen und so nur mit einem einzigen Widerstand auskommen. Leider aber haben diese Pull-ups durchaus Werte im Bereich 10...50 kΩ, was dazu führen kann, dass die LED im Status 1 schwach aber durchaus wahrnehmbar leuchtet. Die Firmware schaltet den Pull-up daher nur solange zu, wie für die Erfassung des Tasters nötig, damit dieser Effekt nicht stört.

Auf jeden Fall muss R2 so dimensioniert sein, dass die Schaltschwelle des Eingangs sicher überschritten wird, denn bei kleinen Strömen sinkt die Flussspannung U_f. Genau das kann bei einer roten LED und im 5-V-Betrieb problematisch werden. In diesem Fall hilft eine ordinäre Silizium-Diode in Serie mit der LED. Die Dimensionierung der Widerstände in Abhängigkeit von der Versorgung ist pro LED-Farbe in einer weiteren Tabelle angegeben. Bei anderen Strömen muss man etwas rechnen.

(130115)

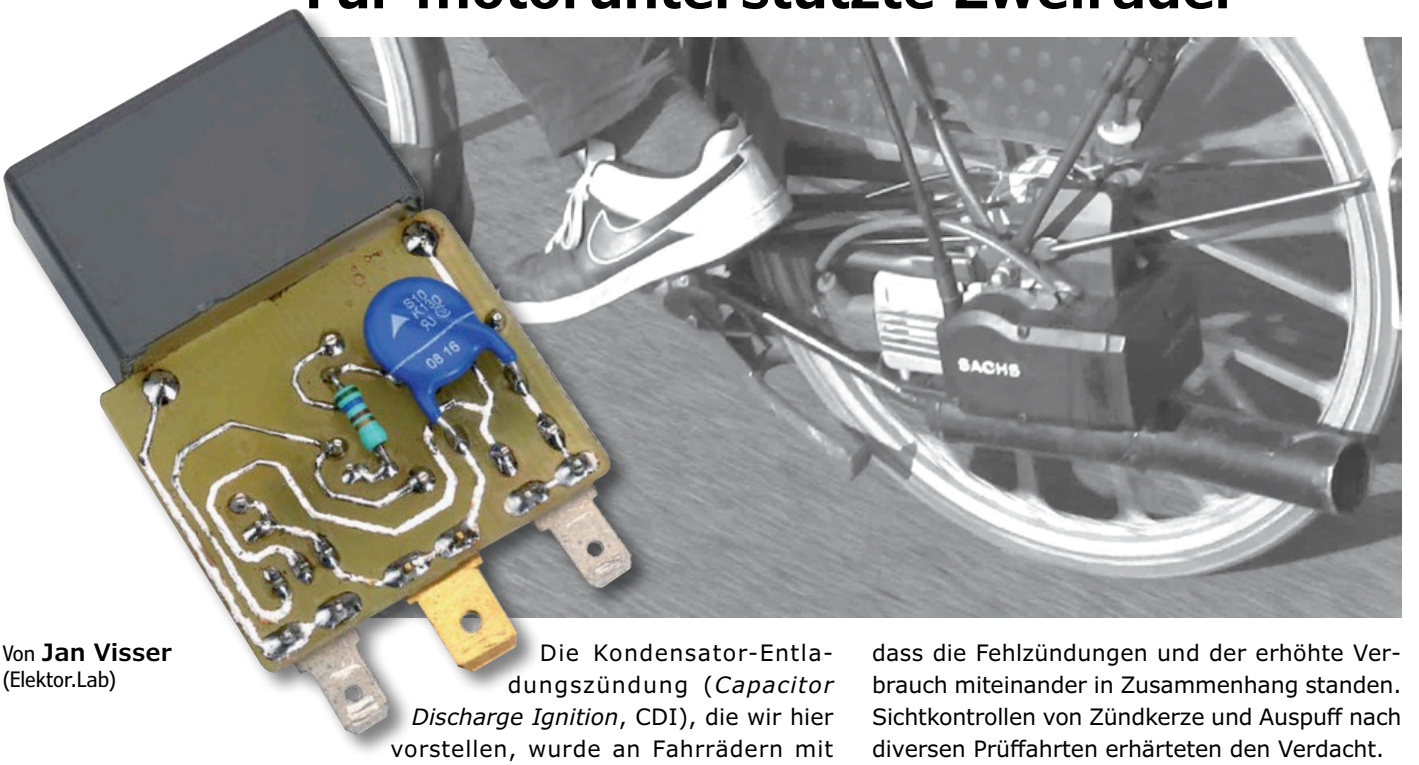
[1] Zweiadriges Interface für Taster mit LED, Elektor April 2012, S. 74, www.elektor.de/110572

[2] Zweiadriges Interface 2.0, Elektor Januar/Februar 2013, S. 105, www.elektor.de/120071

[3] Firmware: www.elektor.de/130115

Kondensator-Zündung

Für motorunterstützte Zweiräder



Von **Jan Visser**
(Elektor.Lab)

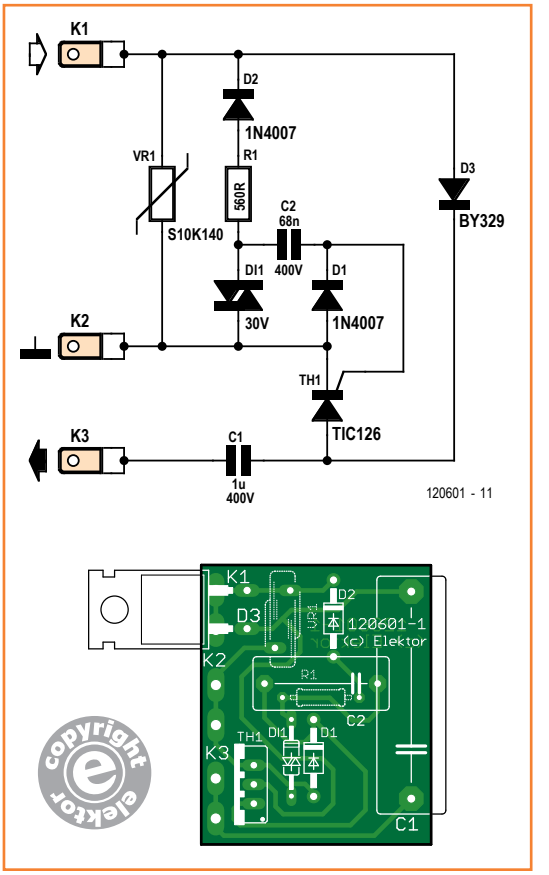
Die Kondensator-Entladungszündung (*Capacitor Discharge Ignition*, CDI), die wir hier vorstellen, wurde an Fahrrädern mit Hilfsmotor der Marken Spartamet und Saxonette erprobt. Ein Hilfsmotor, der den Radfahrer beim Treten in die Pedale unterstützt, ist zweifellos eine nützliche Einrichtung. Mein fahrbarer Untersatz des Herstellers Spartamet wurde von einem Motor angetrieben, der folgende Erscheinungen zeigte: Bei Vollgas und Höchstgeschwindigkeit (etwa 25 km/h) traten in der Zündung Funkenüberschläge auf, gleichzeitig stieg der Kraftstoffverbrauch überproportional an. Bei dreiviertel Gas reichte ein Liter für etwa 50 km Wegstrecke, bei Vollgas waren es nur noch 30 km. Die Vermutung lag nahe, dass die Fehlzündungen und der erhöhte Verbrauch miteinander in Zusammenhang standen. Sichtkontrollen von Zündkerze und Auspuff nach diversen Prüffahrten erhärteten den Verdacht.

Die Überschläge im Zündsystem sind möglicherweise gewollt, dies könnte eine Maßnahme des Herstellers sein, die Höchstdrehzahl des kleinen 30-cm³-Motors zu begrenzen. Allerdings wird nicht gleichzeitig der Vergaser abgeriegelt, er liefert nach wie vor ein Benzin-Luft-Gemisch, das zum wesentlichen Teil unverbrannt in den Auspuff gelangt. Dadurch tritt gleichzeitig der unerwünschte Nebeneffekt auf, dass der Auspuff deutlich schneller verkohlt.

Da das originale Zündsystem vollständig in Gießharz eingebettet war, konnte ich dort keine Modifikationen vornehmen. Deshalb habe ich eine Variante entworfen, die zum Original kompatibel ist, jedoch ohne Drehzahlbegrenzung arbeitet. Das Prinzip ist relativ einfach: Zu definierten Zeiten wird die in einem Kondensator gespeicherte Energie in eine Induktivität (Primärseite der Zündspule) transferiert.

Am Schaltungseingang liegt ein induktiver Sensor, der bei jeder Umdrehung des Motor-Schwungrads einen Impuls liefert. Der Schaltungsausgang ist mit der Zündspule verbunden, sie erzeugt aus den vom Kondensator kommenden Stromstößen Hochspannungsimpulse für die Zündkerze. Der Energiespeicher ist Kondensator C1, er wird über

Diode D3 geladen. Ein Impuls am Schaltungseingang triggert den Thyristor. Der Thyristor wird leitend, er legt C1 an Masse, so dass die Ladung über die Primärwicklung der Zündspule abfließen kann.
Für meine Zündelektronik habe ich eine einseitige Platine entworfen (Platinenlayout siehe [1]), zwei Bauelemente werden auf der Unterseite montiert. Das spart Platz, so dass diese Version kleinere Abmessungen als das Original hat. Die Fotos zeigen den nur 59 · 38 · 24 mm großen Musteraufbau.
Zuerst habe ich D1, D2, DI1 und C2 auf der Platinenoberseite montiert. Dann folgten D3 und Thyristor TH1, sie wurden flach umgebogen, so dass D3 über D2 und TH1 über D1 und DI1 lagen. Wegen seiner Größe fand MKP-Kondensator C1 seinen Platz neben der Platine. Die auf der Unterseite zu montierenden Bauelemente waren Varistor VR1 und Widerstand R1. Zum Schluss wurden noch die drei Flachstecker für die Anschlussleitungen montiert.
Ich habe die bestückte Platine in ein Gehäuse des Typs 001100 von Hammond eingebaut (Conrad 540830). Ein auf Maß angefertigtes Gehäuse, beispielsweise aus Acrylglas, ist eine gleichwertige Alternative.
Nachdem die Schaltung aufgebaut und angeschlossen war, habe ich mich davon überzeugt, dass an der Zündkerze tatsächlich Zündfunken auftraten (Vorsicht! Hochspannung!). Dann habe ich die Schaltung in das Gehäuse eingesetzt und mit Kunstharz vergossen. Ohne diesen Schutz würde die Zündelektronik den Witterungseinflüssen und Fahrzeugschwingungen nicht standhalten, sie würde über kurz oder lang defekt sein.
Fahrräder mit Hilfsmotor sind gegenwärtig überwiegend mit Zündsystemen von Motoplat (rot) oder Prüfrex (blau) ausgerüstet. Bei beiden Typen liegt der Masseanschluss in der Mitte von drei Kontakten. Falls der Eingang und Ausgang vertauscht werden, kann kein Zündfunke entstehen. Schaden nimmt die Zündung nicht, die äußeren Leitungen (rot und blau) müssen lediglich die Positionen wechseln.
Mein alternatives Zündsystem hat die Bewährungsprobe bestanden. Der Motor läuft auch bei Vollgas ruhig und ausgeglichen, Funkenüberschläge gehören der Vergangenheit an. Der Durst nach Kraftstoff ist spürbar zurückgegangen, jetzt sind mit einem Liter mühelos Distanzen von 70 km überbrückbar. Natürlich stößt die Leistung des 30-cm³-Motors schnell an ihre Grenzen. Eine wesentliche Steigerung der Höchstgeschwindigkeit habe ich nicht erwartet, trotzdem lag sie bei ungefähr 6 km/h. Der reduzierte Kraftstoffverbrauch, der ruhigere



Motorlauf und die höhere Geschwindigkeit sind die Pluspunkte meines alternativen Zündsystems.

(120601)gd

Weblink

[1] www.elektor.de/120601

Anschlüsse der Zündmodule

Motoplat: Schwarze Zündspule und rotes Modul
Prüfrex: Blau/grau/rote Zündspule und blaues Modul

Motoplat	Prüfrex
a = gelb	a = schwarz
b = blau	b = rot
c = rot	c = blau

Die Anschlussbezeichnungen befinden sich auf dem Zündmodul.
Beim Verpolen der roten und blauen Leitung ist die Zündung funktionslos.
Schäden entstehen nicht.

Stückliste

Widerstände:
R1= 560 Ω
VR1 = Varistor S10K140

Kondensatoren:
C1 = 1 µ/400 V MKP
C2 = 68 n/400 V MKS

Halbleiter:
D1,D2 = 1N4007
D3 = BY329
DI1 = Diac D30 (oder ER900, DB3)
TH1 = TIC126N

Außerdem:
3 Flachstecker 6,3 mm für Platinenmontage
Platine EPS 120601-1, siehe [1]

Einfacher Servotester

Grundausstattung für Modellbauer

Von **Bernhard Kaiser**
und **Michael Gaus**

Modellbauservos sieht man von außen in der Regel nicht an, wenn sie defekt sind. Deshalb ist ein kleines Testgerät für jeden Modellbauer ein absolutes Muss!

