[Analog • Digital • Embedded • Mikrocontroller • Audio • Messtechnik]

G3078 E März 2012 (D) € 7,60 • CHF 14,50 • (A, B, L) € 8,40

Grundlagen: Transistoren durchmessen

elektor

www.elektor.de

AVR

Software Defined Radio

Hochpräzise Signale mit einem ATtiny-Controller

2x Messen, Steuern, Regeln mit Android

Steuerung mit Bluetooth • Schnell zum User-Interface auf Smartphone und Tablet

2x Projekte für den PC

Flexible Ventilatorregelung • Festplattenaktivitätsanzeige mit 10 LEDs

2x Arduino

Universeller Regler • Eigene Elektronik integrieren

DAS ORIGINAL SEIT 1994

PCB-POOL®

Beta LAYOUT

Oft kopiert – doch nie erreicht:

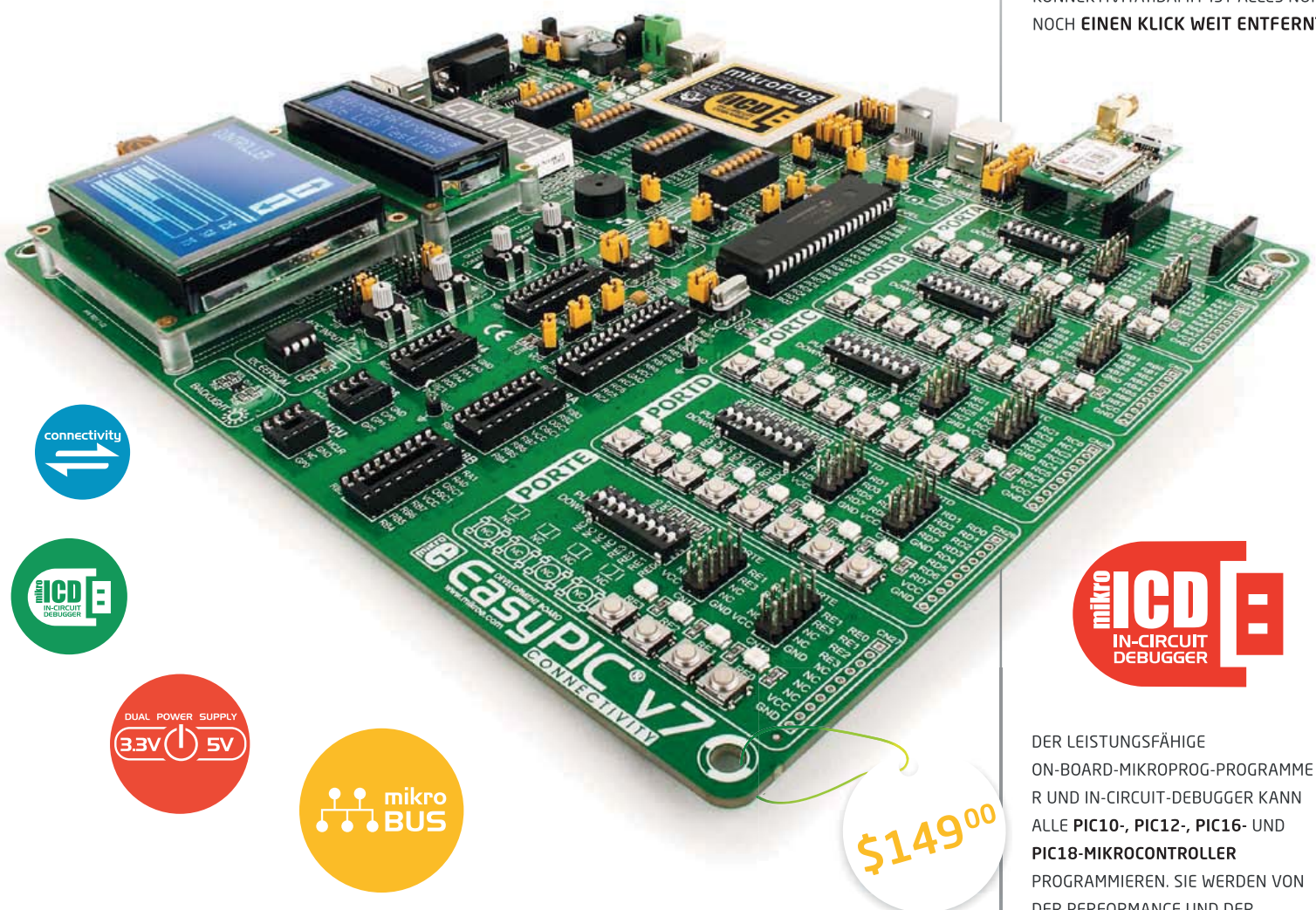
Leiterplatten online kalkulieren	PCB-POOL®	Basista	Euro-circuits	Leiton	WEdirekt	multi-cb
	✓	✓	✓	✓	✓	✓
FREE STENCIL	✓	–	–	–	–	–
Bestückung online	✓	–	–	–	–	–
Kostenlose Layoutsoftware	✓	–	–	–	–	–
Bewertungs-Rabattsystem	✓	–	–	–	–	–
Akzeptierte Layoutformate	16	6	1	3	5	3
Kollisionsprüfung zum Anfassen	✓	–	–	–	–	–
Auftragsverfolgung mit Ansprechpartner	✓	–	–	–	–	–
Watch"ur"PCB	✓	–	–	–	–	–
Pünktlich oder kostenlos	✓	–	–	✓	–	–
8h-Eilservice	✓	✓	–	–	–	–
Online Daten-Restore Service	✓	–	–	–	–	–

Das Original seit 1994!

www.pcb-pool.com

Beta LAYOUT

EasyPIC v7 connectivity



Der weltweite Bestseller unter den PIC-Entwicklungs-Boardserscheint nun in dersiebten verbesserten Version und ist Spitzentechnik in Funktion und Qualität. Mitvier Steckverbindern für jeden Port von EasyPIC v7 ist es sehr **vielfältig anschließbar**. Die Ports sind logisch zu Gruppen für LEDs und Taster zusammengefasst. Der leistungsfähigeintegrierte **mikroProg**-In-Circuit-Debugger und-Programmer unterstützt über 250Chips mit **3,3 V und 5 V**. Ausgestattet mit zwei Display-Typen,seriellem EEPROM, zweiTemperatursensoren, Piezo-Buzzer, USB,RS-232 und FTDI, Oszilloskop-GND-Pins sowie **mikroBus-Support** ist dieses Board die unersetzliche Basis einer effektiven PIC-Entwicklungs-Umgebung.

mikro BUS

MAN BRAUCHT NUR EIN **CLICK-BOARD™** ANSTECKEN UND SCHÖN LÄUFT ES. DAS SPEZIELLE **MIKROBUS™ -KOMMUNIKATIONS-INTERFACE** VEREINFACHT DIE ENTWICKLUNG UND ERMÖGLICHT SIMPLE UND DOCH HOCHEFFEKTIVE KONNEKTIVITÄT.DAMIT IST ALLES NUR NOCH **EINEN KLICK WEIT ENTFERNT!**

mikro ICD IN-CIRCUIT DEBUGGER

DER LEISTUNGSFÄHIGE ON-BOARD-MIKROPROG-PROGRAMME R UND IN-CIRCUIT-DEBUGGER KANN ALLE **PIC10-, PIC12-, PIC16-** UND **PIC18-MIKROCONTROLLER** PROGRAMMIEREN. SIE WERDEN VON DER PERFORMANCE UND DER EINFACHEN FUNKTION VERBLÜFFT SEIN.OB ANFÄNGER ODER PROFI - EINMAL DAMIT GEARBEITET WILL MAN ES NICHT MEHR MISSEN.

DUAL POWER SUPPLY 3.3V 5V

EASYPIC V7 IST DERZEIT WOHL DAS EINZIGE-ENTWICKLUNGS-BOARD, DAS SOWOHL 3,3-V- ALS AUCH 5-V-MIKROCONTROLLER UNTERSTÜTZT. REVOLUTIONÄRE VERFAHREN FÜHREN ZUR UNTERSTÜTZUNG **VON 250 MIKROCONTROLLERN** DURCH EIN EINZIGES BOARD. ES IST, WIE WENN MEHRERE BOARDS IN EINEM STECKEN WÜRDEN!

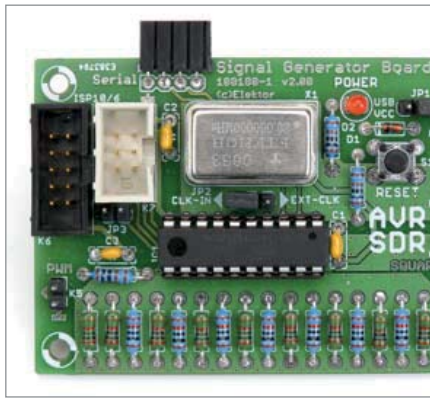
MikroElektronika
DEVELOPMENT TOOLS | COMPILERS | BOOKS

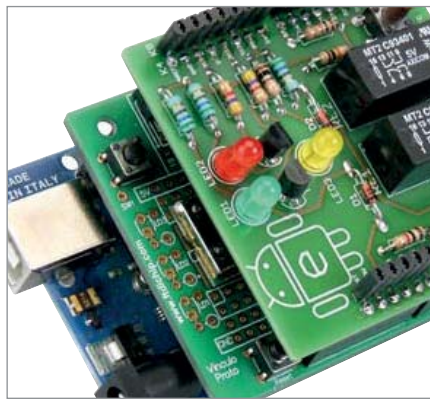
GET IT NOW
www.mikroe.com

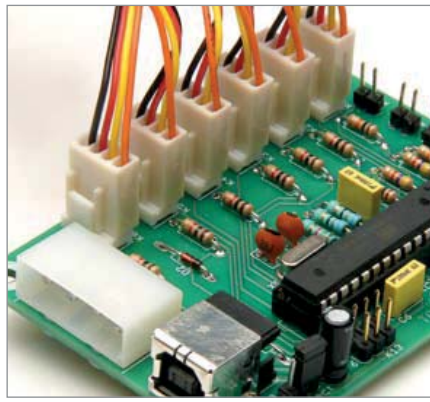
Androino

Im letzten Editorial habe ich unsere neue Community-Website angekündigt – mittlerweile durften wir Redakteure schon einen Prototypen ansehen und testen. Registrierte User können eigene Projekte starten, mit einer Beschreibung und dem Upload von Files. Andere Leser dürfen dann mit Kommentaren und Ideen zum Projekt beitragen, auch der File-Upload soll möglich sein, wenn der Projekt-Autor dies erlaubt. Darüber hinaus gibt's die Möglichkeit, sich selbst mit einem Foto und ein paar Daten zur Person (etwa den eigenen Interessengebieten) vorzustellen. Das Ganze sieht noch recht übersichtlich aus, ohne allzu viel Schnick-Schnack. Wir haben das System auch bewusst nicht allzu komplex angelegt, damit wir rasch starten können und sich möglichst viele Teilnehmer schnell zurechtfinden. Es scharren schon diverse Elektor-Autoren und sonstige Leser ungeduldig mit den Füßen! Daher glaube ich, dass sich die Site nach dem Start im März schnell mit schönen Projekten füllen wird.
Bisher geht alles noch über den traditionellen Weg (per E-Mail an die Redaktion). Was meine Kollegen und mich rund ums Jahr an Projekt-Vorschlägen erreicht, spiegelt natürlich auch die gegenwärtigen Trends in der Elektronik wieder. Da ist es wahrscheinlich kein Zufall, dass wir in den letzten Monaten gleich mehrere Projekte erhielten, die sich mit der Steuerung eigener Elektronik per Android-Smartphone und -Tablet beschäftigen. Die kleinen Computer sind mit reichlich Rechenpower, vielerlei Sensorik und einem tollen Display ausgestattet, da drängt sich die Anwendung als Steuerung der eigenen Elektronik geradezu auf. Wir haben uns daher nicht gescheut, im letzten und in diesem Heft gleich zwei mögliche Lösungen vorzustellen, denn beides sind sehr schöne Entwicklungen, die bestimmt vielen Lesern gefallen werden.
Neben Android ist Arduino ein anderes Trend-Thema bei unseren Lesern. Gleich drei Artikel in diesem Heft drehen sich um das immer beliebter werdende Mikrocontroller-Entwicklungssystem. Und auch die Kombination von Android und Arduino ist im Heft vertreten – diese Verbindung ist naheliegend, weil beides Open source ist und man sich in beiden Fällen mit immer mehr Mitstreitern rund um den Globus austauschen kann.
Viel Spaß mit der neuen Ausgabe!

Jens Nickel







INHALT

43. Jahrgang
März 2012
Nr. 495

Unser Team

Chefredakteur:	Jens Nickel (v.i.S.d.P.) (redaktion@elektor.de)
Ständige Mitarbeiter:	Dr. Thomas Scherer, Christopher Rausch
Internationale Redaktion:	Harry Baggen, Thijs Beckers, Jan Buiting, Eduardo Corral, Wisse Hettinga,Denis Meyer, Clemens Valens
Elektor-Labor:	Christian Vossen (Ltg.), Thijs Beckers, Ton Giesberts, Luc Lemmens, Raymond Vermeulen, Jan Visser
Herausgeber:	Don Akkermans
Grafik & Layout:	Giel Dols, Jeanine Opreij, Mart Schroijen

Unser Netzwerk

Internationale Teams

Großbritannien
Wisse Hettinga
+31 (0)46 4389428
w.hettinga@elektor.com

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

Deutschland
Ferdinand te Walvaart
+49 (0)241 88 909-0
f.tewalvaart@elektor.de

Frankreich
Denis Meyer
+31 (0)46 4389435
d.meyer@elektor.fr

Niederlande
Harry Baggen
+31 (0)46 4389429
h.baggen@elektor.nl

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

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

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

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

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

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

Russland
Nataliya Melnikova
8 10 7 (965) 395 33 36
nataliya-m-larionova@yandex.ru

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

Südafrika
Johan Dijk
+27 78 2330 694 / +31 6 109 31 926
J.Dijk @elektor.com

China
Cees Baay
+86 (0)21 6445 2811
CeesBaay@gmail.com

IMPRESSUM
43. Jahrgang, Nr. 495 März 2012
Erscheinungsweise: 11 x jährlich (inkl. Doppelheft Juli/August)

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

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

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

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

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

Die Elektor Community

215132 Mitglieder in **83** Ländern...

Noch kein Mitglied?

www.elektor.de/community

Unsere Partner und Sponsoren

Sie möchten Partner werden?

Kontaktieren Sie uns bitte unter service@id-medienservice.de (Tel. 0511/616595-0) bis zum 24. Januar 2012.

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

DesignSpark chipKIT™ Design Challenge

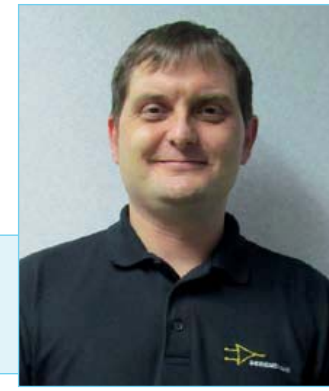
Auch im letzten Monat des Wettbewerbs für energiesparende Anwendungen mit DesignSpark und chipKIT™ können noch Beiträge eingereicht werden – wenn man schnell ist...

Von Ian Bromley (UK)

Der DesignSpark-chipKIT-Wettbewerb erreicht nun die Phase der letzten Wochen, doch es ist immer noch genug Zeit für eine Teilnahme und die Chance auf einen Gewinn. Es sind Preise mit einer Gesamtsumme von 10.000 $ ausgelobt, wobei allein der erste Preis mit 5.000 $ honoriert wird. Die Begeisterung für diesen Wettbewerb ist außerordentlich erfreulich. Besonders beeindruckt sind wir von der Qualität der bisherigen Einsendungen. Wer zum ersten Mal davon hört: Der Wettbewerb mit DesignSpark & chipKIT™ soll Ingenieure, Studenten und Elektronik-Interessierte dazu motivieren, neue und energieeffiziente Anwendungen zu entwickeln, die einen Beitrag zum Umweltschutz darstellen.
Auf dem jetzigen Höhepunkt des Wettbewerbs sind schon einige Teilnehmer mit ihrer Entwicklung auf Basis der Arduino-kompatiblen Entwicklungsplattform chipKIT™ Max32™ von Digilent sehr weit fortgeschritten. Das Kit verwendet den 32-bit-Mikrocontroller PIC32 von Microchip und ermöglicht Entwicklern die einfache und preiswerte Integration von Elektronik in ihre Projekte. Um Sie noch ein wenig neugieriger zu machen, erzählen wir Ihnen nun etwas über die bisherigen Einsendungen, die alle auch auf der DesignSpark Community-Webseite unter www.designspark.com zu finden sind.
Ein sehr interessantes Projekt dreht sich um ein UUV (**U**nmanned **U**nderwater **V**ehicle = unbemanntes Unterwasserfahrzeug), das zwecks Fortbewegung gleitet, was die Betriebsdauer dank des geringen Energiebedarfs steigert. Ein hierfür entwickeltes Erweiterungsboard dient zum Anschluss der für den Betrieb des UUV notwendigen Sensoren und Aktoren. Die Schaltung enthält einen Beschleunigungssensor, ein Gyroskop und ein Magnetometer zur Orientierung unter Wasser – jeweils in dreiachsiger Ausführung. Außerdem gibt es noch einen GPS-Empfänger zur Positionsbestimmung an der Oberfläche. Eine Auftriebsmaschine steuert zusammen mit Aktoren für Nicken und Rollen die Bewegung durchs Wasser. Ein zusätzlicher Sensor misst die Leitfähigkeit, Temperatur und Tiefe des Wassers und legt die Daten im Flash-Speicher ab. Man kann so Profile des Salzgehalts von Gewässern erstellen.
Ein anderes Projekt dreht sich um Hydrokultur, wobei Wasser und Nährstoffe passend zu den klimatischen Bedingungen und den Erfordernissen der in Hydrokultur gehaltenen Pflanzen gesteuert werden. Somit wird nicht nur Wasser, sondern auch Chemie gespart. In den meisten Hydrokulturen wird nach einem etablierten Schema vorgegangen, das nicht an die äußeren Bedingungen angepasst ist. Neben der Schonung der Umwelt kann diese Anwendung auch finanzielle Einsparungen bringen.
Ein Projekt zum Thema „intelligenter Garten“ überwacht die natürlichen Prozesse von Pflanzen bei minimaler menschlicher Intervention. Das Projekt zielt auf die Kombination von erneuerbaren Energiequellen wie Solarenergie mit einem besonders niedrigen Energieverbrauch der Steuerelektronik ab. Das System enthält etliche Sensoren für Feuchtigkeit, Temperatur und Licht sowie Aktoren wie eine Bewässerungspumpe. Außerdem handelt es sich um ein modulares System, das recht einfach erweitert werden kann, um etwa per Bluetooth Meldungen zu generieren. Ein Beispiel auch für das um sich greifende „Internet der Dinge“, von dem man in den nächsten Jahren sicherlich noch mehr hören wird.
Das waren nur drei von sehr vielen, hoch interessanten Projekten. Ein letzter Hinweis an die Teilnehmer und solche, die es noch werden wollen: Alle Beiträge müssen die Entwicklung einer Erweiterungskarte umfassen, die mit der kostenlosen Platinen-Layout-Software DesignSpark von RS Components erstellt wird. Die notwendige Firmware muss mit Hilfe von Microchips MPLAB®-IDE-Software entwickelt werden. Auch Neueinsteiger sind herzlich willkommen und können sich gerne in der Online-Community DesignSpark einbringen und über ihr Projekt informieren, indem sie über Updates und Fortschritte berichten, andere Projekte kommentieren und mithelfen, gute Ideen zu entwickeln. Alle Teilnehmer sind automatisch auch für die Community Choice Awards qualifiziert, die zusätzlich für die beste Kooperation vergeben werden. Hierbei kann man Gutscheine gewinnen, die man gegen Produkte aus dem Sortiment von RS Components / Allied Electronic eintauschen kann.
Die Wettbewerbsbeiträge werden nach dem Grad an Energieeffizienz und der Qualität des Platinen-Designs der Erweiterungskarte beurteilt. Einsendeschluss ist der 27.03.2012. Die Gewinner werden Anfang April 2012 benachrichtigt.

(120188)

Weitere Details und die Registrierung für den DesignSpark & chipKIT™-Wettbewerb finden sich unter **www.chipkitchallenge.com**.



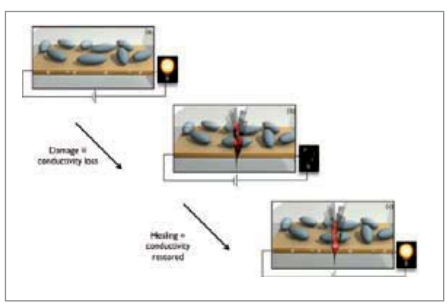
Ian Bromley ist Ingenieur bei RS Components und der Projektleiter der Platinen-Layout-Software DesignSpark. Vor seiner Tätigkeit für RS war er viele Jahre im Support von Texas Instruments sowie als Field Applications Engineer tätig.

Von Phil Knurhahn

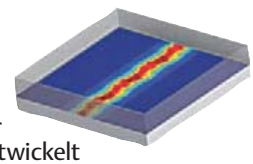
Selbstreparierende Elektronik

Was sich so unglaubwürdig anhört, haben die Professoren Nancy Sottos, Scott White und Jeffrey Moore (Foto: L. Brian Stauffer) von der University of Illinois in Champaign realisiert. Sie haben ein selbstheilendes System entwickelt, dass die elektrische Leitfähigkeit in einem unterbrochenen Schaltkreis wiederherstellt, innerhalb von Millisekunden. Das ist besonders für elektrische Systeme in der Luft- und Raumfahrt interessant, wo Wohl und Wehe von Menschen bedroht sind. Das Team verwendet Polymermaterial als Träger der Leitung, das speziell präpariert wurde. Dazu wurden winzige Mikrokapseln mit etwa 10 µm Durchmesser auf einer Leitung aus Gold angebracht, die mit flüssigem Metall gefüllt sind. Bricht diese goldene Leitung, dann brechen die Mikrokapseln auf, das Metall verteilt sich und füllt die Bruchstelle. Die Unterbrechung in der Funktion dauert nur Mikrosekunden, da das flüssige Metall sich im Nu in die Bruchstelle verteilt und damit die Leitfähigkeit wieder herstellt. Bei den Versuchen wurden 90 % der Bruchstellen wieder „geheilt“ und dabei 99 % der ursprünglichen Leitfähigkeit erreicht (Grafik: Scott White). Da sich die Kapseln nur an der Bruchstelle öffnen, bleibt der Rest der Schaltung unberührt. Gerade in Flugzeugen und Raumkapseln sind viele Kilometer an Leitungen verlegt und eine Reparatur während des Flugs ist schon deshalb völlig ausgeschlossen, weil man gar nicht weiß, wo die Bruchstelle ist. Das System könnte sich für beliebige Schaltkreise (sogar in Chips) eignen. Zunächst konzentrieren sich die Arbeiten aber auf Batterien und deren Zuleitungen, da es hier ja viele Anwendungen mit riesiger Stückzahl gibt.

http://news.illinois.edu/news/11/1220self-healing_ScottWhite_NancySottos_JeffreyMoore.html



Das Ohmsche Gesetz - im atomaren Maßstab



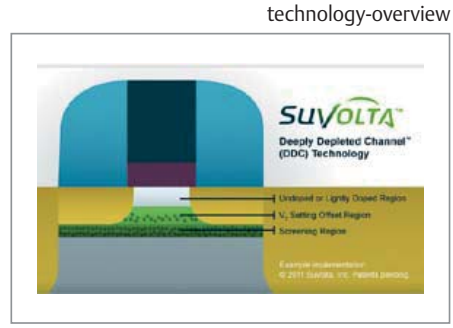
Den kleinsten Silizi-umdraht, der je entwickelt wurde, zeigten Forscher der Purdue Universität und der Universität New South Wales, Melbourne. Er ist gerade einmal ein Atom lang und vier Atome breit (Bild: Purdue Universität, Sunhee Lee, Hoon Ryu und Gerhard Klimeck). Und er scheint in vielerlei Hinsicht den großen Brüdern gleich zu sein: Er hat die gleiche Stromleitfähigkeit wie die dünnsten Leitungen auf den heutigen Chips und auch die (eigentlich unerwünschte) elektrische Kapazität zwischen zwei solchen Leitungen. Dabei sind diese atomaren Drähte 20 Mal dünner sind als die dünnsten heute herstellbaren Kupferleitungen auf den Chips. Aus Computersimulationen und den Mustern zogen die Forscher verschiedene Schlüsse: Ingenieure können nun eine Roadmap für die weitere Entwicklung von Computerbausteinen im Nanomaßstab planen. Bei einem Atom schließlich endet dann auch Moores Law endgültig. Für Computerwissenschaftler rücken silizium-basierte Quantencomputer näher ins Blickfeld. Und Physikern wird das legendäre Gesetz von Georg Simon Ohm aus dem Jahr 1826 bestätigt, das den Zusammenhang zwischen elektrischem Strom, elektrischem Widerstand und angelegter Spannung in einer Leitung beschreibt – was auch fast 200 Jahre später für atomare Siliziumleitungen weiter gilt.

www.purdue.edu/newsroom/research/2012/story-print-deploy-layout_1_17232_17232.html

CMOS mit halbiertem Stromverbrauch

Das 2006 gegründete Start-up-Unternehmen SuVolta hat sich auf die Fahnen geschrieben, unnötigen Energieverbrauch von CMOS-Chips infolge von Leckverlusten drastisch zu senken. Vor allem die stetig wachsende Zahl von Mobilanwendungen auf Smartphones und Tablets ist hier eine wichtige Triebfeder, denn die Anwender wollen immer länger mit einer Batterieladung arbeiten können. Mit einer neuen „PowerShrink“-Plattform wird den Chipherstellern jetzt eine Technologie angeboten, die den Stromverbrauch um satte 50 % senkt, ohne die Leistungsfähigkeit der damit gebauten CMOS-Chips einzuschränken. Wenn mit dieser Technologie auch noch eine entsprechende Spannungsreduzierung einhergeht, dann kann man den Stromverbrauch sogar um 80 % senken. Nach Angaben von SuVolta lässt sich die Technologie derzeit herunter bis zu Chipstrukturen von 20 nm einsetzen. Das dem Verfahren zugrunde liegende Prinzip verwendet einen „Deeply Depleted Channel“, der mit extrem verringerter Dotierung arbeitet (Grafik: SuVolta), die quasi auf null reduziert ist und dem Verfahren den Namen gegeben hat. Das Unternehmen wurde von ehemaligen Mitarbeitern von LSI und Fairchild gegründet.

www.suvolta.com/technology/technology-overview



Galvanische Trennung für USB

Eine häufige Ursache für ungenaue oder verfälschte Messungen ist die gegenseitige Beeinflussung von Messgeräten durch ungewollte Querströme. Wer Messungen über ein am PC angeschlossenes USB-Gerät durchführt, kann diesem Phänomen nun entgegenwirken. Das Unternehmen Cesys bietet ein Gerät an, das in der Lage ist, USB-Komponenten galvanisch voneinander zu trennen und so störende Ströme durch das USB-Kabel zu verhindern. Der USB-Isolator wird dazu zwischen Messgerät und PC geschaltet und ermöglicht so auch Messungen, bei denen sich das Signal nicht auf das Massepotential des Host-PCs bezieht. Da laut Hersteller keine zusätzlichen Signalverzögerungen auftreten, eignet sich der USB-Isolator auch in der Audiotechnik dazu, USB-bedingte Störgeräusche oder Brummschleifen zu verhindern. Sollen über den USB-Port des PCs leistungsstarke Anlagen oder Maschinen gesteuert werden, schützen die integrierten Überspannungsableiter den Steuer-PC zusätzlich vor einer möglichen Zerstörung im Fehlerfall.
Nach Angaben des Herstellers ist das Gerät mit allen Betriebssystemen kompatibel, da für die Nutzung keine weiteren Treiber installiert werden müssen. Für einen Preis von 84,02 € (zzgl. MwSt.) ist der USB-Isolator entweder direkt bei Cesys oder im Fachhandel erhältlich.

www.cesys.com

SO-DIMM-Computer

Die Firma Strategic Test hat ein neues Mitglied ihrer preiswerten Reihe von auf Freescale-Controllern basierenden Computer-Modulen herausgebracht. Das Modell TX-28S ist komplett auf einer SO-DIMM-Platine mit 200 Pins untergebracht, die nur 68 x 25 mm groß ist. Der auf einem ARM926EJ-S-Core basierende i.MX283-Controller hat

Nachrichten aus Forschung und Technik, interessante Produkt-Neuheiten und vieles mehr findet man aktuell unter www.elektor.de!

64 MB an DDR-SDRAM und 128 MB NAND-Flash-Speicher zur Verfügung. Das Modul eignet sich besonders für lüfterlose Anwendungen, wo geringe Kosten, kleine Abmessungen und ein niedriger Stromverbrauch wichtig sind. Für industrielle Anwendungen wichtig ist, dass der Hersteller eine Verfügbarkeit von sieben Jahren garantiert.

Das Controller-DIMM bringt eine 10/100-Mbit/s-Ethernet-Schnittstelle, zwei High-speed-USB-2.0-Ports, einen Controller für Farb-LCDs samt 4/5-Draht-Touchscreen-Interface mit und verfügt über UART, SD-Card-Interface, I²C, PWM und serielle Audio- und SPI-Interfaces. Der Typ TX-28S ist pinkompatibel mit den anderen Modellen auf Basis von Freescales i.MX-Controllern, was den Aufbau skalierbarer Systeme ermöglicht. Alle Module können mit der Hardware-Referenz-Plattform „Development Kit-5“ ausgeliefert werden, womit sowohl Linux als auch Windows Embedded CE 6.0 von Microsoft unterstützt werden.

http://strategic-embedded.com/

Business-Katalog von Conrad

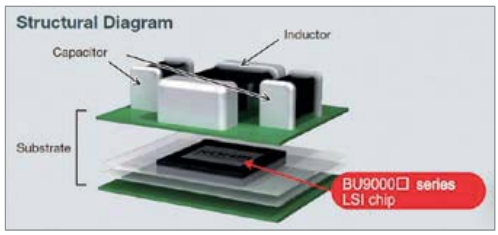
Der Distributor Conrad Electronic hat einen neuen Produktkatalog für seine Geschäftskunden veröffentlicht. Der neue Band I umfasst neben 2.300 Seiten mit Schwerpunkt Bauelemente, Messtechnik, Automation, Werkstatt und Kabel weitere 330 Seiten mit Produktneuheiten aus den Bereichen Computer/IT, Gebäude- und Empfangstechnik. Zusammen mit dem im Juli 2011 erschienen Band II des Produktkatalogs haben Geschäftskunden eine Auswahl von mehr als 220.000 Produkten für den professionellen Einsatz. Die Conrad Exklusivmarken Voltcraft® und Conrad energy feiern dieses Jahr Jubiläum, und so gibt es einige Aktionsangebote. Für Elektroniker interessant ist beispielsweise das digitale 2-Kanal Speicheroszilloskop DSO-3062C mit 60 MHz Bandbreite, 500 MSamples/s Realtime-Samplingrate und einer Speichertiefe von 512 kPts pro Kanal. Als Aktionsangebot ist dieses Einsteiger-Oszilloskop inklusive zwei Tastköpfen für 251,26 € (zzgl. MwSt.) erhältlich.

Der neue Business-Katalog ist ab Ende Januar 2012 auch als Download für die hauseigene iPad-App von Conrad verfügbar.

www.conrad.biz

Extrem kleiner DC/DC-Konverter

Die fortschreitende Miniaturisierung führt zu immer kleineren Bauteilen. Der Hersteller ROHM Semiconductor hat mit dem IC BU90oxx einen mit 6 MHz operierenden Schaltregler-Chip im WLCSP-Gehäuse mit den Abmessungen 1,3 x 0,9 x 0,4 mm entwickelt; außerdem wurden damit aufgebaute ultra-kompakte Module angekündigt, die neben der Spule und den Kondensatoren auch alle anderen Bauteile in einem Volumen von 2,3 × 2,9 × 1 mm unterbringen.
In den letzten Jahren haben gerade mobile Geräte wie Smartphones immer mehr Funktionen bekommen, was die Anzahl an zusätzlichen Bauteilen erhöhte und damit einhergehend zum gesteigerten Bedarf nach mehr unterschiedlichen Versorgungsspannungen führte. Die komplexer gewordenen Stromversorgungen mit diversen Energiesparkonzepten machen kleine und effiziente Module notwendig.
Das DC/DC-Konverter-Modul BZ6A integriert ein IC der BU90oxx-Serie mit einer Schaltfrequenz von 6 MHz direkt in die



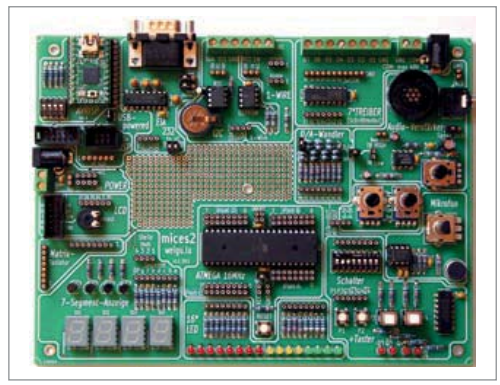
Platine, welche auch alle weitere notwendigen Bauteile mitbringt, sodass sich ein besonders kleines Modul mit hoher Qualität ergibt. Mit solchen Modulen ist eine platzsparende Konstruktion möglich, die aufgrund ihrer Einfachheit auch Entwicklungszeit spart.

www.rohm.com/eu/highlights/product-news/bz6a-series.html

AVR-Experimentierboard

Auf der Internetseite weigu.lu finden Schüler, Studenten und Technikinteressierte viele Informationen rund um das Thema Mikrocontroller und deren Programmierung. Neben einem umfangreichen Assemblerkurs für AVR-Mikrocontroller bietet Weigu unter anderem auch Assembler-Bibliotheken und detailliertes Material zum Selbstbau eines Programmieradapters an.
Neu im Programm ist ein Experimentierboard für AVR-Controller, welches selbst aufgebaut werden kann. Das Experimentierboard MICES2 verfügt über ein integriertes USB-Programmiergerät und neben verschiedenen Schnittstellen (seriell, I2C, 1-Wire) über viele Ein- und Ausgänge. Dazu gehören sowohl 16 LEDs, 4 Taster, 8 Schalter und eine 4-stellige Siebensegmentanzeige als auch Anschlüsse für eine Matrixtastatur, ein LCD oder einen Schrittmotor. Auch ein D/A-Wandler, eine Mikrofonschaltung und ein Audio-Verstärker sowie ein externes EEPROM und eine externe Echtzeituhr (RTC) gehören zu den Komponenten des MICES2-Boards. Mit dieser Ausstattung können alle Aufgaben des Assembler-Kurses durchgeführt und eigene kreative Ideen entwickelt werden.
Das Board wird wahlweise über USB oder ein externes Netzteil mit Strom versorgt. Die einzelnen Komponenten des Boards sind in Baugruppen aufgeteilt, die unabhängig voneinander funktionieren. Dieses modulare Konzept ermöglicht es, die Platine schrittweise aufzubauen und zu erweitern. Die für den Aufbau erforderlichen Dateien sind auf der Internetseite des Entwicklers frei verfügbar. Außerdem steht dort eine umfangreiche Dokumentation zum Download bereit, die Informationen über die einzelnen Baugruppen und nützliche Tipps für den Nachbau des Boards enthält.

http://weigu.lu/b/mices2/



MAILBOX

Spitze Grundlagenartikel

Zurück zu den Wurzeln, Elektor 1/2012, S. 48 (120001)

Den neuen Grundlagenartikel im letzten Heft finde ich einfach spitze. Nicht jeder Leser ist ein ausgemachter Vollblutelektroniker. Neue Themen finden sich genug: Transistortechnik, Operationsverstärkertechnik, Messtechnik usw.

Ich denke, so etwas war längst überfällig. Bitte weiter so!!!

Dirk Pohlemann

Danke für das Kompliment, das wir gerne an den Autor Burkhard Kainka weitergeben!

UPDATES & ERGÄNZUNGEN

2,4-GHz-Fernsteuerung für Modellflugzeuge

Elektor 9/2011, S. 54 (110109)

Für D2 & D3 sollte man statt des Typs ES3A einen neuen Typ verwenden: GF1A.
R15 muss den Wert 3k3 haben, nicht 47k.

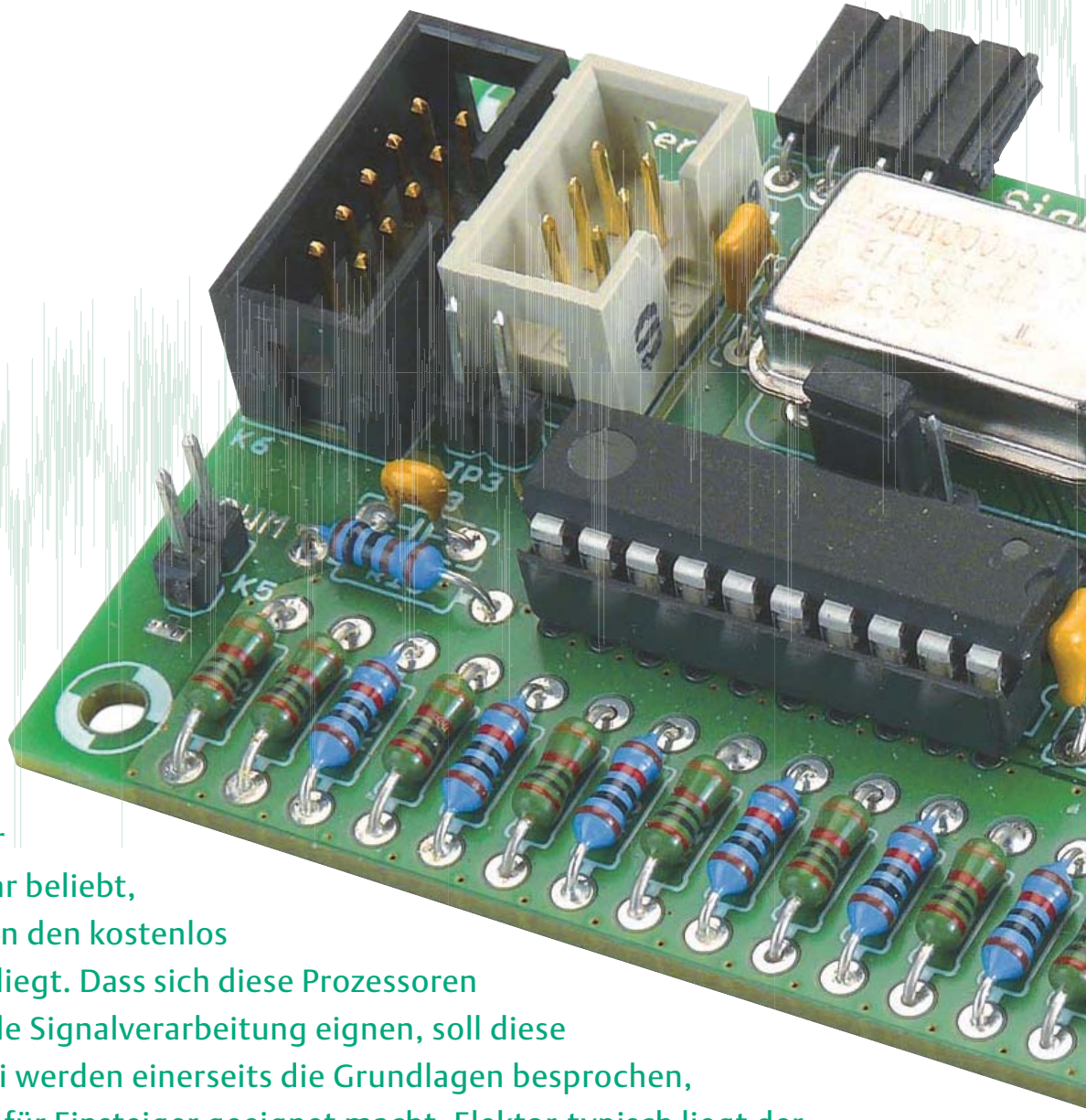
Der Wert des Widerstands R3* sollte 6k8 sein, wenn man ein LCD des Typs Powertip PC2004ARS-AWA-A-Q verwendet.

Es gibt auch ein Software-Update V1.01 (Fehler behoben), im Software-Download finden sich darüber hinaus neue Fotos (www.elektor.de/110109).

SDR mit AVR

ATtiny erzeugt hochgenaue

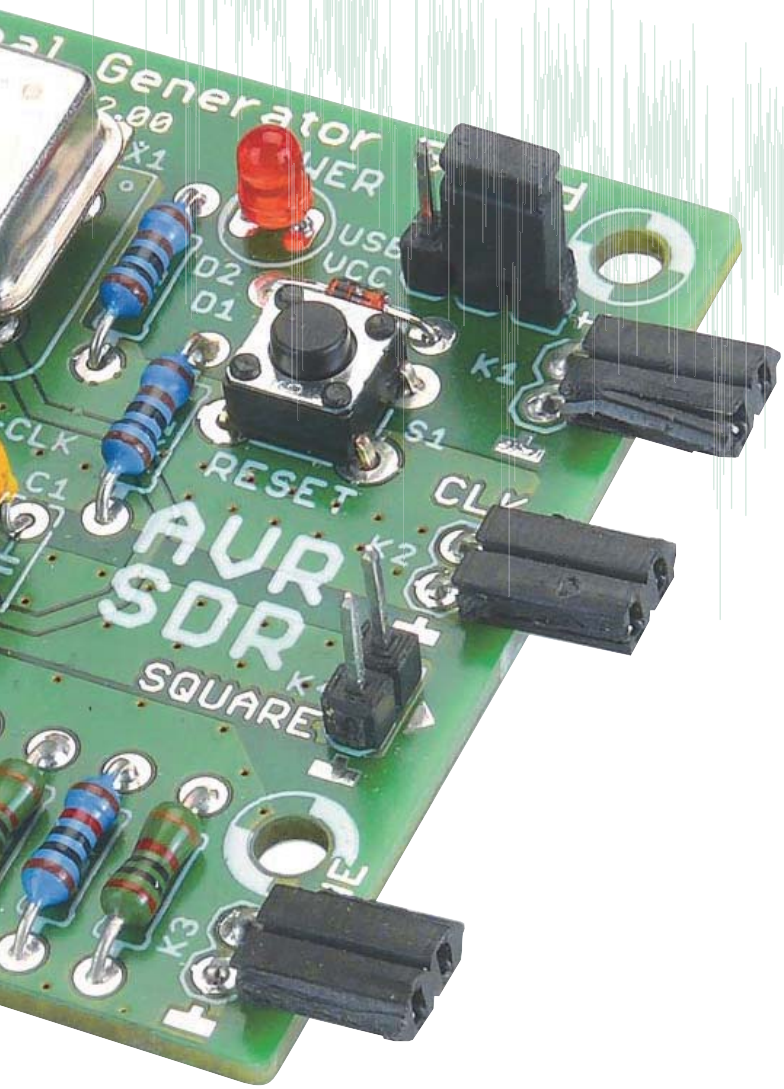
Von Martin Ossmann (D)



Die AVR-Controller von Atmel sind sehr beliebt, was nicht zuletzt an den kostenlos erhältlichen Tools liegt. Dass sich diese Prozessoren auch für die digitale Signalverarbeitung eignen, soll diese Serie zeigen. Dabei werden einerseits die Grundlagen besprochen, was die Serie auch für Einsteiger geeignet macht. Elektor-typisch liegt der Schwerpunkt aber auf den Experimenten. Als Hardware kann man einen eigenen Aufbau nutzen; noch bequemer hat man es mit den Boards, die von Elektor angeboten werden. Die Software kann man wie immer im Quellcode von unserer Website downloaden. Los geht‘s mit der Generierung von Signalen!

Teil 1

Signale



Um noch ein wenig mehr Appetit zu machen: Im Laufe der Serie werden wir nicht nur Signale generieren (wozu die erste Platine dient, die mit einem ATtiny2313, einem 20-MHz-Oszillator und einem R2R-DAC ausgerüstet ist). Wir werden auch Signale aus dem Äther fischen: Eine zweite Platine enthält Hardware, die zum Bau eines digitalen Empfängers (Software Defined Radio) notwendig ist. Neben einer RS232-Schnittstelle ist ein LC-Display integriert sowie ein 20-MHz-VCXO, der später an ein Referenzsignal angekoppelt werden kann. Die dritte Platine dient zum Aufbau einer aktiven Ferritantenne. Die Software für alle Projekte wurde mit dem WINAVR-GCC Compiler im AVR-Studio erstellt. Von der Elektor-Website sind jeweils der C-Quellcode (mit Fuse-Einstellungen) und der Hex-Code downloadbar.

Die Serie ist rund um die Experimente aufgebaut. Freuen darf man sich auf Sinus- und Rechteckgeneratoren, ein RMS-Voltmeter, Versuche zu FM, AM und PM, FIR- und IIR-Filter, eine drahtlose Datenübertragung sowie den Empfang des DCF-Signals, des RTTY-Wetterfunks, verschiedener Langwellen-Signale der BBC ... und vieles mehr!

Vorneweg noch ein Hinweis: Energiesparlampen arbeiten mit Schaltnetzteilen, deren Oberwellen den Empfang von Sendern im Langwellenbereich stark stören. Bei unseren Experimenten heißt es also: Energiesparlampen aus und Hochvolt-Halogenlampe oder Kerze an!

Signalgenerator-Board

Unser Signalgenerator besteht im Wesentlichen aus einem mit 20 MHz getakteten AVR-Mikrocontroller und einem R-2R-Digital-/Analogwandler zur Ausgabe von Spannungswerten. So ein Generator ist natürlich nicht neu. Wir werden aber zeigen, wie man diese einfache Schaltung für viele Zwecke verwenden kann. Insbesondere werden wir Signale erzeugen, um damit Tests an anderen Baugruppen machen zu können. Hierzu gehören frequenz- und phasenmodulierte Signale. Später werden wir den Signalgenerator an eine Taktquelle anschließen, die selbst an ein Frequenznormal (z.B. DCF77 oder TDF auf 162 kHz) angekoppelt ist, was die Präzision erhöht.

