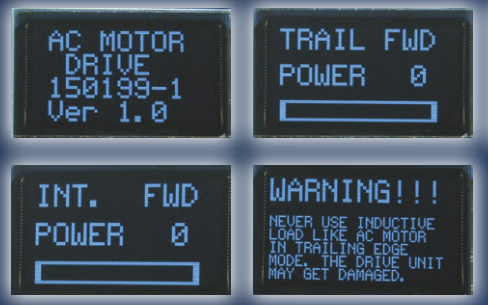
Januar/Februar 2020 | Nr. 571 | www.elektormagazine.de

elektor magazin

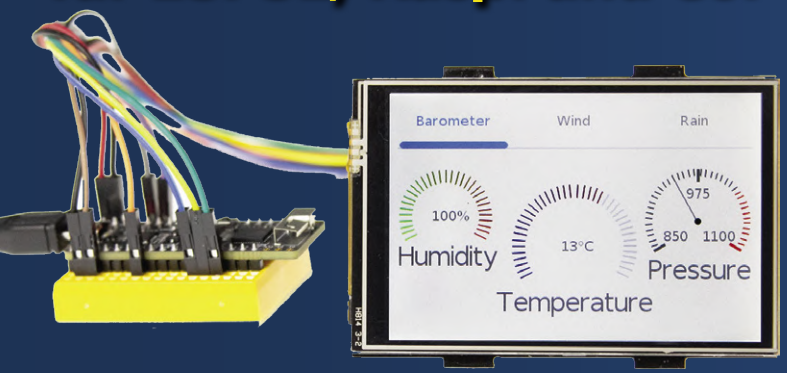
DESIGN BUILD SELL ELECTRONICS

Einstellbarer 1-kW-AC-Motortreiber



Steuerung in drei Modi: Schwingungspakete / Phasenabschnitt / Phasenanschnitt

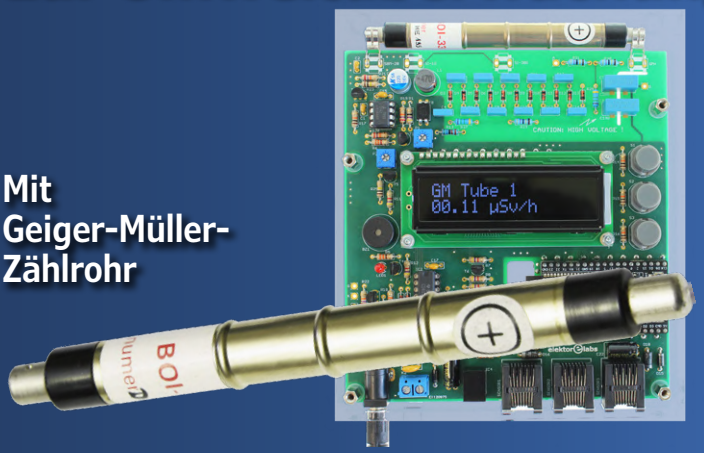
GUI mit Touch für ESP32, Raspi und Co.



Grafische Benutzerschnittstellen mit der Bibliothek LittlevGL

Erweiterbares System zur Umweltüberwachung

Mit Geiger-Müller-Zählrohr



Capaci-Meter mit LED-Anzeige ⊳ Erste Schritte mit RISC-V ⊳ Gewusst wie: Entprellen eines Schalters ⊳ ESP32-Multitasking mit FreeRTOS und Arduino ⊳ Anmeldung im Sigfox-Netz ⊳ Autoscheinwerfer tunen ⊳ Von Entwicklern für Entwickler ⊳ Zwei Wärmebildkameras im Vergleich ⊳ Touch-Display für RPi ⊳ Entwicklung analoger Elektronik ⊳ Bemerkenswerte Bauteile: 7-Segment-LED-Anzeige ⊳ Und vieles mehr!

G3078 E

(D) € 14,90 | CHF 27,00 | (A, B, L) € 16,00

4 198630 314904 02

Die Elektor-Community

82	247235	1045	507	240056
Länder	Mitglieder	Experten & Autoren	Literatur	Monatliche Besucher

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

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

Elektor-Zeitschrift: 9 Ausgaben (6 Doppelhefte + 3 Industry-Ausgaben) pro Jahr voll gepackt mit Elektronik-Projekten, Artikeln, Besprechungen, Tipps und Tricks. **www.elektormagazine.de**

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

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

Elektor Academy: Webinare, Seminare, Präsentationen, Workshops und mehr für praxisorientiertes Lernen. **www.elektor.de**

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

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

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

Treten Sie dem weltweit größten Elektroniker-Netzwerk bei!

GREEN

- ✔ Zugang zum Elektor-Archiv
- ✔ 10% Rabatt auf Shop-Produkte
- ✔ 6x Elektor-Doppelheft (Digital)
- ✔ 3x Elektor Industry (Digital)
- ✘ 6x Elektor-Doppelheft (Print)
- ✘ 3x Elektor Industry (Print)
- ✔ Exklusive Top-Angebote
- ✘ Elektor Jahrgangs-DVD

Unsere Empfehlung

www.elektor.de/green-mitglied

GOLD

- ✔ Zugang zum Elektor-Archiv
- ✔ 10% Rabatt auf Shop-Produkte
- ✔ 6x Elektor-Doppelheft (Digital)
- ✔ 3x Elektor Industry (Digital)
- ✔ 6x Elektor-Doppelheft (Print)
- ✔ 3x Elektor Industry (Print)
- ✔ Exklusive Top-Angebote
- ✔ Elektor Jahrgangs-DVD

www.elektor.de/gold-mitglied

GRATIS

- ✘ Zugang zum Elektor-Archiv
- ✘ 10% Rabatt auf Shop-Produkte
- ✘ 6x Elektor-Doppelheft (Digital)
- ✘ 3x Elektor Industry (Digital)
- ✘ 6x Elektor-Doppelheft (Print)
- ✘ 3x Elektor Industry (Print)
- ✔ Exklusive Top-Angebote
- ✘ Elektor Jahrgangs-DVD

www.elektor.de/newsletter

facebook.com/ElektorDE

twitter.com/ElektorDE

instagram.com/elektorlabs

Impressum

51. Jahrgang, Nr. 571
Januar/Februar 2020

Erscheinungsweise: 9x jährlich
(6x ElektorLabs-Doppelheft + 3x Elektor Industry Magazin)

Verlag
Elektor-Verlag GmbH
Kackertstraße 10
52072 Aachen
Tel. 0241 95509190

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

Hauptsitz des Verlags
Elektor International Media
Postbus 11, 6114 ZG Susteren
Niederlande

Anzeigen
Margriet Debeij (verantwortlich)
Tel. 0241 95509174
Mobil: +49 170 5505396
E-Mail: margriet.debeij@elektor.com

Tanja Pohlen
Tel. 0241 95509186
E-Mail: tanja.pohlen@elektor.com

Es gilt die Anzeigenpreisliste ab 01.01.2020.

Distribution
IPS Pressevertrieb GmbH
Postfach 12 11, 53334 Meckenheim
Tel. 02225 88010
Fax 02225 8801199

Druck: Pijper Media, Groningen (NL)
ISSN 0932-5468

Die nächsten 10 Jahre

Das neue Elektronik-Jahrzehnt ist da. Technologien wie SigFox und LoRa bieten sich für das wachsende Internet der Dinge an; es gibt innovative Prozessor-Architekturen wie RISC-V. Entwickler erwarten, schnell zu fertigen Applikationen zu kommen, mit günstiger Hard- und Software. Sehr rasch ans Ziel gelangt man mit dem immer gängiger werdenden modularen Ansatz. Doch für das Feintuning ist das Wissen, wie einzelne Bauteile zusammenspielen, weiterhin unabdingbar. Gleichzeitig interessieren sich immer mehr Elektroniker dafür, privat entwickelte Lösungen auch zu vermarkten. Elektor – die weltweit führende Elektronikzeitschrift – muss den Anspruch haben, ihre Leser bei all diesen Entwicklungen zu begleiten. In diesem Heft führen wir daher ein paar neue Rubriken ein. Regelmäßig wollen wir neue Controller-Plattformen vorstellen (S. 74). Das analoge Zusammenspiel von Komponenten beleuchten wir in unserer Experten-Rubrik auf S. 88. In unserem Interview kommen die Köpfe hinter erfolgreichen (Open-Source-)Projekten zu Wort – Sie werden dort auch einiges darüber erfahren, wie Sie eigene Projekte bekannter machen können (S. 14). Dazu kommen Seiten für Profi-Entwickler und solche, die es werden wollen (S. 26, 56). Besonders gerne greifen wir den vielfachen Wunsch unserer Leser auf, wieder etwas mehr für Einsteiger zu tun (S. 23, 53, 86, 92).
Das Herzstück unseres Heftes bleiben unsere Projekte, bei denen wir mehr als bisher auch die Software beleuchten (z.B. auf S. 7). Und nicht zuletzt unterziehen wir die interessantesten Produkte aus unserem Shop einem Praxistest (u.a. S. 54).
Unsere Redaktion ist ab jetzt ganz international aufgestellt, mit mir als Chefredakteur (bisher war ich nur für die deutsche Ausgabe verantwortlich). Ohne die Erfahrung und das Wissen meiner Kollegen, die lange Jahre die Länderausgaben geleitet haben, könnte ich die Aufgabe natürlich nicht stemmen. Nur als Team können wir Ihnen eine Zeitschrift und Webseiten bieten, die alle Bereiche der Elektronik bestmöglich abdecken. Wir alle freuen uns mit Ihnen auf die nächsten 10 Jahre!

Jens Nickel

Unser Team

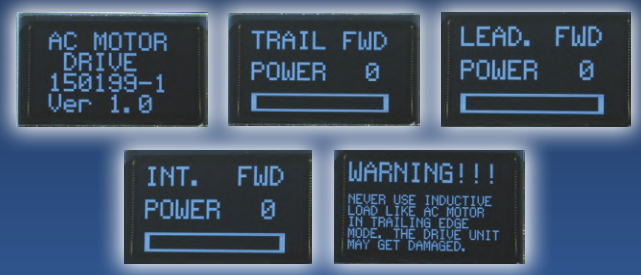
Chefredakteur:	Jens Nickel (v.i.S.d.P.) (redaktion@elektor.de)
Redaktion:	Eric Bogers, Jan Buiting, Rolf Gerstendorf, Denis Meyer, Dr. Thomas Scherer, Clemens Valens
Leserservice:	Ralf Schmiedel
Elektor-Labor:	Mathias Claußen, Ton Giesberts, Hedwig Hennekens, Luc Lemmens, Jan Visser, Clemens Valens
Grafik & Layout:	Giel Dols

Inhalt

51. Jahrgang – Nr. 571
Januar/Februar 2020

Rubriken

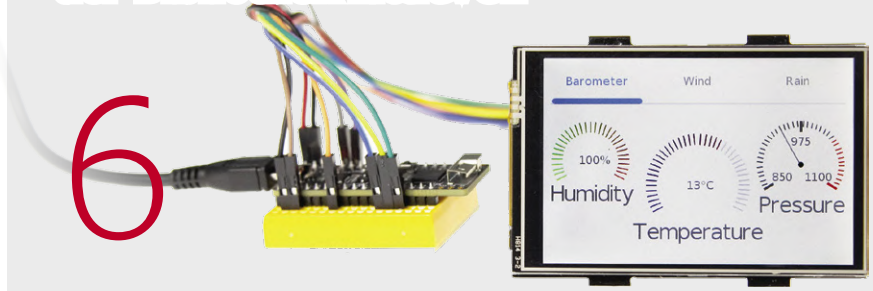
Einstellbarer 1-kW-AC-Motortreiber



Steuerung in drei Modi: Schwingungspakete /

GUI mit Touch für ESP32, Raspi und Co.

Grafische Benutzerschnittstellen mit der Bibliothek LittlevGL



Hintergrund

Interview mit Gábor Kiss-Vámosi, dem Entwickler von LittlevGL

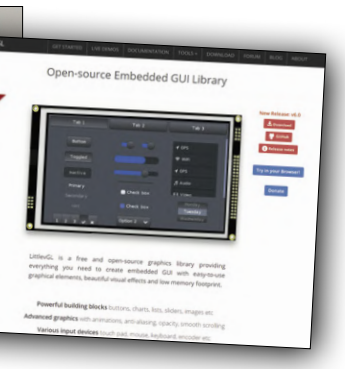
62

Phasenabschnitt / Phasenanschnitt

Erweiterbares System zur Umweltüberwachung

Mit Geiger-Müller-Zählrohr

28



14

elektor
DESIGN BUILD SELL ELECTRONICS

Projekte

Vorschau

Das nächste Heft ist wie immer randvoll gefüllt mit Schaltungsprojekten, Grundlagen sowie Tipps und Tricks für Elektroniker.

Aus dem Inhalt:

- 230-V-Relais mit LoRa-Fernsteuerung
- BASIC für ESP8266
- Günstiger LoRa-Knoten und -Gateway
- ESP32-Türklingel
- IoT-Button
- Optotastkopf
- .NET-Board Meadow
- Entwicklung analoger Elektronik
- Selbstbau-PC fürs Elektronik-Labor
- Interview mit Wienke Giezeman (TheThingsNetwork)

Und vieles mehr!

Änderungen vorbehalten.
Elektor März/April 2020 erscheint am 27. Februar 2020.

GUI mit Touch - für ESP32, Raspi und Co.

Grafische Benutzerschnittstellen mit der Bibliothek LittlevGL

Von **Mathias Claußen** (Elektor-Labor)

Fast alle Projekte mit Mikrocontroller müssen irgendetwas anzeigen. Waren früher oft zweizeilige Text-Displays üblich und ausreichend, werden heute zunehmend grafische LCDs (oder OLED-Displays) eingesetzt. Für die Anzeige von ansprechenden Grafiken und/oder für die Touch-Bedienung gibt es eine ganze Reihe von Bibliotheken. LittlevGL ist eine Library unter der sehr freien MIT-Lizenz, die einfach an die verschiedensten Displays und Controller angepasst werden kann. In diesem Artikel demonstrieren wir das Ganze mit einem ESP32-Board und einem Touch-LCD, das Daten einer Wetterstation anzeigt.

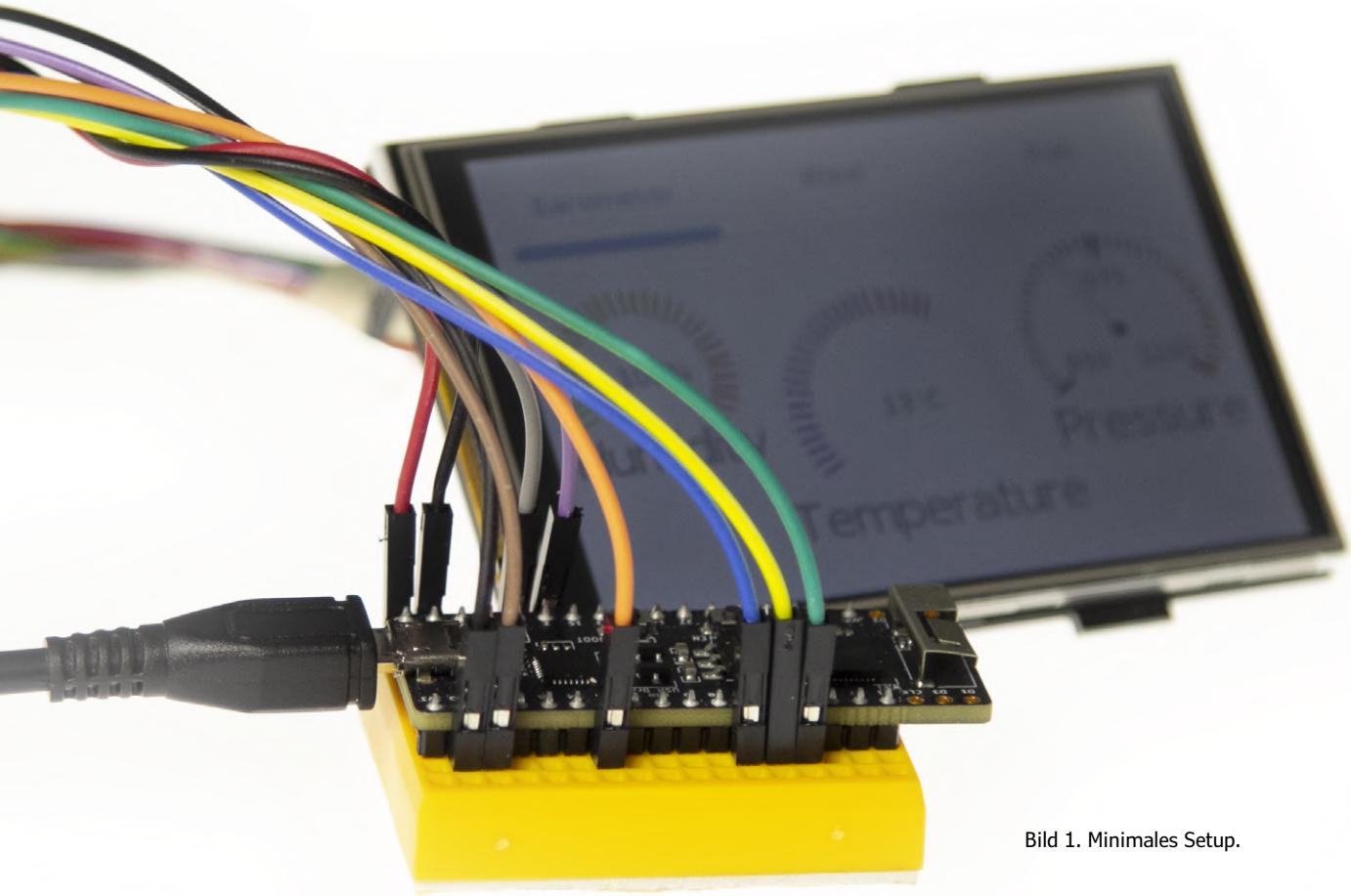


Bild 1. Minimales Setup.

Grafische Benutzerschnittstellen erfordern häufig viel mehr Programmieraufwand als der „eigentliche Code“, der die wichtigen Funktionen des Projekts erledigt. Um sich daher auf die wesentlichen Aspekte des Projekts konzentrieren zu können, werden Fertiglösungen bzw. der Einsatz fertiger Bibliotheken für die grafische Ausgabe von Informationen immer beliebter und wichtiger. Bekannte Namen sind *µGFX*, *emWin* oder *TouchGFX* für STM32-Boards, um nur ein paar zu nennen. Alle haben Vor- und Nachteile, etwa bezüglich kommerzieller Lizenzierung oder der Bindung an bestimmte Controller-Hersteller. Natürlich könnte man auch selbst eine Bibliothek entwickeln, doch das ist eine Riesenmenge an Arbeit und bietet viele Fallstricke, von den vielen Bugs im selbstgestrickten, umfangreichen Code ganz zu schweigen.

Besser sieht es z.B. bei *LittlevGL* von Gábor Kiss-Vámosi [1] aus, denn diese Bibliothek kommt mit einer sehr projektfreundlichen MIT-Lizenz daher. Ein damit entwickeltes GUI eignet sich gut für Touchscreens, kann aber auch mit Maus, Tastatur oder

einigen Tastern bedient werden. Der Code läuft auf gängigen 16-, 32- und 64-Bit-Mikrocontrollern. Minimalkriterien sind 16 MHz, 64 KB Flash und 16 KB RAM. Damit passt die Bibliothek perfekt zu kleinen Boards wie dem ESP32 oder ESP8266 und wurde inzwischen von Espressif auch in ihr IDF aufgenommen. Nachfolgend gibt es Unterstützung für den Einstieg und die Zusammenstellung von Test-Hardware. LittlevGL bietet übrigens auch die Möglichkeit zur Entwicklung und Test von GUIs am PC, was nicht zu verachten ist. Den am PC erstellten Code für ein GUI kann man ohne große Anpassungen auf den Ziel-Mikrocontroller übertragen.

Bibliotheken & ESP32

Man lernt am meisten, wenn man etwas praktisch ausprobiert. Von daher wird hier der Einsatz der Bibliothek bei der Wetterstation von Elektor [2] demonstriert. Ziel ist eine für die Touch-Bedienung geeignete GUI. Sogar eine mehrseitige Anzeige für Daten werden wir realisieren. Doch dazu bedarf es Hardware. Ein ESP32-Modul ist leicht erhältlich. Geeignet sind z.B. ein *ESP32-PICO-D4*, ein *ESP32-DevKitC* oder ein davon abgeleitetes Board. Beim Display hat man die Wahl zwischen einem seriellen Interface oder paralleler Ansteuerung, die dann aber fast alle IOs des ESP32 verwendet. Da auch der Preis eine Rolle spielt, kommt hier ein verbreitetes 3,5“-LCD für den Raspberry Pi zum Einsatz. Die meisten günstigen Displays wie das 3,5“-Exemplar von JOY-iT werden per SPI angeschlossen und arbeiten mit 3,3-V-Pegeln auf den Signalleitungen. Sie eignen sich also perfekt für den Pins sparenden Anschluss an ein EPS32-Board. Zudem haben sie schon einen per SPI anschließbaren Touch-Controller integriert.
Die für RPi gedachten SPI-Displays sind allerdings in der Geschwindigkeit begrenzt, mit der Daten an das Display transferiert werden können. Wer so ein Display schon einmal an einem RPi in Aktion erlebt hat, hat die etwas träge Bildschirmaktualisierung schon zu spüren bekommen.
Wichtig: *LittlevGL* stellt keine Display-Treiber bereit, sondern nur „höhere” Funktionen für das Zeichnen von Objekten. Es ist Sache des Entwicklers, die passenden hardwarenahen Routinen zu entwickeln. Aber auch dafür muss man das Rad nicht neu erfinden, denn für die meisten Display-Controller gibt es bereits fertige Bibliotheken. Hier wird auf die Arduino-Bibliothek *TFT_eSPI* [3] zurückgegriffen, die auch 3,5“-Displays für RPi unterstützt.

Hardware

An Hardware für den Nachbau (siehe **Bild 1**) braucht es:

- ESP32-DevKitC-32D oder ESP32-PICO-Kit V4
- 3,5” Touch-Display für Raspberry Pi von JOY-iT
- Kleine Steckbretter & Drahtbrücken

Software

Auch die erforderliche Software ist übersichtlich. Neben der obligatorischen Arduino-IDE samt Board-Unterstützung für den ESP32 benötigt man noch die Arduino-Versionen der Bibliotheken *LittlevGL* und *TFT_eSPI*.
Um beide Bibliotheken komfortabel zu installieren und zu verwalten, müssen die beiden folgenden Adresszeilen in der Arduino-IDE unter *Preferences -> Additional Boards Manager URLs* eingetragen werden:

```
https://github.com/littlevgl/lv_arduino/library.json
https://github.com/Bodmer/TFT_eSPI/library.json
```

Damit sucht und installiert man *LittlevGL* und *TFT_eSPI* per Library-Manager. Anschließend prüft man, ob im Arduino-Bibliotheksordner „TFT_eSPI“ und „LittlevGL“ vorhanden sind.
Wie schon erwähnt braucht es beide Bibliotheken. *LittlevGL* kümmert sich um das UI, also die Animation und Anordnung von Objekten, die Verwaltung mehreren Szenen und das Rendern der angezeigten Grafiken. Resultat ist eine Bitmap. Diese Daten werden dann von *TFT_eSPI* in passender Weise an die Display-Hardware transferiert. Diese Bibliotheken abstrahieren daher vom konkret verwendeten Display.

Mehr Displays

TFT_eSPI unterstützt nicht nur SPI-Displays für RPi, sondern auch noch weitere mit folgenden Controllern: ILI9341, ST7735, ILI9163, S6D02A1, HX8357D, ILI9481, ILI9486, ILI9488, ST7789 und R61581.
Damit werden viele gängige Farb-Displays unterstützt. Sollte ein RAiO-basierter Controller wie etwa der RA8875 verwendet werden, so kann man stattdessen die *RA8875-Library* von Adafruit [4] verwenden. Hierbei sind Anpassungen nötig, um *LittlevGL* anzubinden. Neben farbigen Displays können in Verbindung mit der *u8g2-Library* [5] auch monochrome Typen verwendet werden. Der folgende Text bezieht sich allerdings auf die gebräuchlichen 3,5“-SPI-LCDs für RPi.

Eigener Treiber

Wenn ein Display mit einem Controller verwendet wird, für den es keinen Arduino-Treiber gibt, muss man wissen, welche Funktionen bereitgestellt werden müssen. Auch für eine Portierung und die Anbindung ist dieses Wissen hilfreich.
Prinzipiell reicht es, einen eigenen Treiber zu schreiben, der einzelne Pixel auf dem Display in einer definierten Farbe setzen kann. *LittlvGL* erwartet eine Funktion wie folgt:

```
/*************************************
* Function    : disp_flush_data
* Description : Sends pixels to the display
* Input       : lv_disp_drv_t *disp, int32_t x1,
*               int32_t y1, int32_t x2, int32_t y2,
*               const lv_color_t *color_array
* Output      : none
* Remarks     : none
*************************************/
void disp_flush_data(lv_disp_drv_t *disp,
 const lv_area_t *area, lv_color_t *color_p){
   /* Here: grab the pixel and send it to the display
   */

   /* Inform the library after job is done */
   lv_flush_ready(disp);
}
```

Diese Funktion wird *LittlevGL* als Funktions-Pointer übergeben. Als Parameter erhält sie die Start- und Endkoordinaten des zu füllenden Zeichenbereichs sowie einen Pointer auf die Bilddaten. Das konkrete Setzen der Pixel hängt vom Treiber für das jeweilige Display ab. Es kann allerdings vorkommen, dass die von *LittlevGL* gewünschten Farben noch für das Display umgerechnet werden müssen (z.B. von RGB nach BGR). Mehr braucht die eigentlichen Zeichenroutine nicht zu können. Andererseits

gibt es Lösungen für eine Hardwarebeschleunigung, wie etwa durch die DMA2D-Engine einiger STM32-Controller.

Touchy

Damit wäre klar, wie eine Grafik auf das Display kommt. Was fehlt ist die Verarbeitung der Daten vom Touch-Controller. Hierfür gibt es eine LittlevGL-Funktion, welche die Koordinaten eines eventuellen Touchs geräteabhängig ausliest und verarbeitet. RPi-Displays sind in der Regel mit einem Touch-Controller des Typs *XPT2046* bestückt, der als weiterer Slave am SPI-Bus angebunden ist. Leider kann dieser nicht mit einem höheren Takt als 2,5 MHz ausgelesen werden. Da das Display und sein Controller mit einem Takt von 20 MHz (bei Raumtemperatur bis zu 26 MHz) laufen, muss bei einem Zugriff der Bus-Takt angepasst und anschließend wieder auf den ursprünglichen Takt zurückgesetzt werden. Auch hier hilft die Bibliothek *TFT_eSPI*, denn sie bietet nicht nur XPT2064-Unterstützung, sondern kümmert sich automatisch um die erforderliche Takt-Umschaltung. Wer keine Touch-Bedienung nutzt, kann das UI auch mit Maus, Tastatur, Drehgebern oder Tastern bedienen. Selbstverständlich benötigen auch diese Eingabemethoden passende Treiber. Sie müssen in *LittlevGL* passend registriert werden.

Bildaufbau

Zunächst ein Wort zu den Stärken von *LittlevGL*: Vorteilhaft ist, dass auch der Quelltext zur Verfügung steht. Dies erleichtert das Debugging des eigenen Codes. Außerdem ist die Bibliothek gut dokumentiert und wird aktiv weiter entwickelt. Auch Einsteiger können mit den Beispielen schnell erste Erfolgserlebnisse erzielen und Oberflächen gestalten. Angefangen von einfachen Labeln über Buttons und Tabellen bis hin zu Dropdown-Listen und Gauges wird eine weite Palette an Bedienelementen bereitgestellt. Weiter werden Fenster und Hinweisboxen sowie Themes für das Erscheinungsbild geboten, um die GUI noch besser an eigene Bedürfnisse anpassen zu können. Die Dokumentation beschreibt alle Elemente detailliert. Unter [1] werden auch Funktionen der Bibliothek und ihr Zusammenspiel demonstriert. *LittlevGL* bietet aktuell (noch) keine Basis-Funktionen zum Setzen von Pixeln oder Zeichnen von Linien. Der Grund liegt in der Art der Bildgenerierung: Wenn sich ein Element ändert, kann der neu zu zeichnende Bildschirmbereich ermittelt und der Puffer im internen RAM vorbereitet werden. Anschließend wird das Bild dann an das Display gesendet. Folglich müssen nicht nur eine Linie als Objekt vorliegen, sondern sogar einzelne Pixel. Diese Einschränkung erleichtert dafür das Neuzeichnen von Bereichen auf dem Bildschirm deutlich.

Nun zu den technischen Details der Generierung und Anzeige von Bildern: Wenn man flimmer- und störungsfreie Animationen oder Bildschirm-Updates haben möchte, könnte man zunächst das komplette Bild im RAM des Controllers vorbereiten und diese Daten anschließend erst ans Display übertragen. In unserem Fall hätte man es hier mit 307 KB Daten zu tun. Man könnte aber auch direkt alle Elemente an das Display übertragen und so weniger RAM belegen. Letzteres erschwert eine flickerfreie Anzeige und behindert Effekte wie Antialiasing, Transparenz und Schatten.

Ein Kompromiss ist das Spiegeln eines Bildschirmteilbereichs im RAM. Mit nur etwas mehr als 10 % des Speicherbedarfs für das Gesamtbild bekommt man schon alle angeführten Features. Für ein Display mit 480 x 320 Pixeln bei 16 Bit Farbtiefe wären dies „nur“ 30,7 KB RAM – für einen ESP32 ist das zwar eine Menge, die er aber durchaus packt.

Tabelle 1. ESP32-Verdrahtung.

Funktion	Display-Pin	ESP32-Pin	Bemerkung
MISO	21	19	
MOSI	19	23	
SCK	23	18	
DC	18	02	
CS	24	05	
RST	22	EN	spart 1 GPIO-Pin ein
T_CS	26	04	
VCC (5 V)	02	5 V	
GND	14 / 25	GND	
T_IRQ (optional)	11	34	ESP32-Pin nur Input

In der aktuellen Version 6 der Bibliothek wird der Speicherbereich nicht durch ein `#define` mitgeteilt, sondern ist per Code bereitzustellen. Vor allem ist dieses Vorgehen praktisch, wenn weiteres externes RAM vorhanden ist und genutzt werden soll. Bei unserer Demo beschränken wir uns auf eine statische Allokation eines Bereichs im Speicher des ESP32, was den Code einfach hält:

```
//Memory for the displaybuffer
static lv_disp_buf_t disp_buf;
static lv_color_t buf[LV_HOR_RES_MAX * 10];
```

Dieser Speicher wird dem Display in der Funktion `hal_init()` mit der folgenden Zeile zugewiesen:

```
lv_disp_buf_init(&disp_buf, buf, NULL, LV_HOR_RES_MAX
   * 10);
```

Bei anderen Mikrocontrollern muss überlegt werden, was möglich ist, denn etliche Exemplare haben deutlich weniger RAM zur Verfügung oder müssten über Klimmzüge externes RAM ansprechen. Neben dem verfügbaren RAM sind auch andere Aspekte wie die verfügbare Rechenleistung oder ein Multi-Threading relevant. *LittlevGL* kann leider kein Multi-Threading, weshalb alle Zugriffe aus demselben Thread erfolgen müssen, in dem auch die Funktion `lv_task_handler()` aufgerufen wird. Die nötige Rechenleistung hängt davon ab, wieviel Interaktion und Zeichnerei auf dem Display geschieht sowie ob und wie Animationen eingesetzt werden. Ein ESP32 hat dank seiner zwei Prozessorkerne genug Rechenleistung für ein GUI.

Experimente

Wer nun selbst experimentieren möchte, auf den lauern gelegentlich Fallstricke. Für eine reibungsarme Inbetriebnahme wird nachfolgend ein Setup beispielhaft beschrieben. Ein ESP32-D4-PICO-Board hat durch die zusätzliche Last eines Displays gelegentlich leichte Startprobleme. Ein zusätzlicher Kondensator von 10 µF zwischen 3,3 V und GND verzögert das Booten soweit, bis die Spannungen in einem definierten Bereich sind. Der Anschluss eines Displays an ein ESP32-Board erfolgt anhand der Belegung von **Tabelle 1**. Damit wäre die Hardware vorbereitet. Nun folgen die Konfiguration und der Test der Software. Zunächst geht es um die Bibliothek *TFT_eSPI* als eigentlichem Display-Treiber und anschließend um die Konfiguration von *LittlevGL*.

Beim Display-Treiber muss man im Bibliotheksverzeichnis der Arduino-IDE den Ordner *TFT_eSPI* suchen, um die Datei *User_*

Setup.h an das genutzte Display anzupassen. Für das verwendete Display müssen die folgenden `#defines` vorhanden sein:

```
#define RPI_ILI9486_DRIVER // max. 20 MHz SPI
#define TFT_MISO 19
#define TFT_MOSI 23
#define TFT_SCLK 18
#define TFT_CS   05 // Chip select control
#define TFT_DC   02 // Data Command control
#define TFT_RST  -1 // set TFT_RST to -1 if display
   RESET is connected to ESP32 RST
#define TOUCH_CS 04 // Chip Select (T_CS) of touch
   screen
#define SPI_FREQUENCY  20000000

// An XPT2046 requires a lower SPI clock rate of
   2.5 MHz:
#define SPI_TOUCH_FREQUENCY 2500000
```

Es werden also die verwendeten GPIOs definiert und ein 20-MHz-SPI-Takt dient als sicherer Anfangswert für das Display. 2,5 MHz eignen sich für den Touch-Controller. Zum Testen wählen wir ein Beispiel (Auswahl: *TFT_eSPI -> 480x320 -> Rainbow480*), das einmal die Regenbogenfarben ausgibt. Wenn alles Kompiliert und richtig angeschlossen ist, sollte das Display wie in **Bild 2** aussehen. Damit ist die Hardware grundsätzlich einsatzbereit.
Als nächster Schritt erfolgt die Anbindung von *LittlevGL* an den Display-Treiber und die Erstellung eines eigenen HMI (Human Maschine Interface). Zur Nutzung von *LittlevGl* muss man zuerst die Konfiguration anpassen. Hierzu sucht man im Arduino-Bibliotheksverzeichnis den Ordner *littlevGL*. In der dortigen Datei *lv_config.h* werden Anpassungen an das verwendete Display vorgenommen und die verfügbaren Elemente der Bibliothek eingestellt. Am Anfang der Datei befinden sich die Einstellungen für die Speicherverwaltung. Die Zeile:

```
#define LV_MEM_SIZE (64U * 1024U)
```

definiert das für Objekte der GUI reservierte RAM. Beim angegebenen Wert von 64 kB wird später der Linker bemängeln, dass so viel nicht bereit gestellt werden kann. Bei statischem Reservieren (zur Compile-Zeit) macht sich der nicht durchgehende Speicherbereich des ESP32 bemerkbar. Man könnte nun zur Laufzeit via `malloc` und `free` entsprechende Blöcke reservieren. Da diese Maßnahme andere Gefahren birgt, wird die Sache anders angegangen. Man ändere hierzu die Zeile in:

```
#define LV_MEM_SIZE (24U * 1024U)
```

Dies ist für unsere ersten Schritte ausreichend.
Das Display hat eine Auflösung von 480 x 320 Pixel. Die zugehörigen `#defines` sind:

```
/* Horizontal and vertical resolution of the library */
#define LV_HOR_RES_MAX (480)
#define LV_VER_RES_MAX (320)
```

Die Auflösung in DPI (Dots Per Inch) ergibt sich nach folgender Formel:

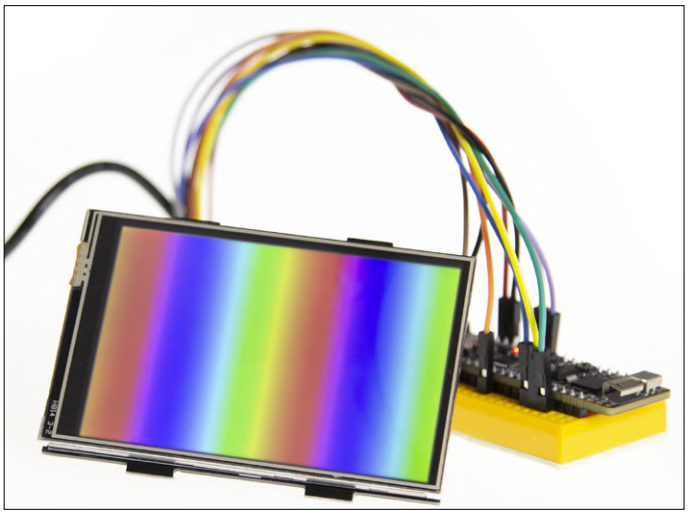


Bild 2. Erster Test: Das Display zeigt Regenbogenfarben.

$$DPI = \frac{\sqrt[2]{(horizontal\ resolution)^2 + (vertical\ resolution)^2}}{screen\ diagonal\ in\ inch}$$

Eingesetzt:

$$DPI = \frac{\sqrt[2]{(480)^2 + (320)^2}}{3,5} = 164,83$$

Ganzzahlig ergibt sich daraus:

```
#define LV_DPI 164
```

Soviel zur Basiseinstellung. Für erste Tests bleiben die restlichen Einstellungen unangetastet; die Änderungen werden gesichert. In der Arduino-IDE kann man nun unter *LittlevGL* das Beispiel *ESP32_TFT_eSPI* auswählen und auf das ESP32-Board laden. Ist alles richtig konfiguriert, sollte sich „Hallo Arduino!“ auf weißem Grund auf dem Display zeigen.

Treiber und *LittlvGL* kooperieren also gut. Allerdings wurden der Touch-Controller des Displays noch nicht ausgelesen und diese Daten nicht an die Bibliothek übergeben. Nachfolgend daher die grundlegenden Code-Teile, womit man ein Grundgerüst für eine eigene Anwendung erstellen kann. Dazu wird das Beispiel *ESP32_TFT_eSPI* aus der LittlevGL-Bibliothek, das gerade in den ESP32 geladen wurde, näher angeschaut.
In der Funktion `setup()` folgen nach der Initialisierung der Bibliothek in Zeile 63 mit `lv_init()` und des TFTs in den Zeilen 69 und 70 mit `tft.begin()` und `tft.setRotation(1)` schließlich die Zeilen 73 und 74 mit der Initialisierung des Structs `lv_disp_drv_t`. Diesem Struct wird ein Funktions-Pointer für das Schreiben auf das Display mitgegeben und danach in der Bibliothek registriert.
Ein ähnliches Vorgehen findet man beim Dummy-Touch-Treiber in den Zeilen 80…84. Als letzter Schritt wird mit Hilfe eines „Tickers“ eine Zeitbasis für die Bibliothek bereitgestellt. Hierbei wird eine Funktion im vorgegebenen Intervall von 20 ms aufgerufen. Jedes Mal wird ein Zeitzähler um 20 ms erhöht.

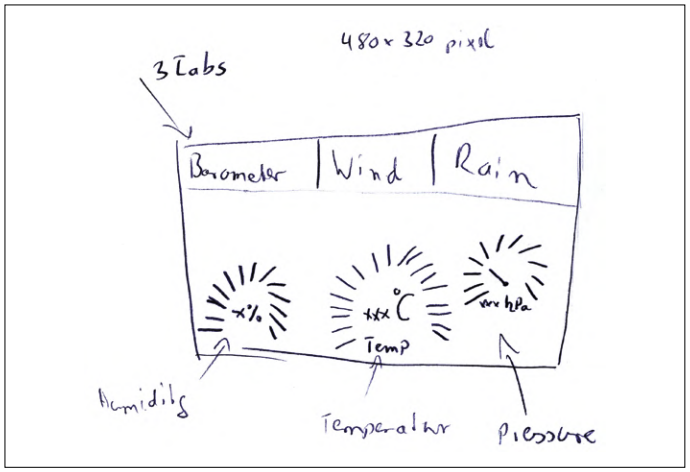


Bild 3. Beispiel einer Handskizze für die Anzeige von Luftdruck, Temperatur und Luftfeuchte mit drei Tabs.

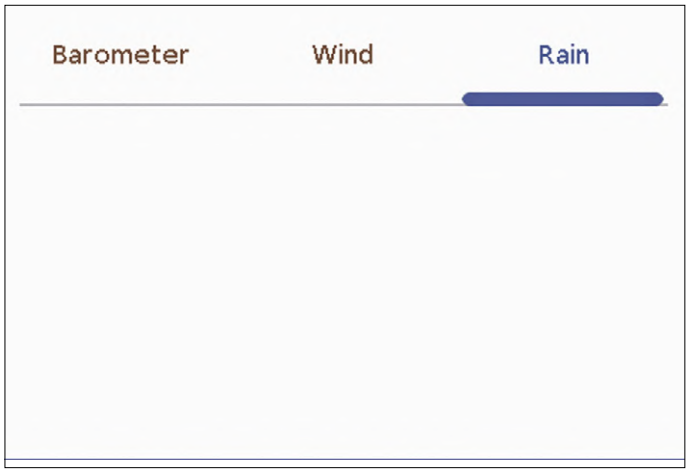


Bild 4. Bildschirm mit drei Tabs ohne weiteren Inhalt.

Abschließend wird dann eine Schaltfläche angelegt und dieser der Text „Hallo Arduino!“ zugewiesen (Zeilen 90…92). Innerhalb der Loop-Funktion wird nun noch `lv_task_handler()` aufgerufen, damit die GUI auf Eingaben reagieren oder den Bildschirm neu zeichnen kann.

Damit man nicht bei jedem Projekt wieder bei Null anfangen muss, hat der Autor ein Basis-Projekt erstellt, in dem die Einstellungen für das Display von JOY-iT und seinen Touch-Controller sowie die Initialisierung der Komponenten vorgenommen werden. In Zeile 139 des Sketches wird die Orientierung des Displays mit `tft.setRotation(3)` eingestellt. Das Bild wird so um 270° gegenüber der Ausgangsposition gedreht. Braucht ein anderes Display z.B. eine Drehung um 180°, muss der Parameter auf 1 gesetzt werden.

GUI-Erstellung

Mit diesem Grundgerüst kann man anfangen, ein eigenes GUI zu erstellen. Dies kann man direkt auf der ESP32-Hardware erledigen, doch das Kompilieren, Hochladen und Testen nimmt Zeit in Anspruch. Die Alternative ist ein PC-Simulator. Seine Installation setzt die Vertrautheit mit *Eclipse* voraus. Die Installation wird unter [6] beschrieben. Sie ist unter Windows etwas schwieriger als unter Linux oder OS X. Im Simulator können nun die ersten Schritte ausprobiert werden, ohne jedes Mal modifizierten Code auf die Hardware laden zum müssen.
Zuerst geht es um das Design der Oberfläche, wozu man am besten zunächst zu Stift und Papier (oder zu Tablet und Stift) greift, denn vor den ersten Code-Zeilen sollten Skizzen angefertigt werden. **Bild 3** zeigt ein Beispiel solch einer Hand-Skizze. So wird klar, an welcher Stelle welches Objekt platziert und mit welchen Objekten navigiert wird.
Da es um eine Wetterstation geht, wird für die Anzeige der Wetterdaten eine einfache Oberfläche mit drei Tabs erstellt. Damit der Arduino-Sketch übersichtlich bleibt, werden die Funktionen und die Erstellung der GUI-Elemente in einer eigenen Datei abgelegt.
Zuerst wird die Szene vorbereitet und ein Tabview-Element angelegt, dem dann drei Tabs hinzugefügt werden: *Barometer*, *Wind* und *Rain*. Für die Vorbereitung der Szene ist der folgende Code zuständig:

```
lv_theme_set_current(th);

/* Next step: create a screen */
lv_obj_t * scr = lv_cont_create(NULL, NULL);
lv_scr_load(scr);
```

Zuerst wird das Theme geladen, das als Funktionsparameter übergeben wurde. Anschließend wird eine leere Szene angelegt und geladen. Das Tabview-Element bekommt die komplette Bildschirmgröße zugewiesen. Der Screenshot in **Bild 4** zeigt die drei leeren Tabs mit den durch das Theme vorgegebenen Schrifteinstellungen. Klickt man auf den Namen des entsprechenden Tabs, wird ein Wechsel der Tabs durch den blauen Marker angezeigt. Da die Tabs leer sind, sieht man nicht mehr als dies.
Mit den folgenden fünf Code-Zeilen

```
/* And now populate the four tabs */
lv_obj_t * tv = lv_tabview_create(scr, NULL);
lv_obj_set_size(tv, LV_HOR_RES_MAX, LV_VER_RES_MAX);
lv_obj_t * tab0 = lv_tabview_add_tab(tv, "Barometer");
lv_obj_t * tab1 = lv_tabview_add_tab(tv, "Wind");
lv_obj_t * tab2 = lv_tabview_add_tab(tv, "Rain");
```

werden die ersten drei Tabs angelegt. Zunächst haben sie noch keinen Inhalt und werden daher anschließend mit Titeln versehen.