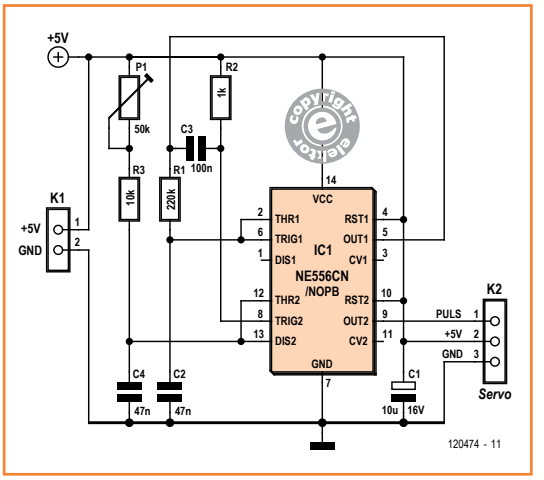
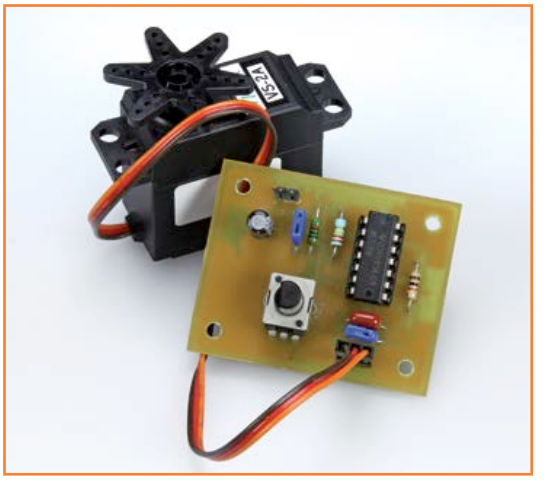


Bild 1. Die Mini-Schaltung des Servo-Testers mit einem Doppeltimer.



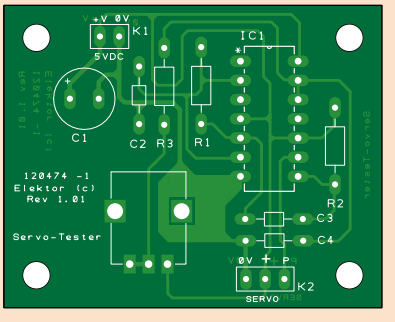
Servomotoren werden gerne im Modellbau eingesetzt, da sie klein, leicht und preiswert sind und auch ohne größeren Aufwand angesteuert werden können. Modellbauservos werden direkt am Funkempfänger angeschlossen. Üblicherweise besitzen Servos drei Anschlussleitungen: positive Versorgungsspannung (+5V), Masse (GND) und eine Steuerleitung (Puls), die die Position des Servoarms bestimmt. Das Signal auf dieser Leitung, das der Funkempfänger zur Verfügung stellt, ist pulsweitenmoduliert. Positive Impulse von etwa 1 ms Länge bedeuten linker Anschlag, von 2 ms Länge rechter Anschlag; logischerweise befindet sich der Servoarm bei 1,5 ms in Mittelstellung. Über die Impulslänge wird also der Winkel bestimmt, den der Servoarm einnimmt. Die Wiederholrate der Pulse beträgt circa 20 ms, es handelt sich also um ein 50-Hz-Signal. Diese Frequenz ist allerdings meistens nicht so kritisch. Wenn das Modell mal nicht so funktioniert, wie es soll, kann neben der Fernbedienung und dem Funkempfänger auch der Motor selbst die Ursache des Übels sein.

Mit dieser Schaltung lassen sich Modellbau-Servomotoren schnell und einfach auf ihre ordnungsgemäße Funktion prüfen und als Fehlerquelle identifizieren (oder eben nicht). Der dazu erforderliche Impulsgeber (**Bild 1**) gehört zu den Brot- und Butterschaltungen eines jeden Elektronikers.

Stückliste

Widerstände:
R1 = 220 k
R2 = 1 k
R3 = 10 k
P1 = Poti 50 k linear

Kondensatoren:
C1 = 10 µ, 16 V, RM7,5
C2,C4 = 47 n
C3 = 100 n



Halbleiter:
IC1 = NE556CN

Außerdem:
K1 = 2-polige Stiftleiste, RM 2,54 mm
K2 = 3-polige Stiftleiste, RM 2,54 mm
Platine 120474-1

DesignSpark-Projektfiles downloadbar unter [1].

Bild 2.
Die Schaltung kann auf dieser Platine bequem aufgebaut werden.

Zwei Timer

Der Impulsgeber ist mit einem Doppeltimer-IC NE556 aufgebaut, die Pulsbreite lässt sich mit einem Potentiometer einstellen. Timer1 im NE556 erzeugt über den Widerstand R1 sowie den Kondensator C2 die Wiederholrate des Servosteuersignals. Das Ausgangssignal an Pin 5 hat ein ungefähr symmetrisches Tastverhältnis. Über den Kondensator C3 wird bei der negativen Flanke der zweite Timer getriggert, der dann einen einzelnen positiven Puls am Ausgang Pin 9 erzeugt. Die Pulsbreite ist abhängig vom Kondensator C4 sowie dem Widerstandswert von R3 und P1. Über P1 kann also diese Pulsbreite verändert werden. Die Pulsbreite konnte im Testaufbau mit den angegebenen Bauteilwerten im Bereich von etwa 0,5...2,6 ms eingestellt, der übliche Bereich von 1,0...2,0 ms somit abgedeckt werden. Das Poti sollte jedoch nicht bis ganz an die beiden Anschläge verstellt werden, da ansonsten der Servo auf den mechanischen Anschlag gefahren wird, was seiner Lebenserwartung abträglich ist.

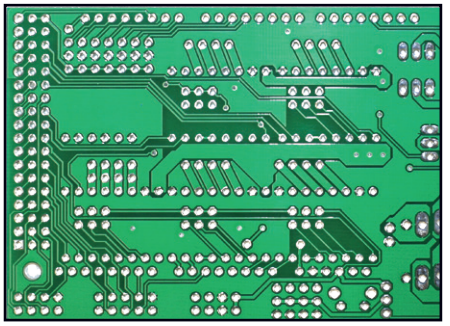
Vor dem Einschalten der Versorgungsspannung sollte sich das Poti etwa in Mittelstellung befinden. Die Wiederholrate der Pulse lag im Testaufbau bei ungefähr 18 ms.

Die meisten Modellbauservos arbeiten im Spannungsbereich von 4,8...6 V. Hier wurde als Versorgungsspannung 5...6 V gewählt, die durch Reihenschaltung von vier (Mignon-) Batteriezellen erreicht wird. Auch der Betrieb an vier NiMH-Akkus mit insgesamt 4,8 V ist problemlos möglich. Um Ihnen den Aufbau der Schaltung so bequem wie möglich zu machen, haben wir eine kleine Platine entworfen (**Bild 2**), die über Elektor [1] erhältlich ist. Die Bestückung sollte Ihnen keinerlei Probleme bereiten.

(120474)

Weblink

[1] www.elektor.de/120474

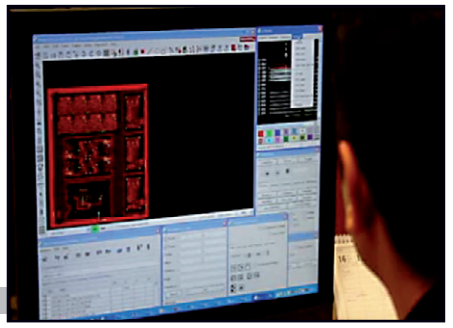


Professionelle Platinenfertigung

Wie eine vierlagige Platine entsteht

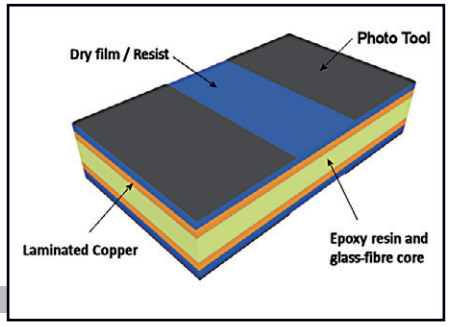
Die Technologie der professionellen Herstellung von Platinen hat sich seit der Veröffentlichung der ersten Layouts in Elektor vor über 50 Jahren enorm weiterentwickelt. Für diesen Artikel durften wir einen Blick in die Hexenküche unseres Elektor-PCB-Service-Partners Eurocircuits werfen.

Wenn man den Herstellungsprozess versteht, dann kann man Platinen entwerfen, die einfacher und preiswerter herzustellen sind sowie eine höhere langfristige Zuverlässigkeit aufweisen, sodass Ihre Kunden gerne wieder auf Ihre Produkte zurückgreifen. Von daher ist der folgende Einblick in die Herstellung einer „4-layer"-Platine nützlich und interessant zugleich.
Professionelle Platinenhersteller fertigen nicht etwa einzelne Platinen. Stattdessen werden etliche Platinen zum so genannten Nutzen (eine große Produktionsplatine) kombiniert. Solche einheitlich großformatigen Platinen sind sehr viel einfacher durch die Produktionsanlage zu schleusen und daher deutlich effizienter zu fertigen. Die Zusammenfassung unterschiedlicher Platinen zu einem Nutzen läuft häufig unter der Bezeichnung „Pool". Auch Eurocircuits produziert Platinen nach diesem rationellen Verfahren. In den Illustrationen kann man vier unterschiedliche Platinen erkennen, die zu einem Nutzen kombiniert wurden.



1

2



3

(1) Von Gerber- zu Produktionsdaten

Beim Platinen-Design erstellt man das Layout mit Hilfe eines spezialisierten CAD-Systems. Da jede Software auf ein eigenes, proprietäres Format setzt, hat sich die Platinen-Industrie ein standardisiertes Daten- bzw. Dateiformat geschaffen, damit die physikalischen Eigenschaften einer Platine einheitlich beschrieben werden können. Dieses Format nennt sich „Extended Gerber" oder „RS274X". Die Gerber-Dateien beschreiben die einzelnen Lagen bzw. Layer mit Leiterbahnen sowie die Lötmasken und die Positionen nebst Bezeichnungen der Bauteile.
Der erste Schritt ist daher, die vom Kunden gelieferten Dateien auf Kompatibilität zum Herstellungsverfahren zu überprüfen. Dies wird meistens automatisch erledigt. Es werden Breite und Abstand der Leiterbahnen, Pads um Bohrungen und die kleinsten Bohrdurchmesser auf Machbarkeit hin gecheckt. Nach dieser Überprüfung stellt ein Ingenieur die benötigten Dateien zusammen, mit denen die Maschinen für die Produktion und einen anschließenden Test gefüttert werden.

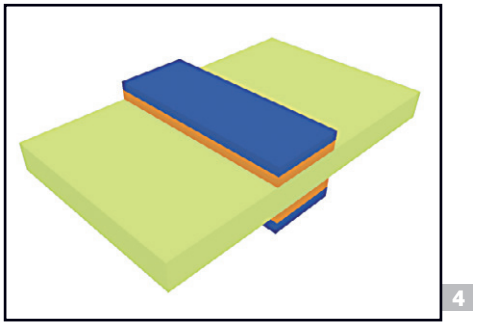
(2) Foto-Tools für die Bildübertragung

Ein Laser-Fotoplotter stellt die für die Produktion nötigen Filme her. Sie werden automatisch entwickelt und für die Produktion bereitgehalten. Alle Layer und Masken benötigen einen eigenen Film bzw. ein Foto-Tool. Die Filme werden mit Passer-Löchern versehen. Diese Löcher passen genau zu den Ausrichtungs-Pins des Belichters und stellen sicher, dass alle Layer genau zueinander ausgerichtet sind.

(3) Innere Layer

Die Produktion der inneren Lagen einer Multilayer-Platine startet bei Herstellern wie Eurocircuits mit einem Laminat aus Epoxid mit Glasfaserkern, das von beiden Seiten mit einer Kupferfolie beschichtet ist. Zuerst wird das Kupfer gereinigt und das Laminat kommt dann in einen Reinraum, damit kein Staub auf die Oberflächen gelangen kann, der Unterbrechungen oder Kurzschlüsse verursachen könnte. Dort werden die Oberflächen mit einem fotoempfindlichen Film (Photoresist) beschichtet. Als nächste Stufe wird das Bild des Films durch Belichtung mit starken UV-Lampen auf die Fotoschicht dieser „inneren Platine“ übertragen. Das Bett dieses „UV-Druckers“ verfügt über Positionierstifte für die genaue Ausrichtung von Film(en) und Platine. Hierzu werden ein Film, dann die Platine und schließlich der zweite Film manuell positioniert.

Nach der Belichtung wird die Platine mit einer alkalischen Lösung eingesprüht, um die nicht belichtete Fotoschicht zu entfernen. Daraufhin wird die Platine mit Druck gewaschen und getrocknet. Die Leiterbahnen sind mit der belichteten Fotoschicht bedeckt. Nun wird die Platine manuell inspiziert, um sicher zu stellen, dass die Kupferoberfläche sauber ist und alle unbelichteten Reste der Fotoschicht entfernt wurden.


4

(4) Ätzen der inneren Layer

Jetzt kommt die Platine in ein alkalisches Ätzbad, um das nicht abgedeckte Kupfer zu entfernen. Dieser Prozess wird genau überwacht, damit die definierten Leiterbahnbreiten eingehalten werden.

Anschließend werden die blauen Reste des Photoresist über den Leiterbahnen entfernt. Wieder wird manuell geprüft, ob dies vollständig gelungen ist.


4

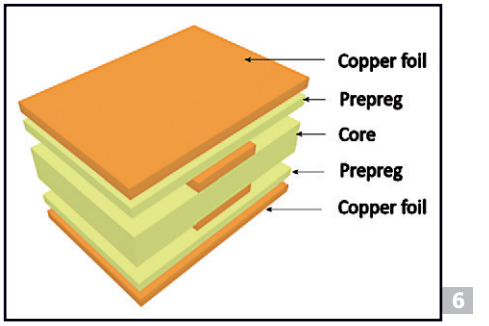
(5) Ausrichtung und Inspektion der inneren Layer

Der innere Kern der Multilayer-Platine ist nun fertig. Nun erhält sie noch Passer-Löcher um die äußeren Layer daran ausrichten zu können. Diese Ausrichtung muss sehr genau erfolgen, denn nach dem Aufbringen der äußeren Schichten kann man nichts mehr ändern. Ein automatisches System überwacht die Platine optisch und vergleicht sie mit einem digitalen Bild, das aus den Layout-Daten errechnet wurde.


