Neu: 8 Seiten mehr!

.magazine
Dezember 2014 | Nr. 528

elektor

Cool Controller Concept

Dreh-Encoder, 17 Zweifarb-LEDs, Taster, Buzzer, ARM-Cortex-Controller:

Modulares User-Interface für Ihre Anwendung!

Pegel- und Distanzmessgerät | Stromsparender Datenlogger | Programmierbarer LED-Weihnachtsbaum | **Breakout-Board für 16-bit-ADC** | Steuer- und Anzeigemodul mit LCD | **Atmel-Studio-Projekte automatisch generiert** | Messplattform Red Pitaya in der Praxis | Schalten via Tablet | Raspberry-Pi-Emulator | FFT-Software in VB | **Dehnung messen mit PSoC** | Bauteile: Tantal-Tropfen

G3078 E

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

4 198630 307906 12

Professionelle Qualität zu attraktiven Preisen!

reichelt.de elektronik

Kundenbewertungen

96.84% zufriedene Kunden

4.84 / 5.00

Rund 97 % unserer Kunden sind vom **reichelt**-Service überzeugt*

*Quelle: Shopauskunft.de (1. 9. 2014)

- 45 Jahre Erfahrung
- schneller 24-Stunden-Versand
- über 50.000 Produkte am Lager
- kein Mindermengenzuschlag

Preis-Tipp!

Digital-Multimeter

mit umfangreichen Messfunktionen

- **Anzeigeumfang: 2.000 Counts**
- **Grundgenauigkeit: 0,5 %**
- **Manuelle Bereichswahl**
- **Hold-Funktion**

- AC/DC Strom-/Spannungsmessung bis 20 A/1000 V
- Widerstands-, Kapazitäts-, Frequenz-, Induktivitäts- und Temperaturmessung
- Transistor-, Durchgangs- und Diodentest

EN 61010-1 CAT III 1000 V

PEAKTECH 2005

29,95

Gleich mitbestellen:

Hochwertige Alukoffer

Der perfekte Schutz für Ihre Messtechnik!

- staub- und spritzwasser-geschützt durch Schließ-kante mit Doppelnut
- gepolsterter Innenraum schützt vor Sturzschäden

PeakTech®

PEAKTECH 7250	298 x 90 x 70 mm	**14,10**
PEAKTECH 7255	295 x 195 x 70 mm	**16,50**
PEAKTECH 7260	370 x 230 x 80 mm	**20,95**
PEAKTECH 7265	390 x 280 x 100 mm	**21,95**
PEAKTECH 7270	500 x 350 x 120 mm	**27,60**

5-in-1-Digital-Multitester

mit integrierten Umweltmessfunktionen

EN 61010-1 CAT III 600 V

- Anzeigeumfang: 4.000 Counts
- Grundgenauigkeit: 1,0%
- AC/DC Strom- und Spannungsmessung bis 10 A / 600 V
- Widerstands-, Kapazitäts-, Frequenz- und Temperaturmessung
- Messung von Lufttemperatur und relativer Luftfeuchte
- Schallpegel- sowie Beleuchtungsstärkenmessung
- Automatische / manuelle Bereichswahl, Hold-Funktion
- Durchgangs- und Diodentest, Relativwertmessung

PeakTech®

PEAKTECH 3690

Sonderpreis! 59,95

% bisher 66,95

FLIR Kompakt-Kamera für industrielle Anwendungen und Messungen im Baubereich

FLIR

Eine der leichtesten & preisgünstigsten Wärmebildkameras weit und breit.

Anvisieren, aufzeichnen und auswerten – so einfach erhalten Sie hochwertige Wärmebilder!

- Empfindlichkeit: 0,1°C
- IR-Auflösung: 140 x 140 Pixel
- Messfunktionen: Messpunkt, Rechteck-bereich mit max./min. Temperaturen, Iso-therme ober- und unterhalb des ge-wählten Temperatur-intervalls

PRICE BURNER

FLIR I7

% bisher 2.493,00

1.779,00

Direkt zum Angebot

http://rch.lt/flir

PeakTech®

Digital-Speicher-Oszilloskop

Extra großes Full-Color-TFT-Display

- Bandbreite: 30 MHz
- Kanäle: 2
- Abtastrate: 125 MSa/s pro Kanal
- vertikale Empfindlichkeit: 2 mV - 10 V/Skt.
- Anstiegszeit: 11,7 ns
- Schnittstellen: USB, USB flash disk, VGA, LAN
- Autoset- und Autoscale-Funktion
- 20 automatische Messmodi und FFT-Funktion
- PASS/FAIL-Funktion
- Sicherheit: EN 61010-1, CAT II

Sonderpreis! 289,00

PEAKTECH 1265

% bisher 329,00

Jetzt bestellen!

Newsletter

Jetzt abonnieren & gewinnen

http://rch.lt/news

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

www.reichelt.de

NEU

Jetzt anfordern!

Katalog 01|2015

Kostenlos!

www.Trace.ME

Incorporating TraceME with your projects

The possibilities of KCS TraceME's M2M connectivity are virtually limitless. It's not a hollow marketing slogan! If you are planning a project which isn't described, chances are that it can be achieved too. Do not hesitate to contact us.

Quick return on investment
Saving on costs is perhaps the most common reason to invest in TraceME technology.

Tracking drones with GPS and GPRS
Flying with drones is quickly becoming a very popular hobby. Drones are equipped with all sorts of electronics, mostly cameras. Add GPS tracking functionality to this flying platform and you can always find the drone via the TraceME.

Personal GPS locater
In certain situations tracing people can be critical. Whether it is your kids, elderly relative, high-profile VIPS, pets. All can be located, on demand or afterwards with TraceME key fob. It is possible to have 2 way audio conversations.

KCS TraceME Micro is the most configurable and smallest tracing device available on the market today.

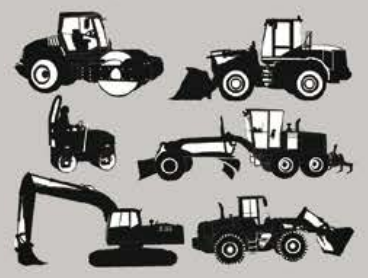
Animal management
Animals tagged with a RFID chip can be remotely monitored and managed. TraceME can be equipped with an RFID reader so it is essentially a checkpoint for keeping administration of which animal has already been fed, milked, shaved, etc.

Bird life tracking
A TraceME Micro is small enough to be a real-time bird tracking system. The on-board connectivity allows the position of the bird to be fully traced while the animal is entering a race or flying around in it's natural habitat.

Extreme Sports
There are a number of Extreme Sports that requires you to travel across some sort of terrain.

Backup power generator
For critical machines running (backup power generator) a black-out can be close to life. TraceME gives you remote solutions.

Security applications
It is the most obvious use of a TraceME module. Keep discretely track of your properties with on-site surveillance or on the go. It essentially becomes a 24/7 security guard on site. Set an alert in the case of alarm via SMS, GSM, GPRS, Wi-Fi, RFID, 3G, etc..



Real-time update of vending machines
TraceME equipped vending machine can give a real-time update of it is inventory and sales of that day.

Rented machinery
If you rent out machinery which needs high level of reliability or service, place a TraceME module inside.

Flexible payment solutions
If your company is a service provider for office facilities, you can deploy the TraceME for flexible payment solutions. Imagine connecting the TraceME to a copying or coffee machine.

TraceME is the affordable alternative to SCADA solutions. To measure is to know.

Advertising displays Control advertising displays from your desktop or mobile phone.

Keeping track of your personal and business mileage for taxation reasons
Do you have to keep track of when you drive a company car for personal use? TraceME can help.

Reading out car statics with ODB-II
A ODB-II plug is fitted with a TraceME inside for an easy connection and reading out car statistics.

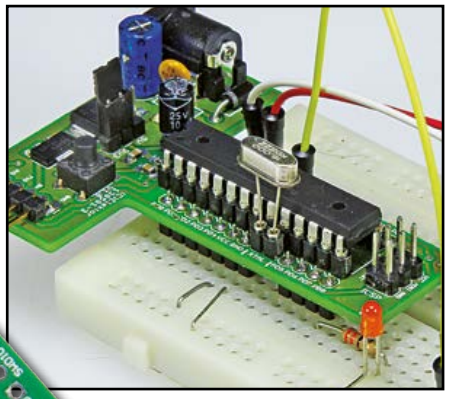
Social Media Connectivity
Sometimes you want to share your location via Twitter and Facebook. TraceME supports this.

Speed camera detection TraceME as a warning system preventing fines for speeding.

www.Trace.ME

Inhalt

elektor magazine

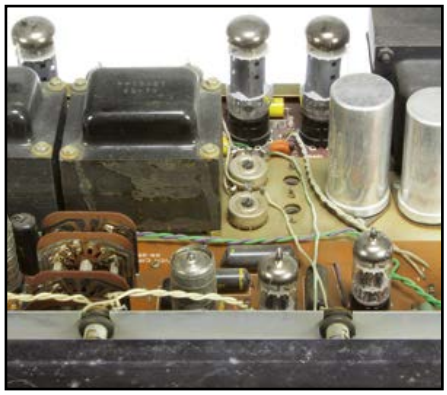


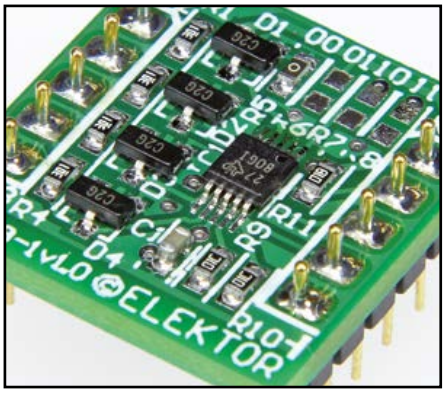
Industry

Labs

Projects

45. Jahrgang – Nr. 528 Dezember 2014





Magazine

Impressum

45. Jahrgang, Nr. 528 Dezember 2014
Erscheinungsweise: 10 x jährlich
(inkl. Doppelhefte Januar/Februar und Juli/August)

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

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

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

Anzeigen:
Margriet Debeij (verantwortlich)
Tel. 02 41/88 909-13 / Fax 02 41/88 909-77
Mobil: +31 6 510 530 39
E-Mail: margriet.debeij@eimworld.com

Julia Grotenrath
Tel. 02 41/88 909-16 / Fax 02 41/88 909-77
E-Mail: julia.grotenrath@eimworld.com

Es gilt die Anzeigenpreisliste Nr. 44 ab 01.01.2014

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

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

Anwendungs-Getrieben

Als Physikstudent hat man oft mit ein-, zwei- oder mehrdimensionalen Arrays von Zahlen zu tun. Mit 20 träumte ich davon, solche Daten mit einer eigenen Skriptsprache bearbeiten zu können. Mein Projekt begann mit einem veritablen Satz von Kommandos und einer ausgefuchsten Fehlerbehandlung recht ambitioniert, doch schlichen sich schnell eine Menge Bugs ein. Nach einer Weile scheiterte das Ganze an mangelndem Ehrgeiz, den damals noch recht begrenzten Computerressourcen und nicht zuletzt an einer fehlenden Killerapplikation.
Gut 15 Jahre später begann ich bei Elektor; von meinem ersten Job als Redakteur brachte ich ein Excel-Sheet für die Seitenplanung mit. Bei Elektor kam ich wieder öfter mit Softwarethemen in Berührung, und ich fing an, unsere Excel-Planung mit etwas VBA zu optimieren. Die zwei Damen vom Redaktionssekretariat drängten mich bald darauf, meine Planung auf MS Access umzustellen, denn sie arbeiteten bereits mit einem access-basierten System, um die Autoren zu verwalten. Ich kaufte mir ein Standardwerk von André Minhorst, und die Arbeit an unserem NRS (Neues Redaktionssystem) begann. Unzählige Abende und Wochenenden später hatten wir eine Datenbank, mit dem wir auch unsere Downloads, die Daten für die Website und vieles andere mehr verwalteten. In gelegentliches Fluchen über eine manchmal nicht ganz intuitive Bedienung mischte sich bei den Kollegen immer mehr der Wunsch, diverse, maßgeschneiderte Darstellungen unserer Daten zu bekommen, in HTML, XML oder als Excel-Blatt. Ich programmierte eine Weile sehr langweiligen, sehr ähnlichen VB-Code und begann, über eine effizientere Lösung nachzudenken. An einem Wochenende skizzierte ich ein paar Skriptkommandos, die Abfragen machen, Text und Tags ausgeben und Excel-Zellen füllen konnten. Ständig kamen nun neue Anforderungen hinzu: das Einlesen von Word-Dokumenten, das Verschieben von Files und schließlich Formulare, die auch über das Web bedienbar waren. Inzwischen entsteht die zweite Version meiner Skriptsprache, die ich auch gerne für lokale PC-Anwendungen einsetze (Seite 52).
Was ich Ihnen damit sagen will?
1. Es lohnt sich oft, erst einmal klein anzufangen. 2. Sie sollten immer von einer guten Anwendung getrieben sein.
3. Wie immer gilt: Bleiben Sie dran!

Jens Nickel
Chefredakteur Elektor

Unser Team

Chefredakteur:	Jens Nickel (v.i.S.d.P.) (redaktion@elektor.de)
Ständige Mitarbeiter:	Dr. Thomas Scherer, Rolf Gerstendorf
Leserservice:	Ralf Schmiedel
Korrekturen:	Malte Fischer
Internationale Redaktion:	Harry Baggen, Jan Buiting, Denis Meyer
Elektor-Labor:	Thijs Beckers, Ton Giesberts, Wisse Hettinga, Luc Lemmens, Mart Schroijen, Jan Visser, Clemens Valens, Patrick Wielders
Grafik & Layout:	Giel Dols

Germany
Ferdinand te Walvaart
+49 241 88 909-17
f.tewalvaart@elektor.de

United Kingdom
Carlo van Nistelrooy
+44 20 7692 8344
c.vannistelrooy@elektor.com

Netherlands
Ferdinand te Walvaart
+31 46 43 89 444
f.tewalvaart@elektor.nl

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

USA
Carlo van Nistelrooy
+1 860-289-0800
c.vannistelrooy@elektor.com

Spain
Jaime González-Arintero
+34 6 16 99 74 86
j.glez.arintero@elektor.es

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

Sweden
Carlo van Nistelrooy
+31 46 43 89 418
c.vannistelrooy@elektor.com

Brazil
João Martins
+31 46 4389444
j.martins@elektor.com

Portugal
João Martins
+31 46 4389444
j.martins@elektor.com

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

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

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

South Africa
Johan Dijk
+31 6 1589 4245
j.dijk@elektor.com

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

Unser Netzwerk

elektor · VOICE COIL · circuit cellar · hifi.nl MUZIEK - FILM - LIFESTYLE · audioXPRESS · techthe future

Die Elektor-Community

Unsere Partner und Sponsoren

Sie möchten Partner werden?

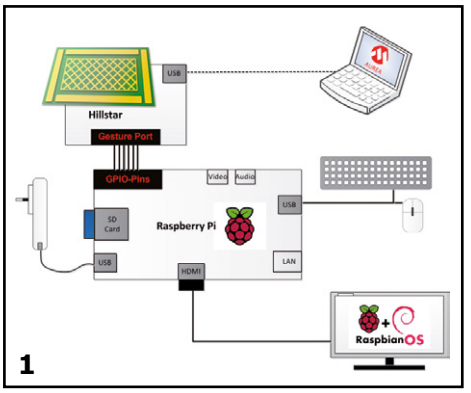
Kontaktieren Sie uns bitte unter m.debeij@elektor.de (Tel. 02 41/88 909-13).

GestIC & 3D TouchPad Kurs (1)

Gestensteuerung für Raspberry Pi

Von **Thomas Lindner** und **Roland Aubauer** (Microchip GestIC® Team, Deutschland) sowie **Jan Buiting** (Elektor)

Letzten Monat haben wir das Microchip/Elektor-Entwicklungskit für eine Gestensteuerung per 3D-Touchpad vorgestellt. Nun zeigen wir Ihnen die Möglichkeiten der Schlüsselkomponente des Systems, des MGC3130 GestIC®-Controllers, an einem RPi.



1

Direkt nachdem wir Ihnen das exklusive Bundle, bestehend aus dem MGC3130-Entwicklungskit zur Gestensteuerung von Hillstar und einem 3D-Touchpad, vorgestellt hatten, kamen die ersten Anfragen: „Wie kann ich das an meinen Mikrocontroller anschließen?“ und „Ich will es mit meinem PC verbinden!“

Das Angebot wurde im letzten Monat [1] angekündigt und die Hardware steht nun (bei Drucklegung dieser Ausgabe) zum Verkauf bereit [2].

In dieser Serie werden wir Schritt für Schritt eine anspruchsvolle Anwendung entwickeln, eine gemischt Touch- und berührungslose 3D-Steuerung des berühmten Raspberry Pi, um das „2048“-Spiel spielen zu können. Diese Serie soll Sie zu eigenen Hardware- und Software-Entwicklungen mit dem MGC3130-Chip auf einem beliebigen Mikrocontroller-System inspirieren.

1. Welche Hardware ist nötig?

Um diesem edukativen Projekt zu folgen, ist folgende Hardware erforderlich:

- Raspberry Pi Modell B v.2, RBCA000 ... 2x USB 2.0 mit 3,5 W
- Stromversorgung: Steckernetzteil über Micro-USB am RPi (1200 mA, 5 V)
- MGC3130-Entwicklungskit von Hillstar

Das letztendliche Sensor-System arbeitet unabhängig vom PC und muss nur von einem USB-Ladegerät oder direkt vom RPi versorgt werden. Die Kommunikation zwischen den Chips verläuft über die GesturePort-Anschlüsse auf der kleinen MGC3130-Platine im Hillstar-Entwicklungskit. Ein PC ist für die Parametrierung und das Flashen der Firmware auf den MGC3130 erforderlich. Ein Blockschaltbild der Hardware-Konfiguration ist in **Bild 1** dargestellt, **Bild 2** zeigt das Touchpad und RPi auf dem Labortisch.

2. Anschlüsse

Der MGC3130 kümmert sich um das komplette kapazitive „Sensing“ der 3D-Gestensteuerung. Die Signale EIO1, EIO2, EIO3, EIO6, EIO7 (und Masse natürlich) der Hillstar-MGC3130-Platine sind wie in **Bild 3** zu sehen mit den GPIO-Anschlüssen des RPi verbunden.

3. Parametrierung mit Aurea

Die MGC3130 ermöglicht es, Ost-West-und Nord-Süd-Handbewegungen in Signale an seinen EIOx-Pins zu verwandeln. **Aurea**, eine freie grafische Shell um den GestIC-Controller, ermöglicht die Parametrierung vieler Größen und Konfiguration von Ebenen, was das kapazitive Sensorpad kalibriert und mit hoher Präzision konfiguriert. Die Parameter für den GesturePort sind in **Bild 4** dargestellt. Beachten Sie die einzigartige Mischung aus 3D-Geste (Bewegung O<->W, Bewegung N<->S) und Touch (Center).

4. Software

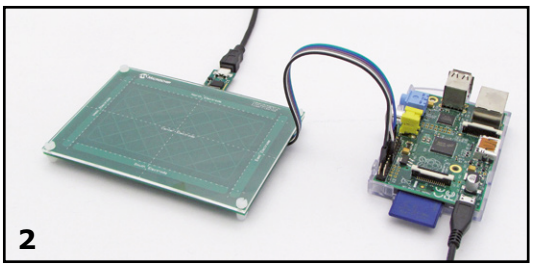
Als Software gebrauchen wir:

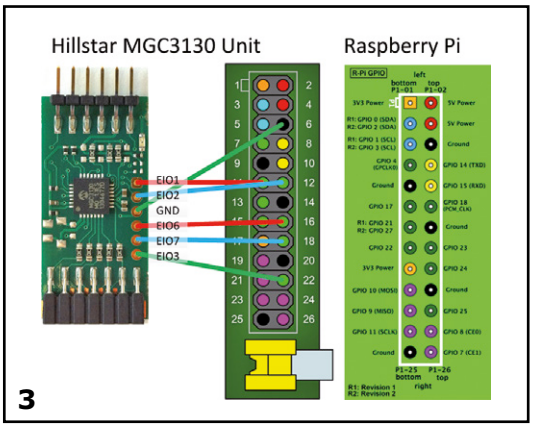
- Betriebssystem: Raspbian (Debian Wheezy), Version: Januar 2014 Veröffentlichungsdatum: 2014.01.07
- Python-Version: 2.7.3 (Standard, 18. März 2014, 05.13.23)
- RPi.GPIO-Library Version: 0.5.4
- Tkinter
- Leafpad-Texteditor
- den Code für das Spiel „2048_with_Gesture_Port_Demo.py“

Alle Programme stehen nach der Installation von Raspbian bereits zur Verfügung, lediglich das Spiel muss zusätzlich installiert werden. Wie das geht (und worum es sich dabei überhaupt handelt), wird im nächsten Monat beschrieben.

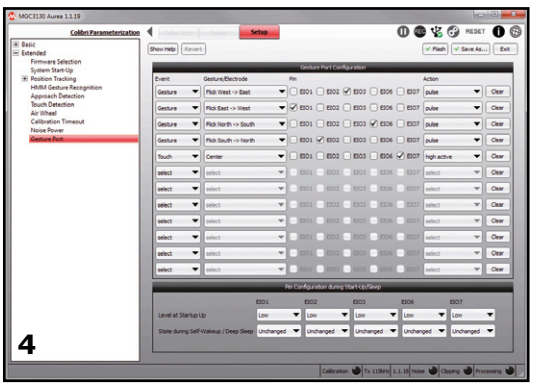
5. Schnellstart

1. Installieren Sie Raspbian auf dem Raspberry Pi.
2. Schließen Sie das Hillstar-Entwicklungskit über USB an Ihren PC an und starten Sie Aurea. Es sollte ein kleines Popup-Fenster erscheinen.
3. Gehen Sie zu „Setup“ auf der oberen Seite und wählen Sie dann „Parameterization“.
4. Klicken Sie nun auf auf der linken Seite auf „Extended“, browsen unter „Parameters“ zu der vorkonfigurierten Datei „Hillstar GesturePort to Raspberry Pi Demo 2048.enz“ und starten Sie die Parametrierung, die eine Weile dauert.
5. Nach dem Abschluss klicken Sie auf der linken Seite auf „GesturePort“, so dass Sie die Konfiguration für die Events sehen. Wenn Sie jetzt auf „Flash“ in der oberen rechten Ecke drücken, wird die dargestellte Konfiguration im MGC3130-Chip gespeichert.
6. Einmal geflasht, läuft das Hillstar-Entwicklungskit im Standalone-Modus. Schließen Sie die Platine wie in Bild 1 gezeigt an den RPi an. Die Schaltung kann entweder von einem externen USB-Ladegerät oder über die entsprechenden Pins der GPIO-Leiste des RPi über den Mini-USB-Anschluss mit +5 V versorgt werden.


2



3


4

Nächsten Monat geht es weiter mit der *GestIC & 3D TouchPad*-Serie. Neuigkeiten zum GestIC-Entwicklungskit und dem 3D-Touchpad aus dem Elektor-Labor gibt es inzwischen auch im Elektor. POST-Newsletter.

(140423)

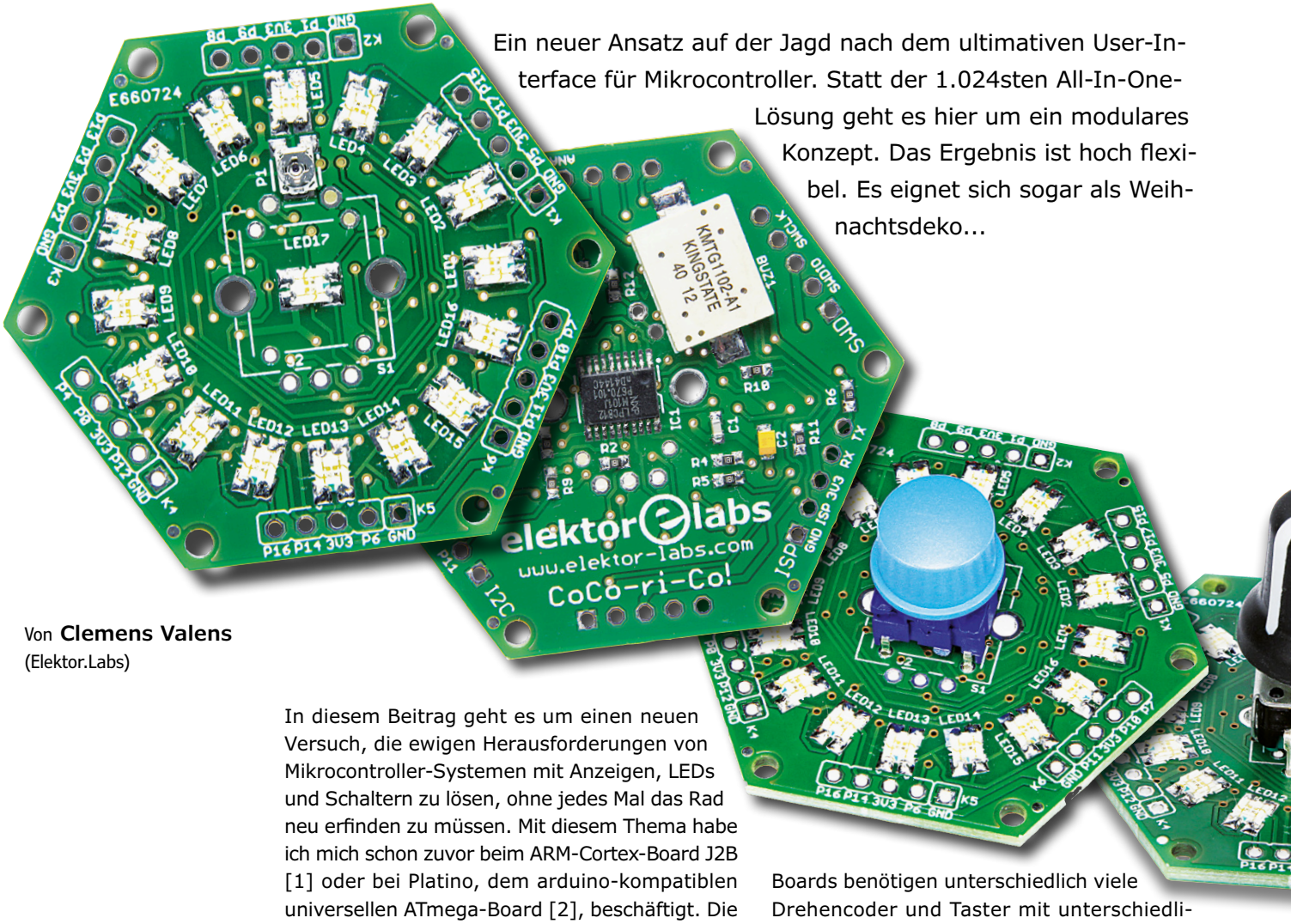
Weblinks

[1] 3D-Steuerung für Ihren Controller oder PC. Elektor November 2014, www.elektor-magazine.de/140408

[2] MGC3130-Entwicklungskits von Hillstar und 3D-TouchPad: www.elektor.de/microchip-dm160218-hillstar-development-kit-and-dm160225-3d-touchpad

[3] www.raspberrypi.org/downloads/

CoCo-ri-Co:
Cool Controller Concept



Ein neuer Ansatz auf der Jagd nach dem ultimativen User-Interface für Mikrocontroller. Statt der 1.024sten All-In-One-Lösung geht es hier um ein modulares Konzept. Das Ergebnis ist hoch flexibel. Es eignet sich sogar als Weihnachtsdeko...

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

In diesem Beitrag geht es um einen neuen Versuch, die ewigen Herausforderungen von Mikrocontroller-Systemen mit Anzeigen, LEDs und Schaltern zu lösen, ohne jedes Mal das Rad neu erfinden zu müssen. Mit diesem Thema habe ich mich schon zuvor beim ARM-Cortex-Board J2B [1] oder bei Platino, dem arduino-kompatiblen universellen ATmega-Board [2], beschäftigt. Die Boards benötigen unterschiedlich viele Drehencoder und Taster mit unterschiedlichen LCDs. Da entstand die Idee, das auf eine Platine zu packen, um dem Entwickler das Leben zu erleichtern.

Technische Daten

- NXP LPC812 32-bit-ARM-Cortex-M0+
- Alle 18 I/O-Pins an Pinheadern verfügbar
- Bis zu 17 zweifarbige LEDs
- Drehencoder und/oder Taster
- Buzzer
- Kommunikation via I²C, SPI (synchron) oder asynchron seriell
- ISP-Port kompatibel mit 3,3-V-FTDI-USB/Seriell-Kabel (kein 5-V-Betrieb)

Dabei wählte ich einen anderen Ansatz: Statt viele Bedienelemente in unterschiedlich möglicher Anordnung gibt es nur ein einziges. Statt einer großen Platine mit allem drauf handelt es sich hier um ein kleines Modul, mit dem man auch andere Projekte veredeln kann. Keine voluminösen, schlecht ablesbaren LCDs mit kryptischen Meldungen in hässlichen Zeichensätzen. LEDs reichen aus. Sie sind hell, aus großer Entfernung

gut abzulesen und können in unterschiedlichen Frequenzen blinken.

Kompromisse der Vergangenheit

Viele elektronische Geräte haben Knöpfe und Taster um einen oder mehrere Parameter einzustellen. Für einige Einstellungen braucht es keine präzise Anzeige von Werten. Beispiel ist die Lautstärkeeinstellung von Verstärkern. Wenn es laut (oder leise) genug ist, hört man einfach auf zu drehen. Andere Einsteller (z.B. zur Auswahl von Eingängen) müssen zwar die jeweilige Stellung anzeigen, aber von hoher Auflösung kann man da nicht reden. Ein paar LEDs sind notwendig und ausreichend. Nur wenn man präzise Werte benötigt, ist eine numerische Anzeige berechtigt. In einigen Fällen nutzt man Up/Down-Taster, um Drehgeber zu ersetzen, obwohl letztere schneller und intuitiver sind. Taster brauchen wenig Platz und sind preiswert sowie einfach an Mikrocontroller anzuschließen. Aber sie sind doch eher ein Kompromiss aus Zeiten, als Ressourcen von Mikrocontrollern (Speicher, I/O, Rechenleistung etc.) noch teuer waren. Das spielt aber keine Rolle mehr. Man löst heute selbst einfachste Probleme mit Controllern, da so ein Chip kaum mehr als einen Euro kostet. Warum also alte Kompromisse beibehalten?

Das Rad neu erfinden

Das vorgestellte Projekt ist quasi revolutionär (englisch „revolution“ = Umdrehung), denn es bietet einen Drehencoder zusammen mit 17 zweifarbigen LEDs (also 34 LEDs total) und einen Buzzer - gesteuert von nichts Geringerem als einem 32-bittigen Mikrocontroller des Typs ARM Cortex-M0+ (ab jetzt MCU genannt). Übertrieben? Kann schon sein, wenn man die Rechenleistung betrachtet. Aber sehr stimmig, wenn man die Kosten und die Verbindungsmöglichkeiten im Blick hat. Hier wird die MCU LPC812M101JDH20FP (abgekürzt LPC81) eingesetzt. Sie hat 18 I/O-Ports, 16 KB Flash, 4 KB RAM, etliche Timer, einen analogen Komparator, ein I²C-Interface, zwei SPI-Controller und drei asynchrone serielle Schnittstellen. Noch besser: Dank einer internen Rekonfigurationsmöglichkeit können die Kommunikationskanäle jeden der 18 verfügbaren I/O-Ports nutzen. Dieses sehr coole Feature hat zur Folge, dass man sehr simpel Multi-Protokoll-Kommunikation mit wenig Verdrahtungsaufwand realisieren kann, da die Schnittstelle per Software so eingestellt werden kann, dass sie entweder als I²C-, synchrone (SPI) oder als asynchrone serielle Schnittstelle konfiguriert wird. Mit einem weiteren Pin ist sogar eine konfigurierbare Vollduplex-Kommunikation möglich. Für den Preis von unter 2 € für Einzelstücke ist das wirklich geschenkt.

Alle Bauteile stecken auf einer kleinen sechseckigen Platine. Die MCU, eine LED und der Drehencoder sitzen in der Mitte. Die verbleibenden 16 LEDs sind in einem Kreis um den Drehencoder angeordnet. An jeder der sechs Seiten der Platine sind Bohrungen für Pinheader, mit denen sich mehrere Platinen miteinander oder mit der „eigentlichen“ Platine eines speziellen Projekts verbinden lassen.

Da nur „coole“ Bauteile vorhanden sind (ARM-MCU, zweifarbige LEDs, Drehencoder), ergibt sich daraus das *Cool Controller Concept* bzw. CoCo-ri-Co (französisch für Kikeriki). Die Platine ist, wie schon gesagt, sechseckig statt rund. Eckig ist technisch besser und einfacher zu handhaben. Auf diese kantige Weise wurde also das Rad neu erfunden ;-).

Wer braucht sowas?

Die zentrale Anwendung für CoCo-ri-Co ist sicherlich die als digitaler Poti mit Positionsanzeige. Jedes Gerät, das einen oder mehrere einstellbare Parameter benötigt, ist ein potentieller Kandidat für den Einsatz von CoCo-ri-Co. Man denke nur an die Einstellungen von Lautstärke und Klang bei Audio-Verstärkern, von Geschwindigkeit und verschiedenen Modi bei Haushaltsgeräten, von Temperatur bei Heizungen oder von Helligkeit und Farbe bei Beleuchtungen – überall da kann man mit Gewinn ein bis drei CoCo-ri-Co-Module einsetzen. Durch die serielle Kommunikation via I²C, SPI oder über eigene Protokolle über einen schlichten 3-Pin-Port kann man CoCo-ri-Co sehr einfach an Projekte anschließen, die einen eigenen Mikrocontroller mitbringen. Bei Geräten ohne eigene MCU kann CoCo-ri-Co sogar deren Funktion mit übernehmen.

CoCo-ri-Co ist klein und passt so in die meisten Gehäuse. Sogar in Wandinstallationsdosen oder in Mehrfachsteckdosen. Aufgrund der sechseckigen Form kann man damit einfach komplexere Bedienfelder konstruieren. **Bild 1** zeigt, wie das im Armaturenbrett eines Fahrzeugs aussehen könnte. Ein CoCo-ri-Co mit Encoder dient als Tacho, andere

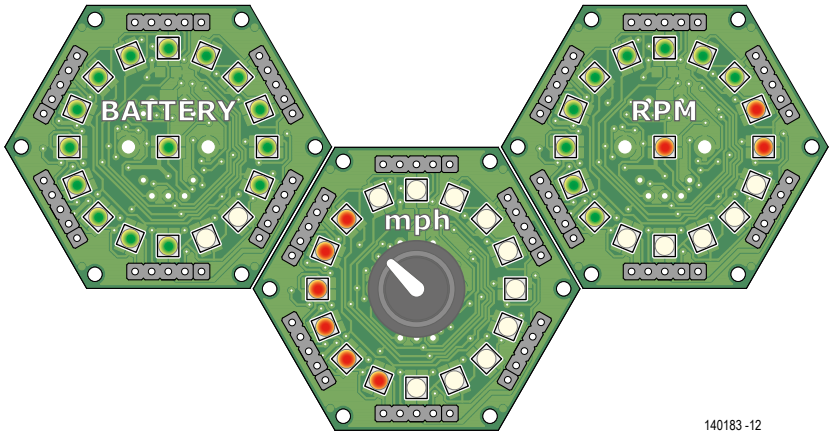


Bild 1.
Mit drei CoCo-ri-Co-Boards kann man ein Armaturenbrett eines futuristischen Fahrzeugs aufbauen.

ohne Drehencoder fungieren als Voltmeter für die Batterie und als Drehzahlmesser. Auch ein Einsatz als Thermometer ist denkbar.
Dank der Abmessungen und des geringen Stromverbrauchs kann man so ein CoCo-ri-Co auch mit Batterie betreiben. Mit einer 3-V-Knopfzelle wird das Modul nicht nur portabel, sondern sogar „wearable“ (z.B. als Geek-Uhr). Die zweifarbigen LEDs ermöglichen dekorative Anwendungen z.B. an Weihnachten, denn die roten und grünen Farben und der Buzzer zum Abspielen von Weihnachtsliedern machen sich da gut (siehe Kasten).
Doch das ist immer noch nicht alles. Dass man durch einfache Kombination auch umfangreichere Bedienfelder konstruieren kann, heißt nicht, dass diese flach und eben sein müssen. Tatsächlich kann man damit auch 3D-Oberflächen bestücken. Damit wirkliche Vielfalt herrscht, kann das Konzept mit dem Cool Controller Concept auf fünfeckige, quadratische und sogar dreieckige Platinen erweitert werden. Damit lassen sich sogenannte Archimedische Körper kreieren. Das sind diese hochsymmetrischen, regelmäßigen geometrischen Körper mit identischen Ecken - ganz wie der ordinäre Fußball. Dieser mathematisch als Ikosaederstumpf bezeichnete Ball besteht aus 20 Sechsecken und 12 Fünfecken. So ein CoCo-ri-Co-Ball (**Bild 2**) hat einen Durchmesser von 12 cm bei 25 mm Kantenlänge. Mit einem 3D-Accelerometer versehen kann man damit LED-Animationen erstellen und Klänge abspielen, die von Position, Geschwindigkeit und Lage des Balls abhängen.

Bild 2.
Polygonale Platinen für mehr Kreativität. Dieser Fußball besteht aus 20 sechseckigen und drei fünfeckigen CoCo-ri-Co-Platinen.

Funktion

Wenn sich die Position des Drehencoders ändert, schickt CoCo-ri-Co einen Wert (abhängig von der Drehrichtung größer oder kleiner werdend) über seine Schnittstelle (Standard ist UART, 115.200, 8N1). Der Wert liegt zwischen zwei Grenzen (0 und 100 als Standard). Der LED-Ring signalisiert den Wert als „Balken“-Grafik oder Leuchtpunkt in rot (Standard) oder grün. Die zentrale LED kann unabhängig angesteuert werden. Ein Druck auf den Taster erzeugt eine „Gedrückt“- und das Loslassen eine „Losgelassen“-Meldung. Die Meldungen können das folgende Format haben:

ASCII-Modus:

- Drehencoder: E*xxxxx*; *xxxxx* ist ein fünfstelliger Wert im Bereich 00000 bis 65535.
- Taster: B*x*; *x* = 0 (gedrückt) oder 1 (losgelassen).

Binärer Modus:

- Drehencoder: E*yz*; = *256 y + z*.
- Taster: B*y*; *y* = 0 (gedrückt) oder 1 (losgelassen).

Man kann das Verhalten des Programms jederzeit ändern, indem man ein paar Befehle an CoCo-ri-Co schickt. So kann man die Farbe und den Bereich (Standard ist 0...15) der LEDs sowie die Grenzen, den Wert oder den Modus des Encoders ändern. Man kann die Balkenanzeige rotieren lassen oder einen Ton ausgeben. Es kommen immer noch weitere Funktionen hinzu. Updates zu diesem Projekt finden sich auf der Webseite von Elektor.Labs [3]. **Tabelle 1** listet die aktuell verfügbaren Befehle auf.

Tabelle 1. Befehle zur Änderung von Parametern (* = Standard). Jeder Befehl besteht aus zwei Bytes.

Befehl	Daten	Beschreibung
LEDs		
0x00 (0)	0|1*|2	LED-Ring Farbe: aus | rot* | grün
0x01 (1)	0*|1	LED-Ring Modus: Balken* | Punkt
0x02 (2)	0*-15	LED-Ring erste LED (0*)
0x03 (3)	0-15*	LED-Ring letzte LED (15*)
0x04 (4)	0*|1	LED-Ring Drehrichtung: rechts* | links
0x05 (5)	0*-15	LED-Ring Rotation (0*)
0x06 (6)	0-15	LED-Ring LEDx an
0x07 (7)	0-15	LED-Ring LEDx aus
0x08 (8)	0|1*|2	Zentral-LED: aus | rot* | grün
Drehencoder		
0x10 (16)	0*|1	Encoder Modus: normal* | beschleunigt
0x11 (17)	0-255 (0)	Encoder Minimum, Low-Byte (0*)
0x12 (18)	0-255 (0)	Encoder Minimum, High-Byte (0*)
0x13 (19)	0-255 (64)	Encoder Maximum, Low-Byte (64*)
0x14 (20)	0-255 (0)	Encoder Maximum, High-Byte (0*)
0x15 (21)	0-255 (0)	Encoder Wert, Low-Byte (0*)
0x16 (22)	0-255 (0)	Encoder Wert, High-Byte (0*)
Buzzer		
0x20 (32)	0|1*	Buzzer: inaktiv | aktiviert*
0x21 (33)	1	Buzzer: Tonausgabe
0x22 (34)	0-255 (232)	Buzzer Frequenz [Hz], Low-Byte (232*)
0x23 (35)	0-255 (3)	Buzzer Frequenz [Hz], High-Byte (3*)
0x24 (36)	0-55 (64)	Buzzer Dauer [ms], Low-Byte (64*)
0x25 (37)	0-255 (0)	Buzzer Dauer [ms], High-Byte (0*)
Außerdem		
0xfe (254)	0*|1	Ausgabe-Modus: ASCII* | binär
0xff (255)	1	Reset auf Standardwerte

Konfiguration

Eine Voreinstellung von CoCo-ri-Co kann mit dem Trimmpoti P1 vorgenommen werden. Man wählt damit z.B. die Art der Schnittstelle (UART, SPI oder I²C). Mit P1 kann man prinzipiell 32 Werte einstellen. Aus praktischen Gründen bzw. wegen der begrenzten Einstellgenauigkeit mit P1 wurde dies auf acht Werte begrenzt.

In **Tabelle 2** findet sich die genaue Zuordnung dieser acht Wertebereiche zu einzelnen Funktionen der Schnittstelle. Die drei generellen Modi sind: kleine Werte von P1 = UART, Mittelstellung = SPI und große Werte = I²C. Wenn man einen dieser Wertebereiche genau treffen will, geht man wie folgt vor:

- Das Board ausschalten.
- Pin P6 und GND verbinden (Jumper auf die Pins 1 und 2 von K5).
- Das Board einschalten.

Tabelle 2. Mit P1 wählbare Konfigurationen. Im SPI-Modus muss beim Einschalten die SCK-Leitung (P12) high sein (3,3 V), sonst bootet CoCo-ri-Co im ISP-Modus.

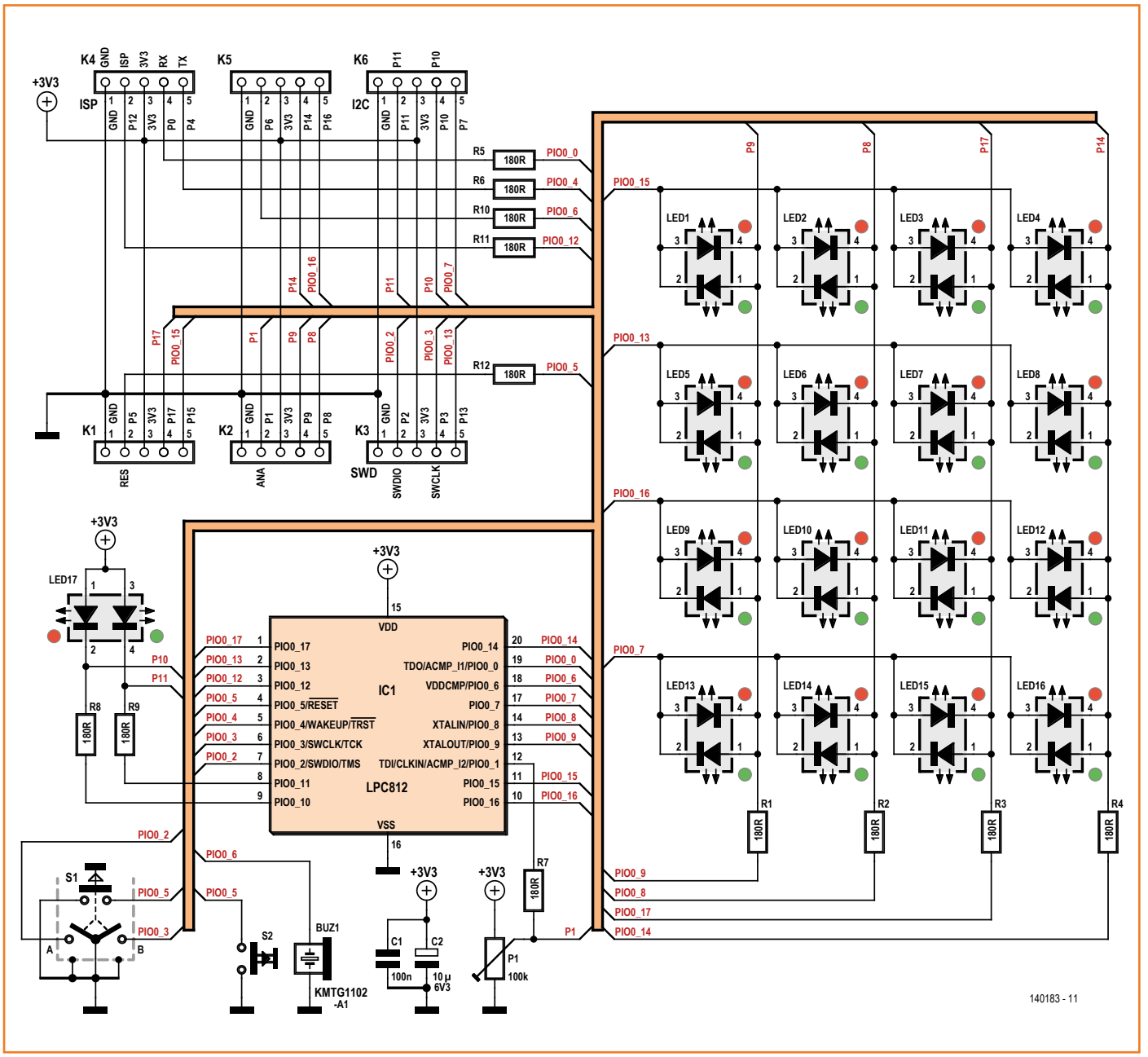
Bereich	Konfiguration	Details	Signale
1	UART, keine Parität, 8 Datenbits, 1 Stoppbit (8N1)	Baudrate = 115.200	P4 = TxD, P0 = RxD
2		Baudrate = 38.400	
3		Baudrate = 9.600	
4	SPI/synchron Slave		P4 = MISO, P0 = MOSI, P12 = SCK (SSEL = ?)
5			
6	I²C, Slave	Adresse 0x1b (29)	P4 = SDA, P0 = SCK (P12 = IRQ)
7		Adresse 0x3a (58)	
8		Adresse 0x74 (116)	

Bild 3.
Die Schaltung von CoCo-ri-Co. Der LPC812-Mikrocontroller des Typs ARM-Cortex-M0+ hat mehr als genug Fähigkeiten für dieses Projekt.

- P1 so einstellen, dass die LEDs den gewählten Bereich anzeigen, und zwar so, dass man die Mitte des zugehörigen Drehbereichs trifft und nicht gerade an den Grenzen liegt, damit sich eine sichere Einstellung ergibt.
- Das Board ausschalten.
- Pin P6 und GND trennen (kein Jumper auf Pin 1/2 von K5).
- Das Board einschalten und Einstellung testen.

Man sollte nicht zu viel mit P1 herumspielen, da so ein SMD-Trimmer eine zerbrechliche Angelegenheit ist.

Hinweis: Bei den Modi SPI oder I²C-Slave muss die angeschlossene Host-Elektronik CoCo-ri-Co regelmäßig pollen. Man kann auch über einen dritten Port den Host darüber informieren, dass neue Daten verfügbar sind. Man beachte, dass ein SPI-Slave-Mode via MCU-SPI-Interface ein SSEL-Signal (Slave Select) benötigt. Sie können frei entscheiden, welchen Port Sie dafür verwenden wollen. Ein „SPI“ via USART im synchronen



Betrieb kommt auch ohne SSEL-Signal aus. CoCo-ri-Co kann mit etwas Programmieraufwand aber auch den SPI-Master spielen, falls nötig.

Die Schaltung

Nachdem nun Funktionen, Einsatzgebiet und Details von CoCo-ri-Co beschrieben sind, wird es Zeit für einen Blick auf den Schaltplan in **Bild 3**. 34 LEDs mit nur 18 I/O-Leitungen ist nicht ganz trivial, oder? Doch wenn man jeweils die zwei Sub-LEDs in einem Gehäuse antiparallel in eine 4x4-Matrix packt, genügen nur acht Leitungen für 16 Zweifarb-LEDs. Diese 16 LEDs bilden den Kreis. Die übrige Zweifarben-LED in der Mitte musste anderweitig angeschlossen werden. Das I²C-Modul der verwendeten MCU unterstützt zwei Interface-Typen: „normal“ mit jedem I/O-Pin und „dedicated“ mit den fix vergebenen Portpins 10 und 11. Diese zwei Portpins können hohe Ströme aufnehmen und eignen sich daher sehr gut als LED-Treiber, weshalb die zentrale LED hier angeschlossen wird. Es steht also nur der Modus „normal“ für I²C zur Verfügung.

Zur Begrenzung des durch die LEDs fließenden Stroms werden noch Vorwiderstände benötigt. Normalerweise braucht es für die rote und grüne Sub-LED unterschiedliche Werte, um auf die gleiche Helligkeit zu kommen, da die rote LED heller als die grüne ist. So krass ist der Unterschied aber nicht und daher befinden sich auf der Platine praktischerweise nur Widerstände mit dem Einheitswert 180 Ω.

Für den Drehencoder wurde eine Version mit integriertem Taster ausgesucht. Ohne Taster sind sie zwar beträchtlich preiswerter, aber die Eingabe von Werten durch Drehen mit abschließendem Tastendruck ist einfach sehr intuitiv. Der Taster ist an die Reset-Leitung der MCU angeschlossen. Das sieht auf den ersten Blick schräg aus. Doch wenn man weiß, dass man die Reset-Funktion dieses Pins via Software abschalten kann, wirkt das weniger merkwürdig. Während der Software-Entwicklung ist eine funktionierende Reset-Funktion sehr hilfreich, aber nach Abschluss der Entwicklung kann man diese getrost deaktivieren. Oder aber man bestückt einen Drehencoder ohne Taster. Wenn statt eines Encoders lediglich ein Taster erforderlich ist, geht auch das: Die Platine hat nämlich auch für Taster passende Bohrungen, die an die Leitungen für den Taster im Encoder angeschlossen sind.