Wetter-Anzeige

Los geht es mit dem Barometer: Hier sollen drei Werte für Luftfeuchtigkeit, Temperatur und Luftdruck angezeigt werden. Für Luftfeuchtigkeit und Temperatur werden `lv_lmeter` und `label` verwendet, welche Wert und Namen der Messgröße anzeigen. Für die Luftfeuchtigkeit wird `lv_gauge` genutzt. Bequemerweise kann man das Aussehen der Elemente noch zur Laufzeit durch Styles beeinflussen und so jedes Element individualisieren.
Bei der Anordnung von Elementen muss auf die Autofit-Funktion der Bibliothek geachtet werden. Man muss also die Elemente passend anordnen oder Autofit abschalten. Elemente lassen sich von mehreren Ursprungskoordinaten aus positionieren – eine Übersicht findet sich unter [7]. Die einzelnen Elemente

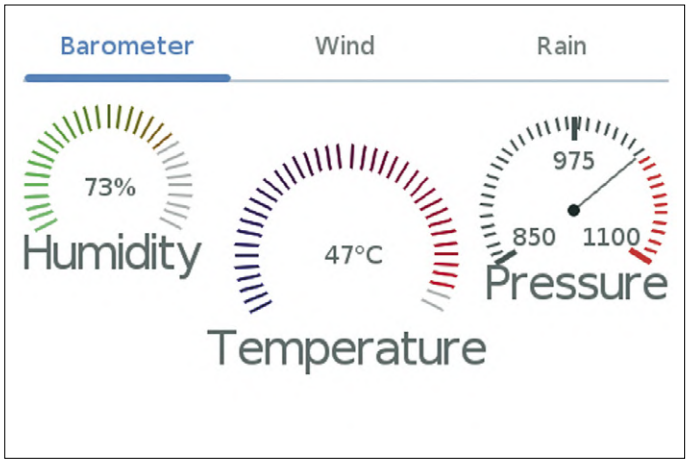


Bild 5. Screenshot des 1. Tabs mit Barometerwerten.

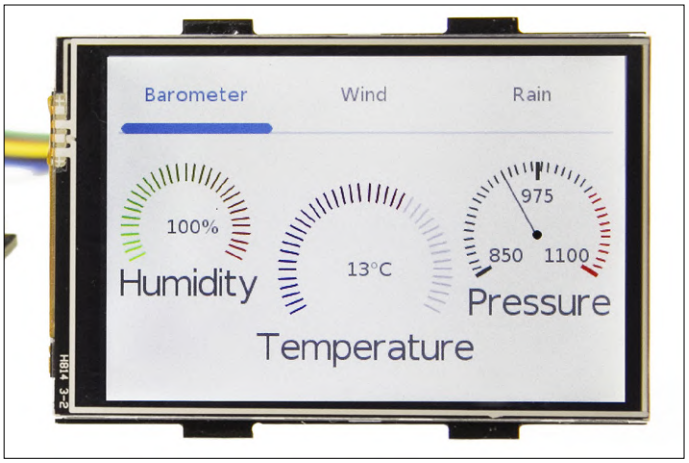


Bild 6. Barometer auf realem Display.

können *Parents* besitzen, also Objekte, von denen ihre Position abhängt. Auf diese Weise ergeben sich elegante Abhängigkeiten der Positionen, wodurch sich bei einer Parent-Verschiebung auch alle *Childs* neu ausrichten. Im Code kann auf die angelegten Elemente später nicht mehr direkt zugegriffen werden. Für *LMeter* und *Gauges* nutzen wir daher Pointer, auf die man global zugreifen kann. Im Beispiel-Code

```
lv_obj_t* humidity_lmeter
lv_obj_t* humidity_label
lv_obj_t* temp_lmeter
lv_obj_t* temp_label
lv_obj_t* air_pressure_gauge
```

fällt auf, dass Funktionen wie `lv_lmeter_create` immer nur Pointer zurückliefern. Die Frage ist nun, wo Speicher reserviert wird. Dies ist etwas tiefer in der Bibliothek vergraben. Der Ausdruck:

```
# define LV_MEM_SIZE (24U * 1024U)
```

legt einen festen Speicher-Pool für die Grafikelemente an. Bei jedem Aufruf einer Create-Funktion wird etwas Speicher aus dem Pool genommen und für das Grafikobjekt reserviert. Das Ergebnis der Operation ist ein Pointer auf die Speicheradresse, der dazu genutzt wird, Eigenschaften des Objekts zu ändern. Falls irgendwann der Speicher ausgeht – bei dynamischen Oberflächen kann das passieren – wird in der Bibliothek ein Fehler ausgelöst und in eine Endlosschleife verzweigt. Der ESP32 stoppt komplett.

Die Pointer sind erst mal nur in der Funktion gültig, in der man sich gerade befindet. Will man später ohne Umwege auf ein Element zugreifen, so müssen die Pointer außerhalb der Funktion abgelegt werden. Der Einfachheit halber dienen hierzu bei uns ein paar globale Variablen. Bei ernsthaften Anwendungen ist von der Verwendung globaler Variablen aber abzuraten.

Via Pointer kann man dann neue Werte in die Anzeigen schreiben. Exemplarisch hierfür ist z.B. die Funktion `UpdateTemperature`. Beim Anzeige-Element *Lmeter* wird ein Wert zwischen 0 und 100 erwartet, doch die Größe hat einen Wertebereich von ±50°. Die Temperatur muss also mit einem Offset von 50 versehen werden. 0° entsprechen dann einem *Lmeter*-Wert von 50. Zusätzlich wird die aktuelle Temperatur noch als Text angezeigt. Dies wird durch `snprintf` und einen kleinen lokalen Puffer erledigt, der als neuer Text in das Textfeld geschrieben wird. Ändert sich die Textlänge, wird der Text nicht automatisch neu ausgerichtet. Die Ausrichtung muss nach dem Setzen des Textes erneut vorgenommen werden. Hierzu wird `lv_obj_align` mit den Parametern für das Label erneut aufgerufen. Luftfeuchtigkeit und Luftdruck werden ganz ähnlich behandelt. **Bild 5** zeigt einen Screenshot des fertigen Tabs und in **Bild 6** kann man sehen, wie das auf dem LCD „in echt“ aussieht.

Weblinks

[1] LittlevGL: https://github.com/littlevgl/lv_arduino

[2] „Wetterstation mit ESP32“, ElektorLabs 1-2/2019: www.elektormagazine.de/180468-03

[3] TFT_eSPI: https://github.com/Bodmer/TFT_eSPI

[4] RA8875-Library: https://github.com/adafruit/Adafruit_RA8875

[5] u8g2-Library: https://github.com/olikraus/u8g2

[6] PC-Simulator: https://docs.littlevgl.com/en/html/get-started/pc-simulator.html

[7] Objekt-Positionierung: https://docs.littlevgl.com/en/html/overview/object.html#object-s-working-mechanisms

[8] „Monster-LED-Uhr“, ElektorLabs 5-6/2019: www.elektormagazine.de/180254-01

[9] Webseite zum Artikel: www.elektormagazine.de/190295-01

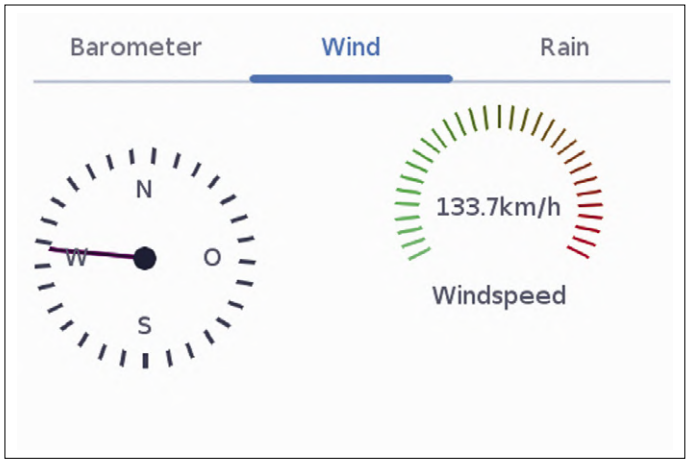


Bild 7. Tab mit Windrichtung und Geschwindigkeit.

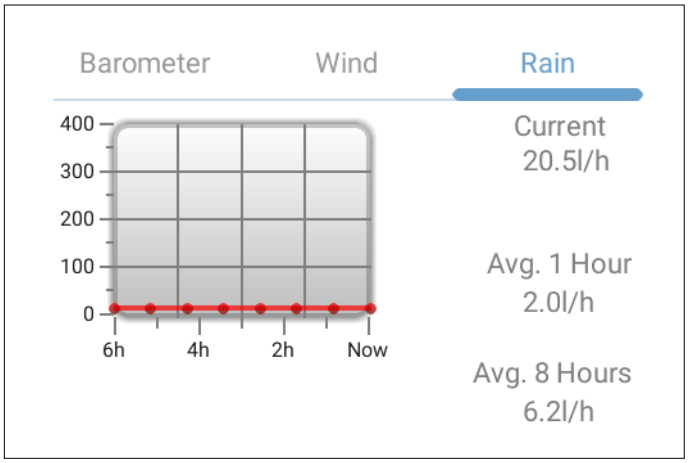


Bild 8. Screenshot des Tabs mit Niederschlagsverlauf und Werten.

Der erste Tab wurde nun mit Inhalten versehen. Beim zweiten Tab wird ähnlich verfahren, jedoch erfordert die Windrichtungsanzeige als Kompass etwas mehr Aufwand. Hier fungiert ein *Gauge* als *Parent* für vier *Label*. Im Code wird zunächst ein *Gauge* mit 0…359° angelegt. Darauf folgen vier *Label*, denen der Kompass als *Parent* verpasst wird. Die *Label* werden auf dessen Mitte bezogen definiert. Somit ergeben sich die vier Himmelsrichtungen. Der Zeiger weist in die Richtung, aus welcher der Wind kommt. Beim *Gauge* ergeben sich 0° nicht beim Wert 0, sondern bei 180. Zur Anzeige der Windgeschwindigkeit wird wieder ein *Lmeter* mit ähnlichem Aufbau wie beim Barometer genutzt. Man merkt, dass sich für das Anlegen der Elemente Vorgänge wiederholen. Zunächst wird ein Style für das Objekt angelegt, dann erfolgt das Erstellen des Objekts und zum Schluss werden ihm seine Eigenschaften zugewiesen. Das fertige Resultat ist im Screenshot von **Bild 7** zu sehen.
Beim Regen bzw. Niederschlag sieht die Darstellung der Werte anders aus. Hier werden die Werte in Textform sowie in einem Diagramm dargestellt, aus dem sich der zeitliche Verlauf ergibt. Die Texte werden wie schon zuvor beschrieben realisiert: Erst werden Styles und Objekte angelegt – dann die Werte zugewiesen. Für den Verlauf der Regenmenge eignet sich ein Liniendiagram, welches seit der Version 6 auch keine Tricks zur Beschriftung der Achsen mehr benötigt. Zur Aktualisierung der Werte muss aber nicht jeder Datenpunkt einzeln bewegt werden, denn `lv_chart_set_next` erledigt diese Aufgabe. Einmal pro Stunde wird ein neuer Wert an das Diagramm übergeben. Die Aktualisierung der Niederschlagsmengen erfolgt wie bei anderen Texten auch durch eine eigene Funktion. **Bild 8** zeigt einen Screenshot mit Pseudodaten des Verlaufs und der Werte des Niederschlags.

Für die Datenanbindung des Displays wird Code des Projekts Monster-LED-Uhr [8] recycelt, da hier schon Daten per MQTT von einem Broker kommend verarbeitet werden. Der Code erwartet, dass der Broker einen JSON-String sendet, in dem die Werte für Luftfeuchtigkeit, Temperatur, Luftdruck, Windrichtung, Windgeschwindigkeit und die Niederschlagsmengen enthalten sind. Wenn vom Broker neue Daten kommen, werden diese in die zugehörigen Elemente eingetragen. Man muss allerdings darauf achten, dass dies nicht durch unterschiedliche Threads passiert. Auch bei der Konfiguration gibt es bis auf die fehlende Uhrzeit-Einstellung keine großen Unterschiede zum LED-Uhr-Projekt [8]. WLAN- und MQTT-Einstellungen wurden schlicht übernommen, es müssen lediglich die Wetterstation und die Anzeige auf das gleiche Topic eingestellt werden. Ab da gelangen die Werte direkt aufs Display. Eine Ausnahme ist im Moment noch die Regenmenge, denn hier wird nur die aktuelle Regenmenge der Wetterstation ausgegeben. Die Berechnung von Stundenwerten und dem Verlauf sind bei der Wetterstation leider noch nicht implementiert. Sobald dieses aber geschehen ist, werden auch diese Werte auf dem Display aktualisiert.

Fazit

Mit diesem Beispiel wurden einige grundlegenden Funktionen von *LittlevGL* praktisch demonstriert. Der zugehörige Code für diese Demonstration kann wie immer kostenlos von der Elektor-Webseite zu diesem Artikel [9] heruntergeladen werden. Wie schon gesagt, stellt *LittlevGL* aber noch deutlich mehr Funktionen und Animationen sowie Gestaltungsmöglichkeiten mit Tabellen, Listen und Dropdown-Menüs zur Verfügung. Wenn Sie nun von der Einfachheit der Nutzung dieser Bibliothek angetan sind, werden Sie *LittlevGL* sicherlich auch in eigenen Projekten ausprobieren wollen. Ein EPS32-Modul in Verbindung mit einem Touch-Display ist ja schließlich eine recht universell einsetzbare Plattform, die viel Leistung für wenig Geld bietet. ⏮

190295-01

UNSER RND SORTIMENT HILFT IHNEN BUDGET-WIDERSTÄNDE ZU UMGEHEN

The best part of your project: www.reichelt.de/rnd

Gute Qualität, niedrige Preise.

RND ist die Eigenmarke von reichelt, die Ihnen einen einfachen Zugang zu preislich attraktiven Komponenten ermöglicht, ohne Abstriche bei der Qualität. Das Sortiment umfasst über 8000 Produkte aus den Bereichen Bauteile, Automation, Kabel, Gehäuse, Werkzeuge und Laborausstattung:

RND lab · RND power · RND components · RND connect · RND cable

Power-Lötkolben mit 150 Watt

RND lab

schnelle Aufheizzeit: auf 480 °C in nur 1 Minute

Für hohe Sicherheit wechselt der Lötkolben bei Inaktivität nach 10 Minuten automatisch in den Standby-Modus und schaltet nach 20 Minuten vollständig ab.

- temperaturgeregelt - Einstellung per Tastendruck
- Temperaturbereich: 250 - 520 °C
- ergnomischer Griff mit integriertem LCD zur Temperaturanzeige

BESTSELLER ~~49,95~~

Bestell-Nr.: **RND 560-00216**

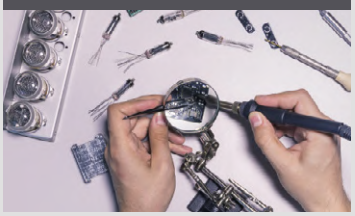
39,95

150 WATT

THEMEN-SPECIAL:
Wissenswertes rund ums Thema Löten

Löten ist ein wichtiger Bestandteil der Arbeit mit und an elektronischen Bauteilen. Für erfolgreiche Projekte bedarf es zum einen des richtigen Equipments, zum anderen der richtigen Löttechnik.

Jetzt informieren
▶ www.rch.lt/MG282



Lötzinn mit Kupferanteil

100 g, Sn99.3/Cu0.7

- Anteil Flussmittel 2.5 %
- Lötdrahtdurchmesser 1 mm
- Schmelzpunkt: 217 °C

PREIS-TIPP

Bestell-Nr.: **RND 560-00169** **7,30**

RND lab

Werkstatt-Lupenleuchte

5-stufig dimmbar

- Glaslinse, 3 Dioptrien
- Farbtemperatur: 6.000 - 6.500 K
- 950 lm Lichtstrom

A+ (A++ - E)

Bestell-Nr.: **RND 550-00122** **84,95**

RND lab

- Top Preis-Leistungs-Verhältnis
- über 110.000 ausgesuchte Produkte
- Zuverlässige Lieferung – aus Deutschland in alle Welt.

Bestellservice: +49 (0)4422 955-333

www.reichelt.de

reichelt elektronik – The best part of your project

„Kein Projekt für jedermann“

Interview mit Gábor Kiss-Vámosi, dem Entwickler von LittlevGL

Fragen von **Mathias Claußen** und **Jens Nickel**

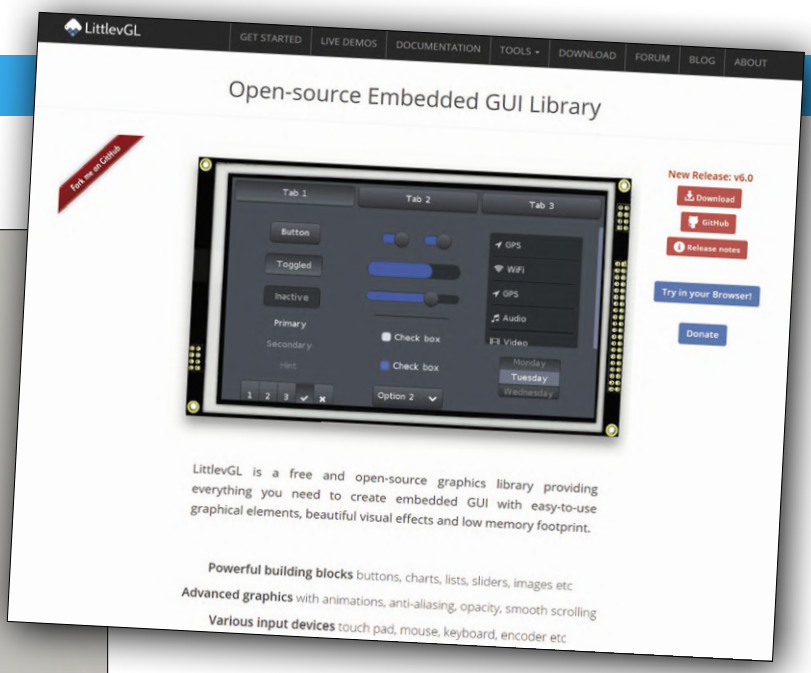
Heutzutage funktioniert Softwareentwicklung nicht mehr ohne Open-Source-Bibliotheken und Frameworks. Einige dieser Bibliotheken werden von jungen Software-Enthusiasten privat entwickelt, weil sie Code für sich selbst benötigen. Dann wird das Projekt schnell größer und professioneller. Ein Beispiel dafür ist die LittlevGL-Bibliothek von Gábor Kiss-Vámosi, die eine grafische Benutzeroberfläche mit Touchfunktion auch für 16-Bit-Mikrocontroller mit kleinem RAM ermöglicht (siehe Artikel in dieser Ausgabe). In diesem Interview spricht Gábor über den Hintergrund, einige interessante Details und die Zukunft seiner beliebten Bibliothek.

Elektor: Gábor, warum sollten wir deine GUI-Bibliothek verwenden? Es gibt ja schon einige, die Open Source sind und (wie µGFX) für nicht-kommerzielle Zwecke frei verwendet werden können.
Gábor: µGFX und LittlevGL haben zwar einige vorteilhafte Gemeinsamkeiten wie Open Source- und Plattformunabhängigkeit, aber LittlevGL kann „glatte“ Animationen, Anti-Aliasing, Opazität und Schattenwürfe ohne doppelte Pufferung verarbeiten. Deshalb benötigt LittlevGL nur einen Frame-Buffer im Display-Controller-Chip oder ein kleines Grafik-RAM neben der MCU. Ein weiterer Vorteil ist, dass LittlevGL eine Navigation und Steuerung mit Tastern oder sogar einem Drehencoder unterstützt. LittlevGL versteht auch MicroPython: Eine Benutzeroberfläche kann mit Python3-Code geschrieben werden, ohne die MCU in der Entwicklungsphase neu programmieren und flashen zu müssen.

Elektor: Was ist dein Hintergrund, dein Beruf? Wieso investierst du so viel Zeit in ein Softwareprojekt?
Gábor: Ich bin Elektronikingenieur und arbeite beruflich an einem Hardware-Security-Projekt. Während des Studiums habe ich mit einem meiner Freunde in der Freizeit und aus Spaß an der Freude Dinge wie Verstärker, Instrumente und andere Gadgets entwickelt. Wir wollten diese Sachen mit einem schönen Grafikdisplay anstelle von schnöden 7-Segment- oder 2x16-LCD-Displays ausstatten. Ich habe gesagt: „Okay, kaufen wir ein TFT-Display und versuchen, etwas darauf zu zeichnen. Wenn ich erst einmal ein Pixel setzen kann, sollte der Rest ganz einfach sein.“ Schließlich hat es nur zwei Stunden gedauert, bis das erste Pixel gesetzt war, und in den folgenden acht Jahren habe ich mich mit dem „einfachen Rest“ beschäftigt!
Ich lerne gerne und finde gerne neue Dinge heraus, es ist wie ein Hobby. Als das Projekt bekannt wurde, bekam ich glücklicherweise viel Feedback von anderen, erfahreneren Entwicklern. Sie haben Verbesserungsvorschläge gemacht und mir gesagt, welche Funktionen sie noch vermissen. Eine solche Herausforderung kann ich schlecht ablehnen, denn ich fühle mich persönlich für mein Projekt verantwortlich, weil die Menschen es nutzen und darauf vertrauen.

Elektor: Wie viel Zeit investierst du in das Programmieren, wie viel in andere Aufgaben?
Gábor: Vor ein paar Jahren war es einfacher, aber jetzt habe ich Familie. Ich muss ein Gleichgewicht zwischen beidem finden. Normalerweise stehe ich früh auf, wenn meine Frau und mein Kind noch schlafen, um E-Mails und Anfragen zu beantworten und an neuen Funktionen zu arbeiten. Am Nachmittag, in meiner Freizeit, versuche ich immer noch ein wenig an der Bibliothek zu arbeiten. Wenn ich die Stunden zähle, ist es neben meiner beruflichen Tätigkeit mehr als eine Freizeitbeschäftigung. Es ist eine ziemliche Herausforderung, aber nicht unmöglich, Zeit für die berufliche Arbeit, LittlevGL, Familie, Freunde und andere Freizeitaktivitäten zu haben.
Am Anfang, als ich LittlevGL veröffentlicht hatte, musste ich eine Menge nicht-programmierbarer Dinge wie Marketing, Suchmaschinenoptimierung, Webentwicklung, Grafikdesign und so weiter erlernen. Ich erinnere mich, dass es Monate gedauert hat, SEO-Tipps und -Tricks zu studieren, und ich habe den ganzen Tag lang Websites analysiert. Aber es hat sich gelohnt, denn wenn man jetzt nach „embedded GUI“ sucht, dürfte bei Google LittlevGL an erster Stelle stehen.

Elektor: Erhältst du beim Code-Schreiben und anderen Aufgaben Hilfe der Community oder von Unternehmen?
Gábor: Glücklicherweise haben sich im Laufe der Zeit immer mehr Mitwirkende dem Projekt angeschlossen und es auf vielfältige Weise unterstützt. Einige von ihnen teilen mir ihre persönlichen Verbesserungen mit, zum Beispiel: „Hey, ich habe dem Textbereichsobjekt eine Platzhalterfunktion hinzugefügt. Bist du interessiert?“ Einige Mitwirkende fügten bedeutende Verbesserungen bei; so stammt die MicroPython-Anbindung von „amir-



Gábor Kiss-Vámosi (29) lebt in Budapest (Fotos: Ramóna Erdei).

gon“ und der neue Font-Konverter von „puzrin“. Andere helfen bei der Beantwortung von User-Anfragen und beheben Bugs. „embeddedt“ zum Beispiel verbringt täglich Stunden damit, andere User zu unterstützen. Was diese Leute tun, ist sehr wichtig und wertvoll. LittlevGL wäre ohne sie nicht das, was es ist!

Elektor: Hast du bei der Arduino-Portierung Hilfe von der Community erhalten?
Gábor: Die Arduino-Portierung war eine knifflige Aufgabe. Die Struktur der Bibliothek musste völlig geändert werden, um mit dem Build-System von Arduino zusammenzuarbeiten. Ich habe die ersten Änderungen selbst vorgenommen, aber vor ein paar Monaten hat „Pablo2048“, der sehr viel mehr Erfahrung mit Arduino hatte als ich, ein überarbeitetes Arduino-Package veröffentlicht, zu dem er auch User-Fragen beantwortet.

Elektor: Benutzer können für dein Projekt spenden. Sind die Spenden eine große Unterstützung für dich oder tragen sie nur einen kleinen Teil bei?
Gábor: Die Spenden reichen in etwa aus, um die Kosten des Projekts wie Webserver, Kauf von Entwicklungsboards und so weiter zu decken, aber ich kann nicht davon leben.

Elektor: Wie kommt man zu neuen Ideen für die Bibliothek? Erhältst du Feedback von Entwicklern?
Gábor: Manchmal fragen User explizit nach einem Feature. Darüber hinaus beschäftige ich mich mit oft gestellten Fragen, da etwas entweder in der Bibliothek oder in der Dokumentation verbessert werden muss. In dieser Hinsicht war beispielsweise für Version 6.0 eine konzeptionelle Änderung der Bibliothek erforderlich. Neben den Vorschlägen der User sind es moderne Trends im Grafikdesign und bei eingebetteten Benutzeroberflächen, die mich anregen, Lösungen zu entwickeln. Ich nutze die Bibliothek auch selbst, bei mehreren Projekten, die ich als Freelancer für Unternehmen entwickelt habe, aber auch in meiner beruflichen Tätigkeit.

Elektor: Gehst du auch zu Messen und Events? Hast du die Möglichkeit, mit den Nutzern deiner Bibliothek persönlich zu sprechen?
Gábor: Um ehrlich zu sein, vermisse ich die persönliche Begegnung mit den Usern. Es wäre schön, sie persönlich zu treffen, aber sie sind auf der ganzen Welt verteilt und das ist dann nicht so einfach.
Einige Leute haben mir angeboten, sie zu besuchen, ich könne sogar bei ihnen übernachten. Und tatsächlich wollen wir in unserem nächsten Urlaub einen LittlevGL-User besuchen. Ich freue mich wirklich über diese Gelegenheit.
Der ganze Background von LittlevGL wird sich zu einer wirklich professionellen Software entwickeln und nicht nur zu einem Projekt für jedermann. Deshalb planen wir auch die Teilnahme an Konferenzen, Webinaren und Seminaren.

Elektor: Für die Verwendung von LittlevGL sind 16-, 32- oder 64-Bit-MCUs erforderlich. Warum gibt es keine Unterstützung für die 8-Bit-MCU-Familie?
Gábor: Im Prinzip gibt es kein Problem mit 8-Bit-MCUs, aber normalerweise haben sie so wenig RAM, dass sie LittlevGL nicht ausführen können. Deshalb sind für den Betrieb von LittlevGL mindestens 16 kB RAM erforderlich, so dass es selbst bei 16-Bit-MCUs schon knapp werden kann.

Elektor: Kannst du uns etwas über die technischen Grundlagen der Bibliothek erzählen?
Gábor: Die Grundidee ist, dass man Objekte (oder mit anderen Worten Widgets) wie Schaltflächen, Labels, Skalen, Schieberegler erstellt und einfach ihre Eigenschaften wie Größe, Position, Stil, Wert oder Status und Callbacks festlegt, so dass der User erkennt, wenn etwas passiert ist (zum Beispiel, dass ein Button angeklickt wird). So werden alle Grafiken und andere zugrunde liegende Dinge von LittlevGL im Hintergrund erledigt, so dass sich der User nur mit den übergeordneten Dingen befassen muss.
Ein Objekt kann in Echtzeit angelegt und gelöscht werden. Auf diese Weise, wenn nur das gerade benötigte Objekt „am Leben gehalten“ wird, kann der Speicher optimal genutzt werden. Wenn man beispielsweise eine Message-Box benötigt, erstellt man sie einfach und löscht sie, wenn der Benutzer auf den OK-Button klickt. Die Message-Box beansprucht also nur dann

Webtools für ein Softwareprojekt

Die Programmierung ist die eine Seite eines Open-Source-Softwareprojekts, aber es ist auch eine Dokumentation vonnöten und die Möglichkeit, Anwender zu unterstützen. So berichtet uns Gábor Kiss-Vámosi über seine Website und andere Tools zur Unterstützung seiner Bibliothek:

Als Elektronikingenieur wusste ich nicht viel über Webentwicklung. Zuerst habe ich WordPress ausprobiert, aber als ich es nach meinen Vorstellungen anpassen wollte, stieß ich schnell an die Grenzen. Später habe ich Bootstrap gefunden und ich konnte die Website von Grund auf neu erstellen. Es ist selbst für Anfänger ziemlich einfach, mit Bootstrap umzugehen, weil es gut aussehende Elemente besitzt und Responsivität quasi integriert ist. Später wurden dann die „side topics“ in eigene Subdomains verschoben.
Der Blog wird auf GitHub gehostet, wo die Beiträge in Markdown geschrieben werden. Eine Engine namens Jekyll erstellt aus den Markdown-Dateien automatisch eine statische Website. Auf diese Weise kann jeder auf einfache Weise Beiträge zum Blog hinzufügen, es muss nur ein Pull-Request gesendet werden.
Auch die Dokumentation, die offline mit Sphinx erstellt wurde, wird auf GitHub gehostet. Die Dokumente können von den Usern auf einer Online-Plattform namens Zanata übersetzt werden.
Das Forum verfügt über eine Diskussionsmaschine, die auf einem DigitalOcean-Virtual-Server läuft. DigitalOcean unterstützt Open-Source-Projekte; ich musste mich nur bewerben und erhielt eine jährliche Patenschaft, über die ich nach Belieben verfügen kann. So habe ich einen Virtual Private Server (VPS) gestartet, um das Forum zu hosten. „seyyah“ von GitHub hat bei der VPS-Konfiguration sehr geholfen.
Mein Ziel war es, eine Gemeinschaft aufzubauen, in der Menschen ihr Wissen teilen und über ihre Projekte und Erfahrungen sprechen können. Deshalb hat LittlevGL einen offenen Blog, in dem jeder Beiträge schreiben kann. Und es gibt eine Kategorie „My projects“ im Forum, die man teilen kann, wenn man etwas Interessantes geschaffen hat.

Platz im Speicher, wenn sie gerade angezeigt wird.
Die Objekte besitzen eine Eltern-Kind-Hierarchie. Man kann beispielsweise einen übergeordneten Container erstellen und ihm drei Schaltflächen als Kinder zuordnen. Wird der Container verschoben, bewegen sich die Tasten mit. Wird der Container gelöscht, verschwinden auch die Schaltflächen.
Ein weiteres interessantes Konzept von LittlevGL ist eine (rudimentäre) Art der Objektorientierung. Jedes Objekt wird vom Basisobjekt abgeleitet, das nur die gängigsten Eigenschaften wie Position, Größe oder Parent definiert. Dieser Mechanismus ist in C++ üblich, aber in C einzigartig.

Elektor: Wir können verschiedene vorgefertigte Widgets für die UI verwenden, aber wenn wir etwas Neues benötigen, gibt es dann eine Anleitung, die bei der Erstellung hilft?
Gábor: Alle Widgets funktionieren auf die gleiche Weise. Sie besitzen einige benutzerdefinierte Eigenschaften, eine Designfunktion, die das Widget zeichnet, und eine Signalfunktion, die interne Events auf einer niedrigen Ebene verarbeitet. Grundsätzlich gilt: Wenn man den Code des Widgets liest und versteht, ist es kein Problem, ein neues Widget auf die gleiche Weise zu erstellen. Es gibt c- und h-Dateien als Vorlagen, die es erleichtern, ein eigenes Widget zu erstellen.

Elektor: Ist das RAM für die UI-Elemente statisch belegt? Oder haben wir es mit einer Art Pool zu tun, in dem uns irgendwann der Speicherplatz ausgehen könnte?
Gábor: LittlevGL besitzt einen eigenen Speichermanager, der dynamisch den Speicher für die Widgets und andere verwandte Dinge zuweist. Er ordnet Daten in einem großen Array an, dessen Größe angepasst oder sogar in einen externen Speicher gestellt werden kann. Im Gegensatz zu den Standards „malloc“ und „free“ überwacht LittlevGL Nutzung und Fragmentierung des Speichers.
Also ja, der Speicher könnte zu knapp werden, aber der Speicherverbrauch lässt sich überwachen und die Größe des für LittlevGL reservierten Speicherbereichs anpassen. Wenn der Speicher dann doch zu knapp wird, erhält man eine Fehlermeldung mit einer Zeilennummer und das Programm wird an dieser Stelle angehalten. Auf diese Weise lässt sich der Fehler leicht an Ort und Stelle beseitigen.

Elektor: Wie wird das Zeichnen verwaltet? Gibt es Tricks, die mehrfaches Zeichnen sich überlappender Objekte verhindert?
Gábor: Es gibt viele Tricks, die den Zeichenvorgang optimieren. Auf der obersten Zeichenebene ermittelt LittlevGL, welche Bereiche neu gezeichnet werden müssen, wenn zum Beispiel eine Taste gedrückt wird, und zeichnet diese Bereiche alle paar Millisekunden (ungefähr 30 ms) neu. Vor dem Neuzeichnen wird im Hintergrund festgestellt, welches Widget das erste ist, das den neu zu zeichnenden Bereich abdeckt. Wenn man also den Text auf der Schaltfläche ändert, werden nur die Schaltfläche und das Label neu gezeichnet, nicht aber der Hintergrund unter der Schaltfläche. Muss jedoch die Position der Schaltfläche geändert werden, so muss auch der Hintergrund, die Schaltfläche und das Label neu gezeichnet werden. Aber das sind ganz triviale Optimierungen.
Die wirklich interessante Sache an LittlevGL ist, dass es nicht wie viele einfache GUI-Bibliotheken direkt auf das Display zeichnet oder wie hoch entwickelte doppelt puffert, sondern mit Kacheln arbeitet. LittlevGL speichert die Daten eines kleineren Teil des

Das Entwickeln beginnt am frühen Morgen.

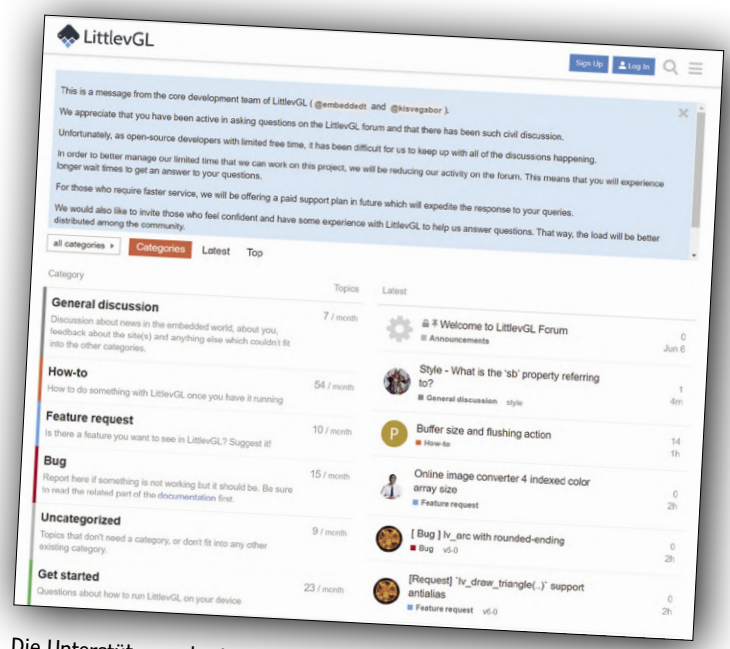
Displays (typischerweise 1/10) und zeichnet den Bereich zuerst dort. Wenn die Zeichnung fertig ist, wird der Puffer mit dem fertigen Bild auf das Display übertragen. Da die Zeichnung gepuffert ist, gibt es während der Übertragung kein Flackern, so dass flüssige Animationen möglich sind.
In der in diesem Jahr erscheinenden neuen Version der Bibliothek ist die Zeichen-Engine komplett überarbeitet. Neben ihrer Flexibilisierung, Erweiterung und einer allgemeinen Verbesserung der Qualität (beispielsweise besseres Anti-Aliasing und Schattenwurf) war das Hauptziel, dass eine Maskierung unterstützt wird. Mit einer Maskierung von Pixeln sind abgerundete Ecken möglich oder Texte mit einem Bildhintergrund oder Farbverlauf. Aber auch einige neue Funktionen wie horizontaler Verlauf, Schattenversatz sowie additive und subtraktive Mischmodi wurden hinzugefügt.

Elektor: Schriftarten können online auf deiner Seite konvertiert werden. Gibt es dazu einen einfach zu bedienenden Offline-Konverter für Schriften? Was ist mit Symbolen?
Gábor: Der Font-Konverter ist in Node.js geschrieben und läuft deshalb online oder offline. Eine Desktop-Anwendung zur Font-Konvertierung haben wir noch nicht, aber man kann den Konverter von GitHub herunterladen und von der Kommandozeile aus verwenden.
Symbole können auch aus Fonts stammen. FontAwesome ist ein bekannter, sehr beliebter Symbolfont, von dem einige Symbole standardmäßig in LittlevGL enthalten sind. Will man jedoch einen eigenen Font erstellen, kann man mehrere Fontdateien auswählen und zu einer einzigen C-Datei verschmelzen. So reicht der angegebene Bereich von Arial bis zu einigen Symbolen von FontAwesome.

Elektor: Die Bibliothek ist nicht threadsicher. Gibt es Pläne, RTOS-Support zu deiner Bibliothek hinzuzufügen?
Gábor: Das stimmt, die Bibliothek ist standardmäßig wirklich nicht threadsicher, aber es ist einfach, sie threadsicher zu machen. Alles, was dafür zu tun ist, ist, einen Mutex (mutual exclusion) zu verwenden, wenn LittlevGL-bezogene Funktionen aufrufgerufen werden, und sie zum geeigneten Zeitpunkt freizugeben.

Elektor: LittlevGL hat eine schöne, auf Eclipse beruhende Umgebung zur schnellen Entwicklung von Benutzeroberflächen, ohne den Aufwand, den Code bei der Entwicklung immer wieder auf echte Hardware flashen zu müssen. Wird es in Zukunft eine Art GUI-Designer geben, der die Entwicklungsphase noch weiter beschleunigt?
Gábor: Auf jeden Fall, ja. Ein solcher Designer wurde bereits mehrfach gefordert. Ich finde es toll, dass einige Benutzer spontan begonnen haben, an GUI-Designern zu arbeiten. Es gibt drei unabhängige Entwicklungen, die alle große Fortschritte gemacht haben. Es ist möglich, dass eine der offizielle GUI-Designer von LittlevGL wird.

Elektor: LittlevGL ist derzeit unter der MIT-Lizenz lizenziert, was eine einfache Integration in kommerzielle Produkte ermöglicht. Bei der letzten Online-Umfrage wurden Fragen zu den Möglichkeiten kommerzieller Lizenzierung gestellt. Was wird sich in Zukunft ändern?
Gábor: Die Popularität von LittlevGL steigt rasant. Ein sehr



Die Unterstützung der User besitzt einen großen Stellenwert. Im Forum beantworten auch erfahrene Bibliotheksnutzer Fragen.

großes Unternehmen hat Interesse an LittlevGL gezeigt. Als Einzelperson zu verhandeln war allerdings schwierig, so dass als nächster sinnvoller Schritt ein Unternehmen gegründet werden sollte, das die Interessen von LittlevGL besser vertreten kann als eine Einzelperson.
Da die Community wächst, wird es immer schwieriger, Entwicklung, Support und Marketing in der Freizeit zu erledigen. Ziel ist es, ein Geschäftsmodell für LittlevGL zu finden, das einigen Personen, die in Vollzeit für LittlevGL arbeiten, ein Gehalt bieten kann, das aber auch für die User akzeptabel ist. In Betracht gezogen werden auch Dienstleistungen wie bezahlter Support mit schnellen Bugfixes, dedizierte professionelle Hilfe und eine Art Lizenzierung. Wir sind noch dabei, das richtige Geschäftsmodell zu finden.

Elektor: Hast du noch Tipps für andere junge Leute, die interessante Open-Source-Softwareprojekte entwickeln?
Gábor: Mein Ratschlag ist, dass man eine Vision und einen Plan haben muss, wenn man ein Open-Source-Projekt startet und veröffentlicht. Handelt es sich nur ein kleines Tool, das nicht wirklich gewartet werden muss? Oder ist es etwas Größeres, das jahrelang viel Zeit in Anspruch nimmt? Was ist das Ziel des Projekts? Was ist die Motivation? Nur zum Spaß? Für Ruhm und Ehre? Für Erfahrungen? Nur, um anderen zu helfen? Man sollte in der Lage sein, diese Fragen im Voraus zu beantworten und entsprechende Entscheidungen rechtzeitig zu treffen. Es kann sehr frustrierend sein, wenn dann Dinge passieren und man nicht weiß, wie man sich entscheiden soll. Sag einfach nein, wenn etwas nicht das ist, was du willst und freue dich über alles, was dich deinem Ziel näher bringt! Wie das Sprichwort sagt: „Es gibt keinen Wind, der richtig weht für den Segler, der nicht weiß, wo der Hafen ist.“

190353-03

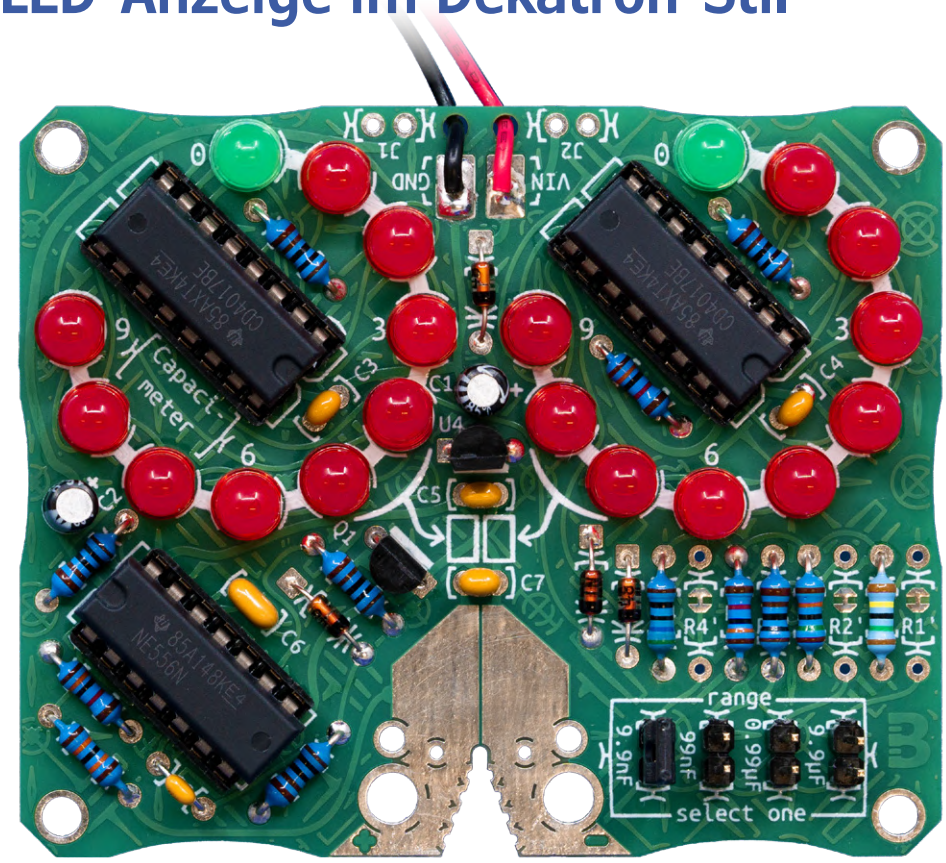
Weblink

[1] LittlevGL Homepage: https://littlevgl.com/

Capaci-Meter

Mit zweistelliger LED-Anzeige im Dekatron-Stil

Von **Jez Siddons** und **Saar Drimer**



Das Kapazitäts-Messgerät basiert auf einem Entwurf, den ich vor über 30 Jahren für einen Technologiekurs entworfen hatte. Anstelle der damaligen dreistelligen 7-Segment-Anzeige habe ich jetzt ein paar LEDs so angeordnet, dass das Gerät den Messwert in zwei Ziffern im Stil einer Dekatron-Zählröhre anzeigt.