5

(6) Innere und äußere Layer verkleben

Die beiden äußeren Layer bestehen aus je einer mit noch nicht ausgehärtetem Epoxid imprägnierten Glasfaserlage (dem „Prepreg“) und je einer Kupferfolie. Zunächst kommen eine Kupferfolie und zwei Lagen Prepreg auf eine Stahlplatte. Darauf kommt dann sauber ausgerichtet die Platine mit den inneren Layern. Dann kommen wieder zwei Lagen Prepreg und eine Kupferfolie. Zum Schluss wird eine Aluplatte aufgelegt und das Ganze unter Erwärmung verpresst. Dieser Vorgang wird von einem Computer gesteuert, damit eine langlebige Verbindung der Schichten der Platine gewährleistet ist.



6

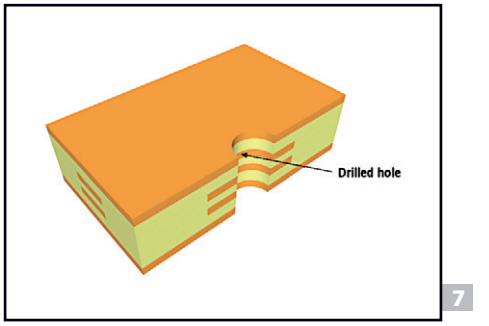
(7) Bohren

Referenz-Löcher

Vor dem Ätzen der äußeren Layer werden zunächst alle Löcher für die bedrahteten Bauelemente und die Durchkontaktierungen (Vias) gebohrt. Eine mit Röntgen-Technik ausgerüstete Bohrmaschine bestimmt hierzu die Bohrpositionen der inneren Layer. Die Maschine bohrt zunächst wieder Passer-Löcher zur Ausrichtung, damit genau durch die Mitte der inneren Layer-Pads gebohrt werden kann.

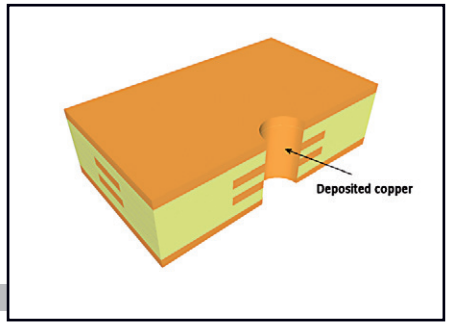
Bohrvorbereitung

Zur Vorbereitung kommt zunächst eine Lage Wegwerfmaterial auf das Bett, damit die untere Kupferlage beim Bohren nicht ausfranst. Dann wird die Platine und eine Lage Alufolie als oberste Schicht aufgelegt.



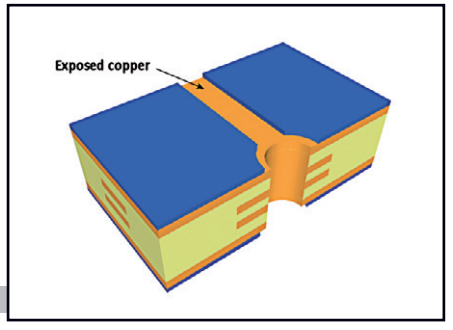
7

7



8

8



9

9

Bohren

Ein passendes Bohrprogramm steuert den Bohrer einer CNC-Bohrmaschine zu den richtigen XY-Koordinaten. Die Spindel der Maschine wird mit Luftantrieb auf bis zu 150.000 rpm gebracht. Hochgeschwindigkeitsbohren sorgt für glatte Bohrwandungen, was wichtig für eine sichere Beschichtung ist.

Bohren ist ein langsamer Prozess, da jedes Loch einzeln gebohrt werden muss. Von daher werden – abhängig vom Bohrdurchmesser – bis zu drei Platinen auf einmal gebohrt. Die Maschine holt sich die gewünschten Bohrer selbst aus der Halterung, überprüft ihren Durchmesser und setzt sie in das Bohrfutter ein.

Epoxid entfernen

Beim Verpressen tritt an den Kanten überschüssiges Epoxidharz aus. Dies wird von einer CNC-Fräsmaschine wieder entfernt. Anschließend ist die Platine bereit für das Beschichten.

(8) Beschichtung – Teil 1

Zunächst werden die Löcherwände leitfähig beschichtet. Hierzu wird die Platine in einen Rahmen eingespannt, um sie durch eine Folge von chemischen und reinigenden Bädern zu schleusen, wodurch die Lochwände zunächst mit Mikropartikeln aus Palladium versehen und darauf dann eine 1 µm starke Kupferschicht abgeschieden wird. Das noch fehlende Kupfer wird jetzt galvanisch auf die leitende erste Schicht aufgebracht. Doch vorher...

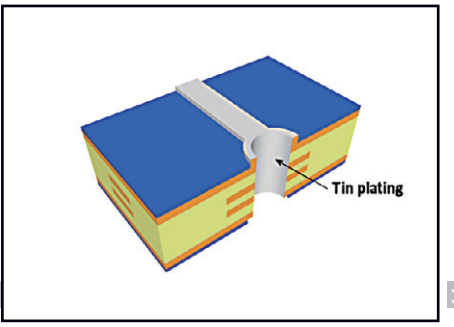
(9) Äußere Layer

... kommt die Rohplatine noch einmal in den Reinraum, wo sie wieder mit Photoresist beschichtet wird. Diese Schichten werden beidseitig auf die Kupferoberflächen thermisch aufgerollt und laminiert. Dann wird der Film mit Hilfe der Pins genau ausgerichtet und die laminierte Platine sowie der zweite Film kommen obenauf. Nach Belichtung und Abziehen des Polyesterfilms zum Schutz des Photoresists wird die nichtbelichtete Fotoschicht entfernt. Nun sind die Leiterbahnen der äußeren Layer freigelegt. Die Platine wird dann nochmals auf saubere Kupferoberflächen und vollständige Entfernung von Photoresist-Resten geprüft.

(10) Beschichtung – Teil 2

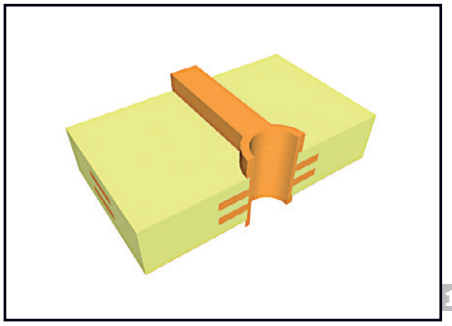
Nun wird die Platine noch galvanisch verkupfert. Hierzu wird die automatische Galvanisierung gestartet, wo die Kupferoberflächen durch einige Bäder zunächst gereinigt, aktiviert und schließlich verkupfert werden. Auch hier sorgt eine Computer-Steuerung dafür, dass jede Platine stets die richtige Zeit in den Bädern verbringt.

Damit die Löcher eine gute Leitfähigkeit erreichen, ist an ihren Wänden eine Kupferschicht von etwa 25 µm erforderlich. Die Galvanisierung wirkt sich überall aus und daher scheiden sich die 25...30 µm auch auf den Leiterbahnen der äußeren Layer ab. Da hier eine Kupferfolie mit 17,5 µm Stärke aufgebracht wurde, sind die Leiter-

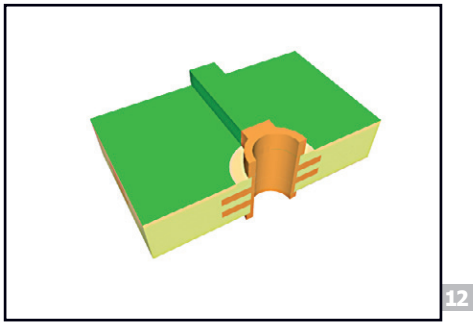


10

10


11


11


12

bahnen anschließend etwa 40...42 µm dick.
Damit abschließend das überflüssige Kupfer außen weggeätzt werden kann, wird auf den Leiterbahnen zunächst noch eine dünne Zinnschicht aufgebracht, und die Platine auf die richtige Stärke von Kupfer- und Zinnschicht geprüft.

(11) Äußere Layer ätzen

Nun zum Ätzen der äußeren Layer: Zunächst muss die Restschicht Photoresist beseitigt werden, die das überflüssige Kupfer bedeckt. Anschließend wird dieses Kupfer mit alkalischer Lösung entfernt. Bei diesem Prozess wird genau darauf geachtet, dass die Leiterbahnen nicht seitlich unterätzt werden und deren Breite stimmt. Dann wird noch die schützende Zinnschicht entfernt.

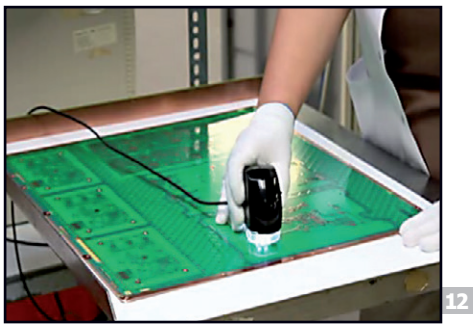
(12) Lötstoppmaske

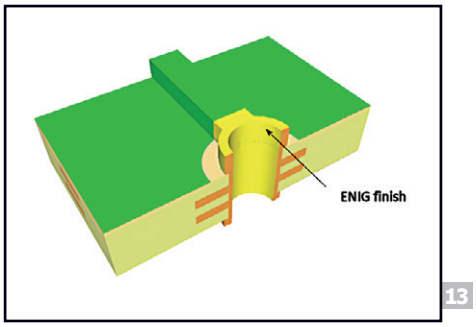
Vor dem Aufbringen der Lötstoppmaske wird die Platine zunächst gereinigt und gebürstet, um ein eventuelles oberflächliches Anlaufen zu beseitigen. Die Platine wird vertikal aufgehängt und beide Seiten werden gleichzeitig mit einer Epoxidschicht kaschiert. Die Platine wird dann eingespannt und durch einen Trockner befördert, wodurch die Maske belastbar genug für das Bedrucken wird. Nun wird die Platine auf vollständige und gleichmäßige Beschichtung geprüft.
Die beschichteten Platinen kommen nochmals in den UV-Belichter. Filme und Platine werden ein weiteres Mal genau ausgerichtet. Wie zuvor bei den Lagen mit Leiterbahnen härtet das Epoxid da aus, wo der Film klar ist. Das sind die Stellen, wo die Lötstoppmaske auf der Platine verbleibt.
Die belichtete Platine wird dann aus dem Reinraum in den Entwickler transportiert, der die nicht ausgehärtete Epoxidschicht entfernt. Jetzt wird nochmals überprüft, ob die Lötstoppmaske genau ausgerichtet war und keine Reste davon auf den Leiterbahnen, Löt-Pads oder in den Löchern vorhanden sind. Zur weiteren Verbesserung der Stabilität der Maske wird die Platine nochmals thermisch getrocknet.

(13) Pads und Löcher

Pads und Löcher für Bauteile sind nicht mit dem Lötstoppfilm bedeckt. Nun kommt hierauf eine lötfähige Oberflächenbeschichtung, damit das Kupfer an den Stellen geschützt ist, wo später gelötet wird. Im Bild ist hier eine Goldschicht zu sehen, wobei hierfür zuvor chemisch eine Schicht von 5 µm Nickel abgeschieden wird und dann auf diese noch 0,1 µm Gold kommt.

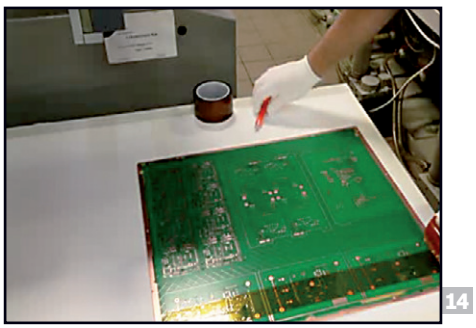
Nach der EU-Richtlinie zu gefährlichen Substanzen (RoHS) darf kein Blei mehr zur Beschichtung verwendet werden. Von daher wird nur noch eine Gold/Nickel-Beschichtung sowie Silber oder die bleifreie Heißluftverzinnung angeboten. Für Letzteres kommt die Platine in ein Bad aus geschmolzenem Zinn. Nach dem Auftauchen bläst heiße Luft das überschüssige Zinn von der Platine und übrig bleibt ein etwa 2 µm dicker Film.

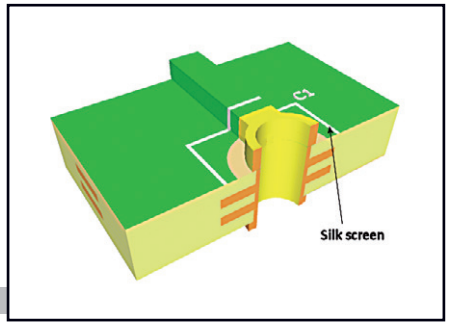

12



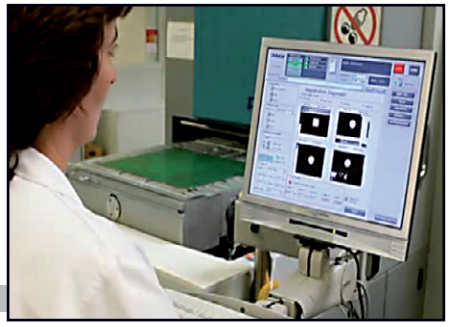
13

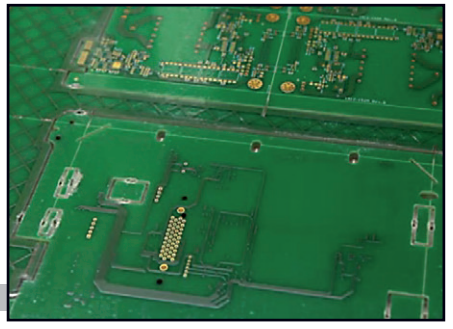

13

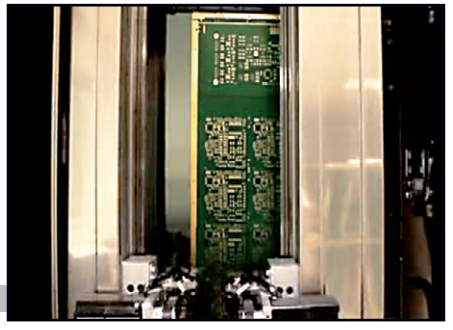

14

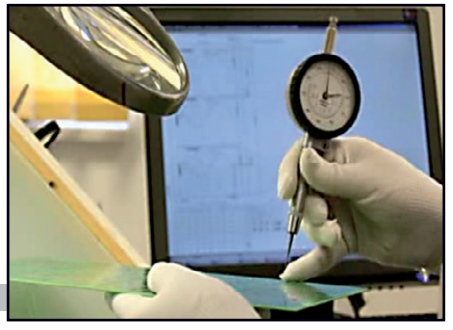


15


15


16


17


18

(14) Hartvergoldung

Für die Kontaktflächen von Platinensteckern (wie beim ISA-Bus) ist eine galvanische Vergoldung notwendig. Zunächst wird die Platine außerhalb dieser Kontaktflächen abgeklebt und diese dann auf einem horizontalen galvanischen Bad positioniert. Nun wird auf eine Nickelschicht von 4...5 µm noch eine Goldschicht von 1...1,5 µm aufgebracht.

