embedded projects JOURNAL
OPEN SOURCE SOFT-AND HARDWARE PROJECTS

Eine Open-Source Zeitschrift
zum Mitmachen!

Im Eilgang ins Weltall

[PROJECTS]

- Erweiterung der Gnublin API
- Experimentierboard
- Dual Licensing
- Ultra Low Power Design mit Atmel Mikrocontrollern
- IPv6 Funkübertragung mit dem Merkurboard
- Controlled Impedance for Edge-Coupled Coated Microstrip
- CAN-Bootloader
- Wetterballon mit Gnublin
- NetIO mit Gnublin

Einleitung

Ausgabe 03/2013

Embedded Projects Journal - Ausgabe No. 18

Einleitung

In dieser Ausgabe werden Artikel quer Beet vom einfachen Mikrocontrollerboard zum Experimentieren, Low-Power Mikrocontroller, CAN Bootloader, einer IPv6 Funkübertragung, Differentielle Signale mit Eagle, ein GNUBLIN Wetterballon und wieder etwas über rechtliche Aspekte mit Open-Source angeboten.

Eine bunte gesunde Mischung für Embedded Systeme. Etwas Hard- und Software gepaart Randthemen, die aber z.T. indirekt auf die Umsetzung der gesamten Lösung Einfluss nehmen.

An dieser Stelle möchten wir auf die Veranstaltung Elektor Live am 12. Oktober in Hanau (siehe Marktplatz) aufmerksam machen. Dort werden wir auch mit unseren Projekten und dem Journal sein :)

Benedikt Sauter

und das embedded projects Team

→shop.embedded-projects.net

HARDWARE FOR YOUR PROJECTS – ONLINESHOP

Design your GNUBLIN

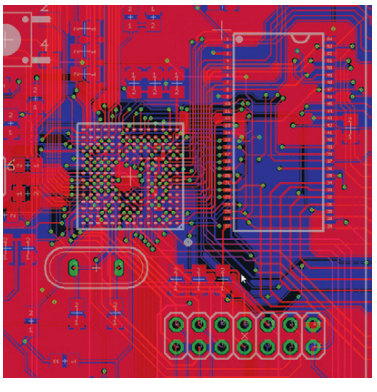
Sie suchen ein Board, das fast so wie GNUBLIN ist und brauchen davon nur eine kleine Menge pro Jahr? Wir passen Ihnen auf kurzem Weg die Schaltung an und ergänzen diese nach Ihrem Wunsch. Dank unserer internen Bestückung können wir Ihnen bereits ab kleinen Mengen ähnliche Stückpreise wie bei den GNUBLIN Boards in diesem Shop liefern.

→ **www.gnublin.org/designer**

Die einfachste Möglichkeit an ein eigenes embedded GNU/Linux Board zu kommen!

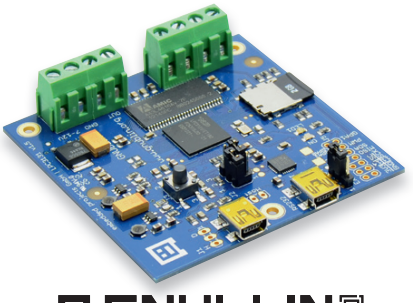
Einfach und genial ist das Produkt „Design your GNUBLIN“. Für alle, die kleine Serien von Platinen mit embedded GNU/Linux benötigen. Basierend auf der Schaltung der Boards der GNUBLIN Familie können wir einfach und schnell kundenspezifische Lösungen erstellen. Innerhalb kürzester Zeit können wir Ihnen die Boards von 1 bis ca. 1000 Stück liefern.

NEU: online GNUBLIN-Designer mit Kalkulator



Wir zeichnen den Schaltplan und das Layout ...

... bestücken mit einem Automaten in Augsburg ...



... und nehmen die Schaltung für Sie in Betrieb.

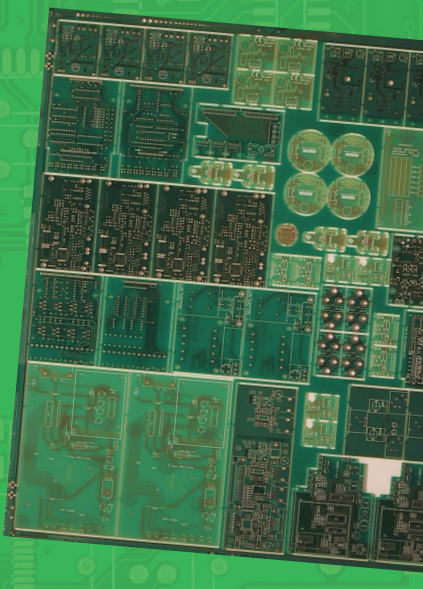
Holzbachstraße 4, D-86152 Augsburg
Tel +49 (0) 821 279599-0
Fax +49 (0) 821 279599-20
shop@embedded-projects.net

embedded projects GmbH

HARDWARE FOR PROJECTS

EURO CIRCUITS

Die Europäische Referenz für PCB Prototypen und Kleinserien



Kostensenkung durch Online-Pooling

- Keine Einrichtkosten
- Keine Mindesbestellwerte - ab der 1. Platine
- Sofortbestellung Online - ohne Vorkasse

Zeitgewinn durch Online-Daten-Check

- PCB Visualizer® - Sofort Design-Rule-Check
- Online Tipps für DFM (Design-For-Manufacturability)
- Online-Optimierung mit Galvano-Simulation

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

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

Preise inkl. 19% MwSt und ohne Transportkosten

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

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

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

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

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

- Leiterplatten mit einlagig isoliertem Metallsubstrat
- ab 3 AT

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

- 1-16 Lagen bis 90µm-Technologie
- ab 2 AT

www.eurocircuits.de

HOLEN SIE SICH DIE NEUESTE VERSION!

Neue Funktionen der Version 6.4:

- Simulation Ihres EAGLE Schaltplans in LTspice IV
- Anzeige und Suchfunktion für Attribute im ADD- und RE-PLACE-Dialog
- Import von Designdaten aus P-CAD, Altium und Protel über das Zwischenformat ACCEL ASCII
- Verbesserte Benutzerführung und Voreinstellungen (Tool-tipps, Shortcuts)

www.cadsoft.de

25 Jahre CadSoft
1988 - 2013

Erweiterung der Gnublin API

Julian Sarcher <sarcher@embedded-projects.net>

In den letzten Wochen ist die Gnublin API wieder stückweise weiter gewachsen. Die Gnublin Library wurde um eine Klasse erweitert, es kann eine weitere Schnittstelle angesprochen werden und es gibt Zuwachs in der Gnublin Modul Familie. Nun können erhobene Sensorwerte leicht mit einfachsten Befehlen in csv Dateien gespeichert werden, sowie die PWM Schnittstelle und ein Digital to Analog Converter (DAC) angesprochen werden. Die neuen nützlichen Funktionen der API stelle ich in diesem Artikel kurz vor.

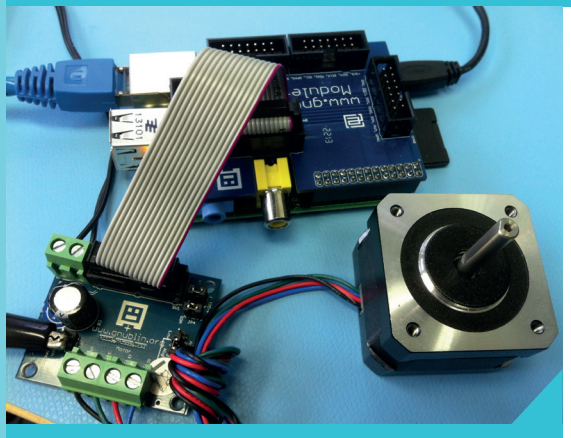


Abb. 1: Aufbau zur Ansteuerung des Motors

Status quo

Aktuell haben wir 12 Klassen in unserer Gnublin API, welche den Umgang mit dem Gnublin bereits um einiges erleichtert haben. Diese bieten neben der Ansteuerung von offiziellen Gnublin Modulen auch Zugriff auf die zahlreichen Schnittstellen. Zudem gibt es nun eine Bibliothek, mit welcher man einfach E-Mails über das Gnublin versenden kann.

Module:

- Temperatur-Sensor
- LCD 2x16
- LCD 4x20
- Relay
- IOExpander
- Step Motor (Abb. 1)
- ADC

Schnittstellen:

- Digital I/O
- Analog to Digital Converter
- I2C
- SPI

Bibliotheken:

- E-Mail

```
#define BOARD _ GNUBLIN
//#define BOARD RASPBERRY _ PI
#include „gnublin.h“
int main()
{
    gnublin _ module _ step motor;
              motor.setAddress(0x76);
    motor.getFullStatus1();
    motor.runInit();
    motor.resetPosition();
    motor.setPosition(1000);
    return 0;
}
```

Listing 1: Beispiel zur einfachen Ansteuerung des Step-Motors

CSV-Export, PWM Signal und Digital to Analog Converter

Mit der neuen Klasse *gnublin_csv* können erhobene Sensorwerte nun einfach und komfortabel in csv-Dateien geschrieben werden. Diese Klasse bietet Methoden zum Erstellen, Öffnen und Schließen einer CSV-Datei, sowie zum Einfügen neuer Zeilen, mit beliebig vielen Spalten. Ein besonderes Highlight sind die frei wählbaren Trennzeichen. Standardmäßig wird das Zeilen Ende mit den Steuerzeichen „\r\n“, die Spalten mit einem Strichpunkt und die Felder mit einem Anführungzeichen markiert (Listing 2).

Weiterhin kann nun auch die PWM-Schnittstelle über die API angesprochen werden. Wie bereits aus den gnublin-Tools bekannt, kann man die Frequenz über 4 verschiedene Clock-Divider einstellen. Ohne Clock-Divider beträgt die Frequenz 1400Hz. Zudem lässt sich der Duty-Cycle des PWM Signals mittels der Methode setValue(float value) einstellen. Wobei 0 einem Duty-Cycle von 0% entspricht und 100, einem 100% Duty-Cycle.

```
“Nummer“;“Temperatur“;“Helligkeit“;
“1“;“21“;“58“;
“2“;“22“;“62“;
“3“;“19“;“57“;
```

Listing 2: Beispielhafte csv Dateiausgabe

Auch gibt es Zuwachs in der Gnublin Modul Familie. Das Gnublin Module-DAC. Auf dem Digital to Analog Converter (DAC) ist ein MCP4728 verbaut. Dieser wird mit I2C angesteuert und hat vier Channel, welche mit unterschiedlichen Werten beschrieben werden können. Zudem besitzt der Chip zu jedem Channel neben dem Register auch ein EEPROM. Dort kann der Wert optional zusätzlich abgespeichert werden. Die API bietet drei verschiedene Schreibkommandos (write, writeAll, writeEeprom), ein Kommando um den Channels einen Gain zuzuweisen und ein Readkommando, welches die aktuellen Werte in den Registern auslesen kann.

```
gnublin _ pwm pwm;
pwm.setValue(50.1);
pwm.setClock(4);
```

Listing 3: Generieren eines PWM Signals

Experimentierboard

Christian Kähler <christian.kaehler@stud.hs-merseburg.de>

Einleitung

Jeder kennt es. Ein Anfänger fragt in einem Forum nach der Grundausstattung zur Controllerprogrammierung. Meist sind fertige Boards für die ersten Gehversuche auf dem Gebiet der Mikrocontrollertechnik recht kostspielig. Ein weiteres Problem stellt sich oft nach kurzer Zeit ein, wenn einem auffällt, dass die eine oder andere Schnittstelle fehlt, oder einfach nicht erreichbar ist. Genau so ging es mir am Anfang meiner zugegeben noch recht kurzen Controllergeschichte. Das bewegte mich dazu, mein eigenes Board (Abb. 1) zu entwerfen. Nach vielen Testaufbauten auf einem Steckbrett ist das Board, welches ich hier kurz vorstellen möchte, entstanden. Ich würde mich freuen, wenn der eine oder andere dieses Board evtl. nachbauen würde. Viel mehr würde ich mich allerdings freuen, wenn das Board nur als Inspirationsgrundlage dient und daraus eigene Ideen entstehen. Jeder hat schließlich andere Anforderungen an „sein“ Board. Weitere Eagle Dateien und die test.hex gibt es hier [1].

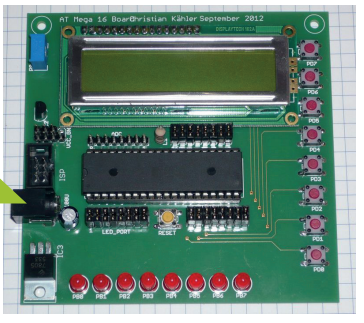


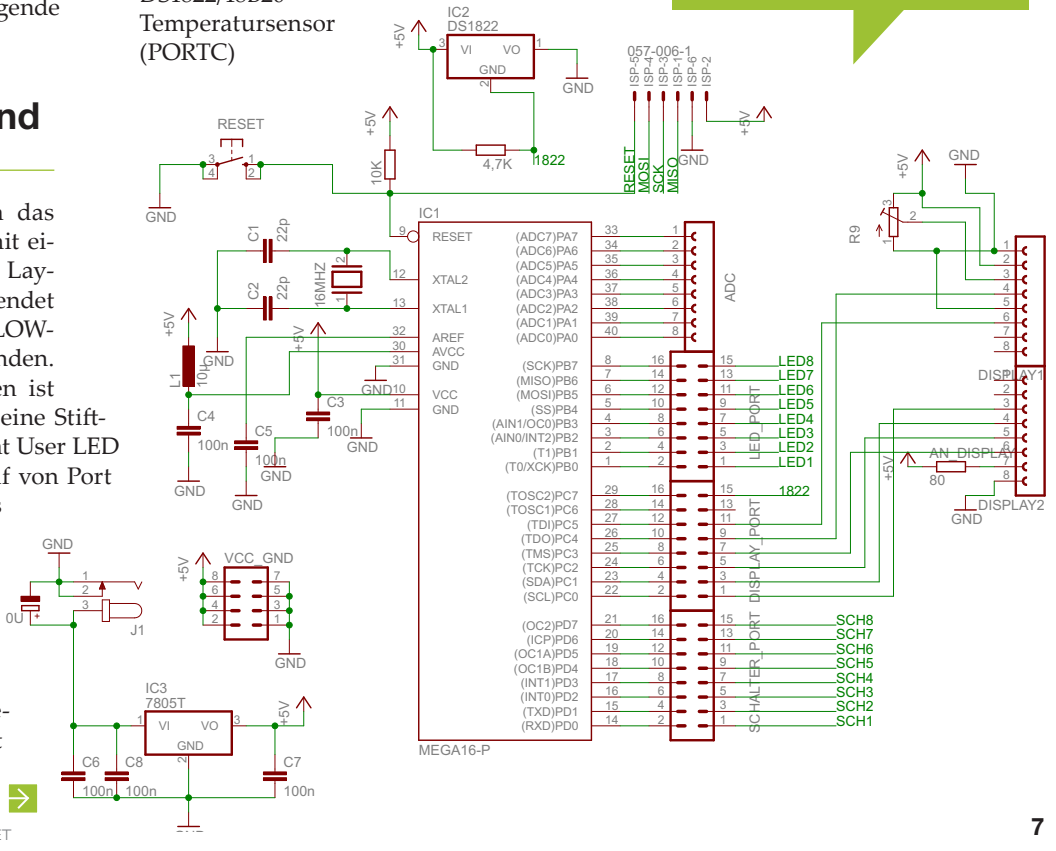
Abb. 1: Das fertig bestückte Board

Funktionsumfang

Was kann dieses Board eigentlich, was ein gekauftes Board nicht kann? Eigentlich nichts. Warum ist dieses Board dennoch entstanden? Ich wollte einfach mein eigenes Board entwerfen, auf dem keine LED oder kein Schalter zu viel oder zu wenig vorhanden ist. Außerdem war es nach dem Grundstudium nochmals eine gute Gelegenheit sich ausgiebig mit EAGLE zu beschäftigen. Einiges ist dabei auch schief gegangen, wie die Beschriftung am oberen Rand, welche nicht in Vektorgrafik portiert wurde. Grob zusammengefasst hat das Board folgende Eigenschaften:

- Pinlayout auf ATmega16/32 ausgelegt
- 7-10V VDC Eingangsspannung möglich
- Betriebsquarz ist frei wählbar (hier 16 MHz)
- Alle Ports sind frei erreichbar (über Stiftleisten)
- Integrierte Peripherie ist über Jumper angebunden
- VCC 5V abgreifbar (3 VCC/GND Paare)
- DS1822/18B20 Temperatursensor (PORTC)
- 2x16 LCD im 4Bit-Mode (PORTC)
- 8 User PushButton (PORTD)
- 8 User LED (PORTB)
- ADC frei verfügbar (d.h. nicht durch integrierte Peripherie vorbelegt)
- SMD 1206 Technik für komfortable Handlötung
- 6-Polige ISP Schnittstelle

Abb. 2: Die Schaltung des Experimentierboards



Schaltungsentwurf und Boardlayout

Die Schaltung (Abb. 2) und auch das Board sind für die Verwendung mit einem ATmega 16/32 ausgelegt. Als Layoutprogramm kann EAGLE verwendet werden. Alle Taster und LED sind LOW-Aktiv an den Controller angebunden. Port A mit seinen ADC-Eingängen ist komplett frei zur Verfügung über eine Stiftleiste verfügbar. An Port B sind acht User LED angebunden. Die Pins null bis fünf von Port C dienen der Ansteuerung des Displays. An Pin 7 hängt der DS1822 Temperatursensor. Alle Schalter (ausgenommen dem RESET) sind an Port D angebunden. Gerade bei meinen ersten Gehversuchen mit einem Pollin-Board hat mich die Belegung des 40 PIN Konnektors meist ziemlich genervt. Die Verwendung eines 40 PIN Konnektors hat sicher

den Vorteil, dass man übliche DIE Kabel zur Anbindung verwenden kann. Andererseits muss meist das Handbuch des Boards konsultiert werden, um herauszufinden wo welcher Port-Pin angebunden ist. Bei dem hier entworfenen Board sind Stiftleisten direkt am Controller angebracht. Das hat auch den Vorteil, dass man komfortabel nur einen Pin extern verwenden kann. Weiterhin gibt es 3 GND/VCC-Paare für die freie Verwendung.

Auch das Boardlayout (Abb. 3) wurde mit EAGLE erstellt. Hierbei wurden vorrangig SMD Bauelemente verwendet. Die Bauform 1206 lässt sich dabei noch komfortabel per Hand löten. Sollte das Board nachgebaut werden, ist darauf zu achten, dass die Schrift (oberer Rand) in eine Vektorgrafik konvertiert wird. Fertigen lassen kann man das Board über einen Anbieter im Mikrocontroller.net – Forum. In diesem Artikel könnt ihr euch über Preise und Vorgaben informieren [2]. Das Verhalten, der unter [1] erhältlichen test.hex, ist hier zu sehen [3].

