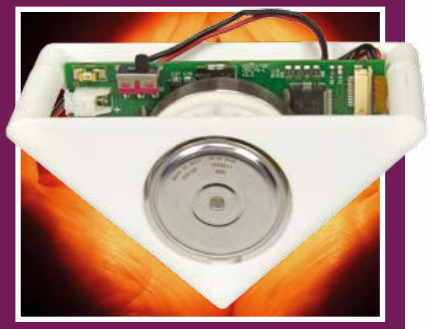
April 2016 | Nr. 544
www.elektormagazine.de

elektor

LEARN | DESIGN | SHARE

ESP8266
auf dem
Android I/O-Board



i-Pendel
Balanciert und
steht selbst auf

LEDitron
LED-Filamente als
7-Segment-Display

Programmer und Debugger

Entwicklungstools für alle

In dieser Ausgabe:
4 Labor-Projekte,
3 Leser-Projekte,
Tipps und Tricks,
Neues aus dem Labor
und mehr ...

Windows auf dem Raspi: SPI und I²C • **Opamp-Experimentier-Kit** • **ESP8266 im Einsatz ohne MCU** • **Laufschrift für Arduino:** Mit 16 x 16 LEDs • **Network Connected Signal Analyzer:** Software & Mathematik • **Embedded World Rundgang** • **Bemerkenswerte Bauteile:** Quecksilberrelais • **Web-Recherche:** Know-How aus dem Netz • **Neues von Elektor-Labs.com**

G3078 E

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

4 198630 307906 04

Professionelle Qualität zu attraktiven Preisen!

reichelt.de elektronik

- mehr als 50.000 Artikel aus Elektronik & IT
- über 1.400 Seiten
- mehr als 5.000 Neuheiten

Der Neue ist da!

Katalog 06|2016

gleich online blättern

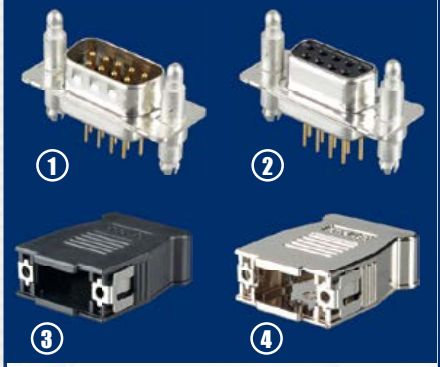
http://rch.lt/15-1

kostenlos anfordern!

D-Sub-Steckverbinder mit SnapLock

- kein Anschrauben nötig
- fester Halt wie bei einer Schraubbefestigung

CONEC



1 D-Sub-Stecker

DSUB ST2093	9-pol	3,25
DSUB ST2113	25-pol	5,20
DSUB ST2123	37-pol	6,90
DSUB ST2133	50-pol	8,40

2 D-Sub-Buchse

DSUB BU2143	9-pol	3,60
DSUB BU2153	15-pol	4,95
DSUB BU2163	25-pol	6,35
DSUB BU2173	37-pol	8,59
DSUB BU2183	50-pol	9,95

3 D-Sub-Haube mit SnapLock, Kunststoff

KAPPE 1810	für 9-pol	2,20
KAPPE 1820	für 15-pol	2,35
KAPPE 1830	für 25-pol	2,50
KAPPE 1840	für 37-pol	2,55
KAPPE 1850	für 50-pol	2,65

4 D-Sub-Haube mit SnapLock, Metall

KAPPE 1750	für 9-pol	3,45
KAPPE 1760	für 15-pol	3,95
KAPPE 1770	für 25-pol	4,45
KAPPE 1780	für 37-pol	4,95

opticalCON, robustes LWL Steckersystem

opticalCON ist ein robustes Glasfasersteckverbindersystem für vielfältigste Anwendungen. Dank des geringen Gewichts, dem kompakten Design, einem robusten Gehäuse und hohen Steckzyklen bietet dieses Steckersystem eine sichere und kostengünstige Verbindung.

NEUTRIK

2-kanal LWL-Verbindungsleitung, 3 m

- ☐ wasserdicht gemäß IP65
- ☐ zuverlässige Push-Pull Verriegelung

145,20 OCON 3

opticalCON DUO Einbaubuchse, schwarz

- ☐ wasserdicht gemäß IP65
- ☐ Schutzklappe mit Silikondichtung schützt die Optische Verbindung

68,95 OCON BU BK

NEUTRIK etherCON

Datenraten bis zu 10 GBit/s

Schwarze CAT6 Einbaubuchse in D-Form

- ☐ CAT6 fähig
- ☐ IP65
- ☐ Push-Pull Verriegelung

21,60 EBU CAT6 BK

Einpolige Federkraftklemmen von MetzConnect

METZ CONNECT

RIA CONNECT BTR NETCOM MCQ TECH

- ☐ lötbar
- ☐ mit praktischem Fingerdrücker

je 0,31

SR 901 CW | SR 901 YE | SR 901 GR | SR 901 OR | SR 901 RD | SR 901 BK

JETZT ABONNIEREN!

Newsletter

Abbonieren und profitieren

- ✓ Neuheiten
- ✓ beste Angebote
- ✓ Preisreduzierungen

Jetzt bestellen!

www.reichelt.de

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

reichelt.de | @reichelt_el | google.com/+reichelt

Tagespreise! Preisstand: 25.02.2016

Preise in € inkl. gesetzl. MwSt., zzgl. Versandkosten
reichelt elektronik, Elektronikring 1, 26452 Sande (D)

Impressum

47. Jahrgang, Nr. 544
April 2016
Erscheinungsweise: 10 x jährlich
(inkl. Doppelhefte Januar/Februar und Juli/August)

Verlag
Elektor-Verlag GmbH
Kackertstraße 10
52072 Aachen
Tel. 02 41/955 09 190
Fax 02 41/955 09 013

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 41/955 09 174 / Fax 02 41/955 09 013
Mobil: +31 6 510 530 39
E-Mail: margriet.debeij@eimworld.com

Julia Grotenrath
Tel. 02 41/955 09 177 / Fax 02 41/955 09 013
E-Mail: julia.grotenrath@eimworld.com

Es gilt die Anzeigenpreisliste Nr. 46 ab 01.01.2016

Distribution:
IPS Pressevertrieb GmbH
Postfach 12 11, 53334 Meckenheim
Tel. 0 22 25/88 01-0
Fax 0 22 25/88 01-199

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

Und wie viele haben Sie?

Und da ist er, der Raspberry Pi 3, leistungsfähiger als die Modelle vor ihm und dazu noch mit On-board-WLAN ausgestattet. Ich könnte wetten, dass schon eine abgespeckte WLAN-Version in Vorbereitung ist, zu einem noch günstigeren Preis. Das Ganze zielt klar auf das Internet of Things, bei dem viele Dinge unseres Lebens über das Netz abfrag- und steuerbar gemacht werden sollen. Und bei den stetig sinkenden Hardwarepreisen werden immer mehr User die Einplatinen-Computer auch „in Stückzahlen“ einsetzen. Zu Beginn dieser Revolution war es Usus, einen einzigen Raspberry Pi zu verwenden, als Media-Center oder Zentrale für ein vernetztes Projekt. Heutzutage geht der Trend zum Dritt- oder Viert-Raspi; und damit ist das Ende der Fahnenstange noch nicht erreicht. Es wird sich in Zukunft kaum mehr lohnen, auf so einem Kerl heute das eine und morgen das andere Programm laufen zu lassen. Das wissen auch die Distributoren. Bisher wurden die Einplatinen-Rechner meist nackt oder mit einer passenden Version eines Betriebssystems ausgeliefert. Nun wird es immer häufiger auch Angebote geben, bei denen auch eine bestimmte Anwendungssoftware schon installiert ist. Die ehemals als billiger Computer-Ersatz gedachten Boards werden damit endgültig zu Geräte-Modulen. In naher Zukunft könnte es Kits geben, die ein Dutzend Raspis oder Arduinos (oder beides) enthalten, fertig konfiguriert mit IoT-Protokollen wie MQTT. DIYer, die ein Smart Home realisieren wollen, müssen dann nur noch Sensoren aufstecken und alles über eine (hoffentlich komfortable) Oberfläche konfigurieren. Und auch in immer mehr Fertiggeräten wird man – wenn man so neugierig ist wie wir – ein Controllerboard finden, das einem sehr, sehr bekannt vorkommt.
Wir bleiben dran!

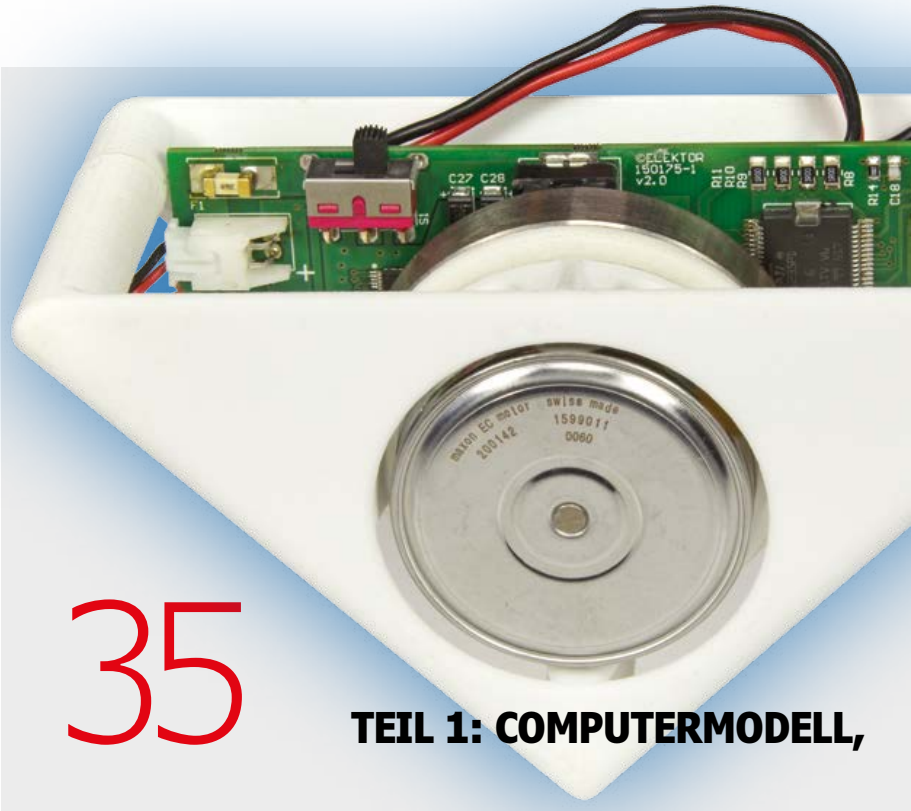
Jens Nickel
Chefredakteur Elektor

Inhalt

47. Jahrgang – Nr. 544 April 2016

LEARN DESIGN SHARE

LEARN DESIGN SHARE



35

TEIL 1: COMPUTERMODELL,

ESP8266 AUF DEM ANDROID I/O-BOARD

FLASHEN NEUER FIRMWARE

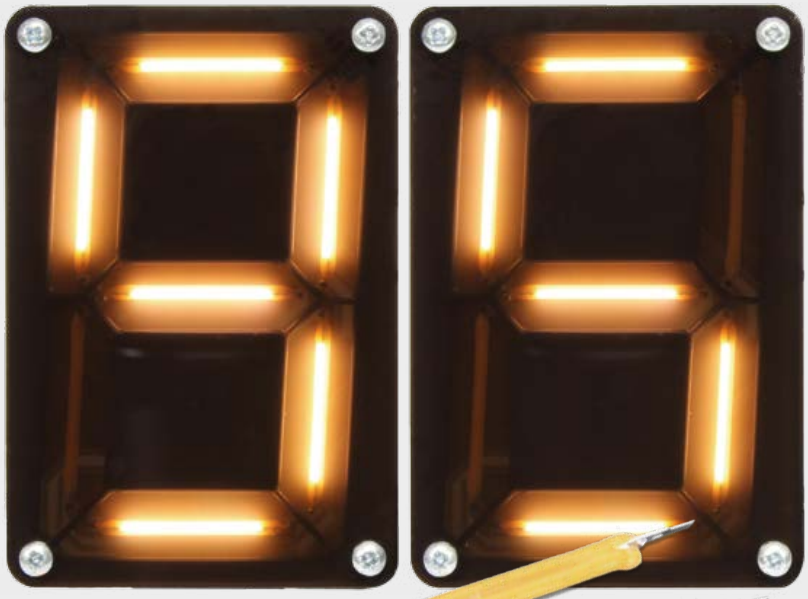
Mit dem Android I/O-Board vom September 2015 lassen sich über Android-Smartphones oder -Tablets drahtlos komplexe Funktionen steuern. Für die Funkverbindung stehen alternativ sieben Module zur Wahl, unter ihnen das verbreitete, kostengünstige WLAN-Modul ESP8266. Allerdings ist der Betrieb nur mit einer daran angepassten Firmware möglich. Hier wird gezeigt, welche Schritte nötig sind.

42

Das inverse Pendel, kurz i-Pendel, kann sich nicht nur selbst im Gleichgewicht halten, sondern sogar selbstständig aufrichten. Wenn man ein solches Projekt beginnt, kann man sich erstaunten Kopfschüttelns sicher sein, genauso aber der begeisterten Reaktionen, wenn man es erfolgreich abschließt und dem staunenden Publikum demonstriert.

i-Pendel

STEUERUNGSREGELN UND KALMAN-FILTER



29

LEDitron

LED-FILAMENTE ALS 7-SEGMENT-DISPLAY

Die alte Glühbirne ist tot, es lebe die LED-Lampe! Die LEDitron verwendet so genannte LED-Filamente, um genau wie bei einer Numitron-Röhre ein 7-Segment-Display zu realisieren. Aber viel energiesparender!

elektor magazine

LEARN | DESIGN | SHARE

Vorschau

Modulares Netzteil

Beim DDS-Generator aus dem letzten Novemberheft wurde ein Netzteil verwendet, bei dem Trafo und Netzfilter viel Platz einnehmen. Dass es kompakter und effizienter geht, beweisen wir mit diesem Projekt. Zum Einsatz kommt hier ein DC/DC-Konvertermodul von Recom. Die Stromversorgung lässt sich auch für viele andere Projekte verwenden beziehungsweise anpassen.

Telepräsenz-Roboter

Fans der Serie „The Big Bang Theory“ kennen den „Sheldonbot“, den Telepräsenz-Roboter des Ober-Nerds Sheldon. Glauben Sie mir: Unserer ist viel cooler, denn er balanciert auch noch auf einer Achse. Erdacht wurde das Ganze von Wheelie-Macher Chris Krohne. Im nächsten Heft erfahren Sie von ihm, wie ein Tablet, ein steuerndes Smartphone und die Elektronik im Fuß des Roboters trickreich zusammenspielen.

Weitere geplante Artikel:

- Nixie-Uhr mit 6 Ziffern
- Entfernungsmessung mit Raspi
- Grundlagen Oszilloskope
- Das Herz der Maker-Szene

Und vieles mehr!

Änderungen vorbehalten.

Elektor Mai erscheint am 20. April 2016.

Verkaufsstellen findet man unter www.pressekaufen.de.

Die Elektor-Community

LEARN | DESIGN | SHARE

Elektor durchbricht die Schranken einer Zeitschrift und wird zur Community aktiver E-Ingenieure - vom Anfänger bis zum Profi – begierig, überraschende Elektronik zu lernen, zu entwickeln, zu teilen.

Elektor-Shop: 24 Stunden an 7 Tagen der Woche für jeden Elektroniker geöffnet! Dauerhafter Rabatt von 10% für alle GOLD- und GREEN-Mitglieder. **www.elektor.de**

Elektor-Zeitschrift: 10 Ausgaben pro Jahr voll gepackt mit Elektronik-Projekten, Artikeln, Besprechungen, Tipps und Tricks. **www.elektormagazine.de**

Elektor-Platinen-Service: Bestellung von Platinen als Einzelstück oder Kleinserie. **www.elektorpcbservice.de**

Elektor wöchentlich & papierlos: Wöchentlicher digitaler Newsletter. Kostenlos und aktuell. **www.elektor.de/newsletter**

Elektor Academy: Webinare, Seminare, Präsentationen, Workshops, DVDs und mehr = praxisorientiertes Lernen. **www.elektor-academy.com**

Elektor-Bücher & DVDs: Arduino, Raspberry Pi, Mikrocontroller und vieles andere mehr. Im Online-Shop mit 10% Rabatt für Mitglieder! **www.elektor.de/bucher**

Elektor.TV: Reviews, Eindrücke, Unboxings und persönliche Journale. Anschauen heißt Erfahrung sammeln. **www.elektor.tv**

Elektor-Labs: Eigene Projekte vorstellen – von Anderen lernen – Anderen helfen und mit Anderen teilen. Elektor macht mit und testet Ihre Ideen! **www.elektor-labs.com**

Treten Sie dem weltweit größten Elektroniker-Netzwerk bei!

GREEN 1,78 €/Woche

- ✔ Zugang zum Elektor-Archiv
- ✔ 10% Rabatt auf Shop-Produkte
- ✔ 10x Elektor jährlich (Digital)
- ✘ 10x Elektor jährlich (Print)
- ✔ Exklusive Top-Angebote
- ✔ Zugang zu elektorlabs
- ✔ Elektor Jahrgangs-DVD

Unsere Empfehlung

www.elektor.de/green-mitglied

GOLD 2,45 €/Woche

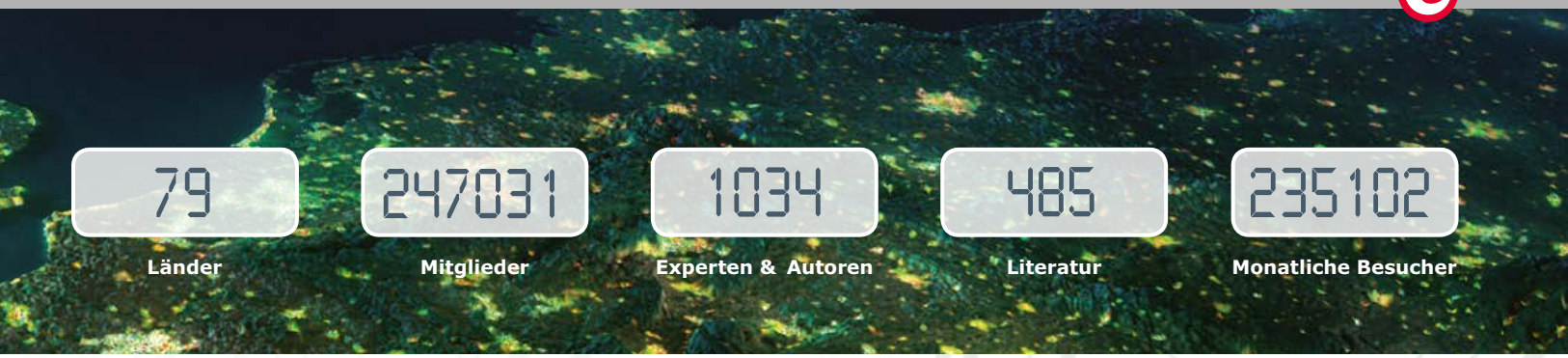
- ✔ Zugang zum Elektor-Archiv
- ✔ 10% Rabatt auf Shop-Produkte
- ✔ 10x Elektor jährlich (Digital)
- ✔ 10x Elektor jährlich (Print)
- ✔ Exklusive Top-Angebote
- ✔ Zugang zu elektorlabs
- ✔ Elektor Jahrgangs-DVD

www.elektor.de/gold-mitglied

GRATIS

- ✘ Zugang zum Elektor-Archiv
- ✘ 10% Rabatt auf Shop-Produkte
- ✘ 10x Elektor jährlich (Digital)
- ✘ 10x Elektor jährlich (Print)
- ✔ Exklusive Top-Angebote
- ✘ Zugang zu elektorlabs
- ✘ Elektor Jahrgangs-DVD

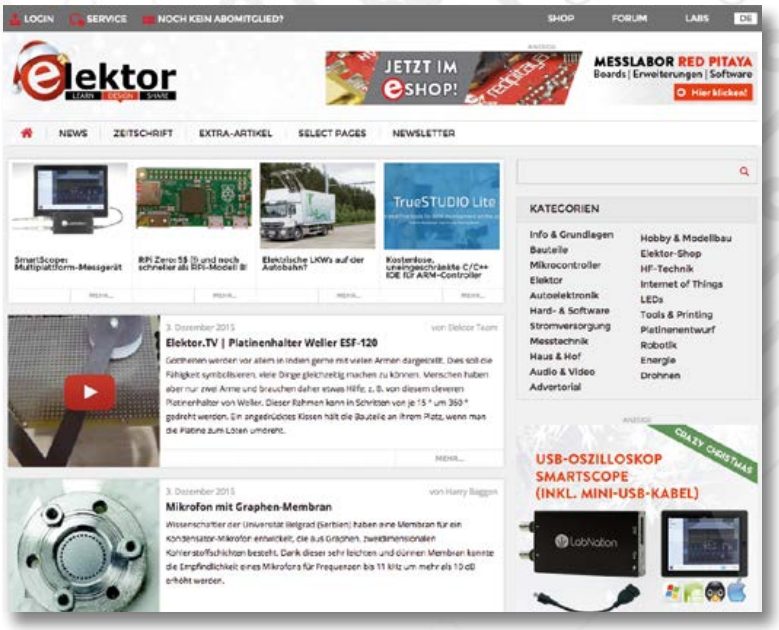
www.elektor.de/newsletter



www.elektormagazine.de

Eine ganze Welt mit Elektronik-News

Elektroniker entdecken eine ganze Welt mit Projekten, News, Videos und mehr auf unserer neu überarbeiteten Website. Rechts oben kann man einfach die Sprache auswählen. Zur Wahl stehen Deutsch, Englisch, Französisch und Niederländisch. Die intelligenten Such-Tools erleichtern das Auffinden von Artikeln und weiterführender Information. Registrieren Sie sich als GREEN- oder GOLD-Mitglied; mit Ihrem persönlichen Login haben Sie vollen Zugriff auf den Online-Shop inklusive vieler Extras. Hier können Sie auch Ihren Account überarbeiten – und das gilt auch für Ihre Mitgliedschaft, das gedruckte Heft und den Newsletter *Elektor*.



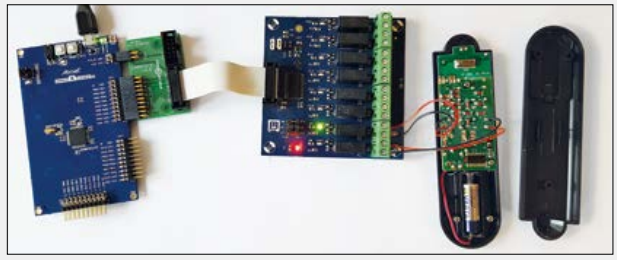
elektor: wöchentlich & papierlos

Ein wöchentlicher Newsletter vollgepackt mit Information

Mehr als 120.000 Elektroniker haben bereits ein Abonnement unseres kostenlosen *Elektor-Newsletters*. Jede Woche gibt es hier News, Tipps, Trends und mehr direkt in ihrem digitalen Briefkasten. Außerdem erhalten Sie so Zugriff auf weitere exklusive Projekte, Spezialangebote und Rabatte für den Online-Shop.
Registrieren Sie sich noch heute:
www.elektor.de/newsletter

Willkommen bei LEARN

Learning by Doing ist ein Prinzip, das ich auf meinem Weg in das Internet of Things gerne beherzige – und Sie können mich begleiten unter www.elektormagazine.de. In Folge 5 meines Blogs habe ich zum ersten Mal ein funktionierendes IoT-Steuerprojekt realisiert. Von einem PC irgendwo auf der Welt kann ich nun die Lampe in meinem Home-Office ein- und ausschalten. Mit Hilfe einer frei verfügbaren .NET-Library habe ich einen kleinen MQTT-Client geschrieben, der Nachrichten senden und empfangen kann. Als Relaisstation im Netz wird dabei der Mosquitto-Testserver (http://test.mosquitto.org) verwendet. Um meine Lampe vom empfangenden PC aus drahtlos anzusteuern, kam eine noch recht grobe Bastelei zum Einsatz. Die Kommandos wurden vom PC über einen virtuellen COM-Port zum bekannten SAM-D20-Board mit Relais-Erweiterung weitergesendet. An die Relais habe ich eine gehackte Funkfernbedienung aus dem Baumarkt angeschlossen, die wiederum eine Funksteckdose mit meiner Lampe schaltet. Immerhin: Es funktionierte! Und dank des MQTT-Protokolls und des Servers im Netz musste ich mir keine Gedanken um eine Router-Konfiguration usw. machen, was bei einem eigenen Webserver nötig gewesen wäre. In den nächsten Folgen optimierte ich die Lösung. Leider fehlt es immer noch an der nötigen „Security“: Meine Lampe kann jedermann schalten, der das MQTT-Topic und die Kommandos kennt… Doch bevor Sie es versuchen: Die Lampe habe ich einstweilen wieder vom Netz genommen… ;-).



Der ESP8266

… scheint sich so langsam zum De-facto-Standard zu entwickeln, wenn man einen Mikrocontroller in ein WLAN-Netzwerk einbinden will. Das Ganze erinnert mich inzwischen an die FT232-Chips von FTDI. Viele Leser und auch die Entwickler in unserem Labor greifen fast automatisch danach, wenn ein Board mit einem USB-Interface zur PC-Kommunikation ausgestattet werden muss. Für eigene Experimente habe ich mir jetzt eines der „Pretzel“-Boards gesichert, die wir auch in unserem Webshop anbieten (www.elektor.de/pretzelboard). Das Modul enthält schon einen ATmega328, einen ESP8266, die Stromversorgung und weitere nötige Beschaltung. Da muss ich nichts löten, keine Käbelchen selbst konfigurieren oder die Kollegen im Labor nach Bauteilen fragen, sondern kann gleich mit dem Probieren und Programmieren anfangen. Ideal also für mich und vielleicht auch für Sie?

Programmer

und Debugger

Wie in der letzten Ausgabe finden Sie auch in diesem Heft einen kleinen Produktüberblick aus dem Embedded-Bereich; diesmal geht es um Programmer und Debugger. Abermals geschrieben wurde der Artikel vom Controller-Enthusiasten Viacheslav Gromov, der einer unserer jüngsten Autoren und trotzdem schon ein erfahrener Kenner der Szene ist. Gerne habe ich ihm daher auch dieses Jahr wieder den Auftrag gegeben, sich für Elektor auf der „Embedded World“ nach neuen Controllern und Boards umzuschauen.

(150731)

Quecksilberrelais

Bemerkenswerte Bauteile

Von **David Ashton** (Australien)

In früheren Zeiten wurde Quecksilber als Kontaktmedium in einigen Relais verwendet. Die meisten alten Hasen wie ich werden sich an die Quecksilberfilmrelais erinnern, die in Telegrafie-Geräten üblich waren. Ziemlich normale Relaiskontakte wurden mit einem Film aus flüssigem Quecksilber beschichtet, damit sich der Kontaktwiderstand durch die größere effektive Kontaktfläche verringerte. Auch das Kontaktprellen wurde weitgehend eliminiert, da, sobald der anfängliche Kontakt hergestellt war, die Oberflächenspannung des Quecksilbers die Verbindung aufrechterhielt, solange die Kontakte nicht zu weit voneinander abprallten. Das bedeutete, dass die Relais klein und schnell sein konnten, schnell genug für Fernschreiber, die mit bis zu 100 Baud (= Hertz in diesem Fall) arbeiteten. Die Relais wurden in regelmäßigen Abständen geschüttelt, um die Kontakte wieder mit Quecksilber zu benetzen, außerdem mussten sie auf eine bestimmte Weise montiert werden, um zu verhindern, dass das überschüssige Quecksilber Kurzschlüsse verursachte. Diese Relais wurden in der Regel entsprechend beschriftet (anders als in **Bild 1**) und wegen des giftigen Inhalts auch mit Warnhinweisen versehen, sie keinesfalls zu öffnen. Unser Retronik-Redakteur Jan hatte einen Einschub für ein Tektronix-Oszilloskop aus dem Jahr 1959, das ein mit Quecksilber benetztes Relais verwendet, um Impulse mit einer Anstiegszeit von 4 ns zu generieren. Ich will Jan hier die Show nicht stehlen, Sie können darüber in einer der nächsten Retronik-Artikel lesen.

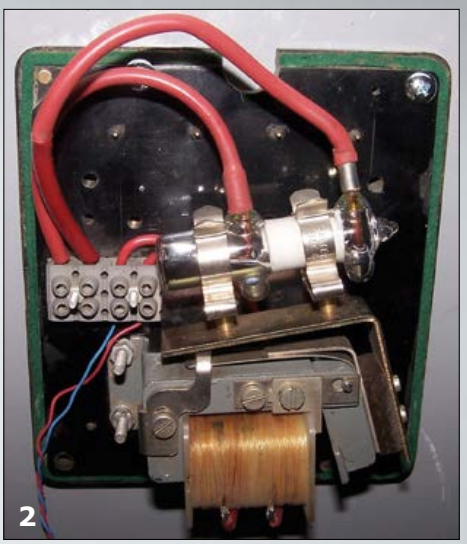
Vor kurzem fand ich eine dieser alten Schönheiten (**Bild 2**), die aus einem ziemlich normalen Relaismechanismus mit einer 240-VAC Spule mit Anker besteht, aber die Arbeitskontakte stecken in einem Röhrchen, in dem sich etwas Quecksilber befindet. Die Kontakte durchmessen etwa 2 mm und sind 6 mm lang. Wenn das Relais betätigt wird, wird der Anker gekippt, das Quecksilber fließt nach rechts und verbindet die beiden Kontakte (**Bild 3**). Wird das Relais wieder gelöst, fließt das Quecksilber zurück in das Reservoir auf der linken Seite. Es reicht nicht mehr aus, um beide Kontakte zu bedecken und elektrisch zu verbinden. Schalter und Relaisspule werden an die Lüsterklemmen mit flexiblen mehradrigen Drähten angeschlossen. Im Hals des Glasrohrs steckt eine kleinere Röhre, die wie Keramikmaterial aussieht. Sie soll den Durchmesser verringern, um sicherzustellen, dass das Quecksilber wirklich nur dann für eine Verbindung sorgt, wenn das Relais betätigt wird. Es steckt übrigens nicht wenig Quecksilber in dem Relais, ich würde die Menge auf etwa 1 cm^3 schätzen.

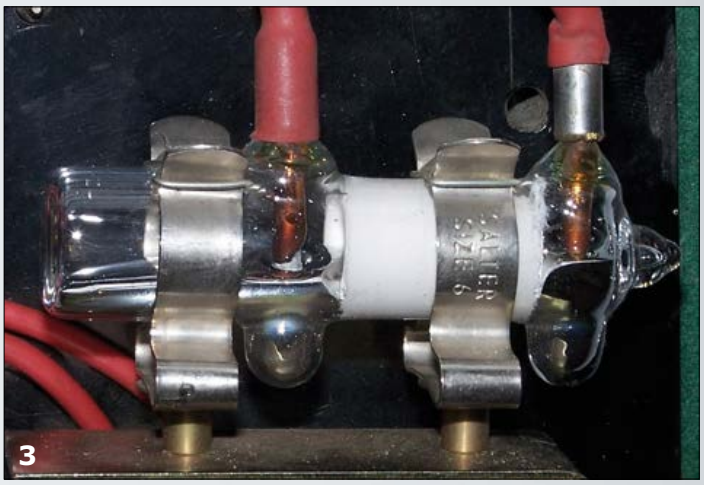
Ich habe über andere Quecksilberrelais gelesen, bei denen ein Eisenkügelchen magnetisch aus einem Quecksilberpool gezogen wird. So wird die Füllhöhe in dem Gefäß geändert, so dass die beiden Kontakte Kontakt miteinander haben oder nicht.

Quecksilber-Relais und -Schalter sind heutzutage nicht mehr oft zu finden. Heutige Energiesparlampen enthalten immer noch Quecksilber, aber das ist eine andere Geschichte. Quecksilber wurde auch weit verbreitet in Neigungsschaltern eingesetzt. Man kann solche gut abgedichteten Schalter auch heute noch kaufen. Ich denke zurück, als Kind mit dem Quecksilber aus einem zerbrochenen Thermometer gespielt zu haben. Mein Quecksilberrelais erinnert mich daran, was für eine faszinierende Substanz dieses silbrige flüssige Metall ist. Und auch an ihren schlechten Ruf, weil sie viel zu oft in die Umwelt gelangt. ⏮

(150672)

1


2


3

Steuern Sie weitere Bemerkenswerte Bauteile hinzu: Mailen Sie an neil@gruending.net.

Programmer und Entwicklungstools für Einsteiger

Von **Viacheslav Gromov**

Wie Sie in der letzten Ausgabe nachlesen konnten, gibt es für viele Mikrocontroller Entwicklungsboards mit integriertem Debugger. Unabhängiger ist man mit einem eigenen Programmer/Debugger, mit dem man Software vom Computer auf die gewünschte MCU übertragen und eine Fehlersuche oder Verbesserungen am Code vornehmen kann.

Für nahezu jede Mikrocontrollerfamilie gibt es inzwischen mindestens einen Debugger beziehungsweise Programmer vom Halbleiterhersteller selbst, aber auch zahlreiche universelle Entwicklungstools von externen und unabhängigen Herstellern. Ein Debugger hat im Vergleich zum „normalen“ Programmer viele Zusatzfunktionen. Vor allem erlaubt er dem Entwickler entweder während der Laufzeit oder beim Anhalten des Programms an einer gewünschten Stelle (Breakpoint) einen Blick in die wichtigsten Speicheradressen und Register. Meist kann man auch ins Programm oder den Speicher aktiv eingreifen, sowie das Programm Befehl für Befehl durchlaufen lassen.

Über diese Basisfunktionen hinaus denken sich die Hersteller immer mehr Debugging-Funktionen aus, etwa verschiedenste Trace-Funktionen, mit denen während der Laufzeit bestimmte Informationen (vom Stromverbrauch bis zu den Speicher-Daten) gespeichert werden, ohne das Programm zu unterbrechen, da nur so möglichst realitätsnahe Bedingungen herrschen.

Bei einem reinen Programmer sind diese Fehlersuch-Funktionen nicht vorhanden. Programmiergeräte sind nur zum (schnellen und einfachen) Beschreiben und Auslesen der MCU-Speicher gedacht. In der Entwicklungsphase kann man Programmer nur zur Fehlersuche einsetzen, indem man die Software auf den Mikrocontroller überträgt, ihre Funktionen von außen beobachtet und das Programm entsprechend anpasst. Wenn die Entwicklungsphase dann zu Ende ist, kann ein Programmer die Software (in Form einer Datei) besser auf den Controller übertragen als ein Debugger, weil er einfach handlicher, günstiger und meist auch schneller ist.

Natürlich muss auch ein entsprechendes Debug-Tool auf dem Rechner laufen. Wenn Sie einen Debugger direkt vom Hersteller nutzen, ist so gut wie immer das Debug-Tool in der passenden IDE integriert, sodass Sie stets innerhalb der Entwicklungsumgebung sofort in den Debug-Modus schalten können. Das funktioniert natürlich erst, wenn die Debug-Einstellungen richtig sind und zuvor die richtigen Treiber installiert wurden. An diesem Punkt bleiben viele Anfänger hängen, weil die Einstellungen doch etwas umfangreicher sind, als man auf den ersten Blick denkt.

Nachfolgend sollten mehrere Debugger und Programmer verschiedener Firmen vorgestellt werden. Zuerst werden in aller Kürze und ohne Anspruch auf Vollständigkeit die von den Halbleiterherstellern selbst (oder indirekt über externe Firmen) hergestellten Debugger beschrieben, anschließend die universellen Debugger und Programmer von unabhängigen Firmen.

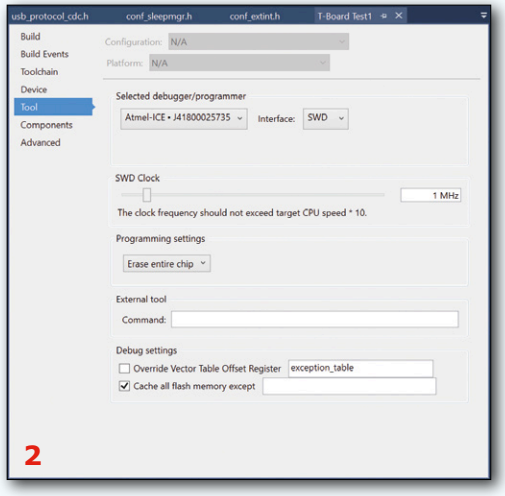
Debugger

Atmel

Atmel ICE [1] (**Bild 1**) ist in der Lage, zwei sehr bekannte und verbreitete Mikrocontrollerfamilien Atmels „in-system“ zu debuggen und zu programmieren: die 32-bit-ARM-Cortex-M-SAM- und die 8-Bit-AVR-Familie. Atmel ICE ist in unterschiedlichen Kits erhältlich. Das günstigste ist die nackte Platine ohne Gehäuse und Kabel für etwas über 40 €. Das Basic-Kit enthält den Debugger samt Gehäuse sowie ein USB- und ein Flachbandkabel zur Programmierung, kostet aber über 60 €. Im größten Kit sind noch ein zusätzliches Flachbandkabel und eine kleine Adapterplatine für die Programmieranschlüsse enthalten, mit etwas mehr als 110 € ist dieses Kit aber deutlich teurer. Selbstverständlich lassen sich Gehäuse, Flachbandkabel und Adapter-Board auch nachrüsten, wenn man mit einer preisgünstigen Lösung eingestiegen ist.

1

Atmel ICE hat abgesehen von dem USB-2.0-Anschluss für den Computer zwei Programmieranschlüsse für die AVR- und für die SAM-Familie. Der Anschluss für die AVR-Familie besitzt die Interfaces aWire, debugWire, ein **P**rogram and **D**ebug **I**nterface (PDI), ein **T**iny **P**rogramming **I**nterface (TPI) und schließlich eine SPI-Schnittstelle (ISP). Beide Anschlüsse verfügen unabhängig von allem anderen über **S**erial **W**ire **D**ebug (SWD) und die **J**oint **T**est **A**ction **G**roup (JTAG) Schnittstelle, wobei sich die Belegung der beiden Anschlüsse voneinander unterscheidet. Je nach Programmierschnittstelle ist Atmel ICE in der Lage, mit (Bus-)Taktfrequenzen von bis zu 7,5 MHz zu arbeiten. Die drei LEDs mit unterschiedlichen Farben am Debugger zeigen den Status an. Die Adapterplatine wird nur benötigt, wenn Anschlüsse in einem bestimmten Layout gebraucht werden, ansonsten reichen auch Flachbandkabel. Die zu programmierende MCU wird grundsätzlich stets mit einer Spannung von 1,6 V bis 5,5 V extern versorgt.



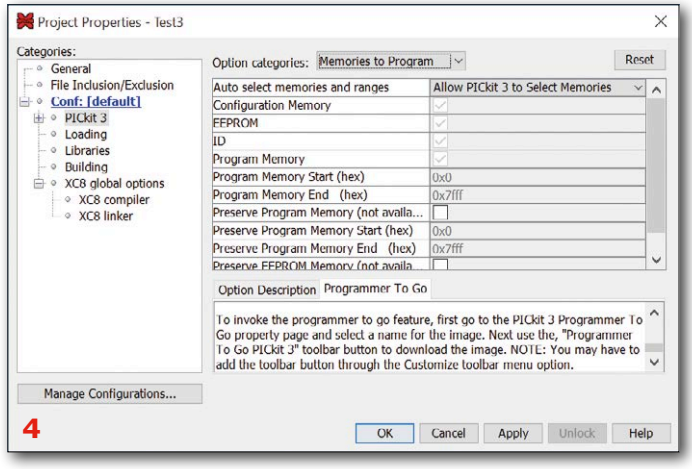
2

Wenn wir einen Blick auf andere Debugger und Programmer werfen, finden wir zum Beispiel den etwa 50 € teuren AVR ISP mkII [2], der ausschließlich die AVR-Controller programmieren kann. Als weiterer Debugger für AVR ist der über 700 € teure, aber extrem leistungsfähige AVR ONE! [3] zu finden, seit Längerem gibt es auch JTAGICE3 [4] für die SAM D und AVR-Familie, der für nicht mehr als 130 € erhältlich ist. Auch ist das Internet voll mit verschiedenen AVR-Programmern von mehr oder weniger bekannten Herstellern. Ein handlicher und mit circa 16 € der günstigste ISP-Programmer ist mySmartUSB light von der Firma Laser & Co. Solutions GmbH [5]. Alle Debugger werden vom kostenlosen Atmel Studio 7 (**Bild 2**) und den meisten anderen gängigen IDEs unterstützt. Das Atmel Studio schlägt automatisch ein Firmware-Update des Debuggers oder Programmers vor, wenn sich darauf noch eine alte Firmware-Version befindet.

Microchip

3

Der rund 70 € teure PICKit 3 Debugger [6] ist für die (ds) PIC-MCU-Familie von Microchip sehr bekannt und weit verbreitet (**Bild 3**). Er liegt auch verschiedenen Kits bei. Außer der USB-Schnittstelle für den Anschluss an den PC, wofür auch gleich ein USB-Kabel mitgeliefert wird, gibt es lediglich den sechspoligen Programmieranschluss mit der PIC-typischen ICSP-Schnittstelle (**I**n-**C**ircuit **S**erial **P**rogrammer) zum In-system-Programmieren aller gängiger PICs. Selbstverständlich kann er auch als Programmer arbeiten. Der sehr leistungsfähige PICKit 3 Debugger verspricht sehr schnelle Datenraten (so schnell wie die jeweilige MCU maximal verträgt). Er kann bei Bedarf 30 mA Strom bei 1,8 bis 5 V zur Versorgung der MCU bereitstellen. Er besitzt drei verschiedenfarbige LEDs zur Anzeige des Debugger-Status. Ein Taster ist für eine besondere Funktion des Debuggers vorhanden: Programmer-To-Go. Die Software kann in diesem Modus auf dem PICKit 3 abgelegt und

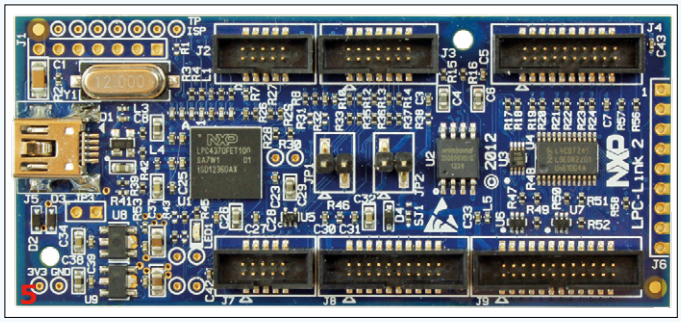


4

später lediglich durch einen Tastendruck auf eine MCU gebrannt werden, unabhängig vom Computer. Für größere Projekte und Produktionen sind aber die größeren Debugger und Programmer wie der Softlog ICP2 Production Quality In-Circuit Programmer [7] für ab 400 € besser geeignet.

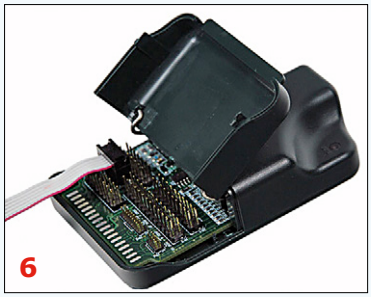
Der PICKit 3 wurde vor allem für die MPLAB X IDE von Microchip entwickelt und lässt sich dort problemlos einbinden (**Bild 4**), nicht zuletzt, weil die HID-Treiber anstandslos von Windows automatisch installiert werden.

NXP



5

Für die „normale“ LPC-Familie von NXP gibt es schon seit Längerem den LPC-Link2 (**Bild 5**), der in etwas einfacherer Form auf den meisten Xpresso-Entwicklungsboards verbaut und alleine für knapp 20 € zu haben ist [8]. Auf dem Board sind viele Wannenstecker zu finden, über die Jumper J6..J8 sind die SWD- und JTAG-Schnittstellen zugänglich. An den restlichen Anschlüssen sind lediglich die Pins des Dreikern-Debug-Controllers LPC4370 herausgeführt. Man kann nämlich den Debugger als Entwicklungsboard nutzen, wenn man den LPC4370-Mikrocontroller von außen mit einem anderen Debugger debuggt oder programmiert. Es wird ein Flachbandkabel mitgeliefert, mit dem man den Debugger mit der Ziel-MCU verbinden kann.


6

Da dieser Debugger auf CMSIS-DAP beziehungsweise der J-Link-Technologie basiert, ist er mit den meisten IDEs wie der gängigen LPCXpresso kompatibel. Mit dem Softwaretool LPCScrypt kann man die Firmware auf dem Debugger ändern und aktualisieren. Für die neulich übernommene Kinetis-Serie (ehemals Freescale) empfiehlt sich der etwa 200 € teure USB Multilink Universal Debugger (**Bild 6**) von der Firma P&E [9]. Sein größerer Bruder mit der Endung „FX“ ist ungefähr doppelt so teuer, aber leistungsfähiger und unterstützt auch mehr Mikrocontroller (Kinetis, ColdFire, LPC, STM32, PSoC 4 usw.). Unter der Plastikkappe dieses Debuggers befinden sich viele Stiftleisten, für die auch Flachbandkabel mitgeliefert werden. Dort befinden sich außer der SWD- und JTAG-Schnittstelle auch viele spezifische Schnittstellen für die zahlreichen MCU-Familien, die mittlerweile von diesem Debugger unterstützt werden. Die Taktfrequenzen können dabei bis zu 50 MHz betragen, der Spannungsbereich muss zwischen 1,6 V und 5,25 V liegen. Der „FX“-Debugger selbst kann je nach Bedarf 3,3 V oder 5 V für die MCU liefern. Grundsätzlich wird dieser Debugger von den meisten üblichen IDEs (Kinetis Design Studio, Keil, IAR usw.) unterstützt, die die P&E-Protokolle beziehungsweise -Treiber akzeptieren.

STMicroelectronics

Der ungefähr 30 € teure ST-Link/V2-Debugger [10] basiert auf einem STM32-ARM-Cortex-M3-Mikrocontroller und ist sowohl für die ARM-basierte STM32- als auch die 8-bit-STM8-Mikrocontrollerfamilie von STMicroelectronics (kurz: ST) geeignet. Er wird mit allen nötigen Kabeln geliefert (**Bild 7**). Mittlerweile ist er auch in abgespeckter Form an vielen ST-Boards vorhanden. Außer der USB 2.0-Schnittstelle für die Anbindung an den PC verfügt er über einen kleinen, vierpoligen Anschluss für die STM8- und einen doppelreihigen 20-poligen Anschluss für STM32-MCUs. Für beide Anschlüsse sind auch Kabel vorhanden, um die Mikrocontroller direkt auf der Platine zu debuggen oder zu programmieren. Die Schnittstelle für STM8-Controller heißt SWIM (**S**ingle **W**ire **I**nterface **M**odule) und benötigt außer den Stromversorgungsleitungen nur noch eine Reset- und eine Datenleitung. Der ST-Link/V2 unterstützt dabei sowohl den Low-speed- (9,7 KB/s) als auch den High-speed-Modus (12,8 KB/s) und das im Spannungsbereich von 1,65 V bis 5,5 V. Die MCU muss extern mit Spannung versorgt werden!

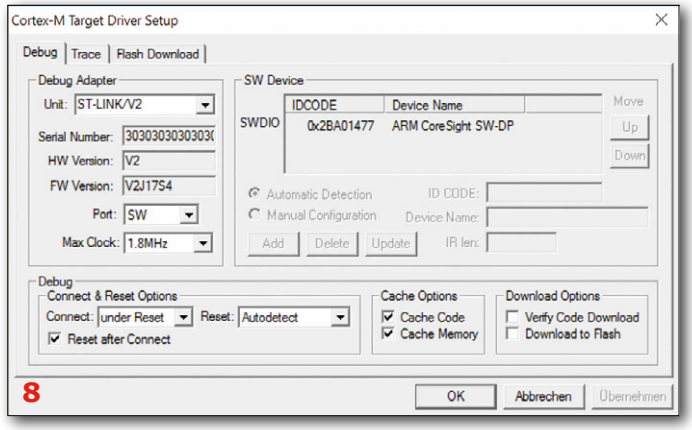

7

Der deutlich größere Anschluss für die STM32-Familie stellt sowohl die ARM-typische SWD- als auch die etwas größere JTAG-Schnittstelle zur Verfügung. Hierbei werden deutlich mehr Anschlüsse benötigt. Die zulässige Versorgungsspannung liegt im Bereich von 1,65 V bis 3,6 V, wobei die Daten-Eingänge auch 5 V verkraften können. Zur Statusanzeige ist eine Bicolor-LED an Bord, die je nach Situation in verschiedenen Farben leuchtet oder blinkt.

Der etwas mehr als doppelt so teure große Bruder heißt ST-Link/V2-ISOL und unterscheidet sich ausschließlich in zwei Punkten: Bei ihm befindet sich zur Sicherheit in kritischen Anwendungen ein Isolator (bis zu 2500 V_{RMS}) zwischen dem USB- und Programmieranschluss, außerdem ist der SWIM-Anschluss innerhalb

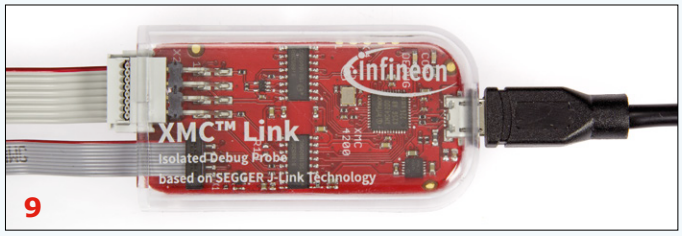
der 20-poligen Wanne vorhanden. Die mitgelieferten Kabel sind entsprechend angepasst.

Selbstverständlich gibt es auch weit teurere Debugger, die wirklich das ganze ST-MCU-Portfolio abdecken (zum Beispiel der etwa 150 € teure STX-RLINK [11]), aber dieser sehr universelle Debugger wird so gut wie von allen bekannten IDEs unterstützt, von IAR und Keil über Atolic, TASKING und STVD bis zu Coo-Cox. Die Einbindung ist meist sehr einfach, es sind nur wenige Einstellungen nötig (**Bild 8**) und alle nötigen Treiber werden meist automatisch von Windows installiert. Ein Firmware-Update dieses Debuggers ist auch möglich, dazu gibt es ein kleines kostenloses Tool auf der Produktseite.