(15) Bestückungsdruck

Der Bestückungsdruck (silk screen) wird direkt mit einem Spezial-Tintenstrahldrucker auf die Platine aufgebracht. Wie bei einem gewöhnlichen Tintenstrahldrucker werden hier kleine Tröpfchen mit Tinte aufgesprüht. Nun werden Lötstoppmaske und Bestückungsdruck final ausgehärtet. Dieser Prozess in einem fünfstufigen Trocknungsofen dauert zehn Minuten.

(16) Vereinzeln des Nutzens

Die Gesamtplatine ist nun bereit für die Auftrennung in die einzelnen Platinen. Zuvor bringt eine CNC-Fräsmaschine die erforderlichen kleinen Löcher und Ausschnitte in der Platine an. Dann wird mit einem 2-mm-Schneidfräser ein neues Werkzeug gewählt; dieser schneidet die einzelnen Platinen aus dem Nutzen aus.

(17) Elektrischer Test

Am Ende der Platinenfertigung muss jede Multilayer-Platine mit Hilfe der Layout-Daten elektrisch getestet werden. Ein Testgerät überprüft, ob keine Leiterbahn unterbrochen ist und auch keine Kurzschlüsse vorhanden sind.
Schneller geht das mit einer optionalen Acceler8-Maschine. Sie hat 4.000 kleine Prüfspitzen, fast wie eine Bürste. Zuerst wird aus einer schon getesteten Platine eine Art elektronischer Landkarte berechnet und diese dann mit den restlichen Platinen verglichen. Auf diese Weise wird die Testzeit um gut 90 % reduziert.

(18) Abschlussprüfung

Zum Abschluss des kompletten Prozesses werden Mitarbeiter mit Adleraugen auf die Platinen losgelassen. Wenn alles in Ordnung ist, wird ein Versionshinweis gedruckt. Die Platinen werden dann vakuumverpackt, um Schmutz und Feuchtigkeit fernzuhalten. Nun folgen Luftpolsterfolie und die Verpackung. Schließlich wird das Paket mit Platinen zum Kunden verschickt.

Nun wissen Sie, wie die Platinen produziert werden, die Sie über den Elektor-PCB-Service bestellen können (www.elektorpcbservice.com). In der nächsten Folge geht es darum, was man als Entwickler beim Platinen-Layout beachten sollte.

(130061)

Illustrationen mit freundlicher Unterstützung von Eurocircuits.

Weblink

www.elektorpcbservice.com

Eurocircuits ist ein europäischer Platinenhersteller. Die Hauptniederlassung befindet sich in der bildhübschen belgischen Stadt Mechelen und die Produktion findet nahe Aachen (Deutschland) und in Eger (Ungarn) statt. Eurocircuits hat sich auf die Fertigung von Prototypen und Kleinserien für Entwickler, Entwicklungsabteilungen, kleine Elektronik-Firmen in Nischenmärkten, Universitäten und Forschungseinrichtungen spezialisiert.

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

NEU

LabVIEW 2

Arrays und serielle Daten

Der zweite Band der LabVIEW-Lehrbuchreihe beschäftigt sich u. a. mit Arrays, Cluster und den seriellen VISA-Funktionen. Als Erstes werden vier neue zusammengesetzte Datentypen (Enum, Ring, Array, Cluster) vorgestellt und deren Verwendung wird anhand zahlreicher praktischer Beispiele und Übungen erläutert.

Danach wird es praktisch: Ein 8051er-Mikrocontrollersystem dient als Datenquelle und -senke für verschiedene LabVIEW-VIs, die einerseits beliebige Messdaten empfangen, auswerten und die Ergebnisse numerisch oder graphisch auf dem Rechner-Monitor darstellen. Anderseits können sie auch Steueranweisungen an das Mikrocontrollersystem ausgeben.
Der gesamte Datentransfer zwischen Mikrocontroller und LabVIEW-VI läuft dabei über serielle COM-Schnittstellen oder mittels USB/Seriell-Adapter über USB-Ports.

Da alle VIs universell aufgebaut und ausführlich erklärt werden, lassen sie sich problemlos mit anderen Mikrocontrollerboards (AVR-, PIC-, ARM-, Arduino-, Raspberry PI-Systeme, E-blocks) kombinieren, sofern diese über eine serielle Schnittstelle verfügen.

ca. 250 Seiten (kart.) · Format 17 x 23,5 cm · ISBN 978-3-89576-274-1
€ 34,80 · CHF 43,20

elektor **Weitere Infos & Bestellung unter www.elektor.de/labview2**

Um noch einmal darauf zurück zu kommen…

LCR-Meter 2013

Höchste Präzision zu niedrigen Kosten

Von
Jean-Jacques Aubry,
Ollioules

Viel positives Echo

Nach Erscheinen des LCR-Meter 2013 war die Resonanz unerwartet hoch, viele Leser fanden Worte der Begeisterung. Das beweist einmal mehr, dass der Selbstbau von Messgeräten hoch im Kurs steht. Autor Jean-Jacques Aubry hat auf seinem französischsprachigen Forum [4] einige Korrekturen veröffentlicht, die wir auch an dieser Stelle weitergeben wollen. Zuerst geht es um die Bemaßungen der Bohrungen für SW1 und den USB-Anschluss in der Seitenfläche. Der Abstand muss nicht 1,16 mm, sondern 0,96 mm betragen. Der Durchbruch für J19, das ist die USB-Buchse, muss nicht 8,03 · 3,97 mm, sondern 5,39 · 6,61 mm sein. Wir haben die korrigierte Bohrschablone ebenso wie die aktualisierte Schaltung auf der Projektseite zum Download bereitgestellt. In der Schaltung wurden nur wenige unwesentliche Änderungen vorgenommen.

Auch die Stückliste der Hauptplatine bedarf einiger kleiner Korrekturen: Statt J17 muss es J9 heißen. Widerstand R81 kommt zweifach vor, fehlerhaft ist R81 = 10 kΩ, der korrekte Wert ist R81 = 7,5 kΩ.

In der Schaltung der Hauptplatine muss R31 = 750 Ω sein, und für die Schaltung der Display-Platine gilt R8 = 1 kΩ, dies ist der Widerstand vor LED D5. Ferner ist Kondensator C30 = 1,5 nF, 5 % NP0.

Die Korrekturen haben keine Auswirkungen auf die Tauglichkeit der Schaltung. Die Platine und das aufgebaut lieferbare Modul bleiben unverändert, denn alles arbeitet nach Wunsch. Nach Erscheinen der dritten Beitragsfolge waren bis zu dem Tag, an dem diese Ergänzung verfasst wurde (21. Mai 2013), keine Updates nötig.

Nur nebenbei bemerkt: Manchmal haben wir auch einen Blick für Kleinigkeiten: Im Download-Dokument *First time Setup* wird die Firmware-Datei fälschlicherweise mit *LCR3A_update_Vxxx.hex* bezeichnet. Das muss natürlich *LCR3A_firmware_Vxxx.hex* heißen.

Einige Aspekte des LCR-Meters sind im theoretischen Teil des Beitrags [1] zu kurz gekommen, weil wir ihn nicht notlos in die Länge ziehen wollten. Deshalb halten wir es für angebracht, bestimmte Facetten noch etwas näher zu betrachten, zumal dies auch unabhängig vom Projekt nützlich sein kann. Ein Beispiel ist eine Routine zum Darstellen von Zeichen und Symbolen, die etwas Geschick auf der Seite der Software erfordert.

Die Fragen von Lesern betrafen häufig die Messgenauigkeit sowie potentielle Quellen für Ungenauigkeiten und Messfehler. Auch darauf wollen wir nachfolgend eingehen. Wir setzen voraus, dass Ihnen der Inhalt des Beitrags vertraut ist und die Schaltung sowie das Messprinzip vorliegen.

Neues für das Display

Der Bootloader und die Firmware setzen Meldungen und Symbole auf das grafische Display (GLCD). Das RAM des Displays ist für 128 Spalten zu 8 Byte eingerichtet, was 128 · 64 Pixeln entspricht. Die darzustellenden Zeichen (Buchstaben, Ziffern, Symbole) sind jedoch höher als 8 Pixel, so dass sie vertikal mehr als ein Byte belegen. Einige Zeichen haben sogar die Höhe 11 oder 16 Pixel. Sämtliche Elemente werden in der Datei *glcd_bitmaps.c* oder in der Kurzversion *bootloader_glcd-bitmaps.c* definiert. In der Kurzversion fehlen die Zeichen und Symbole, die nur von der Firmware verwendet werden.

Um ein Zeichen oder Symbol darzustellen, müs-

sen die zugehörigen Bits auf zwei oder mehr aneinander grenzende Bytes verteilt werden. Schreiboperationen in das Display-RAM sind jedoch nur in ganzen Bytes möglich, denn jedes Byte hat nur eine Adresse. Deshalb muss vor dem Schreiben bekannt sein, welche aktuellen Inhalte die Bytes im RAM haben. Dann ist die Differenz zwischen aktuellen und neuen Inhalten zu berechnen, so dass nicht mehr Pixel als notwendig geändert werden.
Wegen der begrenzten Anzahl der Portleitungen muss das Display seriell gesteuert werden, doch in dieser Betriebsart ist das Lesen des Display-RAMs nicht möglich. Das bedeutet, dass eine Kopie (*mirror*) des Display-RAMs im RAM des Mikrocontrollers angelegt werden muss:

```
uchar xdata GLCD_Array[LCD_COLS]
[LCD_ROWS];
```

Das Umrechnen der alten in die neuen Display-RAM-Inhalte ist fast unmöglich, denn es sind eine Vielzahl von Konstellationen zu berücksichtigen. Dieses Problem lässt sich dadurch lösen, dass das 8-bit-Format der Spalten in ein 64-bit-Format konvertiert wird. Dazu ist folgendes notwendig:

- Anlegen einer Kopie im RAM des Mikrocontrollers, wobei jedem Pixel (Wert nur 0 oder 1) ein ganzes Byte zugeordnet wird:

```
uchar xdata Column_Array[(LCD_ROWS + 1) * 8]; // + 1 for 2nd byte of char in last
line
```

- Erstellen einer Routine, die aus der Kopie spaltenweise liest, in die zugehörigen Bits umsetzt und in Column_Array[] speichert:

```
void GLCD_read_column(uchar col)
{
     uchar  rows, pix, i, j;
     i = 0;
     // read column col, byte after byte
     for (rows = 0; rows < LCD_ROWS; rows++)
     {
            pix = GLCD_Array[col][rows];
            // write pix, bit after bit
            for (j = 0; j < 8; j++)
            {
                   Column_Array[i] = pix & 0x01;
                   pix >>= 1;
                   i++;
            }
     }
}
```

- Erstellen einer Routine, die den Inhalt von Column_Array[] liest, in das Display-RAM schreibt und die Kopie aktualisiert:

```
void GLCD_write_column(uchar col)
{
     uchar  rows, pix, i, j;
     i = 0;
     // write column col, byte after byte
     for (rows = 0; rows < LCD_ROWS; rows++)
     {
            pix = 0;
            // read pix, bit after bit
            for (j = 0; j < 8; j++)
            {
                   pix += Column_Array[i] << j;
                   i++;
            }
            if (GLCD_Array[col][rows] != pix)     // only if GLCD RAM byte modified
            {
```

```
                    GLCD_set_pos(rows, col);
                    GLCD_WriteData(pix);          // draw in GLCD RAM
                    GLCD_Array[col][rows] = pix;  // update RAM mirror
                }
        }
}
```

- Erstellen einer Routine, die ein Zeichen oder Symbol als Bitmap definiert, und zwar für beliebige Positionen auf dem Display:

```
void GLCD_show_icon(uchar code *bitmap, uchar width, uchar height, uchar x, uchar
y, uchar mode)
{
     uchar  tx, ty, pix, hb, i, j, k;
     hb = (height - 1) / 8 + 1; // character height in bytes
     for (tx = 0; tx < width; tx++) // loop for width columns
     {
             GLCD_read_column(tx + x);
             i = y;
             k = 0;
             for (ty = 0; ty < hb; ty++) // read hb bytes of icon
             {
                     pix = *(bitmap + ty * width + tx); //  read one byte
                     if (mode == GLCD_PIXEL_OFF)
                             pix = ~pix;
                     for (j = 0; j < 8; j++) // write 8 bits of pix to Column_Array
                     {
                             if (mode != GLCD_PIXEL_INV)
                                     Column_Array[i] = pix & 0x01;
                             else
                                     Column_Array[i] ^= pix & 0x01;
                             pix >>= 1;
                             if (k == height)
                                     break;
                             i++;
                             k++;
                     }
             }
             GLCD_write_column(tx + x);
     }
}
```