Die Schaltung des Signalgenerators ist in **Bild 1** dargestellt. Zentrales Bauteil ist der Mikrocontroller ATtiny2313. An Port B ist ein 8-bit-R-2R-Kettenleiter als Digital-Analog-Wandler angeschlossen. Das analoge Signal kann am Anschluss K3 (SINE) entnommen werden. Man muss aber beachten, dass die Ausgangsimpedanz 10 kΩ beträgt, der Ausgang also relativ hochohmig ist. Der PWM-Ausgang OC1A des Controllers ist ebenfalls nach außen geführt (K4 SQUARE). Diesen Ausgang werden wir benutzen, um Rechtecksignale mit einigen Hundert kHz zu erzeugen, außerdem dient er uns

Elektor Produkte & Service

- Signalgenerator (Kit mit Platine und allen Bauteilen 100180-71)
- USB/TTL-Konverter BOB FT232, bestückt und getestet 110553-91
- USB-AVR-Programmer, SMD-bestückte Platine plus alle anderen Bauteile 080083-71
- Gratis Software-Download (Hex-Files und Source-Code)

Alle Produkte und Downloads sind über die Website zu diesem Artikel erhältlich: www.elektor.de/100180

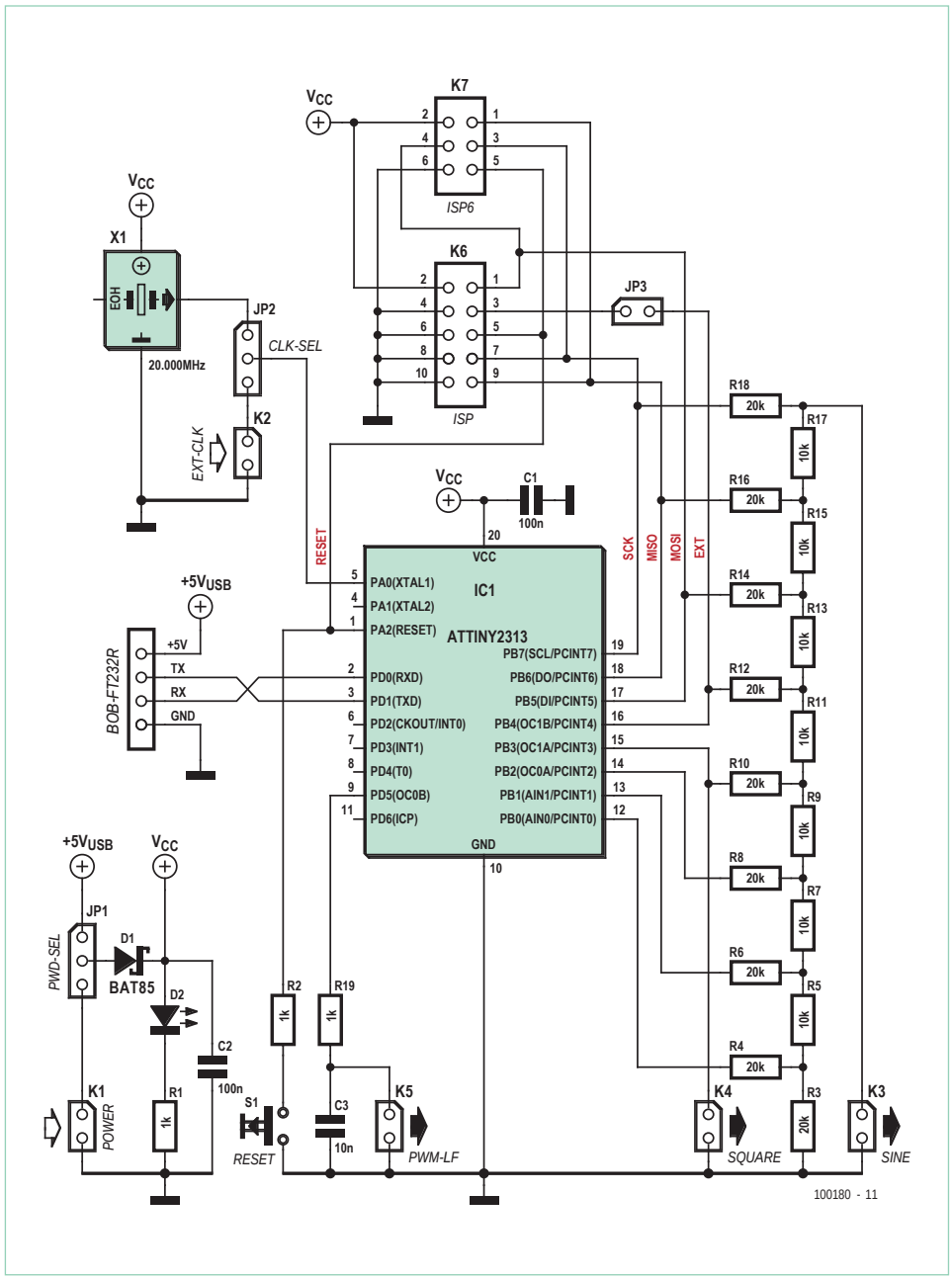


Bild 1. Schaltplan des Signalgenerators.

zur Modulation anderer Signale. Darüber hinaus wird der PWM-Ausgang OC0B über ein Tiefpassfilter R19/C3 nach außen geführt (K5 PWM-LF), womit man einen weiteren analogen Ausgang für langsame Signale hat.
Der Prozessor wird von einem 20-MHz-Oszillator X1 getaktet. Hier sollte man versuchen, ein relativ genaues Exemplar zu erhalten (50 ppm oder besser). Um Oszillatoren verschiedenen Typs (oder verschiedener Frequenz) auszuprobieren, sollte man am besten eine Fassung benutzen. Mit dem Jumper JP2 kann man auf ein externes Taktsignal umschalten, welches am Anschluss K2 (EXT-CLK) zugeführt wird.

Die Signalgenerator-Programme lassen sich teilweise von außen konfigurieren, hierzu dient der UART des Controllers. Auf der Platine, die bei Elektor als Kit mit allen Bauteilen erhältlich ist, sind die erforderlichen Pins auf einen Steckverbinder herausgeführt, an dem der USB/Seriell-Konverter BOB-FT232R [1] angeschlossen werden kann. Dies macht einen Anschluss an einen PC sehr einfach. Über JP1 lässt sich eine Stromversorgung via USB auswählen, wer das Gerät am PC nutzt, kann sich so einen extra Netzadapter sparen.

Die Bestückung der Platine (**Bild 2**) dürfte keine Schwierigkeiten bereiten: Bei den Bauteilen handelt es sich durchweg um bedrahtete Standardbauteile. Neben dem Taktgenerator sollte man auch den Prozessor sockeln. Bei den Programmier-Steckverbindern K6 und K7 ist auf die richtige Orientierung zu achten. Bei der Programmierung (hierzu kann man zum Beispiel den kleinen USB-AVR-Prog [2] von Elektor) benutzen, muss man auch auf die Fuse-Einstellungen achten. Daher sind die Fuses und Compileroptionen jeweils im Quellcode mit angegeben.

DDS-Sinusgenerator

Als erste Anwendung soll ein einfacher Sinusgenerator in C programmiert werden. Der grundlegende Abtast-Takt wird von einem der controllerinternen Timer geliefert, der einen Interrupt triggert. In der Interruptroutine muss dann immer der aktuelle Abstastwert (Sample) der Sinusschwingung erzeugt werden (**Bild 3**). Wir benennen das k-te Sample mit S[k]. Ist p[k] die zugehörige Phase, so gilt:

$$S[k] = \sin(p[k])$$

Zwischen den Samples schreitet die Phase jeweils einen konstanten Wert d (Phaseninkrement) weiter, d.h.

$$p[k+1] = p[k] + d$$

Für einen idealen Sinusgenerator müssten alle Berechnungen exakt durchgeführt werden, was jedoch viel zu lange dauern würde. Wir bedienen uns daher des DDS-Prinzips (Direct Digital Synthesizer). Die aktuelle Phase DDSp wird dabei mit endlicher Genauigkeit als m-Bit-Zahl im so genannten Phasenakkumulator gespeichert. Einer vollständigen Sinusperiode entspricht dann der Bereich von 0 bis 2^m-1. Die Darstellung und Addition des Pha-

Stückliste

Widerstände:
R1,R2,R19 = 1 k
R5,R7,R9,R11,R13,R15,R17 = 10 k
R3,R4,R6,R8,R10,R12,R14,R16,R18 = 20 k

Kondensatoren:
C1,C2 = 100 n (100 V)
C3 = 10 n

Halbleiter:
D1 = Schottky-Diode BAT85
D2 = LED grün
IC1 = ATtiny2313-20PU

Außerdem:
S1 = Taster (Mikro)
K4,K5 = 2x1 Stiftleiste (2,54 mm)
JP3 = 2x1 Stiftleiste (2,54 mm) mit Jumper
JP1,JP2 = 3x1 Stiftleiste (2,54 mm) mit Jumper
K1,K2,K3 = 2x1 Buchsenleiste, rechtwinklig
BOB = 4x1 Buchsenleiste, rechtwinklig

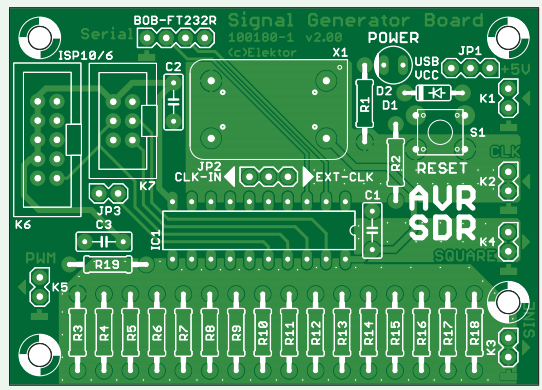


Bild 2. Die Platine ist mit allen Bauteilen als Kit bei Elektor erhältlich.

K6 = Wannenstecker für ISP, 2x5
K7 = Wannenstecker für ISP, 2x3
X1 = Quarz 20 MHz (mit 4x Sockelstift Harwin H3153F01)
BOB-FT232R-001 = USB/TTL-Konverter BOB (fertig bestückt und getestet 110553-91)
Platine

oder
100180-71 Kit mit Platine und allen Bauteilen

seninkrements erfolgen mit der gleichen Genauigkeit. Nun muss man den zum jeweiligen Phasenwert gehörigen Sinuswert bestimmen. Das macht man mit Hilfe einer Tabelle, in welcher die Sinuswerte einer kompletten Periode gespeichert sind. Wollte man für jeden der möglichen 2^m Phasenwerte einen Wert speichern, würde die Tabelle unhandlich groß. Man benutzt daher nur die vorderen n Bit zur Tabellenadressierung, die Sinustabelle muss also nur 2^n Werte beinhalten. Diese Werte selbst werden nun wieder nicht exakt gespeichert, sondern als r-Bit-Zahlen S[k], welche dann einem r-Bit-Digital-Analog-Wandler(ADC) zugeführt werden. Schematisch ist das Ganze in **Bild 4** dargestellt.

In unserem ersten Programm verwenden wir m=32 und n=8. Mit einem 32-bit-Phasenakkumulator lassen sich Signale mit einer sehr präzisen Frequenz erzeugen, da die zugehörigen Phasen entsprechend genau berechnet werden. Wir verwenden eine Sinustabelle mit 256=2^8 Werten und einen 8-bit-DAC (r=8). Im Programm *EXP-SinusGeneratorDDS-T1INT-V01.c* [3] ist eine feste Frequenz von 1 kHz eingestellt. Das Ergebnis sollte man sich mit einem Oszilloskop ansehen (**Bild 5**). **Listing 1** zeigt den Code der Interruptroutine.

Timing

Unseren DDS takten wir mit f_{DDSCLK} = 100 kHz. Um das zu einer zu erzeugenden Frequenz f gehörige Phaseninkrement zu erhalten muss man berechnen:

$DDSd = 2^n * f / f_{DDSCLK}$

Für f = 1 kHz erhält man:

$DDSd = 2^{32} * 1\ kHz / 100\ kHz = 42949673$

Genau diesen Wert findet man im C-Programm auch als Initialisierung für DDSd.

Die Formel lässt erkennen, dass man umso höhere Frequenzen erzeugen kann, je höher die Taktrate ist. Warum wurde sie zu 100 kHz gewählt? Dazu wurde zuerst das Timing der Interruptroutine gemessen. Wie man in obigem Listing sieht, haben wir den eigentlichen Berechnungscode in das Setzen und Rücksetzen von Portpin D.4 eingerahmt. Dann kann die Dauer der eigentlichen

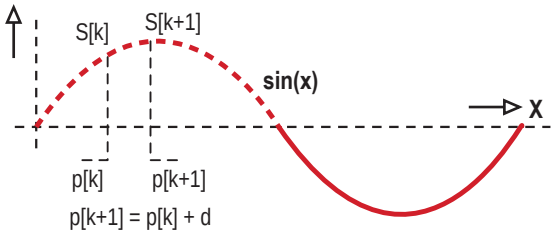


Bild 3. Abtastung eines Sinussignals.

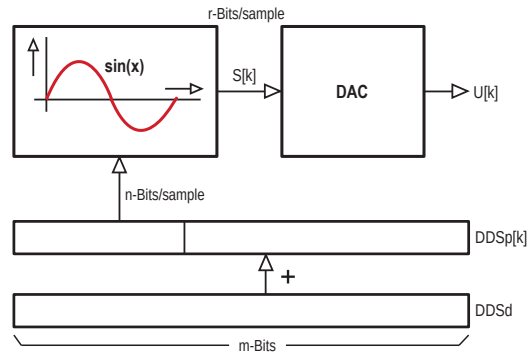


Bild 4. Schema des DDS-Sinus-Generators.

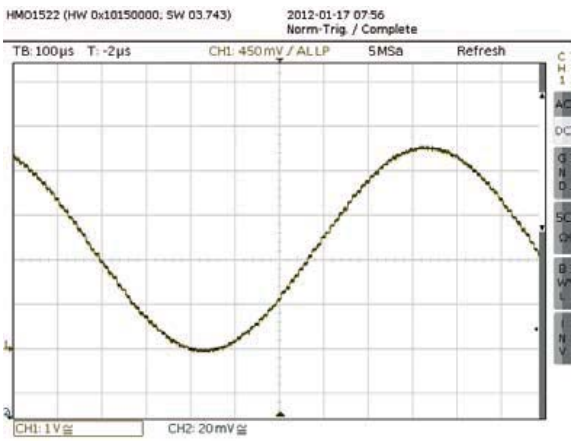


Bild 5. Test des Sinus-Generators.

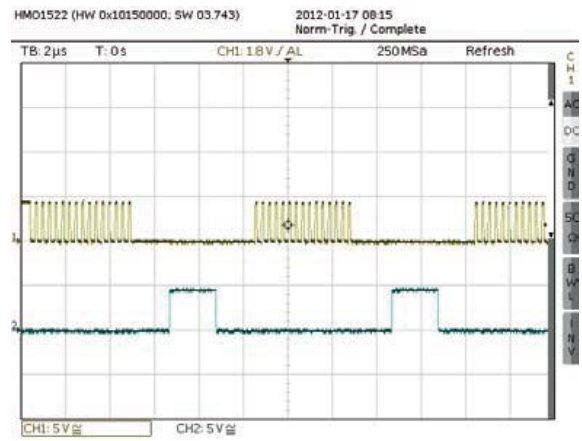


Bild 6. Messung des Interrupt-Timings.

Berechnung mit einem Oszilloskop bestimmt werden; in unserem Fall ergab sich rund 2,2 µs. Doch Vorsicht, damit ist nicht die gesamte Zeit zum Abarbeiten eines Interrupts erfasst! Es fehlen weitere Latenzzeiten, z.B. zum Retten und Wiederherstellen der Register. Diese kann man aber ebenfalls relativ gut unter Praxis-Bedingungen bestimmen.
Man lässt einfach als Hauptprogramm eine Endlosschleife laufen, in welcher ein Portpin (hier PD.5) mit maximaler Geschwindigkeit getoggelt wird. Immer wenn man auf dem Oszilloskop sieht, dass das Toggeln aussetzt, ist gerade die Interruptroutine aktiv. Das Ganze sieht dann wie in **Bild 6** aus.
In unserem Fall ergab sich eine Gesamtzeit zur Abarbeitung eines Interrupts von rund 5,4 µS. Die maximale Interrupt-Rate liegt also unter 180 kHz. Um hierzu genügend Abstand zu haben, haben wir 100 kHz gewählt.
Je höher die zu erzeugende Frequenz f im Vergleich zu f_{DDSCLK} wird, umso mehr machen sich „Unschönheiten“ des DDS-Prinzips durch Jitter, Rauschen und Nebenwellen im Spektrum bemerkbar. Mit unserer Taktrate von 100 kHz kommt man nicht höher als 10 kHz. Vielleicht können wir mit Assembler mehr erreichen?

Schnellerer DDS-Sinusgenerator

Um einen Sinusgenerator zu erhalten, der höhere Frequenzen erzeugen kann, wurde die DDS-Routine in Assembler umprogrammiert. Durch geschickte Anordnung der Variablen in den Registern konnte eine Abtastrate von 2 MHz für den 32-bit-DDS erreicht werden. Die trickreiche Programmierung **(Listing 2)** verwendet das T-Flag als Abbruchkriterium.
Gleichzeitig ist dies ein kombiniertes C/Assembler-Projekt, bei welchem die Sinus-Tabelle an einer bestimmten Speicher-Adresse beginnen muss. Die Konfiguration des Projektes innerhalb von WINAVR ist nicht unbedingt etwas für Anfänger. Wer also nicht selbst Änderungen vornehmen will, beschränke sich am besten darauf, einfach das fertige Hex-File in den Prozessor zu bren-

Listing 1

```
ISR(TIMER1_OVF_vect) {
  PORTD |=  _BV(4) ;                        // start signalize timing
  PORTB=pgm_read_byte( SIN8+(DDSp>>24)) ;   // fetch and output sine-sample
  DDSp += DDSd ;                            // advance DDS phase DDSp by DDSd
  PORTD &= ~  _BV(4) ;                      // end signalize timing
  }
```

Listing 2

```
loop:
    add  DDSphase0,DDSdelta0              // 1     LSB of 32 bit DDS adder
    adc  DDSphase1,DDSdelta1              // 1
    adc  DDSphase2,DDSdelta2              // 1
    adc  ZL       ,DDSdelta3              // 1     MSB is in ZL as pointer
    lpm  R0,Z                             // 3     access sine-table
    out  PORTB,R0                         // 1     out to R-2R DAC at PORTB
    brtc loop                             // 2 (1) loop until T-flag set by interrupt routine
                                          // 10 cycles in total for one loop
```

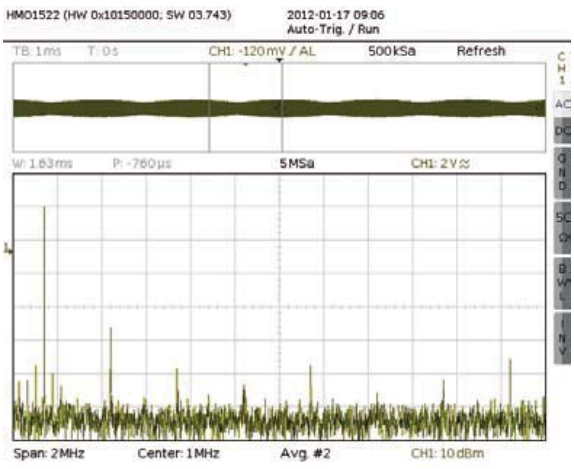


Bild 7. Spektrum des generierten Signals.

nen (an die Fuse-Bits denken). Das Projekt trägt den Namen *EXP-SinusGenerator-DDS-ASM-C-V01*.
Damit der Sinusgenerator flexibel verwendet werden kann, wurde er mit einer Bedienung via UART-Schnittstelle (19200 Baud, 8N1) versehen. Im Terminalprogramm gibt man einfach den Zahlenwert der zu erzeugenden Frequenz ein (plus CRLF) und sendet die Zeichen an den ATtiny. Die maximale Frequenz, bei der das Signal noch brauchbar ist, liegt bei rund 200 kHz. Die Auflösung der Frequenzeinstellung liegt theoretisch bei:

$f_{DDSCLK} / 2^n = 2\ MHz / 2^{32} = 0{,}00046...\ Hz$

Um diese feine Einstellmöglichkeit nutzen zu können, kann man bei der dezimalen Eingabe der Frequenz bis zu drei Nachkommastellen (mit Dezimalpunkt) angeben, z.B. also „1000.045 CRLF“.
Um aus der eingegebenen Frequenz die Parameter der DDS zu berechnen, braucht man eine hohe Rechengenauigkeit. Dazu hat der Autor entsprechende Arithmetikroutinen selbst programmiert, unter anderem zur Division mit Nachkommastellen.
In **Bild 7** ist das Spektrum eines erzeugten Sinussignals mit f = 125,123 kHz von 0 Hz bis 2 MHz dargestellt. Man erkennt, dass Oberwellen auftreten, die aber 30 dB unter dem Nutzsignal liegen. Weiter ist ein breiter Rauschteppich zu sehen, der vom DDS-Prinzip herrührt.

Listing 3

```
uint32_t DDS24 ;                          // DDDS phase, 24 bits used
volatile uint32_t dDDS24 ;                // delta for DDS
uint16_t TOP1 ;                           // integer part of divider for PWM

ISR(TIMER1_OVF_vect) {
  PORTD |=  _BV(4) ;
  DDS24 += dDDS24 ;                       // advance DDS phase
  if (DDS24 & 0x1000000UL ) {             // check bit 24 for overflow
    ICR1 =TOP1 ;                          // on overflow PWM width = TOP1+1
    }
   else {
    ICR1 =TOP1-1  ;                       // else PWM width = TOP1
    } ;
  DDS24 &=0xffffffUL ;                    // make DDS24 24 bits again
  PORTD &= ~  _BV(4) ;
  }
```

Benutzt man einen normalen Quarzoszillator, hat man es mit Ungenauigkeiten von +/-100 ppm zu tun. Da macht es dann nicht unbedingt Sinn, unseren Generator 100,00005 kHz erzeugen lassen zu wollen. Doch hat der Generator auch einen externen Eingang für einen 20-MHz-Takt. In einer späteren Folge werden wir beschreiben, wie man einen 20-MHz-Takt mit Hilfe eines Referenzsenders generiert. Dann lassen sich mit dem Sinusgenerator Signale mit sehr exakten Frequenzen erzeugen.

Abgleich von Schwingkreisen

Im späteren Verlauf der Serie werden wir mit Hilfe eines AVR-Mikrocontrollers Datenaussendungen im Langwellenbereich empfangen und auswerten (DCF 7,5 kHz, France Inter 162 kHz, BBC 198 kHz, ...). Dabei kommt meistens eine Ferritantenne zum Einsatz. Um diese abzugleichen, kann man sehr gut unseren Sinusgenerator verwenden. Man speist damit die Schaltung nach **Bild 8** und stimmt mit dem Drehkondensator auf Maximum ab.

Man kann die Phase zwischen der Ausgangsspannung U_{OUT} und der Eingangsspannung U_{IN} benutzen, um festzustellen, ob die Resonanzfrequenz des Schwingkreises höher oder niedriger als die Frequenz des eingespeisten Sinussignals ist. Eilt die Phase von U_{OUT} gegenüber U_{IN} vor, ist die Sinussignal-Frequenz höher als die Schwingkreis-Resonanzfrequenz. Eilt U_{OUT} der Spannung U_{IN} nach, ist die Signalfrequenz höher als die Schwingkreis-Resonanzfrequenz. Im Resonanzfall sind U_{IN} und U_{OUT} in Phase.

Im Schaltungsbeispiel sind die Bauteil-Werte für einen Schwingkreis von 125 kHz eingetragen. Als Spule L1 wird eine kleine Topfkernspule benutzt. Diese Schaltung werden wir später dazu benutzen, um Testsignale auf 125 kHz zu erzeugen. Mit dem Drehkondensator ist der Schwingkreis auf 125 kHz Resonanz abzustimmen. Als Eingangssignal kann man sowohl das Signal unseres R2R-DACs benutzen (K3), als auch das Rechtecksignal aus dem PWM-Ausgang (K4).

PWM-Rechteck mit fraktionalem Teiler

Wir kommen nun zu einer weiteren Anwendung des DDS-Prinzips. Verwendet man einen Timer mit PWM-Ausgang um ein Rechtecksignal zu erzeugen, kann man normalerweise nur ganzzahlige Bruchteile der Timer-Taktrate als Frequenz realisieren. Ist N der Teiler und f_{CLK} die Taktfrequenz, so ist $f=f_{CLK}/N$ die erzeugte Frequenz. Wenn man aber nun den Teiler laufend variiert (z.B. zwischen N und N+1), so lassen sich auch dazwischenliegende Frequenzen generieren. Verwendet man z.B. abwechselnd N und N+1 als Teiler, so dividiert man im Mittel durch N+0.5. Und wenn man z.B. durch 10,333333... teilen will, dann wählt man mit p=0,33333... „Wahrscheinlichkeit“ den Teilfaktor 11, in den restlichen Fällen aber N=10.
Wie können wir das in der Praxis umsetzen? Wir benötigen ein Verfahren, welches uns in Abhängigkeit vom zu realisierenden Teiler sagt, wann durch N, und wann durch N+1 zu dividieren ist.
Hier hilft unser m-bit-DDS-Generator – bei einem hohen m kann man sehr genaue Berechnungen durchführen. Wir benutzen diesmal den Überlauf des Phasenakkumulators. Die Rate p, mit der ein m-Bit-DDS-Phasenakkumulator Überläufe generiert, ist genau:

$$p = DDSd / 2^m$$

und diese Rate lässt sich über die Variable DDSd sehr präzise einstellen. Den Überlauf kann man nun benutzen, um den Timer anzuweisen, dass er statt durch N nun durch N+1 teilen muss.
Will man z.B. aus einem 20-MHz-Takt ein Signal mit einer Frequenz von 77,5 kHz erzeugen, muss man durch

$$20000 / 77.5 = 258{,}0645161...$$

teilen. Das heißt, man muss zwischen N=258 und N=259 umschalten, und zwar mit p = 0,0645161.... Bei einem 24-bit-DDS erhält man $DDSd = p \times 2^{24} = 1082401$. Eine Timer-Interrupt-Serviceroutine, die dieses Konzept umsetzt, zeigt **Listing 3**.

Das erzeugte Rechtecksignal jittert zwar quasi um die ideale Schwingung herum, stimmt aber im Mittel mit ihr überein.

Die Laufzeit der gesamten Routine inklusive Latenz beträgt wieder rund 6 µs, so dass wir auf diese Art Frequenzen bis ca. 160 kHz erzeugen können. Mit einer Assemblerroutine ließe sich das vermutlich noch um einiges steigern. Damit der Rechteckgenerator einfach verwendbar ist, haben wir ihn wieder mit einer einfachen Bedienung via Terminal versehen. Der Quellcode ist unter dem Namen *EXP-SquareGenerator-DDS-T1INT-V01.c* im Download-Zip [3] zu finden.
Das Prinzip des fraktionalen Teilers hat noch viele weitere Anwendungen. So kann man zum Beispiel eine beliebige Abtastrate aus einem Prozessortakt generieren (und gegebenenfalls noch in Form einer PLL nachregeln).

FM-Generator

Der Rechteckgenerator alleine ist vielleicht nicht so interessant. Da aber der Prozessor durch die PWM-Steuerung noch nicht vollständig ausgelastet ist, bleibt noch Rechenzeit übrig, um die Frequenz dynamisch zu variieren. So entsteht ein FM-Generator!
Der Deutsche Wetterdienst [4] sendet auf der Frequenz 147,3 kHz per Frequenzumtastung (FSK Frequency Shift Keying) Wetternachrichten, und zwar als Funkfernschreiben (RTTY). Später werden wir einen Empfänger für diesen Funkdienst bauen. Um den Empfänger zu testen und abzugleichen, ist ein Testsignal hilfreich. Mittels fraktionaler Teilung und PWM geht das einfach: Gesteuert vom Daten-Bitstrom tasten wir die Frequenz um.
Unseren Testsender programmieren wir vorerst für eine Trägerfrequenz von f = 125 kHz. Um aus dem Rechtecksignal einen Sinus zu machen, verwenden wir die Schaltung nach Bild 8. Die Interruptroutine für den fraktionalen Rechteckgenerator haben wir schon gezeigt. Die zusätzliche Routine *SendBit* **(Listing 4)** dient dazu, ein einzelnes Bit zu senden:

Zuerst werden *COUNT2* Durchläufe von Timer 0 abgewartet; d.h. der Bit-Takt ist die Timer0-Überlaufrate geteilt durch *COUNT2*. Danach wird, in Abhängigkeit des zu sendenden Bits, der Wert von *deltaDDS24* und *TOP1* gesetzt (Frequenzmodulation). Man beachte, dass das Setzen der Parameter jeweils von Aufrufen von cli() und sei() eingerahmt wird. Macht man dies nicht, kann es sein, dass die Interruptroutine aufgerufen wird, wenn nur einer der Parameter geändert wurde, was zu Fehlern führt. Im Programm *EXP-SQTX-FM-RTTY-V01.c* finden sich genau diese Routinen wieder. Mit Hilfe weiterer Hilfsroutinen kann man dann Zeichen im Fernschreibcode (Baudot [5]) senden, um den Sender des Wetterdienstes nachzubilden. In **Bild 9** ist das Spektrum dieses FM-RTTY Signales zu sehen. Man erkennt zwei eng beieinanderliegende Peaks, die von den Frequenzen 125 kHz+/-50 Hz herrühren. Es handelt sich um ein kontinuierliches Spektrum, das zu +/-1 kHz schnell abfällt.

Wenn man nun einen solchen Generator gebaut hat, möchte man natürlich testen, ob die Modulation auch in Ordnung ist. Dazu werden wir uns in der nächsten Folge den ersten Bausteinen eines digitalen Empfängers widmen.

(100180)

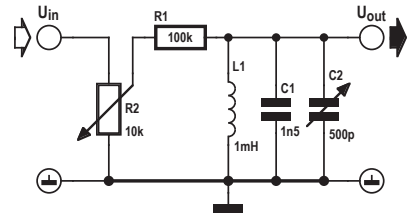


Bild 8. Abgleich eines Schwingkreises.

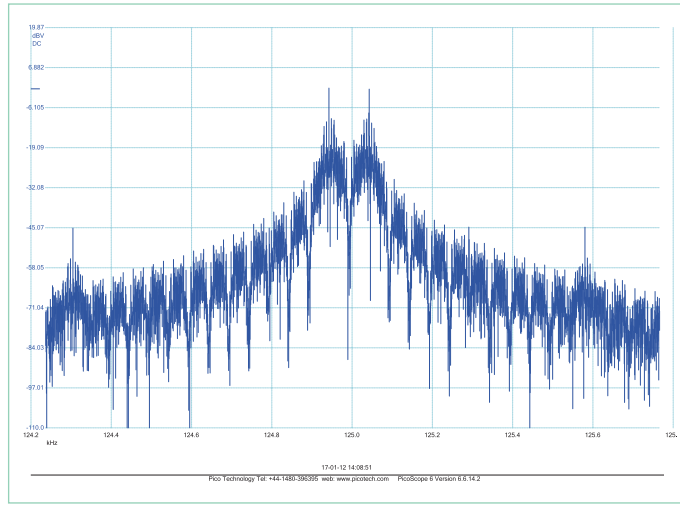


Bild 9. Spektrum bei einer Frequenzmodulation: 125 kHz +/- 50 Hz.

Listing 4

```
void SendBit(uint8_t theBit) {
  uint8_t k ;
  for (k=0 ; k<COUNT2 ; k++){
    while( ( TIFR & (1 << TOV0) )== 0 ) { }
    TIFR |= (1 << TOV0) ;
    }
  if ( theBit==MARK  ) {
    cli() ;
    deltaDDS24=MARK_deltaDDS24 ;
    TOP1=MARK_TOP1 ;
    sei() ;
    }
   else{
    cli() ;
    deltaDDS24=SPACE_deltaDDS24 ;
    TOP1=SPACE_TOP1 ;
    sei() ;
    }
  }
```

Weblinks

[1] www.elektor.de/110553
[2] www.elektor.de/080083
[3] www.elektor.de/100180
[4] http://wapedia.mobi/de/DDH47
[5] http://de.wikipedia.org/wiki/Baudot-Code

AndroPod (2)

Ein einfacher Weg zum maßgeschneiderten User-Interface

Von Jens Nickel

Im ersten Teil haben wir gesehen, wie einfach eigene Elektronik an das AndroPod-Interface und damit an ein Android-Smartphone anzuschließen ist. Daher wollen wir es allen Anwendern auch auf der Softwareseite möglichst einfach machen. Eine Benutzeroberfläche für das eigene Projekt lässt sich ganz einfach mit HTML aufbauen, wenn man unsere kostenlose Android-App verwendet. Wie immer bei Elektor ist diese Software fertig downzuloaden, aber bei Bedarf auch abzuändern, weil alles Open source ist.

Grundsätzlich benötigt man zur Steuerung der eigenen Elektronik eine Android-App, welche eine maßgeschneiderte Benutzeroberfläche enthält. Die naheliegendste Möglichkeit ist, dieses User-Interface mit dem mächtigen Android-Framework und der Programmiersprache Java zu implementieren. Für Anfänger hält das Programmieren unter Android zwar eine recht steile Lernkurve bereit. Der Einstieg könnte sich dennoch lohnen, da in Elektor immer mehr entsprechende Projekte vorgestellt werden. Bernhard Wörndl-Aichriedler [1], Student an der FH Hagenberg und einer der Entwickler des Andropod-Interfaces, hat ein kleines Android-Tutorial in Deutsch und Englisch zusammengestellt (Download unter [2]). Eine englischsprachige Anleitung findet man auch auf der Android-Referenzseite [3]. Unter anderem wird dort erklärt, wie man die Entwicklungsumgebung Eclipse [4] installiert und für die Android-Programmierung erweitert.

Nach dem Start eines neuen Android-Projekts müssen zuerst die Basisfunktionen (das Empfangen und Senden von Bytes über die serielle Schnittstelle) implementiert werden. Die AndroPod-Entwickler stellen hierfür die Java-Klasse `AndropodConnection` zum Download [5] zur Verfügung, die man in eigene Android-Projekte einbinden kann (siehe Tutorial). Dann geht es an die Erstellung der Benutzeroberfläche, die aus Kontroll-Elementen (Controls) wie Textboxen, Buttons usw. aufgebaut wird. Die Struktur einer vollständigen AndroPod-App wird im Textkasten beschrieben. Diese ist im Quellcode downloadbar, so dass man sie als Ausgangspunkt für eigene Entwicklungen einsetzen kann.

Gleiche Steuerung für Smartphone und PC

Es gibt aber noch eine zweite Möglichkeit, eigene Benutzeroberflächen zu erstellen, nämlich mit HTML und Javascript. Der Aufbau einer HTML-Seite mit den benötigten Kontroll-Elementen und das Programmieren einer Steuerung in der Sprache Javascript sind auch für Einsteiger einfach erlernbar. Darüber hinaus ist das Ergebnis prinzipiell unabhängig vom genutzten

Betriebssystem. Man erstellt die Benutzeroberfläche einmal und kann sie dann auf dem Android-Smartphone, einem PC oder einer anderen Computer-Plattform einsetzen. Die Voraussetzung ist natürlich ein entsprechendes Seriell-Interface (für den PC bietet Elektor zum Beispiel einen USB/RS485-Konverter an [6]).
Theoretisch lassen sich die erstellten HTML-Seiten direkt auf dem Smartphone anzeigen, nämlich im Webbrowser, der auf jedem Android-Gerät vorhanden ist. Den normalen Internet-Browser können wir aber nicht so weit bringen, dass er Bytes über unseren AndroPod empfängt und versendet, schon aus Sicherheitsgründen. Daher benötigen wir eine eigene Android-App, eine Art „Spezial-Browser“. Diese stellt die HTML-Nutzeroberflächen dar und realisiert die Kommunikation über die serielle Schnittstelle (**Bild 1**). Der Vorteil ist, dass wir diese App fertig anbieten können, sie muss vom AndroPod-Anwender nicht verändert werden. Die App „ElektorBusBrowserForAndropod“ bieten wir kostenlos im Google Android Market an, das macht den Download und die Installation auf dem Smartphone oder Tablet besonders einfach. Zusätzlich ist die App auch über unsere Website erhältlich, und zwar als .apk-File (was in etwa einem .exe in der PC-Welt entspricht) und im Quellcode [5]. Wenn Sie den Hardware-Test im ersten Artikel-Teil nachvollzogen haben, ist die App schon installiert [7], dennoch sollten Sie nun auf die neueste Version updaten.
Man muss jetzt nur noch die selbst erstellten HTML-Seiten auf das Telefon laden, was aber nicht schwer ist. Doch zuerst wollen wir uns einmal ansehen, wie eine solche HTML-Seite aussehen muss.

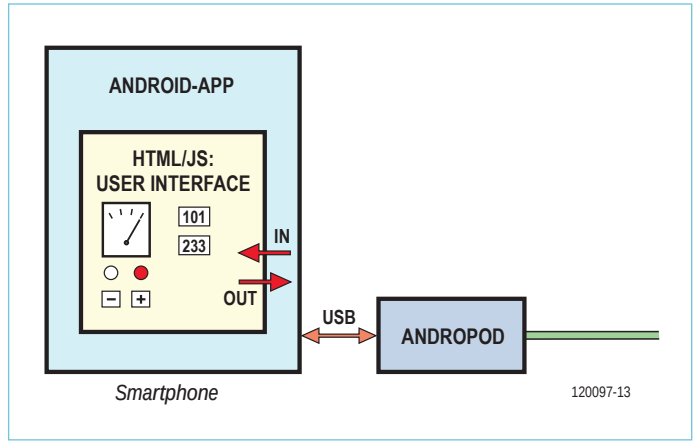


Bild 1. Unsere Android-App stellt HTML-Seiten dar, die vom Anwender für das eigene Projekt maßgeschneidert werden können.

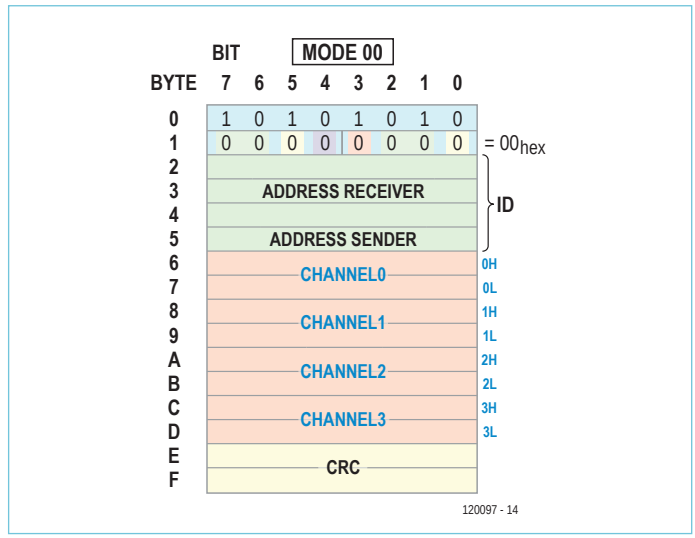


Bild 2. Aufbau einer *ElektorMessage*, die immer 16 Bytes lang ist.

Einfache Messages

Unsere Android-App empfängt und versendet immer 16 Byte lange Nachrichtenpakete (Messages), die nach dem ElektorMessageProtocol aufgebaut sind [6]. Die Datenrate beträgt 9600 Baud. Damit ist man von vorneherein mit vielen kommenden Elektor-Projekten kompatibel. Das Protokoll wurde für den ElektorBus [8] entwickelt, man ist aber keinesfalls auf die ElektorBus-Hardware und nicht einmal auf RS485 festgelegt. Die App funktioniert auch, wenn die Bytes über den Mini-DIN-Steckverbinder oder über die Stiftleisten eingelesen und versendet werden. Hierzu noch eine wichtige Anmerkung: In diesem Fall ist unbedingt der RS485-Treiber zu deaktivieren, indem man den Jumper bei JP4 ganz entfernt!

Eine ElektorMessage hat im einfachsten Fall folgenden Aufbau (siehe dazu auch **Bild 2**):

```
Byte 0        170 = AAhex
Byte 1        0
Byte 2        0
Byte 3        Empfänger-Adresse (1..127)
Byte 4        0
Byte 5        Sender-Adresse (1..127)
Byte 6..13:   Daten
Byte 14..15:  Optional, kann für Checksumme/CRC
              genutzt werden.
```

Das Android-Handy besitzt standardmäßig die Adresse 10 (was aber im Einstellungsmenü der App zu ändern ist). Im einfachsten Fall kommuniziert man nur mit einem einzigen anderen „Teilnehmer“, etwa der eigenen Messelektronik, die zum Beispiel mit einem AVR-Controller ausgerüstet ist. Wenn man diesem Mikrocontroller die Adresse „1“ zuweist, dann haben Nachrichten vom Smartphone zum Controller die Signatur „170,0,0,1,0,10,...“. In umgekehrter Richtung muss es dann „170,0,0,10,0,1,...“ heißen.
Bei den Datenbytes gibt es die Einschränkung, dass kein Wert „170“ vorkommen darf, weil dieser als Startsignal genutzt wird. Wer sich damit (oder etwa mit der festen Nachrichtenlänge) nicht anfreunden möchte, kann natürlich sein eigenes Protokoll entwerfen und die Android-App entsprechend abändern. Wenn man das Grundgerüst der App (siehe Kasten) nicht verändert, dann sind nicht unbedingt tiefgehende Android-Kenntnisse nötig, um etwa die Nachrichtenlänge anzupassen.

Empfangen...

Wenn die Android-App 16 Bytes empfangen hat, decodiert sie die Message zuerst in die Teile „Sender“, „Empfänger“ und

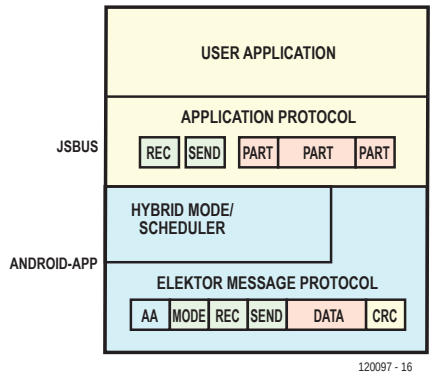


Bild 3. Für die Grob-Decodierung der Messages ist die Android-App zuständig, Javascript decodiert die Nutzdaten weiter.

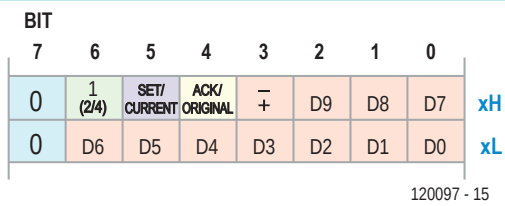


Bild 4. Mit zwei Datenbytes lassen sich Werte zwischen -1023 und 1023 darstellen, was für viele Zwecke schon ausreicht.

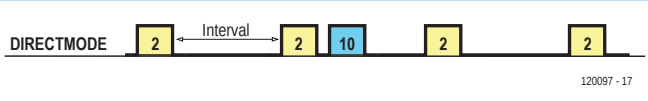


Bild 5. Der einfache *DirectMode* vermeidet Kollisionen bei einer 1:1-Kommunikation.

„8 Nutzbytes“. Diese Daten werden anschließend an die HTML-Seite des AndroPod-Users weitergereicht. Mit den Details muss sich der Anwendungsentwickler nicht herumschlagen, denn es gibt eine Javascript-Bibliothek namens „JSBus“, die man in die selbst erstellten HTML-Seiten einbinden kann. Diese nimmt das 8-Byte-Datenpaket nicht nur entgegen, sondern decodiert es auch weiter (**Bild 3**). Die Voraussetzung ist, dass die Daten nach dem *ElektorApplicationProtocol* aufgebaut sind. Dieses Protokoll ist für die Übertragung von Messwerten (in Ganzzahl- und Fließkommadarstellung), eine Umschaltung der Einheit und Skalierung (z.B. von V auf mV), das Setzen von Sollwerten, das Anzeigen von Grenzwertüber- oder Unterschreitungen und etlichem mehr aus dem Bereich „Messen, Steuern und Regeln“ prädestiniert. Kommende Elektor-Projekte (angedacht ist zum Beispiel ein Multimeter) werden das Protokoll ebenfalls verwenden, und natürlich lässt sich damit auch die ElektorBus-Hardware steuern. Im nächsten Heft wird zum Beispiel eine 230-V-Schaltplatine vorgestellt.
Innerhalb einer Message (also mit acht Nutzbytes) lassen sich zum Beispiel gleichzeitig bis zu vier Messwerte (-1023..1023) auf vier „Kanälen“ (Channels) übertragen, für jeden Wert werden zwei Bytes verwendet (**Bild 4**). Wenn die Javascript-Bibliothek ein 8-Byte-Datenpaket entgegengenommen hat, splittet sie es in bis zu vier Einheiten (*parts*) auf, also zum Beispiel vier Messwerte. Für jeden empfangenen Part ruft sie die Javascript-Routine `ProcessPart{…}` auf, die man in die eigenen HTML-Seiten integrieren muss. Das klingt komplizierter als es ist! Man schreibt einfach in seine HTML-Datei folgende Zeilen:

```
<SCRIPT src='JSBus.txt' Language='javascript'
></SCRIPT>
<SCRIPT Language='javascript' >

function ProcessPart(part)
{
       // eigener Code
}

</SCRIPT>
```

Mit eigenem Code kann man die Messwerte nun verarbeiten, also zum Beispiel eine HTML-Textbox füllen, die den Messwert anzeigt. Auch hierfür bietet die JSBus-Bibliothek Hilfestellung. Nehmen wir an, der erste Messwert (Channel=0) soll in einer Textbox mit dem Namen „MyTextbox“ dargestellt werden. Dann kann man einfach schreiben:

```
if (part.Channel == 0)
{TextboxSetvalue('MyTextbox', part.Numvalue);};
```

In der Variablen part.Numvalue steht immer der empfangene Messwert (von -1023 bis +1023). Eine Übersicht aller Funktionen der Library *JSBus* und viele weitere Hinweise findet man unter [9].
Von den Bus-Projektseiten [8] sind verschiedene Demo-Programme für AVR-Controller downzuloaden, auch auf der Controllerseite muss man das Rad also nicht neu erfinden, sondern kann bereits bestehenden Code für eigene Zwecke abändern. Weitere Controller-Software, die mit den beschriebenen Protokollen kompatibel ist, wird im Laufe des Jahres hinzukommen.

… und Senden

Um vom Handy aus eine Steuernachricht zu versenden, etwa zum An-/Ausschalten zweier LEDs (auf Channel1 und Channel2), kann man folgende Javascript-Zeilen nutzen:

```
function SwitchLed(LedStatus1,LedStatus2)
{
       var parts = InitParts();
parts = SetValue(parts, 10, 1, 1, 0,
LedStatus1);
parts = SetValue(parts, 10, 1, 2, 0,
LedStatus2);
       SendParts(parts, true);
}
```

wobei der LEDStatus1/2 gleich 0 für „aus“ und 1 für „an“ gesetzt werden muss. Der Code erzeugt zuerst ein leeres Array von Parts (Nachrichten-Einheiten). Dann werden zwei Parts hinzuaddiert, die jeweils den Sollwert zum An- beziehungsweise Ausschalten der LEDs auf den zwei Channels transportieren. Zum Schluss versendet die Funktion `SendParts(...)` beide Parts innerhalb einer einzigen Message.
Die Funktion `SwitchLed(...)` kann man durch einen Druck auf einen HTML-Button, als Reaktion auf eine eingehende Nachricht oder auch zeitgesteuert aufrufen lassen. Für alle diese drei Möglichkeiten finden sich in den Artikeln [9] und [10] Beispiele, entsprechende HTML-Files werden auf den zugehörigen Elektor-Webseiten zum Download angeboten. Einsteigern sei empfohlen, sich die meist sehr kurzen Codestücke anzuschauen und dann für eigene Zwecke anzupassen.

Kollisionsvermeidung

Wer die RS485-Erweiterung nutzt, muss sich Gedanken darum machen, wie Nachrichtenkollisionen vermieden werden. Denn die Kommunikation läuft nur über zwei Datenleitungen (halbduplex) ab. Wenn Sender und Empfänger gleichzeitig sprechen würden, gäbe es Bit-Salat. Die Android-App nimmt einem diese Aufgabe aber wiederum weitgehend ab. Grundsätzlich werden zwei Modi unterstützt: Der *DirectMode* und der *HybridMode*.
Ersterer eignet sich speziell für die oben beschriebene 1:1-Kommunikation (zum Beispiel zwischen Handy und Messelektronik). Es wird angenommen, dass die externe Elektronik, also zum Beispiel ein Messgerät, in festgelegten Intervallen Messwerte oder Statusmeldungen sendet. Falls umgekehrt Steuernachrichten vom Smartphone zur externen Elektronik gesendet werden müssen, werden diese unmittelbar nach dem Empfang eines Messwertes verschickt (**Bild 5**). Das ApplicationProtocol sieht übrigens eine Möglichkeit vor, einen Sensor anzuweisen, in einem bestimmten Intervall Messwerte zu verschicken [10]. Bei einer Datenrate von 9600 Baud sind Messintervalle von 100 ms problemlos möglich.
Den *DirectMode* schaltet man in der eigenen HTML-Seite an mit folgender Javascript-Zeile:

```
SetScheduler(SCHEDULER_
DIRECTMODE,0,0,0,0,0,0,0,0);
```

Man kann diesen Befehl auch direkt mit einem Druck auf einen HTML-Button verknüpfen, der beispielsweise die Beschriftung „DirectMode on“ trägt:

```
<BUTTON Type='button' onclick='javascript:
SetScheduler(SCHEDULER_
DIRECTMODE,0,0,0,0,0,0,0,0) ' >DirectMode on</
BUTTON>
```

Die Syntax von HTML und Javascript kann man recht schnell erlernen, einen kleinen Überblick finden Anfänger unter [9].

Im Gegensatz zum *DirectMode* eignet sich der *HybridMode* für eine Kommunikation von mehreren Teilnehmern an einem gemeinsamen Bus. Hier kommt ein Scheduler zum Einsatz, der nichts anderes macht, als den Busknoten Redezeit zuzuweisen. Die Android-App enthält einen solchen Scheduler. Beim Anschalten teilt man ihm die Adressen aller Knoten mit, die regelmäßig berücksichtigt werden sollen. Wenn das AndroPod-Interface beispielsweise an einen RS485-Bus mit den Teilnehmern 1 und 2 angeschlossen ist, wirft man den Scheduler mit folgendem Befehl an:

```
SetScheduler(SCHEDULER_ON,2,1,10,0,0,0,0,0);
```

Insgesamt können bis zu acht Knoten-Adressen als Parameter aufgeführt werden. Die „10“ sollte man nicht vergessen, falls auch das Android-Smartphone des Öfteren etwas mitzuteilen hat.