Mit nur zwei Ziffern auf dem Display kann das Gerät natürlich nicht als Präzisions-Messgerät konzipiert sein. Stattdessen soll es lediglich anzeigen, ob die Kapazität eines Kondensators nahe des aufgedruckten oder gewünschten Sollwerts liegt. Dies kann bei nicht gerade eng tolerierten Kondensatoren nützlich sein, aber auch, wenn es schwierig ist, die auf dem Bauteil gedruckten Werte abzulesen.

Messbereich und Genauigkeit

Das Messgerät verfügt über vier Messbereiche (**Tabelle 1**). Die minimale Kapazität für jeden Bereich wird durch die Zählerauflösung (sowie Jitter) festgelegt, so dass das Minimum typischerweise doppelt so hoch ist wie die Messauflösung. Wie bei jedem Messgerät ist es gut, einen Messbereich so zu wählen, dass der voraussichtliche Messwert dem maximalen Skalenendwert (Vollausschlag) so nahe wie möglich kommt, aber nicht darüber liegt. Die typischen Fehler der vier Messbereiche liegen in der Größenordnung von ±5%, können aber durch eng tolerierte Widerstände weiter verringert werden. Allerdings kann eine zweistellige Anzeige auch theoretisch nur einen maximalen Fehler von ±1% des Skalenendwerts liefern. Darüber hinaus können Messquantisierung und Jitter einen zusätzlichen Fehler von ±1 Digit hinzufügen. Von daher können Sie, realistisch gesehen, einen minimalen Fehler von ±2% über den gesam-

Eigenschaften

- Minimal messbare Kapazität: 200 pF
- Maximal messbare Kapazität: 9,9 µF
- Leerlaufprüfspannung: 3,3 V
- Maximal zulässige Spannung an den Prüfspitzen: -0,6 V bis +5,6 V
- Eingangsimpedanz: 130 kΩ
- Fehler (unangepasst): ±5% Vollausschlag
- Fehler (mit Feinkalibrierung der Messwiderstände): ±2% Vollausschlag
- Stromaufnahme: 15 mA
- Betriebsspannung: 7,5...15 V_{DC}

Tabelle 1. Das Kapazitäts-Messgerät bietet vier Messbereiche.
Wenn eine Kapazität für den aktuell gewählten Bereich zu groß ist, stoppt die Anzeige bei „99“. Testen Sie im nächsthöheren Bereich, ob die Kapazität wirklich überschritten wurde oder ob sie sich vielleicht exakt bei „99“ befindet.

Bereich	Minimale Kapazität	Maximale Kapazität	Auflösung
1	0,2 nF (200 pF)	9,9 nF	0,1 nF (100 pF)
2	2 nF	99 nF	1 nF
3	0,02 µF (20 nF)	0,99 µF (990 nF)	0,01 µF (10 nF)
4	0,2 µF (200 nF)	9,9 µF	0,1 µF (100 nF)

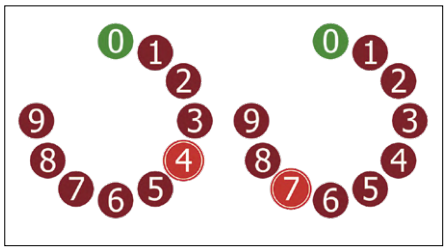


Bild 1. Hier wird der Wert von „47“ angezeigt. Dies entspricht im aktuell gewählten Messbereich 0,0...9,9 µF bei einer Kapazität von 4,7 µF.

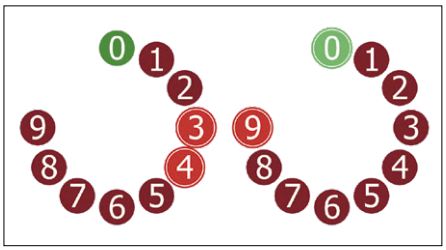


Bild 3. Die Anzeige wechselt zwischen „39“ und „40“. Das kann zunächst verwirren, man könnte auch glauben, sie pendle zwischen „30“ und „49“. Natürlich tritt aber ein solcher Effekt nur zwischen benachbarten Werten (wie 39 und 40) auf.

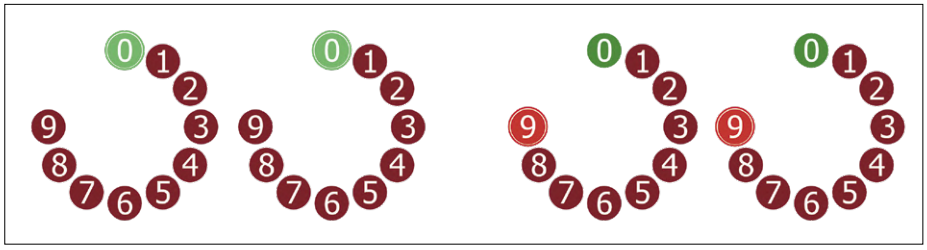


Bild 2. Die Anzeige zeigt „00“ (links) und „99“ (rechts). „99“ kann bedeuten, dass der gemessene Wert exakt das Maximum der Anzeige erreicht oder möglicherweise darüber liegt.

ten Messbereich erreichen, wenn alle vier Bereiche genau abgestimmt sind.

Wie eine Dekatron-Zählröhre

Zwei Kreise mit je zehn LEDs bilden die zweistellige Anzeige des Capaci-Messgeräts. Diese Kreise erinnern an Dekatrons, Zählröhren, die viel früher als Halbleiterchips, LEDs und sogar Transistoren für Zähler/Displays verwendet wurden. Im Gegensatz zu den originalen Dekatrons sind die LEDs hier angeordnet wie Stunden auf einer Uhr, so dass die Anzeige intuitiv abgelesen werden kann (**Bild 1** und **Bild 2**).

Jitter

Manchmal kann, wie in **Bild 3** zu sehen, die Anzeige zwischen zwei Werten schwanken, wenn die Kapazität an der Grenze zwischen zwei Messauflösungen liegt oder wenn elektrisches Rauschen die Messtätigkeit stört. In den meisten Fällen ist es leicht zu erkennen, welcher der beiden Werte der „richtige“ ist, aber das Ganze kann schwieriger zu interpretieren sein, wenn die Zahlen zwischen 9 und 0 wechseln und auch die Zehnerstelle hin und her springt.

Das Messprinzip

Das Capaci-Meter misst fortwährend die Zeit, die der Capacitor-under-Test (Cx) benötigt, um sich über einen bekannten Widerstand zu einer bestimmten Spannung aufzuladen. Die Schaltung besteht aus drei Hauptteilen:

- Cx-Takt (wobei Cx der zu testende Kondensator ist): ein simpler 555-basierter astabiler Multivibrator (U3A, die Hälfte eines 556-Doppeltimers). Die Dauer der Impulse ist proportional zur Kapazität Cx.
- Haupttaktgeber: ein Rechtecksignal, das vom Zähler zur Zeitmessung verwendet wird. Durch die Änderung der Frequenz des Rechtecks lässt sich der effektive Messbereich des gesamten Messgeräts ändern.
- Zähler/Anzeige: Zählt die steigenden Flanken des Haupttakts und misst so die Zeit. Die Zähler treiben auch die LEDs der Anzeige.

Der Haupttakt erzeugt Impulse für die Zähler/Display-Abteilung. Ist das Cx-Signal high, zählt der Zähler die Impulse des Haupttakts, ist es low, so ist der Haupttakt deaktiviert (**Bild 4**) und der Zäh-

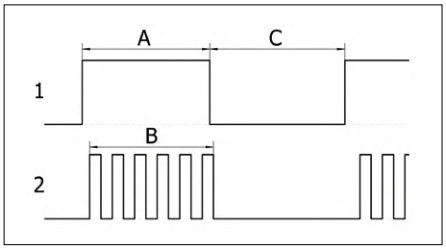


Bild 4. Die Dauer von (A) hängt vom Wert von Cx ab. Der Zähler zählt die Impulse des Haupttakts (2) nur, wenn das Cx-Signal (1) high ist (B). Wenn es low ist, wird der Messwert angezeigt (C).

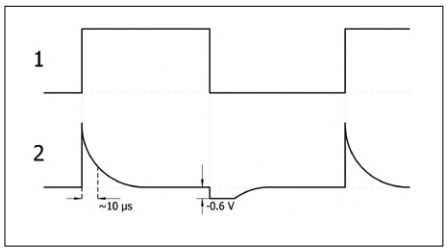


Bild 5. (1) ist der Cx-Takt, (2) ist das Rückstellsignal für die Zähler. Der Differenzierer R12/C8 erzeugt bei einer steigenden Flanke an seinem Eingang einen positiven Impuls und einen negativen Impuls, wenn er eine fallende Flanke wahrnimmt. D4 klemmt den negativen Impuls auf etwa -0,6 V, um eine Beschädigung der Zähler-ICs zu vermeiden. Die Werte von R12 und C8 bestimmen die Ausschwingrate (decay) des differenzierten Signals. Hier haben wir Werte gewählt, die in 10 µs ein Ausschwingen von etwa 66% ergeben, einfach bestimmt durch T = R × C.

ler stoppt. Gleichzeitig werden die LEDs eingeschaltet, so dass der gezählte Wert angezeigt wird.

Start der Zählung bei null

Damit der Zähler bei jedem Zählbeginn bei Null beginnt, muss er bei der steigenden Flanke des Cx-Taktsignals zurückgesetzt werden. Dies wird von einem einfachen Differenzierglied (R12/C8) erledigt, das sehr schmale Impulse bei jeder Flanke des Rechtecksignals liefert (**Bild 5**). Das Differenzierglied erzeugt auch negative Spannungen, die die Zähler nicht verarbeiten können. Diode D4 entfernt diese negativen Impulse, bevor sie den Zähler-ICs schaden können. Die vollständige Schaltung des Messgeräts ist in **Bild 6** zu sehen.

Überlauf verhindern

Obwohl nicht unbedingt notwendig, ist der Zähler auf 99 begrenzt, um Mehrdeutigkeiten zu vermeiden, wenn die

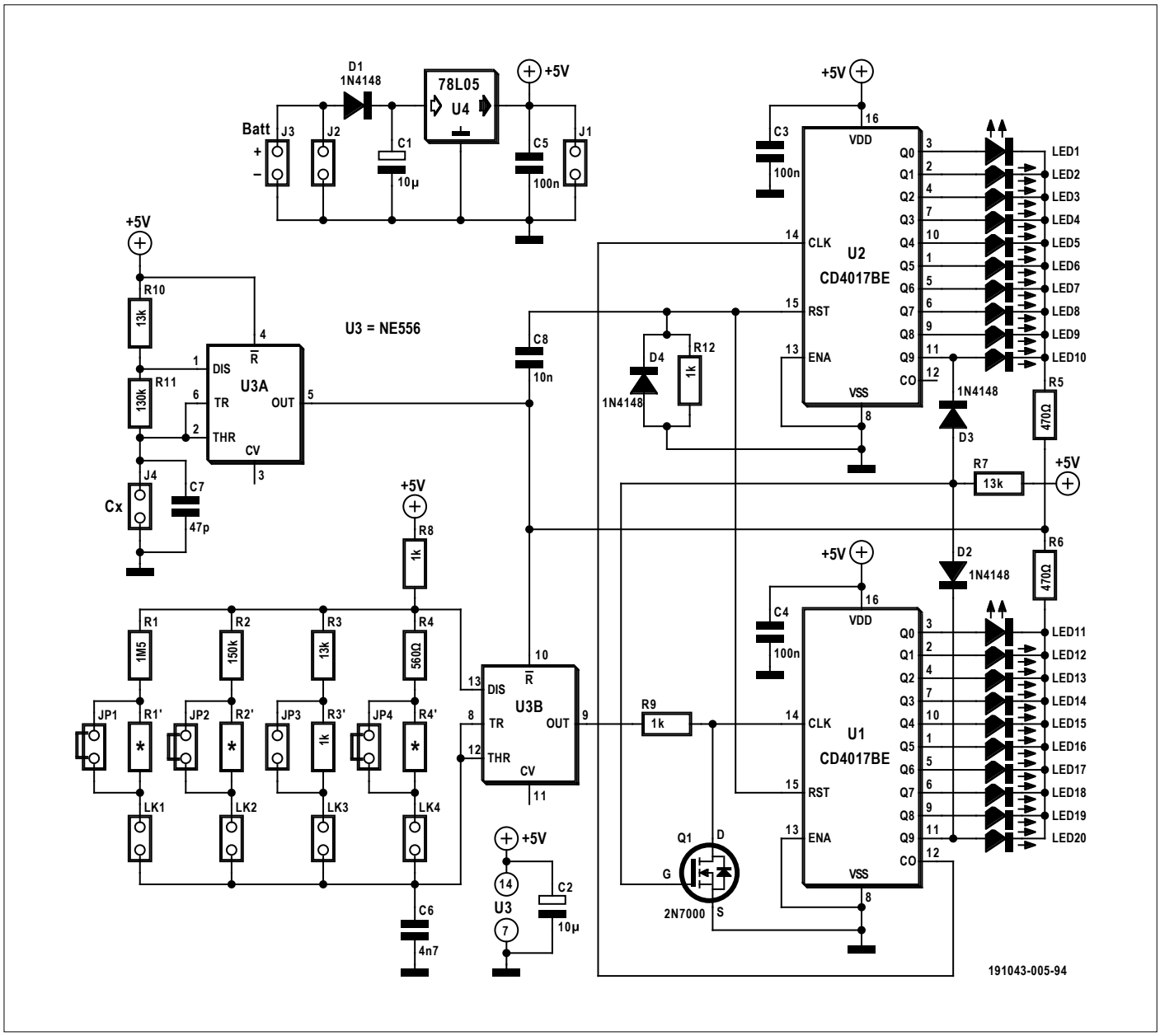


Bild 6. Die vollständige Schaltung des Capaci-Messgeräts.

Anzeige 99 überschreitet und wieder bei 00 beginnt. Deshalb wird die Zählung beendet, wenn die Einser UND die Zehner „9“ sind. Logischerweise benötigen wir dafür ein UND-Gatter, das hier, da nur ein einziges Gatter benötigt wird, aus zwei Dioden (D2, D3 mit R7 als Pull-up) und MOSFET Q1 besteht. Wenn die „9er-Ausgänge“ beider Dezimalzähler high sind, wird der Haupttakt von Q1 unterbrochen, um ein Weiterzählen zu verhindern.

Haupttakt-Frequenzen

Wir möchten, dass der Zähler für eine Kapazität bis 99 zählt, die den aktuellen Bereich vollständig abdeckt. Zum Beispiel: Wenn wir einen Kondensator von 9,9 µF im höchsten Bereich testen, wollen wir, dass der Zähler genau 99 erreicht. In unserer Schaltung wird die Dauer der High-Phase des Cx-Takts durch die Werte von R10, R11 und Cx bestimmt:

$$T_{HIGH} = 0{,}693 \times (R10 + R11) \times Cx \; [s]$$

Mit den angegebenen Werten für R10 und R11 und 9,9 µF für Cx ergibt sich eine Dauer der High-Phase von 0,981 s. Während dieser Zeit wollen wir 99 Impulse des Haupttakts zählen. Die Frequenz des Haupttakts sollte also:

$$f_{CLOCK} = 99 / 0{,}981 = 100{,}9 \text{ Hz}$$

sein. Das ist die Frequenz, die für den Bereich 9,9 µF benötigt wird. Wenn wir den Kapazitätsbereich um den Faktor 10 reduzieren (Vollausschlag 0,99 µF) und wir immer noch bis 99 zählen wollen, dann muss die Haupttakt-Frequenz natürlich um den Faktor 10 höher sein. Gleiches gilt für die noch niedrigeren Bereiche.

Diese Frequenzen für jeden Bereich sind theoretisch; in der Praxis können sie aufgrund von Bauteiltoleranzen leicht abweichen. Um den Fehler zu verringern, können die Messbereichswiderstände durch Hinzuschalten weiterer Widerstände (R1-R1‘, R2-R2‘, R3-R3‘ und R4-R4‘) fein justiert werden. Doch selbst ohne Anpassungen liegt der Fehler wahrscheinlich innerhalb von ±5% Vollausschlag, möglicherweise ist er sogar kleiner.

Messen Sie an Pin 9 von U3 die Haupttakt-Frequenz bei kurzgeschlossenen Kondensator-Messleitungen. Beachten Sie, dass die Pads für die Zusatzwiderstände R1‘, R2‘ und R4‘ (R3‘ nicht!) auf der Platine kurzgeschlossen sind. Wollen Sie einen Zusatzwiderstand bestücken, müssen Sie deshalb die entsprechende Leiterbahn auftrennen.

Abweichung vom theoretischen Wert

Um alle vier Messbereiche abzudecken, muss der Haupttakt Frequenzen von etwa 100 Hz bis 100 kHz erzeugen. Laut Datenblatt wird die Frequenz des 555 wie folgt berechnet:

$$f = 1{,}44 / (C6 \times (R8 + 2 \times Rx)) \; [Hz]$$

Hier ist Rx entweder R1 + R1‘, R2 + R2‘, R3 + R3‘ oder R4 + R4‘. Bei höheren Frequenzen weicht die tatsächliche Ausgangsfrequenz leider vom theoretischen Wert ab. Diese Abweichung wurde bei der Auswahl der bereichsbestimmenden Widerstände zwar schon berücksichtigt, eine weitere Feinabstimmung kann jedoch nicht schaden.

Montage des Kapazitäts-Messgeräts

Wie üblich beginnen wir mit der Montage der kleinen Bauteile wie Dioden und Widerstände und arbeiten uns dann zu den größeren Bauteilen vor. Beachten Sie, dass R1‘, R2‘ und R4‘ (zunächst) nicht montiert werden sollten, R3‘ hingegen schon. Achten Sie darauf, dass alle gepolten Bauteile (in diesem Projekt alles außer den Widerständen, den Keramikkondensatoren und den Stiftleisten) korrekt ausgerichtet sind. Die Verwendung von IC-Fassungen für U1, U2 und U3 wird dringend empfohlen. Führen Sie die Batteriekabel durch zwei der Platinenbohrungen, bevor Sie sie an die Pads löten.

Achtung Test

Überprüfen Sie zunächst noch einmal, ob die gesamte Platine korrekt bestückt wurde und ob es keine Kurzschlüsse gibt. Überprüfen Sie auch, ob die ICs richtig herum eingesetzt sind und alle Beinchen in ihren Schuhen stecken.

Stecken Sie einen Jumper auf einen der Header zur Bereichsauswahl. Befestigen Sie eine 9-V-Batterie an der Batterieklemme. Nun sollten die grünen LEDs leuchten, eventuell eine rote LED im rechten LED-Kreis. Nehmen Sie einen Kondensator mit einem bekannten Wert, der sicher in einem der vier Messbereiche liegt, zum Beispiel 47 nF für den 99-nF-Bereich. Bringen Sie den Testkondensator an den beiden Testpads an oder verwenden Sie, wenn Sie möchten, Messleitungen (die Bohrungen in den Testpads sind für 2-mm- und 4-mm-Bananenstecker geeignet). Der Wert des Kondensators sollte auf der Anzeige erscheinen.

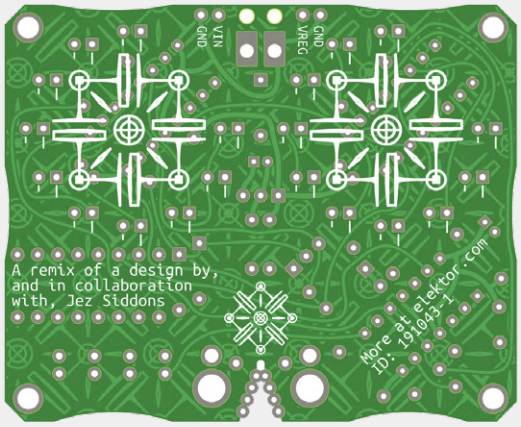
STÜCKLISTE

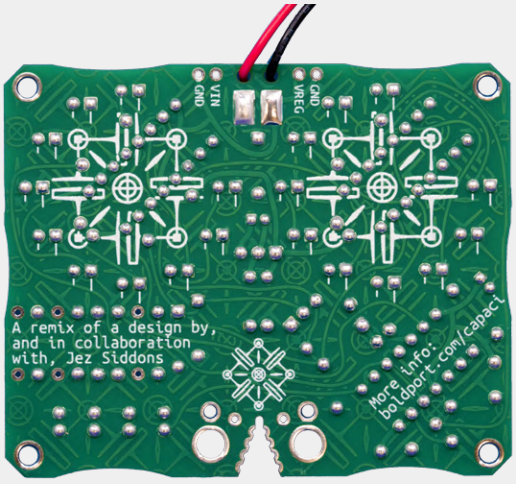
Widerstände
(alle 1 %, 0,25 W)
R1 = 1M5
R2 = 150 k
R3,R7,R10 = 13 k
R3',R8,R9,R9,R12 = 1 k
R4 = 560 Ω
R5,R6 = 470 Ω
R11 = 130 k
R1',R2',R4'= optional zur Feinkalibrierung

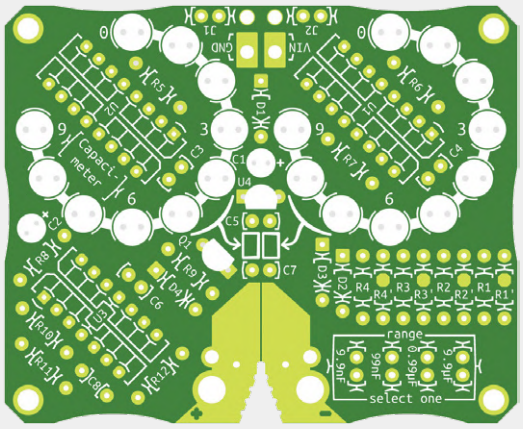
Kondensatoren
C1,C2 = 10 µ, 10 V
C3,C4,C5 = 100 n
C6 = 4n7, 1% oder 5%.
C7 = 47 p
C8 = 10 n

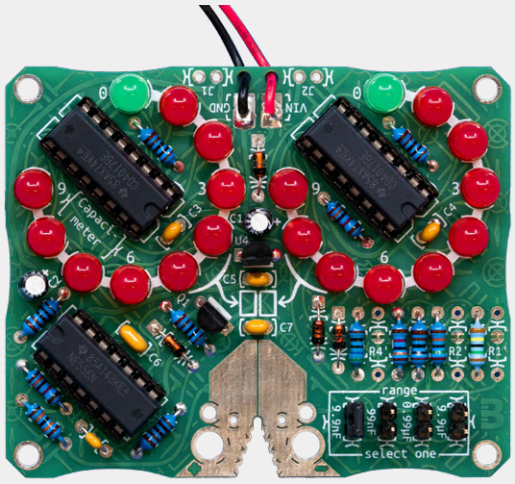
Halbleiter
D1,D2,D3,D3,D4 = 1N4148
Q1 = 2N7000 (BS170)
U1,U2 = CD4017BE
U3 = NE556
U4 = 78L05
LED1,LED11 = LED, grün, 5 mm, low current oder high brightness
LED2...LED10, LED12... LED20 = LED, rot, 5 mm, low current oder high brightness

Außerdem
J1,J2,LK1,LK2,LK3,LK4 = 2-polige Stiftleiste, Raster 2,54 mm
J3 = Clip für 9-V-Block
J4 = Messkabel zum Testkondensator
JP1 = Jumper zur Bereichswahl
2 St. 2x8-polige DIL-Fassung für U1, U2
1 St. 2x7-polige DIL-Fassung für U3
Platine 191043-1









Wenn dies nicht der Fall ist, überprüfen Sie den ausgewählten Bereich und eventuell begutachten Sie nochmals Ihre Lötarbeiten.
Eine letzte Anmerkung, der Vollständigkeit halber: Das Capaci-Messgerät kann entweder mit einer 9-V-Blockbatterie an J3 oder einer externen 7...15-VDC-Spannungsquelle an J2 betrieben werden. Schließen Sie jedoch nicht beide gleichzeitig an! J1 ist nur ein Testpad, schließen Sie hier keine Stromversorgung an!

Abschließend

Ich möchte Stephen Bernhoeft für sein kritisches Auge, seinen Rat und seine Klarheit bei der Verfeinerung und Prüfung dieser Schaltung danken. Vielen Dank auch an Saar Drimer von Boldport für den Entwurf der Platine. Ich hoffe, dass Ihnen der Bau dieses Geräts genauso viel Spaß macht wie mir und dass es seine Messaufgaben brav erfüllt! ⏮

191043-03

GEWUSST WIE:

Entprellen eines mechanischen Kontakts oder Schalters

Ein Schalter ist entweder offen oder geschlossen, oder nicht?

Von **Clemens Valens**

Viele Ingenieure behandeln mechanische Schalter oder Kontakte als einfache Bauteile mit zwei Zuständen, entweder offen oder geschlossen. Obwohl dies meist ausreicht, gibt es Situationen, in denen diese Meinung zu kurz gedacht ist und Ärger verursacht. Kontaktwiderstand ist eine Problemquelle, Kontaktprellen eine andere. In diesem Teil unserer neuen Rubrik „Gewusst wie“ konzentrieren wir uns auf den zweiten Fall.

Eine Frage der Zeit

Wie alles in unserem Universum brauchen Zustandsänderungen Zeit. Schalter, mechanisch oder nicht, sind da keine Ausnahme. Die meisten mechanischen Schalter sind eher langsam und ein Zustandswechsel kann Dutzende von Millisekunden dauern, bis sich die Lage stabilisiert hat. Während des Schließens springt der Kontakt ein paar Mal, wie ein Ball auf dem Boden hüpft. Im Gegensatz zu einem Ball tritt dieses Schalterprellen aber auch auf, wenn ein Kontakt geöffnet wird. Elektronische Systeme, die viel schneller sind als mechanische Schalter, können Kontaktprellen wahrnehmen und dadurch gestört werden. Das System interpretiert dies möglicherweise als mehrmaliges Drücken und durchläuft Menüs, ohne dass der Benutzer eine gewünschte Option auswählen kann.

Entprellen tut Not!

Entprellen ist eine Methode, ein System gegen Kontaktprellen abzusichern. Es gibt mehrere Methoden, von analogen RC-Filtern über spezielle ICs bis hin zu Softwarealgorithmen. Sie alle haben das Ziel, dem System einen definierten, störungsfreien Kontaktzustand zu vermitteln. Dabei ist es völlig unerheblich, ob es sich um eine manuell bediente Taste (oder Schalter) oder einen maschinengesteuerten Kontakt (Mikroschalter, Drehgeber, Relais) handelt.

RC-Netzwerk zum Entprellen

Die Idee ist, an den Schalter eine Schaltung anzuschließen, die noch langsamer ist als der Schalter selbst, so dass sie das Prellen an ihrem Eingang überhaupt nicht bemerkt. Am Ausgang der Schaltung ist der Zustand stabil, high oder low, mit langsamen Übergängen zwischen diesen beiden Zuständen (**Bild 1**). Das klingt nach einer schönen und einfachen Lösung, aber es gibt trotzdem Probleme:

- Wenn die Hysterese des Eingangs nicht ausreichend ist und gleichzeitig das Signal verrauscht ist, kann ein sich langsam änderndes Signal, das ein schnelles System beeinflusst, unerwünschte Effekte erzeugen, die dem Kontaktprellen ähnlich sind. Dies könnte auch zu einem übermäßigen Stromverbrauch führen. Schmitt-Trigger am Systemeingang sind daher sehr zu empfehlen.
- Wenn ein Filter zu langsam ist, können kurze, aber gültige Zustandsänderungen übersehen werden.
- Man benötigt zusätzliche Komponenten, die Platz auf der Platine einnehmen und Geld kosten.
- In einem gewöhnlichen Schalter-nach-Masse-mit-Pull-up-Widerstand-und-RC-Netzwerk führt das Drücken und Loslassen der Taste nicht zu identischen Ergebnissen. Das Hinzufügen einer Diode wie in Bild 1 kann dieses Problem zwar lösen, würde aber die Stückliste weiter verlängern.

Aufwändigere Entprell-Schaltungen

Auch mit einem Set-Reset-Latch (SR) oder einem D-Flip-Flop kann Kontaktprellen vermieden werden. Dazu benötigt man jedoch einen 1xum-Schalter (SPDT), der teurer ist als eine einfache 1xan-Taste (SPST). Diese Methode kann auch ganz einfach in eine Software implementiert werden, wenn Sie bereit sind, dafür zwei I/O-Pins pro SPDT-Schalter zu opfern.

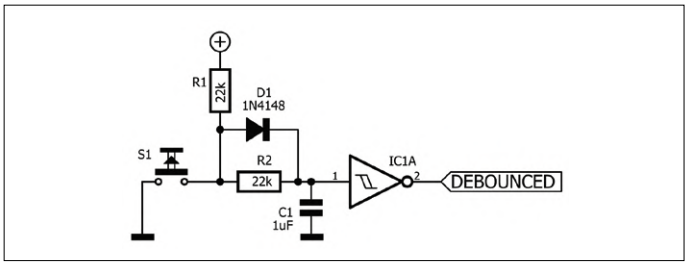


Bild 1. Entprellen eines mechanischen Kontakts mit einem RC-Netzwerk. Obwohl die angegebenen Werte von R1, R2 und C1 in vielen Fällen gut passen würden, hängen sie letztendlich doch von ihrer Anwendung ab. D1 sorgt dafür, dass Öffnen und Schließen des Kontakts zu ähnlichen Verzögerungen führt.

Ein monostabiler Multivibrator oder ein Timer kann eine Lösung sein, erzeugt aber einen Impuls statt eines stetigen Pegels. Es gibt spezielle Entprell-ICs wie den Klassiker MC14490 (**Bild 2**), aber auch andere ICs wie den MAX6816 (mit mehreren Kanälen den MAX6817 und den MAX6818) oder den LTC6994.

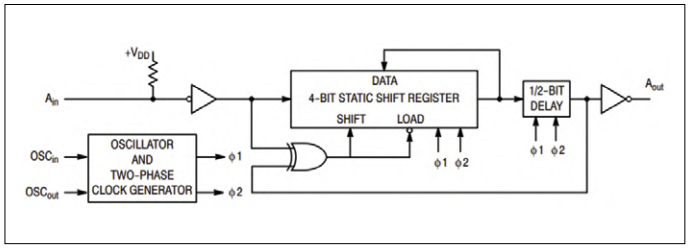


Bild 2. Innenansicht des MC14490 mit sechs Kanälen zur Kontakt-Entprellung.

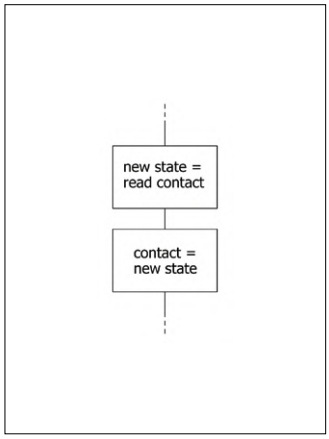


Bild 3. Wenn ein Kontaktzustand nur ab und zu gelesen werden muss, kommt man ohne ausgefeilte Entprell-Techniken aus.

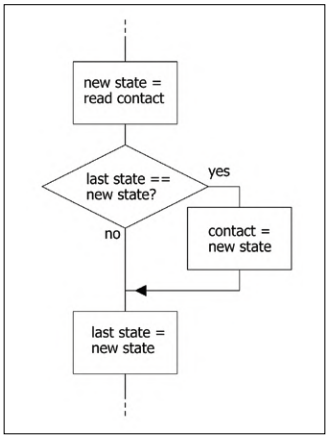


Bild 4. Wenn das Polling nicht zu schnell ist (etwa <0,1 Hz), können zwei aufeinander folgende identische Kontaktzustände ausreichen, um den neuen Zustand zu akzeptieren.

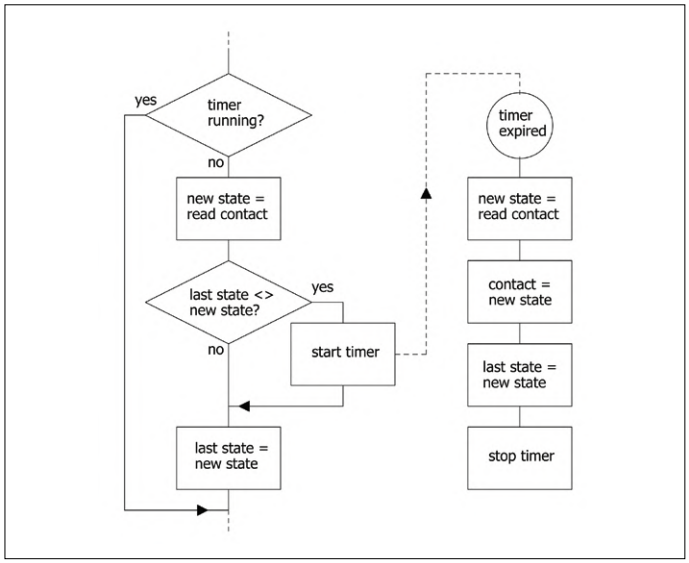


Bild 5. Ein Wechsel des Kontaktzustands startet einen Timer. Die Hauptprogrammschleife ignoriert den Kontakt, während der Timer läuft. Der Zustand des Kontakts wird erst wieder gelesen, wenn der Timer abgelaufen ist, also wenn das Kontaktprellen beendet sein sollte.

Die Firma LogiSwitch scheint sich auf Entprell-ICs spezialisiert zu haben, aber auch Maxim bietet sogenannte „Contact Monitors“ zur Entprellung an. Der Vorteil der Entprellung mit ICs ist, dass mehr als ein Kanal in einem System/Gerät entprellt und gleichzeitig Funktionen wie Überspannungs- und ESD-Schutz hinzugefügt werden können. Hauptnachteile sind natürlich zusätzliche Kosten und Platz auf der Platine.

Entprellen in der Software

Der Grund, mit Hardware zu entprellen, liegt in der Regel darin, dass es aus irgendeinem Grund nicht von der Software erledigt werden kann. Doch in vielen mikrocontroller-basierten Anwendungen hat die MCU ausreichend Ressourcen, um Kontakte zu entprellen. Aber wie? Durch Polling oder durch Interrupts! Beim Polling überprüft der Controller regelmäßig den Kontakteingang. Das kann periodisch durch einen Timer erfolgen oder immer, wenn die MCU gerade Zeit dafür hat. Polling kann deshalb Zustandsänderungen des Kontakts übersehen, wenn sie nur kurz sind oder zu spät erkannt werden (hohe Latenz). Allerdings vermeidet Polling Probleme mit zeitkritischen Prozessen. Interrupts dagegen können Zustandsänderungen eines Kontakts fast so schnell erkennen, wie Sie auftreten (niedrige Latenz). Dies führt zwar zu einem schnell reagierenden System, aber wenn die MCU schnell genug ist, kann Prellen mehrere Interrupts auslösen, was wiederum zeitkritische Prozesse behindern und/oder falsche positive Ergebnisse erzeugen kann. In solchen Fällen kann es helfen, die Pin-Interrupts während der Prellphase zu deaktivieren.

Sag‘ niemals nie!

Einige Leute sagen, dass man niemals einen Schalter an einen Interrupt-Pin anschließen sollte, da das Prellen die interne Logik des Interrupt-Eingangs stören kann (oder aus einem anderen Grund). Das ist natürlich Unsinn, da alles von der Anwendung abhängt. Wenn Kontaktprellen ein System stören kann, dann können EMI und anderes Rauschen das auch. Zu solchen Systemen muss ohnehin eine Rauschfilterung hinzugefügt werden. Die meisten, wenn nicht alle modernen Mikrocontroller besitzen Glitch-Filter an ihren Eingängen. Wenn das nicht genug ist, spendieren Sie ein zusätzliches Rauschfilter zwischen Kontakt und Eingangspin!

Einige Entprell-Algorithmen

Verzögerungsschleife

Sobald ein Kontaktübergang erkannt wird, akzeptieren Sie den neuen Zustand (**Bild 3**). Das ist eine einfache Methode für langsam arbeitende Polling-Systeme (25 Hz oder weniger), allerdings kann sie einen Glitch für einen Übergang halten, wenn der Glitch genau im Moment des Erfassens des Schalterzustands erfolgt. **Bild 4** zeigt, wie diese Methode auf einfache Weise verbessert werden kann, indem der Schalterzustand zweimal nacheinander (mit gleichem Ergebnis natürlich) ermittelt werden muss, bevor der neue Zustand akzeptiert wird.

Timer

Starten Sie nach dem Erkennen des Übergangs einen Timer, um den Eingang etwa 20 Millisekunden später erneut zu überprüfen. Verwenden Sie als Kontaktzustand den Wert am Eingang, wenn der Timer abläuft (**Bild 5**). Das Ganze funktioniert sowohl beim Polling als auch bei Interrupts, erfordert aber einen Timer.

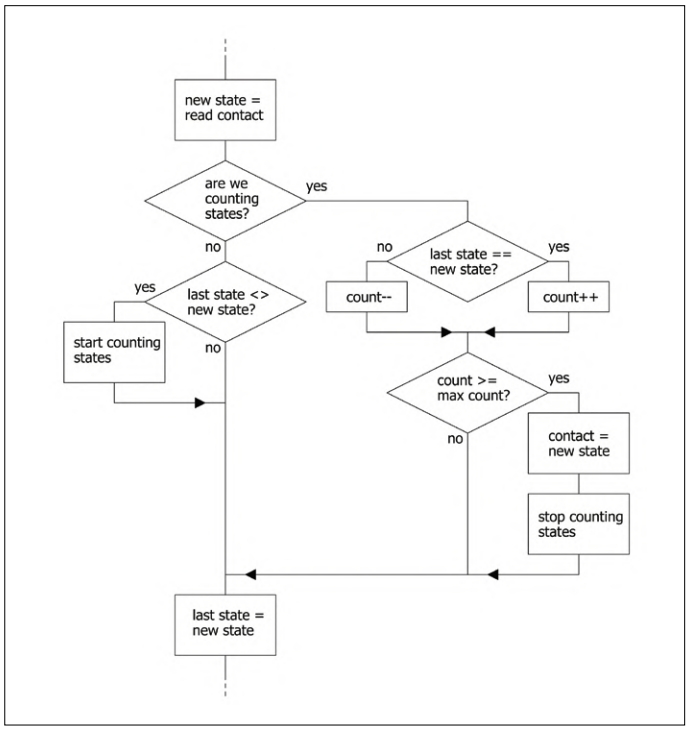


Bild 6. Ein Flussdiagramm, das einem RC-Entprell-Netzwerk entspricht. Es triggert, wenn mindestens „max count“ gültige Samples ermittelt wurden.

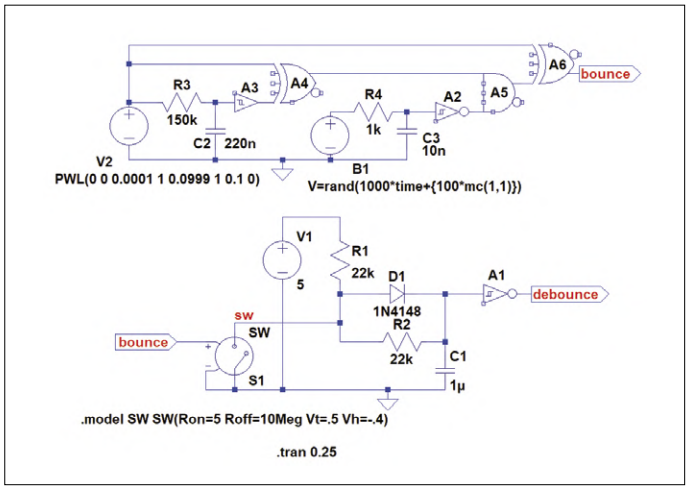


Bild 7. Kontaktprellen und Entprell-Simulationen in LTspice. Spannungsquelle V2 erzeugt den Kontakt-Schließimpuls, Spannungsquelle B1 fügt das Prellen hinzu. Der Wert von B1 bestimmt die Mindestdauer des Prellens (hier 1 ms). Ändern Sie die Werte von R3/C2 und/oder R4/C3, um das Prell-Verhalten zu ändern.

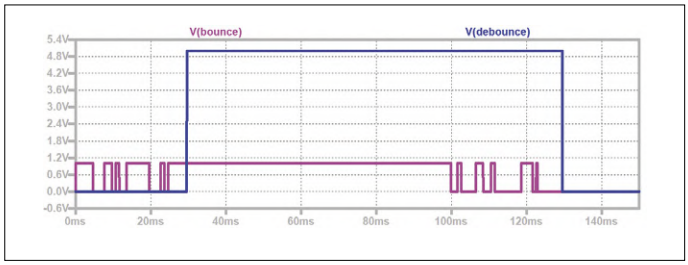


Bild 8. Simulationsergebnisse zeigen Kontaktprellen (rosa) und das entprellte Ergebnis (blau).

Wenn Sie den Pin-Interrupt während des Betriebs des Timers deaktivieren, vermeiden Sie unnötige Probleme durch Prellen.

X aus Y

Einen Schritt weiter geht die Methode, zwei oder mehr identische Samples (aus einer größeren Anzahl von Samples) vor der Annahme einer Zustandsänderung zu verlangen, zum Beispiel drei von vier oder sieben von zehn. Dies ist das digitale Äquivalent zum integrierenden RC-Entprell-Netzwerk (**Bild 6**). Anstatt x gültige von y Erfassungen zu verlangen, können Sie auch z aufeinanderfolgende gültige Samples fordern. Die Reaktionsfähigkeit hängt von der Anzahl der Samples und der Samplefrequenz ab.

Simulation

Bild 7 und **Bild 8** zeigen eine LTspice-Simulation der RC-Entprell-Schaltung, mit der wir diesen Artikel begonnen haben. Damit der Simulator für jedem Durchlauf unterschiedliche Voraussetzungen vorfindet, sollten Sie in LTSpice die Option „Use the clock to reseed the MC generator[*]“ auf dem Hacks!-Tab unter „Control Panel“ (die Taste mit dem kleinen Hammer) wählen (**Bild 9**). Der Abschnitt „100*mc(1,1) “ im Wert für B1 macht dies möglich. Spielen Sie mit verschiedenen R/C-Werten, um die Entprell-Eigenschaften zu ändern.

Und was soll ich jetzt tun?

Wie bei jedem Thema in der Elektronik, in der Technik und überhaupt gibt es noch viel mehr, was man darüber sagen könnte. Letztendlich ist es unmöglich zu bestimmen, welche Entprell-Methode für Sie am besten ist. Lernen und verstehen Sie Ihr System zuerst, bevor Sie sich entscheiden, welche Technik Sie anwenden! ⏮

191015-03

Weblink

[1] Dateien für die LTspice-Simulation: www.elektormagazine.de/191015-03

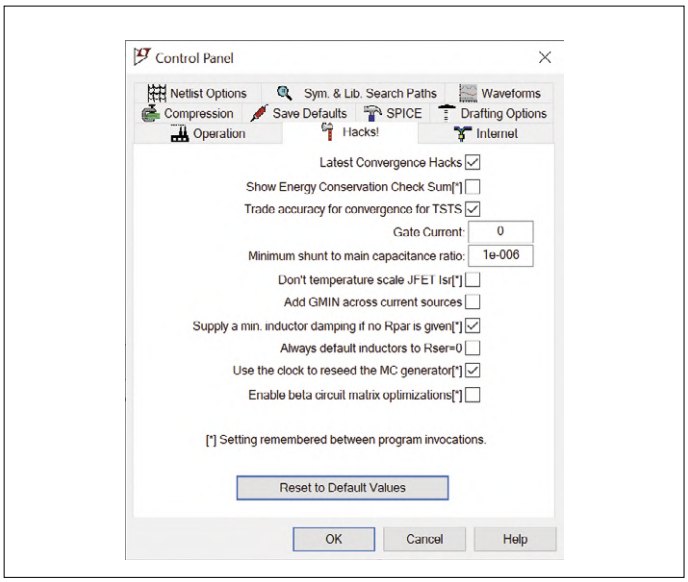


Bild 9. Durch Aktivieren von „Use the clock to reseed the MC generator[*]“ erzeugt LTspice für jeden Durchlauf unterschiedliche Ergebnisse.