Der Encoder wird in der Mitte des Boards direkt über der LED17 bestückt. Warum das? Zum einen kann man so nur das Bestücken, was die Anwendung erfordert, also LED oder Encoder. Zum anderen kann man mit der LED den Encoder beleuchten. In diesem Fall braucht es einen Drehencoder mit transparenter Achse. Auch die Bohrungen für den reinen Taster befinden sich in der Platinenmitte. Transparente Taster sind mir allerdings nicht bekannt. Auf alle Fälle kann man mit LED17 einen schön beleuchteten Hintergrund realisieren. Viele Anwendungen erfordern Töne oder wie es neudenglisch korrekt heißt: ein hörbares Feedback. Ein Modul mit solch universellem Anspruch muss das natürlich auch können. Von daher kann auch ein Buzzer bestückt werden. Aus Platzgründen kommt eine SMD-Version zum Einsatz. Solche Teile können leider recht teuer sein.

Trimmpoti P1 ist an PIO0_1 angeschlossen. Dieser Pin ist in der MCU mit dem analogen Komparator verbunden. Der Komparator kann 32 Werte diskriminieren. Man könnte damit also einen Schalter mit 32 Stellungen für diverse Software-Optionen realisieren. Zwecks einfacher Erreichbarkeit wurde er auf der LED-Seite der Platine platziert, auch wenn er recht klein ist. Man muss P1 nur dann bestücken, wenn man ihn benötigt.

Die MCU kommt ohne externen Quarz oder Oszillator aus, da sie über einen recht guten internen Taktgenerator verfügt. Ist ein präziserer Takt erforderlich, kann man einen Taktgeber an PIO0_8 und PIO0_9 anschließen. Die Bestückung mit Quarz etc. ist auf der Platine allerdings nicht vorgesehen.

Alle Widerstände der Platine verfolgen denselben Zweck: die Strombegrenzung. Sie haben deshalb unisono den Wert 180 Ω. Die meisten der 18 I/O-Ports haben einen, aber nicht alle. Die beiden SWD-Pins PIO0_2 und PIO0_3 sind ungeschützt, weshalb hier etwas Vorsicht bei Prüfspitzen angesagt ist. Die Reihen-Leitungen der LED-Matrix sind ebenfalls ungeschützt, da es unwahrscheinlich ist, dass etwas von außen daran angeschlossen wird und der Platz auf der Platine begrenzt ist.

Die Kondensatoren C1 und C2 puffern die Stromversorgung.

Erweiterungen

Die MCU ist ebenfalls in der Mitte der Platine bestückt. Das geht natürlich nur auf der Rückseite, auf der sich die Widerstände, die Kondensatoren und der Buzzer befinden. Alle 18 I/O-Pins wurden auf die sechs fünfpoligen Pinheader verteilt, sodass an jedem drei I/Os sowie +3,3 V und Masse anliegen. Die Verteilung der I/Os erfolgte

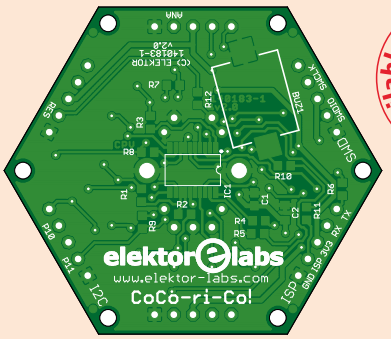
Stückliste

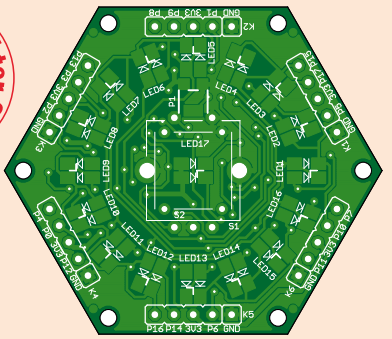
Widerstände:
R1..R12 = 180 Ω, SMD 0603
P1 = 100 k Trimmpoti, SMD 3 mm, z.B. Bourns TC33X-2-104E

Kondensatoren:
C1 = 100 n, SMD 0603
C2 = 10 µ / 6,3 V, SMD 0805

Halbleiter:
IC1 = LPC812M101JDH20FP (TSSOP20)
LED1..LED17 = LED, zweifarbig rot/grün, Dialight 5988610207F

Außerdem:
BUZ1 = Buzzer, Kingbright KMTG1102-A1
S2 = Taster, Multimec 3FTL6





S1 = Drehencoder mit/ohne Taster, z.B. Alps EC12E2424407
K1..K6 = 5-pol. Pinheader, horizontal oder vertikal (optional)

Fertiges Modul: Elektor Shop # 140183-91, siehe www.elektor.de
Nackte Platine: Elektor Shop # 140183-1 (v2.0)

natürlich nicht rein zufällig, sondern nach Funktionen. K1 und K4 eignen sich z.B. für ISP-Zwecke (**I**n-**S**ystem **P**rogramming). K4 ist der eigentliche ISP-Anschluss, jedenfalls beinahe. Er ist kompatibel mit einem 3,3-V-USB/Seriell-Konverter-Kabel von FTDI. An K1 liegt der Reset-Pin. K1 und K4 befinden sich auf entgegengesetzten Seiten der Platine und 38 mm voneinander entfernt. Wenn man möchte, kann man sich sehr einfach einen passenden Programmieradapter auf einer Experimentierplatine aufbauen. So ein Adapter sollte einen kleinen Spannungsregler mitbringen, der aus den 5 V des 3,3-V-FTDI-Kabels (tatsächlich!) die benötigten 3,3 V macht. Noch passender ist der USB/Seriell-Adapter BOB (110553-91) von Elektor [4]: Er enthält nicht nur einen FT232R, sondern stellt auch schon die nötigen 3,3 V zur Verfügung.

K2 kann man als Analog-Schnittstelle bezeichnen, denn seine Pins sind mit dem Komparator des Analog-Eingangs Nr. 2 verbunden (ACMP_I2, PIO0_0 kann als Eingang 1 verwendet werden). Zusammen mit PIO0_8 oder PIO0_9 kann man damit einen kapazitiven Touch-Sensor betreiben. Bei bestücktem P1 steht an Pin 2 von K2 eine einstellbare Spannung zur Verfügung.

Über K3 hat man Zugang zum SWD-Port (**S**erial **W**ire **D**ebug) der MCU. Dieser Port wird zwar auch mit dem Drehencoder geteilt, aber solange man diesen nicht bedient, ist sein Kontakt offen (bei den üblichen rastenden Ausführungen bei Positionierung zwischen den Rastern). Die Belegung von K3 entspricht zwar nicht dem SWD-Standard, ist aber mit einem passenden Adapter verwendbar.

K5 beherbergt digitale I/Os. Sein Pin 2 (PIO0_6) ist mit einem strombegrenzenden Widerstand geschützt, da er keine 5 V toleriert (alle anderen Pins schon). Die anderen Widerstände befinden sich an Ports, die häufig angeschlossen und wieder getrennt werden.

K6 ist ein schnelles I²C-Interface, das mit den Pins 10 und 11 (und der mittigen LED) verbunden ist.

Selbstverständlich sind die meisten Portpins der Pinheader auch mit der LED-Matrix und anderer Peripherie auf der Platine verbunden. Also muss man schon etwas aufpassen, welche Ports man für welche Anwendung mit was verbindet. **Tabelle 3** fasst die Zusammenhänge zwischen I/O-Ports, Pinheadern und Funktionen zusammen.

Tabelle 3. Die Verknüpfung von I/O-Ports mit Pinheadern und Funktionen.

I/O	K	Funktion	I/O	K	Funktion	I/O	K	Funktion
0	4	RxD/frei	6	5	Buzzer	12	4	ISP/frei
1	2	P1/Analog	7	6	Row 3	13	3	Row 1
2	3	SWDIO/S1A	8	2	Column 1	14	5	Column 3
3	3	SWCLK/S1B	9	2	Column 0	15	1	Row 0
4	4	TxD/frei	10	6	LED17R/I²C	16	5	Row 2
5	1	Reset/Taster	11	6	LED17G/I²C	17	1	Column 2

Laut dieser Tabelle ist K4 der beste Anschluss für die Verbindung mit der Platine der eigentlichen Anwendung. Im ISP-Modus (bei einem Reset der MCU durch die Verbindung von PIO0_12 und Masse) bilden PIO0_0 und PIO0_4 zwingend eine serielle Schnittstelle. Im Normalbetrieb sind diese Pins frei und sie können z.B. SPI, I²C oder einer anderen Funktion zugewiesen werden. Aus der Reset-Leitung PIO0_12 kann man einen normalen I/O-Port machen. PIO0_12 benötigt keinen extra Pull-up-Widerstand, weil schon einer in der MCU angeschlossen ist (wie bei allen I/O-Portpins außer PIO0_10 und PIO0_11). Auf diese Weise wird nicht versehentlich der ISP-Modus aktiviert, wenn der Pin offen bleibt.

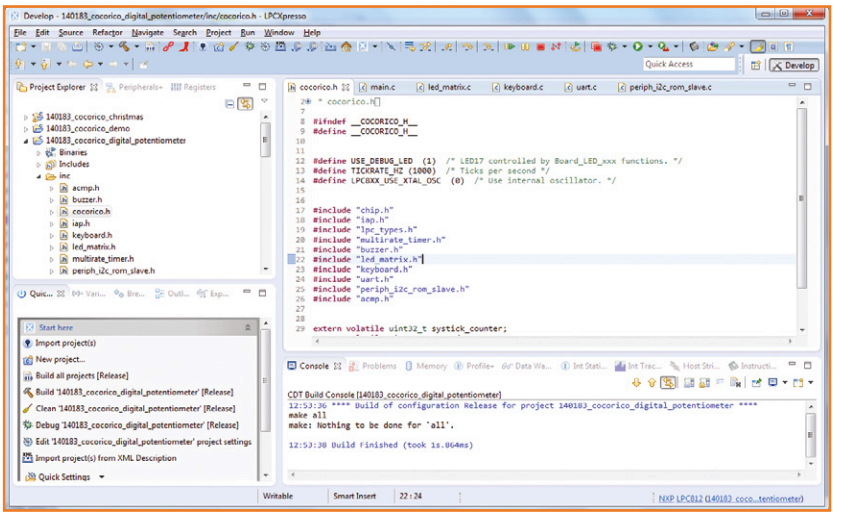


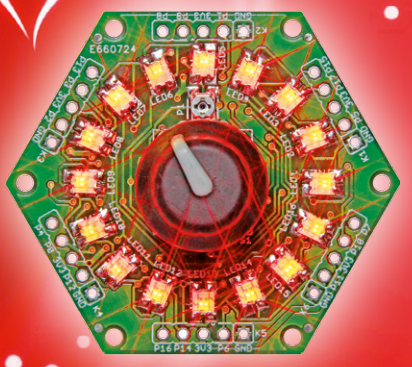
Bild 4.
Zur Entwicklung der CoCo-ri-Co-Software wurde die LPCXpresso-IDE eingesetzt. Auch wenn hier nicht abgebildet: Das Projekt mit dem Namen lpc_chip_8xx_lib ist Teil des Workspace.

Projekt-Software

NXP hat eine Webseite speziell für LPC-Mikrocontroller eingerichtet [5], auf der man Libraries findet und wodurch man sich die Arbeit spart, einen eigenen Low-Level-Hardware-Abstraction-Layer zu konstruieren. Diese Libraries gibt es paarweise: eine für die Chip-Familie und eine andere für das Evaluations- und/oder Demo-Board. Da hier ein LPC812-Chip verwendet wird, muss man die LPC8xx-Pakete suchen. Da die LPCXpresso-Compiler-Tools verwendet werden (**Bild 4**), muss man das Paket für dieses Tool runterladen.

Die Libraries (Version 2.01) sind auch im Download für diesen Artikel [6] enthalten. Es empfiehlt sich, diese nicht durch neuere Versionen von der Webseite zu ersetzen, ohne zuvor umfangreiche Vergleiche anzustellen, denn in den via Elektor verfügbaren Libraries habe ich schon den einen oder anderen Bug gefixt.

Der LPCXpresso-Workspace für CoCo-ri-Co enthält zwei Projekte: `lpc_chip_8xx_lib` (die NXP-Libraries) und `140183_cocorico_digital_potenti-`



Weihnachts-Karaoke

CoCo-ri-Co vereint 17 zweifarbige LEDs in rot und grün mit einem Buzzer auf einer kleinen (fast sternförmigen) Platine. Die Elektronik kann durchaus von einer 3-V-Knopfzelle versorgt werden. An jeder Ecke ist ein Befestigungsloch. Was braucht man mehr für (elektronischen) Weihnachtsschmuck? Einfach in den Weihnachtsbaum hängen bzw. als Brosche, am Ohr oder am Handgelenk tragen. Oder verpacken – ein tolles Weihnachtsgeschenk.

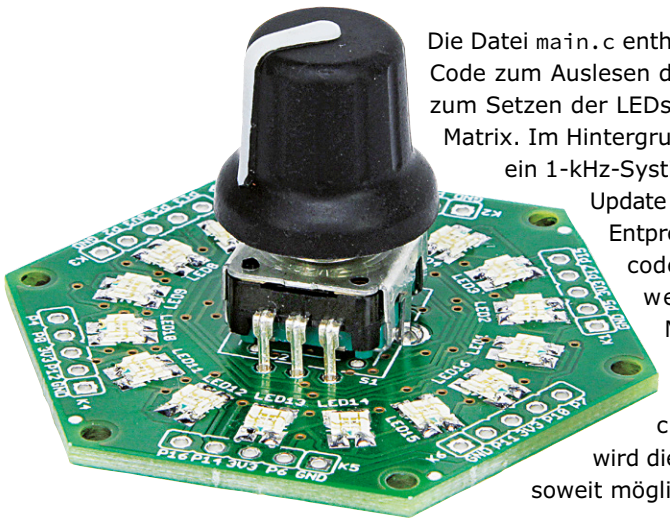
Als Anregung habe ich ein CoCo-ri-Co-Weihnachtsprojekt entwickelt, das weihnachtlich zur Melodie von „Jingle Bells“ seine LEDs glitzern lässt. Man nehme einen passenden Adapter, verbinde ihn mit der seriellen Schnittstelle eines PCs oder Laptops und starte ein Terminal-Programm wie *Tera Term*, um den Text anzuzeigen und die Karaoke-Session zu starten (115.200 Baud, 8N1). Der Text flimmert jetzt über den Bildschirm und sie können im Kreis Ihrer Lieben zusammen mit CoCo-ri-Co singen. Weihnachten mal anders.

Der Takt wird von der mittigen LED angegeben. Das Tempo kann mit dem Drehencoder (auf der Rückseite montieren, damit die zentrale LED sichtbar bleibt) eingestellt werden. Die Dateien finden sich unter [3] und/oder [6]. Das Projekt kann sehr einfach bezüglich Melodie und Text geändert werden. Weihnachten ohne CoCo-ri-Co ist ab sofort undenkbar...

Wissenswertes über LPC812-I/O-Ports

- In Tabelle 3 sind die Ports 10 und 11 als I²C ohne Signalangabe (SDA oder SCK) aufgeführt. Es liegt an Ihnen, welcher Port welches Signal transportiert. Siehe den Abschnitt über die *Switch Matrix* im Manual der LPC81x-Serie. Diese Ports sind Open-Drain-Ausgänge und eignen sich nicht als schnelle SPI- oder UART-Schnittstelle.
- Die Portpins 2, 3 und 5 sind nach dem Einschalten mit SWD und Reset belegt. Man muss sie per Software rekonfigurieren, bevor man sie für andere Zwecke gebrauchen kann. Sie werden via *Pin Enable Register 0* konfiguriert.
- Als digitale Ports tolerieren alle 5 V außer PIO0_6. Ports mit analogen Funktionen (PIO0_0, PIO0_1 und PIO0_6) tolerieren ebenfalls 5 V, wenn ihre analoge Funktion deaktiviert wird.
- Wenn die Reset-Funktion deaktiviert wird, kommt man dennoch in den ISP-Modus, indem PIO0_12 beim Einschalten auf low gehalten wird.
- Die LPC81x-Familie gibt es in unterschiedlichen Gehäusen. Nur in den frühen Ausführungen mit 16- und 20-poligen Gehäusen vor der Version „4C“ wird der ISP-Mode via PIO0_1 statt PIO0_12 aktiviert. Wenn ihre MCU den ISP-Mode verweigern sollte, sollten sie die Version überprüfen.
- Eine jungfräuliche LPC81x-MCU bootet direkt im ISP-Mode, sodass man die MCU zumindest einmal programmieren kann.

`ometer` (der richtige Code). Beide sind schon im herunterzuladenden Archiv enthalten und können direkt in LPCXpresso ohne vorheriges Auspacken importiert werden.



Die Datei `main.c` enthält den zentralen Code zum Auslesen des Encoders und zum Setzen der LEDs in der virtuellen Matrix. Im Hintergrund kümmert sich ein 1-kHz-Systick-Timer um das Update der LEDs und das Entprellen des Drehencoders. Buzzer-Töne werden mit dem Multi-Rate-Timer generiert. Um Programm-Speicher zu sparen wird die Kommunikation soweit möglich von der internen Hardware der MCU übernommen.

Ein Blick in den in reinem C gehaltenen Source-Code (und das Herumspielen damit) ist empfehlenswert. Hierzu kopiert man einfach das Projekt mit dem digitalen Poti unter einem neuen Namen in den gleichen Workspace und kann es dann modifizieren. Es dürfte nicht allzu schwer zu verstehen sein. Anregungen und Kommentare sind jedenfalls willkommen. Bei einem „Build“ wird eine Hex-Datei erzeugt, die man mit dem Tool *Flash Magic* [7] über die serielle Schnittstelle in die MCU laden kann.

CoCo-ri-Co ist auch als fertiges Modul im Elektor Shop erhältlich, genauso wie bloße Platinen zum Selberbestücken. Fragen und Anregungen kann man am besten auf der Projektseite unter Elektor.Labs [3] einstellen.

(140183)

Weblinks

[1] J²B: Elektor September 2011; www.elektor.de/110274; www.elektor-labs.com/node/3832

[2] Platino: Elektor Oktober 2011; www.elektor.de/100892; www.elektor-labs.com/node/2288

[3] CoCo-ri-Co auf Elektor.Labs: www.elektor-labs.com/node/4257

[4] Elektor 110553 BoB-FT232R: www.elektor.de/110553

[5] LPCOpen: www.lpcware.com

[6] Projekt-Downloads: www.elektor.de/140183

[7] Programmier-Tool Flash Magic: www.flashmagictool.com

elektor ACADEMY
the school of electronics

2-tägiges Seminar USB-Treiber für Mikrocontroller

Jedes Gerät – unabhängig ob es für die Industrie oder den Endverbraucher ist – benötigt meist eine Schnittstelle zum Konfigurieren oder Bedienen der Anwendung. Nach vielen Jahren hat USB endgültig die RS232-Schnittstelle abgelöst.

Lange konnte man um USB einen großen Bogen machen, in dem man eine USB-RS232-Bridge für den PC verwendete. Die virtuelle COM-Schnittstelle bietet aber oft nicht die notwendigen Eigenschaften, die von der Anwendung aus gewünscht sind (Problem 1 Millisekunden Frame, eigene Benutzerkennung usw.).

Moderne Mikrocontroller bieten seit einiger Zeit USB-Schnittstellen direkt an. Diese kann mit dem notwendigen Know-how einfach in die eigene Anwendung integriert werden. Viele Beispiele existieren, die man mit einfachen Handgriffen anpassen kann.

In diesem Seminar wird gezeigt, wie man sich basierend auf den integrierten USB-Schnittstellen der Mikrocontroller eine Kommunikation mit Betriebssystemen (Windows und Linux) über Treiber und Bibliotheken aufbauen kann. Das Seminar ist mit vielen praktischen Beispielen abgerundet, um verschiedene Möglichkeiten für eine Anbindung zu sehen. Kennt man erst einmal diese und hat dazu ein paar passende Beispiele, kann jeder USB so verwendet werden, wie es die Anwendung erfordert.

Veranstaltungsort/-termin: **Augsburg, 10. + 11. Dezember 2014**

Referent: **Dipl.-Inf. (FH) Benedikt Sauter**

Teilnahmegebühr: **749,00 Euro (inkl. MwSt.)**

Erwartet: Februar 2015 - SMD Prototypen-Bestückung von BGA-Bausteinen Seminar, am Beispiel des Embedded Linux Boards GNUBLIN (Augsburg)

Im Preis sind Mittagessen, Seminarunterlagen, Dokumentation und Teilnahmezertifikat inbegriffen.

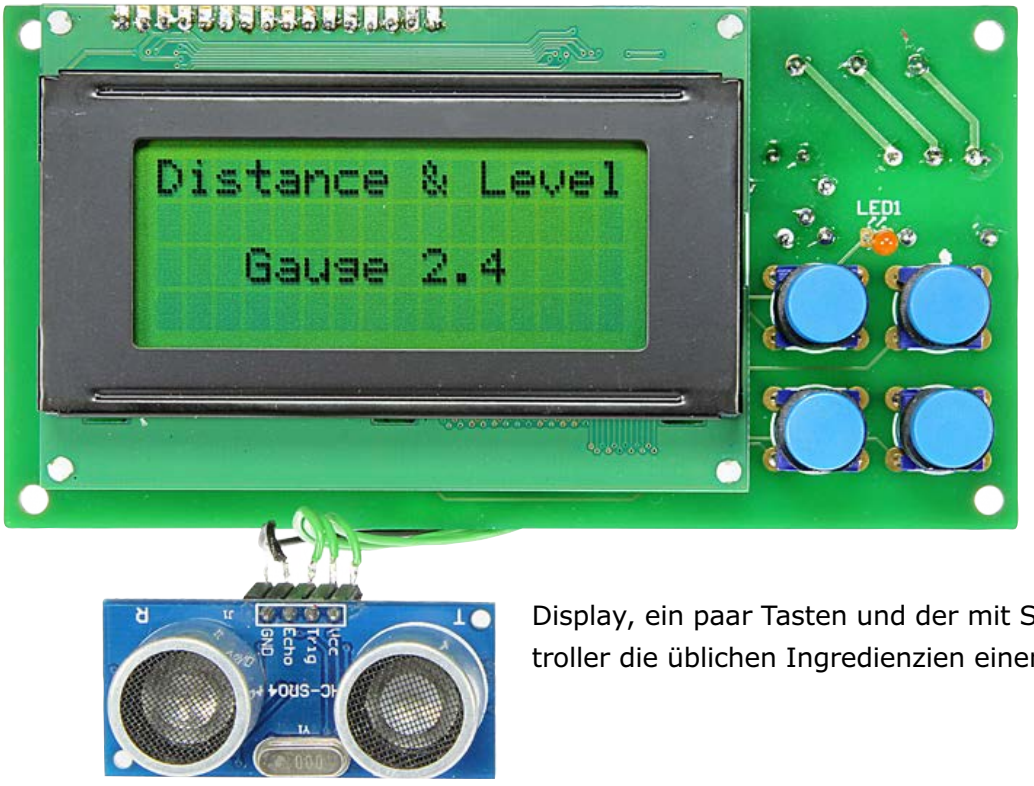
Weitere Infos & Anmeldung: www.elektor.de/events

Elektor-Abomitglieder erhalten 5% Rabatt auf den Workshop-Preis!

Elektor-Verlag GmbH | Süsterfeldstr. 25 | 52072 Aachen
Tel. +49 241 88 909-16
www.elektor.de
j.grotenrath@elektor.de

Pegel- und Distanzmessgerät

Mit Alarmfunktion



Von **Jörg Trautmann**

Kleine, gut erhältliche und spottbillige Ultraschallsensoren ermöglichen den Entwurf von einfachen bis luxuriösen Messgeräten für Entfernungen aller Art. Neben dem Modul sind ein Display, ein paar Tasten und der mit Software gefüllte Mikrocontroller die üblichen Ingredienzien einer solchen Schaltung.

Sie möchten den Füllstand eines Öltanks oder einer Regenwasserzisterne ohne großen Aufwand in Erfahrung bringen? Sie füllen eine große Tonne mit dem Gartenschlauch mit Wasser auf und möchten, dass die Wasserzufuhr automatisch über ein Magnetventil gestoppt wird, wenn ein bestimmter Füllstand erreicht ist? Oder Sie möchten einfach nur die Entfernung zwischen zwei Objekten messen? Dem Ideenreichtum für den Einsatz des hier beschriebenen Pegel-/Entfernungsmessgeräts sind kaum Grenzen gesetzt! Den Füllstand eines zylindrischen oder quaderförmigen Behälters kann man mechanisch mit Schwimmer und Potentiometer, kapazitiv, mit Ultraschall oder mit einem Laser ermitteln. Sowohl die mechanische als auch die kapazitive Variante erfordern den Einsatz zusätzlicher Komponenten wie Schwimmer und Sensoren. Wenn die Genauigkeit der Messung im Vordergrund steht, ist sicherlich ein Laser die beste Lösung, allerdings hat

Eigenschaften

- Pegelstandsmessung von Flüssigkeiten
- Überwachungsfunktion des Pegelstandes mit Relaisausgang und LED-Anzeige
- Stufenlos programmierbarer Min/Max-Alarmpegel
- Speicherung von Min/Max-Kalibrierwerten für bis zu zehn Behälter oder Tanks
- Distanzmessung
- Intuitive Menüführung über LCD-Display

auch diese Messmethode ihre Tücken. So stören beispielsweise Nebel und Dampf die Reflexion des Lasers und verfälschen das Messergebnis.
Von solchen Einflüssen lässt sich Ultraschall kaum beeindrucken. Ich habe mich für diese kostengünstige Variante mit Ultraschallsensor entschieden. Ein weiterer Vorteil dieser Lösung besteht darin, dass kein Kontakt zum flüssigen Medium hergestellt werden muss. Und so nebenbei erhalten wir zusätzlich noch ein Messgerät zur Distanzmessung. Die bei Ultraschallwandlern übliche Frequenz von 40 kHz entspricht bei einer Lufttemperatur von 20 °C einer Wellenlänge von 8,5 mm. Audiowellen in diesem Frequenzspektrum lassen sich gut bündeln und breiten sich keulenförmig mit einem Öffnungswinkel von ungefähr 15° aus. Perfekte Voraussetzungen also für den gewünschten Einsatzzweck, da die Seitenwände des Behälters so gut wie keinen Einfluss auf die Reflexion des Signals gewinnen.

Der hier verwendete Ultraschallwandler ist vor wenigen Monaten, in der Sommerausgabe 2014, schon ausgiebig in Elektor beschrieben worden [1]. Deshalb hier seine Funktion in Kurzform: Ein Trigger-Impuls veranlasst das US-Modul, ein Burst-Signal auszusenden. Dieses Signal wird vom zu messenden Objekt, etwa der Wasseroberfläche in einer Regentonne, reflektiert und vom US-Modul als Echosignal wieder aufgefangen. Da innerhalb der Echozeit der Schall die Distanz zum reflektierenden Objekt doppelt zurücklegt, ist:

Distanz =
0,5 x Schallgeschwindigkeit [m/s] x Echozeit [s]

Schaltung und Bauteile

Die Schaltung in **Bild 1** basiert auf drei Komponenten, dem altbewährten Mikrocontroller ATmega8, einem angeschlossenen LCD und dem Ultraschall-

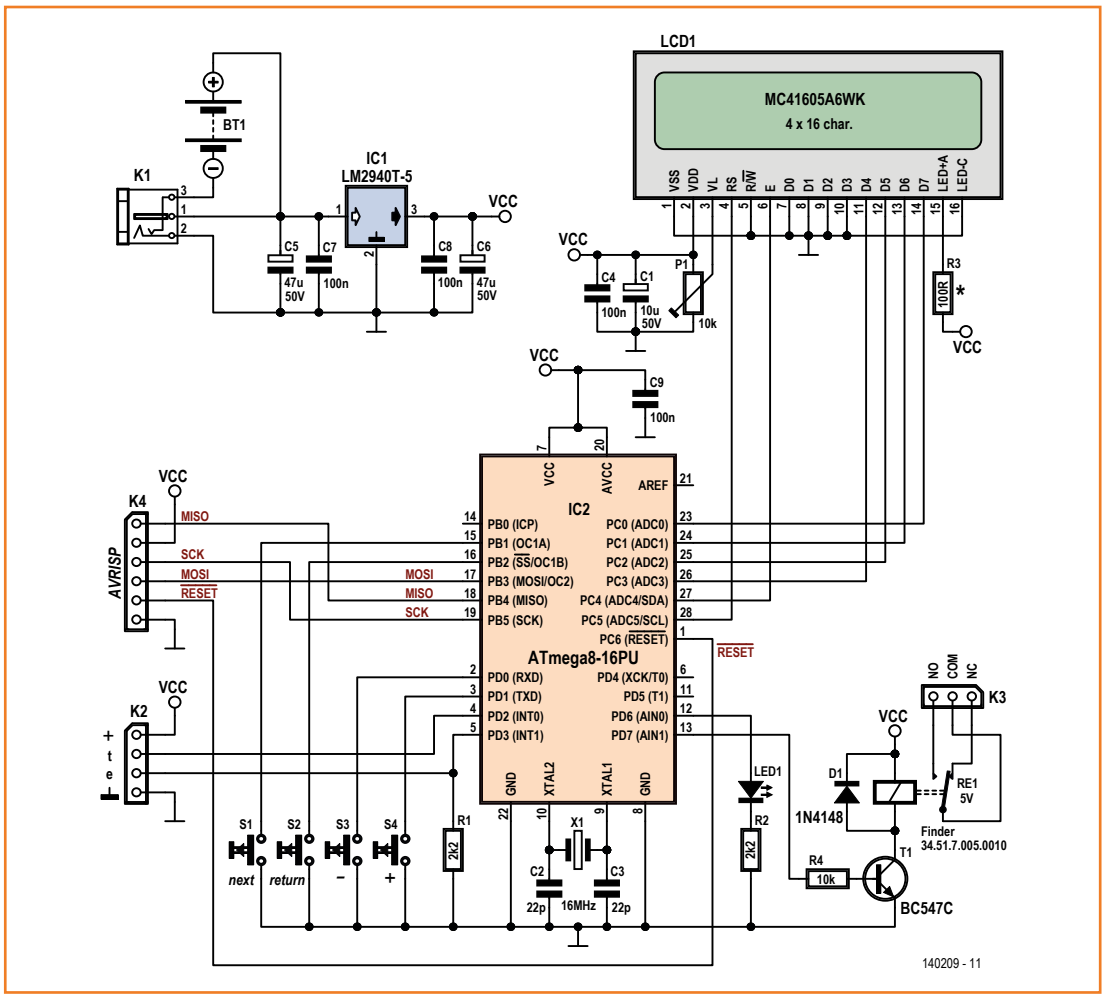


Bild 1.
Die übersichtliche Schaltung des Pegel-/Entfernungsmessers.

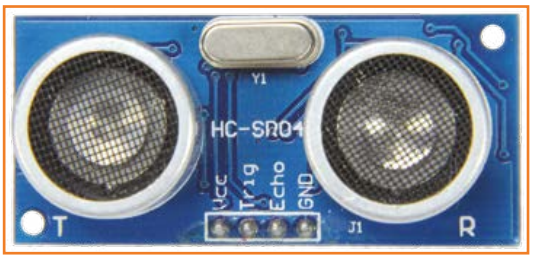


Bild 2.
Das kleine US-Modul HC-SR04.

modul. Neben dem Modul HC-SR04 (**Bild 2**) kann auch ein nahezu baugleiches US-020 zum Einsatz kommen. Beide Module sind mit einem Preis von ungefähr 4 € unschlagbar günstig für den Einsatz in einem selbstgebautem Messgerät.
Hersteller der Module sind nur schwer zu ermitteln, sie sind aber leicht bei Elektronikhändlern erhältlich, vor allem solchen, die sich auf Robotik spezialisiert haben. Wer aber die Datenblätter durchliest, wird diverse Unterschiede entdecken. So wird die Erfassungsreichweite für die beiden Modelle einmal mit 2 cm...3 m ausgewiesen, ein anderes mal mit 3 cm...7 m. Recherchiert man weiter, finden sich noch andere abweichende Angaben; auch für ein- und dasselbe Modell. Demnach liegt die Stromaufnahme je nach Datenblattquelle zwischen 3 mA und 15 mA. Zuverlässige Angaben lassen sich daher nur im Selbstversuch ermitteln. Weitestgehend einig sind sich die Anbieter der Module in folgenden Angaben:

- Betriebsspannung: 5 VDC
- Betriebstemperatur: 0...70 °C
- Ausbreitungswinkel: 15°
- Arbeitsfrequenz: 40 kHz
- Trigger-Inputsignal: 10 µs

Der minimale Abstand von Sender- und Empfängerkapsel ist beim Modul HC-SR04 so klein, dass problemlos Messungen in Kanistern durchgeführt werden können (bei leicht brennbaren Flüssigkeiten wegen der Explosionsgefahr nur mit abgesetztem US-Modul!). Wer Messungen in Behältern mit besonders kleinem Öffnungsdurchmesser durchführen möchte, kann zum US-Modul SRF02 ([3], nicht pinkompatibel, aber mit serieller/I2C-Schnittstelle!) greifen, das nur eine Kapsel sowohl zum Senden als auch zum Empfang des Ultraschallsignals verwendet. Da der Sende-/Empfangswinkel sehr spitz ist, muss man bei diesem Modell allerdings in Kauf nehmen, dass der Mindestabstand zum zu messenden Medium 16 cm beträgt.
Um die Menüführung zu erleichtern, kommt ein vierzeiliges 16-stelliges LC-Display zum Einsatz. An P1 wird der Kontrast festgelegt. R3 ist der Vorwiderstand der Hintergrundbeleuchtung und sollte etwa R3 = $(5V\text{-}V_f)/I_{nom}$ betragen. Hier muss man das Datenblatt des Displays zu Rate ziehen. Bei manchen LC-Displays ist sogar der Vorwiderstand intern schon vorhanden, so dass man R3

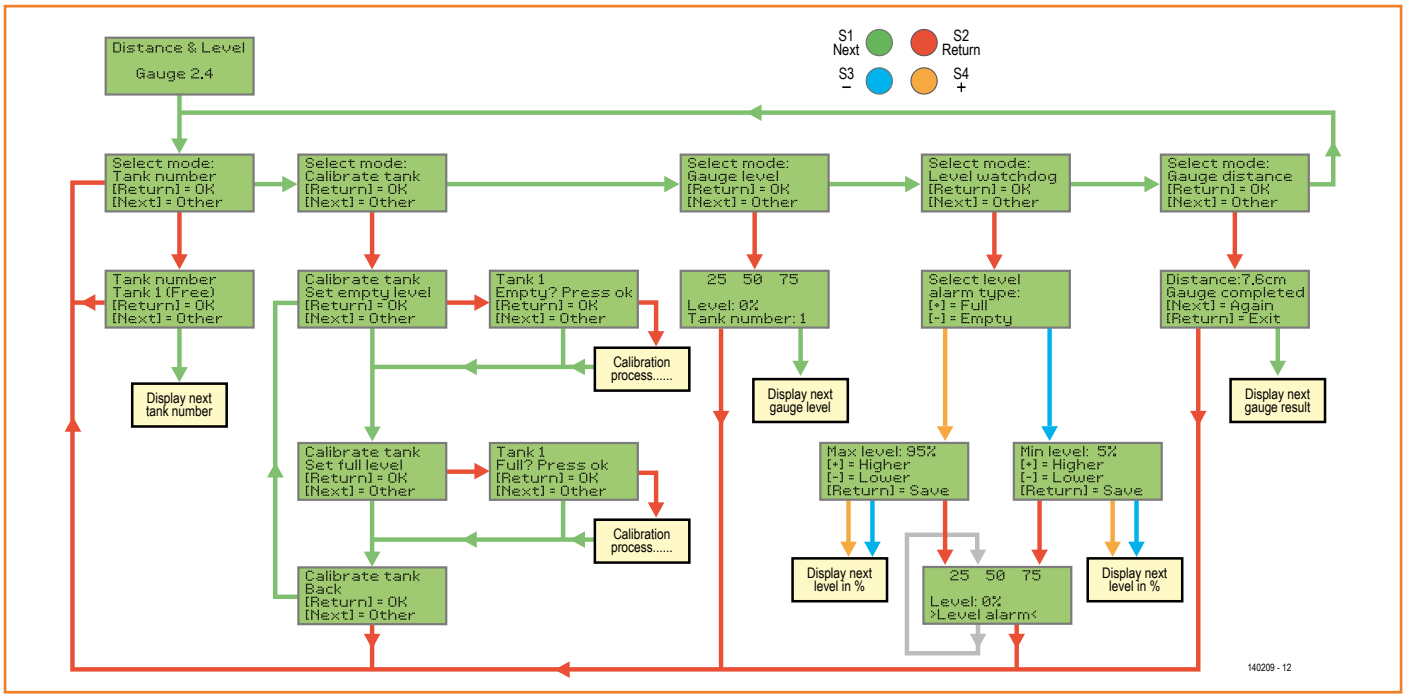


Bild 3.
Ablaufdiagramm der Software.

durch eine Drahtbrücke ersetzen kann. Auch eine Kombination mit einem Schalter ist aus Energiespargründen denkbar.
Das Wechsler-Relais RE1, dessen Kontakte an K3 verfügbar sind, wird über Portpin PD7 und Treibertransistor T1 angesteuert, die Alarm-LED (low-current) über den Portpin PD6. Ein Low-Drop-Spannungsregler LM2940CT-5 sorgt für eine stabile 5-V-Spannungsversorgung aus vier 1,5-V-Batterien. Dieser etwas exotische Spannungsregler liefert bis zu 1 A und ist damit weitaus stärker belastbar als der übliche 78L05 mit seinen 100 mA. Natürlich kann das Messgerät auch über ein Steckernetzteil (<26 VDC) an K1 betrieben werden. Die Schaltung zieht mit eingeschaltetem Relais und LCD-Beleuchtung etwa 150 mA. Um eine möglichst hohe Messgenauigkeit zu erreichen, wird der Mikrocontroller mit einem externen 16-MHz-Quarz betrieben. Ein paar passive Bauteile komplettieren die Schaltung.

Die Software

Das Programm des Mikrocontrollers muss folgende Funktionen abdecken:

- Pegelstandsmessung von Flüssigkeiten
- Überwachungsfunktion des Pegels mit Relaisausgang und LED-Anzeige
- Stufenlos programmierbarer Min/Max-Alarmpegel
- Speicherung von Min/Max-Kalibrierwerten für bis zu zehn Behälter oder Tanks
- Distanzmessung
- Intuitive Menüführung über LC-Display
- Offsetkorrektur bei Distanzmessungen
- Manuelle Divisor-Korrektur zur Anpassung an niedrige und hohe Temperaturen

Die Software folgt dem Ablauf wie in **Bild 3** dargestellt. Angesichts des beschränkten Speicherplatzes im ATmega8 wurden die Texte nur in englischer Sprache verfasst. Die Bedienung erfolgt über die vier Tasten *Return*, *Next*, *Plus* und *Minus*. Wegen der Komplexität des Programmcodes, der zusammen mit der Hex-Datei und den Platinenlayouts unter [2] zum Download bereit steht, wurde dieser in sinnvolle Prozeduren aufgegliedert und mit entsprechenden Kommentaren versehen, so dass sich auch Anfänger schnell zurechtfinden sollten. Der eigentliche Messvorgang: Um das an K2 angeschlossene US-Modul zum Aussenden eines Signals zu veranlassen, wird über Portpin PD2 ein Impuls von 10 µs Länge mit fallender Flanke zum Trigger-Eingang geschickt. Das US-Modul sendet daraufhin nach etwa 250 µs ein Burst-Signal mit einer Frequenz von 40 kHz und einer Länge von 200 µs aus. Der Echo-Ausgang, der über den Portpin PD3 am Mikrocontroller angeschlossen ist, schaltet nun auf H-Pegel und das US-Modul wartet auf den Empfang des reflektierten Signals. Trifft selbiges ein, kippt der Echo-Ausgang wieder zurück auf Low. Währenddessen tickt Timer1, so dass nach dem Stopp des Timers die Entfernung berechnet werden kann. Sollte nach 50 ms kein Echosignal eingetroffen sein, wird der Messvorgang für den aktuellen Zyklus abgebrochen. Um ein möglichst genaues Messergebnis zu erreichen, umfasst ein Messzyklus 16 Messungen. Der Programmcode hierzu kann in der Prozedur `Gauge_distance()` nachgelesen werden.

Ein Hinweis für „Selbstbrenner“, denen über K4 Zugriff auf die Programmierschnittstelle des Controllers gewährt wird: Da das Programm sehr umfangreich ist, ist der 8 K große Flashspeicher zu 100 % voll. Die Demoversion von BASCOM-AVR kann daher nicht zum Kompi-

Kasten 1. Temperaturabhängigkeit der Schallgeschwindigkeit

Die Schallgeschwindigkeit c_0 in Luft beträgt bei einer Temperatur von T = 0°C gemessene 331,5 m/s. Die Schallgeschwindigkeit ist nicht vom Luftdruck abhängig, sondern lediglich von der Lufttemperatur ϑ in Grad Celsius:

$$c_\vartheta = c_0 \cdot \sqrt{1+\alpha \cdot \vartheta}$$

dabei ist α der Ausdehnungskoeffizient von 1/273,15 = 3,661 x 10^{-3} 1/°C. Daraus ergibt sich die Näherung:

$$c_\vartheta = 20{,}063 \cdot \sqrt{\vartheta + 273{,}15} \text{ in m/s}$$

Man sieht, der Temperatureinfluss ist für präzise Messungen nicht unerheblich. Hier einige Beispiele für die Schalllaufzeit bei 0 °C und 20 °C.

Entfernung in cm	Laufzeit bei 20°C in ms	Laufzeit bei 0°C in ms
2	0,117	0,121
10	0,583	0,603
50	2,915	3,017
100	5,831	6,033
200	11,662	12,066
300	17,492	18,100

lieren verwendet werden, da sie nur 4 K, also 50 % Speicherauslastung erlaubt. Beim Brennen kann man schnell in eine Falle tappen: Da der ATmega8 in dieser Schaltung mit einem externen 16-MHz-Quarz arbeitet, ist das Setzen zweier Fuses erforderlich. CKOPT sollte aktiviert werden, damit der Oszillator sicher schwingt. Der Stromverbrauch erhöht sich durch diese Maßnahme etwas, aber man erkauft sich dadurch ein stabiles Schwingverhalten. Unerlässlich ist das Setzen von SUT_CKSEL auf Ext. Crystal/Resonator High Freq.: Start-up time: 1K CK + 64 ms! Die Optionsliste bietet eine große Auswahl an Einstellungen. Wer sich hier vertut, muss seinen ATmega8 nach dem Programmiervorgang unter Umständen beerdigen, da er nicht mehr alleine auf die Füße kommt. Also besser zweimal lesen, welche Einstellung ausgewählt wurde. Im Download-Paket befindet sich ein AVR-Studio-Screendump der korrekten Fuseeinstellungen.

Aufbau und Inbetriebnahme

Der Aufbau gestaltet sich recht einfach angesichts der überschaubaren Anzahl von Bauteilen. Die doppelseitige Platine (**Bild 4**) können Sie bei Elektor erstehen oder auch selber ätzen. Entsprechende Dateien finden Sie im Downloadpaket [2]. Etwas Handarbeit und Geschick ist erforderlich, um am Gehäuse die Aussparungen für die LCD-Anzeige, die Ultraschallkapseln, die Taster und den Schalter anzubringen. Wie die US-Kapseln angeordnet werden, ist eher Geschmackssache. Auch eine separate Installation des US-Moduls über ein längeres abgeschirmtes Kabel ist

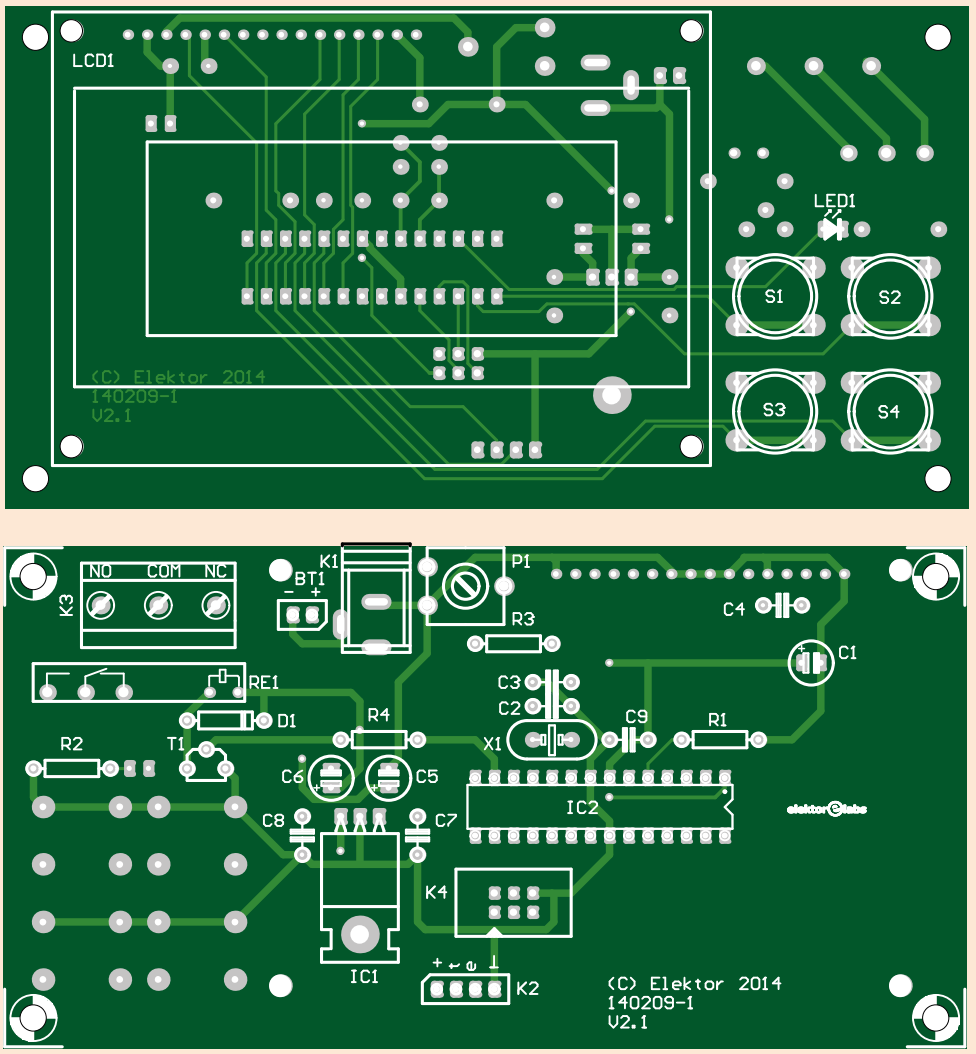


Bild 4.
Ober- und Unterseite der Platine.

Stückliste

Widerstände:
R1,R2 = 2k2
R3 = 0 (Drahtbrücke)
R4 = 10 k (Kohlefilm 250 mW, 250 V)
P1 = 10 k Trimmpoti

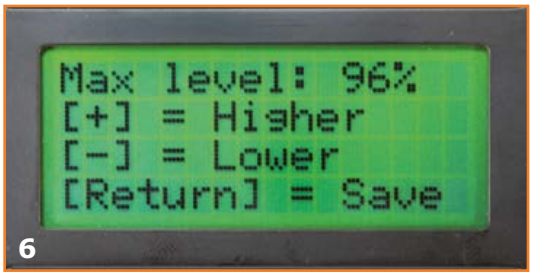
Kondensatoren:
C1 = 10 µ, 50 V, RM 2 mm
C2,C3 = 22 p, 50 V, RM 2,5 mm, C0G/NP0
C4,C7,C8,C9 = 100 n, 50 V, 20%
C5,C6 = 47 µ, 50 V, RM 2,5 mm

Halbleiter:
D1 = 1N4148
LED1 = Low current LED rot, 3 mm
T1 = BC547C
IC1 = LM2940T-4.0
IC2 = ATmega8-16PU, programmiert (140209-41)

Außerdem:
K1 = DC-Hohlbuchse, 1,95-mm-Pin (Farnell 1217037)
K2 = 1x4-polige Stiftleiste
K3 = 3-polige Platinenanschlussklemme RM 5 mm
K4 = 2x3-poliger Header mit Wanne
LCD1 = vierzeiliges LCD-Modul 4x16
RE1 = 5-V-Relais 1 x um (Finder 34.51.7.005.0010 bei Farnell 1169338)
S1...S4 = Taster Multimec RA3FTL6 (Farnell 1132885)
4 Kappen für Multimec 1D03 (Farnell 1132887)
X1 = Quarz 16 MHz, 18 pF
US-Modul HC-SR04 oder US-020 oder SRF02

denkbar, falls die Ablesung in einem anderen Raum erfolgen soll.
Ist der programmierte ATmega8 bestückt und das US-Modul angeschlossen, kann das Gerät erstmals eingeschaltet werden. Zumindest die LCD-Hintergrundbeleuchtung sollte leuchten. Ist dies nicht der Fall, die Stromversorgung unterbrechen und die Bestückung überprüfen! In der Startphase sollte für zwei Sekunden *Distance & Level Gauge 2.4* (**Bild 5**) im Display zu lesen sein.
Nun können die Grundeinstellungen vorgenommen werden. Hierzu das Gerät wieder aus- und mit gedrückter *Return*-Taste wieder einschalten. Nach dem Loslassen der Taste wird nach der Breite des Gehäuses gefragt. Hier kann in 5-mm-Schritten mit den Tasten *Plus* und *Minus* ein Offset angegeben werden, der die Entfernung der Oberseite des US-Moduls zur Anlegekante des Gehäuses berücksichtigt. So kann man später bequem zum Beispiel den Abstand von Wand zu Wand messen. Am besten nimmt man hierfür eine 2-m-Referenzmessung an einem Zollstock vor. Der Wert wird mit *Return* bestätigt.