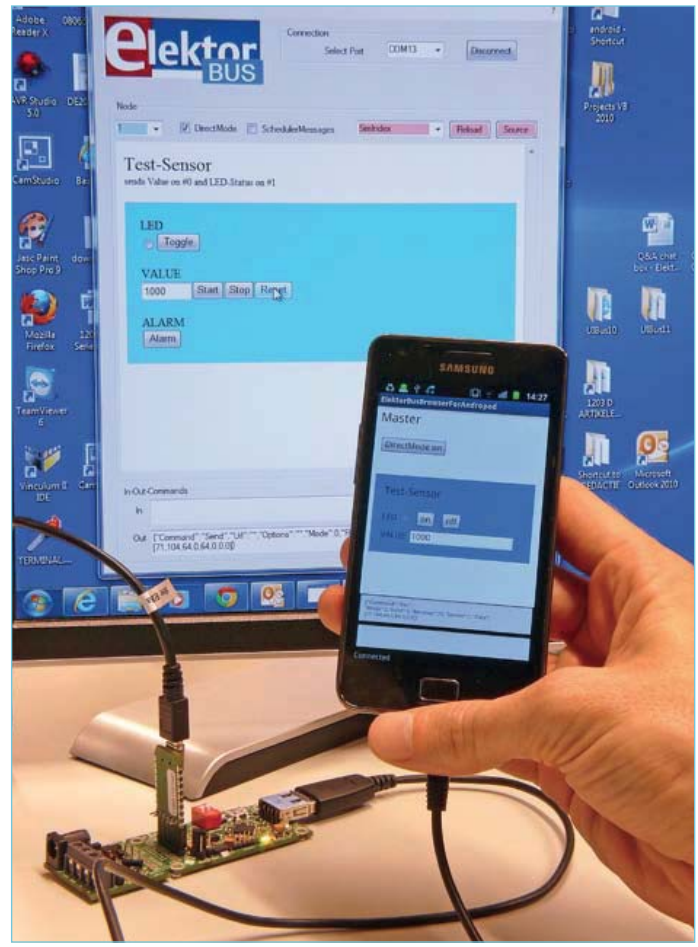


Bild 6. Für einen ersten Test schließt man einen PC an das AndroPod an, zum Beispiel über den USB/TTL-Konverter BOB FT232.

Demo

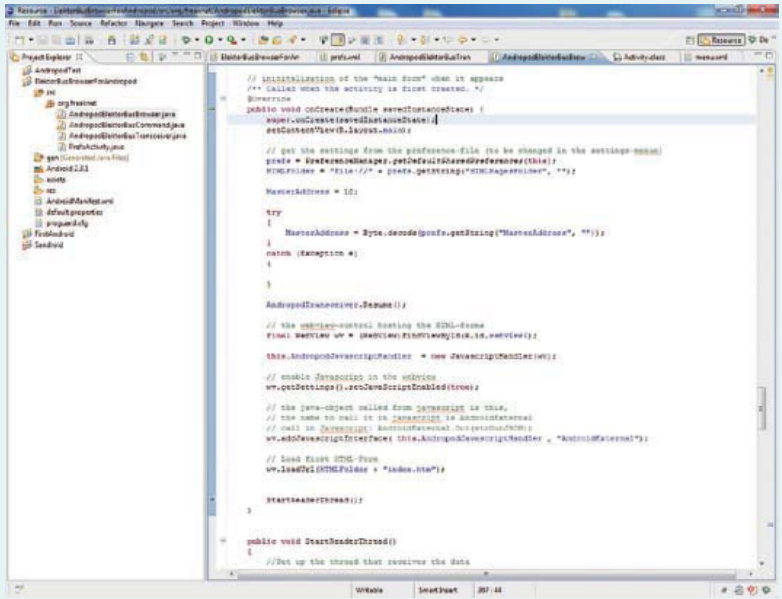
Das Ganze ist jetzt noch recht theoretisch, eine kleine Demo-Anwendung muss her. Zum Test eignen sich natürlich Mikrocontrollerboards wie unser Experimental-Knoten aus dem ElektorBus-Projekt [6]. Es geht aber auch mit einem PC, den man (wie im letzten Teil beschrieben) auf verschiedenen Wegen an das AndroPod anschließen kann, zum Beispiel über den USB/TTL-Konverter BOB-FT232 [5] (**Bild 6**). Im downloadbaren Zip-File findet man die Software „ElektorBusElectronicsSimulator.exe“ [5]. Hier handelt es sich um PC-Software, die Nachrichten versenden und empfangen kann. Man kann damit schön

einen passenden Gesprächspartner für unser Handy simulieren, etwa Mess-Elektronik. Zur Darstellung dieser simulierten Hardware wird ebenfalls eine HTML-Seite genutzt. Damit die PC-Software diese Seite findet, muss der Ordner „UIBus“ im Download-Ordner [5] auf den Desktop des Computers gezogen werden.
Nun müssen wir noch die HTML-Seite, die auf dem Smartphone dargestellt werden soll, auf das Telefon bringen; und auch die Javascript-Library JSBus.txt muss auf dem Telefon landen. Standardmäßig sucht die App in einem Ordner „ElektorBusBrowser“ auf der SD-Karte nach den beiden Dateien (dies kann im Einstellungsmenü angepasst werden). Doch wie kommen die beiden Files dorthin?
Die erste Möglichkeit ist, die Dateien auf einem der traditionellen Wege vom PC auf das Smartphone zu übertragen, zum Beispiel mit einem passenden USB-Kabel oder mit Bluetooth. Wer ein Kabel benutzt, muss das USB-Debugging vorher aus- und nachher wieder einschalten (siehe dazu den ersten Teil [7]). Auch den Ordner für die Files muss man selbst anlegen.
Es gibt aber noch einen bequemeren Weg, der ganz ohne Umstöpseln der Kabel auskommt. Wir benutzen wie im ersten Teil die Software *AdifController* der beiden AndroPod-Entwickler, diesmal ist die Registerkarte „Files“ auszuwählen. Links müsste das Telefon unter „Detected Phones“ zu sehen sein (eventuell muss man das AndroPod-Interface einmal kurz von der Stromversorgung trennen). Man klickt nun auf „Browse“ und wählt den Ordner „UIBus“ auf dem Desktop aus. Dann drückt man den Knopf „Upload“.
Mit dem Dateimanager sollte man nachsehen, ob die Files korrekt auf der Speicherkarte des Telefons abgelegt wurden.

Test

Jetzt starten wir sowohl die App auf dem Telefon als auch das Programm auf dem PC. Die simulierte „Mess-Elektronik“ erscheint im großen Fenster der PC-Software (**Bild 7**). Oben müssen wir noch den

Aufbau einer AndroPod-App



Den Aufbau einer App studieren wir am besten an einem Beispiel wie dem *ElektorBusBrowserForAndropod*, den man von [5] als Android-Projekt (für Eclipse) downloaden kann. Den Quellcode findet man im Projekt-Unterordner „src“. Wie schon des Öfteren erwähnt, fußt das Android-Framework auf der Programmiersprache Java, wobei sowohl die Syntax als auch wichtige Bibliotheken genutzt werden. Zumindest Java-Vorkenntnisse sind also sehr hilfreich. Sinnvoll ist es auch, sich einmal in einem guten Java-Buch die Kapitel zur Thread- und zur Netzwerk-Programmierung anzusehen. Dann sollte man in der Lage sein, den groben Aufbau der App zu verstehen und diese auch für eigene Zwecke abändern können. Wer ein „natives“ (also direkt mit Android verwirklichtes) User-Interface programmieren will, muss zusätzlich Android-Kenntnisse haben oder erwerben. Erste Anlaufstelle ist dabei natürlich die Android-Referenz [11]. Als gutes Buch kann ich [12] empfehlen.

Unsere Andropod-App verwendet drei Klassen. Der `AndropodElektorBusTransceiver` initialisiert eine TCP-Verbindung auf Port 1337, schließt diese und baut sie bei Bedarf wieder auf. Die Methoden `TransmitPacket` und `ReceivePacket` versenden beziehungsweise empfangen ein 16-Byte-Paket, wobei der Wert „$170=AA_{hex}$“ als Markierung für den Beginn einer neuen Nachricht genutzt wird. Wer eine andere Nachrichtenlänge verwenden will, muss den Code also hier anpassen.

Objekte der Klasse `AndropodElektorBusCommand` dienen als Container für die Daten einer empfangenen oder zu versendenden Nachricht. Sie enthält ein Byte-Array namens `Rawdata`, das alle 16 Bytes einer Nachricht aufnimmt. Die Methode `parseRawData` decodiert das Nachrichtenpaket. Dabei werden die Variablen `Mode, Receiver, Sender` und `Data` gefüllt, wobei `Data` wiederum ein Array ist, das die Nutzdaten der Message (8 Bytes) enthält. Umgekehrt wird das 16-Byte-Array `Rawdata` gefüllt, wenn man dem Konstruktor von `AndropodElektorBusCommand` die Werte `Mode, Receiver, Sender` und `Data` übergibt. Darüber hinaus sorgt die Klasse auch für die Verpackung dieser Werte in einen speziellen String, um eine decodierte Nachricht an die gerade dargestellte HTML/Javascript-Seite übergeben zu können (mehr über den Aufbau dieser *In-* und *OutCommands* in [9]). Alle

passenden (virtuellen) COM-Port einstellen und den Knopf „Connect“ betätigen.
Der PC simuliert einen Sensor, der periodisch Werte übermittelt, man könnte zum Beispiel an einen Stromzähler im Keller denken. In der HTML-Oberfläche starten wir das Versenden der Werte mit dem Knopf „Start“. Mit dem HTML-Button „Toggle“ können wir außerdem das Ein- und Ausschalten einer Kontroll-LED simulieren.
Die Werte müssten nun auf dem Smartphone zu sehen sein, das unsere „Master“-Oberfläche darstellt (**Bild 8**). Man achte auf die etwas verzögerte Anzeige der Kontroll-LED. Dies rührt daher, dass ein geänderter LED-Status vom PC nicht unmittelbar verschickt wird, sondern erst, wenn ein neuer Messwert rausgeht. LED-Status und Messwert werden auf verschiedenen Channels innerhalb einer Message versandt.
Jetzt wollen wir einmal etwas vom Master zum Sensor zurückschicken. In der Master-Oberfläche benutzen wir die Knöpfe „On“ und „Off“, um die Kontroll-LED auf dem Sensor setzen oder zurücksetzen zu können. Bevor die Übermittlung von Nachrichten in die andere Richtung funktioniert, müssen wir aber noch den DirectMode anschalten. Da das Handy eine kleinere Auflösung als ein PC-Monitor hat, ist der entsprechende Knopf innerhalb der HTML-Oberfläche untergebracht. Man achte einmal darauf, wie die LED-Anzeige im Smartphone reagiert, wenn die Buttons „On“

Bild 7. Die PC-Software simuliert die externe (Mess-)Elektronik.

Entwickler, die eine andere Aufteilung der Bytes innerhalb einer Nachricht wünschen, können diese Klasse entsprechend anpassen.

Die Klasse `AndropodElektorBusBrowser` sorgt für die Darstellung der empfangenen Daten, außerdem werden Nutzereingaben entgegengenommen, die den Versand von Nachrichten auslösen. Um die im Fließtext beschriebene Plattformunabhängigkeit zu verwirklichen, geschieht beides innerhalb einer HTML-Seite. `AndropodElektorBusBrowser` sorgt nur dafür, dass diese HTML-Seite in einem Android-Control vom Typ `WebView` dargestellt wird. Empfangene Nachrichten (genauer gesagt der daraus generierte String) werden über die Methode `In(...)` an HTML/Javascript weitergereicht. Umgekehrt übergibt das in die HTML-Seite eingebettete Javascript einen String an die Methode `Out(...)`, wenn eine Nachricht versendet werden soll. Darüber hinaus ist in dieser Klasse auch der Scheduler angesiedelt. Die Methode `Out(...)` versendet die Nachricht daher auch nicht direkt, sondern speichert sie nur zwischen. Verschickt wird die Message erst, wenn der Scheduler dies erlaubt (mit dem *DirectMode* als Ausnahme). Ein Teil der Klasse ist auf die ElektorBus-Protokolle zugeschnitten; für alle Leser, die eine eigene App schreiben wollen, wird der Code ab der Methode `onCreate` interessant. Bei der Klasse handelt es sich nämlich um eine Android-Activity, und zwar jene, die beim Start der App aufgerufen wird (sozusagen das Haupt-Fenster der Anwendung). Android-Kenner wissen, dass dabei neben `onCreate` (Initialisierung) auch die Methoden `onStop, onPause` und `onResume` überschrieben werden sollten; hier steht Code, der beim Beenden der Anwendung, beim Wechsel zu einer anderer Activity und bei der Rückkehr abzuarbeiten ist. Für das AndroPod ist noch die Methode `StartReaderThread` wichtig, sie wird im Folgenden beschrieben.

Beim Start der Anwendung wird zuerst `onCreate` von `AndropodElektorBusBrowser` (oder von der selbst programmierten Activity) aufgerufen. Die Codezeile `AndropodTransceiver.Resume` aktiviert den „Transceiver“ in der Klasse `AndropodElektorBusTransceiver`. Zuerst wird dort in der Methode `Resume(...)` ein `ServerSocket` auf Port 1337 geöffnet. Dann wird ein Extra-Thread gestartet. Abgearbeitet wird dabei der Code, der in der Methode `run(...)` in `AndropodElektorBusTransceiver` steht. Wie leicht zu erkennen ist, ist der Code in eine Endlosschleife eingefasst. In dieser Schleife wird kontinuierlich überprüft, ob schon eine Verbindung zum AndroPod-Interface besteht, ansonsten wird diese hergestellt. Dieser Thread macht es möglich, dass das Interface bei laufender App angesteckt werden kann.

Am Ende der Initialisierungsroutine `onCreate` wird mit der Codezeile `StartReaderThread();` der eigentliche Lese-Thread gestartet. Der dazugehörige Code findet sich ein paar Zeilen darunter, wieder innerhalb einer run()-Methode. Die while-Schleife dort wird kontinuierlich durchlaufen, solange die Verbindung zum Interface besteht. Mit der Zeile `ReceivedMessage = AndropodTransceiver.ReceivePacket();` werden die nächsten 16 Bytes (ab Startbyte AA_{hex}) eingelesen. Danach kann das Nachrichtenpaket weiterverarbeitet werden. Der Vollständigkeit halber sei hier erwähnt, dass auch der Scheduler in einem eigenen Thread läuft.

Um eine Nachricht zu versenden, ist der Aufwand nicht ganz so groß: Man ruft einfach die Methode `AndropodTransceiver.TransmitPacket(...)` auf (mit einem `AndropodElektorBusCommand` als Parameter).

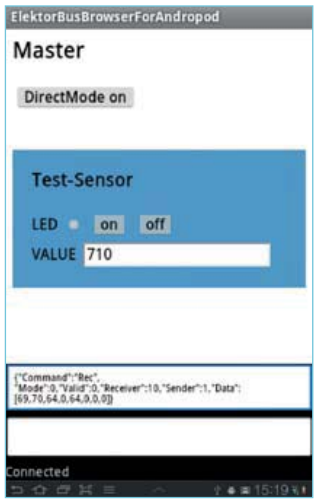


Bild 8. Screenshot der Steuerung auf dem Smartphone.

und „Off“ betätigt werden. Wer gut aufgepasst hat, weiß warum: Der geänderte LED-Status wird vom Smartphone erst verschickt, wenn eine neue Nachricht (samt Messwert und dem alten Status der Kontroll-LED) vom PC eingetrudelt ist. Denn dann ist der Bus garantiert frei.
Mit einem zweiten Handy kann man auch gleich die SMS-Funktion ausprobieren. Wir müssen der App vorher noch die Nummer dieses Handys mitteilen, das im „Alarm“-Fall angesimst werden soll. Dies wird im Settings-Menü der App erledigt, das wir schon aus dem ersten Teil [7] kennen. Nach einem Druck auf die Menü-Taste des Handys unten links und einem anschließenden Klick auf „Settings“ lassen sich drei Parameter einstellen:

- Der Ordner, in dem die App die HTML- und Javascript-Files sucht.
- Eine voreingestellte SMS-Nummer.
- Die ElektorMessage-Adresse des Android-Smartphones (auf „10“ voreingestellt).

In ganz seltenen Fällen kann die App nach dem Verändern der Einstellungen abstürzen; dann muss man die App beenden und neu starten.
Nach der Eingabe der Nummer kann man die SMS-Funktion testen, indem man innerhalb der PC-Software auf den Button „Alarm“ klickt. Auf dem noch freien Channel2 wird dann innerhalb der nächsten Nachricht eine „1“ verschickt, was einen Alarm des Sensors anzeigt.
Die HTML-Seite, die auf dem Handy dargestellt wird, veranlasst daraufhin das Versenden einer SMS; und zwar mit folgender Javascript-Zeile:

```
SendSMS("1", "Alarm on Test-Sensor!");
```

Die „1“ als erster Parameter bedeutet, dass die SMS an die in der App voreingestellte Nummer versendet wird, hier könnte man aber auch direkt eine beliebige Handy-Nummer angeben.

Man kann sich den Quellcode der zwei benutzten HTML-Seiten einmal mit der PC-Software anzeigen lassen. Die Seite, welche die Messelektronik simuliert, heißt „SimIndex“; die auf dem Smartphone dargestellte Seite heißt „Index“. Mit der rötlich unterlegten Combobox wählt man die Seite aus, dann klickt man auf den Knopf „Source“.
In einer richtigen Anwendung müssten wir noch einen Bestätigungsmechanismus einbauen, so dass wichtige Nachrichten nicht durch eine Störung verloren gehen können. Das ElektorMessageProtocol und das ApplicationProtocol stellen hierfür verschiedene Möglichkeiten zur Verfügung [8].

Ausblick

Das Konzept mag auf den ersten Blick ein wenig komplex erscheinen, doch sind die Vorteile dieses HTML-basierten Ansatzes immens. Eine HTML/Javascript-Seite kann viel einfacher und hemdsärmeliger programmiert werden wie ein User-Interface mit Android-Java, hinzu kommt die Plattformunabhängigkeit (gleiche Oberfläche lässt sich auch auf dem PC verwenden). Darüber hinaus befindet sich HTML gegenwärtig im Aufwind; es ist davon auszugehen, dass immer mehr PC-Software für Elektor-Projekte in einem Browser laufen wird.
Im nächsten Heft wird eine „Installationsplatine“ mit zwei Relais und zwei Eingängen vorgestellt, wir werden hierzu ElektorBus-kompatible Firmware anbieten. Dann kann man Leuchten und andere Verbraucher per Android-Smartphone schalten; und wer mag, versendet Statusmeldungen per SMS. Weitere Projekte sind in Vorbereitung, zum Beispiel aus dem Bereich Messtechnik.

(120097)

[1] www.xdevelop.at
[2] www.xdevelop.at/#category=projects&subcategory=1&anchor=6
[3] http://developer.android.com/sdk/installing.html
[4] www.eclipse.org/downloads/
[5] www.elektor.de/120097
[6] www.elektor.de/110258
[7] www.elektor.de/110405
[8] www.elektor.com/elektorbus
[9] www.elektor.de/110517
[10] www.elektor.de/110708
[11] http://developer.android.com/guide/index.html
[12] „Learning Android”, Marko Gargenta, O’Reilly: http://shop.oreilly.com/product/0636920010883.do

FTDI CELEBRATES 20TH ANNIVERSARY

Visit FTDI at Embedded World and see its largest **NEW** product release **EVER**

USB MADE EASY

STAND 322 IN HALL 4A

embeddedworld2012
Exhibition&Conference
... it´s a smarter world

FTDI Chip

Sensible PC-Lüfter-Regelung

Für sechs PWM-gesteuerte Lüfter

In modernen PCs sind Lüfter verbaut, sie sorgen dafür, dass sich empfindliche Bauteile nicht überhitzen. Weil die PC-Hauptplatine die Lüfter nicht individuell steuern kann, wurde dieses kleine Board entworfen. Bis zu sechs Lüfter können aktiv geregelt werden, mehrere Temperatursensoren lassen sich im PC-Gehäuse verteilen. Die Regelung wird mit einem PC-Programm konfiguriert und überwacht, das seine Informationen über USB erhält.

Von Ivo Pullens (NL)

Vor mehreren Jahren veröffentlichte Intel einen Standard, der sich auf vierpolig anzuschließende, pulsbreiten-gesteuerte PC-Lüfter bezieht [1]. Diese Lüfter sind kaum teurer als die bekannten zwei- oder dreipoligen Ausführungen, sie haben jedoch die willkommene Eigenschaft, dass die Drehzahl über die Pulsbreite eines Rechtecksignals (PWM) gesteuert werden kann. Außerdem geben diese Lüfter ein Tachosignal zurück, das Auskunft über die tatsächliche Lüfterdrehzahl gibt.

Auf PC-Hauptplatinen befindet sich meistens nur ein einziger Lüfteranschluss, der ein PWM-Signal liefert. In aller Regel verrichten jedoch im PC-Gehäuse mehrere Lüfter ihren Dienst. Hier müssen der zweite und alle weiteren Lüfter zwangsläufig auf feste Drehzahlen eingestellt werden. Auf anderen Hauptplatinen können zwar mehrere PWM-gesteuerte Lüfter parallel geschal-

tet werden, doch die auf der Hauptplatine untergebrachte Steuerung ist meistens nicht beeinflussbar.

Das hier beschriebene Lüftersystem regelt individuell die Drehzahlen von sechs Lüftern, abhängig von den im PC-Gehäuse gemessenen Temperaturen. Alle Lüfter laufen nur mit den aktuell notwendigen Mindestdrehzahlen, so dass die Geräuschentwicklung auf ein Minimum reduziert wird.

Ein ATmega

Aus der Schaltung in **Bild 1** geht hervor, dass ein Mikrocontroller ATmega168 alle Aufgaben der Steuerung und Regelung übernimmt. Außer dem Mikrocontroller und wenigen passiven Bauelementen sind diverse Steckverbinder für den Anschluss der Lüfter und Sensoren, für die Betriebsspannung sowie eine USB-Buchse vorhanden. Die Leitungen der vierpolig angeschlossenen Lüfter liegen unmittelbar am Mikrocontroller, denn ihre elektrischen Daten sind zum Mikrocontroller kompatibel. An den Mikrocontroller-Eingängen ziehen interne Pullup-Widerstände die offenen Kollektorausgänge der Tachogeneratoren, die in den Lüftermotoren eingebaut sind, auf die Betriebsspannung.
RC-Glied R3/C2 glättet das von der Hauptplatine kommende PWM-Signal, so dass dieses Signal an einen analogen Mikrocontroller-Eingang gelegt werden kann.

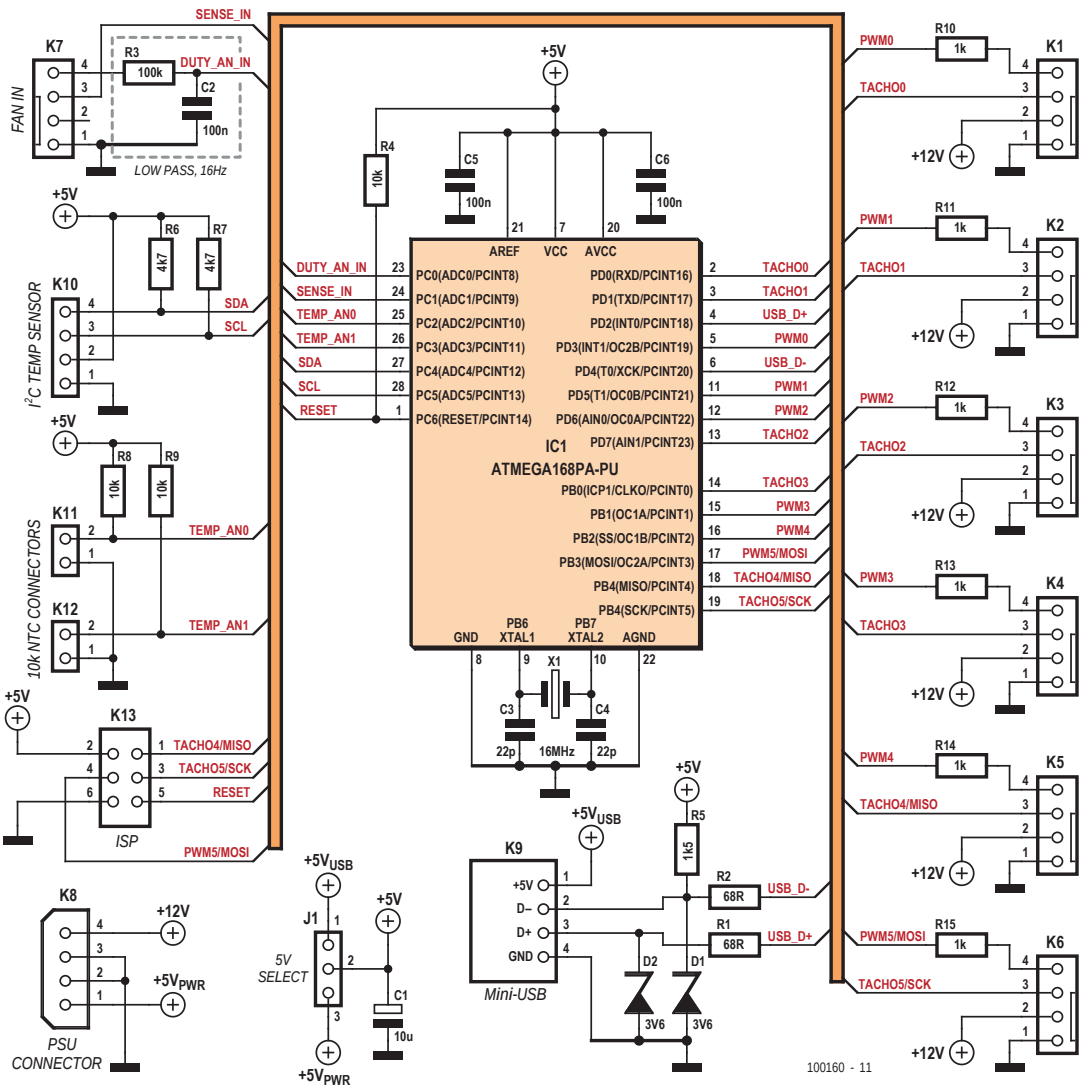


Bild 1. Mittelpunkt der Hardware ist ein Mikrocontroller ATmega168.

Stückliste

Widerstände:
R1,R2 = 68 Ω
R3 = 100 k
R4,R8,R9 = 10 k
R5 = 1k5
R6,R7 = 4k7
R10..R15 = 1 k

Kondensatoren:
C1 = 10 µ/16 V stehend, Raster 2,5 mm
C2,C5,C6 = 100 n, Raster 5 mm
C3,C4 = 22 p

Halbleiter:
D1,D2 = Zenerdiode 3,6 V/400 mW
IC1 = ATmega168PA-PU (programmiert erhältlich, 100160-41)

Außerdem:
X1 = Quarz 16 MHz
J1 = Stiftleiste 3-polig mit Jumper
K1..K6,K7 = Lüfter-Steckverbinder 4-polig
K8 = PC-Stromversorgungsanschluss 4-polig (männlich), für Platinenmontage
K9 = Mini-USB-B-Buchse, für Platinenmontage
K10 = Stiftleiste 4-polig
K11,K12 = Stiftleiste 2-polig
K13 = Stiftleiste 2 x 3-polig
Platine 100160-1 (siehe [3])

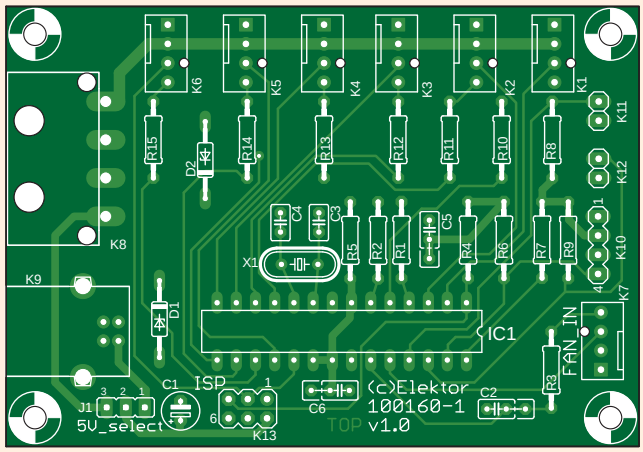
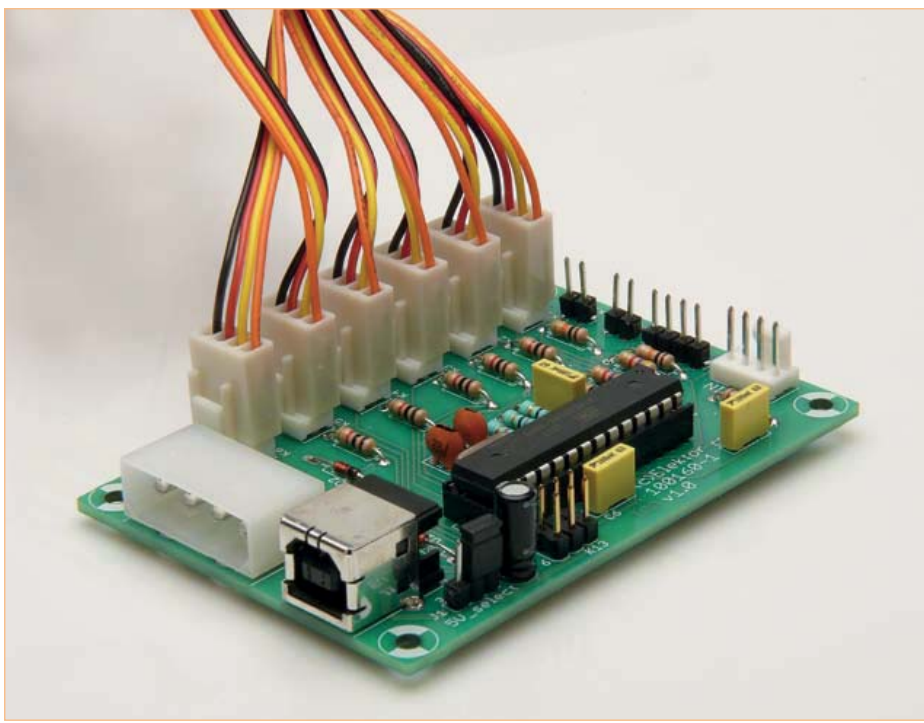


Bild 2. Auf dieser kleinen Platine ist die Lüfterregelung schnell aufgebaut.



Zwei NTC-Widerstände, angeschlossen an K11 und K12, sind die analogen Temperatursensoren. Zusammen mit R8 und R9 bilden die NTCs zwei Spannungsteiler. Die an den Spannungsteilern abgegriffenen Spannungen werden den Mikrocontroller-Eingängen ADC2 und ADC3 zugeführt.
An K10 können bei Bedarf bis zu acht I²C-Temperatursensoren angeschlossen werden, hier sind R6 und R7 die für den I²C-Bus obligatorischen Pullup-Widerstände. Die Firmware wurde mit Sensoren des Typs MCP980x von Microchip getestet. Vergleichbare Sensoren wie der Typ TCN75 des gleichen Herstellers sind (gegebenenfalls mit kleinen Anpassungen) ebenfalls verwendbar. Nach dem Start erkennt die Firmware, wie viele Sensoren mit dem I²C-Bus verbunden sind.
Mit Jumper J1 lässt sich die Mikrocontroller-Betriebsspannung zwischen USB und PC umschalten. Während der Firmware-Entwicklung ohne angeschlossene Lüfter kann der Betrieb am USB nützlich sein. Normalerweise wird die Schaltung aus dem PC mit Strom versorgt (Jumper von Pin 2 nach Pin 3).

Aufbau

Da die Schaltung ausschließlich aus konventionellen Bauelementen besteht, ist der Aufbau auf der Platine (**Bild 2**) nicht schwierig. Bei Steckverbinder K8 handelt es sich um das männliche Gegenstück zum verbreiteten großen PC-Festplatten-Stromversorgungsstecker, ein Hersteller ist Molex. Notfalls können die Kabeladern auch unmittelbar an die Kupferflächen der Platine gelötet werden.
Die Lüfter-Steckverbinder in der 4-Pin-Ausführung K1...K7 sind (anders als die 3-Pin-Version) in Einzelstückzahlen nicht überall erhältlich. Da die Kontaktabstände und die übrigen Daten beider Versionen identisch sind, können 3-Pin-Lüfter auch an 4-Pin-Steckverbinder angeschlossen werden. Wenn ein 3-Pin-Steckverbinder und ein Einzelstift anstelle des vierten Pins montiert werden (siehe **Bild 3**), entsteht ein 4-Pin-Steckverbinder mit Verpolungsschutz für PWM-Lüfter.

Firmware

Die Mikrocontroller-Firmware wurde in C geschrieben, für die Kommunikation mit

Wichtige Eigenschaften

- Individuelle Regelung für bis zu sechs pulsbreiten-gesteuerte, vierpolige Lüfter
- Unabhängige Drehzahlmessung und Erkennung stehender oder blockierter Lüfter
- Eingänge für maximal acht Temperatursensoren:
  - zwei NTC-Widerstände
  - sechs I2C-Temperatursensoren (MCP980x, TCN75 oder kompatibel)
  - Lüfterdrehzahlen sind auch als Messwerte erfassbar
  - Virtuelle „Sensoren", Werte können über USB eingestellt werden
  - PWM-Signal einer externen Lüftersteuerung, beispielsweise von der Hauptplatine
- Drehzahlsteuerung wahlweise nach drei Methoden:
  - Konstante Drehzahl
  - Lineare Steuerung, Drehzahl hängt linear von einer Sensor-Messgröße ab
  - PI-Regelung, Drehzahl wird abhängig von einer Sensor-Messgröße und einem Sollwert gesteuert
- Konfiguration und Überwachung mit PC-Programm über USB-Verbindung
- Software-Bibliothek für die Kommunikation mit externen Anwendungen, für Windows und Linux

dem PC macht sie vom Software-USB-Stack *V-Usb* Gebrauch (siehe [2]). Im ATmega88 konnten nicht alle Funktionalitäten untergebracht werden, so dass die Wahl auf den größeren ATmega168 mit der doppelten Flash-Speicher-Kapazität fiel. Das Quellprogramm ist mit Einschränkungen auch für den ATmega88 compilierbar (siehe Datei *config.h*, enthalten im Download auf der Projektseite [3]).

Der Standard für PWM-gesteuerte PC-Lüfter von Intel sieht vor, dass die Frequenz des PWM-Signals bei 25 kHz liegt. Die Motordrehzahl muss dem Duty-Cycle linear folgen, sobald der Duty-Cycle mehr als 20 % beträgt. Nach dem Intel-Standard ist die Drehzahl bei niedrigeren Duty-Cycles nicht definiert. Der ATmega168 stellt zwar sechs PWM-Ausgänge bereit, doch nur die zwei vom 16-bit-Timer 1 gesteuerten PWM-Ausgänge können Signale um 25 kHz liefern. Da die übrigen PWM-Ausgänge von 8-bit-Timern gesteuert werden, betragen die Frequenzen ihrer Signale entweder 7,8 kHz oder 62,5 kHz. Auch diese PWM-Ausgänge liefern Signale mit der Frequenz 25 kHz, wenn ein Kunstgriff angewendet wird: Die PWM-Signale werden zwar auf die niedrige Frequenz 7,8 kHz eingestellt, doch der Overflow-Interrupt von Timer 1 startet zyklisch auch die 8-bit-Timer. Dadurch liefern alle sechs PWM-Ausgänge Signale mit Frequenzen um 25 kHz.
Die Frequenzen der impulsförmigen Signale, die von den Tachogeneratoren kommen, sind zu den Drehzahlen proportional. Die Impulse werden mithilfe des Pin-Change-Interrupts gezählt. Im normalen Betrieb drehen die Lüfter nicht schneller als 6000 U/min; die Tachogeneratoren erzeugen bei jeder Umdrehung zwei Impulse. Wenn sämtliche Lüfter mit 6000 U/min laufen, muss die Firmware 1200 Pin-Change-Interrupts in der Sekunde verarbeiten. Die Anzahl mag hoch erscheinen, doch die Programm-Hauptschleife kann der schnellen Impulsfolge verzögerungslos folgen. Außer der Berechnung des Duty-Cycle hat sie nur wenige weitere Aufgaben.

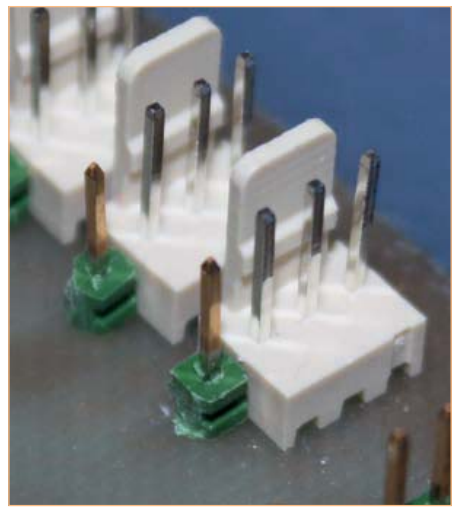


Bild 3. Der vierte Pin macht den Anschluss PWM-gesteuerter Lüfter möglich.

Die Hauptschleife prüft im Sekundenabstand, ob die Lüfter still stehen, weil sie vielleicht blockiert oder defekt sind. Abhängig von der Systemkonfiguration kann ein nicht drehender Lüfter einen vom Mikrocontroller simulierten Tachogenerator stoppen. Der simulierte Tachogenerator gibt auf diese Weise an die Hauptplatine die Information weiter, dass ein Lüfter seinen Dienst versagt. Die Sicherheitsvorkehrungen auf der Hauptplatine bleiben so in Funktion, sie können beispielsweise einen Alarm auslösen.
Die übrigen Aufgaben der Hauptschleife sind das Abfragen der aktuellen Sensor-Werte, das Berechnen der einzelnen Duty-Cycle und das Schreiben der Werte in die Output-Compare-Register der Timer. Die eine volle Sekunde langen Intervalle, in denen die Duty-Cycle berechnet werden, sind für die Regelung vergleichsweise langsamer thermischer Vorgänge kurz genug.
Der simulierte Tachogenerator, der das Signal für die Hauptplatine liefert, muss bis etwa 1200 Impulse in der Sekunde erzeugen, auf ein genaues Timing kommt es hier nicht an. Die Hauptschleife erzeugt diese Impulse, indem sie die Ausgangsleitung abwechselnd hochohmig und nach Masse schaltet.

Die eingestellte Konfiguration kann über die PC-Software im EEPROM des Mikrocontrollers abgelegt werden, so dass sie bei jedem folgenden Neustart wirksam ist.
Ein eigener Teil der Firmware sind der USB-Stack und die Routinen für den Datenaustausch mit dem PC. Die Quasi-Realtime-Kommunikation arbeitet mit dem INT0-Handler, der die oberste Priorität besitzt.

PC-Programm

Die PC-Software wurde in C++ mit dem kostenlosen *Visual C++ 2010 Express* von Microsoft entwickelt. Die grafische Oberfläche verwendet *wxWidgets*, so dass das Programm sowohl unter Windows als auch unter Linux lauffähig ist. Unter OS X wurde es vom Autor nicht getestet. Für die Kommunikation mit dem Mikrocontroller über USB wird *LibUsb* eingesetzt. Eine separate Bibliothek ist für spezielle Funktionen der Kommunikation und des Datenaustauschs zuständig. Die Lüfterdrehzahlen können als Umdrehungen in der Minute eingestellt werden, und die Werte der Sensoren werden unmittelbar angezeigt.

Die grafische Benutzeroberfläche des PC-Programms besteht aus folgenden Tabs:

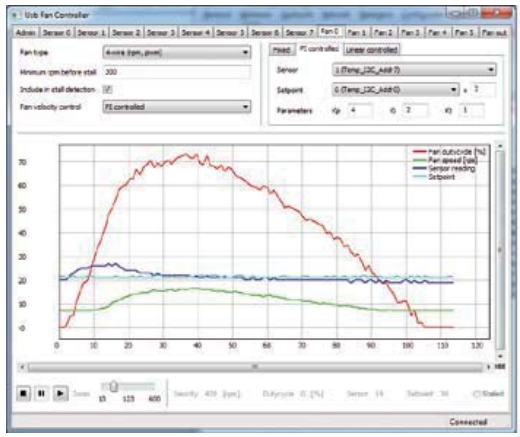


Bild 4. Mit dem PC-Programm wird die Lüfterregelung konfiguriert und überwacht.

- Admin: Die Konfiguration kann im EEPROM gespeichert, von dort abrufen und modifiziert werden. Wenn die aktuelle Konfiguration nicht im EEPROM gespeichert wird, geht sie beim Abschalten verloren!

- Sensor 0…7: Die angeschlossenen Sensoren können wahlweise einem Sensorkanal zugewiesen werden, die Messwerte werden grafisch abhängig von der Zeit dargestellt.

- Fan 0…5: Hier lassen sich die angeschlossenen Lüfter konfigurieren.

- Fan Out: Einstellen des simulierten Tachogenerators, der das Ersatzsignal für die Hauptplatine erzeugt.

Bevor die Sensorsignale für die Regelung wirksam werden können, müssen sie den Sensorkanälen zugewiesen werden. Das in **Bild 4** wiedergegebene Programmfenster zeigt als Beispiel eine Konfiguration für Lüfter 0 (Fan 0), angeschlossen an Steckverbinder K1. Links oben werden der Lüftertyp (Tacho oder PWM) und die Methode der Steuerung oder Regelung eingestellt. Der I²C-Sensor mit der Adresse 0 (Temp_I2C_Addr0) misst beispielsweise die Temperatur der von außen angesaugten Luft, während der Sensor mit der Adresse 7 (Temp_I2C_Addr7) die Temperatur des PC-Gehäuses misst. Da der PC Wärme produziert, kann die Gehäuse-Innentemperatur nicht gleich der Außentemperatur sein. Deshalb ist die Regelung hier so eingestellt, dass die Innentemperatur die Außentemperatur um maximal 2 °C übersteigt (Wert „+2“ hinter dem Feld *Setpoint*). Die gewählte PI-Regelung berechnet die Differenz zwischen der Solltemperatur (Außentemperatur + 2 °C) und der gemessenen Gehäuse-Innentemperatur. Die Regelung steuert die Lüfterdrehzahl so, dass sie proportional steigt (Faktor Kp), gleichzeitig berücksichtigt sie integrierende Fehler über die Zeit (Faktor Ki).

Wie das Diagramm zeigt, läuft der Lüfter langsam an, wobei der Duty-Cycle des PWM-Signals 0 % beträgt. Sobald die Innentemperatur (dunkelblaue Kurve) die Solltemperatur übersteigt (hellblaue Kurve), erhöht die Regelung die Lüfterdrehzahl (grüne Kurve, in U/min). Wird nach einiger Zeit wieder die Solltemperatur erreicht, senkt die Regelung die Lüfterdrehzahl langsam ab.

Eine andere Anwendung des beschriebenen Systems ist das Synchronisieren mehrerer Lüfter, um mechanische Schwingungen im PC-Gehäuse zu minimieren. Dazu muss das System als PI-Regler arbeiten, die Messgröße ist die Drehzahl eines Lüfters, der als Referenz dient, und die Faktoren Kp und Ki erhalten negative Werte.

Im letzten Tab kann eingestellt werden, von welcher Größe das Signal des simulierten Tachogenerators abgeleitet werden soll. Für jeden Lüfter ist auch eine Mindestdrehzahl einstellbar. Unterhalb der Mindestdrehzahl wird der Lüfter als stillstehend interpretiert, das zur Hauptplatine übertragene Signal des simulierten Tachogenerators wird unterbrochen.

Von der Projektseite im Internet [3] können sowohl die Firmware für den Mikrocontroller als auch das PC-Programm heruntergeladen werden.

(100160)gd

Der Autor

Ivo Pullens ist freiberuflich in der Embedded-Software-Entwicklung tätig. Während der zurückliegenden Jahre war er mit umfangreichen Aufträgen aus der Industrie beschäftigt, bei denen es um Prozesse in der Halbleiterproduktion sowie um industrielle Feldbussysteme ging. In seiner Freizeit entwickelt der Autor gern Lösungen für Alltagsprobleme, meistens sind daran Mikrocontroller beteiligt.

Website des Autors: www.emmission.nl

Weblinks

[1] www.formfactors.org/developer/specs/4_Wire_PWM_Spec.pdf

[2] www.obdev.at/products/vusb/index.html

[3] www.elektor.de/100160

Neue 8-bit-Mikrocontroller mit integrierter konfigurierbarer Logik in 6- bis 20-poligen Gehäusen

Mit den neuen 8-bit-Mikrocontrollern PIC10F32X, PIC12F150X und PIC16F150X von Microchip können Sie zusätzliche Funktionalitäten in Ihre Anwendungen aufnehmen, die Größe reduzieren, Energie sparen und Kosten senken. Sie sind insbesondere für preisgünstige Anwendungen oder Einwegprodukte geeignet. Onboard befinden sich konfigurierbare Logikzellen (CLCs), ein komplementärer Funktionsgenerator (CWG) und ein numerisch gesteuerter Oszillator (NCO).

Die kombinatorische und sequentielle Logik lässt sich über die konfigurierbaren Logikzellen (CLCs) per Software steuern. Dies hat den Vorteil, dass Funktionalitäten hinzugefügt, externe Komponenten eliminiert und Codeplatz eingespart werden können. Der komplementäre Funktionsgenerator (CWG) hilft bei der Verbesserung der Schalteffizienz zwischen den verschiedenen Peripherien, während der numerisch gesteuerte Oszillator (NCO) eine lineare Frequenzeinstellung und höhere Auflösung der Anwendung ermöglicht, wie zum Beispiel in Tongeneratoren und Vorschaltgeräten.

Zusätzlich zur Einführung dieser neuen Peripherien bieten der PIC10F/LF32X und der PIC12/16F/LF150X MCUs einen internen 16-MHz-Oszillator, einen ADC, bis zu vier PWMs sowie ein integriertes Temperaturmessmodul zur preisgünstigen Temperaturmessung. Das alles ist in kompakten 6- bis 20-poligen Gehäusen untergebracht.

ENTWICKLUNGSWERKZEUGE FÜR DEN SCHNELLSTART

PICDEM™ Lab Entwicklungs-Kit - DM163045

PIC16F193X F1 Evaluationsplattform - DM164130-1

PICkit™ Demoplatine für geringe Anschlusszahlen - DM164120-1

Freie CLC-Konfigurationswerkzeuge:
www.microchip.com/get/euclctool

Erfahren Sie mehr über PIC® MCUs mit der Peripherie der nächsten Generation und geringer Anschlusszahl: www.microchip.com/get/eunew8bit

microchip DIRECT
www.microchipdirect.com

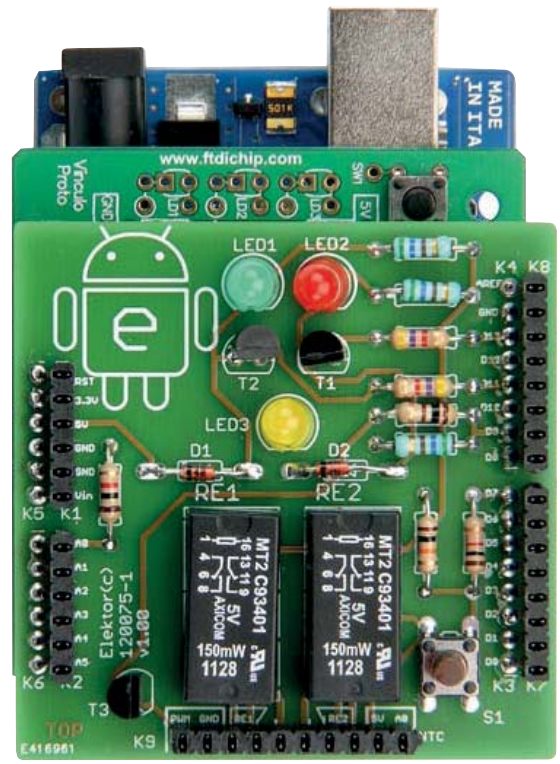
www.microchip.com

MICROCHIP

Schalten mit Android und Bluetooth

Android-Smartphone als Bedienteil und Fernsteuerung für Mikrocontroller-Projekte

Von Jos van Kempen (NL)



Hier zeigen wir, wie ein preisgünstiges Android-Smartphone als Benutzeroberfläche für ein Mikrocontroller-System eingesetzt werden kann. Zusammen mit einem Arduino-Board, ergänzt durch ein Bluetooth-Shield, realisieren wir drahtlose Mess- und Schaltfunktionen. Wir zeigen auch, wie die dazugehörige Android-App programmiert wird und welche kostenlose PC-Software für die Programmierung nötig ist.

Die Benutzeroberflächen von Mikrocontroller-Systemen bestehen typischerweise aus Drucktastern, Drehencodern, LEDs, LC-Displays oder Touchscreens. Nützlich für viele Zwecke können auch drahtlose Fernbedienungen sein, insbesondere wenn sie mit Funk arbeiten. Normalerweise verursachen attraktiv gestaltete Benutzeroberflächen zusätzliche Entwicklungsarbeit, sie treiben die Kosten in die Höhe. Wir wollen zeigen, dass eine solche Benutzeroberfläche einschließlich Fernbedienungsfunktion das Budget nicht übermäßig strapazieren muss und dass sie schnell und unkompliziert programmiert werden kann.