- Was noch fehlt, ist eine Schreibroutine, die eine Zeichenkette in einer bestimmten Schriftart auf das Display setzt:

```
void GLCD_draw_text( uchar x, uchar y, uchar *text , uchar mode )
{
     uchar i, posx, posy;
     uchar *pt;
     posy = y - font_height + 1;
     for( pt = text, i = 0; *pt; i++, pt++ )
     {
               posx = x + i * font_width;
               if( posx + font_width > LCD_COLS )
               {
                     i = 0;
                     posx = x;
                     posy += font_height;
               }
               GLCD_show_icon( font + (*pt - font_offset) * font_charsize, font_
width, font_height, posx, posy, mode );
     }

}
```

Messgenauigkeiten

Die zu messende Impedanz ist:

$$Z_x = \frac{V_p + jV_q}{I_p + jI_q} \times \frac{G_i\ R_{sense}}{G_v} = \left[\frac{V_p\ I_p + V_q\ I_q}{I_{p^2} + I_{q^2}} + j\frac{V_q\ I_p - V_p\ I_q}{I_{p^2} + I_{q^2}}\right] \times \frac{G_i\ R_{sense}}{G_v}$$

Mit *Gi* und *Gv* als Verstärkung (*Gain*) von Strom und Spannung in der Verstärkerkette und R_{sense} als Widerstand von *IU_converter*:

$$G\,i = G_{INA128} \times G_{BUFFER} \times G_{PGA}i \times G_{DAC}i \qquad G\,v = G_{INA128} \times G_{BUFFER} \times G_{PGA}v \times G_{DAC}v$$

Impedanzen werden meistens wie folgt geschrieben: $Z = R_S + j\ X_S$ Darin sind:

$$R_S = \frac{V_p\ I_p + V_q\ I_q}{I_{p^2} + I_{q^2}} \times \frac{G_{PGA}i\ \ G_{DAC}i\ \ R_{sense}}{G_{PGA}v\ \ G_{DAC}v} \quad \text{und} \quad X_S = \frac{V_q\ I_p + V_p\ I_q}{I_{p^2} + I_{q^2}} \times \frac{G_{PGA}i\ \ G_{DAC}i\ \ R_{sense}}{G_{PGA}v\ \ G_{DAC}v}$$

Wenn der durch die Digitalisierung verursachte Fehler vernachlässigt wird (wie unter *Auf die Verstärkung kommt es an* im ersten Teil [1] beschrieben), dann ist der Messfehler:

$$\frac{\Delta R_S}{R_S} = \frac{\Delta X_S}{X_S} = \left[\frac{\Delta G_{PGA}i}{G_{PGA}i} + \frac{\Delta G_{PGA}v}{G_{PGA}v}\right]_{i \neq v} + \left[\frac{\Delta G_{DAC}i}{G_{DAC}i} + \frac{\Delta G_{DAC}v}{G_{DAC}v}\right]_{i \neq v} + \frac{\Delta R_{sense}}{R_{sense}}$$

Der Gesamtfehler besteht aus zwei Komponenten:
- Fehler infolge der Ungenauigkeit der Verstärkungen des PGA103 und des DAC8811C,
- Abweichung vom tatsächlichen Wert der Widerstände von *IU_converter* (Kalibrierfehler).

In die zweite Komponente gehen folgende Fehler ein:
- Fehler der Anzeige, ±1 bit der letzten Ziffer,
- Rest-Phasenfehler nach Kompensation des Phasenfehlers,
- Fluktuationen durch Verstärkerrauschen sowie durch Störsignale, die von den Messkabeln aufgefangen werden (Netzbrummen und andere Störsignale).

Außerdem tritt in den Messbereichen 1 und 8 ein Digitalisierungsfehler auf, falls die Amplituden von Strom und Spannung stark unterschiedlich sind. Eine besser geeignete Verstärkung ist nicht verfügbar.

Ungenauigkeit

***Verstärkungsfehler* (gain error)**

Burr-Brown (Texas Instruments) gibt in der Dokumentation zum **PGA103** folgendes an:

Gain	1	10	100
Typical gain error	±0,005%	±0,02%	±0,05%
Max gain error	± 0,02%	±0,05%	±0,2%

In den Messbereichen 3...6 hat der PGA103 die Verstärkung 1, so dass der Ausdruck

$$\left[\frac{\Delta G_{PGA}i}{G_{PGA}i} + \frac{\Delta G_{PGA}v}{G_{PGA}v}\right]$$

gleich 0 ist.

Für Messbereich 2 und 7 beträgt der maximale Fehler ±0,07 % (typischer Wert ±0,025 %), für Messbereich 1 und 8 beträgt der maximale Fehler ±0,22 % (typischer Wert ±0,055 %).

Texas Instruments beziffert in der Dokumentation des **DAC8811C** die maximale relative Genauigkeit mit ±1 LSB. Der maximale Verstärkungsfehler ist dann ±1/N, wobei N der Wert ist, der für die Verstärkung Unendlich gilt.

Stufe post_Ampli.	0	1	2	3	4	5	6	7
N	**7 500**	**8 700**	**10 000**	**11 600**	**13 500**	**15 500**	**18 000**	**20 700**
Stufe post_Ampli.	8	9	A	B	C	D	E	F
N	**24 000**	**27 600**	**32 000**	**36 900**	**42 600**	**49 100**	**56 700**	**65 500**

Wenn die Stufen *post_amplification_U* und *post_amplification_I* gleich sind, ist der Ausdruck

$$\left[\frac{\Delta G_{DAC}i}{G_{DAC}i}+\frac{\Delta G_{DAC}v}{G_{DAC}v}\right]$$ gleich 0.

Bei Ungleichheit tritt das Maximum dann auf, wenn eine Stufe 1 und die andere 0 ist.

In dem Fall folgt: $\frac{1}{7500}+\frac{1}{8700}$ gleich 0,025%.

Anmerkung: Für die B-Version des DAC (**DAC8811B**) verdoppelt sich dieser Fehler.

Die offene Schleifenverstärkung von *IU_converter* ist nicht Unendlich, auch daraus entsteht ein Messfehler. Die geschlossene Schleifenverstärkung ist ≤1 in den Messbereichen 3...6. Bei 10 kHz beträgt die offene Schleifenverstärkung ungefähr 80 dB (=10000), so dass ein Messfehler von etwa 0,01 % entsteht. Dieser Messfehler kann bei niedrigen Frequenzen vernachlässigt werden.

Anmerkung: Auf dem PC ist der Verstärkungsfehler mit dem Programm AU2011 darstellbar.

Phasenfehler

Der Verstärker *IU_converter* ist breitbandig (50 MHz!), und außerdem sind verschiedene Phasenkompensationen vorhanden (beschrieben in *First Time Setup* des Downloads [3]). Deshalb ist der Phasenfehler zwar minimal, er lässt sich aber nicht vollständig eliminieren. Die Phase der Endstufe ist praktisch unabhängig von der Verstärkung, doch dies gilt nur für die erste Stufe. Dann folgt der DAC8811C, der mit Daten im Bereich 0x2000...0xFFF ein fast gradliniges Durchlassband aufweist (etwa 8 MHz).
Der Restfehler wirkt sich geringfügig auf die sekundären Parameter aus, er dürfte noch etwas niedriger als der Phasenfehler beim *device under test* sein, dessen Phase an 0° oder ±90° dicht heranreicht.

Kalibrierfehler

Der initiale Fehler wird von den Präzisionswiderständen auf der Platine verursacht, er beträgt ±0,05 %. Dieser Fehler lässt sich weiter herabdrücken, indem die Anweisungen in Abschnitt 7 des Dokuments *First Time Setup, Resistance Calibration* [3] durchgeführt werden.

Weblinks und Literatur

[1] LCR-Meter 2013, Teil 1
www.elektor.de/110758

[2] LCR-Meter 2013, Teil 2
www.elektor.de/130022

[3] LCR-Meter 2013, Teil 3
www.elektor.de/130093

[4] www.elektor.fr/forumLCR

(130174)gd

elektor ACADEMY
the school of electronics

Arduino - Programmierung und Projektentwicklung

Nach einer kurzen Einführung und der Inbetriebnahme des Arduino-Boards erfolgt eine systematische Einführung in verschiedene Themengebiete. Dabei wird neben den erforderlichen theoretischen Grundlagen stets größter Wert auf eine praxisorientierte Ausrichtung gelegt. So werden wichtige Techniken wie AD-Wandlung, Timer oder Interrupts anhand von Praxisprojekten ausführlich erläutert. Den Abschluss des Seminars bildet eine Einführung in die eigenständige Entwicklung von Projekten und Systemen.

Referent: Dr. Günter Spanner - Teilnahmegebühr: 349,00 € (inkl. MwSt.)

Eagle PCB und Design

In diesem Kurs werden Sie lernen, wie man mit dem Programm Eagle der Firma Cadsoft GmbH Leiterplatten entflechten kann. Begonnen wird mit dem Zeichnen von Schaltplänen unter Verwendung von Standard-Eagle-Bibliotheken. Sie lernen, wie man Schaltpläne über mehrere Seiten hinweg zeichnet und wie man eigene Bibliotheken und Bauteile erstellt. Anschließend werden Sie lernen, wie man aus dem Schaltplan eine Leiterplatte definiert und diese dann entflechtet (layoutet). Dabei werden auch Spezialkenntnisse zum Layouten von HF-Schaltungen, wie Leitungswellenwiderstand, vermittelt. Nach erfolgreichem Layout werden Produktionsdaten erzeugt, die man benötigt, wenn man die Platine fertigen lassen möchte. Dabei wird auch auf die verschiedenen Produktionsarten wie fräsen und ätzen eingegangen. Zum Abschluss gibt es Tipps und Tricks zum Umgang mit EAGLE.

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

50% Sonderrabatt für Elektor-Abomitglieder: 224,50 € (inkl.MwSt)

3-tägiges Seminar 'Embedded Linux in Theorie und Praxis – ein Crashkurs'

Der Einstieg in ein so mächtiges Werkzeug wie Linux ist nicht trivial! Ziel des Kurses ist es, Ihnen grundlegende Embedded-Linux-Konzepte sowie die Handhabung von Linux zu vermitteln. Was sind z. B. Vor- und Nachteile? Sie werden Ihren eigenen Bootloader und Kernel cross-kompilieren, diverse Programme auf einem PC erstellen/cross-kompilieren und auf einem eingebetteten System ausführen und debuggen. Eine Kombination aus Theorie und praktischen Übungen wird es Ihnen ermöglichen, das neu erworbene Wissen bei Eigenentwicklungen einzusetzen. Nach dem Kurs sind Sie wahrscheinlich noch kein Embedded-Linux-Experte, aber hoffentlich in der Lage sein, sich selbständig zurechtzufinden.

Referent: Robert Berger Teilnahmegebühr: 1.898,00 € (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.

INKLUSIVE elektor ZERTIFIKAT

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

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

Linux Debugging
Dortmund: 19.09. + 20.09.2013
München: 12.12. + 13.12.2013

Eagle PCB und Design
München: 12.09.2013
Braunau am Inn: 13.09.2013
Dortmund: 19.09.2013
Hanau: 17.10.2013

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

Embedded Linux
Hanau: 16.09. bis 18.09.2013
München: 09.12. bis 11.12.2013

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

Elektor-Verlag GmbH | Süsterfeldstr. 25 | 52072 Aachen
Tel. 0241/88 909-16
www.elektor.de
m.debeij@elektor.com

Arduino auf Kurs (6)

Reisen im Namen der Elektronen

Von **David Cuartielles**
(Spanien)

Wenn ich an Elektronik denke, fallen mir zwei Dinge ein: die zunehmende Vernetzung der Dinge, aber auch die Möglichkeit, Jugendlichen etwas Sinnvolles beizubringen. Ich finde, dass wir im ersten Bereich schon sehr weit gekommen sind. Bei der Ausbildung junger Leute gibt es dagegen noch vielfach Raum für Verbesserungen. Über eine Initiative auf diesem Gebiet berichtet dieser Artikel.

Normalerweise braucht es etwa 2.500 Worte, um ein kleines Projekt für Elektronik-Interessierte zu behandeln. In diesem Beitrag aber möchte ich etwas Anderes versuchen. Ich möchte eine Geschichte erzählen, wie mich Atome und Bytes sehr weit weg von meinem gewöhnlichen Aufenthaltsort brachten. Sie handelt davon, wie uns Elektronik auf eine andere Art mit der Welt in Verbindung treten lässt. Diese Folge hätte daher auch *„Von der Kunst der Elektronik zum Schreinerhandwerk"* heißen können, doch das wäre lang und weniger griffig gewesen.

Bild 1.
Die Kirche von Zegache (Mexico). Bild: Talleres Comunitarios.

Stückliste

Zum Nachbau der Projekte dieses Beitrags braucht man:

- Ein offizielles Arduino Starter Kit
- Einen Mikro-Servomotor (180°)
- Einen kleinen 8-Ω-Lautsprecher
- Einen IR-Sensor von Sharp

Erfahrung 1: Ankommen

Santa Ana Zegache ist ein kleiner Ort südlich von Oaxaca in Mexico. Von Mexico City aus braucht man mit dem Auto gute acht Stunden. Man kann auch nach Oaxaca fliegen, was die Reisedauer halbiert, doch bei meinem ersten Besuch entschied ich mich aus Interesse am Land für das Auto (siehe **Bild 1)**.
Sta. Ana ist ein Dorf mit etwa 5.000 Einwohnern und schwer zu beschreiben. Mein erster Besuch

fand am 27. Dezember 2011 statt. Ich war völlig überrascht, dass die Straßen nicht asphaltiert waren. Außerdem waren viele Gebäude unfertig und oft ragten gar Moniereisen aus dem Beton. Es gibt einen einzigen Platz im ganzen Dorf. Das Rathaus, die Schule und die beiden Internet-Cafés befinden sich dort. Die Kirche wurde dank einer Stiftung renoviert, die von dem berühmten Maler Rodolfo Morales gegründet wurde. Hier ein Zitat von der Webseite des Projekts, das mich nach Mexico brachte:

„Der Maler Rodolfo Morales betrachtete Zegache als außerordentlich wertvoll. Den letzten Abschnitt seines Lebens widmete er der Wiederherstellung und Restaurierung der Kirche. Leider konnte er die Vollendung seiner philanthropischen Vision der Bewahrung des künstlerischen Erbes und der Reaktivierung der Talente dieser Region nicht mehr selbst erleben."