Von Entwicklern für Entwickler

Tipps & Tricks, Best Practice und andere nützliche Infos

Von **Clemens Valens**

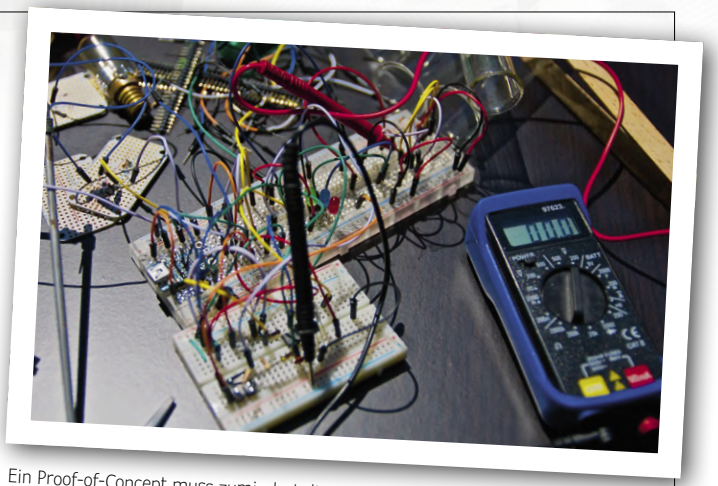
VON DER IDEE ZUM PRODUKT

TEIL 3

In den vorangegangenen Teilen dieser Serie [1][2] haben wir einige Dinge besprochen, die Sie in Betracht ziehen sollten, bevor Sie überhaupt mit der Entwicklung eines Produkts beginnen. Wir gehen in diesem Artikel davon aus, dass Sie diese Probleme bereits behandelt und geeignete Lösungen gefunden haben. Beginnen wir mit dem Design!

ERSTELLUNG EINES PROOF OF CONCEPT

Wenn Sie eine Spezifikation erstellt haben, müssen Sie als Erstes einen Proof-of-Concept (PoC) für Ihr Produkt erstellen. Das ist eine Art Prototyp, der zeigt, ob das, was Ihnen vorschwebt, überhaupt möglich ist. Nicht jede Funktion oder Option aus der Spezifikation muss dabei implementiert werden, solange die Hauptfunktionen wie geplant arbeiten. Ziel ist es dabei, alle technischen oder physischen Unmöglichkeiten aufzudecken, die Sie bisher übersehen haben. Wenn Sie Ihren PoC nicht zum Laufen bekommen, sollten Sie entweder schon jetzt aufgeben oder Ihr Projekt nochmals von vorne beginnen. Es macht wenig Sinn, die Spezifikationen, die Sie nicht erfüllen können, einfach zu vernachlässigen, ohne die Marktforschung zu wiederholen, da das Endprodukt nicht mehr den Wünschen des potenziellen Nutzers entsprechen würde.



Ein Proof-of-Concept muss zumindest die Hauptfunktionen Ihres Produkts implementieren.

Simulieren was das Zeug hält!

SIMULIEREN WAS DAS ZEUG HÄLT

Viele Designer neigen dazu, sich von Schaltungssimulationen fernzuhalten. Dafür gibt es viele Gründe, aber vielleicht sind diese mittlerweile schon überholt? Ein guter Simulator kann Zeit und Geld sparen; und einige davon sind sogar kostenlos. Sie sollten natürlich wissen, wie man die Programme bedient und wie man die Ergebnisse interpretiert, aber so eine Software ist mit Messwerkzeugen ausgestattet, die Sie sich nie leisten werden. Netzteile haben dort eine schier unbegrenzte Leistung. Die Visualisierung jeder möglichen Wellenform, Spannung oder Strom ist nur einen Mausklick entfernt; Sonden fallen nie von den Pads herunter und verursachen Kurzschlüsse. Tausende von Ampere über einen Transistor zu leiten zerstört das Gerät nicht. Und selbst wenn doch, ersetzen Sie es, ganz ohne Lötkolben.

SIMULATIONEN FUNKTIONIEREN NIEMALS IN ECHT

Wenn „eine“ Realität nicht mit ihrer Simulation übereinstimmt, ist diese Realität vielleicht nicht gut genug? Ein Steckbrett mit schlechten Kontakten und langen Kabeln stellt keine saubere, rauschfreie Basis für eine Simulation dar. Bei der Simulation lässt sich ebenfalls noch einiges drehen: Verwenden Sie zum Beispiel SPICE-Modelle von Bauteilherstellern anstelle generischer Modelle. Vergessen Sie nicht, die von Ihnen verwendeten Modelle zu testen. Fügen Sie Dinge wie Toleranzen, Rauschquellen und interne Impedanzen hinzu.

BLEIBEN SIE FOKUSSIERT

Bei guten Kenntnissen im Umgang mit einem Simulator ist es manchmal schwierig, sich nicht mitreißen zu lassen. „Was wäre wenn”-Fragen können dazu führen, dass man Stunden damit verbringt, obskure Schaltungsoptionen und alle möglichen hinzugefügten Komponenten zu untersuchen. Bleiben Sie fokussiert und treten Sie gelegentlich einen Schritt zurück. Stellen Sie sicher, dass Ihre Simulation valide bleibt.

BLEIBEN SIE BEI COTS

Der PoC sollte möglichst aus handelsüblichen (COTS = Commercial off the shelf) Modulen, Bauteilen und Platinen aufgebaut werden. Vermeiden Sie es, zu viel Material von demontierten Geräten zu verwenden, die Sie zufällig herumliegen haben, da solche Teile später Beschaffungs- oder Obsoleszenzprobleme hervorrufen können.

Die Verwendung von handelsüblichen COTS-Produkten erleichtert das Leben eines Designers erheblich.

HÜTEN SIE SICH VOR „ETWAS ÄHNLICHEM"

Wenn das Endprodukt einen Mikrocontroller in einem mikroskopischen Gehäuse benötigt, mit dem Sie nicht umgehen können, ist es ok, ihn im PoC durch ein vom Hersteller entwickeltes großes Entwicklungsboard zu ersetzen. Seien Sie jedoch sehr vorsichtig, wenn ein solches Board nicht existiert, und Sie es durch „etwas Ähnliches” ersetzen. Schnell hat man sich beim Design auf die spezifische Funktion eines Teils verlassen, aber wenn sich herausstellt, dass dieses Teil für das endgültige Design ungeeignet ist, können Sie in Schwierigkeiten geraten. Das Gleiche gilt für ein Dev-Board; vermeiden Sie es, solch ein Entwicklungsboard zum Herzstück Ihres PoC zu machen.

SUBTILE PROBLEME SIND DIE SCHWIERIGSTEN

Während der PoC-Entwicklung können viele subtile Probleme auftreten, wie inkompatible Signalpegel, Synchronisations- und Zeitunterschiede, unverständliche Beschreibungen in Datenblättern und Interferenzen. Manchmal verschwinden diese einfach, wenn Sie ein Bauteil ersetzen. Ein Triple-A-, militärspezifischer, weltraumtüchtiger Operationsverstärker mit Terahertz-Bandbreite kann ein Designproblem lösen, aber ist es eine brauchbare Lösung, um Ihr Ziel zu erreichen? Wenn Sie aber andererseits feststellen, dass Sie zu viele Bauteile benötigen, um ein Problem zu lösen, können Sie auch in eine Sackgasse geraten. In beiden Fällen kann es sich lohnen, zuerst einige Funktionsprinzipien zu überarbeiten, bevor man weitermacht. Zögern Sie nicht, zum Simulator zurückzukehren, um die Probleme zu lösen.

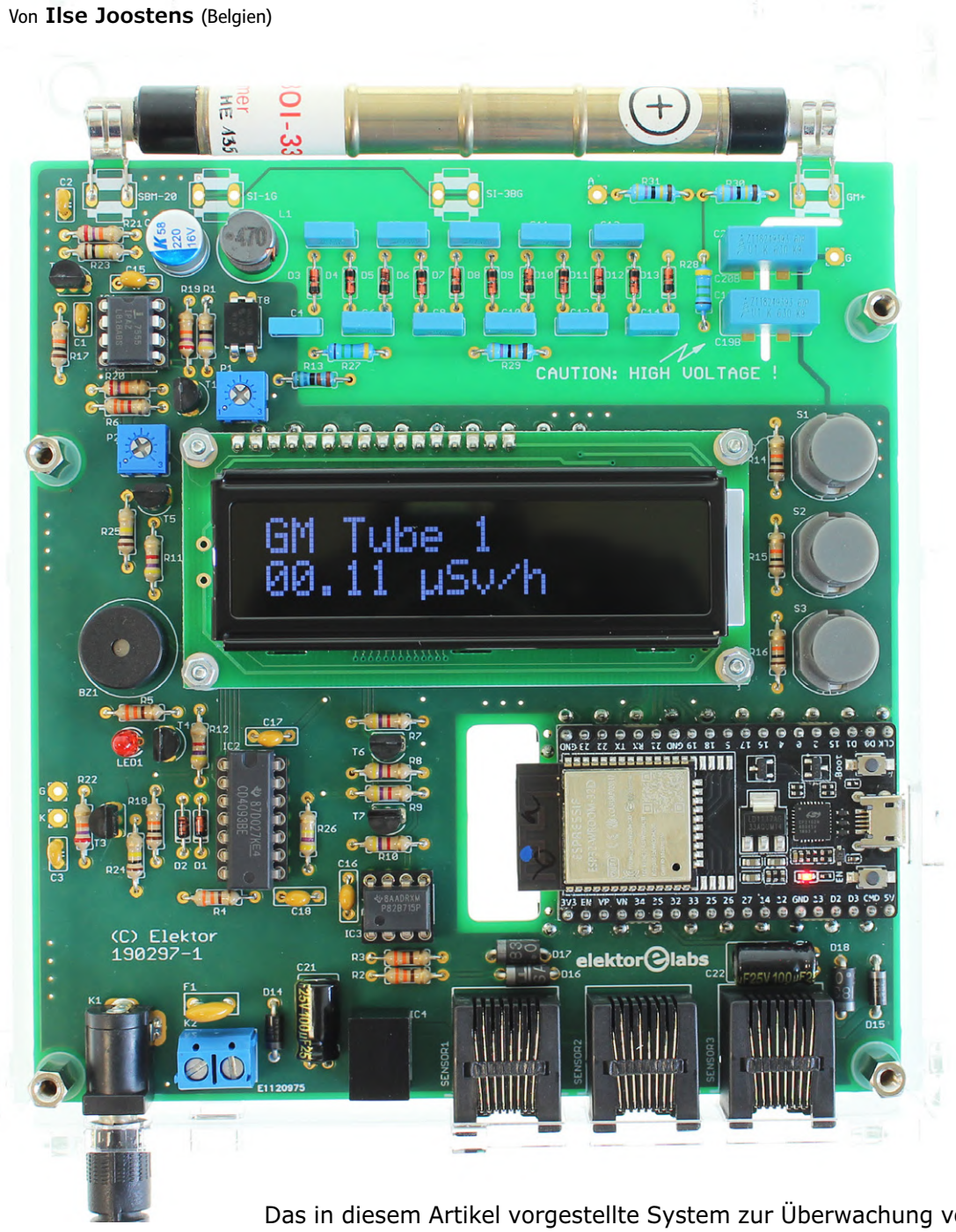
UND WAS KOMMT DANN?

Angenommen, Ihr PoC läuft wie vorgesehen, dann können wir diesen nun in einen Schaltplan umwandeln. Klingt einfach, oder? Aber das ist es nicht!

Erweiterbares System zur Umweltüberwachung

Veröffentlicht Umweltparameter auf IoT-Plattformen

Von **Ilse Joostens** (Belgien)



Das in diesem Artikel vorgestellte System zur Überwachung von Umweltparametern misst die Hintergrundstrahlendosis und veröffentlicht die Messwerte auf IoT-Plattformen wie openSenseMap und ThingSpeak.

INFOS ZUM PROJEKT

Strahlung | Hochspannung | Geiger-Müller-Röhre | Arduino | ESP32 | IoT

Einsteiger
➡ Fortgeschrittene
Experte

etwa 4 Stunden

Lötkolben, Arduino-IDE, Labornetzgerät

circa 70 €

Eigenschaften

- 9…20 V_{DC} Versorgungsspannung
- WLAN-Konnektivität mit ESP32
- Ein oder zwei Geiger-Müller-Röhren
- Einstellbare Röhrenspannung (bis zu 1,1 kV)
- Erfassung von Gammastrahlung (und Beta-Strahlung)
- Anzeige der Daten auf lokalem LCD und Remote-IoT-Plattform
- In hohem Maße konfigurierbar
- Alarm bei hoher Strahlendosis mit einstellbarem Schwellwert
- Drei gepufferte I^2C-Verbinder für Erweiterungen

Hintergrundstrahlung

Die Hintergrundstrahlung, die sowohl aus natürlichen als auch aus künstlichen Quellen stammt, ist die Gesamtmenge der ionisierenden Strahlung an einem bestimmten Ort. Als ionisierende Strahlung wird jede Form von Strahlung bezeichnet, die in der Lage ist, Elektronen von Atomen oder Molekülen zu lösen. Das ist nicht identisch mit Radioaktivität, aber radioaktiver Zerfall führt häufig zu ionisierender Strahlung [1]. Nahezu alle in der Natur vorkommenden Materialien erzeugen diese Strahlung aufgrund natürlicher Radionuklide, von denen uns hauptsächlich Uran-238/235, Thorium-232, Kalium-40 und Rubidium-87 betreffen. In der Umgebungsluft finden wir Radon-222 und dessen Tochter-Nuklide und sowie kosmogene Radionuklide [2]. Schließlich tragen auch radioaktive Materialien in Baustoffen, Konsumgütern und Haushaltsgegenständen (**Bild 1**) zur Hintergrundstrahlung bei [3].

Messung der ionisierenden Strahlung

Bei der Überwachung der Hintergrundstrahlung sind Alpha- und Beta-Strahlung in der Regel weniger interessant, da ihre Reichweite in der Luft sehr begrenzt ist. Deshalb sollte ein Geiger-Müller-Röhrchen mit hoher Gamma-Empfindlichkeit verwendet werden. Für ein Einsteigersystem ist die häufig verwendete russische

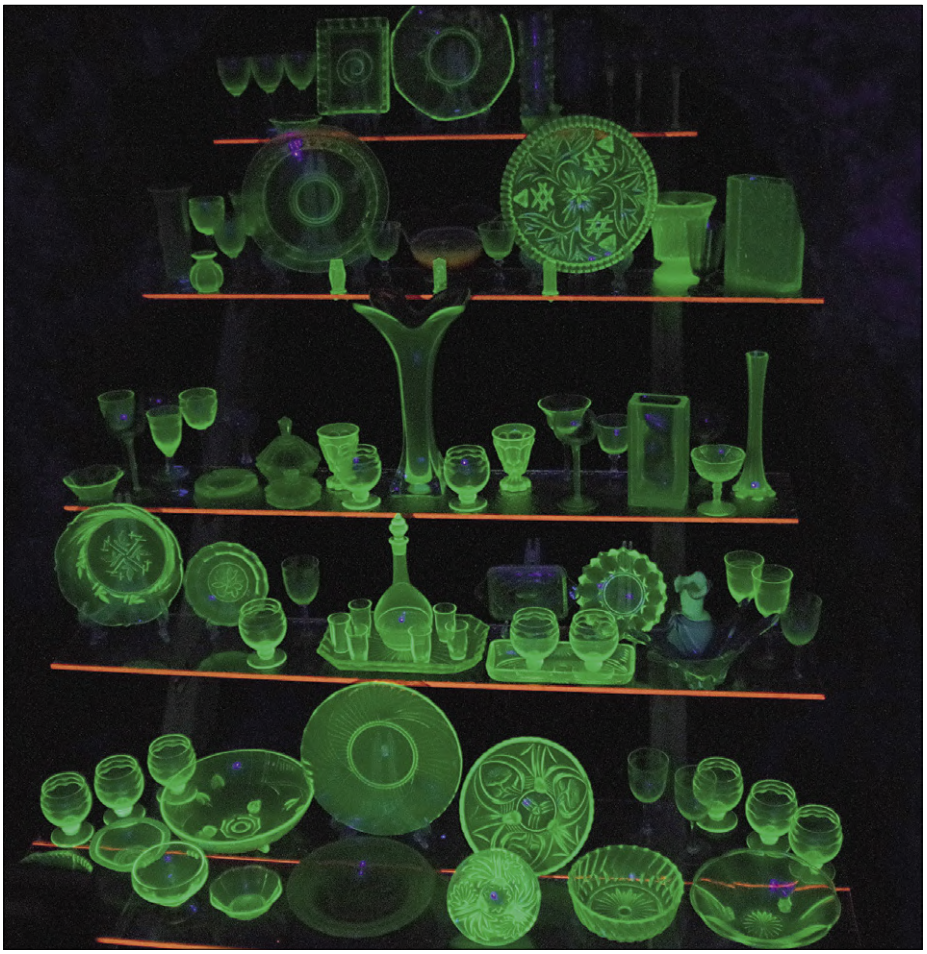


Bild 1. Eine interessante radioaktive Quelle zur Überprüfung des Umweltüberwachungssystems ist Uranglas. Die Perlen dieses Glases, das in der Tschechischen Republik noch immer hergestellt wird, sind billig und sicher zu verwenden, da die Uransalze im Glas gebunden sind.

Bild 2. Zwei Beispiele für gängige Geiger-Müller-Zählröhren mit Metallwänden.

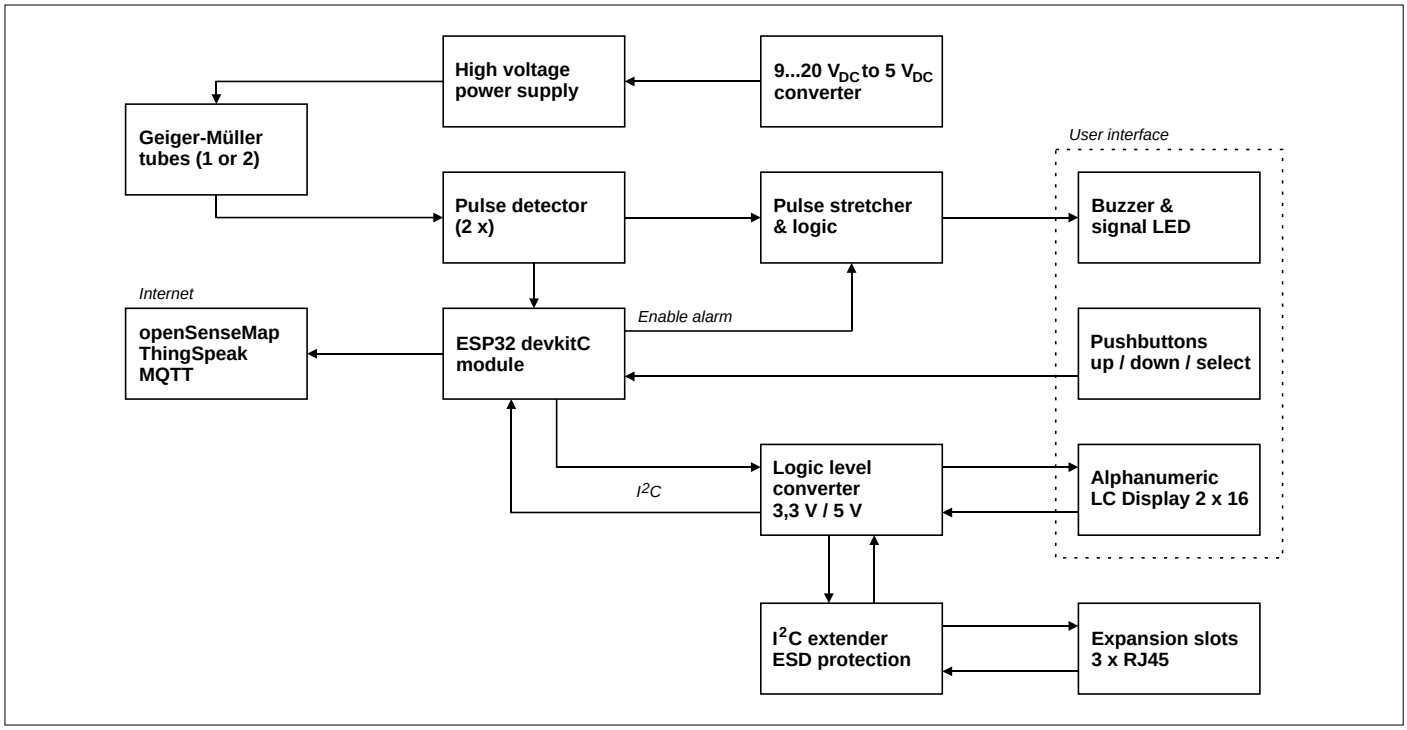


Bild 3. Das Blockschaltbild des Umweltüberwachungssystems zeigt, dass es mehr ist als „nur ein weiterer" Geiger-Müller-Zähler.

G-M-Röhre SBM-20 eine gute Wahl. Hier heißt es aber aufgepasst, da kompatible, aber schlechtere Nachahmerprodukte verkauft werden! Eine andere Option ist die preiswertere LND712, die aber nicht sehr empfindlich auf Gammastrahlung von Cäsium-137 reagiert, so dass sie die von Atomkatastrophen wie in Tschernobyl freigesetzte Strahlung nicht optimal erkennen kann. Wir haben auch die Röhren LND71217 und LND78017 ausprobiert. Die erste ist so empfindlich wie die SBM-20, die zweite ist viel empfindlicher.

Geiger-Müller-Zählröhren

Zwei deutsche Physiker stehen Pate für die G-M-Röhren: Johannes Wilhelm „Hans" Geiger erfand 1908 das Funktionsprinzip, während Walther Müller die Idee 1928 zu einem praktisch einsetzbaren Sensor weiterentwickelte.

G-M-Röhren bestehen normalerweise aus einer Metallröhre als Kathode und einem zentralen Anodendraht (**Bild 2**). Einige Röhren sind mit einem Glimmerfenster zum Nachweis von Alpha-Partikeln und niederenergetischer Betastrahlung, die nicht in die Metallwand der Röhre eindringen können, ausgestattet. Auch Röhren aus Glas statt aus Metall und sogenannte Pfannkuchenröhren sind weit verbreitet [4].

Die Röhre ist mit einer Mischung aus einem Zähl- und einem Löschgas gefüllt. Das inerte (reaktionsträge) Zählgas wie Neon oder Argon und das Löschgas (Ethanol oder ein Halogen wie Chlor oder Brom) besitzen nur einen sehr niedrigen Druck von wenigen Zehntel des Atmosphärendrucks. Zwischen Anode und Kathode wird eine Hochspannung - typischerweise zwischen 300 V und 1200 V - angelegt.

Ein Alpha- oder Beta-Partikel, das in die Röhre eintritt, ionisiert das Gas. Dasselbe geschieht indirekt durch den photoelektrischen Effekt, wenn ein Gamma-Photon auf die Röhre trifft und ein Elektron von der Innenwand der Röhre in das Füllgas schlägt.

Die Townsend-Lawine

Bei der Ionisierung des Füllgases entstehen Paare positiv geladener Ionen und freier Elektronen. Das elektrische Feld im Inneren der Röhre beschleunigt die positiven Ionen zur Kathode und die negativen Elektronen zur Anode. Wenn die freien Elektronen genügend „Schwung" gewinnen, ionisieren sie auf ihrem Weg andere Gasmoleküle und erzeugen zusätzliche freie Elektronen, die wiederum noch mehr Gasmoleküle ionisieren, und so weiter. Das Ergebnis dieser sogenannten Townsend-Lawine ist ein leicht detektierbarer elektrischer Impuls.

Die Lawinen verebben, wenn die Stärke des elektrischen Feldes aufgrund der Ansammlung von positiven Ionen um die Anode herum abnimmt. Das Löschgas hat die Aufgabe, diesen Vorgang abzukürzen und störende Sekundärentladungen zu vermeiden, indem es verhindert, dass UV-Photonen emittiert werden, wenn die positiven Ionen die Kathode erreichen. Das Löschgas macht die Röhre also „schneller". Wenn die Lawinenentladung stoppt, ist die Röhre vorübergehend unempfindlich für ein neues ionisierendes Ereignis. Diese so genannte Totzeit ist natürlich unerwünscht.

Vor- und Nachteile von G-M-Röhren

G-M-Röhren sind relativ preiswert, können alle Arten von Strahlung erkennen, sind langlebig und transportabel. Im Vergleich zu Szintillations- und Halbleiterdetektoren ist der Ausgangsimpuls immer gleich, unabhängig vom Energieniveau und der Art der erfassten Strahlung. G-M-Röhren haben einen sehr niedrigen Wirkungsgrad (ε) und erfassen nur einen kleinen Teil der ionisierenden Strahlung, die auf sie trifft. Sie sind in der Regel nicht empfindlich genug, um zum Beispiel eine radioaktive Kontamination von Lebensmitteln zuverlässig zu erkennen.

Schaltung für G-M-Röhren

Jetzt, da wir verstehen, wie G-M-Röhren funktionieren, können wir eine Schaltung für sie entwerfen. Grundsätzlich benö-

tigen wir nur eine Hochspannungsversorgung (HV) der G-M-Röhre und einen Verstärker, der die Ausgangsimpulse hör- und sichtbar macht.

Die Blockschaltung in **Bild 3** ist etwas komplizierter als gerade beschrieben, da einige Funktionen hinzugefügt wurden. Die Grundschaltung besteht aus der HV-Versorgung, der G-M-Röhre, einem Impulsdetektor/verlängerer sowie einem Signalgeber (Summer und LED). Der Entwurf eignet sich für den Anschluss von zwei G-M-Röhren (nützlich für Vergleiche), weshalb es auch zwei Impulsdetektoren gibt. Zur Verarbeitung der Daten

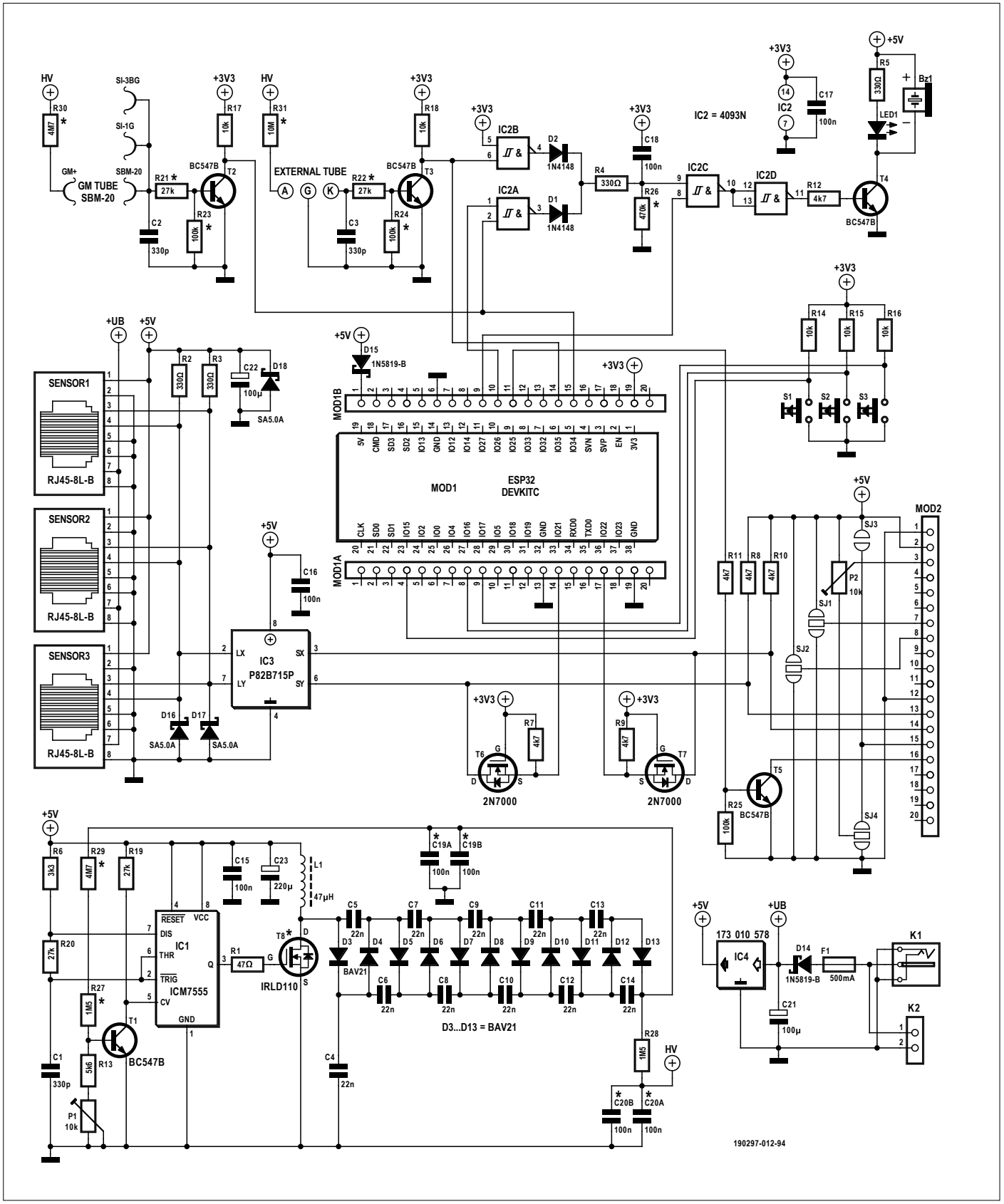


Bild 4. Die elfstufige Diodenkaskade erzeugt die Hochspannung für die Zählröhre, das ESP32-Modul übermittelt die gewonnenen Daten ins Internet.

Bild 5. Die Hochspannungskaskade versorgt die Geiger-Müller-Röhre.

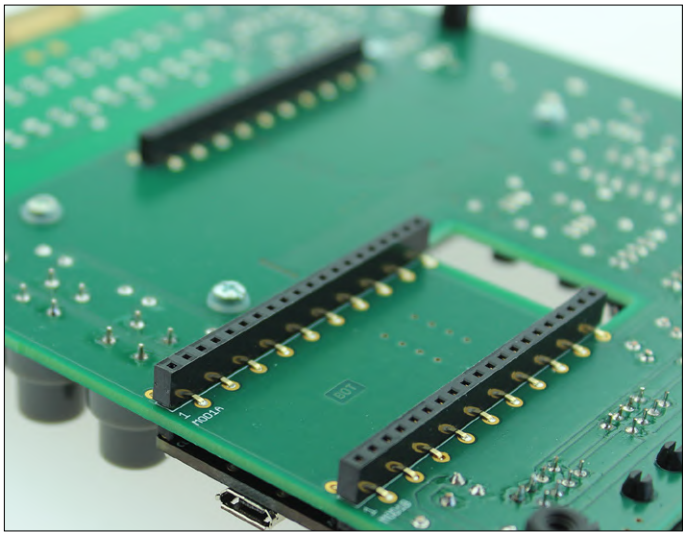


Bild 6. Diese Low-Profile-Fassungen sorgen für eine „schlanke“ Platine.

kommt ein Mikrocontroller zum Einsatz. Die MCU in Form eines ESP32-DevKits zeigt die Messwerte auf einem LCD an und überträgt sie drahtlos ins Internet. Und last not least haben wir drei gepufferte I²C-Erweiterungsanschlüsse für andere geeignete Sensoren (oder auch Aktoren) hinzugefügt. Die „Übersetzung“ der Block- in eine detaillierte Schaltung (**Bild 4**) ist relativ einfach.

Erstens, eine Hochspannungsversorgung

Die Hochspannungsversorgung (HV) ist mit einem 7555-Timer-IC aufgebaut, das mit L1 als Aufwärtswandler arbeitet. Die 11-stufige Hochspannungskaskade erhöht die Ausgangsspannung in etwa um den Faktor sechs (**Bild 5**). Um eine möglichst saubere Gleichspannung zu erhalten, wird der Ausgang der Kaskade vom RC-Glied R28/C20 (A oder B) gefiltert. Da T8, ein MOSFET namens IRLD110, Spannungen von bis zu 100 V schalten kann, beträgt die maximal erreichbare Ausgangsspannung der Kaskade theoretisch 600 V. In der Praxis lässt sich eine Hochspannung von 280...550 V einstellen, was für viele Arten von G-M-Röhren passend ist. Wird für T8 ein IRFD220 verwendet und C19B und C20B (mit höherer Spannungsfestigkeit) anstelle von C19A und C20A eingesetzt, ist eine noch höhere Röhrenspannung möglich. In diesem Fall muss der Wert der Rückkopplungswiderstände R27 und R29 auf je etwa 5,6 MΩ erhöht werden.

Die Hochspannung wird aus stabilisierten 5 V gewonnen und ist somit unabhängig von den 9...20 V_{DC} der Eingangsspannung der Schaltung. Bei einer Ausgangsspannung von 550 V verheizen die Widerstände R27 und R29 etwa 50 mW und werden deshalb etwas warm. Ersetzt man sie durch Typen mit niedrigem Temperaturkoeffizienten, verbessert dies die Stabilität der HV-Versorgung noch weiter.

Zweitens, ein Impulsdetektor mal 2

Die Impulsdetektoren arbeiten auf der „Low-Voltage“-Seite der G-M-Röhre. Sie bestehen aus je einem NPN-Transistor (T2, T3), einem Kondensator und einigen Widerständen. Die Anodenwiderstände R30 und R31 und die Bauteile der Detektorschaltungen sind entsprechend der verwendeten G-M-Röhre(n) auszuwählen. Als Faustregel gilt, dass die Summe der Widerstandswerte an der Transistorbasis etwa 1/45 (0,022) des Anodenwiderstands betragen sollte. Die Kollektoren sind direkt mit der MCU und über 10-kΩ-Pullup-Widerstände mit der 3,3-V-Schiene verbunden.

Und eine Impulsverlängerung

Die Impulse an den Kollektoren sind nur wenige Mikrosekunden lang und damit zu kurz, als dass man sie durch LED1 oder Summer Bz1 wahrnehmen könnte. Deshalb werden sie von der Schaltung R4, R26, C18 und IC2C „gestreckt“. Treffen Impulse vor dem Ablauf dieser Verzögerung ein, so wird die Schaltung kontinuierlich neu getriggert: Ein anhaltender Pfeifton ist die Folge. Spielen Sie mit den Bauteilwerten in der Impulsverlängerung, um diesen Effekt wenn gewünscht zu mildern.

Die Impulsverlängerung hätte auch in die Software implementiert werden können, aber bei hohen Pulsraten (mehrere hundert Impulse pro Sekunde oder höher) würde dies wertvolle Rechenleistung beanspruchen, die wir lieber für wichtigere Aufgaben in Reserve halten.

MCU plus I²C-LCD

Die ESP32 ist ein beliebter Mikrocontroller mit Drahtlos-Netzwerkfunktionen (WLAN). Es gibt zahllose verschiedene ESP32-Module für wenig Geld im Handel. Wir haben uns für das Modul ESP32 DevKitC entschieden und eine Benutzeroberfläche hinzugefügt, die aus drei Drucktasten („Up“, „Down“ und „Select“) und einem 2×16-stelligen alphanumerischen LCD besteht. Dieses Display besitzt praktischerweise eine direkte I²C-Schnittstelle (was nicht üblich ist). Normalerweise werden dazu normale Parallel-Displays mit einer kleinen I²C-zu-Parallel-Wandlerplatine auf der Unterseite ausgestattet, was zwar prima funktioniert, aber das Display recht sperrig macht. Da der ESP32 mit 3,3 V und das Display mit 5 V betrieben werden, werden Pegelwandler (T6, T7) benötigt.

Obwohl das ESP32-Modul über zwei 19-polige Stiftleisten und das LCD nur über 16 Pins verfügt, haben wir uns für 20-polige Buchsenleisten entschieden, da diese leicht erhältlich sind. Die überzähligen Pins bleiben einfach frei.

Mehr I²C wagen!

Drei RJ45-Buchsen stehen für den Anschluss externer Sensoren über übliche Ethernet-Patchkabel zur Verfügung.

Die Buchsen führen die Eingangsspannung UB, die stabilisierten 5 V, Masse und den von IC3, einen P82B715-Busverlängerer gepufferten I²C-Bus. TVS-Dioden schützen die 5-V-Schiene und die I²C-Leitungen vor eingestreuten Hochspannungsspitzen.

Software

Ein Mikrocontroller ohne Software ist wie ein Fisch ohne Wasser. Deshalb haben wir einige Stunden damit verbracht, eine Firmware mit vielen Befehlen zur Steuerung des Systems zu entwerfen (siehe **Kasten** und Kommentare im Quellcode *cmd_proc.ino*).

Befehle können über WLAN oder über die serielle Schnittstelle (den USB-Anschluss des ESP32-Moduls) an den Controller gesendet werden. Die serielle Schnittstelle ist auf 115.200 Bit/s eingestellt. Wenn Sie den Seriellen Monitor der Arduino-IDE verwenden, deaktivieren Sie die Option „No line ending“. Senden Sie die WLAN-Befehle mit PuTTY oder einer TCP-Terminal-App auf Ihrem Smartphone über Port 5010 an die IP-Adresse des Geräts. Die Software wurde in der Arduino-IDE Version 1.8.9 (mit installiertem ESP32) geschrieben.

Impulszählung

Die Impulse der G-M Röhren werden während eines einstellbaren Intervalls gezählt und dann in einen Count-per-Minute-Durchschnittswert (CPM) für dieses Intervall umgewandelt. Aus diesem Wert wird eine Dosisleistung in MikroSievert pro Stunde (µSv/h) berechnet und auf einer IoT-Plattform veröffentlicht. Die LCD-Anzeige wird alle 15 s aktualisiert. Da der Wert auf der Anzahl der in der letzten Minute erfassten Impulse basiert, ist das ausreichend empfindlich für Experimente mit einer radioaktiven Quelle. Die vom LCD angezeigten Werte können aufgrund einer geringeren Glättung etwas von den an das Internet gesendeten Werten abweichen.

Von Impulszahl zum Mikrosievert pro Stunde

Korrigieren Sie zunächst die Zählrate R (in CPM) für die Totzeit der Röhre:

$$R_{corrected} = R_{observed} / (1 - (R_{observed} \times t_{dead})) \text{ [Anzahl/min]}$$

Beachten Sie, dass die Totzeit einer Röhre im Datenblatt in der Regel in Mikrosekunden angegeben wird (µs) und in der obigen Formel in Minuten (min) umgerechnet werden muss, wenn die Zählrate R in „Counts per Minute“ dimensioniert ist. Bei der SBM-20 schlägt die Totzeit von 190 µs bei der Berechnung der CPM-Werte mit weniger als 3.000 CPM zu Buche (weniger als 1% Fehler).

Umrechnungsfaktor

Der zweite (und letzte) Schritt der Berechnung ist einfach, wenn Sie den „Umrechnungsfaktor“ Ihrer G-M-Röhre kennen. Leider ist für die Bestimmung dieses Faktors eine radioaktive Quelle mit einer genau bekannten Aktivität erforderlich. Solche Quellen sind nicht nur sehr teuer, sie müssen in der Regel von einer atomrechtlichen Aufsichtsbehörde genehmigt werden. Alternativ können Sie das Internet nach amtlichen Strahlungsmessstellen [5] in Ihrer Nähe durchsuchen und den Umrechnungsfaktor Ihres Systems so einstellen, dass die Messungen mit den offiziellen Werten übereinstimmen. Für die in unserem Prototyp verwendete SBM-20 Röhre erzielten wir gute Ergebnisse mit einem Umrechnungsfaktor von 0,003931. Wenn der Umrechnungsfaktor k bekannt ist, ist die Dosisleistung D:

$$D = k \times R_{corrected} \text{ [µSv/h]}$$

Konfiguration des Umweltüberwachungssystems

Befehle zur Konfiguration der G-M Röhren			
gm N			**G-M Röhre, N=1...2**
	set		Einstellungen
		attached|att YN	die Röhre ist physisch befestigt Ja/Nein
		accperiod|ap N	Zeitraum für die Meldung des kumulierten Wertes (min)
		deadtime|dt N	Totzeit (µs)
		cnvfactor|cf N	Umrechnungsfaktor
		threshold|th N	Schwellwert für Alarm (nSv/h)

Befehle zur Konfiguration des Summers und der LED			
Summer			**Summer und LED**
	alarm|al YN		Ein/ Ausschalten des Alarmtons
	set		Einstellungen
		aktiviert|ena YN	(De-) Aktivieren des Summers

Wenn Sie die Messungen auf einer IoT-Plattform veröffentlichen möchten, können Sie openSenseMap und/oder ThingSpeak wählen (Konten erforderlich). Für ThingSpeak benötigen Sie einen Schreib-API-Schlüssel und einen Feldnamen für jede G-M-Röhre, bei openSenseMap eine SenseBox-ID und eine Sensor-ID.

thingspeak|thsp				**ThingSpeak-Client**
	set			**Einstellungen**
		gm N		G-M Röhre, N=1...2
			writekey|wk "..."	API-Schlüssel schreiben
			fieldname|fn "..."	Feldname
opensensemap|osmap				**OpenSenseMap-Client**
	set			**Einstellungen**
		gm N		G-M Röhre, N=1..2
			senseboxid|sbid "..."	SenseBox-ID
			sensorid|sid "..."	Sensor-ID

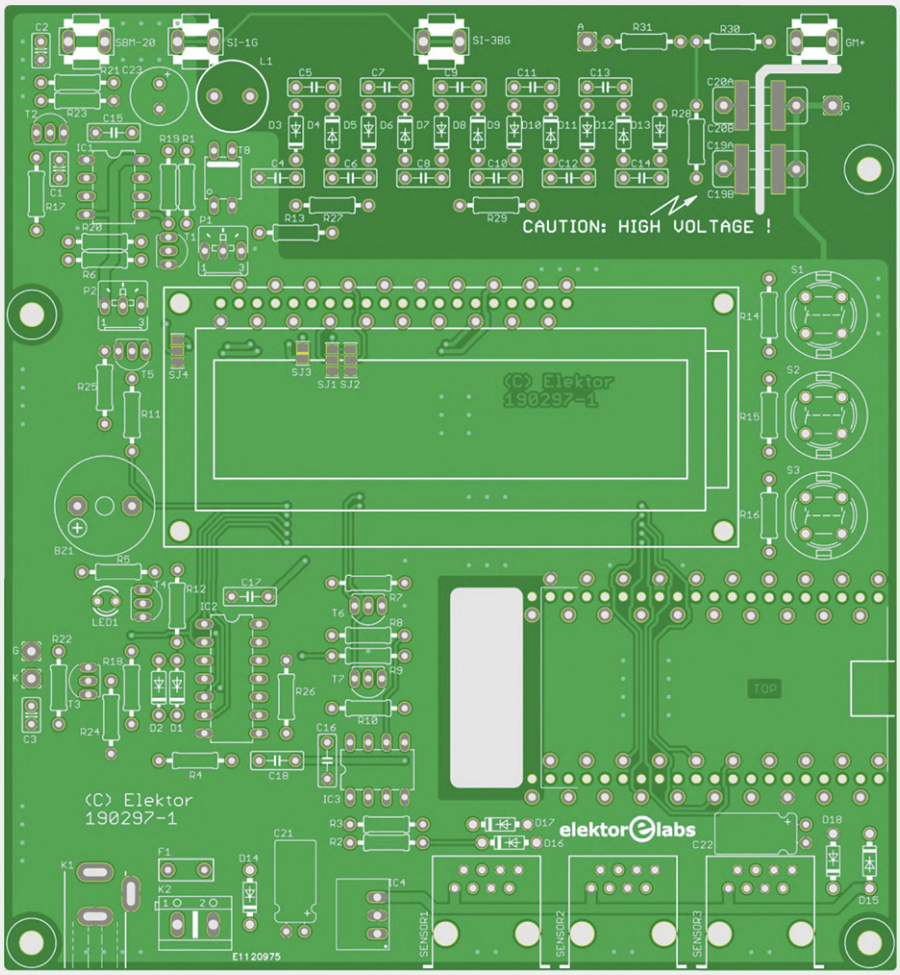
STÜCKLISTE

Widerstände
R1 = 47 Ω
R2...R5 = 330 Ω
R6 = 3k3
R7...R12 = 4k7
R13 = 5k6, 1%
R14...R18 = 10 k
R19...R22 = 27 k
R23,R24,25 = 100 k
R26 = 470 k*
R27,R28 = 1M5, 1,6 kV
(z.B. Vishay HVR2500001504JA100)*
R29,R30 = 4M7, 1,6 kV
(z.B. Vishay VR2500000004704FA500)*
R31 = 10 M, 1,6 kV
(z.B. Vishay VR2500000001005FA500)*
P1,P2 = Trimmpoti 10 k
(z.B. Vishay T73YU103KT20)

Kondensatoren
C1,C2,C3 = 330 p, NP0/COG
C4...C14 = 22 n, 250 V Folienkondensator
(z.B. Epcos B32529C3223J000)
C15...C18 = 100 n, X7R
C19A,C20A = 100 n, 630 V Folienkondensator
(z.B. Epcos B32671P6104K000)*
C19B,C20B = 100 n, X7R 1,5 kV
(z.B. Kemet C2225C104KFRACTU)*
C21,C22 = 100 µ, 25 V
C23 = 220 µ, 16 V, Low ESR

Induktivität
L1 = 47 µ, radial
(z.B. Würth Elektronik 744 772 047 0)

Halbleiter
D1,D2 = 1N4148
D3-D13 = BAV21
D14,D15 = 1N5819-B
D16,D17,D18,D18 = SA5.0A
LED1 = LED 3mm rot
T1...T5 = BC547B
T6,T7 = 2N7000



T8 = IRLD110PBF*
IC1 = ICM755555
IC2 = CD4093NN
IC3 = P82B715P
IC4 = Abwärtsregler Vin 8-28V, Vout 5V, 1A,
Würth Elektronik 173 010 578

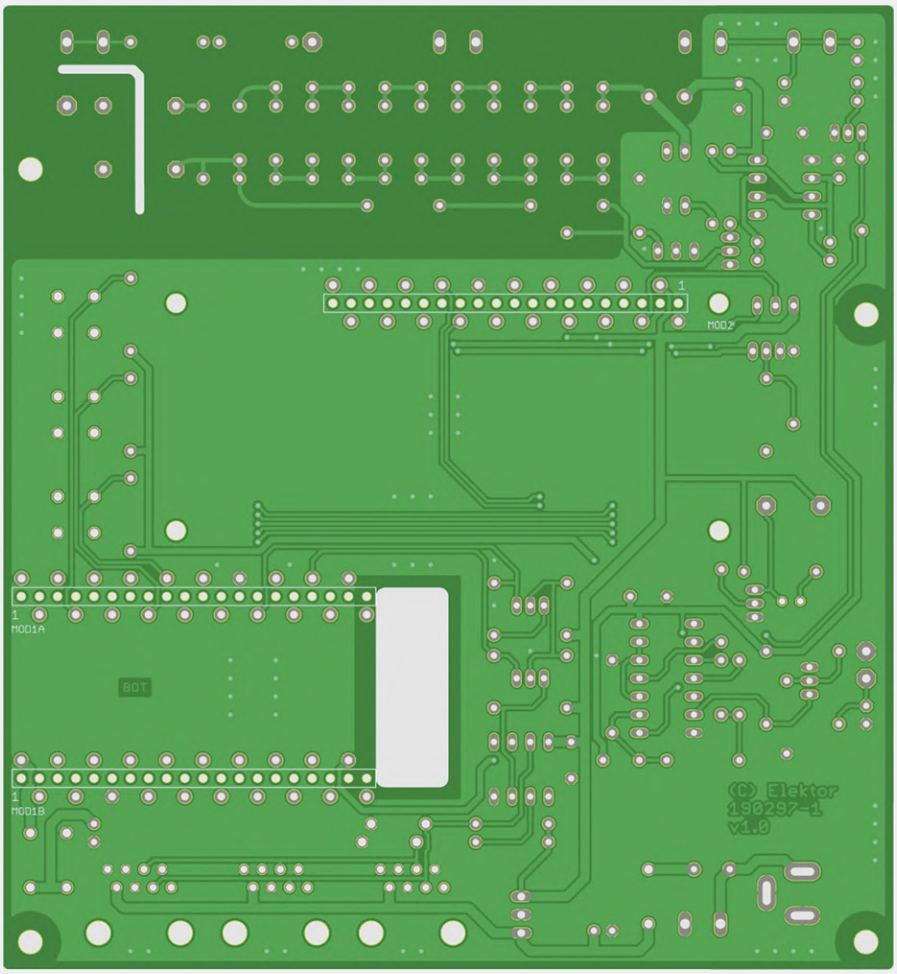
Außerdem
BZ1 = Elektromagnetischer Summer
(z.B. Loudity LD-BZEG-1205/3)

F1 = Rückstellende Sicherung 500 mA
(z.B. Littlefuse 60R050XPR)
K1 = DC-Buchse 2,1/5,5 mm
(z.B. Ninigi PC-GK2,1)
K2 = 2-polige Platinen-Anschlussklemme,
Raster 5 mm
MOD1A,MOD1B,MOD2 = 1x20-polige
Buchsenleiste, Raster 0,1“, mit bottom entry
(z.B. Würth Elektronik 61302015721)

Bestückungsarbeiten

Wir beginnen den Aufbau der Schaltung wie gewohnt mit den kleinsten Bauteilen wie den BAV21-Dioden (D3...D13) und arbeiten uns zu den voluminösesten Bauteilen vor. Montieren Sie alle Teile, aber nicht das LCD und das ESP32-Modul. Stecken Sie IC1 in seine Fassung, lassen Sie aber IC2 und IC3 noch unbestückt. Montieren Sie entweder die Kondensatoren C19A und C20A oder C19B und C20B (nicht alle vier). Die TVS-Dioden D16 und D17 stecken recht straff in ihren Bohrungen. Biegen Sie die Drähte von D17 vorsichtig um und stellen Sie sicher, dass D16 nicht der benachbarten RJ45-Buchse im Weg steht beziehungsweise liegt.