Oft ist ein Smartphone mit Bluetooth bereits vorhanden, es ist als Benutzeroberfläche und Fernbedienung wie geschaffen. Auf manchen Mikrocontroller-Boards gehört Bluetooth bereits zum Standard, andere Boards können mit Adaptern oder Shields nachgerüstet werden. Wir haben uns für ein Arduino-Standard-Board entschieden, das durch ein Bluetooth-Shield ergänzt wird (*ITEAD, ca. 15 €*). Dazu kommt ein kleines I/O-Shield, das von uns entworfen wurde. Elektor hat bereits im November 2004 und Januar 2010 beschrieben, wie Bluetooth zu eigenen Mikrocontroller-Projekten hinzugefügt werden kann. Da viele Elektor-Leser mit der Kunst der Mikrocontroller-Programmierung vertraut sind, jedoch vielleicht noch nie ein Smartphone programmiert haben, wird die Mikrocontroller-Seite nachfolgend nur kurz gestreift. Der Schwerpunkt liegt auf dem Programmieren des Smartphones, vom Download und Installieren der Werkzeuge bis zur funktionierenden Benutzeroberfläche. Alle Quellcodes, die wir verwenden, können frei von der Elektor-Website [1] heruntergeladen werden.

Hardware und Mikrocontroller-Programm

Auf dem Arduino-Board wird ein Bluetooth-Shield montiert, die Daten werden über die UART-Schnittstelle übertragen. In Bascom können mit Befehlen wie *input* oder *print* Kommandos empfangen werden, ebenso ist das Übertragen von Daten, beispielsweise von Messwerten, zum Arduino-Board möglich.

Für unser Projekt haben wir ein einfaches Shield entworfen, auf dem sich einige LEDs, zwei Relais (digitale Ausgänge), ein PWM-Ausgang mit einem FET (quasi-analoger Ausgang), ein Schalter (digitaler Eingang) und ein NTC als analoger Sensor (analoger Eingang) befinden. Die Schaltung geht aus **Bild 1** hervor, **Bild 2** zeigt das Platinenlayout. Mit diesem Shield können alle Kommunikationskanäle zwischen Android-Smartphone und Arduino-Board über Bluetooth getestet werden. Sämtliche Relais-, Schalter- und Sensor-Anschlüsse sind über den Steckverbinder K9 zugänglich.

Das Mikrocontroller-Programm, geschrieben in Bascom (**Listing 1**), ist wie folgt strukturiert:

In einer Schleife wird geprüft, ob ein Zeichen empfangen wurde. Wenn der Arduino das Zeichen „R“ empfängt, wird Ausgang D11 aktiviert, so dass Relais RE1 einschaltet und LED1 aufleuchtet. Wird ein „r“ (kleiner Buchstabe) erkannt, schaltet Relais RE1 ab und LED1 verlischt. Beides gilt sinngemäß auch für Ausgang D13, RE2 und LED2, dieser Ausgang reagiert jedoch auf die Zeichen „O“ und „o“. Nach einem „P“, abgeleitet von „PWM“, wird mit *input* auf einen Wert gewartet. Abschließend muss das Smartphone den Zeilenende-Code „r/n/“ senden. Der übertragene Wert steuert das Signal an FET T3 (PWM-Ausgang), der FET ist bis 60 V/0,5 A belastbar. Die LED3 gibt das Impuls-Pause-Verhältnis des PWM-Signals wieder.

Ferner werden während jedes Schleifenzyklus die Werte von NTC R9 im Format T;adc(0);t und der logische Zustand des Eingangs D7 ausgegeben („G“ bei 1 oder „g“ bei 0). Die Geschwindigkeit beträgt 9600 Baud, nach jeder Übertragung wird eine kurze Wartezeit (30...40 ms) eingefügt.

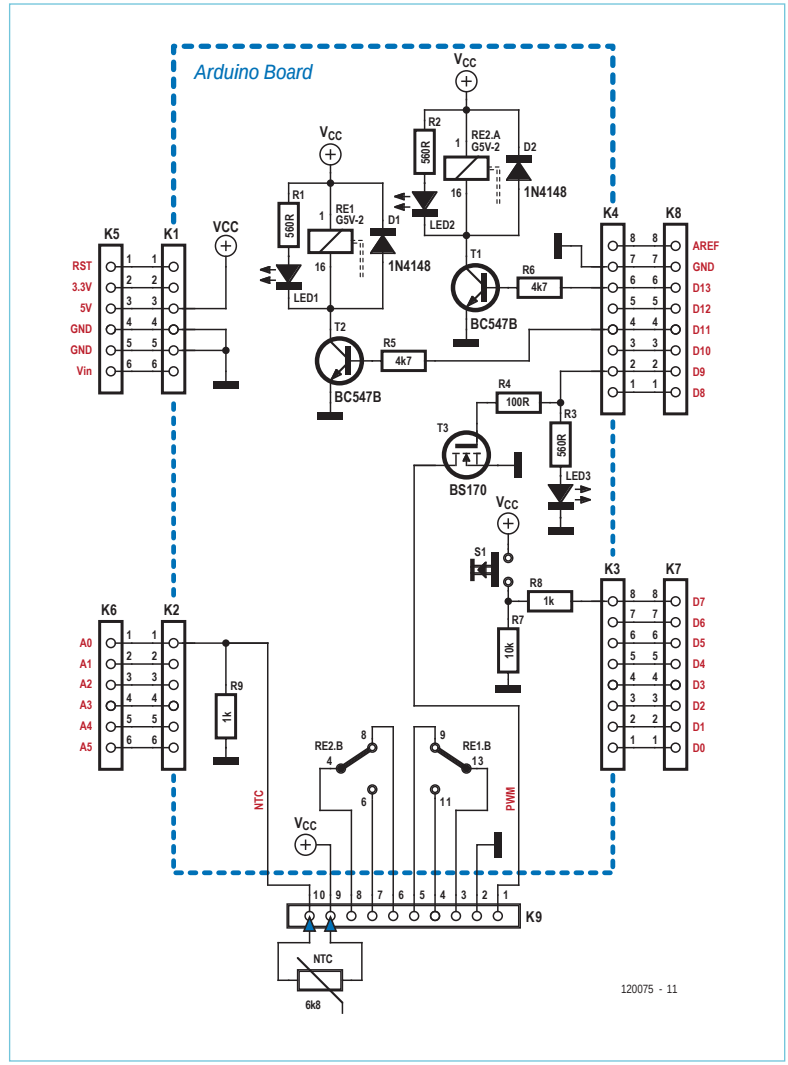


Bild 1. Auf dem Arduino-Shield befinden sich mehrere LEDs und Relais, ein NTC-Widerstand, ein Drucktaster und ein PWM-Ausgang mit Anzeige-LED.

Stückliste

Widerstände:
R1,R2,R3 = 560 Ω
R4 = 100 Ω
R5,R6 = 4k7
R7 = 10 k
R8,R9 = 1 k

Halbleiter:
D1,D2 = 1N4148
LED1,LED2,LED3 = LED rot, 5 mm
T1,T2= BC547
T3 = BS170

Außerdem:
K1,K2 = Stiftleiste 6-polig
K3,K4 = Stiftleiste 8-polig
K5,K6 = Buchsenleiste 6-polig
K7,K8 = Buchsenleiste 8-polig
K9 = Buchsenleiste 10-polig
RE1,RE2 = Relais 5 V (z.B. TE Connectivity MT2-C93401 oder OMRON G5V-2-H1)
S1 = Drucktaster mit Arbeitskontakt (z.B. B3F-1000)
Platine 120075-1 (siehe [1])

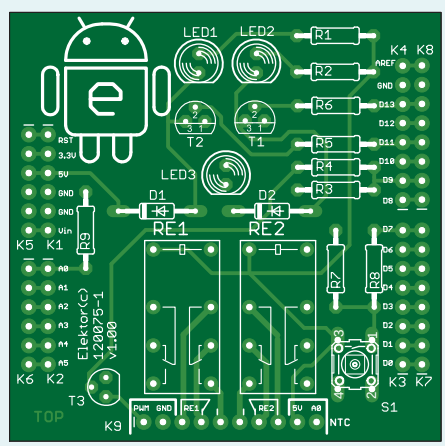


Bild 2.
Die Platine ist so ausgelegt, dass die Steckverbinder zum Arduino-Standard-Board passen.

Listing 1. Das Mikrocontroller-Programm, geschrieben in Bascom.

```
$baud = 9600 : Ucsr0a = &H00                                    'Thanks to J.F. Theinert
'Software under CC-BY-NC-SA licence by Jos van Kempen.
Config Adc = Single , Prescaler = Auto , Reference = Avcc       'ADC (analog) input initialize
Start Adc                                                       'PWM (analog) output initialize
Config Timer1 = Pwm , Pwm = 8 , Compare A Pwm = Clear Down , Prescale = 8
 Config Pinb.1 = Output                                         'PB1 =digpin9=pwm1a
 Pwm1a = 0
 Dim Pwm_str As String * 5                                      'text 0-255
 Dim Pwm_b As Byte
 Dim Value As Integer
 D13 Alias Portb.5 : Config D13 = Output                        'Dig13 no resistance needed for LED
 D11 Alias Portb.3 : Config D11 = Output                        'Dig11
 D7 Alias Pind.7 : Config D7 = Input                            'Initialize DigInput Dig7
 Declare Sub Set_pwm
 Dim B As Byte
 Do
  B = Ischarwaiting()
  Print B
   If B = 1 Then                                                'if incoming command
        B = Waitkey()
        Select Case B
        Case "R" : D11 = 1
        Case "r" : D11 = 0
        Case "O" : D13 = 1
        Case "o" : D13 = 0
        Case "P" : Call Set_pwm
        End Select
  End If
  Waitms 300
  Pwm_b = Pwm_b + 3
  Value = Getadc(0)                                             'A0
  Print "T" ; Value ; "t"
  Waitms 40
  If D7 = 1 Then Print "G" Else Print "g"
  Waitms 30
 Loop
End
Sub Set_pwm
  Input Pwm_str Noecho
  Pwm_b = Val(pwm_str) : Waitms 30
  Pwm1a = Pwm_b
  Print "*" ; Pwm_b : Waitms 30
End Sub
```

Software für die App-Entwicklung

Die gesamte Software zum Programmieren einer App für ein Android-Smartphone kann kostenlos aus dem Internet heruntergeladen werden:

- Die Programmiersprache ist Java, das Java Development Kit (JDK) können Sie von der Website von Oracle [2] herunterladen.
- Das Software Development Kit für Android finden Sie bei android.com [3]. Werfen Sie nach der Installation auch einen Blick auf die übrige Website. Sie enthält zahlreiche Informationen und Tipps zum Programmieren, ebenfalls vorhanden sind Hilfe-Dokumente und USB-Treiber zum Laden der Apps vom PC in das Smartphone. Wenn Sie Musik oder Fotos über USB zwischen PC und Smartphone austauschen, ist der passende Treiber möglicherweise schon installiert.
- Laden Sie das ADT-Plugin für Eclipse von der gleichen Website herunter, notieren Sie den Ordner, in dem das Plugin abgelegt wird.
- Als Entwicklungsumgebung (IDE) dient Eclipse Classic 3.6.2, sie ist von eclipse.org [4] herunterladbar. Als dieser Beitrag geschrieben wurde, war ADT12 das neueste Android-Plugin, es ist mit Eclipse 3.7 nicht kompatibel.
- Nach der Installation von Eclipse muss das Android-Plugin über *Help, Install New Software, Archive* eingebunden werden. Selektieren Sie die ZIP-Datei des ADT und geben sie ihr den Namen *Android plugin*. Gehen Sie anschließend nach *Windows, Preferences* und suchen Sie den Ordner, der das entpackte Android-SDK enthält. Dort sind Unterordner vorhanden, zum Beispiel *Tools*.

Gehen Sie nach *Windows, Android SDK and AVD Manager*, wählen Sie *New (Virtual Device)*, beispielsweise Gingerbread für Version 2.3.3 (oder auch Samsung_GIO). Damit kann das Programm ohne Smartphone auf dem PC simuliert werden.

- Laden Sie den Ordner mit dem Projektbeispiel „Bluetoothinterface“ von der Elektor-Website [1] herunter, Ziel des Downloads darf nicht der Workspace von Eclipse sein.

Projekt „Bluetoothinterface“

Die Software, die über Bluetooth andere Bluetooth-Geräte findet, Verbindungen herstellt und Daten austauscht, ist ziemlich komplex. Zum Android-SDK gehört jedoch ein Beispiel, das Orientierungshilfe geben kann. Leider hat sich herausgestellt, dass dieses Beispiel für viele Bluetooth-Geräte nicht geeignet ist. Im Internet gibt es einen Ersatz für die Datei *BluetoothRfcommClient.java*, mit dem dieses Problem nach Anpassen einiger Deklarationen lösbar ist (siehe [5] und [6]).

Nach dem Starten von Eclipse kann mit *File, New, Android project* ein neues Projekt ins Leben gerufen werden, gegebenenfalls auf der Grundlage eines Beispiels. Im diesem Fall gibt es jedoch schon ein Projekt, das nur in den Workspace importiert werden muss. Das geschieht mit *File, Import, Existing Projects into Workspace*. Die Option *Copy Projects into Workspace* muss aktiviert sein, anderenfalls geht das Original verloren. Gehen Sie zu dem Ordner, in den Sie das Projekt „Bluetoothinterface“ abgelegt haben.

Aus der Verzeichnisstruktur des Projekts geht hervor (eventuell nach Klicken auf die Ordner), dass bereits diverse .java- und .xml-Dateien existieren (**Bild 3**). Hier sind die Dateien *BluetoothChat.java* (Hauptprogramm), *main.xml* (Interface auf dem Smartphone) und *strings.xml* (Deklaration der Interface-Variablen) von Bedeutung.

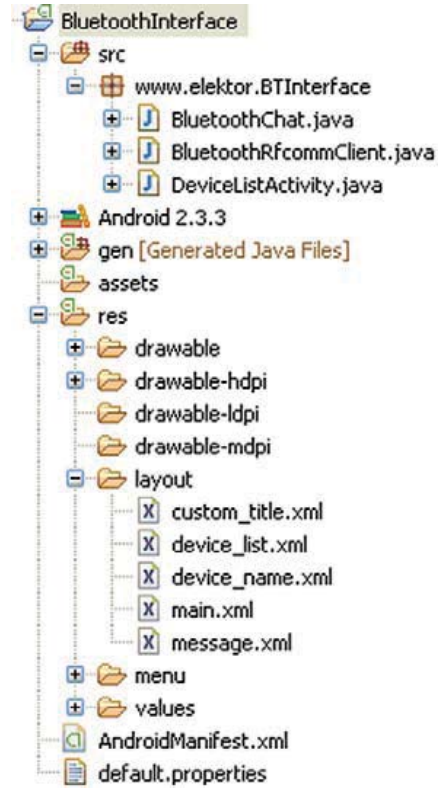


Bild 3. Dieser Verzeichnisbaum gehört zum Projekt „Bluetoothinterface”.

Realisieren der Benutzeroberfläche

Nach einem Doppelklick auf *main.xml* wird die Benutzeroberfläche auf dem Smartphone sichtbar (**Bild 4**). Mit den darunter befindlichen Tabs kann zwischen dem Aussehen und dem erzeugten Code umgeschaltet werden.

Auf der Benutzeroberfläche ist bereits ein horizontales Layout definiert, das unter anderem eine Listenansicht (für die erkannten Bluetooth-Geräte), einen Texteditor (für den Text während eines Chats) und eine Schaltfläche (Button) für das Absenden enthält.

Meistens ist es am einfachsten, die Steuerelemente durch Verschieben in das Fenster *Outline* zu platzieren. Wenn der PC für die Notation mit Kommata konfiguriert ist, können beim Verschieben Fehlermeldungen erscheinen. In diesem Fall muss das Komma-Zeichen (,) in der Datei *main.xml* durch den Punkt (.) ersetzt werden.

Beim Platzieren einer Schaltfläche oder eines Radiobuttons ist gleichzeitig die Prozedur definierbar, die beim Anklicken ausgeführt wird. Es genügt, bei der Eigenschaft *On click* den Namen der Prozedur einzutragen.

Die LEDs und Relais (digitale Ausgänge) auf dem Shield werden über nebeneinander angeordnete Schaltflächen oder Checkboxen bedient. Dazu wird auf der Benutzeroberfläche oder im Fenster *Outline*, was oft einfacher ist, ein horizontales Layout (*linear layout*) mit Checkboxen oder (Toggle-)Schaltflächen platziert.

Den Zustand des Schalters (digitaler Eingang) zeigen auf der Benutzeroberfläche die beiden Radiobuttons „On“ und „Off“ an (*Radiogroup orientation horizontal*).

Eine *Horizontal Progressbar* (*Style Horizontal*), ein *Textview* und ein *ImageView* geben den Wert des analogen Eingangs wieder.

Der PWM-Ausgang mit dem FET wird über einen Schiebebalken (*SeekBar*) gesteuert.

Bis jetzt haben wir das Smartphone noch nicht programmiert, doch wir können uns ansehen, wie die Benutzeroberfläche aussehen wird. Der PC kann eine Simulation durchführen, bei der ein virtuelles

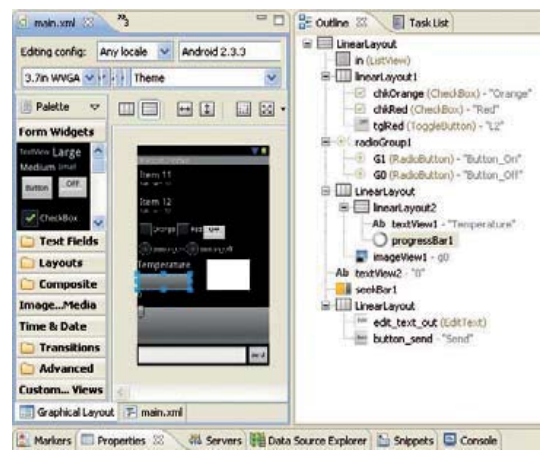


Bild 4. Hier wird das Layout der Benutzeroberfläche erstellt. Die Bedienelemente können im „Outline”-Fenster platziert werden.

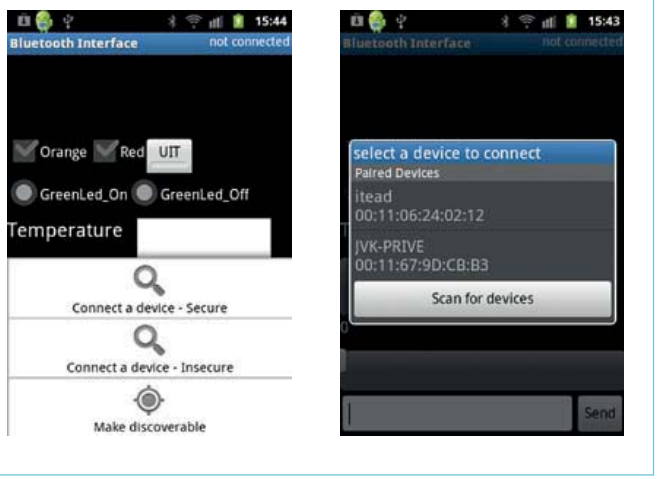


Bild 5. Nach Anklicken von „Connect a device – Secure" kann ein verbundenes Gerät ausgewählt oder nach einem „sichtbaren" Gerät gesucht werden. Die Verbindung kommt nach Eingabe eines Codes zustande (oft „0000" oder „1234").

Smartphone auf dem Bildschirm erscheint. Die Bluetooth-Verbindung lässt sich jedoch nicht simulieren.

Verbinden Sie daher das Smartphone über USB mit dem PC, selektieren Sie den zugehörigen Projektordner, drücken Sie die rechte Maustaste und wählen Sie *Run As, Android Application*. Die App wird compiliert und in das Smartphone geladen.
Jetzt können die Informationen vom Smartphone empfangen und sichtbar gemacht werden, vorausgesetzt natürlich, der Bluetooth-Sender ist eingeschaltet.

Tests sind auch zusammen mit einem Bluetooth-USB-Stick am PC möglich. Ein Terminalprogramm wie *Hyperterminal* oder *Advanced Serial Port Terminal* kann die Bluetooth-Schnittstelle steuern. Zuvor müssen im Terminalprogramm der COM-Port und die Übertragungsgeschwindigkeit eingestellt werden.
Nachdem Sie die App geladen und gestartet haben, stellen Sie eine Verbindung her, indem Sie die Schaltfläche links unten drücken. Nach *Connect to device - secure* wählen Sie das Bluetooth-Gerät aus. Wenn alles nach Plan verlief, erscheint die Meldung „Connected" und kurz darauf „Connected: <Gerätename>" (siehe **Bild 5**).

Bild 6. Die Benutzeroberfläche auf dem Smartphone im Einsatz. Das Feld ganz unten dient hier zum Senden von Kommandos, was während der Testphase nützlich ist. Wenn es später entbehrlich wird, kann es durch eine andere Funktion ersetzt werden.

Der Inhalt der Quellcode-Datei *BluetoothChat* sieht beispielsweise wie folgt aus:

```
private CheckBox chkD13, chkD11,chkDIM;
   (Falls die Checkbox-Eigenschaften im Programm gelesen oder geschrieben werden, ist die Variable zu deklarieren.)

chkD13 = (CheckBox) findViewById(R.id.chkD13);
   (Bei der Prozedur onCreate wird der Link zwischen dem Namen in main.xml (Bildschirm-Layout) und dem Namen im Programm hergestellt.)

public void chkD13Click(View view){
       if (chkD13.isChecked()==true)
          sendMessage(„O"); else sendMessage(„o");
}
```

Der Programmcode der Prozedur ist einfach. Abhängig vom Status der Checkbox werden entweder ein „O" oder ein „o" gesendet. Für den Mikrocontroller sind dies die Kommandos, Relais RE2 und LED2 ein- oder auszuschalten.

Der Code, der einen vom Fortschrittsbalken abhängigen Wert sendet, ist etwas komplizierter. Bei der Eingabe weist die „Quick Fix"-Hilfe auf Eingabefehler hin, und der größte Teil des Codes wird selbsttätig erzeugt. Ferner wird bei der Deklaration der Klasse automatisch *implements OnSeekBarChangeListener* hinzugefügt.

```
public void onProgressChanged(SeekBar seekBar, int
progress, boolean fromUser) {
      // TODO Auto-generated method stub
      textView2.setText(Integer.toString(progress));
}
public void onStartTrackingTouch(SeekBar seekBar) {
      // TODO Auto-generated method stub
      textView2.setTextColor(Color.rgb(255, 48, 48));
}
public void onStopTrackingTouch(SeekBar seekBar) {
      // TODO Auto-generated method stub
      textView2.setTextColor(originalTextColor);
      sendMessage("P");
      try{
         Thread.sleep(1000);
      } catch (InterruptedException e) {
         // TODO Auto-generated catch block
         e.printStackTrace();
```

```
        }
        sendMessage (seekBar1.getProgress()+
"\r\n");
}
```

Selbst implementiert werden muss das Aktualisieren der Textbox mit dem aktuellen Wert, wenn sich der Fortschrittsbalken verschiebt. Die Textfarbe wird geändert, solange das Verschieben stattfindet, und ferner wird das Zeichen „P“ gesendet. Für den Mikrocontroller ist dies das Signal, dass die Prozedur aufgerufen werden muss, in der mit *Input Pwm_str* eine Zeile mit dem gewünschten PWM-Wert gelesen wird. Anschließend sendet das Smartphone eine Zeile mit dem Wert, abgeschlossen durch ein Zeilenende-Zeichen. Es ist auch eine Prozedur vorhanden, die Informationen vom Mikrocontroller empfängt und im Textfeld ausgibt. Maßgeblich ist der folgende Programmcode:

```
case MESSAGE_READ:
   byte[] readBuf = (byte[]) msg.obj;
   // construct a string from the valid bytes
      in the buffer
   String readMessage = new String(readBuf, 0,
      msg.arg1);
   mConversationArrayAdapter.
add(mConnectedDeviceName+
      ":  " + readMessage);
break;
```

Nun müssen nur noch die Radiobuttons und der Fortschrittsbalken abhängig von den empfangenen Informationen die entsprechenden Werte erhalten. Die Radiobuttons werden von den Zeichen „G“ (Ein) oder „g“ (Aus) gesteuert:

```
   if (readMessage.contains("G")==true)
G1.setChecked(true);
      if (readMessage.contains("g")==true)
G0.setChecked(true);
```

Um den Fortschrittsbalken zu steuern, wird geprüft, ob ein Code im Format „TXXXXt” empfangen wurde. Dabei ist „XXXX” eine Zahl 0...1023, die den digitalisierten Wert des analogen Eingangs wiedergibt.
Da diese Information oft in zwei Stücken übermittelt wird, ist etwas mehr Aufwand nötig:

```
   if (readMessage.charAt(0)=='T')
      if (readMessage.contains("t")==true){
         Tempoud=Temp;Temp=readMessage.indexOf("t");
         OldMessage=readMessage.substring(1,Temp);
         Temp=Integer.parseInt(OldMessage);
```

Die Roh-Werte müssen für die Textdarstellung in Grad Celsius, den Fortschrittsbalken und die Grafik umgerechnet und formatiert werden.

```
RTemp=Temp;RTemp=RTemp/6;// 0-255 naar grdC
   // nooit hoger dan maximum progressbar
A0.setProgress(Temp*4); //schaal progressbar
Temp=Temp/2;         //schaal grafiek
DecimalFormat formatter = new DecimalFormat("#.#");
textView1.setText(formatter.format(RTemp));}
```

Die Werte werden grafisch dargestellt. Die Grafik wird gelöscht, wenn die Anzahl der Werte größer als 150 ist. Danach wird ein neues Diagramm einschließlich Hilfslinien erzeugt.

```
xcoordoud=xcoord;xcoord+=1;
if (xcoord==1){   //tekenen kader en hulplijnen
paint.setColor(Color.BLUE)
canvas.drawLine(0,105,150,105,paint);//10 grd
canvas.drawLine(0,85,150,85,paint);//15 grd
canvas.drawLine(0,65,150,65,paint);//20 grd
canvas.drawLine(0,45,150,45,paint);//25 grd
canvas.drawLine(0,25,150,25,paint);//30 grd
paint.setColor(Color.YELLOW);
canvas.drawRect(1, 1, 149, 124, paint);
}
if (xcoord==150) {xcoord=0;
   canvas.drawColor(Color.BLACK);}
else
{canvas.drawLine(xcoordoud,
   125-Tempoud,xcoord,125-Temp,paint);
}
OldMessage=readMessage;
```

Damit die Anwendung nicht mit der Anzeige einer Tastatur startet, ist in *Androidmanifest, Application, Window* die Option *soft input mode stateHidden* eingestellt.
Die vorstehenden Informationen sollten genügen, um eine Android-Smartphone-Benutzeroberfläche für eine Mikrocontroller-Anwendung maßzuschneidern. Zum Download auf der Projektseite [1] gehört eine vollständige App, die unverändert in das Smartphone geladen werden kann. Mit der App können alle beschriebenen Möglichkeiten erprobt werden.

(120075)gd

Weblinks

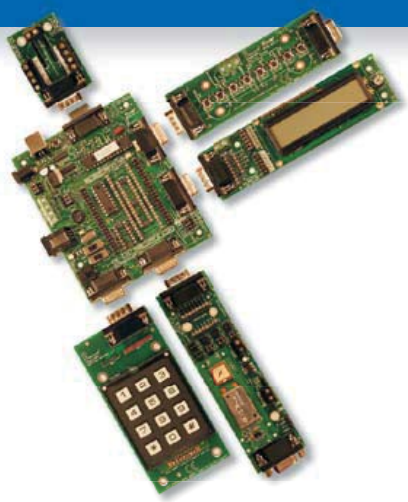
[1] www.elektor.de/120075
[2] www.oracle.com/technetwork/java/javase/downloads/
[3] http://developer.android.com/sdk/index.html
[4] www.eclipse.org/downloads/
[5] http://projectproto.blogspot.com/2010/09/android-bluetooth-oscilloscope.html
[6] http://code.google.com/p/android-bluetooth-oscilloscope/

Entwickeln und Lernen

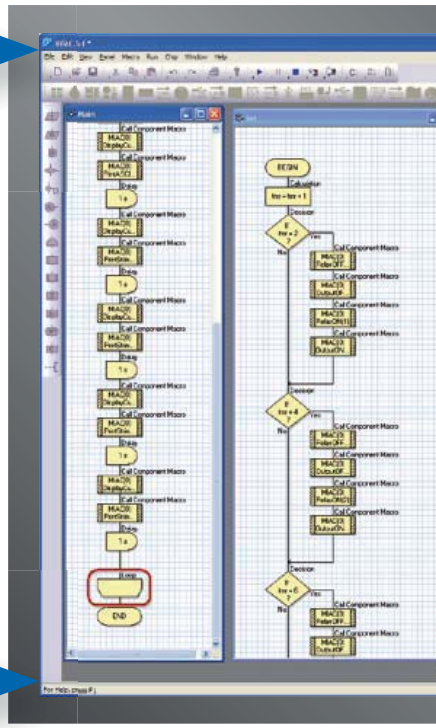
FLOWCODE5

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 40 unterschiedliche Platinen. Die Module reichen von einfachen LED-Boards 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.



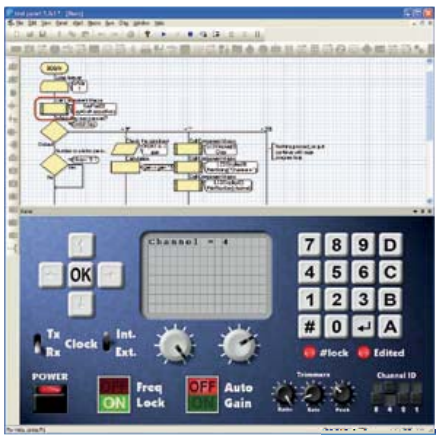
… für Industrie-Steuerungen

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 welchen mehrere MIACs vernetzt werden können. Flowcode gehört zum Lieferumfang aller verfügbaren MIAC-Bundles.

FlowKit

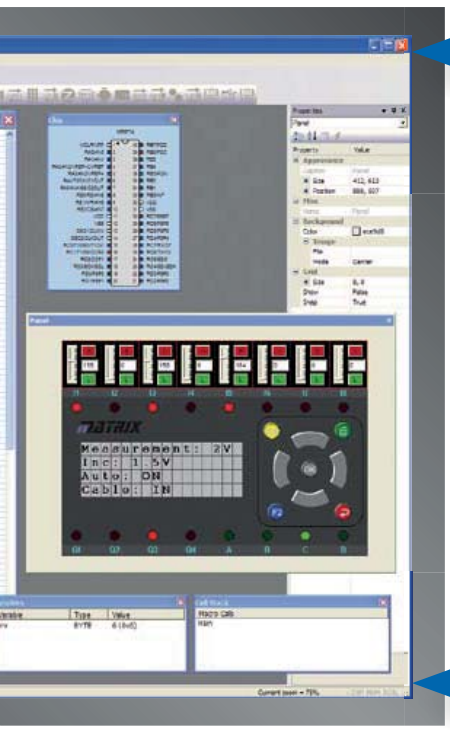
Das FlowKit-Modul ermöglicht In-Circuit-Debugging für Flowcode-Anwendungen in PIC- und AVR-Projekten:
- Start, Stopp, Pause und Schritt für Flowcode-Programme in Echtzeit
- Anzeige der Programm-Variablen
- Ändern von Variablenwerten
- In-Circuit-Debugging für Formula Flowcode Buggy, ECIO- und MIAC-Projekte

mit Flowcode 5 ...

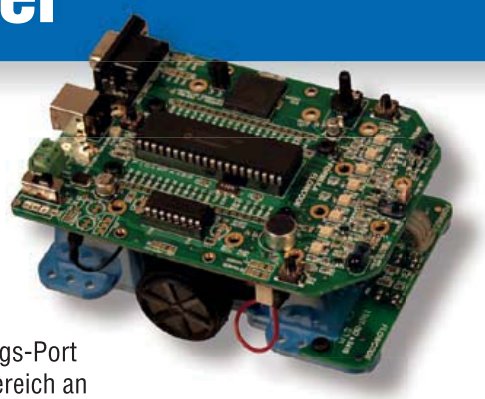


NEU in Flowcode 5:

- 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 Erweiterungsport 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

Bauteil-Tipps

Von Raymond Vermeulen (Elektor-Labor)

USB-Port-Schutz

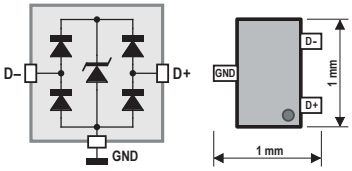
USB-Ports gelten als unproblematisch, trotzdem sind einige Punkte zu beachten. Damit der Strom der +5V-Leitung den erlaubten Wert nicht übersteigt, wird diese meistens gegen Überstrom und Kurzschluss gesichert. Die konventionelle oder elektronische Sicherung schaltet häufig mit beträchtlichen Toleranzen. Die Differenz zwischen dem höchsten Strom, bei dem die Sicherung noch nicht reagiert, und dem Mindeststrom, bei dem die Sicherung garantiert auslöst, beträgt oft 100 % (!). Ferner müssen die Datenleitungen eines USB-Ports gegen Überspannung geschützt werden. Konventionelle Schutzvorkehrungen beanspruchen Platinenfläche, und die Datenleitungen werden kapazitiv belastet, was die Signalform beeinträchtigen kann. Wir stellen hier zwei Bauelemente vor, die solche Randprobleme lösen können.

(120095)gd

TPD2EUSB30ADRTR

Dieses IC schützt vor Überspannungen (ESD) auf den Datenleitungen. Das ist nicht sensationell, auch der Preis liegt im üblichen Rahmen, trotzdem gibt es einige Highlights: Die Abmessungen betragen nur 1 x 1 mm (!), so dass das IC auf der Platine zwischen die USB-Datenleitungen passt. Nach Herstellerangaben eignet es sich auch für Datenleitungen, die Signale mit sehr hohen Frequenzen (USB 3.0, SATA, PCIe) transportieren. Wahrscheinlich sind hier auch andere Bausteine brauchbar, doch diese dürften die Signalform stärker beeinflussen. Eine Betriebsspannung benötigt der TPD2EUSB30ADRTR nicht. Die Breakdown-Spannung beträgt 4,5 V, so dass 3,3V-Eingänge zuverlässig geschützt werden. Gleichzeitig bedeutet dies, dass das IC nicht an Datenleitungen mit höheren Signalspannungen als 3,3 V einsetzbar ist. Der Hersteller bezeichnet einen Anschluss mit D+, den anderen mit D-, obwohl die Polarität eigentlich vertauschbar sein müsste. Wie dem auch sei, der USB-Eingang ist gegen ESD bis ±8 kV geschützt, gleichzeitig ist der Einfluss auf das Datensignal minimal.

TPD2EUSB30ADRTR

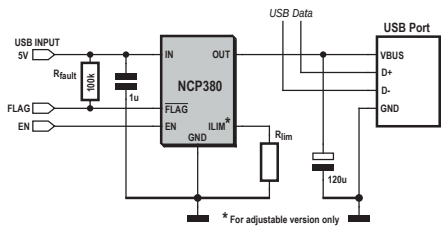


PARAMETER	CONDITION	VALUE
Reverse stand-off voltage	D-, D+	3,6 V
Breakdown voltage	I_{io} = 1 mA	4,5 V
ESD protection	D-, D+	8 kV
C_{io-io}	V_{io} = 2,5 V	0,05 pF
C_{io-gnd}	V_{io} = 2,5 V	0,7 pF

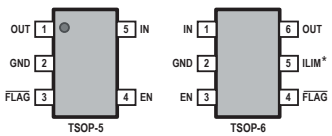
Weblink: www.ti.com/lit/ds/symlink/tpd2eusb30a.pdf

NCP380LSN05AAT1G

Mit diesem IC lässt sich der Strom auf der +5V-Leitung zuverlässig begrenzen. Zuerst fällt der Preis auf, er liegt um etwa die Hälfte niedriger als bei manchem vergleichbaren Produkt. Das IC ist zwar für den Einsatz in USB-Hosts vorgesehen, doch in USB-Clients kann es ebenso nützliche Dienste leisten, zum Beispiel wenn dort der Strombedarf nahe der Obergrenze liegt. Falls ein USB-Client nicht nur aus dem USB-Bus, sondern auch aus einer externen Quelle mit Strom versorgt werden kann, schaltet der NCP380LSN05AAT1G ab, sobald die Spannung am Ausgang um mehr als 100 mV über der Spannung am Eingang liegt. Für USB 2.0 bietet sich die Variante mit dem Grenzstrom 500 mA an. Das IC schützt nicht nur gegen Überströme und Kurzschlüsse, sondern auch gegen ESD. Darüber hinaus sind ein Soft-Start und ein Soft-Shutdown integriert.
Im leitenden Zustand beträgt der Übergangswiderstand nur 70 mΩ, bei konventionellen Sicherungen kann dieser Wert um das Zehnfache höher sein. Falls sich das IC überhitzt oder die Eingangsspannung unter einen Grenzwert sinkt, schaltet sich das IC selbsttätig ab. Diese und andere Fehlerzustände werden über den „Flag“-Anschluss nach außen gemeldet.



NCP380LSN05AAT1G

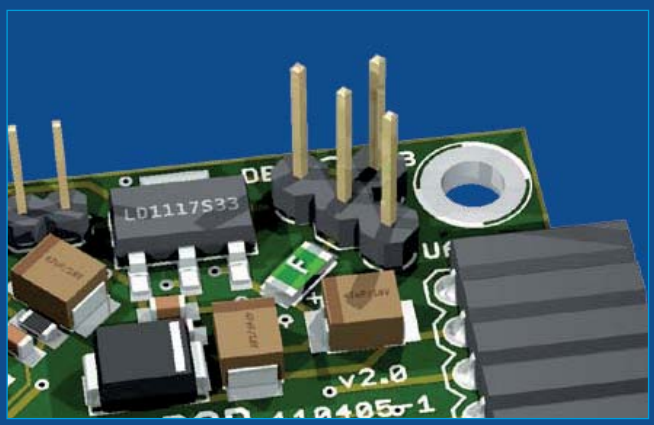


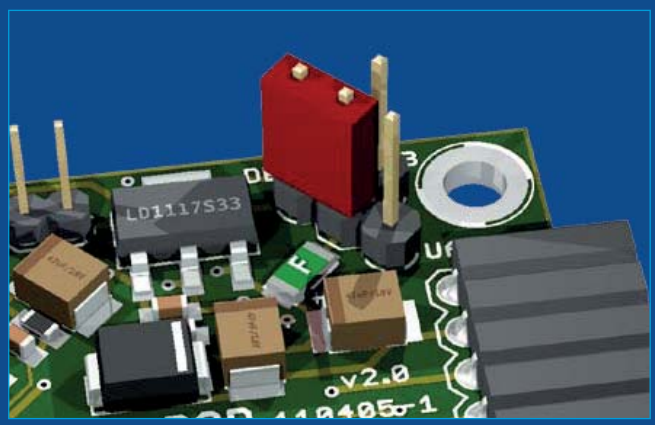
PARAMETER	CONDITION	VALUE
V_{in}	Operational voltage	2,5 V...5,5 V
V_{rev} (V_{out} – V_{in})	Trip point	100 mV
$R_{DS(on)}$	V_{in} = 5 V	70 mΩ
Current limit	V_{in} = 5 V	[Min 0,5 A] [Typ 0,58 A] [Max 0,65 A]

Weblink: www.onsemi.com/pub_link/Collateral/NCP380-D.PDF

AndroPod: Jumper mit drei Richtungen

Zunächst sieht man vielleicht sogar vier davon, aber genauer hingeschaut…





DEBUG-Position

Von Antoine Authier (Elektor Labs)

AndroPod ist ein Interface von Elektor, das die Kommunikation zwischen Android-Smartphones und anderen Schaltungen erlaubt. Bemerkenswert ist die Vielfalt an Stromversorgungs-Möglichkeiten: per USB, UART, Debugging-Modul oder RS485-Erweiterung. Diese Flexibilität forderte den Entwickler heraus: Wie vermeidet man ohne teure mechanische Bauteile Konflikte der Stromversorgungen? Das Ergebnis des Wälzens von dicken Bauteile-Katalogen und Diskussionen mit chinesischen Freunden war, dass es kaum möglich bis komplett unmöglich ist, einen kleinen Schiebeschalter mit vier Positionen zu bekommen, der 500 mA oder gar 1 A aushalten kann.

Das Problem wurde richtig virulent, als es ans Design der Platine ging, denn ich war der Platinen-Layouter. Mein Ziel war, dass das Interface praktisch, einfach zu benutzen und leicht nachzubauen sein sollte. Eine Lösung wäre die Verwendung mehrerer Jumper gewesen, doch ist das riskant, da sich damit viele Steckmöglichkeiten ergeben. Bei nachlassender Konzentration ist schnell etwas falsch gesteckt und das Ergebnis dann destruktiv. Die Lösung brachte mein Kollege Ton Giesberts, der wohl Mitleid mit mir hatte, als ich mich in Jumper-Irrgärten zu verlaufen drohte. Er meinte: „Sind das nicht etwas viele Jumper?“. Und ohne auf eine Antwort zu warten fuhr er fort: „Warum nicht als Stern anordnen?“

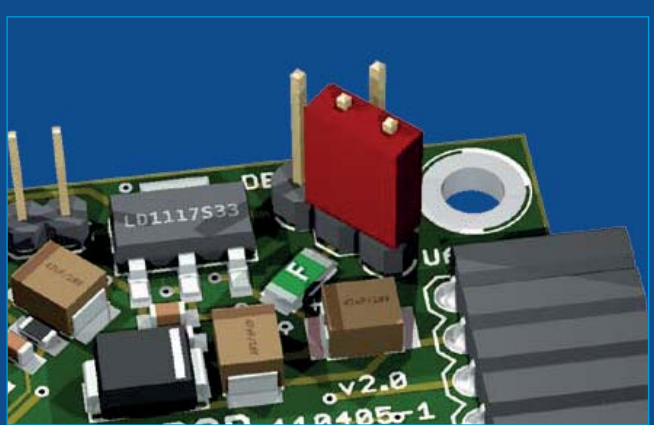
Heureka! Donnerwetter! Das war’s! Alle Achtung, Ton! Denn diese Anordnung schaut nicht nur ungewöhnlich aus, sondern verhindert auch ein falsches Stecken, da sie die Versorgung durch eine einzige Spannungsquelle erzwingt.

Viel Spaß mit AndroPod!

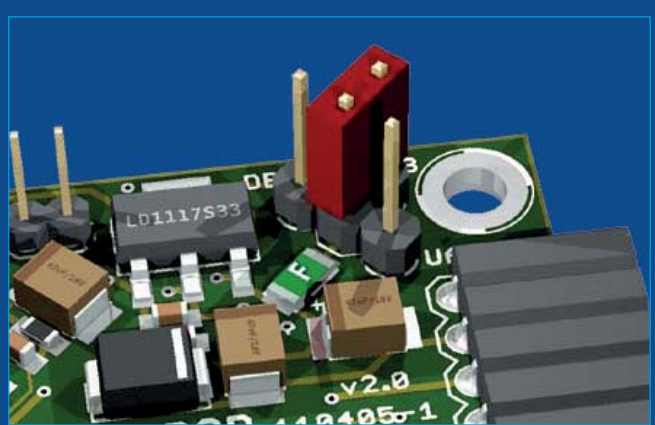
Antoine

P.S.: Nochmals vielen Dank, Ton!

(120076)



USB-Position



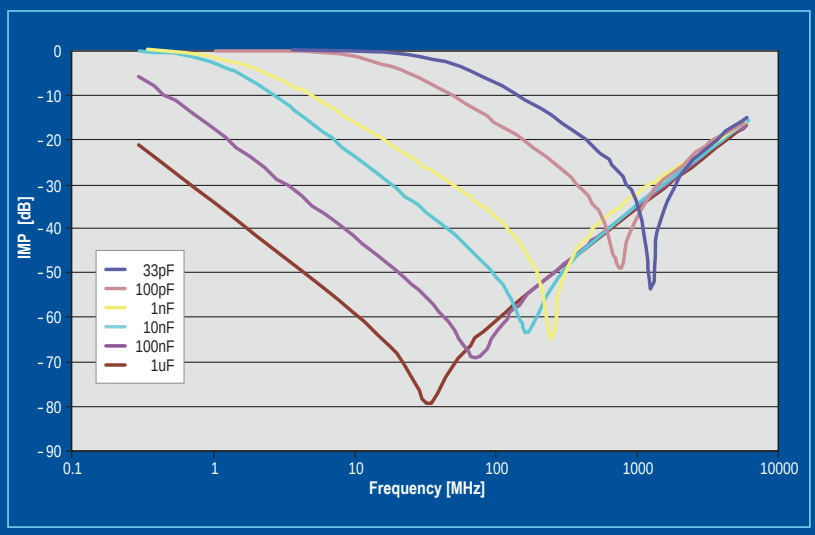
UART-Position

Wanted: Schaltnetzteil-Ausgangsfilter

Von Raymond Vermeulen (Elektor Labs)

DC/DC-Konverter können sehr nützlich sein! Manchmal sind sie sogar unentbehrlich und zum Glück meistens sehr effizient. Ich selbst habe sie nach und nach immer öfter eingesetzt, richtig professionell eben.

Als ich eines Tages an einer per Batterie versorgten Schaltung arbeitete, schien der Einsatz eines DC/DC-Konverters mit sehr hohem Wirkungsgrad zur Erzeugung der notwendigen Spannungen unerlässlich. Während ich den Siebkondensator der Stromversorgung berechnete, kam mir plötzlich Lernstoff aus Studienzeiten zum Thema passive Filter und Schaltnetzteile in den Sinn.



Anders als lineare Regler wie die bekannte 78XX-Serie produzieren Schaltnetzteile oft ordentliche Störpegel am Ausgang. Digitalen Schaltungen macht das in der Regel weniger aus als analogen. Daher ist der zu treibende Filteraufwand abhängig vom Anwendungszweck. Datenblätter für Schaltnetzteile suggerieren häufig recht geringe Kapazitäten zur Filterung, beispielsweise lediglich einen 1-µF-Kondensator zwischen Ausgang und Masse. Und vielfach tendiert man automatisch zu größeren Werten wie etwa 10 µF. Doch hier ein Aber:

Eine zu wenig beachtete Daumenregel besagt, dass mit der Kapazität eines Kondensators auch seine Induktivität steigt, was bei höheren Frequenzen relevant wird (siehe **Grafik**). Weiter liegt es in der Natur von Schaltnetzteilen, dass die Störpegel ein sehr breites Spektrum aufweisen. Das macht die Sache nicht einfacher, da sich etliche Elektronik eher von den höheren Spektralanteilen des Rippels auf der Versorgung gestört fühlt als von niederfrequenten.

Um die spektrale Filterwirkung zu verbreitern, werden oft mehrere Kondensatoren mit unterschiedlichen Kapazitäten parallel geschaltet. Im Beispiel wäre also statt eines einzelnen 10-µF-Kondensators noch zusätzlich ein parallel geschaltetes 100-nF-Exemplar mit guten HF-Eigenschaften (beispielsweise eine keramische Low-ESR-Ausführung wie der Typ X5R) sinnvoll. Auf diese Weise werden auch die hochfrequenten Störungen reduziert.

Wenn diese Maßnahme immer noch nicht reicht, kann man immer noch eine kleine Spule mit beispielsweise 0,1 µH zwischen den Ausgang mit 10 µF und den zusätzlichen 100-nF-Kondensator schalten, sodass sich ein LC-Filter ergibt. Gute Praxis ist es auch, analoge ICs möglichst nahe an ihren Stromversorgungs-Pins mit RC-Filtern zu versehen. All diese Maßnahmen sind - gemessen am Gesamtpreis des Geräts - in der Regel zu vernachlässigen und brauchen kaum Platinenfläche. Sie bannen aber Schwierigkeiten, die ansonsten zu recht viel Frust führen können!

(120144)

LABCENTER

Steckiquitäten

Von Thijs Beckers (Elektor Editorial)

Bei der Suche nach Elektronik-Antiquitäten für unsere Rubrik „Retronik“ stolperte mein Kollege Jan Buiting schon über viele merkwürdige Dinge aus der Welt der Unterhaltungselektronik. Hier und da zeigt sich sein Faible für Audio- und TV-Geräte von Philips. Vor kurzem zeigte mir Jan einige Stücke aus seiner Sammlung an Steckern, die er über die Jahre zusammengetragen hat. Da ich zu jung bin um die praktische Anwendung dieser Steckvorrichtungen selbst erlebt zu haben, war ich sehr neugierig zu erfahren, für was diese Stecker früher gut gewesen sind. Nachfolgend eine willkürliche Auswahl. Die meisten Exemplare sind sogar mechanisch verpolungssicher ausgeführt, damit das Gegenstück immer richtig angeschlossen wird. Manchmal kann man nur staunen.