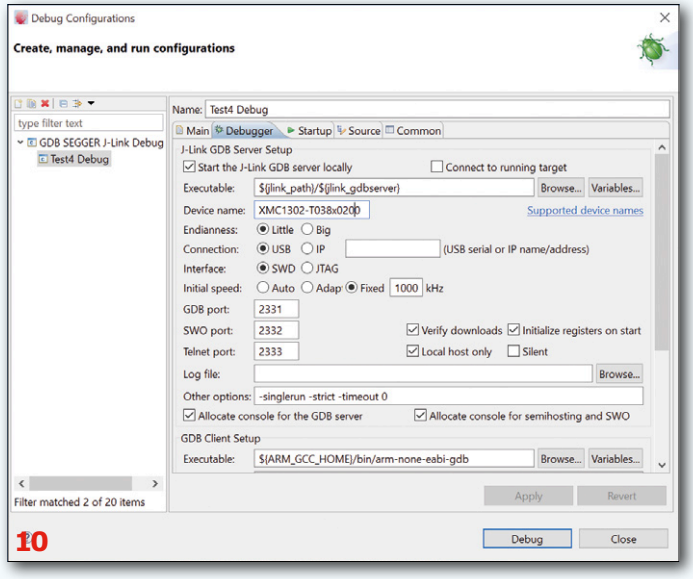
8

Infineon



9

Es gibt einen neuen Debugger (Debug-Probe) für die verbreitete XMC-ARM-Cortex-M-MCU-Familie von Infineon, den XMC Link [12] (**Bild 9**). Dieser kleine, auf einem XMC4200-Controller (mit SEGGER J-Link basierter Software) aufgebaute Debugger erfüllt alle Wünsche, die man beim Umsetzen eigener Projekte mit den XMC-MCUs hat. Er verfügt über einen achtpoligen XMC-MCU-Debug-Anschluss für die XMC1000-Cortex-M0- und XMC4000-Cortex-M4-Familie sowie einen typischen und standardisierten zehnpoligen Cortex-Debug-Anschluss, ausschließlich für die leistungsfähigere XMC4000-Cortex-M4-Familie.
Der erste Anschluss führt eine **S**ingle **P**in **D**ebug (SPD-) und eine SWD-Schnittstelle heraus, auch ist eine UART-to-USB-Bridge vorhanden, eine virtuelle serielle Schnittstelle für die Entwicklungsphase. Am großen Anschluss findet man zusätzlich eine weitere Leitung (SWO) für den **S**erial **W**ire **V**iewer (kurz: SWV, eine Trace-Funktion) und die JTAG-Schnittstelle. Beide Anschlüsse müssen von außen von der MCU-Stromversorgung im Bereich von 2,5 V bis 5,5 V versorgt werden. Außerdem hat der Debugger zwei Isolator-ICs an Board, die den PC vor Spannungen bis zu 1 kV an den Programmieranschlüssen schützen. Die beiden LEDs neben der USB-Buchse zeigen, ob der Debugger mit Strom versorgt wird und ob sich etwas tut. Der Debugger kostet weit unter 100 € (der genaue Preis war bis Redaktionsschluss unbekannt) und wird mit je einem Kabel pro Anschluss ausgeliefert.
Alle gängigen IDEs unterstützen diesen neuen Debugger. Dazu zählt natürlich auch die DAVE-IDE direkt von Infineon, wo der Umgang mit dem Debugger (**Bild 10**) überhaupt keine Hürde darstellt. In Kürze erscheint auch ein XMC Flash Tool, das das Programmieren noch weiter vereinfacht.



10

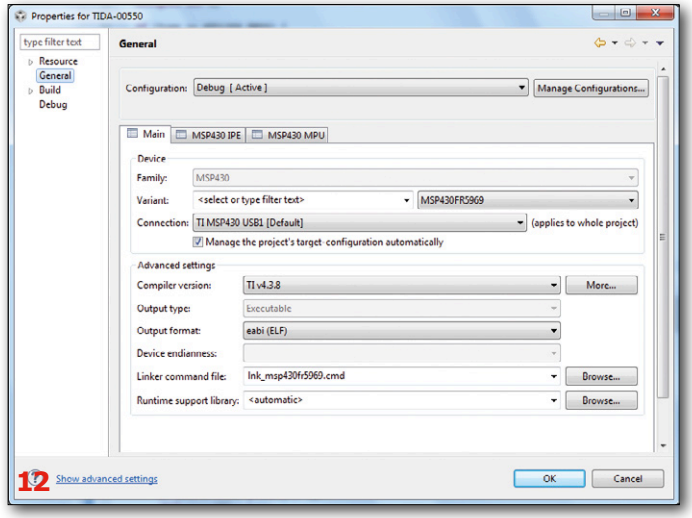
Texas Instruments

Der etwas über 100 € teure MSP-FET [13] in **Bild 11** deckt die ganze MSP430- und CC430-Mikrocontrollerfamilie von Texas Instruments (TI) ab. Die CC430-Familie hat im Vergleich zur schon länger bekannten MSP430 16-bit-Mikrocontrollerfamilie zusätzliche Peripherie für Funkübertragung an Bord. Der MSP-FET ähnelt den auf den LaunchPads verbauten eZ-FETs, bietet allerdings viel mehr Funktionen. Gegenüber der USB-Buchse findet man den 16-poligen Programmieranschluss, der eine JTAG-Schnittstelle enthält. Diese kommt im Spy-Bi-Wire-Modus mit weniger Leitungen als üblich aus. Die EnergyTrace(++)-Funktion, die der MSP-FET als Besonderheit unterstützt, hilft beim Verbessern des Codes hinsichtlich des Leistungsverbrauchs. Dieser wird während der Laufzeit gemessen und angezeigt. Auch sind UART, I²C und SPI vorhanden. So kann der Mikrocontroller Daten mit dem PC austauschen. Über UART oder I²C lässt sich sogar die MCU programmieren, wenn zuvor ein entsprechender Bootloader geladen wurde. Der MSP-FET ist in der Lage, den Mikrocontroller mit bis zu 100 mA

11

zu versorgen und das bei einer einstellbaren Spannung von 1,8 V...3,6 V. Außer der Programmierschnittstelle gibt es zwei verschiedenfarbige LEDs zur Anzeige des aktuellen Status. Im Lieferumfang befinden ein USB-Kabel und ein Flachbandkabel für den Programmieranschluss.

Außer dem **C**ode **C**omposer **S**tudio (CCS) in der normalen und in der Cloud-Variante wird dieser Debugger auch von IAR unterstützt (**Bild 12**). Für alle anderen Compiler gibt es im Notfall auch den MSP-Flasher, der die Ausgabedatei eines beliebigen



12

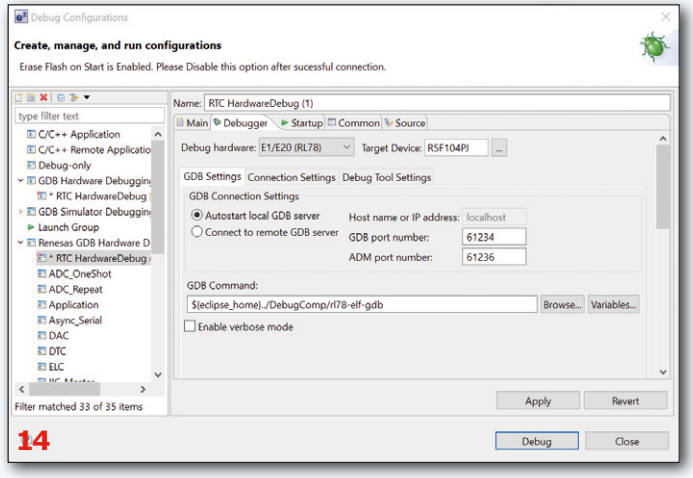
Compilers mit dem MSP-FETs auf einen MSP430-Mikrocontroller brennen kann.

Renesas

13

Da Renesas immer mehr interessante Mikrocontrollerfamilien auf den Markt bringt, muss natürlich auch das Debugger- und Programmer-Angebot für diese Controller berücksichtigt werden. Der günstige, aber dennoch universelle E1-Emulator (**Bild 13**) kann sowohl als Debugger als auch als Programmiergerät fungieren und kostet rund 200 € [14]. Geliefert wird er zusammen mit einem USB- und einem Programmierkabel. Er ist gleich für mehrere Renesas-MCU-Familien geeignet, nicht nur für die relativ bekannten Familien RL78, RX oder R8C, sondern auch für V850, RH850 (Automotive) und die Smart Analog MCU-Familie. Es gibt zwei 14-polige Anschlüsse am Emulator für die üblichen Debug- und Programmierfunktionen. Der Anschluss direkt gegenüber der USB-Buchse führt die UART, SPI- und JTAG-Programmierschnittstellen, der von einer Plastikklappe verdeckte, aber sonst genauso aussehende Anschluss an der Vorderseite ist der Self-Test-Funktion vorbehalten. Man kann mit einem Tool, das auf der Produktseite (nach Anmeldung) zur Verfügung steht, den Debugger Schritt für Schritt in seinen Funktionen testen, wobei man bei einem Schritt auch die beiden 14-poligen Anschlüsse zusammenschließen muss. Am „echten" Programmieranschluss kann die angeschlossene MCU mit 3,3 V oder 5 V und bis zu 200 mA versorgt werden. Die LEDs an den Seiten zeigen verschiedene Zustände des Debuggers an. Der E20-Emulator [15] ist ein größerer Bruder des E1- Emulators mit mehr Funktionen, kostet aber weit über ein 1000 €.

Selbstverständlich wird dieser Debugger vom e²studio, der CubeSuite+ und vielen weiteren bekannten IDEs unterstützt (**Bild 14**). Auf der Produktseite finden Sie auch externe und kostenlose Programmiertools wie den Renesas Flash Programmer.



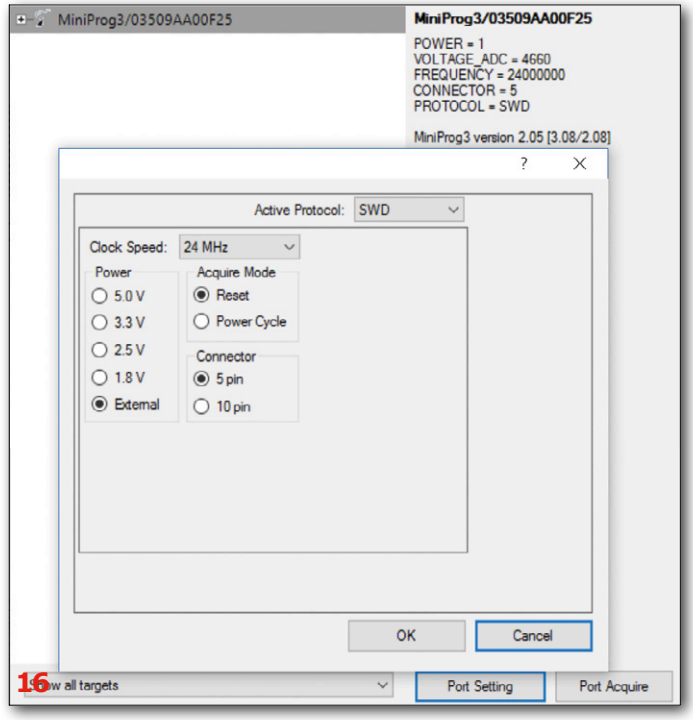
14

Cypress

Für die durchaus besondere PSoC-Familie gibt es einen kleinen Debugger namens MiniProg3 (**Bild 15**) zu einem Preis von fast 80 € [16]. Er sieht aus wie ein USB-Stick und hat zwei Programmieranschlüsse. Er kann PSoC 1 bis -5LP programmieren, allerdings nur PSoC 3 bis PSoC 5LP debuggen. Eine fünfpolige Buchsenleiste zum Aufstecken auf eine Platine und ein normaler zehnpoliger Anschluss, für den auch gleich ein Flachbandkabel mitgeliefert wird, sind vorhanden. Am fünfpoligen Anschluss

15

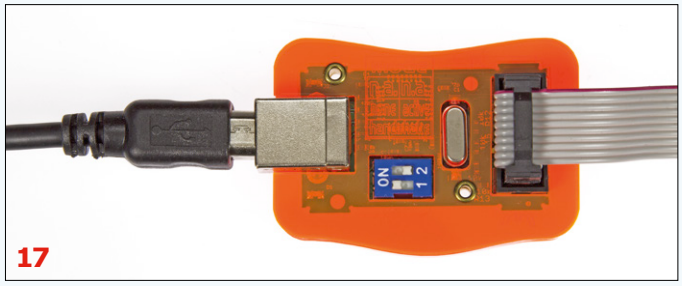
befindet sich die **I**n-**S**ystem **S**erial **P**rogramming Schnittstelle (ISSP) und die SWD-Schnittstelle, außerdem eine I²C-Schnittstelle, die allerdings nur für den Datenaustausch mit dem Computer gedacht ist (I²C-zu-USB-Brücke).
Am zehnpoligen Anschluss ist die JTAG- und die SWD- beziehungsweise SWV-Schnittstelle zu finden. JTAG und SWV werden nur von PSoC 3 und PSoC 5LP unterstützt, die SWD-Schnittstelle ist für alle debug-fähigen MCUs geeignet. Für PSoC 1 muss man die ISSP-Schnittstelle nutzen. MiniProg3 ist in der Lage, die zu programmierbare MCU mit maximal 200 mA zu versorgen. Es stehen dabei vier verschiedene Spannungen zur Wahl: 1,8 V, 2,5 V, 3,3 V und 5 V. Die fünf im Debugger eingebauten LEDs zeigen den aktuellen Status an.
Das verbreitetste Software-Tool für diesen Debugger heißt PSoC Programmer, dieses ist im PSoC Creator beziehungsweise Designer integriert (**Bild 16**), sodass es direkt innerhalb der IDE verwendet werden kann.



16

Universelle Debugger und Programmiergeräte

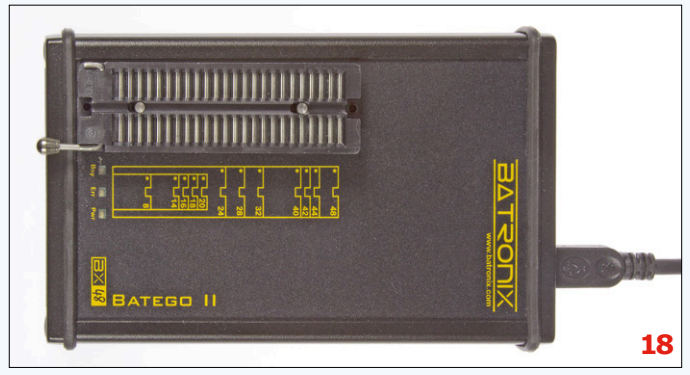
Kommen wir nun zu herstellerunabhängigen und universellen Debuggern und Programmern. Es gibt zahlreiche, weit verbreitete Typen, aus Platzgründen können nur drei preiswerte und am weitesten verbreitete im Überblick behandelt werden.



17

Beginnen wir mit dem etwa 20 € teuren PROG-S [17] Programmiergerät der Firma Diamex (**Bild 17**). Mit diesem Programmer ist man in der Lage, bestimmte Mitglieder der AVR-, LPC- und STM32-Mikrocontrollerfamilie auszulesen und mit Software zu beschreiben. Ein USB- und ein Programmierkabel werden mitgeliefert. Je nach zu programmierender MCU-Familie muss die Stellung der beiden Micro-Schalter am Debugger angepasst werden. Die beiden LEDs zeigen den Status des Debuggers an. Außer der USB-Schnittstelle für den Computer gibt es lediglich einen 10-poligen Programmieranschluss mit UART und SPI (ISP). SPI ist für die AVR-Mikrocontroller zum ISP-Programmieren gedacht. PROG-S arbeitet dann wie ein STK500 und kann direkt vom Atmel-Studio angesprochen werden. Mit UART und zwei weiteren Anschlüssen (Reset- und Boot-Pin) kann man mit dem PROG-S-Programmer LPC- (Software: Flash Magic) oder STM32-Mikrocontroller (Software: STM32Prog) programmieren, wenn die mit einem entsprechenden Bootloader ausgestattet sind. Dabei müssen die zu programmierenden MCUs immer extern mit Strom versorgt werden. Es gibt auch eine Schalterstellung, bei dem das Programmiergerät als UART-zu-USB-Brücke dient. Welche Mikrocontroller von diesem Programmer unterstützt werden, können Sie auf der Produktseite nachlesen. Seit Kurzem gibt es auch einen etwas leistungsfähigeren Nachfolger namens PROG-S2 [18].

Ein schwereres Geschütz stellt der BX48 Batego II von Batronix dar (**Bild 18**), die mit ungefähr 500 € teuerste hier beschriebene Hardware [19]. Dieser Programmer, bei dem die MCUs einzeln auf die Fassung gesteckt werden müssen, ist für relativ große Produktionsreihen gedacht und entsprechend stabil und hochwertig gebaut. Als Zubehör werden außer dem USB-Kabel mehrere Software-Programme auf der beiliegenden CD mitgeliefert. Dieses Programmiergerät ist vorwiegend (und seine kleineren Brüder sogar ausschließlich!) für Speicher-ICs aller Art gedacht, was besonders für größere Projekte von Vorteil ist, bei dem die MCU externe Speicher nutzt. Sollte das IC beziehungsweise der Mikrocontroller etwa in einem SMD-Gehäuse stecken,



18

kann man bei Batronix entsprechende Adapter erwerben. Bei Bedarf gibt es auch einen Adapter, mit dem man den Batego in einen In-system-Programmer verwandeln kann.
Es werden einige Mikrocontrollerfamilien unterstützt, von den bekannten AVR- und PIC-MCUs bis zu wenig bekannten 80C51ern von Goldstar sowie Dallas-, Intel- und Philips-MCUs. Die Programmierung verläuft sehr schnell; die dazu gehörende Software Prog Express ist sehr intuitiv und bietet alle Funktionen, die man in einer kleinen Serienproduktion benötigt. Man kann zum Beispiel unbekannte ICs vom Batego automatisch erkennen lassen oder auf jedem gebrannten IC eine zusätzliche Seriennummer mit abspeichern. Alles in allem ein sehr professionelles Tool!

Wenn man von ARM und Debugging spricht, denkt man fast schon automatisch an die SEGGER J-Link-Debugger (**Bild 19**). Oft lassen Halbleiterhersteller auf die Evaluationboards auch einen J-Link-Lite layouten oder entwickeln ihren eigenen Debugger auf Basis der J-Link-Technologie. Doch die J-Links sind auch einzeln zu bekommen [20], und zwar in verschiedenen hochwertigen und gut ausgestatteten Ausführungen.
Das günstigste Studenten-Modell kostet rund 60 €, das BASE-Modell etwas mehr als 300 €. Im Gegensatz zur Studentenversion ist die Zahl an möglichen Breakpoints im Flash-Speicher nicht limitiert. Gibt man noch etwas mehr aus, bekommt man das PLUS-Modell mit all der benötigten Lizenz-Software für etwa 600 €. Es gibt noch leistungsfähigere Modelle mit 3 MB/s statt 1 MB/s Downloadgeschwindigkeit und zum Beispiel Netzwerk- oder besondere Trace-Funktionen.
Alle J-Link-Debugger unterstützen alle Mikrocontroller, die auf ARM 7, ARM 9 oder ARM 11 basieren sowie alle ARM-Cortex-Mikrocontroller. Unabhängig davon werden auch die 32-bit-MCUs der Renesas RX- und die Microchip PIC32-Familie unterstützt. Zum Programmieren sind u.a. eine SWD- und eine JTAG-Schnittstelle am 20-poligen Programmieranschluss vorhanden (USB und Programmierkabel werden mitgeliefert). Der J-Link kann bei Bedarf auch die MCU-Schaltung mit maximal 5 V und 300 mA versorgen. Ein Isolator kann zusätzlich erworben werden.

19

Grundsätzlich haben die J-Links viele besonders interessante Funktionen wie den **R**eal **T**ime **T**ransfer (RTT) Modus zum Datenaustausch mit der MCU während der Laufzeit oder ein Softwareprogramm namens System View zur Visualisierung und Analyse bestimmter Parameter während der Laufzeit. Software für den J-Link gibt es jedenfalls genug, zum Programmieren kann man das Softwaretool J-Flash nutzen. Zum reinen Programmieren bietet Segger auch die Flasher-Serie an. Mit J-Link haben Sie in der 32-bit-(ARM)-Welt einen universellen, von so gut wie allen IDEs unterstützten Debugger an Ihrer Seite!

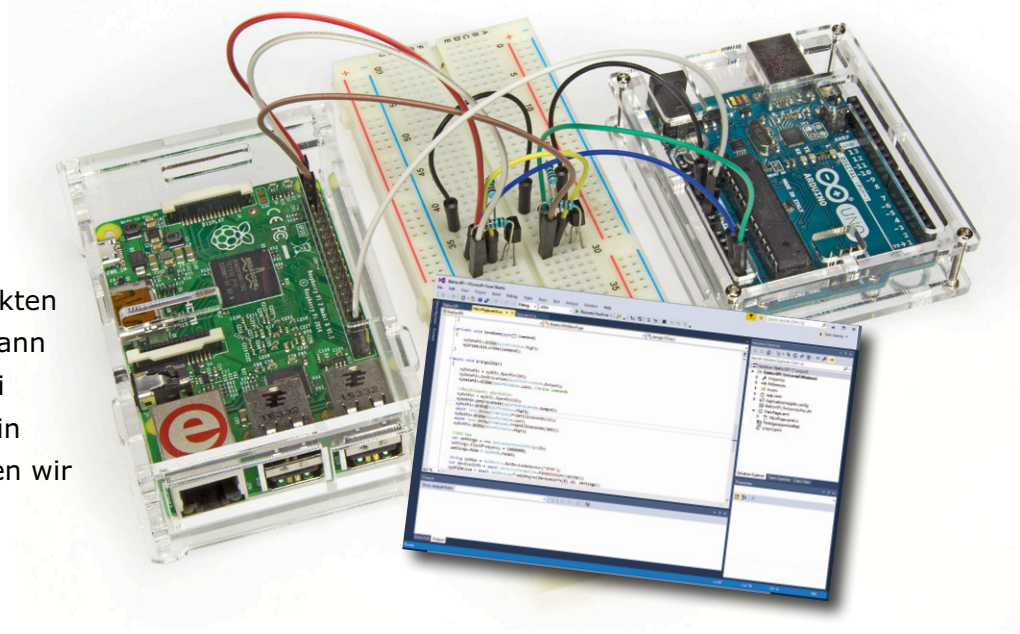
(150725)

Weblinks

[1] www.atmel.com/tools/atatmel-ice.aspx
[2] www.atmel.com/tools/avrispmkii.aspx
[3] www.atmel.com/tools/avrone_.aspx
[4] www.atmel.com/tools/jtagice3.aspx
[5] http://shop.myavr.de/Topseller/mySmartUSB%20light.htm?sp=article.sp.php&artID=200006
[6] www.microchip.com/DevelopmentTools/ProductDetails.aspx?PartNO=PG164130&utm_source=&utm_medium=MicroSolutions-&utm_term=&utm_content=DevTools&utm_campaign=PICkit+3
[7] www.microchip.com/Developmenttools/ProductDetails.aspx?PartNO=TPG100001
[8] www.nxp.com/products/microcontrollers-and-processors/arm-processors/lpc-cortex-m-mcus/lpc-cortex-m4-single-multi-core/lpc4300-series/lpc-link2:OM13054
[9] www.nxp.com/products/interface-and-connectivity/wireless-connectivity/sub-1-ghz-wireless-solutions/universal-multilink-development-interface:UMultilink
[10] www.st.com/web/catalog/tools/FM146/CL1984/SC724/SS1677/PF251168?sc=internet/evalboard/product/251168.jsp
[11] www.st.com/web/catalog/tools/FM146/CL1984/SC724/SS1677/PF122903
[12] www.infineon.com/cms/en/product/productType.html?productType=5546d462501ee6fd015023aeb65733b3
[13] www.ti.com/tool/MSP-FET#
[14] www.renesas.com/products/tools/emulation_debugging/onchip_debuggers/e1/
[15] www.renesas.com/products/tools/emulation_debugging/onchip_debuggers/e20/index.jsp
[16] www.cypress.com/documentation/development-kitsboards/cy8ckit-002-psoc-miniprog3-program-and-debug-kit
[17] www.diamex.de/dxshop/mediafiles//Sonstiges/Prog-S-Anleitung.pdf
[18] www.diamex.de/dxshop/USB-ISP-Programmer-fuer-AVR-STM32-NXP-Cortex-Prog-S2
[19] www.batronix.com/versand/programmiergeraete/BX48/batego-II.html
[20] www.segger.com/j-link-debugger.html

Windows auf dem RaspPi (3)

SPI und I²C



Von **Tam Hanna**

Zur Kommunikation mit externer Elektronik werden in Controllerprojekten oft SPI und I2C genutzt. Auch das kann der Windows-Raspi, wie wir mit zwei Demos beweisen. Erst steuern wir ein kleines OLED-Display an, dann setzen wir einen Arduino als Hilfsarbeiter ein.

Microsoft ist nicht seit gestern im Embedded-Bereich tätig – der Gadgeteer darf auf eine jahrelange Tradition zurückblicken. Allen redmond'schen Kindern ist ihre miserable Performance gemein: Es gibt kein preiswertes Prozessrechnersystem microsoft'scher Bauart, das stringente Echtzeitkriterien erfüllt.
In Windows 10 für den Raspi sieht die Situation insofern nicht besser aus, als Applikationen in Form von Managed Code vorliegen müssen. Das bedeutet, dass Zugriffe auf die Hardware über einen zeitraubenden Umweg in das Betriebssystem erfolgen. Erfreulicherweise enthält der von Broadcom entwickelte RPi-Hauptprozessor Engines für I2C und SPI. Die Kommunikation über die beiden Busse erfolgt in Hardware; das Betriebssystem setzt nur Order ab und sammelt Ergebnisse ein, was beides nicht sonderlich zeitkritisch ist.

OLED-Display

Der RPi ist insbesondere wegen der Fähigkeit zur Ausgabe von HDMI interessant. Leider sind HDMI-fähige Paneele meist groß und stromgierig. Chinesische Hersteller fluten den Markt aber mit 128 x 64 Pixel großen OLED-Displays, die dank monochromatischer Darstellung mit wenig Bandbreite auskommen. Das Display in **Bild 1** hat der Autor bei AliExpress gekauft. Identische Bauteile gibt es bei eBay und in gut sortierten Elektronikshops. Der in diesen Displays verbaute Controller ist für seine Flexibilität bekannt. Er spricht neben I2C auch zwei Varianten von SPI; die gerade aktive Kommunikationsform wird über auf der Rückseite verlötete Wiederstände festgelegt. **Bild 2** zeigt die von Haus aus verlöteten Komponenten, die das Modul in einen proprietären, stark an SPI angelehnten Kommunikationsmodus versetzen. Die Verbindung zum RPi erfolgt gemäß dem in **Bild 3** gezeigten Schema. Das Projekt kann man einfach auf einem Steckbrett aufbauen (**Bild 4**).
Aufmerksame Leser bemerken an dieser Stelle das Fehlen einer Rückfütterung vom Display zum Prozessrechner. Dies ist kein Schaltungsfehler: Unser Display kann keine Daten an den Host senden. Dem Displaycontroller kann man abwechselnd Daten und Steuerkommandos senden, um was es sich handelt, wird ihm über den DC-Pin mitgeteilt. Die Verbindung von Reset-Eingang und GPIO-Pin ist erforderlich, da der Controller beim Start einen Reset-Puls fordert.
Mit diesem Wissen können wir mit der Realisierung einer weiteren Universal-App beginnen. Der gesamte Code kann wie immer als Visual-Studio-Projekt von Elektormagazine.de heruntergeladen werden [3].

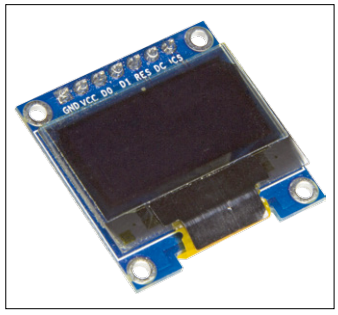


Bild 1. Das OLED-Display – klein, aber oho.

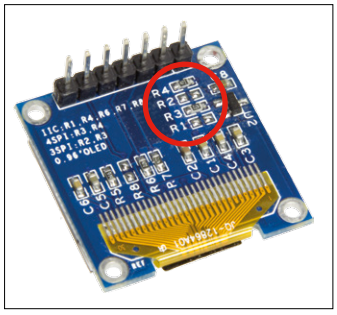


Bild 2. SMD-Widerstände legen den Ansteuermodus des Displays fest (hier SPI).

Display-Befehle

Als erstes deklarieren wir eine Gruppe von Variablen, die für die Kommunikation erforderlich sind:

```
GpioController myGCIc;
GpioPin myDataPin;
```

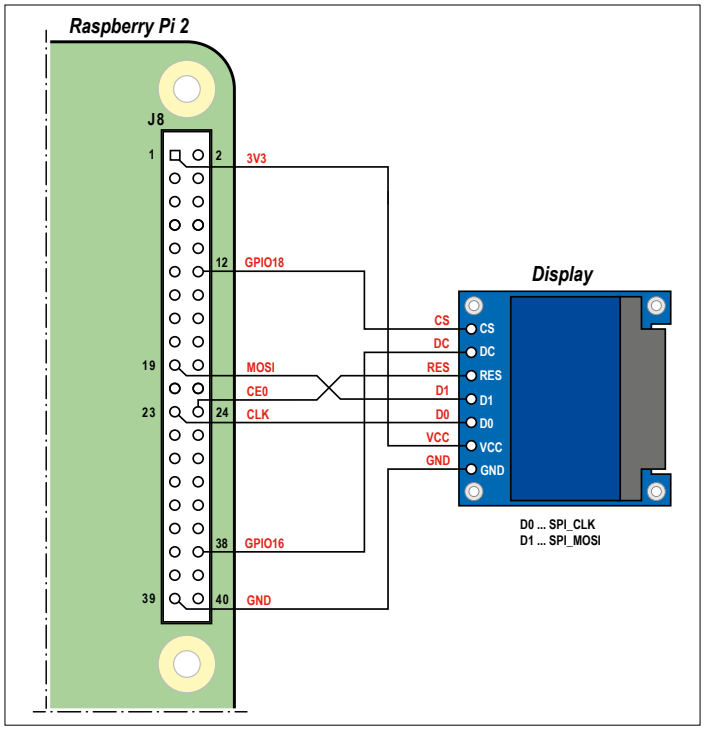


Bild 3. Zur Ansteuerung des Displays sind nur fünf Leitungen nötig.

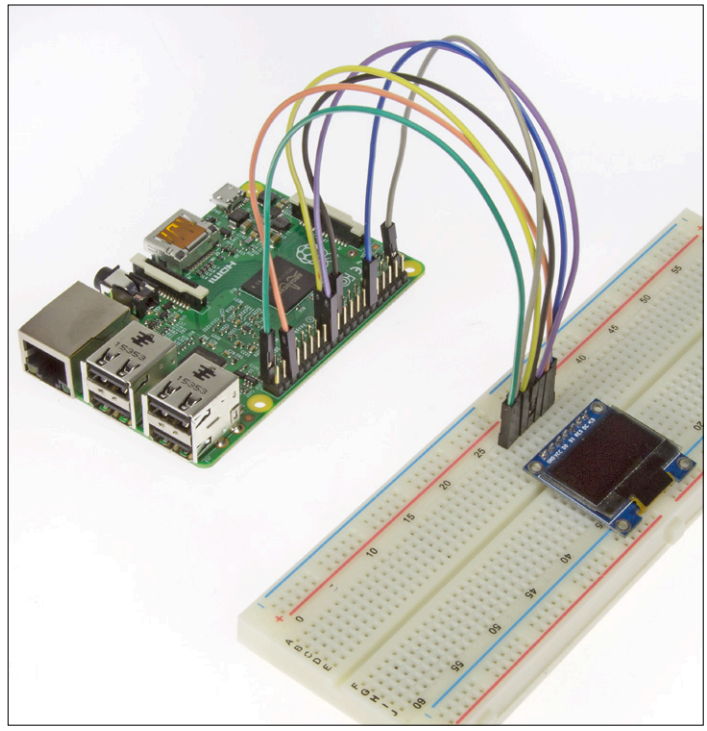


Bild 4. Unser Display-Projekt können wir einfach auf einem Steckbrett aufbauen.

```
SpiDevice mySPIDevice;

GpioPin myRstPin;

byte[] gfxBuf = new byte[128 * 64 / 8];
```

Hier sind zwei Sachen relevant. Wir erzeugen ein Byte-Array, das als Speicher für die auf den Bildschirm zu schiebenden Pixel-Daten dient. Zweitens erzeugen wir globale Instanzen aller Pins und des SPI-Devices – das ist wichtig, weil der Garbage-Collector (siehe Kasten) sonst für Unruhe sorgt.

Danach folgen einige globale Variablen, die mit den zu sendenden Kommando-Bytes gefüllt werden. Hier kann man sich häufig an Treibern für Arduino orientieren, die zum Beispiel von Adafruit oder anderen Herstellern/Distris angeboten werden. In unserem Fall war die Situation besonders günstig, weil Microsoft selbst Beispielcode bereitstellt:

```
private static readonly byte[] CMD_DISPLAY_OFF =
   { 0xAE };
private static readonly byte[] CMD_DISPLAY_ON =
   { 0xAF };
private static readonly byte[] CMD_CHARGEPUMP_ON =
   { 0x8D, 0x14 };
private static readonly byte[] CMD_MEMADDRMODE =
   { 0x20, 0x00 };
…
```

Im Konstruktor der MainPage (siehe Teil 1 [1]) rufen wir die Funktion `bringLCDUp` auf, und bevölkern den Display-Speicher mit zufälligen Daten (**Listing 1**). Die zufälligen Daten sind

Listing 1. Display hochfahren.

```
public MainPage()
{
  this.InitializeComponent();
  myGCIc = GpioController.GetDefault();
  Random myRND = new Random();
  myRND.NextBytes(gfxBuf);

  bringLCDUp();
}

sync void bringLCDUp()
{
  myDataPin = myGCIc.OpenPin(16);
  myDataPin.SetDriveMode(GpioPinDriveMode.Output);
  myDataPin.Write(GpioPinValue.Low); //Write commands

  //Resetsequenz abarbeiten
  myRstPin = myGCIc.OpenPin(18);
  myRstPin.SetDriveMode(GpioPinDriveMode.Output);
  myRstPin.Write(GpioPinValue.High);
  await Task.Delay(TimeSpan.FromMilliseconds(1));
  myRstPin.Write(GpioPinValue.Low);
  await Task.Delay(TimeSpan.FromMilliseconds(100));
  myRstPin.Write(GpioPinValue.High);

…
}
```

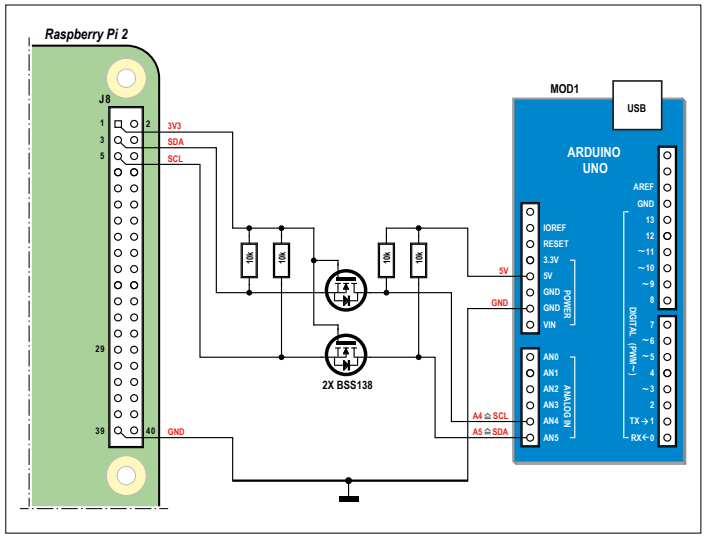


Bild 5. Die Pegelwandler sorgen dafür, dass der Raspberry Pi 2 ob der 5-V-Signalspannung des Arduino Uno keinen Schaden nimmt.

wichtig, weil ein komplett schwarzes OLED keinerlei Lebenszeichen abgibt – es gibt mehr als einen Computertechniker, der beim Programmieren von Spielen für OLED-Smartphones sein Gerät neu gestartet hat, da er von einem Totalausfall ausging. `BringLCDUp` beginnt mit der Durchführung der im Datenblatt vorgesehenen Resetsequenz.
.NET-erfahrene Entwickler vermissen an dieser Stelle die Verzögerungsfunktion `thread.Sleep`. Die altbewährte Methode ist nicht mehr implementiert: Windows-RT-Applikationen dürfen ja nicht blockieren, sondern müssen immer responsiv sein.

SPI

Im nächsten Schritt folgt die Initialisierung des SPI-Busses. Das Ansteuern von externen Hardware-Bussen erfolgt unter Windows 10 für IOT immer über ein Objekt, das nach folgendem Schema deklariert wird:

```
//SPI bus
var settings = new SpiConnectionSettings(0);
settings.ClockFrequency = 10000000;
settings.Mode = SpiMode.Mode3;

string spiAqs = SpiDevice.GetDeviceSelector("SPI0");
var devicesInfo = await DeviceInformation.
 FindAllAsync(spiAqs);
mySPIDevice = await SpiDevice.
 FromIdAsync(devicesInfo[0].Id, settings);
```

Für das anzeigefertige Display fehlt uns nun nur noch das Übertragen der Initialisierungskommandos und das Schreiben der im Displaypuffer befindlichen Informationen. Dies wird durch den letzten Teil von `bringLCDUp` erledigt, der so aussieht:

```
SendCommand(CMD_CHARGEPUMP_ON);
SendCommand(CMD_MEMADDRMODE);
SendCommand(CMD_SEGREMAP);
SendCommand(CMD_COMSCANDIR);
SendCommand(CMD_DISPLAY_ON);

SendCommand(CMD_RESETCOLADDR);
SendCommand(CMD_RESETPAGEADDR);
SendData(gfxBuf);
```

In den letzten Zeilen wird der Controller darüber informiert, dass neue Daten angeliefert werden, die in der oberen linken Kante zuhause sind. Danach wandert das gesamte Bitmap in einem Schritt in Richtung des Displaycontrollers, der es auf den Bildschirm schreibt und anzeigt.
Die Funktionen `SendCommand` und `SendData` unterscheiden sich nur insofern voneinander, als sie den DC-Pin vor dem Aktivieren der SPI-Übertragung auf einen anderen Wert setzen.

```
private void SendCommand(byte[] Command)
{
    myDataPin.Write(GpioPinValue.Low);
    mySPIDevice.Write(Command);
}

private void SendData(byte[] Command)
```

Raspi Stand-Alone

Unser Raspi ist im Moment ein denkbar schlechter Prozessrechner: Nach einem Verlust der Stromversorgung startet er wieder in die „Steuerungsapp“ (siehe Teil 1 [1]), und arbeitet sein Prozessrechenprogramm erst nach Extraeinladung per Visual-Studio weiter ab. Zur Behebung dieses Problems müssen Sie sich per Power-Shell mit dem Raspberry Pi 2 verbinden. Dazu öffnen sie die Power-Shell am Desktop-Computer und geben den folgenden Befehl ein:

```
IotStartup list
```

Windows 10 reagiert darauf mit dem Anzeigen einer Liste der im Raspi befindlichen Applikationen. Falls keine Namen angezeigt werden, können Sie ihr Programm anhand der in den Projekteigenschaften angezeigten UUID erkennen.

Im nächsten Schritt lässt sich das Programm durch Eingabe von `IotStartup add …` einpflegen. Weitere Informationen dazu finden sich unter https://ms-iot.github.io/content/en-US/win10/tools/CommandLineUtils.htm. Microsoft ändert die Syntax des Kommandos relativ häufig, weshalb Sie vor der Nutzung einen Blick auf die Webseite werfen sollten.

Zu guter Letzt sei hier noch – schon aus Gründen der Vollständigkeit - darauf hingewiesen, dass ein Windows-10-Raspi ein vollwertiger Windows-Computer ist. Das bedeutet, dass er gerne heruntergefahren werden möchte. Eine Aufgabe, die sich durch Eingabe von `shutdown /s /t 0` bewerkstelligen lässt (wenn der Startbildschirm erscheint, kann der Raspi abgesteckt werden).

```
{
    myDataPin.Write(GpioPinValue.High);
    mySPIDevice.Write(Command);
}
```

Über SPI lassen sich auch Daten einlesen, mehr darüber findet man unter dem Weblink [4].

1-Wire-Bus …

SPI mag ein sehr schneller Übertragungsstandard sein. Bei der Temperaturerfassung hat sich aber der 1-Wire-Bus von Maxim etabliert.
Er ermöglicht das Errichten von Sensornetzen, die über eine Daten- und eine Masseleitung sowohl mit Energie als auch mit Informationen versorgt werden. Unter Linux kann der Raspberry Pi derartige Sensoren nativ ansprechen: Die Raspbian-Entwickler spendierten ein Kernel-Modul, das 1-Wire per Bit-Banging emuliert.
Unter Windows RT ist dies nicht ohne weiteres möglich, da der asynchrone Aufbau des Betriebssystems das Realisieren von präzisen Verzögerungen (auf Mikrosekundenebene) unmöglich macht. Alles, was kleiner als eine Millisekunde ist, lässt sich wegen des Garbage-Collectors nicht genau timen.
Bisherige Versuche der Realisierung von Bit-Banging scheiterten: Es gibt die eine oder andere Diskussion, in der auch Microsoft-Mitarbeiter offen zugeben, dass es nicht möglich ist, Windows 10 als Basis für Bit-Banging zu nutzen. Als Alternative wird - wie so oft – die Nutzung eines externen Controllers vorgeschlagen. Maxim bietet mit dem DS2482 einen Chip an, der sein 1-Wire-Netzwerk per I2C ansprechbar macht.

… und Arduino

Wer sich mit den Lehren des Misanthropen Murphy auseinandergesetzt hat, weiß über die Problematik der Beschaffung von Bauteilen Bescheid. Wenn sie den erwähnten 1-Wire-Adapter am dringendsten benötigen, ist er europaweit nicht zu beschaffen. Schon aus diesem Grund sollten wir stattdessen einen Arduino Uno zum Mess-Sklaven machen, der uns die Werte per I2C zuschickt. Zur Demonstration der Kommunikation haben wir den Uno hier mit einer einfachen Prozessrechensoftware (**Listing 2**) ausgestattet, die auf Anfragen unseres Raspi (vier Bytes 123, 100, 128, 200) mit einer vorgefertigten, 16 Bytes langen Antwort („HELLO…“) reagiert. Das Abfragen von 1-Wire-Sensoren wurde hier nicht implementiert, da das Programmieren eines Arduinos nicht Thema dieses Artikels ist.

Verbinden sie den Arduino wie in **Bild 5** gezeigt mit dem RPi 2, das Ganze müsste dann wie in **Bild 6** aussehen. Auf der Seite von Windows 10 ist abermals eine Universal-App erforderlich, die diesmal eine Instanz der I2C-Device-Klasse erstellt. Die eigentliche Initialisierung verhält sich im Großen und Ganzen wie im Fall von SPI. Dabei wird auch die Geschwindigkeit festgelegt; entweder kommt der mit rund 400 kHz arbeitende Standardmodus oder eine schnellere Variante zum Einsatz.

In **Listing 3** sieht man die Raspi-Software. Wir lesen die I2C-Daten vom Arduino mit der Funktion `ReadPartial` ein. Deren Ergebnis `result` enthält Informationen über die Kommunikation, unter anderem die Zahl der Bytes, die vom Arduino zurückkamen. Dies können wir zur Auswertung nutzen.

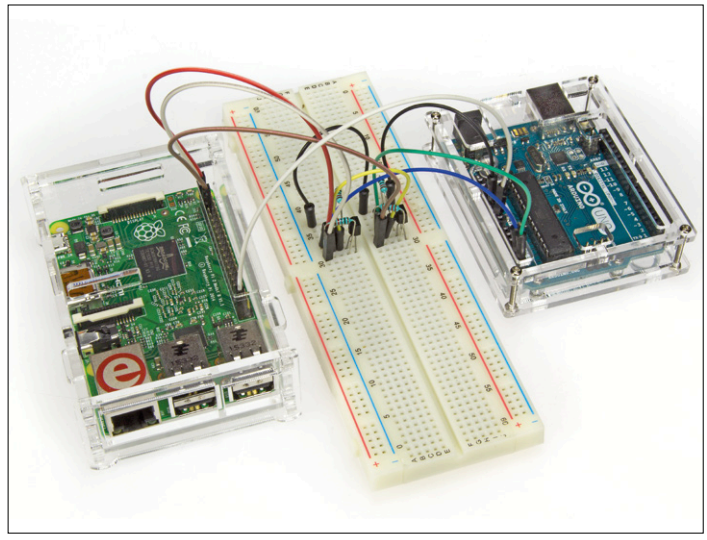


Bild 6. Arduino und Raspberry Pi arbeiten hier zusammen.

Fazit

Der Raspberry Pi 2 mit Windows 10 ist nicht unbedingt das, was sich ein Entwickler von MSR-Anwendungen wünscht. Es steht außer Frage, dass der GUI-Stack und die Netzwerkanbindung erstklassig sind – leider ist die Performance für harte Regelungsaufgaben alles andere als befriedigend. Das Anschließen von per I2C oder SPI ansprechbaren Sensoren erleichtert die Datenakquise, kann aber nichts in Bezug auf Echtzeit ausrichten: rennt der Garbage Collector, so rennt er.
Im Zusammenspiel mit einem Arduino lassen sich diese Schwächen umschiffen. Wer den Echtzeitteil aus Einzelteilen (AVR-MCU

Listing 2. Arduino als „Mess-Sklave“.

```
#include <Wire.h>

void setup() {
  Wire.begin(13);
  Wire.onReceive(receiveEvent);
  Serial.begin(9600);
}

void loop() {
  delay(100);
}

void receiveEvent(int howMany) {
  char b1=Wire.read();
  char b2=Wire.read();
  char b3=Wire.read();
  char b4=Wire.read();
  if(b1==123 && b2 == 0 && b3 == 128 && b4==200)
  {
    Wire.write("HELLOHELLOHELLO!");
  }
}
```

Listing 3. Raspi-I2C-Software.

```
namespace ElektorI2C
{
…

  public sealed partial class MainPage : Page
  {
    int myArduinoAdress = 13;
    private I2cDevice myI2CArduino;

    public MainPage()
    {
      this.InitializeComponent();
      initI2c();
    }

    async void initI2c()
    {
      string aqs = I2cDevice.GetDeviceSelector();
      var dis = await DeviceInformation.FindAllAsync(aqs);
      var settings = new I2cConnectionSettings(myArduinoAdress);
      settings.BusSpeed = I2cBusSpeed.StandardMode;
      myI2CArduino = await I2cDevice.FromIdAsync(dis[0].Id, settings);

      //Prozessrechner abfragen
      myI2CArduino.Write(new byte[] { (byte)128, (byte)0, (byte)128, (byte)200 });
      byte[] i2cReadField = new byte[16];
      var result = myI2CArduino.ReadPartial(i2cReadField);
      if (result.Status == I2cTransferStatus.PartialTransfer || result.Status == I2cTransferStatus.FullTransfer)
      {
          if (result.BytesTransferred >= 4)
          {
              System.Diagnostics.Debug.WriteLine("Arduino works!");
          }
      }
    }
  }
}
```

und Platine) aufbaut, kann nochmal den einen oder anderen Euro sparen. Die Frage nach der Aufteilung – was macht der RPi, was macht der Echtzeitrechner - ist indes ein Thema, das ganze Lehrbücher füllt!

(150520)

Weblinks

[1] www.elektormagazine.de/150465

[2] www.elektormagazine.de/150519

[3] www.elektormagazine.de/150520

[4] https://msdn.microsoft.com/en-us/library/windows.devices.spi.aspx

Garbage-Collector

Wer den Garbage-Collector in Aktion sehen möchte, entfernt die globale Deklaration von myRstPin und deklariert sie stattdessen lokal in der betreffenden Methode.
Wenn sie ein derartiges Programm ausführen, so erscheint der Displayinhalt kurz, um danach wieder zu verschwinden. Das liegt daran, dass der Garbage-Collector das GPIO-Pin-Objekt abträgt und den Pin im Rahmen dessen wieder in den Urzustand zurückversetzt.

Tipps und Tricks

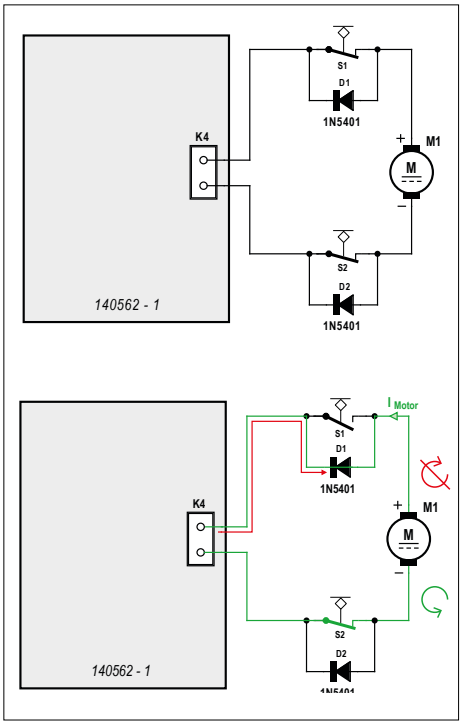
Von Lesern für Leser

Hier kommen wieder clevere Lösungen, die das Elektronikerleben leichter macht.

Gleichstrommotor am Endanschlag

Von Martin Weiß

Gleichstrommotoren werden zum Bewegen oder Verfahren eines Gegenstands eingesetzt. Des Öfteren darf aber die Bewegung bzw. das Drehen eines Gleichstrommotors gewisse Grenzen nicht überschreiten. Endschalter, die die Motorspannung unterbrechen, sind hier eine praktische Lösung. Der Öffner-Kontakt eines Endschalters wird direkt in Reihe zum Motoranschluss geschaltet, sehr oft kommen sogar zwei Endschalter zum Einsatz, um die Bewegung in beide Richtungen zu begrenzen.



Allerdings gibt es nun ein Problem: Ist ein Endschalter ausgelöst, lässt sich der Motor nicht mehr in die Gegenrichtung zurückdrehen bzw. zurückfahren, da ja die Stromzufuhr unterbrochen ist.

Die Lösung sind Dioden, die den ausgelösten/betätigten Öffner-Kontakt überbrücken. In die ursprüngliche Drehrichtung des Gleichstrommotors sperren sie den Stromfluss, doch in die gewünschte Gegenrichtung geht es weiter, zumindest gemächlich. Die Versorgungsspannung des Motors ist aufgrund der Durchlassspannung der Diode verringert, was sich im ersten Moment auf die Drehzahl und/oder das Anzugsdrehmoment des Motors auswirkt. Nachdem der Öffner-Kontakt freigefahren ist, erreicht der Motor wieder die gewohnte Geschwindigkeit.

Wichtig bei der relativ simplen und kostengünstigen Lösung ist allerdings, dass man die Dioden so auswählt, dass sie dem Stromfluss des Motors gewachsen sind. Bei der Schaltung 140562-1 aus dem Artikel „Steuerung für Gleichstrommotoren“ [1] könnten zum Beispiel gängige Standard-Dioden vom Typ 1N5401 zum Einsatz kommen, die mit bis zu 3 A belastbar sind.

[1] www.elektormagazine.de/140562

(150662)

Der Knopfzellen-Trick

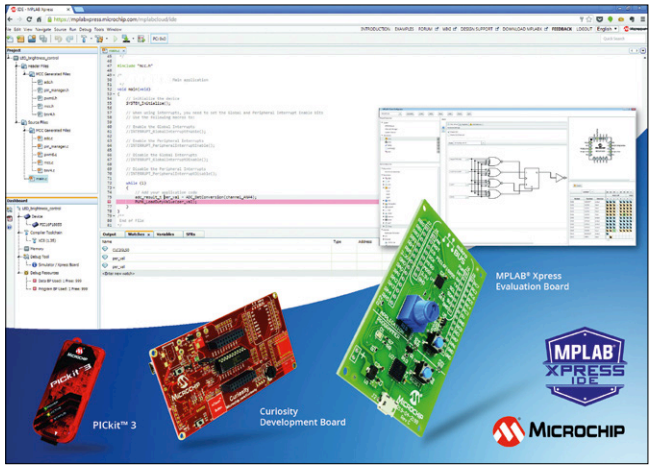
Von Jean-Robèrt Pecheur

Einige Geräte oder Projekte verwenden Knopfzellen als Energiequelle. Natürlich ist eine solche Knopfzelle immer genau dann leer, wenn man sie braucht. Irgendwann war ich es leid und habe mir eine Lösung des Problems ausgedacht: Ein einfaches, beidseitig beschichtetes und auf Maß ausgesägtes Stück Leiterplatte passt in fast jeden Knopfzellenhalter.

Lötet man an den beiden Kupferflächen Kabelstückchen an, kann man, wie im Foto zu sehen, die Schaltung zum Beispiel mit zwei AA-Zellen versorgen. Ober, wenn man ein Labornetzgerät hat, auch dieses anschließen (einstellen auf 3 V). Der Trick funktioniert auch für dickere Knopfzellen, wenn man ein paar Leiterplattenabschnittchen übereinander stapelt! ⏮

(150596)

Sie haben selbst eine clevere Lösung für etwas wirklich Fummeliges? Wenden ein Bauteil oder Werkzeug auf ungewöhnliche Weise an? Haben eine Idee, wie man ein Problem einfacher oder besser angehen könnte, als das bisher gelöst wurde? Schreiben Sie uns – für jeden Tipp, den wir veröffentlichen, loben wir 40 Euro aus!