Danach wird nach dem Divisor-Wert zur Messwertkorrektur gefragt. Dies ist nötig, da die Schallgeschwindigkeit temperaturabhängig ist. Im Temperaturbereich von -20°C bis +40°C verändert sich die Schallgeschwindigkeit von 312,85 m/s auf 349,32 m/s. Die Abweichung ist nahezu linear, so dass man pro 10°C Temperaturänderung eine Ungenauigkeit von etwa 2 % erwarten kann. **Kasten 1** enthält einige Formeln und eine Tabelle mit Werten zu erwartender Laufzeiten. Keinen Einfluss auf die Schallgeschwindigkeit hat übrigens der Luftdruck. Wer sich mit einer bis zu 5-%-igen Abweichung zufrieden gibt, braucht hier gar nichts tun. Wenn

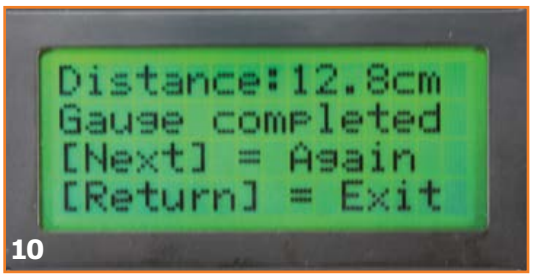


6

7

8

9



10

man allerdings Messungen bei Lufttemperaturen unter 15 °C oder über 25 °C durchführen und eine höhere Genauigkeit erzielen möchte, muss ein Divisor eingeführt werden. Je höher die Temperatur, desto größer muss der Divisor sein. Die Werte für Gehäusebreite und Divisor bleiben auch nach einem Batteriewechsel erhalten und können jederzeit neu eingestellt werden.

Nun sollte das Menü *Select mode:* zu sehen sein. Durch Drücken der *Return*–Taste gelangt man in das Tankauswahl-Menü. Es besteht die Möglichkeit, Kalibrierungen für bis zu zehn unterschiedliche Tanks oder sonstige Behälter vorzunehmen. Einzige Voraussetzung: Die Behälter müssen möglichst quaderförmig oder zylindrisch geformt sein, da ansonsten keine lineare Messung möglich ist.

Nach der Auswahl der Tanknummer landet man wieder im Hauptmenü. Nach dem Drücken der *Next*–Taste wird der Menüpunkt *Calibrate tank* angezeigt. Mit *Return* geht es zum Kalibriermenü. Nach Auswahl der Leerpegel-Kalibrierung wird der Benutzer gefragt, ob er für die Kalibrierung bereit ist. Hierzu muss der betreffende Behälter nicht zwangsläufig geleert werden, man kann einfach von der Höhe des Tankstutzens aus neben dem Tank auf den Boden messen. Ist der Leerpegel kalibriert, landet man im Menü Vollpegel-Kalibrierung (**Bild 6**). Der Defaultwert ist auf das Minimum von 3 cm eingestellt und kann durch Drücken von Next übernommen werden. Möchte man dennoch eine Vollpegel-Kalibrierung durchführen, bringt man das Gerät über dem gefüllten Behälter in Position und drückt auf *Return.*

Nach der Rückkehr zum Hauptmenü kann eine erste Pegelmessung durchgeführt werden. Hierzu den Menüpunkt *Gauge level* auswählen und mit *Return* bestätigen. Die Messung startet nun und das Ergebnis wird angezeigt. *Next* führt eine neue Messung durch, *Return* leitet zurück ins Hauptmenü.

Zum Ausprobieren der Pegelüberwachung wählt man das Menü *Level watchdog* aus. Im Folgemenü (**Bild 7**) kann der Alarm-Typ ausgewählt werden: *Full* bedeutet Alarmauslösung bei Überschreitung der erlaubten Füllmenge, *Empty* bei Unterschreitung. Nach Auswahl des Alarm-Typs kann im Menü *Level-Alarm* (**Bild 8**) der Schwellwertpegel geändert werden. Als Defaultwerte sind für den Maximalpegel 95 % und für den Minimalpegel 5 % voreingestellt, sie können mit den Tasten *Plus* und *Minus* geändert werden. Alle vorgenommenen Kalibrierwerte werden im EEPROM gespeichert und bleiben somit auch nach einem Batteriewechsel erhalten.

Dann startet die erste Messung und der aktuelle Pegel wird angezeigt (**Bild 9**). Nun werden im 5-Sekunden-Takt Messungen durchgeführt. Sollte der Minimalwert unterschritten oder der Maximalwert überschritten sein, zeigt das Display > *Level alarm* < an, die rote LED leuchtet und das Relais zieht an, um zum Beispiel eine Pumpe einzuschalten. Erreicht der Pegelstand nach 5 s wieder den erlaubten Bereich, schaltet sich die Alarmfunktion ab. Durch den 5-s-Messzyklus erspart man sich eine Hysterese-Funktion, um Relaisflattern auszuschließen.

Um aus diesem Modus wieder zurück ins Hauptmenü zu gelangen, muss man die *Return*-Taste drücken (bis zu 5 s lang). Nun kann das Menü *Gauge distance* ausgewählt werden. Mit jedem Druck auf die *Next*-Taste wird eine Entfernungsmessung gestartet (**Bild 10**). Auch aus diesem Modus kann man mit *Return* wieder zum Hauptmenü zurückkehren.

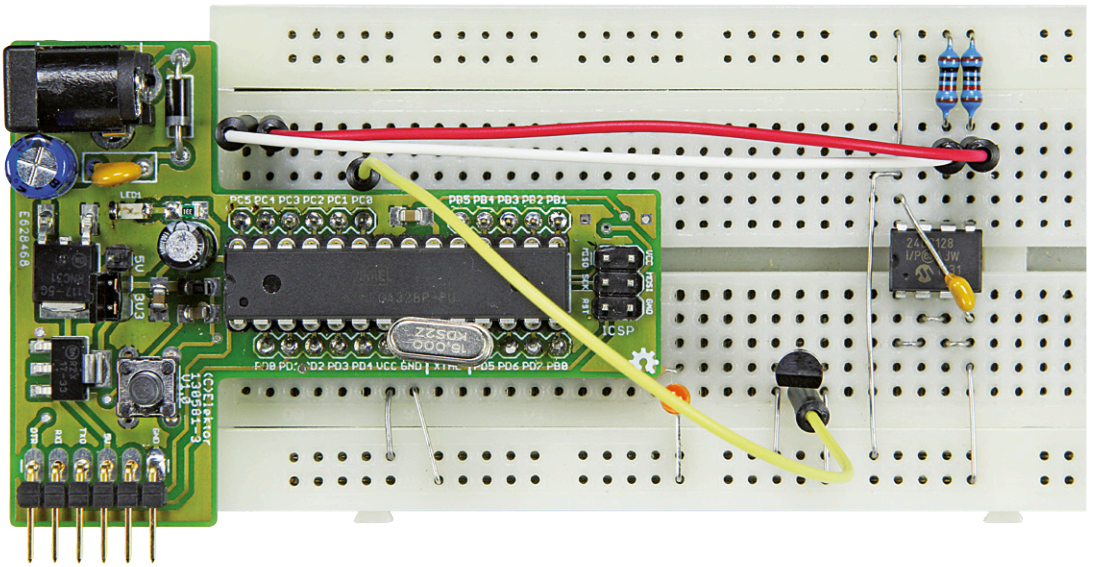
(140209)

Weblinks

[1] www.elektor-magazine.de/130546
[2] www.elektor-magazine.de/140209
[3] HC-SR04: www.cytron.com.my; SRF02: http://rn-wissen.de/wiki/index.php/Sensorarten#SRF02_Ultraschallsensor

T-Board 28: Übung in lowest Power

Temperatur-Logger schlägt Arduino-Lösung



Von **Andrew Retallack**
(Südafrika)

Da Mikrocontroller-Systeme für das Internet of Things immer wichtiger werden, ist der Energieverbrauch bei jedem Projekt ein kritischer Punkt. Jedes Mikroampere zählt, wenn entfernte Sensoren über Monate oder Jahre mit einer Batterie Daten liefern sollen. Elektors neues T-Board 28 ist die perfekte Basis für den Prototypen Ihres nächsten Projekts. Demonstriert wird sein Einsatz als stromsparender Temperatur-Logger.

Das T-Board 28 [1] unterstützt Entwickler bei der Optimierung stromsparender Projekte und Prüfung des Resultats. Das ist einer der großen Vorteile eines T-Boards gegenüber anderen Plattformen wie Arduino. Da viele Erstanwender wohl Erfahrung mit Arduino haben, wird Sie mein Weg interessieren, wie schrittweise der Strombedarf des T-Board 28 als Temperatur-Logger reduziert werden konnte. Auf dem T-Board 28 steckt ein ATmega328. Dabei handelt es sich um das größte aus der T-Board-Serie. Die Boards samt passendem T-Shirt gibt es im Elektor-Store.

Warum ein Temperatur-Logger?

Um die diversen Stromspartechniken auszuprobieren entschied ich mich für ein möglichst einfaches Projekt, das zwar von praktischem Nutzen sein, aber mich nicht vom Stromsparen ablenken sollte. Ein Temperatur-Logger passt da perfekt. Das Projekt eignet sich für die Messung von Temperaturen in regelmäßigen Intervallen mit einem preiswerten Sensor und der Sicherung der Daten in einem EEPROM. Wenn das Board mit einem PC (im Freien eher mit einem Notebook) Verbindung hat, sollen die angesammelten Daten über

eine serielle Schnittstelle ausgegeben werden. Ursprünglich waren SD-Karten vorgesehen. Doch da die einen bis zu 3,5-fachen Stromverbrauch aufweisen und keine 5 V vertragen, entschied ich mich für ein EEPROM. Das Projekt kann später erweitert werden, damit es die Daten per Funk übertragen kann. Die Optimierung des Stromverbrauchs von Funk-Modulen ist aber eine andere Baustelle.

Ein kurzer Blick auf die Schaltung

Wie in **Bild 1** zu sehen, ist die Schaltung sehr einfach gehalten - ein weiterer Vorteil für das T-Board. Das EEPROM 24LC128P von Microchip ist über I²C an das T-Board 28 angebunden. Hinzu kommen noch die Pull-up-Widerstände der I²C-Leitungen. Der Temperatursensor LM60 eignet sich gut, da er nur wenig Strom benötigt und auch negative Temperaturen messen kann (für die Leser, die nicht das Glück haben, im sonnigen Südafrika zu leben ;-)). Außerdem funktioniert er schon mit Spannungen ab 2,7 V. Schließlich gibt es noch eine LED zur Signalisierung der Logger-Aktionen. Während der Strommessungen ist das hilfreich. Sie kann im endgültigen Entwurf dann weg gelassen werden.

Der 24LC128 hat wie alle I²C-Chips eine Device-Adresse. Diese kann man durch den Logikpegel an den Pins 1, 2 und 3 (A0, A1 und A2) beeinflussen. Seine 7-bit-Adresse beginnt fix mit „1010“, gefolgt vom Pegel der Pins A2, A1 und A0. In dieser Schaltung ist A0 „high“, weshalb die Adresse „1010001“ resultiert. Wenn man den Chip auf eine Experimentierplatine packt, sollte man sehr sorgfältig mit allen Pins umgehen. Ich bemerkte erst spät, dass bei mir der Schreibschutz-Pin 7 mit V_{CC} statt mit V_{SS} (GND) verbunden war. Da klappte das Schreiben nicht wirklich. Man verliest sich eben schnell in einem Datenblatt.

Eine serielle Schnittstelle für das Debuggen und zum Auslesen der gespeicherten Daten wird über den UART im ATmega328 realisiert. Mit Hilfe eines FTDI-Kabels oder -Moduls am T-Board-FTDI-Anschluss kann sie mit der USB-Schnittstelle eines PCs verbunden werden. Eine Weiterentwicklung wäre z.B., den Projekt-Code für ein T-Board 8 zu portieren. Hierbei müsste dann ein Software-UART realisiert werden.

Der Code

Der Sourcecode wurde mit Hilfe von Atmel Studio 6.2 erstellt, der neuesten Version der Atmel IDE. Es gibt einfache Module zur Behandlung des ADC, des TWI (Two-Wire Interface) und von UART-Funktionen. An dieser Stelle ist ein kurzer Ausflug in die Terminologie angesagt. Wegen Schutzrechten beim I²C-Protokoll nennt Atmel die I²C-kompatible Version der Schnittstelle TWI. Sie ist zu 100 % mit I²C kompatibel, aber schlicht anders benannt. Deshalb muss man daran denken, im Datenblatt nach TWI statt nach I²C zu suchen.

Der Code steht unter [7] zum Download zur Verfügung und sollte nachvollziehbar sein. Etwas Sorgfalt ist bei den Compiler-Symbolen angesagt. Zugriff erhält man über die Option *Symbols* auf dem Registerblatt *Toolchain* der Projekt-Properties. Zur Berechnung von Datenraten für die Kommunikation sowie für die Genauigkeit der Intervall-Timer ist in F_CPU der CPU-Takt in Hz enthalten. Die Spannung in mV steht in SYSTEM_MILLIVOLTAGE. Damit kann man rohe ADC-Werte in richtige Spannungen umrechnen. Das Symbol DEBUG schickt bei einem Wert von „1“ Debug-Informationen über die serielle Schnittstelle.

Beim Start der MCU wird zunächst ein einfaches Menü seriell ausgegeben, mit dem man die Log-Daten durch Kommas getrennt ausgeben oder die geloggten Daten löschen kann. Das Programm wartet zunächst für 10 s auf die Aus-

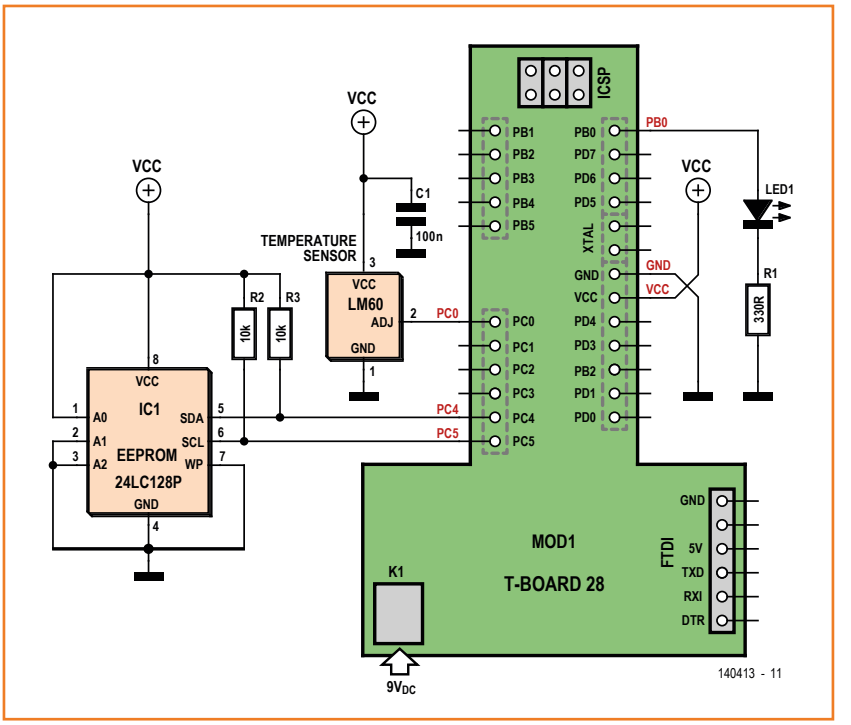


Figure 1. “Circuigram” of our exercise aimed at getting T-Board 28 to draw as little current as possible while acting as a full blown Temperature Logger.

Durchgebrannte Sicherungen

Fuses dienen bei AVR-Mikrocontrollern zur Konfiguration vieler Funktionen und Optionen. Man muss sie mit einem externen Programmer brennen. Vom Programm aus können sie mit nur wenigen Ausnahmen nicht geändert werden. Fuses machen manche Elektroniker richtig nervös, denn manche Fuses falsch gesetzt und schon hat man ein wertloses Stück Elektronikschrott vor sich. Um solche „verschossenen“ Chips zu retten benötigt man einen speziellen Programmer, der mit höheren Spannungen operiert.

Es gibt drei Fuse-Bytes beim ATmega328 des T-Board 28: Die „high“ bzw. H-Fusebits, die „low“ bzw. L-Fusebits und die „extended“ bzw. E-Fusebits. Bei den beschriebenen Experimenten werden hauptsächlich die L-Fusebits verändert, die für den Takt zuständig sind. AVR-Mikrocontroller kann man mit vielen internen und externen Taktquellen betreiben. Hierzu muss man die passenden Fusebits richtig setzen.

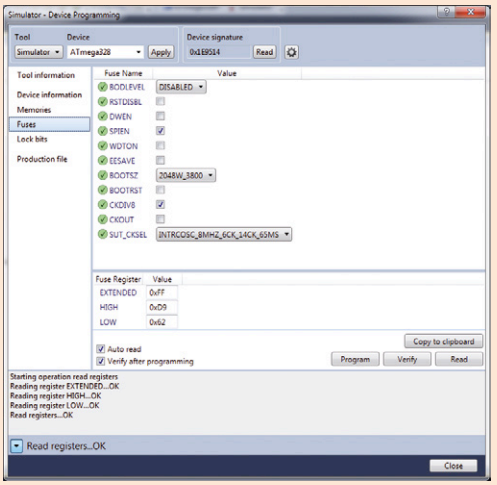
Zur Berechnung der korrekten Werte der Fusebits kann man sich durch die kryptischen Passagen des Datenblatts durchbeißen - oder man benutzt einen simplen Fuse-Calculator. Davon gibt es einige im Internet, unter denen ich den Rechner von Engbedded [4] bevorzuge. Es gibt auch passende Apps für jedes Smartphone.

Das Setzen von Fuses mit Atmel Studio ist ziemlich unkompliziert. Man gibt einfach die entsprechenden Werte in einem Dialog ein. Wenn Ihr Programmer nicht direkt von Atmel Studio unterstützt wird, dann benötigt man AVRDude [5]:

1. Man öffnet eine Befehlszeile und navigiert zum Ordner, in dem AVRDude.exe steckt.
2. Die aktuellen Fusebits liest man wie folgt ein:
   `avrdude -c <programmer> -p <MCU> -U lfuse:r:-:h`
3. Mit diesen Befehlen schreibt man ein Fusebyte:
   `avrdude -c <programmer> -p <MCU> -v -U lfuse:w:0xFF:m`

Dabei gilt:

`<programmer>` ist der Code von AVRDude für einen spezifischen Programmer (z.B. *usbtiny*). In der Datei *avrdude.conf* ist eine Liste mit Bezeichnungen enthalten.



Fuses in Atmel Studio

`<MCU>` ist der Code von AVRDude für eine spezifische MCU (z.B. *m328* für ATmega328).

`lfuse` ist das jeweilige Fusebyte, also *lfuse*, *hfuse* oder *efuse*.

`0xFF` steht hier für den Wert des jeweiligen Fusebytes. Zum Schreiben kann er nach Bedarf geändert werden.

Bei komplexeren Experimenten mit Fuses sollte man Google nutzen, denn hierzu gibt es im Netz viele gute Anregungen.

Timing the Timer

Timer2 ist ein 8-bit-Timer. Er kann also lediglich bis zum Wert 255 zählen. Bei jedem Taktzyklus wird sein Wert inkrementiert bis 255 erreicht ist. Man kann ihn so konfigurieren, dass dann ein Interrupt ausgelöst wird. Bei einem hochfrequenten Takt sind die 255 schnell erreicht. Bei einem 1-MHz-Takt sind die 255 nach nur 256 µs erreicht! Es gibt aber gute Methoden, auf längere Zeiten zu kommen. Als ersten Schritt kann man einen Vorteiler verwenden. Der größte verfügbare Vorteiler hat einen Wert von 1.024. Mit ihm wird der Zählerstand von 255 dann erst nach 262,144 ms erreicht. Das ist aber für ein Thermometer immer noch zu schnell. Also bleibt nur eine Taktreduktion.

Wenn man eine serielle Datenübertragung benötigt, kann man den Systemtakt aber nicht nach Belieben reduzieren. Doch für Timer2 kann man auch eine besondere externe Taktquelle verwenden. Allerdings ist dieser Takt dann nicht mit dem Systemtakt des Chips synchronisiert.

Auf jeden Fall kann man hierzu richtig langsame Quarzfrequenzen nutzen. Ein klassischer Uhrenquarz oszilliert mit 32,768 kHz. Mit so einem Quarz und einem Vorteiler von 1.048 wird der Zählerstand 255 erst in 8 s erreicht! Das passt sehr viel besser! Und niedrigere Taktfrequenzen bedeuten zudem weniger Strom.

wahl durch den Anwender, bevor es in die normale Logging-Schleife springt.
Es gibt zwei Code-Ausführungen: Version 1 (**Bild 2**) zeigt den noch rohen, nicht optimierten Code. Version 2 (**Bild 3**) enthält stromsparende Optimierungen. Die finale Version wurde merklich gekürzt, um den Code lesbar und nachvollziehbar zu gestalten. Es sind natürlich noch weitere kleine Stromspartechniken denkbar. Man betrachte dies als Herausforderung. Wenn Sie Ihre Überlegungen austauschen wollen, finden Sie unter www.elektor-labs.com und http://forum.elektor.com

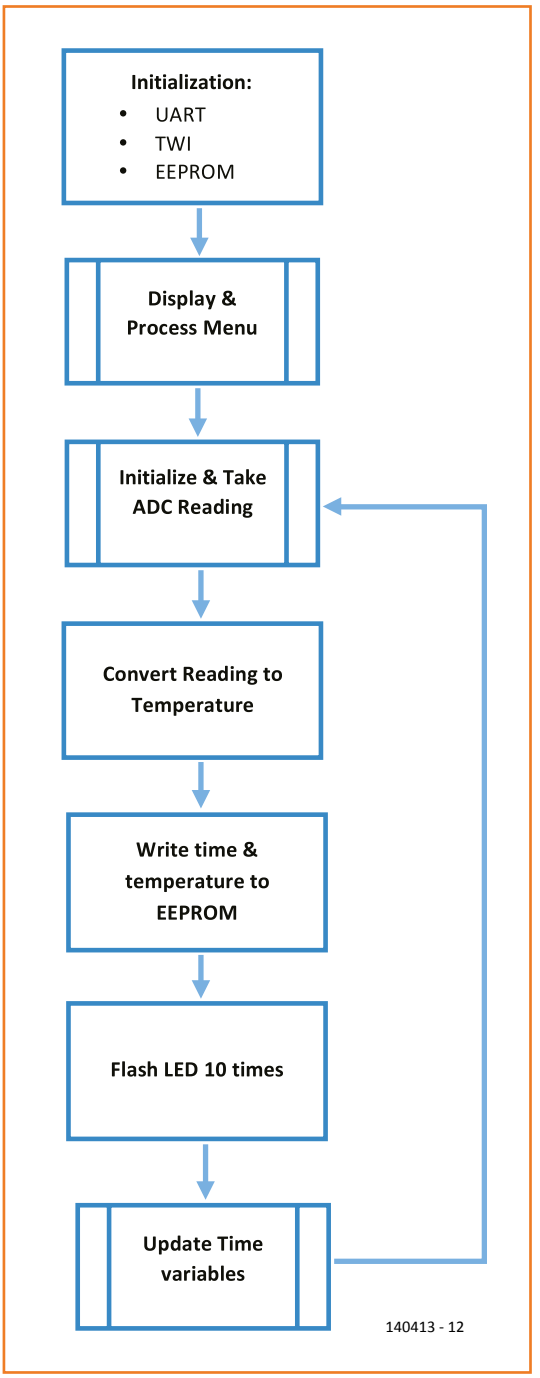


Bild 2. Flussdiagramm des rohen Codes von „Version 1“. Es stimmt zwar alles, doch der Code ist noch nicht optimiert.

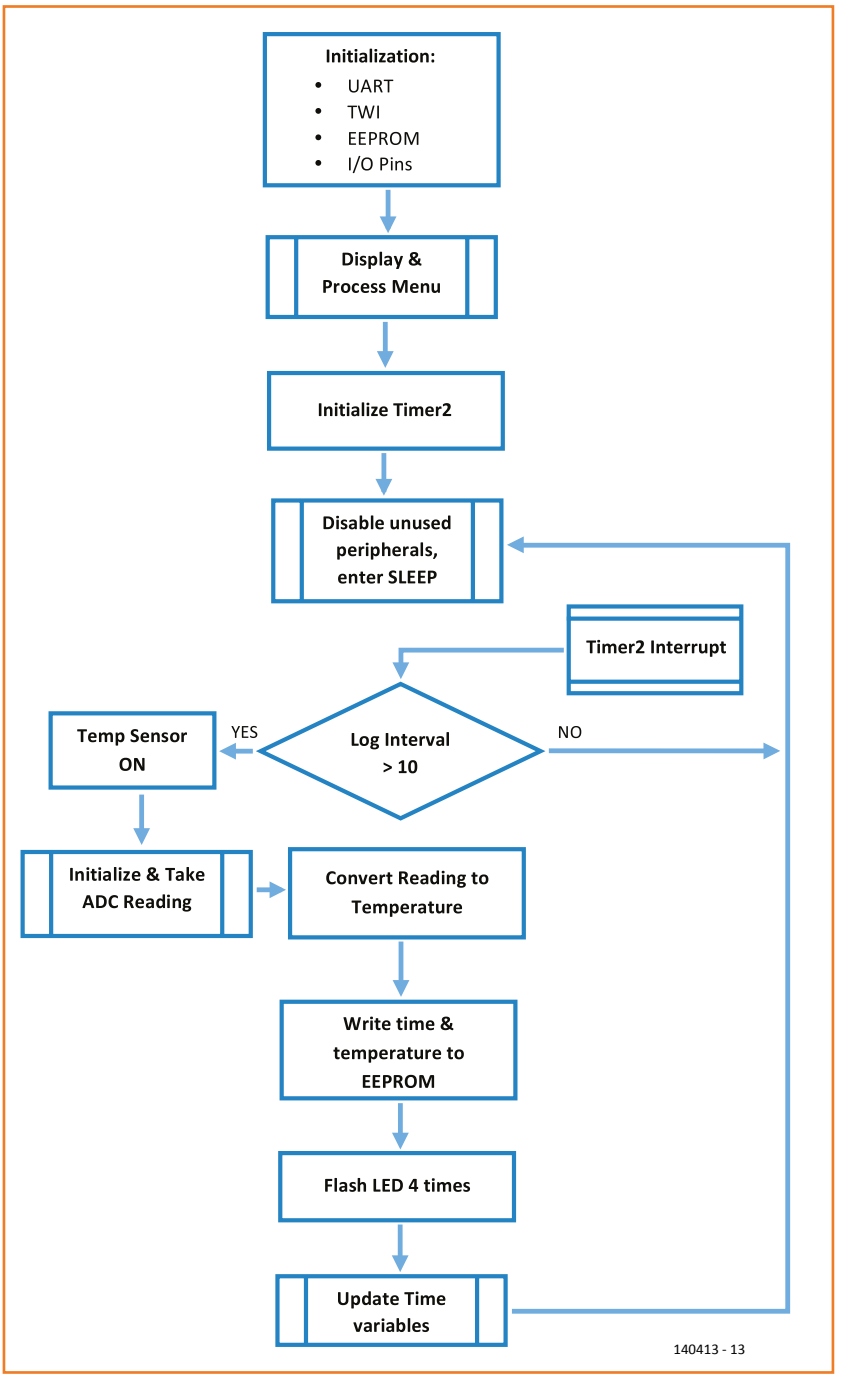


Bild 3. Flussdiagramm von „Version 2“. Die Firmware für das T-Board 28 wurde auf niedrigen Stromverbrauch getrimmt.

Optimierungsexperimente

1. Die „Baseline“

Wenn das Projekt wie in Bild 1 aufgebaut ist, wird das T-Board zuerst ähnlich wie ein Arduino Uno konfiguriert. Die Versorgungsspannung wird mit dem Jumper zur Spannungswahl auf 5 V festgelegt, und ein externer 16-MHz-Quarz wird angeschlossen. Da ja der Strom erfasst werden soll, wird der Jumper wieder entfernt und stattdessen das DMM angeschlossen. Die rote Leitung kommt an den 5-V-Pin und die schwarze an den mittleren Pin. Zunächst wird das DMM in den Ampere-Bereich geschaltet. Jetzt werden die Fuses anhand der Einstellungen nach **Tabelle 1** gebrannt. Zum Schluss muss der Code von „Version-1“ (siehe Download unter [7]) in Atmel Studio geladen, kompiliert und dann auf den Mikrocontroller übertragen werden. Diese Version nutzt eine einfache Verzögerungsschleife (in der Funktion `_delay_ms()` implementiert), um für Intervalle zwischen den Datenerfassungen zu sorgen. Das Flussdiagramm in **Bild 2** gibt einen Überblick über den Code.

Mit diesen ersten Einstellungen sollte sich eine Stromaufnahme von etwa 12,4 mA ergeben, während das Programm wartet. Um diesen „idle“-Status anzuzeigen, blinkt die LED zehnmal. Ein gültiger Messwert ergibt sich also, wenn die Anzeige des DMMs nach diesem Blinken stabil geworden ist.

Ein einzelnes kurzes Aufleuchten signalisiert, dass jetzt eine Temperaturmessung vorgenommen und das Resultat ins EEPROM geschrieben wird. Die hierfür benötigte Zeit ist viel zu kurz, um hierfür einen vernünftigen Stromwert ablesen zu können.

2. Reduktion der Spannung

Die einfachste Maßnahme ergibt die deutlichste Wirkung: Das T-Board wird einfach mit 3,3 V statt 5 V versorgt. Für unsere Messung wird wieder das DMM verwendet. Die schwarze Leitung bleibt, wo sie ist, und die rote Messleitung kommt an den 3,3-V-Pin. Es sind auch Änderungen im Code notwendig, da die Temperaturberechnungen die geänderte Referenzspannung von nur noch 3,3 V berücksichtigen müssen. Dazu muss aber lediglich das Symbol `SYSTEM_MILLIVOLTAGE` geändert werden. Dies erfordert im Projekt-Browser einen Rechtsklick auf das Projekt und die Auswahl der *Properties*. Dann klickt man im Tab *Toolchain* im Abschnitt *AVR/GNU C Compiler* auf *Symbols*. Jetzt wird `SYSTEM_MILLIVOLTAGE` von 5000 UL auf 3300 UL geändert. „UL“ bedeutet, dass der Wert als „unsigned long“ betrachtet wird. Nach der Neukompilierung flasht man den Code auf die MCU. Bei einer Versorgung mit 3,3 V fließen jetzt nur noch rund 5,76 mA – eine Stromreduktion von immerhin 53 %. Die Energieaufnahme des Loggers sinkt sogar um 69 %!

3, 4, 5. Langsamere MCU

Mit den nächsten drei Experimenten kann man den Einfluss des Controller-Takts überprüfen. Hierzu wird eine Kombination von langsameren Quarzen und interner Teilung des Takts angewendet. Mit jeder Taktänderung muss das Symbol `F_CPU` laut **Tabelle 2** angepasst werden. Die Leerzeichen in der Spalte `F_CPU` erleichtern das Lesen. Bei der Eingabe in Atmel Studio müssen sie natürlich entfallen. Für jede Spalte in Tabelle 2 müssen die schon erwähnten Schritte erneut durchgeführt werden: Also Programmer anschließen, Fuses updaten, `F_CPU` im Code ändern, neu kompilieren und das Resultat in die MCU flashen, Programmer entfernen und dann gegebenenfalls den Quarz wechseln.

Man kann einen klaren Zusammenhang zwischen Takt und Stromaufnahme erkennen. Es wäre nun einfach, z.B. einen Takt von 500 kHz zu wählen. Aber da taucht ein neues Problem auf: Bei 500 kHz weichen die Baudraten des USART eines ATmega stärker von den etablierten Werten ab. Im Datenblatt sowie unter [6] findet man nützliche Hinweise zur Berechnung der Genauigkeit der Baudrate in Abhängigkeit vom Takt. Hier gilt es also aufzupassen.

6. Der interne Oszillator

Ein Experiment nutzt den internen 8-MHz-Oszillator zusammen mit einem Takt-Teiler von 8, was einen effektiven Takt von 1 MHz ergibt. Dieser Takt reicht für stabile serielle Kommunikation mit niedrigen Datenraten bis 4.800 Bd aus. Das genügt auch für das Menü am Anfang beim Neustart. Wenn die L-Fusebits auf 0x62 gesetzt sind, muss der Wert für das Symbol `F_CPU` auf 1000000UL geändert und das neu kompilierte Programm auf die MCU übertragen werden. Dann kann man den externen Quarz entfernen. Die resultierende Stromaufnahme ist mit 748 µA zwar etwas höher als beim 500-kHz-Takt, es ist aber immer noch eine große Verbesserung gegenüber dem Ausgangswert von 12,36 mA zu erkennen!

7. BOD deaktivieren

Ein ATmega328 verfügt über einen BOD (**B**rown-**O**ut **D**etector), der die MCU resettet, wenn die Versorgungsspannung unter einen gewissen Wert fällt. Leider braucht dieses Feature extra Strom. Mit Hilfe der E-Fusebits kann man diese Funktion aber auch abschalten und so weitere µA einsparen. Ein Wert von 0x07 für die E-Fusebits deaktiviert die BOD. Schon fließen 6 µA weniger. Nicht sehr viel, aber jedes Elektron zählt!

8. Sleep-Mode

Nun wird der Code optimiert. In Version 1 (Bild 2) wurde eine Verzögerungsschleife für die Zeitintervalle der Temperaturabfragen genutzt. Man könnte nun denken, während einer solchen Wartephase tut die MCU schlicht nichts – aber das ist falsch. Stattdessen muss da die Zeit gezählt werden, bis es etwas zu tun gibt, und dieses Zählen kostet Strom. Die meisten MCUs verfügen aber über Schlafmodi. Dabei werden die eigentliche CPU samt einiger Peripherie abgeschaltet um Strom zu sparen. Bei Bedarf wird das Ganze per Interrupt wieder aufgeweckt. Solche Interrupts sind etwa Pegeländerungen an speziellen Pins, einlaufende Kommunikation oder eine voreingestellte Zeit. Ein AVR ist da keine Ausnahme: Es gibt sechs verschiedene Sleep-Modes mit unterschiedlich guten Stromspar-Eigenschaften. Sie werden hier nicht bis ins letzte Detail beschrieben. Im Modus „Power-save“ weckt Timer2 die CPU nach einer gewissen Zeit wieder auf. Nur Timer2 kann das.

Um weiter Strom zu sparen und präzise Zeiten zu ermöglichen, wird Timer2 von einem externen Uhrenquarz getaktet. Genaueres hierzu im **Kasten** „Timing the Timer“.
Für dieses Experiment müssen keine Fusebits geändert werden. Man muss nur einen Uhrenquarz mit 32,768 kHz richtig mit dem T-Board 28 verbinden. Man muss dabei aufpassen, denn die Anschlussdrähte sind sehr dünn, und sie sollten doch guten Kontakt zum Board haben.
Jetzt muss man den Code von Version 2 laden, kompilieren und in die MCU übertragen. Bei Version 2 gibt es weniger LED-Geblinke: nur noch vier Lichtblitze am Ende einer Messung. Nachdem sich die Anzeige meines DMMs nach der Blinkerei stabilisiert hatte, war ich hoch erfreut über das Resultat: Nur noch 59 µA! Damit reicht eine Batterie mit 800 mAh für nahezu 1,5 Jahre Dauerbetrieb. Doch das geht noch besser...

9. Ungenutzte Peripherie abschalten

Beim ATmega328 kann man mit dem speziellen Register PRR (**P**ower **R**eduction **R**egister) ungenutzte Peripherie deaktivieren. Die Funktion `reducePower()` deaktiviert Timer0, Timer1, SPI und den USART. Das TWI und Timer2 wurden aktiv gelassen, da man Ersteres zur Kommunikation mit dem EEPROM und Letzteres zum Aufwecken der MCU benötigt. Der ADC wurde ja schon optimiert, weshalb man hier nichts machen muss. Zusätzlich aktiviert diese Funktion Pull-up-Widerstände an ungenutzten Eingängen, um zu verhindern, dass diese Pins floaten. CMOS-Eingänge brauchen dann nämlich zusätzlichen Strom. Mit meinem DMM konnte ich hier keine weitere Stromeinsparung erkennen. Laut Datenblatt wäre diese Reduktion eh gering ausgefallen. Die wirkliche Einsparung ergibt sich hier nämlich eher im aktiven Modus der MCU.

10. Der letzte Versuch

Nach dem Vergleich mit den Angaben im Datenblatt war ich schon ganz glücklich mit dem, was ich erreicht hatte. Doch die MCU ist nicht das einzige Bauteil, das Strom benötigt. Da gibt es noch das EEPROM und den Temperatursensor LM60. Beim EEPROM beträgt die Stromaufnahme im Standby-Betrieb gerade mal 100 nA. Vernachlässigbar. Anders beim LM60, denn dieses benötigt laut Datenblatt einen Ruhestrom von immerhin 82 µA. Wenn man die Schaltung so ändert, dass der LM60 von Pin PD7 der MCU versorgt wird, kann man den Sensor abschalten. Die maximal 20 mA von PD7 werden ja bei weitem unterschritten. Jetzt musste noch der Code so geändert werden, dass der Sensor nur während der Messung versorgt wird. Und tata! Der nun gemessene Strom lag sensationell bei 1 µA. Weniger kann mein DMM gar nicht messen. An dieser Stelle beendete ich die Experimente und köpfte eine Flasche Sekt, denn das war ja nun ein Erfolg, den man feiern muss! Weitere Messungen zeigten dann, dass selbst der Spitzenstrom während der Temperaturmessung nur bei etwa 65 µA lag. Diesen Wert „rundete“ ich sicherheitshalber auf 1 mA auf und berechnete die mittlere Stromaufnahme. Nach meiner Berechnung reicht eine 800-mAh-Batterie jetzt gute sechs Jahre. Das dürfte genügen. Und ich wollte Sie nun nicht noch sechs Jahre bis zur empirischen Bestätigung dieser Laufzeit warten lassen ;-).

Tabelle 1. Ergebnisse der Experimente

Experiment	Beschreibung	F_CPU	SYSTEM_ MILLIVOLTAGE	LFUSE	EFUSE	Strom (mA)
1	16 MHz und 5 V	16 000 000 UL	5000 UL	0xFF	0x05	12,360
2	Reduktion auf 3,3 V	16 000 000 UL	3300 UL	0xFF	0x05	5,760
3	Externer Quarz 8 MHz	8 000 000 UL	3300 UL	0xFF	0x05	3,360
4	Externer Quarz 4 MHz	4 000 000 UL	3300 UL	0xFD	0x05	2,130
5	Teiler /8	500 000 UL	3300 UL	0x7D	0x05	0,560
6	Interner Oszillator 8 MHz + Teiler /8	1 000 000 UL	3300 UL	0x62	0x05	0,748
7	BOD inaktiv	1 000 000 UL	3300 UL	0x62	0x07	0,742
8	Sleep Mode mit Timer2	1 000 000 UL	3300 UL	0x62	0x07	0,059
9	Ungenutzte Peripherie abschalten	1 000 000 UL	3300 UL	0x62	0x07	0,059
10	Ungenutzten Temp.-Sensor abschalten	1 000 000 UL	3300 UL	0x62	0x07	0,001

Tabelle 2. Quarze und Teiler für Experiment 3, 4 und 5.

Experiment	Quarz	Teiler/8?	Effektiver Takt	F_CPU	LFUSE	Current
3	8 MHz	nein	8 MHz	8 000 000 UL	0xFF	3,36 mA
4	4 MHz	nein	4 MHz	4 000 000 UL	0xFD	2,13 mA
5	4 MHz	ja	500 kHz	500 000 UL	0x7D	0,56 mA

Weblinks

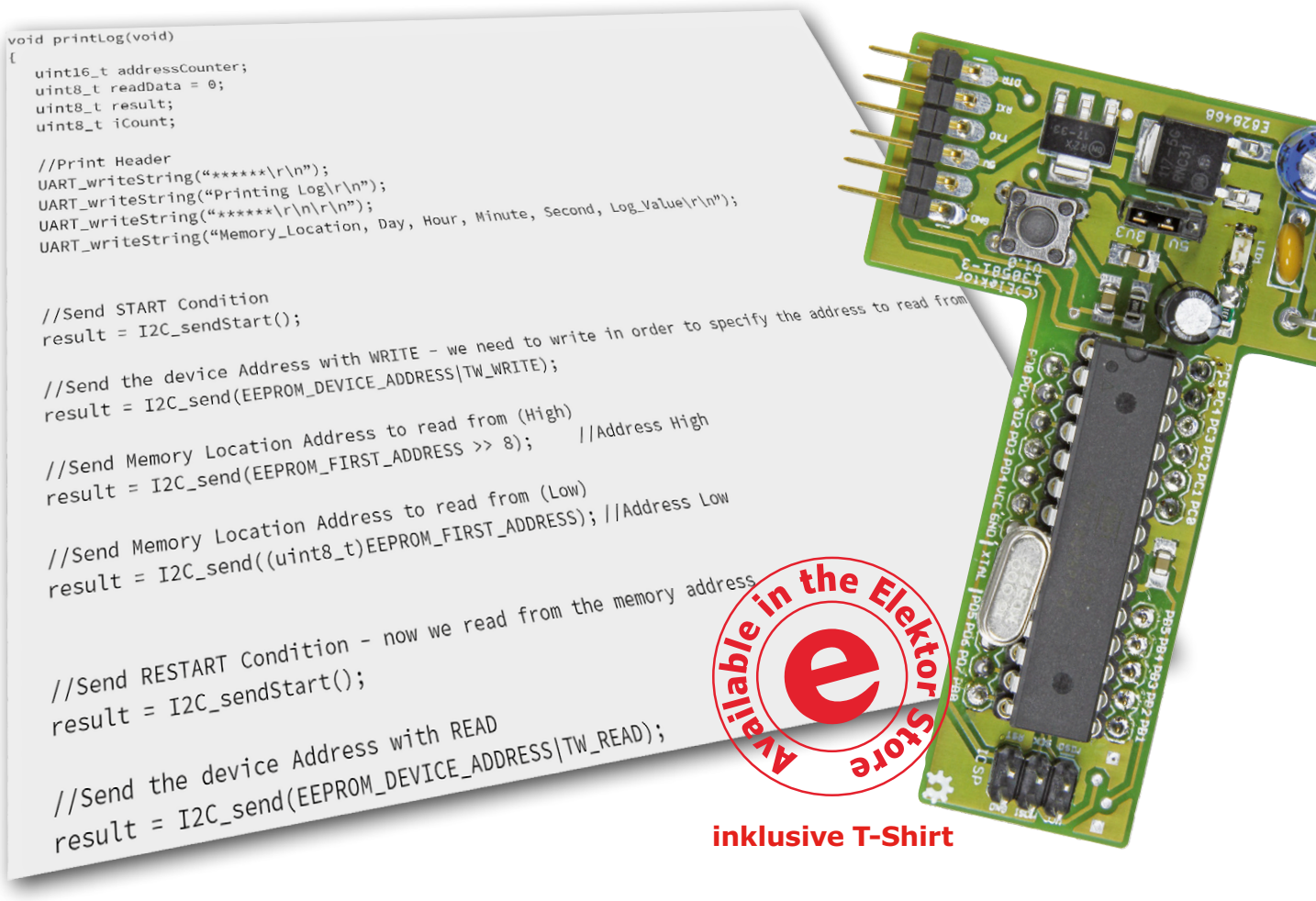
[1] T-Board 8/14/28, Elektor September 2014, www.elektor-magazine.com/130581
[2] Atmel Studio: www.atmel.com/atmelstudio
[3] USBTiny: www.crash-bang.com/using-usbtiny-with-atmelstudio/
[4] Fuse Calculator: www.engbedded.com/fusecalc/
[5] AVRDude: http://savannah.nongnu.org/projects/avrdude
[6] Baud Calculator: www.wormfood.net/avrbaudcalc.php
[7] Code Version 1 & 2: www.elektor-magazine.de/140413

(Topic: Mikrocontroller & Embedded) die richtige Plattform dafür.

T-Board hilft beim Optimieren

Beim Thema Stromsparen bei Mikrocontroller-Projekten fokussiert man prinzipiell vier Aspekte. Zuerst versucht man die Spannungen zu reduzieren und nutzt somit schlicht das Ohmsche Gesetz. Als zweite Strategie taugt die Reduktion der Taktfrequenz der MCU. Je langsamer, desto weniger Strom. Als drittes Verfahren nutzt man die Sleep-Modi der MCU. Der letzte Aspekt ist die Deaktivierung nicht genutzter MCU-Peripherie und sonstiger Funktionseinheiten.

Das T-Board 28 unterstützt von Hause aus die ersten beiden Punkte: Spannung und Taktfrequenz. Die Punkte drei und vier stecken im Code. Der Spannungswähler auf dem T-Board ist der Punkt, an dem man den benötigten Strom als



Maß für die Effektivität der Stromsparbemühungen erfassen kann.

Ergänzend zu den angeführten vier Punkten gibt es noch weitere Verfahren. Effektiverer Code wäre ein Beispiel, da so weniger Takte für eine Aufgabe benötigt werden, was den mittleren Strombedarf reduziert. Wie schon erwähnt liegt auf der Code-Effizienz aber nicht der Schwerpunkt dieses Artikels. Code-Optimierung in C/C++ ist Stoff für eine ganze Serie spezieller Artikel.

Messen der Stromaufnahme

Die meisten Nutzer von T-Boards dürften so wie ich kein voll ausgestattetes Profi-Labor zur Verfügung haben. Von daher ist meine Strategie etwas grob, aber trotzdem wirkungsvoll. Zur Erfassung der Stromaufnahme reicht ja auch ein preiswertes Digitalmultimeter (DMM). Wenn das DMM „auf Strom“ steht und man mit den Prüfspitzen die Spannungsauswahl auf dem T-Board vornimmt, misst man den realen Strom zu exakt diesem Zeitpunkt. Dieser ist nicht von der Stromaufnahme von Spannungsreglern etc. kompromittiert. Zwecks möglichst genauer Messungen habe ich zusätzlich die LED zur Spannungsanzeige entlötet, da sie mehr Strom als die MCU verbraucht. Durch den dann geringeren Stromverbrauch kann man in bestimmten Fällen das DMM in den µA-Bereich schalten und so sehr hoch aufgelöst die Auswirkungen einzelner Maßnahmen bewerten.

Natürlich hat diese Messmethode so ihre Grenzen. Abgesehen von der Auflösung wird der Strom nur zu gewissen Zeitpunkten erfasst. Idealerweise würde man den Strom über eine ganze Zeitspanne erfassen und dann die mittlere Stromaufnahme errechnen, was selbst wieder ein eigenes Elektor-Projekt wäre. Eine LED an PB0 ist sehr nützlich für die Anzeige von Programmaktivitäten. Damit weiß man genau, wann man messen sollte.

Experimente

Nun zu den konkreten Stromsparexperimenten. Hierbei wird detailliert beschrieben, welche Änderungen durchgeführt wurden und welche Auswirkungen dies hatte. Um die aufeinander aufbauenden Optimierungen durchzuführen, werden folgende Schritte vorgenommen:

- Der ISP-Programmer wird angeschlossen und die Fuses eingestellt.
- Der geänderte Code wird kompiliert und auf das T-Board übertragen.
- Der ISP-Programmer wird wieder entfernt.
- Eventuelle Änderungen an der Schaltung werden vorgenommen.
- Die 9-V-Batterie kommt an die 2,1-mm-Buchse und der Strom wird per DMM gemessen.

Während einer Messung sollte kein FTDI-Kabel oder entsprechender Adapter angeschlossen sein. Die Ergebnisse der Experimente sind samt Änderungen in **Tabelle 1** zusammengefasst.

Bevor es losgeht, noch ein Hinweis zum Programmer: Atmel Studio [2] unterstützt diverse ISP-Programmer direkt, die dann keine oder nur eine minimale Konfiguration erfordern. Daneben gibt es eine Vielzahl – meistens preiswerterer - anderer Programmer, die nicht direkt unterstützt werden, die man aber so konfigurieren kann, dass sie ebenfalls mit Atmel Studio kooperieren (z.B. USBTiny, USBasp etc.). Hier gibt es größere Unterschiede, wie die Fuses gesetzt werden müssen. Hierzu empfiehlt sich dann die Lektüre der Tipps im **Kasten** „Durchgebrannte Sicherungen“. Außerdem gibt es online noch viele Hilfestellungen für die Konfiguration von Atmel Studio, damit diese Programmer funktionieren [3].

Weitere Schritte

Ich fand diese Erfahrung sehr lehrreich. Sie hat mich voll von den vielen Möglichkeiten dieser T-Boards überzeugt. Insgesamt gelang eine Reduktion der Stromaufnahme von arduino-typischen 12,4 mA auf einen Durchschnitt von nur 13 µA (berechnet auf der Grundlage der aktiven gegenüber der inaktiven Zeit der MCU). Das entspricht einer Einsparung von 99,89 % des Stroms! Und nun fordere ich Sie dazu auf, dies noch einen Schritt weiter zu treiben und das Ganze auf einen noch kleineren Mikrocontroller wie den des T-Board 8 zu übertragen. Oder wie wäre es, die Daten per Funk zu übermitteln? Ein Hinweis noch: Das klappt natürlich nur mit dem passenden T-Board-T-Shirt am Leib.

(140413)

Programmierbare Edeltanne

Bringt (blaues) Licht ins Dunkel

Text: **Harry Baggen**
(Redaktion NL)

Entwurf: **Eurocircuits**

Wenn der Herbst über das Land zieht und die Tage kürzer werden, ist Weihnachten nicht mehr weit. Jetzt ist es traditionell an der Zeit, dass wir überlegen, was das Fest elektronisch verschönern könnte. Diesmal ist es ein Weihnachtsbaum, an dem 62 blaue LEDs programmiert funkeln.

Mehr oder weniger stilvolle Weihnachtsdekorationen im Baumarkt kaufen, das kann schließlich jeder. Der kreative Elektor-Leser ist anspruchsvoll, er setzt auf Originalität und Individualität, und natürlich darf ein gehöriger Schuss Hightech nicht fehlen. Daraus ist im Lauf der Jahrzehnte eine Elektor-Tradition gewachsen, wir setzen sie in diesem Jahr mit ungebrochenem Enthusiasmus fort.