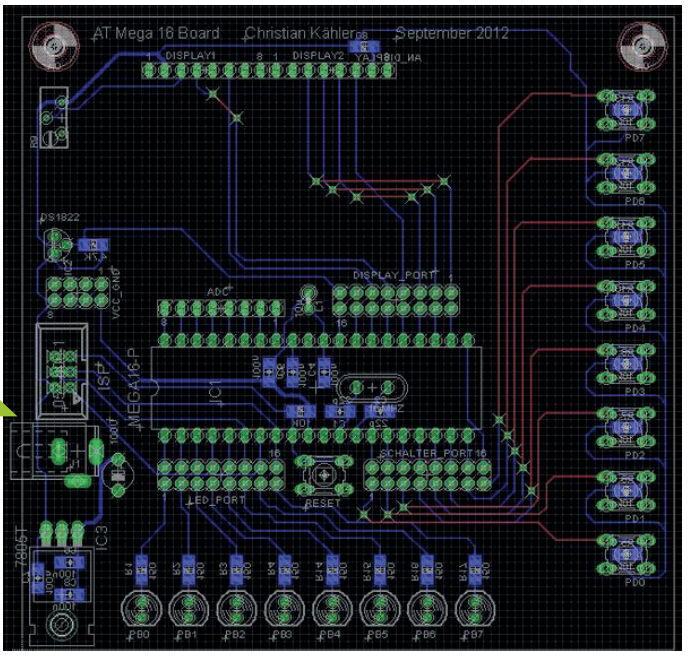


Abb. 3: Die Schaltung des Experimentierboards

QR-Code: Link zum Video aus [3].

Literatur

[1] www.christiankaehler.lima-city.de

[2] https://www.mikrocontroller.net/articles/Platinensammler

[3] https://www.youtube.com/watch?v=imdhV2YO3OY

Dual Licensing

Bernd Suchomski, LL.M. [1]

Die Lizenzierung von Open Source Software (OSS) erfordert Fingerspitzengefühl. Selbst wenn ein rechtlicher Schutz von Software erreicht wurde, ist es ein Balance-Akt, diesen auch effizient an den Abnehmer zu lizenzieren. In umgekehrter Weise müssen sich die Lizenznehmer darüber klar werden, dass eine Mehrzahl von verwendeten Lizenzen von Drittentwicklern auch das eigene Entwicklungsprojekt blockieren kann. Verschiedene Lizenzmodelle bieten die Möglichkeit einer Verbindung von freier und kommerzieller Software. Jedoch besteht dabei die Möglichkeit, offene juristische Flanken zu übersehen oder die Abnehmerschaft mit widersprüchlichen Angeboten zu verärgern. Dieses Phänomen kann Anhand der Java Virtual Machine (JVM) von Sun Microsystems erklärt werden. Deren Lizenzierung stärkte die Marktposition von Sun bzw. dessen Nachfolger Oracle und war Gegenstand eines Gerichtsprozess gegen den Internetkonzern Google, Inc. Google hatte zunächst mit Sun bzw. Oracle bei der Softwareentwicklung kooperiert. Als diese Kooperation aufgelöst wurde, hatte der Internetkonzern mit anderen IT-Firmen das Programm Dalvik für Android-Smartphones entworfen – einer Virtual Machine, welche die JVM ersetzen aber u.a. Teile von deren Schnittstelleninformationen verwenden sollte.

Das Prinzip der Virtual Machine

Eine Virtual Machine oder Runtime Environment ermöglicht die Kommunikation zwischen den Anwenderprogrammen (Applications oder Apps) und dem eigentlichen Betriebssystem eines Computers. Solche Computer finden sich derzeit vor allem in Smartphones wieder.

Dazu steuern die Apps die Schnittstellen (API) der Virtual Machine an. Das ist nur möglich, wenn sie genau die gleichen Schnittstellenbefehle nutzen, welche auch im Code der Virtual Machine (VM) niedergeschrieben sind. Zwar könnten die Apps auch unmittelbar mit einem Betriebssystem kommunizieren. Dazu müssten sie jedoch vorher auf ein bestimmtes Betriebssystem (z.B. Android, iOS, Windows, Linux etc.) abgestimmt sein. Eine einzelne App kann damit nicht auf einer Vielzahl von Betriebssystemen zum Einsatz kommen. Sie müsste vielmehr für jedes einzelne Betriebssystem geschrieben werden, was einen hohen Programmieraufwand erfordert.

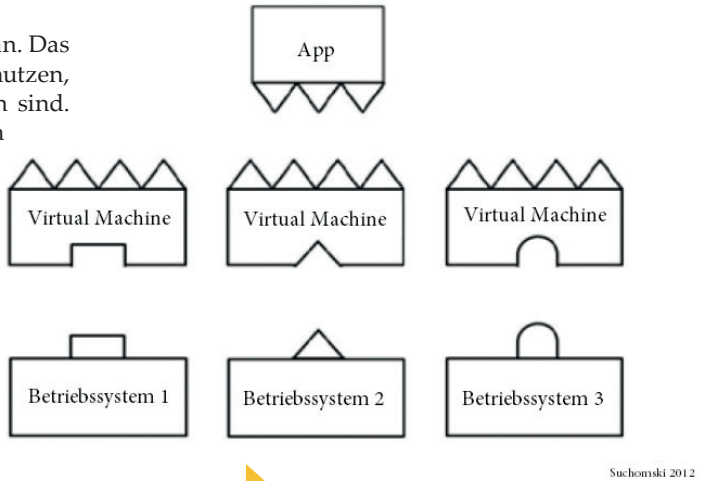


Abb. 1: Die Virtual Machine bietet eine einheitliche Plattform

Eine Virtual Machine löst dieses Problem. Sie hat den Vorteil, dass sie den App-Programmierern eine einheitliche Plattform auf verschiedenen Betriebssystemen bietet (Abb. 1). Die Entwickler müssen deshalb ihre Apps nur noch für eine einzige Virtual Machine schreiben und können sich dabei sicher sein, dass sie gleichzeitig auf verschiedenen Betriebssystemen zum Einsatz kommen können. Damit wird eine Unabhängigkeit der Anwenderprogramme von der Vielzahl der unterschiedlichen Betriebssysteme geschaffen.

Die Java VM und Oracles Lizenzstrategie auf dem Smartphonemarkt

Bei Smartphones werden Apps oft für die beliebte Java Virtual Machine von Oracle bzw. dessen Vorgänger Sun Mircosystems geschrieben [2]. Oracle vertrieb den Quellcode seiner JVM und die dazu gehörigen Schnittstellenbefehle praktisch nach einem Double-Licensing [3]. Das heißt, die Rechte daran werden grundsätzlich in zwei Formen lizenziert: einmal als Standardpaket mit einer kostenlosen OSS-Lizenz und alternativ als unter einer kommerziellen Lizenz (sog. Double-Licensing Abb. 2) [4].

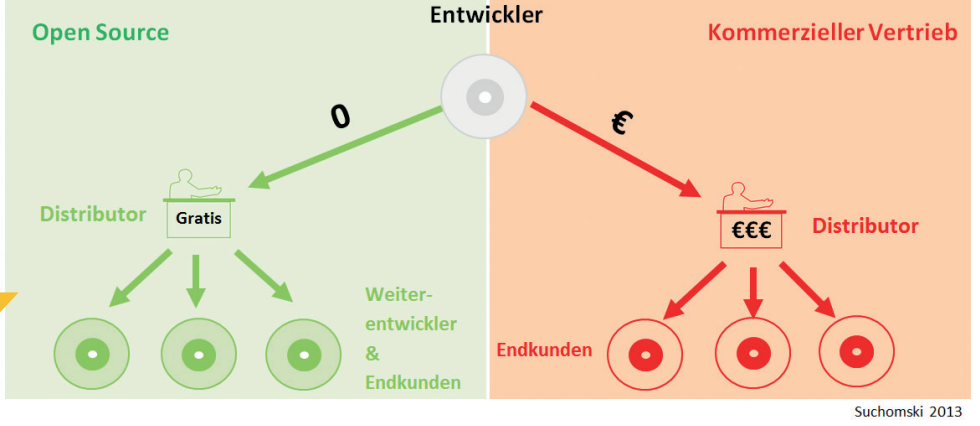
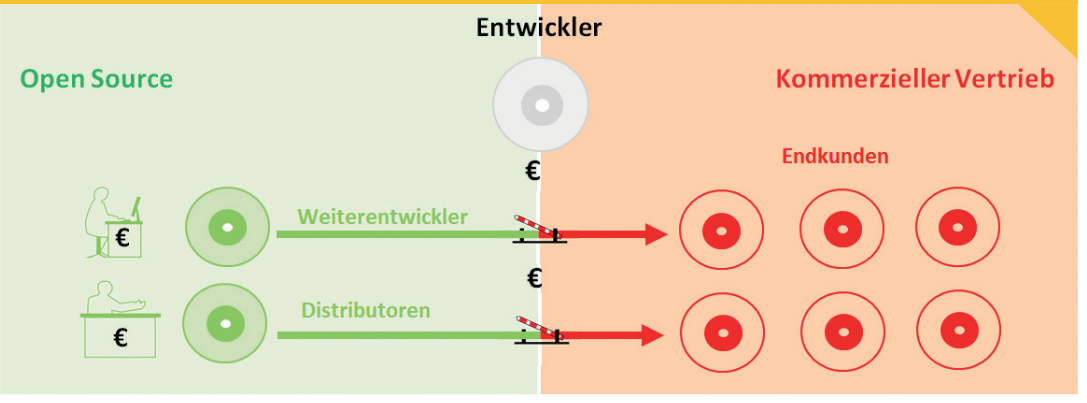


Abb. 2: Double Licensing Teil I

Dieses Prinzip dient zunächst der Verbreitung und Akzeptanz der Software. Durch die kostenlose OSS-Lizenzierung kann die Software schnell und weit verbreiten. Ferner würden deren Nutzer und Weiterentwickler zur Verbesserung der Software beitragen. Schließlich gilt das kommerzielle Angebot bei einem Double Licensing den Nutzern, welche die Software oder die damit ausgestatteten Computer für Geld weiterverkaufen wollen. Das konnten auch freie Entwickler und Distributoren sein (Abb. 3). Der ursprüngliche Entwickler und Lizenzgeber könnte für diese Umstellung von freier auf kommerzieller Verwertung noch eine Lizenzgebühr kassieren (in Abb. 3 durch die Schranke dargestellt). Damit verdient er und die freien Entwickler und Distributoren am Dual Licensing. Soweit so gut. Das Vorgehen von Oracle diente aber auch dem eigenen Wettbewerb: So ist die Java-Programmiersprache zwar auf dem Markt frei erhältlich. Das heißt, es bestand stets die Gefahr, dass andere Entwickler damit eine eigene Virtual Machine programmieren und veröffentlichen, welche der JVM Konkurrenz machen konnte. Allerdings hatte Sun schon früher seine Virtual Machine fertig gestellt und damit einen Wettbewerbsvorsprung.

Diesen Vorsprung hatte das Unternehmen in unerwarteter Weise genutzt. Anstatt ihn weiter kommerziell zu nutzen, stellte das Unternehmen Teile seiner JVM als Open Source Software gratis und quellcode-offen zur Verfügung. Als OSS-Lizenz wurde eine GPLv2ce-Lizenz gewählt. Sie verpflichtete den Lizenznehmer, bestimmte Weiterentwicklungen wieder unter derselben GPLv2ce-Lizenz dem Verkehr anzubieten, wenn er diese Entwicklungen verbreitet (sog. Copyleft). Das bedeutete, dass keiner der Weiterentwickler mit seinen Lizenzen der JVM Lizenzgebühren verdienen konnte.

Abb. 3: Double Licensing Teil II: Kommerzielle Lizenzierung freier Software - der Entwickler verdient mit



Suchomski 2013

Auch verbot die GPLv2ce, dass die Software bei einem Weitervertrieb hinsichtlich ihres Anwendungsbereiches beschränkt wird [5]. Die verbesserte Software dürfte danach auf jedem Gerät und zu jedem Zweck eingesetzt werden. Dabei blieb es jedoch nicht. Die GPLv2ce war nicht das einzige Lizenzabkommen, dass die Entwickler annehmen mussten, um die JVM-Codes verbreiten zu können. Zum Hauptsoftwarepaket – der Java Standard Edition – gehörte auch die Lizenz des Technology Compatibility Kit (TCK). Diese Lizenz musste abgeschlossen werden, um das selbst entwickelte Programm als offizielle Java-Entwicklung verbreiten zu dürfen. Sie ließ aber dagegen keine Verbreitung des Software-Codes in sog. Embedded-Systemen zu – d.h. als fest integrierten Bestandteil eines Computers [6]. Das bedeutet, der Entwickler hätte danach zwar frei Embedded-Programme für die JVM schreiben, sie aber nicht an Dritte weitergeben dürfen – es entstand ein Widerspruch zwischen der GPLv2ce und der TCK-Lizenz.

Diese Beschränkung galt demnach auch für Smartphones, da auf ihnen die Software auch fest installiert ist. Oracle ging davon aus, dass vor allem der Handy-Ausrüster Google an dem Vorhandensein der beliebten JVM auf seinen Smartphones interessiert war. Die App-Programmierer hätten dadurch einen Anreiz gehabt, neue Java-Programme für die Android-Smartphones von Google zu schreiben. Sie kannten sich ja schon mit der Java-Entwicklung für PCs aus. Das damit größer werdende App-Angebot für Android-Smartphones würde zur Beliebtheit dieses Betriebsystems von Google und so zum besseren Verkauf der damit bespielten Handys beitragen. Nach Oracles Lizenzmodell wäre Google deshalb gezwungen gewesen, über neue Lizenzen zu verhandeln, die eine Verwendung der JVM und ihrer Schnittstellen auf Handys ermöglicht [7]. Denn die Android-Handys sollten mitsamt ihrer Software vertrieben werden – ein Umstand, welcher der bisherigen TCK-Lizenz widersprach.

Oracles JVM-Lizenz in der Praxis

Sun hatte damit eine Lizenzstrategie mit besonderer wirtschaftlicher Tragweite entworfen. Diese kam durch das Phänomen zum Ausdruck, dass zunächst viele freie Entwickler deren Lizenzverträge akzeptierten, um Apps für Handys zu entwickeln. Dabei waren viele sich der Beschränkungen beim späteren Vertrieb für Embedded-Systeme durch die TCK-Lizenz nicht bewusst. Sie hatten zunächst nur die freie GPLv2ce-Lizenz im Blick.

Es ist anzunehmen, dass viele Entwickler die Bedingungen der TCK-Lizenz erst zur Kenntnis nahmen, als sie ihr bereits fertiges Programm verbreiten wollten. Sie realisierten deren Lizenzbeschränkungen also erst, nachdem sie viel Arbeit und Zeit in ihre Entwicklungen investiert hatten. In einem solchen Moment standen die Entwickler dann vor der Entscheidung: in den sauren Apfel beißen und eine kommerzielle JVM-Lizenz erwerben – oder die eigene Entwicklung aufgeben. Denn eben nur die kostenpflichtige JVM-Lizenz erlaubte die Verbreitung von Software für Embedded-Produkte.

In einem offenen Brief warnte die OSS-Ikone Richard Stallman deshalb vor dieser „Java-Falle“. Zwar hätten Suns Programme viele praktische Vorzüge. Bei einer Verbreitung der Programme sind ihre Weiterentwickler jedoch dem rechtlichen Konflikt zwischen ihren Lizenzverträgen ausgesetzt [8]. Einerseits sind sie verpflichtet, ihre OSS-Entwicklungen unbeschränkt unter der GPLv2ce weiterzugeben [9]. Andererseits unterliegen sie die den Verpflichtungen der TCK-Lizenz, die eine Verwendungsbeschränkung derselben Software für Embedded-Systeme vorsah. Damit bestand die Gefahr, bei einer Verbreitung der Software die TCK-Lizenz zu verletzen und verklagt zu werden [10].

Doch auch auf andere Weise hatte sich Sun bzw. Oracle einen Einfluss auf den Markt für Embedded-Software gesichert. Denn wer sich gegen die Java Standard Edition und das TCK entschied, um seine eigene Virtual Machine zu programmieren, war von Oracles Patenten bedroht. Für ihn bestand die Gefahr eines Patentverletzungsprozesses. Oracle hatte einige solcher Patente auf Funktionen seiner Virtual Machine. Damit kontrollierte die Firma nicht nur den Quellcode auf die JVM selbst, sondern auch Teile von deren abstrakten Funktionen. Das Unternehmen kontrollierte so zum Teil Parallelentwicklungen von Virtual Machines, welche die JVM ersetzen könnten.

Durch Oracles Lizenzstrategie entstand damit ein Multiplikator-Effekt. Die Firma stärkte neben der Kontrolle fremder Virtual Machines die Verbreitung und die Bekanntheit seiner eigenen JVM durch seine Gratis-Lizenzierung. Die wirtschaftliche Abschöpfung fand schließlich auf den interessanten Einsatzgebieten statt – hier auf dem neuen Markt für Smartphones. Diesen sicherte sich das Unternehmen für seine kommerzielle Lizenzierung.

Diese Lizenzstrategie hat aber auch Grenzen. Sie beginnen in den Bereichen, in denen der Patentschutz des Lizenzgebers nicht mehr greift – d.h. bei der Entwicklung von funktional unterschiedlichen Programmen. Hier darf es fremden Entwicklern nicht verwehrt bleiben, neue Software zu entwickeln, die zu den Programmen des Lizenzgebers kompatibel ist. Dazu können die Entwickler auf die Schnittstelleninformationen des lizenzierten Programms zugreifen, vgl. § 69e Abs. 1 UrhG. Hintergrund ist das Allgemeinwohl und der Wettbewerb, die von solchen neuen Programmen profitieren, welche sonst nie oder selten umgesetzt würden.

Oracle musste hier eine Niederlage vor Gericht einstecken, als die Firma versuchte, Google wegen der Verwendung der JVM-Schnittstellen-Deklarationen zu verklagen [11]. Google war es so gestattet, die Schnittstellenbefehle zu verwenden, um damit seine eigene Virtual Machine namens Dalvik auszustatten. Dass Dalvik auch die JVM funktional ersetzen konnte, lag allerdings laut der Gerichtsentscheidung nicht an einer Lücke im Patentschutz der JVM. Vielmehr hatten die Verantwortlichen von Sun und Oracle durch ihr missverständliches Verhalten bei der Zusammenarbeit mit Google den Anschein erweckt, als wollten sie dem Internetkonzern eine Ausnahmegenehmigung erteilen [12].