Um die Gesamthöhe der bestückten Leiterplatte gering zu halten, werden das LCD und das ESP32-Modul über spezielle (Bottom-Entry)-Buchsen (MOD1A, MOD1B & MOD2) auf der Lötseite der Leiterplatte montiert (**Bild 6).** Sie kön-



S1,S2,S3 = Taster
(z.B. C&K Components D6R10LFS)
SENSOR1-SENSOR3 = RJ45-Buchse
(z.B. Molex 85503-5001)
Alphanumerisches LCD mit 2x16 Zeichen, I2C (z.B. Raystar Optronics RC1602B5-LLH-JWV)
ESP32-DevkitC mit Stiftleisten
2 St. 5-mm-Sicherungshalter-Clips (für G-M-Röhre)

* siehe Text

nen die Pins der MOD2-Fassung ein wenig kürzen, so dass sie nicht die Unterseite des LCD-Moduls berühren.
Stellen Sie die I²C-Adresse des LCDs mit einem Tropfen Lötzinn auf den Jumper-Pads SJ1 und SJ2 ein. Wenn Ihr LCD (wie das in der Stückliste angegebene) eine negative Kontrastspannung (VEE) an Pin 15 benötigt, lassen Sie die Lötbrücke SJ3 offen und verbinden die Pads 1/2 bei SJ4.

Prüfung

Mit Trimmpoti P1 in Mittelstellung wird eine Stromversorgung (9...20 V_{DC}) an K1 oder K2 angeschlossen, die beim Einschalten des Geräts etwa 500 mA (bei 9 V_{DC}) liefern können muss. Ist die 5-V-Spannung vorhanden? Messen Sie die Spannung an C19 und stellen Sie P1 so ein, dass die Hochspannung einen für Ihre G-M-Röhre geeigneten Wert erreicht (400 V für die SBM-20). Seien Sie dabei sehr vorsichtig; das Berühren der Hochspannung ist zwar nicht tödlich, aber sie zwirbelt doch ganz gemein!
Sind alle Versorgungsspannungen korrekt, schalten Sie das Netzteil wieder aus und installieren Sie IC2, IC3, das LCD und das ESP32-Modul. Sie können auch die G-M-Röhre anschließen (Polarität prüfen!).
Schalten Sie das System wieder ein und verbinden Sie es per USB-Kabel mit einem Computer, auf dem eine neue Arduino-IDE installiert ist. Kompilieren Sie den Sketch und laden Sie ihn zum ESP32-Modul hoch. Alternativ können Sie auch die Binärdateien direkt auf den ESP32 hochladen.
Stellen Sie das Kontrast-Trimmpoti P2 so ein, dass der Text auf dem LCD gut lesbar ist.
Mit den Tasten „Up“ und „Down“ wählen Sie einen Sensor aus (derzeit nur G-M-Röhre 1 und G-M-Röhre 2) und drücken Sie „Select“, um zwischen der Anzeige in µSv/h und CPM zu wechseln. Sie können auch den Summer und die LED ein- oder ausschalten und einen Strahlungsalarm zurücksetzen, Datum und Uhrzeit von einem NTP-Server anzeigen und eine Zeitzone auswählen. Und schließlich können Sie Ihr Umweltüberwachungssystem an eine IoT-Plattform anbinden.

Zweite Röhre

Es können zwei G-M-Röhren angeschlossen werden, allerdings müssen sie für die gleiche Betriebshochspannung ausgelegt sein. Sie können die zweite Röhre auch über ein doppelt abgeschirmtes Kabel anschließen, das aber so kurz wie möglich sein sollte. Verbinden Sie den Anodenwiderstand direkt mit der G-M-Röhre. Da die Hochspannung das Kabel bei Berührung durchschlagen könnte, sollte man zwei „halbe“ Widerstände in Reihe schalten, einen auf der Platine und einen an der G-M-Röhre. ⏮

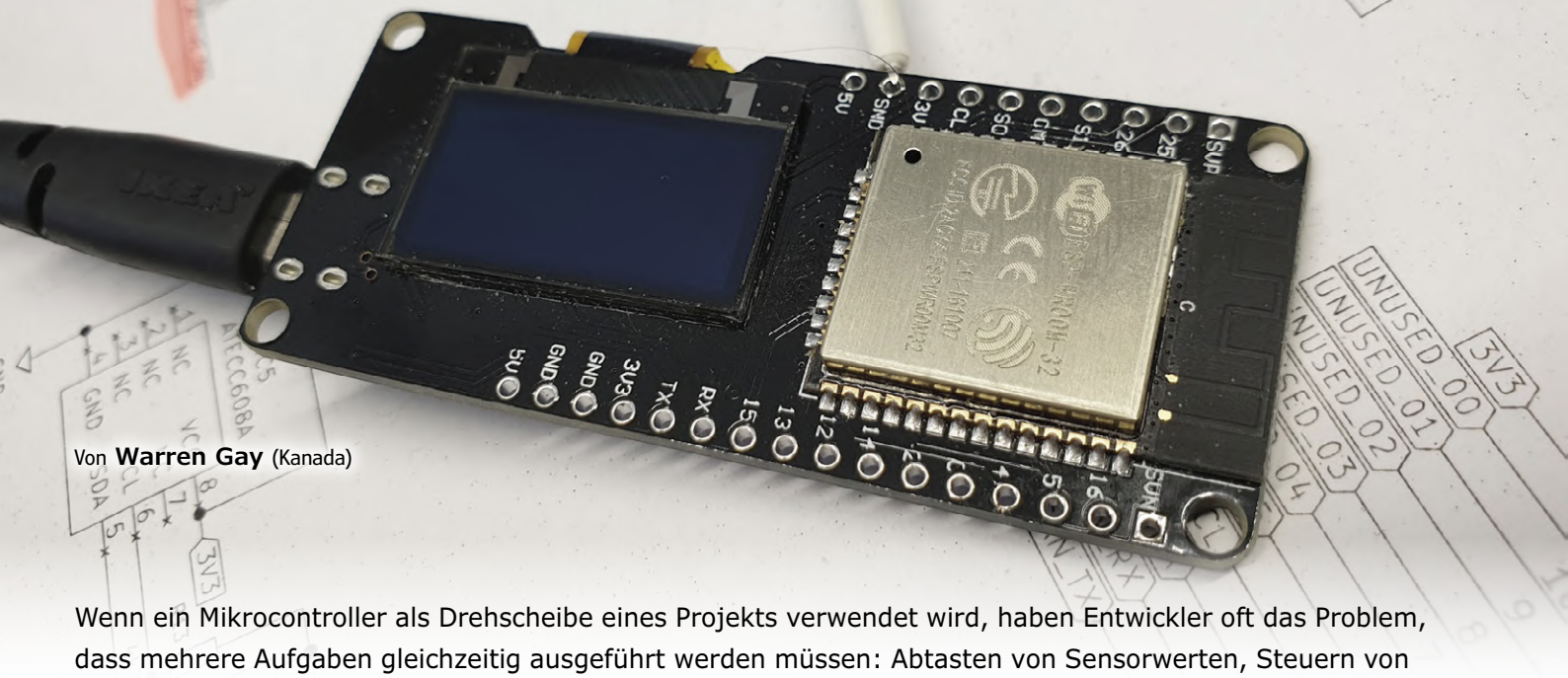
190297-03

Weblinks

[1] Radioactivity is in the Air for You and Me - bionerd 23 @ EHSM - v4: www.youtube.com/watch?v=6Z5mRpFdT5I
[2] Kosmische Luftschauer: https://de.wikipedia.org/wiki/Luftschauer
[3] Radioaktive Stoffe im Haushalt: www.youtube.com/watch?v=LAdVS0KC2UM
[4] Geiger-Müller-Röhren: www.orau.org/ptp/collection/GMs/GMs.htm
[5] ODL-Info: https://odlinfo.bfs.de/DE/index.html

Praktisches ESP32-Multitasking

Task-Programmierung mit FreeRTOS und der Arduino-IDE



Von **Warren Gay** (Kanada)

Wenn ein Mikrocontroller als Drehscheibe eines Projekts verwendet wird, haben Entwickler oft das Problem, dass mehrere Aufgaben gleichzeitig ausgeführt werden müssen: Abtasten von Sensorwerten, Steuern von Stellgliedern, Visualisieren von Gerätezuständen und/oder Warten auf die Eingabe eines menschlichen Nutzers. Glücklicherweise kann dieses Problem mit der Task-Programmierung auf Basis leichtgewichtiger Embedded-Betriebssysteme auf sehr elegante Weise gelöst werden. FreeRTOS ist ein solches einfaches und weit verbreitetes Open-Source-Betriebssystem, das für viele Mikrocontroller-Plattformen verfügbar ist. Das sehr beliebte Tandem ESP32 und Arduino-IDE macht es besonders einfach, FreeRTOS für die Task-Programmierung zu nutzen, da es bereits in die Core-Bibliotheken integriert ist. Vermutlich haben Sie FreeRTOS schon immer benutzt, ohne es zu wissen!

Die ESP32-Plattform der Firma Espressif ist ein spannender Mikrocontroller für Maker-Projekte. Seine Hardwarefähigkeiten, die umfangreiche Software-Unterstützung und der moderate Preis machen ihn für viele zu einem „must have“. Es ist sowohl eine Arduino-kompatible als auch eine native Entwicklungsumgebung (ESP-IDF, Espressif IoT Development Framework) verfügbar. In diesem Artikel wird die vertraute Arduino-Umgebung verwendet. Der größte Teil des API (Application Programming Interface) ist für beide Entwicklungsumgebungen geeignet [1]. Einige unserer Leser wissen vielleicht, dass Espressif das beliebte Open-Source-Embedded-Betriebssystem FreeRTOS verwendet, um die ESP32-Bibliotheksfunktionen für WLAN, Bluetooth und viele weitere Funktionen des Mikrocontrollers zu realisieren. In diesem Artikel werden wir sehen, dass FreeRTOS [2] und die Task-Programmierung auch für die einfachen Arduino-Funktionen `setup()` und `loop()` verwendet wird. Zuerst werden wir ein einfaches ESP32-Demoprojekt beschreiben [3], dann schauen wir hinter die Kulissen und werden später unsere eigenen Tasks für die Demo festlegen.

Applikation

Für diesen Artikel haben wir eine kleine Testanwendung entwickelt, die die Stellung eines Potis mit dem Analog-Digital-Wandler (ADC) einliest und in einem OLED-Display als Balkendiagramm anzeigt. Zusätzlich steuern wir eine LED mit Pulsweitenmodulation (PWM) an und variieren sie ja nach Potistellung von schwach bis hell leuchtend.

Dieser Artikel verwendet das ESP32-Board von Lolin, das ein integriertes OLED-Display besitzt (**Bild 1**) [4]. Wenn Sie ein anderes ESP32-Board verwenden, das über kein SSD1306-kompatibles OLED-Display verfügt, können Sie das OLED-Display im Programm deaktivieren und sich stattdessen auf die Helligkeitssteuerung der LED konzentrieren.
Die LED und das Potentiometer werden wie in **Bild 2** an den ESP32 angeschossen.
Wenn die Anwendung läuft, kann durch Drehen des Potentiometers die Eingangsspannung an GPIO36 verändert werden. Diese Spannung wird mit einer Auflösung von 12 Bit abgetastet, der Wert liegt also zwischen 0...4095. Wenn Sie das Poti im Uhrzeigersinn bis zum Anschlag drehen, wird ein Wert von 4095 angezeigt und die LED leuchtet hell. Sollte das nicht so sein, vertauschen Sie einfach die äußeren Anschlüsse des Potis. Der mittlere Anschluss ist der Schleifer des Potis, an dem man Spannungen im Bereich 0...+3,3 V messen kann.

Achten Sie darauf, dass Sie das Poti nicht an der +5-V-Versorgung anschließen, da dies die maximal erlaubte Spannung des GPIOs überschreitet.

Programm-Konfiguration

Kommen wir nun zur Software. Unser Programm definiert C-Makros (siehe **Listing 1**), um eine Neukonfiguration zu vereinfachen. Setzen Sie Makro `CFG_OLED` auf Null, wenn Sie keine Anzeige verwenden.
Der Wert des Makros `CFG_ADC_GPIO` bestimmt, dass Kanal 0 (GPIO36) des ADC1 verwendet wird. `CFG_LED_GPIO` konfiguriert GPIO13 für den Anschluss der LED.
Der vollständige Quellcode ist bei github im Unterverzeichnis freertos-tasks1 [3] verfügbar.

Initialisierung

Abgesehen von den Tasks findet sich in **Listing 2** das `setup()` für unsere Anwendung, eine typische Initialisierung in einem Arduino-Sketch.
Die Anzeige wird nur initialisiert, wenn der entsprechende Code kompliliert wurde. Danach wird der ADC mit der Arduino-Routine `analogReadResolution()` so konfiguriert, dass er 12-Bit-Werte liest. Die Funktion `analogSetAttenuation()` wird aufgerufen, um einen Wert von 0...+3,3 V zu lesen. Der ADC besitzt nämlich einen internen Verstärker, so dass diese Konfiguration wichtig ist, damit der richtige Bereich verwendet wird.
Danach wird der GPIO-Pin für die LED (`gpio_led`) als Ausgang und durch Aufrufe von `ledcAttachPin()` und `ledcSetup()` für eine PWM konfiguriert.

Balkenanzeige

Über den OLED-Displaycontroller SSD1306 zeichnet die Applikation ein Balkendiagramm, das den ADC-Wert anzeigt (siehe **Listing 3**).

Listing 1. Konfigurationsoptionen.

```
// Set to zero if NOT using SSD1306
#define CFG_OLED            1

// I2C address of SSD1306 display
#define CFG_OLED_ADDRESS    0x3C

// GPIO for display I2C SDA
#define CFG_OLED_SDA        5

// GPIO for display I2C SCL
#define CFG_OLED_SCL        4

// Pixel width of display
#define CFG_OLED_WIDTH      128

// Pixel height of display
#define CFG_OLED_HEIGHT     64

// GPIO for ADC input
#define CFG_ADC_GPIO        36

// GPIO for PWM LED
#define CFG_LED_GPIO        13
```

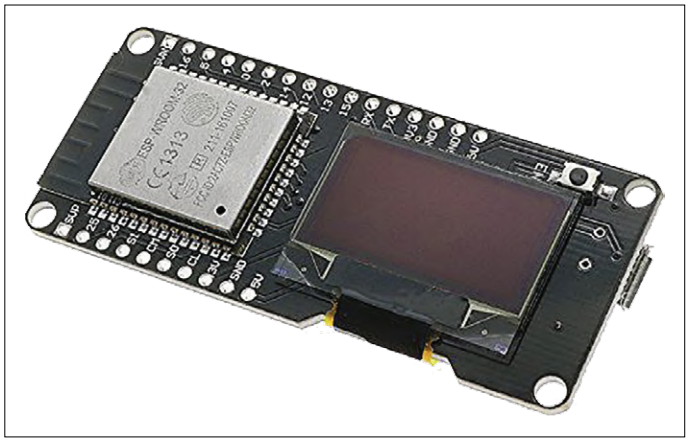


Bild 1. Lolin-ESP32 mit OLED-Display.

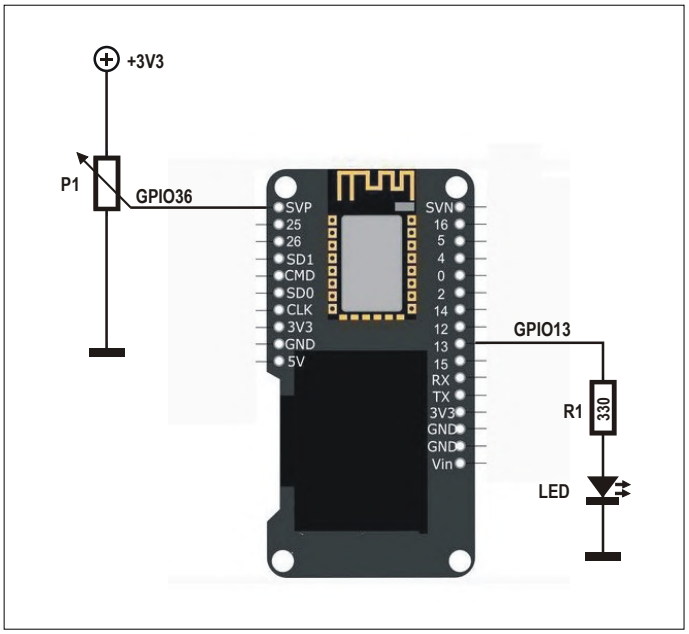


Bild 2. Lolin-ESP32-Modul mit Potentiometer und LED.

Listing 2. Die setup()-Funktion.

```
void setup() {

#if CFG_OLED
  display.init();
  display.clear();
  display.setColor(WHITE);
  display.display();
#endif

  analogReadResolution(12);
  analogSetAttenuation(ADC_11db);

  pinMode(gpio_led,OUTPUT);
  ledcAttachPin(gpio_led,0);
  ledcSetup(0,5000,8);
}
```

Listing 3. Die Balkenanzeige-Funktion.

```
#include "SSD1306.h"
...
void barGraph(unsigned v) {
  char buf[20];
  unsigned width, w;

  snprintf(buf,sizeof buf,"ADC %u",v);
  width = disp_width-2;
  w = v * width / 4095;
  display.fillRect(1,38,w,disp_height-2);
  display.setColor(BLACK);
  display.fillRect(w,38,disp_width-2,
    disp_height-2);
  display.fillRect(1,1,disp_width-2,37);
  display.setColor(WHITE);
  display.drawLine(0,38,disp_width-2,38);
  display.drawRect(0,0,disp_width-1,
    disp_height-1);
  display.setTextAlignment(TEXT_ALIGN_CENTER);
  display.setFont(ArialMT_Plain_24);
  display.drawString(64,5,buf);
  display.display();
}
```

Die Funktion `barGraph()` akzeptiert einen Eingangswert `v` (den ADC-Wert) und formatiert diesen mit `snprintf()` zu einer kurzen Nachricht im Array `buf`. Die Rechtecke werden teils schwarz und teils weiß „gezeichnet“, um den ADC-Wert als Balkendiagramm darzustellen. In der Methode `display.drawString()` wird auch ein formatierter Text auf dem Display platziert. Am Ende wird die Methode `display.display()` aufgerufen, um das Anzeigebild aus dem Arbeitsspeicher in den OLED-Controller zu übertragen.

Die loop()-Funktion

Würde es sich um ein normales Arduino-Programm handeln, so könnten wir eine Schleife wie in **Listing 4** schreiben.
In diesem Code sind die Schritte einfach:

- Lesen des 12-Bit-ADC-Werts (von 0 bis 4095)
- Anzeige im Balkendiagramm (wenn konfiguriert)
- Ausgabe von „ADC“ und des Wertes im seriellen Monitor
- LED-Helligkeit durch Ändern des PWM-Parameters einstellen
- Verzögerung um 50 Ticks.

Listing 4. Typischer Schleifencode bei Arduino.

```
void loop() {
  uint adc = analogRead(ADC_CH);

#if CFG_OLED
  barGraph(adc);
#endif

  printf("ADC %u\n",adc);
  ledcWrite(0,adc*255/4095);
  delay(50);
}
```

Tabelle 1.
Tasks, die beim Arduino-Start ausgeführt werden.

Task-Name	Task #	Priority	Stack	CPU
loopTask	12	1	5188	1
Tmr Svc	8	1	1468	0
IDLE1	7	0	592	1
IDLE0	6	0	396	0
ipc1	3	24	480	1
ipc0	2	24	604	0
esp_timer	1	22	4180	0

Das Programm ist bewusst einfach gehalten. Aber wenn diese Anwendung komplizierter wäre, würde man sie in kleinere Aufgaben (Tasks) unterteilen.

Was sind Tasks?

Mit einer ESP32-CPU mit zwei Kernen können zwei Programme *gleichzeitig* ausgeführt werden. Das ist eine spannende Angelegenheit! Jede CPU arbeitet mit einem eigenen Programmzähler und nutzt andere Register zur Ausführung von Anweisungen. Dies entspricht zwei Ausführungssträngen (*Threads*), im Gegensatz zu einer Single-Core-CPU mit nur einem Thread zu jedem Zeitpunkt.
Um noch mehr als zwei Programm-Aufgaben (*Tasks*) simultan auszuführen, wird ein Trick verwendet, nämlich eine Aufgabe auszusetzen und eine andere aufzunehmen. Dies ist die Hauptaufgabe von FreeRTOS. Dieses Wechselspiel vollzieht sich mit einer solchen Geschwindigkeit, dass es so erscheint, als würden die Tasks gleichzeitig ausgeführt, auch wenn in Wirklichkeit zu einem bestimmten Zeitpunkt (bei einer Dual-Core-CPU) nicht mehr als zwei Tasks gleichzeitig ausgeführt werden.
Um die Ausführung mehrerer scheinbar gleichzeitiger Tasks zu verwalten, speichert die FreeRTOS-Zeitsteuerung die Register des aktuellen Tasks und stellt die Register des nächsten auszuführenden Tasks (wieder) her. Auf diese Weise können mehrere Aufgaben unabhängig voneinander ausgeführt werden. Zusätzlich zum *program counter register* für jeden Task müssen auch die *stack pointer* gespeichert und wiederhergestellt werden. Dies ist notwendig, da C/C++-Programme Variablen und Return-Funktionsadressen auf dem Stack speichern. Jeder Task besitzt natürlich einen eigenen Stack.
Auf dem ESP32 laufen alle Tasks im gleichen Speicherbereich (im Gegensatz zum Beispiel zum Raspberry Pi). Aus diesem Grund muss ein Task sehr sorgfältig programmiert werden, damit er den von anderen Tasks verwendeten Speicherplatz nicht beschädigt. Wenn Sie einen Multi-CPU-Kern verwenden, gibt es noch andere Überlegungen, die aber nicht Thema dieses Artikels sein sollen (sondern eines künftigen).

Arduino-Tasks

Wenn Ihr ESP32 ein Arduino-Programm startet, ist das dann ein Task? Auf jeden Fall! Zusätzlich zu Ihrem eigenen Task beim Starten werden noch FreeRTOS-Tasks ausgeführt. Einige dieser Tasks stellen Dienste wie Timer, TCP/IP oder Bluetooth bereit, zusätzliche Tasks können vom Anwender gestartet werden.
Tabelle 1 zeigt ein Beispiel für FreeRTOS-Tasks, die ausgeführt werden, wenn die Arduino-Funktionen `setup()` und `loop()` aufgerufen werden. Der *loopTask* ist der Haupt-Arduino-Task. Die Spalte mit der Bezeichnung Stack zeigt die von FreeRTOS

nicht genutzten Stack-Bytes. Die Planung der Stackgrößen wird zu einem späteren Zeitpunkt behandelt.
Die Tabelle ist in umgekehrter chronologischer Reihenfolge nach Task-Nummern sortiert. Der Haupt-Task (*loopTask*) ist der zuletzt erstellte Task. Fehlende Task-Nummern deuten darauf hin, dass einige Tasks gestartet und beendet wurden, als ihre Aufgabe erledigt war. Die Spalte *Priority* zeigt die zugewiesenen Taskprioritäten (mit Null als niedrigster Priorität). Im Moment sollten Sie nur im Hinterkopf behalten, dass Prioritäten in FreeRTOS etwas anders funktionieren als beispielsweise unter Linux. Schließlich werden die Aufgaben zwischen CPU 0 und CPU 1 aufgeteilt. Espressif lässt seine Support-Tasks in CPU 0 ablaufen, während Applikations-Tasks CPU 1 verwenden. Dadurch laufen die Dienste für TCP/IP, Bluetooth und ähnliche reibungslos und ohne besondere Beeinflussung durch Ihre Anwendung. Trotz dieser Konvention können Sie aber beiden CPUs zusätzliche Tasks zuweisen.

Arduino-Startup

Es ist gut zu wissen, wie der ESP32 Arduino-Programme vor dem Aufruf von `setup()` initialisiert. Betrachten Sie den (vereinfachten, die Watchdog-Timer-Elemente wurden weggelassen) Programmschnipsel in **Listing 5**. Aus diesem Beispiel können wir einige interessante Dinge lernen:

- Beachten Sie, dass es sich um ein C++-Startup handelt (daher die externe „C“-Deklaration von `app_main()`).
- Einige Arduino-Initialisierungen werden von `initArduino()` durchgeführt.
- Der *loopTask* wird durch den Aufruf von `xTaskCreatePinnedToCore()` mit einer Reihe von Argumenten erstellt und ausgeführt.

Das erste Argument ist die Adresse der Funktion, die für den Task (loopTask) ausgeführt werden soll. Wenn der Task erstellt wird, führt die Funktion `loopTask()` zunächst einen Aufruf von `setup()` und dann von `loop()` aus einer „ewigen“ For-Schleife aus.
Das zweite Argument ist ein String, der den Namen des Tasks als C-String („`loopTask`“) beschreibt. Das dritte Argument gibt die Stackgröße mit 8192 *Bytes* an. Beachten Sie, dass sich FreeRTOS hier von anderen Plattformen unterscheidet, die stattdessen 4-Byte-Wörter verwenden. Wenn der Task erstellt wird, wird diese Menge von Bytes aus dem Heap-Speicher dem Task zugeordnet, bevor die Funktion aufgerufen wird. Eine nicht unbeträchtliche Menge an SRAM wäre verschwendet, wenn der Main-Task nicht verwendet werden würde.
Alle FreeRTOS-Tasks akzeptieren einen Void-Pointer als Argument. In diesem Fall wird der Wert nicht verwendet und auf NULL gesetzt. Das fünfte Argument ist die FreeRTOS-Priorität, die dem Task zugewiesen wird, in diesem Beispiel „1“. Bevor die FreeRTOS-Prioritäten weiter erläutert werden, verwenden wir erst einmal die Priorität 1.
Nach dem Prioritätsargument ist ein Pointer auf eine Variable gesetzt, die das Handle zum Task enthält. Hier wird das Handle nicht verwendet und auf NULL gesetzt. Das letzte Argument gibt an, auf welchem CPU-Kern der Task laufen soll. Beim dualen ESP32 kann es „0“ oder „1“ sein. CPU 1 wird verwendet, um den Main-Task zu starten. Bei Single-Core-Prozessoren kann das Argument nur Null sein.

Erstellen eines Tasks

Stellen wir uns einmal vor, unsere Anwendung wäre kompliziert und wir müssten den Code in zwei Tasks unterteilen. Wir

Listing 5. Vereinfachter ESP32-Arduino-Start.

```
void loopTask(void *pvParameters) {
  setup();
  for (;;) {
    loop();
  }
}

extern "C" void app_main() {
  initArduino();
  xTaskCreatePinnedToCore(
    loopTask,       // function to run
    "loopTask",     // Name of the task
    8192,           // Stack size (bytes!)
    NULL,           // No parameters
    1,              // Priority
    &loopTaskHandle, // Task Handle
    1);             // ARDUINO_RUNNING_CORE
}
```

haben bereits den `loopTask()`, der immer wieder `loop()` aufruft. Um den ihm zugewiesenen Stackplatz zu nutzen, würden wir dort normalerweise unseren stackintensivsten Programmteil programmieren.

In unserem Beispieltask führen wir in der Funktion `loop()` folgendes aus:

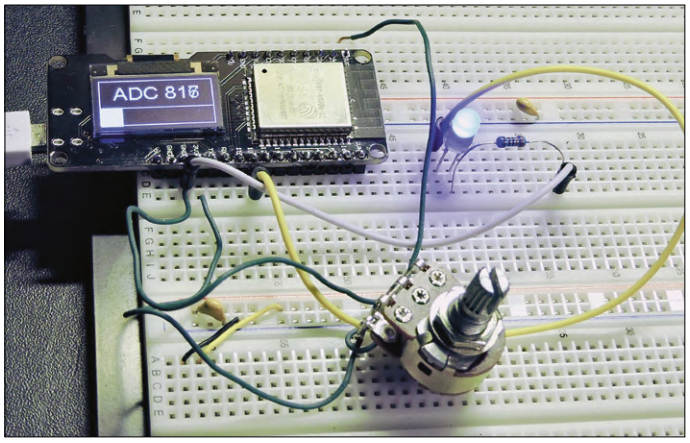


Bild 3. Ausführen der Demo.

- 12-Bit-Wert aus dem ADC lesen
- Wert auf den seriellen Monitor ausgeben
- ADC-Wert an den zweiten Task senden
- 50 Ticks Pause und zurück (von `loop()`)

Damit bleibt für den zweiten Task zu tun:
1. ADC-Wert von der `loop()`-Funktion (im *loopTask*) holen
2. ADC-Wert im Bargraph anzeigen (wenn konfiguriert)
3. PWM für die LED-Helligkeit entsprechend ändern

Der einzig sinnvolle Ort, um den zweiten Task zu erstellen, ist der `setup()`-Task, da diese Aufgabe nur einmal durchgeführt werden muss. Lassen Sie uns also zum `setup()`-Task hinzufügen:

```
void setup() {

#if CFG_OLED
  display.init();
  display.clear();
  display.setColor(WHITE);
  display.display();
#endif

  analogReadResolution(12);
  analogSetAttenuation(ADC_11db);

  pinMode (gpio_led,OUTPUT);
  ledcAttachPin(gpio_led,0);
  ledcSetup(0,5000,8);

...
  xTaskCreatePinnedToCore(
    dispTask,   // Display task
    "dispTask", // Task name
    2048,       // Stack size (bytes)
    NULL,       // No parameters
    1,          // Priority
    NULL,       // No handle returned
    1);         // CPU 1
}
```

Dieser Call startet einen neuen Task namens *dispTask* mit einem Stack von 2048 Bytes. Wir haben ihn der CPU 1 mit der Priorität 1 zugewiesen. Er wird die (noch nicht definierte) Funktion `dispTask()` ausführen. Da der Task auf derselben CPU und mit derselben Priorität wie unser `loopTask()` ausgeführt wird, wird die CPU zwischen `dispTask()`, `loopTask()` und allen anderen Tasks der Priorität 1 geteilt.

Queues

Es fehlt noch ein Bestandteil: Wie senden wir den ADC-Wert von einem Task zu einem anderen? Sollen wir etwa etwas mit dem Speicher versuchen, was für Tasks auf derselben CPU funktionieren könnte? Aber die Angelegenheit ist hier komplizierter, weil Informationen von einer CPU an die andere weitergegeben werden. Die Lösung des Problems sind FreeRTOS-Queues! Die Queue ermöglicht es einem Task, ein Element auf „atomare“ Weise zu speichern. Es ermöglicht dem Empfänger ebenfalls, dieses Element auf atomare Weise abzuholen. Mit „atomar“ ist gemeint, dass das Element nicht nur teilweise gesendet oder empfangen werden kann. Bei einer Dual-Core-CPU mit zwei simultan ausgeführten Befehlen kann es Timing-Probleme und Race Conditions geben. Der Queue-Mechanismus ergreift besondere Maßnahmen, damit die Information atomar übermittelt wird. Eine FreeRTOS-Queue wird wie folgt erstellt:

```
static QueueHandle_t qh = 0;
...
  qh = xQueueCreate(8,sizeof(uint));
```

Das erste Argument betrifft die Queuetiefe (die maximale Anzahl der aufnehmbaren Queueeinträge). Das zweite Argument gibt die Größe jedes in der Queue aufgenommenen Elements an. In diesem Fall ist es die Größe eines *uint*-Wertes in Bytes. Der Rückgabewert ist das Handle der Queues. Die Queue sollte unmittelbar vor der Erstellung des *dispTask* in `setup()` generiert werden, damit der Wert sofort verfügbar ist.
Bauen wir nun die Queue in die Schleife ein:

```
void loop() {
  uint adc = analogRead(ADC_CH);

  printf("ADC %u\n",adc);
  xQueueSendToBack(qh,&adc,portMAX_DELAY);
  delay(50);
}
```

Weblinks

[1] ESP32 API-Referenz: https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/
[2] FreeRTOS-Homepage: http://www.freertos.org
[3] Quellcode des Projekts: https://github.com/ve3wwg/esp32_freertos/blob/master/freertos-tasks1/freertos-tasks1.ino
[4] ESP32-Lolin-Board mit OLED: www.elektor.de/lolin-esp32-oled-module-with-wifi

Das Balkendiagramm und die LED-PWM werden in unserer neuen Funktion `dispTask()`, die immer noch nicht beschrieben ist, aktualisiert. Der `loopTask()` liest jedoch den ADC-Wert in `adc` ein und übergibt ihn dem seriellen Monitor. Danach wird er mit der Funktion `xQueueSendToBack()` auf die „Rückseite“ der Queue geschoben. Wir spezifizieren das Queue-Handle im ersten Argument. Der Wert, der in die Queue gestellt werden soll, wird mit einem Pointer im zweiten Argument bereitgestellt. Das letzte Argument gibt an, wie lange gewartet werden soll, wenn die Warteschlange voll ist. Durch die Angabe des Makros `portMAX_DELAY` geben wir an, dass die Ausführung blockiert werden soll, bis Platz im Queue ist.

Der Display-Task

Nun untersuchen wir den Display-Task mit dem folgenden Code:

```
void dispTask(void *arg) {
  uint adc;

  for (;;) {
    xQueueReceive(qh,&adc,portMAX_DELAY);
    barGraph(adc);
    ledcWrite(0,adc*255/4095);
  }
}
```

Diese Funktion erhält, wie alle Task-Funktionen, beim Start ein Pointer-Argument. Hier wird es nicht genutzt und ist deshalb NULL, da der Task so angelegt wurde.
Der Hauptteil des Tasks ist eine for-Schleife. Hier gibt `xQueueReceive()` den Datenwert der Queue zurück, wartet aber ansonsten „ewig“, wenn die Queue leer ist (der dritte Argumentwert ist `portMAX_DELAY`). Nach dem Empfang eines Wertes aus der Queue wird die Balkenanzeige aktualisiert und der LED-PWM-Wert geändert.
Beachten Sie, dass in diesem Display-Task kein `delay()` aufgerufen werden muss. Der Grund dafür ist, dass der Task automatisch in der Funktion `xQueueReceive()` wartet, wenn die Queue leer ist. Das Tempo der Ausführung wird also vom sendenden Task bestimmt.

Ausführen der Demo

Nach dem Aufbau der Schaltung und der Programmierung sollten Sie die Balkenanzeige und das LED-Licht sehen (**Bild 3**). Wenn die LED dunkel ist, sollte ein Dreh am Poti sie erhellen. Wenn Sie das Programm ohne Displaynutzung kompiliert haben, dann starten Sie den seriellen Monitor und schauen Sie die ausgegebenen Zeilen an, in der Form „ADC 3420“. Dreht man das Poti gegen den Uhrzeigersinn, wird die PWD-LED dunkler, mit dem Uhrzeiger heller. Ebenso werden die ADC-Werte niedriger beziehungsweise höher.

Zusammenfassung

Wir haben in diesem Artikel die ESP32-Startprozedur kennengelernt. Auch die Erstellung des Haupttasks namens `loopTask` wurde abgedeckt. In der Demo haben wir einen eigenen zusätzlichen Task zur Steuerung des OLED-Displays und der LED-PWM erstellt und ihr über eine Queue Daten übermittelt. Dieser Code läuft unabhängig vom Haupttask.
Zusätzlich haben wir beim Haupttask gelernt, dass ihm (aus dem Heap) ziemlich viel Stackplatz zugewiesen wird. In einem zukünftigen Artikel werden wir besprechen, wie Sie den Stack für neue, noch zu erstellende Tasks bestimmen können.
Zugegeben, das Beispiel war nicht besonders komplex. Aber Sie haben erkannt, dass Tasks komplizierte Anwendungen vereinfachen können, indem Sie das Problem in kleine Portionen aufteilen. So können beispielsweise bei einem MIDI-Controller Sende-, Empfangs- und Steuerungsaufgaben separiert werden. Der empfangende Task kann sich auf die Aufgabe konzentrieren, eingehende serielle Daten zu einzulesen und in Events zu dekodieren, die dann vom Steuerungstask ausgeführt werden. Einige Steuerungsevents können wiederum dazu führen, dass serielle Daten vom Sendetask gesendet werden. Dies ist eine vorbildliche Arbeitsteilung und erhöht nicht nur die Korrektheit des Codes. Auch die Wartung des Programms wird sehr erleichtert. ⏮

190182-02

Interaktiv

Korrekturen & Updates || Fragen & Antworten

Von **Clemens Valens**

Aktualisierungen und Ergänzungen zu Projekten, die in Elektor veröffentlicht wurden, ergänzt mit Tipps und Tricks, technischen Ratschlägen und Antworten auf Leserfragen.

Fragen und Antworten zur 9-Kanal-Relaisschaltplatine

Nach der Veröffentlichung dieses Projekts in der Ausgabe Mai/Juni 2019 kamen einige Fragen zu diesem vielseitigen Projekt auf. Hier sind sie, zusammen mit hoffentlich zufriedenstellenden Antworten.

F: Die Stückliste gibt für R3, R4 und R6 Werte von 0 Ω an, aber der Schaltplan 1 kΩ, 1 kΩ und 10 Ω. Was ist richtig?

A: Stückliste und Schaltplan sind in gewisser Art beide richtig. Diese Widerstände begrenzen den Strom aus und zu den Mikrocontroller-Portpins, für den Fall, dass ein unvorsichtiger User etwas an den Programmierverbinder K2 und/oder die serielle USB-Schnittstelle anschließt, ohne vorher die Versorgungsspannung abzuklemmen. Der genaue Wert dieser Widerstände ist unwichtig, solange er nicht zu hoch ist. Wenn das Breakout-Board permanent bestückt ist, kann man die Widerstände R3 und R4 auch durch eine Drahtbrücke ersetzen.

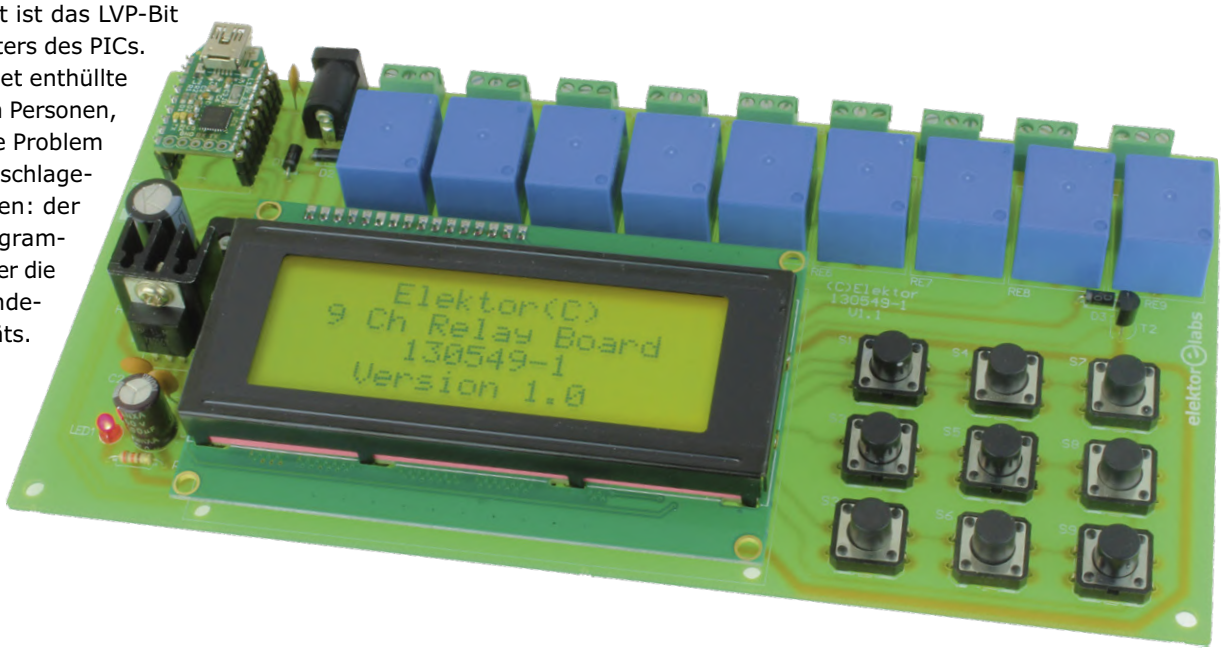
F: Beim Programmieren des PIC-Mikrocontrollers erhielt ich die Fehlermeldung „At address 300006 Expected Value 81 Received Value 85“. Der Austausch der MCU oder die Programmierung auf einem Breadboard führte zu dem gleichen Ergebnis. Ich benutze einen PICkit 3-Klon und die MPLAB IPE-Software, die in anderen Projekten gut funktioniert hat. Obwohl die Meldung ja besagt, dass die Programmierung gescheitert ist, scheint die Schaltung gut zu funktionieren. Was bedeutet diese Fehlermeldung?