Gemeinsam mit dem Platinenspezialisten Eurocircuits haben wir einen blau funkelnden Weihnachtsbaum kreiert, dem der Leser eine individuelle Note verleihen kann. Wer in diesen hektischen Tagen noch Zeit und Muße hat, kann natürlich selbst ans Werk gehen. Allen gestressten Zeitgenossen empfehlen wir, einen Blick in den Elektor-Shop zu werfen, der den Nadelbaum zu einem Weihnachtssonderpreis ins Haus liefert. Das ist sicher kein schlechter Rat, denn die winzigen SMDs lassen sich mit konventionellem Lötgerät nicht im Eiltempo auf dem Stamm des Baums montieren.

Die Vorderseite der Blautanne zieren 62 hellblau leuchtende SMD-LEDs, das kühle Blau lässt die frostige Kühle weihnachtlicher Winternächte erahnen. Auf der Rückseite ist die steuernde Elektronik inklusive eines leistungsstarken Mikrocontrollers untergebracht. Wir haben zwar schon früher blinkende Weihnachtsdekorationen in Elektor präsentiert, doch in diesem Baum steckt eine Lightshow, die ungleich facettenreich ist. Über die LEDs laufen nicht nur Lichteffekte in programmierbarer Vielfalt, darüber hinaus können Texte und Laufschriften persönliche Botschaften transportieren. Ein Fundus attraktiver Formen und Muster ist bereits implementiert. Neue Kreationen können Lightshow-Künstler am PC-Bildschirm entwerfen und über ein simples USB-Kabel in den Tannenbaum laden. Der Phantasie sind keine Grenzen gesetzt.

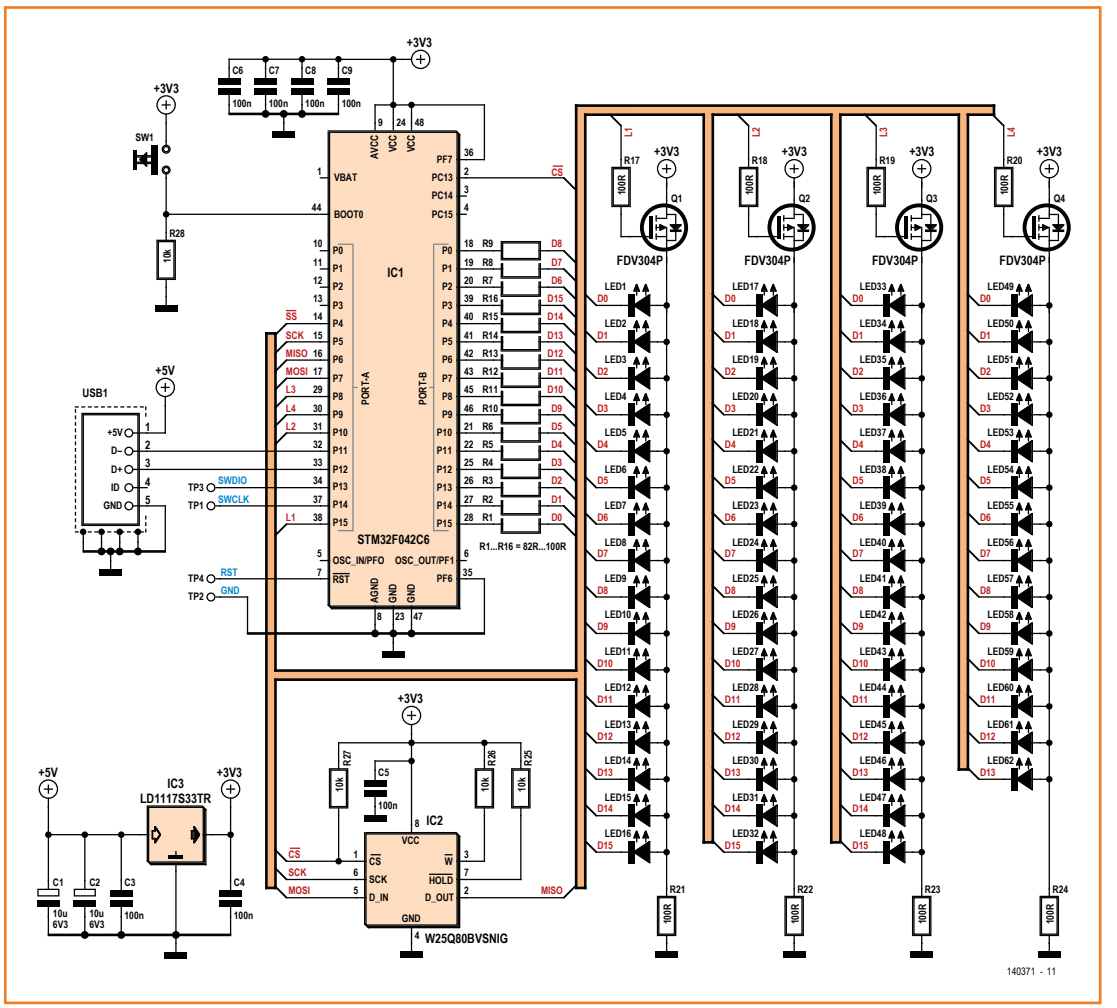


Bild 1.
Zu den Zutaten unserer Edeltanne gehören 62 blaue SMD-LEDs und ein 32-bit-Mikrocontroller von STM.

Stückliste

Widerstände:
R1..R16 = 82..100 Ω (SMD 0805)
R17..R24 = 100 Ω (SMD 0603)
R25..R28 = 10 k (SMD 0603)

Kondensatoren:
C1,C2 = 10 µ/6,3 V Tantal (SMD A)
C3..C9 = 100 n/10 V keramisch (SMD 0603)

Halbleiter:
LED1..LED62 = LED blau (SMD 1206)
Q1..Q4 = FDV304P (SOT23)
IC1 = STM32F042C6 (LQFP48)
IC2 = W25Q80BVSNIG (SO8)
IC3 = LD1117S33TR (SOT223)

Außerdem:
USB1 = USB-Micro-Buchse für Platinenmontage (47346-0001)
SW1 = Drucktaster für Platinenmontage (TACTB-64K-F)
Platinenlayout 140371-1

Rechenstark

Das Kernstück unseres Tannenschmucks ist ein Mikrocontroller ARM Cortex-M0 von STM, **Bild 1** zeigt den Plan der Hardware. Dieser preisgünstige 32-bit-Mikrocontroller bietet außer hoher Rechenleistung eine USB-2.0-Schnittstelle auf dem Chip, so dass er unkompliziert mit einem PC verbunden werden kann. Über den integrierten Bootlader lassen sich die kreierten Leuchtmuster und Animationen fast mühelos in den seriellen Flash-Speicher laden. Der Speicher ist 1 MB groß (8 · 1 Mbit), was verglichen mit gängigen USB-Sticks und Speicherkarten recht wenig erscheinen mag. Der Platz reicht jedoch vollkommen, um mehr als 90.000 Leuchtmuster dauerhaft aufzubewahren.

Die LEDs sind in einer Matrix 4 · 16 angeordnet, 3 · 16 + 1 · 14 Matrixpunkte sind mit LEDs beschaltet. Der Mikrocontroller steuert über die Portleitungen P0...P16 vier LED-Gruppen, die aus 16 (oder 14) LEDs bestehen. Die vier LED-Gruppen werden im Multiplex-Betrieb fortlaufend durchgeschaltet. Wegen der Trägheit des Auges ist das Umschalten nicht sichtbar, es scheint, als ob die LEDs kontinuierlich leuchten. Das schnelle Schalten übernehmen vier SMD-p-Kanal-MOSFETs FDV304P. Die Widerstände R1...R16 begrenzen die LED-Ströme, sie sind unmit-

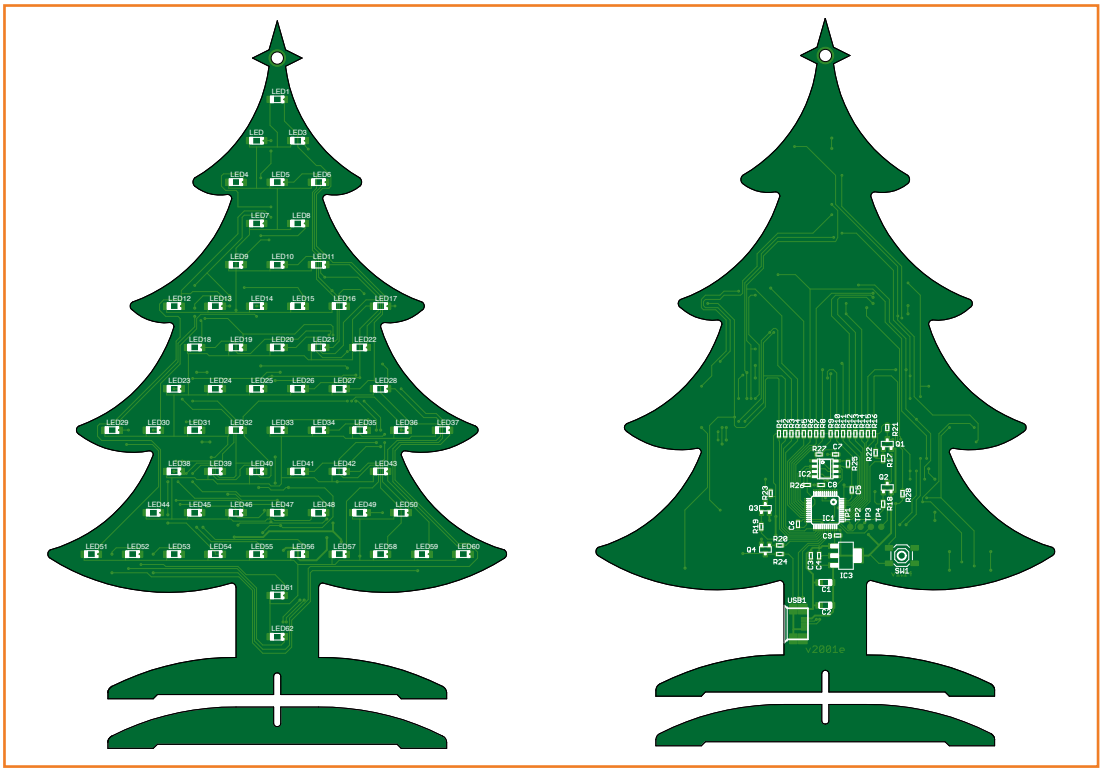


Bild 2.
Auf der Frontseite der doppelseitigen Platine ist eine so genannte PCB-Pixture-Lötmaske mit Eiskristallen aufgetragen.
(60% der tatsächlichen Größe)

telbar mit den Port-Ausgängen des Mikrocontrollers verbunden. Durch den Verzicht auf zwischengeschaltete Treiber werden Bauelemente eingespart, andererseits ist der Mikrocontroller nur mit einem Gesamtstrom von 80 mA belastbar. Der Strom jeder einzelnen LED darf folglich 5 mA nicht übersteigen. LEDs mit hohem Wirkungsgrad leuchten auch bei diesem Strom noch genügend hell, um deutlich wahrnehmbar zu sein.

Die Weihnachtsbaum-Steuerung wird über die Micro-USB-Buchse mit Strom versorgt. Spannungsregler IC3 stellt die stabilisierte Betriebsspannung 3,3 V für den Mikrocontroller und die LED-Matrix bereit.

Dass die Platine unserer Weihnachts-Edeltanne stilecht aussieht, belegt das Layout in **Bild 2**. Die LEDs befinden sich auf der Vorderseite, die Steuerung mit dem Mikrocontroller ist auf der Rückseite versteckt. Diese Platine ist eine Sonderausführung, sie ist mit einer PCB-Pixture-Lötmaske mit Eiskristallen versehen worden (siehe [1]). Die Leser, die sich das Anfertigen und Bestücken der Platine in eigener Regie zutrauen, müssen wir vorsorglich warnen. Einige SMDs lassen sich nur mit viel Ausdauer und einer Portion Glück erfolgreich montieren.

Der Quell- und Hexcode der Firmware ist auf der Projektseite [2] zum freien Download verfügbar. Wir haben das Programm in C geschrieben und mit einem Compiler von Keil in den Hexcode überführt. Der größte Teil gehört zur Kommunikation über die USB-Schnittstelle, er basiert auf der ST Firmware Library.

Wird der Drucktaster auf der Rückseite beim Verbinden mit dem PC gedrückt gehalten, wechselt der Mikrocontroller in den Bootloader-Modus. In diesem Modus kann mit dem Tool DfuSe von ST [3] die Firmware geladen oder aktualisiert werden.

Kreativ

Im Weihnachtsbaum, wie ihn der Elektor-Shop liefert, sind bereits attraktive Animationen eingebaut. Wenn Sie nach mehr streben und als Lightshow-Künstler gestalterisch tätig sein möchten, ebnen wir Ihnen den Weg: Wir haben eine interaktive englischsprachige Webseite eingerichtet [4], auf der unser Baum mit den LEDs erscheint. Ihre Animationen erzeugen Sie, indem Sie die virtuellen LEDs mit der Maus einschalten oder ausschalten. Wenn das LED-Muster vollendet ist, können Sie die Dauer des Aufleuchtens und die Helligkeit wählen. Die entstandenen LED-Muster werden unten im Bild aufgereiht. Nach diesem simplen Verfahren generieren Sie alle Bausteine Ihrer Lightshow. Klicken Sie auf das Textsymbol, wenn Sie eine Laufschrift generieren wollen. Ihre Animation können Sie für eine spätere Verwendung speichern. Wenn Sie auf den Preview-Button klicken, können Sie die Animation vorab auf Ihrem Bildschirm betrachten.

Einzelne Animationen lassen sich zu ausgedehnten Lightshows verknüpfen. Sobald das Werk vollendet ist, können Sie es als Ganzes in Ihren Weihnachtsbaum herunterladen. Den Baum verbinden Sie über ein Micro-USB-Kabel mit Ihrem PC. Beim ersten Mal installiert Windows einen Treiber, Windows betrachtet den Baum als so genanntes „HID-Device“. Von jetzt an können Sie Ihre Kreationen unmittelbar von der Webseite in Ihren Baum übertragen.

Damit die Lightshow auch ohne PC stattfinden kann, verbinden Sie ihn über die Micro-USB-Buchse mit einem 5-V-Steckernetzteil, das mit einem Micro-USB-Stecker ausgestattet ist.

Wir wünschen Ihnen kreative Festtage!

(140371)gd

Gestalten und gewinnen!

Eurocircuits hat einen Wettbewerb um die beliebteste Programmierung des Weihnachtsbaums ausgeschrieben. Setzen Sie Ihre Kreation auf die Webseite des Baums [4] und veranlassen Sie möglichst viele Besucher, das Votum „I like it“ anzuklicken. Sie können einen attraktiven Preis gewinnen! Der Wettbewerb läuft bis zum 07.01.2015, die Gewinner werden persönlich benachrichtigt.

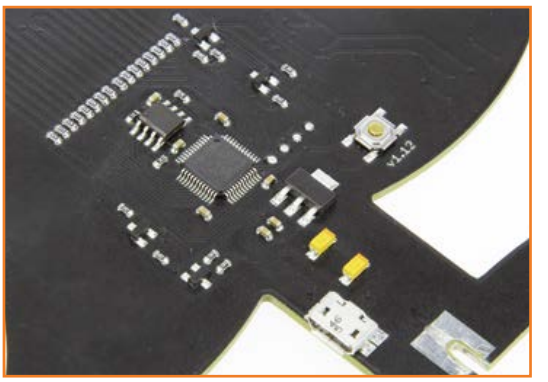


Bild 3.
Die Steuerung mit dem Mikrocontroller verbirgt sich auf der Rückseite.

Weblinks

[1] http://www.eurocircuits.com/blog/171-PCB-PIXture-launched
[2] www.elektor-magazine.de/140371
[3] www.st.com/web/en/catalog/tools/FM147/CL1794/SC961/SS1533/PF257916
[4] www.eurocircuits.com/x-mas

Breakout-Board mit ADS1115

16-bit-ADC mit 4 Kanälen und I²C-Anschluss

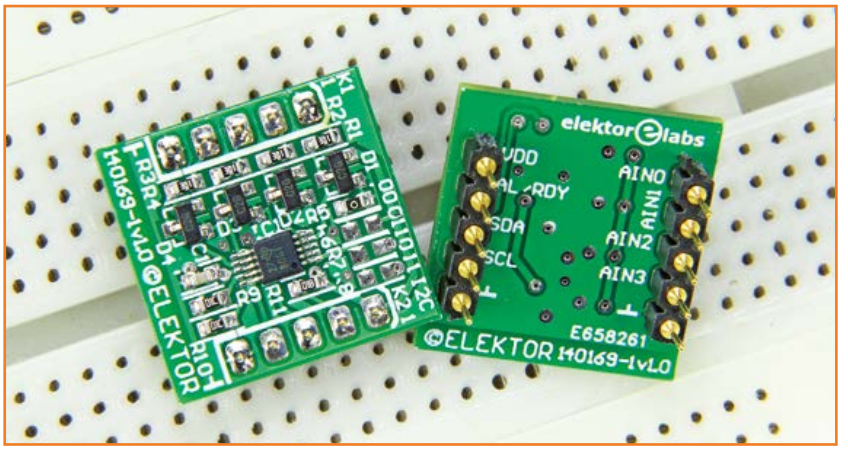
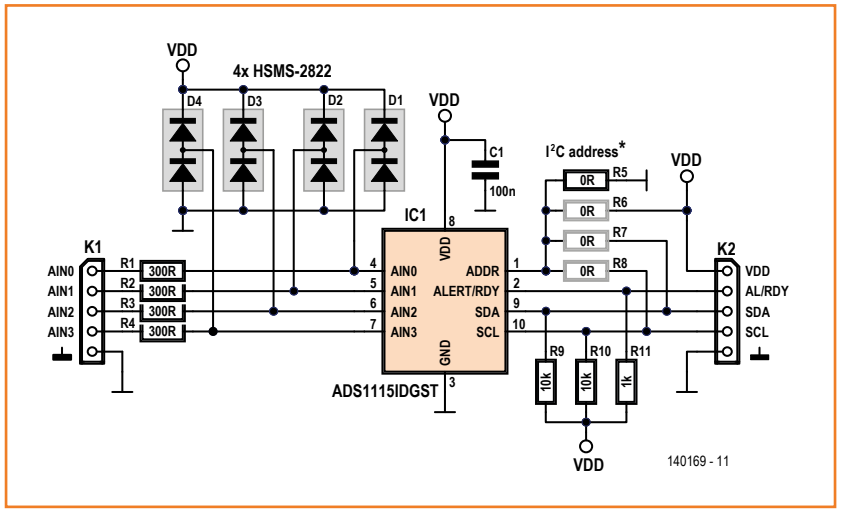


Bild 1.
Das ADS1115-Breakout-Board zeichnet sich durch eine geringe Anzahl von Bauteilen aus.

Von **Ton Giesberts** und **Jan Buiting** (Elektor)

Damit Mikrocontroller (und manchmal auch deren User) analoge Signale verstehen können, müssen diese zunächst digitalisiert werden. Diese Arbeit übernimmt dieses kleine, aber leistungsfähige ADC-Breakout-Board (BoB).

ADCs und ihre Kollegen namens DACs sind hilfsbereite Vermittler zwischen den einander so fremden Welten der analogen und digitalen Signale. Bereits in den Artikeln zum experimentellen ELF-Empfänger [1] und zum 16-bit-Datenlogger [2] wurde der Analog/Digital-Wandler ADS1115 von Texas Instruments [3] genutzt. Der Wandler stellt ein perfektes Frontend für relativ langsame (<1 kHz) aber präzise Datenerfassungssysteme (DAQs) dar. Wir haben deshalb den Chip extra für Sie „geBoBt".
Beim 16-bit-Datenlogger befand sich der Wandler zusammen mit Operationsverstärkern, einigen Anschlüssen und einem LED-Treiber auf der Platine 130485-1. Sie ist ausgelegt für einen einfachen Anschluss an eine Reihe von Elektor-Embedded-Boards einschließlich Linux-Board, Xmega-Webserver-Board und Extension-Shield. Der ELF-Receiver ist dagegen eine hauptsächlich analoge Anwendung zum stundenlangen Lauschen merkwürdiger Geräusche.



Handlöten schwer gemacht

Das Problem mit diesen neuen ICs ist, dass man sie nicht gerade leicht mit dem Handlötkolben auf einer Platine befestigen kann. Von der guten Resonanz auf die beiden obengenannten Artikel angeregt, haben wir uns entschlossen, den ADS1115DGST mit wenigen erforderlichen peripheren Bauteilen auf eine kleine Breakout-Platine zu setzen und diese im Elektor-Store anzubieten.

Auf das Minimum reduziert

Die Schaltung in **Bild 1** zeigt, dass herzlich wenig erforderlich ist, um einen 4-Kanal-/16-bit-ADC

mit einer maximalen Flexibilität zu realisieren - das große Plus eines BoBs. Hier haben wir gemäß den Empfehlungen der Datenblätter für einen Eingangsschutz mit den Widerständen R1... R4 und den Schutzdioden D1...D4 gesorgt. Die 300-Ω-Widerstände begrenzen den Leckstrom jeder Schottky-Diode auf einen vernachlässigbaren Wert von etwa ½ LSB.

Der ADS1115 ist ein I²C-Baustein mit vier Slave-Adressen, die mit den 0-Ω-Widerständen (aka Brücken) R5...R8 am ADDR-Pin eingestellt werden können. Es wird nur einer dieser Brücken bestückt, Standard ist R5. Informationen über die Adressauswahl finden Sie in [2] und [3].
Bild 2 zeigt den Bestückungsplan (tatsächliche Größe 19,1 mm x 19,7 mm) und die Stückliste des ADS1115-BoBs. Obwohl wir die Platine fertig bestückt im Elektor-Store anbieten, können Sie auch eine Leerplatine erwerben, um die Bestückung mit geeigneten Lötwerkzeugen @home zu erledigen.

Beachten Sie, dass die Stiftleisten K1 und K2 auf die Unterseite gehören, damit Sie das Board auf ein Steckbrett stecken können. Eine Stiftleiste führt die vier Eingänge, die andere die Spannungsversorgung (VDD und Masse), den I²C-Bus sowie eine Alarm-/Bereitschaftsleitung AL/RDY. Beim Einsatz als Breakout-Board sollten Sie Leisten mit beidseitigen Stiften verwenden, ansonsten reichen auch „gewöhnliche“ Stiftleisten aus.

Obwohl dieser Artikel sich nur mit der Hardware und dem Komfort einer langsamen, aber sehr präzisen ADC-Messung beschäftigt, wollen wir doch auf die Codebeispiele und Bibliotheken in [2] hinweisen. Die Software steckt sowohl im Download-Paket zu [2] als auch in dem Paket zu diesem Artikel [4].

(140169)

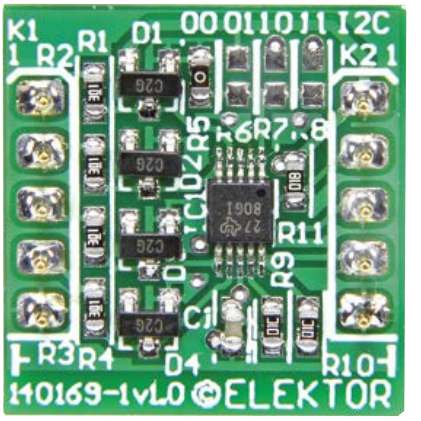
Weblinks

[1] ELF-Empfänger, Elektor September 2014, www.elektor-magazine.de/140035

[2] Daten loggen mit 16 bit, Elektor September 2014, www.elektor-magazine.de/130485

[3] ADS1115-Dokumente: www.ti.com/lit/ds/symlink/ads1115.pdf

[4] Projekt-Software: www.elektor-magazine.de/140169



Available in the Elektor Store

Stückliste

Widerstände
R1...R4 = 300 Ω, 1%, 0,1 W, SMD 0603
R5...R8* = 0 Ω 1%, 0,1 W, SMD 0603
R9,R10 = 10 k 1%, 0,1 W, SMD 0603
R11 = 1 k, 5%, 0,1 W, SMD 0603
*nur eine Brücke bestücken, default = R5

Kondensator
C1 = 100 n, 10%, 16 V, 0603 X7R

Halbleiter
D1...D4 = HSMS-2822-TR1G (Avago Industries)
IC1 = ADS1115IDGST (Texas Instruments)

Außerdem
K1,K2 = 1x5-polige Stiftleiste, Raster 0,1”, eventuell auch mit beidseitigen Stiften (Harwin D01-9923246, Farnell-Nr. 1022218)

Fertig bestückte Platine im Elektor-Shop: 140169-91
Leerplatine im Elektor-Shop: 140169-1, v1.0

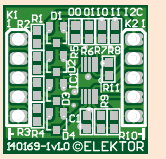
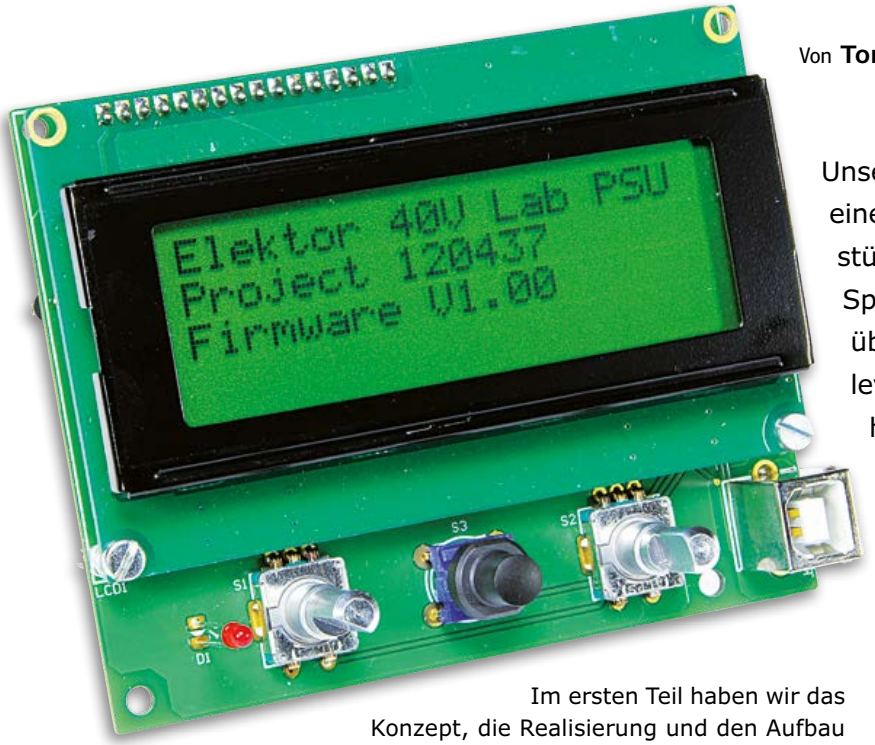


Bild 2.
Das Breakout-Board mit ADS1115 ist dank durchgehender SMD-Bestückung winzig. Damit Sie beim Aufbau keinen Stress haben, ist die Platine fertig bestückt im Elektor-Store erhältlich.

VariLab 402 (2)
Steuer- und Anzeigemodul



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

Unser Labornetzgerät VariLab 402 wird von einem Mikrocontroller gesteuert, er unterstützt den Benutzer beim Einstellen von Spannungen und Strömen. Außerdem überwacht der Mikrocontroller systemrelevante Größen und sorgt für die Sicherheit. Mess- und Einstellwerte werden auf einem vierzeiligen LC-Display ausgegeben, für die Weiterverarbeitung ist eine USB-Schnittstelle vorhanden.

Im ersten Teil haben wir das Konzept, die Realisierung und den Aufbau unseres Labornetzgeräts „VariLab 402“ beschrieben. An dieser Stelle folgt das noch fehlende, separate Steuer- und Anzeigemodul. Die wichtigsten Bauteile sind ein Mikrocontroller ATxmega128A4U-AU sowie ein LC-Display, das 4 · 20 Zeichen darstellen kann. Der Mikrocontroller hat bereits diverse ADCs und DACs zum Messen und Stellen von Spannungen und Strömen an Bord, so dass sich der Schaltungsumfang in Grenzen hält.

Schaltungsbeschreibung

Der ATxmega128A4U-AU, IC1 in **Bild 1**, übernimmt die meisten Aufgaben, die mit dem Einstellen, Überwachen und Anzeigen von Spannungen und Strömen zu tun haben. Spannungsregler IC3 stellt für ihn die Betriebsspannung 3,3 V aus der symmetrischen Betriebsspannung ±5 V bereit, die von der Hauptplatine kommt.

Sämtliche Mess- und Einstellspannungen sind gemeinsam mit der Betriebsspannung ±5 V auf Steckverbinder K3 zusammengeführt. Die Signale für die Ausgangsspannung und den Ausgangsstrom gelangen über K3 zu ADC3 und ADC4 des ATxmega, das Strommesssignal wird zuvor von Spannungsfolger IC2D gepuffert. Widerstand R5 ist so dimensioniert, dass zum Messen der Ausgangsspannung annähernd der gesamte Bereich von ADC3 genutzt wird.

Die Spannungen Vsmps‘ und 48Vin werden ebenfalls zum Mikrocontroller geführt, sie werden von ADC5 und ADC6 gemessen. Widerstand R6 setzt Vsmps‘ auf 1/20 herab, so dass sich die Spannung im Messbereich von ADC5 bewegt. Die Dioden D1…D5 schützen die ADC-Eingänge vor überhohen oder negativen Eingangsspannungen.

Ausgangsspannung und Ausgangsstrom werden mit zwei Spannungen eingestellt, die Stellbereiche betragen 0…+4 V für die Spannung und 0…+2 V für den Strom. IC6, der bewährte LM336, liefert die Referenzspannung +2,5 V für die DACs. Die interne Referenzspannungsquelle des Mikrocontrollers hat sich als zu ungenau erwiesen. Mit Mehrgang-Trimmpoti P1 lässt sich die Referenzspannung exakt einstellen. Normalerweise kann sie auf Maximum gesetzt werden, denn die Kalibrierung wird über die Software vorge-

nommen. Die Opamps IC2A und IC2B verstärken die Ausgangsspannungen der DACs um den Faktor 2, so dass theoretisch eine Spannung bis +5 V verfügbar ist.

Ein Handicap der internen DACs ist die Eigenschaft, dass der Ausgangsspannungsbereich nicht ganz bis 0 V herabreicht, das Datenblatt gibt maximal +0,15 V an. Da der Einstellbereich des Labornetzgeräts bei 0 V beginnen soll, erhalten die Opamps Offsetspannungen in Höhe von +0,3 V. Diese Spannung wird mit R11 und R12 aus der Referenzspannung +2,5 V des Spannungsnormals IC6 abgeleitet und von IC2C gepuffert.

Die Opamps IC2A...IC2D gehören zu einem Vierfach-Opamp MCP6004-E/SL von Microchip, der Rail-to-Rail-Eingänge und -Ausgänge besitzt. Der Vierfach-Opamp wird über Diode D8 an +5 V/-0,6 V betrieben, weil die Betriebsspannung nur maximal +6 V betragen darf. Die Maßnahme ist notwendig, damit der Einstellbereich der Netzgerät-Ausgangsspannung bei 0 V beginnen kann. Das vierzeilige LC-Display liegt an +5 V, es wird vom Mikrocontroller im 4-bit-Modus gesteuert. Da der Mikrocontroller an +3,3 V arbeitet, haben die steuernden Signale diesen niedrigen Pegel, trotzdem erfüllt das Display uneingeschränkt seine Funktion. Der Kontrast ist mit Trimmpoti P2 optimierbar, und die Hintergrundbeleuchtung kann über Jumper JP3 geschaltet werden. In der aktuellen Software-Version ist die Display-Beleuchtung nur eingeschaltet, solange die Strombegrenzung wirksam ist. Wenn die Hintergrundbeleuchtung ständig eingeschaltet sein soll, muss Spannungsregler IC5 auf der Hauptplatine gekühlt werden. Zum Einstellen von Spannung und Strom sind zwei Drehencoder mit integrierten Drucktastern (S1 und S2) sowie ein einzelner Drucktaster (S3)

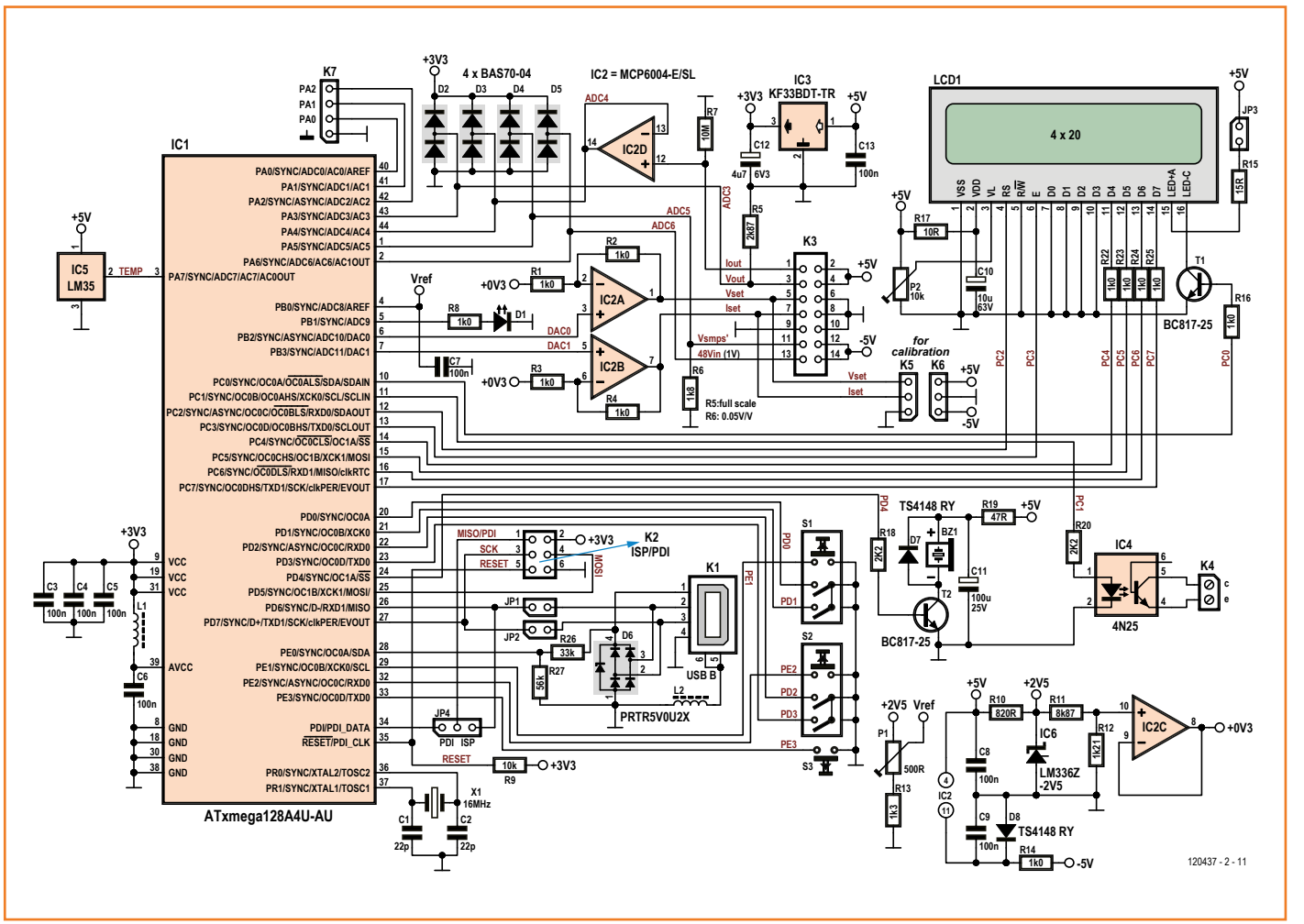
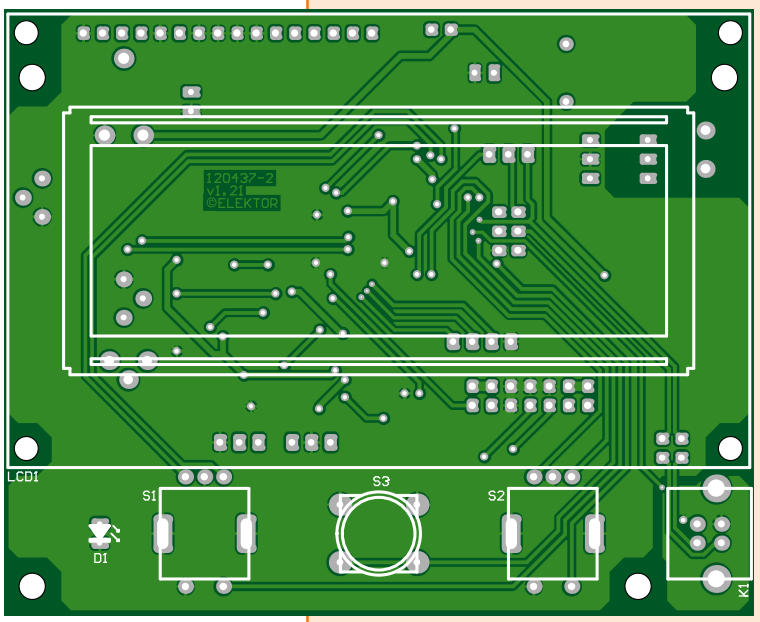
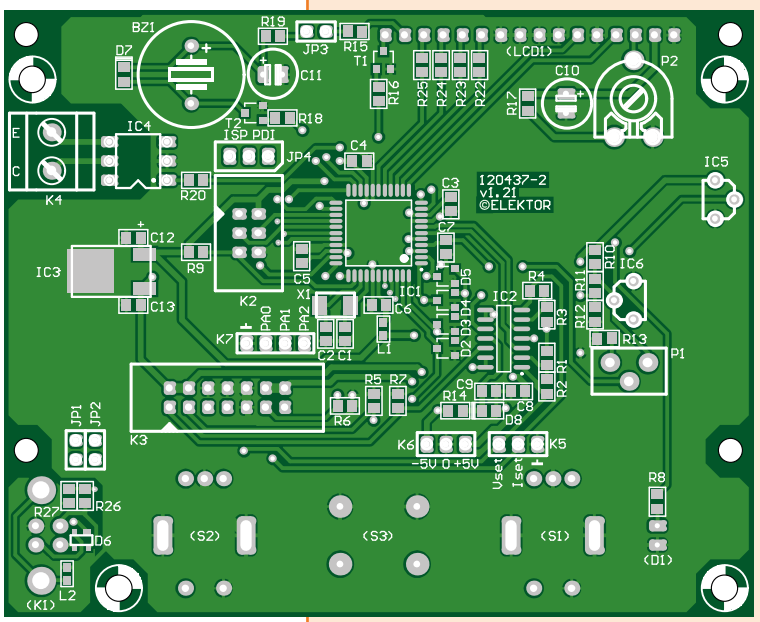


Bild 1.
Schaltung des Steuer- und Anzeigemoduls. Ein ATxmega128 übernimmt fast sämtliche Funktionen.

Bild 2.
Bestückungsplan für die doppelseitige Platine. Das Löten der SMDs erfordert die nötige Erfahrung





Stückliste

Widerstände:
(0W125, SMD 0805, wenn nicht anders angegeben)
R1..R4,R8,R14,R16,R22..R25 = 1 k, 1 %
R5 = 2k87, 1 %, 0W1
R6 = 1k80, 1 %
R7 = 10 M, 1 %
R9 = 10 k, 5 %
R10 = 820 Ω, 1 %
R11 = 8k87, 1 %
R12 = 1k21, 1 %
R13 = 1k30, 1 %
R15 = 15 Ω, 1 %, 0W25
R17 = 10 Ω, 5 %
R18,R20 = 2k2, 5 %
R19 = 47 Ω, 5 %
R21 = nicht vorhanden
R26 = 33 k, 5 %
R27 = 56 k, 5 %
P1 = 500 Ω, 10 %, 0W5, Trimmpoti 25-Gang, Drehschraube oben (Bourns 3296Y-1-501LF)
P2 = 10 k, 20 %, 0W15, Trimmpoti, Drehschraube oben

Kondensatoren:
C1,C2 = 22 p/50 V, 5 %, SMD 0805 C0G/NP0
C3...C9,C13 = 100 n/25 V, 10 %, SMD 0805 X7R
C10 = 10 µ/63 V, 20 %, ∅ 6,3 mm, Raster 2,54 mm
C11 = 100 µ/25 V, 20 %, ∅ 6,3 mm, Raster 2,54 mm
C12 = 4µ7/6V3, 10 %, SMD 0805 Tantal (Case R, AVX TAJR475K006RNJ)

Induktivitäten:
L1,L2 = 330 Ω @ 100 MHz, 0,08 Ω/1,7 A, SMD 0603

Halbleiter:
D1 = LED 3 mm rot, mit Drahtanschlüssen
D2,D3,D4,D5 = BAS70-04, SMD SOT-23
D6 = PRTR5V0U2X, SMD SOT-143B
D7,D8 = TS4148 RY, SMD 0805
T1,T2 = BC817-25, SMD SOT-23
IC1 = ATxmega128A4U-AU, SMD TQFP-44 (programmiert, 120437-41)
IC2 = MCP6004, SMD SOIC14
IC3 = KF33BDT-TR, DPAK-3
IC4 = 4N25, DIP-6
IC5 = LM35, TO-92
IC6 = LM336Z-2V5, TO-92

Außerdem:
K1 = USB-Buchse Typ B für Platinenmontage, vertikale Ausführung (Lumberg 241101)
K2 = Stiftleiste 2x3-polig, Raster 2,54 mm
K3 = Stiftleiste 2x7-polig, Raster 2,54 mm
K4 = Kabelschraubklemme 2-polig, Raster 5 mm
K5,K6,JP4 = Stiftleiste 1x3-polig, Raster 2,54 mm
K7 = Stiftleiste 1x4-polig, Raster 2,54 mm
JP1,JP2,JP3= Stiftleiste 1x2-polig, Raster 2,54 mm und Jumper für JP1,JP2,JP3,JP4
S1,S2 = Drehencoder 18 PPR, mit Drucktaster (z. B. Alps EC11E183440C)
S3 = Drucktaster für Platinenmontage (z. B. Multimec RA3FTL6)
und Kappe für S3, Höhe 19 mm (z. B. Multimec 1S09-19.0)
BZ1 = Piezo-Tongeber, 1,5..16 $V_{=}$, max. 8 mA, ∅ 14 mm, Raster 7,62 mm
X1 = Quarz 16 MHz, C_{load} 18 p, 5x3,2 mm (Abracon ABM3-16.000MHZ-B2-T)
LCD1 = LCD 4 x 20 Zeichen, mit Beleuchtung (Elektor 120061-73)
Platine 120437-2

vorhanden. Mit S3 lässt sich die Ausgangsspannung unmittelbar ein- oder ausschalten. Normalerweise werden mit S1 die Spannung und mit S2 der Strom eingestellt.
Akustische Signale, unter anderem beim Einsatz der Strombegrenzung, erzeugt Tongeber BZ1, er wird über Transistor T1 vom Mikrocontroller aktiviert. LED D1 signalisiert, dass der Mikrocontroller läuft und die Software abarbeitet. Temperaturfühler IC5, ein LM35, misst die Temperatur im Labornetzteil-Gehäuse, sie kann auf dem LC-Display angezeigt werden. Der LM35 hat seinen Platz auf der Platine, eingebaut im vorgeschlagenen Gehäuse ist der Abstand zum Kühlkörper der Hauptplatine kurz. Bei abweichender mechanischer Konstruktion kann der LM35 über ein Kabel mit der Schaltung verbunden werden.
Ausgang K4, mit Optokoppler IC4 galvanisch vom Mikrocontroller getrennt, ist für allgemeine Schaltzwecke vorgesehen. In der aktuellen Software-Version ist die Funktion dieses Ausgangs noch nicht implementiert.
Der USB-Anschluss K1 dient zum Datenaustausch mit einem PC. Es ist möglich, dem PC Messdaten zur Verarbeitung zu übergeben, in umgekehrter Richtung kann der PC über ein Terminalprogramm Spannung und Strom steuern. Die Datenleitungen der USB-Buchse liegen normalerweise über die Jumper JP1 und JP2 am Mikrocontroller. Die Verbindung ist durch Entfernen der Jumper trennbar. ESD-Schutzdiode D6 schützt die Mikrocontroller-Eingänge vor Überspannungen auf den USB-Datenleitungen. Spannungsteiler R26/R27 ist hinzugefügt, um das Hängenbleiben der Mikrocontroller-Software zu verhindern, falls die USB-Verbindung unterbrochen wird.
Mit Jumper JP4 lässt sich die Programmierung des Mikrocontrollers zwischen ISP und PDI umschalten. Im Normalfall ist PDI die richtige Wahl, sie wird zum Beispiel vom Programmer Atmel AVRISP mkII unterstützt. Der Anschluss für den Programmer ist bei beiden Programmierarten der Steckverbinder K2.
An Kontaktleiste K5 liegen außer Masse die Spannungen Vset und Iset, so dass diese Spannungen beim Kalibrieren bequem gemessen werden können. Wenn das Steuer- und Anzeigemodul nicht mit der Hauptplatine des VariLab 402, sondern mit einer anderen Schaltung kombiniert wird, ist K6 der Anschluss für die symmetrische Betriebsspannung ±5 V. Der Strombedarf beträgt +50 mA/-5 mA, er steigt bei aktiver LCD-Hintergrundbeleuchtung auf +105 mA/-5 mA.

Zu erwähnen bleibt noch K7, hier sind die noch freien Portleitungen PA0, PA1 und PA2 des Mikrocontrollers herausgeführt.

Aufbau

Das Layout der Platine für das Steuer- und Anzeigemodul unseres VariLab 402 ist in **Bild 2** wiedergegeben. Auf der kompakten Platine haben das vierzeilige LC-Display, die Bedienelemente (siehe **Bild 3**) und die übrigen Bauelemente ihren

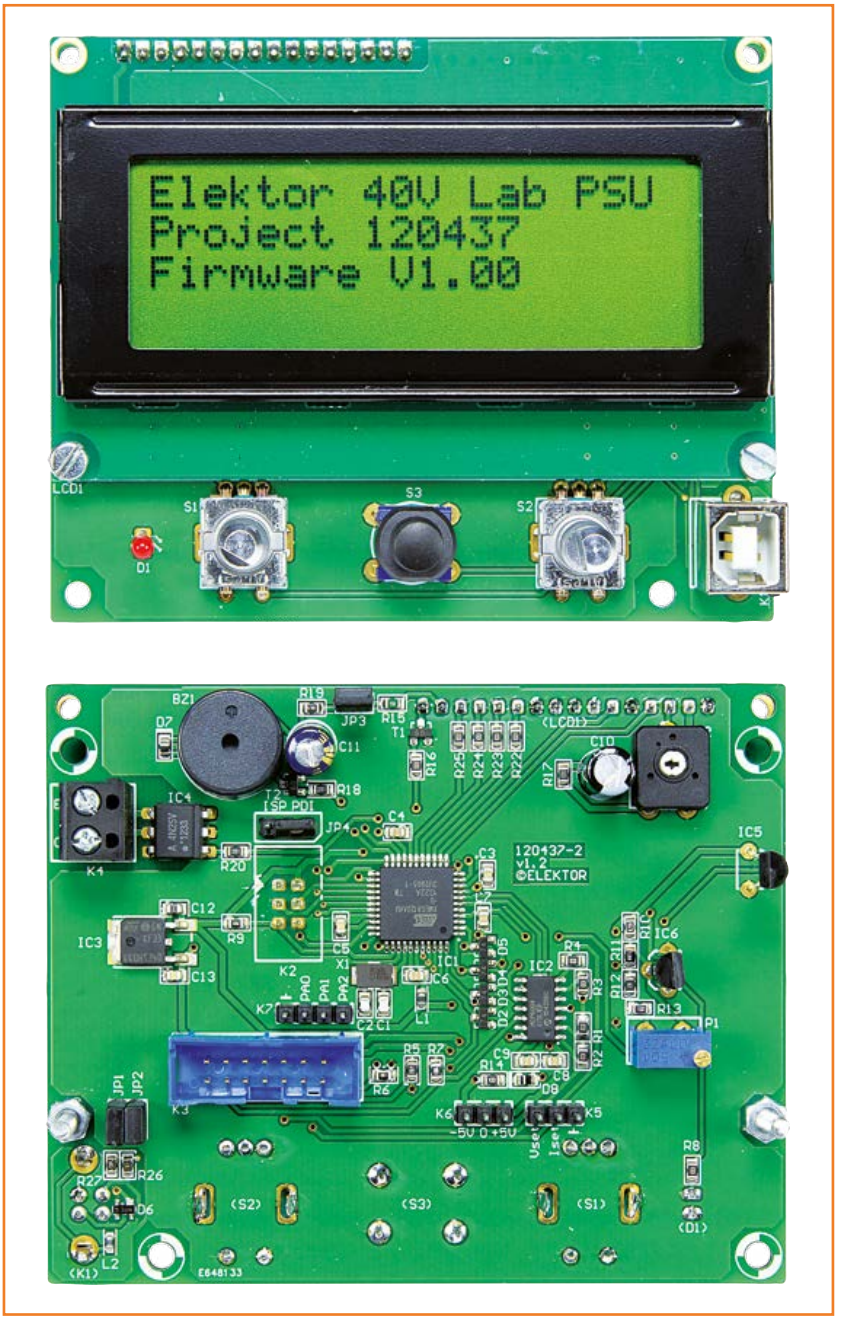


Bild 3.
Vorder- und Rückseite des aufgebauten Moduls. Das LC-Display und die Bedienelemente befinden sich auf der Oberseite.

Bild 4.
Das LC-Display ist auf der Platine mit Gewindeschrauben M3 befestigt.

Platz. Bauelemente sind sowohl auf der Ober- als auch auf der Unterseite der Platine montiert. Auf der Seite mit dem Bestückungsaufdruck befinden sich die SMDs, die bedrahteten Bauelemente und die Kontaktleisten. Die andere Seite ist der Platz des LC-Displays, der Drehencoder, der Drucktaster, der senkrecht stehenden USB-Buchse und der Betriebs-LED.