Zur Renovierung dieser Kirche wurde ein Workshop installiert, durch den viele Einwohner zu schreinern lernten. Dank der Großzügigkeit der Stiftung von Alfredo Harp Helú, der Rockefeller Foundation, von La Curtiduría und von Herr und Frau Sandretto konnten die Dorfbewohner diese Arbeit über mehr als zehn Jahre fortführen. Sie brachten sich dabei nicht nur das Schreinern bei, sondern auch die Restauration aller Arten von Holzarbeiten. **Bild 2** zeigt einen solchen Workshop.

Auch heute noch beschäftigen sie sich mit der Restaurierung von Kirchen und religiöser Ikonographie in der Umgebung. Der Workshop quillt über von gefalteten Händen, Christusbildern, Spiegeln und noch mehr Spiegeln.

Eine Zeit lang war die Herstellung von Spiegeln neben der Restaurierung die Haupteinnahmequelle. Es wurden Künstler aus der ganzen Welt für kurze Programme eingeladen, wo sie alles über das Dorf, den Workshop und die Spiegel erfuhren. Es gibt einen traditionellen Spiegel-Typ, den die Gäste „hacken" sollten. Die dahinter stehende Idee war, dass der Workshop das Recht erhält, fünf identische Kopien der Spiegelvariante des Gastkünstlers zu machen und diese für einen festen Preis zu verkaufen. Diese begrenzte Auflage hat eine doppelte Funktion: Auf der einen Seite werden bescheidene Einnahmen erzielt, aber zusätzlich machen diese Spiegel den Workshop, das Dorf und seine Bewohner in der Welt bekannt. **Bild 3** zeigt solche Spiegel.

So verlief mein erster Kontakt: Wir fuhren hin, verbrachten ein paar Stunden beim Besuch des

Bild 2.
Der Workshop von Zegache.

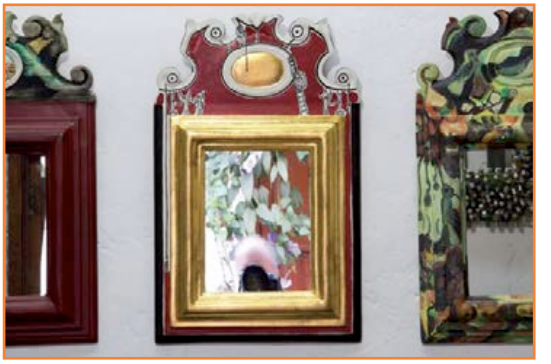


Bild 3.
Spiegel aus Zegache.

Bild 4.
Der improvisierte Arduino-Workshop in Zegache.

Workshops und versuchten uns vorzustellen, wie es wohl wäre, hier einen Workshop zu Thema Arduino zu halten. Das war nicht ganz einfach, denn es gab keinen Elektronikladen im Umkreis von 40 km und auch keinen lokalen Internetanschluss. Nur ein paar PCs waren vorhanden.

Erfahrung 2: Lehre was Du kannst

Im Februar 2012 kam ich zum zweiten Mal nach Zegache. Wir kannten ja nun die Infrastruktur und so hatten wir alles Nötige mitgebracht: Neben meinem PC einen Lötkolben sowie acht Arduino Starter Kits, einige Sensoren, Proto-Shields, eine Ladung Lautsprecher und etliche USB-Sticks mit Software (diese in den örtlichen Internet-Cafés downloaden zu wollen wäre aussichtslos). Von meiner ersten Reise kannte ich die Elektronik-Läden in der eine Autostunde entfernten Regionalmetropole Oaxaca. Ich wusste, was man dort kaufen kann. LEDs und Batterien waren kein Problem.

Wir besorgten außerdem große Papierbögen und Stifte, denn die üblicherweise vorhandenen Hilfsmittel wie Whiteboard oder Projektoren gab es hier nicht. Damit war der Arduino-Workshop vorbereitet (siehe **Bild 4**).

Die Teilnehmer wurden auf Dreiergruppen verteilt und begannen mit dem üblichen Stoff von Arduino-Workshops. Jedes der fünf Teams verfügte über einen PC, ein Starter-Kit und die komplette benötigte Software.

Wir brauchten eine Woche für die Elektronik-Grundlagen: Digitale Ein- und Ausgänge, Analog/Digital-Konvertierung oder die Frage, wozu eine IDE gut ist. Einmal konnte ich eine Teilnehmerin dabei beobachten, wie sie immer wieder die Tastaturkürzel für Kopieren und Einsetzen (Ctrl+C & Ctrl+V) ausführte. Ich hätte nie geglaubt, dass dies so faszinierend sein könnte, aber das war es wohl. Stellen Sie sich nur einmal vor, Sie könnten eine Sache per Tastendruck magisch von einem Ort zum nächsten kopieren! Vielleicht haben wir einfach verlernt darüber zu staunen, wie magisch digitale Elektronik wirken kann.

Nach drei Tagen voller Bits und Bytes wandten wir uns dem Bau von Dingen zu. Wir regten einen Ideenfindungsprozess an und luden die Teilnehmer ein, ihre eigenen interaktiven Spiegel zu erfinden, indem sie diese mit Sound und LEDs ausstatten. Der Rest der beiden Wochen verging beim Bau hölzerner Gehäuse, der verzierenden Schnitzerei, der Lackierung von Hand, dem Löten von LEDs und der Programmierung von Interaktionen. Wie **Bild 5** beweist, waren alle fleißig.

Diese Erfahrung bereicherte mich auf vielerlei Art: Die Menschen waren vielfältige Zusammenarbeit gewohnt. Es spielte keine Rolle, womit sie gerade am eigenen Projekt beschäftigt waren – sie hatten immer Zeit um anderen zu helfen. Fast überflüssig zu erwähnen, dass alle fünf Projekte aller fünf Teams in dieser begrenzten Zeit fertig wurden.

Nie haben wir überzogen: Der Arbeitstag endete um 18:00 Uhr und anschließend gingen die Leute nach Hause und kümmerten sich um ihre Familien.

Alles wird recycelt. Der Einsatz digitaler Produktionstechnik wie Laser-Schneideanlagen würde

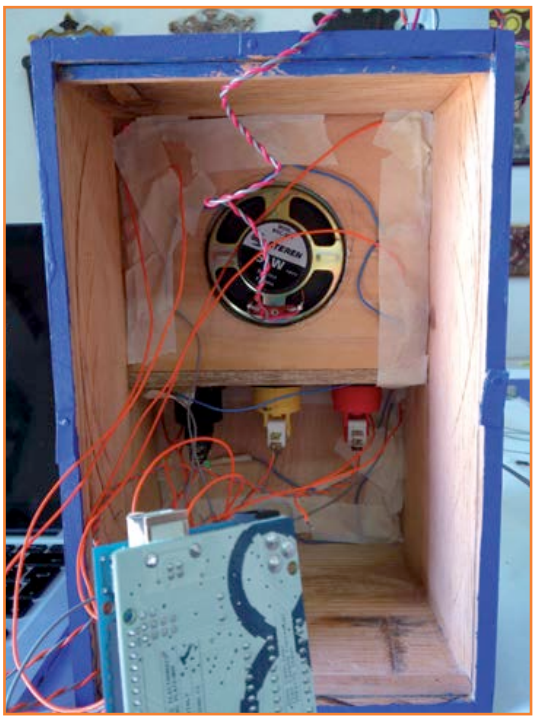


Bild 5.
Der Bau interaktiver Spiegel und Spielzeuge.

Bild 6.
Einige Projekte aus dem ersten Workshop.

fraglos den Materialverbrauch optimieren. Für diesen Workshop aber wurde nicht ein einziges Stück Holz gekauft. Wenn ein 40 cm langes Stück Holz fehlte, wurden einfach zwei 20 cm lange Stücke aneinander geklebt und schon hatte man das benötigte Teil in der Hand.
Die Teilnehmer lernten nicht nur die Elektronik-Grundlagen, sondern auch das Löten und Bestücken von Hand. Alle Projekte funktionierten schließlich und bei einem weiteren Besuch ein Jahr später stellte ich fest, dass die Technik lange und problemlos arbeitete.

Bild 6 zeigt Projekte, die während dieses Workshops entstanden:

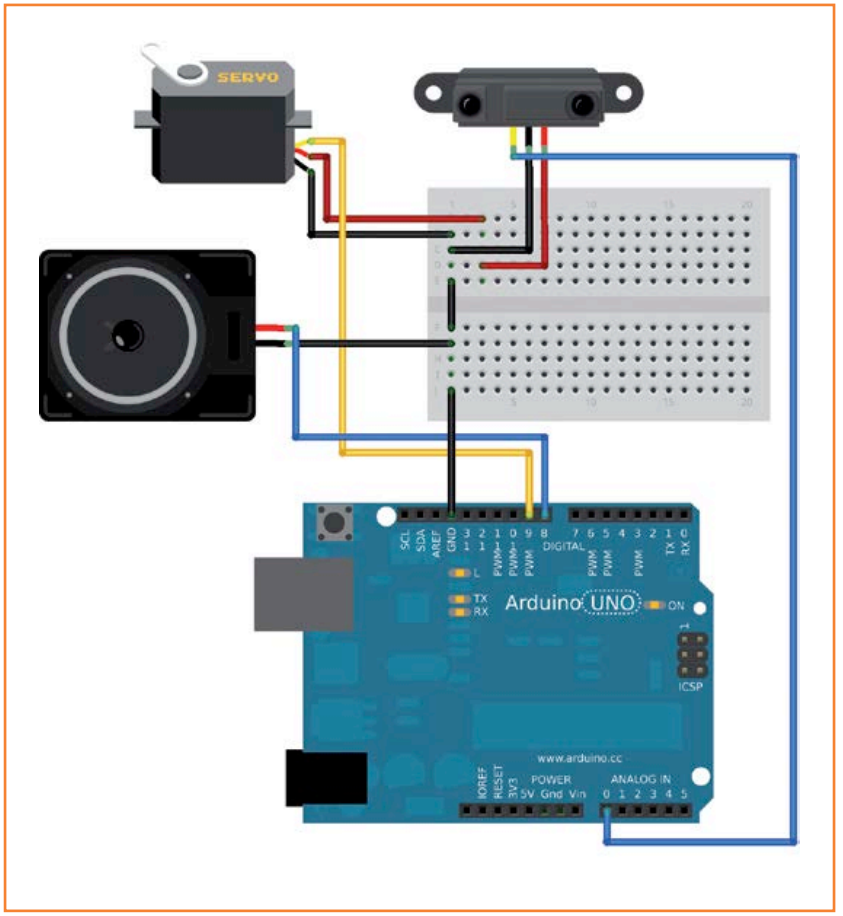
- Einen magischen Spiegel, der Tastendrücke mit Tönen und Licht beantwortet.
- Eine Jukebox, die Instrumente zu einer Sound-Loop hinzufügt.
- Ein Brett für Trinkspiele (kein Kommentar).
- Ein schmeichelnder Spiegel, der beim Vorbeigehen Komplimente macht.
- Ein Schmuck-Kasten, der bei jedem Öffnen andere Musik abspielt.

Erfahrung 3: Lerne soviel Du kannst

Was noch nicht gesagt wurde: Unser Besuch war Teil eines EU-Kunst-Projekts namens Euroaxca, bei dem Künstler und Handwerker beiderseits des Atlantiks zu unterschiedlichen Themen kooperierten und Erfahrungen austauschten. Als Teil des Projekts plante ich eine Fortsetzung, doch angesichts der guten Ergebnisse des ersten Workshops war ich etwas unsicher, wie ich die nächste Projektphase angehen sollte.
Meiner Ansicht nach sollte es auch für einen Schreiner in Zegache möglich sein, das Grundwissen zum Bau interaktiver Spielzeuge zu erwerben und diese dann so zu kopieren, wie er das auch mit Spiegeln macht. Später sollte er sogar eigene Modifikationen durchführen und eigene Entwürfe realisieren können.
„Leider" war dieses Ziel ja schon erreicht. Wir erlebten, dass es möglich ist, fachfremden Menschen in nur zwei Wochen den Umgang mit Elektronik beizubringen, damit diese ihre eigenen Projekte auf die Beine stellen können. Die Frage war also: Was jetzt?
Die Antwort kam von der Koordinatorin des EU-Projekts. Sie schlug vor, so etwas wie animierte Figuren zu bauen. Ein Vogel wäre doch optimal dafür, meinte sie. Ich entschied mich dazu, das einfach auszuprobieren. „Machen wir also Robotervögel!" sagte ich zu meinem Team, als wir gerade Schweden in Richtung Mexiko verließen. Exakt ein Jahr nach dem ersten Besuch kamen

Bild 7. Teile eines handgearbeiteten Robotervogels.

Bild 8. Das elektronische Herz eines Robotervogels.



wir zurück.
In Zegache und dem Workshop gab es nicht viel Veränderung. Einige Leute waren nicht mehr dabei, dafür waren ein paar neue Gesichter zu sehen. Am Dorfplatz war eine große Struktur aufgebaut, die Schatten spendete. Außerdem gab es da noch zwei Tore, damit die Kinder Fußball spielen konnten. Sie können sich sicher vorstellen, wie heiß die mexikanische Sonne tagsüber brennen kann. Mittlerweile war es gelungen, das Internet zum Workshop zu bringen: Jemand im Dorf hatte nämlich DSL bekommen. Aus ominösen Gründen scheint es nicht so einfach, das geneh-

Listing 1.
Die Software-Seele des Roboter-Vogels

```
//*
 * Zegache Robot
 *
 * (c) 2013 D. Cuartielles
 */


#include <Servo.h>

Servo myservo;

int sensorPin = 0;
int speakerPin = 8;
int servoPin = 9;
int val;

void setup()
{
  myservo.attach(servoPin);
  pinMode(speakerPin, OUTPUT);
}

void loop()
{
  val = analogRead(sensorPin);

  if ( val < 250 && val > 50) {
    myservo.write(50);
    tone(speakerPin, val);
    delay(200);
    myservo.write(100);
    noTone(speakerPin);
    delay(100);
    myservo.write(120);
    tone(speakerPin, val);
    delay(200);
    myservo.write(0);
    noTone(speakerPin);
    delay(200);
 }
}
```

migt zu bekommen. Der DSL-„Besitzer" machte den Zugang zum Internet über mehrere WLAN-Repeater durch das ganze Dorf möglich. Und so konnten wir glücklicherweise gleich an zwei PCs E-Mails lesen.