A: Die Meldung bedeutet, dass das Programmiergerät das Bit 2 bei der MCU-Adresse 300006 nicht löschen kann. Das hat mit der 9-Kanal-Relaisschaltplatine überhaupt nichts zu tun. Das fragliche Bit ist das LVP-Bit des CONFIG4L-Registers des PICs. Eine Suche im Internet enthüllte mehrere Beiträge von Personen, die genau das gleiche Problem hatten. Zu den vorgeschlagenen Lösungen gehören: der Einsatz eines HV-Programmers, „bulk erase“ oder die Verwendung eines anderen Programmiergeräts.

F: Wenn das Board von der 12-V-Versorgung getrennt, aber an einem aktiven USB-Anschluss angeschlossen ist, blinkt die LED1 mit reduzierter Helligkeit. Ein Oszilloskop zeigt eine etwa dreieckige Wellenform auf der 5-V-Versorgungsleitung, die bei einer Frequenz von etwa 8,5 Hz von 1,4 V auf 1,9 V ansteigt. Verwendet man einen separaten USB-zu-Seriell-Wandler oder überbrückt man das USB-zu-Seriell-Modul, tritt der gleiche Effekt auf. D2 ist in Ordnung, und auch nach gründlichster Suche kann ich keinen Fehler in der Schaltung finden. Was ist hier los?

A: D2 soll verhindern, dass das USB-zu-Seriell-Modul die Schaltung mit Strom versorgt, da es für die Relais nicht genügend Power hat. Unter normalen Umständen ist es IC1, das das USB-zu-Seriell-Modul mit Strom versorgt, nicht umgekehrt. Was Sie beobachten, ist wahrscheinlich die Leckage durch die RxD/TxD-Leitungen vom USB-zu-Seriell-Modul zu den Portpins der MCU. Diese Portpins verfügen über interne Schutzdioden, die mit dem Versorgungsanschluss der MCU verbunden sind. Der Pin lädt C4 auf, so dass die LED1 aufleuchtet, wenn die Spannung hoch genug ist. Dies wiederum führt dazu, dass C4 leicht entladen wird, wodurch die LED1 weniger hell leuchtet und C4 wieder aufgeladen werden kann und so weiter. Das Ergebnis ist eine dreieckartige Spannung, die allerdings kein Problem darstellt, besonders wenn Sie 1 kΩ für R3 und R4 verwenden, um den Leckstrom auf harmlose Werte zu begrenzen.

www.elektor-labs.com/1778



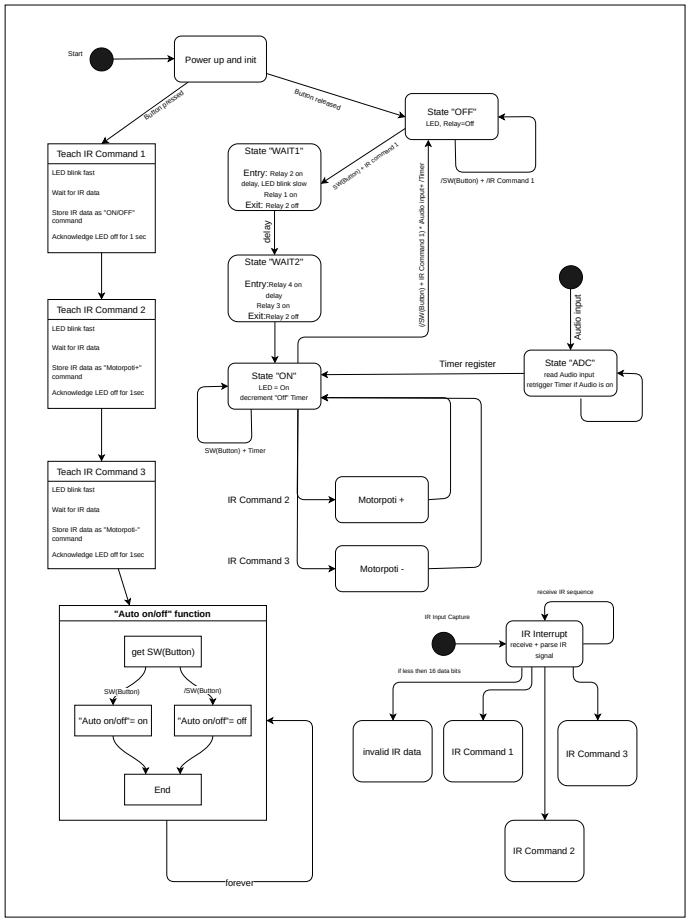
Statische IP-Adresse für den ESP32-Zeit-Server

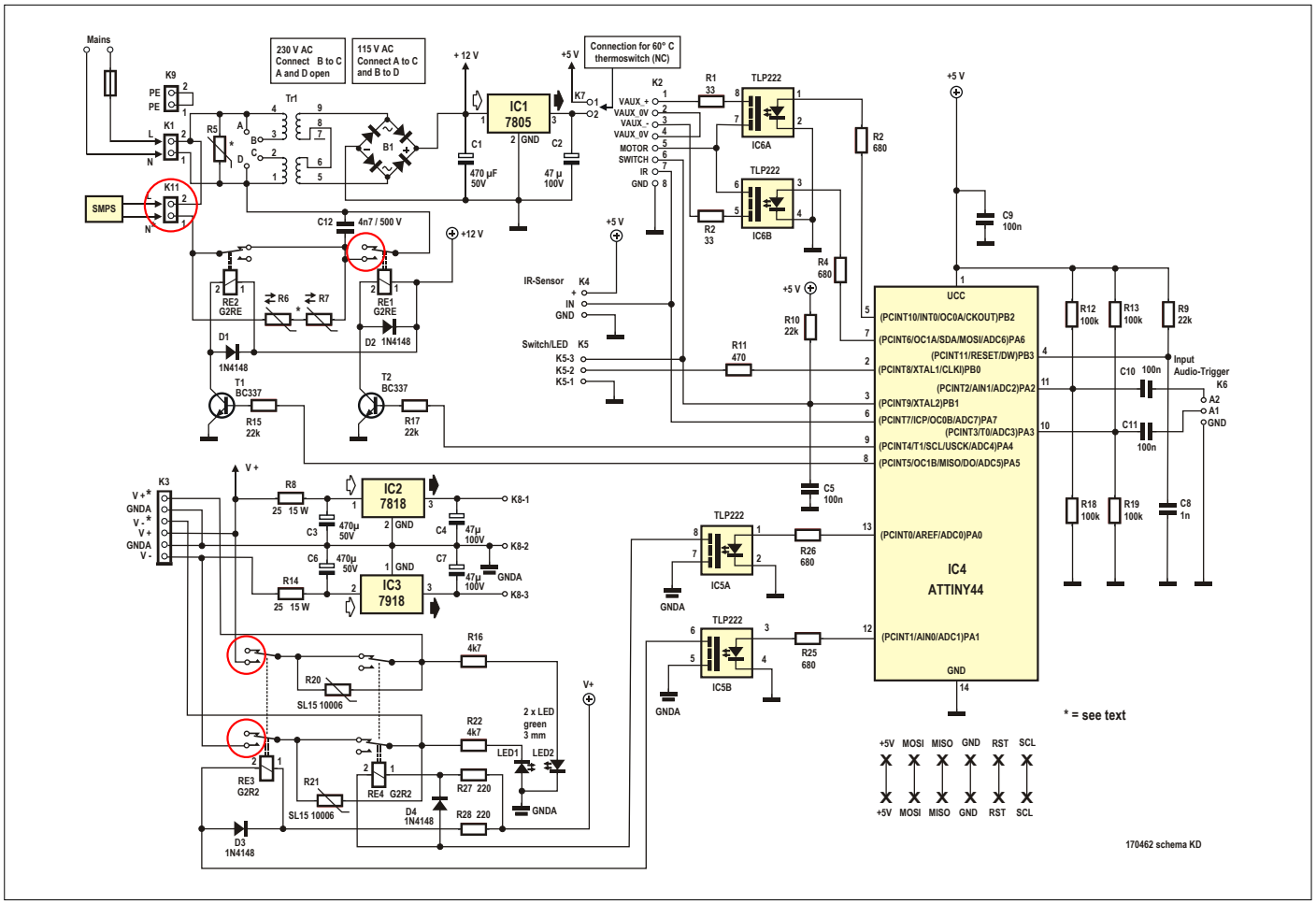
In der Ausgabe Juli/August 2019 haben wir einen kleinen NTP-Server (Network Time Protocol) vorgestellt, der die (echte oder angepasste) Zeit in einem Netzwerk bereitstellt. Um eine IP-Adresse zu erhalten, wurde das DHCP bemüht, was in vielen Fällen praktisch ist, aber nicht immer, wenn beispielweise im Netzwerk überhaupt kein DHCP-Server vorhanden ist. Aus diesem und anderen Gründen haben wir uns entschieden, der Software eine Option hinzuzufügen, dass auch eine statische IP-Adresse für den NTP-Server festgelegt werden kann. Dazu aktivieren Sie ganz einfach auf der Seite „IPv4 Settings“ die Checkbox „Use static IPv4“ und geben die gewünschte IP-Adresse in die Subnetz-Maske ein. Das war es!

Softstart für Audioverstärker

Nach der Veröffentlichung dieses Projekts in der Ausgabe November/Dezember 2019 wurden in zwei Zeichnungen einige Fehler entdeckt. Im Schaltplan (Bild 4) sind die Schließer (NO) und Öffner (NC) der Relais RE1 und RE3 vertauscht. Auch die Beschriftung und Verdrahtung der Klemme K11 war fehlerhaft. Hier ist jetzt der korrigierte Schaltplan abgebildet.

Bild 6 im veröffentlichten Artikel zeigt eine alte Version des Zustandsdiagramms. Hier ist nun die richtige Version zu sehen. Im IR-Lernmodus blinkt die LED, während die Software auf ein IR-Signal wartet. Der Empfang wird bestätigt, indem die LED für eine Sekunde dunkel bleibt. In den Formeln zur Berechnung von R_{NTC} und der Gesamtenergie ist das Ohmzeichen Ω Opfer des Druckers geworden. Die korrekten Berechnungen finden Sie in der Projektdokumentation unter www.elektor-labs.com/1343.





Autoscheinwerfer tunen

Legal, illegal, nicht egal!

Von Dr. Thomas Scherer

Besseres Licht steht auf der Wunschliste vieler Autofahrer, deren Fahrzeuge nicht schon vom Hersteller mit Xenon- oder LED-Lampen ausgerüstet sind. In diesen Fällen ist man auf Halogenlampen angewiesen, die gelber und weniger hell leuchten sowie in manchen Fällen nicht lange leben. Daran lässt sich etwas ändern - und das ganz legal.

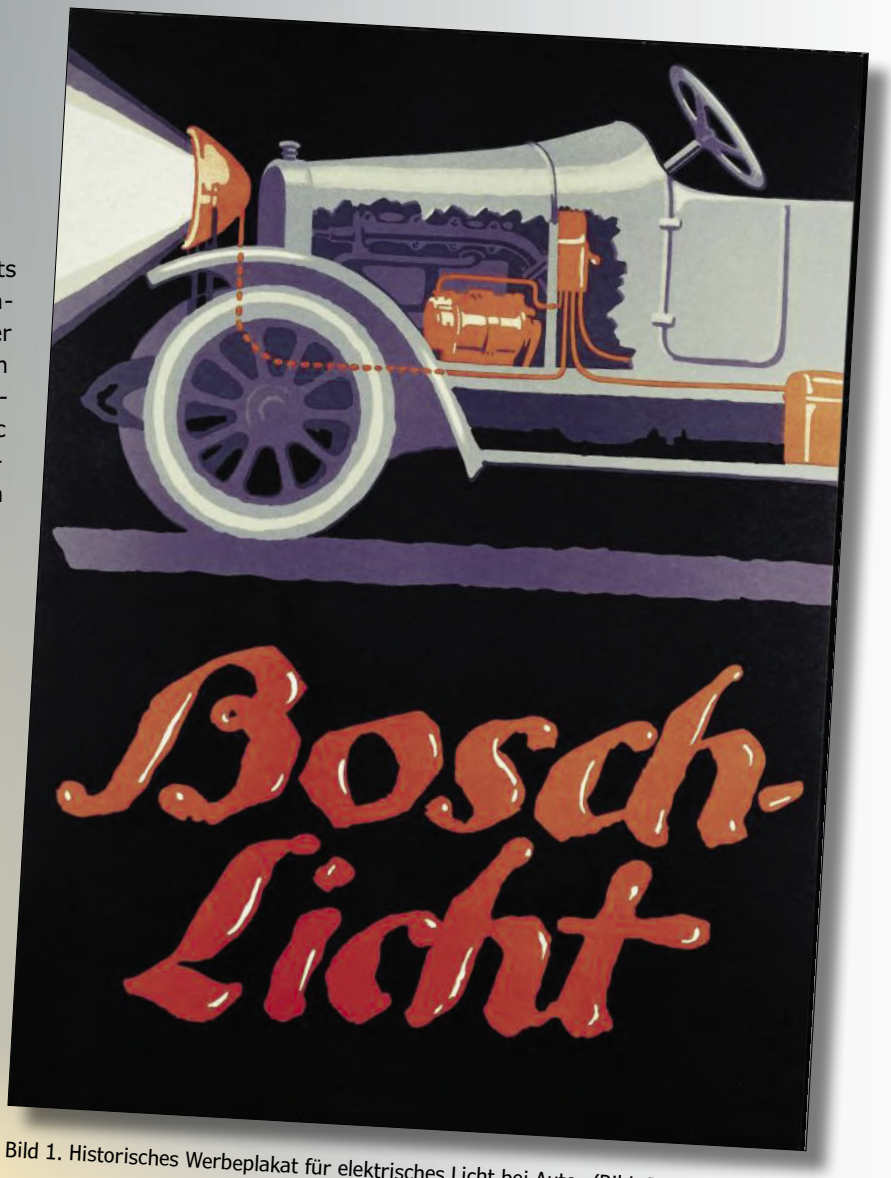


Bild 1. Historisches Werbeplakat für elektrisches Licht bei Autos (Bild: Bosch).

Die ersten Autos hatten Ende des 19. Jahrhunderts vier Räder, einen Motor, eine Lenkung und Bremsen. Die Beleuchtung unterschied sich nicht von der Konkurrenz mit Bio-Pferdestärken: Wie in Kutschen gab es Lampen mit Kerzen oder Karbid. Diese Funzeln wurden 1911 zum ersten Mal in einem Cadillac durch elektrisches Licht ersetzt. Schon 1913 perfektionierte Bosch die mobile Stromerzeugung durch den Einsatz von Batterie samt reglerbestücktem Generator (**Bild 1**). Sofort fühlten sich Zeitgenossen durch das hellere Licht geblendet – modern wird oft nicht gut gefunden, weil meist mit Neuerung verbunden!

Also verpasste man den Cadillacs ab 1917 eine Abblendmöglichkeit. Das fand man auch diesseits des großen Teichs toll und in typisch deutscher „Regelungsfreude“ wurde 1921 im Deutschen Reich das dauerhafte Abblenden des Lichts gesetzlich vorgeschrieben.

Doch das war nicht der Weisheit letzter Schluss: Auf deutschen Straßen wurden die Autos jetzt zwar gesehen, aber die Fahrer selbst konnten des Nachts nicht mehr gut sehen, weshalb sich nächtliche Unfälle häuften. Es musste also weiter getüftelt und ausprobiert werden. Getrennte Scheinwerfer für Fernlicht und Abblendlicht oder auch komplexere Systeme mit beweglichen Optiken wurden modern. In den USA wurde dann 1924 die Zweifadenlampe erfunden, mit dem beide Leuchtweiten mit einem Scheinwerfer möglich waren. Diese Technik wurde dann 1925 von Bosch mit den „Bilux“ genannten Lampen perfektioniert.

Und so blieb die Situation für Jahrzehnte unverändert, auch wenn 6-V-Bordnetze nach dem Zweiten Weltkrieg zugunsten von 12-V-Technik aus der Mode kamen, was die nötigen Ströme und die resultierenden Spannungsabfälle halbierte. Wirklich besser wurden Autoscheinwerfer erst mit der Einführung von helleren Halogenbirnen Ende der 1960er Jahre. Doch erst 1971 war mit dem MB 350SL ein Serienauto mit der Halogen-Zweifadenlampe des Typs H4 bestückt.

Bild 2. Asiatisches Xenon-Nachrüst-Kit (Bild: TXVSO).

Soviel zur Ur- und Frühgeschichte der Autoscheinwerfer. Heute haben alle modernen Autos mindestens Halogenlicht. Besseres wie Xenon-Lampen oder gar LED-Scheinwerfer gibt es bei Kleinwagen und Autos der Mittelklasse zur Zeit nur gegen satte, vierstellige Aufpreise.

Besseres Licht

Nachdem in den 1990er Jahren Limousinen der Ober- und Luxusklasse zunehmend mit Xenonlicht „protzten“, gab es zwei Reaktionen: Neben dem „Habenwollen-Effekt“ fühlten sich – Sie ahnen es bereits – andere, an gelbliches Halogenlicht gewöhnte Verkehrsteilnehmer häufig vom neuen Luxus-Licht geblendet. Fakt ist, dass Xenon-Lampen nicht nur heller sind, sondern auch „blauer“. Während die besten 55-W-Halogenlampen auf höchstens 1.500 lm kommen, kann das Gasentladungslicht von Xenonlampen bei nur 35 W mit bis zu 3.200 lm beeindrucken. Hinzu kommt die Farbtemperatur von 4.200 K gegenüber etwa 3.000 K bei Halogen. Das ist definitiv heller, und wer schon einmal ein Auto mit Xenon-Licht gefahren ist, der weiß, dass sich das beim Abblendlicht besonders positiv auf die Sicht auswirkt.

Bild 3. LED-Retrofit-Lösung mit aktiver Kühlung (Bild: Banggood).

Und was ist mit LEDs? Bis 2008 waren in der EU die technisch besseren und gegenüber Xenonlicht preiswerteren LEDs nur für die Rücklichter zugelassen. Dann aber gab es erste Autos mit echten LED-Frontscheinwerfern. Von den Daten her können sie gut mit Xenonlampen konkurrieren und sind prinzipbedingt viel zuverlässiger. An die Aufpreise für besseres Licht gewohnt, langt die Autoindustrie aber auch heute noch ordentlich hin, wenn die Wahl zwischen Halogen- und LED-Scheinwerfern geboten wird. Diese Kosten tragen genauso wie die „relative Neuheit“ dazu bei, dass Autos mit Halogenlampen immer noch die Mehrheit auf den Straßen bilden, auch wenn sich das stetig aber langsam ändert. Kein Wunder also, dass sich bei mit Halogenlicht geschlagenen, aber technisch begabten Auto-Piloten der Gedanke aufdrängt, dass man hier vielleicht etwas basteln könnte.

Spielregeln

Der Konjunktiv ist genau richtig. Es ist ja auch wirklich naheliegend, dass man schlicht und einfach die drögen Halogenbirnen der H1…H19-Klassen durch Xenon- oder LED-Nachrüstungen ersetzt, die mechanisch passen. Dazu muss man schließlich nicht einmal löten können! Die findige fernöstliche Industrie bietet passende Lösungen für fast alle Zwecke. **Bild 2** zeigt das Beispiel eines Xenon-Kits passend für H8/H9/H11-Sockel, das man samt Vorschaltelektronik für weniger als 30 € nicht nur bei den üblichen Verdächtigen wie Alibaba oder Banggood direkt aus China, sondern auch bequem bei Amazon und sonstigen Shops hierzulande bekommt. Und aus denselben Quellen sind auch Ersatzleuchten mit LED-Technik zu Preisen erhältlich, die für bessere Lösungen immer noch weit unter 100 € liegen und im Extremfall sogar „gute“ Halogenlampen unterbieten.
Doch darf man das? Leider nein!
Während die Regeln erstaunlicherweise in den USA und erwartbar in vielen asiatischen Ländern recht locker sind, werden wir dummen Europäer durch Helikopter-Politik vor allzu viel Freiheit geschützt. In der gesamten EU ist es strikt untersagt, für Halogenlampen gedachte und von den Zulassungsbehörden entsprechend „abgenommene“ Scheinwerfergehäuse mit andersartigen Leuchtmitteln zu bestücken. Punkt.
Tun Sie das doch, verlieren Sie die Zulassung für Ihr modifiziertes Auto und Versicherungen können bei Unfällen den Versicherungsschutz verweigern. Ein ziemlich hohes Risiko, finden Sie nicht?
Die juristische Lage hat dabei nichts mit Argumenten zu tun, denn für den Ersatz von Halogenlampen durch Xenon- oder LED-Leuchtmittel gibt es gute Gründe. Die (asiatische) Industrie stellt sogar mehr oder minder geeignete LED-Retrofit-Leuchtmittel her, also LED-Lampen passend für den direkten Ersatz von Glühlampen. Der ADAC hat den Einsatz solcher LED-Retrofits untersucht [1] und kam dabei zu durchaus positiven Ergebnissen. Aber das hilft alles nichts: Verboten bleibt verboten. Solange der Gesetzgeber sich nicht bemüht, klare Regeln und Prüfnormen für LED-Retrofit-Lampen zu erstellen und zu verabschieden, kümmern sich auch große und seriöse Konzerne wie Osram oder Philips nicht um diesen eigentlich interessanten Markt. Diese Situation schützt nicht nur die Profite der Autokonzerne, sondern führt zu einer illegalen Szene der

Licht-Aufrüster und Tuner, die so auf zweifelhafte Produkte asiatischer Fertigung angewiesen sind. Schaut man sich die Angebote in den diversen Webshops an, stellt man schnell fest, dass HTML geduldig ist und Angaben bezüglich Leistung, Lichtstrom und Farbtemperatur sowie Lebensdauer nicht sehr vertrauenswürdig sind.

Diese Retrofits „trotzdem“ zu verwenden ist also auch aus technischer Sicht keine gute Idee. LEDs sind bei ähnlicher Leistung nämlich flächenmäßig größer als die Glühwendeln von Halogenlampen. Die Ausleuchteigenschaften können damit insbesondere bei Scheinwerfern mit Parabolspiegel durchaus abweichen. Bei sogenannten „Projektionsscheinwerfern” ist das vom Prinzip her nicht ganz so tragisch, da hier eine per Blende begrenzte Leuchtfläche via Linse auf die Straße projiziert wird. Auch thermisch und elektrisch ergeben sich Probleme, denn gegenüber einem Scheinwerfer, der direkt vom Hersteller mit LEDs bestückt ist, führen LED-Retrofits oft zu sehr begrenzten Platzverhältnissen. Es braucht nicht nur eine Vorschaltelektronik (geringeres Problem), sondern irgendwie muss da die Wärme von LED-Chips nach außen abgeleitet werden, denn LEDs und Wärme stehen auf Kriegsfuß. Bei entsprechender Leistung reicht ein passiver Alu-Träger hier nicht aus, sondern es muss aktiv per Lüfter gekühlt werden, wie beim Beispiel in **Bild 3** zu sehen ist. Dieser Klotz mit Kühlrippen und Lüfter steht nicht nur hinten raus und passt aufgrund der beengten Bauweise moderner Autos nicht in jeden Scheinwerfer. Schlimmer noch: Wenn diese Mini-Lüfter verdrecken und blockieren, dann brennen die LEDs schnell durch und unter Umständen sind anschließend nicht nur die Leuchtmittel, sondern wegen des Niederschlags von Rauch und Dämpfen auch die Scheinwerfer selbst defekt.

(I)legale Alternativen

Die Wünsche der Autobastler-Szene wurden von der Industrie schnell erkannt. Seit dem häufigeren Auftreten von Xenon-Licht im Straßenverkehr kann man vermehrt Halogenlampen kaufen, bei denen blaueres Licht versprochen wird. Realisiert wird dies durch schlichte Einfärbung des Glaskörpers mit blauer Farbe. Wenn man **Bild 4** betrachtet, ist klar, dass das nicht nur nicht viel hilft, sondern die Lichtausbeute reduziert. Das Lichtspektrum von Halogenlampen ist eben sehr rotlastig. Die blaue Beschichtung dämpft lediglich das langwelligere Licht, bietet aber kein Mehr an blauem Licht. Das Resultat sieht tatsächlich etwas blauer aus, ist aber nicht heller, sondern dunkler. Wer so etwas kauft, zeigt damit nur, dass er von Physik viel Nichtahnung hat.

Und bevor ich es vergesse: Es gibt auch 100-W-Versionen üblicher Halogenleuchtmittel. Da die aber weit über der zulässigen Leistung (zwischen 55 und 65 W) liegen, ist das nicht nur illegal, sondern auch riskant. Die Kunststoffgehäuse der Scheinwerfer sind für solche Leistungen nicht gebaut und können durch die höhere Wärmeabgabe beschädigt werden.

Möglich wäre auch der Austausch der kompletten Halogenscheinwerfer durch echte Xenon- oder LED-Scheinwerfer. Leider gibt es sowas kaum im Zubehörmarkt, wo man preiswert nachgebaute Ersatzteile von Drittherstellern bekommt. Die Originalteile aber sind in der Regel schweineteuer. Neben dem Preis gibt es auch noch eine weitere Hürde: Der Gesetzgeber schreibt bei Lichtströmen > 2.000 lm das Vorhandensein einer Scheinwerferreinigungsanlage vor. Und nicht nur das: Auch eine automatische Leuchtweitenregulierung ist zwingend erforderlich. Beides zusätzlich nachzurüsten ist nicht nur sehr aufwändig, sondern auch ökonomisch unsinnig. Eine Ausnahme gibt es: Vom VW-Konzern werden neuerdings Xenonlampen mit nur 25 W und einem Lichtstrom von nicht mehr als 2.000 lm verbaut, was die Scheinwerferreinigungsanlage einspart. Eine Leuchtweitenregulierung ist aber nach wie vor integriert. Wen es interessiert: Osram baut diese Lampen unter der Bezeichnung D8.

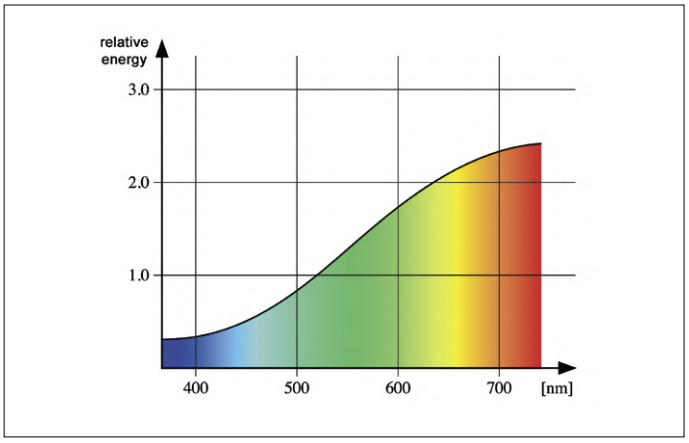


Bild 4. Form des Lichtspektrums von Halogenlampen. Da ist nicht viel Blau (Bild: Jean-Jacques MILAN, GNU CCL 1.2, CCASA 3.0 [5]).

Wer auf dem Teppich bleiben und trotzdem etwas helleres Licht haben will, dem bleibt nur die Möglichkeit, spezielle Halogenlampen zu kaufen, die von Markenherstellern für absolut wenig aber relativ viel Geld verkauft werden. Von Philips gibt es für knapp unter 20 € pro Pärchen neben unsinnig blau lackierten Lampen auch die Typen „X-tremeVison“ mit 130 % und „RacingVision“ mit 150 % der Lichtausbeute normaler Halogenlampen. Angeblich haben sie eine leicht erhöhte Farbtemperatur von 3.500 K, die aber durch blassblauen Lack erkauft wird, und interessanterweise beide einen Lichtstrom um 1.500 lm. Doch Prospektangaben [2][3] sollte man nicht ganz wörtlich nehmen, denn erstens werden diese Lampen mit 130 und 150 % „mehr“ Licht beworben, aber es sind nur 30 und 50 % „mehr“ und zweitens kommen diese Unterschiede zustande, obwohl die schwächere Lampe mit 1.550 lm und die stärkere mit 1.500 lm angegeben ist. Dass Erstere maximal 350 h und Letztere nur 200 h lebt (im Gegensatz zu den 1.000 h einer Standard-Lampe) ist hingegen glaubhaft und nicht paradox.

Auch der andere große Lampenproduzent Osram lässt sich nicht lumpen und bietet für knapp über 20 € leistungsgesteigerte Halogenlampen des Typs „Night Breaker“ [4] (**Bild 5**) an, die angeblich „bis zu 150 % heller im Vergleich zu den gesetzlichen Mindestanforderungen“ leuchten. Worauf auch immer sich das bezieht: Der Lichtstrom beträgt hier erstaunlich moderate 1.350 lm und die Lampe soll bis zu 250 h leben. Dieses Exemplar habe ich mir gekauft.

Physik des Lampen-Tunings

Liest man die Bewertungen bei Amazon etc., so sieht man bei den Verwendern dieser Hochleistungs-Halogenlampen viel Frust. Der Löwenanteil der Enttäuschung geht auf das Konto „Haltbarkeit“. Je nach Bordspannung sind die teureren Halogenlampen ruck-zuck futsch. Kein Wunder, denn die Hersteller bauen sie für eine Spannung an den Klemmen von etwa

Bild 5. OSRAMs Night Breaker versprechen 150 % mehr Licht.

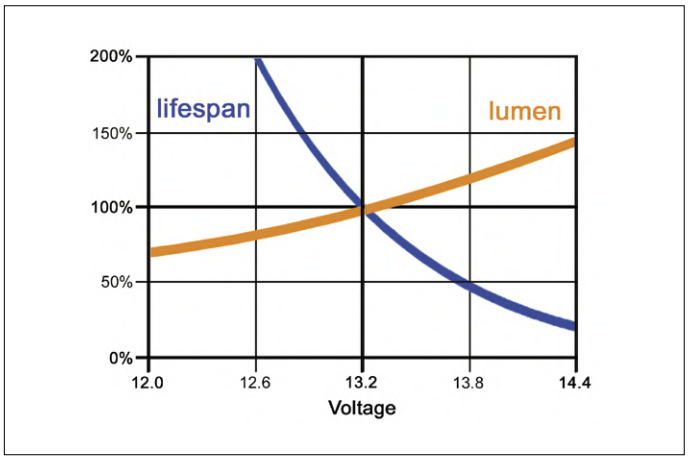


Bild 6. Der Einfluss der Betriebsspannung auf Lebensdauer und Lichtausbeute einer Halogenlampe.

13,2 V. Die Lichtmaschinen in den meisten Autos sind auf die Ladeschlussspannung von Blei-Akkus abgestimmt und liefern etwa 13,8…14,2 V. Je nach der Höhe dieser Spannung und dem Spannungsabfall über Zuleitungen und Relaiskontakte können an den Lampen also auch deutlich unter oder über 13,2 V anliegen. Das Diagramm von **Bild 6** macht klar, was das bedeutet: Nur rund 5 % mehr Spannung halbiert die Lebensdauer! Dafür wird die Lampe rund 20 % heller.

Wirklich heller? Nein, denn diese 20 % mehr Lichtstrom sind an der Grenze der Wahrnehmung, da der Helligkeitseindruck dem Lichtstrom leider nur logarithmisch folgt. Für die doppelte wahrgenommene Helligkeit würde man unter diesen Sichtbedingungen mehr als den vierfachen Lichtstrom benötigen. Die zweite Enttäuschung ist also der Nutzen dieser Lampen, denn man nimmt eine um 50 % gesteigerte Helligkeit leider nur als „etwas heller“ wahr. Positiver macht sich da schon die tatsächlich wahrnehmbare Verschiebung der Farbtemperatur in die blaue Richtung bemerkbar. 500 K Differenz zum normalen Halogenlicht kann man sehen.

Was man auch sehen kann: Diese tollen Hochleistungs-Halogenlampen wurden faktisch so konzipiert, dass die Glühwendel heißer und dadurch heller und blauer wird, aber dafür kürzer lebt. Im Prinzip handelt es sich also schlicht um eine Lampe, die mit leichter Überspannung betrieben wird – mehr nicht. Dafür ist dann auch noch ein Aufpreis fällig, denn schließlich muss das reißerische Marketing ja auch bezahlt werden. Bleibt die Frage nach dem Sinn des Ganzen. Meiner Ansicht nach kann man so etwas machen. Der Aufpreis hält sich in Grenzen und alle anderen Tuning-Maßnahmen sind entweder verboten oder echt teuer.

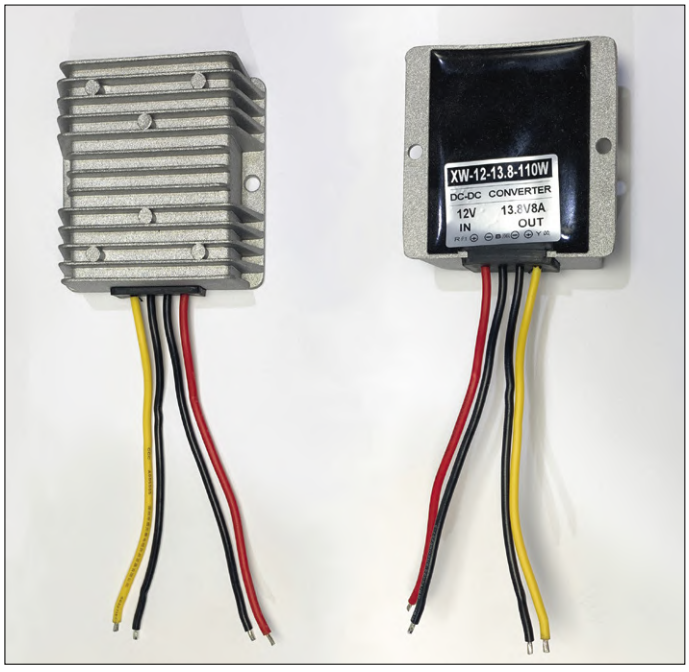


Bild 7. DC/DC-Konverter: 9…14 V in und 13,8 V out, wetterfest in stabilem Alugehäuse vergossen.

Stabiles Licht

Wie schon erwähnt, klagen viele Nutzer dieser Halogenlampen über häufigen Birnenwechsel. Bei mir allerdings hielt ein Satz nun immerhin schon drei Jahre und gut 50.000 km. Dafür empfand ich sie auch nicht wesentlich heller als normale Halogenbirnen, was mich aufgrund der Psychophysik des Sehens nicht sehr wunderte. Trotzdem schöpfte ich Verdacht, denn irgendwas musste doch mit meinem Auto nicht stimmen, oder etwa nicht?

Des Rätsels Lösung: Ich fahre einen Prius III, und der hat gar keine Lichtmaschine. Der für Beleuchtung, Servolenkung und Bordcomputer nötige 12-V-Blei-Akku wird einfach per Elektronik sehr schonend aus dem Hochvolt-Akku geladen. Der Blei-Akku wird (gemessen) mit nicht mehr als maximal 20 A belastet und auf knapp 13 V gehalten. Außerdem sitzt er hinten im Kofferraum. An den Lampen vorne kamen (gemessen) gerade einmal 11,8 V an. Laut Bild 6 kann ich daher mit rund 400 % Lebensdauer rechnen. Das wäre geklärt. Die lange Lebensdauer bezahle ich aber mit 30 % weniger Lumen. Und da beim Prius ein Projektionsscheinwerfer fürs Abblendlicht verbaut ist, brauche ich mich über das echt miese Licht nicht wundern.

Das hätte ich auch anders haben können, aber für LED-Licht hätte ich eine höhere Ausstattungslinie wählen müssen, was neben dem Aufpreis einen um 0,1 l pro 100 km höheren Verbrauch bedeutet hätte. Beim Kauf dachte ich mir „Wenn schon umweltfreundlich, dann richtig“. Und jetzt hatte ich die Bescherung.

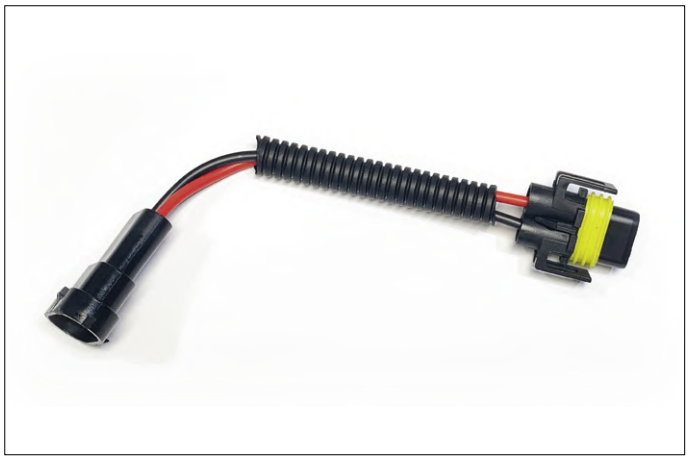


Bild 8. Verlängerungskabel für Halogenlampen. Damit wurden die Konverter mit passenden Buchsen und Steckern zum Einschleifen versehen.

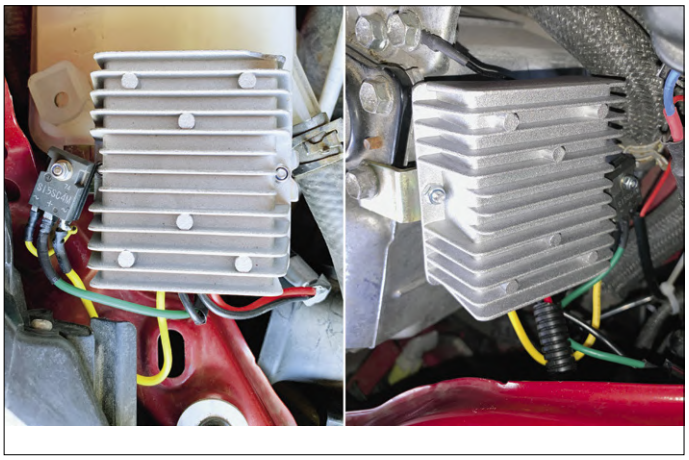


Bild 9. Links und rechts je ein Konverter mit Power-Schottky-Diode in Serie mit dem Ausgang verbaut.

Lässt sich da etwas machen? Wie wäre es mit einem leistungsstarken DC/DC-Konverter von 12 V auf etwa 13,2 V? Kaum erschien in mir dieser Gedanke, war auch schon eBay geöffnet. Tatsächlich: Dort gab es passende Boost-Konverter mit einer Belastbarkeit von 8 A – etwas Reserve ist ja nicht verkehrt. Das Ganze für nur 12 € pro Stück in schönem Alugehäuse vergossen. Leider boten sie nicht einstellbare 13,8 V am Ausgang. Aber dafür würde mir schon etwas einfallen.

Also bestellt und drei Wochen später hatte ich das, was man in **Bild 7** sehen kann, in der Hand. Tests zeigten, dass dieser Konverter passt, denn er liefert auch bei wechselnden Lasten bei Eingangsspannungen von 9…14 V stabile 13,8 V am Ausgang. Mitbestellt hatte ich aus anderer Quelle zwei H11-Verlängerungskabel (**Bild 8**) für je 2 €, die ich in der Mitte auseinanderschneiden und an die Konverter löten wollte. Der Betrieb mit 13,8 V hätte die Lebensdauer halbiert. Also beschloss ich, etwas Spannung mit einer Schottky-Diode zu verbraten. 15-A-Typen aus alten PC-Netzteilen in Serie mit dem Ausgang bringen bei dieser Last theoretisch einen Spannungsabfall von 0,4 V. Meine Messungen bestätigten das: An der angeschlossenen Halogenlampe lagen 13,4 V. Damit konnte ich leben.

Also baute ich die beiden modifizierten DC/DC-Konverter mit Hilfe von Metallbügeln in der Nähe der Scheinwerfer ein. Wie **Bild 9** zeigt, waren die Platzverhältnisse links etwas beengt. Und wenn Sie **Bild 10** betrachten, dann sehen Sie das leicht blauere Licht des äußeren Abblendlichts im Vergleich mit dem Fernlicht. Oben drüber ist übrigens das echt gelbliche 5-W-Standlicht, ganz konventionell und nicht einmal in Halogen. Nach einem Monat Betrieb und über 1.000 Nachtkilometern habe ich keine Beschwerden. Bringt es was? So rein subjektiv: immerhin ein bisschen was.

Übrigens kann so ein Mod auch für Nicht-Prius-Fahrer interessant sein, denn schließlich wird damit das Licht etwas schonender eingeschaltet und ohne Spannungsschwankungen betrieben. Das kommt der Lebensdauer der Lampen zugute. Und da zugelassene Leuchtmittel und unmodifizierte Scheinwerfer verwendet werden, ist diese Verbesserung nach bestem Wissen und Gewissen legal. Sonst darf man ja an der Außenbeleuchtung eines Autos nichts verändern. Aber bei der Innenbeleuchtung Ihres Autos können Sie sich austoben: Hier sind Retrofit-LED-Lampen erlaubt. Zumindest in meinem Auto findet sich innen keine einzige Glühlampe mehr. ⏮

191016-01

Bild 10. Tataaa: Mein Prius mit getuntem Abblendlicht. Man sieht die etwas weniger gelbliche Lichtfarbe des Abblendlichts, das jetzt mit 13,4 V (Fernlicht mit nur 11,8 V) betrieben wird.

Weblinks

[1] ADAC-Test: www.adac.de/infotestrat/technik-und-zubehoer/licht/lichttechniken/

[2] X-tremeVision: www.philips.de/c-p/37166628/x-tremevision-fahrzeugscheinwerferlampe

[3] RacingVision: www.philips.de/c-p/12972RVS2/racingvision-fahrzeugscheinwerferlampe

[4] Night Breaker: https://bit.ly/2nm5E5g

[5] CCASA 3.0: https://creativecommons.org/licenses/by-sa/3.0/deed.en

REVIEW

Digitale Lötstation von Toolcraft

Von **Luc Lemmens** (Elektor-Labor)

Wenn ich mich für eine Lötstation entscheiden muss, bevorzuge ich auf jeden Fall die Top-Marken. Das heißt jetzt nicht, dass ich ein Snob bin: Im Allgemeinen möchte ich nur Werkzeuge kaufen, von denen ich jahrelang etwas habe. Qualität und Zuverlässigkeit sind gewichtige Argumente, aber auch die langfristige Verfügbarkeit von Zubehör oder Ersatzteilen spielt für mich eine wichtige Rolle.

Bei allem Respekt muss ich zugeben, dass ich Voltcraft und Toolcraft - die Eigenmarken von Conrad - nicht für Top-Marken halte. Doch die Lötstation ST-100D sieht interessant genug aus, um sie auf die Probe zu stellen. Und um so herauszufinden, was man bekommt, wenn man sich für das mittlere Preissegment entscheidet.

Der erste Eindruck

Man hat nur eine Chance für einen ersten Eindruck. Ich kann nur sagen, dass die ST-100D in dieser Hinsicht gut abschneidet: ein schön verarbeitetes, robustes Metallgehäuse für die Lötstation und ein sauberer Ständer für den Lötkolben. Ich habe mich nicht tiefgehend mit der Geschichte dieses Sets beschäftigt, aber im Internet habe ich gesehen, dass es mindestens einen Vorgänger mit der gleichen Typenbezeichnung gibt, was darauf hindeutet, dass sich dieses Modell über die Jahre bewährt und so einen neuen Look verdient hat.
Ich habe keine Ahnung, ob nur das Äußere ein (erfolgreiches) Facelifting erhalten hat oder ob mehr verändert wurde, aber das ist auch für eine Beurteilung der aktuellen Version nicht wirklich wichtig. Es ist jedoch beruhigend zu wissen, dass es sich anscheinend nicht um eine Eintagsfliege handelt, was ausreichend Vertrauen in die zukünftige Verfügbarkeit eines Ersatzlötkolbens und von Ersatzteilen aufbaut.