Wenn der Aufbau erfolgreich sein soll, muss die Stückliste beim Beschaffen der Bauelemente strikt beachtet werden. Ebenso wie auf der Hauptplatine ist die Montage der SMDs von Hand nur mit geeignetem Lötgerät und SMD-Löterfahrung möglich. Das LC-Display wird stehend auf kurzen Drahtabschnitten oder auf einer 16-poligen Stiftkontaktleiste einige Millimeter über der Platinenfläche montiert. Mit Anschlussdrähten, die von der Gegenseite durch die Platine ragen, darf das LC-Display nicht in Berührung kommen. An der Unterkante wird das LC-Display mit zwei Gewindeschrauben M3 befestigt. Zwei Muttern stellen, wie **Bild 4** zeigt, den Abstand zwischen Platine und LC-Display her. An der Oberkante ist diese Maßnahme nicht notwendig, dort sorgen die Anschlussdrähte oder Kontaktstifte für den notwendigen Halt.

Wenn die Bauteile montiert sind, können die Jumper auf JP1 und JP2 (USB-Verbindung), JP3 (LC-Display-Hintergrundbeleuchtung) und JP4 (Programmiermodus PDI) gesteckt werden. Der letzte Schritt ist das Herstellen der Verbindung von K3 mit der Hauptplatine über ein 14-adriges Flachkabel. Diese Verbindung soll möglichst kurz sein, damit Störeinflüsse minimiert werden. Weitere Verbindungen zwischen den Platinen sind nicht erforderlich.

Für das Programmieren des Mikrocontrollers stehen zwei Alternativen zur Wahl: Der Elektor-Shop kann einen gebrauchsfertig programmierten Mikrocontroller ins Haus liefern (Bestellnummer 120437-41, siehe [2]), oder der Mikrocontroller wird mit einem geeigneten Programmer, beispielsweise Atmel AVRISP mkII und Atmel Studio, in eigener Regie programmiert. Der Programmer ist mit Steckleiste K2 zu verbinden. Die Quell- und Hex-Dateien stehen auf der Projektseite [2] zum freien Download bereit.

Im dritten Teil dieses Beitrags, der in der nächsten Elektor-Ausgabe erscheint, geht es um die Software, den Einbau in das vorgeschlagene Gehäuse, die Inbetriebnahme und die Praxis mit unserem VariLab 402.

(140373)gd

Weblinks

[1] www.elektor.de/120437
[2] www.elektor.de/140373
[3] www.elektor-labs.com/120437

Elektor.Labs wird tragbar

Von **Clemens Valens**
(Elektor-Labor)

Tragbar ist in!

In den vergangenen Monaten gab es viele für Elektroniker interessante Neuigkeiten von Intel, des weltweit größten Herstellers von PC-Mikroprozessoren. Nicht nur, dass Big Blue die neuen Galileo-V2- und Edison-Boards mit Arduino-Unterstützung vorstellte, das Unternehmen war auch Hauptsponsor der oben genannten Messe in Rom. Dort kündigte Intel an, Marktführer in Sachen tragbarer (oder besser „anziehbarer“) Elektronik werden zu wollen und präsentierte das MICA-Armband (My Intelligent Communication Accessory). Intel hat in Start-ups investiert und gemeinsam mit Mode-Designern Referenz-Produkte entwickelt, um den Markt anzustoßen und zu zeigen, was neben der Anzeige der Herzfrequenz und der Körpertemperatur alles möglich ist. Schauen Sie sich einmal den (auf einem Edison basierenden) Mimo-Baby-Monitor an!

Elektor mag da natürlich nicht zurückstehen und deshalb hier ein Aufruf für Wearable-Electronics-Projekte auf Elektor.Labs: Haben Sie eine „wearable“ Elektronik entworfen? Haben Sie eine Idee für ein solches Gerät? Veröffentlichen Sie Ihre Ideen und Projekte auf Elektor.Labs und wir werden sehen, wie man die besten in reale Produkte umwandeln kann.
www.elektor-labs.com/wearable

3D-Druck - the never ending story

Vor einigen Monaten habe ich über die nutzlosen Dinge gewettert, die Menschen mit ihren 3D-Druckern produzieren. Ich machte den Vorschlag, mit dem 3D-Drucker neue „Tinte“ (Filament) zu drucken, um unseren Planeten vor dem Ersticken in Polymer-Müll zu retten. Bei einem Wettbewerb der italienischen Zeitschrift *Focus* auf einer Messe Anfang Oktober in Rom fand ich viele nutzlose gedruckte 3D-Objekte. Um ehrlich zu sein, es gab zwar einige sehr nützliche Anwendungen, aber leider waren diese in der absoluten Minderheit. Das Siegerprojekt war zu meiner großen Überraschung der ... 3D-Drucker-Filament-Extruder! Eine Art 3D-Drucker, der 3D-Drucker-Tinte druckt. Da es sich um ein italienisches Projekt handelte, war die Aufregung natürlich groß. Das Objekt war wunderbar ästhetisch und würde die Zierde jeder Küche sein - im Gegensatz zu einem hässlichen 3D-Drucker. Die Polymer-Zutaten werden oben hineingegeben und vorne heraus kommt ein Filament, das auf eine abnehmbare Spule gewickelt wird. Sehr wahrscheinlich ist es auch geeignet, um Spaghetti herzustellen.

Im Wettbewerb waren noch einige ähnlich tolle Objekte, aber keines so schön wie diese Spaghettidruckmaschine. Fotos gibt es hier:
www.elektor-labs.com/3d-printing

(140365)

DesignSpark Tipps & Tricks

Mit Bauteilen arbeiten

Von **Neil Gruending**
(Kanada)

In der letzten Folge ging es um die Neunummerierung von Bauteilen. Heute tauchen wir noch weiter ein in die Materie.

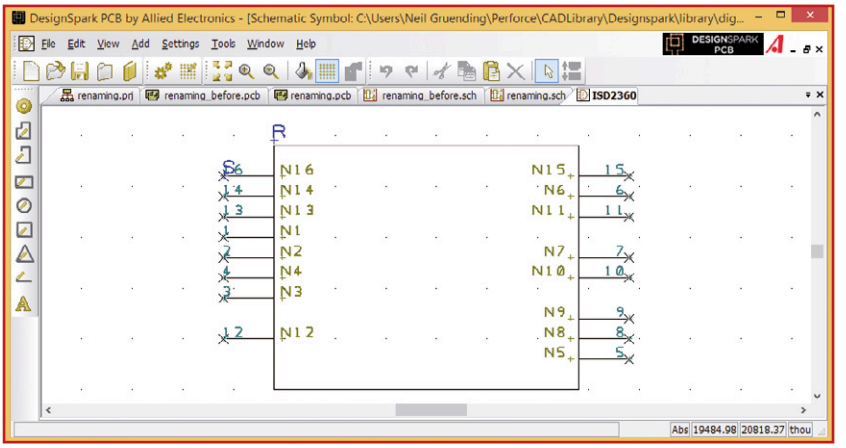


Bild 1.
Ergebnis des Schaltplan-Wizards.

Wenn es um Bauteile geht, sollte man mehr können als nur Nummern vergeben und platzieren. Nun folgen zusätzliche Details. In der dritten Folge ging es um die Verwendung von Libraries in DesignSpark, in der alle Infos zu Bauteilen zusammenlaufen. Von dieser beziehen die Tools für Schaltplan und Platine ihre Informationen, ohne sich gegenseitig zu stören.

Erstellen und Bearbeiten von Bauteilen

Das Erstellen und Bearbeiten von Bauteilen kann in massive Arbeit ausarten, wenn man jedes Mal mit einer leeren Seite anfängt. Wenn man in der Library aber ein ähnliches Bauteil findet, dann kopiert man es einfach mit dem Befehl „Copy To“ im Fenster des Library-Managers und ändert es anschließend. Wenn das nicht möglich ist, sind die Bauteile-Wizards von DesignSpark beim neuen Anlegen von Bauteilen eine große Hilfe. Am Beispiel eines Audio-ICs wird demonstriert, wie diese Wizards funktionieren.

Am liebsten fange ich mit dem Schaltplan-Symbol an. Hierzu öffnet man den Schaltplan-Wizard durch Klick auf den Wizard-Button im Library-Manager. Nach der Beantwortung einiger Fragen wird automatisch ein Schaltplan-Symbol erstellt, das man anpassen kann. Mit etwas Bearbeitung ergibt sich das Symbol von **Bild 1**. Die Pins sind zwar an der richtigen Stelle, doch bei genauerer Betrachtung bemerkt man die vorgegebenen Bezeichnungen N1 zu N16. Für den Moment ist das in Ordnung, da man sie später noch auf der Bauteile-Ebene ändern kann. Mit temporären Texten über den Pin-Namen kann man kurz ausprobieren, ob sie in das Symbol hineinpassen. DesignSpark verfügt auch über einen Wizard für das Platinen-Layout des Bauteils, mit dem sich viele Optionen ergeben. Das Beste daran ist, dass man damit alle Pads mit dem richtigen Abstand erzeugen lassen kann. Im letzten Schritt verknüpft man mit dem Bauteile-Wizard das Schaltplan-Symbol mit seinem Platinen-Layout.

Wenn man das gerade erzeugte Symbol und sein Layout auswählt, erscheint das Fenster für die Zuordnung der diversen Pins (siehe **Bild 2**). Hier kann man die Pins benennen und die Pins des Schaltplan-Symbols mit den Pins des Layouts verknüpfen. Im vorliegenden Fall stimmen die Nummern der Pins von Symbol und Layout schon überein. Man wählt daher einfach „Assign 1-to-1“. Man kann natürlich auch jede beliebige andere Pin-Verknüpfung vornehmen. In der Spalte „Terminal Name“ kann man beliebige Pin-Bezeichnungen eingeben.

Ein Klick auf „Next“ erstellt das Bauteil und man kann es in die Library sichern. **Bild 3** zeigt, wie das eben erstellte Bauteil im Schaltplan aussieht. Doch was ist, wenn man Änderungen macht und die Schaltung updaten muss?

Bauteile-Update

Wenn man ein Bauteil in einer Library ändert, dann wird diese Änderung nur in der Library-Datei und noch nicht im Schaltplan und auch nicht in der zugehörigen Platine übernommen. Deren Dateien enthalten nämlich eine Kopie des Bauteils, sodass nachträgliche Änderungen der Library nicht plötzlich einen Schaltplan oder eine Platine verändern. Also muss man diese lokalen Bauteilkopien irgendwie updaten.

Ein simpler Weg dazu ist, das betreffende Bauteil zu löschen und wieder neu hinzuzufügen. Doch dieses kann unerwartete Auswirkungen auf

die Netzverbindungen haben. Speziell bei Platinen riskiert man dadurch die Verknüpfung zum zugehörigen Schaltplan. Es geht aber einfacher und sicherer: Man geht ins Menü „Tools“, wählt „Update Components“ und dann aus, welches Bauteil man updaten will. Das funktioniert gleichermaßen im Schaltplan und im Platinen-Layout. Man kann nach diesen Bauteilen suchen oder aber direkt die ausgewählten Bauteile updaten. Ein automatisches Update aller Komponenten macht normalerweise keinen Sinn, außer wenn man sich sicher ist, dass dadurch keine unerwünschten Änderungen folgen, die man dann nicht immer auf ein bestimmtes Bauteil zurückführen kann. Auf jeden Fall zeigt sich dann das Fenster „Update Components“ von **Bild 4**. Bei der Platine sieht das Update-Fenster sehr ähnlich aus.

Die Option „Only update item when version is different in library“ erklärt sich selbst: Ohne geändertes Bauteil in der Library passiert nichts. Die Option „Keep value positions“ sorgt dafür, dass alle sichtbaren Bauteil-Felder an der ursprünglichen Stelle bleiben. Das funktioniert zwar meistens, doch sollte man das Resultat trotzdem überprüfen. Auch wenn DesignSpark den Ankerpunkt beibehält, so kann sich doch z.B. die Textausrichtung auf den Wert ändern, der in der Library steht. Wenn man die Option „Keep existing component values“ aktiviert, werden die Felder mit den Bauteilwerten im Projekt beibehalten. Andernfalls werden sie durch die Werte in der Library ersetzt. Das ist wichtig, wenn man den Wert eines Bauteilfelds in der Library geändert hat und will, dass sich diese Änderung auch im Projekt zeigt. Bei der Platine gibt es noch die zusätzliche Option „Remove pad style exceptions“. Normalerweise bleibt die Option aktiviert, so dass DesignSpark alle Fehler löscht, die sich mit dem vorherigen Layout des Bauteils ergeben haben.

Ersetzen und Ändern von Bauteilen

Manchmal muss man Bauteile aber einfach ersetzen oder ändern. Ich persönlich ziehe unterschiedliche Bauteile für jeden Widerstandswert vor. Auf diese Weise kann ich jedem Bauteil nicht nur den richtigen Wert, sondern auch die korrekte Teilenummer des Herstellers zuweisen. Als Resultat kann ich dann sehr einfach eine korrekte und vollständige Stückliste erstellen. Wenn ich also einen anderen Widerstandswert brauche, ändere ich nicht den Wert des Bauteils, sondern gleich das ganze Bauteil. Wenn das Bauteil noch nicht auf der Platine positioniert ist, dann spielt es keine Rolle, ob man ein Bauteil löscht und es durch ein anderes ersetzt. Falls es aber schon auf der Platine steckt, wird das Bauteil zunächst von der Platine gelöscht und dann das andere Bauteil hinzugefügt. Besser ersetzt man das Bauteil, ohne es zuvor zu löschen. Hierzu wählt man zuerst das Bauteil aus und dann seine Eigenschaften. Dann klickt man auf „Change“, worauf sich das Fenster „Change Component“ öffnet, in dem man das neue Bauteil aussucht. Bei dieser Methode bleibt das neue Bauteil auf der alten Stelle auf der Platine und man muss auch nicht überprüfen, ob die Bauteilenummer noch stimmt.

Fazit

In dieser Folge ging es um das Erstellen von Bauteilen mit den DesignSpark-Wizards und darum, wie man Projekt-Updates bei geänderten Libraries durchführt. In der nächsten Folge wird es um multiple Bauteil-Gates und versteckte Pins für die Stromversorgung gehen.

(140367)

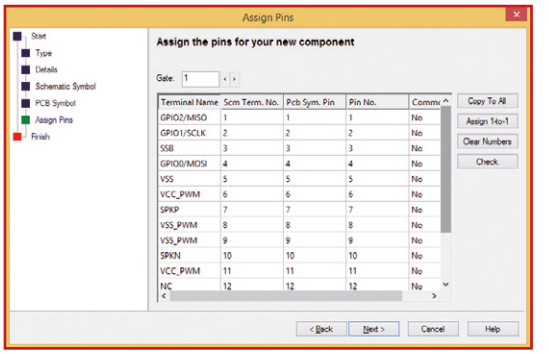


Bild 2.
Fenster zur Verknüpfung von Pins.

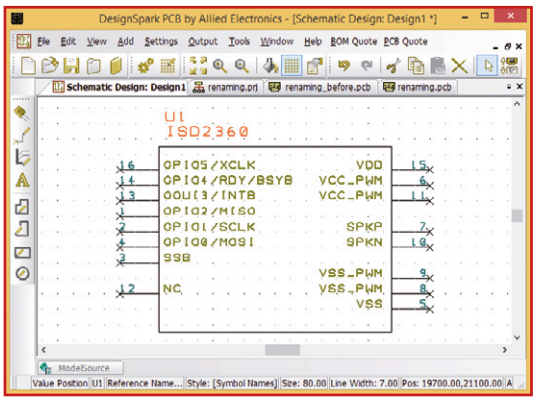


Bild 3.
Fertiges Bauteil.

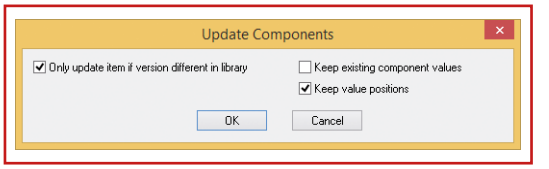


Bild 4.
Das Fenster „Update Components“.

Tantal-Tropfen
Seltsame Bauteile (11)

Von **Neil Gruending**
(Kanada)

Tantal-Tropfen wie in **Bild 1** sind eine Form von Elektrolytkondensatoren. Tantal-Pulver wird um einen Tantaldraht zu Pellets gepresst und dann gesintert, um das Pulver zu verbacken und einen sicheren Anschluss des Anodendrahtes zu gewährleisten. Anschließend wird der Sinterblock durch Anlegen einer hohen Polarisationsspannung (genannt Formierungsspannung) im Elektrolysebad mit einer dünnen (Tantalpent-) Oxidschicht versehen. Diese Schicht stellt das Dielektrikum dar. Der Sinterkörper wird nun mit flüssigem Elektrolyten ausgestattet, der in alle Poren des Tantalblocks eindringt, und zum Abschluss mit einem Kathodenanschluss versehen, anschließend erhitzt (der Elektrolyt wird hart) und mit Lack oder Kunststoff umhüllt.


1

Source: Wikipedia

Da die im Pulver eingeschlossenen Poren des Sinterblocks eine sehr große Oberfläche und die Oxidschicht eine geringe Permittivität aufweisen, erreichen diese Kondensatoren eine für ihre Größe sehr hohe Kapazität. Als weitere Eigenschaften seien der sehr breite Temperaturbereich und der sehr niedrige äquivalente Serienwiderstand (ESR) im Vergleich etwa zu Kondensatoren mit Aluminium-Elektrolyt genannt.

2

Diese Eigenschaften machen Tantalkondensatoren zur guten Wahl in Stromversorgungen. Sie sind lange Zeit statt kleiner Aluminiumkondensatoren verwendet worden, allerdings sind sie auch anfällig für Transienten und Brummspannungen. Tantalkondensatoren können kleine Spannungsschwankungen innerhalb ihres Betriebsbereichs absorbieren, aber wenn die Transienten zu groß sind, können sie durch die dünne Oxidschicht schlagen und den Kondensator beschädigen. Durch den niedrigen ESR können Spannungsspitzen unbeabsichtigt verstärkt werden und auch schädliche Stromspitzen verursachen. Sobald ein Kondensator beschädigt ist, werden seine Spezifikationen schlechter bis hin zum Kurzschluss. Ist die Stromquelle groß genug, dann kann der Kondensator tatsächlich explodieren und in die Leiterplatte brennen.

Aber Tantal-Kondensatoren sind so lange sicher, wie ihre Spannungs- und Stromwerte beachtet werden. Sie trocknen auch im Gegensatz zu Elkos mit nassem Elektrolyten nicht aus. Aluminiumelkos können nahe an ihren Grenzwerten betrieben werden, bei Tantalelkos sollten Sie es bei etwa 50% der Spannungsfestigkeit belassen. Sie sollten auch sicherstellen, dass der Brummstrom nicht zu hoch wird, indem Sie mehrere Tantals parallel schalten. Der Nachteil wiederum ist, dass der kleinere ESR mit der Induktivität der Leiterbahn einen Schwingkreis darstellt, was wieder zu unerwarteten Spannungsspitzen in Stromversorgungsanwendungen führen kann.

Elektronische Geräte aus den späten 80ern bis Mitte der 90er Jahre verwendeten oft in Epoxy eingetauchte SMD-Tantalkondensatoren in ihren Netzteilen und zur Entkopplung. Leider sind diese Kondensatoren für eine ziemlich hohe Ausfallrate (**Bild 2**) berüchtigt. In manchen Reparaturfällen ist es deshalb sinnvoll, alle Tantalkondensatoren in der Schaltung zu testen und zu ersetzen. Die vielen Ausfälle sind eigentlich schwer nachzuvollziehen, aber nach Kemet [1] wurden Tantalkondensatoren anfänglich nicht wirklich für Stromversorgungsanwendungen konzipiert. Sie benötigen eine erhebliche Menge an externem Widerstand, um zuverlässig zu arbeiten (in der Regel mehrere Ohm pro Volt). Es ist auch möglich, dass in vielen Entwürfen aus dieser Zeit die Tantals zu nahe an ihren Nennspannungsgrenzen betrieben wurden und deshalb viele auch nur bei kleinen transienten Ereignissen versagten.

Dank MLCC-Keramik- und deutlich verbesserten (und verkleinerten) Aluminiumelkos werden heute Tantals weniger oft eingesetzt. Auch „politisch“ verhalten sich Tantalelkos nicht unbedingt korrekt, die Abbaubedingungen von Tantalerz sind doch eher als fragwürdig zu bezeichnen. Aber wenn Sie viel Kapazität auf kleinem Raum benötigen, sind Tantalkondensatoren kaum zu schlagen. Moderne Tantals sind auch meist gut gegen Überspannungen gesichert. Man muss nur sicherstellen, dass sie innerhalb ihrer Grenzen eingesetzt werden.

(140366)

Weblink: [1] http://canada.newark.com/pdfs/techarticles/kemet/Tantalum-in-Power-Supply-Applications.pdf

Aufgeblähte Caps

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

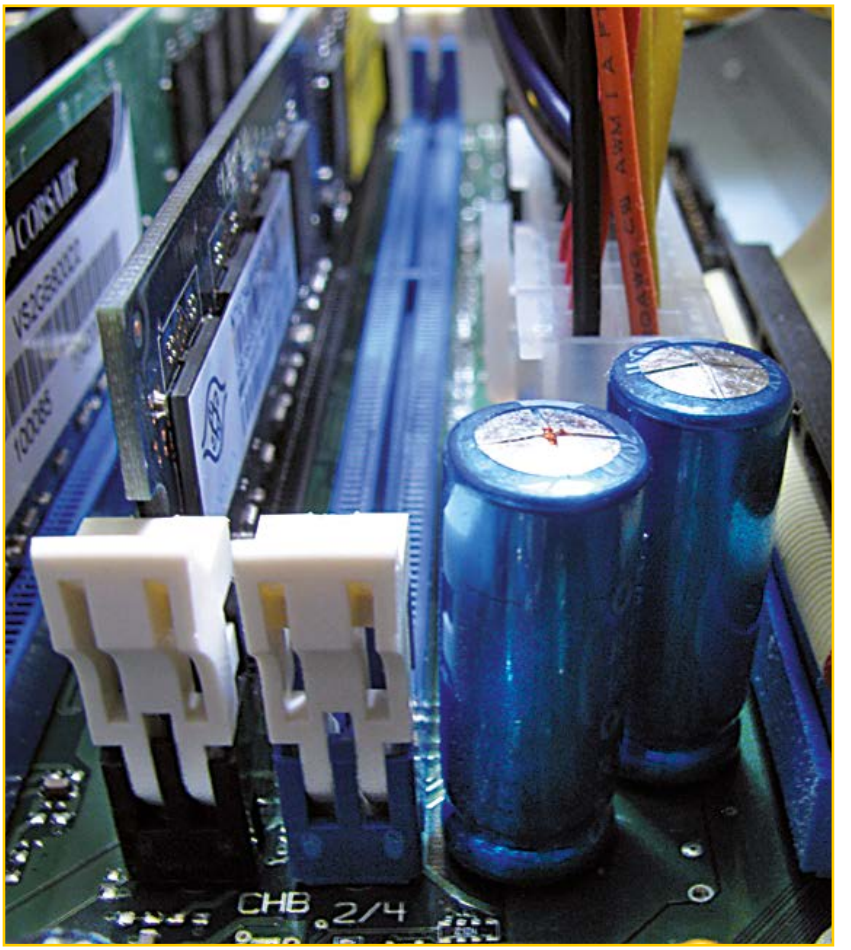
Wertstoffsammelstellen sind ja was Tolles. Ich wollte dort eigentlich nur ein paar ausgenudelte Drucker loswerden, da fiel mein Blick auf einen noch recht rüstig erscheinenden Schrott-PC. Auch ohne Tauchgang im Müllcontainer konnte ich einen Aufkleber „Intel Core™ 2 Duo“ ausmachen und einen weiteren mit dem Produktschlüssel für „Windows Vista™ Home Premium OEM“. Das interessierte mich schon sehr, läuft doch mein aktuelles Laptop-Modell auch noch unter Vista. Und Zeug wegzuschmeißen, das noch funktioniert, auch wenn es nicht mehr strahlend neu ist und etwas zerbeult, ist nicht meine Sache. So habe ich beschlossen, diesem ausrangierten PC zu Hause Asyl zu gewähren und etwas über CPU, RAM, Festplatte und so weiter rauszubekommen.
Zu Hause öffnete ich das Gehäuse. Alle Komponenten mit Ausnahme der Festplatte waren vorhanden und letztere war nicht brutal herausgerissen, sondern der Besitzer hatte sie ordentlich entfernt. Also den Montitor, eine Tastatur und eine Maus angeschlossen und alles unter Strom gesetzt! Es brummte, piepste und das BIOS erschien auf dem Monitor. Hier das, was ich im BIOS über das Gerät herausfand:

- CPU: Intel Core 2 Duo E4500 @ 2.2 GHz
- RAM: 2 GB DDR2 @ 667 MHz
- Mainboard: Fujitsu D2151-A21
- Graphics: NVidia Geforce 8400GS 256 MB
- Optics: DVD-ROM and DVD-R/RW DL drives
- 350-W PSU
- Internal Card Reader
- Cooler Master case

In $2^{n>3}$ Monaten werden wir sicher über diese Konfiguration lachen, aber heutzutage ist es doch eigentlich ausreichend für gelegentliches Surfen, Mails und sogar etwas Musizieren. Bei einem zweiten Besuch der Wertstoffsammelstelle dürfte mir sicher auch eine passende Festplatte in die Hände fallen, so dass ich das Schätzchen wirklich ans Laufen bekommen sollte.
Aber zuerst etwas, was ich bemerkte, als ich das Mainboard genauer betrachtete und überhaupt nicht mag: Sehen Sie die Kondensatoren neben dem ATX-Stecker? Obwohl das Mainboard einwandfrei funktionierte, sagte mir mein elektrotechnischer Sachverstand, dass diese aufgeblähten Kondensatoren kurz vor ihrer Explosion standen. Das ist natürlich schlecht! Doch wenn ein paar Ersatz-Kondensatoren das einzige sind, was benötigt wird, um diesem PC ein zweites Leben zu gewähren, bin ich immer noch ein glücklicher Mensch. Ich bestellte ein paar anständige Low-ESR-Caps (Panasonic FM-Serie), entfernte die Zeitbomben und setzte die neuen Bauteile ein.

Nach der Installation eines neuen Betriebssystems lief der PC ordentlich. Und nach einem Monat tut er es immer noch. Habe ich erwähnt, dass ich einen Hackintosh aus ihm gemacht habe? Vielleicht sollte ich das aber nicht so laut sagen ☺.

(140363)



Mit einem Klick zum C-Projekt

Konfigurator für modulare Software in Atmel Studio

Modulare Softwarebibliotheken wie die EFL haben viele Vorteile. Doch das Anlegen eines eigenen Projekts in einer Entwicklungsumgebung erfordert diverse Schritte, was vor allem Einsteiger abschrecken dürfte. Eine hier vorgestellte, skriptbasierte PC-Software generiert ein neues EFL-Projekt per Mausklick und erzeugt auch gleich den Anwendungscode mit allen Setup-Befehlen. Für Fortgeschrittene gibt es Stellschrauben, mit denen der Speicherverbrauch des Programms minimiert werden kann. Am Beispiel einer Relais-Steuerung mit Display zeigen wir, wie man ein Projekt für kleinere Controller optimiert.

Von **Jens Nickel**

In der letzten Ausgabe haben wir die Vorteile von modularen Softwarebibliotheken wie der EFL dargestellt [1]. Jedes Software-Modul bildet die Eigenschaften des Controllers oder Boards ab, zu dem es gehört, ist aber unabhängig von der sonst verwendeten Hardware im Projekt. Damit kann man die Module ohne Änderungen am Code für ein eigenes Software-Projekt kombinieren; ganz ähnlich wie man fertig entwickelte und ausgetestete Hardware-Module einfach zusammensteckt, um binnen Minuten zu einem funktionierenden Prototypen zu kommen. Leider erkaufen wir uns das Prinzip der Hardware-Unabhängigkeit mit einer wachsenden Anzahl von einzelnen Files, die in das eigene (EFL-)Projekt integriert werden müssen. Für den Controller, das Controllerboard und jedes Erweiterungsboard ist jeweils ein Filepaar (.h/.c) zuständig, dazu kommen mehrere Blockfiles, die für Peripherieeinheiten wie ein Display Low-Level-Funktionen anbieten (sie abstrahieren beispielsweise von der Verdrahtung zwischen Controller und Display). Desweiteren brauchen wir die höheren Bibliotheken (Libraries), welche die Programmierung für den Anwender bequem machen, weil jeder Peripherie-Block desselben Typs auf allen Boards über die gleichen Funktionen angesprochen werden kann. Und natürlich benötigen wir die Common-Files, die Basis der EFL.

Schritt für Schritt

Leider ist es bei modernen Entwicklungsumgebungen wie Atmel Studio nicht damit getan, einfach die benötigten Files in einen Windows-Ordner zu ziehen. Vielmehr müssen alle Codefiles mit Hilfe von Atmel-Studio-Befehlen in das Projekt eingebunden werden. Starten wir in Atmel Studio mit einem „nackten“ Projekt, dann müssen wir zuerst auch die drei EFL-Projekt-Ordner „Common“, „Hardware“ und „Libraries“ anlegen. Die

Pfade in diese Ordner müssen wir überdies dem Compiler bekannt machen. Doch damit ist es natürlich noch nicht getan. Im Hauptfile der Firmware, in dem sich der eigentliche Anwendungscode befindet, müssen alle Board-, Extension- und Library-Header-Files per Include-Anweisung eingebunden werden. Und schließlich darf man auch die Init- und Setup-Funktionen für die Hardware- und Library-Module nicht vergessen.
Natürlich kann man sich einmal ein passendes EFL-Projekt zusammenstellen (oder eines unserer Demos verwenden) und dann durch einfaches Kopieren des gesamten Atmel-Studio-Projektordners neue Projekte erzeugen, die man dann passend abändert. Leider wird dabei auch der Name des ursprünglichen Projektes beibehalten. Das manuelle Umbenennen des Projekts im „Solution Explorer“ löst das Problem nur teilweise, es verbleiben dann Reste des ursprünglichen Projektnamens im Projekt.
Um saubere Demo-Projekte für die vorangegangenen EFL-Artikel zu erzeugen, musste auch ich alle obengenannten Schritte wieder und wieder durchlaufen. Sicher keine Arbeit, die besonderen Spaß macht.

Mit drei Dateien zum Projekt

Bei der Suche nach einer Lösung habe ich zuerst einmal in Atmel Studio ein ganz frisches C-Projekt erzeugt und mir die generierten Ordner und Dateien angesehen. Alle Files befinden sich in einem Hauptordner, der den Namen des Projektes trägt (der Atmel-Studio-User wählt den Namen vor dem Anlegen des Projekts, voreingestellt ist „GccApplication1“). Im Hauptordner findet sich ein File mit dem Projektnamen und der Endung .atsln und ein weiterer Ordner mit dem Namen des Projekts (**Bild 1**). Dieser enthält zwei Files, die ebenfalls den Projektnamen tragen, einmal mit der Endung .cproj, und einmal mit der Endung .c. Daneben findet man noch einen leeren Ordner namens „Debug“.
Die .atsln- und .cproj-Dateien lassen sich mit einem Text-Editor öffnen; es präsentiert sich reinrassiges XML. Das ist schon mal ein großer Schritt auf dem Weg zum automatischen Generieren eines solchen Projekts (ohne dass wir Atmel Studio selbst verwenden müssen). Denn Ordner und Textfiles mit bestimmten Namen und Inhalten anlegen, das können wir mit allen handelsüblichen PC-Programmiersprachen. Etwas Studium des .atsln-Files zeigt, dass hier nur der Projektname relevant ist, die anderen Informationen

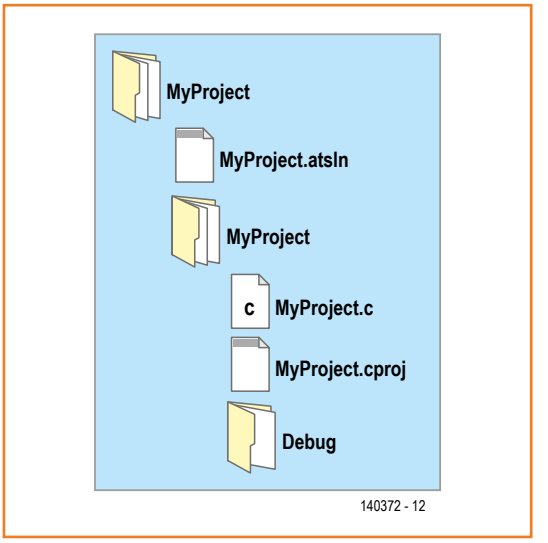


Bild 1.
Diese Filestruktur generiert Atmel Studio beim Anlegen eines neuen C-Projektes.

sind nicht direkt projekt-bezogen. Um uns ein maßgeschneidertes .atsln-File anzulegen, können wir daher eine von Atmel Studio generierte .atsln-Datei als Template verwenden. Dessen Text lesen wir mit einer eigenen Software zunächst ein. Dann ersetzen wir den darin enthaltenen Projektnamen im (XML-)Text durch den gewünschten Projektnamen und speichern das File mit dem neuen Inhalt und dem neuen Namen (nämlich Projektname plus Endung .atsln) ab.
Das .atsln-File gibt es übrigens nur, weil Atmel Studio auf Visual Studio von Microsoft basiert; diese IDE wird unter anderem für VB.NET und C# im Bereich der PC-Programmierung verwendet. In Visual Studio wird zwischen einer „Solution“

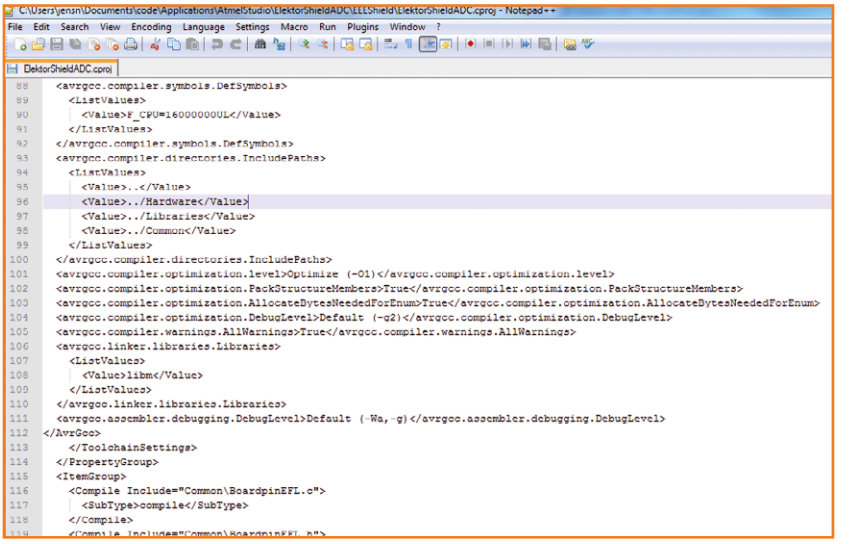
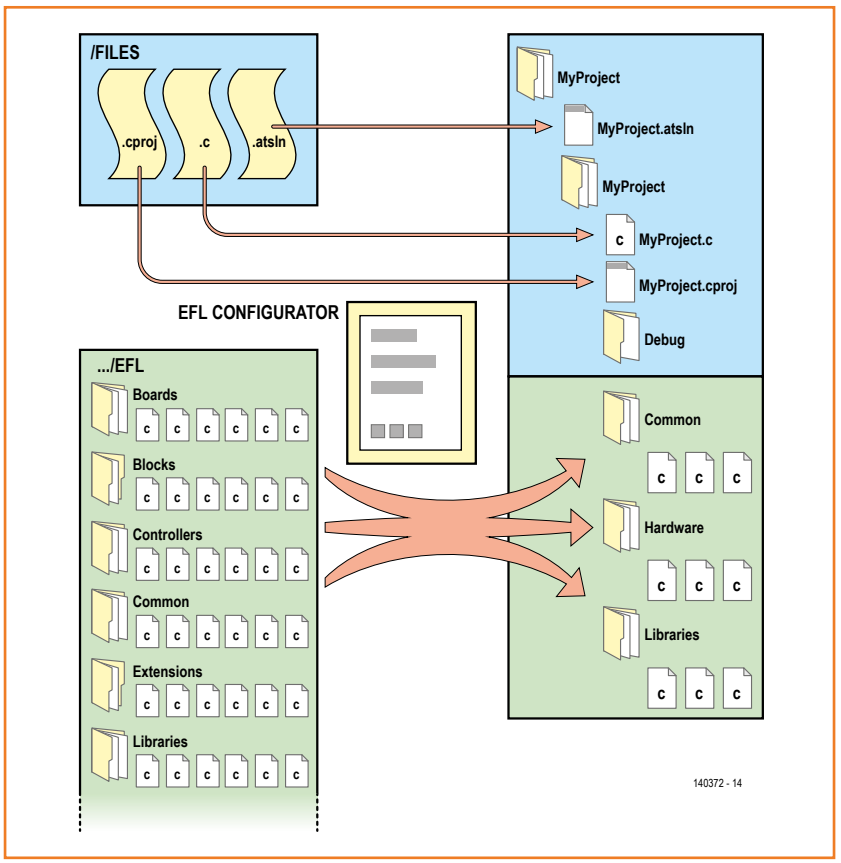


Bild 2.
Die .cproj-Datei enthält alle Informationen zu einem Atmel-Studio-Projekt in XML-Form.

und einem „Project“ unterschieden, eine Solution kann dabei mehrere Projekte enthalten. Bei uns besteht eine Solution aber grundsätzlich immer nur aus einem Projekt, und beide tragen den gleichen Namen.
Ein Blick in das .cproj-File zeigt, dass in dieser Datei alle relevanten Informationen zum eigentlichen Atmel-Studio-Projekt gespeichert sind. Zum Vergleich habe ich mir ein cproj-File eines „manuell“ erzeugten EFL-Projekts angesehen (**Bild 2** zeigt einen Ausschnitt im Editor Notepad++). Alles, was wir benötigen, ist dort in Form von XML-Elementen niedergeschrieben:

- Die Namen für die Unterordner des EFL-Projekts („Hardware“, „Libraries“, „Common“).
- Die passenden Pfade für den Compiler.
- Alle Quellcodefiles, die zum Projekt gehören.
- Die definierten Symbole für den Präprozessor. Bisher haben wir in der EFL schon die Definition F_CPU=16000000UL (Taktfrequenz des ATmega328 auf dem Arduino Uno) benutzt. Mit diesem Symbol kann im Controller-Code beispielsweise der nötige Registerinhalt zum Einstellen der UART-Baudrate berechnet werden.

Bild 3.
Aus drei Template-Dateien und dem Ordner mit allen EFL-Files generiert unser Konfigurator ein vollständiges EFL-Projekt für Atmel Studio.



Bleibt noch ein Blick in die .c-Datei. Es handelt sich um das Hauptfile des Quellcodes mit der main-Funktion, die von Atmel Studio bereits angelegt wird. Natürlich ist auch diese Datei ein Textfile, dessen Inhalt wir leicht manipulieren können.

In **Bild 3** ist auf der rechten Seite dargestellt, wie die Ordnerstruktur eines EFL-Projektes in Atmel Studio aussehen muss.
Zur Erzeugung eines solchen Projekts (oder eines anderen modularen C-Projekts) in Atmel Studio bietet sich nun der folgende Lösungsweg an. Wir deponieren in einem bestimmten Ordner drei Template-Dateien als Ausgangspunkt für die Files ...atsln, ...cproj und ...c. Zuerst erzeugt unsere PC-Software die nötigen Ordner für das Atmel-Studio-Projekt. Dann werden die Template-Dateien der Reihe nach eingelesen, der Inhalt entsprechend manipuliert und schließlich unter neuem Namen an der richtigen Stelle innerhalb der Projekt-Ordner-Struktur abgespeichert. Damit wir es beim automatischen Editieren der Template-Dateien einfacher haben, legen wir innerhalb des Textes einfach zu findende Ausdrücke wie „AAAAA“, „BBBBB“ usw. ab. Sie werden dann durch den erforderlichen Inhalt ersetzt, wie zum Beispiel die XML-Elemente zur Einbindung von Quellcodefiles in das Projekt.
Außerdem legen wir für das EFL-Projekt die Unterordner „Common“, „Hardware“ und „Libraries“ (neben dem Unterordner „Debug“) an und kopieren hier alle benötigten EFL-Files hinein. Die PC-Software weiß, wo die jeweiligen Files im Gesamtbestand der EFL-Dateien zu finden sind („Controllers“, „Boards“, „Blocks“ usw.).

Skript-System

Nun war es zu einer passenden PC-Software nicht mehr weit. Grundsätzlich ist jede Progammiersprache geeignet, die Befehle zum Anlegen eines Ordners, zum Einlesen und Abspeichern von Textdateien und zur Manipulation von Texten und Strings bietet. Ich habe mich hier für ein von mir entwickeltes, skriptbasiertes Framework entschieden, das wir im Verlag unter anderem einsetzen, um Listen und Dateien aus den Daten unseres redaktionellen Planungssystems zu generieren. Bei einer lokalen Anwendung wie der nun folgenden Software sind die Skripts als Textdateien abgespeichert. Sie werden von einer Exe-Anwendung (dem Interpreter bzw. „Skripts-Prozessor“) eingelesen und abgearbeitet. Das erste eingelesene Skript generiert eine HTML-basierte Benut-

zeroberfläche, die von der Exe-Anwendung in einem Web-Steuerelement dargestellt wird. Falls man in der HTML-Oberfläche einen Button betätigt, wird das Ereignis von etwas Javascript „nach außen“ an die Exe-Anwendung weitergereicht, die dann ein weiteres Skript aufrufen kann. Das klingt kompliziert, hat aber einen wesentlichen Vorteil. Wenn wir unserer Software weitere Elemente für die Benutzeroberfläche und/oder weitere Funktionen hinzufügen wollen, dann müssen wir nur die Texte in den Skript-Dateien ändern; die Exe-Datei bleibt immer gleich. Prinzipiell sind die Skripts auch plattformunabhängig. Den Skripts-Prozessor (der mit C# für .NET realisiert ist) gibt es bisher allerdings nur für Windows.

Die Exe-Datei, die außerdem benötigten .DLL-Files und die Skripts befinden sich alle in einem gemeinsamen Ordner, denn man von der Projektseite zu diesem Artikel herunterladen kann [Proj]. Die Skripts sind zusammen mit Konfigurationsfiles und weiteren Dateien, die zur eigentlichen Skript-Applikation gehören, im Unterordner „APP“ untergebracht. In diesem „App-Ordner“ können wir auch unsere drei Templatefiles unterbringen (im Unterordner APP\FILES\).
Fortgeschrittene PC-Programmierer schauen natürlich einmal in die Skriptdateien (im Unterordner APP\SCRIPTS\) und die Daten-Dateien (APP\DATA\) hinein. In den Datenfiles werden zum Beispiel benutzerdefinierte Einstellungen gespeichert. Einen kleinen Überblick über die verwendete Skriptsprache findet man im Kasten.

Projekt-Generator ...

Nach einem Klick auf die Exe-Datei startet die App namens „EFL Configurator“. Zuerst wird die HTML-basierte Benutzeroberfläche aufgebaut und dargestellt (**Bild 4**).
Klein fängt man an: Aus Zeitgründen musste ich mich beim Konfigurator noch auf ein ganz bestimmtes Hardware-Setup beschränken. Es sind noch viele Erweiterungen denkbar, etwa dass man sich zuerst den verwendeten Controller und das Controllerboard aussucht und dann zu einer neuen Oberfläche geleitet wird, wo passende Erweiterungsboards ausgewählt werden können. Bisher ist der EFL Configurator auf das im Sommerheft vorgestellte Elektor Extension Shield [ExtShield] zugeschnitten, das auf einem Arduino Uno sitzt. An dieses können wahlweise angeschlossen werden:

- ECC-RS485-Modul und/oder
- EEC-Modul mit acht Relais oder
- EEC-Modul mit externem 16-bit-ADC

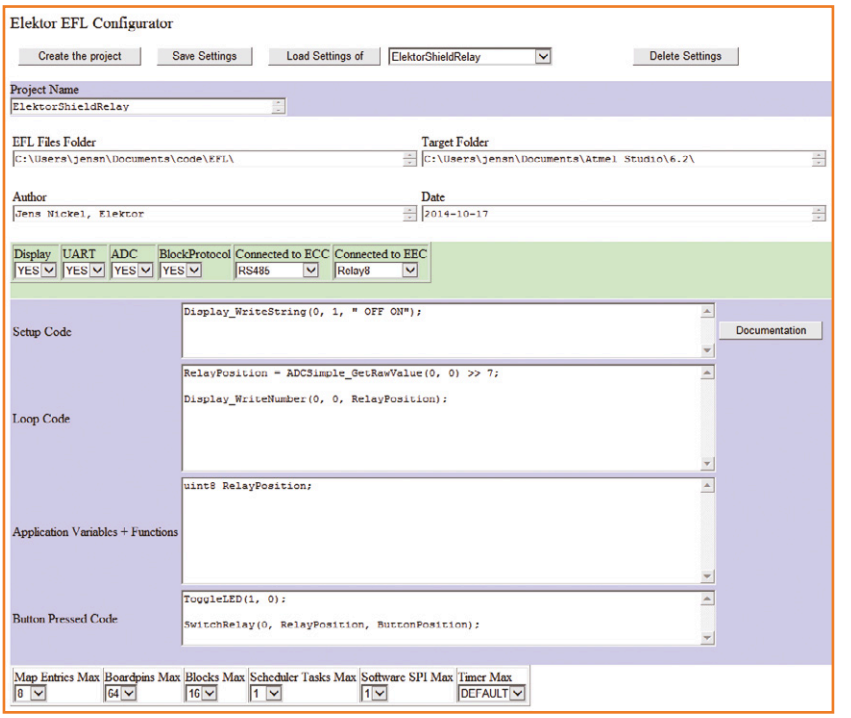


Bild 4.
Der EFL Configurator gewinnt keinen Schönheitspreis, nimmt uns aber viel Arbeit ab. Mit ein paar Klicks können verschiedene Versionen einer Firmware erzeugt werden, um den Einfluss auf den Code- und RAM-Bedarf zu untersuchen.

Alle drei Module und dazu passende Demo-Projekte haben wir in der letzten Ausgabe beschrieben [CModule].

Ganz oben in der Benutzeroberfläche trägt man einen Namen für das Projekt ein. In den Feldern darunter stellt man denjenigen Pfad ein, unter dem man die neueste Version des EFL-Code-Bestands auf dem eigenen Rechner abgespeichert hat (findet sich ebenfalls im Download zu diesem Artikel [Proj]). Der Pfad sollte in den Ordner „EFL“ hineinführen, muss also mit „\EFL\“ enden. Außerdem muss man den Pfad angeben, unter dem die generierten Atmel-Studio-Projekte abgespeichert werden sollen. Dann folgen Textboxen für den Projekt-Autor und das Datum, dessen Format man natürlich nach Wunsch selbst anpassen kann. Beides wird sich später als Kommentar im Quellcode wiederfinden.
Darunter finden sich Auswahlboxen, in denen man einstellt, welche Libraries man im Projekt benutzen möchte. Immer im Projekt enthalten ist die LEDButtonEFL-Library. Die DisplayEFL-, UART-InterfaceEFL- und ADCSimpleEFL-Library nimmt man hinzu, wenn man ein Display, den UART zur Kommunikation oder einen Analogeingang

verwenden will. Die Default-Einstellung ist hier drei Mal „YES“, um alle Funktionen des Elektor Extension Shields und des Arduino (Display, Poti, UART) auf einfache Weise nutzen zu können. Mit dem BlockProtocol [4] lassen sich die Peripherie-Einheiten auf dem Board und den Erweiterungsmodulen von einem Terminalprogramm aus fernsteuern. In den Dropdown-Boxen daneben stellt man ein, welche Erweiterungsmodule man einsetzen möchte.