1. Ein Lautsprecher-Stecker für ein Radio aus den 1950er Jahren mit einem zentralen Pin, damit man ihn nicht versehentlich in eine Netz-Steckdose stecken kann. Mehr hierzu unter Punkt 4.
2. Eine schwere, dreipolige Ausführung mit Sicherheits-Schraubdeckel für hohe Ströme und/oder hohe Spannungen. Diese Steckverbindung wurde oft zum Anschluss von Lautsprechern in 100-V-Audio-Systemen verwendet.
3. Ein DIN-nach-5-pol-IEC-Plattenspieler-Adapter von Philips mit fünf versilberten Pins (mit Glasfaserpinsel gereinigt). Sehr selten.
4. Lautsprecherstecker zum Anschluss von Hochtöner-Satelliten an das edle Radio B6X62A – gut geraten: von Philips. Ohne Verpolungsschutz zwar, aber dafür mit einem roten Punkt zur Kennzeichnung der Phase versehen. Die braune Version (1) ist für den Anschluss eines Woofers gedacht. Das Radio hatte sogar zwei eingebaute Aktivfilter mit einer Trennfrequenz von 400 Hz!
5. Kabelversion eines fünfpoligen IEC-Audio-Steckers samt passender Buchse. Wurde nur von Philips verwendet und konnte sich nie als Standard durchsetzen. Zum Öffnen braucht es Spezialwerkzeug.
6. Dreipoliger Stecker für Philips-Mikrofone oder 800-Ω-Lautsprecher samt passender Buchse. Ein typischer Lautsprecher-Anschluss für OTL-Röhrenverstärker (trafoloser Ausgang) der Jahre 1955–1964. Verpolungssicher. Der dritte Pin ist optional und für den Hochtöner (>400 Hz) vorgesehen. Das dynamische Mikrofon ist per geschirmtem Kabel mit seiner Kupplung verbunden.
7. Frühe IEC-Verbindung zur Stromversorgung von Zusatzgeräten. Verpolungssicher aber nicht geerdet.
8. VHF-Antennenstecker für 240-Ω-Flachbandkabel mit Zentralelektrode.
9. Wie 8., aber mit runden Pins ohne Zentralelektrode.
10. VHF/UHF-Antennenstecker für Fernseher. Geeignet für 240-Ω-Flachbandkabel. Die genaue Anwendung ist leider unbekannt.
11. Wie 8. mit Verpolungsschutz. Die genaue Anwendung ist unbekannt.

Dies war nur eine kleine Auswahl aus der schier unendlichen Menge an Steckern, Buchsen und Kupplungen aus dem Vor-Internet-Zeitalter. Scheinbar gibt es immer noch nicht genug unterschiedliche Versionen, denn ständig werden neue entwickelt. Heute sind Steckverbinder für USB, USB 3, Firewire (4- und 6-polig), Thunderbolt, HDMI, DisplayPort, DVI (-A, -D, -I, -D HDCP, mini-, micro-), VGA, S-Video, Cinch, TOSLINK (für S/PDIF), XLR, Klinke (2,5 mm, 3,5 mm und 6,35 mm), Bananenstecker, Sata und RJ-45 besonders wichtig. Und dabei sind Stecker und Dosen für Netzspannung noch gar nicht eingerechnet.

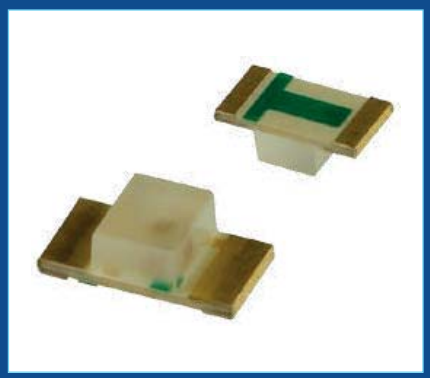
Wann wird diese beschleunigte Evolution enden? Auch bei Elektor weiß das niemand. Doch immerhin kann man herausfinden, womit diese Geschichte begann. Und aus dieser Frage machen wir gleich einen Aufruf bezüglich „*Iunctio Antiquissima*“ und bitten Sie, uns ein gutes Foto ihres ältesten Steckers an connectorcontest@elektor.com zu schicken. Möglicherweise wird Ihre Antiquität dann Thema eines „Retronik“-Artikels oder sie wird in einer der nächsten Ausgaben besonders erwähnt.

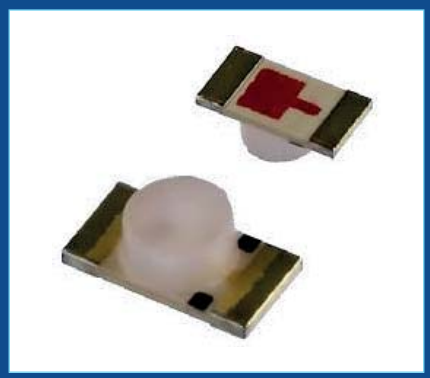
(120027)

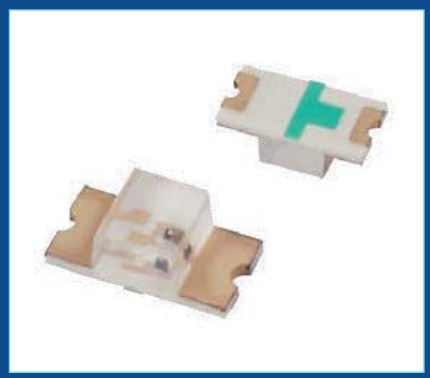
Polarität von SMD-LEDs

Von Raymond Vermeulen (Elektor-Labor)

Beim Löten von gepolten Bauteilen braucht es immer etwas zusätzliche Aufmerksamkeit. Besonders bei SMD-Bauteilen ist nicht immer gleich klar, welcher Pin welcher ist. Das wurde mir neulich wieder bewusst, als ich zwei SMD-LEDs im winzigen 0603-Gehäuse bestücken wollte. Nicht nur dass diese LEDs wirklich sehr klein sind – ich fand auch keinerlei Markierung zur







Kennzeichnung von Anode oder Kathode...
Ich riskierte es, lötete die beiden LEDs einfach ein und hoffte, dass die Sache auf Anhieb funktionieren würde. Doch wie es der Zufall wollte: Nun waren beide LEDs verkehrt herum eingelötet. Enttäuscht lötete ich sie aus und richtig herum wieder ein.

Die Geschichte hätte hier zu Ende sein können, doch ein Nachgeschmack blieb, da mir bei der Schaltungsentwicklung wohl nicht alle Details bekannt waren. Nach einem Blick ins Datenblatt zeigte sich, dass die LEDs wohl doch eine Markierung besaßen, selbst in der 0603-Ausführung. Es wäre ja auch erstaunlich gewesen, wenn nicht. Auf der Rückseite der LEDs findet sich immer eine Kennzeichnung von Anode oder Kathode. Doch da es hierzu keine Standardisierung gibt, kocht jeder Hersteller sein eigenes Süppchen und verwendet Symbole, die ihm sinnvoll scheinen. Die Illustration zeigt einige erfolgreich „decodierte" Exemplare, doch ist diese Sammlung bei weitem nicht vollständig. Es existiert eine breite Palette von Streifen, Punkten, Rechtecken und Pfeilen - sogar Kombinationen davon kommen vor.

Um sicher zu gehen muss man also das passende Datenblatt studieren oder schlicht die Polarität messen – das war die Erkenntnis, die ich aus dieser Pleite gewann, und die ich hiermit gerne an Sie weitergeben möchte, damit Sie es in Zukunft einfacher haben.

(120145)

Ein Kabel für den Bus

Von Jens Nickel

Um unser ElektorBus-Projekt weiterzuentwickeln, haben wir bisher nur kleinere Systeme aufgebaut, die bequem auf einem Schreibtisch Platz hatten. Doch nun soll es ernst werden: In der nächsten Ausgabe wollen wir eine „Installationsplatine" vorstellen, die das Schalten von 230-V-Verbrauchern erlaubt; danach geht es an den Aufbau eines Pilot-Busses in unserer Verlagszentrale. In unserem Elektor-Castle (siehe Bus-Logo) können schon einige zig Meter zusammenkommen, wenn wir zwei Räume oder gar Stockwerke verbinden wollen. Höchste Zeit also, sich Gedanken um ein geeignetes Kabel zu machen.
Für einen kleinen Test haben wir zwei Experimental-Knoten [1] und unsere AndroPod-Platine [2] verkabelt, die zusammen mit einem Smartphone einen dritten Busteilnehmer darstellt. An einen der Knoten war ein Fotowiderstand angeschlossen, die Werte wurden kontinuierlich über den Bus gesendet und auf dem Handy angezeigt.
Los ging es mit einem Kabel, das wir in unseren Labor-Beständen entdeckten (KROSCHU Schaltflex CY Style 2571 [3]). Es verfügt über 10 Leiter, davon ist ein Paar verdrillt (Twisted Pair). Wir machten zwei Versuche: Einmal verwendeten wir für unsere vier Busleitungen (Daten A und B, 12 V, GND) jeweils einen einzelnen Leiter; einmal nutzten wir für A und B das verdrillte Leitungs-Paar, so wie für RS485 empfohlen. Das schlichte Ergebnis: Bis zu Kabellängen von 36 m (mehr hatten wir nicht zur Verfügung) konnten wir in beiden Fällen keine Störungen entdecken, die Messwerte flitzten kontinuierlich über den Bus. Auch die Stromversorgung schien über solche Strecken robust zu funktionieren. Jedenfalls konnte am zweiten Busknoten ein Relais problemlos geschaltet werden, das ebenfalls über den Bus gespeist wurde. Mein Kollege Raymond Vermeulen kontrollierte schließlich noch die Flanken des Signals auf dem Oszilloskop: Perfekt!
Dann kam ein CAT5E-Kabel an die Reihe (4 x Twisted Pair, geschirmt). Vorschriftsmäßig haben wir eines der verdrillten Paare für A und B genutzt und für 12 V und GND jeweils ein anderes Paar verwendet (beide Leiter zusammengeschaltet). Leitungslängen von rund 30 m waren auch hier kein Problem. Ein kleines Video dieses Experiments [4] kann man auf dem Elektor-YouTube-Kanal ansehen!

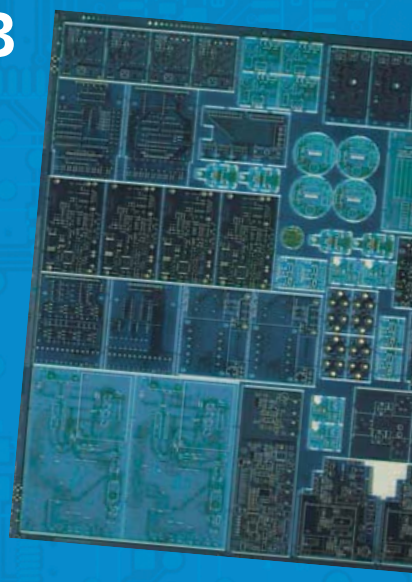
(120198)

[1] www.elektor.de/110258
[2] www.elektor.de/110405
[3] www.kroschu-cable.de/documents/downloads/Schaltflex%202008-07-08%20%5Be%5D.pdf
[4] www.youtube.com/watch?v=rbDSTXNARmw

elektor PCB Service

präsentiert von Eurocircuits

Die europäische Referenz für PCB Prototypen und Kleinserien



Sparen Sie Entwicklungszeit und -kosten mit unseren einfach zu nutzenden Leiterplatten-Pooling-Services.
Wir bieten Ihnen:

- Top Qualität zu niedrigen Pooling Preisen
- Schnelle Lieferung nach Ihren Bedürfnissen
- Umfangreiche Technologie-Unterstützung
- Keine Einmal- oder Werkzeugkosten
- Keine Mindestbestellwerte - ab der 1 Leiterplatte
- Online-Bestellung ohne Vorauskasse
- Schablonen Service

PCB proto – *spezieller Prototypen-Service für Entwickler, preiswert und schnell*

- 1 oder 2 LP in 2, 3, 5 oder 7 Arbeitstagen
- DRC-geprüft, professionelle Ausführung inkl. 2x Lötstopplack und 1x Bestückungsdruck, 150µm Technologie
- 1 x 100 x 80mm in 7AT - 2 Lagen 46.26 € - 4 Lagen 93.94 €
- 2 x 100 x 80mm in 7AT - 2 Lagen 36.28 € je LP - 4 Lagen 73.52 € je LP

Preise inkl. 19% MwSt und ohne Transportkosten

STANDARD pool – *die größte Auswahl an Eurocircuits Pooling Optionen*

- 1-8 Lagen 150µm Technologie-Leiterplatten
- ab 2 AT

TECH pool – *100µm-Technologie mit allen Pooling-Vorteilen*

- 2-8 Lagen 100µm Technologie-Leiterplatten
- ab 4 AT

IMS pool – *Aluminiumkern-Leiterplatten für hohe Wärmeableitung (z.B. LED-Anwendung)*

- Leiterplatten mit einlagig isoliertem Metallsubstrat
- 1.5mm Alukern mit 75µm thermisch leitfähigem Dielektrikum und 35µm Kupferfolie
- ab 3 AT

On demand – *Alle Optionen im Nicht-Pooling für Spezialanwendungen*

- 1-16 Lagen bis 90µm-Technologie
- RF- und Hoch-Tg-Materialien
- ab 2 AT

www.elektorPCBservice.com

Platino mit Arduino

Anleitung zur Integration eigener Hardware

Platino, das besonders vielseitige AVR-Board in der Version vom Oktober 2011, ist weitgehend mit der Arduino-Programmier-Umgebung kompatibel. Doch hie und da lauern noch Schwierigkeiten, denn Arduino unterstützt nicht alle bei Platino möglichen Mikrocontroller – etwas Code-Portierung kann schon nötig sein. Glücklicherweise stehen die dazu erforderlichen Werkzeuge im Internet kostenlos zur Verfügung. Dieser Artikel beschreibt, wie das funktioniert.

Von Clemens Valens (Elektor-Redaktion)

Bekannterweise fußen Arduino Uno und seine Nachfolger auf der 28-poligen Controller-Familie ATmegaXX8 von Atmel. Der größere Nachfolger Arduino Mega basiert auf dem ATmega2560 im 100-poligen SMD-Gehäuse. Platino aber unterstützt nicht nur 28-polige AVR-Controller, sondern auch die mit 40 Pins, welche Arduino nicht kennt. Gerade wenn man eine eigene Platine entwickelt, liegt es oft nahe, AVR-Controller in SMD-Gehäuse mit abweichender Pin-Anzahl einzusetzen, die Arduino von Hause aus nicht unterstützt. In diesem Beitrag geht es darum, wie man die Arduino-Umgebung so modifiziert, dass sie mit eigener Hardware harmoniert.

Eigene Hardware & Arduino 1.0

Die Integration kompatibler Hardware in die Arduino-Programmier-Umgebung ist ziemlich einfach. Man muss lediglich einen zusätzlichen Ordner im Hardware-Verzeichnis der Arduino-Distribution (Beispiel: arduino-1.0\Hardware\) erstellen. Das neue Verzeichnis – eine aussagekräftige Bezeichnung wie Platino empfiehlt sich – enthält die Dateien, die spezifisch für die verwendete Hardware sind. Gelegentlich sind noch so etwas wie Links auf Standard-Dateien des Arduino-Kerns erforderlich.

Um Probleme mit zukünftigen Versionen der Arduino IDE zu vermeiden, empfiehlt sich eine Kopie der Dateien der Arduino-Hardware (arduino-1.0\Hardware\arduino) in den neuen Ordner (arduino-1.0\Hardware\platino), sodass man lediglich die Kopien modifiziert. Der Nachteil dieser Methode ist, dass nicht automatisch die Verbesserungen und Updates der Core-Dateien einfließen. Dafür braucht man bei Updates von Core-Dateien auch nicht immer wieder etwas zu ändern.

Man muss ein Verzeichnis mit vier Unterverzeichnissen und zwei Dateien kopieren:

- `bootloaders` (Ordner)
- `cores` (Ordner)
- `firmwares` (Ordner)
- `variants` (Ordner)
- `boards.txt` (Datei)
- `programmers.txt` (Datei)

Im Verzeichnis bootloaders stecken logischerweise Bootloader für die eigene Hardware. Bei einem Controller, der von den Standard-Bootloadern nicht unterstützt wird, muss man einen davon modifizieren. Das hört sich schlimmer an als es ist, solange es sich um einen AVR-Controller handelt. Mehr hierzu später.

Das Verzeichnis cores enthält die Core-Dateien von Arduino. Damit werden die Arduino-Funktionen bestimmt, wovon einige an die eigene Hardware angepasst werden. Im Embedded-Jargon würde man hier von einem BSP (**B**oard **S**upport **P**ackage) sprechen.

Im Verzeichnis firmwares stecken die ausführbaren Dateien, die Boards mit einem ATmega8U2 für die Kommunikation mit der IDE nutzen. Wenn die eigene Hardware (so wie Platino) keinen solchen Controller beinhaltet, kann man den Ordner löschen.

Das Verzeichnis `variants` ist neu in Arduino 1.0. Es ermöglicht auf einfache Weise die Definition von Board-Varianten mit einer großen Anwenderbasis. Bezüglich Platino können hier die Varianten mit 28- und 40-poligem Controller definiert werden. Für jede Variante gibt es ein eigenes Unterverzeichnis mit der Datei pins_arduino.h, die in früheren Arduino-Versionen im Ordner cores abgelegt war. Bei Bedarf können hier auch andere Varianten-spezifische Dateien platziert werden.

Die Datei programmers.txt sollte gelöscht werden, denn man benötigt sie nur dann, wenn man einen eigenen Programmer definieren will. Wenn man sie ohne Änderung behält, gibt es doppelte Einträge

im Menü Tools->Programmer.
Dank der Datei boards.txt wird die IDE über die richtigen Ordner und Protokolle für die verwendete Hardware informiert. Die in dieser Datei aufgeführten Boards zeigen sich auch im IDE-Menü Tools->Board, mit dem man sie auswählen kann. Diese Datei wird daher zuerst modifiziert, bevor es an die Core-Dateien geht.

boards.txt

Hier finden sich die Board-Definitionen für die IDE. Sie sorgen dafür, dass die eigene Hardware von der IDE erkannt wird. Zur Anpassung genügt ein einfacher Text-Editor wie der Windows-Editor. Die Datei enthält Blöcke zusammengehörender Zeilen wie den von **Listing 1** (ohne Zeilennummern).
Man löscht alle Blöcke bis auf den, welchen man für die eigene Hardware anpassen will. Ohne das Löschen der anderen Blöcke ergeben sich doppelte Einträge im Menü Tools->Boards.
Alle Zeilen des Blocks in Listing 1 fangen mit uno an, was als Kennzeichen des Boards in der Datei boards.txt und allen anderen Dateien der Arduino-Installation fungiert. Die IDE erkennt nur Boards mit einer identischen Kennzeichnung.
Die Reihenfolge der Zeilen ist unwichtig.
Zeile 1 enthält das Label name, das den in der IDE angezeigten Namen des Boards definiert. Man kann den Namen recht lang machen. Ich weiß nicht, wie viele Zeichen akzeptiert werden, doch gibt es sicherlich eine Grenze.
Die Zeilen 2 & 4 upload.protocol und upload.speed sind an AVRDUDE übergebene Parameter. AVRDUDE ist der von der IDE verwendete Standard-Programmer. Arduino 1.0 enthält die neueste Version von AVRDUDE, die mit dem Protokoll arduino umgehen kann. Mit älteren Arduino-Distributionen mitgelieferte Versionen können das nicht. Hier nutzt man besser das Protokoll stk500.
Zeile 3 definiert den maximalen von der Anwender-Applikation belegbaren Speicher. Er wird bestimmt, indem man vom maximalen Speicher den Bedarf des Bootloaders abzieht. Bei einem Controller mit 32 KB und einem 512 Byte langen Bootloader sind 32.256 Byte verfügbar.
Die Zeilen 5 bis 11 definieren den von der IDE verwendeten Bootloader (Tools->Burn Bootloader) und wie er geladen wird. Auch hier enthalten die Zeilen Parameter für AVRDUDE. Die Fuse- und Lock-Einstellungen hängen vom Controller ab. Der in Zeile 9 angegebene Bootloader sollte im in Zeile 8 angegebenen Unterverzeichnis des Ordners bootloaders stecken. Solange man nicht versucht, den Bootloader von der IDE aus zu brennen, können diese Parameter fiktional sein. Diese Zeilen erlauben auch die Spezifikation eines Bootloaders, der nicht mit dem Arduino-Kommunikationsprotokoll kompatibel ist, solange AVRDUDE damit umgehen kann.
Zeile 12 spezifiziert den verwendeten Controller. Gerade bei AVR-P-Typen gibt es unterschiedliche Varianten. Dies ist deshalb wichtig, weil sich deren ID-Bytes von Nicht-P-Typen unterscheiden. Das Suffix „A“ macht hingegen keinen Unterschied.
Zeile 13 definiert die Taktfrequenz des Controllers in Hz. Dieser Wert sollte mit der Frequenz des Quarzes oder des internen Taktgebers übereinstimmen. Die Frequenzangabe ist wichtig für die Timer und den UART. Wen interessiert wo solche Angaben stecken, kann die

Listing 1. Diese Struktur definiert ein Board für die IDE von Arduino 1.0 in der Datei „boards.txt“. Die Zeilennummern sind lediglich zu Referenzzwecken hinzugefügt.

```
 1  uno.name=Arduino Uno
 2  uno.upload.protocol=arduino
 3  uno.upload.maximum_size=32256
 4  uno.upload.speed=115200
 5  uno.bootloader.low_fuses=0xff
 6  uno.bootloader.high_fuses=0xde
 7  uno.bootloader.extended_fuses=0x05
 8  uno.bootloader.path=optiboot
 9  uno.bootloader.file=optiboot_atmega328.hex
10  uno.bootloader.unlock_bits=0x3F
11  uno.bootloader.lock_bits=0x0F
12  uno.build.mcu=atmega328p
13  uno.build.f_cpu=16000000L
14  uno.build.core=arduino
15  uno.build.variant=standard
```

Core-Dateien nach dem Label F_CPU durchsuchen.
Apropos Core-Dateien: sie sollten in dem Unterverzeichnis des Ordners Hardware stecken, das in Zeile 14 definiert ist (hier arduino-1.0\Hardware\arduino). Zeile 15 spezifiziert das Unterverzeichnis des Ordners variants, das die Datei pins_arduino.h enthält. Sie enthält Angaben zum eigenen Board.
Die letzten beiden Zeilen können auf beliebige Ziele verweisen. Man kann damit auf Core-Dateien anderer Boards verweisen, mit denen man arbeitet. Es handelt sich dabei um die am Anfang erwähnten Links.
Listing 2 zeigt, wie ich Listing 1 an mein spezielles Platino-Board mit einem ATmega164p angepasst habe. Das Gleiche habe ich mit allen vorstellbaren Ausführungen gemacht, sodass insgesamt 16 neue Boards in der IDE auftauchen. Ich habe eine Variante ATmegaXX4 für 40-polige Chips und auch eine Variante ATmegaXX8 für 28-polige Chips erstellt.

Listing 2. Die Board-Struktur für Platino mit einem ATmega164p. Die letzten beiden Zeilen verlinken zu den richtigen Core-Dateien für dieses Board.

```
platino164p.name=Platino 164p(a) @ 16 MHz
platino164p.upload.protocol=arduino
platino164p.upload.maximum_size=15872
platino164p.upload.speed=115200
platino164p.bootloader.low_fuses=0xff
platino164p.bootloader.high_fuses=0xdc
platino164p.bootloader.extended_fuses=0xfd
platino164p.bootloader.path=optiboot
platino164p.bootloader.file=optiboot_
     platino164p.hex
platino164p.bootloader.unlock_bits=0x3f
platino164p.bootloader.lock_bits=0x0f
platino164p.build.mcu=atmega164p
platino164p.build.f_cpu=16000000L
platino164p.build.core=platino
platino164p.build.variant=ATmegaXX4
```

```
int analogRead(uint8_t pin)
{
    uint8_t low, high;

#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
    if (pin >= 54) pin -= 54; // allow for channel or pin numbers
#elif defined(__AVR_ATmega32U4__)
    if (pin >= 18) pin -= 18; // allow for channel or pin numbers
#elif defined(ATMEGA_X4) || defined(__AVR_ATmega16__) || defined(__AVR_ATmega32__)
    if (pin >= 24) pin -= 24; // allow for channel or pin numbers
#else
    if (pin >= 14) pin -= 14; // allow for channel or pin numbers
#endif

#if defined(__AVR_ATmega32U4__)
    pin = analogPinToChannel(pin);
    ADCSRB = (ADCSRB & ~(1 << MUX5)) | (((pin >> 3) & 0x01) << MUX5);
#elif defined(ADCSRB) && defined(MUX5)
    // the MUX5 bit of ADCSRB selects whether we're reading from channels
    // 0 to 7 (MUX5 low) or 8 to 15 (MUX5 high).
    ADCSRB = (ADCSRB & ~(1 << MUX5)) | (((pin >> 3) & 0x01) << MUX5);
#endif

    // set the analog reference (high two bits of ADMUX) and select the
    // channel (low 4 bits).  this also sets ADLAR (left-adjust result)
    // to 0 (the default).
#if defined(ADMUX)
    ADMUX = (analog_reference << 6) | (pin & 0x07);
#endif
```

Bild 1. Ein Teil der Datei „wiring_analog.c“ mit Platino-Code-Modifikation.

pins_arduino.h

Wenn das eigene Board sauber definiert ist, kann man mit der Anpassung der Pin-Definitionen in der Datei pins_arduino.h im Ordner variants beginnen (für ATmega164p: arduino-1.0\Hardware\platino\variants\ATmegaXX4).
In dieser Datei wird die Anzahl an Pins angegeben, die als GPIO, als analoge Eingänge, als PWM-Ausgänge, als Timer-Pins, als SPI-Port etc. fungieren können. Alle diese Definitionen sind Makros und Datenstrukturen. Man braucht das passende Datenblatt, um alles fehlerfrei zu erledigen.
Die Reihenfolge in den Datenstrukturen wird durch die Nummern der digitalen Pins bei Arduino bestimmt. Der digitale Pin 0 ist der erste Eintrag jeder Datenstruktur. Man muss die gleiche Reihenfolge in allen Strukturen verwenden, sonst werden Pins und Strukturen durcheinander gebracht. Analoge Eingänge werden anders behandelt und sind nicht in dieser Datei enthalten.
Auch wenn es sich hier nicht um eine geheimes Alien-Know-how handelt, sollte man bei Datei-Änderungen wissen, was man tut.
Meine Datei enthält #ifdef-Direktiven bezüglich der Controller ATmega16 und ATmega32. Ich hätte auch eine weitere Variante für diese 40-poligen Chips anlegen können, doch sind die Unterschiede zu gering, um dies zu rechtfertigen. Zum Beispiel resultiert die Controller-Definition __AVR_ATmega16__ (und Ableitungen davon) aus der Definition von Zeile 12 aus Listing 1.

Core-Dateien ändern

Der Ordner cores (arduino-1.0\Hardware\arduino\cores\) enthält das Unterverzeichnis arduino, in dem die Arduino-Core-Dateien abgelegt sind - aktuell 36 Dateien. Das sieht nach viel Arbeit aus, aber nur ein Teil davon muss geändert werden. Bei meinem Platino-Board wurden nur vier Dateien modifiziert (von meiner lokalen Kopie von cores). Hier meine Core-Änderungen:

Arduino.h
Diese Datei wird von vielen anderen Core- und Library-Dateien inkludiert, weshalb dies ein guter Ort für Links zum eigenen Code ist. Meinen Code für die Platino-Funktionen habe ich in C++ geschrieben, weshalb die Anweisung #include „Platino.h“ am Ende der Datei direkt hinter der Anweisung #ifdef __cplusplus steht.
Außerdem musste ich noch eine Definition für die internen Referenzspannungen INTERNAL1V1 und INTERNAL2V56 der 40-poligen Chips anlegen. Dies wurde mit der Anweisung #elif defined(ATMEGA_X4) erledigt. Das Makro ATMEGA_X4 ist in der von mir erstellten Header-Datei devices.h enthalten. Hierfür wurde eine Include-Anweisung am Anfang der Datei eingefügt.

Tone.cpp
Es ist nicht wirklich notwendig, diese Datei zu ändern, denn sie ist etwas seltsam. Sie enthält sogar einen Kommentar, der das bestätigt. Doch für zukünftige Kompatibilität ist es wohl besser, sie nicht zu ignorieren. Ich fügte die Anweisung #elif defined(ATMEGA_X4) hinzu und kopierte den gleichen Code wie für die anderen Chips hinein.

Wiring_analog.c
Die Funktion analogRead muss modifiziert werden, sodass man Nummern digitaler Pins auch analogen Eingängen zuweisen kann. Es handelt sich nur um eine Code-Zeile, die mit #elif defined(ATMEGA_X4) eingebunden wird (**Bild 1**).

Wiring_private.h
Diese Datei enthält lediglich Interrupt-Definitionen, und da 40-polige AVR-Controller drei externe Interrupt-Quellen kennen, muss hier eine Definition hinzugefügt werden. Bei der Anweisung #elif defined(ATMEGA_X4) kann man sehen, wie ich das gelöst habe.

Falls Sie sich fragen, warum ich nur für die 40-polige Variante Compiler-Direktiven verwendet habe: Die 28-poligen Controller werden ja schon von Arduino unterstützt und Platino ist damit kompatibel, sodass es nichts zu ändern gibt.

Hinzufügen von Core-Dateien

Wenn Ihre Hardware nur Peripherie unterstützt, die auch bei einem Standard-Arduino-Board vorhanden sind, können Sie diesen Abschnitt überspringen. Platino integriert allerdings ein LCD, Taster und Drehgeber sowie einen Buzzer, eine RGB-LED und Konfigurations-Jumper. Diese zusätzliche Peripherie kann natürlich von den Libraries (arduino-1.0\libraries) behandelt werden — die Arduino-LiquidCrystal-LCD-Library funktioniert zum Beispiel auch gut mit Platino — aber das impliziert komplexere Modifikationen an mehreren Stellen, die berücksichtigt werden müssen. Von daher entschied ich mich für die Ausführung der Platino-Funktionen als integraler Bestandteil der Arduino-Umgebung und integrierte meinen eigenen Code in eine Core-Kopie.

Ein Objekt für sämtliche Platino-Peripherie

Zusätzliche Core-Dateien sind kein Problem, da der Compiler einfach alles kompiliert, was er im Core-Verzeichnis findet. Nachfolgend alphabetisch sortiert was hinzukam und warum:

devices.h
Einige Definitionen braucht es zur bedingten Kompilierung von Core-Dateien für Platino. Da Platino mehrere unterschiedliche AVR-Controller unterstützt, wären #ifdef-Anweisungen recht lang. Ich entschied mich daher für zwei Gruppen von 40-poligen (ATMEGA_X4) und 28-poligen (ATMEGA_X8) Controllern. Hier kann man neue Chips hinzufügen.

LiquidCrystal cpp & h
Wie schon erwähnt harmoniert Platino mit der Arduino-LCD-Library, doch für die Benutzung braucht es noch Änderungen (siehe nachfolgend). Zur einfacheren Kompilierung kopierte ich diese Library in meinen Core-Ordner.
Da diese Library über keinen Default-Constructor verfügt, fügte ich die Zeile LiquidCrystal(void) {} zur Klassendefinition in der Header-Datei hinzu. Falls Sie nicht so firm in C++ sind: Meine Platino-Klasse definiert ein LiquidCrystal-Objekt, ohne es zu initialisieren. Die dazu notwendige Funktion fehlte in der Library.

Platino.cpp & h
Zwecks einfachem Zugriff auf Platino-Peripherie erstellte ich ein Objekt in der Form einer Klasse. Der Aufruf der Klasse erlaubt den direkten Zugriff auf Platino-Peripherie. Tatsächlich handelt es sich mehr um einen Wrapper für die individuellen Peripherie-Klassen, doch so werden auch die Löt-Jumper auf einem Platino-Board zwecks Konfiguration berücksichtigt. Hierzu später mehr.

PushButton cpp & h
Diese beiden Dateien implementieren eine Klasse für ein Taster-Objekt. Sie unterstützt entprellten und direkten Zugriff auf Taster und wird auch von der Klasse für Dreh-Encoder genutzt (siehe unten).

RotaryEncoder cpp & h
Dreh-Encoder sind tatsächlich zwei (oder drei bei Varianten mit Druck-Funktion) mechanisch gekoppelte Taster. Die zuständige Klasse behandelt sie auch so. Sie nutzt keine Interrupts und keine Timer, weshalb sie der Anwender regelmäßig abfragen muss. Man ruft sie in der Loop-Funktion eines Sketches auf, muss aber potentielle Blockierungen oder Warteschleifen anderer Libraries im Auge behalten.
Der optionale Drucktaster von Dreh-Encodern wird von dieser Klasse nicht behandelt. Er sollte als normaler Taster via Taster-Klasse abgefragt werden.

Anwendung

Nachdem alle Änderungen gemacht sind, kann man erste Tests starten. Zuerst wählt man sein Board mit dem IDE-Menü Tools->Boards. Wenn die Modifikation der Datei boards.txt stimmt und man einen der IDE bekannten Programmer verwendet, sollte man den Boot-

Bild 2. Platino innerhalb von Arduino. Die rechte untere Ecke zeigt die Board-Auswahl. Das Platino-Object wird am Sketch-Anfang (in der Mitte des Screenshots) aufgerufen.

loader direkt aus der IDE (Tools->Burn Bootloader) in den Controller übertragen können. Mit programmiertem Bootloader kann man mit einem Sketch weitermachen.
Es kann unter Umständen Probleme geben, einen geeigneten Bootloader zu finden. Meine Experimente führten zu einer fast universellen Lösung, doch auch hier kann es hier und da an Stellen klemmen, die sich im Rahmen dieses Beitrags nicht klären lassen. Im kostenlosen Download von der Webseite zu diesem Artikel ist ein Bootloader enthalten, der für (fast) jeden AVR-Controller taugt, den Platino unterstützt. Dennoch gilt: Anwendung auf eigene Gefahr! Es kann zwar nichts kaputt gehen, aber es kann auch prima nicht funktionieren. Bei mir funktioniert ein ATmega164p sehr gut mit diesem Bootloader.
Es empfiehlt sich, einen ersten Sketch zu schreiben, mit dem man die Pins auf korrekte Funktion überprüft. Wichtig ist, dass die Pins reagieren, die man adressiert. Die analogen Eingänge und die SoftwareSerial-Library sollten nicht übergangen werden. Es sollten auch die Interrupts getestet werden. Wenn alles soweit klappt, kann man recht sicher sein, dass man eine gute Portierungsarbeit geleistet hat. Jetzt kann es losgehen.

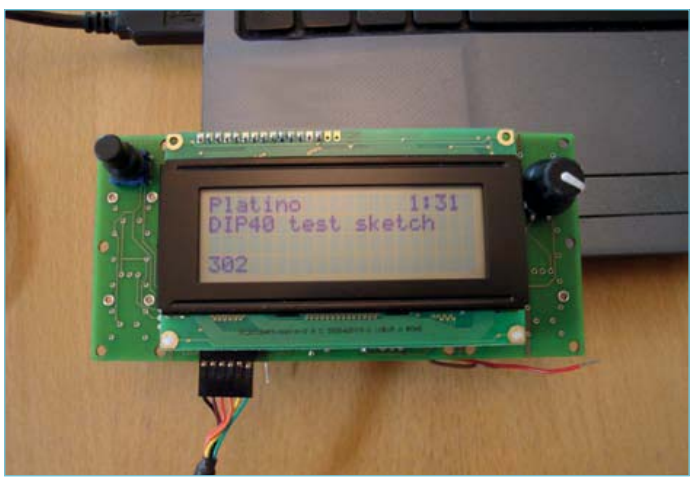


Bild 3. Test-Sketch von Bild 2, auf Platino laufend. Der Wert unten links (302) entspricht dem Status des Dreh-Encoders rechts. Der Timer-Wert oben rechts kann durch Betätigung des Tasters links zurückgesetzt werden.

Anwendung der Platino-Klasse

Schon der Aufruf der Platino-Klasse belegt rund 2,5 KB des Programmspeichers. Diese Investition zahlt sich durch Programmier-Komfort aus. Zum Aufruf schreibt man am Anfang eines Sketches (**Bild 2**):

```
Platino platino(4,20,'B','C','B','B','B','B','B');
```

Das bedeutet hier, dass ein LCD mit vier Zeilen à 20 Zeichen verwendet wird. Außerdem befinden sich die Löt-Jumper 1, 3...7 & 14 in den Positionen Port B oder C ('B' oder 'C'; ein offener Jumper wird durch '0' indiziert). Der Zugriff auf die Platino-Peripherie geht so:

```
platino.led(state); // state can be 1 or 0.
if (platino.pushButton(1)==0) … // check
     pushbutton 1 (active low)
if (platino.encoder2.tick(counter)!=0) … //
     call often, 1 kHz if possible.
platino.beep(frequency,duration_in_ms);
platino.lcd.setCursor(0,1); // goto column 0 of
     row 1.
platino.lcd.print("the quick brown fox");
```

Elegant, nicht wahr?
Zeile 3 benötigt Erläuterungen: Hier wird die Tick-Funktion von Dreh-Encoder-Zähler 2 aufgerufen, wodurch der counter aktualisiert wird. Wenn das Resultat der Funktion von null abweicht, dann indiziert das einen neuen Wert für counter. Wie schon erwähnt muss man diese Funktion häufig aufrufen, wenn man keine Impulse übersehen will.
Natürlich sind alle öffentlichen Funktionen von LiquidLcd und anderen Klassen verfügbar, solange ein platino vorangestellt wird.

Im Download zu diesem Artikel (www.elektor.de/120094) befinden sich auch einige Test-Sketche, welche die Verwendung der Platino-Klasse demonstrieren.
Viel Spaß damit!

(120094)

Coming Soon!

The RL78 Green Energy Challenge

$20,000 in cash prizes

Competition starts March 26, 2012

With an incredible ecosystem of hardware, software and third-party vendors, Renesas' family of RL78 MCUs are optimized for efficient power consumption and deliver up to 41 DMIPS at 32MHz. These versatile MCUs offer a true low-power platform for the most demanding 8- and 16-bit embedded applications.

Renesas, along with *Circuit Cellar* and *Elektor*, invites you to experience true low power by developing a green energy application using the RL78 MCU and IAR tool chain. **Succeed and win a share of $20,000 in cash prizes!**

RENESAS RL78

In association with *Elektor* and *Circuit Cellar*

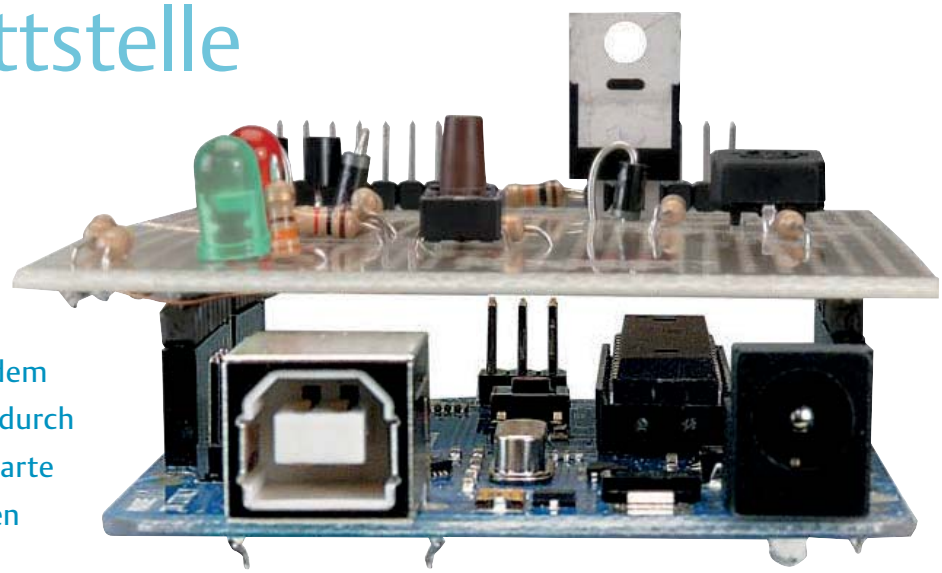
Be the green you see in the world and get ready for the RL78 Green Energy Challenge.

www.circuitcellar.com/RenesasRL78Challenge

Regeln mit Arduino und PC

Universeller µC-gesteuerter Regler mit PC-Schnittstelle

Von Jos van Kempen (NL)



In technischen Systemen sind Regelungen allgegenwärtig. Realisieren lassen sie sich mit Mikrocontroller-Systemen wie dem bekannten Arduino-Board. Ergänzt durch eine unkomplizierte Schnittstellenkarte entsteht ein System, das sich mit den unterschiedlichsten Sensoren und Aktoren kombinieren lässt. Die hier beschriebene Regelung arbeitet autonom, sie kann aber auch über einen PC gesteuert werden.

Eine Reglung ist ein System, das einen Istwert mit einem Sollwert vergleicht. Abweichungen gleicht das System aus, indem es den Istwert nachsteuert. Im Kühlschrank und in der Heizungsanlage werden Temperaturen geregelt, im Auto wird die Bordspannung konstant gehalten und in der Toilettenspülung sorgt ein Regelungsmechanismus dafür, dass der Wasserbehälter nicht überläuft.

Ein typisches Beispiel für eine Temperaturregelung ist die Heizungsanlage. Einfache Regelungen schalten den Brenner ein, wenn die Raumtemperatur unter der eingestellten Temperatur liegt. Der Brenner wird abgeschaltet, wenn die Raumtemperatur die eingestellte Temperatur übersteigt. Die Schwankungen, die bei solchen Zweipunktregelungen auftreten, sind relativ hoch, die Regelung reagiert vergleichsweise träge. Mit einem so genannten PID-Regler und einer steuerbaren Brennstoffzufuhr lassen sich präzisere Regelungssysteme realisieren. Noch höhere Anforderungen an die Reaktionsgeschwindigkeit und die Konstanz des Istwerts erfüllen Regelungssysteme, die komplexer aufgebaut sind und nach anderen Methoden arbeiten.

Die Steuereinheit einer Regelung muss Eingangssignale schnell verarbeiten und flexibel in Ausgangssignale umsetzen. Von universell einsetzbaren Regelungen wird gefordert, dass sie mit möglichst vielen Sensoren und Aktoren kompatibel sind. Solche Systeme müssen fähig sein, unterschiedliche physikalische Größen (beispielsweise °C, hPa oder U/min) zu verarbeiten. Ferner stehen eine einfache Bedienung trotz komplexer Funktionen und nicht zuletzt ein wirtschaftliches Kosten-Nutzen-Verhältnis auf der Wunschliste. Es liegt nahe, die zentralen Aufgaben einer Regelung einem Mikrocontroller zu übertragen.

Das hier beschriebene Regelungssystem arbeitet mit dem Mikrocontroller-Board „Arduino“, auf das eine Schnittstellenkarte, dort *Shield* genannt, aufgesteckt wird. Dieses Duo ist mit einer Vielzahl unterschiedlicher Sensoren und Aktoren kombinierbar. Auf der Schnittstellenkarte können Einstellungen vorgenommen werden, eine zweifarbige LED zeigt den Betriebszustand an. Der Regler arbeitet autonom (*stand alone*), zum Zweck der Kontrolle, zum Einstellen von Parametern und für Datenanalysen ist er auch über einen USB-Anschluss mit einem PC koppelbar. Daten gibt das Regelungssystem im Textformat oder als CSV-Datei für die Verarbeitung mit Excel aus. Das Ändern von Parametern ist auch im laufenden Betrieb möglich.

Input

Die Eingangsgrößen können mit Sensoren unterschiedlicher Art gemessen werden. Analoge Sensoren geben häufig variable Spannungen oder Ströme ab (beispielsweise 0...5 V, 0...10 V, 0...20 mA, 4...20 mA), oder ihre elektrischen Widerstände ändern sich. Der Regler stellt drei analoge Eingänge des Mikrocontrollers mit dem Spannungsbereich 0...5 V zur Verfügung (siehe **Bild 1**). Höhere Sensorspannungen können unkompliziert über Spannungsteiler an diesen Bereich angepasst werden. Auf der Schnittstellenkarte sind Lötstützpunkte für solche Anpassungen vorhanden.

Die drei analogen Eingänge haben folgende Funktionen: Eingang AIn2 (Pin 4 auf Steckleiste K2) ist für die Eingangsgröße reserviert, über Pin 6 auf K2 wird dem Mikrocontroller die Spannung eines Potentiometers zugeführt, das auf der Platine von Hand bedienbar ist, und an Eingang AIn1 (Pin 5 auf K2) kann ein externes Signal gelegt werden, das den Sollwert repräsentiert.

Für digitale Eindraht-Sensoren (*One Wire*)

sind zwei Eingänge vorhanden, dies sind die Pins 4 und 8 auf Steckleiste K5. Ferner gibt es einen Eingang für ein pulsförmiges Signal (Pin 1 und 3 auf K3), an den beispielsweise ein Dreh-Encoder angeschlossen werden kann. Die Impulse lösen Interrupts des Mikrocontrollers aus, sie werden bis etwa 25 kHz erkannt.

Output

Über zwei als Ausgänge konfigurierte Leitungen steuert der Mikrocontroller eine Zweifarben-LED, über drei weitere Leitungen gibt er pulsbreitenmodulierte Rechtecksignale aus. Zwei Rechtecksignale steuern bipolare Transistoren (T2 und T3), das dritte Rechtecksignal steuert einen Power-FET (T1). Die bipolaren Transistoren können Ströme bis insgesamt etwa 400 mA bei 5 V liefern. Der Power-FET kann (theoretisch) Ströme bis 65 A eines externen Stromkreises schalten, bei 12 V und 50 W Schaltleistung ist noch keine Kühlung erforderlich. Wichtig ist hier, dass die Spannung im ausgeschalteten Zustand nicht hochläuft, stark variierende Spannungen dürfen nicht geschaltet werden. Kleinverbraucher bis 45 V/100 mA schaltet Kollektorausgang +Trans (Pin 5 auf K4). Der Kollektorausgang +Dir (Pin 6 auf K4) kann genutzt werden, um beispielsweise die Drehrichtung eines Motors oder die Betriebsart eines Peltier-Elements umzuschalten.

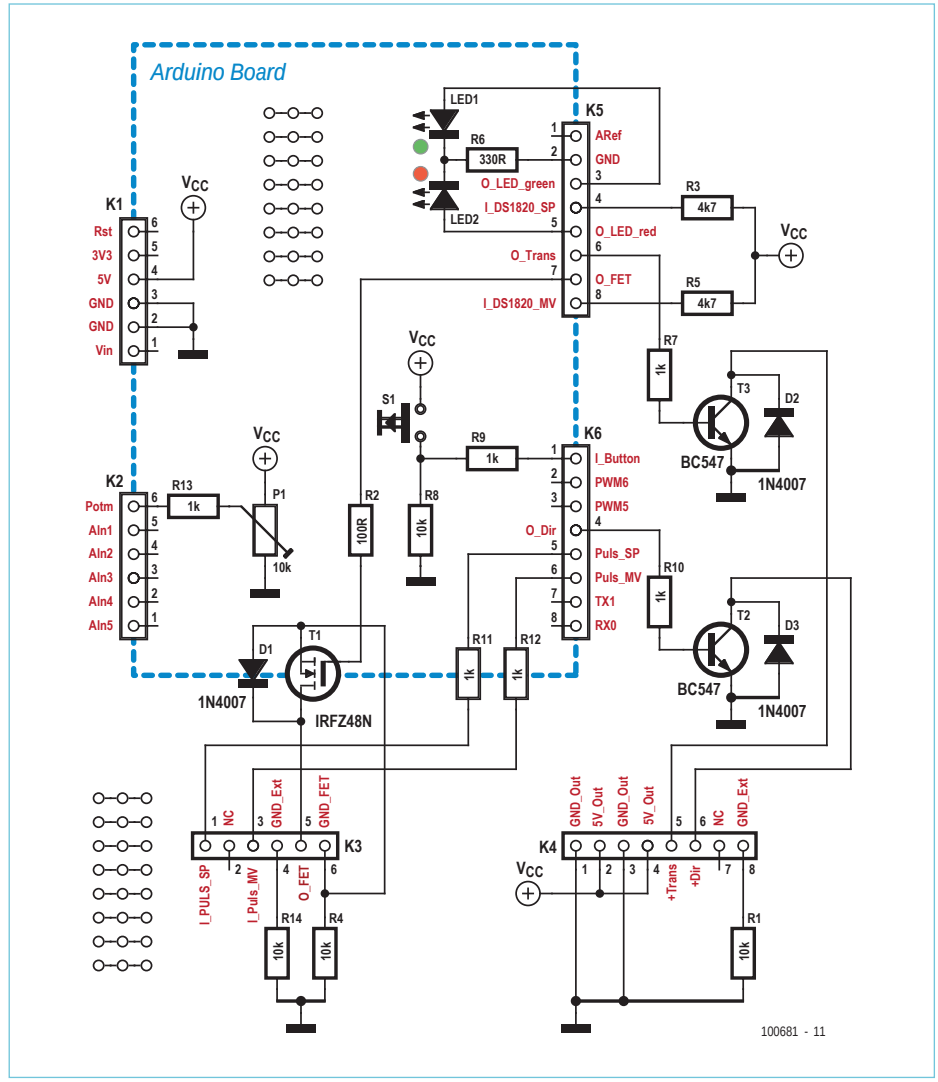


Bild 1. Auf der Schnittstellenkarte befinden sich diverse Steckleisten und einige Halbleiter. Die senkrechten Steckleisten passen in die Kontaktleisten des Arduino.