Schließlich sollte man sich auch aus kartellrechtlichen Gründen nicht immer auf einen Softwareschutz durch das geistige Eigentum verlassen. Das zeigt die Entscheidung der EU-Kommission gegen Microsoft aus dem Jahr 2004. Darin wurde der Microsoft-Konzern gezwungen, seine Schnittstellen für Konkurrenzentwicklungen gleich offenzulegen, um eine Marktabschottung zu verhindern. Das Verfahren führte zu einer der höchsten, je verhängten Bußgeldsummen in der EU und wurde damals u.a. von Sun Microsystems initiiert [13]. Auch ist es denkbar, dass eine so versteckte und widersprüchliche Lizenzstrategie schnell gegen deutsches Recht verstoßen hätte. Gemäß § 305c BGB gehen solche überraschenden und widersprüchlichen Allgemeinen Geschäftsbedingungen – wie sie z.B. in Lizenzverträgen vorliegen können – zu Lasten des Verwenders bzw. des Lizenzgebers. Ferner wird diskutiert, ob es nicht auch Konkurrenten möglich sein soll, gegen die unwirksamen AGB eines Mitbewerbers gemäß §§ 3, 4 Nr. 11 UWG vorzugehen [14].

Damit ist es an den Entwicklern, sich aus den verschiedenen Möglichkeiten der Softwarelizenzierung zu bedienen, um eine zukunftsträchtige und verlässliche Lizenzstrategie zu bilden. In rechtlicher Hinsicht kann diese Strategie mit verschiedenen technischen und urheberrechtlichen Schutzrechten abgesichert werden. Sie sollte auch klar und verständlich kommuniziert werden, um Widersprüche und Irreführungen zu vermeiden. Andererseits ist es für die Weiterentwickler und Lizenznehmer wichtig, dass sie vorab alle Lizenzen prüfen, die sie verwenden möchten. Dies erspart ihnen im Einzelfall viel Entwicklungszeit und Geld für weitere Lizenzen.

```
public static int max (int x, int y)   //Deklaration

{  if (x > y) return x;                 //Implementierung
   else return y; }
```

Abb. 4: Vergleich Schnittstellen-Deklaration und -Implementierung

Literatur

[1] Der Autor ist Rechtsanwalt und berät zum Schutz und zur Lizenzierung von Softwaretechnik, insbesondere Open Source Software. Der Beitrag ist seinem Aufsatz „Exit through the Java Trap“ in GB 3/2012, S. 178 ff. entnommen [http://www.recht.uni-jena.de/z10/gb/gbarchiv/GB_03_2012_screen.pdf#page=4].

[2] „[W]rite once run anywhere“, Oracle's Trial Brief, Docket No 568 v. 27. 10. 2011, S. 9 Tz. 12 ff., US District Court for the Northern District of California, Selected Filings: [http://cand.uscourts.gov/filelibrary/934/Doc%20568%20Oracle%20 Trial%20Brief%20--%20Unredacted.pdf].

[3] Grundlegend: Jaeger/Metzger, Open Source Software, 3. Aufl. 2011, S. 94 ff., Rn. 114 ff.

[4] Vgl. Riehle, The Java IP Story, 8 – OpenJDK Strategie (Dual Licensing Model): [http://dirkriehle.com/2011/06/30/the-java-ip-story/#S08].

[5] "General Public License - Version 2 with Classpath Exception": Ziffer 2 b) GPLv2ce: "If [you form] a work based on the Program (…) [y]ou must cause any work that you distribute or publish (…) to be licensed as a whole at no charge to all third parties under the terms of this [GPLv2ce] License.", wobei diese Regel bei der Einbeziehung bestimmter gekennzeichneter Programmbibliotheken nicht greift, vgl. "Classpath Exception" (-ce) am Ende der Lizenz, vgl. OpenJDK: [http://openjdk.java.net/legal/gplv2+ce.html].

[6] Sog. "stand-alone implementation", vgl. Ziffern 1.6, 1.12 a.E. 2.1 (b) (v), Exhibit A II TCK License, Java Community Process: [http://jcp.org/aboutJava/communityprocess/licenses/STANDaloneTCK7Final.docx]. Die Beschränkung wurde durch die zur Java Standard Edition notwendige Zusatzzertifizierung mit Oracles Technology Compatibility Kit (TCK) ausgelöst, siehe dazu Ziffer 2 (c) Java Standard Edition: "[License will be granted if the software] (c) passes the Technology Compatibility Kit (including satisfying the requirements of the applicable TCK Users Guide) for such Specification („Compliant Implementation"). In addition, the foregoing license is expressly conditioned on your not acting outside its scope.", vgl. Groklaw: [http://www.groklaw.net/articlebasic.php?story=20120221094600287#JavaLicense].

[7] Vgl. Oracle's Trial Brief [Fn. 2], S. 6 Tz. 4 ff., S. 20 Tz. 11 ff.

[8] Vgl. Stallman, Free Software – Free Society, FSF 2010, S. 215 ff.

[9] Ziffer 2 b) GPLv2ce, so. Fn. 5.

[10] Vgl. Riehle, The Java IP Story, 13 – Problems for OpenJDK Forks: [http://dirkriehle.com/2011/06/30/the-java-ip-story/#S13].

[11] Dazu der Autor in GB 3/2012, S. 173 – „Exit through the Java Trap" [http://www.recht.uni-jena.de/z10/gb/gbarchiv/GB_03_2012_screen.pdf#page=4]

[12] Ebd., S. 180

[13] 497 Millionen Euro: KomE 23. 3. 2004, ABl. 2004 Nr. C 900 final, S. 4 Tz. 2, S. 297 Tz. 1078.

[14] Vgl. BGH, Urteil vom 31. 3. 2010 - I ZR 34/08, GRUR 2010, 1117, 1118.

Ultra Low Power Design mit Atmel Mikrocontrollern

Dipl.-Ing. Andreas Riedenauer <riedenauer@ineltekmitte.de>

Der Energieverbrauch ist in den letzten Jahren zunehmend ins Blickfeld von Entwicklern und Anwendern gerückt. Batterie- und Akku-Betrieb sowie Energy Harvesting verlangen nach sparsamen Lösungen. Aber auch bei netzbetriebenen Geräten wird auf geringen Verbrauch Wert gelegt. Im Vortrag werden Energie- sparmaßnahmen am Beispiel der AVR und Cortex M4 Mikrocontroller von Atmel demonstriert [3]. Der vorliegende Beitrag gibt Hinweise zum Entwurf energiesparender Mikrocontroller-Schaltungen, aber auch zu einigen damit verbundenen Einschränkungen.

Bauteileauswahl

Neben den Controllern sollten alle relevanten Bauelemente in Betracht gezogen werden, da Einflüsse auf Gestaltung, Bedienung und Funktion des Endprodukts zu erwarten sind. Hier einige Anregungen:

Tasten und Folientastaturen sind unkritisch, da nur während der Betätigung ein geringer Querstrom durch die notwendigen Pullup- oder Pulldown-Widerstände fließt. Endschaltern und Reedkontakten, die dauerhaft im Ein-Zustand verharren können, ist jedoch Aufmerksamkeit zu widmen. Oft können Pullups nicht so hochohmig gewählt werden wie man es gerne hätte, da die Zuleitungen Störsignale auffangen. Werden zur Störunterdrückung zusätzlich Kondensatoren an den I/O-Pins vorgesehen, kann sich gerade bei sehr hochohmigen Pullups ein schwer zu findendes Fehlverhalten einschleichen: die Zuleitungen bilden aufgrund ihrer Induktivität zusammen mit solchen Kondensatoren Schwingkreise, die durch die plötzliche Entladung des Kondensators bei Tastenbetätigung zu Schwingungen mit einer weit über Vcc liegenden Amplitude angeregt werden. Die Amplitude wird dann zwar durch die Eingangsschutzdioden begrenzt, dies führt aber zu derart starken Impulsströmen, dass ein Latchup am I/O-Pin ausgelöst werden kann! Sicherste Abhilfe ist ein Serienwiderstand am Eingangspin, der den Strom in jedem Fall auf zulässige Werte begrenzt. Der Energieverlust durch Querströme lässt sich durch kurzzeitiges, periodisches Abfragen und anschließendes Abschalten der Tasten minimieren.

Für Potentiometer gilt ähnliches: nur Zuschalten zur Abfrage der Schleiferstellung, dann wieder einseitig trennen. Man erkennt, dass sich leicht ein höherer Bedarf an I/O-Pins ergibt als bei klassischen Designs.

Kapazitive Touch-Eingabe als Tastatur- und Poti-Ersatz wird zunehmend populär. Da entweder ein Spezialbaustein oder der mit einer Softwarelösung programmierte Mikrocontroller in Betrieb sein muss, bietet es sich an, auch hier periodisch abzufragen. Dabei beschränkt man sich zunächst auf einen Kanal, der eventuell bereits auf Annäherung reagiert, und aktiviert den Rest nur bei Bedarf. So lassen sich im Wartebetrieb durchschnittliche Stromaufnahmen im Mikroampere-Bereich erzielen. In die neuen SAM4L Controller mit Cortex M4 Core wurde die hardwaremäßige Unterstützung für kapazitive Sensoren übernommen, wie man sie bereits von den 32 Bit AVRs der UC3L Serie kennt.

Magnetische Winkel- und Lineargeber schieden bis vor kurzem aufgrund des durch das Hall-Prinzip bedingten Stromflusses für Geräte mit niedrigstem Strombedarf aus. Neuere Entwicklungen machen sie wegen ihrer technischen Vorteile bei extrem gesenktem Eigenverbrauch auch hier interessant [1].

Sensoren für chemische und physikalische Größen bilden ein weites Feld, so dass in diesem Rahmen nur kurz auf sie eingegangen werden kann. Zumeist gibt es zahlreiche Prinzipien, die miteinander zu vergleichen sind. Bisweilen lässt sich das Erfassen einer Größe sogar mit Energiegewinnung verbinden, wie bei der Helligkeitsmessung über Fotozellen, der Messung von Strömungsgeschwindigkeiten mittels durch die Strömung selbst betriebener Kleingeneratoren, Messungen an Hochspannungsleitungen, wo dem Feld ausreichend Energie zum Betrieb autarker funkvernetzter Sensoren entnommen werden kann oder der Schwingungserfassung über Piezo-Elemente. Ist solches Energy Harvesting nicht möglich, wird man sich auf periodisches Messen beschränken und ansonsten möglichst viele Komponenten deaktivieren. Auch sind die in der Regel erforderlichen analogen Bauteile sorgfältig auszuwählen; viele Hersteller bieten spezielle Low-Power Analogprodukte an [5]. Bei ADCs muss die erste Messung nach dem Ein- bzw. Umschalten des Multiplexers auf einen anderen Kanal meist verworfen werden.

Aktive Displays wie TFT und OLED verbieten sich aufgrund ihrer vergleichsweise hohen Stromaufnahme in vielen Fällen. Immerhin lässt sich durch bedarfsweises Ein- bzw. Hell-Schalten eine gewisse Einsparung erreichen. Um Größenordnungen sparsamer sind passive LC-Displays, gegebenenfalls mit zuschaltbarer Hintergrundbeleuchtung. Diese Option setzt allerdings transflektive Displays voraus, die im Vergleich zu reflektiven im passiven Modus kontrastärmer sind. Xmega (Abb. 1) und SAM4L Controller gibt es auch mit besonders sparsamen integrierten LCD Controllern. Elektrophoretische Displays (E-Paper) und die neuen bistabilen LCDs halten den letzten Bildinhalt auch ohne Stromaufnahme bis zum nächsten Bildwechsel. Während des

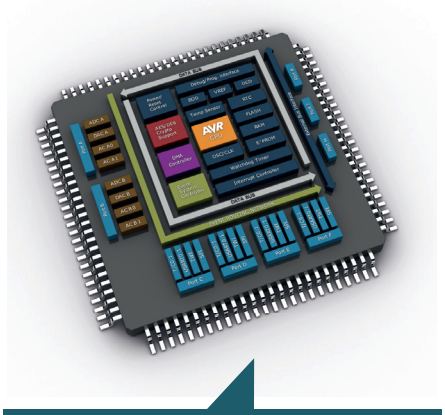


Abb. 1: Xmega Microcontroller

Wechsels benötigen E-Paper mehr Energie als LCDs, auch gibt es noch Einschränkungen bei Geschwindigkeit und Temperaturbereich. Für Etiketten, Türschilder etc. sind sie jedoch hervorragend geeignet. Vorsicht ist geboten bei Displays für Warn- oder Freigabemeldungen, da bei Stromausfall Falschanzeigen möglich sind! Die Ansteuerung Elektrophoretischer Displays ist komplex und erfordert spezielle Ansteuerchips oder damit bereits ausgestattete Controller.

Beschränkt sich die Anzeige auf LEDs, so wählt man hocheffiziente Typen und wird diese möglichst nur kurz blinken oder aufblitzen lassen. In letzterem Fall kann mittels einer zusätzlichen Induktivität eine Spannungsanhebung bei gleichzeitiger Strombegrenzung erfolgen, so dass auch aus niedriger Batteriespannung ein Betrieb möglich ist, ohne Energie in einem Vorwiderstand zu verschwenden. Rote LEDs haben eine geringere Durchlassspannung als andersfarbige. Dagegen ist das menschliche Auge im grünen Bereich am empfindlichsten und im blauen am unempfindlichsten.

Bistabile Relais benötigen nur während des Umschaltens kurze Stromimpulse. Es ist aber oft aus Sicherheitsgründen darauf zu achten, dass auch bei Stromausfall ein zuverlässiges Abschalten des Geräts gewährleistet bleibt. Bei Standardrelais kann mittels PWM der Haltestrom nach dem Umschalten abgesenkt werden. Ähnliche Techniken erlauben Ersparnisse bei Schrittmotoren im Haltebetrieb.

Zur akustischen Signalisierung bieten sich Piezo-Signalgeber an. Der Betrieb auf der Eigenresonanzfrequenz bewirkt besonders hohe Lautstärke. Hier kann eine relativ große Bauform vorteilhaft sein, damit diese Resonanz in einem auch von älteren Menschen gut hörbaren, nicht zu hohen Frequenzbereich liegt.

LF/RF Bausteine oder entsprechende Fertigmodule wie „active Tags“ werden im Bereich Zugangskontrolle, Personenüberwachung, Diebstahlschutz u.v.m. eingesetzt. Oft ist ein extrem sparsamer Wake-Up Receiver für das Langwellenband mit einem Transceiver für Hochfrequenz kombiniert [4]. Gerade unter den Wake-Up Receivern findet man Ausführungen, die im aktiven empfangsbereiten Zustand nur wenige Mikroampere aufnehmen. Allerdings gibt es erhebliche Unterschiede bei der Empfindlichkeit der Empfänger und anderen Merkmalen mit Einfluss auf die Energiebilanz. Auf der HF-Seite beeinflussen u.a. Frequenzbereich, Antenne und verwendetes Protokoll in hohem Maße den Energiebedarf [2].

Controller-Auswahl

Aufgrund des Pflichtenhefts ergeben sich bereits zahlreiche Anforderungen an den zu wählenden Mikrocontroller. Nicht nur technische Vorgaben zur Erfüllung der eigentlichen Aufgabe sind maßgebend, auch weitere Faktoren wie Safety-Funktionen, vorhandene Libraries, Vorerfahrungen mit der Controllerfamilie, Entwicklungsumgebungen, (Langzeit-)Verfügbarkeit, Qualität, Support durch Hersteller und Lieferant, Einsatz bei anderen Produkten und damit verbundene Lagerhaltung sowie eventuelle Preisvorteile durch Bündelung u.v.m. spielen eine wichtige Rolle. Nun kommt der Aspekt der Energieersparnis noch hinzu. In manchen Fällen wird man einen speziell auf niedrigsten Energiebedarf hin konstruierten Controller wählen (müssen), oft jedoch reicht es aus, einen sparsamen modernen Standard-Controller einzusetzen und die Schaltung hard- und softwaremäßig zu optimieren. Wer extrem an Energie sparen muss und einen Spezialbaustein ins Auge fasst, sollte besonders sorgfältig vergleichen, denn die inzwischen gestiegene Sensibilität der Entwickler in puncto Energieverbrauch hat manchen Hersteller zu Marketingaussagen veranlasst, die überzogene Erwartungen wecken können. Hier ist gesundes Augenmaß angesagt. Liegt die geringstmögliche Stromaufnahme im Tiefschlaf eines bestimmten Controllers bei einigen Mikroampere, so ist das für Automobilanwendungen und im Industriebereich meist unerheblich. Für einen Langzeitdatenlogger dagegen, der über viele Jahre aus einer kleinen Batterie gespeist wird, ist das deutlich zu viel. Andererseits tappt in eine Falle, wer sich hier allein durch extreme Angaben von wenigen Nanoampere beindrucken lässt, wenn in der realen Anwendung eine auch nur geringfügig schlechtere Performance sich in einer Verlängerung der aktiven Phase niederschlägt und alle im Tiefstschlaf erzielten Nanowatt-Einsparungen zunichtemacht. Nur die genaue Betrachtung von Peripherie, Programmlaufzeiten und verfügbaren Energiesparoptionen ermöglicht einen fairen Vergleich.

Sleep Modus	Main Clock	RTC	Wakeup	SPM & EE-PROM Ready Wakeup	ADC Complete Wakeup	RTC Wake-up	Sonstige Interrupt Wakeups	Anmerkung
Idle	Ein	Ein	Schnell	Ja	Ja	Ja	Ja	
ADC Noise Reduction	Ein	Ein	Schnell	Ja	Ja	Ja	Nein	Wie Idle, aber weniger Module aktiv
Power Down	Aus	Aus	Langsam	Nein	Nein	Nein	Nein	Nur externes Wecken
Power Save	Aus	Ein	Langsam	Nein	Nein	Nein	Nein	Wie Power Down, aber Selbstwecken möglich
Standby	Ein	Aus	Schnell	Nein	Nein	Nein	Nein	Power Down, nur Mainclock an
Extended Standby	Ein	Ein	Schnell	Nein	Nein	Nein	Nein	Power Save, nur Mainclock an

Tabelle 1: Sleep Modi bei AVR Controllern

Unbenutzte I/O-Pins sind als Eingänge zu konfigurieren, da so der Strombedarf etwas niedriger ist als bei offenen Ausgängen. Digitale Eingänge sollten mit einer Abweichung von weniger als 0.5 Volt auf Masse oder Vcc liegen, was durch ihre externe Beschaltung oder durch Aktivierung der internen Pullups erreicht wird. Die Einhaltung der Logikpegel für High und Low reicht nicht aus! Die Pullup-Widerstände sind im RESET-Zustand noch nicht aktiviert, da die zugehörigen Register noch ihre Initialwerte enthalten. Um auch in dieser Phase den Stromverbrauch zu minimieren, sind externe Pullups erforderlich. ATxmegas haben auch Pulldown-Widerstände integriert.

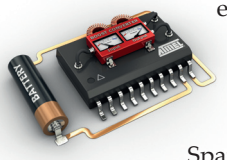


Abb. 2: Atmel ATtiny

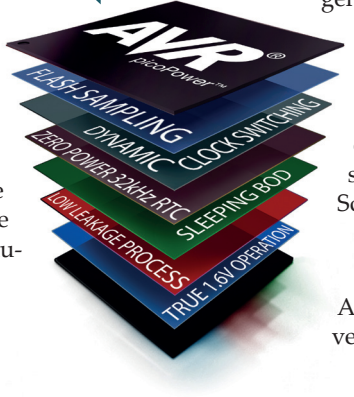
Die Stromaufnahme ist bei CMOS-Stufen etwa proportional zur Taktfrequenz. Daher wählt man diese bei Dauerbetrieb nur so hoch, dass die benötigte Rechenleistung erzielt wird. Falls kein Dauerbetrieb erforderlich ist, zeitweise jedoch eine schnelle Abarbeitung benötigt wird, sollte man die verschiedenen Sleep-Modi nutzen (Tabelle 1). Dabei wird die CPU die meiste Zeit über vom Systemtakt getrennt oder der Oszillator wird ganz abgeschaltet. Bei Bedarf wird die CPU geweckt solange nötig, danach wieder schlafen gelegt.