Kostenlose Cloud-basierte PIC-Entwicklungsplattform

Die Cloud-basierte IDE von Microchip benötigt weder Downloads oder eine Registrierung noch ein Setup um die populärsten Eigenschaften der MPLAB-X-IDE auf mit dem Internet verbundene PCs, Laptops oder Tablets zu bringen. MPLAB-Xpress enthält auch eine Library von durch Microchip validierten Code-Beispielen sowie ein Interface zum MCC 3.0 (MPLAB Code Configurator) für GUI-basierte Konfiguration von MCU-Peripherie und automatischer Code-Erzeugung, integrierte MPLAB-XC-Compiler, Unterstützung für Programmer/Debugger-Hardware und 10 GB an sicherem Online-Speicher via myMicrochip-Account. Man kann eigene Projekte auch sehr einfach auf die komplette, downloadbare MPLAB-X-IDE migrieren. Zusätzlich bietet die MPLAB-Xpress-Community die Möglichkeit zum Teilen von Code, Ideen und Erfahrungen.

Das MPLAB-Xpress-Evaluation-Board bietet einen integrierten Programmer, eine MCU des Typs PIC16F18855 und einen mikroBUS-Header für Systemerweiterungen durch die über 180 Click-Boards von MikroElektronika. Die MPLAB-Xpress-IDE unterstützt auch das Entwicklungs-Board Curiosity von Microchip, ein kostengünstiges Tool mit integriertem Programmer und Debugger sowie Erweiterungsmöglichkeiten für Add-on-Boards und externe Anschlüsse. Zusätzlich kann diese Online-IDE auch mit dem beliebten In-Circuit-Debugger/Programmer PICkit 3 von Microchip verwendet werden, der über 1.000 PIC-Typen unterstützt. **(150737-1)**

Eigene Projekte mit Funksensoren für Feuchtigkeit und Temperatur

Das neue Entwicklungs-Kit *SHT31 Smart Gadget* für Feuchtigkeits- und Temperaturmessungen von Sensirion besteht aus einer Platine mit SHT31-Sensor, LCD, Taster und BLE-Modul samt Batterie. Dabei ist auch eine Kurzanleitung. Der Sensor SHT31 gehört zur neuen Generation von Feuchtigkeits- und Temperatursensoren. Er bietet höhere Intelligenz, Zuverlässigkeit und verbesserte Genauigkeit. Geboten werden zudem eine erweiterte Signalverarbeitung sowie zwei unterschiedliche, vom Benutzer wählbare I2C-Adressen. Es sind Baudraten bis zu 1 MHz möglich. Der Sensor hat ein DFN-Gehäuse mit nur 2,5 × 2,5 x 0,9 mm. Die Versorgungsspannung kann 2,4 V bis 5,5 V betragen und bei einer Messfrequenz von 1 Hz beträgt der Betriebsstrom nur etwa 2 µA. Im Ruhemodus sinkt der Strombedarf auf 0,2 µA. Die digitalen Signale des SHT31 sind voll kalibriert, linearisiert und temperaturkompensiert. Die typische Genauigkeit beträgt ±2 % RH und ±0,3 °C.

Das Entwicklungs-Kit SHT31 Smart Gadget ist bei Mouser Electronics erhältlich.

www.sensirion.com/humidity-development-kit (150737-4)

LVDS Digital Isolators für Harsh Industrial Environments

Der LVDS-Isolator ADN465x von Analog Devices eignet sich mit verbesserter Leistung, Zuverlässigkeit und Energieverbrauch für industrielle Instrumentations- und Steuerungsaufgaben. Die Lösung basiert auf der digitalen Isolationstechnik iCoupler, die für LVDS-Anwendungen dank galvanischer Isolation die Sicherheit und Zuverlässigkeit optimiert, wobei dennoch Datenraten von 600 Mbit/s bei einem Jitter von lediglich 70 ps und einer Signalverzögerung von nur 4,5 ns möglich sind. Mit den Isolatoren des Typs ADN465x lassen sich schnelle serielle LVDS-Signale direkt isolieren, ohne sie zuvor erst deserialisieren zu müssen, was das Isolieren deutlich vereinfacht.

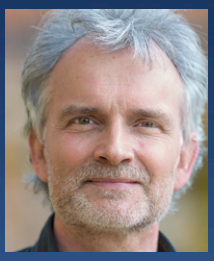
www.analog.com/en/products/interface-isolation/isolation/isolated-lvds.html (150737-3)

Advertorial

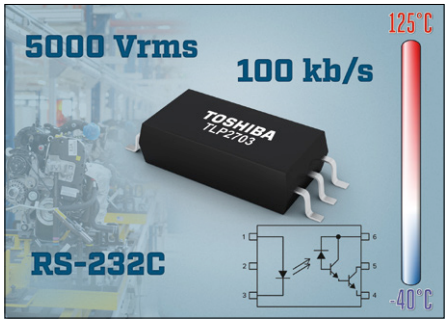
Die Rubrik ElektorBusiness in Elektor ist der Ort für Artikel, Neuigkeiten und andere Beiträge von Firmen und Institutionen, die im Bereich Elektronik tätig sind.

Redaktion ElektorBusiness: **Jan Buiting**

Beiträge nimmt er gerne entgegen unter newsdesk@elektor.com.



Optokoppler für serielle Kommunikation mit 100 kbit/s



Mit einer zulässigen Datenrate von bis zu 100 kbit/s kann der Optokoppler TLP2703 von Toshiba preislich den 1-Mbit/s-Optokopplern Konkurrenz machen. Dabei genügt ihm ein niedriger Eingangsstrom von nur 1 mA. Das neue IC bietet ein hohes Stromübertragungsverhältnis (I_C/I_F) von minimal 900 % bei einem Eingangsstrom von 0,5 mA. Die garantierte Signalverzögerung beträgt maximal 25 µs bei einem Eingangsstrom von 1,6 mA, bzw. nur 7 µs bei 12 mA. Das IC eignet sich sehr gut für die Isolation serieller Kommunikation nach RS232C. Die Isolationsstrecke beträgt 8 mm und die Isolationsspannung 5 kV_{rms}. Der Optokoppler ist auch für hohe Anforderungen an die Isolation geeignet und bietet einen Temperaturbereich von bis zu +125 °C.

www.toshiba.semicon-storage.com **(150737-6)**

Kompakte Strom-Messtrafos

Die neuen Strom-Messtrafos der CST7030-Serie im SMD-Gehäuse von Coilcraft sind lediglich 5,2 x 7 x 3 mm groß und eignen sich zur Messung von Strömen bis zu 20 A mit Frequenzen im Bereich 10 kHz bis 1 MHz. Eine typische Anwendung ist die Messung des Laststroms in Schaltnetzteilen zum Überlastschutz. Die Bauteile sind nach AEC-Q200 Grade 1 (–40 °C bis +125 °C) qualifiziert und eignen sich somit ideal für konventionelle und elektrisch angetriebene Fahrzeuge, z.B. zur Stromüberwachung von Stellmotoren oder für Batterie-Management-Systeme in 48-V-Bordnetzen.

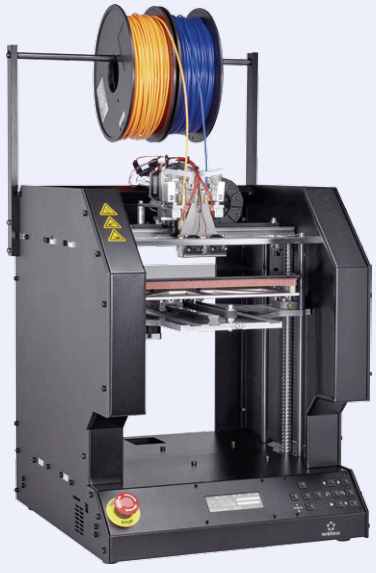
In der CST7030-Serie sind fünf Übertragungsverhältnisse von 1:20 bis 1:150 verfügbar. Der Widerstand der Primärwicklung ist mit nur 1,5 mΩ sehr gering. Die Isolationsspannung beträgt 500 V_{rms} für 1 Minute. Coilcraft hat mit der CU8965-AL-Serie auch Ausführungen mit höheren Isolationsspannungen bis 1,5 kV_{DC} im Programm. Auf Anfrage sind auch Evaluations-Samples erhältlich.

www.coilcraft.com

(150737-8)

Dual-Extruder-3D-Drucker und Steuereinheit

Mit doppeltem Extruder, doppeltem Farbdruck und der Fähigkeit zum Drucken wasserlöslicher Objekte erweitert Conrad Business Supplies sein Ein-Extruder-Modell RF1000 von 2013 jetzt um den Typ RF2000. Zusätzlich gibt es jetzt die autonome Steuereinheit 3D Printbox, die diese 3D-Drucker auch ohne PC ansteuern kann.



Im Vergleich zum Vorgänger RF1000 bietet das neue Modell RF2000 eine verbesserte Ausleuchtung des Druckbereichs, was die Kontrolle während des Druckprozesses erleichtert. Hinzu kommt eine heizbare Basis aus Glas-Keramik, die während des Drucks eine bessere Adhäsion bietet. Last not least wäre das große, kontrastreiche LCD zu nennen, wodurch die Bedienbarkeit optimiert wird. Weitere Verbesserungen betreffen das Design, das eine durch einen Stopp-Taster verbesserte Sicherheit und durch einen zusätzlichen Lüfter optimierte Kühlmöglichkeiten mitbringt.

Die 3D-Printbox fungiert als „plug and play“ Steuereinheit für die Modelle RF1000 und RF2000 sowie viele andere kompatible 3D-Drucker von Herstellern wie etwa MakerBot. Die Printbox kann in ein lokales Netzwerk oder aber auch in die kostenlose „Astroprint cloud“ integriert werden. Sie bietet genug Speicher für 3D-Modell-Daten mit hoher Auflösung. Als Zubehör gibt es auch eine Webcam, mit der der Druckprozess überwacht werden kann.

www.conrad.com **(150737-5)**

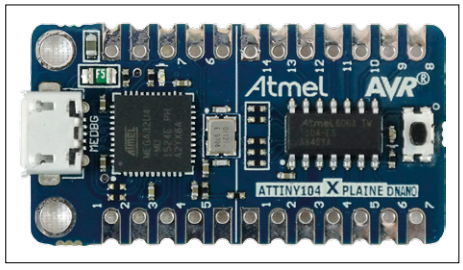
Embedded World 2016

Wieder war die größte Embedded-Messe der Welt mit rekordverdächtigen 939 Ausstellern und 30.063 Besuchern **die** Plattform für Ankündigungen neuer Produkte. Schwerpunkt war diesmal IoT und Sicherheit – ein aktuelles, weitreichendes Thema und gleichzeitig eine schwierige, aber notwendige Kombination.

Von **Viacheslav Gromov** (D)

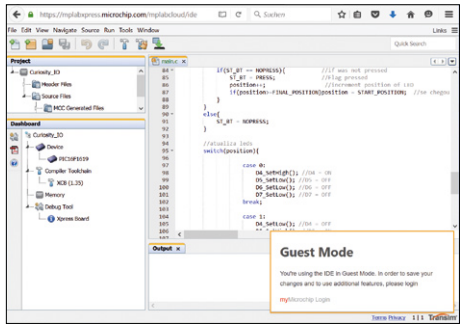
Wenn man am Eingangstor von der Menge der Besucher in die riesigen Messehallen gespült wurde, erblickte man als Erstes die größeren Stände der großen Elektronik-Unternehmen. So gut wie jeder namenhafte Halbleiterhersteller hatte irgendetwas Neues zum Thema IoT-Kommunikation (WLAN, BLE, NFC und/oder LTE), dazu Crypto-Chips und entsprechende Sicherheitssoftware dabei. Es folgt eine kleine Auswahl der spannendsten Produkte.

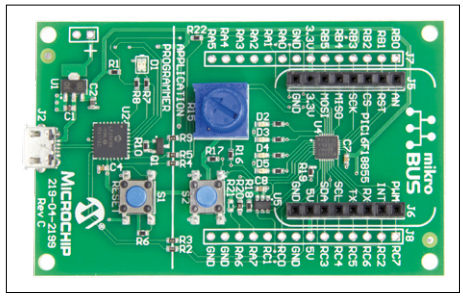
Fangen wir ganz konservativ mit der altbekannten AVR-Familie von **Atmel** an, die mit den Modellen ATtiny102 und ATtiny104 Zuwachs bekommen hat. Letztere sind ideal



für sehr kleine Anwendungen. Man erhält von allem ein wenig, beispielsweise 1 KB Flash, bis zu zwölf Pins, die übliche Peripherie und vier Stromsparmodi, was sich einerseits in einem günstigen Preis niederschlägt, andererseits aber doch für viele kleine und stromsparende Anwendungen völlig ausreichend ist. Zum Testen ist ein ATtiny104 Xplained Nano Board im Arduino Nano-Format für unter 10 € erhältlich [1]. Zum Thema Sicherheit hat Atmel den achtbeinigen I²C-Krypto-Chip namens ATECC508A vorgestellt, der ECD(S)A-, SHA-, HMAC- und ECC-Schlüssel erzeugen und Daten entschlüsseln kann [2].

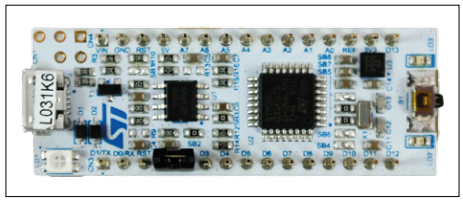
Microchip stellt stolz die neue MPLAB Xpress IDE in der Cloud vor [3]. Diese Mbed-ähnliche Entwicklungsumgebung





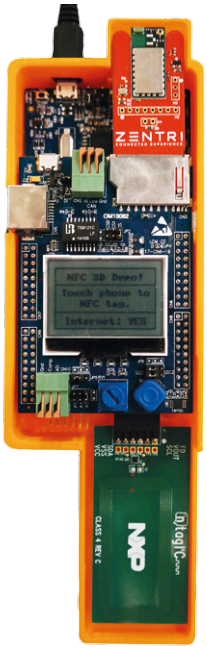
spart deutlich Prozessorleistung und benötigt kein auf dem Rechner installiertes Softwaretool. Die Xpress-IDE lässt sich mit vielen Beispielprogrammen im Browser (noch einfacher als mit der normalen MPLAB X IDE) bedienen. Nach der Anmeldung/Registrierung stehen dem Anwender 10 GB Onlinespeicher für Projektdateien zur Verfügung. Bis zum Sommer sollen voraussichtlich alle PIC-MCUs von dieser IDE unterstützt werden.
Der hilfreiche Code-Generator ist bereits eingebettet. Zu dieser IDE gibt es zu einem Preis von wenig mehr als 10 € ein so genanntes Xpress-Board für den Einstieg in die neueste PIC16F18855-MCU (20 MHz, 14 KB Flash, 1 KB SRAM und 256 B EEPROM) mit ein wenig Peripherie in Form von Tastern, vier LEDs und einem Potentiometer [4]. Das Board meldet sich wie ein USB-Speichermedium beim PC an. Die von der Cloud-IDE erzeugte Hex-Datei kann einfach darin abgelegt werden, um die Firmware auf die MCU zu übertragen.

Dass **STMicroelectronics** an seiner (Mbed-)Nucleo-Board-Familie stark gefeilt hat, war am Stand sofort ersichtlich. Immer mehr günstige Extension-Boards sind erhältlich – von WLAN über Motorsteuerung bis zu einem Lichtsensor. Es gibt so ziemlich alles, was eine schnelle Prototyping-Phase verspricht. Abgesehen davon und auch unabhängig von der sich ständig vermehrenden STM32-Familie sind die zwei neuen Boards Nucleo-32 und Nucleo-144 angekündigt. Beide Boards verfügen wie die Nucleo-64 über einen ST-Link/V2-Debugger, nur sind unterschiedliche MCU-Typen darauf. Da die 32-Pin-Boards nach dem Format des Arduino Nano gelayoutet sind, kann man sie in sehr kompakte Anwendungen packen. Die 144-Pin-Boards sind für deutlich anspruchsvollere Projekte gedacht. Eine Übersicht aller Nucleo-Boards fin-



den Sie unter [5]. Die Nucleos-32 sind für rund 10 € erhältlich, die Nucleos-144 für knapp 22 €.

Das spannende und neue „LPC54114 Audio and Voice Recognition“-Kit von **NXP** [6] mit dem LPC53114-Mikrocontroller (100 MHz, 256 KB Flash, 192 KB SRAM), der auf einem ARM-Cortex-M4F als Haupt- und einem ARM-Cortex-M0+ als Hilfskern basiert, zeigt, wie man mit Spracherkennung einen Prozess in Gang setzen kann. Dazu befindet sich im Kit ein Mikrofon/Codec/OLED-Shield mit allem, was nötig ist, um das Signal des digitalen Mikrofons zu verarbeiten und um Informationen auf dem Display anzuzeigen.
Zum Thema „IoT und Sicherheit“ stellt NXP das „LPC43S67-A70CM Cloud Connectivity“-Kit [7] vor. Im Kit enthalten sind ein LPCXpresso43S67-Board

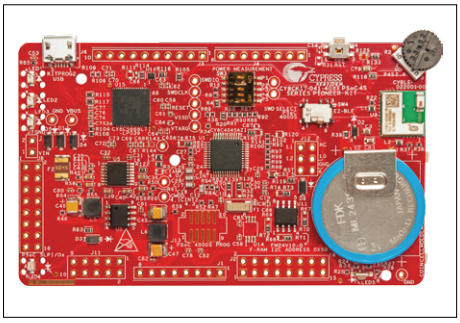


mit USB-Kabel, ein „General Purpose Shield“ und eine WLAN- sowie NFC-Erweiterung. Auch die Lizenz für das ZentriOS, das entsprechende SDK und viele weitere Zentri-Tools gehören zum Kit. Diese Oberfläche erlaubt es, sichere IoT-Anwendungen umzusetzen. Sicherheit wird aber nicht nur von der Software gewährleistet, sondern auch von der Sicherheits-MCU LPC43S67 (204 MHz, M4F + M0+) und einem Sicherheits-IC namens A70CM an Board. Für knapp 90 € ist dieses Kit für viele (sensible) IoT-Projekte geeignet.

Texas Instruments unterstützt die schon länger bekannten (MSP430-) LauchPads mit den immer umfangreicheren BoosterPacks intensiv, vor allem in Hinblick auf das IoT. Unter [8] finden Sie zahlreiche, meist kostenlose Möglichkeiten, Lauchpads kinderleicht mit spezieller Firmware und APIs an das Internet beziehungsweise die Cloud anzubinden.

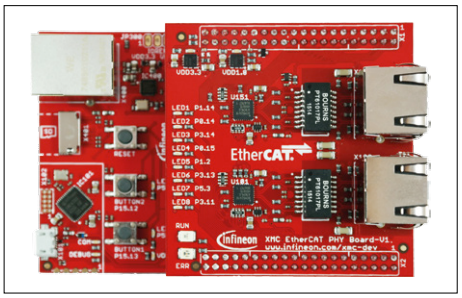
Ob Amazon-Cloud oder die Dienste von IBM – alles lässt sich auf einfache Weise einbinden. Sehr überzeugend und einfach gestaltet ist die Seite Temboo [9], die (nach kurzer Registrierung) unzählige fertige Programme für das CC3200-LaunchPad mit integriertem WLAN oder auch für die anderen TI-LaunchPads mit dem aufgestecktem CC3100-BoosterPack zur Verfügung stellt, übrigens auch für den Arduino! Von der Paket-Sendungsverfolgung über Yahoo-Wetter bis zu Dropbox-Dateien - alles wird für das LaunchPad zugänglich, wenn es mit dem Internet verbunden ist. Sie müssen lediglich den angezeigten Code in die kostenlose Energia-IDE (ähnlich der Arduino-IDE) einfügen und laufen lassen. Selbstverständlich können Sie den Code eigenen Wünschen anpassen oder erweitern. Der Energia-Code lässt sich problemlos in das weit professionellere Code Composer Studio (CCS oder CCS Cloud) importieren.

Außer einem neuen Update des PSoC-Creators hat **Cypress** die bekannte PSoC-Familie, deren Peripherie per Mausklick zusammenstellen kann, um die 4S-Unterfamilie [10] erweitert, die auf dem ARM-Cortex-M0+-basiert. Sie verfügt gegenüber den PSoC-4-Modellen in manchen Bereichen über eine leistungsfähigere Peripherie, besonders was die Verarbeitung analoger Signale betrifft. Dazu gibt es Besonderheiten wie spezielle GPIO-Pins, die über vorgeschaltete „Look Up Tables“ (LUTs) verfügen, die (einfache) frei-programmierbare Logik-Funktionen



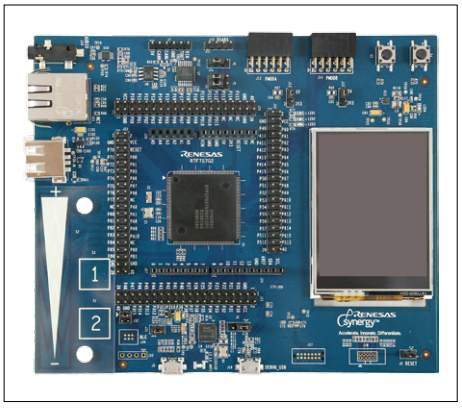
erlauben. Auch die „Cap Sense“-Technologie wurde bei diesen MCU-Typen deutlich verfeinert.
Für das IoT stellt Cypress ein spannendes „Solar Powered IoT Device“-Kit [11] für rund 50 € vor. Es besteht im Wesentlichen aus zwei Platinen. Eine hat die Form eines USB-Sticks für den Computer und die andere ist mit einer Solarzelle ausgestattet. Beide Seiten enthalten einen PRoC-BLE-Chip (**B**luetooth **L**ow **E**nergy), die Platine mit der Solarzelle das Energy-Harvesting-IC S6AE101A, so dass diese Platine samt BLE nur vom Umgebungslicht mit Energie versorgt werden kann. Dies zeigt natürlich auch, wie stromsparend die sehr flexiblen PRoC-BLE-Bausteine sind.

Infineon war auf die neuen Mitglieder XMC4800 und XMC4300 der ARM-Cortex-M-XMC-MCU-Familie sichtlich stolz. Highlight dieser beiden Typen, die in verschiedensten Ausführungen zu haben sind, ist die Unterstützung von EtherCAT (mit



Slave-ICs). Dies ist eine in der Industrie sehr verbreitete Schnittstelle für relativ hohe Datenübertragungsraten. Dennoch verfügen diese Cortex-M4F-basierten 144-MHz-Mikrocontroller über fast dieselbe umfangreiche Peripherie wie viele andere XMC4000-Vertreter. Sie sind mit reichlich Schnittstellen gesegnet wie CAN, USB und Ethernet und verfügen über ausreichend anderweitige Peripherie wie Timer oder ADCs. Für den leistungsfähigeren Typ XMC4800 mit maximal 2 MB Flash, 352 KB RAM und 8 KB Cache gibt es das „XMC4800 Relax EtherCAT“-Kit [12] mit einem zweistöckigen Board für rund 50 €. Es enthält alles, was man zum Testen der MCU-Funktionen, besonders aber des EtherCATs, benötigt.

Renesas hat das erweiterte und verbesserte „Ökosystem“ der relativ neuen Synergy-32-bit-ARM-Cortex-M4/0+-Mikrocontrollerfamilie vorgestellt, die je nach Typ bis 240 MHz getaktet werden kann und die auch in punkto Peripherie im Vergleich zur RL78-Familie deutlich besser aufgestellt ist. Sie beherrscht unter anderem alle gängigen Schnittstellen wie USB 2.0 oder Ethernet, Touch-Funktionen und Grafik-LCD-Ansteuerung. Dank vieler neuer Tools wie einem Code-Generator, die im bekanntem e²studio integriert sind, ist der Umgang mit dieser Mikrocontrollerfamilie sehr leicht. Die Synergy-Gallery im Internet [13] bietet viele kostenlose Bibliotheken und Beispielprogramme. Eine



gute Möglichkeit für den Einstieg in diese Synergy-Welt stellt das SK-S7G2-Starterkit für etwa 74 € [14] dar. Das Kit bietet sehr viel Peripherie und die dazu gehörende Dokumentation ist gut.
Mit dem neuen RL78/G1D wurde ein BLE-4.1-Funkchip mit einer On-board-CPU (32 MHz, 256 KB Flash, 20 KB SRAM und 8 KB Data Flash) vorgestellt, der durch sein besonderes Konzept ganz neue Maßstäbe beim Stromverbrauch setzt [15].

Silicon Labs erweitert das Angebot an Funk-ICs um die Module Blue Gecko BGM111 [16] BLE 4.1 und Wizard Gecko WGM110 [17]. BGM111 ist mit einem ARM-Cortex-M4F-Mikrocontroller (256 MB Flash und 32 KB RAM) und damit mit der üblichen, und – wenn man sich den DC/DC-Wandler anschaut - sogar besonderen Peripherie ausgestattet. Der WGM110 hat ähnliche Eigenschaften, nur basiert er auf einer ARM-Cortex-M3-MCU (1 MB Flash und 128 KB RAM). Beide Module können in den meisten Anwendungsfällen ohne zusätzliche Mikrocontroller auskommen. Sie sind auch je nach Betriebsmodus mehr oder weniger stromsparend. Der BGM111 kann durch Änderung der Software auch zu einem BLE-4.2-Modul mutieren. Für beide Module gibt es Starterkits.

Um die schon länger erhältliche EVE-Touch-LCD-Treiber-Familie für Einzel- und Kleinabnehmer noch interessanter zu machen, versucht der USB-Spezialist **FTDI** durch Crowdfunding die Plattform CleO beziehungsweise NeO zu entwickeln. Und es läuft gut, erste Prototypen sind bereits vollfunktionsfähig. CleO35 ist eine Platine mit einem 3,5 Inch-HVGA-TFT-Touchdisplay (320x480 Pixel), einem MicroSD-Kartenslot sowie mehreren Anschlüssen für eine optionale Kamera, Lautsprecher und Mikrofon. Das Board ist wie ein Arduino-Shield aufgebaut, da es trotz der 32-bit-on-board-MCU FT903 und dem EVE810-Display-, Touch- und Audio-Controller von außen angesteuert werden muss. Da die ICs die Hauptarbeit übernehmen, kann man dazu einen Arduino Uno mit wenig Rechenleistung verwenden. Da man bei einem normalen Uno noch Stiftleisten für den Anschluss an CleO verlöten muss, wurde der vollständig Uno-kompatible NeO mit schon angelöteten Stiftleisten extra für das CleoO-Board entwickelt. Zum Programmieren eignet sich die Arduino-IDE, in die die CleO-Bibliothek eingebunden ist. Mit dieser Bibliothek ist es ein Kinderspiel, Grafiken, Touch-Elemente oder Fotos von der SD-Karte auf dem Display ohne Vorkenntnisse anzuzeigen. Wenn Sie und genügend andere Interessenten das Projekt unterstützen, kann es spätestens Mitte dieses Jahres in die Serienproduktion gehen [18]. Für 63 $ gibt es den CleO und NerO zusammen. ⏮

(150707)

Weblinks

[1] www.atmel.com/tools/attiny104-xnano.aspx
[2] www.atmel.com/devices/ATECC508A.aspx
[3] www.microchip.com/mplab/mplab-xpress
[4] http://ww1.microchip.com/downloads/en/DeviceDoc/MPLAB%20Xpress.pdf
[5] www.st.com/web/catalog/tools/FM116/SC959/SS1532/LN1847
[6] www.nxp.com/products/microcontrollers-and-processors/arm-processors/lpc-cortex-m-mcus/lpc-cortex-m4-single-multi-core/lpc54000-series/lpc54114-audio-and-voice-recognition-kit:OM13090?tid=vanOM13090
[7] www.nxp.com/products/identification-and-security/authentication/lpc43s67-a70cm-cloud-connectivity-kit:OM13086
[8] www.ti.com/ww/en/simplelink_embedded_wi-fi/ecosystem.html
[9] www.temboo.com/library/
[10] www.cypress.com/products/32-bit-arm-cortex-m0-psoc-4
[11] www.cypress.com/documentation/development-kitsboards/s6sae101a00sa1002-solar-powered-iot-device-kit
[12] www.infineon.com/cms/de/product/evaluation-boards/KIT_XMC48_RELAX_ECAT_V1/productType.html?productType=5546d46250cc1fdf0150f6bdd1236ec8
[13] https://synergygallery.renesas.com/auth/login
[14] http://am.renesas.com/products/tools/introductory_evaluation_tools/renesas_starter_kits/sk_s7g2/index.jsp
[15] www.renesas.com/products/mpumcu/rl78/rl78g1x/rl78g1d/
[16] www.silabs.com/products/wireless/bluetooth/Pages/BGM111-bluetooth-smart-module.aspx
[17] www.silabs.com/products/wireless/wi-fi/wi-fi-modules/Pages/wgm110-wi-fi-module.aspx
[18] www.indiegogo.com/projects/cleo-the-smart-tft-display-for-arduino#

Willkommen bei DESIGN

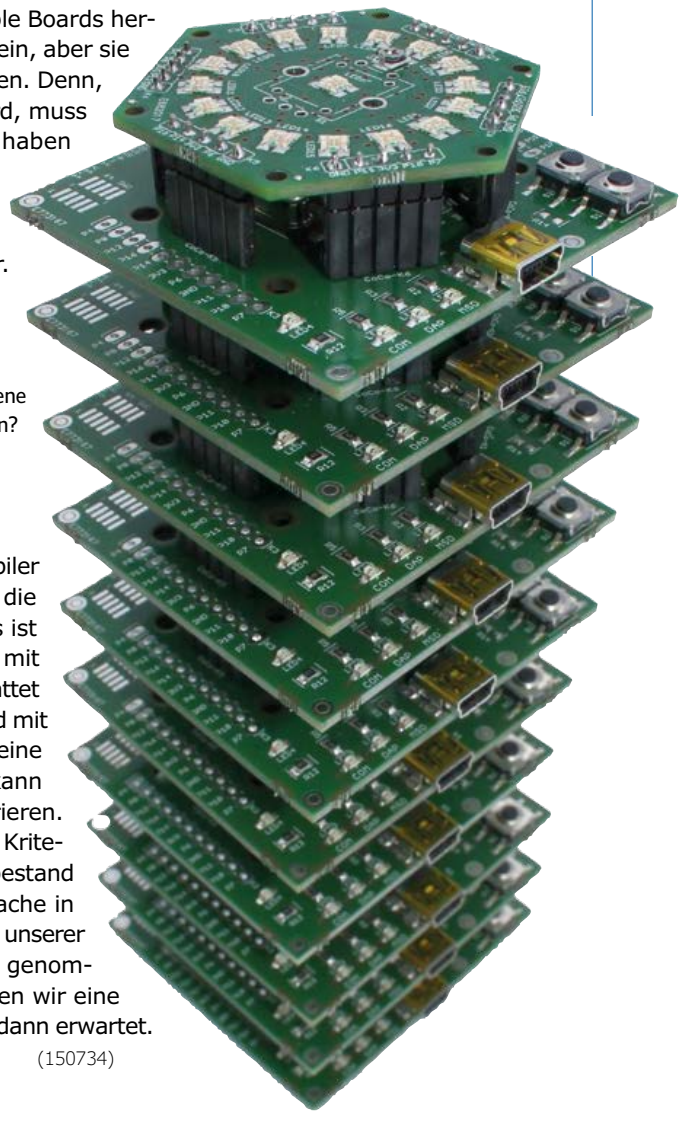
Von **Clemens Valens,** Elektor Labs

Gibt es ein Leben nach Arduino?

Arduino hat in den letzten Jahren eine Menge Aufmerksamkeit auf sich gezogen. So viel, dass andere Plattformen zum schnellen Prototyping (mit Ausnahme vom Raspberry Pi) etwas von diesem Hype verdeckt wurden. Dennoch gibt es einige andere interessante Plattformen, die es geschafft haben, eine ganze Community aufzubauen. mbed von ARM ist eine solche Plattform. Elektor hat mbed vorgestellt, als es im Jahr 2010 ins Leben gerufen wurde, und einen äußerst erfolgreichen Wettbewerb mit Tausenden von Teilnehmern durchgeführt.

Aber, wie es manchmal so kommt, wurde etwa zur gleichen Zeit Arduino und kurz danach Raspberry Pi mit Pauken und Trompeten eingeführt, so dass mbed bei den meisten Menschen vom Schreibtisch fiel. Doch anstatt sich still in eine Ecke zum Sterben zurückzuziehen, wurde mbed fortgesetzt und entwickelte sich langsam von einer Single-Board-Plattform zum Multi-Vendor-Multi-Target-Ökosystem, das von vielen Halbleiterherstellern wie NXP/Freescale, ST, Atmel, Renesas und Silicon Labs unterstützt wird. Arduino und mbed sind beide Open-Source und Open-Hardware. Der Hauptunterschied liegt darin, dass Arduino mehr oder weniger unkontrolliert „ins Kraut geschossen" ist, während mbed eher geordnet von Unternehmen weiterentwickelt wurde. Die Industrie hat viele Arduino-kompatible Boards hervorgebracht, nur um ein Teil der Bewegung zu sein, aber sie scheint auch bereit zu sein, in mbed zu investieren. Denn, damit ein Board als „mbed-fähig" akzeptiert wird, muss es Tests und Prüfungen bestehen. Bei Elektor haben wir uns entschieden, durch diesen Feuerreifen zu springen. Werden wir Erfolg haben? Bleiben Sie dran und lesen Sie über unsere Erfahrungen und das Schicksal des ersten mbed-Boards von Elektor.

https://developer.mbed.org



Werden diese zehn Boards das angesehene „mbed-enabled"-Label erhalten?

mbed-enabled, erste Schritte

mbed besteht aus einem kostenlosen Online-Compiler und einem Board mit einem ARM-Prozessor, der die Binaries, die der Compiler erzeugt, ausführt. Dies ist nicht so schwer zu realisieren, so lange das Board mit einem der unterstützten ARM-Prozessoren ausgestattet ist. Die nächste Stufe ist „mbed-enabled". Ein Board mit diesem Etikett wird vom Online-Compiler durch seine eindeutige ID (an)erkannt. Nur das mbed-Team kann solche IDs vergeben und sie in seine Tools integrieren. Und das geschieht nur, wenn das Board bestimmte Kriterien erfüllt, die das Team überprüft. Unser Board bestand die Tests in unserem Labor, doch nun liegt die Sache in fremden Händen. Genauer gesagt: zehn Exemplare unserer Hardware werden derzeit von ARM in die Mangel genommen. Wenn die Hardware akzeptiert wird, kommen wir eine Runde weiter, aber wir wissen noch nicht, was uns dann erwartet. Fortsetzung folgt… ⏮ (150734)

LEDitron

Led-Filamente als 7-Segment-Display

Von **Ilse Joostens** und **Peter S'heeren** (Belgien)

Die alte Glühbirne ist tot, es lebe die LED-Lampe! Die LEDitron verwendet so genannte LED-Filamente, um genau wie bei einer Numitron-Röhre ein 7-Segment-Display zu realisieren. Aber viel energiesparender!

Eigenschaften

- Ausgangsspannung einstellbar 35...100 V
- Maximaler Ausgangsstrom 100 mA
- Eingangsspannung 12 V, 1 A
- Geeignet für 4...6 LED-Filament-Displays (abhängig von eingestellter Helligkeit)
- Entwurf ausschließlich mit Durchsteck-Bauteilen

Das Aufkommen der Energiesparlampen in den 80er Jahren ist ja noch nicht so lange her. Die ersten Compact Fluorescent Lamps (CFL) waren noch lange nicht so kompakt wie ihr Name behauptet, sie sahen eher aus wie Marmeladengläser mit Schraubgewinde. Durch das schrittweise EU-Verbot von Glühbirnen von 2008 an wurde nicht nur die Entwicklung von Energiesparlampen stark vorangetrieben, um das Jahr 2010 kamen auch LED-Lampen als Konkurrent und letztendlich als Ersatz von CFLs auf den Markt.

Zurzeit findet man energiesparende LED-Lampen mit so genannten Filamenten oft im Einkaufskorb des Verbrauchers, die den klassischen Glühlampen sehr ähneln (**Bild 1**), ja, es gibt sogar Modelle auf dem Markt, die so aussehen wie die alten Kohlefadenlampen.

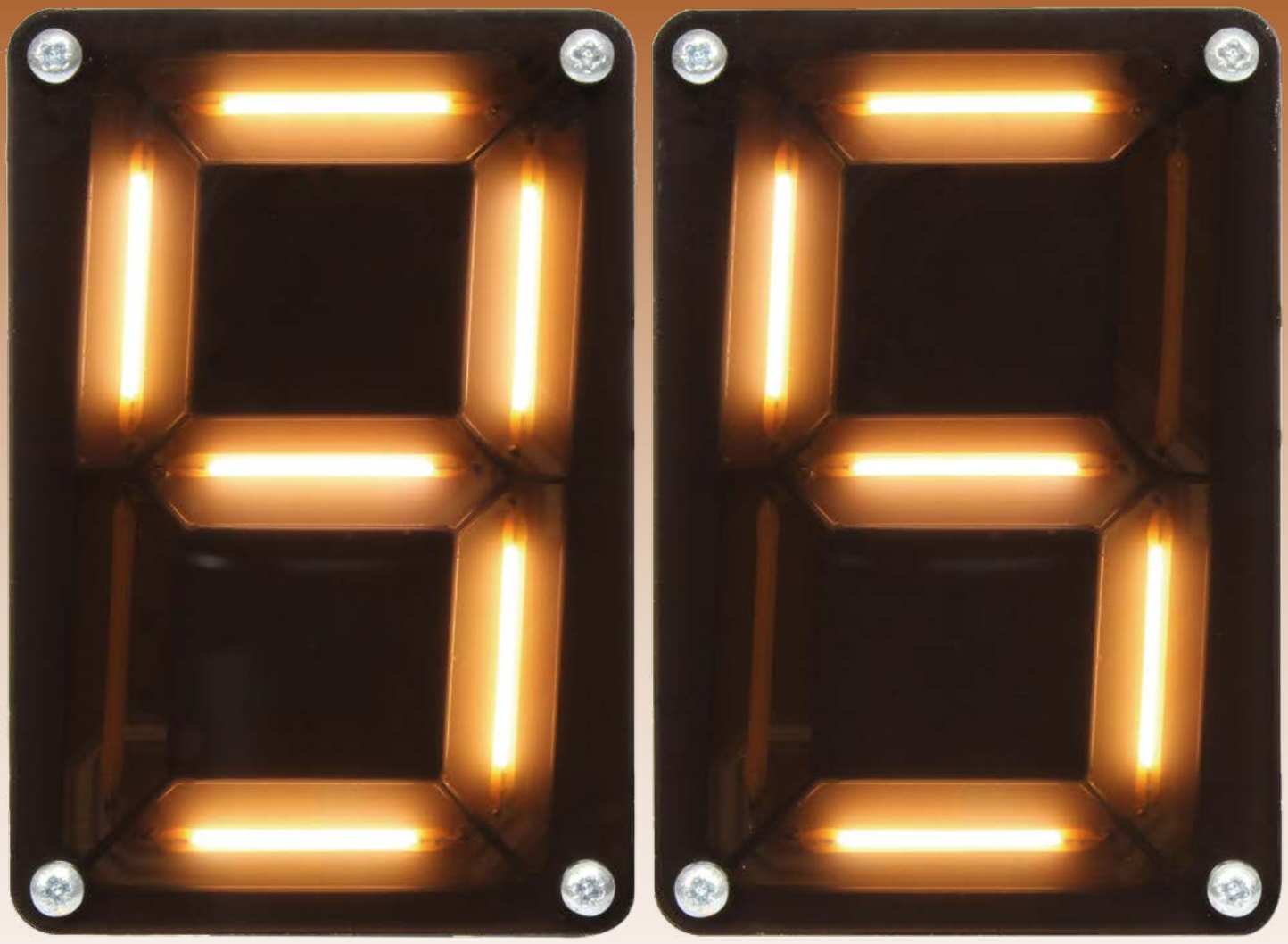
Wer sich ein wenig mit altertümlicher Elektronik auskennt, dem sind bestimmt die Nixie- und die Numitron-Röhren ein Begriff. Eigentlich handelt es sich dabei um Glühlampen mit mehreren Leuchtfäden, die gemeinsam ein 7-Segment-Display bilden. Seit einiger Zeit findet man bei meist chinesischen Anbietern via eBay & Co. LED-Filamente, die „lose" zu kaufen sind (**Bild 2**), was das Experimentieren und vor allem den hier beschriebenen „Missbrauch" herrlich einfach macht. Numitrons haben nämlich außer ihrem hohen Energiebedarf einen weiteren großen Nachteil: Sie sind ziemlich klein! Viel größere und exakt gleich aussehende 7-Segment-Displays lassen sich aber gut mit LED-Filamenten herstellen. Darüber hinaus sind diese Displays sehr gut ablesbar, selbst im grellen Sonnenlicht.

So funktionieren LED-Filamente

Die Erfindung der blauen LED mit großer Helligkeit durch Shuji Nakamura (Nichia Corporation) im Jahr 1993 machte es möglich, LEDs für Beleuchtungszwecke einzusetzen. Eine weiße LED ist nichts anderes als eine superhelle blaue LED, die mit einem phosphoreszierenden Stoff beschichtet ist, der das blaue Licht in ein Farbenspektrum umwandelt, das dem menschlichen Auge als weiß erscheint.

LED-Leuchtfäden sind nichts anderes als eine Weiterentwicklung der blauen LED. Ein Filament besteht aus einem länglichen Substrat, meist aus Kunststoff, Glas oder Saphir, auf dem eine große Zahl blauer LEDs in Reihe geschaltet ist. Wenn man den Strom durch ein Filament sehr stark begrenzt, kann man die einzelnen LEDs auf dem Streifen gut erkennen (**Bild 3**). Diese Anordnung wird mit einer Phosphorschicht bedeckt und mit Anschlussdrähten an den Enden versehen, so dass eine Montage in „Glühbirnen" durch Punktschweißen möglich ist.

Bei der großen Zahl blauer, in Reihe geschalteter LEDs ist natürlich eine hohe Betriebsspannung (meist um die 70 V) vonnöten, um die gemeinsame Vorwärtsspannung zu überwinden. Bei netzspannungsversorgten LED-Lampen ist das eher ein Vorteil, weil die notwendige Elektronik im Lampenfuß recht bescheiden ausfallen kann. In unseren Experimenten



müssen wir aber der hohen Betriebsspannung Rechnung tragen.

Versorgung und Steuerung der Displays

Grob gesagt besteht die Schaltung aus drei Teilen (**Bild 4**): einem Aufwärtswandler, der die Versorgungsspannung von 12 V auf 70...80 V umsetzt, einem 7-Segment-Treiber und zwei Anodentreibern. Die Ausgangsspannung des Aufwärtswandlers ist mit P1 im Bereich von ungefähr 35...100 V einstellbar, wobei der maximale Ausgangsstrom rund 100 mA beträgt. Da der Ausgangskondensator eine nominale Arbeitsspannung von 100 V besitzt, ist es besser, die Ausgangsspannung nicht höher als ungefähr 80...85 V zu wählen, was für unsere Anwendung aber völlig ausreichend ist.

Als Boost-Wandler arbeitet ein gewöhnliches 555-Timer-IC in der Schaltung. Der ICM7555 ist recht gut und preiswert erhältlich, außerdem schön stabil und nicht anspruchsvoll, was das Platinenlayout angeht.

Selbstredend haben wir das Layout der Platine so weit wie möglich optimiert. Die Verbindungleiterbahnen zwischen Eingangskondensatoren, MOSFET, Spule, Diode und Ausgangskondensator sind so kurz und so breit wie möglich gehalten. Der Feedback-Kreis mit R38, R10, P1 und T1 ist so weit von den Platinenbereichen, in denen große Ströme fließen, entfernt, dass keine Störungen auftreten können.

Bild 1. LED-Filamente in einer „Glühlampe“ in Kerzenform.

Bild 2. Lose Filamente fordern zum Experimentieren heraus.

Bild 3. Wenn der Strom sehr stark begrenzt wird, kann man die einzelnen LEDs auf dem Substrat sehen.

Als Ausgangskondensatoren werden Low-ESR-Typen verwendet. Das Ergebnis: eine Schaltung, die trotz ihres einfachen Aufbaus gute Eigenschaften zeigt und auch bei einer hohen Belastung von 100 mA im wahren Sinne des Wortes cool bleibt. Bei einer Ausgangsspannung von 82 V und einer Last von 820 Ω fließen etwa 100 mA (8,2 W) zur Last, während am Eingang bei 12 V ungefähr 790 mA fließen (9,48 W). Der Wirkungsgrad beträgt also mehr als 86 %.

Da die Platine fest montiert und ja Teil eines größeren Geräts werden soll, haben wir auf einen Verpolungsschutz verzichtet. Bei dem recht hohen Strom könnte eine Diode beträchtlich warm werden, was dem Wirkungsgrad abträglich wäre, und eine Lösung mit p-Kanal-MOSFET wäre doch übertrieben (und kostenintensiv). Es gibt allerdings eine 2-A-Polyfuse am Eingang.

Die LED-Filamente werden in dieser Schaltung als Display mit gemeinsamer Anode angeschlossen. Die Segment- beziehungsweise Kathodentreiber sind mit den „Hochspannungs“-NPN-Transistoren MPSA42 aufgebaut. Ein logisches High an einem Pin 1...7 von K2 schaltet über den Transistor das entsprechende LED-Segment ein, wobei der Strom durch R2...R8 begrenzt wird. Durch eine Änderung der Spannung am Boost-Wandler oder der Werte von R2...R8 kann man die Leuchtkraft der Anzeige anpassen. Eventuell ist dies auch softwaremäßig zu erreichen, indem man das Tastverhältnis der Ansteuersignale ändert.

Schließlich gibt es noch zwei Anodentreiber, damit man zwei Displays gemultiplext betreiben kann. Diese Treiber bestehen aus PNP/NPN-Pärchen MPSA42 und MPSA92. Wenn man mehr als zwei Displays einsetzen möchten, kann man einfach zusätzliche Anodentreiber auf einem Stückchen Lochraster aufbauen und in die Schaltung aufnehmen. Es lassen sich,

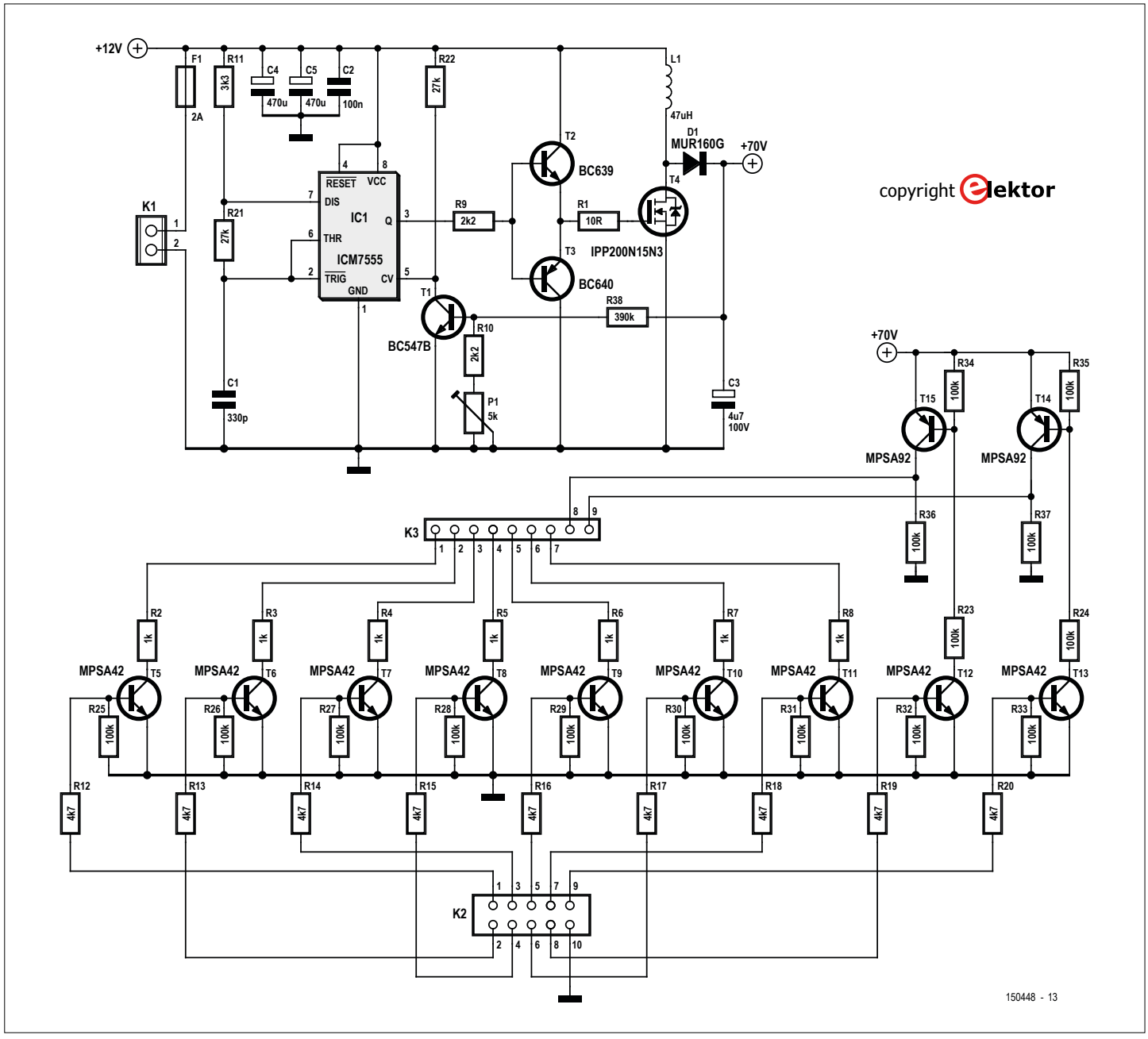


Bild 4. Das Schaltnetzteil ist sehr effizient und erzeugt wenig Abwärme.

Stückliste Netzteilplatine

Widerstände:
R1 = 10 Ω
R2...R8 = 1 k
R9, R10 = 2k2
R11 = 3k3
R12...R20 = 4k7
R21,R22 = 27 k
R23...R37 = 100 k (R36, R37 optional)
R38 = 390 k
P1 = Trimmpoti 5 k

Kondensatoren:
C1 = 330 p
C2 = 100 n
C3 = 4,7 µ/100 V, low ESR
C4,C5 = 470 µ/25 V, low ESR

Spulen:
L1 = 47 µ (Würth WE-PD3 744 591 47)

Halbleiter:
D1 = MUR160G
T1 = B547B
T2 = BC639
T3 = BC640
T4 = IPP200N15N3 G
T5...T13 = MPSA42
T14,T15 = MPSA92
IC1 = ICM7555

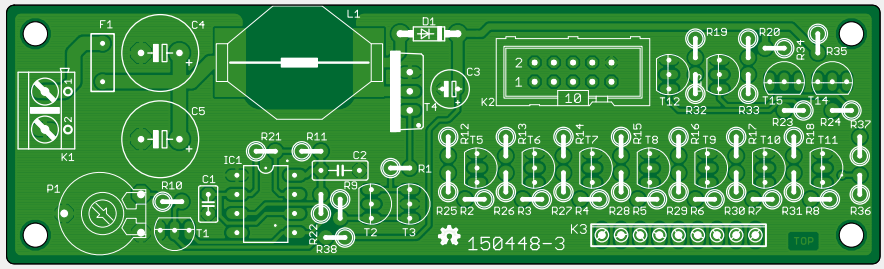
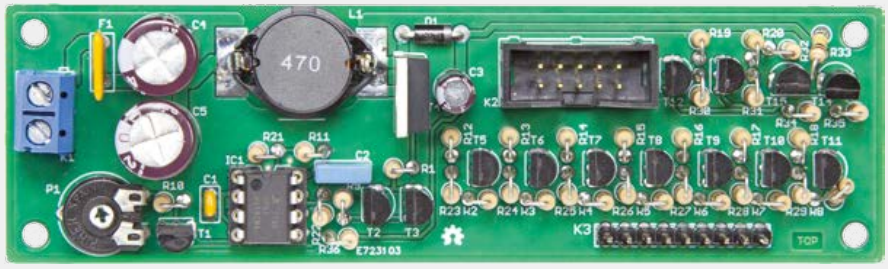


Bild 5. Die kleine Platine erleichtert den Aufbau der Schaltung



Außerdem:
F1 = 1,85 A/30 V MCC33161
K1 = 2-polige Platinenanschlussklemme, RM 5 mm
K2 = 2x5-polige Stiftleiste mit Wanne, RM 2,54 mm
K3 = 1x9-polige Stiftleiste
Platine 150448-3

abhängig von der gewünschten Helligkeit, maximal vier bis sechs Displays an diesem Netzteil betreiben.
Am den Aufbau zu vereinfachen, haben wir für die Steuerung der LEDs die Platine in **Bild 5** entworfen, die LED-Filamente werden auf einer eigenen Platine untergebracht (Schaltung in **Bild 6,** Platinenlayout in **Bild 7**). Für einen Aufbau wie im Titelbild gibt es zudem eine Zwischenplatine (**Bild 8**). Alle Platinenlayouts können von der Projektseite [1] kostenlos heruntergeladen werden.