Die Lötstation

Über das schwere Metallgehäuse ist wenig zu sagen. Es steht bombenfest auf dem Tisch und bewegt sich nicht, wenn man die Tasten bedient, genauso, wie es sein sollte. Ein kleines, aber übersichtliches LC-Display zeigt Soll- und Ist-Temperatur

der Lötspitze an. Laut Betriebsanleitung deutet ein Balkendiagramm auf der linken Seite des Displays die Wärmeleistung der Station an, es scheint jedoch eher so zu sein, dass sie die Differenz von Ist- und Solltemperatur anzeigt.
Die Temperatur wird auf der Vorderseite mit einem Drehknopf eingestellt, was besser funktioniert als mit den Auf- und Ab-Tasten, die man heutzutage an den meisten Lötstationen findet. Darüber hinaus gibt es Tasten für drei Präferenztemperaturen, die sich so „auf Knopfdruck“ einstellen lassen, was auch recht praktisch ist.
Natürlich gibt es auch einen Ein-/Ausschalter auf der Vorderseite, den Anschluss für den Lötkolben selbst und eine 4-mm-Buchse für den Potentialausgleich. Die Station verfügt leider über keinen Timer, der die Temperatur automatisch herunterregelt, wenn der Lötkolben über einen längeren Zeitraum nicht benutzt wird. Wer lange etwas von den Lötspitzen haben will und Energie sparen möchte, sollte also deshalb nie vergessen, die Lötstation rechtzeitig abzuschalten.
Die einzige „versteckte“ Funktion in diesem Gerät ist die Temperaturkalibrierung, die gemäß der Anleitung immer dann durchgeführt werden muss, wenn ein anderer Lötkolben angeschlossen wird. Allerdings hat nicht jeder das dafür notwendige Thermometer, das solch hohe Temperaturen (bis 450 °C) messen kann. Die Lötstation ist bereits werkseitig auf den mitgelieferten Lötkolben eingestellt. Die Temperatur kann leicht abweichen, wenn man eine andere Lötspitze einsetzt. In diesem Fall muss ein Offset für die Anzeige eingestellt werden, aber dann benötigen Sie natürlich wiederum ein geeignetes Thermometer.

Bild 1. Die Front ist übersichtlich, der Drehknopf für die Temperatureinstellung funktioniert super.

Der Ständer

Von der Lötkolbenablage bin ich ehrlich gesagt etwas enttäuscht. Ich mag es, wenn ein Ständer bombenfest auf dem Tisch steht und sich nicht bewegt, wenn ich den Lötkolben ablege oder herausnehme. In dieser Hinsicht kann die Lötstation keine Pluspunkte sammeln. Etwas mehr Masse für den Ständer hätte sicher nicht geschadet. Ein Schwamm zur Nassreinigung und Messingwolle mit passender Abdeckplatte zur trockenen Entfernung von Löt- und Harzrückständen sind im Lieferumfang enthalten. Der Ständer bietet jedoch nur Platz für jeweils eines von beiden. Ich ziehe es vor, während des Lötens den nassen Schwamm zur Reinigung der Spitze zu verwenden, und benutze die trockene Drahtwolle nur für eine gründlichere Reinigung. Der Wechsel ist bei diesem Ständer etwas umständlich.

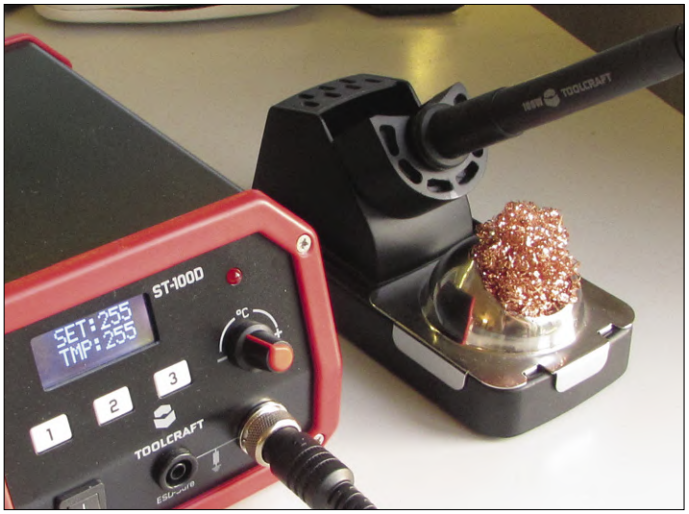


Bild 2. Der Ständer hätte etwas schwerer sein können, er bietet auch nur Platz für einen nassen oder einen trockenen „Schwamm“.

Spitzen

In der Anleitung steht nichts darüber, welche Spitzen für den Lötkolben geeignet sind. Aber wenn Sie den ST-100D im Webshop von Conrad suchen, finden Sie unter „Zubehör“ eine Reihe von Spitzen mit einem Durchmesser von 1,2 mm bis 3,2 mm, Meißel-, Bleistift- sowie abgeschrägte Modelle. Ein interessantes Detail ist, dass eine der Lötspitzen das Logo und den Namen der Firma Hakko sowie die Typenbezeichnung 900M-T-B zeigt. Diese Lötspitzenserie ist viel umfangreicher und zu sehr günstigen Preisen „überall im Internet“ zu bekommen. Was die Lötspitzen betrifft, hat Conrad hier eine ausgezeichnete Wahl getroffen. Die günstigeren Lötspitzen aus dem Internet sind höchstwahrscheinlich Kopien der originalen Hakko-Modelle und garantiert von zweifelhafter Qualität.
Hakko selbst liefert diese Lötspitzen übrigens nur noch als Ersatzteil, verwendet sie aber nicht mehr für neuere Lötkolben. Nachfragen haben ergeben, dass Hakko jetzt die verbes-

Bild 3. Eine feine bleistiftförmige Spitze (1,2 mm) wird standardmäßig mitgeliefert.

serte T18-Serie einsetzt, die ohne weiteres als Ersatz für die 900M-Serie verwendet werden kann.
Unser Toolcraft ST-100D wurde mit einer bleistiftförmigen Spitze mit 1,2 mm Durchmesser geliefert, was sehr praktisch für die feinere (SMD-)Arbeit ist. Ich hätte mir zum Beispiel auch eine etwas größere meißelförmige Spitze von 2,4 mm gewünscht, die universell einsetzbar ist und sich daher auch für schwerere Lötarbeiten eignet. Zum Glück sind die Lötspitzen nicht übermäßig teuer und es schadet sowieso nicht, verschiedene Größen oder Modelle zur Hand zu haben. Also, ich würde sofort eine oder mehrere zusätzliche Spitzen dazubestellen!

Und wie arbeitet es sich damit?

Im Endeffekt ist dies die wichtigste Frage, wenn man über Werkzeuge spricht. Das Kabel zwischen Lötkolben und Station ist ziemlich, aber nicht zu dick, es ist lang genug und schon gar nicht hinderlich steif oder schwer beim Löten. Selbst bei höchster Temperatureinstellung wird der Griff nicht wirklich heiß. Es ist und bleibt dennoch Geschmackssache, ob Sie einen Lötkolben mögen oder nicht. Ich finde dieses Toolcraft-Modell gut und angenehm in der Handhabung, während ein Kollege die Spitze zu lang findet. Der Abstand von der Hand zur Lötstelle ist ihm zu groß, um fein löten zu können.
Die Lötstation hält den Lötkolben auf der richtigen Temperatur. Die mitgelieferte Spitze ist nicht wirklich für größere Lötverbindungen geeignet, aber mit etwas Geschick (den Lötkolben flach halten) können Sie auch mit dieser dünnen Spitze ganz einfach größere Pads auf die richtige Temperatur bringen.
Alles in allem finde ich, dass sie mit der Toolcraft ST-100D für diesen Preis eine prima Lötstation bekommen, obwohl ich - wie erwähnt - einige Punkte sehe, die bei diesem Set verbessert werden könnten. Aber ... gäbe es die perfekte Lötstation, wären nicht so viele Marken und Modelle auf dem Markt. ⏮

190385-02

Arduino-Pro-IDE

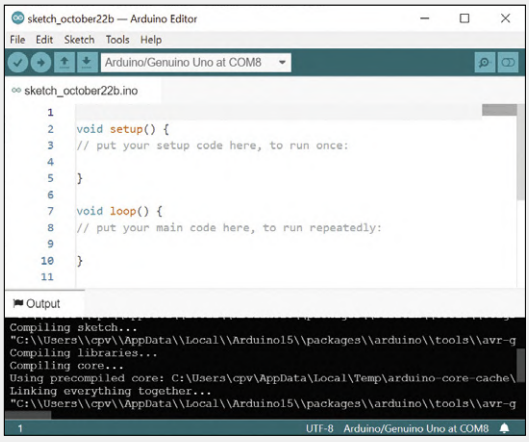
Erste Eindrücke

Von **Clemens Valens**

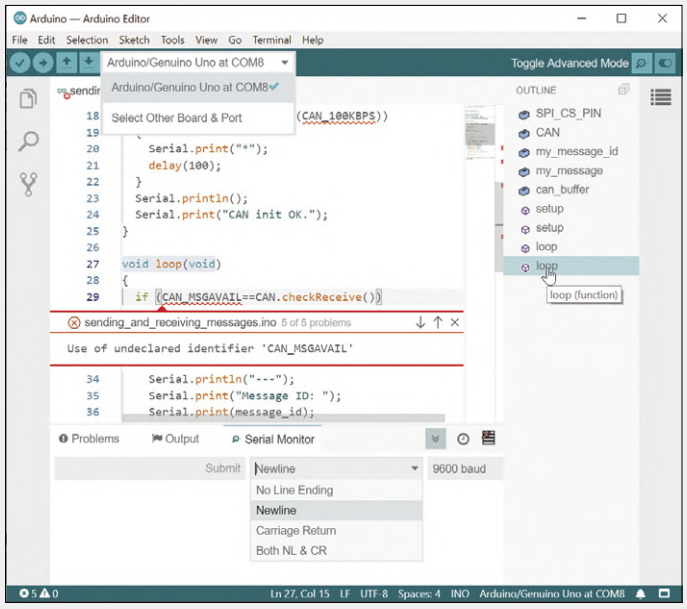
Die auf Eclipse Theia basierende Arduino-Pro-IDE wurde im vergangenen Oktober auf der Maker Faire in Rom vorgestellt. Die klassische Arduino-Kommandozeile (CLI) bietet nach wie vor alle wichtigen Arduino-Funktionen, läuft aber nun im Daemon-Modus. Die Arduino-Pro-IDE ist für Windows, Mac OS X und Linux 64 konzipiert, war aber, als diese Zeilen geschrieben wurden, erst in der (englischsprachigen) Version *0.0.2-alpha-preview* verfügbar, so dass potentielle User besser ein paar Releases abwarten sollten, bevor sie die neue IDE übernehmen. Auch wer die IDE selbst aus dem Quellcode kompilieren möchte, sollte sich etwas gedulden.

Wichtigste Eigenschaften

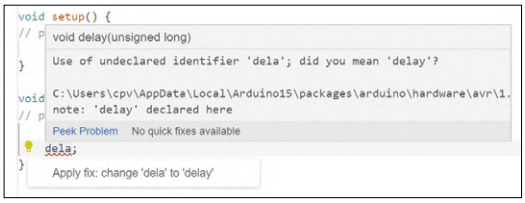
- Dualer Modus: klassisch und fortgeschritten
- Neuer Boardmanager
- Neuer Bibliotheksmanager
- Board-Liste
- Einfache Auto-Vervollständigung
- Git-Integration
- Ansprechenderer Serial Monitor



Im Klassik-Modus sieht die Arduino-Pro-IDE tatsächlich wie die klassische IDE aus.

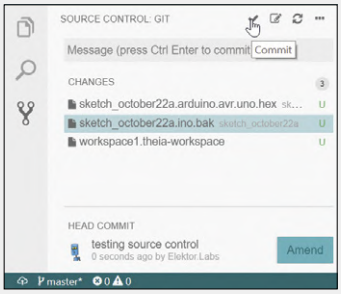


Einige Funktionen der neuen IDE: Board- und Port-Auswahl, Sketchdarstellung, Konfiguration des Serial Monitor und Fehlermeldungen.



Geht die Kompilation schief, hilft der Editor mit.

Die Auto-Vervollständigung erfordert eine gewisse Gewöhnung, bevor man Tippfehler vermeidet und wirklich Zeit spart.



Git-Integration eingebaut, aber wie nutzt man sie?

191146-03

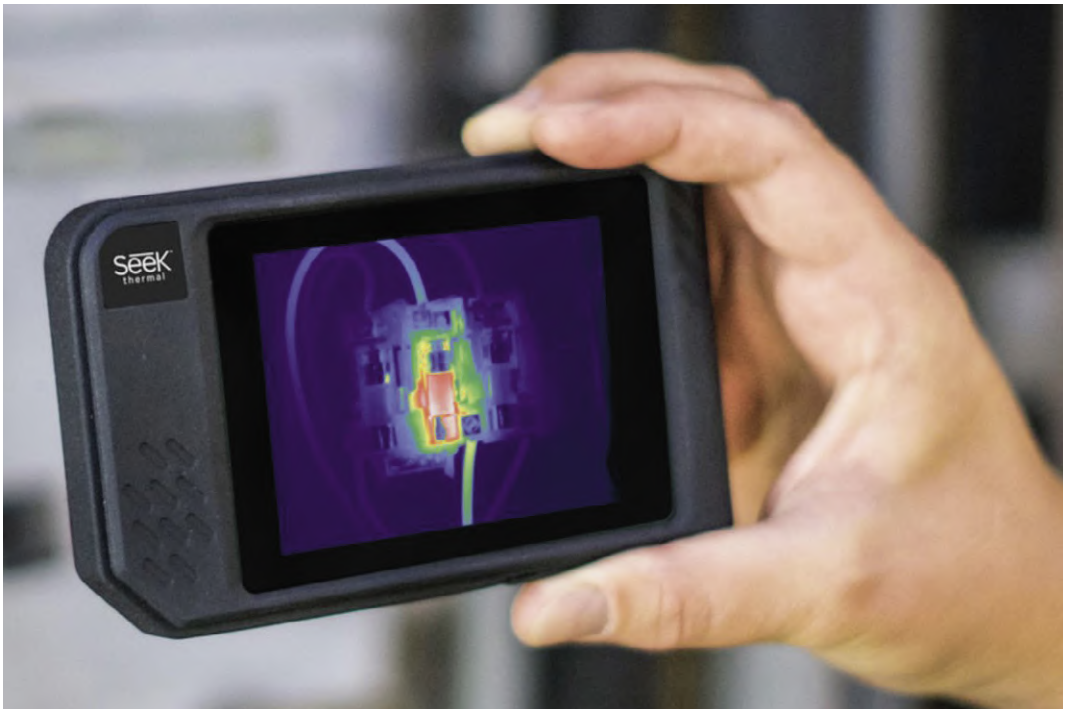
Weblink

[1] Video: Erste Eindrücke der Arduino-Pro-IDE: https://youtu.be/AfvCBJ8By-0

Zwei Wärmebildkameras im Vergleich

Von **Harry Baggen**

Infrarotkameras sind sehr nützlich, um Schwachstellen und Lecks in Isolierungen von Räumen, Verstopfungen in Ableitungen und Hotspots in elektrischen Systemen zu erkennen. Aber auch für einen Elektriker bietet eine solche IR-Kamera die Möglichkeit, Ursachen von Wärmeproblemen in Schaltungen auf die Schliche zu kommen.



In diesem Review vergleichen wir zwei IR-Kameras des amerikanischen Herstellers Seek, der sich von anderen Marken vor allem dadurch unterscheidet, dass die Sensoren mit einer viel höheren Auflösung arbeiten als bei anderen Geräten der gleichen Preisklasse.
Wahrscheinlich müssen wir Ihnen nicht mehr erzählen, wie eine Wärmebildkamera arbeitet. Im Gegensatz zu einer normalen Kamera ist sie für (vom Menschen) sichtbares Licht nicht empfindlich, sondern nur für Infrarotstrahlung. Und diese Strahlung ist ein Maß für die Temperatur des Messobjekts. Die verschiedenen Temperaturbereiche des Messobjekts werden in der Darstellung üblicherweise durch verschiedene Farben dargestellt.

Groß und klein

Wir haben hier zwei Produkte der Firma Seek in sehr unterschiedlichen Preissegmenten unter die Lupe genommen. Auf der einen Seite haben wir die semiprofessionelle Kamera **Seek ShotPro** mit einem Preis von 850 €, auf der anderen Seite die **Seek Compact**, ein Plug-in-Modul für Smartphones mit einem Preis von circa 300 €, das in einer Android- und einer iPhone-Version erhältlich ist. Beide Kameramodelle haben eines gemeinsam: Sie liefern sehr detaillierte Bilder, auf denen die Wärmeverteilung genau sichtbar ist. Zum Vergleich: Eine IR-Kamera in der 1000-€-Preisklasse besitzt gewöhnlicherweise einen Sensor mit einer Auflösung von maximal 160x120 Pixel, während die Seek ShotPro mit 320x240 Pixel glänzen kann. Und selbst die relativ billige Seek Compact verfügt bereits über einen Sensor mit 206x156 Pixel.

Seek ShotPro

Die ShotPro ist eine Standalone-IR-Kamera in der Größe eines Smartphones. Das Gehäuse ist durchgängig mit einer rutschfesten Gummibeschichtung versehen und sieht so aus, als könnte es ein paar Stöße und Schläge schadlos überstehen. An der Vorderseite befindet sich die Linse in einer zusätzlichen Schutzkappe, auf der anderen Seite ein 3,5“ großer Touchscreen. Die Kamera lässt sich drahtlos über WLAN mit einem PC verbinden.
Wenn Sie das Bild auf dem Touchscreen berühren, erscheinen oben und unten Leisten für verschiedene Einstellungen. Die obere Leiste führt Sie zu den „bevorzugten“ Einstellungen, unter anderem die der Wahl der Temperatureinheit, der Uhrzeitdarstellung und des Emissionswertes. Die untere Leiste ermöglicht den Zugriff auf die Bildeinstellungen. Hier stehen verschiedene Farbmuster der Temperaturanzeige zur Verfügung, welche Spot-Temperaturen auf dem Bildschirm angezeigt werden und wie man auf die gespeicherten Screenshots zugreifen kann.
Ein sehr wichtiges Symbol auf den Leisten ist das „Auge“, mit man das Wärmebild auf das Bild der ebenfalls integrierten „normalen“ Kamera (mit frei wählbarem Überlagerungsgrad) überlagern kann. Das Ergebnis ist ein Mischbild, das viel klarer

Bild 1. Die Seek ShotPro (Foto: Seek Thermal).

Bild 2. Die Seek Compact (Foto: Seek Thermal).

zeigt, was und wo Sie messen. IR- und sichtbares Bild lassen sich exakt gegeneinander verschieben, damit sie so genau wie möglich übereinander fallen. Dieses Feature lässt sich auch bei professionellen IR-Kameras finden.

Seek Compact

Die Seek Compact ist mit weniger als 4,5x2x2x2 cm³ die kleinste Kamera der Seek-Familie, verfügt aber dennoch über einen Sensor mit 206x156 Pixel. Die Seek Compact ist als Plug-in-Modul für ein Smartphone konzipiert und in zwei Versionen erhältlich: mit einem Micro-USB-Anschluss für Android-Geräte und mit einem Lightning-Anschluss für iPhones und iPads. Die Kamera wird mit einer stabilen Aufbewahrungsbox geliefert, damit man sie sicher transportieren kann.

Nach der Installation der entsprechenden App kann man die Seek Compact sofort verwenden. Die Funktionen der App sind denen der ShotPro sehr ähnlich. Es gibt eine Reihe von Farbpaletten zur Auswahl, verschiedene Möglichkeiten, um Temperaturpunkte im Bild zu messen, natürlich auch eine Screenshot-Funktion und ein Einstellungsmenü. Im Vergleich zum ShotPro gibt es aber keine Möglichkeit, den Emissionspegel einzustellen. Das dürfte der durchschnittliche Anwender aber leicht verschmerzen können.

Wie bei der ShotPro ist es auch hier möglich, das Wärmebild mit dem der Smartphone-Kamera zu kombinieren. Allerdings ist das Ergebnis etwas anders: Auf der einen Seite des Bildschirms sieht man das Wärmebild, auf der anderen das Bild der normalen Kamera. Die Bilder werden nicht wie bei der ShotPro übereinander projiziert. Es gibt verschiedene Einstellungen, um Größe und Position der Bilder anzupassen, aber ich fand das Ergebnis aufgrund der unterschiedlichen Blickwinkel und Positionen der Kameras nicht wirklich überzeugend.

Nebeneinandergelegt

Ich habe die beiden Kameras ausprobiert und viele verschiedene Objekte gemessen.

In der Praxis erweist sich die Seek ShotPro als ein sehr handliches und robustes Gerät mit sehr guter Bildqualität. Die Auflösung ist ausgezeichnet und vor allem mit dem überlagerten Bild der IR- und normalen Kamera erhält man viele nützliche Informationen. Nach Angaben des Herstellers beträgt die Akkulaufzeit vier Stunden, was eine recht lange Zeit ist. Natürlich ist das Gerät kein Billigheimer, aber angesichts der Qualität und der Leistung ist der Preis sicherlich gerechtfertigt.

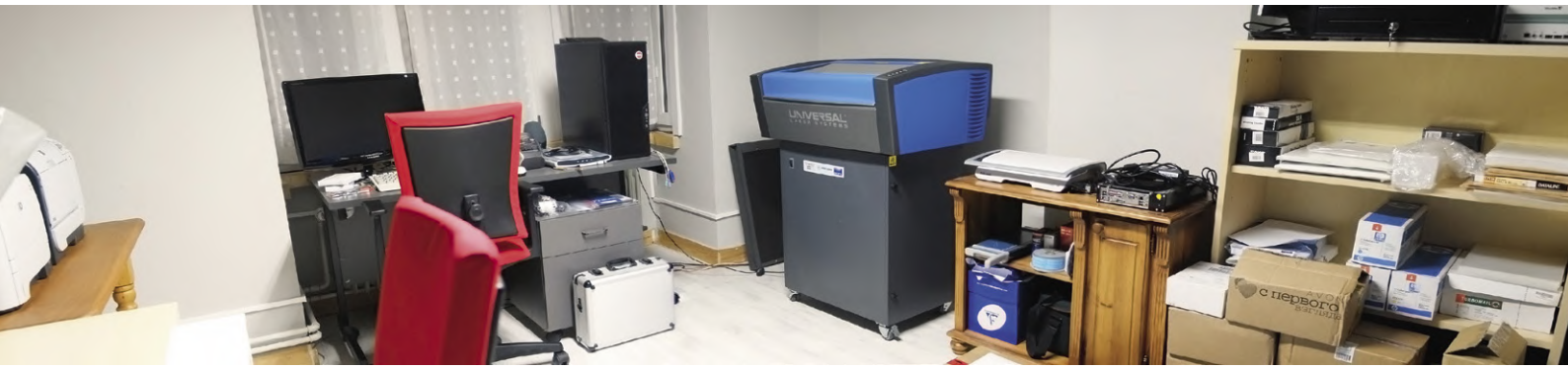
Bei der Seek Compact blieben die Konturen im Bild etwas undeutlicher, auch nachdem ich mit den verschiedenen Paletten eine Zeit lang experimentiert hatte. Obwohl die Auflösung deutlich geringer ist als bei der ShotPro, kann man diesen Nachteil ausgleichen, indem man näher an das Messobjekt herangeht. Die Lösung des Doppelbildes (statt der Überlagerung) ist nicht elegant gelöst, aber das ist halt der Nachteil des Plug-in-Moduls, das nicht über zwei Kameras verfügt. Wir sollten aber nicht so kritisch sein, denn auch der Preis ist mit 300 € ebenfalls sehr unterschiedlich.

Beide Lösungen eignen sich auf jeden Fall hervorragend zur Temperaturmessung zum Beispiel in elektronischen Schaltungen und Geräten. Man kann aber damit noch viel mehr anfangen: Gehen Sie einfach durch Ihr Labor oder vor Ihr Haus und richten Sie die Kamera auf alle möglichen Wände, Fenster und Türen. Sie werden erstaunt (und manchmal auch unangenehm überrascht) sein, was Sie da alles zu sehen bekommen!

(190321)

Aus dem Leben gegriffen

Die Planung eines Labors und eines Arbeitsbereichs



Von **Ilse Joostens** (Belgien)

Wenn ich etwas Zeit übrig habe, stöbere ich gerne in Online-Elektronikforen, um zu sehen, was bei Hobby-Elektronikern und auch bei Profis angesagt ist. Dort wird regelmäßig die Einrichtung eines Labors oder Arbeitsbereichs diskutiert, manchmal mit notwendigem Bildmaterial. Bei den Hobbyisten lassen sich zwei Arten unterscheiden, der „Poser“ und der „Low-Budget-Mann”. Während die ersten meist Fotos eines Arbeitsbereichs pos(t)en, der mit allen möglichen obskuren Messgeräten bis zur Decke gefüllt ist, kauft der Low-Budget-Mann (oder -Frau) seine/ihre Sachen lieber im örtlichen Baumarkt und heutzutage auch immer mehr über das Internet in China.

Wenn Sie professionell arbeiten und Ihre eigenen Produkte entwickeln und vermarkten wollen, werden Sie mit einem billigen Lötkolben aka Schürhaken nicht sehr weit kommen. Lassen Sie uns versuchen, ein paar Dinge über das professionelle Elektroniklabor und seine Ausrüstung klarzustellen.

Mess- und andere Elektronikgeräte

Im Moment benutze ich im Labor nicht viel mehr als drei Labornetzgeräte, zwei Multimeter, ein älteres Oszilloskop, zwei Lötstationen, eine Heißluftstation, ein Universalprogrammiergerät, ein Stereomikroskop und eine Wärmebildkamera (die ich eigentlich für andere Zwecke gekauft habe). Ich persönlich ziehe Qualität der Quantität vor, und wenn es um Geräte und Werkzeuge für intensiven Gebrauch geht, kaufe ich lieber namhafte Marken – aber dann nur das, was ich wirklich benötige. Benutzerfreundlichkeit, Zuverlässigkeit und Langlebigkeit stehen im Vordergrund. Natürlich spielt auch der Preis eine Rolle, aber auch viele andere, manchmal weniger offensichtliche Aspekte.

Nicht selten erscheinen meine Messgeräte auf YouTube oder in einer Montageanleitung für einen Bausatz. Dann macht ein ordentliches, professionelles Multimeter wirklich einen besseren Eindruck als ein preiswertes aus dem Baumarkt. Sinneseindrücke sind sehr wichtig und Sie wollen bei Ihren Kunden ja einen professionellen und zuverlässigen Eindruck hinterlassen.

Die Steuerlast in Belgien gehört zu den höchsten in der westlichen Welt. Manchmal kann eine Investition einen attraktiven Weg darstellen, diese Steuern ein wenig zu senken. Ende letzten Jahres habe ich auf Anraten meiner Buchhalterin einige Geräte gekauft und erneuert. In diesem speziellen Fall war das Verhältnis von Steuerabzug und Qualität besonders wichtig, so dass ich unter anderem zwei sehr solide, aber recht teure Lötstationen gekauft habe.

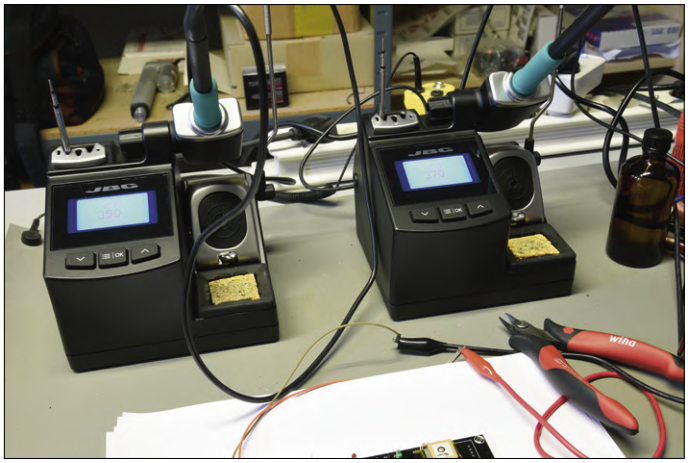


Bild 1. Zwei teure Lötstationen – Danke, liebes Finanzamt!

Geräte, die ich weniger oft benutze (zum Beispiel den Universalprogrammer) oder die weniger bedeutsam sind, kaufe ich auch ab und zu gebraucht, obwohl das immer mit Risiken verbunden ist.

Produktionsmaschinen

Wenn Sie Ihre eigenen Produkte vermarkten und nicht alles auslagern wollen, ist die Wahrscheinlichkeit hoch, dass Sie die eine oder andere Produktionsmaschine kaufen müssen. Bei mir handelte es sich um eine Laserschneidemaschine und einen Reflowofen.

Geben Sie vor allem der Versuchung nicht nach, in billigere Maschinen zu investieren, die eigentlich für den Hobbygebrauch bestimmt sind. Nichts macht weniger Spaß als Hunderte von Rückständen und unzufriedene Kunden, weil eine Maschine mitten in einem Produktionsprozess plötzlich ihren Geist aufgibt. Auf lange Sicht kostet Sie eine billigere Maschine garantiert mehr, ganz zu schweigen von immer wieder auftretenden Ärgernissen (wie dem x-fachen Austausch einer Laserröhre). Mit einer professionellen Maschine können Sie außerdem versichert sein, dass Ersatzteile langfristig verfügbar bleiben und Sie auf Wunsch auch einen Servicevertrag abschließen können.

Sonstige Ausstattung

Wenn Sie Ihre eigenen Produkte vermarkten wollen, benötigen Sie zusätzlich zu den Klassikern wahrscheinlich eine Menge anderer Geräte und Materialien, die für den durchschnittlichen Hobby-Elektroniker normalerweise von geringem Interesse sind. Beispiele dafür sind eine solide Spiegelreflex-Kamera für Produktfotos, Beleuchtung, Fotohintergründe, Heißsiegelgeräte, ein Farblaserdrucker, eine Papierschneidemaschine, Regale für fertige Produkte und Teile, Elektrowerkzeuge und so weiter. Das ist etwas, das schnell ins Geld gehen kann, daher sollten Sie das von Anfang an berücksichtigen.

Software

Natürlich benötigen Sie auch viel Software und mehrere Computer. Was diese Computer betrifft, so werde ich Ihnen keinen Rat geben; es ist besser, das selbst herauszufinden. Elektor ist keine Computerzeitschrift und nein, ich werde Ihren Computer nicht reparieren!

Was die Software betrifft, so verwende ich neben einem klassischen Office-Paket derzeit ein Paket für den Platinenentwurf, zwei verschiedene CAD-Programme (2D), Bildbearbeitungssoftware, DTP-Software und Software für eine Reihe sehr spezieller Anwendungen. Einer meiner Bekannten sagte immer: „Man kauft keine Software, man kopiert sie.” Vielleicht sind sie in der glücklichen Lage, kostenlose Open-Source-Software verwenden zu können, meist landen Sie aber bei kommerziellen Anwendungen mit in der Regel teuren Lizenzen. Leider hat sich das Lizenzmodell vieler Softwarepakete in den letzten Jahren von einem einmaligen Kauf zu einem Jahres- oder Monatsabonnement entwickelt. Für mich ist es kein unüberwindbares Problem, einmal einen angemessenen Betrag für eine Lizenz zu zahlen, aber mit den aktuellen Abonnements gestaltet sich das nicht so einfach. 600...700 € pro Jahr für die Nutzung eines Softwarepakets - das ist meiner Meinung nach fast schon Erpressung. Für all meine Software können da schnell mal ein paar tausend Euro pro Jahr zusammenkommen. Und was passiert, wenn Sie Ihr Abonnement kündigen? Können Sie Ihre Dateien dann noch öffnen? Es bleibt fraglich... Und bei all den Steuern, die ich hier in Belgien bereits zu zahlen habe, wird es allmählich wirklich zu viel.

Vorläufig werde ich bei meinem alten vertrauten Windows 7 bleiben (für die Microsoft-Hasser unter Ihnen: Ja, ich benutze auch Linux). Übrigens, alle meine älteren Softwarepakete mit „Lizenz fürs Leben” laufen noch super. Okay, es mag etwas weniger schick aussehen, aber sie tun, was sie tun sollen. All dieser neumodische Firlefanz in den neuesten Versionen macht ein Programm noch langsamer. Ich bekomme auch kaum noch Mails und Bitten, auf ein Abonnement umzusteigen. Wahrscheinlich haben die die Hoffnung inzwischen aufgegeben, hurra!

191152-03

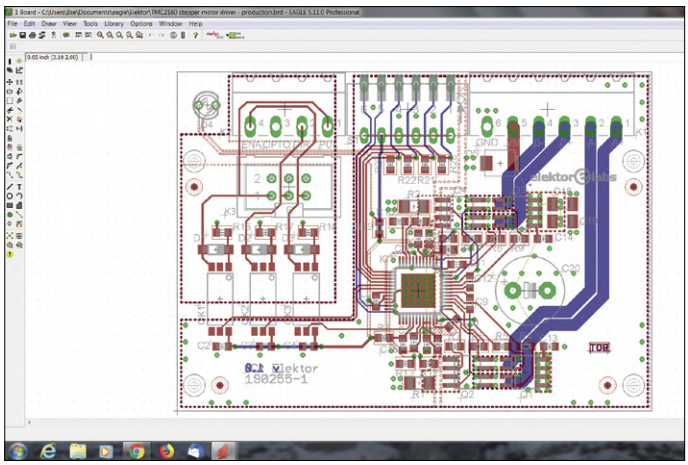


Bild 2. Version 5 von Eagle ist nicht die aktuelle, aber für mich gut genug.

Bild 3. Zwei Drucker und ein Stereomikroskop.

7-Segment-LED-Anzeige Monsanto MAN1

Bemerkenswerte Bauteile

Von **Neil Gruending** (Kanada)

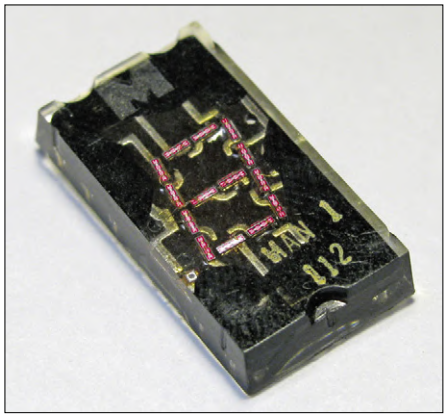


Bild 1. Monsanto MAN1 Konstruktion [1].

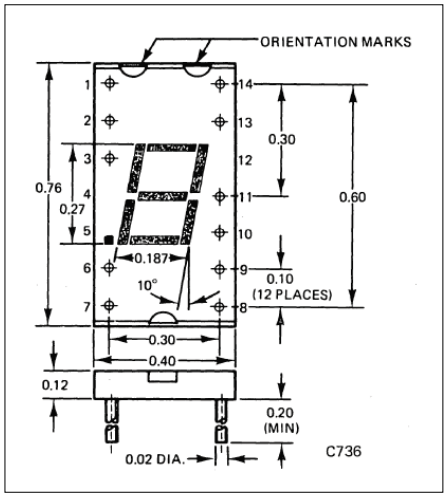


Bild 2. Abmessungen der Monsanto MAN1 [2].

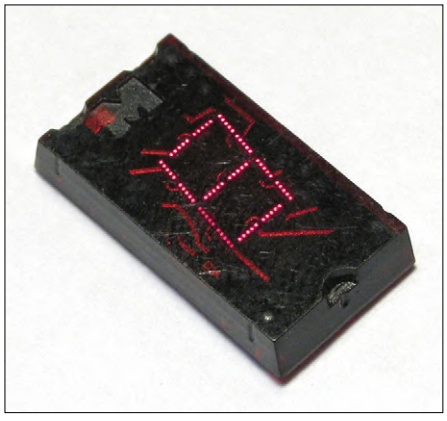


Bild 3. Die 7-Segment-Anzeige Monsanto MAN1A [1].

Die Idee, alphanumerische Zeichen mit sieben Segmenten darzustellen, entstand Anfang des 19. Jahrhunderts, um Zeichen per Telegraf übertragen und dann auf ein Papierband drucken zu können. Kurz darauf wurden 7-Segment-Großdisplays gebaut, bei denen Glühlampen die einzelnen Segmente beleuchteten. Aber 7-Segment-Displays waren nicht besonders populär, bis Anfang der 1970er Jahre miniaturisierte LED-Versionen auf den Markt kamen. Werfen wir einen Blick auf eines der ersten Exemplare namens Monsanto MAN1 (Monsanto AlphaNumeric).

7-Segment-Displays mit LEDs waren ein wichtiger Meilenstein, da sie viel weniger Strom benötigten als andere damals verfügbare Technologien wie Nixie-Röhren, Minitrone und Panaplex-Anzeigen. Ihr relativ geringer Strombedarf ermöglichte auch den Betrieb in batterieversorgten Geräten wie die Pulsar-Time-Computeruhr und Taschenrechner wie den HP35. Die Akkulaufzeit lag im Stundenbereich, was für moderne Verhältnisse zwar ziemlich schlecht ist, aber die genialen Entwickler haben sie so weit wie möglich verlängert, indem sie das Display, wann immer möglich, ausgeblendet haben.

Das Display Monsanto MAN1 besaß für jedes Segment zwei Dies aus Gallium-Arsenid-Phosphat (GaAsP) mit jeweils vier LED-Dioden. Für alle Ziffernsegmente einschließlich Dezimalpunkt waren also 57 LEDs erforderlich (**Bild 1**)! Das Zeichen war nur 6,8 mm (0,27 Zoll) hoch (**Bild 2**), so dass das Display in vielen tragbaren Anwendungen wie Taschenrechner eingesetzt werden konnte. Es war auch üblich, Vergrößerungslinsen über dem Display anzubringen, damit die Ziffern größer erschienen. Die Dioden waren auch nicht sehr hell, so dass sie normalerweise mit einem Rotfilter zur Erhöhung des Kontrasts abgedeckt wurden. Dies verlieh den damaligen Displays ihren charakteristischen Rotton. Das MAN1-Dispaly war in einem durchsichtigen Kunststoffgehäuse untergebracht, das die gesamte interne Elektronik offenbarte. Bei einer späteren Version namens MAN1A wurde roter Kunststoff verwendet, um die Innereien des Displays zu verbergen (**Bild 3**).

Der Wettbewerb war damals heftig und die Technologien für LED-Displays änderten sich schnell. Es dauerte nicht lange, bis HP 5×7-Punktmatrix-LEDs mit integrierten binären Dezimaldecodern (BCD) einführte. Litronix, ein Spin-off von Monsanto, brachte 1977 ein intelligentes LED-Display mit Zeilen/Spaltentreibern und integriertem Zeichen-ROM auf den Markt. Aber es ist trotzdem interessant, diese coolen kleinen MAN1-Displays in der klassischen Elektronik und Testausrüstung zu entdecken. Falls Sie mehr darüber erfahren möchten: Detaillierte Kataloginformationen [2] für diese Displays sind von General Instruments, die 1979 Monsanto übernahmen [3], verfügbar. ⏮

190383-02

Weblinks

[1] Aufbau Monsanto MAN1: www.decadecounter.com/vta/articleview.php?item=463

[2] Abmessungen Monsanto MAN1: https://bit.ly/2BuRRgf

[3] Geschichte der Monsanto-LED: www.datamath.org/Display/Monsanto.htm

HOMELAB PROJECT

Teeuhr

Fingerübung in Sachen Energy-Harvesting

Von **Lothar Göde**

Die „Thermoelektrische Teeuhr“ begrenzt die Zeit, die ein Teebeutel im heißen Wasser zieht. Die Steuerung der Mechanik erfolgt elektronisch, wobei die dazu benötigte elektrische Energie mit moderner Energy-Harvesting-Technologie aus der Differenz zwischen der Temperatur des heißen Tees und der Raumtemperatur gewonnen wird. Somit kann bei diesem Projekt auf Batterien, Akkus oder gar Netzversorgung verzichtet werden.

Vor einiger Zeit habe ich einen Artikel zum Thema Energy-Harvesting gelesen. Diese Technologie war für mich neu und ich fand sie sehr interessant und spannend zugleich. Über einen Verein, der unter anderem Bausätze für junge Menschen zur Verfügung stellt, um sie im Rahmen der schulischen Ausbildung an die Elektronik heranzuführen, erhielt ich einen Energy-Harvesting-Bausatz. Kernstück dieses Bausatzes ist der LTC3108 [1] der Firma Linear Technology (heute Analog Devices). Faszinierend war für mich an diesem IC, dass aus einer Gleichspannung von nur 20 mV, bei der normalerweise noch kein Transistor oder MOSFET arbeitet, eine wesentlich höhere Spannung erzeugt wird, die zum Beispiel einen Mikrocontroller versorgen kann. Der LTC3108 ist dafür konzipiert, dass er seine Eingangsspannung aus der Temperaturdifferenz eines Thermoelektrischen Generators (TEG) bezieht.

Thermoelektrischer Generator

Ein Thermoelektrischer Generator kann elektrische Energie aus einer Wärmedifferenz erzeugen oder mit Zufuhr elektrischer Energie eine Wärmedifferenz erzeugen. Ein TEG besteht aus zwei isotropen, homogenen Halbleitermaterialien mit unterschiedlicher (p-,n-)Dotierung (**Bild 1**). Die populärste Anwendung des thermoelektrischen oder Peltier-Effekts sind diese preiswerten Kühlboxen, die man im Auto in den Zigarettenanzünder stöpselt und die dann in kurzer Zeit die Autobatterie leersaugen (ohne dass das Bier wirklich kalt wird).
Doch es geht auch anders herum: Die Umkehrung des Peltier- ist der Seebeck-Effekt. Eine vorhandene Wärmedifferenz wird in elektrische Energie umgewandelt. Die erzeugte Spannung hängt von den Eigenschaften der thermoelektrischen Materialien und natürlich der Temperaturdifferenz ab. Ein thermoelektrischer wird übrigens ähnlich wie ein thermovoltaischer Generator betrieben: Der entnommene Strom wird so gewählt, dass die Klemmenspannung nur leicht abfällt, die Leistung aber maximal ist (Maximum Power Point, MPP).

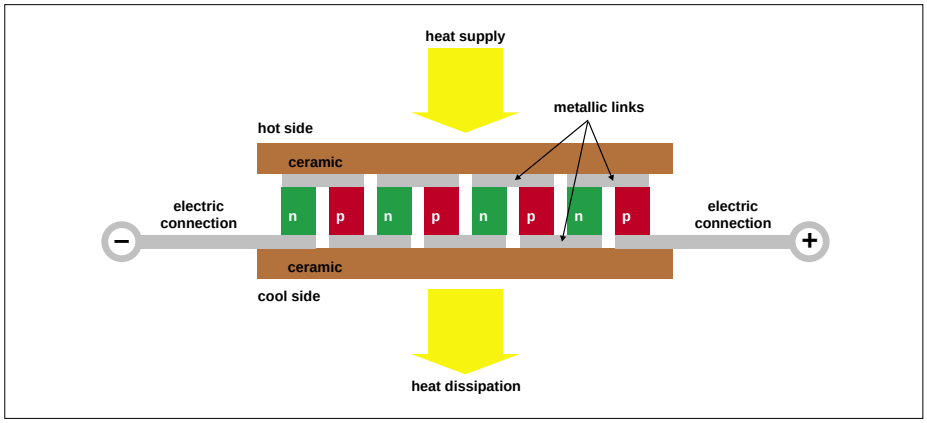


Bild 1. Prinzipieller Aufbau eines Thermoelektrischen Generators.

Heißer Tee, kalter Tee?