Steht ein externes Wecksignal wie Tastendruck oder ein Schnittstellen-Signal zur Verfügung, kann der Controller in den sparsamsten Sleep-Mode , den Power Down Mode mit typisch 0,1 µA, versetzt werden. Soll der Controller von selbst aufwachen, muss ein Timer laufen - in der Regel der 32kHz Uhrenoszillator bei einer Stromaufnahme von 0,5 bis 10 µA. Solche Angaben sind typische Werte und beziehen sich auf eine Temperatur von 25 °C. Die maximalen Werte können deutlich höher liegen. Hier empfiehlt sich ein Blick ins Datenblatt. Noch sparsamer sind moderne externe RTC-Bausteine, die auf Werte bis herab zu 130 nA kommen und so konfiguriert werden können, dass sie den Controller periodisch aufwecken [6].

Im Intervallbetrieb mit langen Sleep-Pausen und kurzen aktiven Phasen ist es nicht gleichgültig, ob in den aktiven Phasen schnell getaktet und kurz gerechnet oder langsamer getaktet und dafür länger gerechnet wird. Kurze, schnell getaktete aktive Phasen sind etwas günstiger. Allerdings setzt die Betriebsspannung Grenzen, und die internen RC-Oszillatoren können bei vielen Controllern nur die halbe Maxi-malfrequenz erzeugen. Externe RC-Kombinationen ermöglichen aber auch höhere Frequenzen.

Ist beispielsweise durch die Baudrate des UART ein Zeitfenster vorgegeben, sollte man zeitweise den Takt auf das gerade noch ausreichende Maß absenken und erst für Berechnungen wieder auf den höheren Takt umschalten. Dabei ist es günstiger, zwischen geeigneten Taktquellen umzuschalten, als nur den Teilerfaktor des Prescalers zu variieren. Grundsätzlich sollte man höhere Frequenzen als maximal benötigt erst gar nicht erzeugen. In dieser Hinsicht ist der XMEGA besonders flexibel. Wird er mit 16 MHz betrieben, ist es besser, diese Frequenz mittels 2-MHz-RC-Oszillator und PLL zu erzeugen, als einen 32-MHz-Takt herunter zu teilen. Benötigen CPU und Peripherie unterschiedliche Takte, sollte der Hauptoszillator, aus dem diese abgeleitet werden, so langsam wie möglich laufen.

Abb. 3: Schema AVR picoPower



Taktoszillatoren unterscheiden sich in Stromaufnahme, Einschwingzeit und Stabilität. Bei Nutzung der Sleep-Modi und begrenzten Anforderungen an Genauigkeit und Frequenzstabilität ist der interne RC-Oszillator zu empfehlen. Das spart nicht nur den vibrationsempfindlichen Quarz und Platz auf der Platine, sondern auch Zeit beim Einschwingen und damit Energie: Ein Quarz benötigt über 15.000 Schwingungen, bis er sauber oszilliert, ein RC-Oszillator nur sechs! Keramikresonatoren liegen mit 200 bis 1000 Schwingungen dazwischen. Ihre Genauigkeit reicht für asynchrone serielle Schnittstellen, nicht aber für eine Uhr. Neuere AVRs verfügen über einen Temperatursensor, der die Kalibrierung des RC-Oszillators im Register OSCCAL abhängig von der Temperatur mit 1-2% Abweichung ermöglicht, was für UARTs ausreicht. ATxmegas verfügen über einen im kompletten Spannungs- und Temperatur-Bereich auf 1% kalibrierten 32-kHz-RC-Oszillator. Auch die neuen ATtiny1684 (Abb. 2) sowie die SAM4L Serie sind mit besonders sparsamsten Oszillatoren versehen. Die RC Oszillatoren der neuesten Xmegas sind derart genau, dass nicht einmal für USB ein Quarz benötigt wird! Die Quarzoszillatoren moderner AVRs können wahlweise für „Low Power“ oder „Full Swing“ Betrieb konfiguriert werden. Durch die geringere Amplitude im Sparmodus sinkt aber naturgemäß die EMC-Festigkeit. Eine Übersicht der Oszillatortypen zeigt Tabelle 2.

Oszillatortyp	Genauigkeit	Einschwingzeit in Zyklen
Quarz	10-50 ppm	16k
Uhrenquarz 32kHz	10-50 ppm	16k - 32k
Keramikresonator	0,5-1 %	200 - 1000
RC-Oszillator	1-2% (kalibriert)	6
Externer Takt		6

Tabelle 2: Oszillatortypen

Die Stromaufnahme ist nahezu proportional zur Betriebsspannung. Je niedriger UB, desto niedriger ist allerdings auch die maximal zulässige Taktfrequenz. Einige Controller enthalten einen Aufwärtswandler, der den Betrieb an nur einer Batteriezelle erlaubt. Beim ATtiny23U/43U ist sicheres Anlaufen ab 0.9 V gewährleistet, wodurch die Batterie sehr gut ausgenutzt wird. Um Tiefentladung zu vermeiden wird unterhalb von 0,6 V der gesamte Chip abgeschaltet. Der Schaltregler selbst benötigt 17 µA, was für Schaltungen mit ständig laufendem Timer noch zu viel sein kann.

Bei zu weit absinkender Betriebsspannung ist das Verhalten von Mikrocontrollern nicht mehr vorhersagbar. Dies lässt sich durch eine Brown Out Detection (BOD) vermeiden, die den Controller bei Unterschreiten des Mindestwertes in den RESET-Zustand versetzt. Pico-Power-AVRs (Abb. 3) verfügen über eine Sleeping-BOD, die sich im Sleep-Mode abschalten lässt, ATxmegas über eine „Sampled BOD“, die nur zeitweise die Spannung überprüft. Bei Batterien sinkt die Spannung so langsam, dass es oft genügt, sie periodisch über den ADC oder den analogen Komparator zu überprüfen. So lässt sich zudem bereits deutlich vor Erreichen der BOD-Schwelle eine Batteriewarnung ausgeben.

Auch das Softwaredesign hat Einfluss auf den Stromverbrauch: Unterprogramme sparsam einsetzen,

schnelle statt Speicherplatz sparende Algorithmen einsetzen, Look-Up-Tabellen anstelle langwieriger Rechnungen, Interrupts statt Polling, relative Sprünge verwenden, häufig benutzte Variablen in Arbeitsregistern halten. Bei Hochsprachen die Optimierung zugunsten schneller Codeausführung wählen.

Besonderheiten bei AVR Mikrocontrollern

AVR und SAM4(L) sind aufgrund des statischen CMOS-Designs und der HARVARD-Struktur schon sehr sparsame Mikrocontroller, sowohl absolute als auch auf die Performance bezogen. Beim AVR sind Features wie der effiziente RISC-ähnliche Befehlssatz, Zero Flag Propagation und das Flash-Sampling zu nennen, bei dem der Programmspeicher nur während der Zugriffsphase mit Strom versorgt wird. Innerhalb der AVR-Familie variiert die Ausstattung mit Stromsparfunktionen. Bei den klassischen AVRs sind die Pico-Power-Ausführungen (Abb. 4) am sparsamsten und darunter wiederum die A-Typen (z.B. ATmega 48/88/168PA), die um zirka 30 Prozent sparsamer sind als ihre Vorläufer. XMEGAs sind grundsätzlich mit modernsten Techniken ausgestattet, insbesondere die Versionen mit USB (z.B. ATxmega 128A1U).

Nicht benötigte Peripherie sollte abgeschaltet werden. Moderne AVRs haben ein Power Reduction Register (PRR), über das Timer, ADC, USART und TWI (I2C) durch Setzen eines Bits vom Takt getrennt werden können; sie verbleiben dann in ihrem letzten Zustand, bis der Takt wieder frei gegeben wird. Der Analogkomparator muss extra abgeschaltet werden, nämlich durch Setzen des Bits ACD im Register ACSR. Die Ersparnis beträgt etwa 60 µA bei 3 V. Dabei muss der Analog Comparator Interrupt gesperrt sein, sonst wird er durch Ändern des Bits ausgelöst. Dazu ist das ACIE-Bit im Register ACSR zuvor zu löschen.

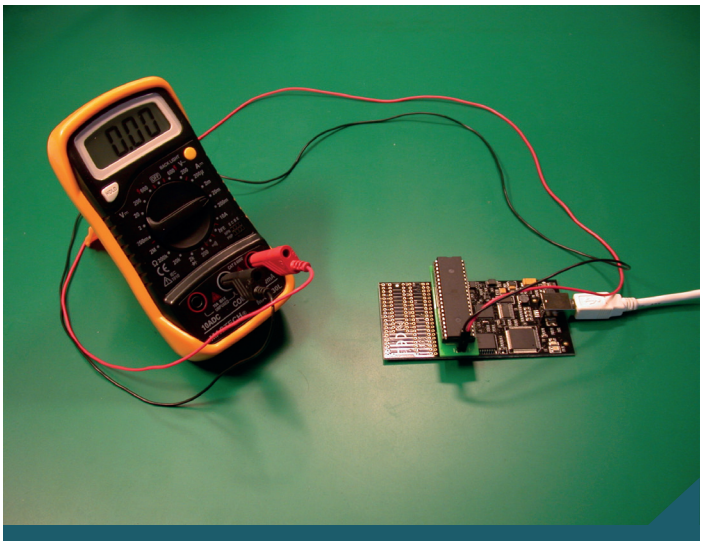


Abb. 4: Messung der Stromaufnahme des AVR picoPower

Wenn die Bandgap-Diode (Vref) intern mit einem Komparator-Eingang verbunden ist, fließen etwa 15 µA, und zwar auch im Sleep Mode. Deshalb sollte das ACBG Bit im Register ACSR auf null gesetzt werden. Tabelle 3 fasst die Einsparmöglichkeiten zusammen.

Der Adressraum für die Special Function Register (SFR), in dem schnelle Spezialbefehle angewandt werden können, ist beschränkt, so dass einige Peripherie-Register in den SRAM Bereich gelegt werden mussten. Durch geschickte Auswahl der Peripherie können einige Takte eingespart werden. Zum Ausgleich stehen außerdem General Purpose I/O-Register zur Verfügung, auf die ein schnellerer Zugriff möglich ist als auf das SRAM. Bei XMEGAs kann man darüber hinaus die aktive Phase durch so genannte Virtual Ports verkürzen, die den Zugriff auf die Register DIR, IN, OUT und INTFLAGS von bis zu 4 Ports mit Single-Cycle-Befehlen zu Datentransfer und Bitmanipulation gestatten.

Modul	Einsparung aktiv	Einsparung Idle Mode
USART	2 %	6 %
Asynchroner Timer (RTC)	4 %	15 %
Timer/Counter	2 %	6 %
ADC	4 %	14 %
SPI	3 %	11 %

Tabelle 3: Einsparungen bei abgeschalteter Peripherie

Der ADC wird nur bei Bedarf eingeschaltet und dann möglichst im Single-Conversion-Mode betrieben. Im Noise-Reduction-Mode wird die CPU während der Wandlung abgeschaltet, was nicht nur Strom spart sondern auch die Genauigkeit erhöht: Störungen des ADC durch die Aktivitäten der CPU fallen weg. Reichen 8 Bit Auflösung anstelle der möglichen 10 Bit, verkürzt sich die Wandlungszeit des ADC von 65 µs auf 12 µs. Da aber kein eigener 8-bit-Modus zur Verfügung steht, muss man den ADC-Takt auf einen entsprechend hohen Wert stellen und durch Setzen des ADLAR-Bits das signifikante Ergebnisbyte ins ADCH-Register schieben. Dieses Register liest man 12 µs aus und schaltet dann den ADC ab.

Liegen analoge Spannungen an Eingangspins, die auch digitale Funktion haben, fließen in den digitalen Eingangsstufen Querströme, da beide Transistoren teilweise leiten. Bei Pico-Power-Typen können die digitalen Eingangszweige über das „Digital Input Disable Register" (DIDR) bitweise getrennt werden. Beim XMEGA kann man den Non-Volatile-Memory-(NVM-)Controller stromsparend konfigurieren. Dabei werden das EEPROM sowie nicht benutzte Teile des Flashs abgeschaltet. Pageweise in das EEPROM zu schreiben ist sparsamer als byteweise.

Den Watchdog-Timer (WDT) abzuschalten spart 6 µA bei 3 V, er wird möglichst nur in der Entwicklungsphase benutzt. Bei den XMEGAs gibt es dank DMA und Event-System mehr Möglichkeiten, die CPU anzuhalten und dennoch auf Ereignisse zu reagieren und sogar Teilaufgaben selbständig in der Peripherie ablaufen zu lassen. Ein Datenlogger etwa kann fast ohne CPU arbeiten: Über das Eventsystem löst der RTC periodisch A/D-Wandlungen aus, deren Ergebnisse über die DMA direkt im Speicher unter fortlaufenden Adressen abgelegt werden. Die CPU wird nur bei Bedarf per Interrupt geweckt, zum Beispiel, wenn das Ergebnis außerhalb eines zuvor definierten Bereichs liegt. Die DMA reduziert die CPU-Belegung um 96 % bei SPI-Kommunikation mit 4 Mbps und um 57 % bei UART-Kommunikation mit 2 Mbps. Die

Crypto-Unit erlaubt AES-Verschlüsselung beim XMEGA (und auch beim SAM4LC) unabhängig von der CPU. Bei den Controllern mit LCD-Controller wählt man spezielle Ansteuer-Signalformen aus und hält die Frame Rate möglichst niedrig, siehe im Datenblatt unter „Minimizing Power Consumption“ und „Low Power Waveform“. Bei synchronem LCD-Clock kann das LCD-Modul auch im Idle- und im Power-Save-Mode betrieben werden, bei asynchronem LCD-Clock dazu im ADC-Noise Reduction-Mode.

Fazit

Durch die genannten Verfahren kann der Strombedarf mikrocontrollergesteuerter Schaltungen erheblich vermindert werden, wobei viele Maßnahmen noch nicht einmal die Performance einschränken oder die Systemkosten erhöhen – und wo letzteres doch zutrifft, amortisiert sich der Mehraufwand mit der Zeit zumeist vielfach. Bei weltweit milliardenfacher Verbreitung von Mikrocontrollern ist schließlich auch der Umweltschutzaspekt nicht zu vernachlässigen.

Literaturnachweise

- Atmel Application Notes (http://www.atmel.com)
- AVR040: EMC Design Considerations
- AVR042: AVR Hardware Design Considerations
- AVR105: Power Efficient High Endurance Parameter Storage in Flash Memory
- AVR1010: Minimizing the power consumption of Atmel AVR XMEGA devices
- AVR1509: Xplain Training – Low Power Training

[1] AS3932 http://www.ams.com/eng/Products/RF-Products/Low-Frequency/AS3932

[2] Magnetic Encoders http://www.ams.com/eng/Products/Magnetic-Encoders

[3] Atmel Microcontroller http://www.atmel.com/products/microcontrollers

[4] Atmel ATA5580 http://www.atmel.com/devices/ATA5580.aspx

[5] MAS http://www.mas-oy.com

[6] RTC http://www.mcrystal.ch/Products/Real-Time-Clock-Modules.aspx

[7] Jedes Mikrojoule zählt, Andreas Riedenauer, Elektor 2/2010 http://www.elektor.de/jahrgang/2010/marz/jedes-mikrojoule-zahlt.1255269.lynkx

[8] Innovative Techniques for Extremely Low Power Consumption with 8-bit Microcontrollers, A.M. Holberg und A. Saetre (Atmel White Paper) http://www.atmel.com/technologies/lowpo wer/default.aspx

IPv6 Funkübertragung mit dem Merkurboard

Harald Pichler <harald@the-develop.net>

In diesem Artikel wird durch Einsatz von zwei Merkur-Boards und dem Betriebssystem Contiki-OS für Mikrocontroller gezeigt, wie ein batteriebetriebener Funkknoten als Lichtschalter und ein weiterer Funkknoten als Lichtaktor eingesetzt werden kann. Durch Drücken eines Tasters wird die LED des Merkurbord an- und ausgeschaltet und der Befehl an das zweite Modul weitergeleitet, welches ebenfalls seine LED an- und ausschaltet. Die Kommunikation erfolgt über IPv6 auf Basis der IEEE-802.15.4 Funktechnik auch 6LoWPAN genannt. Als Applikationsprotokoll wird COAP eingesetzt.

Einleitung

Der Funkstandard IEEE-802.15.4 [1] wurde speziell für batteriebetriebene Sensoranwendungen mit niedrigen Datenraten im Bereich von 20 bis 250 kbit/s spezifiziert, wodurch mit diesen Funkknoten eine Batterielebensdauer von mehreren Jahren erreicht werden kann. Im IEEE-802.15.4 Standard sind die physikalische Datenübertragung (PHY-Layer) und der Zugriff auf den Funkkanal (MAC-Layer) beschrieben. Das 6LoWPAN [2] Protokoll setzt auf dem 802.15.4-MAC-Layer auf und ermöglicht die Kommunikation von Funkknoten untereinander über mehrere Knoten hinweg (Mesh) und über einen Router auch mit dem lokalem IP Netz oder dem Internet.

Merkurboard

Das Merkur Board ist ein Arduino- und Contiki-OS-kompatibles Mikrocontrollerboard mit integriertem 2.4Ghz IEEE802.15.4 Funksender. Der Arduino-Bootloader ist vorprogrammiert. Das Funkteil unterstützt alle auf 802.15.4 basierenden Funkprotokolle wie 6LoWPAN, Zigbee und RF4ce.