Software

Man kann die LED-Filament-Displays an schon existierende Schaltungen anschließen, die 7-Segment-Displays verwenden, eventuell, wenn nötig, über logische Inverter. Natürlich kann man zur Ansteuerung auch einen Mikrocontroller, einen Arduino oder sogar einen Single-Board-Computer verwenden.
Mit Hilfe eines Arduinos und den Programmbeispielen des früher veröffentlichten VFD-Shields (Elektor September 2015 [2]) kann man mit diesen Displays eine Uhr, ein Thermometer oder ein Voltmeter realisieren. Dabei muss allerdings das Timing für die Ansteuerung der Displays etwas angepasst werden, damit die Displays nicht flackern. Diese angepassten Beispiel-Sketches mit einer von 250 Hz auf 500 Hz erhöhten Wiederholfrequenz sind ebenfalls unter [1] zu finden.

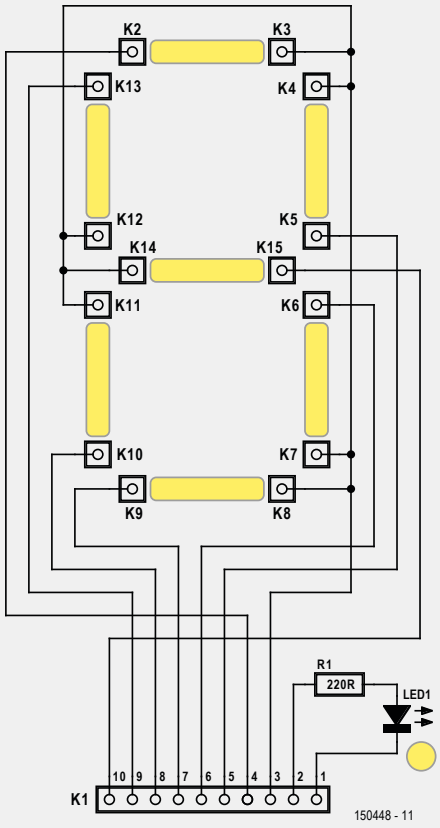
Aufbau der 7-Segment-Displays

Die Anschlüsse der LED-Filamente sind nicht lötbar, aber sie können am leichtesten mit den Buchsenkontakten aus einer IC-Fassung oder einer Buchsenleiste montiert werden. Die Buchsenkontakte steckt man entweder in eine (Lochraster-) Platine oder eine schwarze Acrylglasscheibe (vorgebohrt 3 mm) und lötet auf der Rückseite normalem Kupferdraht an. Dann biegt man die Anschlussstreifen der LED-Filamente um 90 ° ab und drückt sie ganz vorsichtig in die Buchsen. Achten Sie gut auf die Polarität; die Anode eines LED-Filaments ist durch einen roten Punkt oder durch ein Löchlein im Anschlussstreifen markiert. Als Dezimalpunkt kann man eine weiße oder gelbe LED verwenden.
Wir haben übrigens die Erfahrung gemacht, dass „lose“ LED-Filamente mechanisch sehr empfindlich sind und oft den Transport aus dem fernen China nicht überleben. Eine sicherere Alternative kann es sein, im Baumarkt eine Lampe mit LED-Filamenten zu kaufen und diese vorsichtig (!) auszuschlachten.
Um die Lesbarkeit zu erhöhen, sollte man die LED-Filamente untereinander optisch abschirmen, beispielsweise mit schwarzem Acrylglas, und auch ein optisches Filter darauf zu montieren. Beim Filter haben wir mit 3-mm-Acrylglas der Farbe „umbra“ gute Erfahrungen gemacht. Zudem erhielten die Displays durch diese graubraune Abdeckung ein schönes Retro-Design. Natürlich können Sie auch mit anderen Farben und Acrylglas-Ausführungen experimentieren.
Wer an eine Laserschneidemaschine kommt, kann mit den Gerber-Dateien aus [1] aus dem gewünschten Material ein 7-Segment-Display in der Größe von ungefähr 67 × 100 mm² anfertigen (lassen). Ein Bausatz mit Umbra-Plexiglas und einer schwarzen Displayplatine wird ebenfalls im Elektor-Shop angeboten.

Aufbau und Test der Schaltung

Der Aufbau der Schaltung ist recht einfach. Man beginnt mit den kleinen Bauteilen wie Spule, Diode, Trimmpoti, IC-Füßchen und keramischen Kondensatoren. Dann sind die Widerstände an der Reihe, gefolgt von den Transistoren in den TO92-Gehäusen, den Verbindern, dem 4,7-µF-Ausgangskondensator und der Polyfuse. Zum Schluss verlötet man die hohen Bauteile wie die Eingangskondensatoren und den MOSFET, steckt IC1 in seine Fassung und dreht P1 ganz nach links.
Nun verbinden Sie die Schaltung mit einer 12-V-Spannungsquelle, die mindestens 1 A liefern kann. Messen Sie die

Stückliste 7-Segment-Display



Widerstand:
R1 = 220 Ω

Halbleiter:
LED1 = LED 5 mm

Außerdem:
K1 = 10-polige Buchsenleiste, RM 2,54 mm
Platine 150448-1

Bild 6. Die sieben LED-Filamente werden über einen zehnpoligen Verbinder angesteuert ...

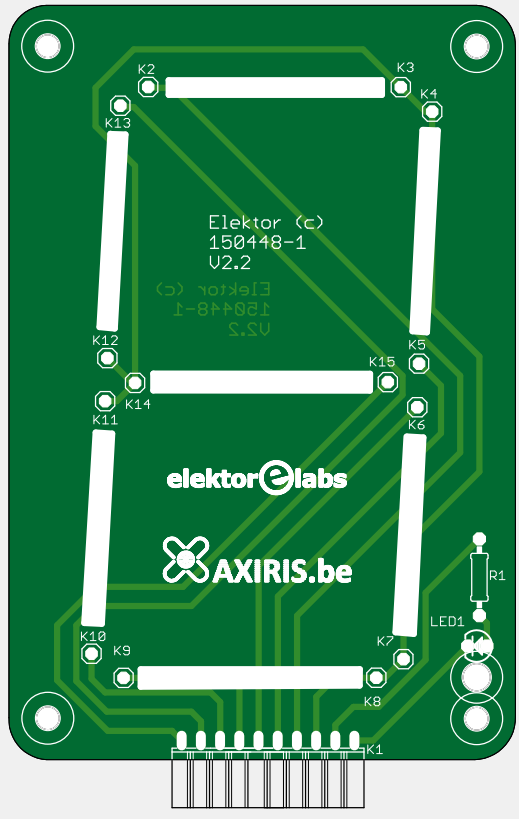


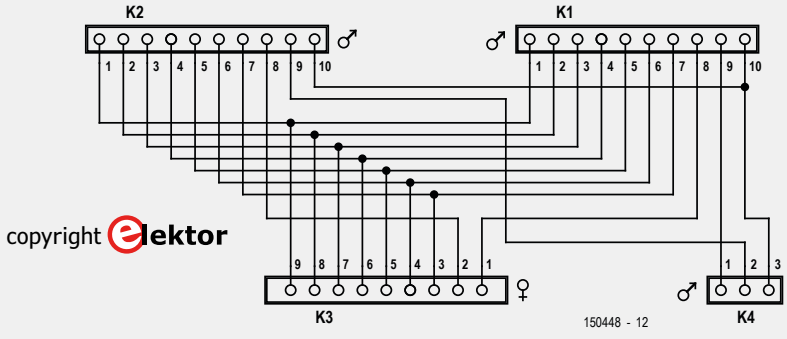
Bild 7. ... und bilden auf der Platine zusammen ein 7-Segment-Display.

Ausgangsspannung zwischen Masse und der Kathode von D1. Sie muss ungefähr 35 V betragen. Nun dreht man an P1 die Ausgangsspannung langsam auf ungefähr 75...80 V hoch. Damit ist die Schaltung erst einmal für den Einsatz vorbereitet. Wenn später die Filament-Displays angeschlossen sind, kann man die Helligkeit der LEDs noch nach Wunsch einstellen. Manchmal kann der Leckstrom durch T14 und T15 im gesperrten Zustand groß genug sein, dass die LED-Filamente zwar nicht kräftig, aber doch ein wenig leuchten. Das ist abhängig von der jeweiligen Applikation. Sollte der Effekt hinderlich sein, kann man Widerstände von 100 kΩ (R36, R37) zwischen den Anodenanschlüssen der Displays und Masse schalten.

Stückliste Zwischenplatine

Außerdem:
K1,K2 = 10-polige Stiftleiste, RM 2,54 mm
K3 = 9-polige Buchsenleiste, RM 2,54 mm
K4 = 3-polige Stiftleiste, RM 2,54 mm
Platine 150448-2

Bild 8. Mit einer „Zwischenplatine“ kann man zwei Displays gleichzeitig ansteuern.



Zum Schluss noch eine Warnung:
Die Ausgangsspannung ist nicht ganz ungefährlich. Ein Berühren bei Aufbau und Test sollte tunlichst vermieden werden. Später sollte die Schaltung in einem berührsicheren Gehäuse untergebracht werden. ⏮

(150448)

Weblinks

[1] Zusätzliche Informationen zu diesem Projekt: www.elektormagazine.de/150448

[2] Artikel über VFD-Shield: www.elektormagazine.de/150064

i-Pendel

Teil 1: Computermodell, Steuerungsregeln und Kalman-Filter

Von **Jean-Sébastien Gonsette** (Belgiën)

Das inverse Pendel, kurz i-Pendel, kann sich nicht nur selbst im Gleichgewicht halten, sondern sogar selbstständig aufrichten. Wenn man ein solches Projekt beginnt, kann man sich erstaunten Kopfschüttelns sicher sein, genauso aber der begeisterten Reaktionen, wenn man es erfolgreich abschließt und dem staunenden Publikum demonstriert.

Das Besondere an diesem inversen Pendel ist, dass sein Schwerpunkt *über* der Rotationsachse liegt. Es muss deshalb vermeiden, dass es umfällt; genau wie ein Mensch. Aber ein Mensch hat Augen und Ohren, deren Signale vom Gehirn interpretiert werden, um die Muskeln so zu steuern, dass der Schwerpunkt des Rumpfes immer über den Fußgelenken bleibt, wenn wir gehen oder stehen. Das inverse Pendel arbeitet nach dem gleichen Prinzip, wie man im Youtube-Video [1] sehen kann. Wenn man ein wenig im Internet surft, findet man eine zehnjährige Geschichte dieses Projekts, das für Studenten eine hervorragende Gelegenheit ist, die verschiedensten Regelungstechniken auszuprobieren und Erfahrungen mit einem interdisziplinären Entwurf zu sammeln. Und nicht unwesentlich: Das inverse Pendel hat einen außergewöhnlich hohen *Wow*-Faktor!

Selbst-balancierende Erfindungen gibt es in vielen Ausführungen und Maßen. Sie kennen sicherlich den einachsigen Roller Elektor-Wheelie, es gibt Stäbe, einfache, doppelte und dreifache, die so stabil auf einem Wägelchen stehen wie ein Besenstiel auf dem Finger, Quad-Copter, die eine Stange aufrecht balancieren, selbst mit einem Glas darauf. Zirkusreif, doch ich wollte etwas anderes. Hier das Paket meiner Anforderungen:

- Geringe Abmessungen, etwa 10 cm, und leicht genug, um auf meinem Schreibtisch zu stehen.
- Völlig autonom, also mit einer Energiequelle (Akku) an Bord.
- Einfach genug, um das Prinzip ohne Hokuspokus einfach erläutern zu können und ohne überflüssige Komplexität hinzuzufügen.

Ich fand im Internet ein schönes Projekt namens *Cubli*. Cubli wurde erdacht und realisiert von Wissenschaftlern der Technischen Universität Zürich, Spezialisten für die Steuerung dynamischer Systeme. Cubli ist ein Würfel, der sich selbst im Gleichgewicht hält, auf einer seiner Kanten oder sogar einer seiner Ecken, und der auch von einer in die andere Position springen kann [2].

Cubli erfüllt meine ersten beiden Bedingungen, aber nicht die dritte. Es ist nämlich recht kompliziert, ein Computermodell für einen Gegenstand mit drei Freiheitsgraden zu entwerfen, der sich also in allen drei Richtungen im Gleichgewicht halten muss. Die mathematischen Gleichungen für die Bewegungen sind alles andere als linear, was die Steuerung enorm verkompliziert. Ich habe den 3D-Würfel durch ein etwas dickliches 2D-Viereck ersetzt, das sich auf einer Ecke im Gleichgewicht halten kann. Und da das Viereck nicht „rollen“ muss, habe ich es noch diagonal durchgeschnitten und ein handliches Dreieck-Pendel daraus gemacht.

Anatomie des umgedrehten Pendels

Bild 1 zeigt eine Skizze mit den verschiedenen, nicht übermäßig zahlreichen Bestandteilen des inversen Pendels. Das Skelett besteht aus zwei Kunststoffteilen, die mit einem 3D-Drucker angefertigt wurden (darauf kommen wir im zweiten Teil des Artikels noch zurück). Die eine Hälfte nimmt die Elektronik auf, die andere ein Schwungrad mit einem Motor. Das „Hirn“ des Pendels ist ein PIC-Mikrocontroller, der den Motor ansteuert. Der in Modellbaukreisen beliebte LiPo-Akku steckt in einem Halter unter der Platine. Der „Bewegungsapparat“ des Pendels besteht aus einem Motor mit aufgesteck-

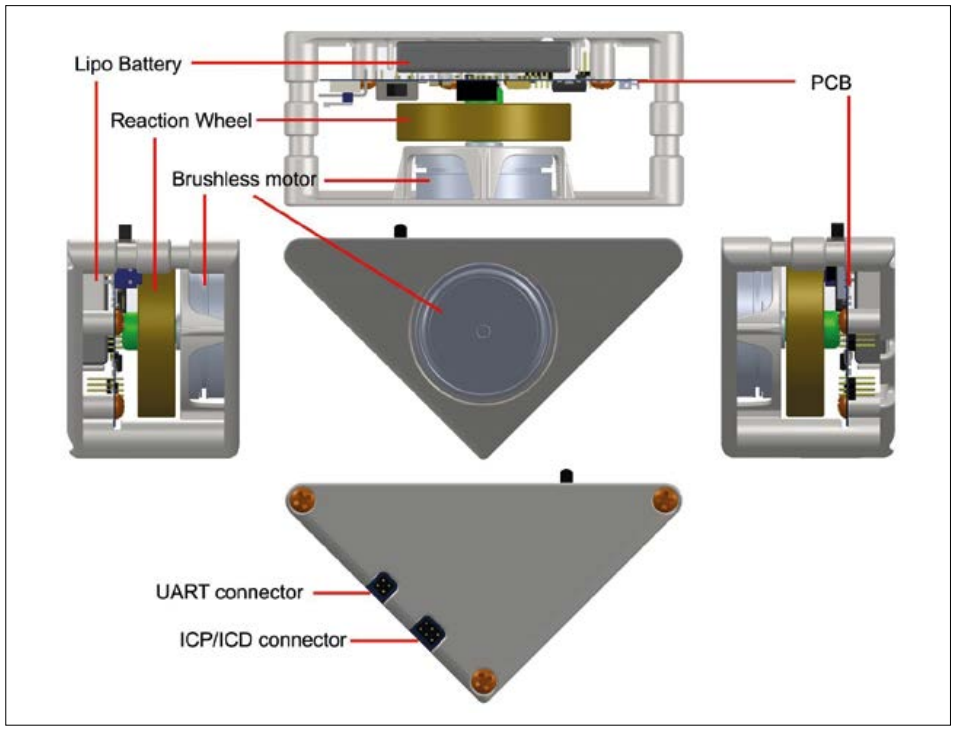


Bild 1. Innenansicht des umgekehrten Pendels.

tem Schwungrad. Auch das Schwungrad wurde mit dem 3D-Drucker erstellt und, um ihm eine ausreichende Masseträgheit zu verleihen, mit einem Metallreifen ausgestattet.

Wie funktioniert es?

Genau wie Cubli folgt dieses Pendel dem physikalischen Gesetz der *Erhaltung des Drehmoments* (oder des *Impulsmoments*). Es bleibt dank des Schwungrades im Gleichgewicht. Man könnte vermuten, dass das drehende Schwungrad einen gyroskopischen Effekt hat, doch dies ist hier nicht der Fall beziehungsweise nicht wesentlich. Erinnern wir uns an den Physikunterricht und an die Newton'schen Gesetze:

1. Ein Körper bleibt in seinem Ruhezustand oder bewegt sich geradlinig mit konstanter Geschwindigkeit, wenn auf ihn keine Kraft einwirkt (oder sich alle einwirkenden Kräfte gegenseitig aufheben).
2. Eine Änderung der Bewegung ist proportional der (Summe der) auf den Körper einwirkenden Kräfte und findet in Richtung der Kraft statt. In einer Formel heißt das:

$$\sum \vec{f}_i = \frac{d\vec{p}}{dt} \quad ,$$

was in der Praxis meist durch die Euler'sche Gleichung $f = m \cdot a$ ausgedrückt wird. Auf gut Deutsch heißt das: Will man den Bewegungszustand eines Körpers verändern, muss man eine proportionale, gerichtete Kraft ausüben.

3. Jeder Körper A, der eine Kraft auf einen anderen Körper B ausübt, verursacht eine Kraft gleicher Stärke in entgegengesetzter Richtung. Dies ist das Prinzip „actio est reactio".

Diese Gesetze gelten für physikalische Größen wie beispielsweise den berühmten Massepunkt. So etwas gibt es in der Realität aber nicht, in der Praxis arbeiten wir mit dem Schwerpunkt und mit der Rotation eines Körpers um seinen Schwerpunkt. Dabei gibt es zwei Arten von Kräften:

- Kräfte, die die lineare Bewegung des Schwerpunkts verändern;
- Kräfte, die die Drehung um den Schwerpunkt des Körpers verändern.

Beim zweiten Punkt ist es bequemer, mit dem *Drehmoment* zu arbeiten. Die Drehung eines Körpers um einen Drehpunkt hängt von der Länge des Hebelarms zum Drehpunkt und der Kraft ab, die senkrecht auf den Hebelarm wirkt (Drehmoment = Kraft mal Hebelarmlänge).

In **Tabelle 1** werden die Eigenschaften einer linearen und einer Drehbewegung miteinander verglichen.

Die Wirkungsweise des umgedrehten Pendels ist gut mit der der Schwungräder in Satelliten vergleichbar, die in Fachkreisen *Reaktionsräder* oder *reaction wheels* genannt werden.

Ein Satellit im Orbit muss meistens seine Antennen auf einen bestimmten Punkt auf der Erdoberfläche ausrichten. Er muss deshalb seine Lage in Bezug auf die Erdoberfläche korrigieren können. Dies geschieht mit Hilfe von Reaktionsrädern. Vor der Korrektur sei der Satellit stationär; er kreiselt nicht, auch nicht ganz langsam. Die Reaktionsräder drehen stationär im Leerlauf. Wenn man nun die Drehzahl eines Reaktionsrad erhöht, ergibt dies ein bestimmtes Drehmoment. Dadurch erfährt

Tabelle 1. Vergleich von linearer und Drehbewegung.

geradlinige Schwerpunktsverschiebung des Systems		Drehung um den Schwerpunkt des Systems	
Masse in kg	m	Trägheitsmoment in kg·m²	I
Geschwindigkeit in m/s	$\vec{v}$	Winkelgeschwindigkeit in rad/s	$\vec{w}$
Beschleunigung in m/s²	$\vec{a} = \frac{d\vec{v}}{dt}$	Winkelbeschleunigung in rad/s²	$\vec{\alpha} = \frac{d\vec{w}}{dt}$
Kraft in N	$\vec{f}$	Drehmoment in Nm	$\vec{\tau}$
Linearer Impuls	$\vec{p} = m \cdot \vec{v}$	Drehimpuls	$\vec{L} = I \cdot \vec{w}$
Kinetische Energie	$K = \frac{m \cdot v^2}{2}$	Kinetische Rotationsenergie	$K = \frac{I \cdot w^2}{2}$
Zweites Gesetz von Newton	$\sum \vec{f} = m \cdot \vec{a}$	Zweites Gesetz von Newton	$\sum \vec{f} = I \cdot \vec{a}$
Drittes Gesetz von Newton	$\vec{f}_{A \to B} = -\vec{f}_{B \to A}$	Drittes Gesetz von Newton	$\vec{\tau}_{A \to B} = -\vec{\tau}_{B \to A}$

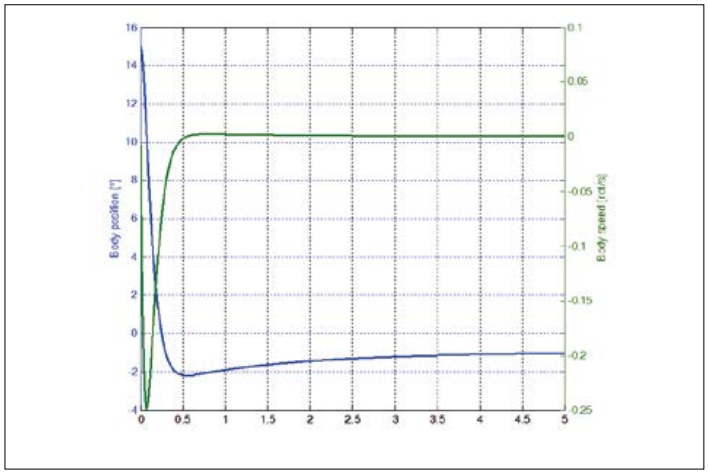


Bild 2. Simulation der Stabilisierung.

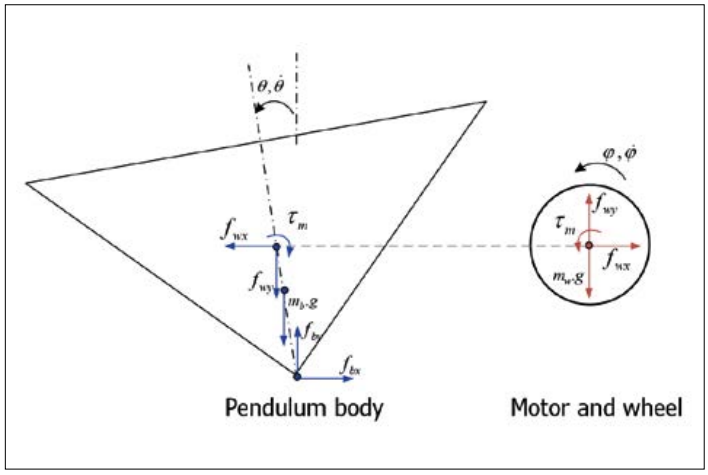


Bild 3. Dynamisches Modell des inversen Pendels und des Reaktionsrads.

der Satellit als Ganzes ein gleich großes Drehmoment in der umgekehrten Richtung (Newton 3) und kippt entsprechend. Der Clou der Sache ist, dass keine einzige Kraft von außen wirken muss. Der Satellit, gewichtslos im Weltraum dahinschwebend, kann seine Lage mit den Reaktionsrädern selbsttätig verändern. Schaut man sich den gesamten Satelliten inklusive Reaktionsräder an, ändert sich am Drehmoment des gesamten System nichts (Newton 1). Mit anderen Worten: Die Summe der Drehmomente von Reaktionsrad und dem Rest des Satelliten verändert sich nicht, sofern keine anderen Kräfte auf den Satelliten wirken.

Für unser i-Pendel ist die Situation ähnlich, nur ist die Kraft auf den Körper nicht Null, sondern gleich der Schwerkraft. Wenn wir versuchen, das Dreieck auf seiner Spitze im Gleichgewicht zu halten, wird ein minimales Ungleichgewicht durch die Schwerkraft (die man nicht ohne Grund auch Fall*beschleunigung* nennt) umgesetzt in eine Beschleunigung größer Null, so dass das Dreieck mit einer seiner kurzen Seiten platt auf den Schreibtisch kippt. Um dies zu verhindern, muss man dafür sorgen, dass der Schwerpunkt des Pendels genau über dem Drehpunkt gehalten wird. Doch das ist praktisch unmöglich, weil der Drehpunkt sehr klein ist. Aber man kann dafür sorgen, dass der Schwerpunkt *gemittelt* über dem Drehpunkt liegt. Möchte das Dreieck nach rechts fallen, so muss man es nach links ziehen und umgekehrt. Wenn man also entgegen der Schwerkraft wirkt, bleibt das Pendel im Gleichgewicht. Dazu ist, wie beim Satelliten, ein Reaktionsrad und ein Motor erforderlich. Wenn das Reaktionsrad die eine Seite beschleunigt, erfährt das Pendel eine Beschleunigung in die andere Richtung. Man muss nur ermitteln, zur welchen Seite das Dreieck gerade kippt und daraus das nötige Drehmoment des Motors ableiten, um das Ungleichgewicht zu kompensieren. Wenn das klappt, so wird das Pendel vielleicht mal um seinen Drehpunkt hin und her wackeln, aber nicht auf eine Seite fallen.

Und wie steht i-Pendel auf?

Liegt das Dreieck auf einer seiner kurzen Seiten auf dem Boden, kann es sich selbst auf den Winkel aufrichten. Wenn man das zum ersten Mal sieht, erscheint es reichlich wundersam, doch die Erklärung ist nicht kompliziert. Zunächst erzeugen wir mit dem Reaktionsrad ein Drehmoment, das vom Boden weg gerichtet ist. Dreht sich das Rad schnell genug, wird es abrupt gestoppt. Durch die Masseträgheit erhält das Pendel ein Drehmoment, das es auf seinen Drehpunkt aufrichtet. Noch mal zurück zu Newton:

a. Liegt das Pendel auf einer kurzen Seite und steht das Schwungrad still, so ist das Drehmoment Null.
b. Das Pendel schaltet sein Schwungrad an und gibt ihm ein Drehmoment, das vom Boden weg zeigt. Ohne festen Boden würde das natürlich nicht funktionieren.
c. Wenn die Umdrehungsgeschwindigkeit ausreichend ist, wird das Rad abrupt gestoppt. Sein Drehmoment wird dann Null, doch der Impuls wirkt auf den gesamten Körper des Pendels, so dass es sich aufrichtet.

Computermodell für das dynamische Verhalten des Pendels

Wollen wir unser Pendel steuern, müssen wir dahinter kommen, welchen Gesetzen seine Bewegungen unter dem Einfluss der Steuerbefehle folgen. Dies können wir am besten durch ein Computermodell analysieren. Aus dieser Analyse folgt die Dimensionierung der diversen Bauteile: Höhe des Pendels, Masse des Akkus, Drehmoment des Motors, Leistung der Elektronik und so weiter. Da ich bei null angefangen habe, konnte ich bestimmte Parameter mehr oder weniger willkürlich festlegen und danach erforschen, wie mein Modell sich verhielt. In **Bild 2** ist im Beispiel die Simulation der Reaktion auf meine Steuerbefehle zu sehen. Erst danach habe ich mich an den Bau eines Prototyps gemacht. Ein Herumspielen mit einem solchen Computermodell minimiert aber nicht nur das Risiko, einen unbrauchbaren Prototypen zu bauen, sondern es ist auch unverzichtbar bei der Erstellung von Algorithmen für die Motorsteuerung, der Analyse der Sensorsignale und der Bestimmung des Zustands, in dem sich das Pendel zu einem bestimmten Augenblick befindet.
Um die Steuerung des Pendels in Gleichungen und Algorithmen zu beschreiben, die sich in eine Firmware für einen Mikrocontroller unterbringen lassen, muss zunächst bekannt sein, wie viele Freiheitsgrade das System hat. Danach macht man eine Aufstellung aller Kräfte und Drehmomente, die auf jedes Teil des Pendels einwirken und beschreibt dann, ob und wie sich die Kräfte auf die Bewegung des Pendels auswirken. Das Pendel hat zwei Freiheitsgrade:

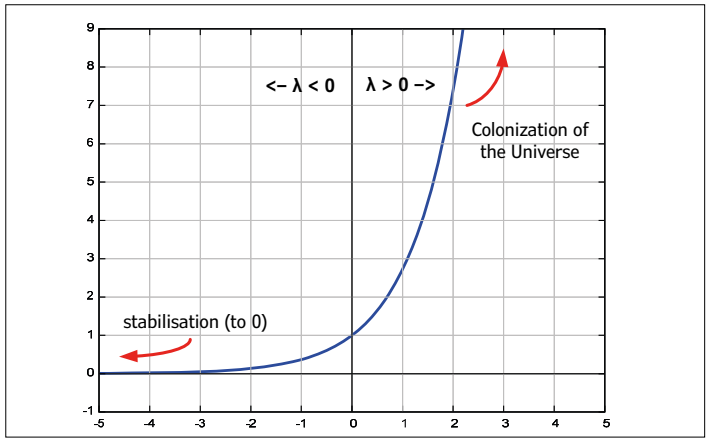


Bild 4. Der Systemwert bestimmt alles (oder nichts).

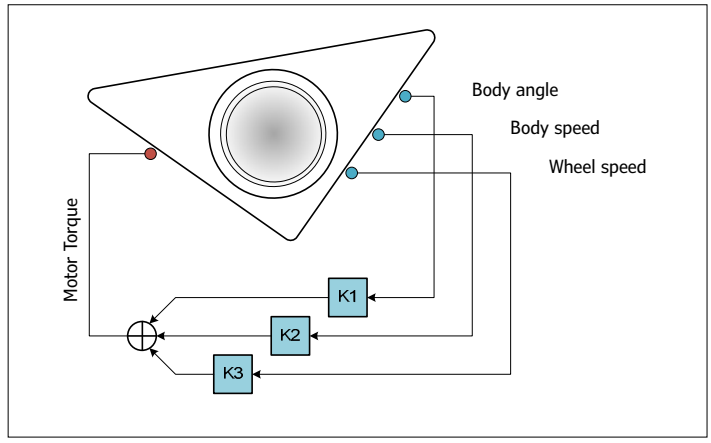


Bild 5. Rückkopplungs- oder Feedback-Regler.

- das ganze Dreieck, das auf seinem Drehpunkt mit dem Winkel θ auf dem Boden balanciert.
- die Drehung des Motors und des Schwungrads mit einem Winkel φ.

Nun wollen wir wissen, was die zweiten Ableitungen dieser beiden Parameter sind, mit anderen Worten, was ihre Beschleunigung als Funktion des Drehmoments des Motors ist. Alle Kräfte und Größen, die Einfluss haben auf die Bewegung des Pendels als Ganzes und auf das Schwungrad, sind in **Bild 3** dargestellt.

- Schwungrad: Das Drehmoment des Motors τ_m, das Reibungsmoment zwischen Motor und Stator τ_f und die zwei Kräfte, die den Rotor an seinem Platz halten, nämlich f_{wx} und f_{wy}.
- Pendel: dieselben Kräfte, die auf das Rad wirken, jedoch in umgekehrter Richtung, plus zwei Kräfte zwischen Pendel und Boden, die das Pendel auf seinem Platz halten: f_{bx} en f_{by}.

Die Schwerkraft wirkt sowohl auf die Masse des Pendels m_b als auch auf die des Rades m_w. Die komplette Herleitung dieser Formeln kann ich Ihnen ersparen, aber Sie können mir glauben, dass sie in den folgenden Bewegungsgleichungen mündet:

$$\ddot{\theta} = \frac{M \cdot g \cdot \sin\theta - \tau_m + \tau_f}{I}$$

$$\ddot{\varphi} = \frac{(I + I_w) \cdot (\tau_m - \tau_f)}{I \cdot I_w} - \frac{M \cdot g \cdot \sin\theta}{I}$$

mit $M = m_b \cdot l_b + m_w \cdot l_w$ und
$I = I_b + m_b \cdot l_b^2 + m_w \cdot l_w^2$

Die Terme m_b, I_b, m_w und I_w bezeichnen die Masse beziehungsweise die Trägheit des Pendels als Ganzes und des Reaktionsrades. l_b und l_w sind die Höhe des Schwerpunks des gesamten Pendels und des Schwungrades über dem Boden. Mit diesen Gleichungen kann man simulieren, wie das Pendel sich verhält und darüber hinaus feststellen, wie die Steuerungsgleichungen aussehen müssen.

Stabilität des Pendels

Die Stabilität des Pendels soll anhand einer biologischen Analogie erläutert werden. Stellen Sie sich eine Petrischale mit sich vermehrenden Mikroorganismen und deren Verhalten vor. Wir gehen von folgenden Grundregeln aus:

- y ist die Anzahl der Anzahl der Organismen oder die Population der Schale am Anfang der Untersuchung,
- n ist die Zahl der geborenen, d die Anzahl der gestorbenen Mikroorganismen in der Population,
- wenn n und d proportional zum Umfang der Population ist, vollzieht sich die Entwicklung der Populationsgröße im Rahmen der Formel $y' = n \cdot y - d \cdot y$.

Sind n und d konstant, dann ist die Zu- oder Abnahme der Bevölkerung gleich dem Integral dieser letzten Gleichung. Mit einer Anfangspopulation y_0 erhält man:
$y(t) = y_0 \cdot e^{(n-d)\cdot t} = y_0 \cdot e^{\lambda \cdot t}$
Es ist eine exponentielle Funktion! Die Größe des Terms $n - d$ hat entscheidenden Einfluss auf die Entwicklung der Populationsgröße. Ist die Zahl der Todesfälle größer als die der Geburten, so ist der Exponent negativ und die Bevölkerung der Petrischale stirbt asymptotisch aus. Im umgekehrten Fall wächst die Bevölkerung exponentiell, bis sie das gesamte Weltall kolonisiert hat (theoretisch). Der Term $n - d$ oder λ in der Formel wird *Eigenwert* des Systems genannt und bestimmt, ob ein System stabil ($\lambda < 0$) oder instabil ($\lambda > 0$) ist (**Bild 4**).
Variiert man den pH-Wert in der Petrischale, kann man Leben und Sterben der Organismen beeinflussen. Angenommen, pH-Wert- und Population-Veränderung wären proportional, ist:

$$y' = n \cdot y - d \cdot y + k \cdot y \text{ en } y(t) = y_0 \cdot e^{(n-d+k)\cdot t}.$$

Hier finden wir einen wichtigen Hebel: k ist frei zu wählen (innerhalb der physikalischen Grenzen des Systems), so dass wir einen negativen Eigenwert positiv und einen positiven negativ machen können. Durch die Steuerung von k kann man das gesamte System stabil halten!
Die Steuergleichungen für das Pendel sind dann doch ein wenig komplizierter als die im Beispiel. Die verschiedenen Zustände wie Winkel oder Geschwindigkeit von Pendel oder Motor müssen als Vektoren eingefügt und außerdem mehrere Differentialgleichungen in Matritzen transformiert werden. Und wenn man die Matritzen durchrechnet, ist es nicht so einfach zu sehen, ob ein System stabil ist oder nicht und warum. Es muss nur irgendwo ein Eigenwert positiv werden, im System also ein $e^{\lambda \cdot t}$ mit $\lambda > 0$ auftauchen und die ganze Angelegenheit funktioniert nicht mehr.

Regelung

Ein System mit einer Eingangs- und einer Ausgangsvariablen lässt sich meist gut mit einem PID-Regler steuern. Unser Pendel besitzt allerdings zwei Freiheitsgrade, den Winkel des Pendels als Ganzes und das Drehmoment des Schwungrades. Diese beiden Größen müssen gleichzeitig gesteuert werden, und zwar mit nur einem Steuersignal für das *Drehmoment*

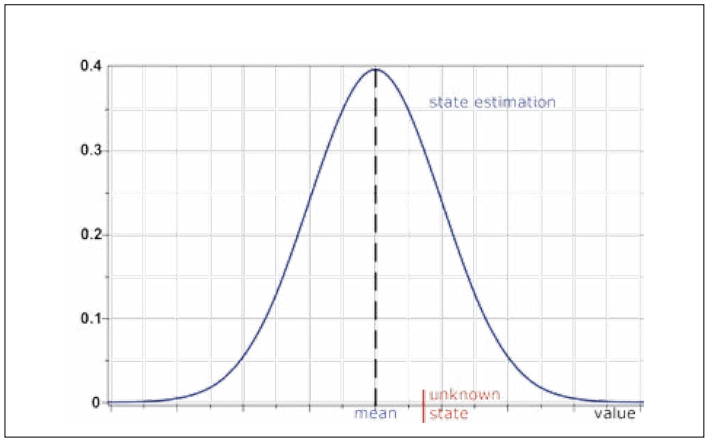


Bild 6. Normal- oder Gauß-Verteilung.

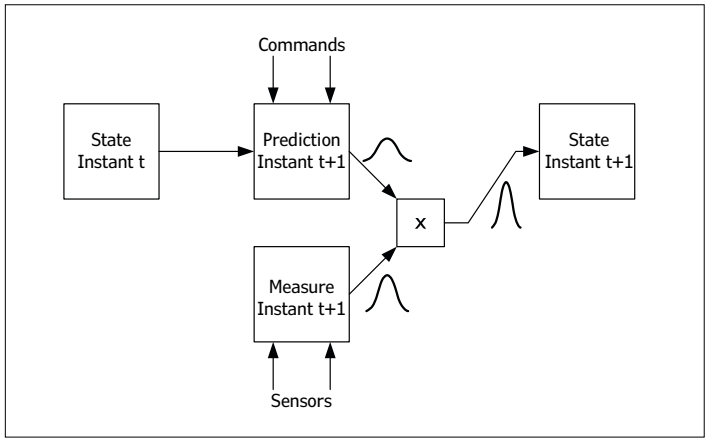


Bild 7. Prinzip des Kalman-Filters.

des Motors. Das Problem ist nun, dass ein Motor ein bestimmtes Drehmoment nur in einem bestimmten Drehzahlbereich liefern kann. Also muss ich auch die *Drehzahl* regeln, sonst wird alles unkontrollierbar und das Pendel ist nicht mehr zu steuern. Aus und vorbei, PID-Regler!
Stattdessen habe ich ein Rückkopplungssystem gewählt, das Messwerte aus dem System einfließen lässt. Jede Ausgangsvariable wird mit einer bestimmten Verstärkung oder Abschwächung (*gain*) zum Eingang zurückgeführt. Es gibt drei Ausgangsvariablen: der Winkel und die Drehgeschwindigkeit des Pendels sowie die Drehzahl des Schwungrads. In der Regelschleife werden diese drei Signale, jedes mit individueller Verstärkung, addiert und zum Eingang zurückgeführt (**Bild 5**). Die Verstärkungsfaktoren können so eingestellt werden, dass die Eigenwerte stets negativ sind und das Pendel stabil bleibt. Dies lässt sich auf verschiedene Arten bewerkstelligen, doch die beste Wahl ist ein linearer quadratischer Regler (LQR, *Linear Quadratic Regulator*), der mit einem Algorithmus arbeitet, der die Verstärkungsfaktoren automatisch aus Systemgleichungen berechnet, wobei bestimmte Größen gewichtet behandelt werden. Je „gewichtiger" eine Größe ist, je mehr der Algorithmus den Zustand des Befehls für diese Größe beeinflusst, desto schneller wird der Wert der Größe dem gewünschten Punkt angenähert.

Am wichtigsten ist es, dass das Gleichgewicht mit einem minimalen Drehmoment hergestellt werden sollte, nicht aber die Zeit, in der dies passiert. Das Motor-Drehmoment ist endlich, deshalb müssen wir das Maximum, das dem Motor möglich ist, ansetzen, um das größte anzunehmende Ungleichgewicht zu kompensieren.
Ich will es bei diesen Anmerkungen zur Theorie belassen. Es ist eine ziemlich komplexe Materie, die über die Möglichkeiten eines Elektor-Artikels herausreicht. Wenn Sie sich weiter darin vertiefen wollen, schauen Sie einmal auf die Tutorials der Universität Michigan [3] über Steuersysteme in Matlab. Dort gibt es auch ein System für ein umgedrehtes Pendel.

Gleichgewichtszustand nach Kalman-Filterung

Wir haben drei Ausgangsvariablen: den Winkel des Pendels, die Geschwindigkeit des Pendels und die Geschwindigkeit des Motors. Wie können wir wissen, wie diese Variablen zu einem bestimmten Zeitpunkt sind? Wir können sie berechnen oder messen. Wenn ich den Startwert weiß, kann ich den Verlauf als Funktion des Drehmoments, das an den Motor gelegt wurde, berechnen. Ich integriere die Differentialgleichungen, um herauszufinden, in welchem Zustand sich das Pendel im nächsten Augenblick befindet. Und diese Berechnung könnte ich immer wiederholen. Doch das ist so, als würde man ein Auto mit verbundenen Augen steuern: Wenn man sich den Weg vorstellen kann und das Auto perfekt beherrscht, würde das wohl klappen – ein paar Sekunden. Vorausberechnen ist keine Option! Wir benötigen eine vertrauenswürdigere Methode. Doch auch Sensoren weisen eine Messungenauigkeit auf und sind mit Rauschen behaftet. Zwei aufeinanderfolgende Messungen einer absolut konstanten Größe ergeben selten dasselbe Ergebnis. Darüber hinaus kann man nicht alle Größen direkt, sondern nur indirekt über Umwege bestimmen.
So habe ich auf der einen Seite eine Methode, die zu Beginn sehr genau ist (die Berechnung), aber mit der Zeit immer ungenauer wird, und auf der anderen Seite eine Methode (die Sensoren) mit einem festen, bekannten Fehler, der aber nicht vernachlässigt werden kann. Das Beste ist es, eine Methode zu verwenden, die zu jedem Zeitpunkt den kleinsten Fehler verspricht. Wenn ich zum Beispiel einen Widerstand messe, zeigt ein teures Messgerät 68 Ω an, ein billiges 70 Ω. Die erste Messung scheint vertrauenswürdiger, weil das teure Messgerät doch auch viel besser ist. Oder nicht? Wir nehmen besser eine gewertete Mittelung vor, wobei das Messresultat des teuren Geräts ein höheres Gewicht erhält als das des billigen. Dieses Prinzip verwende ich auch für das Pendel. Der Zustand des Pendels wird als Funktion des Modells und der Sensormessungen bestimmt, wobei den Signalen eine Gewichtung mitgegeben wird.

Dieses Prinzip stellt die Basis für ein so genanntes **Kalman-Filter** dar. Dabei werden gemessene und berechnete Werte gewichtet zu einem Mittelwert zusammengefasst, der wieder als Steuersignal verwendet wird. Dadurch wird das Messrauschen unterdrückt und die Vertrauenswürdigkeit der berechneten Werte optimiert. Ein Kalman-Filter arbeitet in zwei Schritten:

1. Bestimmung des Zustands des Systems im Moment $t + dt$ anhand des Systemmodells und der Befehle im Moment t. Diese Berechnung stellt eine Abschätzung der Unsicherheit über den Zustand des Systems dar, die aus dem Modell bestimmt wird. Alle Unsicherheiten (der Begriff Wahrscheinlichkeiten klingt vertrauter) eines Kalman-Fil-

ters haben eine Normalverteilung. Für ein System mit einem Zustand gilt die Gauß'sche Glockenkurve (**Bild 6**), gibt es mehrere Zustände, erhält man eine mehrdimensionale Gauß-Grafik.

2. Messung des Systemzustands zum Zeitpunkt $t + dt$. Diese Messung hat auch seine Wahrscheinlichkeiten, auch wieder normalverteilt. Alle Messungen erhalten eine Gewichtung entsprechend ihrer Unsicherheiten, wie in dem Beispiel mit der Widerstandsmessung.

Danach wird das Resultat der Messungen mit der Berechnung erweitert, wobei Unsicherheiten auf beiden Seiten einkalkuliert werden. In **Bild 7** ist dieser Prozess in einem Blockdiagramm veranschaulicht. Wenn Sie mehr darüber wissen wollen, rate ich Ihnen, das Buch [4] zu studieren. In unserem Fall messen ein Gyroskop und ein Beschleunigungsaufnehmer den Winkel des Pendels mit dem Boden. Das Gyroskop misst ausschließlich längs der Z-Achse im rechten Winkel zur Fläche der Platine. Der Beschleunigungssensor misst auf drei Achsen (X, Y und Z), wobei aber nur X und Y vom Kalman-Filter gebraucht werden. Das Filter ist sehr einfach ausgeführt und gibt dennoch ausreichend genaue Resultate.

Wie werden die Sensorsignale zusammengeführt?

- Das Gyroskop ergibt die Winkelgeschwindigkeit des Pendels. Integriert man diese nach der Zeit mit einer

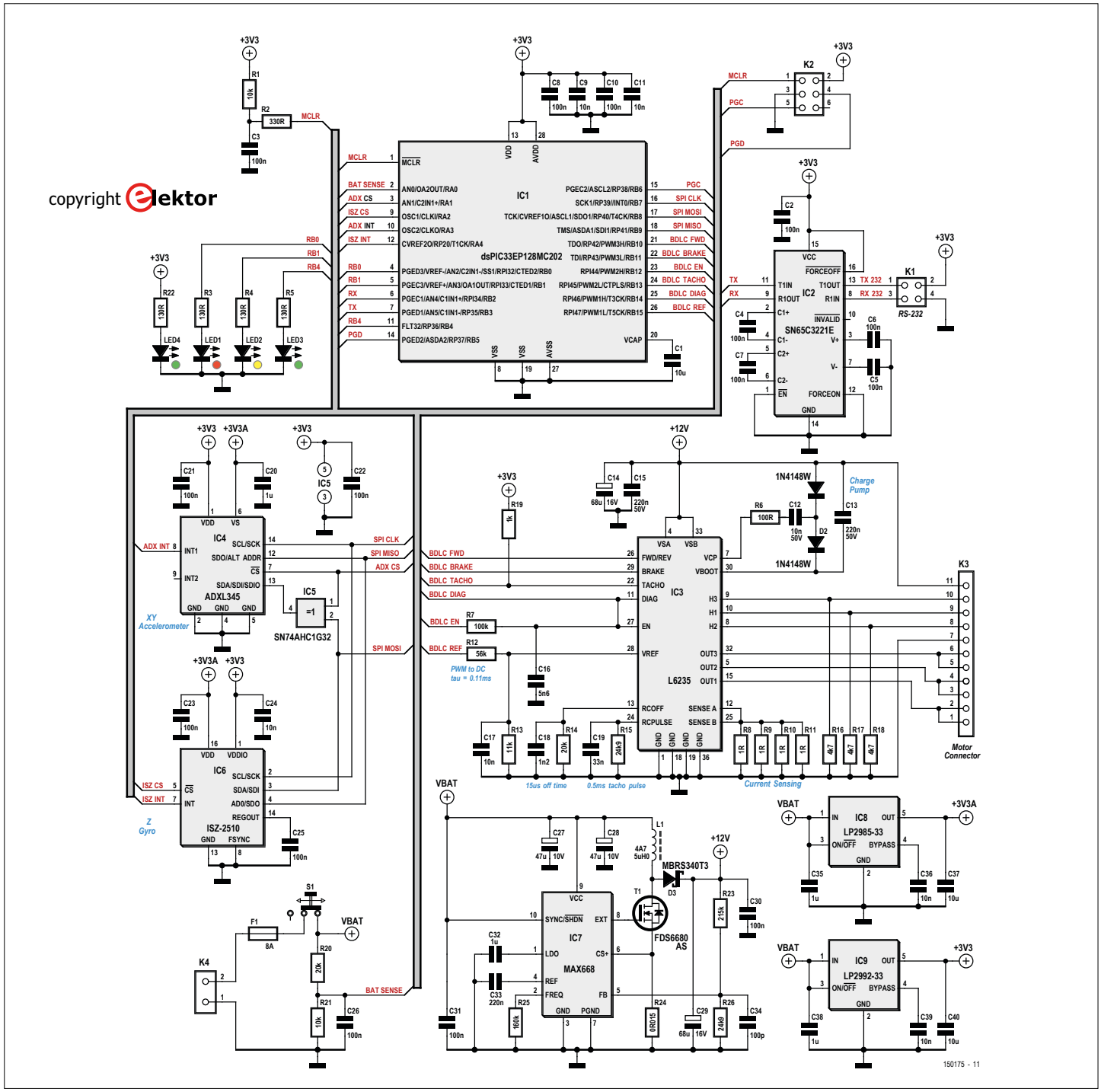


Bild 8. Vollständige Schaltung des inversen Pendels.

dt-Stufe, können wir den zukünftigen Winkel des Pendels berechnen.
- Der Beschleunigungsaufnehmer misst hauptsächlich, wie die Schwerkraft oder besser die Fallbeschleunigung auf der X- und der Y-Achse wirkt. Mit ein wenig Goniometrie (Winkelfunktionen) erhalten wir direkt den Winkel des Pendels mit dem Boden.

Berechnung und Messung werden miteinander verrechnet, bewertet nach dem Fehler der Sensoren. Die Abschätzung der Winkelgeschwindigkeit des Schwungrads beruht auf dem gleichen Prinzip. Mit dem Drehmoment, das der Motor vorgibt, ist die Beschleunigung oder Verzögerung des Schwungrads zu berechnen und damit die Geschwindigkeit nach dem Moment *dt*. Die Berechnung wird mit der Drehzahlmessung eines Tachometers am Motor kombiniert.

Die Elektronik

Nach all diesen theoretischen Betrachtungen ist es höchste Zeit für die Praxis. Die Elektronik des Pendels in **Bild 8** hat vier wichtige Aufgaben:
- die Spannung des LiPo-Akks umsetzen zur Versorgung eines bürstenlosen Motors;
- die Steuersignale für den Motor liefern;
- die Bewegungsparameter des Pendels mit einem Beschleunigungsaufnehmer und einem Gyroskop messen;
- den Motor mit den oben beschriebenen Steuerregeln auf Basis der Sensor-Messwerte steuern.

Netzteil

Di Spannung des LiPo-Akkus variiert zwischen 7 V und etwas mehr als 8 V, je nach Entladezustand. Die Nominalspannung des Maxon-Motors ist mit 12 V angegeben. Deshalb wird der schaltende Aufwärtsregler MAX668 eingesetzt. Der Motor verbraucht zwar 2 A bei einem Drehmoment von 55 mNm, das Netzteil kann aber mindestens doppelt so viel liefern.
Solange Pin 8 des MAX668 auf High liegt, leitet der Leistungs-MOSFET T1 und der Strom durch die Spule L1 nimmt zu. Wird Pin 8 Low, sperrt T1 und die magnetische Energie entlädt sich über die Diode und C29. R25 am MAX668 legt die Frequenz, in der dieser Zyklus abläuft, auf hier 300 kHz fest. Die Ausgangsspannung, gemessen zwischen R23 und R26, wird zur Regelung herangezogen, genau wie der Strom durch die Spule, gemessen von R24. Die 12-V-Ausgangsspannung weist eindeutig eine hochfrequente Welligkeit auf, die vom Laden/Entladen von C29 herrührt. Die Welligkeit wird vom Serienwiderstand (ESR) des Kondensators hervorgerufen. Aus diesem Grund sollte man diesen Kondensator sehr sorgfältig auswählen (Low ESR).