… und Projekt-Verwaltung

In die Textboxen darunter kann man den applikationsspezifischen Quellcode des EFL-Projektes eintragen. In die oberste Textbox werden alle Funktionen hineingeschrieben, die zu Beginn der Anwendung aufgerufen werden sollen. Danach folgt Code, der immer wiederkehrend abgearbeitet wird. Anwendungsspezifische (globale) Variablen und Funktionen kommen in die nächste Textbox. In der letzten Textbox platziert man den Code, der beim Drücken irgendeines Tasters

Skripts

Die verwendeten Skripts sind in der vom Autor entwickelten Skript-Sprache „Sheets“ geschrieben, sie werden bei einer lokalen App wie dem EFL Configurator in Textdateien gespeichert. Ein Skript kann mehrere Eingabewerte und einen Ausgabewert haben. Da die Skripts ursprünglich dazu eingesetzt wurden, ein XML- oder (X)HTML-Dokument dynamisch zu generieren, heißt der Ausgabewert *XMLResult*. Der Skriptcode selbst ist auch in Form von XML notiert, alle Skript-Kommandos sind XML-Elemente. Die Kommandos lassen sich im Skriptcode mit anderen XML- oder (X)HTML-Elementen mischen, die beim Durchlaufen des Skripts einfach in das XMLResult geschrieben werden. Skripts, die HTML-Seiten (z.B. Benutzeroberflächen) oder XML-Dokumente generieren, lassen sich damit recht einfach notieren (siehe zum Beispiel das Skript „Base.txt“ des EFL Configurators). Das Kommando TEXT schreibt einfach Text, der sich zwischen den Tags befindet, in das XMLResult:

```
<DIV>
<TEXT>Hello World!</TEXT>
</DIV>
```

Das XMLResult enthält dann:

```
<DIV>
Hello World!
</DIV>
```

Mit dem Kommando

```
<TEXT Press='DoSomething' >Do something!</TEXT>
```

wird statt bloßen Texts ein HTML-Button mit der Beschriftung „Do something!“ erzeugt. Wenn man später die Schaltfläche im HTML-Formular drückt, dann wird das Skript *DoSomething* aus der Textdatei *DoSomething.txt* eingelesen und abgearbeitet. In diesem Skript hat man Zugriff auf den Inhalt der verschiedenen HTML-Steuerelemente desjenigen User-Interfaces, von dem aus das Skript aufgerufen wurde. Das können zum Beispiel Textboxen sein. Solche Textboxen können wir uns ebenfalls leicht per Skript generieren lassen:

```
<ENTER ID='EnterText' >Replace the text here</ENTER>
```

Im Skript *DoSomething* können wir den aktuellen Inhalt dieser Textbox über den Ausdruck `@EnterText` erreichen. Denn dieser wird beim Aufruf des Skripts in der Variablen *EnterText* gespeichert und dem Skript als Eingangswert mitgegeben.
Solche einfachen Variablen können wir uns auch innerhalb von Skripts selbst anlegen und mit einem Wert befüllen: Die Kommandos

```
<SET RName='TargetFolder' >C:\MyFolder\</SET>
<TEXT>@TargetFolder</TEXT>
```

schreiben den Text „C:\MyFolder\“ in das XMLResult.
Mit dem Zeichen „@“ bekommen wir Zugriff auf den Inhalt von Variablen. Statt „@“ plus einem Variablennamen können wir aber auch das Ergebnis einer Berechnung in unserem Skript verwenden, den Rechenausdruck müssen wir nur mit „@{“ und „}“einrahmen.
Die Kommandos

```
<SET RName='TargetFolder' >
   @{Add(C:\MyFolder\,MySubfolder)}</SET>
<TEXT>@TargetFolder</TEXT>
```

geben den Text „C:\MyFolder\MySubfolder“ aus. Gleiches können wir aber auch direkt erreichen mit

```
<TEXT>@{Add(C:\MyFolder\,MySubfolder)}</TEXT>
```

In Ausdrücken müssen Sonderzeichen wie „Komma“,

(aus dem ersten Tasterblock) durchlaufen werden soll (welcher Taster gedrückt wurde, steht in der Variablen ButtonPosition). Wenn man die Schaltfläche „Documentation“ betätigt, öffnet sich die Doxygen-Dokumentation im Internet Explorer. Hier kann man die Syntax der Library-Funktionen nachsehen.
Selbstverständlich könnte man die Textboxen auch leer lassen und ein EFL-Projekt generieren, das nur die nötigen Initialisierungen und Setup-Befehle für die Bibliotheken enthält. Den eigentlichen Applikationscode kann man ja auch direkt im Atmel Studio Editor nachtragen.
Doch gibt es einen Vorteil, wenn wir die Anwendung bereits im Configurator programmieren. Mit einem Klick auf „Save Settings“ können wir alle Einstellungen inklusive des Codes in einer kleinen textbasierten Datenbank abspeichern (siehe APP\DATA\tblProjects.txt). Der Datensatz wird jeweils unter dem Namen abgespeichert, den man für das Projekt vergeben hat. Über die Dropdownbox ganz oben kann man alle bereits abgespeicherten

„Klammer Auf“, „Klammer Zu“ usw. durch vordefinierte Konstanten ersetzt werden, die immer mit „@@“ beginnen: @@Komma, @@Open, @@Close. Auch Statusvariablen des Skript-Interpreters beginnen mit „@@“: Zum Beispiel enthält „@@DateToday“ immer das aktuelle Datum in der Form YYYY-MM-DD.

Der wichtigste (komplexe) Datentyp der Skriptsprache sind die namensgebenden Sheets (Tabellenblätter), die wiederum aus 1 bis 64 Spalten zusammengesetzt sind, die jeweils Daten verschiedenen Typs enthalten können. Jede Reihe des Sheets enthält einen Datensatz mit den Daten aus allen Spalten. Ein Sheet kann bis zu 1023 Reihen enthalten. Jede Spalte hat einen Namen, um innerhalb eines Datensatzes einen bestimmten Tabelleneintrag zu referenzieren.
Alle Sheets, die in einem Skript verwendet werden (bis zu 32) müssen zu Beginn mit Namen deklariert werden, wobei mehrere Einträge durch ein Semikolon getrennt werden:

```
<SHEETS>Files;Symbols;</SHEETS>
```

Mit dem Kommando SET kann man nicht nur wie oben gezeigt einfache Variablen, sondern auch ganze Sheets befüllen. Das Kommando

```
<SET Array='True' RName='Folder;File;' RS='Files' >
Common;BoardpinEFL.c;
Common;CommonEFL.h;
Libraries;LEDButtonEFL.h;
</SET>
```

legt im Sheet „Files“ zwei Spalten mit Namen „Folder“ und „File“ an und füllt die Tabelle mit den Werten zwischen den Tags. Auch hier sind Referenzen auf Variablen (@...), Statusvariablen (@@...) und Rechenausdrücke (@{…}) erlaubt.
Mit dem Ausdruck @*Sheetname*.*Spaltenname* hat man nun Zugriff auf den Inhalt einer bestimmten Spalte des aktuellen Datensatzes (Reihe) eines bestimmten Sheets. Auch der Ausdruck @*Spaltenname* genügt, wenn der Spaltenname lediglich in einem der Sheets verwendet wird. Nach Befüllen des Sheets ist immer der erste Datensatz aktuell. Mit dem Kommando FOREACHROW kann man das Sheet Reihe für Reihe durchlaufen, zum Beispiel das oben definierte Sheet „Files“. Im Inneren der Schleife hat man dann Zugriff auf die Daten in den einzelnen Spalten. Die Kommandos

```
<FOREACHROW Of='Files' >
<TEXT>@Files.File</TEXT>
<TEXT>;</TEXT>
</FOREACHROW>
```

führen zu folgendem Text im XMLResult:

```
BoardpinEFL.c;CommonEFL.h;LEDButtonEFL.h;
```

READ und WRITE sind die Befehle zum Laden und Speichern von Daten in einer Datenbank. Bei einer lokalen Anwendung wie dem EFL Configurator besteht die Datenbank aus Textdateien, die sich genauso wie die Skripts im APP-Ordner befinden, der Unterordner trägt den Namen der Datenbank (in unserem Beispiel einfach „DATA“). Jede Textdatei enthält eine Tabelle, die genauso wie ein Sheet in Spalten und Reihen organisiert ist.

Mit FILEREAD und FILEWRITE können Text- (und Binärdateien) eingelesen und generiert werden. FILERC (RC= Rename/Copy) verschiebt und kopiert Files bzw. benennt diese um. Mit MAKEFOLDER kann man einen Ordner anlegen, alle Elternordner werden automatisch miterzeugt. MESSAGE gibt eine Nachricht auf dem Bildschirm aus. Darüber hinaus gibt es viele weitere Kommandos. An einer ausführlichen Dokumentation wird gearbeitet.

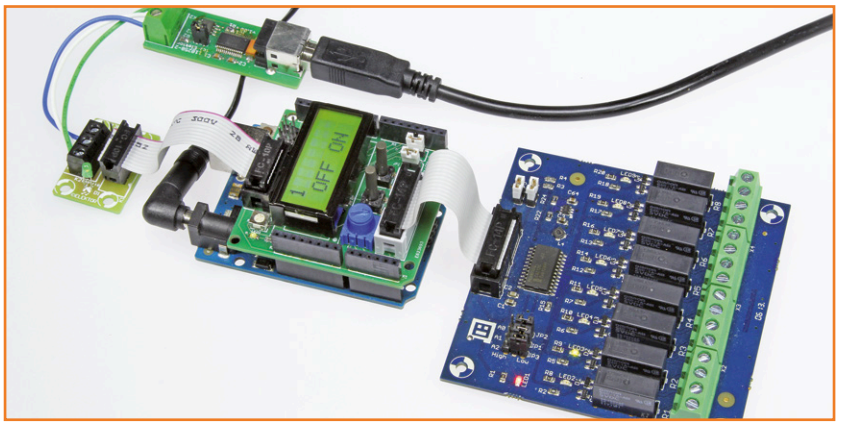


Bild 5.
Als erstes Beispiel generieren wir den Code für die Relais-Steuerung, die in der letzten Ausgabe beschrieben wurde [1].

Projekte anwählen und mit „Load Settings of“ wird das Projekt mit allen Einstellungen und dem Code zurück in den EFL Configurator geladen. Damit entsteht eine kleine EFL-Projekt-Verwaltung, die – und das könnte in der Zukunft noch wichtig werden – unabhängig von der verwendeten Entwicklungsumgebung ist. Aus den Einstellungen und dem Applikations-Code, der hier mit dem Ziel „Atmel Studio“ geschrieben und abgespeichert wurde, könnte ein ähnliches Skript später einmal zum Beispiel ein Eclipse-Projekt generieren. So können Programmierer nicht nur über die Grenzen von Controller und Boards, sondern auch über die Grenzen von Entwicklungsumgebungen hinaus zusammen Anwendungen erstellen.

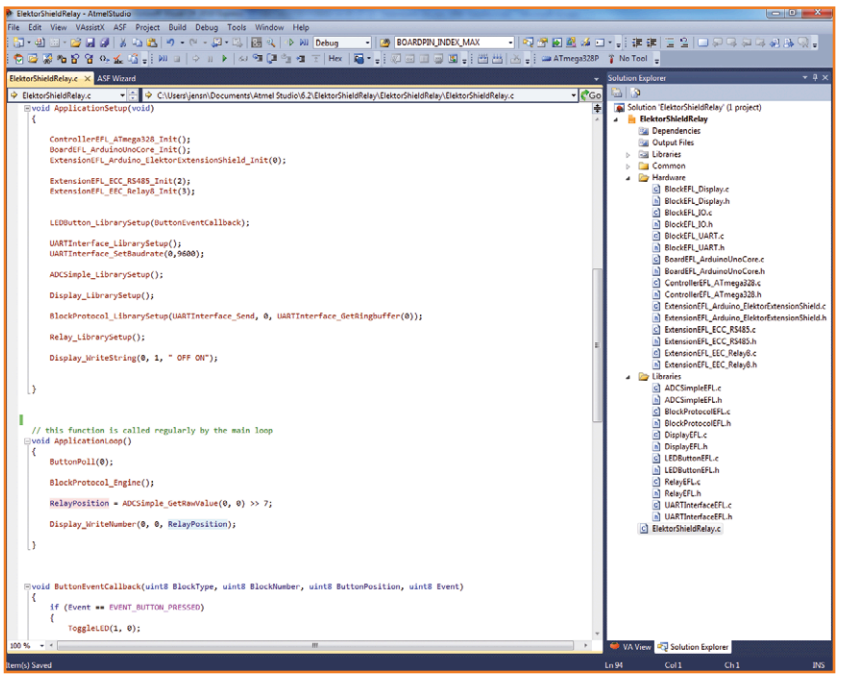


Bild 6.
Hurra, es hat geklappt: Das automatisch erzeugte Projekt in Atmel Studio.

Erster Test

Kein EFL-Artikel ohne Beispiele: Wir haben als Grundstock für die eigene Projekt-Verwaltung bereits die drei Anwendungen aus der letzten Ausgabe vorbereitet [1]. Als erstes Beispiel ist die Anwendung „ElektorShieldRelay“ voreingestellt (Bild 4). Das **Bild 5** zeigt die Hardware des Projekts, die im letzten Heft beschrieben wurde. Acht Relais lassen sich lokal mit den Tastern, dem Poti und dem Display des Extension Shields bedienen. Darüber hinaus auch vom PC aus per Terminalprogramm; als Protokoll wird das einfache BlockProtocol genutzt.

Wenn wir nun den Button „Create Project“ drücken, müsste das vollständige EFL-Projekt an der gewünschten Stelle generiert werden.
Mit einem Klick auf das neu erzeugte .atsln-File öffnen wir das Projekt in Atmel Studio (**Bild 6**). Im Solution Explorer zeigen sich alle benötigten Files in den drei EFL-Ordnern. Das Hauptfile des Quellcodes enthält alle #include-, alle Init- und alle Setup-Anweisungen für die Bibliotheken, plus den applikationsspezifischen Code.

Das Projekt müsste sich nun anstandslos kompilieren und in den Controller laden lassen.
Bei der Entwicklung ist es empfehlenswert, den EFL Configurator und Atmel Studio parallel geöffnet zu haben. Ändern Sie einmal den Code in einer der Textboxen des EFL Configurators und betätigen Sie erneut die Schaltfläche „Create Project“, ohne den Projektnamen geändert zu haben. Im bereits erzeugten Atmel-Studio-Projekt wird das Hauptfile des Codes dann durch ein neues File ersetzt und erscheint (nach Bestätigung in einer Dialogbox) geändert im Atmel Studio. Hier können wir den neuen Code kompilieren und testen.

Runter mit dem Speicherbedarf

Mit Hilfe der Projektverwaltung können wir uns nun ganz einfach verschiedene Versionen einer Anwendung generieren. Das nutzen wir, um unser EFL-Projekt ein wenig auf Speicherbedarf zu optimieren. Ganz unten im Konfigurator findet man einige Auswahlboxen, mit denen wir auf den RAM-Bedarf des Projekts Einfluss nehmen können. Das beginnt mit der Größe der EFL-Tabellen [5], nämlich der Map für Controller-Features wie ADC, UART und Co., der Boardpin-Tabelle für die Abbildung der Verdrahtung und der Block-Tabelle, die alle Peripherieblocks und Erweiterungssteckverbinder beschreibt.

Wenn wir die oben beschriebene Firmware in den Controller gebrannt haben, dann können wir über das BlockProtocol nicht nur die Relais steuern, sondern auch den Inhalt der momentan benutzten EFL-Tabellen in einem Terminalprogramm anzeigen lassen (Kommando x <CR>) [4]. Wir sehen, dass wir vier Einträge in die Map-Tabelle, rund 60 Boardpin-Einträge und zwölf Einträge in die Block-Tabelle benötigen (**Bild 7**). Im Moment wird aber Speicherplatz für acht Map-Einträge (je 4 Byte), 64 Boardpin-Einträge (2 Byte) und 16 Einträge (je 5 Byte) in der Block-Tabelle bereitgestellt. Dies können wir nun optimieren, in dem wir im EFL Configurator die Größe der Map-Tabelle auf 4 und die Größe der Block-Tabelle auf 12 einstellen. Außerdem ordnet der Controllercode einer Software-SPI-Schnittstelle noch einen Ringpuffer von immerhin 64 Bytes zu. Auch für die drei Timer wird eine relativ speicherhungrige Struktur angelegt. Desgleichen noch für den Scheduler, der timergesteuert Funktionen aufrufen kann (wurde bisher noch nicht beschrieben).

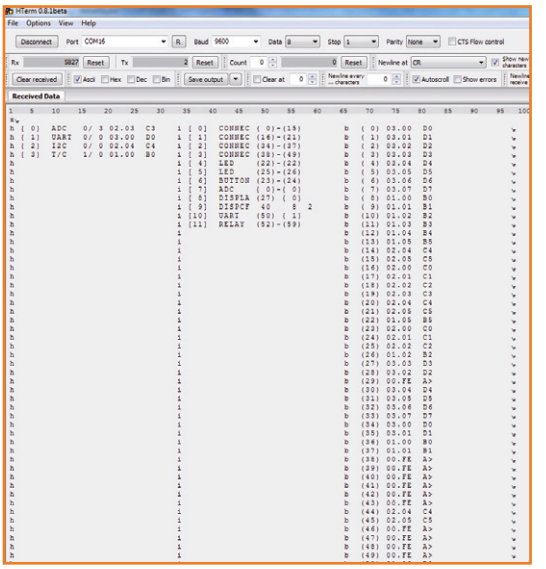


Bild 7.
Im Terminalprogramm sehen wir, wie groß die EFL-Tabellen für unsere Anwendung sein müssen. Nun können wir den RAM-Speicherbedarf optimieren, so dass die Firmware auch in kleineren Controllern läuft.

Wir optimieren das Ganze daher wie unten in **Bild 8** gezeigt, denn wir benötigen weder den Timer noch SPI.

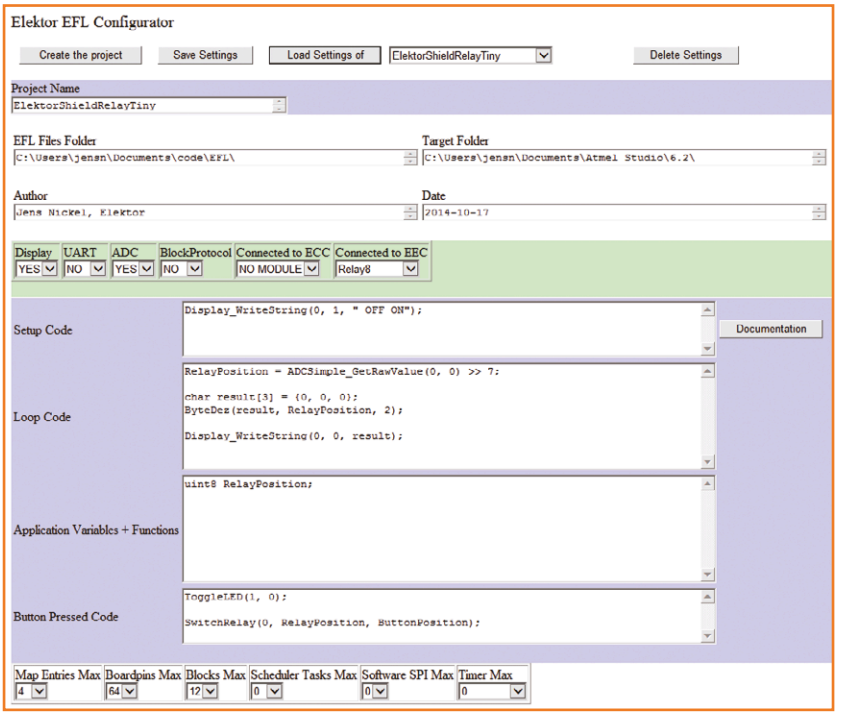


Bild 8.
Den Speicherbedarf für die Software-SPI-Schnittstelle und die Timer setzen wir auf Null (unten rechts). Denn beides benötigen wir nicht.

Wenn wir nun ein abgewandeltes Projekt erzeugen, dann werden in der .cproj-Datei entsprechende Symbole hinterlegt (die Symbole kann man sich in Atmel Studio anzeigen lassen, wenn man auf den Projektnamen rechtsklickt und dann Properties, Toolchain und Symbols auswählt). Beim Kompilieren des Projekts werden nun die optimierten Größen der EFL-Tabellen berücksichtigt. Das Ergebnis (im Out-Fenster) kann sich sehen lassen: Wir benötigen statt ursprünglich 1091 Bytes nur noch 923 Bytes RAM. Und das bei vollständig erhaltener Funktionalität des Programms. Da der Flashspeicher-Bedarf jeweils 16.260 Bytes beträgt, passt das Projekt nun bereits in einen ATmega168.
Wir können weiter abspecken, wenn wir auf die Fernsteuerung der Relais verzichten. Wir lassen das RS485-Modul weg und stellen zusätzlich noch die Dropdown-Boxen bei BlockProtocol und UART auf „NO“.

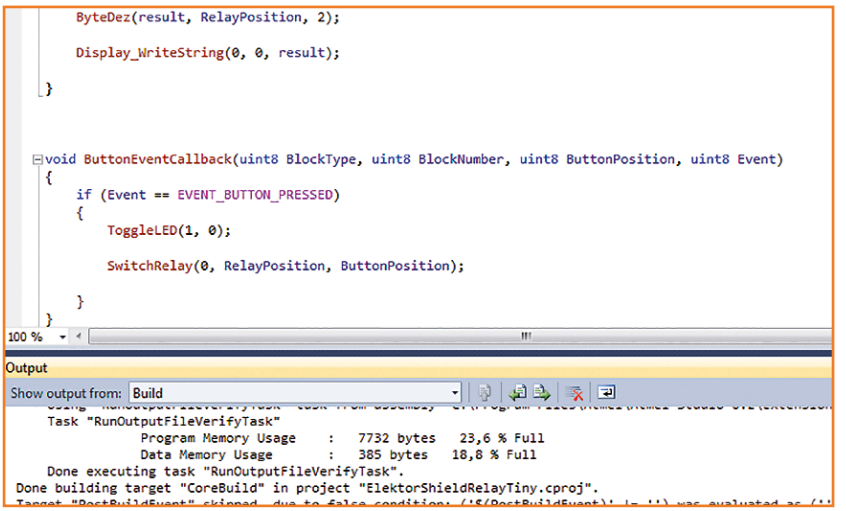


Bild 9.
Ohne den Fernzugriff passt der Code der Relais-Steuerung auch in einen ATmega88.

Nach dem Kompilieren des neu generierten Projekts stellen wir nur noch eine Codegröße von 9.072 Bytes und einen RAM-Speicherbedarf von 389 Bytes fest; besitzen aber immer noch eine schöne Relaissteuerung per Poti und Taster plus Rückmeldung auf dem Display. Leider liegen wir beim Flashbedarf noch über den magischen 8 K, mit der unser Projekt in einen ATmega88 (oder einen ATtiny85) passen würde. Doch findet man im Applikations-Code recht schnell einen absoluten Speicherfresser: Die Funktion `Display_WriteNumber(...)` greift in der DisplayEFL-Library ihrerseits auf die Funktion `sprintf(…)` zurück. Zur Darstellung der Ziffern „0“ bis „7“ brauchen wir eine derart mächtige Funktion nicht. Wir ersetzen deshalb Display_WriteNumber(...) durch die in Bild 8 in der Textbox „Loop Code“ gezeigten Zeilen und generieren das Projekt erneut aus dem EFL-Konfigurator heraus. Das Ergebnis sieht man in **Bild 9**. Das Projekt „ElektorShieldRelayTiny“ findet sich ebenfalls unter den mitgelieferten Projekt-Settings.

Den EFL Configurator werden wir weiter ausbauen, denn die Unterstützung weiterer Hardware und weitere EFL-Anwendungen sind schon in Vorbereitung.

Vielleicht haben wir Sie aber auch auf den Geschmack gebracht, Ihren eigenen Konfigurator für Atmel-Studio-Projekte oder andere Entwicklungsumgebungen zu entwickeln. Ihre Ideen nehmen wir gerne entgegen – auf elektor-labs.com oder per Mail an die Redaktion!

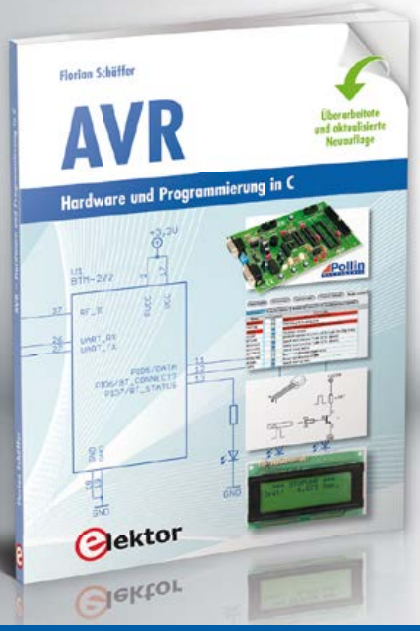
(140372)

Weblinks

[1] www.elektor-magazine.de/140328
[2] www.elektor-magazine.de/140372
[3] www.elektor-magazine.de/140009
[4] www.elektor-magazine.de/130154
[5] www.elektor-magazine.de/120668

AVR
Hardware und Programmierung in C

Überarbeitete und aktualisierte Neuauflage



Der Autor führt Einsteiger und auch Fortgeschrittene gekonnt und professionell in eine hochinteressante Thematik ein. Auch wer seine Elektronik- und Programmierkenntnisse weiter ausbauen und vertiefen möchte, hat dazu gute Möglichkeiten. Die modernen und zeitgemäßen Atmel AVR-Prozessoren sowie die Programmierung in C sind in Kombination eine zukunftssichere Plattform für lange Zeit. Nach Einführung und Vorstellung der notwendigen Entwicklungsumgebung werden Projekte vorgestellt, die schrittweise zum Ziel führen.
Für die meisten Projekte kommt das Atmel AVR-Evaluation-Board zum Einsatz – eine Experimentierplatine aus dem Hause Pollin Electronic. Das gewährleistet den reibungslosen Nachbau der vorgeschlagenen Projekte. Natürlich ist auch die Verwendung eigener Experimentierschaltungen möglich, denn ein erklärtes Ziel des Buches ist es, den Anwender zu selbständigem Arbeiten und Entwickeln zu befähigen.

Tipp: Unter www.elektor.de/avr-buch haben wir ein vorteilhaftes Buch/Board-Bundle für Sie geschnürt.

260 Seiten • Format 17 x 23,5 cm (kart.)
ISBN 978-3-89576-300-7

€ 34,80
CHF 43,95

Weitere Infos & Bestellung unter www.elektor.de/avr-buch

Learn to program
Spielerisch programmieren lernen – ohne Tastatur

NEU!

Learn to program ist ein einfaches Lernspiel für Kinder und Schüler, das im Stil der Anfangsjahre des Computerzeitalters das Programmieren mit nur vier Tasten erlaubt. Auf dem Spielbrett befinden sich alle nötigen Ein-/Ausgabeeinheiten wie etwa eine LED-Ampel, ein Helligkeitssensor und ein Piezopiepser. Damit lassen sich verschiedene Programme entwickeln, wie zum Beispiel eine einfache Eieruhr, ein Wecker (der morgens wie ein Hahn „Kikeriki" ruft) oder auch einfache Reaktionsspiele. Die Batterieversorgung macht die Programmierung netzunabhängig.

Für viele Elektroniker ist es selbstverständlich, Programme zu schreiben, einfache Regelungen zu implementieren oder Steuerungen zu realisieren. Nur ist einem dabei oft nicht klar, was für ein langer Weg es war, um soweit zu kommen.
Mit diesem edukativen Spiel kann man das Mysterium des Programmierens Kindern anschaulich erklären.

Eigenschaften des Boards:
• 16 verschiedene Befehle • 8 Eingabetaster • Tonausgabe (Piezopiepser) • LED-Ampel • 4 weitere LEDs • Lichtsensor

Inhalt der Lernbox:
• Bestückte Mikrocontrollerplatine (Spielbrett)
• 73-seitiges A6-Handbuch

€ 44,95
CHF 56,95

Weitere Infos & Bestellung unter www.elektor.de/learn-to-program

Red Pitaya

Mehr als ein USB-Oszilloskop!

Von **Martin Oßmann** (D)

Mit Red Pitaya gibt es jetzt eine leistungsfähige Open-Source-Messplattform, die man für verschiedene Messaufgaben konfigurieren kann. Es gibt fertige Applikationen (Apps) im Internet, die man einfach herunterladen kann. Man kann aber auch selbst spezielle Messvorgänge programmieren. Dieser Artikel beschreibt einige Messmöglichkeiten und zeigt Programmierbeispiele.

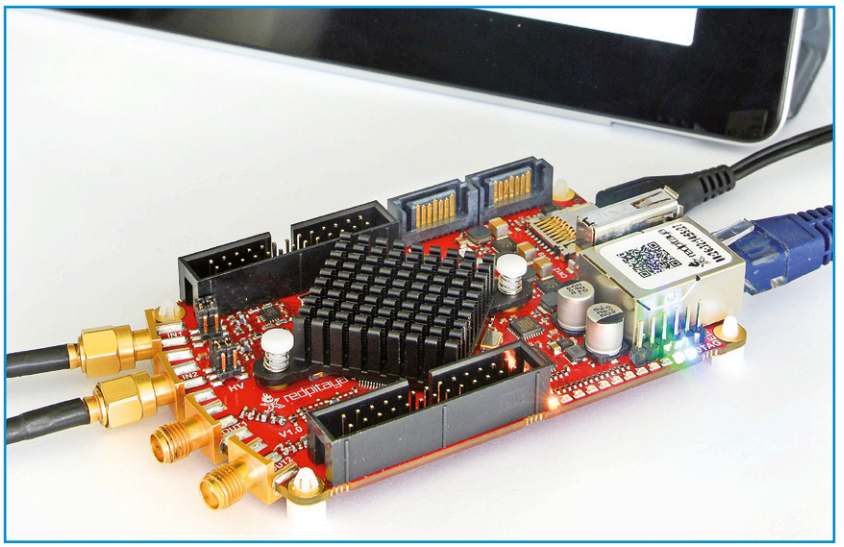


Bild 1.
So kompakt fällt das Red-Pitaya-Modul aus.

Die sich bietenden Möglichkeiten des Boards (**Bild 1**) kann man gut abschätzen, wenn man sich das Blockschaltbild in **Bild 2** anschaut. Zentraler Chip ist ein Xilinx-SoC (**S**ystem-**o**n-a-**C**hip)

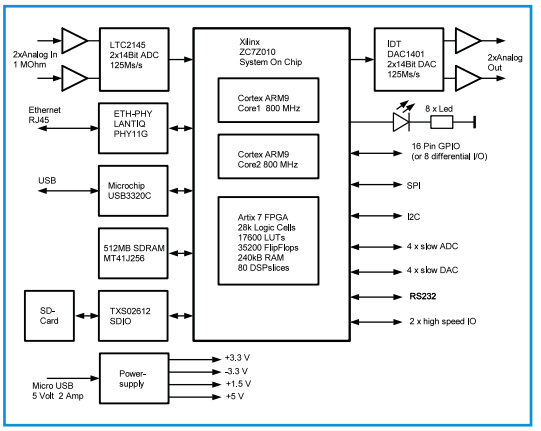


Bild 2.
Das Blockschaltbild des Red-Pitaya-Boards.

des Typs ZC7Z010. Er enthält eine Dual-Core-ARM9-CPU mit 800 MHz Takt, auf welcher bei RP (**R**ed **P**itaya) ein Linux-System läuft. Weiter verfügt der Chip über ein FPGA mit 28 K Logik-Zellen, einer Menge BRAM (Block RAM) und sogenannten DSP-Slices. Mit diesem FPGA kann man extrem schnelle Funktionen realisieren, wie sie z.B. bei der digitalen Signalverarbeitung anfallen. Um den SOC herum gruppiert sich neben 512 MB RAM allerhand Peripherie.

Für Messaufgaben gibt es zwei A/D- und D/A-Wandler mit 14 bit, die eine Samplerate von bis zu 125 MS/s erreichen. Mit diesen Wandlern kann man schon recht anspruchsvolle Aufgaben erledigen. Mit dem FPGA hat man auch die dazu notwendige Verarbeitungsgeschwindigkeit in petto. Das System bietet einen Ethernet-Anschluss, einen USB-Host-Port und einen USB-COM-Port, um auf einfache Weise mit einem PC über eine serielle Schnittstelle kommunizieren zu können. Hinzu kommt noch ein SD-Karten-Slot. Die SD-Karte wird von Linux als Festplattenersatz benutzt. Über Erweiterungssteckplätze sind noch General-Purpose-IOs, I^2C, RS232 und weitere, relativ langsame A/D- und D/A-Wandler zugänglich.

Es ist also alles vorhanden, was man so braucht, wenn man anspruchsvolle Messungen und Hochleistungs-Signalverarbeitung auch bei höheren Frequenzen durchführen will. Genau dafür ist RP gedacht: Es ist eine universelle Messplattform mit einer Bandbreite bis zu 50 MHz.

Inbetriebnahme

Die erste Inbetriebnahme gestaltet sich problemlos, weil es eine gute Anleitung im Internet gibt. Man kopiert einfach den für die SD-Karte

gedachten Inhalt auf eine Mikro-SD-Karte und steckt diese in den Karten-Slot von RP. Anschließend verbindet man RP mit dem eigenen LAN und schaltet den Strom ein. Dann bootet RP und es kann direkt losgehen.
Der Autor verfolgt normalerweise den Bootvorgang auf seinem PC über den USB-COM-Port, denn so kann man gut kontrollieren, welche IP-Nummer RP erhält. Nach dem Booten kann man über den USB-COM-Port direkt mit Linux kommunizieren, oder man verwendet das Ethernet zur weiteren Kommunikation. Letzteres ist die favorisierte Methode.

Messgeräte vom Bazaar

Eine der Hauptintentionen beim RP-Projekt war es, fertige Applikationen als Web-Anwendungen bereitzustellen. Diese findet man im sogenannten „Bazaar“. Wenn man per Webbrowser den Webserver für RP kontaktiert, wird (sozusagen auf der Haupt-Webseite) der Bazaar geöffnet und man kann die gewünschten Applikationen auswählen. Zurzeit sind die in **Tabelle 1** aufgelisteten Apps verfügbar.
Es ist damit zu rechnen, dass hier in Zukunft noch viele interessante Apps für diverse Messaufgaben auftauchen werden. Die Projekte sind alle Open Source, so dass man sie auch gut als Startpunkt für eigene Weiterentwicklungen nutzen kann.
Als Einstieg dienen das Oszilloskop und der Spectrum-Analyzer. Damit wird die in **Bild 3** theoretisch und in **Bild 4** praktisch gezeigte Schaltung vermessen.
Die Eingänge des RP-A/D-Wandlers sind dank Opamps mit 1 MΩ so hochohmig, dass man direkt normale Oszilloskop-Tastköpfe anschließen kann. Man kann zwei Verstärkungen für die Messbereiche 2 V_{SS} und 20 V_{SS} wählen. Wenn man die Tastköpfe an TP1 und TP2 von Bild 3 anschließt und die RP-Scope-App startet, erhält man die Darstellung von **Bild 5**. Es sind alle üblichen Scope-Parameter einstellbar.
Startet man die Spectrum-Analyzer-App, erscheint eine Anzeige wie in **Bild 6**. Man kann die beiden Spektren bequem live verfolgen und vergleichen. Man kann in Bild 6 gut die Oberschwingungen des Quarzoszillators erkennen, wobei der Gatterausgang TP1 stärkere Oberwellen liefert als TP2.
Nachdem dem ersten Eindruck dessen, was man mit der RP-Plattform realisieren kann, geht es nachfolgend um das Programmier-Know-How für eine derartige App (**Bild 7**).

Tabelle 1. Derzeitige Apps im RP Bazaar.

- 2-Kanal-Signalgenerator 0...50 MHz
- 2-Kanal-Oszilloskop, 2x 125 MS/s
- 2-Kanal-Spectrum-Analyzer
- Frequency-Response-Monitor
- PID-Controller

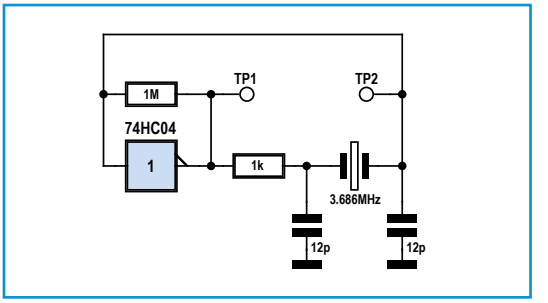


Bild 3.
Schaltung eines Quarzoszillators.

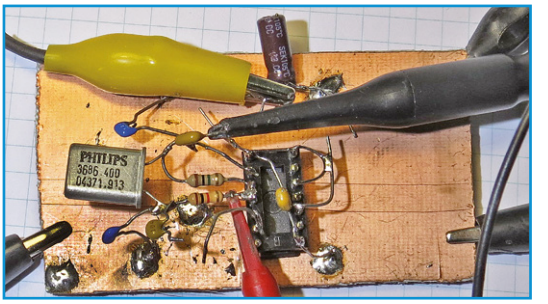


Bild 4.
Testaufbau des Quarzoszillators.

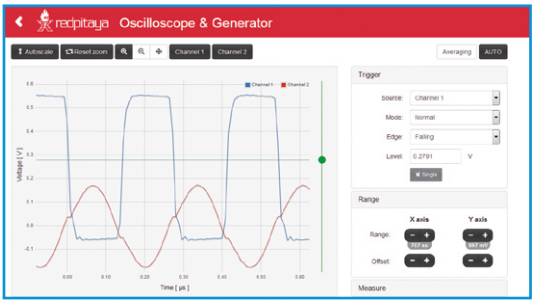


Bild 5.
RP-Scope-App in Aktion.

Bild 6.
RP-Spectrum-Analyzer-App in Aktion.

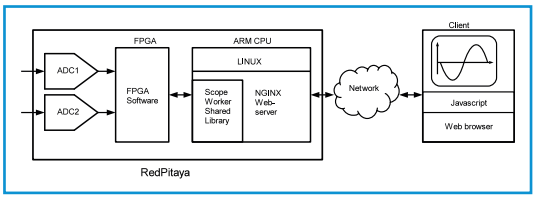


Bild 7.
Beteiligte Komponenten bei RP-Web-Mess-Apps.

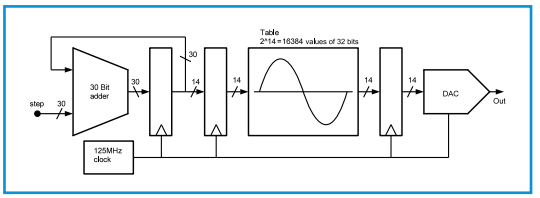


Bild 8.
Prinzipschaltbild des Signalgenerators.

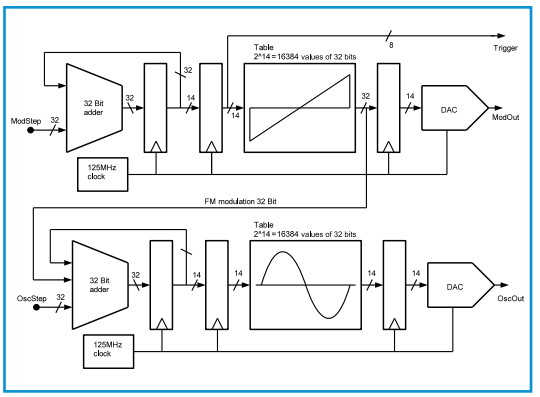


Bild 9.
Prinzip des FM-Generators aus zwei DDS-Baugruppen.

Webservice Messgeräte

Die A/D- bzw. D/A-Wandler werden normalerweise vom FPGA bedient. Hier findet ggf. auch eine Vorverarbeitung statt wie etwa die Speicherung von Frames im FPGA-eigenen schnellen Speicher. Für diese Aufgabe muss man daher entsprechenden FPGA-Code programmieren.

Das FPGA kommuniziert üblicherweise mit einem Programm auf der ARM9-CPU. Dieses Programm übernimmt normalerweise die etwas langsameren aber komplexeren Steuerungsaufgaben. Für neue Anwendungen muss natürlich auch dieser Code programmiert werden. Bei einer Web-App ist dieser Code als „Shared Library“ in den Linux-Webserver eingebunden. Der Webserver selbst vermittelt die Daten zwischen dem Webbrowser auf dem Client und dem ARM-Programm. Das GUI (**G**raphical **U**ser **I**nterface) im Browser selbst wird z.B. mittels JavaScript realisiert. Auch dieser Code muss für eine neue App erstellt werden. Wer also komplexe Mess-Apps selbst erstellen will, ist fachlich erheblich gefordert. Vorteil ist aber, dass alle bisherigen RP-Apps Open Source sind, so dass man genügend Ausgangsmaterial hat, von dem aus beginnend man sich mit geringen Modifikationen vortasten kann. Jetzt geht es um mögliche erste Schritte.

Programmierung der ARM-CPU

Als Erstes wird demonstriert, was man durch einfache Programmierung der ARM-CPU unter Benutzung des vorliegenden FPGA-Codes machen kann. Im Standard-FPGA-Code ist auch die Funktionalität für einen zweikanaligen Signalgenerator enthalten. Die Wellenform bezieht jeder Kanal aus einem Speicher mit 16.384 Samples. Dieser Speicher wird genauso abgetastet wie bei einem DDS-Generator (**D**irect **D**igital **S**ynthesis). Die Samples werden mit den 14-bit-D/A-Wandlern ausgegeben. Die Arbeitsfrequenz (Abtastrate) liegt bei 125 MHz. **Bild 8** zeigt die Prinzipschaltung für einen Kanal.
Die Parameter stehen in Registern des FPGA, die von der CPU aus unter festgelegten Adressen ausgelesen und beschrieben werden können. Dies erledigt ein C-Programm.

FM-RTTY-Generator

Die Frequenz des Generators soll softwaregesteuert zwischen zwei Werten umgeschaltet werden. Dazu wird die Schrittweite (step) des DDS-Generators geändert. Damit ergibt sich ein FM-RTTY-Generator. Zwischen den Änderungen wird jeweils 20.000 µs gewartet, was eine Baudrate von 50 bit/s ergibt. Die beiden Sendefrequenzen sind frei zwischen 0 und 50 MHz einstellbar. In der Beispiel-Software werden die zu sendenden Bits vorab im Baudot-Code berechnet und dann in einer Schleife abgespielt. Diese Schleife ist in **Listing 1** dargestellt.

Indem man einfach auf die Variable „chb-countstep“ zugreift, kann man direkt das FPGA-Register ändern. Das Beispiel zeigt, dass man schon mit einfachen C-Programmen recht interessante Anwendungen mit dem RP realisieren kann. Die C-Programme muss man natürlich mit einem ARM9-kompatiblen Compiler generieren. Der Autor verwendet dazu den ARM9-GCC-Compiler unter LUBUNTU in einer VM unter VirtualBox.
Inzwischen ist auch gut beschrieben, wie man unter Windows geeignete C-Programme für RP erstellen kann. Die Programme überträgt man dann z.B. mit SCP/WINSCP vom PC auf RP und kann sie dort ausführen.

FPGA-Programmierung

Nächster Schritt ist ein Beispiel zur FPGA-Programmierung. Um FPGA-Software zu erstellen verwendet der Autor „Vivado“ von Xilinx in einer freien WebPACK-Edition. Der Standard-RP-FPGA-Code liegt auch als Vivado-Projekt vor, so dass man davon ausgehend durch Modifikation gut eigene Designs entwickeln kann. Insbesondere das Interface zur ARM-CPU und der anderen Hardware braucht man dann nicht selbst neu zu erfinden.
Hilfreich ist ebenfalls, dass man FPGA-Register und -Signale als Variablen im ARM-CPU-Speicher sichtbar machen kann. Das erleichtert das Debugging ungemein, weil man live in das FPGA „hineinsehen“ kann. Doch nun zur eigentlichen Anwendung:

FM-Generator

Im Standard-FPGA ist ja ein zweikanaliger Signalgenerator enthalten, wobei jeder Kanal wie in Bild 8 aufgebaut ist. Das Ganze wird nun so modifiziert, dass der eine Kanal den anderen Kanal frequenzmoduliert. Da die modulierende Wellenform ja selbst im Speicher steht, kann man einen Sweep-Generator programmieren, indem man linear moduliert.

Bild 9 zeigt die Prinzipschaltung des FM-Generators. Die obere Gruppe ist der Modulator-DDS. Die Tabellenwerte werden mit 32 bit Breite gespeichert, um die Frequenz fein genug modulieren zu können. Die Tabellenwerte gelangen zur Kontrolle auf den zweiten DA-Wandler. Entscheidend für die Modulation ist, dass diese in den Addierer der zweiten DDS-Gruppe (unten) gelangen und diesen DDS (Oszillatorteil) damit FM-modulieren. Damit im Sweep-Betrieb ein Oszilloskop getriggert werden kann, wird aus dem oberen Teil ein passendes Triggersignal abgezweigt. Dieses gelangt auf GPIO-Pins und auch auf ein per Software lesbares Register.

Der wesentliche Code des Modulators von **Listing 2** besteht nur aus wenigen Deklarationen und einem Codeblock mit sechs Zuweisungen.
Erzeugt man mit diesem FM-Generator ein 10-MHz-Signal, das mit einem 1-kHz-Sinussignal in einer bestimmten Modulationsstärke moduliert wird, entsteht das in **Bild 10** dargestellte Spektrum. Der Hub ist so eingestellt, dass die Frequenzen 10 MHz ± 2 kHz fehlen.
Der Verilog-Code des FM-modulierten Generators sieht ganz ähnlich wie der Code des Modulators aus (**Listing 3**). Ein wichtiger Unterschied ist die Berechnung der neuen Position: Hier wird zusätzlich das Modulationssignal „fmInput-i“ addiert.
Wenn man den Generator so parametriert, dass er zwischen 9,7 MHz und 11,7 MHz sweept,

Listing 1: RTTY-Sendeschleife

```
uint32_t delay1=20000 ; // 50 Bit/sec
while(1){
  for(int k=0 ; k<nBits ; k++){
    int theBit=bitBuffer[k] ;
    if(theBit){ g_awg_reg->chb_count_step = step2 ; }
          else{ g_awg_reg->chb_count_step = step1 ; }
    usleep(delay1) ;
    }
  }
```

Listing 2: Verilog-Code des Modulator-DDS

```
reg [ 32-1: 0] dac_buf [0:(1<<14)-1] ; // data buffer
16384x32 Bit
reg [ 32-1: 0] dac_rd    ; // DAC value
reg [ 14-1: 0] dac_rp    ; // DAC read pointer
reg [ 32-1: 0] dac_pnt   ; // read pointer

always @(posedge dac_clk_i) begin
   dac_rp   <= dac_pnt[14-1+18:0+18];
   dac_rd   <= dac_buf[dac_rp] ;     // read data value
   signal_o <= dac_rd ;
   dac_o <=dac_rd[14-1+12:0+12] ;    // feed to output
   dac_pnt <= dac_pnt + set_step_i ; // get new position
   trigSignal_o <= dac_rp[8-1+6:0+6] ;
end
```

Listing 3: Verilog-Code des Oszillator-DDS

```
reg  [ 14-1: 0] dac_buf [0:(1<<14)-1] ;
reg  [ 14-1: 0] dac_rd ;
reg  [ 14-1: 0] dac_rp ;
reg  [ 32-1: 0] dac_pnt ; // read pointer

always @(posedge dac_clk_i) begin
  dac_rp    <= dac_pnt[14-1+18:0+18];
  dac_rd    <= dac_buf[dac_rp] ;
  dac_o <= dac_rd[13:0] ;
  dac_pnt <= dac_pnt + set_step_i + fmInput_i ;
end
```

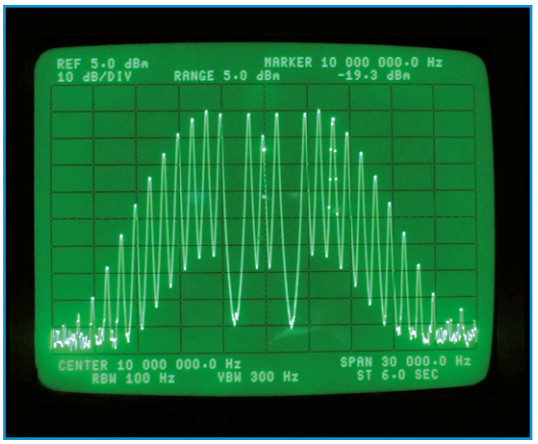


Bild 10.
Spektrum eines FM-Signals.

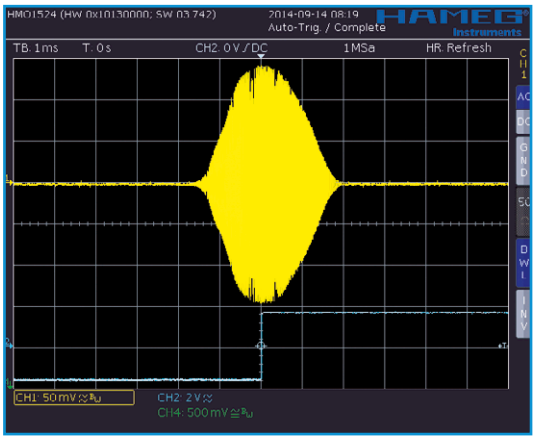


Bild 11.
FM-Sweep eines 10,7-MHz-ZF-Filters.

ergibt sich das Ergebnis von **Bild 11**. Es wird ein 10,7-MHz-UKW-ZF-Filter vermessen. Die blaue Kurve unten zeigt das Triggersignal.

Weitere Projekte

Schon mit einfachen FPGA- und C-Programmen kann man mit RP sinn- und anspruchsvolle Mess- und Testgeräte selbst realisieren. Abschließend noch ein paar kurze Hinweise zu Projekten, die ebenfalls schon realisiert wurden:

AM-Generator

Statt des vorgestellten FM-Generators kann man auch einen AM-Generator realisieren. Mit Hilfe der A/D-Wandler kann man eine Modulation mit externen Signalen ermöglichen.

RMS-Voltmeter

Die Eingänge werden mit 125 MS/s abgetastet. Daraus wird der RMS-Wert berechnet und angezeigt. Auf diese Weise kann man zweikanalig RMS-Werte mit einer Bandbreite von bis zu 50 MHz messen.

Gain/Phase/Impedance-Analyzer

Um den Frequenz- und Phasengang von Verstärkern, Filtern etc. im Bereich von 1 kHz bis 50 MHz zu vermessen, wird ein Sinussignal erzeugt und dann dessen Betrag und Phase am Ein- und Ausgang gemessen. Um Impedanzen zu bestimmen werden Strom und Spannung gemessen und daraus die Impedanz berechnet. Diese Anwendung ist im RP-Blog beschrieben und die Software im RP-GitHub verfügbar.

AM-Empfänger von 0 bis 50 MHz

Das Signal einer Drahtantenne gelangt direkt an einen A/D-Wandler. Es wird mit einem Oszillatorsignal auf der Empfangsfrequenz gemischt (125 MS/s Samplerate). Die Quadraturkomponenten werden tiefpassgefiltert und bis auf 120 kS/s reduziert. Diese Signale gelangen dann per FIFO in die ARM-CPU zur Demodulation. Das demodulierte Signal wird über einen D/A-Wandler ausgegeben. So entsteht ein einfacher AM-SDR-Empfänger.

SSB-Generator

Das Sprachsignal wird per A/D-Wandler gesampelt und von der ARM-CPU mit zwei Filtern in ein Hilbert-I/Q-Paar konvertiert. Diese I/Q-Signale gelangen mit 120 kS/s ins FPGA. Dort werden sie interpoliert, gefiltert und modulieren dann Sinus und Cosinus des Trägersignals. So kann man einen SSB-Sender von 0...50 MHz aufbauen.

Ausblick

Man kann sicher sein, dass die äußerst leistungsfähige Hardware von Red Pitaya noch für viele andere interessante Aufgaben eingesetzt wird. Mit der Zeit werden vermutlich immer mehr „Geräte“ als Open-Source-Projekte entstehen, wodurch sich dann auch der Anschaffungspreis amortisiert.

(140277)

Professionelle Hard- & Software zum Sonderpreis!

Exklusiv für Studenten!

Als neuer Vertriebspartner von National Instruments bietet Elektor ab sofort die Produkte der NI-Plattform für Ausbildung und Lehre für Studenten und schulische Einrichtungen an. Diese edukative Plattform vereint Hardware, Software und Unterrichtsmaterial, um Schülern und Studenten ein attraktives und inspirierendes Lernumfeld zu ermöglichen.