Technische Details:

- 8-Bit AVR 16Mhz Microcontroller
- 2.4 Ghz Transceiver IEEE 802.15.4
- IPv6/6LoWPAN Support
- 128k Flash, 16k RAM, 4k EEPROM
- SPI, i2c, Serial, GPIO, JTAG auf Stiftleisten

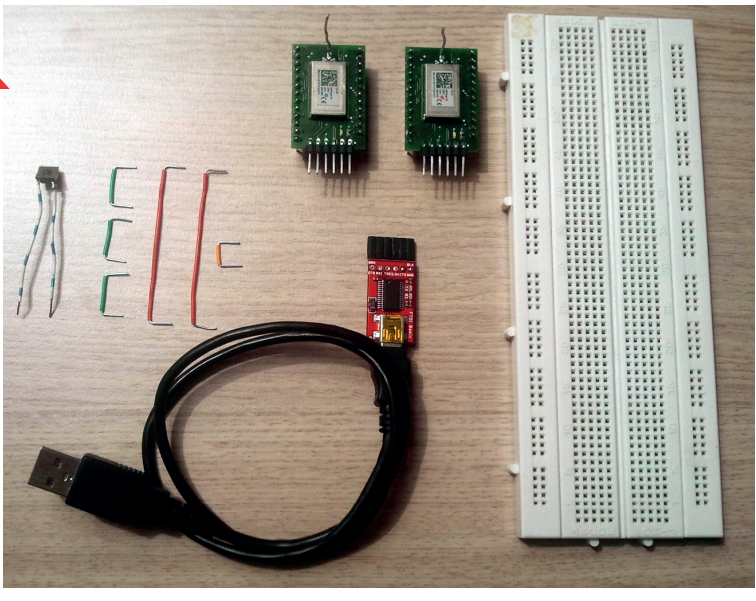


Abb. 1: Benötigte Bauteile

Das Projekt

Unsere Aufgabenstellung: Nach dem Drücken des Tasters geht die LED auf unserem Client-Knoten an oder aus, danach wird ein Coap Befehl über IPv6 (6LoWPAN) an den Server-Knoten gesendet und deren LED geht ebenfalls an oder aus.

Für unseren Aufbau benötigen wir zwei Merkur Boards, ein Steckbrett, einen 3,3V Serial-USB Wandler und einen Taster (Abb. 1).

Die Kommunikation

Heutige Webanwendungen basieren auf Techniken wie HTML, XML oder Soap und werden über HTTP und TCP übertragen. Dies resultiert aber in einer Paketgröße von hunderten von Bytes bis zu mehreren Kilobytes. P2P-Anwendungen auf 6LoWPAN Basis sollten Protokolle auf UDP Basis benutzen und die Daten wenn möglich komprimiert versenden. Eine interessante Alternative ist das COAP- Protokoll.

Das von der IETF entwickelte „Constrained-Application-Protocol“ (COAP) bildet eine „Representational State Transfer“ (REST) -ähnliches Interface in der Anwendungsschicht ab. Es wurde auf niedrige Bandbreiten optimiert und kommt mit einer geringeren Komplexität als HTTP basierte REST-Schnittstellen aus. COAP verwendet Mechanismen wie HTTP-Ressourcen und Abstraktionen wie URIs, RESTful Interaktionen und erweiterbare Header-Optionen. Coap nutzt eine kompakte binäre Repräsentation, die leicht zu analysieren ist. Im Gegensatz zu HTTP, das über TCP kommuniziert, benutzt Coap das UDP Protokoll.

Zur Vereinfachung und Minimierung des Aufwandes zur Programmierung der Firmware des Funkknotens wird ein netzwerkfähiges Mikrocontroller-Betriebssystem eingesetzt. Die Wahl fiel auf Contiki-OS 2.6. Es ist ein in C geschriebenes und unter einer BSD Lizenz stehendes, modular aufgebautes Betriebssystem, welches auf verschiedene Mikrocontroller portiert wurde. Alternativ könnte auch TinyOS eingesetzt werden.

Contiki-OS bietet verschiedene Coap Implementierungen an. Wir entscheiden uns für unser Experiment den Erbium Rest Server/Client mit draft-ietf-core-coap-08 Implementierung zu verwenden.

Wir setzen in unserem „Lichtschalter“-Merkurboard (Abb. 2) einen Coap Client ein. Unser Coap Client sendet an den Lichtaktor die Uri „/actors/toggle“ und wechselt damit den Zustand der Lichtaktor-LED zwischen an und aus.

In unserem „Lichtaktor“-Merkurboard (Abb. 3) setzen wir einen Coap-Server ein und implementieren die nachstehend aufgelisteten Resourcen (Listing 1).

```
Uri Path            Funktionalität
/actors/leds        Schaltet die LED
                    an oder aus
/actors/toggle      Schaltet die LED
                    in den inversen Zustand
/sensors/battery    Spannung in mV im
                    Textformat
/sensors/button     Lesen des
                    Button-Zustandes
/sensors/cputemp    Lesen der CPU Temperatur
```

Listing 1: Resouren für “Lichtaktor”-Merkurboard

Software

Installation der Toolchain unter Debian

```
sudo apt-get install  avr-libc binutils-avr gcc-
avr gdb-avr simulavr avrdude
```

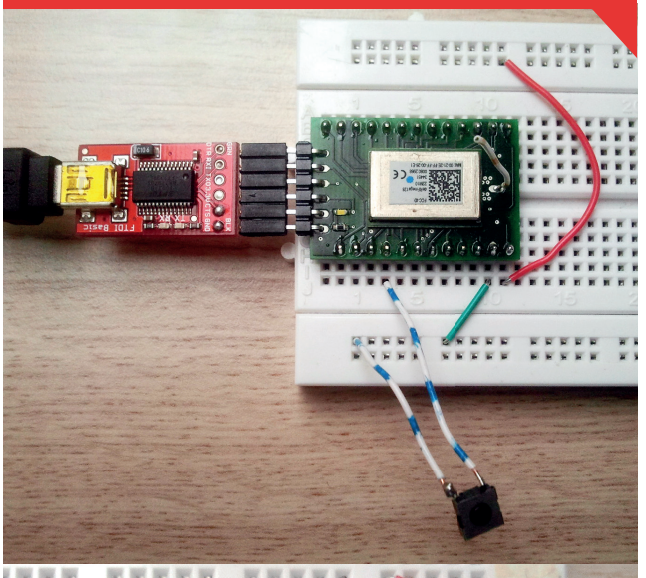
OSD-Contiki besorgen [3].

Entpacke es in deinem Home Directory.

```
cd ~/osd-contiki/examples/osd/er-rest-examp le-mer-
kurboard
```

Durch Anstecken des 3,3V Serial-USB Wandlers an den Pc und das Merkurbaord an den Serial-Wandler kann die Firmware in das Modul eingespielt werden. Der Bootloader meldet sich durch 5 mal Blinken und danach leuchtet die LED für 5 Sekunden. Ist noch keine Firmware eingespielt, beginnt die LED wieder zu blinken. Das Script flash.sh lädt die Server Firmware in den Knoten und das Script flashclient.sh lädt die Client Software für unseren Lichtschalter in das Merkurbaord. Nach dem Einspielen der Firmware in die Module ist unser Aufbau einsatzbereit. Durch Drücken des Tasters leuchtet die LED am Client-Board, überträgt den Coap Befehl Toggle an das Serverboard und auf diesem beginnt die LED ebenfalls zu leuchten.

Abb. 2: Aufbau des „Lichtschalter“-Merkurboards



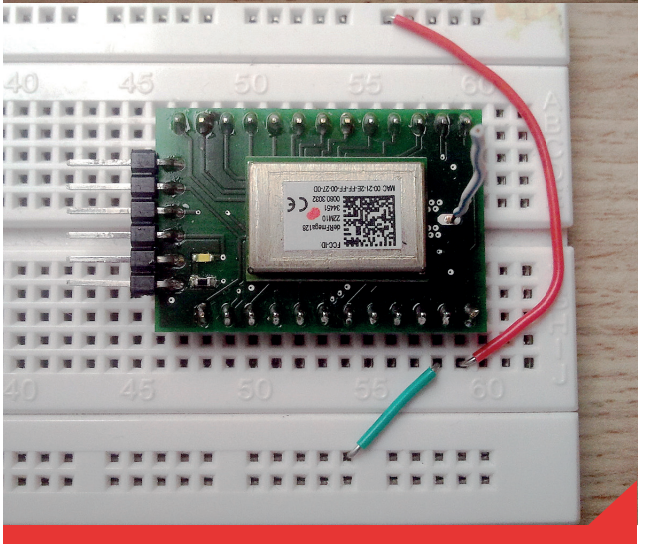


Abb. 3: Aufbau des „Lichtaktor“-Merkurboards

Wer die Firmware compilieren und oder anpassen möchte findet im OSD-Wiki [4] Anleitungen dazu.

Ausblick

Das vorgestellte Beispiel-Projekt zeigt, wie mit geringem Aufwand eine einfache Lichtsteuerung realisiert werden kann. Durch den Einsatz von Funkmodulen und dem Betriebssystem Contiki-OS reduziert sich der Aufwand bei der Hardware- und Software-Entwicklung. Das vorgestellte Merkur-Board wird in kommenden Beiträgen als Router-Stick zum Einsatz kommen. Die Software zum Artikel findet man auf Github [5].

Bezugsquellen und Literatur

Merkur-Board

- http://shop.embedded-projects.net/
- http://www.coxcoon.at/
- http://www.hackerspaceshop.com/

[1] IEEE_802.15.4 http://de.wikipedia.org/wiki/IEEE_802.15.4

[2] 6LoWPAN http://de.wikipedia.org/wiki/6LoWPAN

[3] OSD-Contiki https://github.com/osdomotics/osd-contiki/archive/master.zip

[4] OSD-Wiki http://osdwiki.open-entry.com/doku.php/de:projekte:merkur

[5] Software https://github.com/osdomotics/osd-contikiw

Controlled Impedance for Edge-Coupled Coated Microstrip with Cadsoft Eagle V6

Steffen Mauch <steffen.mauch@gmail.com>

Abstract – Due to the fact that the rise time of signals are getting faster and faster and furthermore the electrical length is nowadays often significant less than the propagation delay, the impedance of a trace on a PCB is not longer simply negligible or can be considered as simply a resistance. Any discontinuity along a signal path will cause reflections and this signal degeneration can result in system failures. The major layout tools as Mentor, Altium to name a few have built in support for controlled impedance. Due to the fact that Eagle does not have built in support for controlled impedance does not mean that it is not possible. In high-speed interfaces such as CameraLink, PCIe or even USB, controlled impedance for the differential signals is highly recommended or even necessary and this can even be realized with Eagle. Index Terms— eagle, cadsoft, controlled impedance, differ- ential signals, LVDS, high-speed signals, printed board circuit .

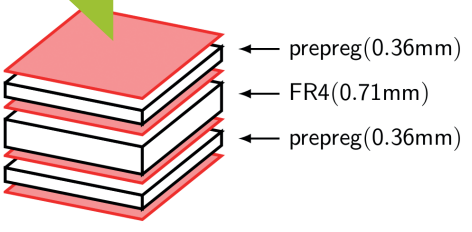
Introduction

In this article, the aim is to show that it is possible to route differential signals in Eagle whereas the impedance of the wire is controlled. Of course the use of other layout tools is recommended, but sometimes one has to route just a few wires which are critical or you don't have the time to learn another tool for a single printed board circuit (PCB). Furthermore, most of the work remains the same, independent of the used layout tool but one has to admit that some checks could not be done with Eagle so far.

Edge-coupled coated microstrip

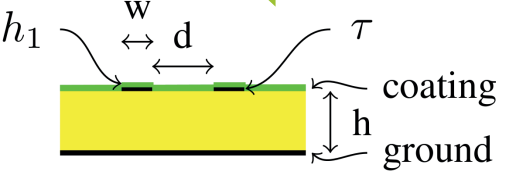
Impedance controlled circuits are feasible in different techniques, concerning the layer stacking, as well as their wire structure. In Fig. 1 a typical layer stack for a four layer printed circuit board is shown. This layer stack is in this configuration not very well suited for edge-coupled coated microstrip with 100 Ohm due to the relative thick prepreg. The resulting wire width and distance is with this setup very huge for 100 Ohm. The red layers are the coper planes and the prepreg consist normally of two 180 µm prepreg. If a total width of 1.59, mm for the PCB is needed then either use a thicker core such that the prepreg could be thinner or change to six layers. Most inexpensive manufacturer do not have thicker cores or this special request is very expensive. Therefore it could be cheaper to use a six layer standard setup and this is in general better suited for edge-coupled coated microstrip with 100 Ohm. This article only covers edge-coupled coated microstrip, which is drawn in Fig. 2. A high impact on the impedance has the thickness of the dielectric followed by the width of the wire. To solve the impedance based on given parameters, one has to solve the Maxwell equations. This is usually done with special programs, which use approximations or solve the equations with finite-elements methods, see [1].

Fig. 1: A typical layer stack for a four layer printed circuit board



The typical parameters for edge-coupled coated microstrip are: h_1 thickness of the soldering stop, w width of the trace, d distance between the differential pair, τ thickness of the trace, h height and ε_r of the dielectric material.

Fig. 2: Edge-coupled coated microstrip



For more specific information about the different microstrip realizations as well as stripline, see i.e. [2].

Remark:

The results from the different solvers could differ quite a lot, due different assumptions and requirements. Best way is to contact the PCB producer, because they will have usually quite a lot of practical experience. Due to fabrication tolerances, the calculated value could differ by ±10%, which is standard in practical applications. But nevertheless this is no excuse to use uncertain parameters for the impedance calculation, because the error will most likely accumulate.

HowTo

As differential routing feature was established in Eagle 6, Eagle is now capable of routing differential signals simultaneously. Nevertheless controlled impedance is not integrated till now. However, with some creativity it is not very difficult to use the existing features for controlled impedance. As is shown in Fig. 3, the layer stack could be manipulated with the drc command. This setup is the default six layer setup from www.eurocircuits.com. The resulting impedance due to layer setup, distance and

material could be solved with special programs. One program, which is very easy to handle and delivers good results, is Saturn PCB Design Toolkit [3]. With the given thickness, layer stack (Fig. 3) and a common ε_r = 4.3, an impedance of 100 Ohm could be achieved with a wire width of 9 mil and a clearance between the two signals of 9 mil. For each wire the impedance is then 60 Ohm, which are the recommended values for PCIe.

Eagle interprets signals with the same name, but a suffix of _N and _P , as differential signals. Therefore the differential signals are renamed in the schematic to meet the requirement. When differential signals are selected for routing, Eagle automatically tries to route the signals in parallel. The clearance between the differential signals are defined by specifying a new class with the 'class' command. As in Fig. 4 shown, the values are entered for the '100ohm' class. Afterwards when the class is assigned to the signals, the routing command automatically maintains the clearance.

In Fig. 5 is the maximum length difference of the differential pairs specified, as well as the gap factor for the meanders to accommodate the length of the pair. For the routing of PCIe, it is recommended to have a maximum length difference of 5 mil. In general this setting could also be very helpful, because all the differential signals should ideally have the same length due to a propagation time preferably equal.

In Fig. 6 an exemplary layout for PCIe 2.0 x4 is given. Layer two and five are both defined as ground layers for having the possibility of edge-coupled coated microstrip at layer one and six.

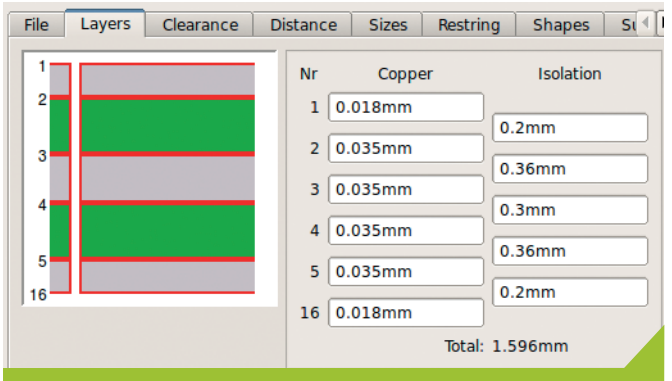


Fig. 3: Default six layer setup from eurocircuits

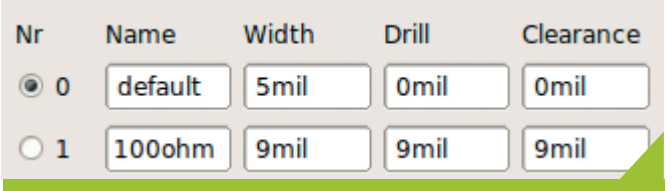


Fig. 4: Specifying a new class

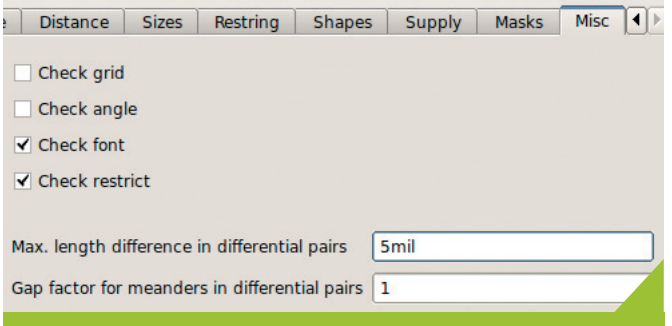


Fig. 5: A maximum length difference of 5 mil is recommended

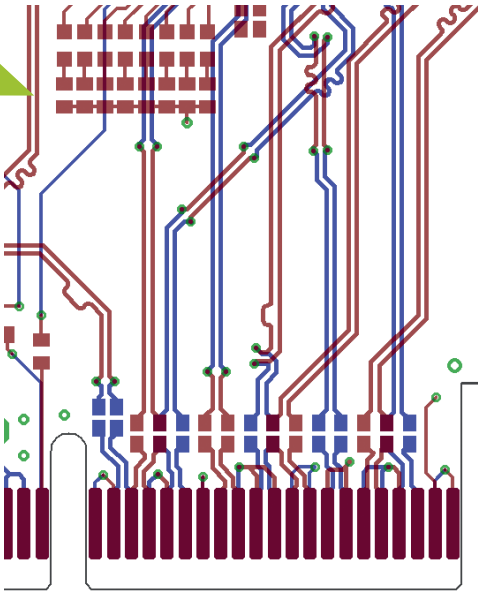


Fig. 6: Layout for PCIe 2.0 x4

Conclusion

As shown in the article, controlled impedance is well manageable even with Eagle without heavy work. Important to remember is, that it is not sufficient to simply use differential routing but also that the parameter, the layer stack as well as the right PCB material is correctly defined and checked that the producer could manufacture the board. A PCB with controlled impedance is in almost all cases not as cheap as a 'standard' one, due to the fact that the manufacturer has to meet the given specifications, i.e. the prepreg should not change due to possible differences in ε_r. Claiming to measure the impedance after production is often possible but this extra step will increase the price. Some manufacturer are specifying the distances as well as the ε_r exactly and even adjust an existing layout to meet the requirements. Needless to say, compared to a standard approach the overhead for controlled impedance is considerably higher but most high speed interfaces do need specific treatment of the impedance due to their rising time and data rate.

Bibliography

[1] http://dcchapters.ipc.org/assets/pnw/presentations/20030925_Impedance.pdf

[2] http://www. vps.nu/img/image/Docs/INTRO.PDF

[3] http://saturnpcb.com/pcb_toolkit.htm

Praktikum: CAN-Bootloader

Pamela Bogner <bogner@hm.edu>, Konstantin Werner <kw@lessmueller.de>

In meinem ersten Artikel USB zu CAN Adapter (Embedded Projects Journal, Ausgabe 15) schuf ich die Grundlage für meinen jetzigen Artikel. Diesmal war meine Aufgabe einen CAN Bootloader für den Mikrocontroller LPC11C24 zu schreiben.