Motor

Ich habe einen bürstenlosen Motor aus der EC-Flat-Reihe von Maxon ausgesucht. Bei einem gewöhnlichen Gleichstrommotor hätte man mit dem Verschleiß der Kohlebürsten und auch mit einer Welligkeit des Kommutators zu kämpfen. Mit einem bürstenlosen Motor vermeidet man dies und erhält darüber hinaus einen höheren Wirkungsgrad. Der Nachteil ist, dass die Verkabelung und die Ansteuerung etwas komplizierter sind. Man kann ihn nicht einfach an einer Spannungsquelle betreiben, sondern muss für eine dreiphasige Ansteuerung sorgen, weil die Spulen den Anker nacheinander anziehen müssen. Glücklicherweise gibt es zu diesem Zweck integrierte Lösungen wie IC3, ein L6235, der sowohl die Steuerlogik wie auch die Leistungsstufen für die Motorsteuerung enthält. Das IC besitzt folgende Ein- und Ausgänge:
- *Fwd/Rev* gewünschte Drehrichtung;
- *Brake* aktiviert die elektromagnetische Bremse (die gebraucht wird, um das Pendel aufzurichten);
- *En* (enable) gibt die Drehung des Motos frei;
- *Vref* analoger Eingang für die Strombegrenzung des Motors;
- *Diag* gibt an, dass eine interne Absicherung in Funktion ist;
- *Tacho* gibt Tachometer-Impulse aus.

Mit dem Tiefpassfilter R12, R13 und C17 wird das PWM-Signal in eine analoge Spannung umgesetzt, die den Grenzwert des Drehmoments bestimmt. Die drei Phasen des Motors werden über die Ausgänge Out1...Out3 gesteuert, während die Position des Motors, ermittelt von Hall-Sensoren, über die Eingänge H1...H3 dekodiert wird. Die Schaltung stellt den Motorstrom ein, indem sie die Spannung über den parallelen Shunt-Widerständen R8...R11 mit dem Wert von Vref vergleicht. Das Motormoment ist proportional zum Motorstrom und deshalb gut zu regeln.

Sensoren

Das Pendel kann nur im Gleichgewicht bleiben, wenn es seine Position misst und sie aufgrund der Messung korrigiert. Der Winkel des Pendels zum Boden wird durch ein Gyroskop mit einer Achse (IC6, ein ISZ-2510) und einem Beschleunigungsaufnehmer mit drei Achsen (IC4, ein ADXL345) ermittelt. Diese analogen Sensoren besitzen ein SPI-Interface zur Kommunikation mit dem Mikrocontroller. Sie befinden sich in einem Teil der Schaltung, der mit 3,3 V versorgt wird. Dadurch hat die analoge Messung wenige Probleme mit dem Rauschen, das durch den Rest der Schaltung verursacht wird.

Mikrocontroller

Alle Intelligenz des Pendels ist in einem 16-bit-dsPIC-Mikroconttroller untergebracht. Für dieses Projekt ist dieser eine bessere Wahl als ein 32-bit-Controller, weil Reaktionsgeschwindigkeit wichtiger ist als Rechenkraft. Andererseits besitzt der Controller ausreichend RAM und ROM, um komplexere Applikationen zu ermöglichen, im Gegensatz zu 8-bit-Controllern. Die dsPIC-Reihe besitzt PWM-Ausgänge, die hier gebraucht werden, um den Motor anzusteuern. Der Controller wird auf übliche Weise über den ICP-Port an K2 programmiert. Einer der UARTs wird für einen seriellen Port über IC2, einem SN65C3221, verwendet.

Der nächste Teil des Artikels ist „konkreteren“ Dingen wie der Verdrahtung der Schaltung, der mechanischen Konstruktion und der Kalibrierung der Sensoren gewidmet. Sehe ich Sie in einem Monat wieder? ⏮

(150175)

Weblinks

[1] www.youtube.com/watch?v=6xe19XnX5L0

[2] www.youtube.com/watch?v=bMuCACqwI4s

[3] http://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=SystemModeling

[4] Buch von Dan Simon, *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*

ESP8266 auf dem Android I/O-Board

Flashen neuer Firmware

Von **Elbert Jan van Veldhuizen** (NL)

Mit dem Android I/O-Board vom September 2015 lassen sich über Android-Smartphones oder -Tablets drahtlos komplexe Funktionen steuern. Für die Funkverbindung stehen alternativ sieben Module zur Wahl, unter ihnen das verbreitete, kostengünstige WLAN-Modul ESP8266. Allerdings ist der Betrieb nur mit einer daran angepassten Firmware möglich. Hier wird gezeigt, welche Schritte nötig sind.

ESP8266 ist die Typenbezeichnung eines preiswerten WLAN-Kommunikationsmoduls, das unkompliziert zu handhaben ist. Wir hatten bereits angekündigt, dass der Betrieb zusammen mit dem Android I/O-Board das Thema eines eigenen Beitrags sein wird, sobald die nötige Firmware stabil arbeitet.

Inzwischen ist die Entwicklung so weit vorangeschritten, dass wir unser Versprechen einlösen können.

ESP8266

Das WLAN-Funkmodul ESP8266 ist als Access Point (AP) oder als Client (STA, Station) einsatzfähig. Im AP-Modus loggt das Android-Gerät ohne Umweg in das WLAN-Netzwerk des ESP8266 ein, im STA-Modus stellt es die Verbindung über einen Router her. Wenn das Android-Gerät ebenfalls mit dem Router verbunden ist, kann die Verbindung über das lokale Netzwerk laufen. Ferner kann das Android-Gerät die Rolle des AP übernehmen (Tethering). In diesem Fall ist die Verbindung des ESP8266 im STA-Modus möglich. Im STA-Modus müssen dem ESP8266 die SSID und das Passwort des Netzwerks bekannt sein. An anderer Stelle in diesem Beitrag wird beschrieben, wie diese Informationen in das ESP8266 gelangen.

Verfügbar ist das ESP8266 in mehreren Versionen, die Typenbezeichnungen unterscheiden sich durch angehängte Nummern. Das Android I/O-Board ist für die Version ESP-01 ausgelegt, bei dem, wie **Bild 1** zeigt, zwei mal vier Kontakte im Rastermaß 2,54 mm vorhanden sind. Diese Variante passt auf Steckverbinder MOD4 des Android I/O-Boards. Auf dem ESP8266-01 befinden sich zwei LEDs, sie signalisieren, dass Betriebsspannung anliegt oder Daten übertragen werden. Verbreitet ist auch die Version ESP8266-12, bei der mehr Anschlussleitungen nach außen geführt sind. Diese Version passt nicht unmittelbar auf das Android I/O-Board. Nachdem das Modul mit doppelseitigem Klebeband fixiert ist, müssen die Verbindungen über Drähte hergestellt werden. **Bild 2** zeigt einen „Adapter“, der aus Drähten und einem Stück Rasterplatine angefertigt wurde. Bei einigen Untertypen des ESP8266-12 muss der Anschluss GPIO15 über einen Widerstand bis 1 kΩ an Masse gelegt werden. Über diesen Anschluss kann das Modul bei Bedarf prüfen, ob eine bootfähige SD-Speicherkarte vorhanden ist. Auf dem ESP8266-12 befindet sich nur eine LED, sie gab beim Modul des Autors kein Lebenszeichen von sich, während das Modul ordnungsgemäß arbeitete.

Die Speichergrößen der ESP8266-Module können ebenfalls unterschiedlich sein. Die älteren, meistens blauen Ausführungen sind mit 512 KB (4 Mbit) ausgestattet, die neueren, meistens schwarzen Varianten verfügen über 1 MB (8 Mbit). Für den Einsatz zusammen mit dem Android I/O-Board sind beide Ausführungen geeignet.

Alle Varianten des ESP8266 generieren relativ hohe, der Betriebsspannung überlagerte Störspannungen. Die Puffer-Elkos auf dem Android I/O-Board reichen nicht aus, um das Modul stabil arbeiten zu lassen. Um dem abzuhelfen, kann wie in **Bild 3** gezeigt am Steckverbinder MOD6 ein Elko 100 µ zwischen die Anschüsse 2 (+) und 3 (–) geschaltet werden.

Firmware ESP8266

Das WLAN-Modul ESP8266 wird mit einer Firmware ausgeliefert, die mit AT-Kommandos arbeitet. Wenn beispielsweise Daten über den seriellen Port geschickt werden sollen, muss zuvor die Anzahl der Zeichen mit einem AT-Kommando deklariert werden.
Ferner ist beim Original die Baudrate auf 115200 Baud eingestellt, sie ist mit den 9600 Baud des Android I/O-Boards nicht kompatibel.

Die beschriebenen Handicaps werden dadurch ausgeglichen, dass für das ESP8266 diverse Firmware-Versionen aus fremden Quellen verfügbar sind. Der Autor hat sich für die Firmware des Projekts „beckdac“ entschieden [1], die stabil und ohne Schnörkel läuft. In eigener

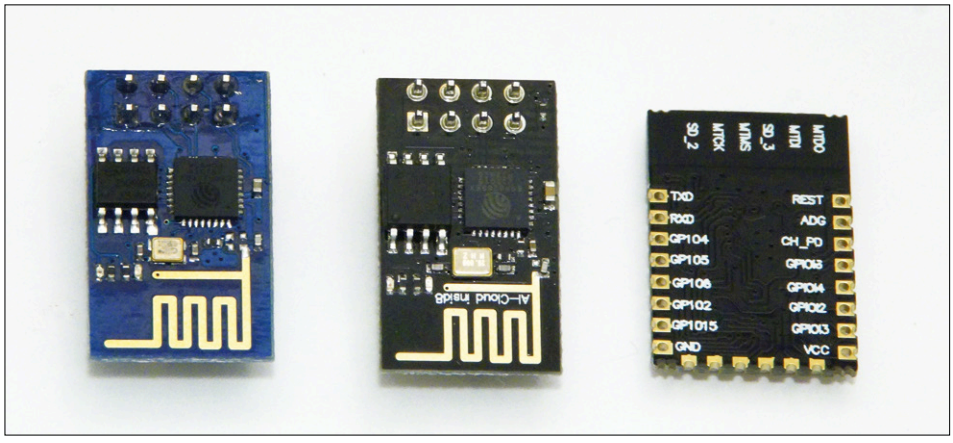


Bild 1. Drei Mal ESP8266: ESP8266-01 mit 512 KB und ESP8266-01 mit 1 MB von oben, ESP-12E mit 1 MB von unten.

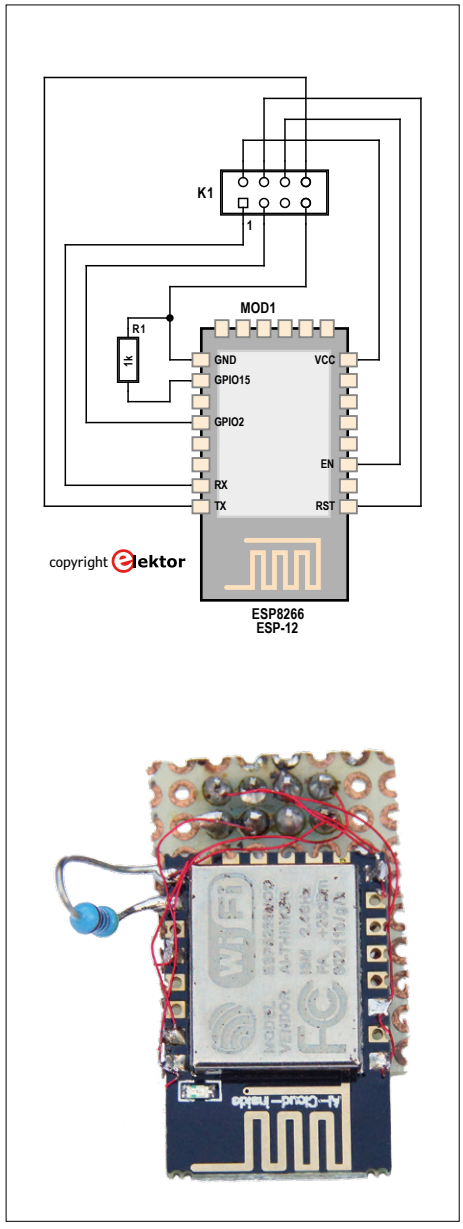


Bild 2. Anpassen eines ESP8266-12 an das Layout eines ESP8266-01.

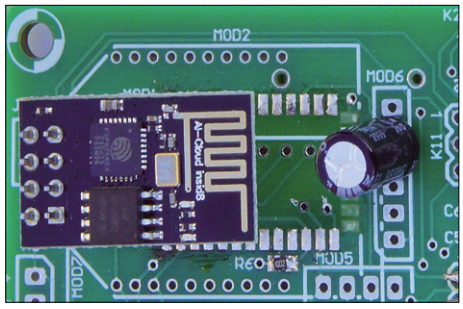


Bild 3. Puffer-Elko an den Anschlüssen des Steckverbinders MOD6.

Regie muss das ESP8266 nicht mit dieser Firmware geflasht werden, die bereits geflashte Version ist im Elektor-Shop erhältlich [2].

Kommandos (AP und STA)

Das ESP8266 lässt sich über die WLAN-Verbindung mit Kommandos konfigurieren, die mit „+++AT“ beginnen. Nachstehend sind nur die wichtigsten Kommandos aufgelistet, eine vollständige Übersicht ist unter [3] zu finden.

`+++AT BAUD baudrate` – Einstellen der Baudrate des seriellen Ports; für das Android I/O-Board gilt: +++AT BAUD 9600. Das Kommando ohne Parameter gibt die aktuelle Einstellung zurück (dies gilt für alle Kommandos!). Auf +++AT BAUD antwortet das Modul mit BAUD=9600 8 N 1.

`+++AT MODE mode-nummer`. Bei Mode-Nummer 1 befindet sich das ESP8266 im STA-Modus, bei Mode-Nummer 2 im AP Modus. Mode-Nummer 3 bedeutet, dass sowohl der STA-Modus als auch der AP-Modus aktiv sind.

`+++AT STA SSID password` stellt die SSID und das Passwort des Routers ein, mit dem sich das Modul verbinden soll.

`+++AT AP SSID password authentication_method` gibt die Sicherheitsparameter des WLAN-Netzwerks vor, mit denen das Modul arbeiten soll. Die Zahl 2 steht für WPA, und 3 bedeutet WPA2. Mit +++AT SSID password 3 ist das ESP8266 gegen unerwünschtes Eindringen sicher geschützt.

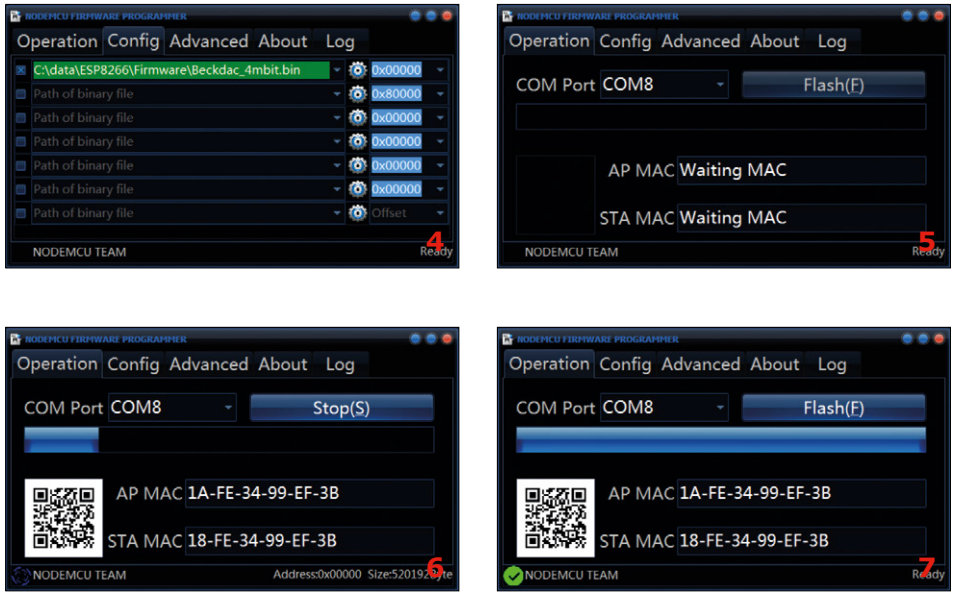


Bild 4...7. Der „esp8266flasher“ bei der Arbeit.

Mit dem ESP8266 können sich mehrere Geräte gleichzeitig sowohl im STA-Modus als auch im AP-Modus verbinden. In diesem Fall bündelt das Modul die Verbindungen und gibt die Antworten parallel an alle verbundenen Geräte aus.

Flashen des ESP8266

Die zu flashende Firmware besteht aus mehreren Teilen, die an unterschiedlichen Speicheradressen stehen müssen. Die Prozedur ist nicht nur umständlich, auch kommen einige Programmer damit nicht zurecht. Der Autor vermutet, dass in den dazwischen liegenden Bereichen Daten stehen bleiben, mit der Folge, dass einige Funktionen des ESP8266 gestört sind. Deshalb hat der Autor eine einzige, zusammenhängende Binärdatei erstellt, mit der auch die Zwischenbereiche überschrieben werden. **Tabelle 1** enthält eine Übersicht über die Bestandteile der Datei, sie setzt sich aus Code des Beckdac-Projekts und originärem Code zusammen [4]. Die Dateien können von der Elektor-Website [2] heruntergeladen werden. Abhängig von der Speichergröße des verwendeten Modul-Typs 512 KB (blau) oder 1 MB (schwarz) ist entweder die Datei beckdac_4mbit.bin oder beckdac_8mbit.bin die zutreffende Wahl, obwohl theoretisch die 4-Mbit-Version auf beiden Modul-Typen laufen müsste.

Zum Flashen des ESP8255 sind mehrere Programmer geeignet, der Autor hat drei Typen getestet: FLASH_DOWNLOAD_TOOLS [5], esp8266_flasher [6] und esp8266flasher [7] (der einzige Unterschied zwischen den beiden letzten ist der Unterstrich im Namen). Zwar gelang das Flashen mit allen Programmern, das stabilste Ergebnis wurde jedoch mit dem esp8266flasher erzielt.

Tabelle 1. Adressräume der Dateien beckdac_4mbit.bin und beckdac_8mbit.bin.

Adresse	Bin-Datei	Quelle
0x00000	0x00000.bin	Beckdac [1]
0x3E000	blank.bin	Espressif [4]
0x40000	0x40000.bin	Beckdac [1]
0x7E000	blank.bin	Espressif [4]
Außerdem bei der 8-Mbit-Version (1 MB):		
0x80000	0x40000.bin	Beckdac [1]
0xFE000	blank.bin	Espressif [4]

Anleitung

Folgende Schritte müssen durchlaufen werden:

- Vorbereitung: Herunterladen und Installieren der Software und Hardware.
- Optional: Testen des ESP8266 vor dem Flashen.
- Flashen der erstellten Firmware.
- Konfigurieren durch Einstellen der Parameter.
- Testen.

1. Vorbereitung

Laden Sie die Firmware von [2] herunter und entpacken Sie die ZIP-Datei.
Laden Sie die Datei esp8266flasher.exe von [7] herunter.

Halten Sie einen Umsetzer USB/Seriell 3,3 V bereit, beispielsweise das Modul FT232R USB/Serial Bridge/BOB von Elektor. Falls noch nicht geschehen, installieren Sie die nötigen Treiber. Fertigen Sie einen Adapter an, der den USB/Seriell-Umsetzer mit der linken Seite von JP2 auf dem Android I/O-Board verbindet. Dies betrifft nur Masse und die beiden Datenleitungen, die Betriebsspannung des Umsetzers bleibt frei. Das Android I/O-Board muss aus einer eigenen Spannungsquelle versorgt werden, da die meisten Umsetzer nicht genügend Strom liefern.

2. Testen des ESP8266 vor dem Flashen

Stecken Sie das ESP8266 auf das Android I/O-Board und verbinden Sie den USB/Seriell-Umsetzer und die externe Stromversorgung mit dem Android I/O-Board.

Auf dem ESP8266 leuchtet die rote LED auf, sobald die Betriebsspannung anliegt. Die blaue LED blinkt beim Starten.

Auf dem PC starten Sie ein Terminalprogramm, beispielsweise Teraterm. Stellen Sie die Verbindung mit dem USB/Seriell-Umsetzer her, stellen Sie die Baudrate auf 115200 Baud ein.
Wenn Sie auf dem PC die Zeichen „AT“ tippen, erhalten Sie „Error“ zurück. Dies ist korrekt.

Suchen Sie nach WLAN-Netzwerken in Ihrer Umgebung. Sie finden ein neues Netzwerk mit beispielsweise AI-THINKER_xxxxx (abhängig vom Hersteller), wobei die letzten sechs Zeichen die hexadezimalen Werte der letzten drei Bytes der MAC-Adresse sind.

Das ESP8266 arbeitet!

3. Flashen des ESP8266

Stecken Sie Jumper JP1 auf das Android I/O-Board auf, so dass sich das ESP8266 im Flash-Modus befindet. Wenn Sie einen Widerstand verwenden, darf sein Wert nicht über 1 kΩ liegen.
Verbinden Sie den USB/Seriell-Umsetzer mit JP2 und schließen Sie die Stromversorgung am Android I/O-Board an.

Starten Sie das Programm esp8266flasher.exe.

Klicken Sie auf den Tab „Config“.

Klicken Sie auf das Zahnrad, laden Sie die Datei beckdac_4mbit.bin oder beckdac_8mbit.bin, setzen Sie den Offset auf 0x00000 und markieren Sie die Datei mit dem „x“ (**Bild 4**).

Wählen Sie im Tab „Operation“ den zutreffenden seriellen Port aus, falls er noch nicht aktiv ist (**Bild 5**).

Klicken Sie auf „Flash“.

Das Programm muss jetzt innerhalb einiger Sekunden die MAC-Adressen anzeigen (**Bild 6**). Falls dies nicht geschieht, starten Sie das Android I/O-Board neu, indem Sie die Stromversorgung kurz unterbrechen.

Das Flashen der 4-Mbit-Datei dauert etwa 90 Sekunden, die 8-Mbit-Datei braucht 180 Sekunden. Warten Sie, bis wieder die Meldung „Ready“ erscheint (**Bild 7**).

Damit ist der Vorgang abgeschlossen. Entfernen Sie Jumper JP1 und den USB/Seriell-Umsetzer, bringen Sie die Jumper auf JP2 wieder an und starten Sie das Android I/O-Board neu, indem Sie die Stromversorgung kurz unterbrechen.

4. Konfigurieren

Stellen Sie das Netzwerk auf ESP_xxxxxx ein.
Bauen Sie über das Terminalprogramm (beispielsweise Teraterm) eine Telnet-Verbindung zur Adresse 192.168.4.1, Port 23 auf.

Geben Sie folgende Kommandos ein:

- `+++AT FLASH 1`
- `+++AT BAUD 9600`
- `+++AT MODE 3`

Die nächste Eingabe ist die SSID und das Passwort Ihres lokalen Netzwerks:

- `+++AT STA SSID Password`

Danach folgt die für das ESP8266 gewählte SSID mit dem zugehörigen Passwort:

- `+++AT AP SSID Password 3`

Das ESP8266 führt jetzt den eigenen Reset herbei, dann baut es ein neues WLAN-Netzwerk auf. Warten Sie, bis das ESP8266 neu gestartet ist.

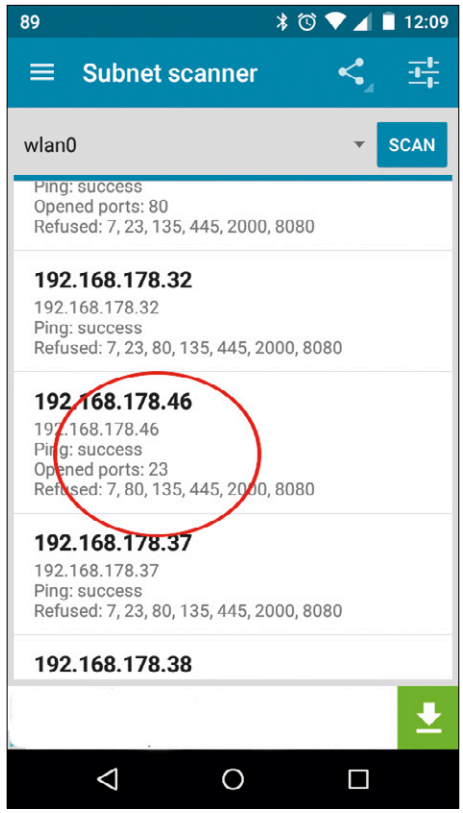


Bild 8. Ein Subnet-Scan auf Port 23 findet das ESP8266 im Netzwerk.

Damit ist die Konfiguration abgeschlossen. Wenn das ESP8266 ausschließlich als STA oder AP arbeiten soll, ist MODE 3 durch MODE 1 oder MODE 2 zu ersetzen, und der AP beziehungsweise der STA müssen nicht konfiguriert werden.

5. Testen

Sie können sich mit dem Android-Smartphone oder –Tablet im WLAN-Netzwerk des ESP8266 mit den Zugangsdaten einloggen, die Sie zuvor mit dem Kommando +++AT AP festgelegt haben.

Wenn sich das ESP8266 innerhalb der Reichweite Ihres lokalen WLAN-Netzwerks befindet, loggt es sich dort ein. Mit der App Ping Tools [8] können Sie die IP-Adresse ermitteln, indem Sie einen Netzwerk-Scan nach Port 23 ausführen (**Bild 8**).

Das Android I/O-Board können Sie dadurch testen, dass Sie die Demo-App öffnen, oder Sie starten eine Telnet-App [9] und geben ein Kommando ein, zum Beispiel „G Z“.

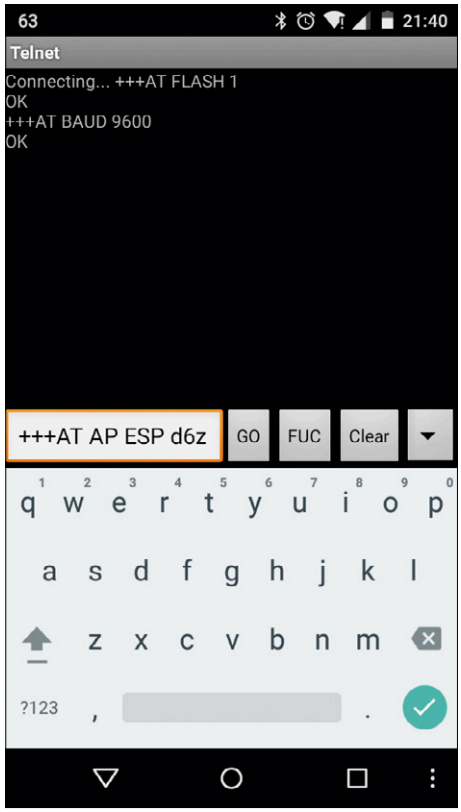


Bild 9. Konfigurieren des ESP8266 mit „Simple Telnet“.

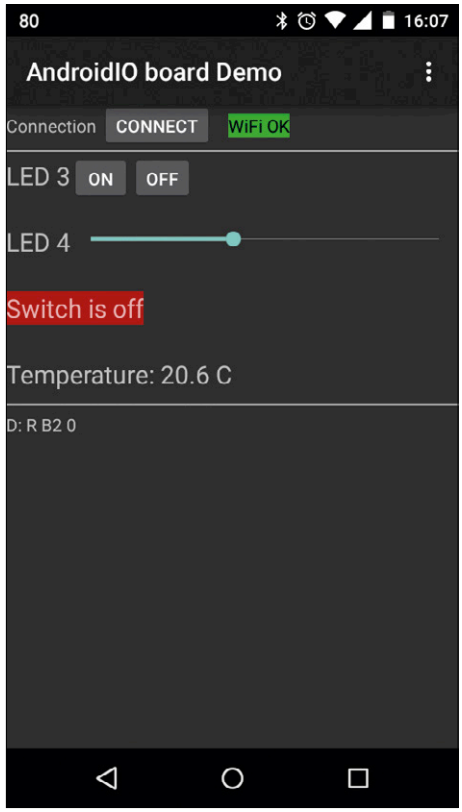


Bild 10. Demo-App für die Kommunikation des ESP8266 mit dem Android I/O-Board.

Wenn sich nichts tut

Haben Sie oben stehende Anleitung genau befolgt, kann eigentlich nichts schief gehen. Sollte dies doch passiert sein, gehen Sie folgende Punkte durch:

Das ESP8266 generiert hohe Störsignale auf der Betriebsspannung, so dass eine ergänzende Glättung nötig ist. Bringen Sie den zusätzlichen Elko an.

Das ESP8266 nimmt Spitzenströme in der Größenordnung 200...300 mA auf. Die meisten FT232-Umsetzer (auch der BoB von Elektor) können diese Ströme nicht liefern. Beim Flashen muss eine externe Stromversorgung verwendet werden.

Haben Sie Geduld! Beim Autor trat das Phänomen auf, dass das ESP8266 nach dem Flashen zwar ein WLAN-Signal sendete, darüber hinaus jedoch nicht reagierte. Am nächsten Tag arbeiteten sämtliche Funktionen ohne weiteres Zutun (!) absolut ordnungsgemäß. Über die Ursache lässt sich nur mutmaßen.

Leider kann der Speicherinhalt des ESP8266 nach dem Flashen nicht verifiziert werden. Wenn die Parameter fehlerhaft gesetzt wurden, können Fehler auftreten. Bei den Flash_Download_Tools sind die Parameter einstellbar. Das Schreiben bei 20 MHz mit DIO ist eine sichere Option.

Das Einloggen in das ESP8266 (WLAN-Netzwerk ESP_xxxxxx) ist unter Windows 8.1 häufig nicht möglich. Android-Smartphones oder –Tablets und PCs mit Linux haben damit selten Probleme, auch wenn es manchmal bis zu einer Minute dauert. Der Autor benutzte deshalb Simple Telnet aus dem Play Store zum Konfigurieren (**Bild 9**).

Manchmal hat es den Anschein, dass die Telnet-Verbindung nicht zustande gekommen ist („Connecting...“). Trotzdem werden eingegebene Zeichen korrekt übertragen.

Die +++AT-Kommandos haben nur Wirkung, wenn sie über WLAN zum ESP8266 gelangen, nicht über den seriellen Port. Vergessen Sie nicht, die Baudrate auf 9600 Baud zu setzen!

Demo-App

Zum Download [2] dieses Beitrags gehört eine Demo-App, die zwei LEDs, den NTC und den Touch-Button einbezieht (siehe **Bild 10**). Mit der App kann LED 3 auf dem Android I/O-Board geschaltet werden, und LED 4 ist im Bereich 0...100 % dimmbar. Aktionen am Touch-Button werden ohne Verzug zur App übertragen. Außerdem liest die App die vom NTC gemessene Temperatur im Sekundenabstand aus. Der Übertragungsweg (Bluetooth, WLAN, USB Accessory oder USB Host) muss auf das verbaute Kommunikationsmodul eingestellt werden. ⏮

(150703)gd

Weblinks

[1] https://github.com/beckdac/ESP8266-transparent-bridge
[2] www.elektormagazine.de/150703
[3] README.md auf https://github.com/beckdac/ESP8266-transparent-bridge
[4] https://github.com/espressif/esp8266_at/tree/master/bin
[5] FLASH_DOWNLOAD_TOOLS_v1.2_150512.zip: https://drive.google.com/file/d/0B_ctPy0pJuW6V2ViNDR0NGdnYTQ/view?pli=1
[6] esp8266_flasher: https://drive.google.com/file/d/0B3dUKfqzZnlwVGc1YnFyUjgxelE/view?usp=sharing
https://github.com/Stadslab/ESP8266_example/tree/master/ESP8266_flasher_V00170901_00_Cloud%20Update%20Ready
[7] esp8266_flasher: https://github.com/nodemcu/nodemcu-flasher
[8] Ping Tools: https://play.google.com/store/apps/details?id=ua.com.streamsoft.pingtools
[9] Simple Telnet: https://play.google.com/store/apps/details?id=com.telnet

Opamp-Experimentier-Kit für myDAQ

Zehn Lehrstücke in zehn Konfigurationen

Von **Sunil Malekar** und **Roy Aarts** (Elektor-Labor)

Das Label myDAQ steht für ein kostengünstiges Messsystem von National Instruments, das vornehmlich als Lehr- und Schulungsmittel konzipiert ist. Mit externer Hardware kombinieren lässt sich myDAQ über eine von außen zugängliche Kontaktleiste. Hier stellen wir eine unkomplizierte Erweiterung vor, mit der myDAQ zum Experimentierfeld für die elementaren Funktionen von Operationsverstärkern wird.

National Instruments brachte unter der Bezeichnung myDAQ ein unkompliziertes, aber vielseitiges Messsystem heraus, das acht oft benötigte Messgeräte in sich vereint. In den Bereichen Schule und Lehre hat sich myDAQ bereits seit einiger Zeit etabliert, für den täglichen Einsatz im Elektronik-Labor ist es gleichermaßen gut geeignet. In myDAQ sind unter anderem ein Zweikanal-Oszilloskop, ein Digitalmultimeter und ein Zweikanal-Funktionsgenerator integriert, und für die Verbindung zur Außenwelt sind diverse digitale und analoge Eingänge und Ausgänge verfügbar. Bedient wird myDAQ über einen PC oder Laptop, der mit dem myDAQ-System über eine USB-Schnittstelle kommuniziert. Hardware-Erweiterungen lassen sich in das System über eine Steckleiste an der Gehäuseseite einbinden.

Die Experimentierplatine für myDAQ, die wir hier vorstellen, soll Newcomer und Lernende mit der Welt der Operationsverstärker (kurz: Opamp) vertraut machen. Außer den elementaren Opamp-Funktionen lassen sich auch diverse Filter realisieren. Auf der Platine befinden sich elektronische Schalter, über die vom PC aus die Opamps zusammengeschaltet und konfiguriert werden können. Signalparameter wie Schwingungsform, Frequenz und Amplitude sind über virtuelle Bedienelemente auf dem Bildschirm einstellbar. Die Signale an den Opamps misst das

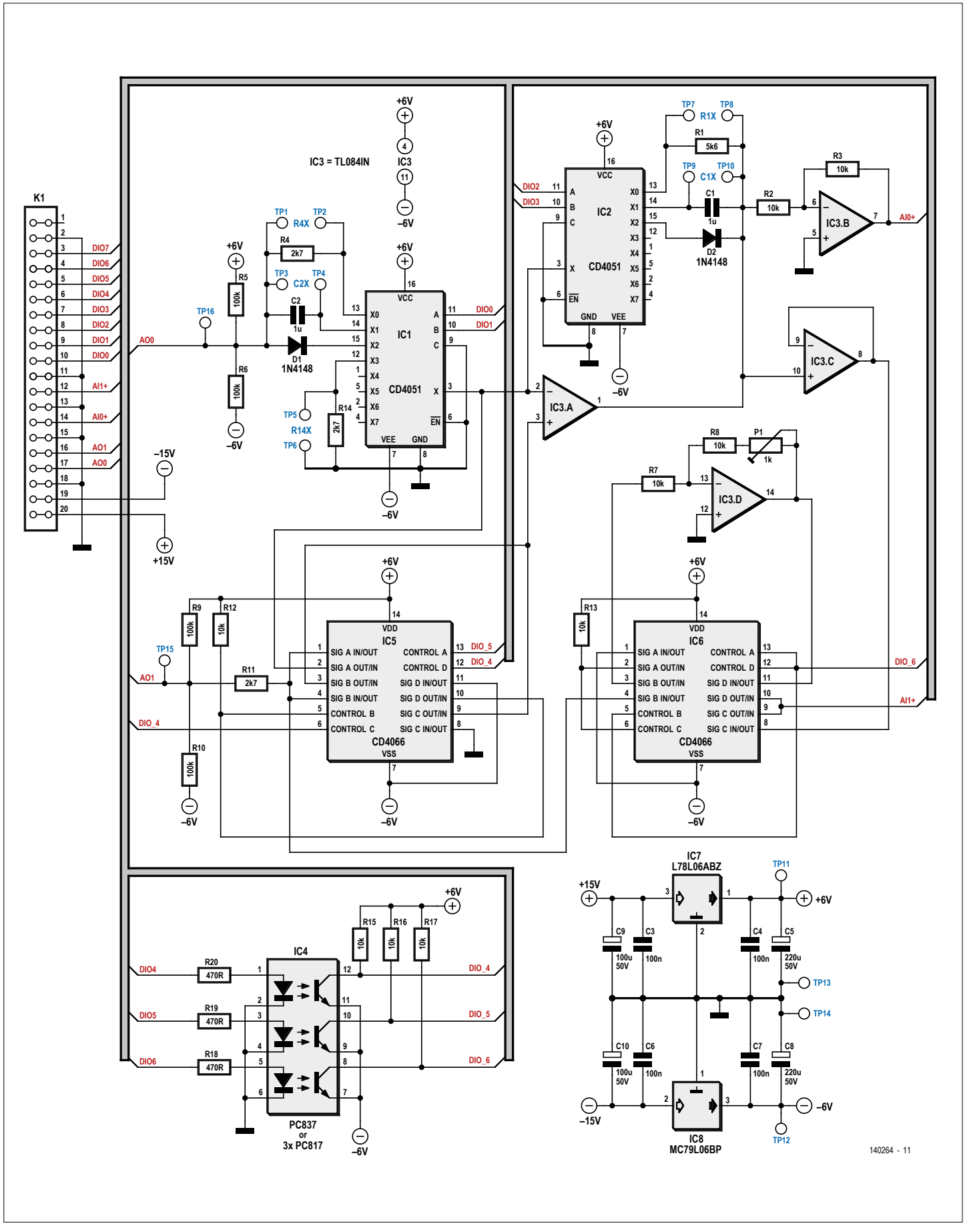


Bild 1. Die Hardware erscheint kompliziert, doch sie besteht nur aus mehreren Opamps, die über elektronische Schalter in diverse Konfigurationen versetzt werden können.

System in Echtzeit. Die Eingangs- und Ausgangssignale der beiden Kanäle werden auf dem Bildschirm in vier Oszilloskop-Fenstern dargestellt.
Die Software für das Opamp-Experimentier-Kit wurde, wie könnte es anders sein, in LabVIEW geschrieben. Deshalb kann die Kombination mit dem myDAQ auch demonstrieren, wie mit LabVIEW Ausgangssignale programmiert werden und Eingangssignale auf dem Bildschirm erscheinen.

Die myDAQ-Hardware

Das Messsystem myDAQ besitzt zwei analoge Eingänge (AI) mit der maximalen Sample-Rate 200 Kilosamples/s und der Auflösung 16 bit, zwei analoge Ausgänge (AO) mit identischer Sample-Rate und Auflösung, acht digitale Eingänge oder Ausgänge (DIO), einen 32-bit Counter oder Timer, ein 3,5-stelliges Digitalmultimeter (DMM) sowie eine Stromversorgung, die +5 V/100 mA und ±15 V/32 mA liefern kann. Die an der Gehäuseseite befindliche Steckleiste, die für Hardware-Erweiterungen vorgesehen ist, hat 20 Kontakte.
Die Verbindung zum PC oder Laptop stellt eine Schnittstelle USB 2.0 High-Speed her. Im Lieferumfang des myDAQ sind diverse Software-Messinstrumente für Standardfunktionen enthalten, die selbstständig laufen (ELVISmx Software Instruments). Wenn eigene Anwendungen realisiert werden sollen, wird eine Version von LabVIEW benötigt.

Die Schaltung

Auf den ersten Blick mag die Schaltung in **Bild 1** komplex erscheinen, doch näher betrachtet ist sie leicht überschaubar. Opamp IC3.A hat die Funktion des Versuchsobjekts, er kann über die analogen 8-Kanal-Multiplexer/Demultiplexer IC1 und IC2 in die unterschiedlichen Konfigurationen geschaltet werden. Über IC1 wird die Komponente gewählt, die am Opamp-Eingang liegt. Dies kann eine Diode (D1), ein Kondensator (C2) oder ein Widerstand (R4) sein, der dem invertierenden Opamp-Eingang vorgeschaltet wird. Für die nicht invertierende Konfiguration kann der Eingang über einen Widerstand (R14) an Masse gelegt werden. IC2 schaltet die Komponente um, die im Gegenkoppelzweig des Opamps liegt: Eine Diode (D2), ein Kondensator (C1) oder ein Widerstand (R1). Damit lassen sich bereits zahlreiche Konfigurationen realisieren. Außerdem befinden sich Einsteck-Buchsen auf der Platine, die zusätzliche Widerstände oder Kondensatoren aufnehmen können (R1X, C1X, R4X, C2X, R14X). Diese Komponenten liegen parallel zu den zugehörigen, auf der Platine montierten Komponenten. Der Ausgang von Versuchsobjekt IC3.A steuert den invertierenden Opamp IC3.B sowie die nicht invertierende Stufe IC3.C. Der Ausgang der invertierenden Stufe ist fest mit dem ersten analogen Eingang des myDAQs verbunden. Abhängig von den Einstellungen auf dem PC-Bildschirm gelangen das nicht invertierte Signal oder das Ausgangssignal des einstellbaren Verstärkers IC3.D zum zweiten analogen Eingang des myDAQs.
Ferner sind noch zwei CMOS-ICs 4066 vorhanden (IC5 und IC6), in denen jeweils vier analoge Schalter integriert sind. IC5 ist dafür zuständig, den zweiten Ausgang des myDAQs über R11 entweder mit dem invertierenden oder dem nicht invertierenden Eingang von IC3.A zu verbinden. Damit sind entweder ein Addierer oder ein Subtrahierer verfügbar. Außerdem kann das Signal, das hinter R1 liegt, den einstellbaren Verstärker IC3.D steuern. Über Schalter C von IC5 wird der nicht invertierende Eingang von IC3.A bei verschiedenen Konfigurationen an Masse gelegt. Die Schalter von IC6 leiten entweder das Ausgangssignal von IC3.B oder das Ausgangssignal von IC3.C zum zweiten Eingang des myDAQs weiter (über Schalter C). Der einstellbare Verstärker IC3.D wird über Schalter B (Eingang) und Schalter D (Ausgang) von IC6 ausgewählt.
Die analogen Schalter und die Multiplexer werden vom myDAQ über die digitalen Leitungen DIO0...DIO6 gesteuert. In **Tabelle 1** ist angegeben, welche logischen Zustände zu den einzelnen Konfigurationen gehören. Für die Leitungen DIO4, DIO5 und DIO6, die die analogen Schalter von IC5 und IC6 steuern, gilt eine Besonderheit. Da die Leitungen 3,3-V-Signale liefern, IC5 und IC6 jedoch an der symmetrischen Betriebsspannung ±6 V liegen, ist eine Pegelanpassung notwendig. Diese Aufgabe übernehmen die Optokoppler, die in IC4 integriert sind. Wenn eine Eingangsleitung auf hoher Spannung liegt, wird der zugehörige Fototransistor leitend. Der Fototransistor zieht den Optokoppler-Ausgang von +6 V nach –6 V, was gleichzeitig ein Invertieren des schaltenden Signals bedeutet.

Die Stromversorgung der Schaltung arbeitet mit einem positiven und einem negativen Spannungsregler (IC7 und IC8), sie setzen die vom myDAQ kommende Betriebsspannung ±15 V auf ±6 V herab. Die 20-polige Steckleiste K1 stellt die Verbindungen von der Platine zum myDAQ her.

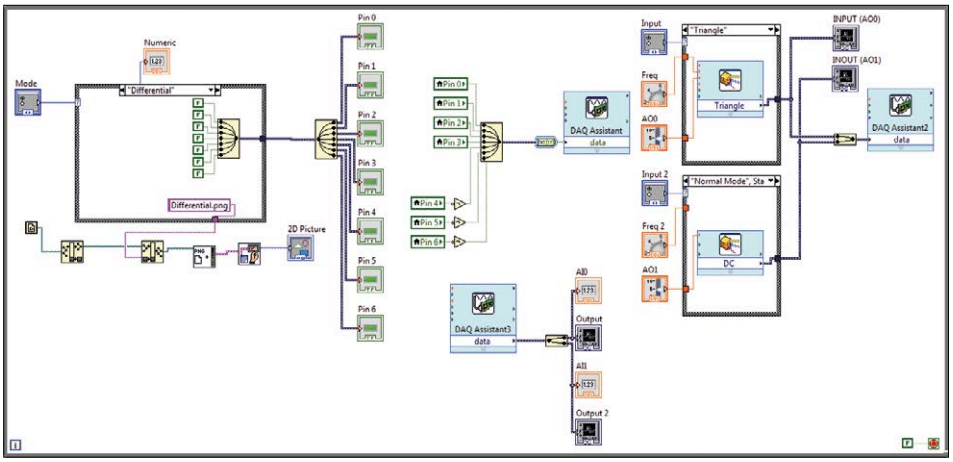


Bild 3. Funktionsschema des LabVIEW-Programms, das für das Opamp-Experimentier-Kit entwickelt wurde.

Aufbau

Das Layout unserer myDAQ-Experimentierplatine ist in **Bild 2** wiedergegeben. Die ausschließlich bedrahteten Bauelemente haben reichlich Platz, so dass der Aufbau problemlos möglich ist. Für die ICs können Fassungen montiert werden, insbesondere wenn die Opamps versuchsweise gegen andere, pinkompatible Typen austauschbar sein sollen. Auf der Platine sind nur niederfrequente Signale vorhanden, so dass Störungen von außen äußerst unwahrscheinlich sind. Die Einsteck-Buchsen TP1...TP6 können einer 10-poligen Buchsenleiste durch Abbrechen entnommen werden. Auch einzelne Buchsenkontakte sind im Handel, sie sind jedoch weniger gängig.

Nachdem die Platine fertig gestellt ist, wird sie auf die Seitenkante des myDAQ aufgesteckt. Wir empfehlen, an der Platinenaußenkante, die der Steckleiste gegenüber liegt, kleine Stützen anzubringen. Dann wird die Platine nicht allein von der Steckleiste getragen, wenn beim Einstecken zusätzlicher Bauelemente oder Drehen am Trimmpoti auf die Platine Druck ausgeübt wird.

Wir haben unter der Bestellnummer 140264-71 einen kleinen Bausatz zusammengestellt, der außer der Platine den seltenen Steckverbinder K1 und den wenig gebräuchlichen Spannungsregler MC79L06BP-AP enthält.

Die Software

Für den Einsatz unserer Experimentierplatine haben wir ein Software-Modul in LabVIEW geschrieben. Damit ist es möglich, unter diversen Opamp-Konfigurationen auszuwählen, die Eingangssignale zu variieren und die Ausgangssignale zu betrachten. Das Software-Modul kann frei von der Projektseite [1] heruntergeladen werden, der Einsatz setzt jedoch eine auf dem PC installierte Version von LabVIEW voraus.

Eine schematische Übersicht über die Funktionen der Software zeigt **Bild 3**. Auf der linken Seite ist die Auswahl der Opamp-Konfigurationen dargestellt, abhängig davon werden die digitalen Ausgänge des myDAQs gesteuert. Gegenüber, auf der rechten Seite, werden die Signale der analogen myDAQ-Ausgänge generiert. Unten in der Mitte des Diagramms sind die Bildschirmfenster angedeutet, die die analogen Signale auf dem Bildschirm darstellen. In Software wurden hier außer der digitalen Schaltlogik ein Funktionsgenerator und ein Oszilloskop mit vier Fenstern realisiert.

Nach dem Start präsentiert sich die Software wie in **Bild 4** wiedergegeben. Die zehn Schaltflächen auf der linken Seite dienen zur Auswahl der Opamp-Konfiguration. Darunter wird die elektrische Schaltung der gewählten Konfiguration eingeblendet. Die sieben mit Pin 0...Pin 6 beschrifteten Felder zeigen die logischen Zustände der digitalen myDAQ-Ausgänge an. Die Signalkombinationen schalten

Stückliste

Widerstände:
(5 %/250 mW)
R1 = 5k6
R2,R3,R7,R8,R12,R13,R15,R16,R17 = 10 k
R4,R11,R14 = 2k7
R5,R6,R9,R10 = 100 k
R18,R19,R20 = 470 Ω
P1 = Trimmpoti 1 k

Kondensatoren:
C1,C2 = 1 µ/50 V, stehend
C3,C4,C6,C7 = 100 n
C5,C8 = 220 µ/50 V, stehend, RM 10 mm
C9,C10 = 100 µ/50 V, stehend, RM 8 mm

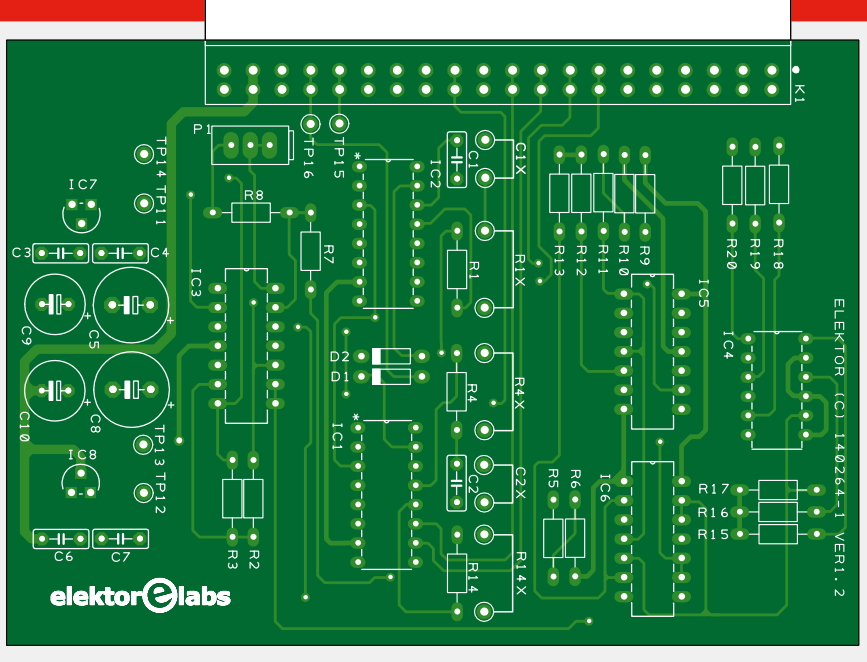


Bild 2. Die übersichtlich gestaltete Platine ist schnell aufgebaut, bestückt wird sie ausschließlich mit bedrahteten Bauelementen.

die analogen Multiplexer auf der Experimentierplatine, ihre Bedeutungen sind in **Tabelle 1** zusammengefasst. Davon rechts sind die Einstellelemente für die myDAQ-Ausgangssignale angeordnet. Als Signalformen sind Gleichspannung, Sinus, Dreieck und Rechteck wählbar. Mit den senkrechten Schiebepotentiometern lassen sich die Höhe und Polarität der Gleichspannung und die Amplitude der wechselförmigen Signale einstellen. Die Drehknöpfe dienen zum Einstellen der Signalfrequenzen. Die Frequenzen liegen im niedrigen Bereich, damit die Signale in den Oszilloskop-Fenstern, rechts davon, gut erkennbar sind. In den linken Fenstern erscheinen die Eingangssignale, während in den rechten Fenstern die Ausgangssignale dargestellt werden.

Wir wünschen Ihnen viel Freude mit myDAQ und unserem Opamp-Experimentier-Kit!

(140264)gd

Tabelle 1.