LabVIEW

Mit der Systemdesignsoftware *LabVIEW* können Studenten praxisorientiert anhand von Projekten und Systemen in einer einzigen Umgebung lernen und sich so Fähigkeiten und Verfahrensweisen aneignen, die im späteren Berufsleben unschätzbar sind.

NATIONAL INSTRUMENTS LabVIEW

Circuit Design Suite

Die *Circuit Design Suite* umfasst *Multisim* und *Ultiboard* und ist eine vollständige Plattform für Entwurf, Simulation und Validierung von Schaltplänen sowie den Leiterplattenentwurf. Die Suite verfügt über Funktionen, die speziell auf die Anforderungen von Studenten zugeschnitten sind, die bei der Entwicklung von elektronischen Konzepten hilfreich sind.

myDAQ

Bei *myDAQ* handelt es sich um ein kostengünstiges Datenerfassungsgerät, das überall und jederzeit Messungen und Analysen physikalischer Signale ermöglicht. *myDAQ* ist kompakt und portabel, sodass Studenten auch außerhalb des Labors und unter Einsatz branchenüblicher Werkzeuge und Methoden praktische Erfahrungen sammeln können.

Studentenversionen:

- Der Preis von ***LabVIEW*** und ***Circuit Design Suite*** beträgt jeweils nur 23,95 Euro.
- Das ***myDAQ Education-Kit*** bestehend aus *myDAQ* + 3 Software-DVDs (*LabVIEW, Circuit Design Suite* und *DIAdem*) kostet nur 174,95 Euro.

Jetzt bestellen unter www.elektor.de/ni-plattform!

Schalten via Tablet
Mit E-blocks und Flowcode

Von Bert van Dam (NL)

Über einen Tablet-Computer drahtlos LED-Leisten oder andere Objekte schalten, das muss kein schwieriges Unterfangen sein. Wie sich dies unkompliziert mit E-Blocks und Flowcode bewerkstelligen lässt, erklärt dieser Beitrag.

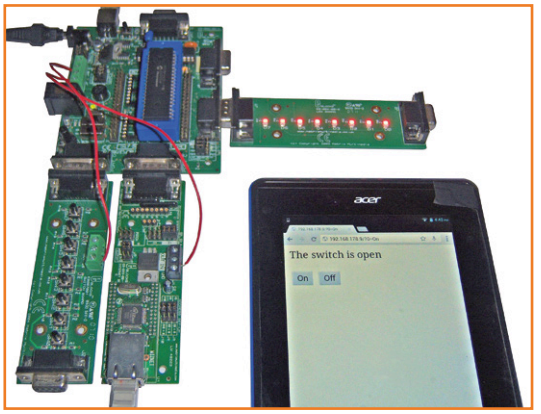


Bild 1.
Hier wird ein LED-Strip über den Tablet-Computer geschaltet.

Kleine und auch große Projekte lassen sich vorteilhaft mit E-blocks realisieren, diese werden in Flowcode grafisch programmiert. Für unser Vorhaben brauchen wir folgende Komponenten: Flowcode in der Version 6 für PIC, PIC-Multiprogrammer EB006 mit Mikrocontroller PIC18F4620, LED-Board EB004, Drucktaster-Board EB007, Internet-Board EB023 sowie ein kurzes Ethernet-Kabel. Mit den genannten E-blocks bauen wir ein lokales Internet, ein sogenanntes Intranet, auf.

Intranet

Zuerst verbinden wir das Internet-Board EB023 über das Ethernet-Kabel mit dem lokalen Router. Am Router angeschlossene Geräte fordern entweder selbstständig eine IP-Adresse an, oder wir müssen ihnen, wie dem E-block EB023, eine IP-Adresse zuweisen. Die IP-Adresse ist hier eine statische Adresse, da sie stets unverändert bleibt. Um eine IP-Adresse zuweisen zu können, rufen wir über den Webbrowser den Router auf und stellen fest, welche Adressen für das Intranet verfügbar sind. Die ersten drei Zifferngruppen, beispielsweise 192.168.178, sind im Router hinterlegt, während die vierte (ebenfalls aus drei Stellen bestehende) Zifferngruppe wählbar ist. Belegt werden mindestens zwei Adressen, denn eine Adresse gehört bereits zum Router. Die Anzahl der möglichen Adressen hängt vom Router ab, beim Router des Autors war sie einstellbar. Falls wir mehrere statische IP-Adressen vergeben, notieren wir die Adressen, so dass eine Adresse nicht mehrfach zugewiesen werden kann. **Tabelle 1** ist ein Beispiel für eine lokale Adressenliste.
Tablet-Computer erfordern keine weiteren Schritte, sie sind über das WLAN mit dem Router verbunden. Wenn kein Tablet vorhanden ist, kann ein Smartphone oder ein Desktop-PC die Aufgabe übernehmen.

E-blocks

Die Hardware unseres Projekts ist blockschematisch in **Bild 1** skizziert. Das Zusammenstellen und Konfigurieren der E-blocks ist spielend einfach:

- **Programmer**
  EB006, Schalter auf XTAL und Fast, Jumper LVP auf I/O-Port, J29 auf PSU, J12-14 auf USB, externe Stromversorgung. Der Mikrocontroller ist ein PIC18F4620, nicht der Stan-

Tabelle 1. Statische IP-Adressen im Router des Autors.

IP-Adresse	Apparatuur
192.168.178.9	Flowcode EB023
192.168.178.8	ARM mbed
192.168.178.7	Raspberry Pi über Wi-Fi

Tabelle 2. IP-Adressen im Webserver-Modul.

IP-Adresse	Item
192.168.178.1	Gateway (Router)
255.255.255.0	Subnet-Maske
192.168.178.9	EB023

dard-PIC, dieser hat nicht genügend Speicher.

- **Port B**
  EB004, LED-Board.
- **Port C**
  EB023, Internet Board. Jumper auf A, +5 V, Adresse auf 1-1-1, Netzwerkkabel zum Router, Kabel für Stromversorgung zum Programmer.
- **Port D**
  EB007, Drucktaster-Board. Kabel für Stromversorgung zum Programmer.

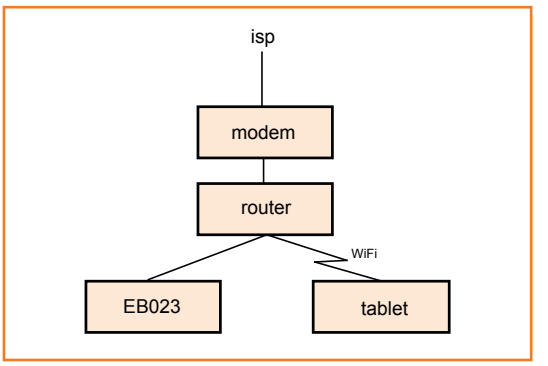


Bild 2.
Vom Internet-Board EB023 führt ein Kabel zum Router, der Tablet-Computer ist drahtlos angebunden.

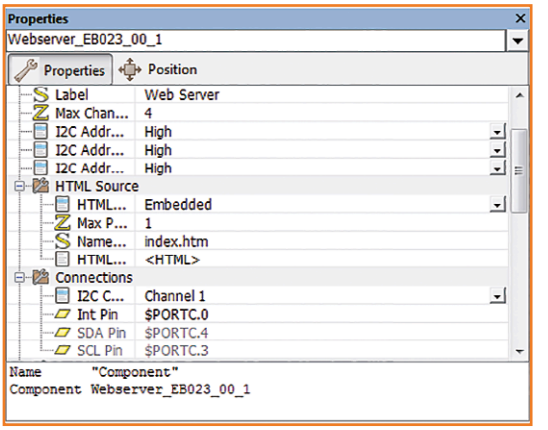


Bild 3.
Einstellungen des I²C-Interfaces beim Webserver EB023.

Software

Das Programm wurde in Flowcode 6 geschrieben. E-block EB023, das Internet-Board, ist an den Mikrocontroller über einen I²C-Bus angebunden. Im Programm wird die Komponente „Webserver EB023“ verwendet, dadurch werden die Einstellungen übernommen (**Bild 3**). Die mit den Jumpern einzustellende I²C-Adresse lautet 1-1-1. Außerdem müssen wir die IP-Adresse eintragen, wie sie im Router festgelegt ist (siehe **Tabelle 2**). Als MAC-Adresse kann die Vorgabe des Webservers übernommen werden.

Der nächste Schritt ist das Erstellen der Seite, die auf dem Tablet-Display bei der Verbindung mit dem EB023 erscheint. Das zu schaltende Objekt, hier der LED-Strip, soll mit zwei Schaltflächen bedient werden. Gleichzeitig soll das Display Auskunft über den Schaltzustand geben.

Wenn ein Benutzer die IP-Adresse des E-Blocks EB023 aufruft, gibt der Webserver eine Seite mit zwei Schaltflächen aus. Ruft sich diese Seite beim Betätigen einer Schaltfläche selbst auf (mit „_self“), folgen auf die IP-Adresse der Name sowie der Wert der Schaltfläche. Die erste Schaltfläche erhält den Namen „0“, ihr wird der Wert „On“ zugewiesen. Mit der Erweiterung „?0=On“ lautet die vollständige, vom Webserver übergebene Adresse „192.168.178/?0=On“. Über die Anweisung *GetInValue(0)* im Flowcode-Programm gibt der Webserver den Wert als String zurück, der bei der Variablen mit dem Index „0“ in der Adresse steht. In diesem Fall ist dies der Wert „On“. Dieser Wert schaltet das Objekt, beispielsweise den LED-Strip ein.

Da die Schaltflächen auf der Webseite keine weiteren Funktionen haben, kann auch die zweite Schaltfläche den Namen „0“ erhalten, diesmal jedoch mit dem Wert „Off“. Dieser Wert schaltet das Objekt wieder aus.

Tablet-Computer stellen Texte meistens sehr klein dar. Deshalb vergrößern wir mit „width:100px;height:50px;“ die Schaltflächen, und die Zeichenhöhe legen wir mit „font-size:18pt;“ auf 18pt fest.

Ein variabler Wert, in diesem Fall der Schaltzustand, wird zum Tablet-Computer gesendet, indem im HTML-Text das Zeichen „%“, gefolgt von einem Index eingefügt wird. Mit einem Makro wird der Variablen „%1“ der tatsächliche Wert zugewiesen.

```
<HTML>
<FORM>
<P></P>
<P STYLE="font-size: 18pt;">The switch is
   %1</P>
<P><INPUT NAME="0" TYPE="submit" STYLE
   ="width:100px;height:50px;font-size:
   18pt" VALUE="On" onClick="window.
   open('','_self')">
<INPUT NAME="0" TYPE="submit" STYLE="
   width:100px;height:50px;font-size:
   18pt" VALUE="Off" onClick="window.
   open('','_self')">
</P>
</FORM>
</HTML>
```

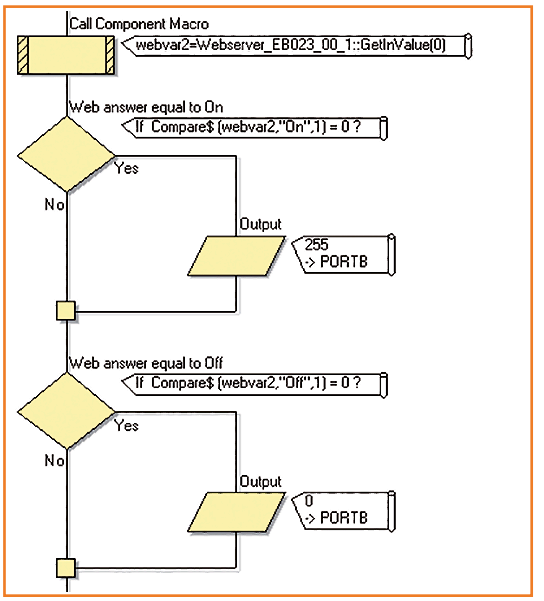


Bild 4.
Flussdiagramm für den Daten-Empfang in Flowcode.

Den vorstehenden Code tragen wir als *HTML Source* im Feld *HTML* ein (Bild 3). Um Platz zu sparen, werden hier die HTML-Konventionen nicht streng beachtet, es fehlen der *Header* und der *Body*. Mit der Code-Verkürzung kommen alle aktuellen Browser zurecht.

Beim Programmstart wird ein so genannter *Socket* geöffnet. Ein Socket ist mit einem Telefon vergleichbar. Beide Teilnehmer müssen im Besitz eines Telefons sein, um miteinander sprechen zu können. Außerdem müssen die Telefone miteinander verbunden sein. Der EB023 öffnet einen Socket und wartet auf einen Anruf. Sobald der Tablet-Computer eine Verbindung herstellt, sind die Sockets des EB023 und des Tablets miteinander verbunden. Das Makro *CheckSocketActivity* prüft in der Hauptschleife des Programms, ob auf den Sockets Aktivitäten stattfinden. Die weitere Behandlung läuft automatisch ab.

Mit einem *GetInValue(0)*-Makro wird geprüft, ob das Tablet neue Daten gesendet hat (Index „0“), und mit einem *SetOutValue(1,stand)* wird der Platzhalter „%1“ durch den tatsächlichen Stand des Schalters ersetzt.

Bedienung

Warten Sie, bis das Programm gestartet ist und geben Sie dem E-block EB023 und dem Router einige Sekunden Zeit, um die Verbindung herzustellen. Rufen Sie im Browser Ihres Tablets die IP-Adresse des EB023 auf, beim Musteraufbau war dies die Adresse 192.168.178.9. Nachdem die Seite geladen ist, berühren Sie eine Schaltfläche, um das Objekt (den LED-Strip) ein- oder auszuschalten.

Die Seite gibt auch den Schaltzustand des Tasters SW1 auf dem Drucktaster-Board EB007 wieder. Drücken Sie den Taster und berühren Sie auf dem Tablet eine Schaltfläche, damit der aktuelle Schaltzustand von SW1 in die Anzeige übernommen wird.

Im Menü Ihres Routers können Sie den weltweiten Zugriff auf den E-Block EB023 aus dem Internet erlauben. Wenn Sie den Zugriff freigeben, können Sie das Objekt von jedem Ort der Welt schalten, an dem das Internet zugänglich ist. Im Prinzip können Sie über Ihr Tablet ein beliebiges Objekt schalten. Denken Sie daran, dass das E-blocks-Relais-Board EB038 zum direkten Schalten der Netzspannung 230 V wegen der elektrischen Sicherheit nicht geeignet ist.

Hilfestellung

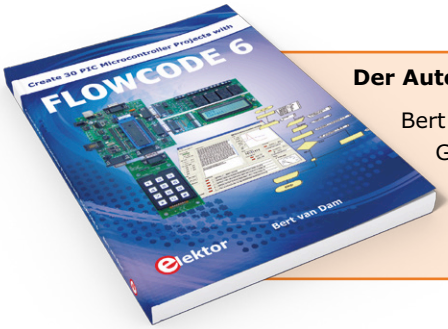
Ausführliche Informationen zu den E-blocks stehen auf der zugehörigen Elektor-Webseite [1] bereit, der Quellcode des Programms in Flowcode kann von der Projektseite [2] heruntergeladen werden. Andere nicht alltägliche Projekte werden in dem englischsprachigen Elektor-Buch „Flowcode V6 – 30 Projects for PIC Microcontrollers“ vorgestellt.

(140351)gd

Weblinks

[1] www.elektor.de/devtools/eblocks und www.elektor.de/devtools/flowcode-1-user/

[2] www.elektor-magazine.de/140351



Der Autor

Bert van Dam ist freiberuflicher Autor praxisorientierter Fachliteratur zu ausgewählten Gebieten der Elektronik. Die Schwerpunkte seiner Themen sind die Mikrocontroller-Familien PIC und ARM, die Einplatinencomputer Arduino und Raspberry Pi, die Künstliche Intelligenz (KI) sowie die Programmiersprachen JAL, C, Assembler, Python und Flowcode.

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

elektor GOLD membership

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

- Sie erhalten **10 Elektor-Hefte** (8 Einzelhefte + 2 Doppelausgaben Januar/Februar und Juli/August) pünktlich und zuverlässig frei Haus.
- **Extra:** Jedes Heft steht Ihnen außerdem als PDF zum sofortigen Download unter www.elektor-magazine.de (für PC/Notebook) oder via App (für Tablet) bereit.
- **Exklusiv:** Sie erhalten alle 2 Wochen per E-Mail ein neues Extra-Schaltungsprojekt (frisch aus dem Elektor-Labor).
- **Exklusiv:** Wir gewähren Ihnen bei jeder Online-Bestellung 10% Rabatt auf unsere Shop-Produkte – dauerhaft!
- **Exklusiv:** Der Online-Zugang zum Community-Bereich www.elektor-labs.com bietet Ihnen zusätzliche Bauprojekte und Schaltungsideen.
- **Extra:** Die neue Elektor-Jahrgangs-DVD (Wert: 27,50 €) ist bereits im Mitgliedsbeitrag inbegriffen. Diese DVD schicken wir Ihnen sofort nach Erscheinen automatisch zu.
- **Extra:** Als Dankeschön gibt es einen Elektor-Gutschein in Höhe von 25 € GRATIS obendrauf!

UMWELTSCHONEND – GÜNSTIG – GREEN

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

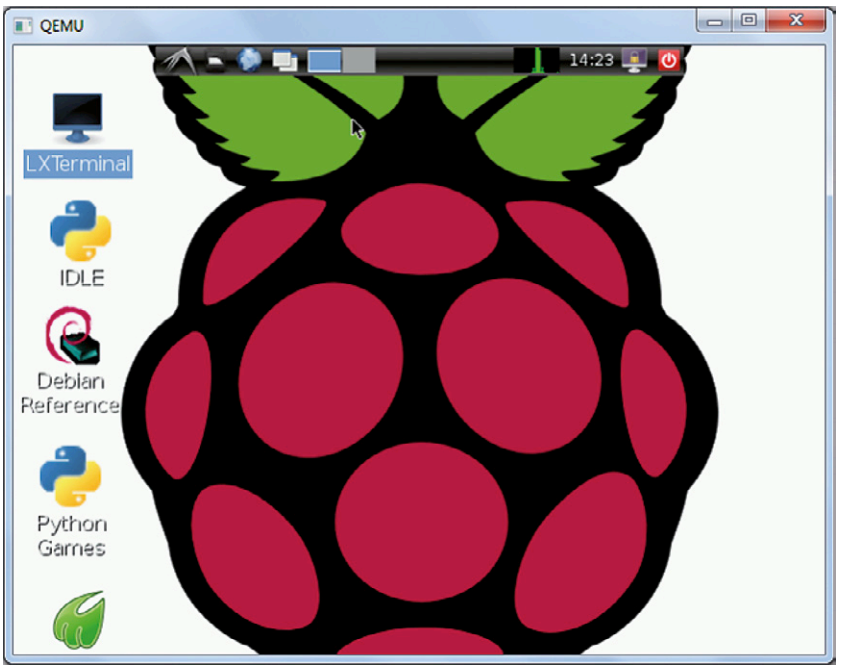
elektor GREEN membership

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

Raspberry-Pi-Emulator

Von **Bert van Dam** (NL)

Einen Raspberry Pi haben Sie (noch) nicht, trotzdem möchten Sie auf dem Laufenden sein? Der Emulator Qemu, der zur Klasse der Open-Source-Software gehört, ebnet den Weg. Qemu emuliert diesen inzwischen recht beliebten und verbreiteten Single-Board-Computer auf einem Windows-PC.



Vielleicht sind Sie über einen Elektor.POST-Beitrag [1] mit dem Raspberry Pi in Berührung gekommen, oder das Elektor-Buch „Raspberry Pi“ von Bert van Dam [2] hat Sie inspiriert. Vielleicht möchten Sie aber auch erst einmal schauen, was es mit diesem vielversprechenden System auf sich hat. Hier ist der Emulator Qemu in seinem Element, der den Raspberry Pi unter Windows auf dem PC emuliert. Allerdings kann die Emulation nicht absolut originalgetreu sein, denn auf dem PC sind einige für den Raspberry Pi spezifische Hardware-Komponenten nicht verfügbar. Trotzdem lässt der Emulator die Leistung und Vielseitigkeit des Raspberry-Pi-Systems vorausahnen. Damit Sie nicht lange suchen müssen, bieten wir die Emulator-Software auf der Elektor-Website zum Download an. Die nachstehende Anleitung bezieht sich auf das Betriebssystem Linux, das als Standard auf dem Raspberry Pi läuft. Die Linux-Distribution befindet sich auf der SD-Speicherkarte, die zum Raspberry-Pi-Buch von Elektor gehört. Andere Debian-Distributionen sind gleichermaßen nutzbar.

- Gehen Sie zu der unter [4] genannten Elektor-Website und laden Sie die Emulator-Software *Qemu* sowie den *DiskImager* herunter. Beide Programme laufen unter Windows, wir haben sie unter der 64-bit-Version von Windows 7 getestet.
- Legen Sie einen neuen Ordner an, beispielsweise `c:\qemu`, und entpacken Sie die heruntergeladenen Dateien in diesen Ordner. Die entpackten Programme müssen nicht installiert werden.
- Verwenden Sie den DiskImager, um eine Kopie der zum Buch gehörenden SD-Karte zu erstellen. Beim Starten des DiskImagers werden Sie von Windows gefragt, ob Sie Änderungen an Ihrem Windows-System zulassen wollen. Beantworten Sie diese Frage mit „Ja“. Windows wird Sie eindringlich warnen, doch die Warnung können Sie ignorieren, das Programm arbeitet trotzdem korrekt. Geben Sie der Kopie den Dateinamen `rpibookbertvandam.img` und speichern Sie die Kopie im Ordner `Qemu`.
- Starten Sie die Batch-Datei `fix.bat` im Ordner `Qemu`, indem Sie auf diesen Dateinamen doppelklicken. Jetzt startet die Emulation, sie stoppt mit der letzten Zeile:
  `root@(none):/#`
  Dies ist das Linux-Prompt.
- Geben Sie das folgende Kommando ein:
  `nano /etc/ld.so.preload`
  Das Kommando startet einen Texteditor, der diese Textzeile (oder sehr ähnlich) enthält:
  `/usr/lib/arm-linux-gnueabihf/libcofi_rpi.so`
  Fügen Sie vor dieser Zeile eine Raute (#) ein und drücken Sie CTRL-X. Drücken Sie

anschließend Y und danach ENTER.

- Damit haben Sie den Editor verlassen, erkennbar daran, dass Sie zum Linux-Prompt zurückkehren:
  `root@(none):/#`
  Geben Sie das Kommando `halt` ein, drücken Sie ENTER und warten Sie, bis wieder der Prompt erscheint.
  Eine eventuelle Fehlermeldung können Sie ignorieren.
  Danach schließen Sie das Windows-Fenster.
- Starten Sie noch einmal die Batch-Datei `run.bat` im Ordner `Qemu`, indem Sie auf den Dateinamen doppelklicken. Nun wird die Kopie geladen, die Sie vom Inhalt der SD-Speicherkarte erstellt haben. Nach kurzer Wartezeit erscheint das Begrüßungsfenster des Raspberry-Pi-Systems. Die Meldung `failed` beim Laden des Betriebssystems und die Fehlermeldungen beim Laden von `libmod` sind unbedenklich. Diese Meldungen rühren daher, dass die Software bestimmte Hardware-Komponenten erwartet, die auf einem PC-System nicht vorhanden sind.

Nun steht den ersten Gehversuchen im Emulatorfenster nichts im Weg. Sie können sogar die Beispiele der ersten Kapitel im Raspberry-Pi-Buch nachvollziehen. Der Emulator kann natürlich keine Projekte aus dem Buch abarbeiten, die an Hardware-Komponenten geknüpft sind. Er emuliert ausschließlich den Raspberry Pi, nicht jedoch angebundene Hardware. Wenn Sie die Maus innerhalb des Emulationsfensters bedienen, werden die Aktionen auf die Raspberry-Pi-Emulation bezogen. Um die Maus an Windows zurückzugeben, drücken Sie die Tasterkombination CTRL-ALT.
Da die Emulation vom PC ein hohes Maß an Rechenleistung verlangt, läuft der emulierte Raspberry Pi möglicherweise deutlich langsamer als die Original-Version.

Die Emulation lässt sich jetzt immer mit `run.bat` starten und mit dem Kommando `sudo shutdown -h now` in einem Terminalfenster anhalten. Alternativ können Sie die Taskleiste aufrufen (rotes Power-Symbol) und `Shutdown` aktivieren. Warten Sie, bis die Meldung `system halted` erscheint, erst danach schließen Sie das Fenster.

(130539)gd

Wir danken dem Projekt XEC Design (xecdesign.com) für die Vorabinformationen.

Weblinks

[1] www.elektor-magazine.de/post
[2] www.elektor.de/rpi-buch
[3] www.elektor.de/rpi-software
[4] www.elektor-magazine.de/130539

Dehnung messen
Präzisionsmessung mit PSoC

Von **Kendal Castor-Perry** und **Nidhin MS** (Cypress Semiconductor)

Viele Druck- und Kraftsensoren basieren auf sich verändernden Widerstandselementen in einer Brückenschaltung. In der Elektronikwelt sind „Brücken“ nichts Ungewöhnliches – man denke nur an die häufig vorkommenden Brückengleichrichter. Kaum ein Netzteil kommt ohne diese Gleichrichtervariante aus.

Eine Brücke mit Impedanzen, deren Werte extern beeinflusst werden können, ist eine sehr vielseitige Messanordnung. Wenn einige Impedanzen frequenzabhängig sind, kann man damit interessante Filter- und Oszillatorschaltungen bauen. Jeder Elektroniker hat zumindest schon einmal vom Wien-Brücken-Oszillator [1] gehört, der die frequenzabhängigen Eigenschaften der Brücke von **Bild 1** ausnutzt.
Brückenschaltungen tauchen auch in anderen passiven Filtern auf, doch werden sie häufig „verdreht“ als Gitter gezeichnet. Bei der Darstellung in **Bild 2**, das wie Bild 1 auch aus Wikipedia entnommen wurde, sind links die beiden Eingänge und rechts die beiden Ausgänge angeordnet. Wenn man hier die Z-Zweige als Widerstände und die Z’-Zweige als Kondensatoren ausführt, resultiert ein Allpass 1. Ordnung.

Prinzip und Probleme

In diesem Beitrag dreht es sich selbstverständlich um Brückenschaltungen aus Widerständen, deren Werte von physikalischen Faktoren beeinflusst werden. Mögliche Faktoren wären Temperatur (die absichtlich erfasst wird oder stört), Magnetfelder, Helligkeit, Feuchtigkeit oder eine der wichtigsten industriellen Anwendungen: die physikalische Dehnung eines mechanischen Systems.
Wer sich nicht mehr richtig an Mechanik erinnert: Dehnung ist ein Phänomen, das allen physikalischen Objekten passiert, wenn daran gezogen wird. Dehnung ist eine Änderung der physikalischen Dimensionen. Gemeint ist dabei eine Länge, der Querschnitt oder beides. Da diese Effekte die strukturellen Eigenschaften des Objektmaterials beeinflussen, kann sich damit auch die Leitfähigkeit für elektrischen Strom (oder umgekehrt sein Widerstand) ändern.
Eine Widerstandsbrücke wird praktisch so ausgeführt, dass ein oder zwei Widerstände von den mechanischen Verformungen des zu messenden Objekts betroffen sind. Der hierfür verwendete Widerstandstyp wird als Dehnungsmessstreifen oder englisch „strain gauge“ bezeichnet. Konkret bestehen diese Streifen aus mäanderförmigen dünnen Metallschichten auf einem Kunststoffsubstrat, das auf das zu messende Objekt aufgeklebt wird. Als mit Messtechnik befasster Elektroniker interessiert daran vor allem, wie genau man die Information über eine Dehnung in Form von Spannungen oder deren Veränderung erfassen kann.
Wäre das trivial, gäbe es kaum die umfangreiche Literatur zum Thema und Halbleiter-Hersteller würden kaum spezielle Produkte für exakt diese Messprobleme entwickeln und herstellen. Die Welt der Messung solcher analogen Werte ist nämlich mit ordentlichen Stolpersteinen gepflastert. Die Hauptgründe hierfür liegen darin, dass diese Streifen möglichst klein sein müssen (damit sie sich nicht permanent verformen) und dass

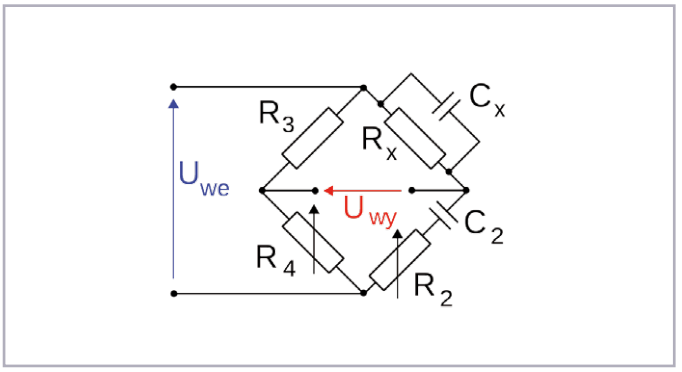


Bild 1. Die Wien-Brücke (aus Wikipedia).

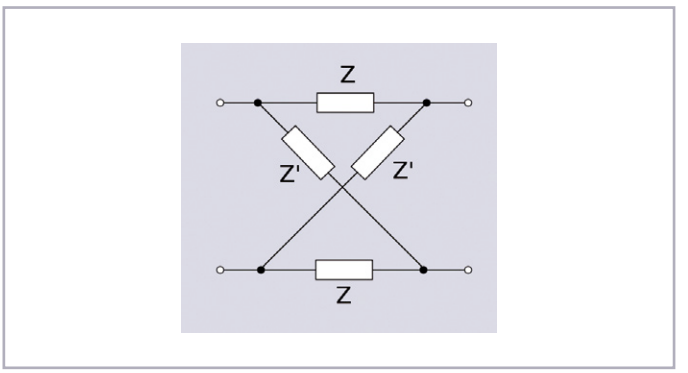


Bild 2. Eine Brücke in Gitteranordnung.

sowohl die Dehnung selbst als auch ihr elektrischer Effekt klein ausfallen, was eine Messbrücke folglich nur wenig verstimmt und die Spannungen klein sein lässt.

Neues aus der Brückenwelt

Die Frage ist, wie man den Effekt einer solcher Messbrücke am besten auswertet und was für einen Effekt man erwarten kann. Die Messung der kleinen Spannung an den Ausgängen einer solchen Brücke ist die klassische Anwendung für einen Instrumentenverstärker. Ein solcher Verstärker soll die Differenz seiner Eingänge verstärken und sich dabei nicht von einer im Vergleich dazu hohen Gleichtaktspannung irritieren lassen. Die Ausgangsspannung eines solchen Verstärkers wird heutzutage dem Eingang eines ADC (AD-Wandler) zugeführt, da rein analoge Messsysteme nicht mehr ganz zeitgemäß sind. Gerade klassische Instrumentenverstärker-ICs verfügen über ausgetüftelte Maßnahmen, um einen möglichst niedrigen Eingangs-Offset sowie niedrige Drift und Rauschen zu erzielen. Sie sind daher in aller Regel richtig teuer. Wo es auf Genauigkeit ankommt, werden heute hochpräzise Delta-Sigma-Wandler mit Auflösungen im Bereich von 20 bis 24 bit eingesetzt, die schon über integrierte Verstärkerstufen verfügen, damit man direkt Spannungen im Mikrovoltbereich erfassen kann. Doch da das Messen von Größen immer weitere Anwendungen erfährt und dies unter dem Druck von Kosten, Größe und Energieaufnahme steht, schauen sich Entwickler gerne nach ökonomischen Lösungen um. Die Qualität der analogen Seite üblicher preiswerter Mixed-Signal-Mikrocontroller allerdings reicht nicht aus. Es gibt aber eine Technik, die besonders bei der Implementierung von Ein-Chip-Sensor-Lösungen ihre Stärke ausspielt, wo es um Brückensensoren mit niedrigen Ausgangsspannungen geht. Der *terminus technicus* lautet CDS (**C**orrelated **D**ouble **S**ampling).

Die modernsten programmierbaren SoCs (**S**ystem **o**n a **C**hip) weisen aber gerade im analogen Sektor deutliche Verbesserungen gegenüber „normalen“ Mikrocontrollern mit Analogfähigkeiten auf. Hervorzuheben wäre da z.B. die Offsetspannung von <1 mV durch Abgleich im Herstellungsprozess, wie das bei der PSoC-Familie 4200 von Cypress Semiconductor erzielt werden konnte. Für viele Anwendungen sind solche Chips eine gute Ausgangsbasis. Sie vereinfachen und verkürzen den Entwicklungsprozess bei der Datenerfassung mittels moderner Sensoren erheblich.
Wenn die Signalpegel aber nicht im Milli- sondern im Mikrovoltbereich liegen, dann müssen auch diese neuesten Produkte der Prozessorkunst durch kluge Schaltungstechnik unterstützt werden. Glücklicherweise verfügen diese PSoCs über flexible analoge Routing-Strukturen, wodurch die Verwendung der schon angesprochenen CDS-Technik zur Präzisionssteigerung möglich wird.
Das Akronym CDS umfasst einen weiten Bereich an Techniken, in denen unterschiedliche Messungen so kombiniert werden,

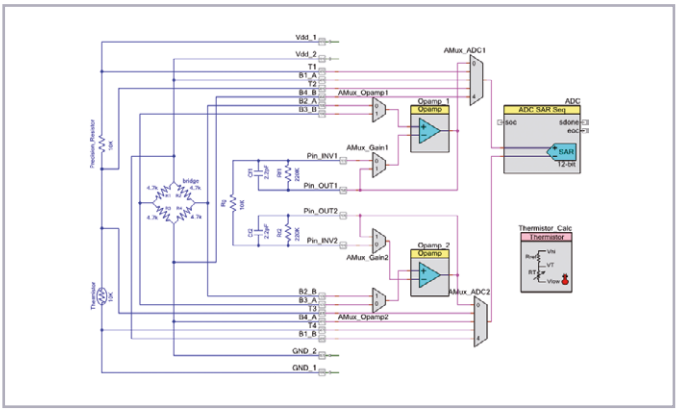


Bild 3. Eine Schaltung, die CDS-Technik für die differentielle Brücken- und unipolare Temperaturmessung verwendet.

dass bestimmte Fehlerarten reduziert werden, die miteinander korreliert sind. Daher der Name.
Das einfachste Verfahren der zweifachen Messung einer Spannungsdifferenz, wobei zwischen beiden Messungen der nichtinvertierende und der invertierende Eingang des Messverstärkers getauscht und die resultierenden Ergebnisse subtrahiert werden, kann schon viele Fehler der analogen Schaltung beseitigen. So löscht diese Maßnahme die inhärente Eingangsoffsetspannung und reduziert auch niederfrequentes Rauschen egal welcher Herkunft (Stromversorgung, thermische Effekte oder das NF-Eingangsrauschen von Verstärkern) die in der Zeitspanne zweier Messungen statisch auftreten. Außerdem werden Gleichtaktstörungen deutlich besser unterdrückt.

Für noch optimiertere Anwendungen, bei denen sich die Fehlerkomponenten zeitlich sehr schnell ändern, kann dieses Verfahren erweitert werden. Hierbei wird der Effekt der zeitlich versetzten Messung mit vertauschten Eingängen des Verstärkers ausgeschaltet. Zu diesem Thema lohnt sich die Lektüre der weiterführenden Informationen zur „Filter-Wizard-Technik“ [2]. Die Schaltung selbst bleibt dabei gleich. Lediglich das folgende Post-Processing wird dann etwas aufwändiger.

Mit einem PSoC

In **Bild 3** ist demonstriert, wie man so ein Prinzip in die Praxis umsetzen kann. Basis ist der schon erwähnte PSoC 4200, der über die nötige analoge Hardware verfügt. Solche Chips unterscheiden sich noch einmal deutlich von konventionellen Mixed-Signal-Mikrocontrollern, da sie über komplexere und bessere Routing- und Umschaltmöglichkeiten für die analogen Signale verfügen. Diese Fähigkeiten werden in der Schaltung von Bild 3 nutzbringend eingesetzt. Zunächst zu dem Schaltungsteil, der den Ausgang der Messbrücke verarbeitet. Analoge Multiplexer (AMux_xxx) wählen Signale für die AD-Wandlung aus. Ein Opamp-Paar bildet einen Differenzverstärker, dessen Verstärkung zwischen niedrig (x 1 für Diagnostikzwecke) und

hoch mit Hilfe externer Präzisionswiderstände umgeschaltet werden kann. Die Qualität dieser Opamps ist zwar gut, doch für Signale im Mikrovoltbereich nicht ganz ausreichend bei Dehnungsmessungen. Deshalb wird noch das CDS-Verfahren durch entsprechende Signalumschaltung an den Verstärkereingängen durchgeführt, was wie schon erwähnt die Messung von einem großen Teil der sonst vorhandenen Fehler wie Offset, Drift und NF-Rauschen befreit.

Alles Berechnung

Zur Berechnung eines Dehnungswerts benötigt man zwei Informationen. Zum einen benötigt man eine möglichst genaue Messung der kleinen Spannung der Messbrücke. Zusätzlich muss noch bekannt sein, mit welcher Spannung die Brücke versorgt wird. Die Versorgung der Brücke erfolgt hier über GPIO-Pins des PSoCs, damit die Dehnungsmessstreifen für geringe Stromaufnahme deaktiviert werden können. Die Versorgungsspannung wird über eine Vierleiter-Anordnung den beiden vielkanaligen Multiplexern des ADCs zugeführt, sodass die Messung sehr genau erfolgt. Der ADC selbst kann Spannungen bis zur Versorgungsspannung messen, da er so konfiguriert ist, diese als Referenzspannung zu nutzen. Letzteres beeinträchtigt die Messgenauigkeit der Dehnung in keiner Weise, da sich so die Versorgungsspannung als Konstante extrahiert. Zu beachten ist, dass hier die generelle Dehnung gemessen wird. Welche Widerstände sich wie verändern, wenn die Struktur belastet wird, auf der die Dehnungsmessstreifen aufgebracht sind, ist unwichtig. Wenn man die konkrete Belastung bzw. Dehnung erfassen will, muss man mehr über die Brücke wissen. Für höchste Genauigkeit muss eine Linearitätskorrektur durchgeführt werden und dazu muss man wissen, welche Widerstände gedehnt werden und welche lediglich als Referenzelemente fungieren.

Die vorliegende Schaltung erlaubt schon eine hohe Messqualität. Der interne ADC des PSoC hat zwar eine Auflösung von „nur“ 12 bit, doch man kann die hohe mögliche Samplerate dieses Chips von bis zu 1 MS/s nutzen und dank der eingebauten Hardware sogar ohne CPU-Belastung mehrere Messungen mitteln, um das Rauschen zu erniedrigen und so die wirksame Auflösung per Oversampling zu erhöhen. Mit der 44-fachen Verstärkung der Eingangsstufe und einer Mittelung

Die Software machts

Auf den ersten Blick liegt es nahe, für Messungen den Kanal mit Differenzeingängen zu verwenden. Das aber ist unnötig. Es reicht aus, die Spannungen an jedem Ende der Widerstände mit Verstärkern ohne Differenzeingänge zu erfassen. Wenn man die beiden Messungen subtrahiert, resultiert die reine Spannung über dem Widerstand. Es ist natürlich klar, dass die Subtraktion zweier fast gleich großer Werte zu Ungenauigkeiten führen kann. Deshalb muss man dafür sorgen, dass die ADC-Auflösung für die gewünschte Genauigkeit ausreicht. Diese Art Messung erfordert eine bestimmte Menge von Bits, „ENOB“ (**E**quivalent **N**umber **O**f **B**its) genannt, die gut über 16 bit liegen können.
Der große Vorteil dieser CDS-Variante ist, dass jeder statische oder quasi-statische Fehler am Eingang ausgemerzt wird, auch wenn keine differenzielle Messung vorgenommen wird. Das Messergebnis ist daher nicht von Fehlern wie Kanal-Offset oder NF-Rauschen beeinträchtigt, wie das bei der Brückentechnik der Fall ist. Es ist daher ein gutes Verfahren um gute Messungen mit einem nichtdifferenziellen Verstärker durchzuführen. Die Messung hängt natürlich noch von der Qualität der Referenzspannung des ADCs ab, doch dieser extrahiert sich, wenn man das Verhältnis aus Präzisionswiderstand und Thermistor bildet. Letztlich erhält man eine systemunabhängige Messung eines Widerstandsverhältnisses. Dieses kann man per Software in eine Temperatur umwandeln. Hier ergibt sich eine weitere clevere Einsatzmöglichkeit eines PSoCs:

Die Temperaturberechnung wird von einer Software-Komponente erledigt, die schon in den Entwicklungs-Tools implementiert ist. Man zieht sie kurz in das Schema und konfiguriert sie für den verwendeten Thermistor.
In diesem Beispiel misst der ADC insgesamt acht unterschiedliche Spannungen bei der Messbrücke und dem Thermistor. Zwischen den Messungen steuert die Firmware die diversen Multiplexer so, dass die nächste Spannung gemessen werden kann. Der ADC im PSoC 4200 verfügt zudem noch über Verarbeitungsmöglichkeiten, die das Mitteln mehrerer Messungen zur Erzielung einer höheren effektiven Auflösung ermöglicht. Diese Mittelung ist völlig von der CPU unabhängig. Wo es auf niedrige Stromaufnahme ankommt, kann sich die CPU die meiste Zeit im Sleep-Mode befinden. Die CPU wird vom ADC nach jeder Messung (nach Mittelung) wieder aktiviert. Die CPU kann dann die Multiplexer passend zur nächsten zu messenden Spannung ansteuern, die nächste AD-Wandlung anstoßen und sofort wieder einschlafen. Nachdem alle acht Messungen durchgeführt wurden, kann die Firmware die Dehnung und die Temperatur berechnen.
Der verwendete PSoC verfügt über eine Menge Ausgabemöglichkeiten für Resultate. Er kann die Ergebnisse analog mit Hilfe seiner IDAQs (Stromquellen als DA-Wandler gegen Masse oder Versorgung) sowie die Daten über I^2C, UART, SPI oder andere Interfaces ausgeben oder aber direkt LCDs ansteuern.

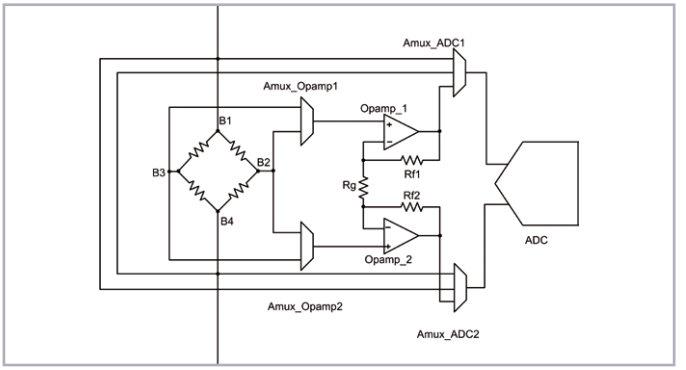


Bild 4. Schaltungsteil mit der Brücken-Messung aus Bild 3.

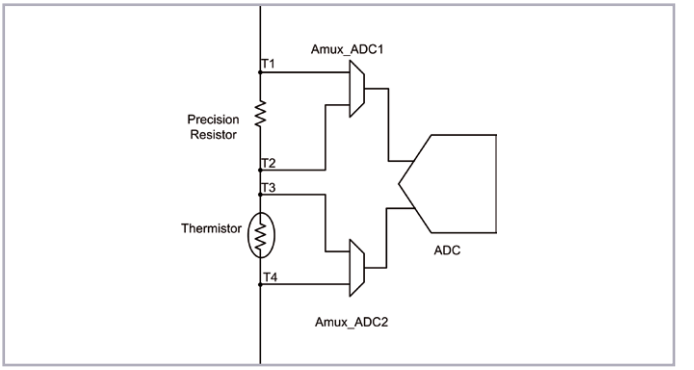


Bild 5. Prinzip der Messung von Widerstandsverhältnissen aus Bild 3.

über 256 Samples erreicht man eine Qualität, die der eines 15-bit-ADCs direkt an der Messbrücke nahe kommt. Dies entspricht einem Rauschpegel von <8 mV_{eff} mit einem Gleichspannungsfehler, den man mit herkömmlichen Mitteln kaum messen kann. All das ohne Verrenkungen beim Platinen-Layout.

Bild 4 zeigt eine vereinfachte Version des Messbrückenteils der Schaltung von Bild 3. Die relative Dehnung entspricht dem Verhältnis der Ausgangsspannung der Brücke zu ihrer Versorgung. Daher müssen für die Berechnung der relativen Dehnung die Spannungen B1-B4 und B2-B3 erfasst werden. Für CDS müssen diese Spannungen doppelt mit Polaritätsumkehr gemessen werden. Der ADC misst dabei die Versorgungspannung in Form von B1-B4 und B4-B1 direkt. Die Brückenausgangsspannungen B2-B3 und B3-B2 werden erst nach dem Differenzverstärker digitalisiert. Die Firmware berechnet dann aus diesen Werten die genaue relative Dehnung nach der Formel:

$$strain = \frac{ADC\,count(B2,B3) - ADC\,count(B3,B2)}{Gain_{preamp}(ADC\,count(B1,B4) - ADC\,count(B4,B1))}$$

wobei gilt: *strain* = relative Dehnung, *ADC count* = ADC-Wert und $Gain_{preamp}$ = Verstärkung des Differenzverstärkers.
Wenn man diese Schaltung mit dem Entwicklungsboard dieses PSoCs realisiert, erhält man eine hohe Stabilität bei Mikrovoltsignalen und kann auch kleine Dehnungen sehr genau messen. Mit der hier vorgesehenen Verstärkung um den Faktor 44 erreicht man eine maximale relative Dehnung von etwa 2 %. Bei den hier benötigten Bandbreiten kann man auch mit anderen externen Widerständen noch höhere Verstärkungen für Systeme mit geringer statischer Dehnung des Sensors realisieren.

Das Beispiel ist außerdem auch noch mit einer Messung der Temperatur ausgestattet. Diese wird simultan zur Dehnung erfasst und nutzt ebenfalls CDS-Technik. Die Messung der Sensortemperatur kann helfen, die Sensor-Empfindlichkeit zu optimieren. Ein Thermistor bietet aktuell das beste Preis/Leistungs-Verhältnis als Temperatursensor. Zur Temperaturmessung muss man dessen Widerstand messen. Für eine optimale Genauigkeit sollten auch hier auf möglichst wenig Offset und Rauschen und eine hohe Stabilität der Referenz und der Verstärkung geachtet werden.

Das grundlegende Messprinzip für die Temperatur in **Bild 5** ist einfach ist einfach: Ein Strom fließt durch den Thermistor und einen Präzisionswiderstand. Man misst die Spannungen über dem Thermistor und dem Widerstand. Das Verhältnis der Spannungen entspricht dem Verhältnis der Widerstände. Die Genauigkeit hängt daher direkt von der Güte des Präzisionswiderstands ab, doch dieser ist heute preiswert erhältlich. Einfluss hat auch die Genauigkeit, mit der man Spannungen über Widerständen messen kann. Hierzu gibt es im **Kasten** weitere Details.

Fazit

Das CDS-Verfahren bietet eine erstaunlich hohe Qualität bei der Erfassung kleiner Sensorspannungen. Man hält es kaum für möglich, mit wie wenig Aufwand und Kosten solche Mikrocontroller-Lösungen machbar sind. Besonders elegant wird die Sache, wenn man so eine Lösung mit einem passenden PSoC realisiert, der schon über die notwendigen internen Analog-Routing-Fähigkeiten verfügt. Die hier beschriebene Datenerfassung erlaubt den Aufbau kleiner und preiswerter aber doch hochqualitativer Messsysteme, die leicht an individuelle Gegebenheiten angepasst werden können.

(140230)

Weblinks

[1] Wien-Brücken-Oszillator:
http://de.wikipedia.org/wiki/Wien-Robinson-Brücke

[2] Filter-Wizard: www.cypress.com/?docID=45637

FFT-Analyse in VB

Spektralen Verlauf auch über der Zeit anzeigen

Von **Kurt Diedrich** (D)

Zum Aufspüren der in einem Signal enthaltenen Frequenzanteile reicht ein Oszilloskop nicht aus. Doch ein PC als Universal-Tool, kombiniert mit der passenden Software, erzeugt wunderschöne spektrale Darstellungen. Dieser Beitrag beschreibt, wie man so eine Aufgabe mit Visual Basic löst.

Die seit 1822 bekannte, vom gleichnamigen Mathematiker entwickelte *Fourier-Analyse* ermittelt rein rechnerisch die Amplitude der in einem Signal enthaltenen Frequenzen. 1965 wurde dann daraus von James Cooley und John W. Tukey die heute technisch gebräuchliche FFT (**F**ast **F**ourier **T**ransformation) entwickelt. Mit ihrer Hilfe kann man ein Signal als zweidimensionales Diagramm darstellen; die Amplitude wird dabei über der Frequenz aufgetragen (**Bild 1**). In Hardware-Form gibt es so etwas schon seit Jahrzehnten: Neben einfachen Varianten wie der eher zu Dekorationszwecken eingesetzten LED-Balkenanzeige auf der Basis von fixen Bandpässen gab es für Laboranwendungen auch spezielle, sehr genaue und ebenso teure Spektrum-Analyzer, mit denen man die in einem Signal enthaltenen Frequenzen als Spektrum darstellen konnte. Heute jedoch reicht die Rechenleistung normaler PCs locker aus, digitalisierte (Audio-)Signale nicht nur in Dateiform zu verwalten, sondern ganz ohne spezialisierte Hardware aus diesen Daten blitzschnell informative spektrale Darstellungen zu berechnen.

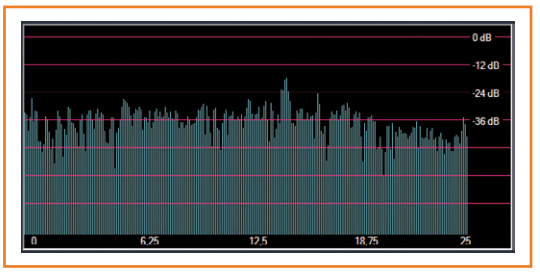


Bild 1.
Ergebnis einer klassischen FFT: Darstellung der Intensität der in einem Signalausschnitt enthaltenen Frequenzen (hier bis 25 Hz) als unterschiedlich hohe Linien.