Die Hardware

Die Hardware besteht wieder aus einer USB zu SPI Bridge, einem MCP2515 und einem MCP 2551. Auch diesmal können die vorgefertigten Enwicklungsboards von FTDI und Microchip verwendet werden. Hinzu kommt ein LPC Entwicklungsboard von NXP (Abbildung 1 und 2).

Der Mikrocontroller LPC11C24 [1] befindet sich auf dem LPCXpresso LPC11C24 Board von NXP. Um dies mit dem MCP2515 PictailTM Demo Board von Microchip zu verbinden, werden auf die herausgeführten Pins Pinleisten gelötet. Danach können die beiden Entwicklungsboards mit Jumperkabel verbunden werden (siehe Abb.1). Auf dem LPCXpresso LPC11C24 Board müssen P0_1 und P0_3 mit GND verbunden werden. Diese Pins aktivieren den CAN Bootloader. Den Strom bekommt der Mikrocontroller über den USB Anschluss.

Alternative Hardware

Um Platz zu sparen wäre auch eine Platine denkbar, auf der MCP 2515 und MCP2551 so verbunden sind, wie im Datenblatt [2] nur ohne PIC16F676 und einer geeigneten Buchse statt des Pinheaders. In der passenden Größe und mit entsprechender Platzierung der Anschlüsse könnte die Platine direkt auf das FTDI Mini-Module aufgesteckt werden.

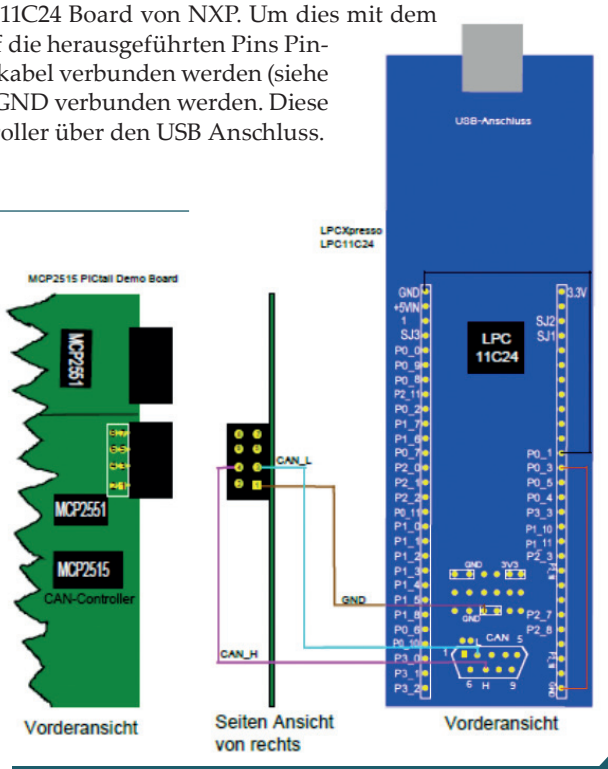


Abb. 1: Verdrahtung der beiden Entwicklungsboards

Der Bootloader

Der LPC11C24 ist ein sehr vielseitiger Baustein, der über ein integriertes CAN-Interface inklusive Transceiver verfügt. Im ROM des Mikrocontrollers befindet sich standardmäßig ein Bootloader, der über CAN und UART ansprechbar ist.

Bei jedem Starten oder Reset springt der Mikrocontroller automatisch in den Bootloader, vorausgesetzt die folgenden Pins sind richtig gesetzt: Liegt PIN0_1 auf GND, so startet nach einem Reset der Bootloader. Zusätzlich muss PIN0_3 auf GND gesetzt werden, um die CAN-Variante des Bootloaders auszuwählen.

Die CAN Nachricht für den Bootloader ist folgendermaßen aufgebaut: Die ID ist immer 0x67D und die Datenlänge immer 8 Byte. Die Antwortnachricht, die auf jeden Befehl folgt, hat die ID 0x5FD und ebenfalls 8 Datenbytes. Die Datenübertragungsrate beträgt 100kb/s. Die 8 Datenbytes müssen für den Bootloader im Format des Service Data Object (SDO) [3] vorliegen. Tabelle 1 zeigt, wie diese aufgebaut sind. Hier wird im ersten Byte definiert, um was für eine Nachricht es sich handelt, um eine zum hoch- bzw. herunterladen der Daten, ob ein Segment oder gleich mehrere gesendet werden sollen (cs) und wie viele Daten gesendet werden (n, e, s). Im zweiten und dritten Byte wird der Index der Nachricht versendet, mit dem 4. Byte der Subindex. Die restlichen 4 Bytes bleiben für die Daten übrig, wie z.B. eine Adresse.

2 Bits	1 Bit	2 Bits	1 Bit	1 Bit	2 Bytes	1 Byte	4 Bytes
Command Specifier (cs)	reserviert 0	n	e	s	Index	Sub-index	Daten

Tabelle 1: Wird genutzt um Befehle zu übertragen oder um eine segmentierte Datenübertragung einzuleiten.

3 Bits	1 Bit	3 Bits	1 Bit	7 Bytes
cs	t = toggle	n	l = last	Daten

Tabelle 2: Wird verwendet, wenn eine stückweise Übertragung der Daten stattfinden soll.

Tabelle 2 zeigt die stückweise Übertragung von Daten. Im ersten Byte muss das 4. Bit immer getoggelt werden und am Ende muss im ersten Byte das letzte Bit auf 1 gesetzt werden, um die segmentierte Datenübertragung abzuschließen. In den restlichen 7 Bytes stehen die Daten.

Die beiden Formate werden in der Software deutlich.

Die Software

Für die LPC Bauteil Reihe existiert schon eine UART Bootloader Software[4] von Martin Maurer, diese diente als Vorbild für das CAN Bootloader Programm. Für die CAN Kommunikation wird die gleiche Software wie im letzten Artikel genutzt. Hinzugekommen sind drei neue Ebenen, die im Folgenden näher erläutert werden.

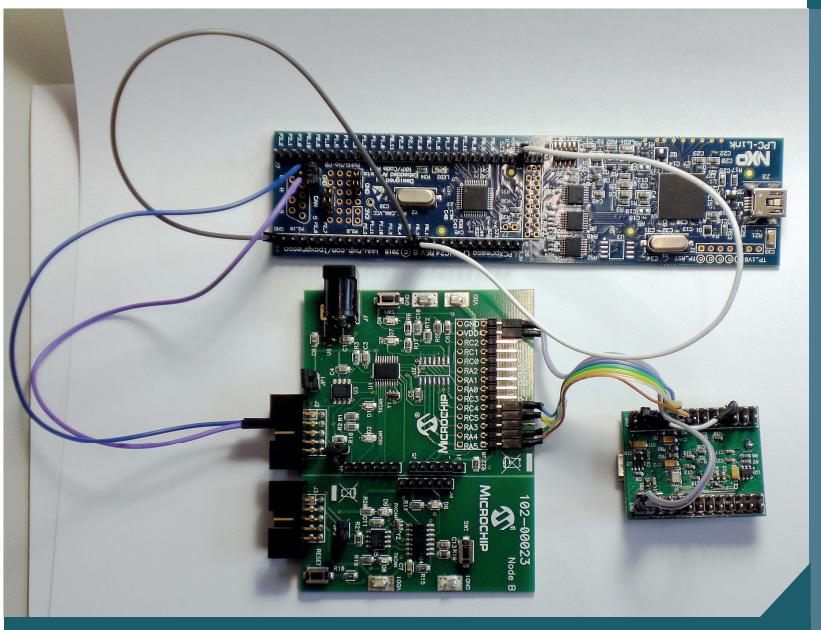


Abb. 2: Aufbau der Schaltung

a. SPO Ebene

Wie im Kapitel Bootloader erwähnt muss die Nachricht im SDO Format gesendet werden. Diese Ebene kümmert sich um das Ver- und Entpacken ins Format der SDO-Befehle, das Verpacken in CAN-Nachrichten sowie das Abfangen von Fehlermeldungen. Zur einfachen Handhabung wurde eine SDO_CMD_-Struktur erstellt, deren Konstruktor das korrekte Formatieren der Daten übernimmt.

Zum eigentlichen Versenden bzw. Empfangen der Befehle über CAN dienen die Funktionen SDO_CMD (siehe Listing 1w), SDO_CMD_REPLY (siehe Listing 2), SDO_SEG und SDO_SEG_REPLY.

Alle vier Funktionen greifen auf die CAN Bibliothek aus dem letzten Artikel zurück, die seit dem jedoch etwas überarbeitet wurde.

Auf jede CAN Nachricht antwortet der Bootloader auch mit einer CAN Nachricht, diese wird mit SDO_CMD_REPLY empfangen. Aus dieser können aufgetretene Fehler gelesen werden. Um die Fehler zu analysieren, wird die Funktion SDO_CHECK_ERROR verwendet.

```
int SDO _ CMD(SDO _ CMD _ *SDO _ CMD)
{
   unsigned char cmd[8];
   cmd[0] = *((unsigned char*) &(SDO _ CMD->flags));
   cmd[1] = (unsigned char) (SDO _ CMD->index);
   cmd[2] = (unsigned char) (SDO _ CMD->index >> 8);
   cmd[3] = SDO _ CMD->subindex;
   for (unsigned char i = 0; i < (4 - SDO _ CMD->flags.n); i++)
                  cmd[4+i] = SDO _ CMD->data[i];
   for (unsigned char i = (4 - SDO _ CMD->flags.n); i < 4; i++)
     cmd[4+i] = 0x00;
   return CAN _ send(0, can _ message _ (SDO _ SERVER _ ID + SDO _ NODE _
ID, 0, 0, 8, cmd));
}
```

Listing 1: Funktion zum Versenden von SDO Befehlen

b. Bootloader Ebene

In dieser Ebene befinden sich die Bootloader-Befehle, so wie sie im User Manual [5] beschrieben werden. Um einen am CAN-Bus angeschlossenen LPC Mikrocontroller zu erkennen, wird der READ_ID-Befehl verwendet. Auf diesen antwortet der Bootloader mit einer typspezifischen ID. Für den LPC11C24 z.B. lautet diese 0x1430102B. Um in den Flash des Mikrocontrollers schreiben zu können, müssen die entsprechenden Befehle zunächst mit dem UNLOCK-Befehl und einem „Passwort“ (0x5A5A) freigeschaltet werden. Der 32kByte große Flash ist in 8 Sektoren á 4kByte unterteilt, die jeweils am Stück gelöscht (ERASE_SECTOR) und vorbereitet (PREPARE_SECTOR) werden. Um das Timing beim Beschreiben des Flashs sicher einhalten zu können, werden die Daten für einen Sektor zunächst per WRITE_TO_RAM in den 8kByte großen RAM geschrieben. Liegt das Abbild eines Sektors vollständig im RAM, so kann es per COPY_RAM_TO_FLASH in den Flash kopiert werden. Diesen Vorgang erledigt der Bootloader vollautomatisch.

```
int SDO _ CMD _ REPLY(unsigned int sdo _ id, SDO _ CMD _ *SDO _ CMD)
{
   unsigned char t = 0;
   signed char buffer;
   can _ message _ reply;
   do {
     CAN _ check _ rx(&buffer, NULL, NULL, NULL);
     Sleep(10);
     switch (buffer) {
            case 0:
            case 1: CAN _ recv(buffer, &reply); break;
            case 2: CAN _ recv(0, &reply);
                  if (reply.id != (sdo _ id))
                  CAN _ recv(1, &reply); break;
     }
     t++;
   } while (t < SDO _ REPLY _ TIMEOUT && (reply.id != (sdo _ id)));
   if (t == SDO _ REPLY _ TIMEOUT) return -1;
```

Listing 2: Funktion zum Empfangen von SDO Befehlen

Anschließend kann noch mit dem COMPARE-Befehl der Inhalt von RAM und Flash verglichen werden, um eine fehlerfreie Über-

tragung sicherzustellen. Alternativ können auch mit der Funktion READ_MEMORY die Daten ausgelesen und dann vom PC mit der gesendeten Datei verglichen werden. Dieser Vorgang wird für alle Sektoren wiederholt, bis das gesamte Programm übertragen wurde. Nun kann der GO-Befehl benutzt werden, um den Bootloader zu verlassen und das neue Programm auszuführen. Beim LPC11C24 ist die Startadresse für den GO-Befehl 0x00000004.

```
    SDO_CMD->flags.raw = reply.data[0];
    SDO_CMD->index = (unsigned short) (reply.data[1]
+ ((unsigned short) reply.data[2] << 8));
    SDO_CMD->subindex = reply.data[3];
    for (unsigned char i = 0; i < 4; i++)
      SDO_CMD->data[i] = reply.data[4+i];
    return 0;
}
```

Fortsetzung: Listing 2

Die Funktionen READ_BOOT_CODE und READ_SERIAL_NUMBER wurden zwar implementiert, aber bisher nicht verwendet.

c. ISP Ebene

Die ISP Ebene fasst die Funktionen des Bootloaders zusammen um Aufgaben wie z.B. das Laden eines neuen Programms oder das Löschen des Flashs einfacher durchführen zu können. Nachdem mit ISP_Connect die Verbindung zum Mikrocontroller hergestellt ist kann nach dem Entsperren durch ISP_Unlock mit den folgenden Funktionen gearbeitet werden:

- ISP_write_firmware: In dieser Funktion werden Daten aus einer Binärdatei gelesen, und wie im Bootloader-Abschnitt erwähnt in den RAM geschrieben, von dort aus in den Flash kopiert und am Schluss verglichen.
- ISP_run: Hiermit wird der Bootloader verlassen und das übertragene Programm gestartet.
- ISP_erase_chip: Mit dieser Funktion wird der gesamte Flash gelöscht.
- ISP_verify: In dieser Funktion werden die Daten aus der Datei gelesen und mit den Daten im RAM verglichen.

Am Schluss muss der CAN Bus mit ISP_disconnect wieder geschlossen werden

Zum Testen des CAN-Bootloaders eignet sich z.B. das „Blinky"-Programm, das NXP zusammen mit der eclipsebasierten LPCX-Presso-IDE kostenlos zur Verfügung stellt [6]. Eine erfolgreiche Übertragung des Programms zeigt sich dann als blinkende LED auf dem Evaluationboard.

d. User Ebene

Um die Verwendung des Bootloaders etwas benutzerfreundlicher zu machen, wurde ein Konsolenprogramm geschrieben, das alle wichtigen Parameter wie die Typ-ID, den Dateinamen, etc. als Parameter entgegen nimmt und oft benötigte Aktionen ausführt. Ein Blick in main() sollte hier als Erklärung ausreichen.

Starten des Bootloaders per IAP Befehl

Um nicht bei jedem Aufspielen eines neuen Programms die Jumper für den Bootloader setzen und einen Reset ausführen zu müssen, kann der Bootloader auch aus dem User-Programm heraus aufgerufen werden. Das erleichtert vor allem Updates in schon verbauten Chips.

Um den Bootloader aus dem User-Programm heraus aufzurufen, wird ein so genannter IAP-Command verwendet. Dieser wird im Programm so eingebaut, dass er z.B. per CAN-Nachricht aufgerufen werden kann.

Leider ist die Dokumentation zur Verwendung der IAP-Befehle im Datenblatt von NXP nicht sehr ausführlich.

Ein einfaches Ausführen des IAP-Befehls 57 (REINVOKE_ISP) führt nur zum Absturz des Mikrocontrollers.

Nach einiger Suche in den einschlägigen Foren wurde jedoch eine Lösung ausfindig gemacht [7] (siehe Listing 3). Diese Funktion bereitet den Controller so vor, dass nach dem Aufruf des IAP-Befehls die richtigen Bedingungen für den Bootloader herrschen. Die komplette Software ist hier zu finden [8].

```
void IAP_InvokeBootloader(void) {
   static unsigned long command[5] = {57, 0, 0, 0, 0};
   static unsigned long result[4]; __disable_irq();
   // Make sure 32-bit Timer 1 is turned on before calling ISP
   LPC_SYSCON->SYSAHBCLKCTRL |= 0x00400;
   // Make sure GPIO clock is turned on before calling ISP
   LPC_SYSCON->SYSAHBCLKCTRL |= 0x00040;
   // Make sure IO configuration clock is turned on before calling ISP
   LPC_SYSCON->SYSAHBCLKCTRL |= 0x10000;
   LPC_SYSCON->SYSAHBCLKDIV = 1;       // AHB clock divider must 1:1
   // Disable the PLL
   LPC_SYSCON->MAINCLKSEL = 0;         /* Enable IRC Oscillator */
   LPC_SYSCON->MAINCLKUEN = 0x01;      /* Update MCLK Clock Source */
   LPC_SYSCON->MAINCLKUEN = 0x00;      /* Toggle Update Register */
   LPC_SYSCON->MAINCLKUEN = 0x01;
   while (!(LPC_SYSCON->MAINCLKUEN & 0x01));  /* Wait until updated */
   command[0] = IAP_CMD_REINVOKE_ISP;      // Reset the stack pointer
to point to the top of the stack minus the 32 byte for IAP
   // For LPC11C14 where top of RAM is 0x1000_2000; MSP = (0x1000_2000
- 0x20) = 0x10001FE0
   _set_MSP(0x10001FE0);
   // The Cortex M0 needs to synchronize the CPU cache
   __ISB();
   iap_entry(command, result);
}
```

Listing 3: Angepasster Code aus [7]

Fazit

Ich freue mich, dass das Programm nun ohne Probleme funktioniert und bin auch sehr dankbar für die Hilfe im mikrocontroller.net Forum und auch für die meines Kollegen (Konstantin Werner, Praktikumsbetreuer).

Die Software ist sehr flexibel einsetzbar. So habe ich z.B. die unteren Ebenen durch die CAN-Befehle des LPC11C24 ausgetauscht und konnte das Bootloader-Programm fast 1-zu-1 auf einen LPC11C24 portieren. So kann ein LPC verwendet werden um einen anderen zu programmieren.

Literatur

[1] http://www.nxp.com/documents/data_sheet/LPC11CX2_CX4.pdf

[2] http://ww1.microchip.com/downloads/en/DeviceDoc/51572a.pdf

[3] http://www.lpcware.com/content/nxpfile/an11238-lpc11cxx-canopen-network-demo-0

[4] http://sourceforge.net/projects/lpc21isp/files/

[5] http://www.nxp.com/documents/user_manual/UM10398.pdf

[6] http://ics.nxp.com/lpcxpresso/

[7] http://knowledgebase.nxp.com/showthread.php?p=14214

[8] https://www.dropbox.com/s/9nj83y2t1djoeyn/CAN_Bootloader.zip

□

Wetterballon mit Gnublin

Dietmar Sach <dietmar.sach@hs-augsburg.de>

Patrick Vogt <patrick.vogt@hs-augsburg.de

Von der Faszination des Alls getrieben und inspiriert durch andere Wetterballonprojekte von Entwicklern aus Nah und Fern, beschlossen wir am Anfang des Sommersemesters 2013 im Fach Embedded Linux ein Wetterballonprojekt zu starten. Unser primäres Ziel ist es, möglichst spektakuläre Bilder und Videos aus großer Höhe zu machen. Des Weiteren wollen wir während des gesamten Flugs Temperaturmessungen durchführen. Mit diesen Daten lässt sich der Verlauf der Temperatur mit steigender Höhe herausfinden.