Konfiguration	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0
Invertieren	X	0	0	1	0	0	0	0
Nicht invertieren	X	0	0	0	0	0	1	1
Addieren	X	0	1	1	0	0	0	0
Subtrahieren	X	0	0	0	0	0	0	0
Differenzieren	X	0	0	1	0	0	0	1
Integrieren	X	0	0	1	0	1	0	0
Logarithmieren	X	0	0	1	0	1	0	0
Exponenzieren	X	0	0	1	0	0	1	0
Komparator	X	0	0	1	1	1	0	0
Verstärkung einstellen	X	1	0	1	0	0	0	0

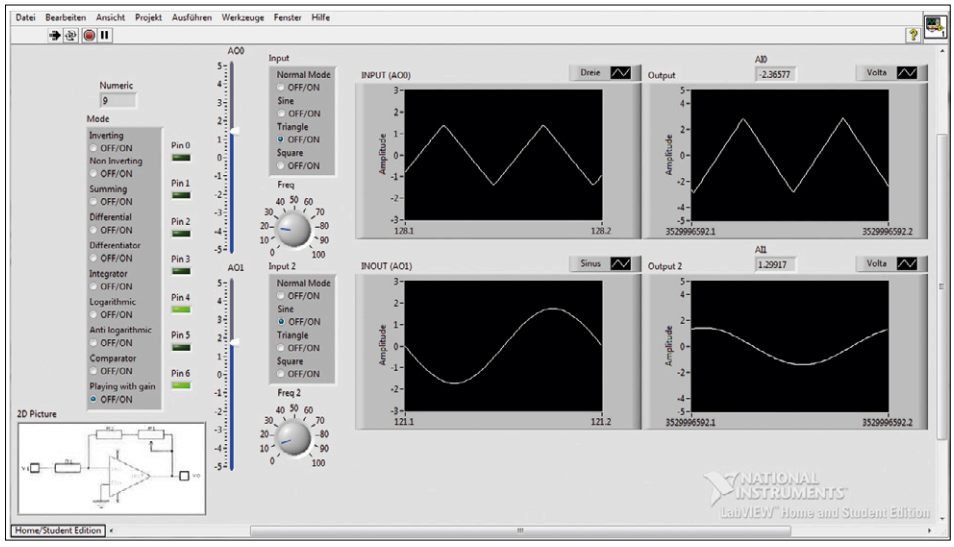
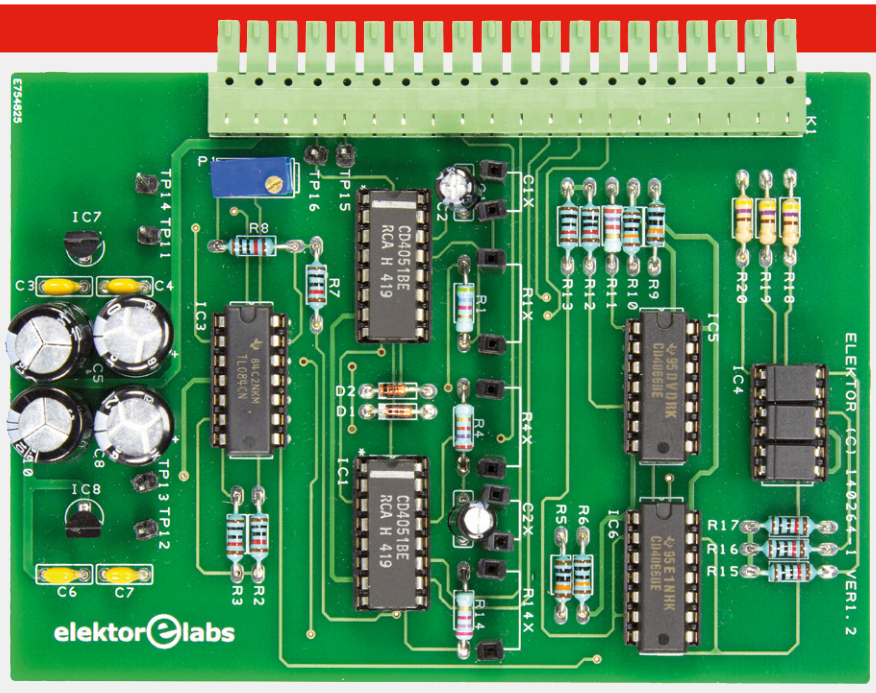


Bild 4. Die Software auf dem Bildschirm: Links werden die Konfigurationen und Eingangssignale gewählt, daneben sind die Oszilloskop-Fenster mit den Eingangs- und Ausgangssignalen angeordnet.

Weblink

[1] www.elektormagazine.de/140264



Halbleiter:

IC1,IC2 = CD4051BE
IC3 = TL084
IC4a,IC4b,IC4c = PC837 oder 3 · PC817
IC5,IC6 = CD4066BE
IC7 = L78L06
IC8 = MC79L06
D1,D2 = 1N4148

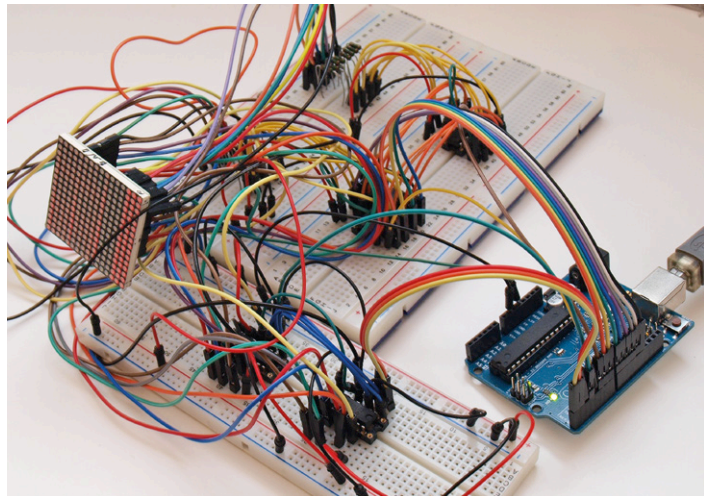
Außerdem:

IC-Fassung DIP-6
IC-Fassung DIP-14
IC-Fassung DIP-16
K1 = Steckverbinder 20-polig, RM 3,81 mm, für Verbindung mit myDAQ
TP1...TP10 = Buchsenleiste einreihig, RM 2,54 mm
TP11...TP16 = Stiftleiste einreihig, RM 2,54 mm
Platinenbausatz 140264-71 (Platine, 20-poliger myDAQ-Steckverbinder und Spannungsregler MC79L06)

Laufschrift für Arduino

Mit 16 x 16 LEDs

Von **Tam Hanna**



Mit einem Arduino, einem LED-Matrix-Display und ein paar zusätzlichen Bauteilen kann man eine Laufschrift verwirklichen. Beim Erstellen der Software hilft ein kostenloses Tool von Mikroelektronika, das einen ganzen Buchstaben-Satz im Nu in Codezeilen umsetzt.

Ob New York, Frankfurt, Wien oder Peking: In der Börse gehören sie einfach dazu, die „Scrolling Banners“, die Kursdaten, Statistiken und nützliche Hinweise anzeigen. Die Realisierung solcher Laufschriften ist in mehrerlei Hinsicht eine interessante Aufgabe, da man Hardware- genauso wie Software-Kenntnisse mitbringen muss. Doch keine Bange, wir zeigen alles Schritt für Schritt.

Schaltung

Wir beginnen unsere Arbeiten mit der in **Bild 1** gezeigten Schal-

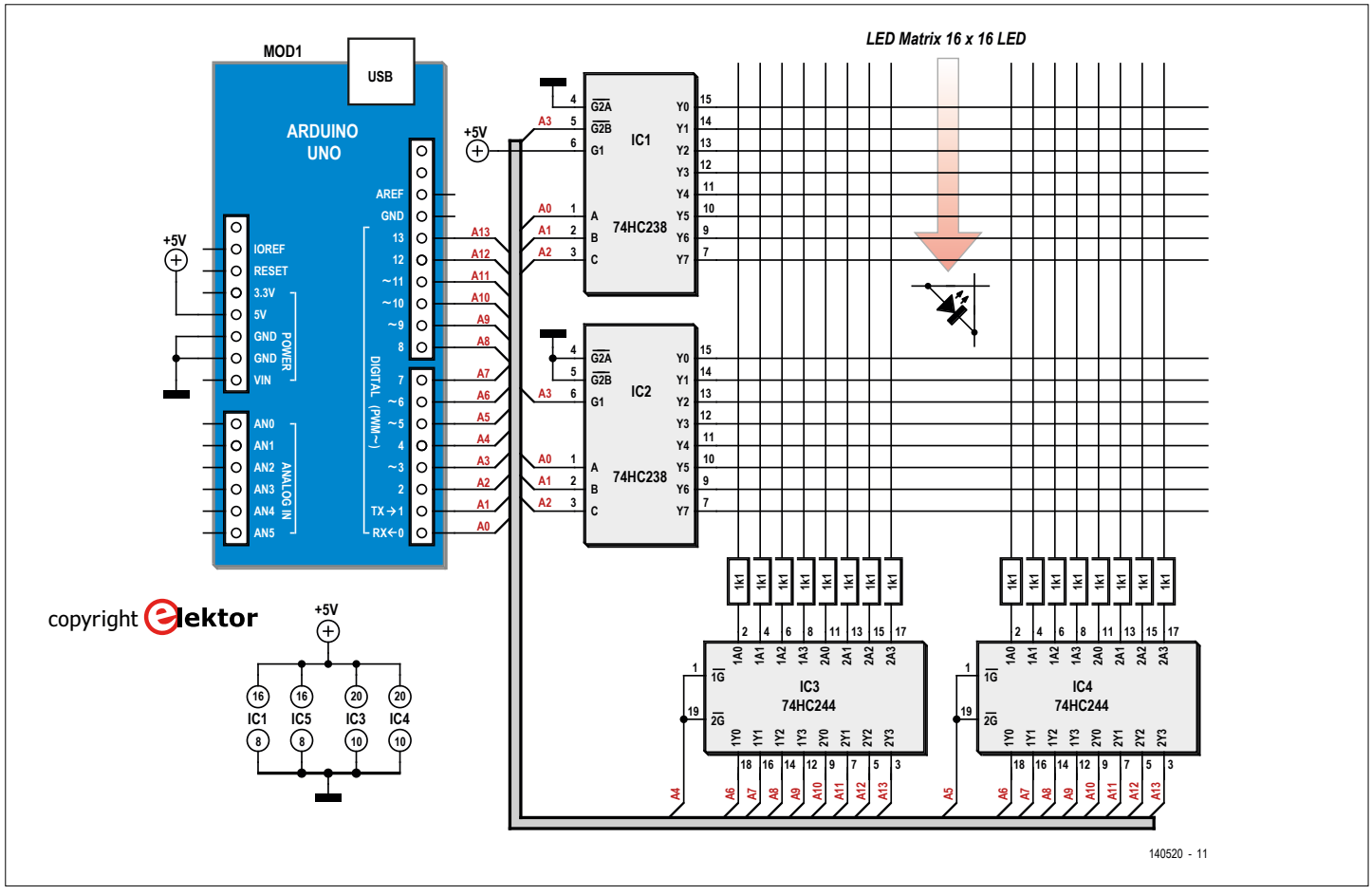


Bild 1. Schaltplan. Die beiden 238er-ICs setzen je eine Reihe der LED-Matrix unter Spannung; das eigentliche Ein- und Ausschalten einzelner LEDs erfolgt über die 244er. Die Ansteuerung nutzt alle digitalen Portpins des Arduinos.

tung. An den Kreuzungspunkten der Zeilen- und Spaltenleitungen befindet sich je eine Leuchtdiode. Unsere 16x16er-LED-Matrix wird für die Ansteuerung per Arduino in vier Segmente unterteilt. Die 238er-ICs haben die Aufgabe, die Zeilen nacheinander zu aktivieren.
Das eigentliche Einschalten der einzelnen LEDs erfolgt in einer aktivierten Zeile durch zwei ICs der Bauart 244, indem auch die Spaltenleitung durch Low-Ziehen des jeweiligen Pins aktiviert wird. Die Ansteuerung nutzt alle digitalen Portpins des Arduinos. Zur Steigerung der Helligkeit bietet sich die Nutzung von Transistoren an – ein Thema, auf das wir hier aus Platzgründen nicht weiter eingehen.

Erste Schritte zum Code

Unser Fernziel ist die Ansteuerung mit einem klassischen Arduino: Zwecks Vereinfachung der Entwicklung beginnen wir die Arbeit aber auf einem leistungsstärkeren Arduino Zero, der dank Debugger und ausreichend Ressourcen die Entwicklung der Algorithmen erleichtert – das Optimieren erfolgt später.
Diese auf den ersten Blick unsorgfältig wirkende Vorgehensweise erweist sich in der Praxis oft als sehr sinnvoll. Ein System lässt sich im Allgemeinen leichter „reduzieren“, wenn es erst einmal funktioniert – beim Erreichen dieses Zustands ist ein Mehr an Leistungsfähigkeit immer hilfreich.
Da die verwendeten ICs aus der HC-Familie sind, funktionieren sie auch mit 3,3 V und überstehen die Übersiedelung zwischen den beiden Arduinos somit ohne Probleme. Achten Sie bei der Arbeit mit dem Zero nur darauf, die Versorgungsspannung aus dem 3V3-Pin des Arduino zu beziehen. Führen Sie danach das Programm in **Listing 1** aus.
Dank der beiden 238er-ICs erfolgt das Adressieren der Reihen durch das Ausgeben einer vierstelligen Binärzahl an den vier Leitungen A0...A3. In der Funktion `activateRow` setzen wir zuerst alle Leitungen auf low, dann innerhalb von vier If-Abfragen – je nach den Bits in der Variablen `_which` - die entsprechenden Leitungen auf high.
Der Code in **Listing 2** bringt die LEDs zum Laufen.
Wir aktivieren mit dem wiederkehrenden Aufruf von `activateRow` Zeile für Zeile. In der inneren Schleife setzen wir zuerst alle Spaltenleitungen auf high, um die Spalten zu deaktivieren. Danach wird Spalte für Spalte aktiviert. Eine kleine Verzögerung mit `delay` führt zu einem laufenden LED-Punkt.

Hardwarefehler

Es kommt in der Praxis manchmal zu Fehlern bei der Verschaltung der Matrix. Anstatt alles nochmal abzustecken und neu aufzubauen, kann man kleine Fehler in der Software ausbügeln. Beim Autor erfolgte dies in Form der Routine `mapRow` – die Methode ordnet die logischen Reihen den physikalischen zu (**Listing 3**).

Font herbei

Damit sind wir zum Ansteuern von einzelnen Punkten befähigt. Zur Darstellung von Zeichen ist eine Schriftart erforderlich: Beim PC liegen die als Fonts bezeichneten Buchstaben-Sätze meist im TrueType-Format vor. Es handelt sich dabei um ein Vektordatenformat, dessen Dekodierung kleine MCUs vor schier unlösbare Aufgaben stellt.
Zudem ist die Breite der meisten TTF-Fonts unterschiedlich: Ein I nimmt normalerweise weniger Platz ein als ein M. Dieses

Listing 1. Aktivieren der Reihen.

```
void setup() {
  for(int i=0;i<13;i++)
  {
    pinMode(i,OUTPUT);
    if(i>=6)digitalWrite(i,LOW);
  }
}

const int TAM_A0=0;
const int TAM_A1=1;
const int TAM_A2=2;
const int TAM_A3=3;

void activateRow(unsigned char _which)
{
  _which=mapRow(_which);
  digitalWrite(TAM_A0,false);
  digitalWrite(TAM_A1,false);
  digitalWrite(TAM_A2,false);
  digitalWrite(TAM_A3,false);
  if(_which&1) {
    digitalWrite(TAM_A0,true);
  }
  if(_which&2) {
    digitalWrite(TAM_A1,true);
  }
  if(_which&4) {
    digitalWrite(TAM_A2,true);
  }
  if(_which&8) {
    digitalWrite(TAM_A3,true);
  }
}
```

Listing 2. Laufender LED-Punkt.

```
void loop() {
   while(1==1)
  {
    for(int j=0;j<16;j++)
    {
      activateRow(j);
      for(int i=0;i<8;i++)
      {
        digitalWrite(13,true);
        digitalWrite(12,true);
        digitalWrite(11,true);
        digitalWrite(10,true);
        digitalWrite(9,true);
        digitalWrite(8,true);
        digitalWrite(7,true);
        digitalWrite(6,true);
        digitalWrite(6+i,false);
        delay(25);
      }
    }
  }
}
```

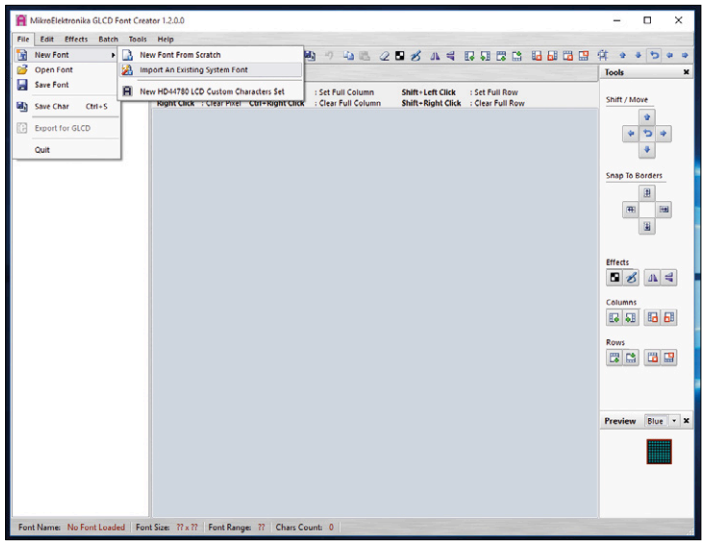


Bild 2. Import einer System-Schriftart in das Tool GLCD Font Creator.

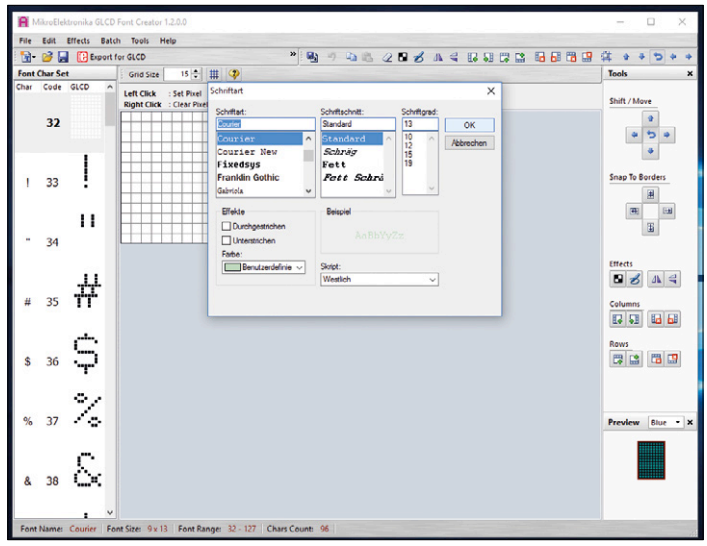


Bild 3. Einstellen des Schriftgrads in der Auswahlbox. Falls in der Auswahl nicht enthalten, muss man die Zahl manuell eingeben.

Problem lässt sich durch die Nutzung von monospaced-Fonts lösen. In den folgenden Schritten verwenden wir Courier.
Der Toolhersteller Mikroelektronika bietet ein hilfreiches Programm an, das Buchstaben-Sätze in Codezeilen umsetzen kann – und dieses Werkzeug ist in einer Basisversion sogar gratis. Laden Sie das unter [1] bereitstehende Programm GLCD Creator herunter, und erstellen Sie eine neue Schriftart auf Basis von Courier. Dazu müssen Sie wie in **Bild 2** gezeigt zuerst eine neue System-Schrift importieren.
Setzen Sie als Zeichenhöhe den Wert 13 fest. Da dies nicht vorgesehen ist, müssen sie den Wert wie in **Bild 3** gezeigt von Hand eingeben. Im nächsten Schritt bestätigen Sie die Importeinstellungen wie in **Bild 4** gezeigt – der darauffolgende Optimierungsprozess nimmt einige Zeit in Anspruch.
Prüfen Sie im nächsten Schritt die Größe der erzeugten Zeichen, die an der unteren Bildschirmkante angezeigt wird. Wenn dort nicht 10x13 steht, so folgt Batch > Columns > Ins Column. Danach ist Batch > Invert All notwendig.
Klicken Sie danach auf Export, und danach in die Rubrik MicroC.

Der Lohn der Mühen ist ein C-Array, das wie in **Bild 5** aussieht. Kopieren Sie den Code im nächsten Schritt in Notepad, und markieren Sie das unnötige Längenbyte wie in **Bild 6** gezeigt. Kopieren Sie den Text des Längenbytes in das Suchen-Feld des Ersetzen-Fensters. Dann beseitigen Sie alle Längenbytes durch Suchen & Ersetzen (**Bild 7**). Tauschen Sie nun noch den Datentyp-Ausdruck „short“ gegen ein „Int“ aus, um die Arbeiten abzuschließen.

Kopieren Sie das Courier10x13-Array nun in die .c-Datei, um die Arbeit fortzusetzen. Unsere Version der LED-Ansteuerung muss noch den linken und den rechten Teil des Displays auseinanderhalten (darum hatten wir uns beim oben gezeigten laufenden LED-Punkt noch nicht gekümmert). Die Methode in **Listing 4** aktiviert entweder den linken oder den rechten 244er-Baustein, je nach dem Bit _hi.
Erfahrene Leser monieren an dieser Stelle, dass man die Auswahl des gerade aktiven Teils des Displays mit einem Inverter und nur einem Pin hätte bewerkstelligen können. Dieser Ein-

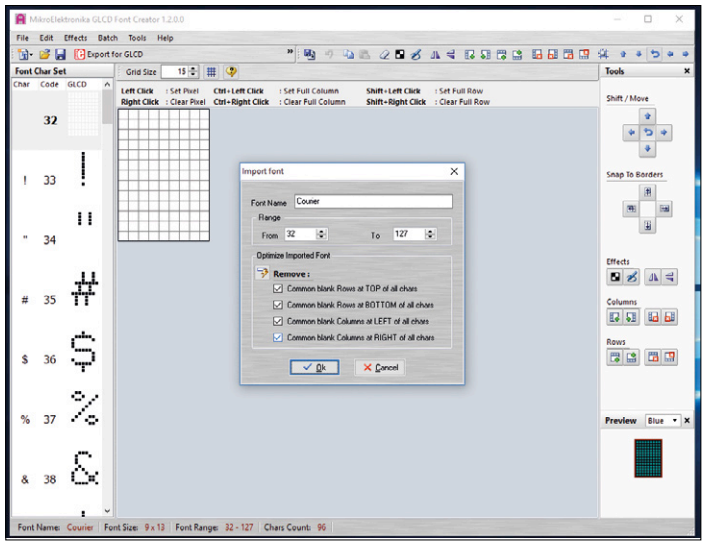


Bild 4. GLCD Font Creator entfernt unnötige Pixel automatisch.

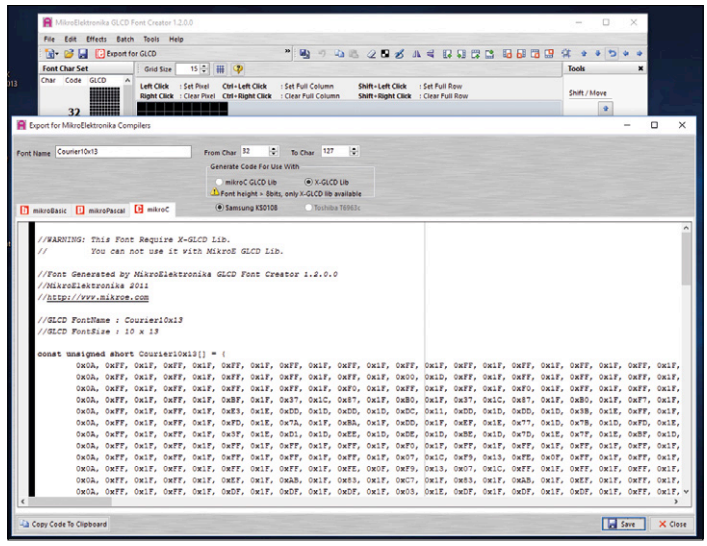


Bild 5. Resultat: Ein C-Array, das die Zeichendarstellung enthält.

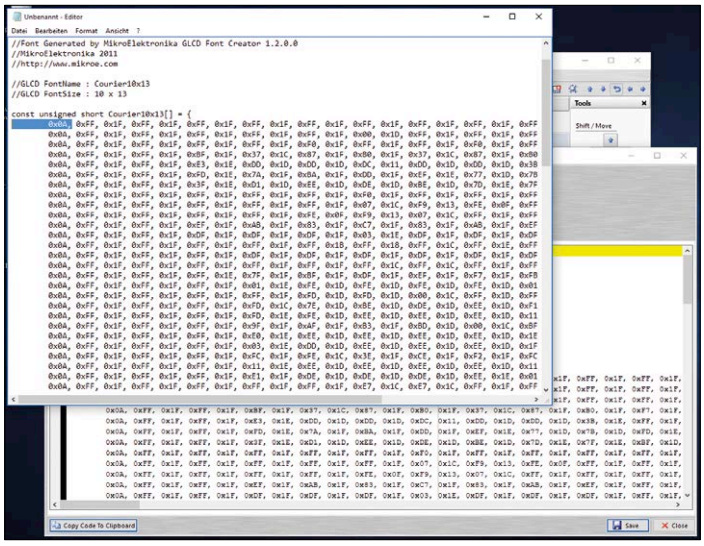


Bild 6. Dieses Byte ist unnötig.

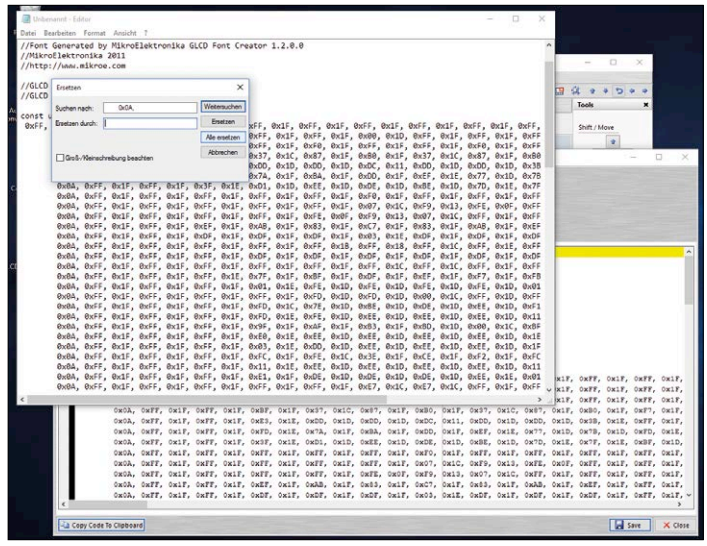


Bild 7. Ist das „Ersetzen durch"-Feld leer, so werden alle Längen-Bytes einfach gelöscht.

wand ist durchaus richtig, er hätte aber eine Steigerung der Hardware-Komplexität verursacht.

Als nächste Aufgabe haben wir das Rendern der einzelnen Zeichen vor uns. Zu Test- und Entwicklungszwecken schreiben wir uns eine Demo-Routine. Sie arbeitet den im Font enthaltenen Zeichenbereich von 32 bis 127 über eine For-Schleife ab. Jedes Zeichen wird zudem 120 Mal hintereinander gezeichnet. Unterbleibt dies, so nimmt das Auge die Darstellung als verwaschen dar (**Listing 5**).
Ein mit GLCD Font Creator erstelltes Font enthält – von Haus aus – die Zeichen von 32 bis 127. Da die ersten Zeichen nicht Teil des Arrays sind, lässt sich der Offset eines bestimmten Zeichens durch Subtraktion von 32 ermitteln.

Damit können wir uns dem eigentlichen Rendern eines Zeichens zuwenden (siehe **Listing 6**).
Nach der Berechnung des Offsets des ersten Bytes des Zeichens durchläuft die Schleife alle 10 Spalten der Zeichendarstellung nacheinander. Zuerst wird ein Byte `whichToWrite` gewonnen, das im Schaltplan die Spalten auf der linken Seite ansteuert (Anmerkung: Die Zeichen werden um 90° verdreht dargestellt). Für jede Spalte der Zeichendarstellung wird im Schaltplan fortlaufend eine LED-Zeile aktiviert.

Das eigentliche Herausschreiben der Datenbits auf die Spaltenleitungen erfolgt über einen kleinen Trick: `digitalWrite` akzeptiert alle Werte größer 0 als true. Wir ANDen den Wert von `whichToWrite` für jedes Bit mit einer Maske, die aus der Bitadresse durch Rechtsschieben ermittelt wird.

Listing 3. Hardware-Fehler ausgebügelt.

```
unsigned char mapRow(unsigned char row)
{
  if(row<8)
  {
    return row+8;
  }
  else
  {
    return row-8;
  }
}
```

Listing 4. Linken und rechten Display-Teil ansteuern.

```
void pickPartOfDisplay(bool _hi)
{

   if(_hi)
   {
       digitalWrite(4,HIGH);
       digitalWrite(5,LOW);
   }
   else
   {
       digitalWrite(4,LOW);
       digitalWrite(5,HIGH);
   }
}
```

Listing 5. Buchstaben-Satz durchlaufen.

```
void loop() {
  while(1==1)
  for (int offsetC=0;offsetC<127-32;offsetC++)
    for(int dela=0;dela<120;dela++)
    {
//Rendern
    }
  }
}
```

Die Beschreibung des rechten Teils der LED-Matrix erfolgt analog. Die Variable `offset` wird vorher um 1 erhöht. Damit haben wir Zugriff auf ein Byte `whichToWrite`, das im Schaltplan die Spalten auf der rechten Seite ansteuert.

Und los!

Für die Realisierung eines Laufbanners müssen wir Zeichen an beliebiger Position in der LED-Matrix zeichnen können. Bei der Erstellung von Controller-Firmware bietet es sich häufig an, bisher funktionierende Logik in eine Funktion „umzulagern“ und dort weiter zu verarbeiten.

`displayChar` nimmt neben dem Offset eine weitere Ganzzahl entgegen, welche die Verschiebung nach rechts (positiv) oder nach links beschreibt, siehe **Listing 7**.

Das Grundprinzip der Render-Funktion bleibt unverändert bestehen. Die neue Version unterscheidet sich insofern von ihrem Vorgänger, als die errechneten Koordinaten (für die Zeilen im Schaltplan) nun mit einem Offset `_shoveHowMuch` beaufschlagt werden. Führt dieses zu ungültigen Ergebnissen, so inkrementieren wir den Inhalt von `offset` um zwei und starten den nächsten Schleifendurchlauf.

Das Zeichen wird an derselben Stelle 10 Mal geschrieben (deshalb die Schleife ganz außen), um einem Verwaschen vorzubeugen.

In der Hauptschleife `loop` lassen wir nicht nur ein einzelnes Zeichen, sondern gleich eine kleine Begrüßung aus mehreren Zeichen loslaufen. Die Zeichen werden hier gleich mit einem bestimmten Offset auf die Reise geschickt, damit sie nacheinander über die LED-Matrix rollen. Die For-Schleife sorgt dann für den notwendigen Vorlauf (**Listing 8**).

Listing 6. Rendern eines Buchstabens.

```
int offset=widthOfChar*offsetC;

for(int j=0;j<10;j++)
{
  char whichToWrite=Courier10x13[offset];

  activateRow(j);
  pickPartOfDisplay(false);
  for(int i=0;i<8;i++)
  {
    digitalWrite(6+i,(whichToWrite&(1<<i)));
  }
  delayMicroseconds(20);

  offset++;

  whichToWrite=Courier10x13[offset];

  activateRow(j);
  pickPartOfDisplay(true);
  for(int i=0;i<8;i++)
  {
    digitalWrite(6+i,(whichToWrite&(1<<i)));
  }
  delayMicroseconds(20);

  offset++;

}
```

Listing 7. Zeichen laufen lassen.

```
void displayChar(int _offset, int _shoveHowMuch)
{
  for(int dela=0;dela<10;dela++)
  {
    int offset=widthOfChar*_offset;
    for(int j=0;j<10;j++)
    {
      if((j+_shoveHowMuch)>15 || (j+_shoveHowMuch)<0)
      {
        offset++;
        offset++;
        if((j+_shoveHowMuch)>15) break;
                                 //Speed optimization
        continue;
      }

      char whichToWrite=Courier10x13[offset];

      activateRow(j + _shoveHowMuch);

      pickPartOfDisplay(false);
      for(int i=0;i<8;i++)
      {
        digitalWrite(6+i,(whichToWrite&(1<<i)));
      }
      delayMicroseconds(20);

      offset++;

      whichToWrite=Courier10x13[offset];

      activateRow(j + _shoveHowMuch);

      pickPartOfDisplay(true);
      for(int i=0;i<8;i++)
      {
        digitalWrite(6+i,(whichToWrite&(1<<i)));
      }
      delayMicroseconds(20);

      offset++;
    }
  }
}
```

Adaptierung an AVR

Damit können wir auf den Arduino Uno umsteigen. Die Schriftart passt knapp ins RAM, wegen des ansonsten ressourcen-schonenden Aufbaus des Programms können wir die Compilerwarnung ignorieren. Leider ist das Ergebnis wegen des starken Flimmerns nicht wirklich brauchbar. Daher tauschen wir den Inhalt von Loop gegen folgende Sequenz aus – und siehe da, alles wird ruhig:

```
void loop() {
  displayChar('F'-32,0);
  /*for(int i=0;i<90;i++)
  ...
```

Nun sehen wir, dass die eigentliche Zeichendarstellung den Controller nicht überfordert. Die Probleme treten wegen des suboptimalen Aufbaus der Anzeige längerer Zeichenfolgen auf. Als Lösung bietet sich die Optimierung der Darstellungsroutine an. Die neue Version präsentiert sich in **Listing 9**.

Für uns ist hier – neben der Entfernung der Aufrufe von `delayMicroseconds` - die Überprüfung von `shoveHowMuch` interessant. Die neue Version von `displayChar` prüft die Gültigkeit des Zahlenbereichs sofort nach der Aktivierung, und erspart dem Prozessor im Falle eines Falles somit das Abarbeiten der beiden for-Schleifen. Die äußere ist dabei (wegen der enthaltenen Multiplikation) besonders kritisch.

Fazit

Mit der Anzeige des kleinen Grußes ist unsere Arbeit an dieser Stelle fertig: die Ansteuerung der 16x16-Matrix funktioniert. Bei größeren Projekten gibt es an mehreren Stellen Möglichkeiten zur Optimierung. Erstens würde sich die Nutzung eines Prozessors mit mehr GPIO-Ports anbieten, um so mehr Datenworte gleichzeitig auf das Display schreiben zu können.

Änderung Nummer zwei betrifft die Verwaltung der Zeichen. Die hier gezeigte primitive Schleife verbraucht unnötig Rechenleistung, da jedes Zeichen unabhängig von seinem Aufenthaltsort am Bildschirm geprüft wird.

Zu guter Letzt kommt unser Programm nur knapp in Arbeitsspeicher des verwendeten AVR-Prozessors unter. Die Nutzung des PROGMEM-Kommandos würde an dieser Stelle Abhilfe schaffen – weitere Informationen hierzu finden sich in der Arduino-Dokumentation.

Der gesamte Code kann unter [2] heruntergeladen werden. ⏮

(140520)

Weblinks

[1] www.mikroe.com/glcd-font-creator/

[2] www.elektormagazine.de/140520

Listing 8. Ein kleiner Gruß.

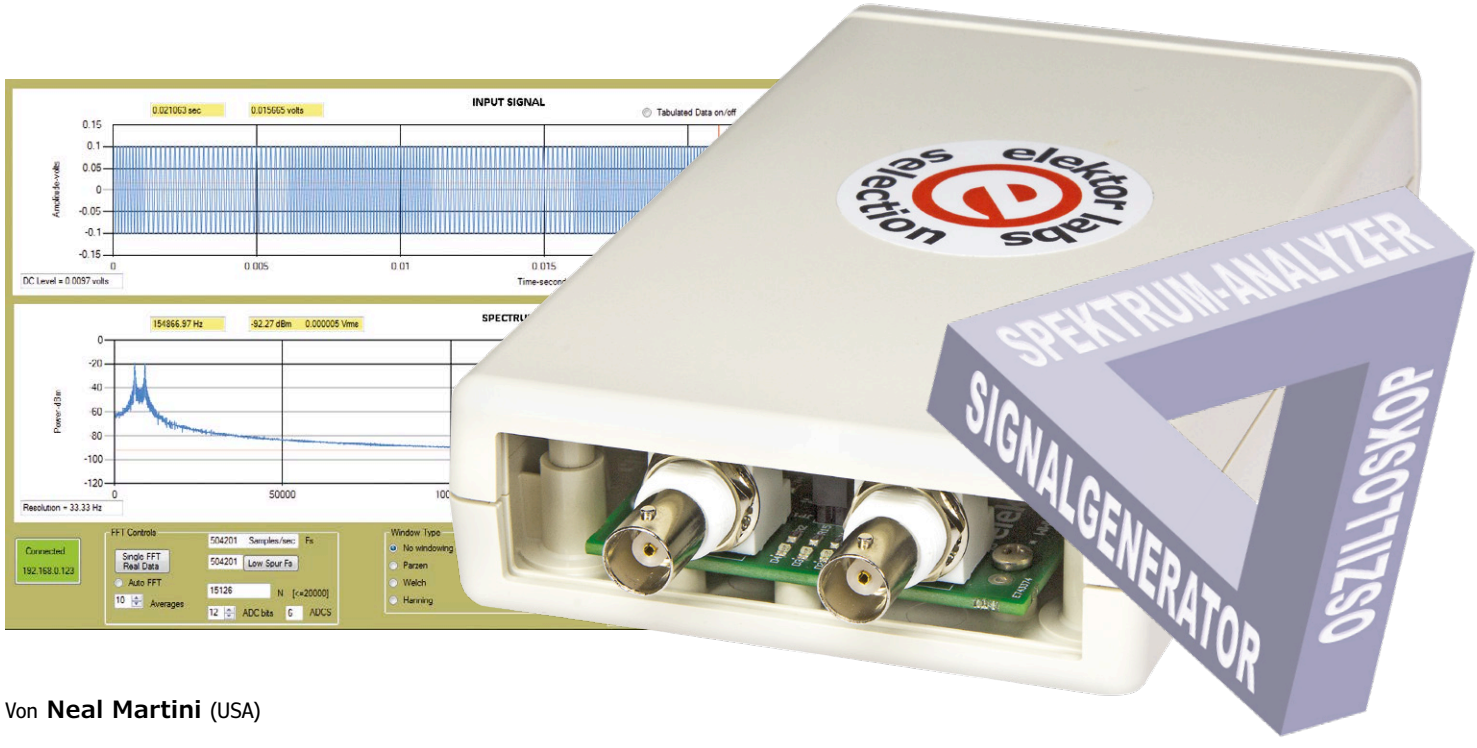
```
void loop() {
  for(int i=0;i<90;i++)
  {

    displayChar(' '-32,0-i);
    displayChar('H'-32,10-i);
    displayChar('A'-32,20-i);
    displayChar('L'-32,30-i);
    displayChar('L'-32,40-i);
    displayChar('O'-32,50-i);
    displayChar('!'-32,60-i);
    displayChar(' '-32,70-i);
    displayChar(' '-32,80-i);
  }
}
```

Listing 9. Verbesserte Routine.

```
void displayChar(int _offset, int _shoveHowMuch)
{
  if(_shoveHowMuch<-10)return;
  if(_shoveHowMuch>16)return;

  for(int dela=0;dela<5;dela++)
  {
    int offset=widthOfChar*_offset;
    for(int j=0;j<10;j++)
    {
      if((j+_shoveHowMuch)>15 || (j+_shoveHowMuch)<0)
      {
        offset++;
        offset++;
        if((j+_shoveHowMuch)>15)break; //Speed
optimization
        continue;
      }
      char whichToWrite=Courier10x13[offset];
      activateRow(j+_shoveHowMuch);
      pickPartOfDisplay(false);
      for(int i=0;i<8;i++)
      {
        digitalWrite(6+i,(whichToWrite&(1<<i)));
      }
      offset++;

      whichToWrite=Courier10x13[offset];
      activateRow(j+_shoveHowMuch);
      pickPartOfDisplay(true);
      for(int i=0;i<8;i++)
      {
        digitalWrite(6+i,(whichToWrite&(1<<i)));
      }

      offset++;
    }
  }
}
```

NCSA – der Network Connected Signal Analyzer (2)

Software & Mathematik



Von **Neal Martini** (USA)

In Folge 1 ging es um die Hardware, welche die Funktionen eines Oszilloskops, eines Spektrum-Analyzers und eines Signalgenerators zu einem einzigen preiswerten System kombiniert. In diesem Artikel dreht sich nun alles um die Software und um die dahinter stehende Mathematik, durch die viele dieser Funktionen erst möglich sind.

Diese zweite Folge geht auf die Software von Mikrocontroller und PC ein. Dabei wird auch die zugrundeliegende Mathematik gestreift. Wenn Sie dies gelesen und verstanden haben, sind Sie schon (fast) ein Experte für digitale Signalverarbeitung.

Technische Daten:

- Kombination von Oszilloskop, Spektrum-Analyzer und Signalgenerator
- Abtastrate bis zu 1 MHz
- Unterstützt Subsampling
- Max. Eingangspegel: 0 dBm (0,225 V_{RMS})
- Empfindlichkeit: -80 dBm (22,5 µVRMS)
- Ethernet-Anschluss
- Open-Source

Kommunikation via Ethernet

Für die Ethernet-Kommunikation stützt sich der Mikrocontroller auf die Treiber in *ioLibrary_BSD*. Diese erledigen die Kommunikation mit dem W5500-Modul [2]. Zur Verwendung dieser Treiber braucht es vier Funktionen, bei denen die SPI-Peripherie des dsPIC33's zur Steuerung des W5500 eingesetzt wird. Nachfolgend findet sich der dazu nötige C-Code:

```
// Declare W5500 driver SPI
   Functions
reg_wizchip_cs_cbfunc(wizchip_
   select, wizchip_deselect);
reg_wizchip_spi_cbfunc(wizchip_
   read, wizchip_write);

//Functions
void  wizchip_select(void)
{
    WIZCS = 0;
}

void  wizchip_deselect(void)
{
    WIZCS = 1;
}
void  wizchip_write(uint8_t wb)
{
    uint8_t dummy;
    SPI1BUF = wb;  // write to
   buffer for TX
    while( !SPI1STATbits.SPIRBF );
   // wait for TX complete
    dummy = SPI1BUF;
}
```

```
uint8_t wizchip_read()
{
    SPI1BUF = 0x00;  // write to
   buffer for TX
    while( !SPI1STATbits.SPIRBF );
   // wait for TX complete
    return SPI1BUF;  // read the
   received values
}
```

Jetzt braucht es nur noch die beiden Funktionen `recv(sn,buf,size)` und `send(sn,buf,size)` der Treiber in *ioLibrary_BSD*, um die Kommunikation zwischen W5500 und PC zu etablieren.

Auf der PC-Seite benötigt man für die Aufnahme einer Verbindung etwas Code in C#. Die bidirektionale Übergabe von Daten und Befehlen ist dann sehr einfach. Wenn die PC-Applikation gestartet wird, checkt sie zunächst, ob eine der vier IP-Adressen im lokalen Netzwerk vorhanden sind. Eine entdeckte Adresse wird dann für die komplette weitere Kommunikation verwendet. Aus diesem Grund ist es wichtig, dass die NCSA-Hardware schon vor dem Start dieser App an das Netzwerk angeschlossen und eingeschaltet ist. Außerdem sollte man dem W5500 vor dem Start der PC-App einige Sekunden für die Initialisierung gönnen.
Die PC-App nutzt die Windows-Klasse *TcpClient* für die Ethernet-Kommunikation. Nachfolgend der Code in C# zur Kommunikation von PC und W5500:

```
var result = client.
   BeginConnect(IPAddress.
   Parse(ipaddress),4000,null,null);
result.AsyncWaitHandle.
   WaitOne(TimeSpan.
   FromSeconds(1)); //timeout if no
   PCB
if (client.Connected)
{
    clientStream = client.
   GetStream();  //get a client
   stream
}
```

Wenn die Verbindung steht, benötigt man nur noch die beiden Funktionen `clientStream.Write(TxBuff,offset,size)` und `clientStream.Read(RxBuff,offset,size)` um Befehle und Daten zwischen PC und W5500 auszutauschen.

Signalgenerator(en)

Genau genommen gibt es zwei getrennte Signalgeneratoren. Einer steckt in der MCU und generiert via DDS (Direkte Digitale Synthese) analoge Signale. Er eignet sich prima zur Genese von Testsignalen für externe Geräte. Man kann dessen Signal aber auch direkt in den Eingang einspeisen und es analysieren. Das analoge Signal ist zwar gut für Experimente mit dem NCSA-System geeignet, doch ist es kein vollständiger Ersatz für einen kommerziellen Funktionsgenerator.
Der zweite Signalgenerator steckt in der PC-Applikation. Mit ihm kann man Arrays mit Zahlen generieren, die den Signalen entsprechen. Diese Signale können in der App selbst angezeigt und auch via FFT analysiert werden. Doch zunächst zu den analogen Signalen der MCU:
Im Blockschaltbild von **Bild 1** sieht man eine PWM-Einheit, gefolgt von einem Tiefpassfilter (LPF). Diese Kombination entspricht einem Digital/Analog-Konverter. Bei konstantem Duty-Cycle der PWM ergibt sich am LPF-Ausgang eine konstante Gleichspannung. Durch entsprechende Änderungen der PWM bestimmt man daher die gewünschte Frequenz und Amplitude des Ausgangssignals des Generators.
Die Frequenz via dsPIC33-PWM ergibt sich zu:

$$f_{sys} / 256 = 120 \times 10^6 / 256 = 468{,}75 \text{ kHz}$$

Damit ergibt sich auch die DDS-Update-Frequenz. Einmal pro PWM-Zyklus wird die Amplitude der gewünschten Wellenform berechnet und dann die PWM so eingestellt, dass diese Amplitude generiert wird. Für den gewünschten Wert des Duty-Cycles der PWM kann man eine Tabelle mit den Werten für einen Zyklus verwenden. Dann wird schlicht der jeweilige Amplitudenwert der Tabelle für die jeweilige Wellenform ausgelesen und zyklisch der PWM übergeben. Hierzu wird ein Phasen-Akkumulator mit 32 bit (`DDSp` im Code) verwendet. Bei jedem DDS-Update-Zyklus wird der Phasen-Akkumulator durch das Phasen-Inkrement (`DDSd` im Code) inkrementiert, das für die gewünschte Frequenz der Wellenform zuständig ist. Das Phasen-Inkrement wird wie folgt berechnet:

$$\text{Phase increment} = 2^{32} \times \text{Frequency-Desired} / \text{DDS update rate}$$

Ein Beispiel: Für eine Sinuswelle mit 10 kHz (gewünschte Frequenz = FrequencyDesired), ergibt sich ein Phasen-Inkrement von 91.625.968. Dieser Wert wird zum Inhalt des Phasen-Akkumulators addiert, und dies mit der DDS-Update-Frequenz. Daher enthält der 32-bit-Phasen-Akkumulator zu jedem Zeitpunkt die Phase der jeweiligen Kurvenform. Eine Lookup-Table mit 2^{32} Einträgen im RAM ist etwas überdimensioniert. Es werden daher nur 8 bit des 32-bittigen Phasen-Akkumulators als Adressen der Tabelle verwendet, die insgesamt eine Periode des Signals umfasst.
Der NCSA-Source-Code erlaubt die Auswahl einer Sinus-, Dreieck- oder Rechteck-Welle und die Einstellung der Frequenz. Wer mag, kann den Code so ändern, dass auch andere Wellenformen erzeugt werden.
Bei der Festlegung der Grenzfrequenz des LPF hinter dem PWM-Ausgang gilt es zu beachten, dass die Frequenz niedrig genug ist, um die PWM auszumitteln und deren

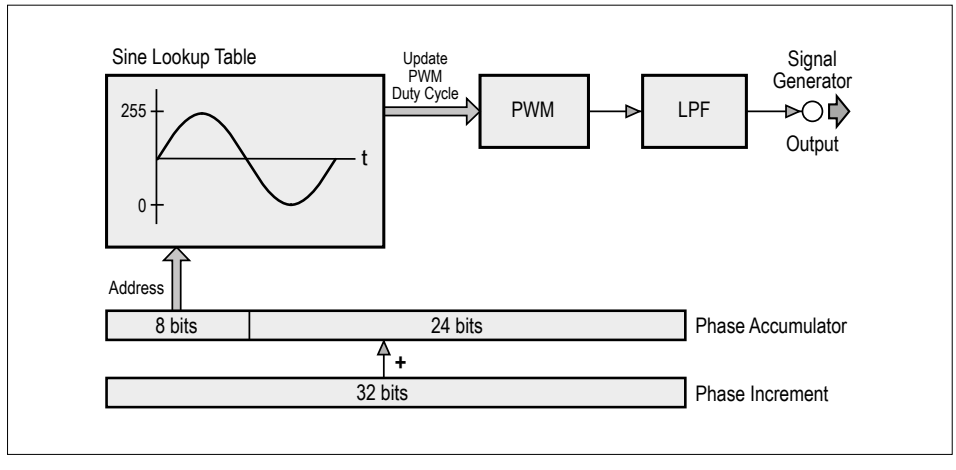


Bild 1. Blockschaltung des DDS-Generators.

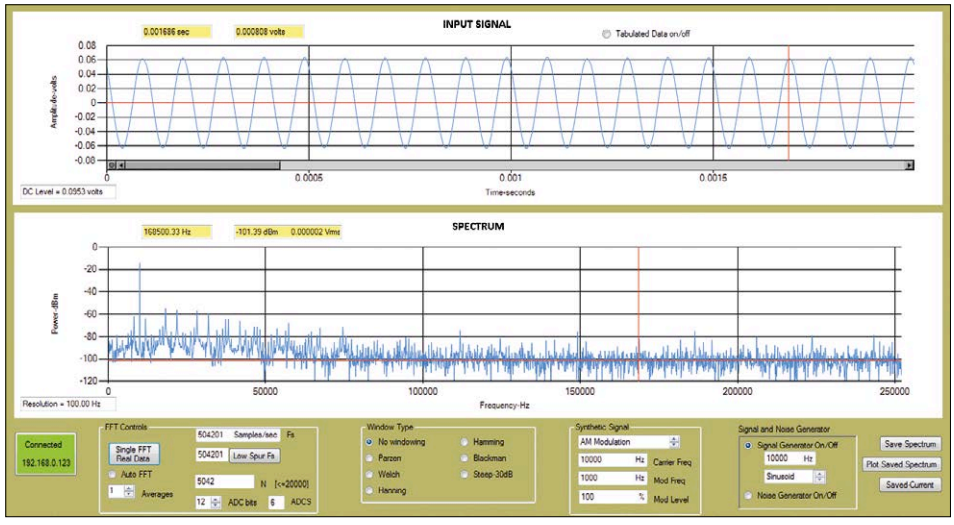


Bild 2. So sieht der Bildschirm aus, wenn der Signalgenerator des NCSA ein Sinussignal mit 10 kHz produziert und dieses wieder an den NCSA-Eingang gelegt wird.

Stufen zu glätten, doch hoch genug, um die höchste Signalfrequenz nicht nennenswert abzuschwächen. Glücklicherweise gibt es einen geeigneten Rechner hierfür im Internet [3]. Wenn man das generierte Rechtecksignal betrachtet, kann man sehen, dass der verwendete LPF seiner Glättungsaufgabe gerecht wird. Bei höheren Frequenzen werden die Kanten des Rechtecksignals verschliffen.

In **Bild 2** (rechts unten) ist der NCSA-Signalgenerator auf ein Sinus-Signal mit 10 kHz eingestellt. Der Ausgang des Signalgenerators wurde hier direkt wieder an den NCSA-Eingang gelegt. Die obere Kurve zeigt die registrierte Sinuskurve. Sie sieht optisch ganz gut aus, nicht wahr? Doch das Spektrum darunter enthüllt die tatsächliche Signalqualität. Links erkennt man den Peak mit der 10-kHz-Grundwelle, und dann folgt ein wildes Gemisch aus Rauschen und Oberwellen etc. Die Verzerrungen haben gerade mal einen Abstand von 40 dB zum eigentlichen Signal. Für einen einfachen Signalgenerator ist das akzeptabel, wenn auch nicht besonders gut.