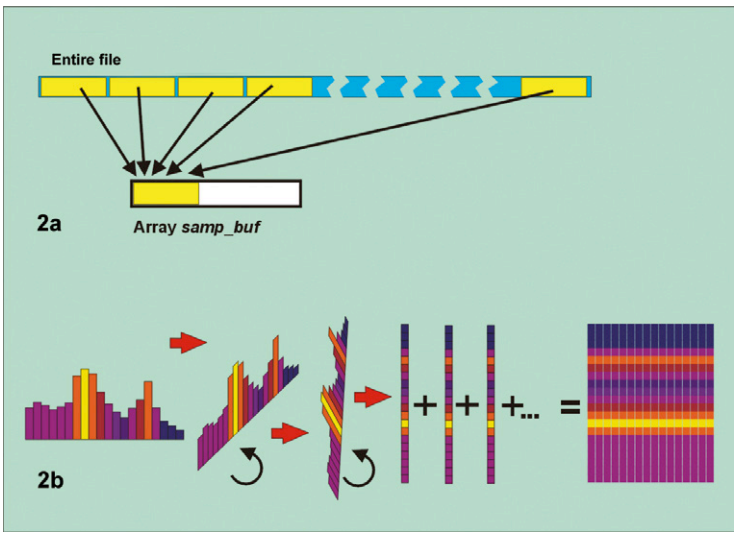


Bild 2. Die Grafik „FFT versus Time“ ergibt sich durch Berechnung aufeinander folgender Zeitintervalle (Bild 2a). Die Darstellung des Spektrums über der Zeit ergibt sich durch Kippen und Drehen (um 90 Grad) sowie Aneinanderreihung der Darstellung von Bild 1. Die Höhe der Balken ist durch intuitiv interpretierbare Farbwerte ersetzt (Bild 2b).

Zeit vs. Frequenz

Bei sich zeitlich verändernden Signalen ergibt sich ein Problem: Je genauer die Analyse sein soll, desto größere Zeitabschnitte muss man untersuchen. Da hierbei die in einem Zeitabschnitt enthaltenen Frequenzamplituden gemittelt werden, kämpft man prinzipbedingt mit einer Art „Fourierscher Unschärferelation“: Je höher die zeitliche Auflösung bei der Analyse sein soll, desto kürzer muss das zu untersuchende Intervall sein, was jedoch wiederum zu einer gröberen Frequenzauflösung führt – und umgekehrt.

Die im hier vorgestellten Programm angewendete FFT-Analyse über der Zeit (*FFT versus Time*) geht einen Schritt weiter. Sie generiert nicht etwa nur das statische Spektrum eines bestimmten

Signalabschnitts, sondern erfasst aufeinander folgende Zeitintervalle (**Bild 2a**). Damit kann die Veränderung der in einem Signal enthaltenen Frequenzen als spektraler Verlauf auch über längere Zeiträume angezeigt werden. Zur Darstellung des Ergebnisses werden hierfür drei Dimensionen benötigt: Zeit, Frequenz und Intensität. Die X-Achse entspricht meistens der Zeit - und die Y-Achse der Frequenz. Die Amplitude wird durch die Farbe bzw. Helligkeit eines Punktes in der Zeit-Frequenzfläche repräsentiert (**Bild 2b**). Es ergibt sich also eine Folge von farblich kodierten Spektren. Ein frequenzmodulierter Sinuston (wie bei einer Polizeisirene) zeigt sich in solch einer Darstellung als helle, horizontale Schlangenlinie vor dunklem Hintergrund.

Formeln und Code

Das Internet bietet zum Thema FFT enorm viel Information, die aufgrund komplizierter Formeln leider mathematische Vorbildung erfordert. Wesentlich einfacher gelingt der direkte Zugang über den Code zum Berechnen einer FFT.

In **Listing 1** ist zunächst der VB-Code für eine Frequenzanalyse zu einem bestimmten Zeitpunkt (Zeitintervall) gezeigt.

Als Beispiel dient eine (längere) WAVE-Datei eines Musikstücks. Gesucht sind die Amplituden der Frequenzen zu einem bestimmten, ab dem Zeitpunkt *t* beginnenden Zeitabschnitt Δt. Hierzu werden ab dem Zeitpunkt *t* beispielsweise 1.024 Samples in das Array *samp_buf* kopiert (siehe **Bild 3**). Der hierfür erforderliche Code ist nicht im obigen Beispiel enthalten.

Zur FFT-Berechnung sind noch die vor jeder Analyse auf Null gesetzten Buffer-Arrays *a* und *b* sowie für die Ergebnisse das Array *grafik* erforderlich, dessen Inhalt später dargestellt wird.

Die beiden verschachtelten Schleifen (mit *n* und *i*) sind für die Berechnung der FFT-Analyse zuständig. Die äußere Schleife unterteilt den in der Datei vorhandenen Frequenzbereich (Abtastrate / 2) in 512 Teile. Jeder Wert von *n* gibt demnach die Intensität der Frequenz im untersuchten Zeitbereich auf einer Skala von 1 bis 512 an.

Die Variable *i* in der inneren Schleife bestimmt die Länge des zu untersuchenden Zeitintervalls. Ein optimales Ergebnis ergibt sich bei $i = 2 \bullet n$. In der inneren Schleife werden die Samplewerte mit *Sinus*- bzw. *Cosinus 2 Pi* und den Schleifenvariablen multipliziert sowie in den Arrays *a* und *b* aufsummiert. Nach einem kompletten Durchlauf der inneren Schleife ist erst ein einziger Intensitätswert auf der Frequenzskala von 0 bis 512 berechnet.

In der letzten Schleife *cnt* werden die Inhalte der Arrays *a* und *b* quadriert, addiert und daraus die Wurzel gezogen. Die Wurzel als endgültiges Ergebnis wird dann im Array *grafik* gespeichert. Es enthält die gemittelten Amplituden der Frequenzen auf einer Skala von 1 bis 512 und repräsentiert das Spektrum des Intervalls von *t* bis *t + 1.024* Samples.

Alle Arrays wurden als *single* deklariert. Bei der grafischen Darstellung der *FFT vs. Time* müssen die im Array *grafik* enthaltenen Ergebnisse in Integer-Farbwerte umgewandelt werden. Mehr dazu in den Kommentaren des Quellcodes.

Rechenzeit sparen

Für das Spektrum eines bestimmten Zeitpunkts innerhalb der Datei wie in Bild 1 reicht der aufgeführte Code völlig aus. Bei einer FFT-versus-Time-Analyse müssen jedoch mehrere hundert

Listing 1.

```
Private Sub FFT()
  For n = 0 To 512
    a(n) = 0
    b(n) = 0
  Next n
  For n = 0 To 512
    For i = 1 To 1024
      a(n) = a(n) + samp_buf(i) * Math.Cos(2 * pi * n * i / 1024)
      b(n) = b(n) + samp_buf(i) * Math.Sin(2 * pi * n * i / 1024)
    Next i
  Next n
  For cnt = 0 To 512
    grafik(cnt) = Math.Sqrt((a(cnt) * a(cnt)) + (b(cnt) * b(cnt)))
  Next cnt
End Sub
```

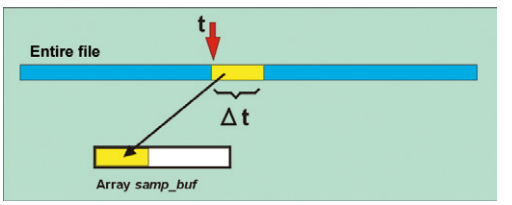


Bild 3.
Die Grafik von Bild 1 entsteht, indem zu einem bestimmten Zeitpunkt *t* ein Zeitintervall Δt mit einer festen Anzahl von Samples einer FFT-Analyse unterzogen wird.

Listing 2.

```
private sub pre_FFT
   For n = 0 To 512
      For i = 1 To 1024
         speicher1(n, i) = Math.Cos(2 * pi * n * i / 1024)
         speicher2(n, i) = Math.Sin(2 * pi * n * i / 1024)
      Next i
   Next n
End Sub
```

oder gar tausend solcher Analysen nacheinander ausgeführt werden. Damit man nicht mehrere Minuten warten muss, sollte die Ausführung des obigen Codes möglichst nicht länger als ein paar Millisekunden dauern. Dies erreicht man z.B. mit einem von vielen Tricks. Der in der inneren Schleife enthaltene Ausdruck:

```
Math.Cos(2 * pi * n * i / 1024)
```

enthält keine Werte der zu untersuchenden Datei, sondern nur Konstanten, die schon vorab berechnet und dann recycelt werden können. Da jede Multiplikation gerade in einer verschachtelten Schleife die Rechenzeit drastisch verlängert, werden diese Werte in einer eigenen Funktion *pre_FFT* vor der eigentlichen FFT berechnet. Die Ergebnisse von *pre_FFT* stehen dann in den Arrays *speicher1* und *speicher2* zur Verfügung. Man muss dann ihren Inhalt in der eigentlichen FFT-Schleife lediglich auslesen. Auf diese Weise wird die Rechenzeit erheblich reduziert. Der Code der Funktion *pre_FFT* steht in **Listing 2**.
Die betreffenden Zeilen in der Funktion *FFT* werden wie folgt geändert:

```
a(n) = a(n) + samp_buf(i) * speicher1(n, i)
b(n) = b(n) + samp_buf(i) * speicher2(n, i)
```

Es gibt noch weitere Möglichkeiten zur Beschleunigung, auf die hier jedoch nicht tiefer eingegangen wird. Interessierte finden sowohl im Programm-Quellcode [1| als auch in der Online-Hilfe des Programms weitere Hinweise.

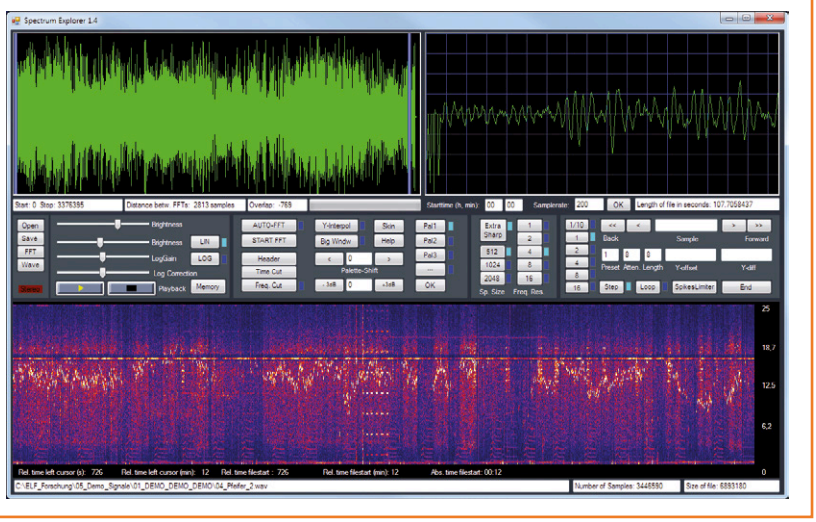


Bild 4. Programmoberfläche mit der Beispieldatei *04_Pfeifer_2.wav*. Man erkennt sofort ein sich in seiner Frequenz änderndes Sinussignal.

Resampling

Angenommen, die Samplerate einer Datei beträgt 400 Hz. Eine FFT mit dem obigen Code ergibt die Amplituden der Frequenzen von 1 bis 200 Hz (aufgrund des Abtast-Theorems) mit einer Auflösung von 1/512 des Gesamtbereiches. Angenommen, die niedrigen Frequenzen zwischen 1 und 10 Hz seien besonders interessant. Man kann hier aber leider nicht viel erkennen, weil sie am unteren Rand der Skala „zusammengequetscht" sind. Hier wäre ein Y-Zoom hilfreich. Nichts einfacher als das: In der inneren Schleife werden zwar immer noch 1.024 Werte berechnet, aber das Programm bewegt sich im array *samp_buf* nicht um ein Sample pro Rechenschritt weiter, sondern um 2, 4, 8 oder gar 16 Samples, was zu einer 2-, 4-, 8- oder gar 16-fachen Dehnung des Frequenzbereiches in Y-Richtung führt. Dazu muss die Variable *i* in der Zeile

```
speicher1(n, i) = Math.Cos(2 * pi * n * i
   / 1024)
```

lediglich mit 2, 4, 8 oder 16 multipliziert werden, oder mit einer Variablen (z.B. *downsampling*), die sich über die Benutzeroberfläche verändern lässt. Genau dies geschieht im hier vorgestellten Programm:

```
speicher1(n, i* downsampling) = Math.
   Cos(2 * pi * n * i / 1024)
```

Noch eine Schleife

Bisher wurde aus der Datei der Länge *l* lediglich ein kurzer Zeitabschnitt *t* herausgegriffen, analysiert und das Ergebnis als Amplitude über der Frequenz dargestellt.
Die nun beschriebene Analyse bezieht sich auf

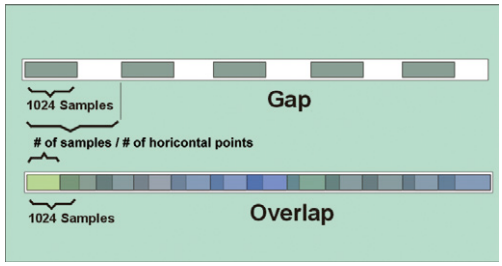


Bild 5. Um jedem horizontalen Pixel der Anzeige eine senkrechte, farbige Linie zuzuordnen, wird an den berechneten, zeitlichen Positionen jeweils eine Analyse für eine stets feste Anzahl von Samples durchgeführt. Je nach Dateilänge kommt es dabei zu Lücken oder Überlappungen (dunkle Quadrate im unteren Balken).

das komplette Signal. Aus diesem Grunde ist es erforderlich, das Gesamtsignal in *n* Abschnitte zu unterteilen und für jeden dieser Abschnitte eine gesonderte Analyse durchzuführen. Das grafische Ergebnis besteht aus aneinander gereihten, senkrechten Linien, wobei jede Linie eines der berechneten Zeitintervalle repräsentiert (Bilder 2a und b).

Angenommen, die auf dem Bildschirm dargestellte FFT-Grafik im unteren Teil von **Bild 4** sei genau 1.000 Pixel breit. Zur Vermeidung von Aliasing-Effekten muss die Datei vor der Analyse daher in 1.000 Segmente unterteilt werden. Für jedes Segment muss die oben beschriebene FFT-Analyse durchgeführt werden. Da immer nur 1024 Samples berechnet werden, ergeben sich je nach Dateilänge entweder Lücken (Gaps) oder Überlappungen (Overlaps), wie in **Bild 5** zu sehen. Da die Anzahl der Segmente im vorliegenden Programm wegen der konstanten Breite des Bildfensters immer gleich ist, hängt es von der absoluten Dateigröße ab, ob sich Überlappungen oder Lücken ergeben. Für die Qualität der Darstellung macht dies im vorliegenden Programm keinen wirklichen Unterschied.

Im Code erfolgt die Segmentierung durch eine zusätzliche Schleife: Der obige Code wird in eine Schleife eingebettet, deren Zählvariable die Werte von 0 bis zur Bildbreite (in Pixeln) durchläuft. Im vorliegenden Programm ist dies der Wert 1.185 (maximale Ausnutzung der Standard-Bildschirmbreite). Zusätzlich muss dafür gesorgt werden, dass in der inneren Schleife nach jedem kompletten Durchlauf bei der Adressierung von *samp-buf* ein Sprung um den Faktor *Dateilänge in Samples / Fensterbreite* erfolgt. Bei einer Datei mit 6 MSamples ergibt sich beispielsweise der Wert *Int (6.000.000 / 1.185) = 5.063*.

Fensterung

Schneidet man aus einem Signal ein Segment einer bestimmten Breite (FFT-Fenster) heraus, so werden auch die im Signal enthaltenen Sinuswellen willkürlich an- und abgeschnitten. Diese steilflankigen Schnittkanten entsprechen starken Oberwellen, die das Resultat verfälschen und eine stark verrauschte FFT-vs.-Time-Grafik ergeben würden. Zur Vermeidung solcher Artefakte werden die zu analysierenden Werte mit einer über das Zeitfenster gelegten Hüllkurve versehen. Von diesen gibt es zahlreiche bewährte Varianten, z.B. solche, die aus Sinus- bzw. Cosinusfunktionen abgeleitet wurden. Auf diese Weise werden die

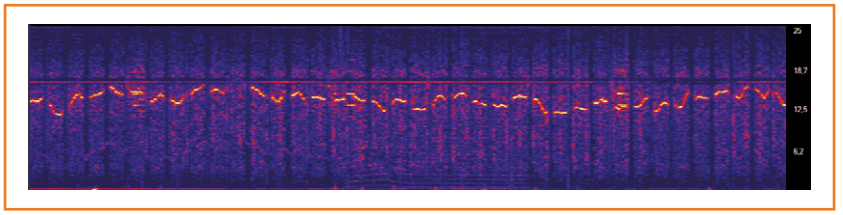


Bild 6.
Zoom durch Verschiebung der Slider im linken oberen Übersichtsfenster.

am Fensterrand gelegenen Werte weniger stark gewichtet und die erwähnten Artefakte somit reduziert. Im hier beschriebenen Programm wird das so genannte *Hanning-Fenster* verwendet, was im Quellcode sehr leicht nachvollzogen werden kann.

Installation und Bedienung

Wer das Programm lediglich anwenden möchte, ohne den Quellcode durch eigene Zeilen zu erweitern, kann sich beim Kopieren der heruntergeladenen Files [1] (unter *Bin\Release*) auf die folgenden drei Dateien beschränken: *Spectrum_Explorer.exe, AxInterop.WMPLib.dll* und *Interop.WMPLib.dll*. Die Exe-Datei kann auf jedem Rechner durch Doppelklick gestartet werden, auf dem das *.Net-Framework 4* installiert ist (getestet auf Windows 7 mit 32 und 64 Bit). Zusätzlich zur Exe-Datei sind noch die beiden DLLs (im selben Verzeichnis) erforderlich, die für die akustische Wiedergabe der Dateien sorgen. VB-Programmierer wissen, wie sie mit dem heruntergeladenen Code umgehen müssen.
Das Programm wurde ursprünglich zur Analyse der Dateien des ELF-Empfängers geschrieben, der im September 2014 in Elektor veröffentlicht wurde [2]. Es ist daher ausschließlich für Mono-WAVE-Dateien mit 16 bit Auflösung geeignet.
Die Beschreibung der Bedienung und der Funktion des Codes bis ins letzte Detail würde diesen Artikel sprengen. Daher ist im Download auch eine PDF-Datei verfügbar, die mit den wesentlichen Features des Programms vertraut macht. Außerdem ist der Quellcode ausgiebig deutsch und englisch kommentiert. Per Help-Taste auf der Benutzeroberfläche wird zudem eine Kopie der Oberfläche aufgerufen, die beim Anklicken eines Buttons die Erläuterung der jeweiligen Funktion bietet. Die **Kurzanleitung** im Kasten müsste für erste Explorationen aber ausreichen. Für die Erforschung der Feinheiten des Programms ist die PDF-Anleitung besser. Probieren Sie ruhig einmal andere Dateien sowie alle vorhandenen Buttons aus…

(140246)

Weblink

[1] www.elektor-magazine.de/140246
[2] www.elektor-magazine.de/140035

Kurzanleitung

- Klicken Sie auf *Open*.
- Laden Sie die Beispieldatei: *04_pfeifer_2.wav* (16 Bit, Mono, 200 Hz Samplerate).
- Im unteren Bereich wird ein Analyseergebnis sichtbar, das dem in Bild 5 entspricht.
- Die Analyse ist auf „schnell – aber unscharf" als Standard voreingestellt.
- Deaktivieren Sie *Y-Interpol* und aktivieren Sie *Extra Sharp*.
- Tragen Sie im Eingabefeld *Samplerate* den Wert *200* ein und klicken Sie auf *OK* (ansonsten stimmt die Frequenzskalierung nicht).
- Durch Klick auf *OK* bzw. auf *START FFT* wird die neue Analyse gestartet.

Die Grafik erscheint jetzt schärfer als zuvor. An der Stellung der Slider im Fenster oben links kann man erkennen, dass nahezu der gesamte Bereich einer fast drei Stunden langen Aufnahme angezeigt wird. Die maximale dargestellte Frequenz beträgt 25 Hz (siehe Skala rechts). Die stark unregelmäßige Linie im mittleren Frequenzbereich zeigt, dass hier Sinussignale vorliegen müssen, die sich in ihrer Frequenz ständig unregelmäßig ändern. Dies kann man wie folgt näher untersuchen:

- Schieben Sie mit der Maus den rechten Slider im linken oberen Fenster soweit nach links, dass er sich nur einen Finger breit vom linken Slider entfernt befindet.
- Klicken Sie nun erneut auf *Start FFT*.

Nach einer leichten Reduzierung der Helligkeit mit dem zweiten *Brightness*-Schieber von oben sieht man das neue Ergebnis: Gleich lange Sinustöne kurzer Dauer (**Bild 6**), die sich in ihrer Frequenz permanent ändern.

Hexadoku Sudoku für Elektroniker

Alle Jahre wieder kommt auch der vorweihnachtliche Stress. Da sollten Sie zwischendurch unbedingt mal zu Ruhe kommen. Sehr gut gelingt das, wenn Sie sich mit unserem monatlichen Hexadoku in eine stille Ecke zurückziehen. Mit etwas Glück und Verstand können Sie außerdem noch etwas gewinnen, denn hier warten einmal wieder fünf schöne Elektor-Buchgutscheine.

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 einen von fünf Buchgutscheinen im Wert von 50 Euro gewinnen!

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. Dezember 2014!

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

Die richtige Lösung ist: **C03EF**.
Einen Elektor-Buchgutschein über je 50 € haben gewonnen:
Philippe Monnard, Ivan Corneillet, Siegfried Kepper, Steve Hasko und Jean-Paul Lagaisse.
Herzlichen Glückwunsch!

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

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

Röhrenverstärker Heathkit AA-100 (1960)

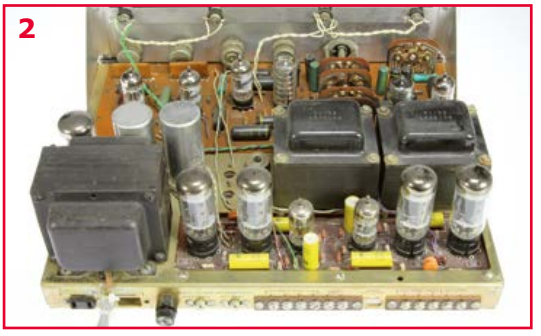
Ein toller Bausatz für nur 85 $

Von **Jan Buiting**
(Chefredakteur englische Ausgabe)

Einst ein Schwergewicht und heute ein Fossil: Als der Röhrenverstärker AA-100 von Heathkit 1960 vorgestellt wurde, konnte er mit sagenhaften 2x25 W in stereo, preiswerten Endstufenröhren sowie vielen Eingängen beeindrucken. Außerdem konnte man ihn selbst zusammenlöten. Und so ein Teil landete neulich bei uns...

Die Firma Heathkit repräsentiert selbst für stromaffine Nichtamerikaner eine besondere Ära in der Elektronik. Wer als Elektroniker schon den runden netzfrequenzartigen Geburtstag hinter sich hat, kennt diesen Firmennamen. Möglicherweise hat er sogar selbst einen Heathkit-Bausatz in den Fingern gehabt oder sich „früher“ eines der vielen Funk-, Mess- oder Netzgeräte von Heathkit gewünscht. Ich selbst hatte schon in den 1970ern mit einem „Heath“-Röhrenvoltmeter und einem Grid-Dipper zu tun, auch wenn sie damals nicht von mir zusammengebaut waren. Die Anleitungen von Heathkit waren bezüglich Klarheit und technischer Illustrationen vorbildlich. Wer sich für den damaligen US-Tech-Speak interessiert, der hat auch heute noch Spaß daran.


1


2

Unser Mann in Belgien

Ein besonders treuer Elektor-Leser wollte mir freundlicherweise „ein paar Sachen aus seiner Sammlung“ überlassen. Unter der Bedingung, dass die imposanteren Stücke in *Retronik* beschrieben würden. Also besuchte ich ihn im Frühjahr 2013. Raymond ist ein emeritierter Professor der Computertechnik. Er studierte und lehrte an unterschiedlichen amerikanischen Universitäten, darunter auch die berühmteren aus Kalifornien. Schließlich kehrte er nach Holland an die Universität von Nijmegen zurück. Mittlerweile genießt er sein Leben als Rentner und lebt gerade mal 40 km vom „Elektor Castle“ entfernt in Belgien. Er beschäftigt sich immer noch mit Elektronik und zwar speziell mit Computer-Funk. Unter den vielen verschiedenen Dingen aus seiner Sammlung war ein großer und schwerer Karton.

Ein Amerikaner!

Nachdem ich beim Auspacken auf den Schriftzug „Heathkit/Daystrom“ stieß, handelte ich so wie jeder gute Sammler und Restaurateur: Ich suchte Unterstützung beim Hersteller, auch wenn die Herstellung lange her war. Google spuckte folgendes aus: „Der Luxus-Stereo-Verstärker AA-100 von Heathkit übertrifft andere Verstärker-Bausätze bei weitem. Dieser aufregende High-Fidelity-50-Watt-Verstärker ist besser als doppelt so teure Geräte. Dieser Bausatz bietet mit zwei verzerrungsfreien 25-Watt-Kanälen genug Leistung für jede Stereo-Anlage. Der Verstärker kostet

$ 84.95 - also nur $ 1.70 pro Watt.“
„Dank seiner Elektronik ist der AA-100 sicher der beste Verstärker, den man kaufen kann. Die besonderen Eigenschaften dieser Schönheit ergeben einen Verstärker von besonderem Wert. Das braune, vinylbeschlagene Stahlblech-Koffergehäuse sieht aus wie Leder und fühlt sich auch so an, aber es ist praktisch unverwüstlich, da es kratzfest und schmutzabweisend ist. Beim AA-100 handelt es sich um den ‚Stereozwilling‘ zum berühmten Heathkit AJ-30 Stereo-AM/FM-Tuner ... sie sehen ähnlich aus und passen zu jeder Raumausstattung in Ihrem Heim.“ [1]
Außerdem gab es in einigen Audio- und Vintage-Foren für Elektronik hilfreiche Ratschläge über Verbesserungen und Restaurierungs-Details. Die endlosen Debatten zu Ersatzröhren, transparentem Klang, Violinen-Solos, Wohlklang-Kondensatoren und sauerstoffarmen Kabeln habe ich ignoriert ;-).

X bedeutet...

Als der AA-100 dann auf meinem Tisch stand, waren die ersten Eindrücke: Das ist kein Leder. Das Ding ist (sau-)schwer! Keine Rückseitenabdeckung (?). Und auch kein Netzschalter (?!). Da hat jemand im Netzteilbereich rumgebastelt (ih!). Das Heathkit/Daystrom-Schild ist auf der falschen Seite. Die Signaleingänge sind unten (!?) und die Lautsprecheranschlüsse werden dem guten Aussehen des Verstärkers nicht gerecht.
Innen drin sah der AA-100 nett aus, mit etwas Staub selbstverständlich (**Bilder 1** und **2**). Alle Lötstellen, Leitungen und Bauteile wirkten auch gut. Die Knöpfe waren eher weiß. Auf den Bildern im Internet waren sie gold-transparent. Merkwürdig. Ich konnte der Versuchung widerstehen, das alte Ding einfach einzuschalten. In diesem Fall fiel das aufgrund der Modifikation hinten am Netzteil etwas leichter (**Bild 3**). Verdächtigerweise war die Netzbuchse „SWITCHED“ gar nicht erst vorhanden. Dafür war die zweipolige Buchse „NORMAL“ von Hand mit „Remote Switch“ beschriftet. Diese beiden Buchsen sind normalerweise dazu gedacht, andere Geräte wie den schönen Tuner AJ 30 zu versorgen. Im Karton war auch noch eine Art Fernbedienung, bei der es sich um ein Stück zweiadriges Kabel mit US-Netzstecker auf der einen und einem Schiebeschalter auf der anderen Seite handelte.
Da Heathkit eine amerikanische Firma ist, sind 117 V bei 60 Hz gefragt. In Europa sind aber eher 230 V bei 50 Hz üblich. Diese Sonderwünsche bezüglich Stromversorgung nerven natürlich. Tief in der Kiste befand sich noch etwas Selbstgebautes: Ein Gehäuse aus Hartplastik (**Bild 4**) mit Schalter, einer Kontrollleuchte und einer IEC-Netzkupplung. Meine Vermutung war, dass hier ein 230/115-V-Trafo drin ist. Leider entpuppte er sich als Spartrafo, der die Netzspannung um 9 V oder 18 V reduzieren konnte. Enttäuschend. Aus guten Gründen sind Liebhaber alter Röhrengeräte vorsichtig, wenn es um die Netzspannung geht. Sie wurde nämlich zwischenzeitlich Stück für Stück von 220 V auf 230 V (Maximum = 240 V) angehoben. Aktuell herrschen im Elektor-Labor 228 V vor.
Meine nächste Vermutung war, dass der Netztrafo des AX-100 zwei 115-V-Wicklungen mit Mittelanzapfung hätte, die für 230-V-Betrieb in Serie geschaltet würden. Doch wieder falsch: Die Inspektion ergab einen simplen Trafo mit nur zwei Drähten auf der Primärseite - genau

Technische Daten:

Nennleistung:	2 x 25 W stereo oder 50 W mono
Musikleistung:	2 x 30 W stereo (0,7 % THD @ 1 kHz) 60 W mono (0,7 % THD @ 1 kHz)
Eingangsempfindlichkeit	(V_{RMS} für 25 W pro Kanal): Mono PHONO* (nur Kanal L): 1,5 mV Stereo PHONO*: 1,5 mV *für magnetische Tonabnehmer
TAPE HEAD:	1,0 mV
TUNER:	0,2 V
AUX 1:	0,2 V
AUX 2:	0,2 V
Eingangsimpedanzen:	PHONO: 47 kΩ, anpassbar an System TAPE HEAD: 470 kΩ TUNER und AUX: je 250 kΩ
Ausgangsimpedanzen:	4, 8 und 16 Ω
Tonband-Ausgang:	ca . 0,5 V max. bei 600-Ω-Quellimpedanz von Kathodenfolger. Min. 150 kΩ
Frequenzgang:	±1 dB, 30 – 15.000 Hz @ 25 W, via AUX
Kanaltrennung:	>42 dB @ 1 kHz
Dämpfung:	15
Harmonische Verzerrungen:	<0,5 % @ 25 W, 1 kHz; <2 % @ 25 W, 30 – 15.000 kHz
Intermodulationsverzerrungen:	<1 % @ 25 W, 60 Hz und 6.000 Hz, 4:1
Brummen und Rauschen:	PHONO: 55 dB TAPE HEAD: 35 dB TUNER und AUX: 70 dB
Entzerrung:	PHONO: RIAA-Kurve TAPE HEAD: NARTB-Wiedergabekurve
Klangeinstellung:	Bass +15/-17 dB; Höhen +12/-20 dB
Röhrenbestückung:	2x EF86, 4x 12AX7, 2x 7199, 4x 7591, 1x GZ34
Abmessungen:	39x11x34 (BxHxT) inkl. Standfüße
Nettogewicht:	13 kg

3


4

5


6

wie im Heathkit-Schaltplan, mit Ausnahme des Netzschalters. Lediglich ein Primäranschluss war statt schwarz eher schwarzgrün. Daher musste es sich um einen anderen Trafo als in der Schaltung handeln! Tatsächlich war seine Bezeichnung „54**X**-89“ (**Bild 5**) und nicht „54-89“. e**X**port? Heathkit, bitte helfen!

Die Verdrahtung des Netzschalters war vorhanden, aber der Schalter nicht. Er lag auch nicht im Karton. Sein viereckiges Montageloch war durch das Heathkit/Daystrom-Schild verdeckt.

An diesem Punkt wurde ich zuversichtlich, dass dieser AA-100 eine 230-V-Ausführung ist und ich den Verstärker an meinen einstellbaren Trafo anschließen kann, um später die Versorgungsspannungen langsam hochzufahren.

Transplantationen...

Beim AA-100 wurde eine Platine mit diskreter Verdrahtung ohne Kabelführungen kombiniert. Nur wenige Bauteile waren freiluftverdrahtet oder auf Lötösen montiert. Bemerkenswert: Zwei Dickfilmschaltungen in den Klangeinstellungen für den linken und rechten Kanal namens „Packaged Electronic Circuit“ (PEC) von Heathkit.

Auch wenn es mir noch nie gefallen hat, wenn man Röhren auf Platinen (speziell die dünnen aus Pertinax) bestückt, so scheint es doch beim AA-100 keine Probleme zu geben – nicht einmal mit den beiden Endröhren, die ziemlich heiß werden. Es wurden nämlich Sockel guter Qualität mit Federkontakten verbaut.

Raymond hatte einige Notizen beigelegt. Laut denen war die Endstufe 1992 mit neuen Kondensatoren und Widerständen überholt worden. Das war vorausschauend! Polypropylen-Kondensatoren ersetzten die leckenden „black bombs“ (alte, vergossene Kondensatoren) und auch die alten, hochohmig gewordenen Kohlenschichtwiderstände erfuhren ein Update. Die originalen Kondensatoren von Heathkit hatten wolkige Bezeichnungen wie PYRAMID, SANGAMO oder MICAMOLD TROPICAP (!) – da sträuben sich die audiophilen Nackenhaare. Beim Austausch blieben wohl etliche Kondensatoren übrig. Ich fand jedenfalls etliche davon im Karton (**Bild 6**). Leider waren einige der abgebildeten neuen Elkos zu groß, um das Gehäuse schließen zu können. Darüber hinaus hatten die blauen Exemplare eine Kapazität von 180 µF - viel zu viel Last für die Gleichrichterröhre GZ34. Dafür war ihre Spannungsfestigkeit mit 450 V zu niedrig.

Bei der 1992er Restaurierung wurden auch Röhren ersetzt. Dabei wurden alle Spannungen gemessen und aufgezeichnet. Die alten Röhren wurden in (unpassende) Schachteln gesteckt (**Bild 7**). Heathkit verwendete neu beschriftete 12AX7-Typen von Mullard (ECC83). Ich habe alle alten Röhren mit meinem uTracer-Röhrentester überprüft. Nur eine Röhre des Typs 7591 war etwas schwach, aber immer noch brauchbar. Ich war neugierig auf die Gleichrichterröhre GZ34 (die original von Philips heute richtig teuer ist). Es gab noch eine andere holländische Spezialität: die EF86! Faktisch waren nur die Treiber und Endröhren US-Typen.

Die Platine mit Vorverstärker und Klangeinstellung werde ich demnächst restaurieren.

Vermutlich verstecken sich da auch noch einige verdächtige Cs und Rs.

Prä-HiFi mit allem Schnickschnack

Die Schaltung von Treiber und Endstufe eines AA-100-Kanals ist in **Bild 8** zu sehen. Die komplette Schaltung findet man an mehr als einer Stelle, darunter *Vintage Radio* [2].

Dieser klassische Gegentaktverstärker wartet mit etwas atypischen Endröhren auf. Die pro Kanal getrennte Einstellung von Höhen und Tiefen ist ebenfalls ungewöhnlich. Außerdem gibt es da noch die ausgefallene Funktion SEPARATION und Anschlüsse für einen CENTER SPEAKER – letztere dienen laut Heathkit zur Vermeidung des „Mittenlochs“, der angeblich einigen Stereo-Quellen anhaften würde. Okay, wenn man die L- und R-Lautsprecher zu weit voneinander aufstellt, hat selbst die „Wall of Sound“ von Phil Spector etwas Unterstützung in der Mitte nötig. Noch was Spezielles: Die Phase des linken Kanals kann „umge-

ESTD 2004

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

dreht“ werden. Der Schalter und die Verkabelung dazu wurden von einem Vorbesitzer entfernt.
Ein Blick auf den Schaltplan: Die beiden 7199 (V7 bzw. V8) arbeiten als Spannungsverstärker und sind direkt mit der „integrierten“ Triode als Phasenumkehrstufe verbunden. Hier werden die Gegentakt-Ansteuersignale für die Endröhren vom Typ 7591 erzeugt (V9/V10 bzw. V11/V12). Ihre halben Beiträge werden im Ausgangstrafo zur Gesamtleistung addiert, an dessen Sekundärseite die Lautsprecher wahlweise in 4, 8 oder 16 Ω angeschlossen werden. Die Röhren werden als normale Pentoden betrieben. Ihre Gitter- und Schirmspannungen sind so gewählt, dass sich die maximale unverzerrte Ausgangsspannung ergibt. Die Gittervorspannung von -16 V wird von einem Halbwellen-Selengleichrichter erzeugt.
Die Gitter-Ableitwiderstände R91/R92 bzw. R85/R86 wurden vielfach wegen zu hoher Werte kritisiert. Das Datenblatt zur 7591 empfiehlt maximal 300 kΩ, aber Heathkit verwendet 470 kΩ. Eine quasi zwingende Änderung ist daher, die Widerstände durch niedrigere Werte von 220 kΩ oder 270 kΩ zu ersetzen. Ich habe auch den Selengleichrichter durch eine Silizium-Diode ersetzt und die zugehörigen Widerstände so gewählt, dass wieder -16 V dabei herauskommen.

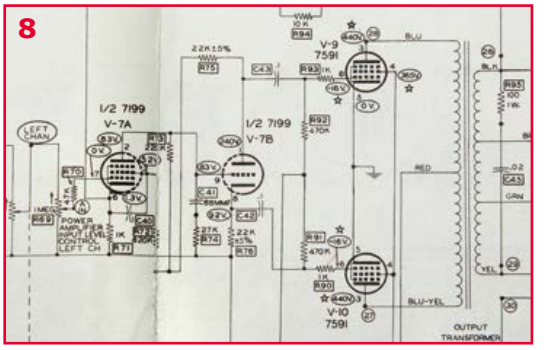
Plaste & Elaste

Zur Kosmetik: Alle Knöpfe der Frontplatte und sogar einige Drähte innen hatten Flecken wie von Schimmel oder Farbspritzern. Am schlimmsten betroffen waren die sechs großen, weiß gewordenen Kunststoffköpfe. Die schwarzen Knöpfe waren mit hellbraunen Flecken übersät. **Bild 9** zeigt rechts das Resultat der schnellen Reinigung eines Knopfpaares nur mit etwas Spülmittel und einer Zahnbürste. Glücklicherweise war der Kunststoff nicht gealtert - ein sehr guter Zustand. Wenn Sie wissen warum, dann schreiben Sie mir.
Die Reinigung der anderen Knöpfe und der Frontplatte wurden zugunsten dieses Artikels verschoben. In *Retronik* werden die Apparate original so gezeigt, wie sie bei uns ankommen.

Doppel-X

Da das Netzteil passte und die Treiber- und Endstufe des Verstärkers schon überholt worden war, konnte ich den AA-100 an meinen Regeltrafo anschließen und die Spannung langsam in 20-V-Schritten verteilt über einen ganzen Tag auf den Normwert hochfahren. Es passierte nichts Schlimmes. Der mickrige Lautsprecher, der beim AA-100 dabei war, kam in den Keller. Stattdessen wurde ein Lautsprecherpaar des Typs Philips 22RH427’s [3] angeschlossen. Dabei handelt es sich um geschlossene Dreiwegeboxen von 1973 mit einem Volumen von 35 Litern und doppeltem 8”-Basslautsprecher. Da dieser Verstärker weit entfernt von seiner 60-Hz-Heimat betrieben wurde, schien mir US-Indie-Sound am besten. Die Stimmen kamen durch ein Mikrofon von 1946 und wurden von einer halbakustischen Gretsch-Gitarre begleitet: *16 Horsepower Live [4].* Im Vergleich zu meinem überholten Philips-Verstärker AG9015 von 1966 klang der Heathkit dunkel und aussagekräftig. Auch wenn er schon ein X hat (54X) braucht dieser AA-100 noch ein weiteres X in Form der Überholung des Vorverstärkers plus einer Reinigung des Kunstleders.

(140333)



Weblinks

[1] Heathkit Museum: www.heathkit-museum.com/hifi/hvmaa-100.shtml
[2] AA-100 Schaltung: www.vintage-radio.info/heathkit/
[3] 22RH427: www.oudio.nl/speakers/22rh427.htm
[4] 16 HorsePower, live @ Montreux 2010: http://youtu.be/G5pV2aQdf3o

Elektor Store

1

Mit Visual Basic in die analoge Welt

VB-Express und die Hardware

Visual Basic zählt nach wie vor zu den sehr weit verbreiteten Programmiersprachen. Seine Beliebtheit resultiert gerade für den Einsteiger aus der schnellen Erlernbarkeit und der einfachen Lesbarkeit des Programmcodes. Dieses Buch ist für Einsteiger in die Programmiersprache Visual Basic.NET gedacht, auch und gerade unter Berücksichtigung der Bedürfnisse von Elektronikern.

287 Seiten (kart.) • ISBN 978-3-89576-270-3
€ 36,80 • CHF 45,95

2

Grundlagen für Visual Basic-Einsteiger und VB6-Umsteiger

VB Express 2010

Dieses Buch unterstützt den Anwender bei den ersten Schritten mit Visual Basic, in dem es sich auf die Werkzeuge der Toolbox und deren Eigenschaften konzentriert, die zum Schreiben praktisch verwertbarer Programme notwendig sind. Zu jedem Thema findet der Leser ausführlich kommentierte Beispielprogramme, die er selbst ausprobieren kann und die sich auf das Mindeste beschränken, was zum Starten der Software notwendig ist.

284 Seiten (kart.) • ISBN 978-3-89576-269-7
€ 34,80 • CHF 43,95

3

45 Experimente mit Hard- und Software für Elektroniker

Raspberry Pi

Dieses Buch beschreibt 45 spannende und interessante Projekte mit Raspberry Pi, wie zum Beispiel ein Wechselblinklicht, eine Motorregelung, Erzeugen und Verarbeiten analoger Signale, ein digitales Thermometer, ein Lichtmesser. Aber auch kompliziertere Projekte wie eine Motor-Geschwindigkeitsregelung, ein Webserver mit CGI (Common Gateway Interface) und Client-Server-Programme werden vorgestellt. Sie können dieses Buch als Projektbuch verwenden und die Projekte nachbauen, um sie dann in der Praxis einzusetzen.

271 Seiten (kart.) • ISBN 978-3-89576-273-4
€ 39,80 • CHF 49,95

4

Arrays und serielle Daten

LabVIEW 2

Der zweite Band der LabVIEW-Lehrbuchreihe beschäftigt sich u.a. mit Arrays, Cluster und den seriellen VISA-Funktionen. Als Erstes werden vier neue zusammengesetzte Datentypen (Enum, Ring, Array, Cluster) vorgestellt und deren Verwendung wird anhand zahlreicher praktischer Beispiele und Übungen erläutert. Danach wird es praktisch: Ein 8051er-Mikrocontrollersystem dient dabei als Datenquelle und -senke für verschiedene LabVIEW-VIs.

248 Seiten (kart.) • ISBN 978-3-89576-274-1
€ 34,80 • CHF 43,95

5

Der professionelle Ratgeber für Funkempfangstechnik

Funkempfänger-kompendium

Wollten Sie schon immer wissen, wie sich die klassische Funkempfängertechnik fortentwickelt hat? Wie funktionieren professionelle Funkempfänger heute und was können sie leisten? Welche Empfangssysteme und Techniken stehen heute zur Verfügung? Möchten Sie

auch ausgefallene Anwendungen von Empfängern kennenlernen und wissen, wie ein Software Defined Radio (Digitalempfänger) nun wirklich funktioniert und was der letzte Stand der entsprechenden Technik kann? In diesem Buch findet man die Antworten!

397 Seiten (geb.) • ISBN 978-3-89576-276-5
€ 49,00 • CHF 61,95

Theorie und Praxis mit Multisim

6 Grundschaltungen der Elektronik

Dieses Buch ist ein Nachschlagewerk über Elektronik mit praxisorientierten Fakten und ausführlichen Erklärungen. Der Autor hat selbst für komplexe Vorgänge oder Formeln praktische kurze Erklärungen und Näherungsrechnungen entwickelt, ohne die Darstellungen zu simplifizieren. Als Ausgangspunkt wurde das Simulationsprogramm Multisim gewählt, das zahlreiche Bauelemente und umfangreiche Messinstrumente zur Verfügung stellt.

360 Seiten (kart.) • ISBN 978-3-89576-286-4
€ 44,00 • CHF 54,95

Schnell und einfach
mit Arduino und Elektor-Shield

7 Mikrocontroller verstehen und anwenden

Mit diesem Buch erweitert der Anfänger auf dem Gebiet der Mikrocontroller, der Arduino-User bzw. -Enthusiast seine Mikrocontroller-Kenntnisse auf Grund eigener Erfahrungen und Erfolgserlebnisse und wird dazu noch ganz nebenbei in die Welt des Arduino und seiner Entwicklungsumgebung eingeführt.

392 Seiten (kart.) • ISBN 978-3-89576-296-3
€ 42,00 • CHF 52,95

Schaltungsprojekte für Profis

8 Arduino

Für den großen Erfolg der Arduino-Plattform lassen sich zwei Ursachen finden. Zum einen wird durch das fertige Board der Einstieg in die Hardware enorm erleichtert; der zweite Erfolgsfaktor ist die kostenlos verfügbare Programmieroberfläche. Unterstützt wird der Arduino-Anwender durch eine Fülle von Software-Bibliotheken. Die täglich wachsende Flut von Libraries stellt den Einsteiger vor erste Probleme. Nach einfachen Einführungsbeispielen ist der weitere Weg nicht mehr klar erkennbar, weil oft detaillierte Projektbeschreibungen fehlen. Hier setzt dieses Buch an. Systematisch werden Projekte vorgestellt, die in verschiedene Themengebiete einführen. Dabei wird neben den erforderlichen theoretischen Grundlagen stets größter Wert auf eine praxisorientierte Ausrichtung gelegt.

270 Seiten (kart.) • ISBN 978-3-89576-257-4
€ 39,80 • CHF 49,95

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

www.elektor.de

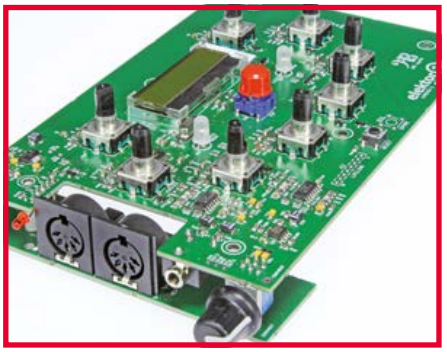
Elektor-Verlag GmbH
Süsterfeldstr. 25
52072 Aachen
Tel. +49 (0)241 88 909-0
Fax +49 (0)241 88 909-77
E-Mail: bestellung@elektor.de

Nächsten Monat in Elektor

Die nächste Elektor-Ausgabe ist eine Doppelausgabe mit extra vielen Projekten, Ideen und Tipps!

Ebenso wie zu Beginn dieses Jahres wird die erste Ausgabe im kommenden Jahr eine doppelte Ausgabe mit extra-starkem Inhalt sein. Wir haben einen spannenden Mix aus Kurzbeiträgen, anspruchsvollen Projekten und aktueller Information vorbereitet. Die Mikrocontroller beherrschen nicht allein das Feld, auch für die Mess- und Regelungstechnik ist Platz, und die analoge Schaltungstechnik kommt ebenfalls nicht zu kurz. Außer dem J:B-Synthesizer und dem Platino-Signalgenerator sind folgende Themen geplant:

- Einfacher FM-Empfänger
- LED-DC/DC-Booster
- Breakout-Board für Feuchte-Sensor
- True-RMS-Vorsatzmodul für DMM
- CMOS IR-Modulator
- ZigBee T-Board
- Uhr mit DCF-Visualisierung
- ELPP-Multitester



J:B-Synthesizer

Nach Jahren der Pause stellt Elektor einen neuen Musik-Synthesizer vor. Die analogen Gerätschaften, die seinerzeit Wandschränke vereinnahmten, sind längst Geschichte. Heute sind kompakte Synthesizer gefragt, die einfach bedienbar sind. Basis unseres neuen Entwurfs war das J2B-Board mit ARM-Prozessor (September 2011), es wurde gründlich überarbeitet. Zum J:B-Synthesizer gehört Software, die natürlich komplett open-source ist. Und so viel sei schon verraten: An Rechenleistung mangelt es nicht!

Platino-Signalgenerator

Auf der Basis des Platino-Boards vom Oktober 2011 haben wir einen Signalgenerator entworfen, der für viele Zwecke geeignet ist. Nicht nur die Standard-Signalformen Sinus, Rechteck, Dreieck, Sägezahn und Einzelimpuls werden erzeugt. Der Anwender kann auch Signalformen nach Bedarf programmieren und diese Signale mit anderen Signalen frequenzmodulieren. Die Bedienung ist unkompliziert und übersichtlich, dazu tragen ein Drehencoder und ein vierzeiliges LC-Display bei.

Änderungen vorbehalten! Elektor Januar/Februar 2015 erscheint am 07. Januar 2015. Verkaufsstellen findet man unter: www.pressekaufen.de.

elektor PCB Service
powered by Eurocircuits

PCBs
Multilayer
Prototypes
Small series

Professional Quality
Trusted Service
Secure Ordering

Elektor PCB Service at a glance:

- 4 Targeted pooling services and 1 non-pooling service
- Free online PCB data verification service
- Online price calculator available
- No minimum order value
- No film charges or start-up charges

Delivery from 2 working days

Further information and ordering at www.elektorpcbservice.com

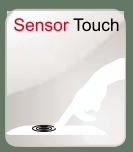
Integrierte 3D-Gesten- und Positionserfassung

mit Microchips GestIC®-Technologie

Microchips patentierte GestIC®-3D-Technologie ist eine Einchip-Lösung für die Gestenerkennung in Echtzeit. Der MGC3130 ermöglicht in jedem Produkt zu sehr geringen Kosten den nächsten Durchbruch bei der Entwicklung von Benutzerschnittstellen.

MGC3130

- Der MGC3130 läuft mit der Colibri Gesture Suite und sorgt für eine sichere Erkennung komplexer 3D-Gesten ohne Host-Datenverarbeitung
- Der MGC3130 liefert die Ergebnisse (erkannte Gesten / XYZ-Position) über eine I²C™-Bus-Schnittstelle oder GPIOs an den Host



microchip DIRECT
www.microchipdirect.com

MICROCHIP

microchip.com/gestic