Die Reise des Wetterballons

Der Wetterballon (siehe Abb. 1) beginnt nach dem Start mit dem Steigflug. Ein GPS- Empfänger zeichnet die Positionen periodisch auf und übermittelt diese automatisch. Nach etwa 1 – 2 Stunden erreicht der mit ca. 1,5 m^3 Helium gefüllte Ballon die erzielte Höhe von 30.000 m in der Stratosphäre. Durch den niedrigen Luftdruck in dieser Region dehnt sich der Ballon bis enorm aus und wird platzen. Danach geht die Nutzlast mit den Ballonresten in den freien Fall über. Ein Fallschirm soll sich nun entfalten, um den Flug abzubremsen. Über die vor und nach dem Aufprall übermittelten GPS-Koordinaten soll die Konstruktion wieder gefunden werden.

Abb. 1: Der Wetterballon kurz vor dem Start

Die Transport-Box

Um die Nutzlast am Ballon zu befestigen, wird ein Behälter benötigt, der sowohl stabil, als auch möglichst leicht sein und isolierend wirken soll. Wir entschieden uns für eine quaderförmige Kiste (Abb. 2) aus Platten aus besonders stabilem Polystyrol (Styropor), das auch zur Wärmeisolierung von Gebäuden verwendet wird. Da uns eine Wandstärke von 35 mm nicht ausreichend erschien, verwendeten wir für jede Wand jeweils zwei Platten übereinander, was eine gesamte Wandstärke von 70 mm ergab. Verbunden wurden die Platten mit einem kälteunempfindlichen, extrem stabilen PU-Schaum-Kleber. Eine zusätzliche Stabilisierung oder Verstärkung der strukturellen Integrität war durch das geringe Gewicht der Nutzlast nicht erforderlich. Nach unseren Schätzungen bleibt die Transport-Box demnach bei bis zu 5-6 kg Last stabil, die tatsächliche Last beträgt jedoch nur 1 – 1,5 kg.

Abb. 2: Die Transportbox wird zur zusätzlichen Wärmeisolierung mit Styroporkugeln aufgefüllt

Hardware

Das Gnublin-Board ist eine ARM-basierte Ausbildungsplattform für Embedded Linux, die aufgrund seines geringen Stromverbrauchs und den vielen Erweiterungen inklusive einer Programmierumgebung gerne als Ersatz für typische Mikrocontroller-Anwendungen genutzt wird. Damit eignet es sich ideal für unser Projekt, in dem es eine Zahl von Sensoren steuern und als interaktive Schnittstelle für den Wetterballon dienen soll.

Bei dem Einsatz im Wetterballon werden Akkus mit 7,2 V über die Anschlussklemmen angeschlossen. Am USB-Device-Port werden GPS- Sensor, UMTS-Surfstick und die Webcam angeschlossen. Am ADC-Anschluss (Analog-Digital-Wandler) wird die Wheatstone'sche Messbrücke mit dem PT1000-Temperatursensor angeschlossen.

Um während des Fluges Bilder aufzuzeichnen, wird eine handelsübliche Webcam verwendet. Die Entscheidung fiel auf die Logitech C170, weil von ihr akzeptable Bilder produziert werden und diese Webcam am Gnublin-Board bereits erfolgreich getestet wurde.

Um die Temperatur außerhalb der Styroporbox zu bestimmen kommt die Wheatstone'sche Messbrücke (Abb. 3)mit einem PT1000-Temperatursensor zum Einsatz, deren Funktionsweise hier kurz erläutert werden soll. Die Wheatstone'sche Messbrücke besteht im Wesentlichen aus einer Anordnung von vier Widerständen und einem Spannungsmessgerät. Die Widerstände R1, R3, R4 sowie der PT1000-Temperatursensor lassen die angelegte Spannung abfallen. Die Widerstände haben 1 kOhm , was dem Widerstand des Temperatursensors bei 0° Celsius entspricht. Der Spannungsabfall beim PT1000 ist von der Temperatur abhängig. Je niedriger die Temperatur wird, desto geringer wird der Widerstand. Die Spannungen beider Seiten werden dann verglichen und im Gnublin- Board über einen Analog-Digital-Wandler in einen Digitalenwert umgerechnet. Der digitale Wert kann dann durch Software in eine „Temperatur" umgewandelt werden.

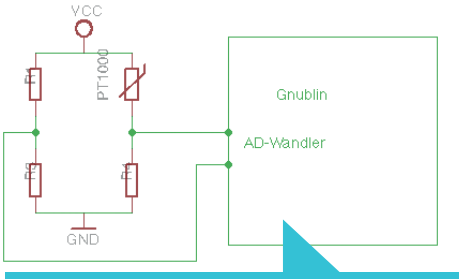


Abb. 3: Messbrücke PT1000

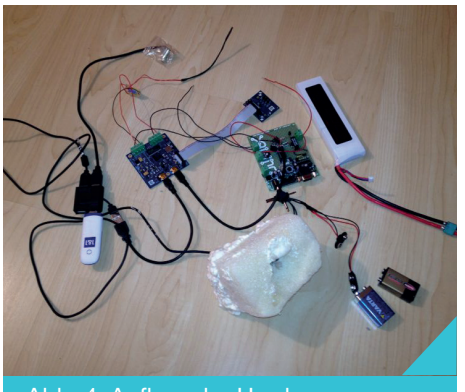


Abb. 4: Aufbau der Hardware

Ein UMTS-Stick baut eine Verbindung zum Mobilfunknetz auf und bietet dadurch Zugriff auf Kommunikationsfunktionen wie UMTS- Datennetz, SMS, MMS oder Telefonie. Für die Kommunikation mit dem Gnublin-Board wird der UMTS-SurfStick ZTE MF190 verwendet. Dieser kann nach einigen kleineren Anpassungen am Linux-Kernel am Gnublin-Board verwendet werden. Der Betrieb des UMTS-Moduls benötigt besonders viel Energie, weshalb von dem Stick keine Datenverbindung aufgebaut werden soll. Stattdessen findet die Kommunikation über periodisch abzurufende und zu sendende SMS-Kurznachrichten statt. Der Stick benötigt eine SIM-Karte für den Betrieb. Dafür wurde eine günstige Prepaid-Karte für 10 Euro mit 3000 Frei-SMS angeschafft.

Für die Ermittlung der Position wird ein GPS-Empfänger benötigt. Dieser sollte möglichst günstig, leicht und stromsparend sein. Daher fiel die Wahl auf den Empfänger Gms-D1 von GlobalTop mit MediaTek MT3329 Chip.

- Stromverbrauch: 37 – 48 mA
- Gewicht: 2 g

Stromversorgung

Die Stromversorgung stellt eine besonders kritische Komponente dar, von welcher sämtliche Systeme abhängen. Zudem reduziert die extreme Kälte in der Stratosphäre von ca. -50° Celsius enorm die Akkulaufzeit. Es wird eine Reduzierung auf schlimmstenfalls 30 – 50 Prozent der Kapazität erwartet. Deshalb ist es ratsam, den Akku zu überdimensionieren.

Es kommen verschiedene Akku-Technologien in Frage: Nickel-Metallhybrid (NiMH), Lithium-Ionen oder Lithium-Polymer. Generell lässt sich sagen, dass NiMH-Akkus kostengünstiger, dafür aber deutlich schwerer und kälteempfindlicher als Lithium-Ionen-Akkus sind. Zudem haben Lithium-Ionen-Akkus nur eine geringe Selbstentladung. Da Lithium-Polymer-Akkus eine etwas höhere Energiedichte als herkömmliche Lithium-Ionen-Akkus aufweisen, fiel die Wahl auf diese. Zur Bestimmung der Akku-Kapazität war es erforderlich, den Stromverbrauch der Geräte zu bestimmen (Tabelle 1).

Gerät	Stromverbrauch
Gnublin	50 mA
GPS-Modul	50 mA
UMTS-Stick	150 mA
Kamera	200 mA
Messbrücke	0,6 mA
Gesamt:	400 mA

Es muss also mit einem maximalen Verbrauch von 400 mA gerechnet werden. Der Akku muss des Weiteren die benötigte Leistung über die gesamte Flugdauer hinweg bereitstellen. Die Flugdauer ist von vielen Faktoren abhängig und daher unmöglich vorher zu sagen. Anhand der Erfahrungen anderer Wetterballon-Teams können wir die Flugdauer auf eine Zeit zwischen zwei und sechs Stunden eingrenzen. Geht man von einem Verbrauch von konstant 400 mA und einer Flugdauer von 6 Stunden aus, ergibt sich folgende Rechnung:

Kapazität = 400mA * 6h = 2400 mAh

Demnach müsste ein Lithium-Polymer-Akku mit mindestens 2400 mAh Kapazität eingesetzt werden. Da aufgrund der Kälte mit einem Verlust von 50 - 70 Prozent der Kapazität zu erwarten ist, sollte der Akku demnach eine Kapazität von 4800 mAh – 8000 mAh aufweisen. Aus Gewichts- und Kostengründen haben wir uns für einen Lithium-Polymer-Akku mit einer Kapazität von 5000 mAh entschieden.

Abb. 5: Die Transportbox wird für den Abflug vorbereitet

Abb. 6: Aufblasen des Ballons mit zwei Helium Gasflaschen

Software

Im Verzeichnis /dev/init.d/ wird beim Booten des Boards ein Skript ausgeführt, das alle wichtigen Programme für das Projekt startet. Im Start-Block des Skripts werden der Watchdog-Timer, die Anwendung zur Messung und Speicherung der Temperatur, die Anwendung zur

Aufnahme von Fotos und Videos sowie die Anwendung für die Kommunikation und Ortung gestartet. Da diese Funktionen essentiell für den Betrieb und die Wiederauffindung der Konstruktion sind, gelten hier ähnliche Anforderungen an die Betriebsstabilität der Software wie in der Luftfahrt. Die Software muss unter allen Umständen die erforderlichen Funktionen ausführen können und sich ohne Außeneinwirkung im Fehlerfall selbst zurücksetzen oder Fehlerbehandlungen durchführen können. Daher sorgt ein ebenfalls automatisch gestartetes Script dafür, dass ein unvorhergesehen beendetes Script sofort wieder gestartet wird. Sämtlicher Code aller Scripts ist mit Fehlerbehandlungsmaßnahmen versehen.

Der Watchdog-Timer (WDT) ist eine Systemfunktion des LPC3131- Prozessors. Diese kann softwareseitig gestartet werden und wartet dann in einem 10 Sekunden-Intervall auf ein Signal. Bleibt dieses aus, wird ein hard-reset des Gnublin-Boardes vorgenommen. Auf diese Weise soll ein permanentes Einfrieren beziehungsweise Abstürzen des Betriebssystems verhindert werden.

Um den internen Watchdog zu aktivieren wurde ein kleines C-Programm geschrieben, welches den WDT-Treiber anspricht. Der Watchdog präsentiert sich als Gerätedatei, die mit libc-Funktionen geöffnet und geschrieben werden kann.

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

elektor GOLD membership

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

- Sie erhalten **10 Elektor-Hefte** (8 Einzelhefte + 2 Doppelausgaben Januar/Februar und Juli/August) pünktlich und zuverlässig frei Haus.
- **Extra:** Jedes Heft steht Ihnen außerdem als PDF zum sofortigen Download unter www.elektor-magazine.de (für PC/Notebook) oder via App (für Tablet) bereit.
- **Neu & Exklusiv:** Sie erhalten alle 2 Wochen per E-Mail ein neues Extra-Schaltungsprojekt (frisch aus dem Elektor-Labor).
- **Neu & Exklusiv:** Wir gewähren Ihnen bei jeder Online-Bestellung 10% Rabatt auf alle unsere Webshop-Produkte – dauerhaft!
- **Neu & Exklusiv:** Der Online-Zugang zum neuen Community-Bereich www.elektor-labs.com bietet Ihnen zusätzliche Bauprojekte und Schaltungsideen.
- **Extra:** Die neue Elektor-Jahrgangs-DVD (Wert: 27,50 €) ist bereits im Mitgliedsbeitrag inbegriffen. Diese DVD schicken wir Ihnen sofort nach Erscheinen automatisch zu.
- **Extra:** Top-Wunschprämie (im Wert von 30 €) gibts als Dankeschön GRATIS obendrauf!

UMWELTSCHONEND – GÜNSTIG – GREEN

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

elektor GREEN membership

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

```
int wdt_file = open(„/dev/watchdog“, O_WRONLY);
```

Falls der Watchdog nicht gestartet, die Datei also nicht geöffnet, werden kann, wird eine Fehlermeldung per SMS verschickt. Wurde der Watchdog erfolgreich gestartet, wird er alle zehn Sekunden durch ein Script mit einem „Lebenszeichen“ versorgt.

```
ioctl(wdt_file, WDIOC_KEEPALIVE, 0);
```

Bleibt diese Benachrichtung aus, wird das Gnublin-Board neugestartet. Beim Beenden des Watchdog wird die Gerätedatei wieder geschlossen und das Programm beendet.

Um die Bewegungen des Ballons verfolgen zu können, wird der UMTS- Stick zusammen mit dem GPS-Empfänger als Trackingsystem verwendet. Über die USB-Schnittstelle wird vom GPS-Empfänger periodisch die Position abgerufen. Diese Koordinaten werden über den UMTS-Stick per SMS an uns versendet. Natürlich setzt das den Empfang im Handynetz voraus. Ein flächendeckender Empfang ist in dieser Region gegeben und bis zu einer Höhe von circa 1000 Meter möglich. Während des Großteils des Flugs werden wir also keine Positionsdaten erhalten, diese werden aber aufgezeichnet für die spätere Auswertung. Besonders wichtig ist das Trackingsystem für die Zeit im Empfangsbereich vor der Landung und nach der Landung. Vor der Landung kann dadurch der ungefähre Aufschlagsort ermittelt werden. Wenn das System nach der Landung noch funktioniert, wird die Position gesendet, bis der Akku erschöpft ist. Ersatzweise für die GPS-Koordinaten können vom UMTS-Stick per SMS auch Netzwerkinformationen, die die aktuelle Funkzellen-ID und Netzwerk-Code bei Empfang übermittelt werden. Diese Codes können über die Internetplattform OpenCellID1 in grobe Koordinaten umgewandelt werden.

Per SMS können auch Befehle an das Gnublin-Board geschickt werden. Ein Python-Script wertet diese Befehle aus und setzt sie in Kommandos um. Hier eine Funktionsübersicht:

Per SMS können auch Befehle an das Gnublin-Board geschickt werden. Ein Python-Script wertet diese Befehle aus und setzt sie in Kommandos um. Hier eine Funktionsübersicht:

- Setzen der Systemzeit (Zeit wird per SMS als String übergeben)
- Abrufen der Systemzeit
- Starten der Übermittlung von GPS-Koordinaten (mit Intervallangabe)
- Beenden der Übermittlung von GPS-Koordinaten
- Netzwerkinformationen übermitteln
- System neu starten
- Beliebigen Systembefehl ausführen (Befehl als String aus SMS)

Dieses Script listener.py wird in /root platziert und mittels chmod +sx ausführbar gemacht und mit Root-Rechten versehen. Um gewisse Systembefehle ausführen zu können, sind volle Privilegien erforderlich. Sicherheitsaspekte können hier vernachlässigt werden. Das Script wird beim Systemstart automatisch ausgeführt. Es wird die Software gammu verwendet, welche eine Abstraktionsschicht für den Zugriff auf verschiedene mobile Geräte und eine Python-API zur Verfügung stellt. Diese vermeidet das Arbeiten mit AT-Kommandos und stellt komfortable Befehle z.B. zum Versenden von SMS zur Verfügung.

Abb. 7: Die letzten Sekunden vor dem Start des Ballons

Um die Temperatur während des Flugs zu messen, wird eine Messbrücke mit einem PT1000-Temperatursensor an den Analog- Digital-Eingang GPA1 des Gnublin-Boards angeschlossen.

Der Analog-Digital-Wandler wird über eine Gerätedatei geöffnet und mit dem entsprechenden Kanal beschrieben. In diesem Fall wurde der Kanal 1 gewählt.

```
os.write(adc_file, „0x0001“)
```

Es werden immer 256 Byte von GPA1 gelesen. Als Wert erhält man einen hexadezimalen String, der die Differenz der Spannung der Messbrücke angibt. Diese Differenz wird dann in eine dezimale Temperatur umgerechnet und in eine Temperatur-Messdatei geschrieben. Die Temperaturanwendung kann per SMS gesteuert werden.

Die verwendete Webcam Logitech C170 wird vom System bereits unterstützt, daher entfallen aufwendige Konfigurationen. Es ist im System das Programm uvccapture enthalten, mit dem Webcams angesprochen werden können. Es soll in einem Intervall, etwa alle 2 Minuten, ein Bild aufgenommen werden. Diese Bilder werden im JPEG- Format auf der lokalen SD-Karte abgespeichert und werden nach der Bergung ausgewertet. Bei ausreichendem Bildmaterial könnte aus den Einzelbildern ein Videofilm geschnitten werden.

Quellcode

Der für das Projekt von uns erstellte Programmcode ist online auf Github zu finden [2].
Erläuterung der Dateien:

```
local.autostart          /etc/init.d/
                         Startet alle Prozesse
wdt.c                    /root/
                         Initialisiert den Hardware-Watchdog
listener.py              /root/
                         Initialisiert den SMS-Listener
listener_start.py        /root/
                         SMS-Listener, verarbeitet GPS-Daten und Kamerabilder
script_watcher.sh        /root/
                         Hält die Prozesse am Laufen
temperature.py           /root/
                         Liest den Temperatursensor aus
_installation.sh         /root/
                         Kopiert die einzelnen Dateien und setzt Rechte
```

Bilder

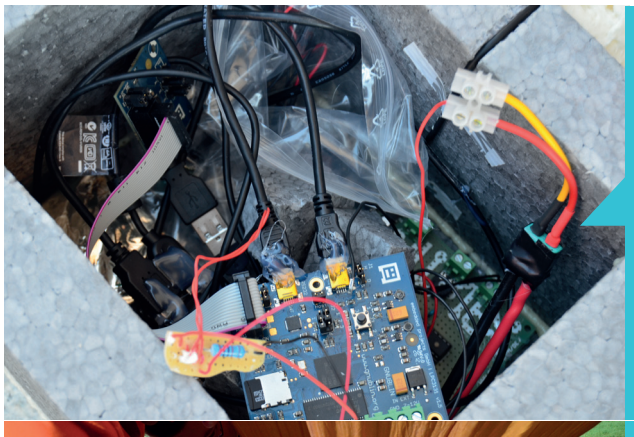


Abb. 8,9: Die bestückte Transport-kiste

Abb. 10,11: Ballonstart

Fazit

Die Planung und Organisation dieses Projektes war für uns eine besonders spannende Herausforderung. Es ist immer wieder eine Freude, interdisziplinär zu arbeiten. Leider lief beim Start jedoch alles anders. Beim Test der Software kurz vor dem Start erlebten wir einen Systemausfall, den wir noch versuchten zu beheben. Bedauerlicherweise erhielten wir bereits kurz nach dem Start keine Nachrichten mehr von der Elektronik, der Wetterballon ist seitdem verschollen. Falls Du ihn findest, kontaktiere uns bitte unter der oben angegebenen E-Mail-Adresse. Vielen Dank!