Der folgende Pseudo-Code demonstriert das Prinzip des DDS-Generators:

```
// Generate a lookup table; one
   cycle of the waveform.
for (i=0; i<256; i++)
{
    Switch (RxBuff[7])  // Selects
   the type of signal to generate.
    {
        case 1: Wave[i] = (uint8_t)
   (128+127*sin((PI*i)/128)+.5);
   //sine
        break;
        case 2: Wave[i] = (i<128) ?
   255 : 0;  // one cycle square
        break;
        case 3: Wave[i] = abs((i
   % 256) - 128);  //one cycle
   triangle
        break;
    }
}

// Calculate the phase increment
DDSd  =  ((FrequencyDesired *
   pow(2,32))/468750.0) + .5;

// Timer1 interrupt service routine
void __attribute__((__interrupt__,
   no_auto_psv)) _T1Interrupt(void)
{
    PDC1 = Wave[(DDSp>>24)]; //do
   table lookup using upper 8 bits
   of DDSp
    DDSp += DDSd;  //increment
   phase accumulator
    IFS0bits.T1IF = 0;  //Clear
   Timer1 interrupt flag
}
```

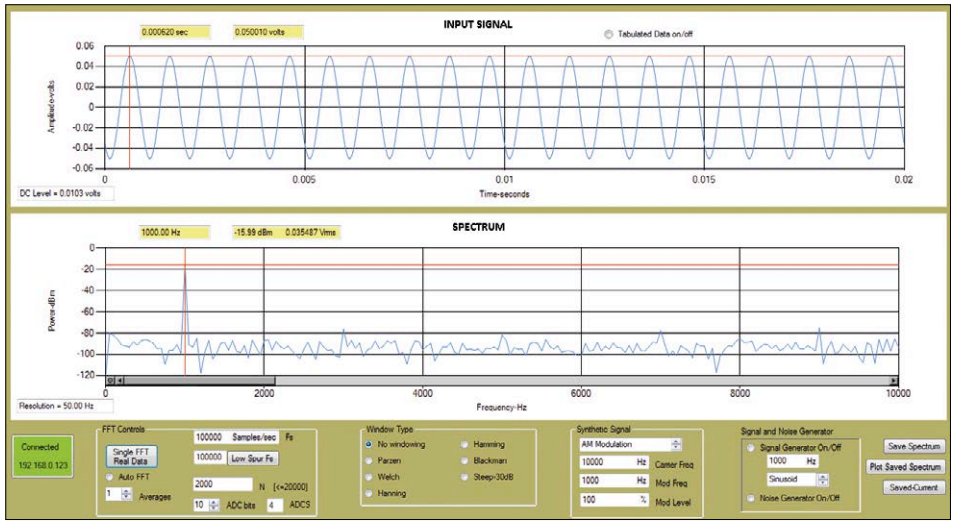


Bild 3. Ein 1-kHz-Sinussignal des NCSA-Signalgenerators. Zeitlicher und spektraler Verlauf werden von der PC-Applikation berechnet und angezeigt

Außerdem hat die PC-App ja auch noch einen synthetischen Signalgenerator eingebaut. Dabei kann man zwischen vier Signaltypen wählen. Sie finden sich in der Funktion `GenerateData` in der Datei *myFFTWstuff.cs*. Auch hier gilt, dass man bei Bedarf natürlich jede gewünschte weitere Signalform in den Code einbauen kann. Man beachte, dass das synthetische Rechtecksignal (case 2) absichtlich kein ideales Rechtecksignal ist. Ich hatte mich dazu entschieden, das Rechtecksignal als Summe von fünf Sinussignalen zu konstruieren. Auf diese Weise kann man schön die zeitlichen Effekte und die Frequenzeffekte auf das resultierende Signal beobachten.

Fourier-Transformation

Nachfolgend wird das Prinzip der Fourier-Transformation beleuchtet. Fourier war ein Pionier darin, mit Hilfe der Aufsummierung von Sinusfunktionen andere Wellenformen zu generieren. Die Fourier-Transformation ist ein Verfahren zur Berechnung der Amplituden und Phasen dieser Sinussignale. Im Prinzip zerlegt die Fourier-Transformation ein Sinussignal in seine Bestandteile. Die DFT (Dis-

krete Fourier-Transformation) ist eine Approximation an die Fourier-Transformation auf der Basis abgetasteter, diskreter Werte. Nachfolgend die Gleichung für den DFT-Algorithmus:

$$Spectrum(m) = \sum_{n=0}^{N-1} \times(n)\cos\left(\frac{2\pi nm}{N}\right) - j\sum_{n=0}^{N-1} \times(n)\sin\left(\frac{2\pi nm}{N}\right)$$

wobei gilt:
x(n) sind die diskreten Daten-Samples,
N ist die Gesamtzahl an Samples,
n ist die Laufvariable der diskreten Samples und
m ist die Variable für die Position im Frequenzraum.

Für jedes m erhält man eine komplexe Zahl der Form (a + jb). Wenn man lange genug drauf schaut, dann sieht man vielleicht, dass die Werte für a und b jedes Werts m der Summe des abgetasteten Signals multipliziert mit dem Cosinus und Sinus diverser Frequenzen entspricht ;-). Je ähnlicher das abgetastete Signal x(n) dem Sinus einer spezifischen Frequenz ist, desto größer die Summe. Umgekehrt gilt: je unähnlicher x(n), desto kleiner die Summe. Der Cosinus/Sinus der niedrigsten Frequenz dieser Gleichung basiert auf einem Sinussignal mit einer Periodendauer, die der Abtastperiode entspricht. Alle folgenden Sinusanteile der DFT sind ganzzahlige Vielfache dieser Frequenz.
Als Beispiel ein konkreter Satz an Werten für N und f_s: Für die Abtastrate gilt f_s = 5000 S/s und es werden N = 1.000 Abtastwerte genommen. Die von diesen Abtastwerten überstrichene Zeitspanne beträgt T = 1.000 / 5.000 = 0,2 s. Der erste Sinuswert der DFT nutzt einen Zyklus in T Sekunden, weshalb sich seine Frequenz zu 1 / 0,2 s = 5 Hz ergibt. Dies ist gleichzeitig die Frequenzauflösung der vorgenommenen DFT. Bei einer DFT über m = 1.000 Werte ergeben sich auch 1.000 komplexe Zahlen. Bei realen Daten entsprechen die ersten N / 2 Werte den zweiten N / 2 Werten. Daher ist nur die Hälfte bzw. nur N / 2 Werte nicht redundant. Folglich ergibt sich der Frequenzbereich des erzeugten Spektrums zu:

(N / 2) × Auflösung = 500 × 5 Hz = 2.500 Hz

Dies entspricht genau f_s / 2, was kein Wunder ist.
Bei Anwendung der Transformation ergeben sich also N / 2 komplexe Zahlen. Was macht man mit denen? In der spektralen Analyse brauchen wir die jeweiligen Pegel der diversen sinusförmigen Komponenten. Dabei entspricht (a + jb) der komplexen Repräsentation von Amplitude und Phase des jeweiligen Sinus. Da die Amplitude an einem 50-Ω-Widerstand am NCSA-Eingang gemessen wird, ergibt sich die Leistung der Sinuskomponente zu:

$$P_{RMS}(m) = (a^2 + b^2) / 50$$

Die Leistungsangaben werden in dBm gewandelt, da sich dies als Standard-Notation für Spektrum-Analyzer durchgesetzt hat. P_{dBm} ist ein Maß für die Leistung bezogen auf 1 mW und berechnet sich wie folgt:

$$P_{dBm} = 10 \log_{10} P_{RMS} / 0{,}001$$

Neben dem Wert in dBm werden oberhalb des Spektrums auch der zugehörige Effektivwert und die Frequenz an der Cursor-Position angezeigt.

Bild 3 illustriert die komplette Verarbeitung von der Eingangsverstärkung bis zur Umwandlung in dBm. Am Eingang liegt ein Sinussignal mit 1 kHz und einer Amplitude von 100 mV_{SS}, das mit f_s = 100 kHz von einem 10-bit-ADC in Blöcken zu N = 2000 S registriert wird. Wie schon beschrieben ergeben sich dadurch Zeitscheiben von T = N / f_s = 20 ms und ein Frequenzbereich von f_s / 2 = 50 kHz (Bild 3 ist gezoomt und zeigt daher nur 0...10 kHz). Nach dem ADC werden die Daten von Gleichspannung befreit und in die Einheit V transformiert, wobei der Einfluss des Verstärkers etc. berücksichtigt wird. Die Spannungswerte werden dann einer DFT unterzogen, was N / 2 = 1.000 komplexe Frequenzwerte der Form (a + jb) ergibt. Da der Eingang eine Impedanz von 50 Ω aufweist, wird die Leistung anhand der beschriebenen Formel berechnet. Wie man schön sieht, ergibt sich an der Cursor-Position der Grundwelle mit 1 kHz wie erwartet ein Pegel von -15,99 dBm = 35,487 mV_{RMS}.
Bis hierher wurde das Akronym DFT einfach so verwendet. Zur Klarheit sei gesagt, dass eine FFT (Fast Fourier Transform) eingesetzt wird, die 1965 von Cooley und Tukey als effektive Variante einer DFT entwickelt wurde. Der FFT-Algorithmus spart viel Rechenzeit, indem er redundante Multiplikationen der DFT-Gleichung vermeidet und ist doch äquivalent zu einer DFT, nur eben schneller.
Die FFTs werden natürlich vom PC erledigt. Das geht zwar auch mit einem Mikrocontroller, jedoch deutlich langsamer. Die verwendeten FFT-Library-Routinen wurden 1999 entwickelt (siehe www.fftw.org). Glücklicherweise gab es im Internet schon eine Version, bei welcher der ursprüngliche C-Code in einer Implementation in C# [4] enthalten war. Hieraus extrahierte ich die Teile, die für die in C# geschriebene PC-App nötig waren und schrieb dafür das passende User-Interface.

FensterIn

Das folgende Beispiel demonstriert, warum für einige DFTs Fenstertechniken benötigt werden. **Bild 4** zeigt das Spektrum eines externen Sinussignals aus exakt 10 Perioden im Abtastzeitraum. **Bild 5** zeigt ein Signal mit etwas mehr als 10 Perioden im gleichen Abtastzeitraum. Der Unterschied im angezeigten Spektrum ist dramatisch. In Bild 5 ist zudem ein

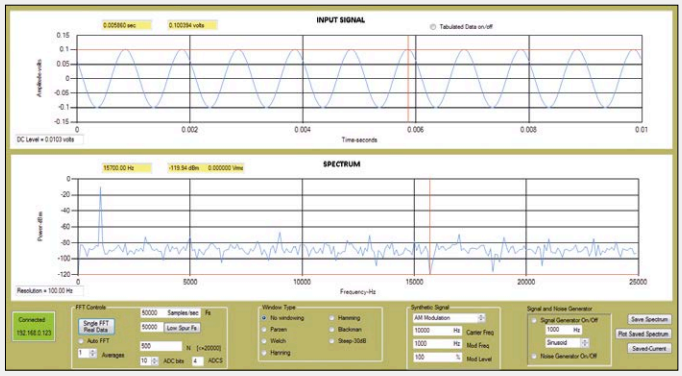


Bild 4. Ein 1-kHz-Sinussignal mit einer ganzzahligen Menge an Perioden im Abtastzeitraum. Das erste und das letzte Datum haben den gleichen Wert.

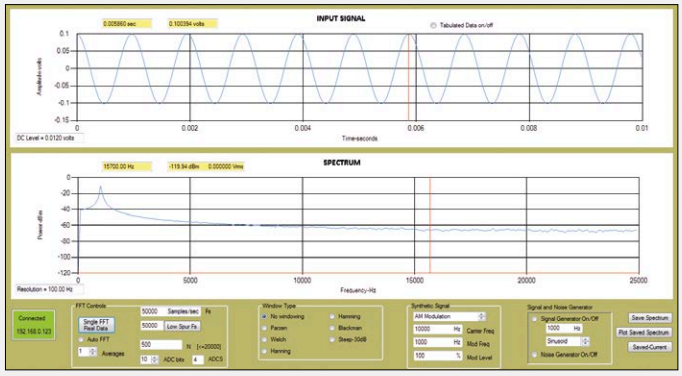


Bild 5. Bei einem (extern eingespeisten) Signal mit 1.020 Hz passen keine vollständigen Perioden in den Abtastzeitraum. Der erste und letzte Abtastwert unterscheiden sich deutlich. Der Peak der Grundwelle im Spektrum hat sich verbreitert und der HF-Pegel ist so angehoben, dass er wichtige Teile des Ergebnisses überdeckt.

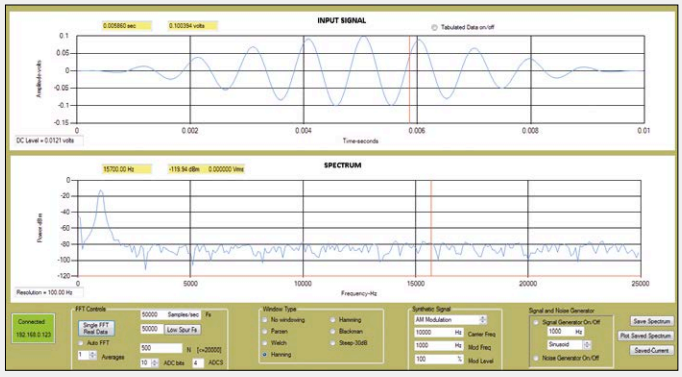


Bild 7. Ein Hanning-Fenster angewendet auf das Signal mit 1.020 Hz von Bild 5. Man sieht sehr deutlich, wie sich das Spektrum verbessert.

hoher Pegel abseits der Grundwelle zu sehen. Auch der Peak der Grundwelle ist verflacht. Die erhöhten Hintergrundpegel werden auch als „leakage“ bezeichnet, und diese können tatsächlich vorhandene Signalanteile überdecken.

Warum ist das so? Da die DFT sich mit den Signalen einer abgetasteten Zeit *T* beschäftigt, geht der Algorithmus quasi davon aus, dass es sich um ein periodisches Signal handelt, bei dem sich diese Zeitabschnitte zyklisch wiederholen. Wenn daher Anfang und Ende des Abschnitts nicht annähernd gleichen Pegel aufweisen, dann sorgt dieser abrupte Unterschied für die artifizielle Produktion hochfrequenter Anteile mit relativ hoher Amplitude, was zu einer verzerrten spektralen Darstellung führt.

Man könnte zwar mit der Abtastfrequenz herumspielen, bis schön vollständige Wellenzüge in den Abschnitt passen, aber manchmal kennt man ja genau diese Signaleigenschaften noch nicht. Hier helfen Fenster.

Um die Unterschiede zwischen Anfang und Ende abzuschwächen, wird der Pegel daher am Anfang und am Ende abgeschwächt. Dies wird erledigt, indem die Daten mit einer Art Hüllkurve multipliziert werden. Genau das ist die Fenster-Funktion. Und die Form der Hüllkurve entspricht dem Fenstertyp. **Bild 6** zeigt ein 500 Datenpunkte fassendes Fenster des Hanning-Typs. **Bild 7** zeigt die Anwendung des Hanning-Fensters

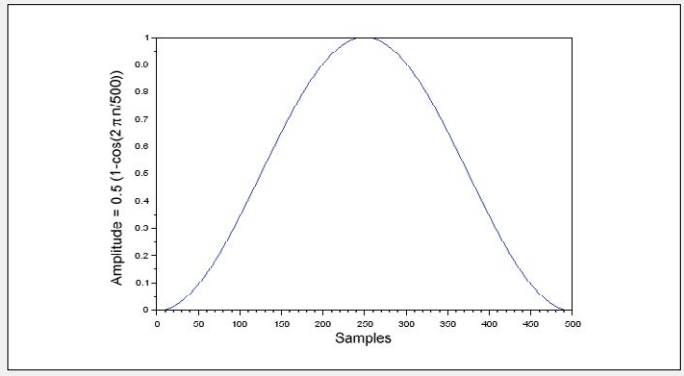


Bild 6. Das Hanning-Fenster.

auf die Daten von Bild 5: Das obere Amplituden/Zeit-Diagramm demonstriert die Abschwächung am Anfang und am Ende. Interessant ist vor allem, wie sich das Spektrum verbessert hat. Die Artefakte durch „leakage“ sind viel geringer. Die Sache ist aber nicht total perfekt repariert, denn die Frequenz-Peaks sind immer noch verbreitert. Die Gesamtleistung bleibt gleich und verteilt sich lediglich etwas über nahe gelegene Frequenzen. Die leicht breiteren Peaks sind üblicherweise nicht so schlimm, und die Vorteile eines präziseren Spektrums durch Fensteranwendung überwiegen.

Insgesamt stehen sechs unterschiedliche Fenstertypen zur Verfügung. Die Fenster unterscheiden sich darin, wie stark sie die Peaks verbreitern und wieviel HF-Störpegel durchgelassen wird. Es gibt eine enorm große Diskussion darüber, welcher Fenster-Typ welche Vor- und Nachteile hat. Man kann das nachlesen. Und mit dem NCSA kann man es auch einfach ausprobieren.

Die Geschwindigkeit des NCSA-Systems wird hauptsächlich von der Blockgröße *N* bestimmt. Ein großes *N* ergibt eine größere spektrale Auflösung und ein kleineres *N* höheres Tempo. Die nachfolgende Tabelle vermittelt einen Eindruck vom Zusammenhang von *N* und der Geschwindigkeit auf meinem Laptop. In der „Update Rate" enthalten ist die Abtastung, die Fenstertechnik, der Datentransfer, die FFT und die Anzeige.

N	Spectrum Update Rate (Hz)
5.000	8,33
10.000	3,96
15.000	2,48
20.000	1,70

Der folgende Code-Schnipsel in C# zeigt den Kern der PC-App. Gezeigt werden der Datentransfer von der NCSA-Hardware und die anschließende Signalverarbeitung:

```
// Receive data over the network
int read = 0, offset = 0, toRead =
   2*N;
while (toRead>0 &&
   (read = clientStream.
   Read(RxBuff,offset,toRead))>0)
{
    toRead -= read;
    offset += read;
}
DCterm = myFunctions.DCtermCalc(N,
   RxBuff, ScaleFactor);    //
   calculates DC term

// Subtracts DC, scales for ADC
   bits, puts data in real part of
   din[]
myFunctions.Filldin(N, DCterm,
   RxBuff, ScaleFactor, AmpGain,
   din);

myFunctions.windowing(N, din,
   WindowType, 1, din);  //window
   data
plotTime.PerformClick();  //plot
   input time series
fftwf.execute(fplan);  //Do FFT

//convert to dBm and average
   spectra (code not shown here)

plotFreq.PerformClick();  // Plot
   frequency domain data (spectrum)
```

Fazit

Dank der Flexibilität durch die vielen einstellbaren Systemvariablen und der Möglichkeit verschiedene Signale zu erzeugen sowie der hochgradigen Interaktivität der Oberfläche kann man mit dem NCSA schnell ein gutes Gefühl für Signalverarbeitung im Allgemeinen und die FFT im Speziellen bekommen. Mit genügend Kreativität kann man die Software nach Belieben um weitere Signalformen der Generatoren erweitern oder aber NCSA an ganz spezielle Einsatzgebiete anpassen. Die gleichzeitige Darstellung des zeitlichen und des Frequenzverhaltens erleichtert das Verständnis der unterschiedlichen Eigenschaften von Signalen. Und außerdem macht das Spielen damit viel Spaß!

(150694)

Web Links

[1] www.elektormagazine.de/150211

[2] http://wizwiki.net/wiki/doku.php?id=products:w5500:driver

[3] http://sim.okawa-denshi.jp/en/PWMtool.php

[4] https://github.com/tszalay/FFTWSharp

Subsampling

Diese Technik ist sehr wichtig, da sie die Anwendbarkeit von digitalen Datenanalysegeräten deutlich erweitert. Ein Beispiel erläutert die Funktion von Subsampling, zu deutsch Unterabtastung, das manchmal auch als „harmonic sampling" oder auch „sub Nyquist sampling" bezeichnet wird.

Wenn man ein 10-kHz-Sinus-Signal eines guten, kommerziellen Funktionsgenerators an den NCSA-Eingang legt und dessen Abtastfrequenz auf 500 kHz einstellt, sieht man genau das, was man erwarten würde. Interessant wird es, wenn man so ein rauscharmes und hochqualitatives Signal beim Generator auf 510 kHz oder 1,01 MHz oder gar 1,51 MHz hochdreht: In jedem Fall sieht man ganz genau so viele Samples und in genau der gleichen Form wie beim 10-kHz-Signal. Der Grund hierfür ist, dass die hochfrequenteren Signale exakt beim gleichen Pegel abgetastet werden, an dem sich zu dieser Zeit auch das 10-kHz-Signal befinden würde. Und man erhält sogar bei den leicht reduzierteren Signalfrequenzen 490 kHz, 990 kHz oder 1,49 MHz (die sogenannten Spiegelfrequenzen) genau die gleiche Kurve wie beim 10-kHz-Signal.

Wenn der Störpegel im Signal (oder Messsystem) steigt, dann wächst auch der Störpegel des Subsample-Spektrums. Vor allem wenn sich bei den Spiegelfrequenzen nennenswerte Signalteile befinden, ergeben sich aber verzerrte Spektren. Doch selbst mit diesen Einschränkungen ist es doch ein kleines Wunder der Technik, dass ein digitales Analysegerät auch Frequenzen oberhalb der Abtastrate f_s vernünftig anzeigen kann, oder? Das Signal eines Mittelwellensenders aus Bild 7 der ersten Folge [1] nutzt genau diesen Ansatz. Der Sender liefert ein amplitudenmoduliertes Sinussignal bei 1,14 MHz und die Abtastfrequenz beträgt 504.201 Hz. Der große Peak im Spektrum liegt bei 131.598 Hz, und das entspricht genau 1.140.000 Hz – 2 × 504.201 Hz.

Die einzige Einschränkung bei der Nutzung dieser Technik mit dem NCSA liegt in der begrenzten Bandbreite des Eingangsverstärkers der Hardware. Wie schon erwähnt beträgt das Bandbreiten-Verstärkungs-Produkt 6 MHz. Wenn man mit einer Abschwächung der Amplitude leben kann, kann man diesen Bereich auch überschreiten (siehe **Bild 8**). Eine coole Sache!

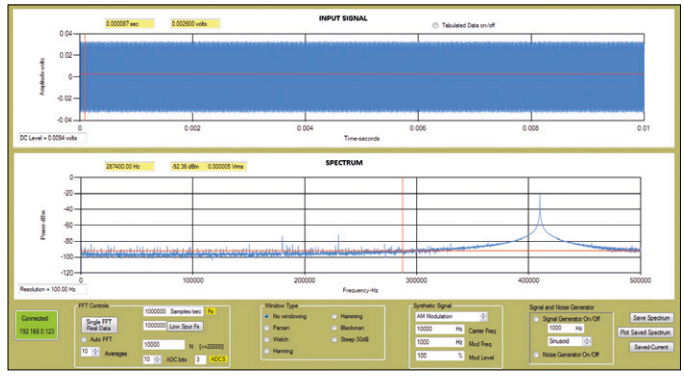
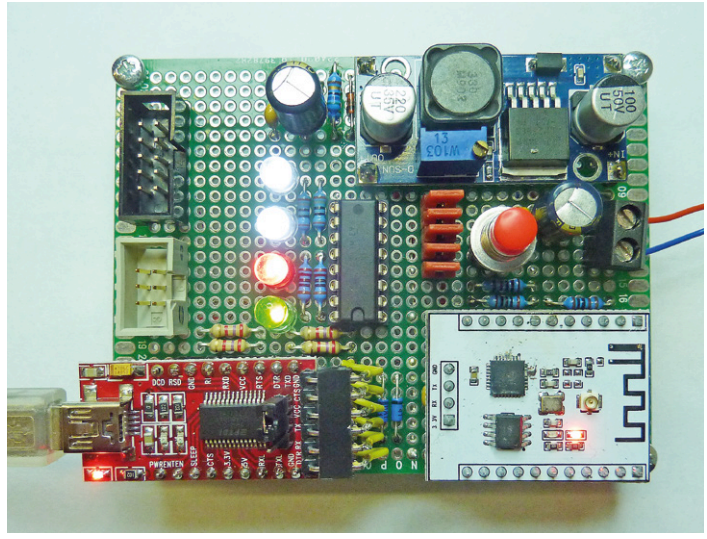


Bild 8. Subsampling ermöglicht die Analyse von Signalen mit deutlich höheren Frequenzen als die maximale Abtastrate des NCSA von 1 MHz. Gezeigt wird hier ein Sinussignal mit immerhin 4,41 MHz. Das Subsampling sorgt dafür, dass der Peak bei 410 kHz liegt.

WLAN kompakt und autonom

Oder: Wie man den ESP8266 auch ohne MCU einsetzen kann



Von **Walter Trojan**

Einen Mikrocontroller ins WLAN zu bringen und damit IoT-fähig zu machen, ist heutzutage kein Problem. Dafür gibt es mittlerweile zahlreiche Zusatzboards oder Chips. Aber kann man diese Kombinationslösungen nicht durch ein kompakteres Design ersetzen, indem man dem WLAN-Chip auch typische MCU-Aufgaben zuordnet?

In der Januar/Februar-Ausgabe 2016 [1] haben wir gezeigt, wie der ESP8266 in Mikrocontroller-Designs eingebettet und mit AT-Befehlen gesteuert werden kann. Aber das Modul kann mehr. Es verfügt nämlich neben seinen dedizierten WLAN-Schaltkreisen über einen leistungsfähigen 32-bit-Prozessor, der mit 80 MHz beziehungsweise 160 MHz getaktet wird. Nach Herstellerangaben wird von dessen Leistung lediglich 20 % für die WLAN-Funktionen verwendet. Da drängt es sich doch auf, die restlichen 80 % für zusätzliche Aktivitäten zu nutzen.
Ein Blick auf das Blockschaltbild (**Bild 1**) zeigt die klare Trennung zwischen analoger Hochfrequenztechnik, einem 2,4-GHz-Transceiver mit Antennenanpassung sowie den notwendigen Taktgeneratoren und der digitalen Steuerung mit der Tensilica L106-MCU. Diese verfügt neben einem RAM mit 32+80 KB über nützliche Peripherie, die über 17 Leitungen von außen zugänglich ist. Abhängig von der Anwendung können diese als GPIOs, UART-, SPI-, I²C- oder PWM-Schnittstellen geschaltet werden. Die Firmware wird in einem externen Flash-Speicher untergebracht, der über eine leistungsfähige SDIO-Schnittstelle (4 Daten-, 2 Steuerleitungen) angesteuert wird. Der Strombedarf, der im Normalbetrieb circa 70 mA und maximal etwa 170 mA beträgt, kann mit den Power Management Units im Sleep-Mode auf ungefähr 10 µA reduziert werden. Basierend auf diesen Eigenschaften lässt sich der ESP8266 autonom in kleinen bis mittleren WLAN-Projekten einsetzen.

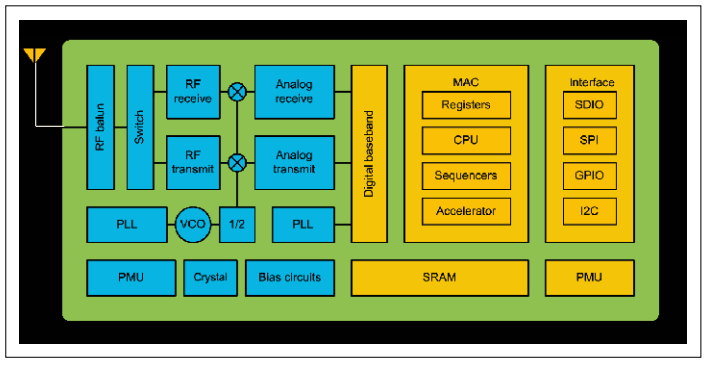


Bild 1. Im Inneren des ESP-Chips

Viele Wege führen zur Firmware

Espressif, der Hersteller des Chips, stellt ein Entwicklungssystem zur Verfügung, das in einer virtuellen Maschine mit Lubuntu-Linux läuft. Da dieses wohl nicht die Lieblingsumgebung vieler User darstellt, hat die sehr aktive Maker-Community für den ESP8266 einige alternative Entwicklungspfade entwickelt. Hier die drei populärsten:

Lua – Skriptsprache aus Südamerika

Lua (portugiesisch für Mond) ist eine imperative und erweiterbare Skriptsprache [2], die plattformunabhängig von einem Interpreter mit geringem Ressourcenbedarf (120 KB) ausgeführt wird. Sie kann auf C-Bibliotheken zugreifen; aus C können wiederum Lua-Routinen aufgerufen werden.
Flashed man den ESP8266 mit dem Lua-Interpreter namens NodeMCU [3], wird der AT-Präprozessor überschrieben. Der Chip nimmt anschließend nur Lua-Instruktionen entgegen und führt sie aus. Es gibt sogar spezielle Boards, die neben einem ESP bereits einen UART/USB-Wandler enthalten und mit dieser Firmware vorgeladen sind. Mit dem ESPlorer [4] steht eine komfortable IDE zur Eingabe und Ausführung von Lua-Skripts bereit. **Bild 2** zeigt einen Screenshot: Im linken Fenster ist ein Lua-Skript, im rechten Fenster der Dialog mit dem ESP abgebildet. An dem abgebildeten „LED-Blink“-Beispiel sieht man die einfa-

che und übersichtliche Lua-Struktur. Nachteilig kann bewertet werden, dass man eine neue Programmiersprache erlernen muss und dass die NodeMCU-Firmware bereits einige Ressourcen des ESP belegt.

Arduino-IDE

Ja, auch die starke Arduino-Fraktion war aktiv und hat den ESP8266 in ihre Arduino-IDE integriert. Die Erweiterung der IDE [5] ist schnell durchgeführt: Unter „Voreinstellungen“ spezifiziert man bei „Zusätzliche Boards Manager URLs“ die Adresse der IDE-Erweiterung [6]. Danach wird vom „Boards Manager“ die Installation des „Generic ESP8266 Module“ angeboten. Nach dessen Installation erscheinen zusätzliche ESP-Module in der Liste der verfügbaren Boards.

Wählen Sie das „Generic ESP8266 Module“. Nun kann mit der Entwicklung der Firmware begonnen werden. Setzt man die jüngste Version der IDE ein, kann es passieren, dass die Bibliotheken der IDE-Erweiterung noch nicht daran angepasst sind. In einem solchen Fall ist die Vorgängerversion zu installieren (zum Beispiel V1.6.5 statt V1.6.6).

Der Ablauf dieser Integration ist sehr gut in einem kostenlosen, rund 300 Seiten starken E-Paper des ESP-Aktivisten Neil Kolban [7] beschrieben. Dieses Werk enthält darüber hinaus viele Grundlagen und Spezifika des ESP8266. Sehr empfehlenswert!

Arduino unterstützt sein WLAN-Shield durch eine leistungsfähige Bibliothek mit über 100 Funktionen. Diese Routinen stehen nun auch für den ESP zur Verfügung. **Bild 3** zeigt einen Screenshot der bekannten Arduino-IDE, bei dem im Hauptfenster der Sketch-Ausschnitt eines Webservers gezeigt wird. Mit schon etwa 100 Befehlen ist ein einfacher Webserver zur Steuerung einiger LEDs realisierbar. Das Browserfenster dazu ist in **Bild 4** zu sehen.

Eclipse und C

Freunde der Entwicklungsumgebung Eclipse können mit einem PlugIn die Sketch-Programmierung in der Arduino-IDE vornehmen (Integration: siehe Neil Kolban), wer allerdings die volle Kontrolle über seine Firmware haben und die Feinheiten des ESP8266-SDKs ausnutzen möchte, der greift zum Profisystem Eclipse und programmiert in C. Eclipse ist eine IDE, die im Jahr 2001 von der Firma IBM quelloffen der Open-Source-Gemeinde zur Verfügung gestellt wurde. Ursprünglich als Java-Entwicklungsplattform vorgesehen, sind mittlerweile angepasste Perspektiven (spezialisierte Ansichten) für zahlreiche Programmiersprachen implementiert.

Für C und C++ stehen die „C Development Tools“ (CDT) bereit. Vorteilhaft ist auch, dass Eclipse unter Linux und Windows läuft und die Weiterentwicklung stetig vorangetrieben wird. Seit 2006 gibt es zehn Versionen. Weit verbreitet ist die Version Luna, die jüngste ist von 2015 und trägt den Namen Mars.

Namhafte Entwicklungsplattformen wie Atollic TrueSTUDIO oder OpenSTM nutzen Eclipse als IDE. Für die ESP-Community hat der Anwender Cherts ein Paket für Windows geschnürt, das aus folgenden Komponenten besteht:

- ESP8266 System Development Kit mit Dokumentation und zahlreichen Beispielen
- Eclipse Mars IDE
- MinGW Paketverwaltung und Compiler
- ein Skript für die automatische Installation zusätzlicher Pakete

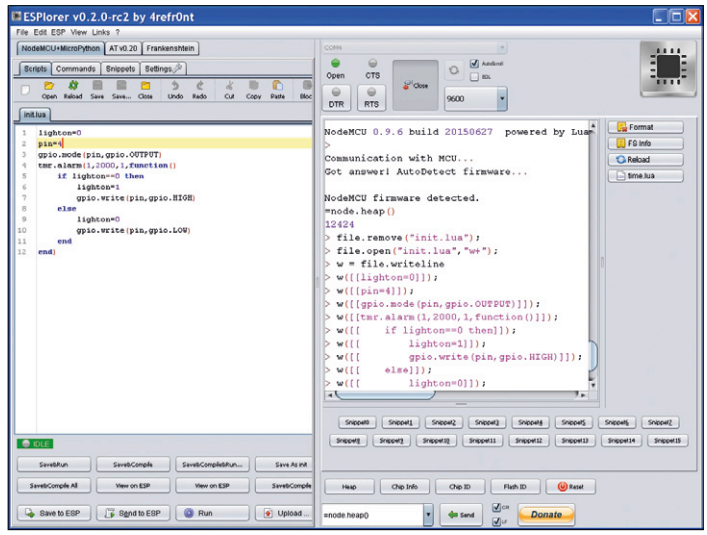


Bild 2. ESPlorer: Im linken Fenster das Lua-Skript, im rechten Fenster der Dialog mit dem ESP.

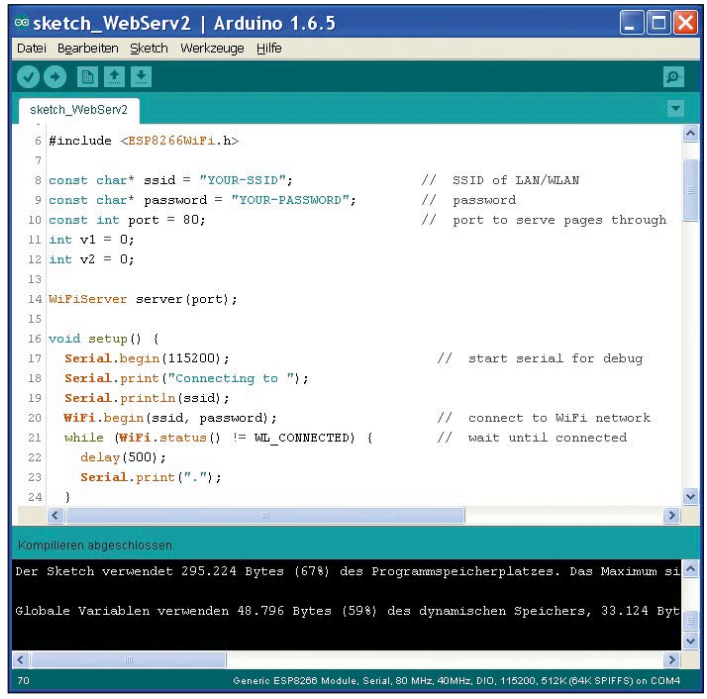


Bild 3. Ein einfacher Webserver in der Arduino-IDE.

Bild 4. So präsentiert sich der „Simple Webserver“.

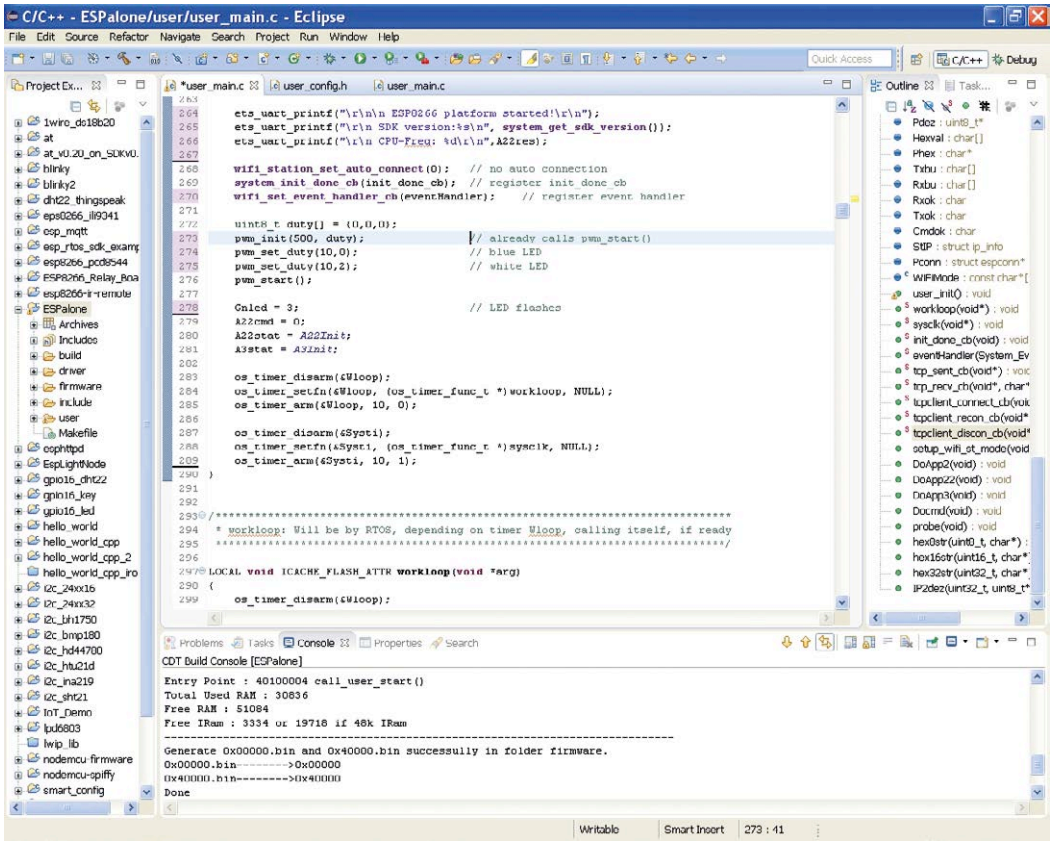


Bild 5. Ein gut gefüllter Eclipse-Bildschirm mit vielen Infos und Zugriffsmöglichkeiten.

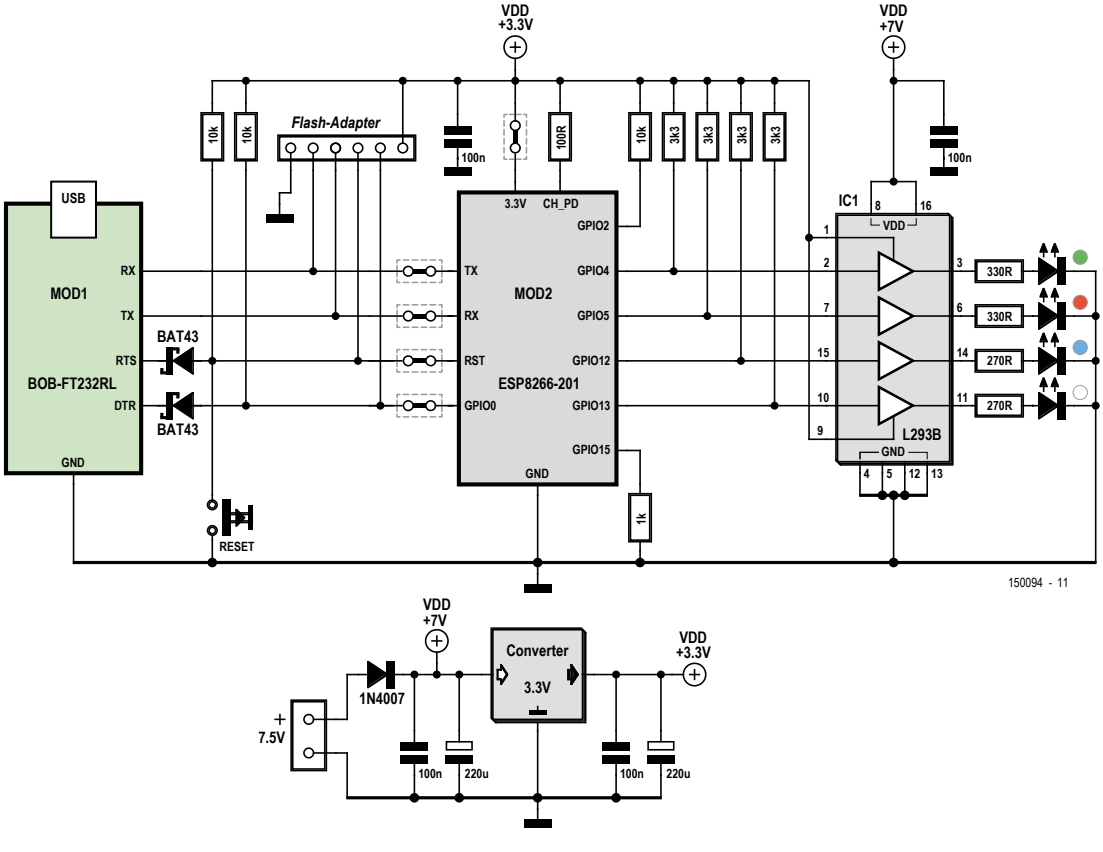


Bild 6. Die Hardware für das „Blink-LEDs“-Projekt.

Anweisungen und Download-Adressen sind unter [8] zu finden. Nach der problemlosen Installation präsentiert sich Eclipse wie in **Bild 5** dargestellt. Links im Projektfenster sind die verfügbaren Beispiele und User-Projekte aufgeführt, im zentralen Hauptfenster der C-Code aufgelistet. Über Outline im rechten Fenster kann man schnell auf die Variablen und Funktionen des Programms zugreifen. Im unteren Fenster werden unter anderem Konsolen-Nachrichten, Fehlermeldungen und Suchergebnisse dargestellt. Zusätzliche Service-Funktionen wie das Flashen des ESP aus Eclipse heraus werden von in den Projekten enthaltenen Make-Files unterstützt.
Da ich Eclipse bereits für die Entwicklung mit STM32-Controllern einsetze, fühle ich mich hier auch bei der ESP-Programmierung wohl und habe damit die weiter unten beschriebene Lösung erstellt.

Jetzt wird es hard!

Nach der Vorstellung der verschiedenen Entwicklungsszenarien wenden wir uns dem konkreten Projekt zu. Die Aufgabenstellung lautet: Steuerung von drei LEDs wie im Januar/Februar-Artikel [1] beschrieben, aber ohne Host-MCU; der ESP8266 mit seinem 32-bit-Prozessor soll alles alleine bewerkstelligen. Die Bedienung soll über das WLAN mit denselben Klienten für PC, iPhone und Android-Tablet erfolgen.
Schauen wir dazu auf die kompakte Schaltung in **Bild 6**. Mittelpunkt ist ein ESP-201, bei dem mehrere GPIOs zugänglich sind. Vier LEDs werden über GPIO 4, 5, 12 und 13 angesteuert. Die grüne LED signalisiert den Kommunikationsstatus:

- Aus: Keine Verbindung mit dem WLAN
- Blinkt: ESP ist mit dem Router verbunden
- An: Ein Client hat sich eingeloggt

Da die maximale Strombelastung eines GPIO-Ports 12 mA beträgt, ist zur Ansteuerung der LEDs ein Treiberbaustein L293B zwischengeschaltet, der auch eine höhere Ausgangsspannung für die Ansteuerung der weißen und blauen LEDs liefert. Ein Treiber ist zwar etwas überdimensioniert, aber ich wollte das Board auch für ein stromintensiveres Motor-Experiment einsetzen. Pulldown-Widerstände an den Eingängen des Treibers sorgen für einen definierten Zustand in der Startphase des ESP. Wichtig für die Selektion der Bootadresse ist die Belegung der GPIOs 0, 2 und 15 bei der Startphase: Mit GPIO 0 und GPIO 2 auf High und GPIO 15 auf Low startet der ESP wunschgemäß vom externen Flashspeicher. Pullup/down-Widerstände stellen die Pegel dazu ein.
Ein USB/RS232-Konverter mit FT232RL-Chip auf dem Board macht das Laden der Firmware komfortabel. Der Transfer der Software zum ESP erfolgt über die TX/RX-Leitungen. Dabei muss zuerst der Flash-Modus eingeschaltet werden, indem GPIO 0 und RST über die FT232-Ausgänge RTS und DTR auf Low gesetzt werden. Zur Entkopplung im Normalbetrieb sind in diese Leitungen die Dioden BAT43 eingefügt. Der Transfer erfolgt mit 115200 Bit/s, die Steuerung übernimmt dabei Eclipse. Der USB/RS232-Konverter wird über den USB-Stecker mit Strom versorgt. Zum manuellen Restart der ESP-Firmware wird die RST-Leitung mit einem Taster auf Low gesetzt.
Wichtig bei der Auswahl des Konverters ist, dass er auf eine **Ausgangsspannung von 3,3 V** eingestellt werden kann. Eine höhere Spannung würde den ESP zuverlässig töten!
Damit auch andere Boards geflasht werden können, habe ich die Programmierleitungen auf einen 6-poligen Wannenstecker herausgeführt und den ESP über Steckbrücken angeschlossen. Bei externer Programmierung werden die Brücken getrennt und das externe Zielboard mit einem kurzen Kabel angeschlossen. Die Stromversorgung ist unkritisch: Eine Eingangsspannung von 7...9 VDC wird über eine Schutzdiode gegen Verpolung dem Treiber (der Betriebsspannungen bis 45 V verträgt) und auch dem Eingang des Schaltreglers zugeführt, der daraus eine feste 3,3-V-Ausgangsspannung erzeugt. Da der ESP im Mittel 70...80 mA, in der Spitze jedoch 170 mA benötigt, sollte die Spannungsquelle entsprechend dimensioniert sein. Stromspitzen werden durch 220-µF-Kondensatoren abgepuffert. Der Aufbau des Boards ist im **Aufmacherbild** zu sehen.

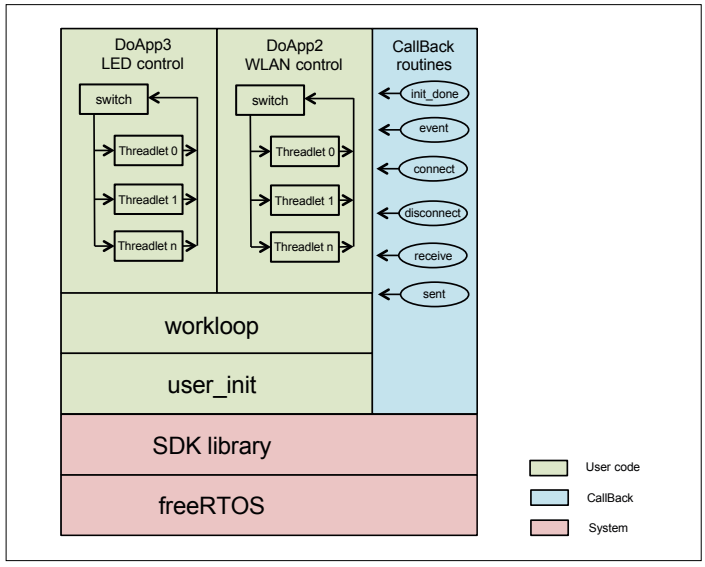


Bild 7. Die Struktur der Firmware.

freeRTOS anstatt „Bare-Metal“

Espressif stellt SDKs mit oder ohne „Real Time Operating System“ zur Verfügung. Dabei sind die Schnittstellen zu den System- und WLAN-Funktionen gleich. Wenn man wie ich bisher nur „Bare-Metal“-Projekte mit unmittelbar auf der MCU-Hardware aufgesetzter Firmware realisiert hat, erfordert der Umstieg auf eine Umgebung mit Echtzeit-Betriebssystem eine gewisse Umstellung. Man zahlt mit Kontrollverlust, erntet aber Komfort. Zur Steuerung des ESP8266 wird das bekannte System freeRTOS eingesetzt. Es koordiniert sowohl die WLAN-Funktionen als auch die Aktivitäten der eigenen Firmware. Bei der Programmierung sind folgende Dinge zu beachten:

- Verzögerungsschleifen mit CPU-Belastung sind zu vermeiden und durch (Soft-)Timer-Konstrukte zu ersetzen.
- Timer-Variable nie lokal, sondern statisch definieren, was eigentlich selbstverständlich ist.
- Eine Anwenderfunktion darf nicht länger als 15 ms dauern, da sonst konkurrierende WLAN-Funktionen behindert werden oder der Watchdog-Timer einen Restart auslöst.

Nach diesen Vorbemerkungen kann es nun ans Eingemachte der Firmware gehen, deren Struktur in **Bild 7** zu sehen ist. Basis ist – wie erwähnt - das Betriebssystem freeRTOS mit den zahlreichen Service-Funktionen des SDKs. Die Gesamtsteuerung übernimmt wie bei allen C-Programmen eine `main`-Funktion,

Listing 1. Endlosschleife unter freeRTOS.

```
LOCAL void ICACHE_FLASH_ATTR workloop(void *arg)
{
  os_timer_disarm(&Wloop); // Schaltet Wloop-Timer aus

  DoApp2(); // State-Machine WiFi
  DoApp3(); // State-Machine LEDs
  os_timer_setfn(&Wloop, (os_timer_func_t *)workloop, NULL);
    // Ruft wookloop wieder auf
  os_timer_arm(&Wloop, 1, 0);// nach einer Pasuse von 1 ms
}
```

Listing 2. Funktion tcpclient_connect_cb, die nach dem Einloggen eines Clients aufgerufen wird.

```
static void ICACHE_FLASH_ATTR tcpclient_connect_cb(void *arg)
{
  Pconn = arg; // Übernahme Argument
  espconn_regist_sentcb(Pconn, tcp_sent_cb);
                                    // Init. tcp_sent_cb
  espconn_regist_recvcb(Pconn, tcp_recv_cb);
                                    // Init. tcp_recv_cb
  ConnState = TCP_CONNECTED; // Signal Status
}
```

die allerdings für den Benutzer nicht sichtbar ist. Sie läuft im Hintergrund und steuert unter anderem alle WLAN-Funktionen. Nach dem Start des ESP8266 wird die Funktion `user-init` einmal aufgerufen. Diese Funktion

- setzt die Parameter für die Peripherie, in diesem Fall für UART und GPIOs,
- startet eine Init-Callback und einen Event-Handler,
- aktiviert eine Systick-Funktion für eigene Soft-Timer (hier 10-ms-Tick),
- und ruft eine eigene Hauptschleife auf, hier `workloop` genannt.