Sollwerteingabe

Für die Eingabe des Sollwerts gibt es vier Möglichkeiten:

- Wenn Taster S1 auf der Schnittstellenkarte gedrückt wird, übernimmt der Regler den aktuellen Istwert als Sollwert. Die Zeit, die S1 betätigt wird, muss mindestens einen Zyklus und höchstens drei Sekunden lang sein. Bei Zyklen, die mehr als drei Sekunden lang sind, beträgt die obere Grenze drei Zyklen.
- Der Sollwert wird über das zugehörige PC-Programm festgelegt. Wenn die Eigenschaften des Sensors in der Software implementiert sind, ist die Eingabe unmittelbar, beispielsweise in °C möglich.
- Der dritte Weg ist das Einstellen des Sollwerts als Spannung mit dem auf der Schnittstellenkarte befindlichen Potentiometer P1.
- Auch über einen Mikrocontroller-Eingang lässt sich der Sollwert festlegen, zum Beispiel wenn die Drehzahl eines Motors von der gemessenen Drehzahl eines anderen Motors abhängt. Als Eingänge sind AIn1 auf K2 (0...5 V), I_Puls_SP auf K3 (Impulszeiten, beispielsweise eines Encoders) oder I_DS1820_SP auf K5 (Eindraht-Signal) vorgesehen.

Eine schnelle Sichtkontrolle ist mit zwei LEDs möglich:

- Die rote LED leuchtet kontinuierlich beim Lesen und Schreiben von Einstellungen (Initialisierung), beim Senden gespeicherter Daten (zum Zeichen dafür, dass der Regler inaktiv ist) sowie in dem Fall, dass der Messwert oder Sollwert 0 oder 1023 beträgt (mögliche Sensorstörung).
- Die grüne LED leuchtet kontinuierlich, wenn der Istwert innerhalb der Toleranzgrenzen liegt (< Error_max), sie blinkt, wenn sich der Istwert in Richtung Sollwert bewegt oder unverändert bleibt. Solange der Istwert außerhalb des Toleranzbereichs liegt, blinkt die rote LED.

Starten mit dem PC

Wenn das Arduino-Board zusammen mit der Schnittstellenkarte am PC angeschlossen ist, erscheint nach dem Programmstart ein Fenster, das nach Anklicken der Schaltfläche *Datastream* einen Überblick über die Funktion der Regelung gibt. Das Fenster gibt auch Auskunft über den eingestellten Sollwert und die Steuerung der Aktoren.
Über die *Datalog*-Funktion sind die Aktionen des Reglers auch im Nachhinein kontrollierbar. Das kann zum Beispiel nützlich sein, falls der Regler zuvor autonom (*stand alone*) gearbeitet hat. Wenn Taster S1 auf der Schnitt-

Stückliste

Widerstände:
R1,R4,R8,R14 = 10 k
R2 = 100 Ω
R3,R5 = 4k7
R6 = 330 Ω
R7,R8,R9,R10,R11,R12,R13 = 1 k
P1 = 10 k Trimmpoti

Halbleiter:
D1,D2,D3 = 1N4007
LED1 = LED 5 mm, grün
LED2 = LED 5 mm, rot
T1= IRFZ48N
T2,T3 = BC547

Außerdem:
K1,K2 = Stiftleiste 6-polig, Raster 2,54 mm
K3,K4 = Stiftleiste 8-polig, Raster 2,54 mm
K5,K6 = Buchsenleiste 6-polig, Raster 2,54 mm
Platine 100681-1 (siehe [3])

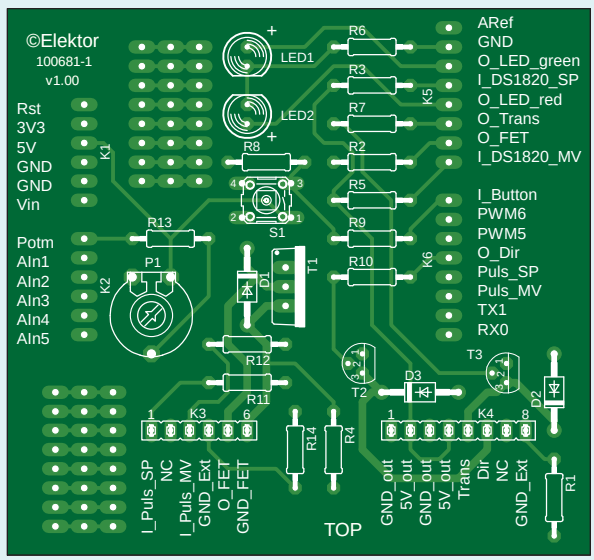


Bild 2. Das Titelfoto zeigt den Aufbau auf einer vorläufigen Platine, inzwischen ist eine Platine im gewohnten Elektor-Design lieferbar.

Bild 3. Im Startfenster wird der COM-Port eingestellt, ferner ist die Quelle der Initialisierung (EEPROM oder PC-Datei) wählbar.

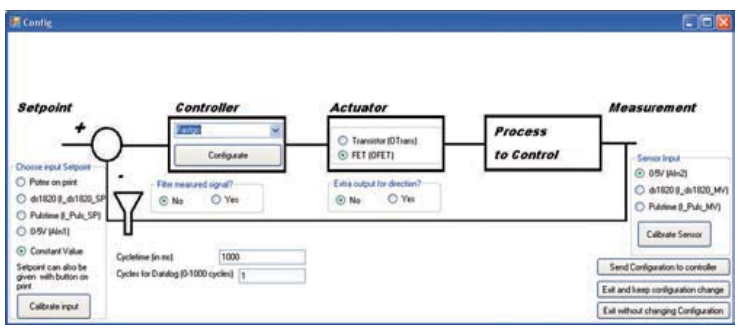


Bild 4. Fenster für die Konfiguration des PC-Programms „Control" mit den Einstellungen für Setpoint, Type Controller, Output und Sensor.

stellenkarte für drei Sekunden (und mindestens die dreifache Zykluszeit) gedrückt wird, speichert der Regler die Daten in seinem EEPROM, so dass sie nicht verloren gehen können, wenn man die Stromversorgung auf USB umschaltet. Dabei muss abgewartet werden, bis die rote LED verlischt, abhängig von der Datenmenge kann dies bis zu fünf Minuten dauern.
Bei jedem Regler-Start werden die Einstellungen aus dem EEPROM gelesen, nur beim ersten Start sind diese Speicherplätze noch leer. Die initialen Daten, die in der Datei *...\Bascom_bas_hex\PIDINIT2.hex* stehen, müssen mit dem Programm *AVRdude* geladen werden. Dieses Programm benutzt den Arduino-Bootlader, so dass eine Programmierung über ISP nicht erforderlich ist. Auf der Kommandozeilenebene oder in einer Stapeldatei kann das Kommando beispielsweise *avrdude -F -p m328p -b 115200 -c arduino -P com3 -U flash:w:PIDINIT2.hex:i* lauten. Die Angabe des seriellen Ports (hier COM3), an den der Arduino angeschlossen ist, muss gegebenenfalls korrigiert werden. Die Baudrate 115200 gilt für den Bootlader der Arduino-Version „Uno", beim „Duemilanove" beträgt sie 57600. Die genannte Hex-Datei muss sich im gleichen Ordner wie das Programm AVRdude befinden.
Nachdem der Arduino programmiert ist, kann auf dem PC das Visual-Basic-Programm *Control* gestartet werden. Dort wird zuerst der serielle Port gewählt, an den der Arduino angeschlossen ist. Im nächsten Fenster muss die Schaltfläche *SendSetup-2EEPROM* angeklickt werden. Nachdem die Setup-Werte im EEPROM stehen, ist der nächste Schritt das Laden von *...\Bascom_bas_hex\PID2.hex* in den Arduino. Die Vorgehensweise ist identisch wie oben, lediglich die Argumente von AVRdude müssen angepasst werden. Dort muss natürlich der zutreffende Dateiname stehen, und dem *-F* muss ein *-D* folgen. Damit wird sichergestellt, dass die Werte im EEPROM erhalten bleiben. Jetzt ist das Visual-Basic-Programm auf dem PC einsatzfähig.

Wie arbeitet die Regelung?

Nach einem Reset liest das Arduino-Programm die Einstellungen der Regelung, die nicht flüchtig im EEPROM gespeichert sind. Falls das EEPROM leer ist, zum Beispiel nach

der ersten Inbetriebnahme, muss zur Initialisierung die Programmversion *pidinit2* in den Arduino geladen werden. Diese Version macht die erste Einstellung über das PC-Programm *Control* möglich. Wenn das geschehen ist und die Einstellungen im EEPROM gespeichert sind, kann das Programm *pid2* geladen werden. Mit diesem Programm arbeitet der Regler autonom, also *stand alone.*

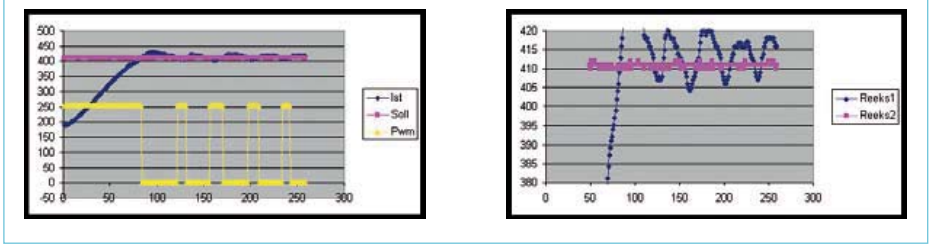


Bild 5. Verhalten einer Zweipunktregelung, die bei Temperaturabweichungen von ±0,1 °C schaltet. Die Systemträgheit bewirkt Abweichungen der gemessenen Temperatur von etwa ±0,7 °C. Die Messdaten können in Dateien gespeichert und in Excel ausgewertet werden.

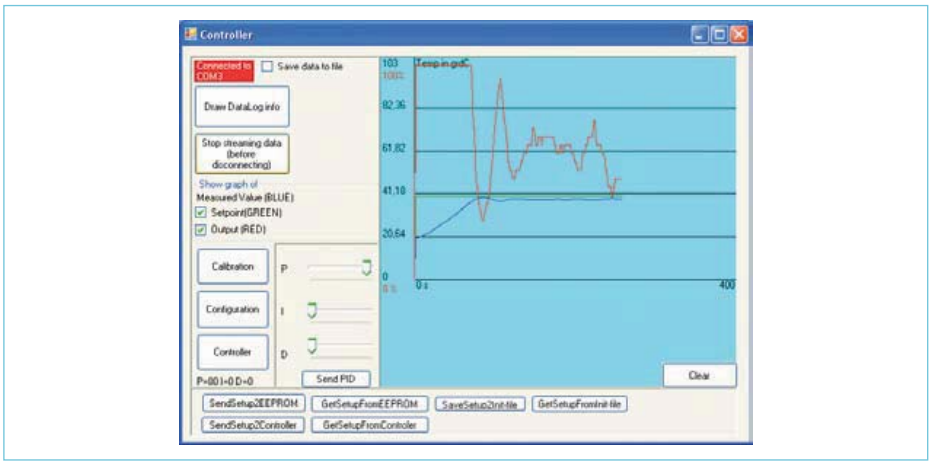


Bild 6. In diesem Fenster lässt sich im laufenden Betrieb das PID-Verhalten einstellen, außerdem sind die Fenster für die Regler-Konfiguration, das Kalibrieren sowie für weitere Einstellungen zugänglich. Weil der Regler hier auf P-Verhalten eingestellt ist, bleibt eine statische Abweichung übrig.

Zweipunktregelung

Der Regler vergleicht den gemessenen Istwert mit einem minimalen und maximalen Wert. Abhängig vom Ergebnis wird die Ausgangsgröße entweder eingeschaltet oder ausgeschaltet. Das Regelverhalten lässt sich wie folgt beschreiben:

Istwert < Minimalwert → Ausgang EIN
Istwert > Maximalwert → Ausgang AUS

PID-Regler

Das Verhalten eines PID-Reglers ist vergleichsweise komplex, weil hier die Eigenschaften eines P-, I- und D-Reglers zusammentreffen. Das gilt für den parallelen Regler, im Programm sind auch der „klassische" Regler oder der kaskadierte Regler aktivierbar [1].

Beim P-Regler ist die Ausgangsgröße proportional zum Fehler und zum einstellbaren Parameter Kp:

P_Aktion = P x Fehler

Beispiel: Wenn die Raumtemperatur gleich dem Sollwert ist, wird keine Leistung umgesetzt. Tritt eine Abweichung auf, hängt die Leistung von P und von der maximalen Leistung ab, die von der Energiequelle und vom Aktor bereitgestellt werden kann.

Beim I-Regler ist die Ausgangsgröße proportional zur Summe der Abweichungen über die Zeit:

I_Aktion = I_Aktion_alt + I x Fehler

Beispiel: Ein I-Regler schaltet die Heizungsanlage aus, solange die Raumtemperatur mit dem Sollwert übereinstimmt. Wenn die Außentemperatur nicht gleich der Raumtemperatur ist, muss die Heizung gedrosselt laufen, damit die Raumtemperatur erhalten bleibt. Der I-Regler steuert die Heizungsanlage so, dass die Wärmeverluste ausgeglichen werden.

Beim D-Regler ist die Ausgangsgröße proportional zur Fehleränderung in der Zeiteinheit (Zyklus):

D_Aktion = D x (Fehler - Fehler_alt)

Beispiel: Die Abweichung steigt sprunghaft an, weil die Raumtemperatur deutlich höher eingestellt wird, oder ein Fenster wird zum Lüften geöffnet. In diesem Fall reagiert der Regler sofort, indem er auf höchste Leistung schaltet.

Einstellung eines PID-Reglers

Die Einstellung eines PID-Reglers hängt wesentlich von der Anwendung ab. Wenn die Bewegung eines Fräskopfs geregelt wird, der von einem Werkstück Material abhebt, darf der Fräskopf in keinem Fall über die Endposition hinausfahren. Das

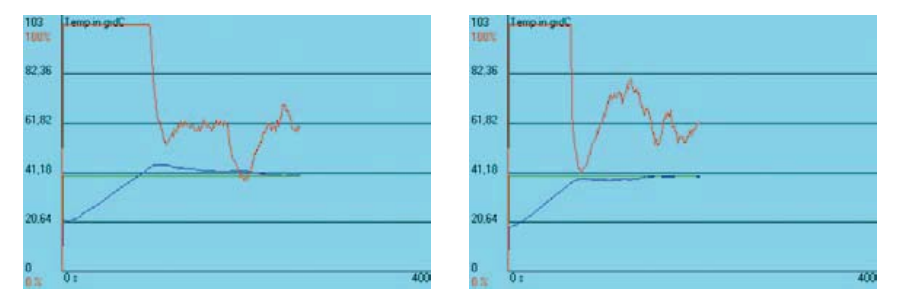


Bild 7. Nach Hinzufügen einer integrierenden Komponente (P=60, I=1) wird der Sollwert exakt erreicht, es entsteht jedoch ein Überschwingen (links). Das Einführen eines proportionalen Bereichs (PIDon=3) dämpft das Überschwingen, außerdem wird der Sollwert schneller erreicht.

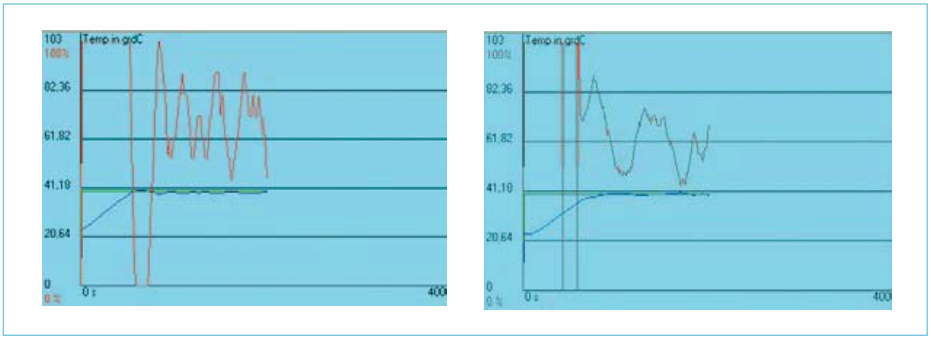


Bild 8. Links das Verhalten eines Reglers des Typs B (P=0,3, I=0,2), rechts das Verhalten eines „Fastgo"-Reglers (P=60, I=5, C1=4).

Werkstück wäre anderenfalls unbrauchbar. Eine Temperaturregelung ist toleranter, dort fallen geringe Abweichungen selten ins Gewicht. Zu den Regler-Typen und ihren Einsatzgebieten sei an dieser Stelle auf die Literatur [1] verwiesen.
Meistens genügt es, einen PID-Regler durch geschicktes Probieren (*trial and error*) einzustellen. Zuerst wird das P-Verhalten so gewählt, dass der Istwert den Sollwert in angemessener Zeit erreicht, dann wird das I-Verhalten so eingestellt, dass kein permanenter Fehler auftritt, und schließlich wird mit dem Einstellen des D-Verhaltens die Reaktionszeit optimiert. Falls nötig muss diese Prozedur wiederholt werden.
PID-Regler haben den Nachteil, dass sie bei nicht optimaler Einstellung entweder zu träge reagieren, oder der Istwert pendelt über den Sollwert hinaus (*overshoot*). Verantwortlich für das Überschwingen ist das I-Verhalten, das den Istwert zwar beschleunigt in Richtung Sollwert steuert, nach Erreichen des Sollwerts klingt dieses Verhalten jedoch nur langsam ab (*integral wind-up* [2]). Auf das I-Verhalten kann bei vielen Anwendungen nicht verzichtet werden, da anderenfalls der Istwert den Sollwert nicht exakt erreicht.
Um das Überschwingen zu dämpfen, wurden spezielle Algorithmen entwickelt, zum Beispiel für Regler der Typen B und C [2]. Ferner ist das Einfügen eines proportionalen Bereichs möglich. Das ist ein Gebiet, in dem sich der Regler bei Abweichungen, die größer als ein vorgegebener Wert PIDon sind, wie ein Zweipunktregler verhält. Im Programm des Arduino-Reglers ist ein proportionaler Bereich wirksam, solange der Vorgabewert (PIDon) nicht zu hoch gewählt wird. Eine Methode für die korrekte Einstellung besteht darin, der Steuerung einen „erwarteten" Wert zuzuweisen, wobei der Sollwert und ein Systemmodell als Basis dienen. Ein einfaches Modell kann beispielsweise dadurch entstehen, dass die Ausgangsgröße für zwei Eingangswerte bestimmt wird. Eine andere Methode, bei der das Systemmodell vom Regler berechnet wird, ist die „Fastgo"-Methode. Wenn die Differenz zwischen Istwert und Sollwert sehr hoch ist, steuert der Regler mit voller Kraft entgegen. Nachdem der Fehler zur Hälfte eliminiert wurde, berechnet der Regler die Richtung der Regelkurve. Die Richtung gibt den Punkt vor, an dem der Regler in den PID-Betrieb übergeht.

Was tun mit der Regelung?

Fast immer gibt es in Haus, Heim oder Garten etwas zu regeln. Beim Autor wird mit dem Arduino die Temperatur eines Heizelements geregelt, das ein PC-Netzteil mit Strom versorgt. Ein Eindraht-Sensor DS1820 misst die Temperatur, während der Power-FET das Heizelement schaltet (Pin 5 auf K3, Steuerung über Leitung O_FET auf K5). Eine andere Anwendung des Autors ist die Drehzahlregelung eines Elektromotors (Drehzahlmessung mit Fotosensor und Stroboskopscheibe, Eingangssignal an Puls_MV auf K6, Motorsteuerung mit T3, Anschlüsse +Trans auf K4 und O_Trans auf K5).
Für Arduino typische Anwendungen sind, um nur einige zu nennen, die Raumheizung in Einfamilienhäusern, die Beheizung des Wassers in Aquarien oder die Stellung der Jalousie-Lamellen abhängig vom Tageslicht. Die Möglichkeiten sind überaus vielfältig, es kommt auf die geschickte Wahl der Sensoren und Aktoren an.

Die Software des Arduino-Reglers kann von der Elektor-Website [3] kostenlos heruntergeladen werden. Zum Download gehört ein Word-Dokument mit ergänzenden Informationen, das allerdings zur Zeit nur in niederländischer Sprache verfügbar ist. Die Software steht unter den Lizenzen Cc, By, Nc und Sa, was bedeutet: Solange die Software nicht kommerziell genutzt wird (Nc) und der Name des Autors in der Software erscheint (By), darf sie frei genutzt und weitergegeben werden. Bearbeitete oder erweiterte Versionen dürfen nur unter den gleichen Bedingungen verbreitet werden (Sa).

(100681)gd

Arduino-Versionen

Die Software dieses Projekts wurde zwischen Juni und Dezember 2010 entwickelt. Vor der Veröffentlichung hat die Elektor-Redaktion die Software mit der aktuellen Arduino-Version „Uno" vom September 2010 getestet. Obwohl die Unterschiede zum „Duemilanove" gering sind, lief das Programm auf der neuen Arduino-Version nicht. Abhilfe war nur durch Laden des alten Bootladers in den „Uno" möglich. Ursache des Problems ist der Wert, den der „Uno"-Bootlader in das Register setzt, das für das Verdoppeln der Baudrate zuständig ist. Das Register wird vom Bootlader nach dem Bootladevorgang nicht zurückgesetzt.

Lösen lässt sich das Problem, indem im Bascom-Programm sowohl die Baudrate als auch das Register auf die zutreffenden Werte gesetzt werden:

```
$baud = 9600
Ucsr0a = &H00
```

Wir danken J.F. Theinert für diesen Hinweis.

Literatur und Weblinks

[1] Jos van Kempen: Regeltechniek
Pearson Education, 2009
ISBN 978-90-430-1811-1
(In niederländischer Sprache)

[2] http://bestune.50megs.com/typeABC.htm

[3] www.elektor.de/100681

First Step

NEU!

Erste Schritte mit dem Mikrocontroller

Sie interessieren sich als Auszubildender, Schüler, Student – oder einfach nur so – für Mikrocontroller-Technik? Mit dem neuen „First Step"-Paket haben Sie den Schlüssel und alle nötigen Werkzeuge für diese faszinierende Welt in der Hand! Das fertig bestückte und getestete „First Step"-Board und drei exakt darauf abgestimmte Arbeitshefte (plus Software-CD) machen die ersten Experimente mit einem Mikrocontroller zum Kinderspiel.

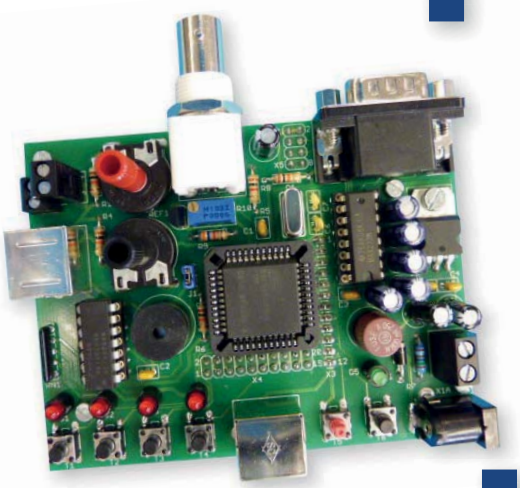
Bestandteile des „First Step"-Pakets:

elektor ACADEMY
the school of electronics

3 Arbeitshefte

(inkl. passendes DIN A4-Ringbuch)

- Beschreibung der Hardware
- Beschreibung der integrierten Entwicklungsumgebung IDE
- Einführung in die Programmiersprache „C"
- Zahlensysteme, Arithmetik, Variable
- Logische Operationen
- Digitale I/O-Ports
- A/D-Wandler, Tiner/Counter



1 „First Step"-Mikrocontroller-Board

- 8051er-Mikrocontroller: AT89C51CC03
- 2,5-V-Referenzspannungsgeber für A/D-Wandler: LT1009
- TTL/RS-232-Pegelwandler MAX232
- Treiber für LEDs und Piezo-Summer: 74HC04
- 4 Taster (Eingabe von binären Signalen)
- 4 LEDs (Ausgabe von binären Signalen)
- Piezo-Summer (Ausgabe von akustischen Signalen)
- BNC-Buchse (Ein-/Ausgabe von externen binären Signalen)
- 2 Potentiometer (Eingabe von analogen Signalen)
- 2 Mini-DIN-Buchsen und eine Doppelstock-Schraubklemme
- Karten-Format: 98 x 75 mm
- Spannungsversorgung: 9 V DC, max. 100 mA, Verpolungsschutzdiode und Miniatursicherung

1 CD-ROM mit Zusatzinfos

- Datenblätter
- Systemdokumentation
- Entwicklungsumgebung
- Beispielprogramme

Das gesamte „First Step"-Paket kostet nur 199,00 Euro.

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

Zurück zu den Wurzeln (3)

Transistoren durchgemessen

Von Burkhard Kainka (D)

Die Elektronik wird immer komplexer, ein einzelner Stromkreis ist da nicht mehr im Blick. So wird es für Einsteiger immer schwieriger, den Anschluss zu bekommen. In dieser Serie wollen wir daher zurück zu den Grundlagen! In Teil 3 führen wir verschiedene Messungen an einem Transistor durch. Schon mit einem einfachen analogen Multimeter kann man viel über so ein Bauteil herausfinden!

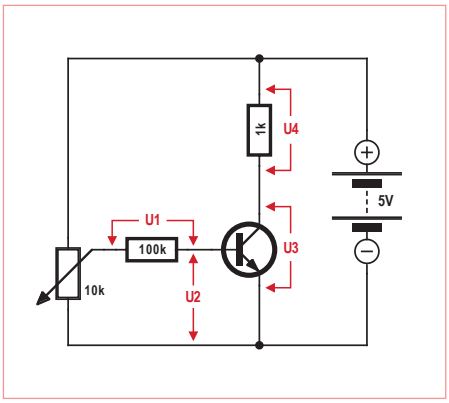


Bild 1. Der Messaufbau.

Tabelle: Messwerte für einen Transistor BC547B

	U1	I_B	$U_2=U_{BE}$	$U_3=U_{CE}$	U4	I_C	$V = I_C/I_B$
1	0 V	0 µA	0 mV	5 V	0 V	0 mA	0
2	0 V	0 µA	400 mV	5 V	0 V	0 mA	0
3	0,07 V	0,7 µA	573 mV	4,9 V	0,1 V	0,1 mA	143
4	0,15 V	1,5 µA	595 mV	4,8 V	0,2 V	0,2 mA	133
5	0,26 V	2,6 µA	612 mV	4,6 V	0,4 V	0,4 mA	153
6	0,47 V	4,7 µA	629 mV	4,2 V	0,8 V	0,8 mA	170
7	0,90 V	9,0 µA	646 mV	3,4 V	1,6 V	1,6 mA	177
8	1,77 V	17,7 µA	665 mV	1,8 V	3,2 V	3,2 mA	181
9	2,63 V	26,3 µA	679 mV	0,3 V	4,7 V	4,7 mA	179
10	3,54 V	35,4 µA	681 mV	0,15 V	4,85 V	4,85 mA	137
11	4,32 V	43,2 µA	683 mV	0,13 V	4,87 V	4,87 mA	113

In den Datenblättern der Hersteller findet man alle möglichen Kennlinien, die beschreiben, wie sich ein Transistor in welcher Situation verhält. Am besten ist es aber, wenn man selbst einmal zum Messgerät greift und so viel wie möglich selbst misst. Das vermittelt ein Gefühl für das Bauteil, und man versteht den Transistor besser, nicht nur in der Theorie.

Man sollte den Basisstrom I_B, den Kollektorstrom I_C, die Basis-Emitter-Spannung U_B und die Kollektor-Emitter-Spannung U_{CE} messen. Wenn man alle Messungen mit demselben Multimeter durchführen möchte, ist es günstiger, nur Spannungen zu messen und den Messbereich möglichst nicht zu wechseln. Die Ströme lassen sich dann aus den Spannungen und den Widerständen in der Schaltung leicht berechnen.

Bild 1 zeigt unseren Messaufbau. Mit dem Poti am Eingang der Schaltung soll die Eingangsspannung in kleinen Schritten zwischen Null und 5 V verändert werden. Bei jeder Einstellung werden die vier Spannungen U1 bis U4 gemessen und notiert. Daraus lassen sich dann die Ströme und der Verstärkungsfaktor V ableiten. Die **Tabelle** zeigt ein

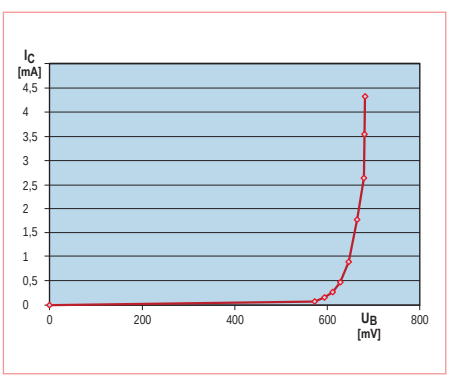


Bild 2. Basisstrom aufgetragen gegen die Basis-Emitter-Spannung.

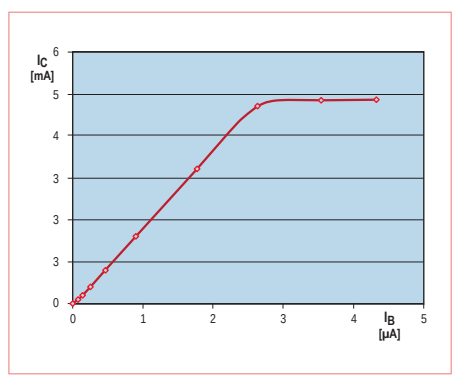


Bild 3. Kollektorstrom in Abhängigkeit vom Basisstrom.

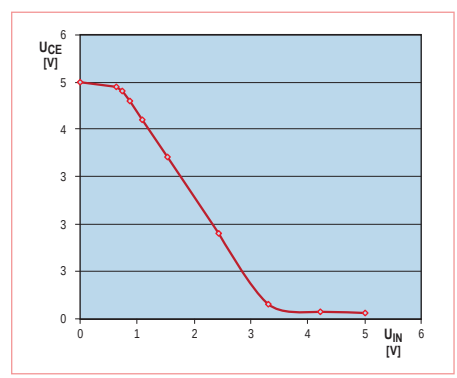
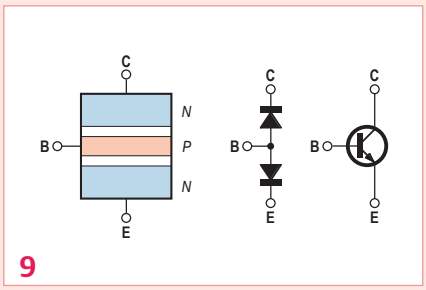


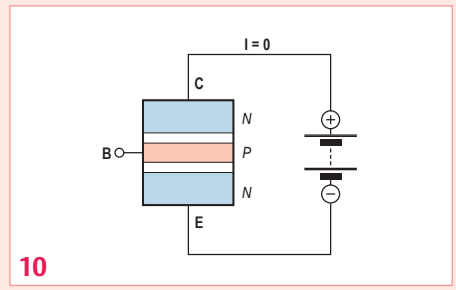
Bild 4. Ausgangsspannung als Funktion der Eingangsspannung.

Transistor-Grundfunktion

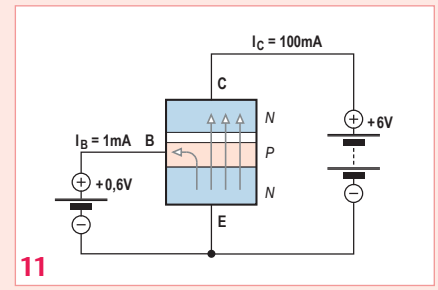
Der Transistor ist ein Halbleiterbauelement mit drei Anschlüssen, das überwiegend als Stromverstärker eingesetzt wird. Wie eine Diode besteht der Transistor aus n- und p-dotiertem Halbleitermaterial. Er hat aber drei Schichten mit zwei dazwischen liegenden Sperrschichten. Die Schichtenfolge kann N-P-N oder P-N-P sein. Hier soll zunächst der NPN-Transistor betrachtet werden, dessen Aufbau und Ersatzschaltbild man in Bild 9 sieht.



9



10



11

Die einzelnen Schichten des Transistors bezeichnet man als Emitter (E), Basis (B) und Kollektor (C). Entscheidend für die Funktion ist, dass die Basisschicht sehr dünn ist. Der Transistor soll zunächst mit freiem Basisanschluss an eine Stromquelle gelegt werden, wobei der Emitter mit dem Minuspol verbunden ist (Bild 10). Es fließt kein Strom, weil die Basis-Kollektor-Sperrschicht in Sperrrichtung liegt.

Nun soll eine zweite Stromquelle zwischen Basis und Emitter angeschlossen werden, wobei der Pluspol an der Basis liegt und die Spannung mit etwa 0,6 V so gering ist, dass nur ein kleiner Strom durch die Basis-Emitter-Diode fließt. Dabei kann man zwischen Emitter und Kollektor einen wesentlich größeren Strom beobachten. Die Erklärung dafür findet sich in der sehr dünnen Basisschicht. Treten nämlich n-Ladungsträger in die Basis ein, gelangen sie sofort in das starke elektrische Feld der Basis-Kollektor-Sperrschicht. Die meisten der Ladungsträger werden zum Kollektor hin abgesaugt. Nur etwa ein Prozent der Ladungsträger, die vom Emitter ausgehen, gelangen zum Basisanschluss (Bild 11). Umgekehrt ist also der Kollektorstrom etwa 100 Mal größer als der Basisstrom. Der Kollektorstrom wird über die Basis-Emitter-Spannung bzw. über den Basisstrom gesteuert. Doch auch wenn sich die Elektronen vom Emitter zum Kollektor bewegen: Aus Gründen der Tradition (Strom fließt von Plus nach Minus) sagt man, dass der Strom „vom Kollektor zum Emitter“ fließt.

Beispiel. Hier wurde ein BC547B durchmessen, wobei für U1, U2 und U3 eigene Digitalmultimeter fest angeklemmt blieben. U4 wurde aus U3 berechnet. Die Ströme und der Verstärkungsfaktor wurden dann aus den gemessenen Spannungen abgeleitet.

Praxistipps

In welchen Schritten sollte man bei der Messung vorgehen? Günstig ist es, den Kollektorstrom zuerst auf 0,1 mA einzustellen (U4 = 0,1 V). Dann verdoppelt man diesen solange bei jeder neuen Messung, bis er nicht mehr weiter ansteigt (U4 wird also auf 0,1 V, 0,2 V, 0,4 V, 0,8 V usw. eingestellt). Dabei kommt nämlich eine interessante Gesetzmäßigkeit zum Vorschein: Jede Verdoppelung des Kollektorstroms wird zwar ungefähr durch eine Verdoppelung des Basisstroms erreicht. Die Basis-Emitter-Spannung vergrößert sich aber jeweils um einen konstanten Betrag von ungefähr 20 mV.

Durch Teilen von I_C durch I_B erhält man den Stromverstärkungsfaktor. Wie die Tabelle zeigt, liegt der maximale Verstärkungsfaktor bei unserem Versuch bei rund 180. Das ist etwas wenig, denn eigentlich sollte mindestens eine 200-fache Verstärkung erreicht werden. Es gibt allerdings einige Fehlerquellen, so wie der endliche Innenwiderstand des Messgeräts (10 MΩ). Dies bedeutet, dass zum Beispiel bei der Messung von U2 ein kleiner Teil des Basisstroms für das Messgerät abgezweigt wird. Messfehler sind normal. Wenn man alle Fehlerquellen und Toleranzen (auch die der Widerstände) berücksichtigt, könnte dieser Transistor gerade an der unteren Verstärkungsgrenze von 200 liegen. Probieren Sie es einmal selbst, vielleicht bringt Ihr Transistor mehr.

Bei so vielen Messdaten sollte man sich auch gleich einmal an eine grafische Auswertung machen. Das geht z.B. mit Bleistift und Papier oder am PC mit einer Tabellenkalkulation. Und dabei kommt folgendes heraus: **Bild 2** (I_C über U_B) zeigt die typische exponentielle Kennlinie einer Silizium-Diode. Im linearen Maßstab sieht man einen Knick bei ca. 0,6 V. Darüber steigt der Strom immer steiler (exponentiell) an. Die Messdaten zeigen, dass sich bei einer Basisspannung von beispielsweise 400 mV noch kein messbarer Basisstrom zeigt – und damit auch kein Kollektorstrom. Man kann sich also merken: Die Basisspannung liegt meist irgendwo zwischen 0,6 V und 0,7 V.

Bild 3 (I_C über I_B) zeigt, dass der Kollektorstrom in erster Näherung linear mit dem Basisstrom ansteigt, bis er dann bei knapp 5 mA nicht mehr weiter zunimmt, d.h. „in die Sättigung“ geht. Mehr als 5 mA können ja gar nicht gemessen werden, weil der Kollektorwiderstand den Strom auf 5 mA begrenzt (5 V / 1 kΩ = 5 mA). Und man sieht nun deutlich, dass sich auch die 5 mA nicht ganz erreichen lassen. Der Transistor ist eben nur „fast ganz“ eingeschaltet; es bleibt eine Kollektor-Emitter-Restspannung von knapp über 0,1 V.

Die Kurve zeigt auch eine geringere Steilheit (also eine kleinere Stromverstärkung) bei sehr kleinen Strömen. Da ist tatsächlich was dran, denn bei sehr kleinen und sehr großen Kollektorströmen nimmt die Verstärkung etwas ab. Aber dahinter versteckt sich auch ein Messfehler, der diesen Effekt noch verstärkt. Bei der Messung von U2 fließt ein kleiner Messstrom, der den Basisstrom gerade im unteren Bereich größer erscheinen lässt als er tatsächlich ist.

Bild 4 zeigt schließlich die Ausgangsspannung (U_{CE}) in Abhängigkeit von der Eingangsspannung (U1 + U2) am Schleifer des Potis. Hier sieht man auf den ersten

Blick: Mehr Eingangsspannung führt zu weniger Ausgangsspannung. Der Grund ist klar. Wenn der Kollektorstrom steigt, erhöht sich der Spannungsabfall am Kollektorwiderstand.

Gegenkopplung

Wenn man eine Schaltung plant und den Verstärkungsfaktor nicht genau kennt, dann muss man sich darauf einstellen. Wenn es um eine Schaltstufe geht, ist die Sache einfach. Man muss den Basisstrom einfach nur so auslegen, dass die Schaltung auch noch mit dem geringstmöglichen Stromverstärkungsfaktor gut funktioniert. Also im Zweifel etwas mehr Basisstrom, dann passt es für alle Transistoren eines Typs.

Anders sieht die Sache aus, wenn eine analoge Größe verstärkt werden soll. Ein zu großer Basisstrom kann dann gerade verkehrt sein, weil der Transistor leicht in die Sättigung kommt. Man möchte aber möglichst oft einen mittleren Kollektorstrom haben, der nach unten und nach oben verändert werden kann. Eine Möglichkeit, dies mit unterschiedlichen Transistoren zu erreichen, ist die Gegenkopplung. Dazu legt man den Basiswiderstand nicht an die Versorgungsspannung, sondern an den Kollektor (siehe **Bild 5**). Ein Transistor mit besonders hoher Verstärkung würde einen größeren Spannungsabfall am Kollektorwiderstand bewirken. Damit sinken die Kollektorspannung und zugleich auch der Basisstrom. Umgekehrt erhalten Transistoren mit geringerer Stromverstärkung automatisch etwas mehr Basisstrom. Im Endeffekt passt es dann für alle Transistoren.

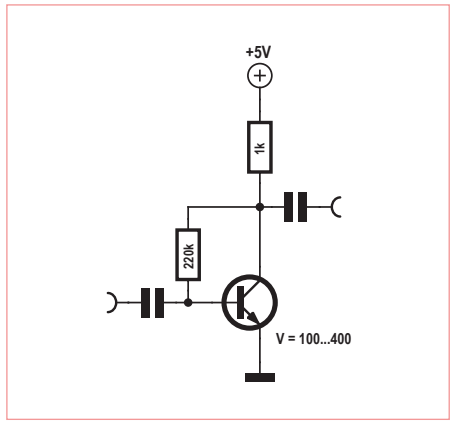


Bild. 5. Arbeitspunkteinstellung mit Gegenkopplung.

Messungen mit dem Ohmmeter

Gerade bei Bauteiletests und bei der Fehlersuche haben analoge Zeigerinstrumente immer noch ihre Vorteile. Man kann nämlich Ergebnisse viel schneller als bei einem Digitalmultimeter ablesen, zumindest einen groben Wert. Das Digitalmultimeter ist dagegen unverzichtbar, wenn es um hochgenaue Messungen geht.

Einfache analoge Vielfachmessgeräte bieten meist auch einen oder mehrere Widerstands-Messbereiche, und mit etwas Übung lassen sich mit einem einfachen Ohmmeter nicht nur Widerstände, sondern auch Transistoren, Dioden, Kondensatoren und viele andere Bauelemente überprüfen. Zur Widerstandsmessung benötigen die Multimeter eine Batterie, die oft für alle anderen Messbereiche ohne Funktion ist. Die Messung beruht im Prinzip auf einer Strommessung bei konstanter Spannung. Die Widerstandsanzeige ist daher nicht linear. Der Endausschlag bei null Ohm muss mit einem Potentiometer abgeglichen werden, um die unterschiedliche Batteriespannung auszugleichen (siehe **Bild 6**). Am anderen Ende der Skala reicht die Messung in jedem Bereich bis Unendlich.

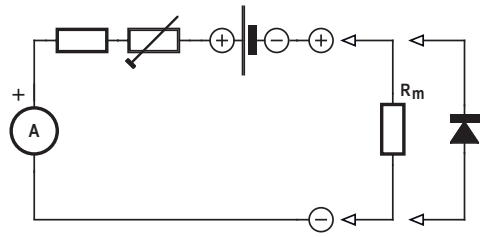


Bild 6. Prinzipschaltung eines analogen Ohmmeters.

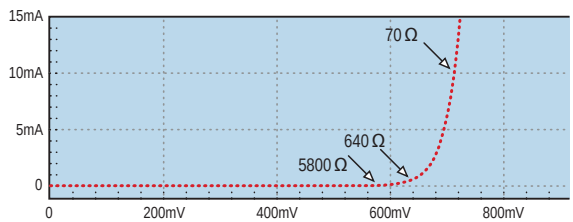


Bild 7. Gleichstromwiderstand einer Si-Diode bei verschiedenen Messströmen.

Die übliche Beschaltung einfacher Analog-Multimeter bringt es mit sich, dass die Spannung an den Anschlussklemmen in den Ohmmessbereichen anders gepolt ist, als es die Bezeichnungen der Anschlüsse für Strom- und Spannungsmessungen vermuten lassen. Am Minusanschluss des Vielfachmessgeräts liegt also der Pluspol des Ohmmeters. Dies ist zu beachten, wenn man ein Ohmmeter zur Überprüfung von Dioden oder Transistoren verwenden will.

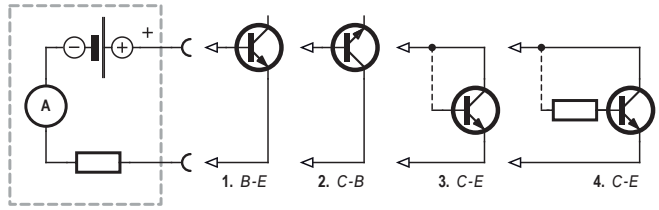


Bild 8. Messungen an einem Transistor.

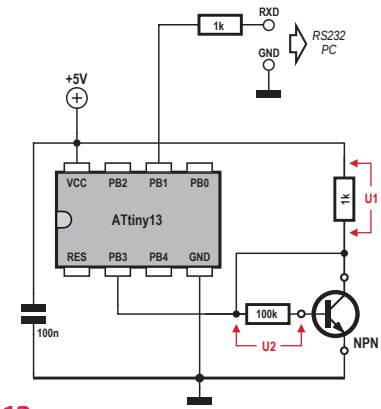
Bei der Messung an Diodenstrecken muss man im Kopf behalten, dass einer Sperrschicht kein konstanter Widerstand zugeordnet werden kann. Der angezeigte Wert hängt vielmehr vom Messstrom und damit vom

Transistor-Prüfer

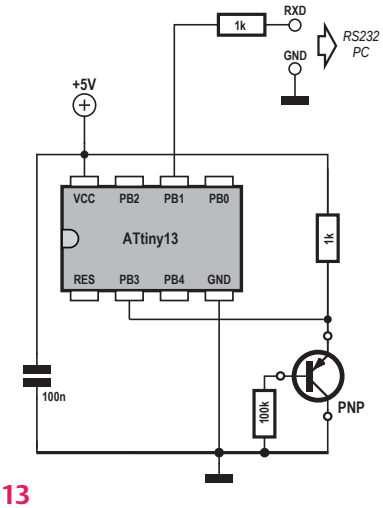
Ein Mikrocontroller mit internem A/D-Wandler eignet sich hervorragend als Messgerät. Wie wäre es z.B. mit einem Transistor-Prüfer? Es soll dabei einfach die Stromverstärkung bestimmt werden. Ein ATtiny13 reicht für diese Aufgabe, wenn das Endergebnis seriell an einen PC geschickt und dort in einem Terminalprogramm angezeigt wird.

Die Schaltung in **Bild 12** ist sehr einfach. Nur die Kollektorspannung wird gemessen. Der Transistor wird in Gegenkopplung betrieben, deshalb können sehr unterschiedliche Verstärkungsfaktoren gemessen werden. Das Programm muss dann etwas mehr rechnen, aber dafür ist ein Mikrocontroller ja da.

```
,Transistor tester
$regfile = „attiny13.dat“
$crystal = 1200000
$hwstack = 8
$swstack = 4              ' 16
$framesize = 4


Dim UC As Word
Dim U1 As Word
Dim U2 As Word
Dim I1 As Word
Dim I2 As Word
Dim V As Word

Config Adc = Single , Prescaler = Auto
Start Adc

Open „comb.1:9600,8,n,1,INVERTED“ For Output As #1

Do
  UC = Getadc(3)          ' PB3=ADC3 -> UC = 0..1023

  UC = UC * 50            ' max 51150 -> 5115 mV
  U2 = UC - 6000          ' 6000 <- U_BE = 600 mV
  U1 = 51150 - UC
  I1 = U1                 ' 1 k
  I2 = U2 / 100           ' 100 k
  V = I1 / I2
  Print #1 , V            ' --> RXD
  Waitms 1000
Loop

End
```



12



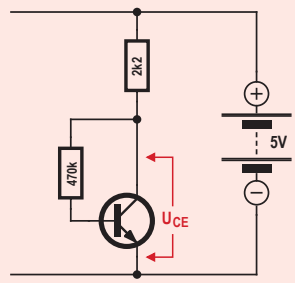
13

Das Programm berechnet den Spannungsabfall U1 am Kollektorwiderstand und den Spannungsabfall U2 am Basiswiderstand. Daraus ergeben sich der Kollektorstrom I1 und der Basisstrom I2. Die Verstärkung V ist dann I1 / I2. Damit alles in den kleinen Mikrocontroller passt, wird ausschließlich mit Ganzzahlen vom Typ Word gearbeitet. Dabei wurde darauf geachtet, dass weder ein Überlauf passieren kann noch durch zu kleine Zwischenergebnisse Genauigkeit verloren geht.

Der Mikrocontroller kann mit demselben Programm auch Messungen an einem PNP-Transistor durchführen. Man muss diesen nur etwas anders anschließen, wie **Bild 13** zeigt.

Quiz

Können Sie auch ohne Mikrocontroller mit einer einzigen Messung die Stromverstärkung eines Transistors bestimmen? Die Schaltung mit einem Kollektorwiderstand von 2,2 kΩ und einem Basiswiderstand von 470 kΩ zum Kollektor zeigt die Ausgangsspannung U_{CE}.



1) Sie messen U_{CE} = 2,8 V. Die Stromverstärkung beträgt

A) ca. 152-fach
B) ca. 214-fach
C) ca. 472-fach

2) Sie messen U_{CE} = 0 V. Was ist zu vermuten?

D) Der Transistor ist durchgebrannt (zweiter Durchbruch).
E) Der Basiswiderstand hat keinen Kontakt.
F) Der Basiswiderstand hat nur 470 Ω statt 470 kΩ.

3) Sie messen U_{CE} = 4,9 V. Wo könnte der Fehler liegen?

G) Unterbrechung im Bereich der Kollektorleitung.
H) Unterbrechung an der Basisleitung.
I) Emitter und Kollektor wurden vertauscht.