Dieses Mal war die Sache sehr einfach. Ich sprach mit einigen Leuten vom Workshop über die Herstellung von Robotervögeln. Sie öffneten ihre Trickkiste und begannen alsbald mit dem Schnitzen unterschiedlicher Vögel, die ihre Flügel, den Schwanz und den Schnabel bewegen oder den Kopf schütteln konnten. Die Resultate sind in **Bild 7** zu bewundern. Der erste Vogel brauchte zwei Tage und dann waren auch die anderen Vögel kein Thema mehr.

Software und Elektronik waren einfach ausgelegt. Die Verdrahtung der Teile sieht man in **Bild 8**. Alle Vögel waren mit einem Mikro-Servomotor ausgestattet, der an einer Schnur zog, um alle beweglichen Teile des Vogels auf einmal zu bewegen. Hinzu kamen noch ein Lautsprecher für einfache Töne und ein IR-Sensor, der registriert, wie weit entfernt sich Personen aufhalten. Die Vögel sollten wie hölzerne Skulpturen aussehen (**Bild 9**) - bis man ihnen zu nahe kommt. Dann sind einfache Interaktionen möglich: Wedeln mit der Hand vor dem IR-Sensor bringt unterschiedliche Töne hervor.

Ein Blick in den Code von **Listing 1** zeigt, wie einfach es sein kann, einen Robotervogel zu beleben. Es bleibt jede Menge Raum für Verbesserungen, gerade wenn es um die Tonerzeugung geht. Doch ich brauchte etwas Einfaches, was die Leute in Zegache selbst verändern konnten. Wir planen gerade einen weiteren Besuch, bei dem es um verbesserte Sound-Spielzeuge gehen wird, doch das ist eine andere Geschichte.

Schlussbemerkung

Durch Elektronik lernt man nicht nur selbst, wie Dinge funktionieren. Mit etwas Offenheit kann man schnell in die Lage kommen, mit anderen Menschen überall auf der Welt Projekte zu realisieren. Nehmen Sie also eine Arduino-Platine in die Hand, sprechen Sie mit Freunden, Bekannten oder Nachbarn und interessieren Sie sie für diese extrem interessante Art, interaktive Objekte zu bauen und dabei Spaß zu haben.

Das ist Hilfe zur Selbsthilfe im wahrsten Sinne des Wortes!

(120751)

Bilder von Laura Balboa (Mexico)

Bild 9.
Die fertige interaktive Vogelskulptur.

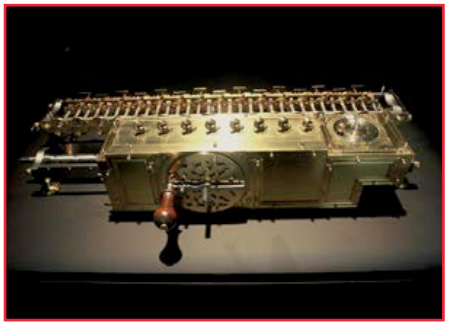
Weblinks

[1] Das Zegache-Workshop-Projekt: http://proyectozegache.com/index_en.php

[2] Das EU-Projekt Euroaxaca: http://euroaxaca.org

Danksagung

An dieser Stelle möchte ich Geska und Robert von „Performing Pictures" danken, die das Euroaxaca-Projekt initiiert und mich freundlicherweise zum Mitmachen eingeladen haben. Ganz besonderen Dank auch den Einwohnern von Zegache für ihre Gastfreundschaft und dafür, dass Sie mir klar gemacht haben, dass in Elektronik oft mehr steckt als nur Bytes.



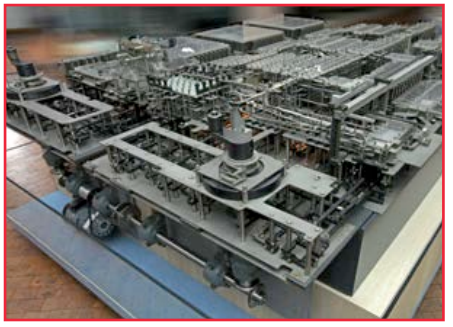
1

Konrad Zuse: Z1 bis Z4 und mehr

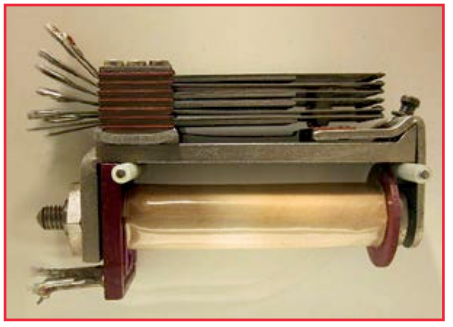
Es begann mit Nullen, Einsen und Algebra

2

Seit sich Menschen mit Mathematik und logischem Denken beschäftigen, wurde auch versucht, sich der damit verbundenen drögen Rechenarbeit zu entledigen. Basierend auf dem Verständnis logischer Funktionen und Beziehungen wurden die raffiniertesten Rechenmaschinen ersonnen. Dieser Artikel unternimmt einen Streifzug entlang der eindrucksvollen Beiträge Konrad Zuses zur Entwicklung des Computers.



3



4

5

Von Peter Beil (D)

Eine der frühesten Rechenmaschinen wurde von Gottfried Leibniz um das Jahr 1700 konstruiert (**Bild 1**). Bis ein Rechner direkt logische Operationen ausführen konnte, sind dann aber doch noch einige Jahrhunderte vergangen. Erst 1938 baute der geniale Denker und Erfinder Konrad Zuse (**Bild 2**) seinen Z1 – einen „richtigen“ Computer, auch wenn er noch mechanisch operierte. Der Begriff „Computer“ entstammt übrigens dem lateinischen „computare“ für zusammenzählen oder berechnen. Der Z1 war „frei programmierbar“ und rechnete mit binären Werten. Das Original wurde im Zweiten Weltkrieg zerstört. Im Deutschen Technikmuseum in Berlin befindet sich ein Nachbau (**Bild 3**).

Z1: Ein schwieriger Anfang

Der Computer mit dem Kürzel Z1 arbeitete bereits mit logischen Verknüpfungen wie „und“, „oder“ etc., die rein mechanisch ausgeführt wurden. Auf Grund von Toleranzen, Reibung und anderen mechanischen Widrigkeiten hatte das Gerät aber recht viele Tücken, da sich die Schaltglieder immer wieder verhakten.

Eines aber hatte der Z1 selbst modernen Computern voraus: den nichtflüchtigen Speicher. Ein bis heute ungelöstes Problem, das eigentlich nur umgangen wird. Konrad Zuse hatte damals schon erkannt, dass logische Operationen nur binär mit vertretbarem technischem Aufwand zu realisieren sind. Im Gegensatz dazu bevorzugten viele andere Forscher und Pioniere in England und den USA zunächst noch das Dezimalsystem.

Z2: 16 bit@10 Hz

Aufgrund der mechanischen Probleme mit dem Z1 startete Zuse 1939 ein Experiment: Der daraus entstandene Z2 basierte auf hunderten Telefonrelais (**Bild 4**). Er lief bei einer Taktfrequenz von angenähert 10 Hz und war dazu in der Lage, die vier Grundrechenarten mit dualer Festkomma-Arithmetik durchzuführen. Der Rechner besaß einen 16-bit-Speicher und ein Gesamtgewicht von eindrucksvollen 300 kg.

Z3: Software auf Film

Im Mai 1941 stellte Zuse seinen Z3 vor. Dabei handelte es sich um den ersten richtig funktionierenden Digitalcomputer (**Bild 5**). Rund 600 Relais waren für

das Rechenwerk nötig und der Speicher umfasste weitere 1.400 Relais. Wie beim Z1 kam binäre Fließkomma-Arithmetik zum Einsatz. Außerdem war der Z3 der erste wirklich universell programmierbare Computer. Angeblich hatte Zuse einen Bekannten bei der UFA (ein traditionsreiches deutsches Filmunternehmen), was ihn auf die Idee brachte, die Eingabe über gelochte Filmstreifen zu realisieren. Das Stanzen des Films erfolgte gleich am Gerät (**Bild 6**). Es gab natürlich auch eine Lese-Einheit, die den Film bzw. seine Löcher mit Hilfe der Perforation präzise abtasten konnte (**Bild 7**).

Auch der Z3 wurde im Krieg zerstört, doch eine voll funktionsfähige Replika steht im Deutschen Museum in München. Dieser Nachbau wurde noch von der mittlerweile aufgelösten Zuse KG realisiert. Ein Blick ins Innere zeigt den ungeheuren Verkabelungsaufwand sowie ein Walzenschaltwerk für den Takt und Schrittschaltwerke, wie sie später in Telefonwählern eingesetzt wurden (**Bilder 8**, **9** und **10**). Die Wortlänge betrug 22 bit, davon 7 bit für den Exponenten, 14 bit für die Mantisse und 1 bit für das Vorzeichen. Der Z3 hatte einen Relaisspeicher für 64 Worte und das Programm lief in einer Schleife. Zahleneingaben erfolgten per Tastatur (**Bild 11**). Die Ergebnisse wurden auf einem Anzeigefeld mit Lämpchen ausgegeben (**Bild 12**). Neben den vier Grundrechenarten beherrschte der Z3 auch die Quadratwurzel.

Z4: Immer noch Mechanik

Die Arbeit am Z4 begann 1942. Erst 1945 war er betriebsbereit (**Bild 13**). Auch dieser Computer basierte auf Relais und stand nun in Konkurrenz zu den amerikanischen Maschinen „Mark I“ von 1944 und „ENIAC“ von 1946 (**Bild 14)**, die aber nach anderen Prinzipien (u.a. mit Röhren und vor allem dezimal) arbeiteten.

Der Z4 hatte wie der Z3 durchaus Ähnlichkeit mit modernen Computern, denn beide konnten bereits Schleifen ausführen. Der Z4 beherrschte sogar Gleitkomma-Arithmetik mit 24 bit für die Mantisse, 7 bit für den Exponenten und 1 bit für das Vorzeichen - insgesamt also 32-bit-Worte. Er verfügte über zwei simultan arbeitende Rechenwerke. Neben den Grundrechenarten konnte er Quadrieren und Wurzeln ziehen. Der Befehlssatz des Z4 umfasste 29 Instruktionen. Die angeschlossene elektrische Schreibmaschine war kein Drucker im heutigen Sinne, sondern diente zur Ausgabe von Rechenprotokollen (**Bild 15**).

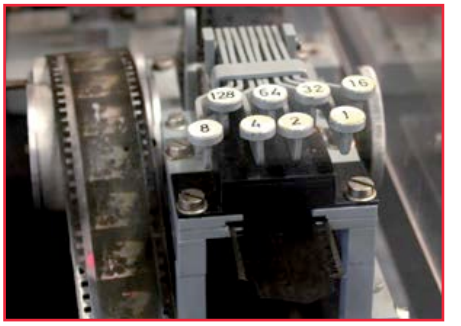
Die Programmierung erfolgte auch hier wieder mit gelochten Filmstreifen (**Bild 16**). Eine Speicherung im heutigen Sinne war nicht möglich: Das Abschalten der Betriebsspannung vernichtete alle Daten. Erste brauchbare Ferritkern-Speicher gab es erst Mitte der 50er Jahre. Der Z4 konnte etwa 30 Operationen pro Minute ausführen und brauchte für eine Addition 0,5 Sekunden. Eine Multiplikation dauerte 3,5 Sekunden. Bereits begonnene Weiterentwicklungen wie Programmsprünge oder Indexregister zur Adress-Umrechnung gingen allerdings in den Kriegswirren unter.

Programmiersprache und Nachkriegsentwicklungen

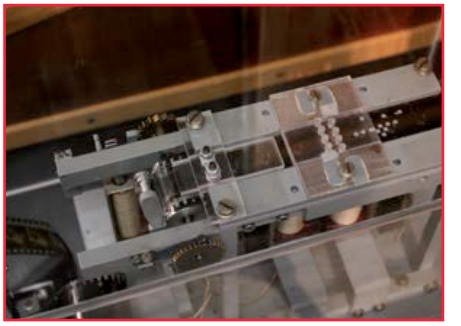
Kaum jemand weiß, dass Konrad Zuse schon damals die Notwendigkeit einer höheren Programmiersprache erkannte. In den Jahren 1945/46 entwarf er sein „Plankalkül“, konnte dies aber nicht mehr veröffentlichen. Damit hatte er aber schon die Grundlagen von Fortran, Algol oder Cobol vorweg genommen.

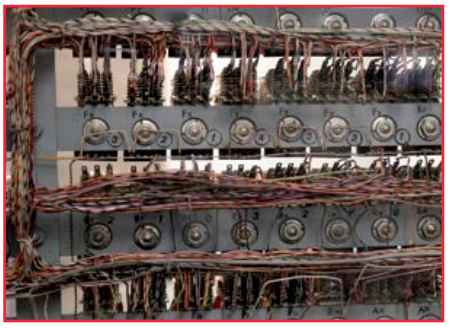
In den Jahren nach dem Krieg war der Z4 auf einer Odyssee durch Deutschland, die Schweiz und Frankreich. Unter anderem interessierte sich IBM für die Schutzrechte, um weitere Entwicklungen zu verhindern. In den 50er Jahren landete der Z4 dann an der ETH Zürich, wo er für wissenschaftliche Berechnungen im Einsatz war. Im Jahre 1950 war die Maschine der einzige funktionierende Computer in Europa. 1960 endete die lange Reise im Deutschen Museum in München.