Listing 3. Mit Print-Befehlen wird der Ablauf der Firmware dokumentiert.

```
ESP8266 platform started!  // Start in user_init
SDK version:1.3.0  // verwendete SDK-Version
CPU-Freq: 80 // CPU-Takt 80 MHz
Init done.  // WLAN-Module sind initialisiert
A2ConnStation  // DoApp2 übernimmt und…
ESP8266 in STA mode configured.
               // schaltet in den Station-Mode
Event: EVENT_STAMODE_CONNECTED
               // ESP erfolgreich am Router
Event: EVENT_STAMODE_GOT_IP
               // ESP hat IP-Adresse erhalten
A2CheckIP      // Diese wird abgefragt und…
IP OK 192.168.178.28 // ausgedruckt
A2StartServer  // TCP-Server wird gestartet
               // Warten auf Client
tcpclient_connect_cb // Client hat sich angeschaltet
IP OK 192.168.178.20 // IP-Adresse des Clients
                     // Warten auf Kommando
tcp_recv_cb B2,Lon   // Client sendet Kommando, LED=ein
A2ProcMsg    // Kommando wird verarbeitet
A2SendMsg Data sent, Txbu: ack
             // Acknowledgement für Client
tcp_sent_cb  // Erfolgreich gesendet
             // Warten auf Kommando
tcp_recv_cb B1,Lof   // Client sendet Kommando, LED=aus
A2ProcMsg            // Kommando wird verarbeitet
A2SendMsg Data sent, Txbu: ack
                     // Acknowledgement für Client
tcp_sent_cb          // Erfolgreich gesendet
tcpclient_discon_cb  // Client hat sich ausgeloggt
```

Jetzt kann man das Konstruktionsprinzip gut erkennen. Man hat es mit zahlreichen Callback-Routinen zu tun. Mit der Init-Callback wird gemeldet, wenn der ESP einsatzbereit ist, also alle WLAN-Einheiten korrekt arbeiten. Auch der Event-Handler ist eine Callback-Routine, die unter anderem folgende hier nützliche Ereignisse meldet:

- EVENT_STAMODE_CONNECTED : ESP ist als Station mit dem Router verbunden
- EVENT_STAMODE_DISCONNECTED : Verbindung mit dem Router getrennt
- EVENT_STAMODE_GOT_IP : ESP hat vom Router eine IP-Adresse erhalten

In diesen, wie auch in allen anderen noch zu erwähnenden Callback-Routinen wird keine nennenswerte Verarbeitung durchgeführt, sondern lediglich in einer globalen Variable ein entsprechendes Signal gesetzt.
Die Verarbeitung findet in den von der `workloop` aufgerufenen State-Maschinen `DoApp2` und `DoApp3` statt, wobei `DoApp2` die WLAN-spezifischen Aktivitäten übernimmt und `DoApp3` sich um die LED-Ansteuerung kümmert. Die `workloop` besteht hier nicht aus einer klassischen Endlosschleife, sondern nach dem sequentiellen Aufruf der oben angegebenen Apps endet sie, aber nicht ohne sich vorher beim Betriebssystem mit einer kurzen Verzögerung selber wieder aufzurufen. So wird der Ablauf nicht blockiert und die Systemfunktionen erhalten die Steuerung zurück. In **Listing 1** sieht man eine Endlosschleife unter freeRTOS.
Im Laufe der Verarbeitung werden noch weitere, von mir definierte Callback-Routinen initialisiert:

- tcpclient_connect_cb : Ein Client hat sich eingeloggt
- tcpclient_disconnect_cb : Ein Client hat sich ausgeloggt
- tcp_recv_cb : Eine TCP-Message wurde empfangen
- tcp_sent_cb : Eine TCP-Message wurde erfolgreich gesendet

Auch in diesen Funktionen wird lediglich das Ereignis in einer Variablen signalisiert und an DoApp2/3 weitergegeben. **Listing 2** zeigt als Beispiel die Funktion `tcpclient_connect_cb`, die nach dem Einloggen eines Clients aufgerufen wird. Sie aktiviert lediglich die Sende- und Empfangsroutinen und vermeldet das neue Ereignis. Die espconn-Funktionen sind in den mitgelieferten Bibliotheken implementiert.
`DoApp2` und `DoApp3` arbeiten im kooperativen Multithreading. Nach der globalen Initialisierung in `user_int` wird die Hauptschleife `workloop` aktiviert, in der die Threads sequentiell auf-

gerufen werden. Diese sind als Statemachines ausgelegt, in der Code-Sequenzen – hier Threadlets genannt - in Abhängigkeit einer Statusvariablen abgearbeitet werden. Ist das Threadlet erfolgreich abgearbeitet, verändert es die Switch-Variable, so dass beim nächsten Aufruf ein anderes Threadlet gemäß der Aufgabenstellung zur Ausführung kommt.
Mit Print-Befehlen wird der Ablauf der Firmware dokumentiert, siehe **Listing 3**.
Von mir definierte Client-Kommandos werden vom ESP wie in **Tabelle 1** aufgeführt verstanden. In den UIs der Clients befinden sich Buttons und Slider, <value> kann Werte zwischen 0..255 annehmen (0 = aus, 255 = ganz hell). Weitere Einzelheiten können dem Quellprogramm entnommen werden, das Sie als komplette Eclipse-Workspace von der Elektor-Projektseite [10] herunterladen können.
Diese autonome Lösung arbeitet genauso zuverlässig wie die unter [1] beschriebene Kombination mit dem STM32.

Fazit und Ausblick

Mit dem ESP8266 verfügt die Maker-Gemeinde über einen Baustein, mit dem man ohne Host-MCU kleine bis mittlere IoT-Projekte realisieren kann. Und für die Entwicklung der Firmware stehen für (fast) jeden Geschmack passende Werkzeuge zur Verfügung. Aber die Entwicklung geht weiter, der große Bruder des Chips ist bereits angekündigt. Elektor bleibt am Ball und hält Sie auf dem Laufenden! ⏮

(150094)

Tabelle 1. Kommandos und (Re-)Aktionen des ESP.

Betätigung	Kommando vom Client	Aktion
Button 1	B1, Lof	Rote LED aus
Button 2	B2, Lon	Rote LED an
Button 3	B3, Lbl	Rote LED blinkt
Button 4	B4, Lzz	Rote LED blitzt
Slider 1	S1,<value>	Helligkeit der blauen LED gemäß <value>
Slider 2	S2,<value>	Helligkeit der weißen LED gemäß <value>

Weblinks

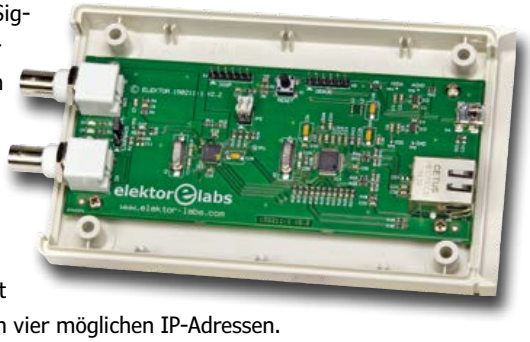
[1] WLAN für Mikrocontroller (Elektor 1-2/2016): www.elektormagazine.de/150093
[2] Lua Skriptsprache: www.lua.org/
[3] NodeMCU: https://github.com/nodemcu/nodemcu-firmware
[4] ESPlorer: http://esp8266.ru/esplorer/
[5] Arduino-IDE: www.arduino.cc/en/Main/Software
[6] Arduino-IDE-Erweiterung: http://arduino.esp8266.com/stable/package_esp8266com_index.json
[7] Kolban´s Book: http://neilkolban.com/tech/esp8266/
[8] Eclipse-basiertes Entwicklungssystem: www.esp8266.com/viewtopic.php?f=9&t=820
[9] FreeRTOS: www.freertos.org/
[10] Elektor-Projektseite: www.elektormagazine.de/150094

Willkommen in Ihrem E-SHOP

ELEKTOR EMPFIEHLT

Network Connected Signal Analyser

Wer gerne ein Oszilloskop, einen einfachen Signalgenerator und einen Spektrum-Analyzer in seinem Elektronik-Labor haben möchte, der ist mit diesem preiswerten Network Connected Signal Analyzer (NCSA) sehr gut bedient. Man kann damit Signale mit Abtastraten bis zu 1 MHz registrieren. Außerdem ist auch noch ein Signalgenerator implementiert. Das digitalisierte Signal wird dabei auf dem Bildschirm eines PCs unter Windows wie bei einem Oszilloskop angezeigt. Mit der Spektrum-Analyzer-Funktion kann man sich die spektralen Eigenschaften des Signals anzeigen lassen. Die Variablen der Fourier-Transformation sind dabei vom Anwender änderbar. Das User-Interface ist sehr grafisch orientiert und die Bedienung ist recht intuitiv. Zur Verwendung muss man den NCSA einfach via häuslichem LAN mit einem Router verbinden. Die PC-Anwendung entdeckt die Hardware automatisch an einer von vier möglichen IP-Adressen.

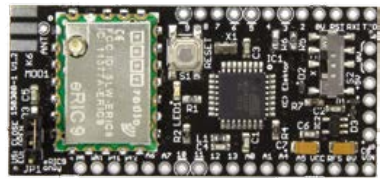


Clemens Valens (Elektor-Labor)

www.elektor.de/150211-91

Elektor-Bestseller

1. eRIC Nitro
   www.elektor.de/eric-nitro



2. Offizieller 7“-Touchscreen für RPi
   www.elektor.de/official-rpi-touch
3. Apps für Elektroniker
   www.elektor.de/apps-fuer-elektroniker
4. Raspberry Pi 2 (Mod. B)
   www.elektor.de/rpi-2
5. Formelsammlung
   www.elektor.de/formelsammlung
6. SmartScope
   www.elektor.de/smartscope
7. Sensoren am Raspberry Pi 2
   www.elektor.de/sensoren-am-rpi-2
8. Red Pitaya (kalibriert)
   www.elektor.de/red-pitaya-calibrated

Digitale Systeme mit FPGAs entwickeln

FPGAs sind Standard-ICs, die dem Anwender die Konfiguration von Hardwarestrukturen ermöglichen. Mit FPGAs ist die Umsetzung digitaler Systeme ab Stückzahl eins machbar. Durch die Verfügbarkeit kostenloser Lizenzen und preiswerter Entwicklungsboards ist die finanzielle Einstiegsschwelle in diese Technik niedrig. Bei der Überwindung der fachlichen Schwelle hilft dieser 6-teilige Kurs.

Mitgliederpreis: 49,00 €

www.elektor.de/digitale-systeme-fpga

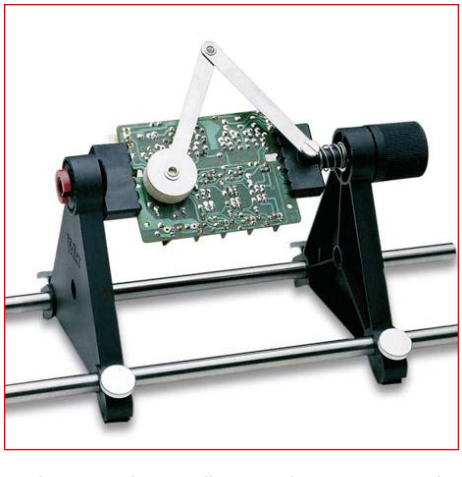
Sensoren am Raspberry Pi 2

Dieses Buch richtet sich an jeden, der seinen Raspberry Pi 2 mit dem aktuellen Windows 10 IoT Core betreiben will. Wie das geht, zeigt der Autor mit dem Entwicklungssystem Visual Studio und Visual Basic als Programmiersprache. Als Einstieg in die Materie wird in diesem Buch auf das auch bei Elektor erhältliche 37 Module umfassende Sensor-Kit zurückgegriffen.

Mitgliederpreis: 29,80 €

www.elektor.de/sensoren-am-rpi-2

Platinenhalter Weller ESF-120 ESD



Gottheiten werden vor allem in Indien gerne mit vielen Armen dargestellt. Dies soll die Fähigkeit symbolisieren, viele Dinge gleichzeitig machen zu können. Menschen haben aber nur zwei Arme und brauchen daher etwas Hilfe, z.B. von diesem cleveren Platinenhalter von Weller. Dieser Rahmen kann in Schritten von je 15 ° um 360 ° gedreht werden. Ein angedrücktes Kissen hält die Bauteile an ihrem Platz, wenn man die Platine zum Löten umdreht.

Mitgliederpreis: 71,96 €

www.elektor.de/weller-esf-120

3D Printing and Autodesk 123D Design

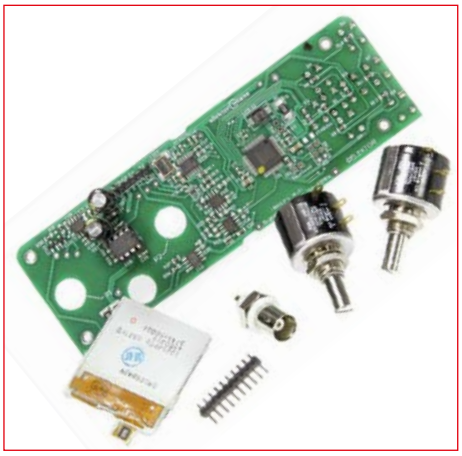
Fachbuch in englischer Sprache

Dank der großen Fortschritte bei Tools und Diensten zum 3D-Druck kann jetzt jeder Dinge herstellen, ohne dafür Mitglied einer großen Organisation sein oder aber spezielle Fähigkeiten aufweisen zu müssen. Dies ist sicher einer der wichtigsten Gründe dafür, warum immer mehr Leute an 3D-Druck-Techniken interessiert sind. Auf der anderen Seite gibt es eine Menge Leute, die 3D-Druck zwar interessiert, die aber nicht wissen, wo sie anfangen sollen. Viele davon haben sogar noch nie direkt einen 3D-Drucker live bei der Arbeit gesehen, sondern kennen diese lediglich über Medien.

Dieses neue Buch versorgt Sie mit dem grundlegenden Wissen und allen notwendigen Informationen zum 3D-Druck, um direkt loslegen zu können.

Mitgliederpreis: 38,20 € (frei Haus)
www.elektor.de/3d-printing

10-MHz-DDS-Funktionsgenerator



Bei diesem Modul handelt es sich um einen kompakten Funktionsgenerator nach dem DDS-Prinzip (Direct Digital Synthesis). Es bietet die Wellenformen Sinus, Dreieck und Rechteck mit einer einstellbaren Amplitude von maximal 15 VSS und einem justierbaren Gleichspannungspegel von ±10 V. Außerdem verfügt das Modul über ein grafisches LCD mit 128x128 Pixel. Der Frequenzbereich beträgt 1 Hz bis 10 MHz.

Mitgliederpreis: 161,96 €

www.elektor.de/150210-91

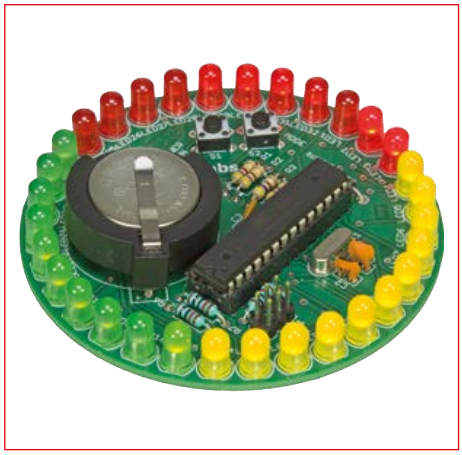
Apps für Elektroniker

Apps für Smartphones gehören mittlerweile vollkommen selbstverständlich zum Alltag und sind in täglich wachsender Zahl in den entsprechenden Stores kostenlos oder für wenig Geld zu haben. Dieses Buch veranschaulicht anhand verschiedener Beispiele, wie man eigene Apps programmieren kann, um damit gekaufte oder selbst gebaute Elektronik auf unterschiedlichen Wegen anzusprechen.

Mitgliederpreis: 34,80 €

www.elektor.de/apps-fuer-elektroniker

Programmierbarer LED-Ring



Wenn die Tage kurz und die dunklen Stunden lang sind, ist die Saison der Licht- und Leuchtobjekte gekommen. Der programmierbare LED-Ring passt in diese Jahreszeit, er ist eine Zierde für die Fensterscheiben der Wohnung oder auch eine stilvolle Dekoration für den Garten oder Balkon.

Mitgliederpreis: 26,06 €

www.elektor.de/150455-71

REVIEW DES MONATS

Von **Ernie Woollard**

Ich bin ein Elektronik-Ingenieur und kürzlich in Rente gegangen. Als solcher war ich auf der Suche nach einem Buch, das alles Wissenswerte zu Raspberry Pi systematisch vermittelt, damit ich bald mit Linux und Python loslegen konnte.

Dieses Buch erfüllte meine Ansprüche perfekt. Es kümmert sich um RPi-Hardware, die Linux-Kommandozeile, GUI, Python, Hardware-Interfaces und enthält eine Auswahl an Beispiel-Projekten. Die Beispiele ermöglichen tiefe Einblicke sowohl in die Hardware- als auch die Software-Aspekte von Projekten. Es gibt auch ein Kapitel zu Programmiertechniken mit dem Titel „Program Description Language“, der eine gute Ausgangsbasis für zuverlässigen und effizienten Code bietet.

Alle Kapitel sind gut geschrieben und die verwendeten Konzepte sind sauber anhand von Code-Beispielen erklärt, was das Erläuterte gut verständlich macht.

Die Kapitel zur Programmierung von „Networking“ und „Systems“ in Python sind besonders interessant und informativ, da hier die wichtigsten Eigenschaften fortgeschrittener Programmierung klarwerden.

Das Buch eignet sich für alle, die schon ein grundlegendes Wissen zum RPi haben und einen RPi in anspruchsvolleren Projekten einsetzen wollen, die über die Basics hinausgehen.

Mehr Infos finden Sie unter www.elektor.de/rpi-advanced-programming

Wenn Sie uns einen ähnlichen Bericht über ein Elektor-Produkt schicken, der in Elektor veröffentlicht wird, erhalten Sie als Dankeschön einen 100-EUR-Gutscheincode, den Sie im Elektor-Shop einlösen können.

Mehr Infos gibt's unter **www.elektor.de/review**

Das neue „Formula AllCode“ ist ein kompletter Roboterkurs, der sich sowohl für fortgeschrittene Robotik-Anwender als auch für Einsteiger eignet. Die Spezifikationen des Roboterfahrzeugs sind beeindruckend: Ausgestattet mit Bluetooth kann er von der großen Rechenleistung externer Plattformen wie Android und OS X/iOS oder auch Raspberry Pi und Arduino profitieren.

Das System ist mit praktisch allen Lösungen programmierbar – inklusive Flowcode 6, MATLAB, LabVIEW, Python, App Inventor, Visual Basic/C#/C++ und mehr.

Mitgliederpreis: 224,10 €

www.elektor.de/formula-allcode

Formula AllCode Robot Buggy

3D Printing and Autodesk 123D Design

Fachbuch in englischer Sprache

Elektor-Shop

Das gesamte Elektor-Sortiment finden Sie auf www.elektor.de.

Formelsammlung

Dieses Buch ist ein Nachschlagewerk mit praxisorientierten Fakten – kein Lehrbuch mit ausführlichen Erklärungen. Der Autor hat auch für komplexe Vorgänge praktische kurze Erklärungen, Näherungsformeln und Rechenbeispiele entwickelt, ohne die Darstellungen zu simplifizieren. Die logische Gliederung in zehn Kapitel vereinfacht das Nachschlagen und Aufsuchen der gewünschten Themen.

Mitgliederpreis: 29,80 €

www.elektor.de/formelsammlung

Internet of Things

Das Internet of Things (Internet der Dinge) ist eine unumkehrbare Entwicklung. Wir möchten gerne alles im Haus mit unserem Smartphone oder Tablet erledigen – von Facebook bis Fernsehen, Lampen steuern oder die Heizungstemperatur einstellen. In diesem Buch stellen wir 35 interessante und nützliche Projekte vor, die demonstrieren, wie Sie selbst ein Internet-of-Things-System anlegen können.

Mitgliederpreis: 39,80 €

www.elektor.de/iot-buch

Maker Kit Internet of Things

Das Internet der Dinge verbindet die reale Welt mit der digitalen Welt. Mit diesem Lernpaket können Sie selbst Projekte für das Internet der Dinge umsetzen. Mit dem beiliegenden IoT-Board (Einzelpreis: 29,95 €) legen Sie direkt los und steuern die Hardware über das Internet. Sie lernen alles, was Sie für den Einstieg benötigen: Schaltungsaufbau sowie Netzwerk- und HTML-Programmierung.

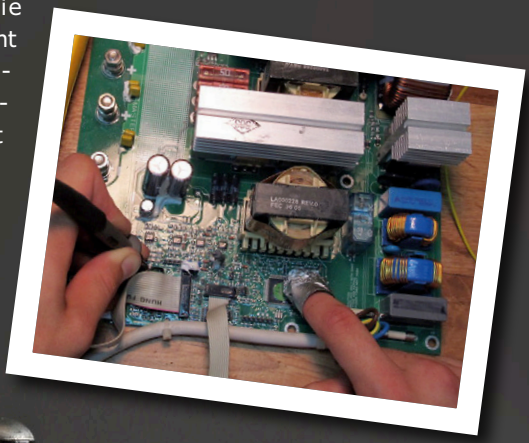
Mitgliederpreis: 71,96 €

www.elektor.de/franzis-maker-kit-iot

Willkommen bei SHARE

Von **Thijs Beckers** (Elektor NL)

Echt praktisch: Alufolie um den Zeigefinger!



In SHARE finden Sie wie gewohnt eine kleine Übersicht von bemerkenswerten Projekten, die auf unsere Elektor-Labs-Webseite gestellt wurden (www.elektor-labs.com). Das schöne Projekt *Vakuum-Fluoreszenz Displays* von Autor „MiBu“ zeigt die Wiederverwendung eines Displays aus einem alten Küchenradio. Noch mehr Recycling: Ein sehr praktisches Projekt stellt aus Teilen einer Satellitenschüssel eine mechanische Positionierungseinrichtung beispielsweise für Kameras oder Antennen her.
In der Rubrik „Aus dem Labor“ geben wir einen Tipp zu LED-Filamenten, der sehr praktisch für das Projekt LEDitron ist, das wir in diesem Heft beschreiben. Wenn Sie sich an das Projekt LEDitron wagen wollen, sollten Sie diesen Tipp nicht verpassen!
Praktische Elektroniktipps finden sich auch in dem kleinen Artikel von Harry, er war wieder im Netz auf der Suche nach sehens- und lesenswerten Sites. Den Tipp mit der Alufolie um den Zeigefinger finde ich persönlich am interessantesten...
Zum Abschluss kommen mit dem Hexadoku natürlich auch die Rätselfreunde wieder auf ihre Kosten. Viel Spaß beim Lesen und beim Lösen!

Verbesserter Platino

Die neue Version des Multifunktions-Controller-Boards wird gut angenommen. Sie erinnern sich an die letzte Ausgabe? Wir haben den zuerst im Oktober 2011 vorgestellten Platino in einigen Punkten verbessert. Es wurden unter anderem eine 3,3-V-Stabilisierung hinzugefügt, ein zweiter serieller Port, eine Reset-Schaltung und mehr übersichtliche Beschriftungen. Manche Leser setzen die neue Platine natürlich auch ein, um bereits veröffentlichte Platino-Projekte nachzubauen, wie zum Beispiel den Platino-Funktionsgenerator. Einen nützlichen Hinweis dazu finden Sie unter www.elektormagazine.de/150555.

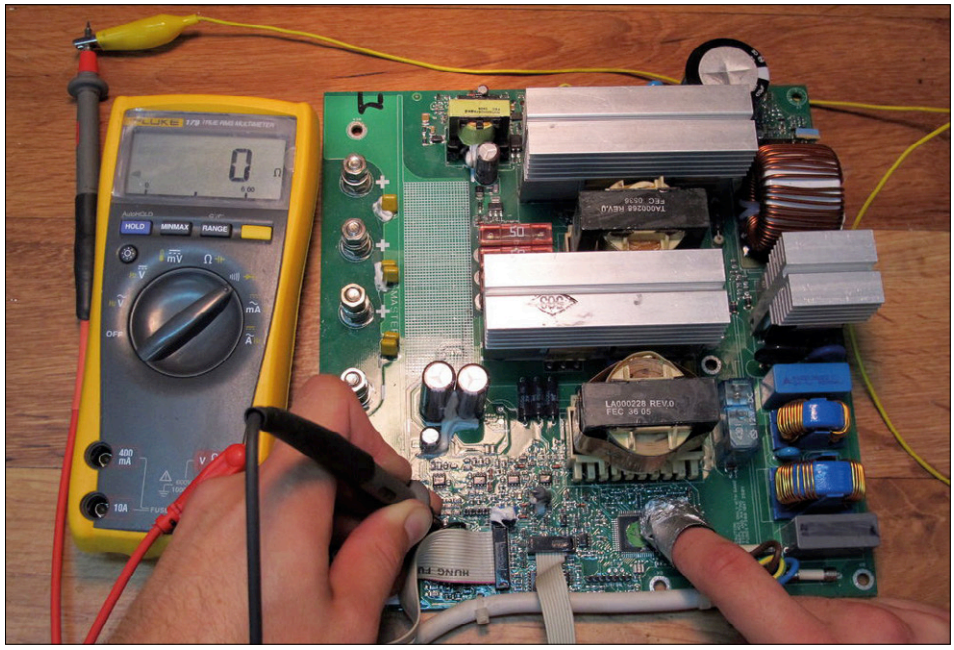
(150744)

Know-How aus dem Netz

Für Labor und Service

Von **Harry Baggen** (Elektor-Labor)

Ohne Zweifel, auch bei Elektor sind intelligente Köpfe am Werk, die kreative Lösungen für alltägliche oder auch eher seltene Aufgaben entwickeln. Doch das weltweite Netz überrascht oft mit klugen und originellen Ideen, auf die der Mensch erst einmal kommen muss. Wir stellen eine kleine Auswahl vor.



Wohl jeder Elektroniker hat sich im Lauf der Jahre einige individuelle Rezepte zurechtgelegt, auf die er bei seinen beruflichen oder privaten Aktivitäten zurückgreift. Ein zielgerichteter Streifzug durch das Web fördert manches fast schon geniale „Gewusst Wie“ zutage, das die Autoren dort der Welt offenlegen.

Zu den leidigen Problemfällen gehört das Reparieren elektronischer Geräte, für die keine technischen Unterlagen verfügbar sind. Übrig bleibt dann nur das zeitraubende Verfolgen von Leiterbahnen auf der Platine. Muss beispielsweise ergründet werden, wohin eine Leiterbahn vom Anschluss eines bestimmten Bauteils führt, lässt sich dies mit einem Durchgangsprüfer (Multimeter) bewerkstelligen. Doch es kann viel Zeit und Geduld kosten, bis mit zwei Messfühlern der gesuchte Platinenpunkt gefunden ist. Diese investigative Arbeit lässt sich verkürzen, indem ein Finger mit einem Stück Alufolie umwickelt wird. Die Folie wird über eine Klemme und eine Leitung mit einem Pol des Messgeräts verbunden. Der andere Pol muss natürlich mit dem Ausgangspunkt der Leiterbahn verbunden sein (siehe [1]). Wird die Platine mit dem umwickelten Finger abgetastet (kräftig drücken!), werden eine ganze Anzahl Lötpunkte und Durchkontaktierungen gleichzeitig erfasst. Der lokale Bereich der Platinenfläche, in dem der gesuchte Punkt liegt, ist schnell eingegrenzt. Jetzt ist es nicht mehr schwierig, den gesuchten Lötpunkt mit der Prüfspitze zu finden.

Einen kleinen, aber nützlichen Trick, die Durchlassspannung einer Diode zu senken, fanden wir bei Daycounter [2]. Soll beispielsweise ein Gerät, das mit 5 V arbeitet und an einem 6-V-Akku betrieben wird, durch eine Diode gegen Verpolung geschützt werden, darf der Spannungsabfall an der Diode nur gering sein. Verglichen mit einer einfachen Siliziumdiode liegt die Durchlassspannung einer Schottky-Diode niedriger. Doch auch hier können bei Strömen von 100...200 mA schnell 1 V oder mehr abfallen. Dem lässt sich entgegen wirken, indem zwei oder drei Dioden parallel geschaltet werden. Diese simple Methode drückt den Spannungsabfall nicht selten unter 0,5 V. Zwar existieren auch elegantere Methoden wie der Einsatz eines FETs, doch das Parallelschalten mehrerer Dioden führt zum gleichen Ziel.

Auf der Seite „Electronics for Bharat“ [3] sind zahlreiche Tipps und Tricks für den Einsatz von Steckplatinen versammelt. Viele Ratschläge dürften dem Praktiker bereits bekannt sein, doch manches ist durchaus mit dem legendären „Ei des Columbus“ vergleichbar. Dazu zählen wir die vorgeschlagene Methode zur Einbindung einer Knopfzelle CR2032 in die Schaltung: Eine Stiftkontaktleiste mit zweimal zwei Kontakten klemmt die Knopfzelle so ein, dass sie kontaktsicher auf der Steckplatine montiert werden kann. Ebenso nützlich ist die vorgeschlagene Methode, mit einer siebenpoligen Stiftkontaktleiste eine SD-Speicherkarte auf der Steckplatine zu platzieren.

Zu einer ganz anderen Kategorie von Tipps gehört das Prozedere, originale Chips der Typen RT232R oder RT232RL des Herstellers FTDI von minderwertigen Nachahmungen zu unterscheiden. Wir haben diesen Tipp auf der Website von Starlino Electronics [4] gefunden. In den letzten Jahren war der Markt von solchen Plagiaten überschwemmt worden. Die Nachahmungen können gravierende Probleme mit den Geräten verursachen. ⏮

(150749)gd

Weblinks

[1] www.instructables.com/id/How-To-Quickly-Find-And-Trace-PCB-Tracks-1/

[2] www.daycounter.com/LabBook/Electronics-Tips-Tricks.phtml

[3] http://m8051.blogspot.de/2012/08/updated-bread-board-tips-and-tricks.html

[4] www.starlino.com/ftdi-chip-real-of-fake-how-to-spot-a-fake-rt232r-rt232rl-and-others.html

Sensible LED-Filamente

Dünnes Glas, wie leicht bricht das!

Von **Harry Baggen** (Elektor-Labor)

LED-Lampen gehören längst zum Alltag, sie sind in vielen Formen und Formaten auf dem Markt. Die äußere Gestalt vieler Typen ist an die konventionellen Glühlampen angelehnt. Leider lässt die Rundstrahlcharakteristik mancher preiswerter LED-Lampen einige Wünsche offen. Die Hersteller bemühen sich, diesen Schönheitsfehler durch geschicktes Anordnen der LEDs oder Einbau von Reflektoren in den Kolben zu korrigieren. Die Lampenform gleicht dann zwar der altvertrauten Glühlampe, doch das anheimelnde Licht eines sichtbar hinter dem Glas glimmenden Glühfadens erreichen sie nicht.

Vor einiger Zeit ist ein neuer Typ der LED-Lampe auf dem Markt erschienen, die LED-Filament-Lampe. Darin sind anstelle einzelner LED-Chips so genannte LED-Filamente verbaut. Im Prinzip sind dies gläserne Träger, auf denen eine größere Anzahl kleiner LED-Chips aufgebracht ist. Die Durchlassspannung der in Reihe geschalteten LEDs liegt so hoch, dass mehrere in Reihe geschaltete Filamente über einen Gleichrichter unmittelbar an die Netzspannung gelegt werden können. Vom Erscheinungsbild her ähneln die Filamente den Glühfäden, die Rundstrahlcharakteristik ist befriedigend bis gut. Bei der Produktion der Fila-

Weblink

www.ledinside.com/knowledge/2015/2/the_next_generation_of_led_filament_bulbs

mente wird eine Chip-on-Glass-Technologie angewendet, die im Wesentlichen darin besteht, dass das Halbleitersubstrat der LED-Chips unmittelbar auf den Glasträger aufgebracht wird. Der Träger erhält ein Leuchtstoff-Coating, wobei die Leuchtstoff-Zusammensetzung die spektrale Verteilung des Lichts bestimmt. An den Trägerenden werden metallische Elektroden für die Verbindung mit der Stromquelle angebracht.

LED-Filamente sind bei chinesischen Herstellern auch einzeln, also ohne Lampe erhältlich. Das brachte uns auf die Idee, sie nicht für die Beleuchtung, sondern für andere Zwecke zu verwenden. Wir wollten ein überdimensionales Sieben-Segment-Display konstruieren, was dann zu dem „LEDitron“ an anderer Stelle in dieser Elektor-Ausgabe führte. Kurzerhand hatten wir einen Vorrat an LED-Filamenten in China bestellt, doch als das Paket ankam, gab es eine böse Überraschung: Offensichtlich waren die gläsernen Träger extrem empfindlich gegen mechanische Schwingungen, so dass mehr als die Hälfte die weite Reise trotz sachgemäßer Verpackung nicht überstanden hatte.

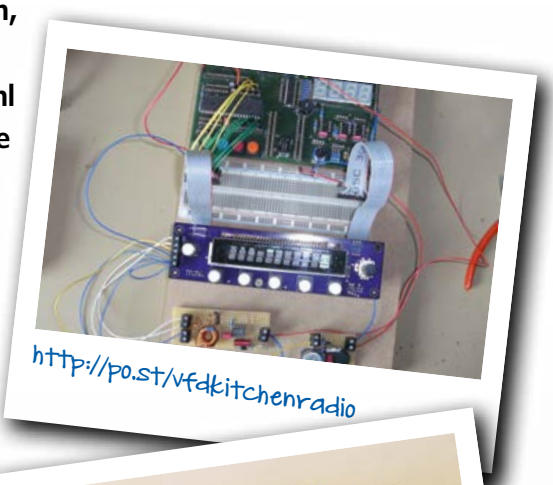
Nach dieser enttäuschenden Erfahrung hatten wir eine neue Idee: Wir kauften bei einem örtlichen Großhändler eine größere Stückzahl LED-Filament-Lampen zu einem niedrigen Einzelpreis und schlachteten diese Lampen aus. Dazu deckten wir ein Tuch über den Glaskolben und schlugen ihn vorsichtig in Scherben. Die Ausbeute waren bei jeder Lampe drei oder vier Filamente, die für unser „LEDitron“ genau passend waren. Hinzu kam, dass diese Filamente mechanischen Stress deutlich besser vertrugen als die in China georderten Exemplare. Die Träger waren nicht aus Glas, sondern aus einem biegsameren Material.

(150747)gd

Elektor-Labs.com

Vielfalt ist Trumpf

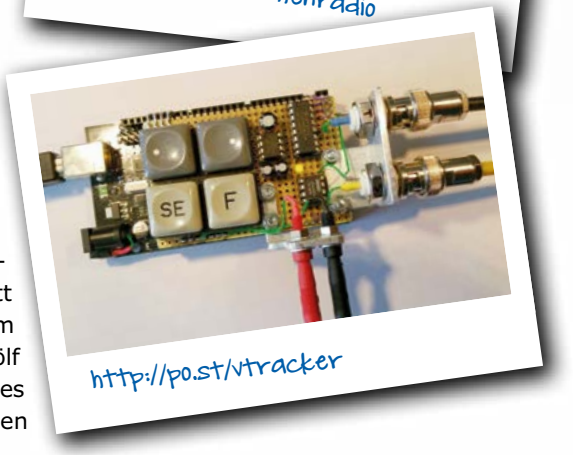
Das Wort „Vielfalt“ kommt dem in den Sinn, der die Projekte von Elektor-Labs.com durchsucht. Hier ist eine vielfältige Auswahl an Projekten, die ihre Initiatoren über viele Wochenenden beschäftigt hielten.



http://po.st/vfdkitchenradio

Spaß mit Fluoreszenz-Displays

Nach dem Ableben eines alten Küchenradios nahm es der Autor dieses Projekts auseinander, um die Anzeige, ein Vakuum-Fluoreszenz-Display (VFD), zu recyceln. Ja, das ist eines jener schönen Displays, die gespenstisch blau oder grün glimmen statt schwarz auf einem hässlichen grünlich beleuchteten Hintergrund. Unterstützt von einem Raspberry Pi hat der Autor mit dem Display ein neues Küchenradio samt einer Uhr und einem Timer zum Eierkochen und zum Backen aufgebaut. Ein ATmega32 steuert die Anzeige und die Tasten, während der RPi als Internet-Radio fungiert. Künftige Erweiterungen zur Home-Automation vorbehalten...



http://po.st/vtracker

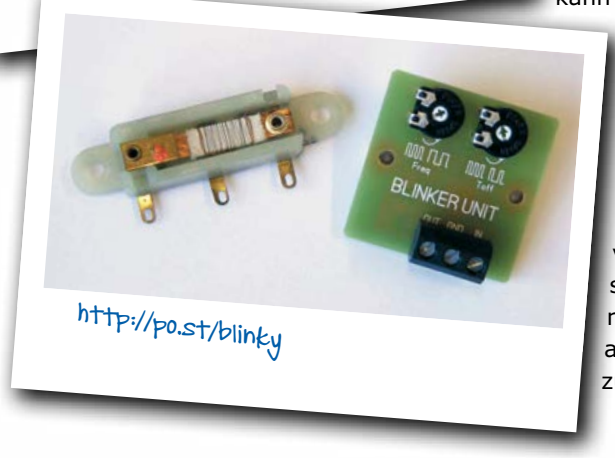
Spannungstracker für Oszilloskope

Wenn Sie ein sehr langsames Signal messen wollen, zum Beispiel eine Batterieentladungskurve, benötigen Sie ein Oszilloskop mit einer Zeitbasis von Stunden/div statt Sekunden/div. Dieses Projekt basiert auf einem Arduino-Mega2560-Board mit einem selbst entworfenen Add-on-Shield, das die Tracking-Zeit eines Oszilloskops auf zwölf Stunden und mehr erweitert. Als Bonus werden alle Parameter auf dem Bildschirm des Oszilloskops angezeigt. Natürlich funktioniert das System auch mit den guten alten analogen Oszilloskopen.

http://po.st/diseqc

Satellitenschüssel-Positionierung mit neuer Aufgabe

Stellmotoren/Rotoren für Satellitenschüsseln sind durch ihre Schneckengetriebe extrem robust und leistungsstark. Warum nicht die Vorteile dieser Systeme nutzen, um zum Beispiel terrestrische Antennen oder Überwachungskameras unter rauen Bedingungen zu positionieren? Der Drehwinkel von etwa 180 Grad dürfte für die meisten Anwendungen ausreichen. Das Protokoll *Digital Satellite Equipment Control* (DiSEqC), das ein Satelliten-Receiver verwendet, um die Drehung der Schüssel zu steuern, besteht aus einer bestimmten Sequenz von 22-kHz-Impulsen: Startbyte, Adressbyte, Befehlsbyte und Datenbyte, wobei jedes Byte von einem Paritätsbit abgeschlossen wird. Mit einem ATtiny2313-Mikrocontroller kann dieses Protokoll einfach in Bascom implementiert werden.



http://po.st/blinky

Blinky ist nicht wählerisch!

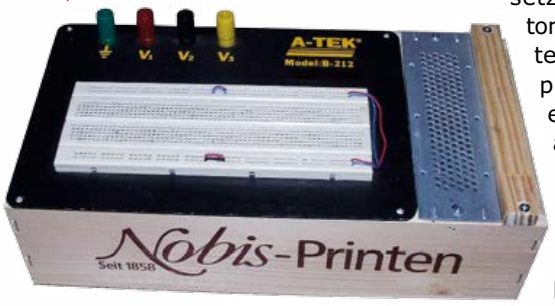
No, I’m not picky. I’m not rude. Why, I’ll eat any kind of food. Dies sind die ersten Zeilen eines Kindergedichts von Kenn Nesbitt, das gut zu diesem Elektor-Labs-Projekt passt. Die kleine Schaltung ist für alle Modellbauanwendungen geeignet, in denen ein Blinklicht erforderlich ist. Blinky verdaut Spannungen von 6 V bis 30 V AC und 9...42 V DC, ohne Magenschmerzen zu bekommen. Blinky kann Bimetall-Blinker ersetzen und ist nur 36x36 mm² groß. Die Schaltung ist mit einem klassischen 555-Timer aufgebaut, allerdings mit der CMOS-Version. Sie besitzt zwei Trimmpotis zur Einstellung der Frequenz und des Tastverhältnisses. ⏮

(150736)

Zusammengestellt von **Aniek Reuling**

Wem der Schuh passt....

Manchmal sind die Dinge so herrlich einfach. Unser Übersetzer David Ashton aus Australien testete alte Displays. Er wollte ein Steckbrett auf die Testschaltung setzen und montierte es auf einer Platte. Nur, in welches Gehäuse sollte er das alles stecken? Da erinnerte sich David an das Weihnachtspäckchen von Elektor mit einem Kistchen köstlichster *Aachener Printen* darin. Und wie Aschenputtels Schuh hatte es eine perfekte Passform!



Elektor-ID

Seit Ende 2015 gibt es die Elektor-ID, um die Anmeldeverfahren für unsere verschiedenen Plattformen (für Green- und Gold-Mitglieder) zu vereinheitlichen.

ELEKTOR-ID

Derzeit erlaubt Ihr Elektor-ID-Account den Zugriff auf die Magazine- und die Labs-Website.

READ ONLY MEMORY

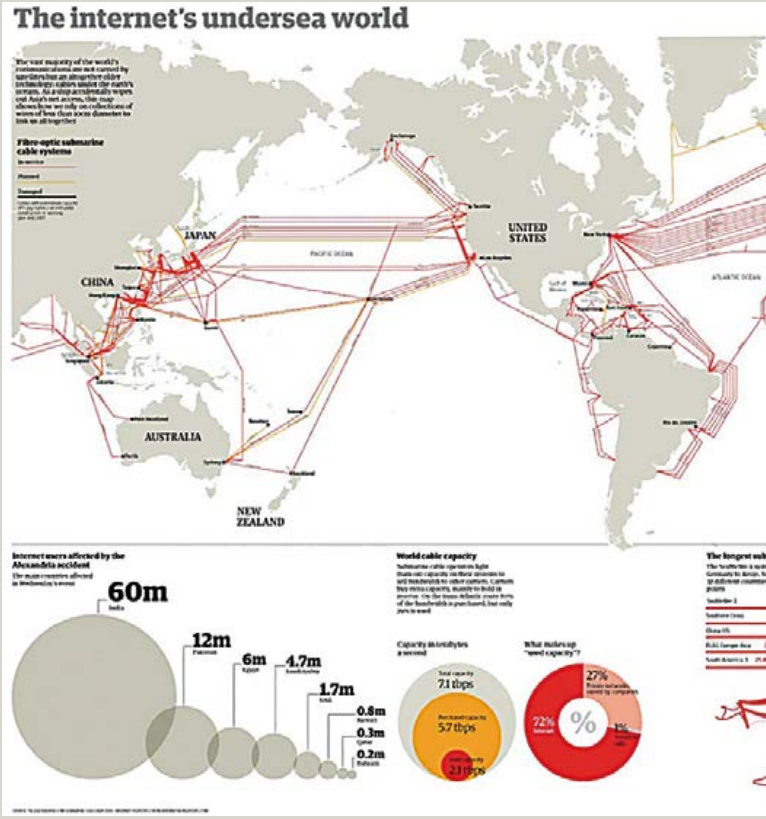
Unser Magazin kann auf eine lange Geschichte zurückblicken. In diesem Kasten zeigen wir stolz Vergangenes aus alten Tagen.

Vor dem Touchscreen war der Membranschalter. In den frühen 1980er Jahren waren Membranschalter der letzte Schrei, vor allem im Vergleich zu normalen Tastaturen. Diese „Raumzeitalter-Technologie" konnte zu einem Bruchteil der Kosten herkömmlicher Tastaturen hergestellt werden und war zudem vollständig Kaffee-resistent. Die Schalter konnten jede Form und jede Farbe annehmen und hatten keinerlei Verschleißteile. Die Verbindung zur Außenwelt verlief durch einen flexiblen Verbindungsstreifen, der als Zebra bezeichnet wurde.

Bessere Nutzung der Vorteile

elektorethics von **Tessel Renzenbrink**

Von digitalen Technologien kann die Menschheit stark profitieren. Sie können das Wirtschaftswachstum ankurbeln, Arbeitsplätze schaffen und die Menschen mit neuen Dienstleistungen versorgen. Nach Ansicht der Weltbank wird dieses Potential aber nicht optimal ausgenutzt. Am nachteiligsten ist, dass fast 60% der Weltbevölkerung noch keinen Zugriff auf das Internet besitzt. Diese digitale Kluft zwischen Internet-Besitzen-



den und Besitzlosen verursacht eine wachsende Ungleichheit. In ihrem *World Development Report 2016: Digital Dividends* fasst die Weltbank die Vorteile der digitalen Technologien in den drei Hauptkategorien zusammen: Inklusion, weil mehr Menschen, auch in ländlichen Gebieten, an der nationalen und sogar globalen Wirtschaft teilhaben können. Effizienz, weil Handel für Käufer und Verkäufer effizienter geworden ist und Innovation, weil E-Commerce-Plattformen, die auf Automatisierung und Skalierungseffekte angewiesen sind, die Transaktionskosten auf nahezu Null reduziert haben.

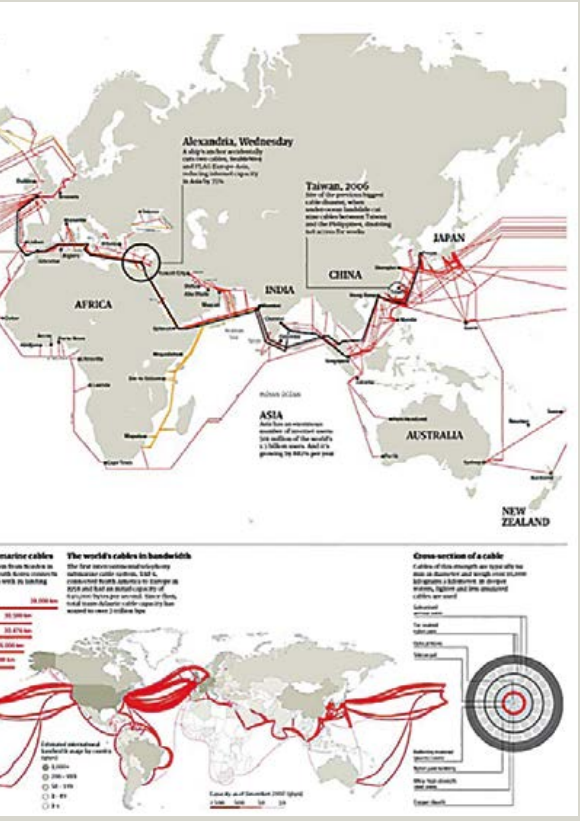
Aufkommende Risiken

Die Weltbank warnt aber, dass diese Vorteile nicht voll ausgeschöpft werden können, weil sie neuen Risiken gegenüberstehen wie Ungleichheit, Konzentration und Kontrolle. Der

PEOPLE NEWS • Viele Elektor-Mitarbeiter trafen sich in Aachen zum Office-Warming der neuen deutschen Labor sind stolz darauf, ein neues Gehäuse für die Nixie-Uhr mit Steampunk-Einflüssen zu präsentieren. Elektor, namentlich Margriet Debeij, Raoul Morreau und Johan Dijk, hat hart an der Einrichtung einer unterstützen: www.elektormagazine.de/elektor-business • Ab nächsten Monat wird Elektor World von

Digitaler Technologien

fehlende Zugang ist eine Triebfeder von Ungleichheit. Im Jahr 2015 hatten mehr als vier Milliarden Menschen keinen Zugang zum Internet. Wenn digitale Technologien weiterhin die Lebensbedingungen nur derjenigen verbessern, die online sind, erhöht dies die Ungleichheit zwischen On- und Offline-Menschen.
Das Risiko, die Ungleichheit zu erhöhen, lauert auch



in Gesellschaften mit hoher Vernetzungsdichte. Automatisierung kann mehr Arbeitsplätze vernichten als sie schafft. Wenn Gesellschaften sich solchen Veränderungen nicht anpassen, wird sich die Kluft wirtschaftlicher Ungleichheit erweitern.

Der zweite Risikokategorie der Konzentration liegt in Geschäftsinteressen, politischer Unsicherheit und Mangel an Wettbewerb begründet, der zu einer schädlichen Konzentration oder sogar Monopolisierung in vielen Bereichen führen kann. Und schließlich besteht die Gefahr, das Gleichgewicht der Kräfte für wenige Auserwählte außer Kraft zu setzen. Unternehmen und Regierungen nutzen digitale Technologien, um die Bürger kontrollieren, anstatt sie zu stärken.

Zentrale von Elektor • Die Leute aus dem Elektor-Mehr dazu in der Mai-Ausgabe • Das Client-Team von Online-Datenbank gearbeitet, um Elektor Business zu Robert van der Zwan zusammengestellt. ... ...

EXPERTENPROFIL

Elektor arbeitet mit mehr als 1.000 Experten und Autoren bei der Produktion von Büchern, Artikeln, DVDs, Webinaren und Live-Events zusammen. In jeder Ausgabe wollen wir einen von ihnen ins Rampenlicht stellen....

Name: **Robert van der Zwan, MA**

Alter: **56**

Ausbildung: **Philosophie**

Berufliches Interesse: **Technologie soll jedermann Spaß machen, Schreiben kristallklarer Manuals**

Wer ist Robert van der Zwan?
Auf der einen Seite bin ich Redakteur mit Interesse an Elektronik und Informationstechnologie. Ich schreibe und bearbeite Artikel über Elektronik und IT seit 1987. Auf der anderen Seite sehe ich mich als Verfasser von Handbüchern. Eine gute Anleitung zu schreiben ist eine echte Herausforderung!

Warum haben Sie angefangen, über Technik zu schreiben?
Ich wollte mit Leuten zusammen sein, die es toll finden, eigene Dinge zu entwerfen. Das ist tatsächlich sehr anregend! Ja, ich habe meine eigenen PCs zusammengebaut mit einer eigenen Anwendung hier und da. Wer würde nicht gerne mit Menschen zu tun haben wollen, die ihre Türklingel individuell abstimmten, ihr GPS-Navigationsgerät entwerfen oder die Avionik für ihr eigenes Flugzeug?

Wer ist Ihr größtes Vorbild?
Ich würde sagen, das ist „Mr. iFixit“, Kyle Wiens. Als sein iBook herunterfiel, wollte er das geliebte Gerät reparieren. Aber eine Reparaturanleitung für sein iBook war nirgendwo im Web zu finden - auch nicht auf der Website von Apple selbst. So entschied er sich, eine Website einzurichten mit nun mehr als 18.000 Reparaturhandbüchern für alle Arten von Geräten. Es ist kostenlos und wirklich von hoher Qualität! Sein Geschäftsmodell? Der Verkauf von Werkzeugen und Teilen, die bei einer Reparatur benötigt werden. Hervorragend!

Auf welches Projekt sind Sie am meisten stolz?
Wie ein Lehrer bin ich jedes Mal stolz, wenn ich Menschen helfen kann, ihre Angst vor dem Schreiben eines effektiven Handbuchs zu überwinden.

Über welche Themen werden Sie in Zukunft schreiben?
Die Themen werden zweifellos so vielfältig sein wie das Universum, solange sie nur mit Elektronik zu tun haben.

Was wird die wichtige Entwicklung der nächsten Jahre sein?
Schwer zu sagen. Ich bin nicht so sicher, was Wearables betrifft. 3D-Druck scheint ein Thema zu bleiben. Aber der Reihe nach: Ich hoffe, dass Batterien bald der Lage sein werden, so lange wie möglich zu funktionieren.

Angenommen, Sie bekommen 500 €, um damit im Elektor-Shop einzukaufen. Was würden Sie kaufen? Und warum?
Bücher, Bücher und ... Bücher. Ich will alles wissen, auch wenn ich weiß, dass ich nicht alles wissen kann. ⏮

(150739)

Hexadoku

Sudoku für Elektroniker

Kinder, wie die Zeit vergeht! Während ich diese Zeile schreibe, sehe ich draußen noch ein paar flüchtige Restchen von Schnee; doch wenn Sie diese Zeilen lesen, sind wir schon mit der Sommer-Doppelausgabe beschäftigt. Ein wenig Ruhe in diesen rastlosen Zeiten können Sie beim gemütlichen Lösen unseres monatlichen Hexadokus finden!

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 drei Gutscheinen 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 30. April 2016.

Die Gewinner des Hexadokus aus der Januar/Februar-Ausgabe stehen fest!

Die richtige Lösung ist: **BE840**

Einen Elektor-Wertgutschein über je 50 € haben gewonnen: Gilbert Dufieux, Klaus Kohlmann und Susanne Müller-Furrer.

Herzlichen Glückwunsch!

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

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

Das 21. Jahrhundert hat bereits begonnen
... und vieles muss noch entdeckt werden ...

Ich entwickle Elektronik
Ich lese elektor

www.elektormagazine.de

Autorisierter globaler Distributor für die NEUESTEN elektronischen Komponenten.

mouser.de/new

Die Zukunft lässt sich nicht mit Produkten aus der Vergangenheit gestalten.

Sichern Sie sich schon frühzeitig die **NEUESTEN** PRODUKTE

MOUSER ELECTRONICS

Die neuesten Produkte für Ihre neuesten Designs™