Wer uns die richtige Lösung zusendet, kann ein „Minty Geek Electronic 101 Kit“ gewinnen!

Senden Sie dazu den Lösungscode (ergibt sich durch Aneinanderreihung der Buchstaben der drei richtigen Anworten) per E-Mail an: basics@elektor.com.

Als Betreff der E-Mail bitte nur den Lösungscode angeben.

Einsendeschluss ist der 31. März 2012.

Die Lösung des Quiz aus der letzten Ausgabe finden Sie im nächsten Heft!

gewählten Messbereich ab. Trotzdem lassen sich Aussagen machen. Beobachtet man bei einem Ohmmeter mit einer internen Spannung von 1,5 V einen Zeigerausschlag von etwa der Hälfte der Skala, dann muss der Spannungsabfall am Messobjekt etwa 0,75 V betragen. Der Ausschlag ändert sich wegen der exponentiellen Kennlinie einer Diode nur geringfügig, wenn man den Bereich umschaltet. Es wird daher in jedem Messbereich ein anderer Widerstand angezeigt, der Zeigerausschlag ist dagegen ähnlich, da der Spannungsabfall immer um 0,6 V beträgt. Die Diodenspannung lässt Rückschlüsse auf den Diodentyp zu. In **Bild 7** dürfte es sich um eine Si-Diode handeln.

Transistorprüfung

Bei der Prüfung eines Transistors lassen sich nur mit einem Ohmmeter verschiedene Aussagen über den Typ und den Zustand des Messobjekts machen. Auch bei einem völlig unbekannten Transistor kann man so wenigstens die Anschlussbelegung herausfinden.
Mit drei typischen Messungen lässt sich ein Transistor vollständig prüfen. Zunächst werden die Basis-Emitter- und die Basis-Kollektor-Diode (siehe Kasten: Transistor-Grundfunktion) vermessen, womit sich bereits Si- und Ge-Transistoren unterscheiden lassen und mögliche Kurzschlüsse verraten (**Bild 8**, Teil 1 und 2). Danach misst man den Widerstand zwischen Emitter und Kollektor ohne und mit Basisstrom (3). Bei offener Basis zeigt ein intakter Transistor keinen Strom, also einen unendlichen Widerstand. Bei leitender Verbindung der Basis mit dem Kollektor muss sich ein etwas größerer Strom zeigen als bei der Basis-Emitter-Diode allein. Der letzte Test (4) sollte mit einem kleinen Basisstrom über einen Basis-Kollektor-Widerstand durchgeführt werden. Den Basisstrom kann man auch durch Berühren von Kollektor und Basis mit dem angefeuchteten Finger generieren. Der erzielte Ausschlag am Ohmmeter vermittelt einen groben Eindruck von der Stromverstärkung des Transistors. Auch bei vertauschtem Emitter und Kollektor zeigt sich aber noch eine geringe Stromverstärkung, so dass man den Transistor im Zweifelsfall noch einmal umdrehen sollte, wenn die Anschlüsse nicht sicher bekannt sind.

Digitalvoltmeter verwenden im Ohmmessbereich meist eine völlig andere Innenschaltung. Hier beruht die Widerstandsmessung auf einer Messung des Spannungsabfalls bei konstantem Strom. Damit ergibt sich eine lineare Anzeige und eine eindeutige Messbereichsgrenze. Ein Abgleich des Nullpunkts ist hier nicht erforderlich. Ein weiterer Unterschied gegenüber Zeigerinstrumenten ist, dass die Polung bei der Spannungs-/Strommessung sowie der Widerstandsmessung gleich ist.
Mit einem Digitalmultimeter im Ohmbereich lassen sich im Prinzip die gleichen Tests an Bauteilen durchführen wie mit einem analogen Gerät. Oft gibt es zusätzlich einen Messbereich speziell zur Messung an Diodenstrecken. Er arbeitet zwar wie der Widerstandsmessbereich, angezeigt wird jedoch der Spannungsabfall in Millivolt oder ein Messwert, der proportional zur Diodenspannung ist.

(120003)

DESIGNSPARK

Deadline for Entries: March 27, 2012

Turn a hot idea into a cool solution.

DesignSpark chipKIT™ **Challenge**

Get ready to win your share of $10,000 in cash prizes!

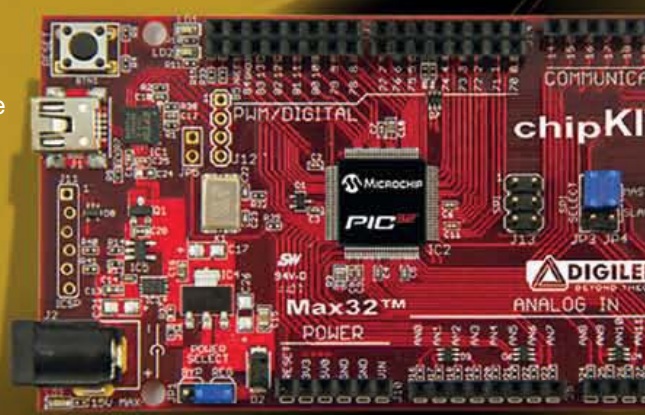
It's time to see if your hard work and superior engineering skills have paid off. The deadline for the **DesignSpark chipKIT™ Challenge** is just around the corner. It's time to finalize your design and prep your entries for the judges!

Will your design change the world? Reduce power consumption? Improve energy efficiency? There's only one way to find out.

Manage your project entry by clicking on the 'My Project' tab at **www.designspark.com/chipkitchallenge-projects/latest**. Be sure to upload, and clearly label, all materials necessary for judging your entry including an abstract, complete documentation, and source code.

For more information and tips on how to enter, visit **www.designspark.com/chipkitchallenge/faq.**

Don't delay! The DesignSpark chipKIT™ Challenge ends on March 27, 2012 at 18.00 GMT (13.00 EST).



Visit **www.chipkitchallenge.com**

for complete rules and details.

IN ASSOCIATION WITH:

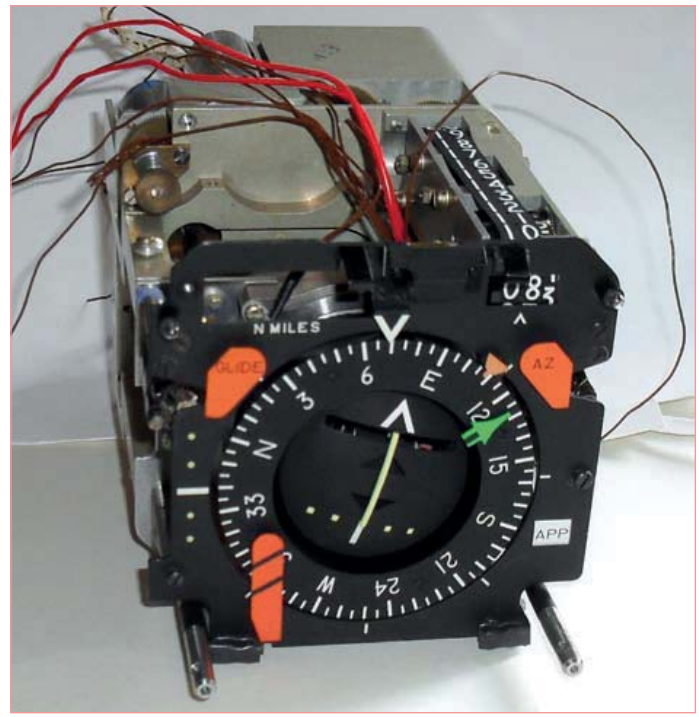
elektor · MICROCHIP · RS · CIRCUIT CELLAR · ALLIED ELECTRONICS · DIGILENT BEYOND THEORY

DesignSpark chipKIT™ Challenge

Avionik-Hack

Ansteuerung eines Horizontal Situation Indicator

Von Martin Oßmann (D)



Normalerweise beschäftigt sich der Verfasser eher mit Elektronik, aber gleichzeitig ist er auch fasziniert von mechanischen Wunderwerken aller Art. Als beim örtlichen Surplus-Händler ein interessant aussehendes Flugzeug-Instrument angeboten wurde, konnte er nicht widerstehen. Er hat dieses mechatronische Meisterstück namens HSI nach allen Regeln der Kunst unter die Lupe genommen und Aufbau und Funktionsweise ergründet. Hier sein überaus erhellender Bericht.

Der erwähnte lokale Surplus-Händler [1] war schon einmal Gegenstand eines Artikels in Elektor [2] und ist wahrscheinlich einem größeren Leserkreis bekannt. Bei dem hier untersuchten Instrument (**Bild 1**) handelt es sich um einen so genannten „Horizontal Situation Indicator“ (HSI) [3], ein Kombinationsinstrument, das die Anzeigen für das UKW-Drehfunkfeuer (VOR), den Gleitpfad (ILS) und den Kurskreisel (HI) in sich vereint.
Wenn dann so ein Gerät vor einem liegt, erhebt sich natürlich die Frage, ob man das Teil nicht selbst ansteuern kann. Dieser Artikel beschreibt, welche Technik in dem Instrument verwendet wird und wie man die verschiedenen Anzeigen selbst steuern kann. Damit kann man dann zum Beispiel versuchen, das Instrument an einen Flugsimulator zu koppeln oder eine „HSI-Uhr“ fürs Büro oder den Clubraum zu bauen. Wer sich für das Resultat interessiert: Der Autor hat auf Youtube ein Video eingestellt [4].

Hack1: Synchros und Motoren

Ein Blick von hinten auf das Innenleben (**Bild 1**) offenbart die Vielzahl der elektromechanischen Komponenten des Instruments. Dabei gibt es Teile mit einem blauen Deckel und solche mit unlackierter Metallkappe. Bei den blauen Teilen handelt es sich um Komponenten des Aerospace- und Avionik-Teile-Herstellers Muirhead [5], auf denen angegeben ist, dass drei Drähte zur Primärseite und zwei zur Sekundärseite gehören.
Diese Bezeichnungen legen nahe, dass es sich um so genannte Synchros [5] oder Resolver handelt. Eine gute Beschreibung dieser Komponenten findet man unter [6] und [7]. Die prinzipielle Aufgabe von Synchros ist es, Winkelinformationen zu erfassen und auf Anzeigeinstrumente weiterzuleiten. Die Winkelinformation kann beispielsweise von einem Kreiselkompass (korrekt: Kurskreisel) kommen. Um die Winkelinformation einer mechanischen Achse elektrisch weiterzuleiten, wird zuerst ein so genannter Synchro-Control-Transmitter (CT) eingesetzt. Er besitzt drei Statorwicklungen, die jeweils um 120 Grad mechanisch versetzt sind. Diese Wicklungen bilden die Sekundärseite. Die Primärseite wird aus einer Rotorwicklung

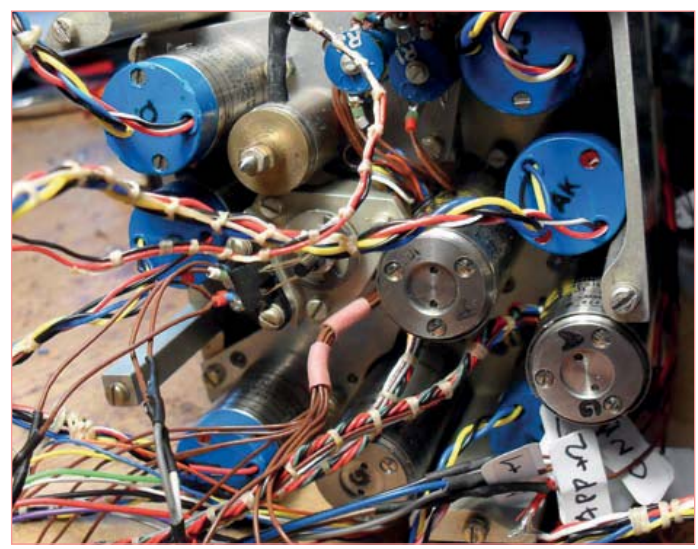


Bild 1. Ein Blick von hinten auf die Motoren und Synchros des HSI.

gebildet, die sich mechanisch mit der Achse dreht. (**Bild 2** links). Die Rotor-Primärwicklung wird von einem 400-Hz-Sinussignal $U_R(t) = \hat{U}\ sin(2\pi t400\ Hz)$ gespeist.
In den drei Sekundärwicklungen hängt die induzierte Spannung nun vom mechanischen Rotorwinkel θ ab. Es gilt:

$U_{S1}(t) = \hat{U}\ sin(\theta)\ \sin(2\pi t400\ Hz)$

$U_{S2}(t) = \hat{U}\ sin(\theta + 120°)\ sin(2\pi t400\ Hz)$

$U_{S3}(t) = \hat{U}\ sin(\theta - 120°)\ sin(2\pi t400\ Hz)$

Diese drei Spannungen enthalten nun die Winkelinformation, und mit den drei Leitungen kann man die Winkelinformation weiterleiten. Das Schaltsymbol eines CX ist in **Bild 2** rechts angegeben. Um aus der elektrischen Information wieder einen mechanischen Drehwinkel zu gewinnen, wird ein Regelkreis eingesetzt. Wie so ein System aussieht, ist in **Bild 3** dargestellt.
Die elektrische Winkelinformation wird auf einen sogenannten Synchro-Control-Transformer CT gegeben. Die Primärseite (Stator) des CT besteht (wie die Sekundärseite des CX) aus drei um je 120 Grad versetzten Wicklungen. Der Rotor bildet die Sekundärwicklung.
An der Sekundärseite entsteht eine 400-Hz-Wechselspannung, deren Amplitude proportional zu sin(A - B) ist, wobei A der Winkel des CX und B der Winkel des CT ist. Diese Spannung wird genau Null, wenn beide Winkel gleich sind. Die Spannung kann man als Fehlersignal benutzen, um den Winkel B mit einem Motor nachzusteuern, bis A = B ist. Genauso arbeiten die verschiedenen Winkelanzeigen in unserem HSI.

Alternative Ansteuerung

Wollte man dieses Prinzip zur Ansteuerung einsetzen, müsste man pro Anzeige-Winkel die drei elektrischen Signale des CX nachbilden und den Regelkreis aufbauen. Es geht aber auch anders.
Speist man nämlich die Windungen, wie in **Bild 4** gezeigt, mit zwei phasenverschobenen Signalen, kann man am Rotor ein 400-Hz-Sinussignal entnehmen, dessen elektrische Phasenverschiebung genau dem Achsenwinkel entspricht. Mit einem A/D-Kanal kann man dieses Signal verarbeiten und die Phase messen, um den Motor entsprechend anzusteuern. Alle CT kann man mit den gleichen Sinussignalen speisen, so dass man diese nur einmal erzeugen muss.

PWM-Signalerzeugung

Ein Sinussignal mit einer Frequenz von 400 Hz kann man gut mit einem Prozessor per PWM erzeugen. Wir verwenden einen Prozessortakt von 20 MHz und eine schnelle PWM mit 8 bit Auflösung. Die PWM-Frequenz ist damit 20 MHz/256 = 78.125 kHz, ist also sehr schnell gegenüber 400 Hz. Als Interruptfrequenz wird 36×400 Hz gewählt, indem man 20 MHz durch 1389 teilt. Mit jedem Interrupt wird dann jeweils ein neuer Sinus- bzw. Cosinuswert an die PWM gesandt. Das PWM Signal muss nun noch mit einem Tiefpass gefiltert und verstärkt werden, um die CT-Wicklungen speisen zu können. Für einen der beiden Kanäle ist die Schaltung in **Bild 5** dargestellt.

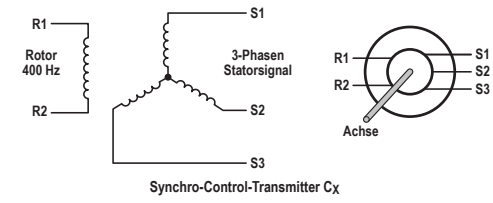


Bild 2. Aufbau eines Synchro-Transmitters CX.

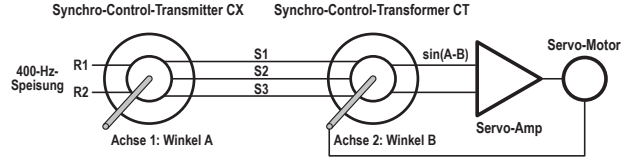


Bild 3. Elektrische Verlängerung einer Achse.

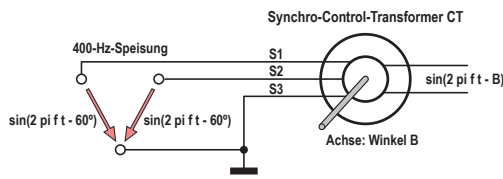


Bild 4. Alternative Ansteuerung eines Synchro-Transformators CT.

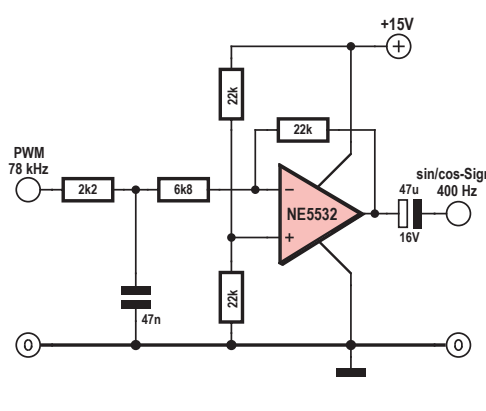


Bild 5. Tiefpass und Verstärker für das 400-Hz-Sinussignal.

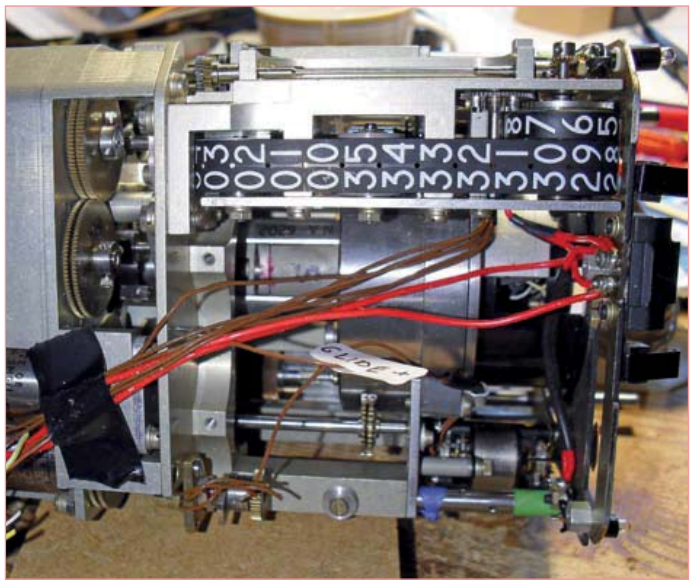


Bild 6. Blick von oben auf die Mechanik.

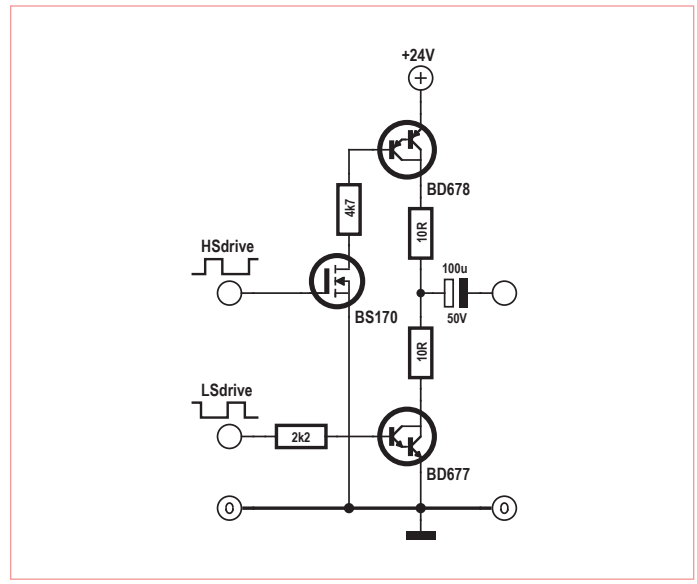


Bild 9. 24-V-Halbbrücke.

Phasenmessung

Um nun zu berechnen, welchen mechanischen Winkel die Achse einnimmt, müssen wir die Phase des Rotorsignals bestimmen. Das Signal geben wir auf einen A/D-Kanal und tasten diesen mit der gleichen Interruptroutine ab, wie sie zur PWM Erzeugung verwendet wird, also mit 400×36 Hz = 14400 Samples/s. Eine Periode entspricht dann 36 Werten. Aus diesen 36 Werten wird dann die Phase bestimmt, genau so wie bei einem Phasen-Demodulator in einem SDR-Empfänger. Das Signal wird in Sinus- und Cosinus-Anteil zerlegt, und aus diesen Quadraturkomponenten wird der Winkel bestimmt.

Damit erhalten wir den Winkel der verschiedensten Zeiger unseres HSI und können diese Information in Regelkreisen verwenden, um gewünschte Positionen anzusteuern.

Motorsteuerung

Die nächste zu lösende Aufgabe ist die Ansteuerung der Motoren. Aus einem dieser Motoren kommen immerhin 10 Kabel. Davon gehören vier, wie man der Beschriftung entnehmen kann, dem Tachogenerator. Dann gibt es noch drei weitere Wicklungen, von welchen eine mit „REF 26V 400c/s“ beschriftet ist. Diese Wicklung ist also an 26 V/400 Hz anzuschließen. Die Suche nach der Anschluss-Information für einen derartigen Motor war nicht ganz einfach, schließlich wurden aber doch Dokumente gefunden [8]. Offensichtlich handelt es sich um einen sogenannten Drag-Cup-Motor [9] und Tacho-Generator. Der Läufer des Motors ist wie bei einem Asynchronmotor aufgebaut, allerdings rotiert der magnetische Rückschluss im Inneren des Läufers nicht mit. Der Stator besteht aus einer sogenannten Referenzwicklung, die ständig mit 400 Hz bestromt wird. Im rechten Winkel dazu ist die zweite zweiteilige Wicklung angebracht. Bestromt man diese mit einem 400-Hz-Signal, das zur Referenzwicklung um 90 Grad phasenverschoben ist, dreht der Läufer sich. Über die Polarität kann man dann die Drehrichtung beeinflussen. Die Wicklung ist zweiteilig ausgeführt, um bestimmte Regelkreise einfach aufbauen zu können.

In **Bild 7** ist so ein Motoraufbau schematisch dargestellt. Der glockenförmige Läufer dreht sich um den Eisenkern im Inneren. Außen sind die zwei Wicklungen angebracht, die, wie bei einem Zweiphasen-Asynchronmotor, ein Drehfeld erzeugen [10].

Durch Experimentieren wurde herausgefunden, dass der Motor auch dann läuft, wenn die Spannung nicht sinusförmig, sondern

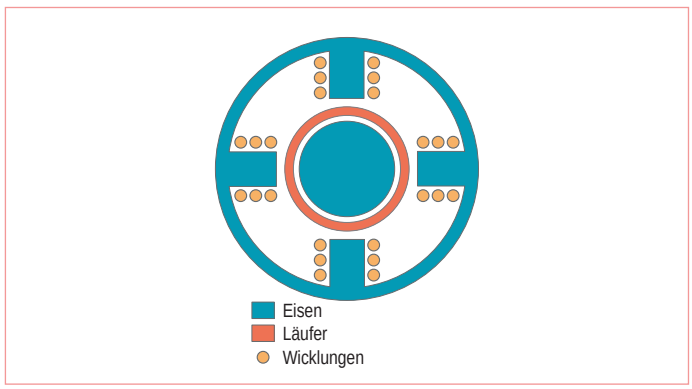


Bild 7. Aufbau eines Drag-Cup-Motors.

Bild 8. Ein Motor/Tachogenerator aus dem HSI.

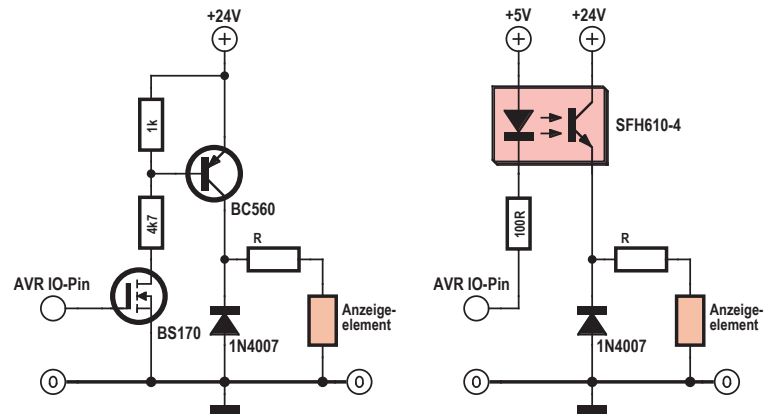


Bild 12. 24-V-Treiber.

rechteckförmig ist. Dazu wurde eine einfache Halbbrückenschaltung nach **Bild 9** entworfen. Die Ansteuersignale erzeugt der Controller, wobei zwischen dem Highside- und dem Lowside-Puls jeweils eine Totzeit liegt, damit die beiden Darlingtons nie gleichzeitig leiten.

Zwei solche Halbbrückenschaltungen versorgen dann, wie in **Bild 10** dargestellt, einen Motor. Die Drehzahl kann über die Phasenverschiebung eingestellt werden.

Tachogenerator

Interessant ist auch, wie der Tachogenerator aufgebaut ist. Er besteht aus einer „Erregerwicklung“, die mit einem 400-Hz-Sinussignal angesteuert wird. Um 90 Grad gedreht gibt es die Sensorwicklung, in die bei Motorstillstand keine Spannung induziert wird. Im Feld befindet sich nun wieder ein Kurzschlussläufer, genauso wie im Motorteil nach **Bild 7**.

Dreht sich der Rotor, so zieht er sozusagen das Erregerfeld mit. Dadurch verschwindet die Symmetrie und es wird eine Spannung in der Sensorwicklung erzeugt, die proportional zur Drehzahl ist. Da keinerlei Bürsten verwendet werden, ist dieses Konzept sehr robust.

Kurszeiger CDI

Die Anzeige für das VOR (Kurs-Signal des UKW-Drehfunkfeuers) ist ein CDI (Course Deviation Indicator) genannter Kursanzeiger. Das ist eine Nadel, die nach links oder rechts ausschlägt, und deren Ausschlag die Größe der Kursablage angibt. Technisch wird der CDI mit einem Drehspulinstrument realisiert. Um positive und negative Ströme zu realisieren, wird die in **Bild 11** gezeigte Schaltung eingesetzt. Bei einer PWM mit 50 Prozent Tastverhältnis ist die Brücke abgeglichen und der CDI steht in der Mitte. Je nachdem, ob der Dutycycle nun kleiner oder größer als 50 Prozent ist, schwenkt der CDI nach links oder rechts aus. Mit dem Trimmer kann man die Empfindlichkeit einstellen.

To/From-Anzeige

Die TO/FROM-Anzeige des VORs ist ebenfalls mit einem Drehspulinstrument realisiert, das allerdings digital angesteuert wird. Es kommt wieder die Schaltung nach **Bild 11** zum Einsatz, aber man kann statt eines PWM-Ausgangs einen TRI-State-Ausgang nehmen. Ist der Ausgang hochohmig, ist der gelieferte Strom Null, und weder das TO-Dreieck noch das FROM-Dreieck erscheinen. Gibt man nun eine Null oder eine Eins auf den Ausgang, dann kann man damit das TO- beziehungsweise FROM-Dreieck aktivieren.

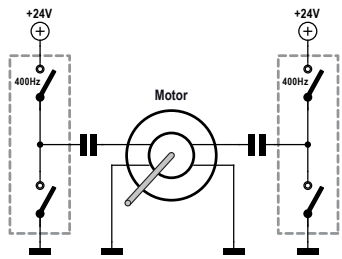


Bild 10. Ansteuerung des Zweiphasenmotors.

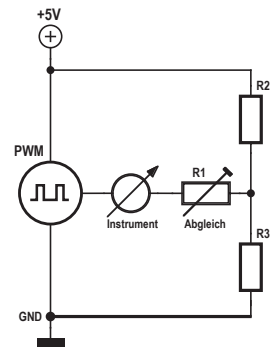


Bild 11. Brücke zur CDI-Ansteuerung.

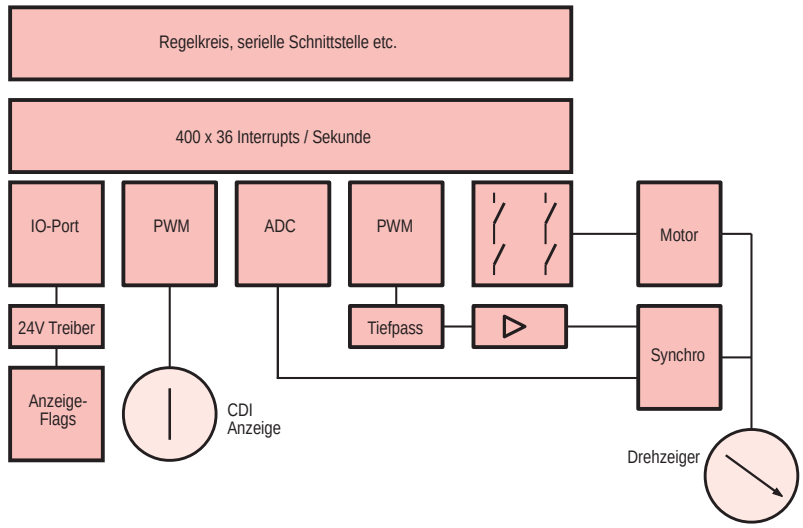


Bild 13. Systemaufbau.

Weitere Anzeigeelemente

Der HSI besitzt noch einige Anzeigelemente, die durch einfache Elektromagnete betätigt werden, zum Beispiel die roten Klappelemente in der Frontansicht am Artikelanfang. Mehrere Elemente haben teilweise eine gemeinsame Masse und benötigen mehr als 12 Volt. Um diese anzusteuern, wurden die Treiberschaltungen aus **Bild 12** verwendet.

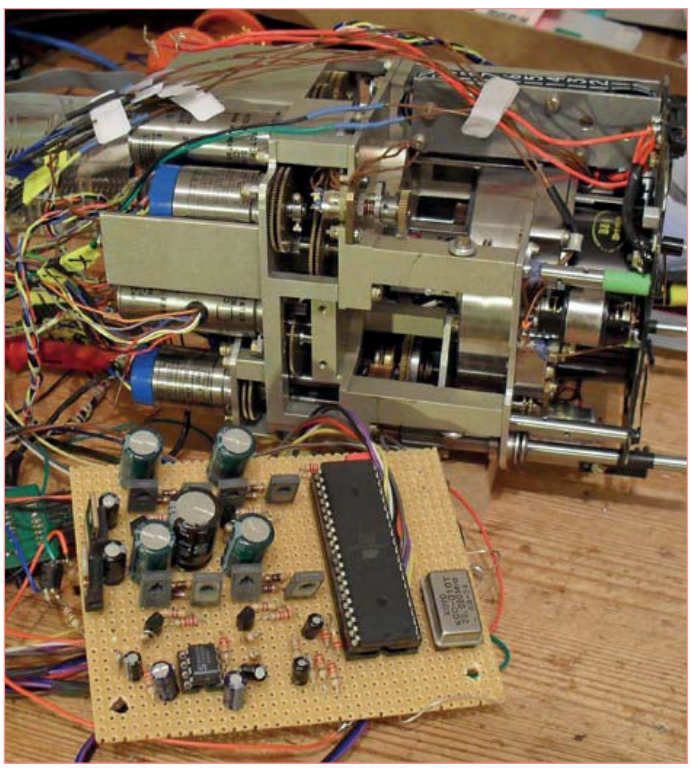


Bild 14. Testplatine.

Bei geringen Strömen reicht ein Optokoppler mit hoher Stromverstärkung. Für etwas höhere Strome wird ein BS170 als Level-Shifter und ein BC560 als Schalter verwendet. Mit dem Widerstand R kann man dann jeweils den Strom einstellen.

Fazit

Mit Hilfe der vorgestellten Schaltungen kann man nun den HSI komplett fernsteuern. Die Gesamtkonzeption ist in **Bild 13** dargestellt. Es kommt ein ATmega644 zum Einsatz. Dieser bietet neben 20 MHz Taktfrequenz auch die benötigten Timer und A/D-Wandler.
Ein erstes, auf Lochrasterplatine aufgebautes Testexemplar ist in **Bild 14** zu sehen. Die Funktion ist im eingangs erwähnten Video auf Youtube zu sehen. Der Ausstattung des eigenen Simulator-Cockpits mit echten Instrumenten steht damit nichts mehr im Wege.

(110756)

Links:

[1] www.helmut-singer.de
[2] www.elektor.de/090287
[3] http://de.wikipedia.org/wiki/Horizontal_Situation_Indicator
[4] www.youtube.com/user/ossimodding
[5] www.muirheadaerospace.com/motion-technology/synchros.html
[6] http://en.wikipedia.org/wiki/Synchro
[7] www.ddc-web.com/documents/synhdbk.pdf
[8] www.hnsa.org/doc/neets/mod15.pdf
[9] www.google.com/patents (US-Patent 3641376)
[10] http://books.google.de
(A Textbook of Electrical Machines von R. K. Rajput, Seite A-5)

Markt elektor | Elektronik & Mikrocontroller

Kontakt: Tel. 0511/61 65 95-0 – Fax 0511/61 65 95-55 – E-Mail: service@id-medienservice.de

Regelungstechnik

Theorie und Praxis mit WinFACT und Multisim

Die heutige Regelungstechnik hat Verknüpfungspunkte mit fast jedem technischen Gebiet. Ihre Anwendungen reichen von der Elektrotechnik über die Antriebstechnik und den Maschinenbau bis hin zur Verfahrenstechnik.
Will man nun die Regelungstechnik anhand der fachlichen Regeln dieser einzelnen Gebiete erklären, so müsste man von einem Regelungstechniker verlangen, jedes Fachgebiet, in dem er Regelungen vornehmen will, fundiert zu beherrschen. Dies ist aber bei dem heutigen Stand der Technik nicht möglich.
Bei der Regelung einer Antriebsaufgabe, einer Druck- oder einer Temperatur-regelung tauchen Gemeinsamkeiten auf, die man mit einer einheitlichen Vorgehens-weise beschreiben kann. Die Grundgesetze der Regelungstechnik gelten in gleicher Weise für alle Regelkreise, ganz unabhängig davon, wie verschieden sie im Einzelnen auch apparativ aufgebaut sein mögen.
Dieses Buch richtet sich an den Praktiker, der gründlicher in die Regelungstechnik eindringen möchte, auf ausschweifende theoretische Exkursionen in die Mathematik aber gerne verzichten kann.

NEU!

365 Seiten (kart.)
Format 17 x 23,5 cm
ISBN 978-3-89576-240-6
€ 49,00 CHF 60,80

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

Festplatten-Aktivität sichtbar gemacht

Zehn LEDs sagen mehr als eine!

Von Karsten Böhme (D)

Fast jedes klassische PC-Gehäuse ist mit einer LED zur Anzeige der Aktivität der eingebauten Festplatte(n) ausgestattet. Diese flackert bei Plattenzugriffen, was besser als nichts ist. So richtig gut wird die Anzeige aber erst, wenn man 10 LEDs zur Verfügung hat!

Eigenschaften

- Anzeige der Festplattenaktivität mit 10 LEDs
- Anzeige der Festplattenaktivität in 10-%-Schritten
- Absturzsichere Anzeige (keine PC-Software nötig)
- Kompaktes Modul zum Einbau in ein PC-Gehäuse

Auch wenn moderne PCs über eine enorme Rechenkraft verfügen und die Festplatten schnell und schneller geworden sind: Gerade bei Power-Usern und professionellen Anwendern mit entsprechender Software gibt es immer noch Situationen, bei denen man nicht sicher ist, ob das grade aktive Programm „hängt“ oder ob es einfach schwer beschäftigt ist und zum Beispiel gerade die Festplatte quält. In solchen Momenten ist es ausgesprochen informativ, über eine Anzeige der Festplatten-Aktivität „in Hardware“ zu verfügen, um die Geschehnisse im Innern der „Blechkiste“ einschätzen zu können.

Um so eine verbesserte Anzeige der Festplattenaktivität geht es bei diesem Selbstbauprojekt, das gegenüber einer denkbaren Lösung in Software den Vorteil hat, nicht nur keine zusätzliche Rechenleistung zu verbrauchen und unabhängig vom jeweiligen Betriebssystem zu operieren, sondern (wenn man von der Firmware absieht) auch prinzipiell bug- und absturzfrei ist. Interessanterweise scheint man auf dem riesigen Markt für Case-Modding-Zubehör keine solche Anzeige zum Nachrüsten kaufen zu können. Doch wozu gibt es Elektor?

Mehr LEDs!

Die typische, eher qualitative 1-LED-Anzeige informiert mit ihrem Flackern nur recht grob über die Aktivitäten der eingebauten Festplatte(n). Die hier vorgeschlagene Lösung erledigt dies mit ihren insgesamt zehn LEDs eindeutig quantitativ über die Intensität der Festplattenaktivität (in 10-%-Schritten). Dabei ist die Anzeige durchaus reaktionsschnell, denn sie wird fünf Mal pro Sekunde aktualisiert.

Eine spannende Frage ist, wie der Autor seine Balkenanzeige der Festplattenaktivität verwirklicht hat. Schließlich sind bei Festplatten weder Analogausgänge noch digitale Schnittstellen vorhanden, mit denen Zugriffsdaten übertragen würden. Die Lösung des Autors ist, das Geflackere der vom Motherboard angesteuerten Gehäuse-HDD-LED als eine Art pulsbreitenmoduliertes Signal zu betrachten. Das funktioniert erstaunlich gut. Hierzu wird das Flackersignal vom Motherboard abgegriffen und einfach gemessen, wie groß der „High“-Anteil während der Messzeit von 200 ms ist. Bei 100 % „high“ leuchten selbstverständlich alle zehn LEDs auf und bei 0 % logischerweise keine einzige. Das Foto der im PC des Autors eingebauten Anzeige zeigt alle zehn LEDs aktiv. Für die ersten sieben LEDs (Bereich 1...70 %) sind grüne Exemplare vorgesehen. Die beiden gelben LEDs zeigen eine Auslastung der Festplatte für den Bereich 71...90 % an und die rote LED leuchtet bei einer Aktivität zwischen 91 und 100 %.

Decoder und LED-Treiber

Das beschriebene Funktionsprinzip verlangt fast zwingend nach einem Mikrocontroller. Und wenn man auf die ziemlich übersichtliche Schaltung von **Bild 1** blickt, findet man außer einem einfachen Atmel-Mikrocontroller vom Typ ATtiny2313 nur noch einen Optokoppler, LEDs und wenige weitere Bauelemente. Das Flackersignal wird schlicht von den Pins des Motherboards abgegriffen, auf die normalerweise der Stecker für die HDD-LED des Gehäuses gesteckt wird. Die Signalübertragung wird elegant mit einem Optokoppler erledigt, was als adäquater LED-Ersatz gleichzeitig jedes Pegelanpassungsproblem vermeidet. Damit die Gehäuse-LED nun nicht arbeitslos wird, verfügt die Schaltung über einen extra

Leserprojekte sind Beiträge von Elektor-Lesern für experimentelle Zwecke oder zur Anregung für andere Leser.
Die in dieser Rubrik vorgestellten Schaltungen wurden vom Elektor-Labor nicht auf Reproduzierbarkeit und Funktion getestet.

Ausgang, über den sie vom Mikrocontroller so angesteuert wird, als würde sie direkt mit dem Motherboard verbunden sein.
IC1 treibt die zehn LEDs über einfache 470-Ω-Vorwiderstände. Mit anderen Controllern und entsprechend angepasster Firmware könnte man noch mehr LEDs treiben. Man muss lediglich drauf achten, dass man weder den individuellen Maximalstrom pro Pin (hier 10 mA) noch den zulässigen Gesamtstrom für alle Pins (hier 60mA) wesentlich überschreitet. Wenn man eine sehr helle Anzeige will (was an einem PC allerdings eher stört), dann empfiehlt sich die Verwendung von so genannten Low-Current-LEDs.

Die komplette Schaltung wird einfach von einer der möglichen 5-V-Spannungsquellen (Kabel für Festplatten oder Floppy-Drive) versorgt. Zur Programmierung des Controllers ist der Atmel-typische zehnpolige Pinheader vorgesehen.

Auf- und Einbau

Der Autor hat für seine Anzeige eine Platine entworfen, die nicht nur die Elektronik, sondern auch die LEDs enthält. Wie man an den Layouts für Oberseite (Datei PCB_top.jpg unter [1]) und Unterseite (Datei PCB_top.jpg unter [1]) der Platine sieht, geht es trotz SMD-Bauteilen recht luftig zu. Für die Widerstände hat der Autor ein noch gut von Hand zu lötendes 1206-Gehäuse vorgesehen. Besondere Schwierigkeiten sollte die Bestückung also nicht machen. Alle Bauteile bis auf die LEDs kommen auf die Oberseite. Die LEDs platziert man in passendem Abstand auf der Unterseite, sodass man die Platine als kompaktes Anzeigemodul mit Hilfe von Abstandshaltern von innen an eine passende Stelle des PC-Gehäuses schrauben kann, in das man zuvor die zehn Löcher für die LEDs gebohrt hat. Ein Ausdruck des Layouts auf Papier im Maßstab 1:1 kann als Bohrschablone die mechanischen Arbeiten vereinfachen. Man kann es allerdings auch so wie der Autor machen (siehe Prototype.jpg unter [1]) und die LEDs abgewinkelt einlöten, sodass sie auf einer Stirnseite der Platine angeordnet sind.
Beim Flashen der Firmware sollte man die Fuses so einstellen, dass der interne Taktgenerator mit 4 MHz verwendet wird. Ohne diese Maßnahme läuft der Controller deutlich langsamer, was die Aktualisierungsfrequenz beeinflusst. **Bild 2** zeigt einen Screenshot der Takt-Fuses bei Verwendung von Atmels Entwicklungsumgebung AVR-Studio 4.

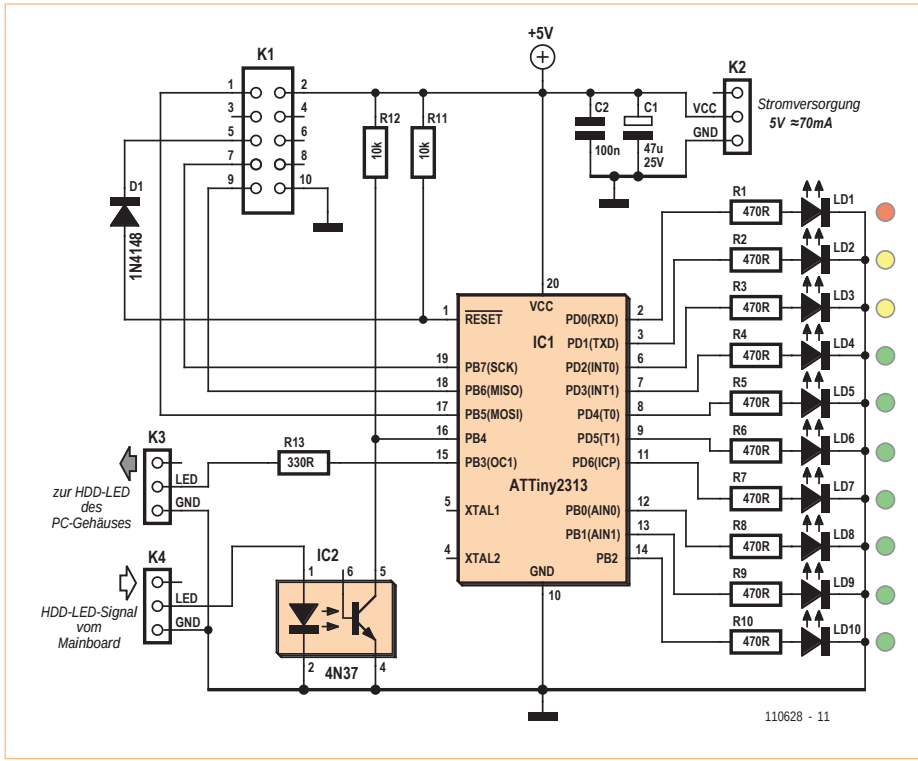


Bild 1. Die Schaltung der Festplattenaktivitätsanzeige ist recht übersichtlich.

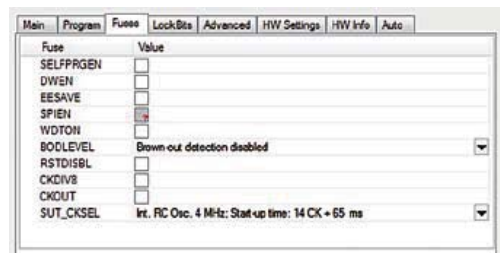


Bild 2. Die Fuses für einen internen Takt von 4 MHz mit AVR-Studio 4.

Die Firmware selbst wurde vom Autor mit dem C-Compiler CodeVison geschrieben und ausreichend kommentiert, sodass Änderungen leicht fallen. Sowohl der C-Source-Code als auch die fertig kompilierte Hex-Datei der Firmware können von der Elektor-Webseite zu diesem Artikel [1] kostenlos herunter geladen werden. Aus derselben Quelle lassen sich die Layout-Dateien im Eagle-Format beziehen. Außerdem kann man sich dort ein vom Autor aufgezeichnetes Video seiner Anzeige im Betrieb herunterladen.

(110628)

Weblink

[1] Video, Software und Layout-Dateien des Autors: www.elektor.de/110628

Elektors „Einfacher Funktionsgenerator“ (1978)

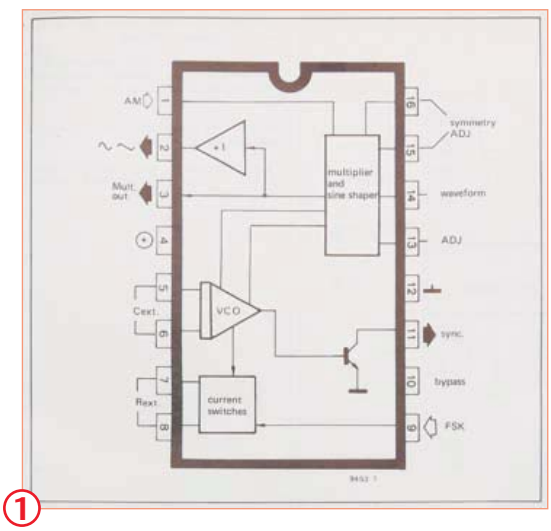
Von Jan Buiting (Elektor UK/US)

Wenn man schaut, was in einem (nicht überwiegend auf Mikrocontroller fokussierten) Elektronik-Labor dauernd eingeschaltet ist, dann wird an erster Stelle ein Lötkolben vor sich hin heizen und dann noch ein einstellbares Labornetzteil aktiv sein. An nächster Stelle kommt wohl ein Oszilloskop und dann gleich ein Funktionsgenerator (oft wird hierfür ein PC zweckentfremdet). Auf jeden Fall ist so ein Generator ein Problemlöser schlechthin, denn zwischen seinem Ausgang und dem Eingang des Oszilloskop klemmt normalerweise ein „Etwas“, das

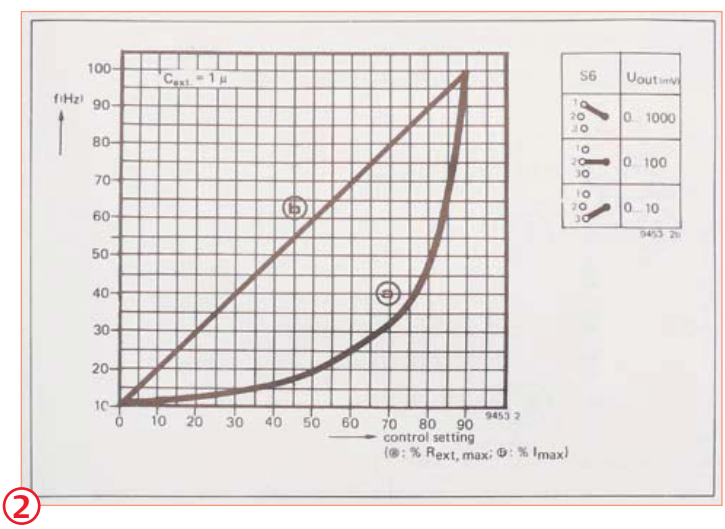
- ☐ kaputt ist,
- ☐ nicht richtig funktioniert,
- ☐ erforscht werden soll,
- ☐ gerade entwickelt wird,
- ☐ seine Funktion nicht gerne preisgibt,
- ☐ für Freunde „mal angeschaut“ werden soll,
- ☐ qualmt / stinkt / blinkt / knistert oder
- ☐ das man von eBay gebraucht (= defekt) ersteigert hat.