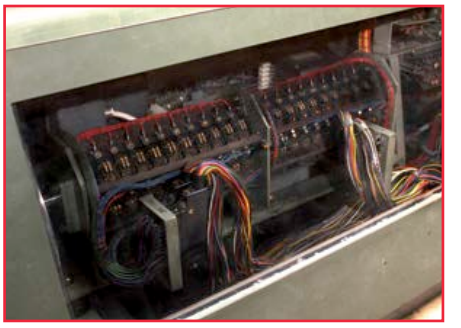
Die Firma Zuse beschäftigte sich noch einige Jahre weiter mit der Entwicklung wissenschaftlicher Computer. Im Jahre 1961 gelang die Herstellung des



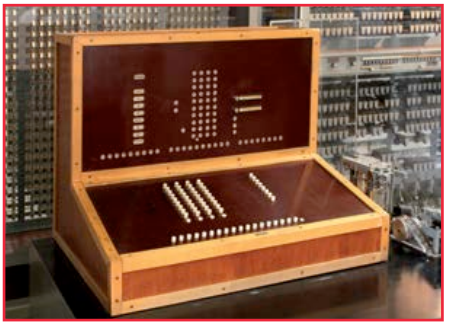
6


7

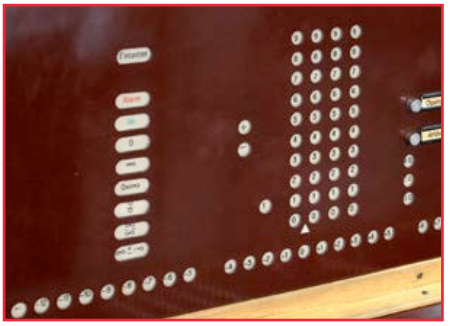

8


9

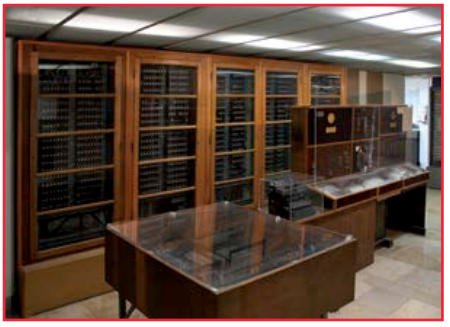

10



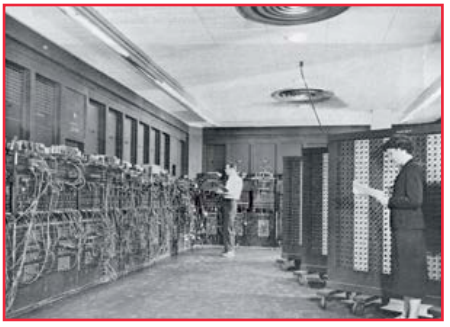
11



12

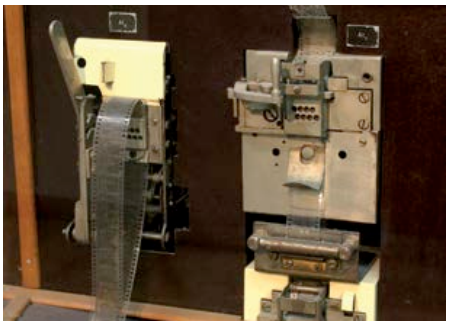


13



14

15



16

So „denkt“ der Z3

K. Zuse (l) and H. Nixdorf

Jede Rechenoperation basiert auf der Addition bzw. Subtraktion zweier Integer-Zahlen.
Eine *Addition* zweier Fließkommazahlen wird durch Berechnung der Differenz der Exponenten realisiert; damit erfolgt dann die Angleichung der Mantisse und dann erfolgt die Addition der Mantissen.
Eine *Subtraktion* erfolgt wie eine Addition mit dem zusätzlichen Schritt der Berechnung des Zweierkomplements der zweiten Mantisse.
Eine *Multiplikation* ist eine Addition der Exponenten mit anschließender Multiplikation der Mantissen durch eine iterative Addition.
Eine *Division* entspricht einer Multiplikation, nur werden hier die Exponenten subtrahiert und für die Division der Mantissen eine iterative Subtraktion verwendet.
Das *Wurzelziehen* nutzt einen Algorithmus, der eine iterative Division realisiert. Das Rechenwerk besteht aus zwei Teilen. Ein Werk rechnet mit Exponenten und eines wird für die Mantissen-Berechnung genutzt. Für Befehle, bei denen iterative Methoden zum Einsatz kommen, wird ein Sequenzer verwendet, um einzelne Teile des Rechenwerks anzusteuern. Dies entspricht grob dem Mikro-Code moderner CPUs.

ersten voll funktionsfähigen Plotters mit der schönen Bezeichnung „Graphomat“. Zuletzt war die kleine Firma der übermächtigen Konkurrenz aus dem USA nicht gewachsen und so wurde sie 1964 von BBC (**B**rown **B**overi & **C**ie) und 1967 schließlich von der Siemens AG geschluckt.

(130040)

Bildquellen

Die Fotos 1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 und 16 stammen vom Autor, mit der freundlichen Genehmigung des Deutschen Museums in München.
Die Bilder 14 und 15 sind der deutschen Wikipedia entnommen.
Bild 3: Deutsches Technikmuseum, Berlin.
Bild 2: Prof. Horst Zuse.

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).

CIRCUIT CELLAR
CC-WEBSHOP
ARCHIVE·STORE

MISSED an issue?
LOST an issue?

Dog ATE your issue?
No problem!

Circuit Cellar issue PDFs are always available at **www.cc-webshop.com. Save $2 off every issue PDF** now through the end of July. **Use coupon code CCPDF713**

Each issue of *Circuit Cellar* contains:
- Analysis of the newest embedded technologies
- Electronics engineering insight
- Hardware design
- Programming tips
- Techniques for testing

www.cc-webshop.com

PLUS!

Want the complete issue archive? Get a CC Gold USB drive today. It's packed with every *Circuit Cellar* issue published through date of purchase.

Ends July31,2013

Hexadoku Sudoku für Elektroniker

Auch in der Sommerausgabe muss niemand auf sein gewohntes Hexadoku verzichten. Nehmen Sie das Rätsel einfach mit in den Urlaub oder an den Baggersee, sicher findet sich dort das eine oder andere ruhige Stündchen. Wer uns die richtigen Ziffern in den grauen Kästchen zuschickt, kann einen von vier schönen Gutscheinen gewinnen!

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 31. August 2013!

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

Die richtige Lösung ist: **3D1AE**.
Der Eurocircuits/Elektor-PCB-Service-Gutschein im Wert von 100 € geht an: Dirk Neerijse (Belgien).
Einen Elektor-Gutschein über je 50 € haben gewonnen: József Nagy, Sigrid Scheel und Joe Young.
Herzlichen Glückwunsch!

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

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

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

NEU

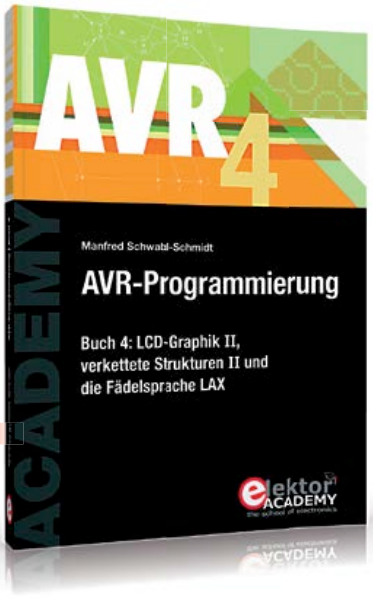
Ein Muss für jeden Elektor-Leser!

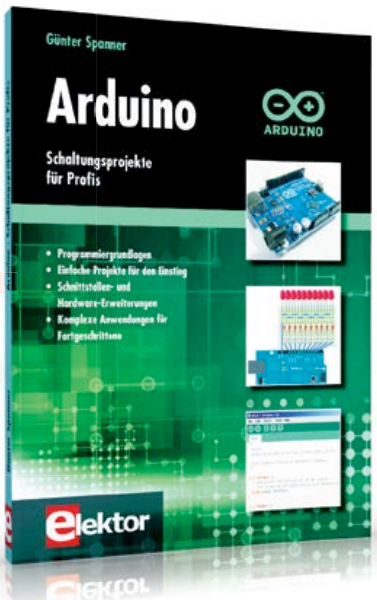
ISBN 978-3-89576-263-5
€ 69,00 • CHF 85,60

elektor

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

Elektor Store





1 2 3

30 Projekte in C für Fortgeschrittene

1 ARM-Mikrocontroller 2

Die im Buch beschriebenen Projekte mit dem mbed-Board sind für Einsteiger in C und ARM-Mikrocontroller ausgelegt. 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.

243 Seiten (kart.) • ISBN 978-3-89576-271-0
€ 39,80 • CHF 49,40

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

2 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

3 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

Apps programmieren – Schritt für Schritt

4 Android

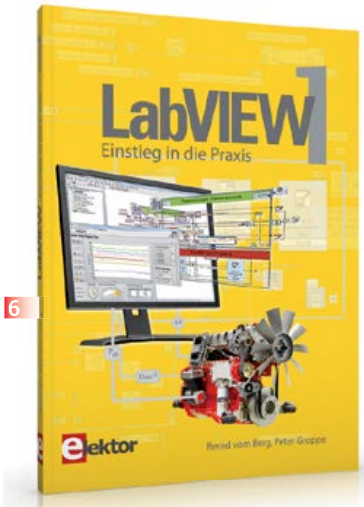
Smartphones und Tablet-Computer mit dem Betriebssystem Android finden immer weitere Verbreitung. Die Anzahl der Anwendungsprogramme – die sogenannten Applikationen oder kurz Apps – mit denen sich die Geräte individuell an die Vorlieben und Wünsche ihrer Benutzer anpassen lassen, steigt täglich an. Man 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

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:

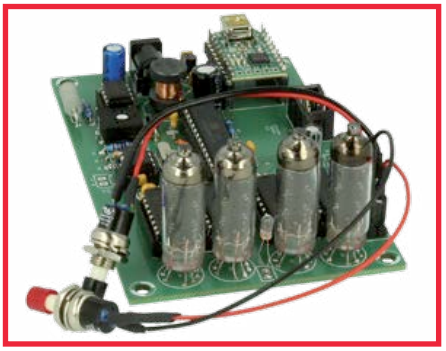
www.elektor.de/shop

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

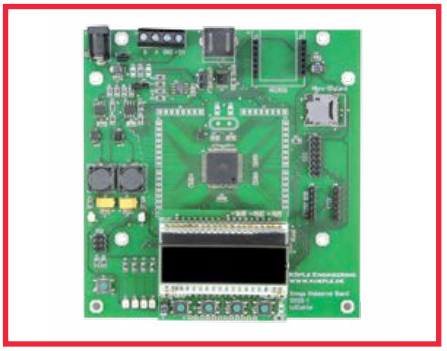
Kompakte Audio-Endstufe

Kein Zweifel, in letzter Zeit sind die Audio-Enthusiasten unter unseren Lesern etwas zu kurz gekommen. Das haben wir uns zu Herzen genommen und ein neues, attraktives Audio-Projekt entwickelt. In der nächsten Ausgabe erscheint ein Endverstärker, dessen Kernstück ein spezieller Treiberbaustein von Texas Instruments ist. Der Baustein arbeitet klaglos an Betriebsspannungen bis ±100 V. Kombiniert mit den Endstufentransistoren lassen sich mehr als 200 W Dauerleistung an 4 Ω herausholen, selbstverständlich bei ultraniedrigen Verzerrungen. Auf der kompakten Platine haben auch eine Einschaltverzögerung und eine elektronische Sicherung ihren Platz.



Numitron-Uhr

Unter den Elektor-Projekten gewinnt die Arduino-Plattform zunehmend an Boden. Diesmal haben wir uns eine stilvolle Synthese aus nostalgischer und moderner Technik einfallen lassen: Ein mit Arduino kompatibles Mikrocontroller-System, einige ergänzende Bauelemente und mehrere so genannte Numitron-Röhren sind die Komponenten für eine Digitaluhr, die auch Temperaturen anzeigt. Numitron wird eine „antike" Siebensegment-Anzeigeröhre genannt, in der die Segmente ähnlich einer Kleinglühlampe aus Glühfäden bestehen. Heute werden solche Anzeigeröhren im Internet zu erschwinglichen Preisen gehandelt.



Xmega-Webserver-Board

Im nächsten Heft stellen wir ein vielseitig verwendbares Mikrocontroller-Board vor, das mit einem besonders leistungsfähigen Controller aus der beliebten AVR-Familie ausgerüstet ist. Zur Ein- und Ausgabe dienen 4 LEDs, 4 Taster und ein (abnehmbares) Display. An Schnittstellen stehen RS485 und verschiedene UART/TTL-Steckverbinder, an die zum Beispiel der USB/TTL-Konverter BOB angeschlossen werden kann, zur Verfügung. Über den Embedded Extension Connector ist das Board vielfältig erweiterbar. Dank eines aufsteckbaren TCP/IP-Moduls lassen sich Webserver- und andere Netzwerk-Anwendungen realisieren; ein Mikro-SD-Karten-Steckplatz rundet das Ganze ab.

Änderungen vorbehalten! **Elektor September 2013 erscheint am 21. August 2013.**

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

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

- Sie erhalten **10 Elektor-Hefte** (8 Einzelhefte + 2 Doppelausgaben Januar/Februar und Juli/August) pünktlich und zuverlässig frei Haus.
- **Extra:** Jedes Heft steht Ihnen außerdem 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!

elektor GOLD membership

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!