Literatur

[1] OpenCellID http://www.opencellid.org/

[2] GitHub https://github.com/ramteid/gnublin-weatherballoon

NetIO mit Gnublin

Julian Sarcher <sarcher@embedded-projects.net>

Benedikt Sauter <sauter@embedded-projects.net>

Mit NetIO erstellt man sehr schnell Oberflächen für Fernbedienungen. Man kann diese Oberfläche dann ganz komfortabel auf ein Smartphone oder Tablet laden. Als Gegenstück kann man auf dem GNUBLIN die Kommandos der Fernbedienung entgegennehmen, mit Python auswerten bzw. einfache Aktionen direkt starten.

HowTo

- Account auf NetIO anlegen [1].
- App auf Smartphone installieren (iPhone oder Android)
- UI-Designer per Browser starten und Config öffnen: „NetIO with Gnublin“ [2]
- Aktuelle IP Adresse von GNUBLIN in Connection-Settings eingeben
- Save Online klicken
- Im Smartphone Zugangsdaten eingeben und synchronisieren (siehe unten)
- „NetIO with Gnublin“ auswählen
- Auf GNUBLIN wechseln, Python API und Server kopieren und starten
- Relay + GPIO über das Smartphone steuern (Abb. 1)

Wir steuern als Demo [3] die rote LED (an GPIO 3) auf dem GNUBLIN Board an, welches im gleichen Netzwerk wie unser Telefon hängt.

Im Wesentlichen denkt man sich Kommandos aus, die man in der Oberfläche für jeden Button definiert. Diese Kommandos werden dann an den kleinen Server im GNUBLIN gesendet. Dort kann man diese einfach auswerten.

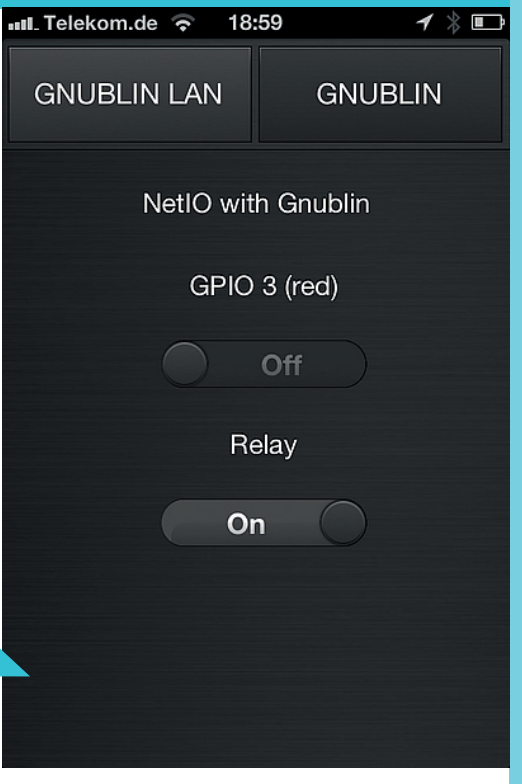


Abb. 1: Die im UI-Designer erstellte Oberfläche wird auf das Smartphone synchronisiert

Funktionsweise

Der erste Schritt ist es, einen Account auf der NetIO Webseite zu erstellen und die UI-Designer config von unserer Projektseite herunterzuladen [4]. Im UI-Designer kann man die Kommandos festlegen, welche beim Betätigen des Buttons gesendet werden sollen. In Abb. 2 ist ein Switch zu sehen. Dieser sendet, wenn er auf „on“ geschaltet wird: „LED1 on“. Dieser String wird auf dem Socket Server im Gnublin wiederrum abgefangen (Listing 1).

```
def handle_command(self, line):
    if line == ,LED1 on‘:
        self.send(,on‘)
        gpio.digitalWrite(3,1)
    elif line == ,LED1 off‘:
        self.send(,off‘)
        gpio.digitalWrite(3,0)
    elif line == ,RELAY open‘:
        self.send(,off‘)
        gpio.digitalWrite(18,0)
    elif line == ,RELAY close‘:
        self.send(,off‘)
        gpio.digitalWrite(18,1)
    else:
        self.send(,unknown cmd‘)
```

Listing 1: Auszug aus dem Socket Server

Hinweis: Bei Connection muss die aktuelle IP des Gnublin eingetragen werden. Die IP des Gnublin findest du mit folgendem Befehl heraus:

```
root@gnublin:~$ ifconfig
```

Anschließend wird die NetIO App auf dem Smartphone installiert. Die App baut einen TCP Socket auf um dort dann selbst definierte Strings zu versenden. Um diese mit dem Online Account zu synchronisieren, muss man bei einem Android Smartphone mit zwei Fingern über den Bildschirm streichen, das iPhone muss man schütteln.

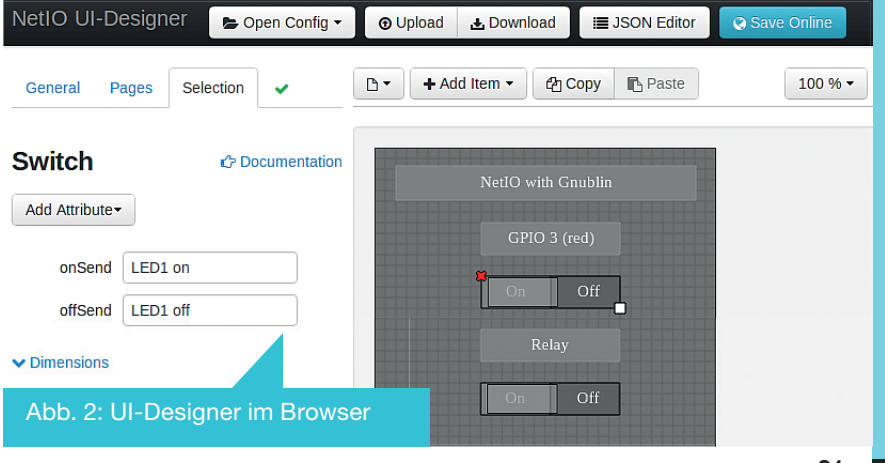


Abb. 2: UI-Designer im Browser

Nun benötigt man auf dem Gnublin noch einen TCP Socket Server der die Verbindung annimmt und entsprechend reagiert. Dieser wird im Beispiel in Python implementiert. Dazu muss das Gnublin mit dem Netzwerk verbunden sein [5].

Zudem ist zu überprüfen, ob die Python API bereits auf dem Gnublin installiert ist [6].

Sobald die Phyton API also auf dem Gnublin installiert und das Netzwerk eingerichtet ist, kann man sich den Socket Server direkt auf den Gnublin laden, entpacken und starten:

```
root@gnublin:~$ wget http://netio.davideick hoff.de/
media/attachments/gnublin _ netio _ 1.zip
root@gnublin:~$ unzip gnublin _ netio _ 1.zip
root@gnublin:~$ python server _ netio _ 1.py
```

Die Anleitung und den Quelltext gibt es auch bei uns im Gnublin-Wiki [7].

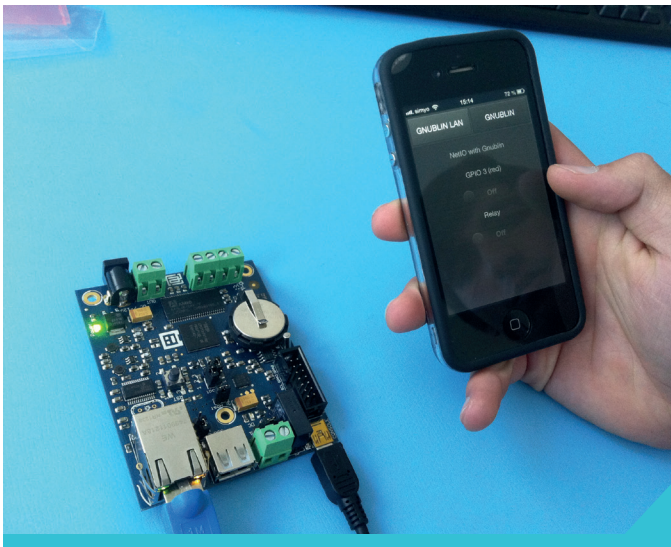


Abb. 3: Nun lässt sich der Gnublin bequem fernsteuern

Literatur

[1] http://netio.davideickhoff.de/

[2] http://netio.davideickhoff.de/editor2?config=6932

[3] http://youtu.be/Ng2PX9XzsUs

[4] http://netio.davideickhoff.de/projects/262

[5] http://wiki.gnublin.org/index.php/WLAN

[6] http://wiki.gnublin.org/index.php/API_mit_Python

[7] http://wiki.gnublin.org/index.php/NetIO

Marktplatz / Neuigkeiten

Die Ecke für Neuigkeiten und verschiedene Produkte

Seminartag für Elektronik-Profis und -Hobbyisten

elektorLive!

Am 12. Oktober 2013 veranstaltet das bekannte Elektronikmagazin Elektor zum zweiten Mal den Seminartag „ElektorLive!“. In Hanau bei Frankfurt am Main haben Elektronik-Profis und Hobby-Elektroniker Gelegenheit, attraktive Seminare zu besuchen und sich auf einer Ausstellung über neue Produkte zu informieren. Im Seminarprogramm sind Trendthemen wie „Android“, „Raspberry Pi“ und „Arduino“ zu finden. Doch auch die klassischen Grundlagen wie das Schaltplan- und Platinen-Design haben ihren Platz.

Hier ein Auszug aus dem Programm:

- Apps progammieren mit Android
- Raspberry Pi Grundlagen
- Arduino Praxisprojekte
- Anwendungen mit Embedded Linux
- Labview für den Praktiker
- Platinendesign mit „Eagle“
- Rapid SMD-Prototyping

Im Eintrittspreis von 49,50 Euro sind zwei Seminare inbegriffen, die man bei der Anmeldung auswählt. Studenten, Azubis und Schüler zahlen nur 19,50 Euro! Wie immer gilt: „Der frühe Vogel fängt den Wurm“, denn die Teilnehmerzahl ist bei allen Seminaren begrenzt.

Mehr Infos und Anmeldung: www.elektor-live.de

Picosafe - ein verschlüsselter GNUBLIN

Benedikt Sauter <sauter@embedded-projects.net>

Picosafe ist ein kleiner komplett verschlüsselter Computer für die Hosentasche. Die Schaltung basiert auf GNUBLIN, jedoch wird ein LPC3143 eingesetzt. Dies ist ein Prozessor, der komplett verschlüsselt booten kann. Einmalig wird zu Anfang ein Schlüssel in den Prozesssor gebrannt, der anschließend den Bootloader nur noch verschlüsselt startet.

Abb. 1: Picosafe im Gehäuse

Der Bootloader kopiert und entschlüsselt den Kernel in den Arbeitsspeicher. Der Kernel wiederum startet ein intern mit Luks verschlüsseltes Debian Linux. Das Passwort für die Entschlüsselung gibt man über den Browser per HTTPS geschützt ein.

Nur derjenige, der den Schlüssel kennt, kann einen neuen Bootloader oder Kernel erstellen bzw. modifizieren. Der Stick selber emuliert eine Netzwerkschnittstelle und ein Massenspeicherlaufwerk. Jetzt kann man sich per SSH oder HTTPS verbinden. In der Standardversion bringt Picosafe eine Weboberfläche mit, in der man schnell weitere virtuelle USB-Sticks zum Speichern von Daten erstellen kann. Mitverwendet werden kann ein Passwort-Tresor bzw. ein privates Wiki. Aktuell arbeiten wir an einer E-Mail und Chat Anwendung, so dass man von Picosafe zu Picosafe schnell und verschlüsselt kommunizieren kann.

Warum benötigt Picosafe einen Akku? Ganz einfach: Möchte man z.B. Daten zwischen zwei PCs übetragen muss man sich nicht jedes mal authentifizieren, sondern kann selbst bestimmen, wann und wie lange der Stick „offen" bleiben soll. Picosafe bringt als eine komfortable Absteckfunktion für verschiedenste Anwendungen von Haus aus mit.

Weitere Infos:

www.picosafe.de

Abb. 2: Die Weboberfläche erreicht man per Browser.

embedded - projects.net
JOURNAL
OPEN SOURCE SOFT-AND HARDWARE PROJECTS

Werdet aktiv!

Das Motto: Von der Community für die Community !

Das Magazin ist ein Open Source Projekt.

Falls Du Lust hast, Dich an der Zeitschrift durch einen Beitrag zu beteiligen, würden wir uns darüber sehr freuen. Schreibe deine Idee an:

journal@embedded-projects.net

Regelmäßig

Die Zeitschrift wird über mehrere Kanäle verteilt. Der erste Kanal ist der Download als PDF - Datei. Alle Ausgaben sind auf der Internetseite [1] verfügbar. Diejenigen, die lieber eine Papierversion erhalten möchten, können den zweiten Kanal wählen. Man kann sich dort auf einer Internetseite [2] in eine Liste für die gesponserten Abos oder ein Spendenabo eintragen. Beim Erscheinen einer neuen Ausgabe wird dank Sponsorengeldern an jeden auf der Liste eine Ausgabe des aktuellen Journal versendet. Falls man den Versandtermin verpasst hat, kann man das Heft auch über einen Online - Shop [2] beziehen.

[1] Internetseite (Anmeldeformular gesponserte Abos): http://journal.embedded-projects.net

[2] Online - Shop für Journal:
http://www.embedded-projects.net

Sponsoren gesucht!

Damit wir weiterhin diese Zeitschrift für jeden frei bereitstellen können, suchen wir dringend Sponsoren für Werbe- und Stellenanzeigen. Bei Interesse meldet Euch bitte unter folgender Telefonnummer: 0821 / 2795990 oder sendet eine E-Mail an die oben genannte Adresse.

Bitte frei machen

embedded projects GmbH
Holzbachstraße 4
D - 86152 Augsburg

Name / Firma

Straße / Hausnummer

PLZ / Ort

Email / Telefon / Fax

◯ Ich möchte jede zukünftige Ausgabe erhalten

◯ Wir möchten als Hochschule / Ausbildungsbetrieb jede weitere Ausgabe bekommen. Bitte gewünschte Anzahl der Hefte pro Ausgabe ankreuzen. 5 10

◯ Ich möchten im embedded projects Journal werben oder eine Stellenanzeige aufgeben. Bitte schicken Sie mir Infomaterial, Preisliste etc. zu.

Impressum

embedded projects GmbH
Holzbachstraße 4
D-86152 Augsburg
Telefon: +49(0)821 / 279599-0
Telefax: +49(0)821 / 279599-20

Verteilt durch:

embedded projects GmbH
HARDWARE FOR PROJECTS

Veröffentlichung: 4x / Jahr
Ausgabenformat: PDF / Print

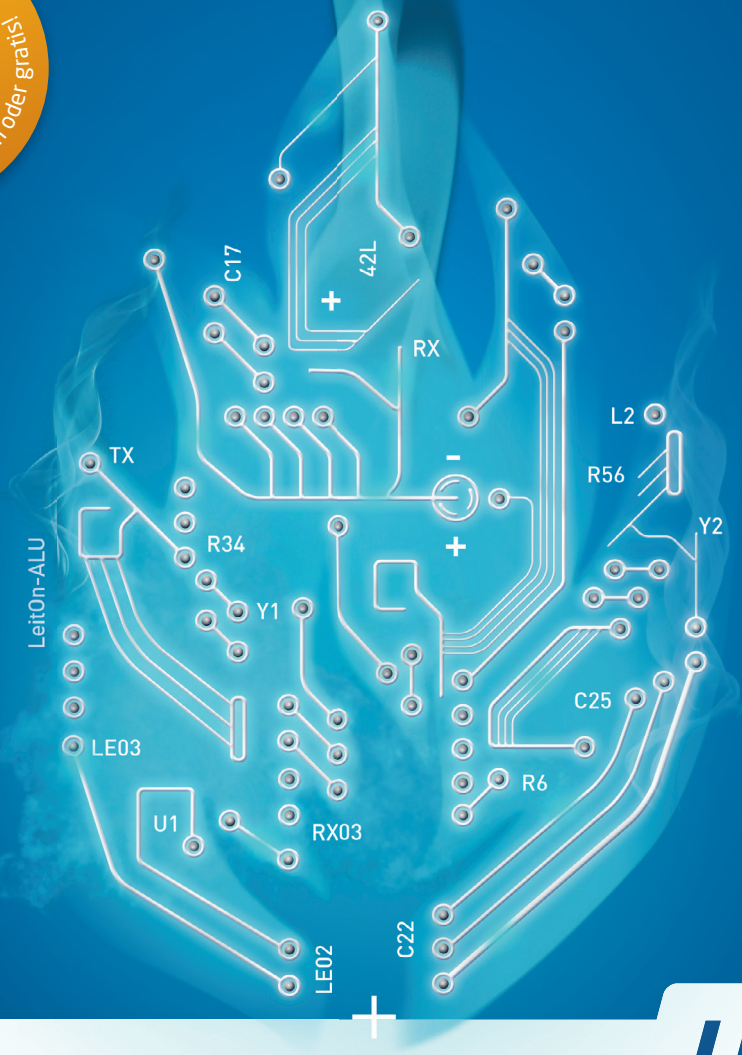
Auflagen Print: 2500 Stk.
Einzelverkaufspreis: 1 €

Layout / Satz: EP
Druck: flyeralarm GmbH
Titelbild: Markus Musielak

GERADE WENN'S MAL HEISS HERGEHT.

LEITERPLATTEN AUS ALUMINIUM ONLINE BESTELLEN.

ab 12h Leiterplatten-Express! Pünktlich oder gratis!



www.HTQN.de

LEITON
RECHNEN SIE MIT BESTEM SERVICE

Vorsicht, heiß! Starke Hitzeeinwirkung ist eine echte Herausforderung – besonders in der Hochleistungs-LED-Technik oder wenn wichtige Leistungsbauteile permanent hohen Temperaturen ausgesetzt sind. Aluminium-Platinen von LeitOn überzeugen durch **hohe Hitzeresistenz** sowie einen Wärmeleitwert von bis zu 3,0 W/mK in der Premiumvariante „Polytherm TC-Lam 3.0". Die vielen Vorteile liegen auf der Hand: **Kostenreduktion** durch höhere Lebensdauer und Zuverlässigkeit, **Platzersparnis** dank Integration der aufwendigen Kühlmechanismen sowie **Performancesteigerung** durch höhere Leistungsdichte der Anwendung. Sie möchten mehr darüber wissen? Wir bieten persönliche Beratung am Telefon und einen kompetenten Außendienst. Sie können bei LeitOn immer mit dem besten Service rechnen.

www.leiton.de | **kontakt@leiton.de** | Info-Hotline **+49 (0)30 701 73 49 0**