– war das Oszilloskop für Arme. Bei komplexeren Aufgaben benötigten Service-Techniker und professionelle Elektroniker aber schon damals zusätzliche Wellenformen mit verschiedenen Frequenzen bei einstellbarem Ausgangspegel. Ebenfalls wichtig waren die Signalstabilität und geringe Verzerrungen beim Sinussignal. Diese Anforderungen erfüllte nur Profi-Equipment zu entsprechend gesalzenen Preisen. Auch wenn es ab Mitte der 1970er Jahre viele gebrauchte Funktionsgeneratoren mit Röhrenbestückung zu kaufen gab: Diese waren nicht nur voluminös, sondern immer noch reichlich teuer. Solche Geräte waren also außer Reichweite von Hobby-Elektronikern und kleinen Werkstätten.

Es war mir nicht möglich zu klären, ob der Begriff „Funktionsgenerator“ tatsächlich von den mathematischen Funktionen entlehnt wurde, die zur Ableitung von Signalformen wie Sinus, Dreieck oder Sägezahn aus einer basalen Rechteckwelle nötig sind. Außerdem gibt es Funktionsgeneratoren, die ihre verschiedenen Signalformen von einer Dreieckwelle ableiten. Ein älterer Zeitgenosse erklärte mir: „Jan, diese Apparate haben einfach viele Funktionen.“



①



②

Normalerweise reagiert so ein TuT (Thing under Test) irgendwie auf die Signale des Funktionsgenerators und gibt so Hinweise auf seine Funktion, Fehler oder Defekte. Laut dem legendären Analog-Entwickler Bob Pease [1] sagt der Bildschirm eines Oszilloskops mehr als tausend Worte. An dieser Stelle scheiden sich wahre Elektroniker von Normalsterblichen: Erstere sind fasziniert und Letztere gehen fernsehen.

Historisch und funktional ist der Funktionsgenerator der luxuriöse Nachfolger des alten „Signalinjektors“, der typischerweise ein NF-Rechtecksignal von einigen hundert Hertz lieferte. Ein passender „Signalverfolger“ - nichts anderes als ein empfindlicher Verstärker

Um die Jahre 1974/75 war schließlich ein IC erhältlich, das nahezu alle „Funktionen“ integriert hatte, um damit einen leistungsfähigen und mit vielen „Funktionen“ ausgestatteten Funktionsgenerator zu niedrigen Kosten zu realisieren. Das IC XR2206 von Exar (**Bild 1**) war sehr schnell sehr bekannt. Auch heute, nach immerhin 38 Jahren, kann man es noch bei Conrad Elektronik kaufen! Und im Jahre 1977 nahm sich schließlich der Elektor-Redakteur Ernst Krempelsauer (mittlerweile pensioniert, siehe **Kasten**) das reichlich trockene Datenblatt zum XR2206 vor und verwandelte es in ein außerordentlich erfolgreiches Selbstbauprojekt, das unter der Nummer EPS 9453 in den folgenden Jahren von tausenden Lesern nachgebaut wurde.

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

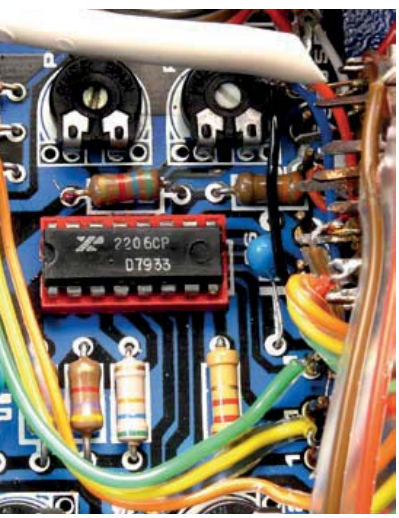
Und neulich erst sah ich in einem YouTube-Video [2], wie jemand eine hastig zusammengebaute Version davon zur Demonstration des Oszilloskops DSO 062 von JYE Tech eingesetzt hat. Die in Elektor veröffentliche Basis-Schaltung mit dem XR2206 findet sich auch auf der Webseite RadioMuseum.org [3], was wohl als Zeichen besonderer Anerkennung gewertet werden kann.
Ein ganz besonderer Trick von Ernsts Entwicklung ist sicherlich die Art, wie er ein gewöhnliches Poti mit einigen Widerständen so verschaltet hat, dass damit eine fast lineare Frequenzeinstellung mit dem XR2206 möglich wurde. Die relevanten Schaltungsteile haben wir in den **Bildern 2** und **3** nochmals abgedruckt. Mit einer konventionellen Beschaltung würde sich eine exponentielle Einstellung ergeben, der man mit einem negativ logarithmischen Poti hätte begegnen können. Das wäre zwar etwas für Mathematiker, aber nichts für die Praxis. Wenn man nämlich einfach und stur ein Poti zwischen die Pins 7 und 8 des XR2206 legt, ergibt sich:

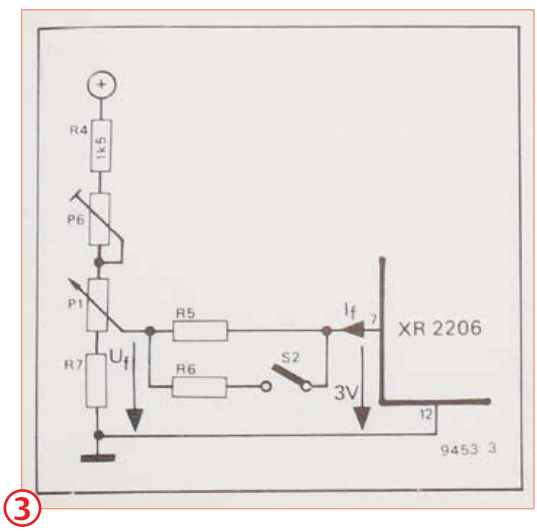
$f = 1 / (R_{ext} C_{ext})$

Mit dem I/U-Konverter à la Krempelsauer am Pin I_f ergibt sich aber:

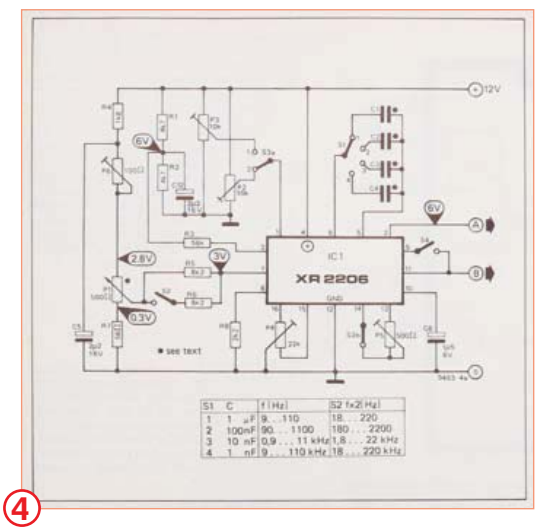
$f = (3 - U_f) / (3 R_s C_{ext})$

Die kompletten Details kann man im originalen Artikel [4] nachlesen. Ergebnis ist jedenfalls, dass der Einstellbereich des linearen Potis P1 auf eine praktisch lineare Frequenzänderung über etwas mehr als eine Dekade (beispielsweise 9...110 Hz) hinausläuft. Als Zugabe kann man die eingestellte Frequenz durch Schließen von S2 einfach verdoppeln (solange R5 = R6).
Von den drei Teilschaltungen des Funktionsgenerators wird hier die relevante Beschaltung des XR2206 in **Bild 4** gezeigt. Das Netzteil mit dem integrierten 12-V-Regler L130 ist konventionell und zum transistorisierten Ausgangsverstärker gibt es lediglich zu sagen, dass er den Ausgang recht niederohmig macht. Interessanterweise ist der Ausgang mit „AC“ beschriftet, was zwar technisch korrekt ist (um Wechselspannung handelt es sich ja), doch assoziiert man im englischen Sprachraum zu diesem Akronym eher Netzspannung.
Die Frequenz-Kalibrierung war erstaunlich einfach, denn sie nutzte die 100 Hz vom Brückengleichrichter des Netzteils. Dieser Netzbrumm wurde zusätzlich in den Ausgangsverstärker eingespeist und die Frequenz so eingestellt, dass sich eine Schwebung ergab, die gegen 0 Hz tendierte.
Das abgebildete Muster des einfachen Funktionsgenerators wurde unter eklatanter Verletzung der Anweisung „den alten Kram zu entsorgen“ aus dem früheren Büro gerettet. Es gäbe einige Geschich-





3



4

Ernst Krempelsauer erinnert sich…

Ich war gerade erst etwas über ein Jahr bei Elektor und wider Erwarten schon stellvertretender Chefredakteur, als ich im Juni 1975 ein erstes Exemplar des Funktionsgenerator-ICs XR2206 von Exar ergattern konnte. Es gab zwar mit dem ICL 8038 von Intersil schon 1973 ein auch von Elektor verwendetes Funktionsgenerator-IC, doch der 2206 war um Klassen besser. Nach Datenblattstudium und gründlichem Ausprobieren entstand dann ein Artikel mit zahlreichen Schaltbeispielen, der in Elektor September 1975 veröffentlicht wurde. Dabei hatte ich schon die Möglichkeit der linearen Frequenzeinstellung über eine Spannungssteuerung ausführlich beschrieben.

Die in der Folge auf dem Markt erhältlichen Funktionsgeneratoren mit dem 2206 verwendeten aber alle die in der Exar-Applikationsschaltung angegebene Stromeinstellung der Frequenz mit einem Poti als variabler Widerstand, was zu einem exponentiellen Skalenverlauf führt. Mir schwebte aber ein Funktionsgenerator mit einer linearen Frequenzskala vor, der in Verbindung mit dekadischen Frequenzbereichen auch ohne Frequenzzähler eine relativ genaue Frequenzeinstellung ermöglicht.

Allerdings war ich im darauffolgenden Jahr mit dem umfangreichen Formant-Synthesizer-Projekt (neben der Redaktionsarbeit!) gut ausgelastet, so dass der Funktionsgenerator erst mal warten musste. Dafür sollte es auch ein möglichst praxistauglicher Entwurf mit möglichst einfachem Aufbau und Abgleich und optimalem Aufwand/Nutzen-Verhältnis werden. Dazu gehörten auch Spannungsangaben an fast allen Punkten des Schaltplans und ein Frontplattenentwurf mit einem klassischem Design, das von der Optik der Formant-Module inspiriert war. Auch die Optimierung der Platine nahm etwas mehr Zeit in Anspruch, so dass die sehr ausführliche Bauanleitung des „Einfachen Funktionsgenerators“ dann (endlich!) im Oktoberheft 1977 erscheinen konnte. Offenbar immer noch zum richtigen Zeitpunkt, denn es wurden tausende Platinen verkauft und neben Bausätzen sogar (nicht lizenzierte) Fertiggeräte angeboten…

ten über das mit einem transparenten Plexiglas-Deckel versehene Exemplar zu erzählen, das für Ausstellungen und Messen gedacht war. Die Aufkleber und Zettel auf der Unterseite des Geräts bezeugen eine exzessive Reisetätigkeit und das Überqueren vieler Grenzen (damals waren europäische Grenzen noch durchaus real). Ein vom Ausgang zum eingebauten Lautsprecher führendes Kabel war durchgeschnitten. Dies war wohl Notwehr der auf Messen akustisch gequälten Elektor-Mitarbeiter, denn sicher wollte jeder Interessierte einmal ausgiebig an den Knöpfen drehen…

Wenn man genau hinschaut, kann man weitere lose Drahtenden erkennen. Auch das essentielle IC XR2206 war verschwunden. Doch zu meiner Rettung konnte Kollege Luc Lemmens ein solches IC aus einem Playtronic-WOG-2206-Signalgenerator von 1980 entwenden. Übrigens erinnert das Playtronic-Gerät verdächtig an das Elektor-Projekt (siehe **Bild 5**). Eine halbe Stunde später lief der Funktionsgenerator wieder wie in alten Zeiten.

Wie es der Zufall wollte erschien 1984 eine neue Variante des Funktionsgenerators in Elektor. Diese optimierte Version beseitigte Probleme mit kleinen Spikes in der Sinuskurve. Es gab noch weitere Verbesserungen, doch im Kern war es die alte Schaltung mit dem XR2206. Soweit ich in Erfahrung bringen konnte, kam dieser Entwurf ähnlich gut an wie die 1977er Version.

Heute können PCs, Laptops und sogar Smartphones mit entsprechender Software für die eingebaute Sound-Hardware in einem Elektronik-Labor als Funktionsgenerator dienen. Bezüglich Stabilität, Genauigkeit und Verzerrungen sind solche Lösungen sogar richtig gut. Ich für meine Person ziehe es aber immer noch vor, an richtigen Knöpfen realer Mess- und Testgeräte zu drehen. Und wenn ich an die enorme Resonanz der beiden Elektor-Funktionsgeneratoren von 1977 und 1984 denke, bin ich damit nicht allein.

(120068)

Zur Feier der Ausgrabungsarbeiten kann man sich eine eingescannte Kopie des originalen Artikels zum einfachen Funktionsgenerator aus Elektor Oktober 1977 kostenlos von [4] downloaden. Leider sind heute weder passende Platinen lieferbar noch können wir technische Unterstützung für das Projekt geben.

Weblinks

[1] http://en.wikipedia.org/wiki/Bob_Pease

[2] www.youtube.com/watch?v=KPtsgFw5Fno

[3] www.radiomuseum.org/tubes/tube_xr2206.html

[4] www.elektor.de/120068

⑤

elektor

Elektor Print

Gewohnter Lesespaß auf Papier

+

Elektor Digital

Neuer Lesespaß auf PC, Notebook oder Tablet

=

Elektor PLUS

Ultimativer Lesespaß zu Hause oder unterwegs

Lesen Sie Elektor im vorteilhaften PLUS-Abonnement!

Jetzt abonnieren oder upgraden: www.elektor.de/abo

Hexadoku

Sudoku für Elektroniker

Wie viele verschiedene Hexadokus sind bei 16 x 16 Ziffern möglich? Wir wissen das nicht genau, aber es sind in jedem Fall ziemlich viele... Hier kommt eine Variante – und die haben Sie bestimmt noch nie irgendwo gesehen. Füllen Sie überall die richtigen Zahlen in die Kästchen und schicken Sie uns die Lösung zu, denn hier warten vier schöne Gutscheine auf die Gewinner!

Die Regeln dieses Rätsels sind ganz einfach zu verstehen: Bei einem Hexadoku werden die Hexadezimalzahlen 0 bis F verwendet, was für Elektroniker und Programmierer ja durchaus passend ist.
Füllen Sie das Diagramm mit seinen 16 x 16 Kästchen so aus, dass **alle** Hexadezimalzahlen von 0 bis F (also 0 bis 9 und A bis F) in jeder Reihe, jeder Spalte und in jedem Fach mit 4 x 4 Kästchen (markiert durch die dickeren schwarzen Linien) **genau einmal** vorkommen.
Einige Zahlen sind bereits eingetragen, was die Ausgangssituation des Rätsels bestimmt.
Wer das Rätsel löst - sprich die Zahlen in den grauen Kästchen herausfindet - kann wie jeden Monat einen Hauptpreis oder einen von drei Trostpreisen gewinnen!

Mitmachen und gewinnen!

Unter allen internationalen Einsendern mit der richtigen Lösung verlosen wir
einen **ELEKTOR-Gutschein** im Wert von 100 €
und
drei **ELEKTOR-Gutscheine** im Wert von je 50 €.

Einsenden

Schicken Sie die Lösung (die Zahlen in den grauen Kästchen) per E-Mail, Fax oder Post an:
Elektor – Redaktion Süsterfeldstr. 25 52072 Aachen
Fax: 0241 / 88 909-77 E-Mail: hexadoku@elektor.de
Als Betreff bitte nur die Ziffern der Lösung angeben!
Einsendeschluss ist der 31. März 2012!

Die Gewinner des Hexadokus aus dem Januarheft stehen fest!

Die richtige Lösung ist: 43ADE.
Der Elektor-Gutschein über 100 € geht an: Christian Klems aus Nijkerk (NL).
Einen Elektor-Gutschein über je 50 € haben gewonnen: Marc Herzog, Raúl Elguezabal Martínez und Antje Völksch.
Herzlichen Glückwunsch!

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

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

Subscribe to *audioXpress* magazine!

Do your **electronics speak** to you? Are the words **"audio", "vacuum tubes",** and **"speaker technology"** music to your ears?

Then you should be reading *audioXpress!*

Recently acquired by The Elektor Group, *audioXpress* has been providing engineers with incredible audio insight, inspiration and design ideas for over a decade. If you're an audio enthusiast who enjoys speaker building and amp design, or if you're interested in learning about tubes, driver testing, and vintage audio, then *audioXpress* is the magazine for you!

What will you find in *audioXpress*?

- In-depth interviews with audio industry luminaries
- Recurring columns by top experts on speaker building, driver testing, and amp construction
- Accessible engineering articles presenting inventive, real-world audio electronics applications and projects
- Thorough and honest reviews about products that will bring your audio experiences to new levels

Choose from print delivery, digital, or a combination of both for maximum accessibility.

Subscribe to *audioXpress* at
www.cc-webshop.com
today!

audioXPRESS

Starke Stücke

Die ganze Welt der Elektronik in einem Shop!

Bücher

Von 0 und 1 zum FPGA

Digitale Logik selbst entwickeln

Dieses Buch nimmt Sie mit auf eine Entdeckungsreise in die Welt der digitalen Elektronik. Nach dem Aufbau einer soliden Wissensbasis hierüber verlagert sich der Schwerpunkt zur programmierbaren Logik. Wie lassen sich mit vorhandenen Bausteinen logische Systeme aufbauen und wie koppelt man sie sicher und störungsfrei an die analoge Außenwelt? Das sind Fragen, die das Buch beantwortet. Moderne logische Systeme sind so komplex, dass deren Aufbau mit separaten Bauelementen wie einzelnen Gattern, Flipfl ops, Zählern oder Teilern eine fast unmögliche Aufgabe ist. Deshalb beschäftigt sich das Buch in der zweiten Hälfte mit der programmierbaren Logik in Form von FPGA- und CPLD-ICs. Sie lernen Methoden kennen, die als Brücke zwischen dem klassischen Design und der zeitgemäßen Entwicklung mit FPGA fungieren. Neben dem schematischen FPGA-Entwurf setzt sich das Buch auch mit Sprachen wie Verilog und VHDL auseinander. Das abschließende Projekt, eine Uhr mit Alarmfunktion in Verilog und VHDL, zeigt den praktischen Nutzen der programmierbaren Logik.

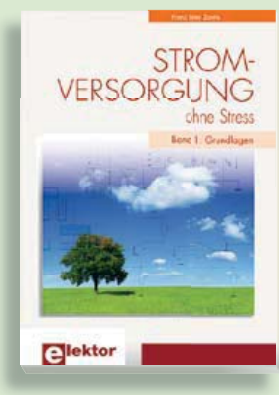
603 Seiten (kart.) • ISBN 978-3-89576-254-3 • € 49,00 • CHF 60,80

Kreative Lösungen aus allen Bereichen der Elektronik

311 Schaltungen

Das mittlerweile zwölfte Buch aus Elektors erfolgreicher „Dreihunderter"-Schaltungsreihe bietet ein weiteres Mal neue Konzepte sowie einen (fast) unerschöpfl ichen Fundus zu allen Bereichen der Elektronik: Audio & Video, Spiel & Hobby, Haus & Hof, Prozessor & Controller, Messen & Testen, PC & Peripherie, Stromversorgung & Ladetechnik sowie zu Themen, die sich nicht katalogisieren lassen.

544 Seiten (kart.) • ISBN 978-3-89576-255-0
€ 36,80 • CHF 45,70

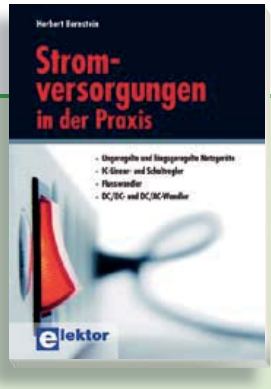


Band 1: Grundlagen

Stromversorgung ohne Stress

Die Funktion alle elektronischen Schaltungen und Geräte steht und fällt mit der Stromversorgung. Schon deshalb muss man dieser Baugruppe besondere Aufmerksamkeit widmen. Dieses Buch beinhaltet Grundlagen und Schaltungen der Stromversorgungstechnik für elektronische Geräte aus der Praxis. Dem aktuellen Trend folgend hat der Autor der mobilen Stromversorgungstechnik und der Schaltnetzteiltechnik besondere Aufmerksamkeit gewidmet.

294 Seiten (kart.) • ISBN 978-3-89576-248-2
€ 38,00 • CHF 47,20

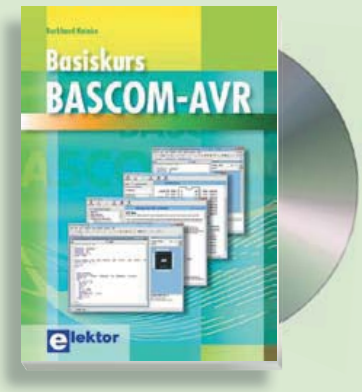


„Die Spannung steigt!“

Stromversorgungen in der Praxis

Die Elektronik bestimmt unser tägliches Leben mehr denn je – Tendenz steigend. Dabei benötigen alle elektronischen Geräte und Systeme eine gut funktionierende Stromversorgung mit spezieller Anpassung an die Betriebsbedingungen. Dieses neue Buch beschreibt die entsprechenden Möglichkeiten vom Transformator bis zum passenden Kühlkörper. Behandelt werden die wichtigsten Merkmale, Einsatzmöglichkeiten und das Betriebsverhalten von vielen unterschiedlichen Stromversorgungsgeräten.

366 Seiten (kart.) • ISBN 978-3-89576-239-0
€ 46,00 • CHF 57,10

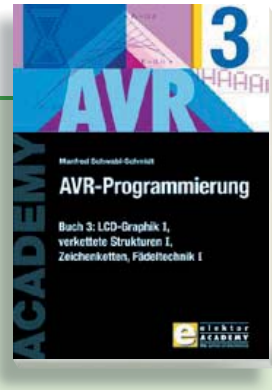


Von Top-Autor und Entwickler Kainka

Basiskurs BASCOM-AVR

BASCOM und AVR-Controller sind ein starkes Team! Was immer man entwickeln möchte, meist hat ein ATmega schon das Wichtigste an Board: Ports, Timer, AD-Wandler, PWM-Ausgänge und serielle Schnittstelle, RAM, Flash-ROM und EEPROM, alles ist reichlich vorhanden. Und BASCOM macht die Anwendung zu einem Kinderspiel. Auch komplexe Peripherie wie LCD, RC5 und I²C lassen sich mit wenigen Befehlen nutzen.

223 Seiten (kart.) • inkl. Software-CD
ISBN 978-3-89576-238-3 • € 39,80 • CHF 49,40



LCD-Graphik I, verkettete Strukturen I, Zeichenketten, Fädeltechnik I

AVR-Programmierung 3

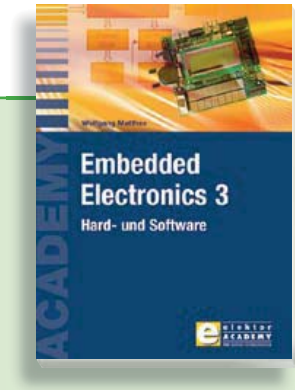
Dieser dritte Band der Buchreihe zur Assembler-Programmierung von AVR-Mikrocontrollern richtet sich nicht nur an Einsteiger. Auch die C-Programmierer von AVR-Prozessoren profitieren von der Erläuterung der Besonderheiten, die es bei der Assembler-Programmierung zu beachten gilt. Nach der Erläuterung der statischen Datenstrukturen in Buch 2 folgt in diesem Buch der Einstieg in die dynamischen Strukturen. Er beginnt nach einer allgemeinen Einführung mit der einfachsten Struktur, der verketteten Liste. Der letzte Teil führt in den Selbstbau von Fädelsprachen ein, die ein äußerst personalisiertes und projektorientiertes Programmieren erlauben.

319 Seiten (kart.) • ISBN 978-3-89576-231-4
€ 46,00 • CHF 57,10

Weitere Informationen zu unseren Produkten sowie das gesamte Verlagssortiment finden Sie auf der Elektor-Website:

www.elektor.de

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



Der 3. Band der neuen Buchreihe

Embedded Electronics 3

Die Bände Reihe wenden sich an jene, die sich – als Auszubildende, Studierende oder Berufseinsteiger – von A bis Z in die professionelle Schaltungs- und Systementwicklung einarbeiten wollen. Sie bieten, was Praktiker und Lernende brauchen: Eine Auffrischung und Vertiefung der Grundlagen, Anregungen zu eigenen Gedanken und Zugänge zu Einzelheiten, Querverbindungen und Spitzfindigkeiten.

412 Seiten (kart.) • ISBN 978-3-89576-185-0
€ 49,00 • CHF 60,80

Visual Studio

C# 2010 Programmierung und PC-Anbindung

Ziel dieses Buches ist, auf einfache Weise zu zeigen, wie mit der populären Hochsprache C# ein PC programmiert werden kann. Am Anfang beschreibt das Buch Datentypen und Programmsteuerungen, die dann um fortschrittliche Konzepte wie die objektorientierte Programmierung, Threads, die Internetkommunikation und Datenbanken erweitert werden. Alle verwendeten Code-Beispiele können kostenlos von der Elektor-Webseite heruntergeladen werden.

349 Seiten (kart.) • ISBN 978-3-89576-244-4
€ 44,00 • CHF 54,60

CD- & DVD-ROMs

Kompletter Elektor-Jahrgang 2011 auf DVD

Elektor-DVD 2011

Die neue Elektor-Jahrgangs-DVD enthält alle Artikel des Jahrgangs 2011. Sie verfügt über eine sehr übersichtlich gestaltete Benutzeroberfläche. Mit der Elektor-DVD 2011 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 978-90-5381-276-1 • € 27,50 • CHF 34,10

Der Logikbaukasten für Jederman

Workshop-DVD „FPGA“

In diesem Workshop werden Sie in die Lage versetzt, auf die neuen Möglichkeiten einzugehen. Sie lernen den Aufbau und die Funktionsweise von FPGAs (*Field Programmable Gate Arrays*) kennen und sind danach in der Lage, den für Sie geeigneten Hersteller und Bausteintyp auszuwählen. Sie werden mit einem Schaltplan-Eingabewerkzeug arbeiten und erfahren, welchen „Logikbaukasten“ der FPGA-Hersteller für Sie in seinen Bibliotheken bereithält.

ISBN 978-3-89576-259-8 • € 29,80 • CHF 37,00

AndroPod

(Elektor Februar 2012)

Android-Smartphones und -Tablets sind aufgrund ihrer Ausstattung (hochauflösender Touchscreen, Rechenleistung satt, WLAN und Telefoniefunktionen) geradezu prädestiniert, als Schaltzentrale in eigenen Projekten eingesetzt zu werden. Bisher war es allerdings nicht ganz einfach, die Geräte an externe Elektronik anzuschließen. Mit der Interface-Platine „AndroPod“, die einen Seriell-TTL- und einen RS485-Ausgang mitbringt, ist dies nun möglich.

Bestückte und getestete Platine mit RS485-Erweiterung

Art.-Nr. 110405-91 • Preis: www.elektor.de

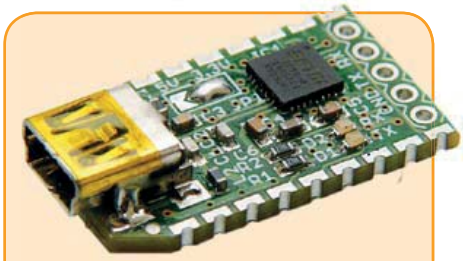
Verbesserter Strahlungsmesser

(Elektor November 2011)

Für die Messung radioaktiver Strahlung braucht man nicht viel mehr als eine PIN-Fotodiode und den passenden Sensorverstärker. Bei diesem Projekt handelt es sich um einen optimierten Vorverstärker mit einem Mikrocontroller-Zähler. Der Controller übernimmt auch gleich die Zeitmessung und zeigt die Impulsrate in „counts per minute“ an. Das Gerät kann mit unterschiedlichen Sensoren für Gamma- und Alphastrahlung verwendet werden. Es eignet sich gut für Langzeitmessungen und für Untersuchungen an schwach strahlenden Proben.

Bausatz mit allen Bauteilen inkl. Platine, Display und progr. Controller

Art.-Nr. 110538-71 • € 39,95 • CHF 49,60



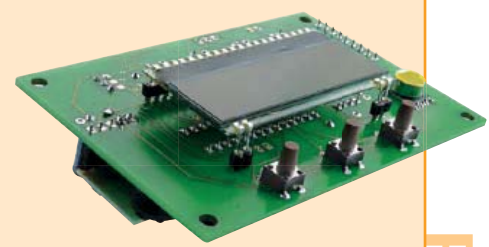
USB-FT232R Breakout-Board

(Elektor September 2011)

Dieser USB-nach-TTL-Wandler ist nicht viel größer als der angegossene Stecker eines USB-Kabels. Seine nützlichen Dienste kann die Mini-Platine unter Windows, Linux und anderen Betriebssystemenentfalten. Mit dem praktischen Konverterboard lassen sich eigene Schaltungen einfach um einen USB-Anschluss erweitern und leicht USB/RS232- oder RS485-Wandler realisieren. Die Platine ist überall dort praktisch, wo TTL erwünscht, aber nur USB vorhanden ist.

Bestückte und getestete Platine

Art.-Nr. 110553-91 • € 15,00 • CHF 18,60



USB-Wetterlogger mit Langzeitspeicher

(Elektor September 2011)

Dieser autonome Datenlogger zeichnet die von I²C-Sensoren gelieferten Daten für Luftdruck, Temperatur und Feuchte auf und zeigt sie auf einem LC-Display an. Die Ergebnisse lassen sich über USB auslesen und mit GNUplot auf einem PC grafisch darstellen. Dank der digitalen Sensormodule ist der Hardwareaufwand gering und ein Abgleich nicht erforderlich. Die Betriebsdauer mit drei Mignonzellen beträgt sechs bis acht Wochen.

Kit bestehend aus Platine, prog. Controller, Feuchte- und Luftdrucksensor

Art.-Nr. 100888-73 • € 34,95 • CHF43,40

Bausätze & Module

Lieferprogramm

März 2012 (Nr. 495) €

+++ Das Lieferprogramm zu dieser Ausgabe finden Sie auf www.elektor.de +++

Februar 2012 (Nr. 494)

AndroPod (1)
110405-91 ..... Bestückte und getestete Platine mit RS485-Erweiterung.....www.elektor.de
110258-91 ..... USB/RS485-Konverter (bestückt und getestet)..... 24,95
110553-91 ..... USB/TTL-Konverter BOB (bestückt und getestet) 15,00
120103-92 ..... USB-Kabel Micro-B auf USB-A (1,8 m) 3,95
120103-94 ..... Netzadapter mit Micro-B-USB-Stecker 5V / 1A (5W)..... 8,95

Pico C-Plus und Pico C-Super
110687-41 ..... Progr. Controller ATtiny2313-20PU (Pico C-Plus)..... 4,95
110687-42 ..... Progr. Controller ATtiny2313-20PU (Pico C-Super) 4,95

Zurück zu den Wurzeln (2)
ELEX-1 ..... Experimentier-Platine Elex-1 Größe 1 5,50
ELEX-2 ..... Experimentier-Platine Elex-2 Größe 2 9,95

Januar 2012 (Nr. 493)

Interface für Breitband-Lambdasonde
110363-41 .....Progr. Controller ATmega8-16AU 9,95

Netzlupe
110461-41 .....Progr. Controller AT89C2051-24PU, 50 Hz (Europa) 9,95

Hier kommt der Bus (11)
110258-1 .....Platine (Experimental-Knoten) 5,95
110258-1C3 ...3 x Platine (Experimental-Knoten) 12,95
110258-91 .....USB/RS485-Konverter (bestückt und getestet) 24,95

Audio-DSP-Kurs
110002-71 .....Teilbestückte Platine mit allen SMD-Bauteilen inkl. Steckverbindern und LED-Balkenanzeige 49,95

Dezember 2011 (Nr. 492)

USB-Stick am Mikrocontroller
110409-1 ..... Platine 10,95
110409-41 ..... Progr. PIC24FJ64GB002-I/sp dil-28s..... 14,95

Hier kommt der Bus (10)
110258-1 ..... Platine (Experimental-Knoten)..... 5,95
110258-1C3 ... 3 x Platine (Experimental-Knoten) 12,95
110258-91 ..... USB/RS485-Konverter (bestückt und getestet)..... 24,95

November 2011 (Nr. 491)

Verbesserter Strahlungsmesser
110538-41 ..... Progr. Controller ATmega88PA-PU 10,50
110538-71 ..... Bausatz mit allen Bauteilen inkl. Platine, Display und progr. Controller 39,95

Hier kommt der Bus (9)
110258-1 ..... Platine (Experimental-Knoten)..... 5,95
110258-1C3 ... 3 x Platine (Experimental-Knoten) 12,95
110258-91 ..... USB/RS485-Konverter (bestückt und getestet)..... 24,95

Fledermaus-Sonar
110550-1 ..... Platine 9,95

OnCE/JTAG-Interface
110534-91 ..... Programmer-Platine (bestückt und getestet) 39,95

Oktober 2011 (Nr. 490)

AVR-Platine Platino
100892-1 ..... Platine 12,95

Hier kommt der Bus (8)
110258-1 ..... Platine (Experimental-Knoten)..... 5,95
110258-1C3 ... 3 x Platine (Experimental-Knoten) 12,95
110258-91 ..... USB/RS485-Konverter (bestückt und getestet)..... 24,95

Audio-DSP-Kurs
110001-91 ..... DSP-Board (bestückt und getestet) 129,95
110001-92 ..... Kit bestehend aus DSP-Board (110001-91) und Programmer (110534-91 / Veröffentlichung im November-Heft / erhältlich ab KW42)..... 149,95

September 2011 (Nr. 489)

USB-Wetterlogger mit Langzeitspeicher
100888-1 ..... Platine 17,95
100888-41 ..... Programmierter Controller ATmega88-20PU 9,95

Bestseller

Bücher

1 **Digitale Logik selbst entwickeln**
ISBN 978-3-89576-254-3 ..... € 49,00 ..... CHF 60,80

2 **311 Schaltungen**
ISBN 978-3-89576-255-0 ..... € 36,80 ..... CHF 45,70

3 **AVR-Programmierung 3**
ISBN 978-3-89576-231-4 ..... € 46,00 ..... CHF 57,10

4 **Stromversorgungen in der Praxis**
ISBN 978-3-89576-239-0 ..... € 46,00 ..... CHF 57,10

5 **Embedded Electronics 3**
ISBN 978-3-89576-185-0 ..... € 49,00 ..... CHF 60,80

CD- & DVD-ROMs

1 **Workshop-DVD 'FPGA'**
ISBN 978-3-89576-259-8 ..... € 29,80 ..... CHF 37,00

2 **ATM18-Collection**
ISBN 978-0-905705-92-7 ..... € 29,50 ..... CHF 36,60

3 **The Audio Collection 3**
ISBN 978-90-5381-263-1 ..... € 21,50 ..... CHF 26,70

4 **ECD 6**
ISBN 978-90-5381-258-7 ..... € 29,50 ..... CHF 36,60

5 **Elektor-DVD 2010**
ISBN 978-90-5381-267-9 ..... € 27,50 ..... CHF 34,10

Bausätze & Module

1 **Verbesserter Strahlungsmesser**
Art.-Nr. 110538-71 ..... € 39,95 ..... CHF 49,60

2 **USB/RS485-Konverter**
Art.-Nr. 110258-91 ..... € 24,95 ..... CHF 31,00

3 **USB-FT232R Breakout-Board**
Art.-Nr.110553-91 ..... € 15,00 ..... CHF 18,60

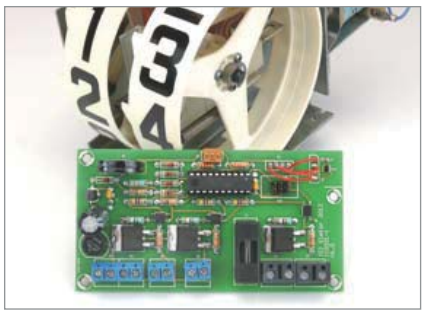
4 **USB-Wetterlogger mit Langzeitspeicher**
Art.-Nr. 100888-73 ..... € 34,95 ..... CHF 43,40

5 **Pico C**
Art.-Nr. 100823-71 ..... € 82,50 ... CHF 102,30

Bestellen Sie jetzt einfach und bequem online unter www.elektor.de/shop oder mit der portofreien Bestellkarte am Heftende!

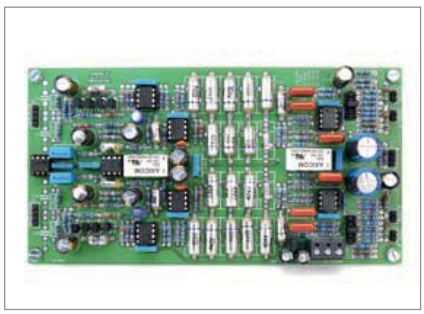
Elektor-Verlag GmbH
Süsterfeldstr. 25, 52072 Aachen
Tel. +49 (0)241 88 909-0
Fax +49 (0)241 88 909-77
E-Mail: bestellung@elektor.de

Preisänderungen und Irrtümer vorbehalten!



Elektromechanisches Thermometer

Elektronische Thermometer mit LCD findet man in jedem Supermarkt für ein paar Euro. Doch als Elektroniker sucht man natürlich einen originelleren Weg, die Temperatur anzuzeigen. Unser Thermometer ist etwas ganz Besonderes: Zur Anzeige wird ein Score-Board eines Flipper-Automaten genutzt! Angesteuert wird dieser elektromechanische Zähler durch einem ATtiny2313, der die Umgebungstemperatur mit Hilfe eines TMP100-Sensors ermittelt.



High-End-Vorverstärker

Nach unserem besonderen „5532-OpAmplifier“ Endverstärker (Oktober/November 2010) hat der bekannte Audio-Spezialist Douglas Self einen passenden Vor-/Regelverstärker entwickelt, der ebenfalls (parallel geschaltete) 5532-OpAmps verwendet. Unser Elektor-Labor hat hierzu Platinen entworfen, die viele Einstell- und Regelungsmöglichkeiten bieten. Im nächsten Heft stellen wir den Aufbau dieses Vorverstärkers vor.

RS485-Schaltplatine

Unser ElektorBus-Projekt zeigt, wie groß das Interesse am Messen-Schalten-Regeln und speziell an der Hausautomatisierung ist. Im nächsten Heft stellen wir daher eine kompakte Platine vor, die mit zwei Relais, einem ATmega88 und einem RS485-Treiber ausgestattet ist. Hiermit lassen sich 230-V-Verbraucher schalten; außerdem sind zwei Controller-Eingänge auf Schraubklemmen geführt, so dass man auch die Stellung von Lichtschaltern o.ä. erfassen kann. Dazu gibt es Software, die mit dem ElektorBus kompatibel ist. Ein weiterer Baustein also für die Haussteuerung über PC, Tablet und Smartphone!

Elektor April 2012 erscheint am 21. März 2012.
Elektor gibt es im Bahnhofsbuchhandel, Elektronik-Fachhandel, an ausgewählten Kiosken und garantiert beim Presse-Fachhändler.
Sie können Elektor auch direkt bei www.elektor.de bestellen.

Änderungen vorbehalten!

Elektor im Netz

Alle Magazin-Artikel ab 1996 können einzeln im PDF-Format heruntergeladen werden (gegen Elektor-Credits). Zu jedem Artikel findet man eine kurze Zusammenfassung, technische Daten und eine Stückliste (sofern vorhanden), was bei der Projektauswahl hilft. Dazu kommen eventuelle Updates/Berichtigungen und die Downloads zum Artikel (z.B. Software und Zusatzinfos). Auch die zum Projekt gehörenden Produkte wie Platinen, programmierte Controller und mehr lassen sich hier bestellen.

Im Elektor-Shop findet man alle anderen Angebote von Elektor, wie CD-ROMs, DVDs, Kits, Module, Software und Bücher. Mit der Suchfunktion können Sie die ganze Site nach Stichwörtern durchforsten.

Ebenfalls unter www.elektor.de:

- News aus der Elektronikwelt
- Leserforum
- Download des E-Magazins
- Spezialangebote (zeitlich begrenzt)
- FAQs, Inhaltsverzeichnisse und Kontaktformular

Ja, ich möchte Elektor im Jahresabonnement

03/2012

(11 Hefte / inkl. Doppelheft Juli/August) pünktlich und zuverlässig frei Haus beziehen*.
Im Vergleich zum Einzelheftkauf am Kiosk spare ich beim Standard-Abonnement 8 €
(bei der PLUS-Variante sogar bis zu 23 €). Als Dankeschön erhalte ich das Elektor-Buch
"Basiskurs R8C/13" (sofort nach Zahlung der Abonnementsrechnung) gratis zugeschickt.

Bitte wählen Sie Ihr Jahresabonnement aus:

☐ **Jahresabonnement-Standard für nur 82,00 €**

☐ **Jahresabonnement-PLUS (inkl. Jahrgangs-DVD 2012** + exklusiver Online-Zugang zu Elektor-Plus.de) für nur 99,50 €**

TIPP

Zahlungsweise ☐ Rechnung ☐ Bankeinzug (gilt nur für D)

Bank

Konto

BLZ

Datum, Unterschrift

*Das Abonnement verlängert sich automatisch um 12 Monate, wenn nicht spätestens zwei Monate vor Ablauf schriftlich gekündigt wird.
**Diese DVD-ROM wird Ihnen sofort nach Erscheinen (Februar 2013) zugeschickt.

Ja, ich möchte Elektor kennenlernen!

03/2012

Ich erhalte die nächsten 3 Ausgaben für nur 14,90 € pünktlich und zuverlässig frei Haus.*

Wenn Sie innerhalb von 1 Woche nach Erhalt der dritten Ausgabe nichts von mir hören, möchte ich Elektor im Jahresabonnement für nur 82,00 € weiter beziehen.

*Dieses Angebot gilt nur, wenn Sie während der letzten 12 Monate noch nicht Abonnent waren.

Zahlungsweise ☐ Rechnung ☐ Bankeinzug (gilt nur für D)

Bank

Konto

BLZ

Datum, Unterschrift

Elektor-Bestellkarte 03/2012

Diesen Streifen an den unten stehenden Streifen kleben!

Ich bestelle folgende Elektor-Produkte:

Bezeichnung	Preis	Anzahl	Gesamtpreis
Regelungstechnik NEU	49,00 €		
Elektor-DVD 2011 NEU	27,50 €		
Digitale Logik selbst entwickeln	49,00 €		
Workshop-DVD "FPGA"	29,80 €		
311 Schaltungen	36,80 €		
Stromversorgung ohne Stress 1	38,00 €		
Embedded Electronics 3	49,00 €		
AVR-Programmierung 3	46,00 €		
zzgl. Porto- und Versandkosten			5,00 €
Gesamtbetrag			€

Datum: **Unterschrift:**

Tragen Sie bitte Ihre Anschrift auf der Rückseite ein!

Diesen Streifen an den oberen Streifen kleben!

Hier ist meine Anschrift:

Firma

Vorname

Name

Straße, Nr.

PLZ, Ort

Land

DE

Kunden-Nr.

E-Mail

Innerhalb Deutschlands kein Porto nötig!

Antwort

Elektor-Verlag GmbH
Süsterfeldstr. 25
52072 Aachen

Hier ist meine Anschrift:

Firma

Vorname

Name

Straße, Nr.

PLZ, Ort

Land

DE

Kunden-Nr.

E-Mail

Innerhalb Deutschlands kein Porto nötig!

Antwort

Elektor-Verlag GmbH
Süsterfeldstr. 25
52072 Aachen

Hier ist meine Anschrift:

Firma

Vorname

Name

Straße, Nr.

PLZ, Ort

Land

DE

Kunden-Nr.

E-Mail

Innerhalb Deutschlands kein Porto nötig!

Antwort

Elektor-Verlag GmbH
Süsterfeldstr. 25
52072 Aachen

Abonnieren Sie jetzt die führende US-amerikanische Fachzeitschrift für Mikrocontroller-Anwendungen und Embedded Systems!

Wählen Sie unter
www.elektor.de/cc-abo
Ihr gewünschtes Abonnement aus!

12 Original-Ausgaben jährlich für nur

Digital:	38 US-Dollar
Print:	63 US-Dollar
Digital + Print:	90 US-Dollar

CIRCUIT CELLAR®
THE MAGAZINE FOR COMPUTER APPLICATIONS

Elektronikring 1, 26452 Sande
Telefon: +49 (0)4422 955-333
Telefax: +49 (0)4422 955-111

Anzeigenforum: www.explore.reichelt.de
Tagespreise! - Preisstand: 2. 2. 2012

reichelt.de elektronik

LOGO START RCE 2 **309,00**

SIEMENS LOGO!

NEU! Jetzt mit Ethernet!

Starter-Box, 230 Volt

Logikmodul im kostengünstigen Komplettpaket: Einfachste Projektierung von Schaltprogrammen
- direkt am Gerät/PC mit LOGO! Soft Comfort V7
- Verbinden von Funktionen per Drag and Drop
- Offline-Simulation und Online-Test
- professionelle Dokumentation mit individuellen Blocknamen und Ausdruck aller notwendigen Projektinformationen

12/24-V-Version: LOGO! Starter Box 12/24 RCE
- mit zusätzlichem LOGO! Power-Modul (24 V/1,3 A)

LOGO START RCE 1 349,00

LOGO! Logikmodul, einzeln

LOGO 230 RCE	115/240 Volt	209,00
LOGO 12/24 RCE	12/24 Volt	209,00

SIEMENS

Ein Plus an Automatisierungs-lösungen für Sie!

SIEMENS LOGO!

Erweiterungs-module

Für SIEMENS LOGO! Kleinsteuerung

Analog:
- 2 Analogeingänge 2 x PT100, 2- oder 3-Leiter
- Temperaturbereich: -50 bis +200°C
- Versorgungsspannung: 12/24 V DC
- 2 Analogausgänge: 0 ... 10 V

LOGO AM2 RTD analog 99,65

Digital:
- 4 digitale Eingänge
- 4 Relaisausgänge 5(3) A
- Versorgungsspannung: 12/24 V DC

LOGO DM8 12/24R digital 73,50

Industrielle Kommunikation

Ob für das Steuern von Maschinen, das Überwachen von Fertigungslinien oder die Koordination ganzer Produktionsbereiche - ohne Industrielle Kommunikation wären diese komplexen Aufgaben nicht denkbar.

Markenqualität • Top-Service • günstige Preise
www.reichelt.de

SIEMENS simatic

Kommunikations-modul

- RS 232, Punkt-zu-Punkt-Kommunikation
- 9-polig, SUB-D-Buchse
- unterstützt FREEPORT

CM 1241 RS232 107,95

Stromver-sorgung
geregelt

- Eingang: AC 120/230 V
- Ausgang: DC 24 V/2,5 A

PM 1207 AC-DC 79,95

SIEMENS simatic

Signalmodule

Digitaleingabe:
- 8 x 24 VDC
- 8 Eingänge (2 Gruppen)
- Versorgungs-spannung: 24 V DC
- IEC Typ 1 P-lesend

SM 1221 24V 89,95

Digitalausgabe:
- 8 Relaisausgänge (2 Gruppen)
- Versorgungsspannung: 5 - 30 V DC
- Ausgangsspannung: 5 - 250 V AC / 5 - 30 V DC
- Ausgangsstrom: 2 A

SM 1222 RLY 89,95

SIEMENS simatic

Signalboard

- 4 digitale Eingänge (1 Gruppe)
- Eingangsspannung: 0-5 V , 24 V und 15-30 V
- 200-kHz-Ein- und Ausgänge für schnelle Zähler und Impuls-generatoren
- 24-V-DC-Ausführung für HTL-Signale
- Maße (BxHxT): 38 x 62 x 21 mm

SB 1221 24V 57,95

Surfen Sie gleich los und entdecken unser gesamtes Sortiment in den Bereichen:

SIEMENS LOGO!
SIEMENS simatic

Einfach QR-Code per Smartphone scannen ...

... oder Kurzlink eingeben:

http://rch.lt/1T

www.reichelt.de
Schaltschrank-/Anlagenbau: Siemens Logo! & simatic

Starter Kit
SIMATIC S7-1200

Die SIMATIC S7-1200 Steuerung ist modular, kompakt und vielseitig einsetzbar. Eine Kommunikations-schnittstelle, die die höchsten Anforderungen der Industrie erfüllt, und eine ganze Palette leistungsstarker und integrierter Technologie machen diese Steuerung zu einem integralen Bestandteil einer umfassenden Automatisierungslösung.

- CPU 1212C AC/DC/RLY
- Eingang-Simulator
- STEP7-Basic-CD
- Handbuch (CD), Infomaterial
- IE TP Cord, 2 m
- TANOS-Systainer-Transportbox

S7-1200 STARTER **429,00**

SIEMENS simatic

Katalog kostenlos!
Tagesaktuelle Preise:
www.reichelt.de