Beim Experimentieren mit dem interessanten Lernpaket trank ich nebenbei oft Tee. Doch wie das so kommt, man vergisst vor lauter Begeisterung meist die Teetasse und dann ist der Tee bitter und kalt. Zur Lösung des Problems stellte ich mir die Frage: „Ist es möglich, aus der Temperaturdifferenz eines mit kochendem Wasser befüllten Teeglases und dessen Umgebung so viel Energie zu gewinnen beziehungsweise umzuwandeln, dass eine energiesparende Elektro(mecha)nik den Teebeutel nach Ablauf der vorgesehenen Brühzeit aus dem Teeglas hebt?“ Zur Beantwortung dieser Frage musste ich zunächst einige Experimente mit dem TEG durchführen. Zur effektiven Wärmeableitung von der kalten Platte des TEGs kommt ein Aluminiumkühlkörper eines CPU-Kühlers aus der Bastelkiste zum Einsatz. Auf der anderen Seite muss

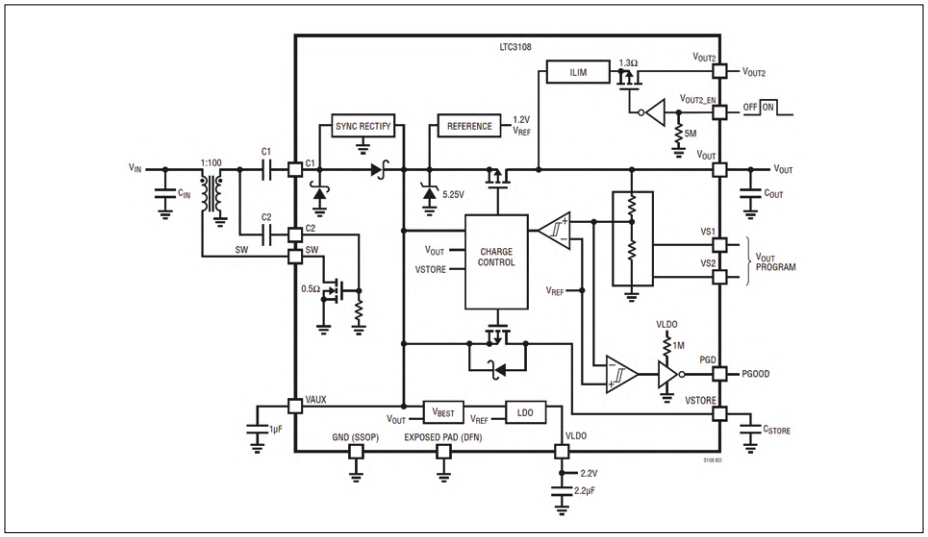


Bild 2. Der LTC3108 kann quasi aus dem Nichts Energie zur Versorgung einer kleinen Schaltung ernten (Quelle: Analog Devices).

man diverse Teegläser und Wärmeleitmaterialien ausprobieren, um eine optimale Wärmeübertragung zu erreichen. Sind diese Probleme gelöst, kann man in fünf Minuten Aufgusszeit aus einem Teeglas mit kochendem Wasser so viel Energie ernten, dass der LTC3108 einen Elko mit 2200 µF (CSTORE in **Bild 2**) zuverlässig auflädt, so dass die Steuerelektronik ihren Dienst verrichten kann.

Beförderungsmechanismus

Was lässt sich mit der geernteten Energie anfangen? Natürlich sind „stromfressende“ Funktionen nicht möglich, alle Teile einer Teeuhr müssen sehr sparsam mit Energie umgehen. Zunächst zur Mechanik: Ein Teeglas wird auf den TEG gestellt, der Teebeutel (oder das Teeei) manuell in die Tasse gesteckt und das Teebeutelschildchen über eine Klemme mit dem Draht einer „Hebemechanik“ verbunden.

Bei einer gewöhnlich versorgten Schaltung könnte man jetzt einfach die Hebemechanik mit einem Motor ausstatten, der den Beutel aus dem Sud zieht. Das ist hier aber nicht sinnvoll. Stattdessen wird wie in **Bild 4** zu sehen der Draht über eine Umlenkung geführt und mit einem kleinen Gegengewicht (etwas schwerer als ein Teebeutel) ausgestattet. Auf diese Weise ernten wir weitere Energie, nämlich die der Schwerkraft. Da das Gegengewicht

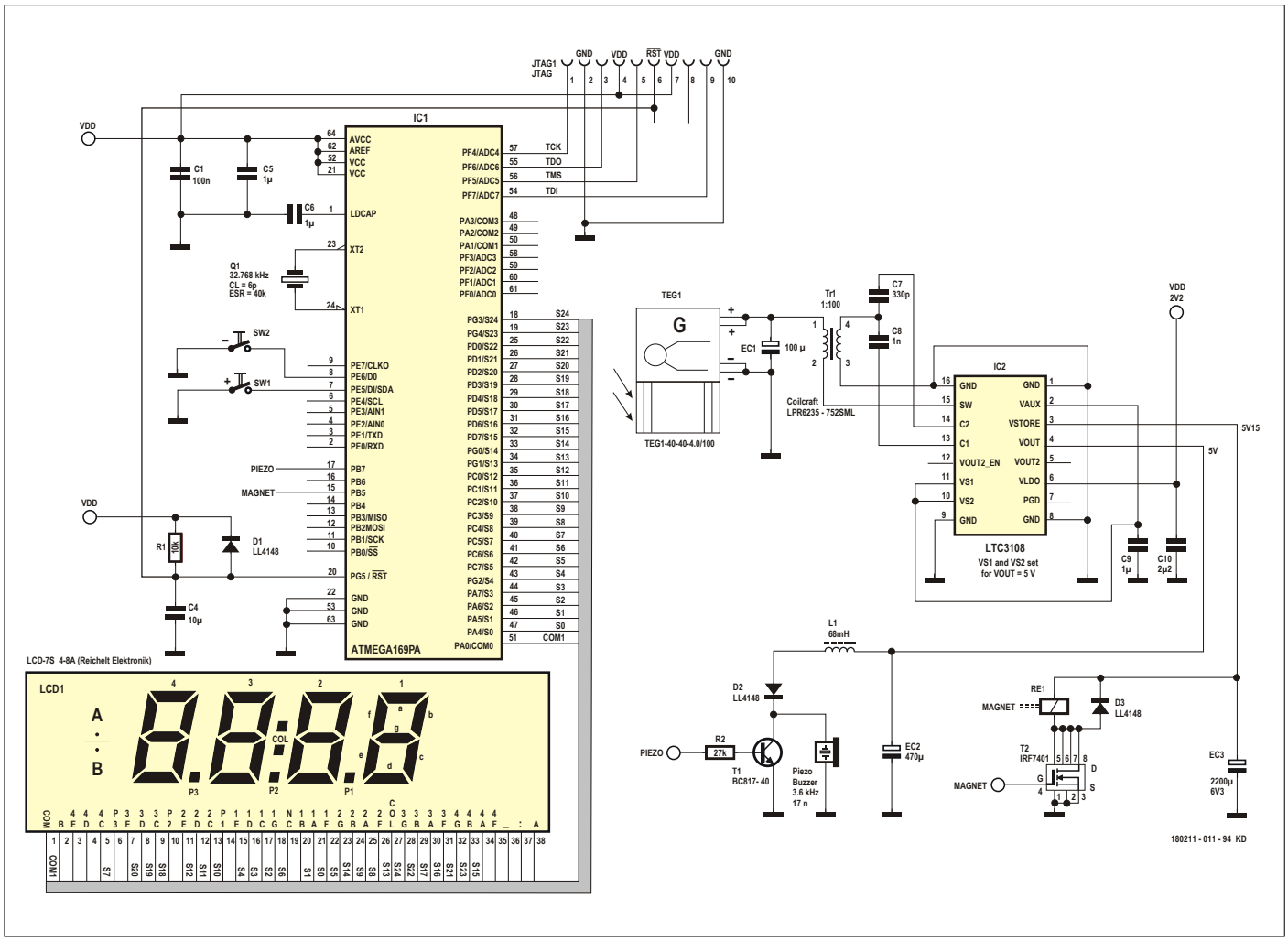


Bild 3. Das Schaltbild der Thermoelektrischen Teeuhr.

schwerer ist als der Teebeutel, muss es von einem Bolzen arretiert werden, der durch Federkraft den Weg des Gegengewichtes versperrt, sobald dieses eine bestimmte Position erreicht hat. Die Freigabe des Bolzens erfolgt elektromagnetisch mit einem einzigen Stromstoß. So muss nicht ein relativ kräftiger Strom über einen längeren Zeitraum geliefert werden. Für die Bolzenarretierung habe ich ein Relais umgebaut und den Bolzen am Kontaktsatz des Relais befestigt. Es sind auch spezielle Hubmagnete im Handel, die hier ebenfalls eingesetzt werden könnten. Damit der Tee beim Herausheben des Beutels nicht spritzt, habe ich das Gegengewicht mit einem Gummidämpfer versehen müssen.

Der Mikrocontroller als Timer und Treiber

Weil wir so sparsam mit der Energie für den Mechanismus umgehen, können wir uns bei der Steuerzentrale der Teeuhr, einem Mikrocontroller ATmega169PA von Atmel [2], einigen Luxus erlauben. Im ATmega169PA ist ein LCD-Controller eingebaut, was den bequemen Anschluss eines Displays zur Anzeige der Ziehzeit erlaubt. Übrigens ist es tatsächlich auch energiesparender, wenn man eine Hardwarefunktion eines Controllers nutzt, als diese Funktion per Software zu emulieren.

In das Teeglas wird nun kochendes Wasser geschüttet, so dass der TEG elektrische Energie erzeugt. Sobald die Energy-Harvesting-Schaltung die Versorgungsspannung für den Mikrocontroller bereitstellt, wird die Ziehdauer im Display blinkend angezeigt. Bis die Anzeige erscheinen kann, vergehen circa 30 s. Die meisten Teesorten müssen 5 min ziehen, so dass als Standardwert eine Zeit von 4:30 min angezeigt wird.

Am Mikrocontroller sind zwei Tasten angeschlossen, mit denen sich (unter anderem) die Ziehdauer einstellen lässt. Wird für einige 12 s keine der Tasten betätigt, dann blinkt die Anzeige nicht mehr und die eingestellte Ziehzeit wird im Sekundentakt heruntergezählt.

Einige Sekunden vor Ablauf der Brühdauer ertönt das akustische Signal eines Piezopiepsers. Dies soll ein Hinweis darauf sein, dass der Tee gleich zur Abholung bereit steht. Wenn die Ziehzeit endgültig abgelaufen ist, wird vom Mikrocontroller ein kurzes Signal zur Relaisspule geschickt, die den Arretierbolzen freigibt, so dass das Gegengewicht den Teebeutel in gedämpfter Bewegung aus dem Teeglas zieht.

Das vom Piezosignalgeber erzeugte akustische Signal muss leicht zeitversetzt vor Ablauf der Ziehdauer ausgegeben werden, um in der verbleibenden Zeit die verpiepste Energie wieder kompensieren zu können, so dass ausreichend elektrische Energie für die Betätigung der Relaisspule vorhanden ist. Wurde das akustische Signal ignoriert, lässt die Teeuhr nicht locker; weitere akustische Signale folgen im Minutentakt. Dieser Alarm lässt sich durch Betätigung einer Taste abstellen. Nötig sollte das aber nicht sein, denn wenn man die heiße Tasse vom TEG nimmt, bricht die Spannungsversorgung ja in kurzer Zeit ohnehin zusammen.

Fazit

Eine Schaltung so zu konzipieren, dass sie mit der geringen geernteten Energie beim Energy-Harvesting auskommt, ist eine große Herausforderung. Dies betrifft nicht nur die Hardware-Konstruktion, sondern auch und vor allem das Energiemanagement bei der Erstellung der Firmware. Beispielsweise musste die Firmware so optimiert werden, dass bei der Einstellung der Ziehdauer durch die betätigten Tasten nur ein minimaler Strom benötigt wird, damit die Versorgungsspannung für den Mikrocontroller nicht zusammenbricht. Aber genau dies ist ja Sinn und Zweck solcher Experimente!

Die Thermoelektrische Teeuhr kann zusätzlich mit einem Funkmodul (Bluetooth Low Energy, BLE) erweitert werden, damit eine Nachricht an ein Smartphone gesendet wird, wenn der Tee fertig ist. Auch könnte das Herausheben des Teebeutels aus dem Teeglas ein stromsparender Schrittmotor übernehmen.

Bild 4. Noch im Versuchsstadium: Der funktionierende Prototyp der Teeuhr.

Auch wenn es im Foto des Prototyps (**Bild 4**) nicht so aussieht, stellt die Thermoelektrische Teeuhr (im Gegensatz zu vielen kommerziellen Produkten mit einer ähnlichen Aufgabe, aber geradezu antiker Herangehensweise) durch die Verwendung der modernen Energy-Harvesting-Technologie ein ausgesprochenes Hightech-Gerät dar [3]. Lästige Pflichtaufgaben wie Batteriewechsel oder Aufladen von Akkus entfallen. Da auf Batterien und Akkumulatoren verzichtet wird, können auch keine chemischen Substanzen auslaufen. Das Gerät ist (bis auf die Reinigung) wartungsfrei. Es entstehen keinerlei gefährliche elektrische Spannungen, so dass auf Schutzmaßnahmen verzichtet werden kann. Die Thermoelektrische Teeuhr ist einzigartig! ⏮

180211-01

Das Projekt wurde auch bereits im Reichelt Elektronik Magazin veröffentlicht.

Weblinks

[1] LTC3108: www.analog.com/media/en/technical-documentation/data-sheets/LTC3108.pdf

[2] Mikrocontroller ATmega169PA: www.microchip.com/wwwproducts/en/ATmega169PA

[3] Video zur Teeuhr: www.youtube.com/watch?v=z-ias4IwbSQ

Einstellbarer 1-kW-AC-Motortreiber

Steuerung in drei Modi: Schwingungspakete, Phasenabschnitt oder Phasenanschnitt

Entwicklung: **Elektor Labs Indien**

Dieser kostengünstige und einfach zu realisierende Motortreiber wurde entwickelt, um Wechselstrommotoren in Haushaltsgeräten und Elektrowerkzeugen zu steuern. Die Software im Mikrocontroller deckt verschiedenartige (induktive und kapazitive) AC-Lasten ab. Die am Mikrocontroller angeschlossene Hardware wird mit Lasten bis zu 1 kW fertig. Dank der Anpassungsfähigkeit der Hardware kann die Schaltung auf viele persönliche Anforderungen zugeschnitten werden.

Viele Elektrogeräte im Haushalt benötigen für den Betrieb ihrer Motoren die Netzwechselspannung von 230 V_{AC} mit einer Frequenz von 50 Hz aus der Steckdose. Normalerweise erreicht der Wechselstrom ein Gerät über einen elektronischen Leistungsschalter. Für einen „sanften" Betrieb der Last und die Möglichkeit, die Motordrehzahl zu steuern, muss die an die Last angelegte Wechselstromleistung variiert werden. Dazu werden verschiedenartige elektronische Leistungsschalter eingesetzt.

Zwar könnte man in leistungsschwachen Geräten wie Ventilatoren, Pumpen oder Kompressoren auch bürstenlose Gleichstrommotoren einsetzen, erste Wahl sind aber nach wie vor preisgünstigere, gut verfügbare und weniger komplexe Einphasen-Wechselstrommotoren.

Designüberlegungen

Die Schaltung wurde nicht nur für gewöhnliche Wechselstrommotoren, sondern auch für viele andere Wechselstromlasten ausgelegt, bei denen eine Leistungsregelung erforderlich oder vorteilhaft ist. Um diese Variabilität zu bieten, müssen in einer einfachen Benutzeroberfläche unterschiedliche Methoden der Leistungssteuerung für die verschiedenartigen Wechselstromlasten ausgewählt werden können. Der Leistungstreiber soll die Steuerung einer Reihe von Hochleistungs-LEDs genauso problemlos beherrschen wie der Halogenlampe im Bastelkeller oder Omas Mini-Elektroheizkissen. Der Treiber soll für den Betrieb an Lasten von maximal 1 kW ausgelegt sein (er wurde an Lasten bis zu 350 W getestet). Weiterhin soll durch ein klares „Nutzerinterface" in Form eines OLED-Displays und eines Drehgebers (mit Drucktaste) die Auswahl der Parameter bequem gestaltet werden. Und dieses Nutzerinterface sollte auch noch um zusätzliche Funktionen erweiterbar sein.

Leistungssteuerungsverfahren

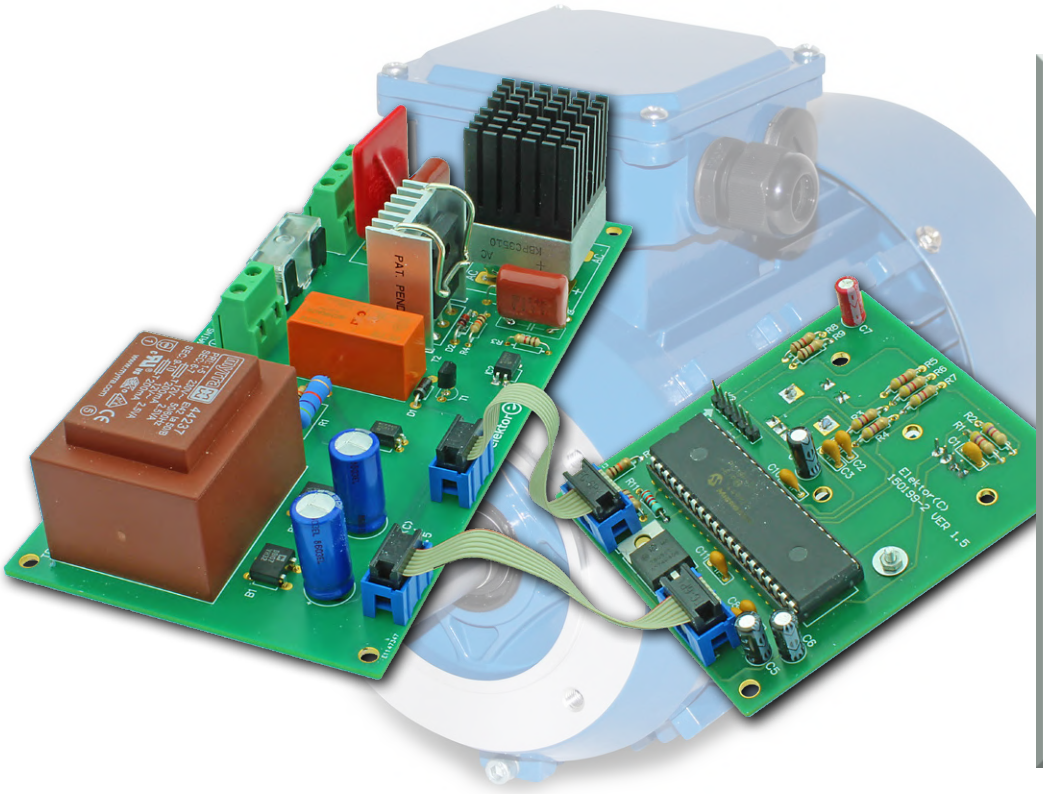
Lassen Sie uns einige etablierte Methoden zur Leistungssteuerung einer Wechselstromlast wie einen Motor betrachten.

1. Phasenschnitt-Verfahren

Die Last wird zur Leistungssteuerung nur während eines Bruchteils der Dauer einer Halbwelle mit der Netzwechselspannung verbunden. Das die Last steuernde Bauteil wird dazu abhängig vom Nulldurchgang der Netzspannung in jeder Halbwelle abgeschaltet, so dass die Last für diesen Zeitraum von ihrer Energieversorgung abgeschnitten ist. Je länger die

Eigenschaften

- Wechselstromausgang (eine Phase): 230 VAC, max. 1 kW
- Betriebsarten
  - Schwingungspaketsteuerung
  - Phasenabschnitt
  - Phasenanschnitt
- 0,96-Zoll-OLED-Display
- 3-polige Anschlussklemme für Motor/Last
- Relaisgesteuerter Vorwärts/Rückwärts-Betrieb des Motors
- Treibersteuerung durch PIC18F45K22
- Intern isolierte MCU-Steuerelektronik
- Vorwärts/Rückwärtslauf über Drehgeber-Taster
- Steuerung der Ausgangsleistung/Motordrehzahl durch Drehgeber
- EEPROM-Speicherung der Motor/Last-Treiberparameter
- Softstart im Nulldurchgang bei Änderung des Modus oder der Motorrichtung



INFOS ZUM PROJEKT

Wechselstrommotor
Leistungsregelung
Dimmer PIC

Einsteiger
➡ Fortgeschrittene
Experte

etwa 3 Stunden

Standard-Laborwerkzeuge, CCS-Compiler und PicKit-Programmieradapter von Microchip (optional)

etwa 75 €

Abschaltzeit, desto geringer die Leistung. Phasenschnittschaltungen sind einfach zu realisieren, haben aber große Nachteile: starke Strom- oder Spannungsspitzen und, da je nach Eigenschaft der Last der Strom nicht wie die Spannung sinusförmig verläuft, eine hohe Blindleistung. Es gibt hauptsächlich zwei Möglichkeiten, die Phase zu beschneiden:

1a. Phasen**an**schnitt

Der Phasenanschnitt (engl. Leading Edge LE oder Forward-Phase-Control FPC) ist die preiswerteste Methode, eine Last zu „dimmen“. Die Last wird dabei eine einstellbare Zeit (Anschnittwinkel) nach dem Nulldurchgang der Netzspannung eingeschaltet. Das steuernde Glied in einer solchen Schaltung ist in der Regel ein Triac, der nach der Zündung im Anschnittwinkel selbsttätig leitet (ohne weitere Zündimpulse), bis ein Mindeststrom (Haltestrom) unterschritten wird, was kurz vor dem nächsten Nulldurchgang geschieht. Bei diesem Verfahren sind durch das schnelle Einschalten des Triacs die Stromimpulse sehr hoch, was schlecht für viele angeschlossene elektronische Bauteile ist. Dem Triac wird (hoffentlich) meist eine Induktivität zur Seite gestellt, die den Anstieg des Stroms etwas abbremst.
Solche Dimmer eignen sich für ohmsche und induktive Lasten wie Motoren, Glühlampen, Neonröhren, Kaltkathoden- und Niedervoltlampen (induktiv/magnetisch). Dagegen ist der Phasenanschnitt nicht für stark kapazitive Lasten wie Schaltnetzteile und elektronische Vorschaltgeräte geeignet.

1b. Phasen**ab**schnitt

Bei der Phasenabschnitt-Methode (engl. Trailing-Edge TE oder Reverse-Phase-Control RPC) wird das steuernde Element direkt im Nulldurchgang aktiviert und nach einer einstellbaren Zeit wieder abgeschaltet. Eine solche Schaltung erfordert MOSFETs oder IGBTs und ist deshalb anspruchsvoller zu realisieren (und daher teurer) als der Phasenanschnitt mit Triac und Spule. Dafür wird der Anwender mit einer weichen und geräuschlosen Leistungsregelung entschädigt. Ein Phasenabschnittdimmer erfordert eine geringere Mindestlast (oft nur 5 W) als sein anschneidendes Gegenstück und ist daher die bessere Wahl für das Dimmen von Lichtanlagen mit geringer Leistung. Auch Schaltnetzteile, die in der Regel ein stark kapazitives Verhalten aufweisen, benötigen zur Leistungssteuerung einen Phasenabschnitt.
Phasenabschnittdimmer besitzen quasi einen integrierten „Soft-Start“, der die angeschlossene Elektronik schont und auch verhindert, dass Glühlampen beim ersten Einschalten einen Thermoschock erleiden. Phasenabschnittdimmer sind die beste Wahl auch für die kapazitive Last eines LED-Treibers, um einen flimmerfreien Betrieb zu gewährleisten.
Dieses Verfahren ist für ohmsche und kapazitive Lasten geeignet, aber nicht für induktive, da es starke Gegen-EMK-Spannungsimpulse von der Last zur Folge hat, die den MOSFET und alle Schutzvorrichtungen in der Treiberschaltung gefährden. Da aber die Stoßströme nicht besonders hoch sind, kann unsere Steuerung in diesem Modus kapazitive Lasten bis zu 1 kW steuern.

2. Schwingungspaketsteuerung

Dieses Verfahren ist am aufwendigsten und wird für die direkte Umwandlung von Wechselstrom in Wechselstrom verwendet. Bei der Schwingungs- oder Wellenpaketsteuerung (engl. Integral-cycle Switching) von Wechselspannungsverbrauchern werden keine Halbwellen beschnitten, sondern ganze Halbwellen (vorzugsweise gleich viel positive und negative) oder ganze Perioden übersprungen. Dies geschieht immer genau im Nulldurchgang der Netzspannung/strom, so dass elektromagnetische Störungen, die durch die Schaltvorgänge entstehen, weitgehend eliminiert werden. Die dadurch (im Gegensatz zu Pha-

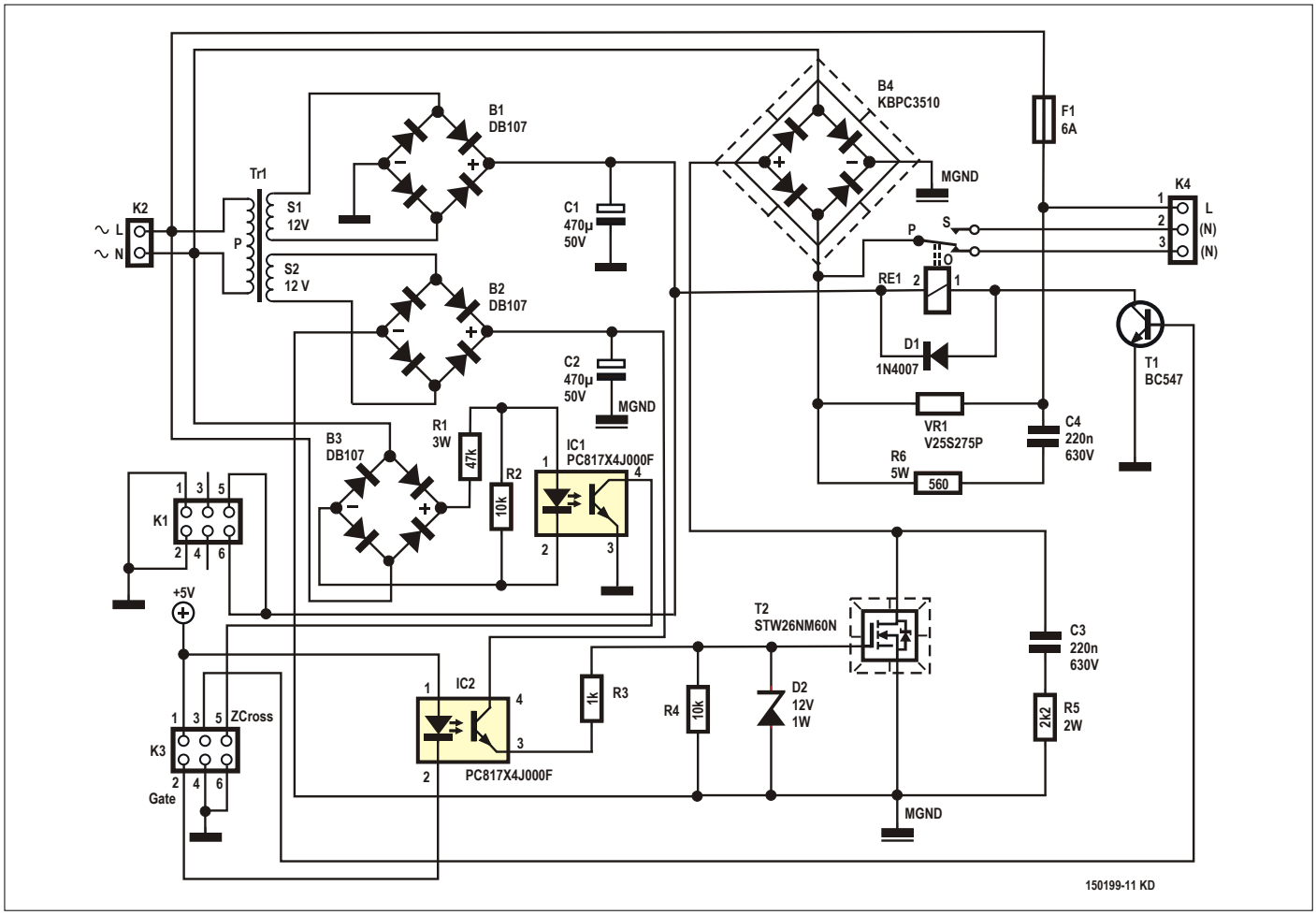


Bild 1. Schaltung der Leistungselektronik. Es gibt zwei Massepotentiale, GND mit dem gewohnten Symbol und die Motortreiber-Masse MGND mit unterschiedlicher Kennzeichnung. Zugunsten der elektrischen Sicherheit sollten beide Massen GND und MGND getrennt gehalten werden.

Mit Vorsicht zu genießen: Phasenanschnittdimmer

Die meisten und billigsten Lampendimmer verwenden Triacs, die zu einem bestimmten, vom Nulldurchgang der Netzwechselspannung entfernten Zeitpunkt innerhalb einer Halbwelle „gezündet" werden und damit den Strom einschalten. Nach dem Einschalten bleibt ein Triac (auch ohne Gatespannung) eingeschaltet, bis der Strom durch ihn unter einen Mindestwert fällt. Erst wenn dieser Haltestrom kurz vor dem nächsten Nulldurchgang unterschritten wird, schaltet der Triac ab und leitet nicht mehr. Da ein Triac bidirektional arbeitet, gilt dieses Verhalten für beide Halbwellen. Je später der Zündzeitpunkt liegt, desto weniger der Halbwelle erreicht die Last und desto geringer ist deren Leistungsaufnahme.

Dimmer, die nach diesem Prinzip arbeiten, werden Phasenanschnittdimmer genannt, da sie die Halbwelle „von vorne" anschneiden. Das Hauptproblem dieser Triac-Dimmer ist es, dass durch das sehr schnelle Einschalten des Triacs ein sehr starker Stromstoß in die Last fließt, der zu Schäden an Kondensatoren und anderen Bauteilen führen kann. Aus diesem Grund sollten dimmbare elektronische Lampen (sowohl Energiespar- als auch LED-Lampen) nicht mit Triac-Phasenanschnittdimmern eingestellt werden.

Fast alle Dimmer, die für den Hausgebrauch angeboten werden, sind nur 2-Draht-Dimmer, und das gilt fast zu 100% für Triac-Dimmer. Der Dimmer belastet nicht nur die Elektronik in der Lampe, sondern verliert auch seine Referenz, wenn eine elektronische Last verwendet wird. Das macht ihn ungeeignet für die meisten elektronischen Verbraucher, auch wenn wir die Probleme mit den Stromstößen ignorieren. Phasenanschnittdimmer sollten deshalb nur für induktiven Lasten wie ferromagnetische Transformatoren oder (Lüfter-)Motoren verwendet werden.

senschnittverfahren) nicht verursachte elektrische Belastung der schaltenden Bauteile erhöht die Zuverlässigkeit.

Ob resistiv, induktiv oder kapazitiv, beinahe alle Arten elektrischer Lasten akzeptieren dieses Verfahren. Die Schwingungspaketsteuerung wird häufig verwendet, um träge elektrische Verbraucher mit hoher Leistung zu steuern, deren Ansprech-Zeitkonstante viel länger ist als die Perioden der Netzspannung, die diese Verbraucher versorgt. Ein Beispiel ist ein elektrisches Heizelement in einer Küche, dessen thermische Zeitkonstante in der Größenordnung einer Sekunde oder mehr liegen kann. Im Modus Schwingungspaketsteuerung kann unsere Schaltung Lasten bis zu 1 kW steuern.

Die Leistungselektronik

Der Schaltplan der Leistungselektronik des Projekts ist in **Bild 1** dargestellt. Tr1 ist ein Netztrafo mit zwei unabhängigen 12-V-Sekundärwicklungen. Die Spannung einer Wicklung (S1) wird

vom Brückengleichrichter B1 (DB107) gleichgerichtet und anschließend für die +5-V-Versorgung der Mikrocontrollerschaltung herangezogen. Die andere Sekundärwicklung speist den Brückengleichrichter B2 (ebenfalls ein DB107) und wird für die Leistungselektronik verwendet, die durch den Optokoppler IC2 (PC817X4J000F) von der Controllerschaltung elektrisch isoliert ist. Dadurch werden nicht nur Rauschen auf und eine zu hohe Belastung der Controller-Versorgungsspannung vermieden. Auch das vom Controller erzeugte Gate-Steuersignal wird durch den Optokoppler vom Leistungstreiber getrennt.

Brückengleichrichter B3 und Optokoppler IC1 detektieren den Nulldurchgang der Netzwechselspannung und geben diese Information zur intelligenten Verarbeitung an den Mikrocontroller weiter. IC1 hält auch das Netzspannungspotential (elektrisch gesehen) weit von der Controllerabteilung entfernt.

Der eigentliche Lastregler besteht aus dem Hochleistungs-MOSFET T2 (STW26NM60N) und dem Leistungsbrückengleichrichter B4 (KBPC3510). Der MOSFET ist über die (Gleichspannungs-) Anschlüsse +VE- und -VE des Brückengleichrichters B4 geschaltet, dessen Wechselspannungsanschlüsse mit dem Neutralleiter (N an K2) und über die Last und F1 mit dem Außenleiter (L an K2) verbunden sind. Diese clevere Anordnung ermöglicht es, die Netzspannung in jedem Phasenwinkel zu trennen oder zu schalten, ohne dass für jede Halbwelle ein MOSFET erforderlich wäre. Das nennt man angewandte Schaltungsökonomie! Obwohl ein Leistungs-MOSFET mit niedrigem RDS-on-Widerstand gewählt wurde, beträgt die Worst-Case-Verlustleistung immer noch etwa 3,5 W. Dies erfordert einen Kühlkörper für den MOSFET, wenn der Treiber bis an die Grenzen von 1000 W betrieben werden soll. Die Gleichrichterbrücke B4 muss ebenfalls mit einem Kühlkörper ausgestattet werden, da die Verlustleistung im schlimmsten Fall (also wieder bei 1 kW) etwa 10 W beträgt.

Da anstelle des traditionellen Triacs ein Leistungs-MOSFET gewählt wurde, ist ein Betrieb im Phasenabschnittmodus möglich, mit den oben genannten Vorteilen gegenüber dem Phasenanschnitt. Widerstand R5 und Kondensator C3 bilden ein so genanntes Snubber-Netzwerk für den MOSFET T2 und tragen dazu bei, Spannungsspitzen durch Induktivitäten (Gegen-EMK) zu unterdrücken. Der Widerstand R6 und der Kondensator C4 erfüllen die gleiche Aufgabe an der Netzspannungsleitung. Der Metalloxid-Varistor VR1 unterdrückt Spannungstransienten. Der Motor oder jede andere Art von Last wird an Platinenanschlussklemme K4 angeschlossen.

Das Relais RE1 wird vom Controller über Transistor T1 angesteuert. Mit dem Relais kann man die Drehrichtung des Motors wechseln, wenn man an K4 verschiedene Motorwicklungen anschließt. Die träge Sicherung F1 schützt die Treiberschaltung vor zu hohem Lastströmen.

Die Mikrocontroller-Steuerung

Wir wechseln zu **Bild 2**. Der programmierte Mikrocontroller IC2, ein PIC18F45K22-E/P, ist für die Steuerung der Wechselstrom-Ausgangsleistung verantwortlich, indem er entsprechende Signale über den I²C-Bus zur Treiberplatine sendet. Der Controller kann im ISCP-Modus über den Verbinder K2 programmiert werden. Die Versorgungsspannung gelangt von K1 der Treiberplatine über K1 der Controllerplatine zum Spannungsregler IC1, der für stabile +5-V-Verhältnisse auf der Controllerplatine sorgt. Alle Signale zwischen der MCU- und der Leistungstreiberplatine laufen über die Klemme K3.

Der Mikrocontroller empfängt die Signale des Encoders ENC1A sowie dessen Tasters ENC1B (Auswahl der Motor-Drehrich-

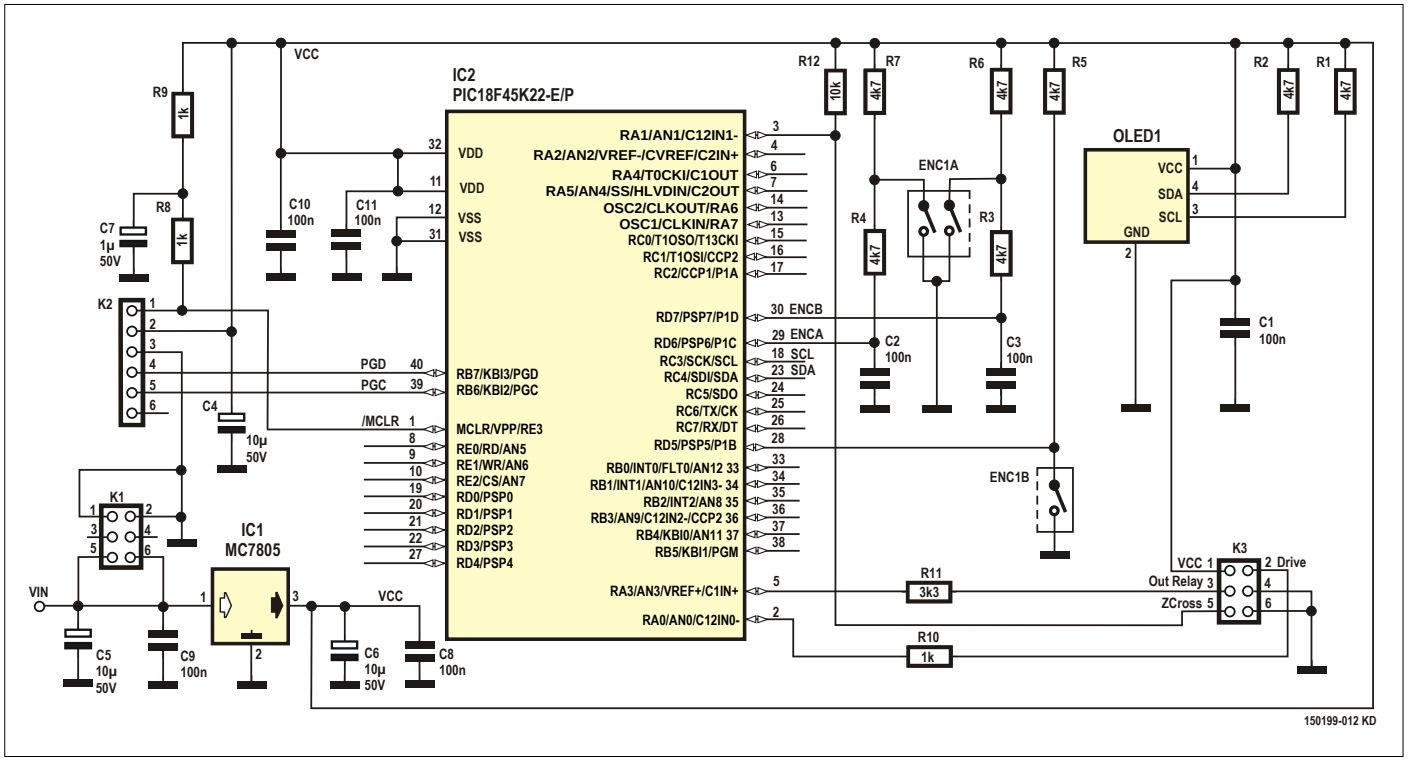


Bild 2. Schaltung der Mikrocontroller-Abteilung mit der Benutzeroberfläche. Zu sehen gibt es einen PIC-Mikrocontroller und eine „Mensch-Maschine-Schnittstelle“ in Form eines Drehgebers und eines OLED-Displays.

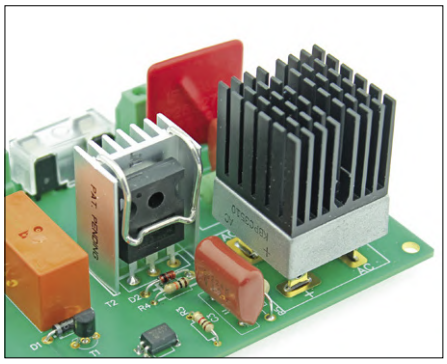


Bild 3. Auf dem Brückengleichrichter B4 wird mit Hilfe einer thermisch leitenden Klebefolie ein Kühlkörper angebracht. Auch MOSFET T2 wird mit einem Kühlkörper geholfen, die Verlustwärme abzuführen. Das keramische Isolationsplättchen ist so eben noch sichtbar.

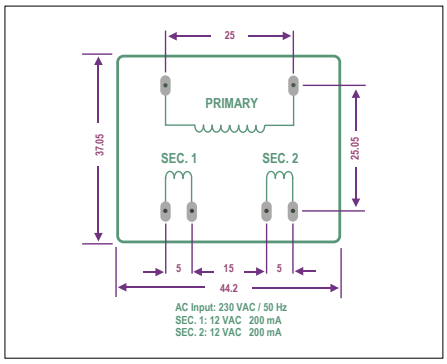


Bild 4. Footprint und Anschlussbelegung des Netztransformators Myrra 44237.

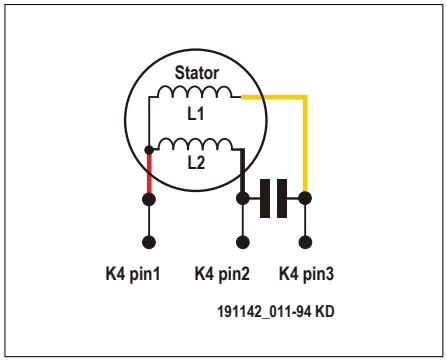


Bild 5. So wird ein AC-Motor an K4 angeschlossen. Lernen Sie Ihren Motor kennen und identifizieren Sie Haupt- und Hilfswicklung sowie den Phasenschieberkondensator.

tung und eine der drei Betriebsarten, in dem der Motortreiber arbeiten soll) und steuert das 0,96-Zoll-Display OLED1 über den I²C-Bus an. Im Display wird die an den Motor abgegebene Leistung in Ziffern und als Balkendiagramm, die Drehrichtung und der Betriebsmodus angezeigt.

Zeit für Software!

Die Software für den Mikrocontroller PIC18F45K22 wurde mit dem CCS-Compiler von Microchip geschrieben, der kostenlos von [1] heruntergeladen werden kann. Der Controller arbeitet mit einer internen Oszillatorfrequenz von 64 MHz, so dass der Core-Takt 16 MHz beträgt. Beim Einschalten ist auf dem OLED-Display eine Startmeldung zu sehen, während der Controller alle Systemparameter und Einstellungen liest.

Bei der Abtastung der Netzwechselspannung wird der Nulldurchgang auf Interrupt-Basis ermittelt, da der MOSFET je nach gewähltem Modus zu exakten Zeitpunkten ein- und ausgeschaltet werden muss. Der für den Phasenschnitt erforderliche Impuls wird von Timer0 abgeleitet, der beim Phasenabschnitt einen Interrupt am Ende des Impulses und beim Phasenanschnitt einen Interrupt am Anfang des Impulses erzeugt.

Anschließend schaltet der Controller je nach eingestellter AC-Leistung den MOSFET und versorgt die Last mit Strom, der langsam von Null ansteigt, bis die eingestellte Leistung erreicht ist. Eine „Softstart-Routine“ sorgt für einen sanften Start, auch dann, wenn bei laufendem Betrieb die Drehrichtung oder der Modus geändert wird. Die Software lässt keine plötzlichen Änderungen der Ausgangsleistung zu und trägt so dazu bei, Schäden am Motor oder Leistungseinbußen zu vermeiden.

Die Benutzeroberfläche besteht aus dem OLED-Display und einem 24-stufigen Drehgeber mit Drucktaste. Dies ermöglicht eine äußerst einfache Bedienung und eine bequeme Einstellung der abgegebenen Leistung, der Drehrichtung sowie der Wahl des Betriebsmodus. Die bereitgestellte Leistung lässt sich durch Drehen des Encoders zwischen „0“ und „100“ wählen. Ein langer Druck auf den Encoder-Taster wählt das Regelverfahren aus: Schwingungspaket, Phasenabschnitt oder Phasenanschnitt. Ein kurzer Druck auf den Taster kehrt die Motorrichtung um.

Wichtiger Hinweis: Der Phasen**ab**schnittmodus darf niemals bei induktiven Lasten wie ferromagnetischen Trafos oder Motoren verwendet werden. Dies führt zu extrem hohen Strom- und Spannungsspitzen und kann den Leistungstreiber, die Last oder beides beschädigen oder gar zerstören. Aus Sicherheitsgründen gibt die Software für 10 s einen Warnhinweis auf dem Display aus, wenn der Phasenabschnittsmodus gewählt wird. Nur wenn die Software innerhalb dieses Zeitraums einen Tastendruck erkennt, wird tatsächlich der Phasenabschnittmodus aktiviert, ansonsten kehrt das Gerät in den Phasenanschnittmodus zurück.

Bei Umkehr der Drehrichtung wird die Ausgangsleistung zuerst auf Null reduziert, damit der Motor Zeit bekommt, abzubremsen. Ohne diese Maßnahme könnten die Motorwicklung, der Antrieb oder beides beschädigt werden.

Konstruktion

Die Aufteilung der Elektronik auf zwei Platinen hilft, Probleme mit der elektrischen Isolation zu vermeiden. Die Platinen sind zwar miteinander verbunden, aber ausreichend elektrisch isoliert, um den elektrischen Sicherheitsvorschriften Genüge zu leisten.

- Platine für den Leistungstreiber (Elektor-Store 150199-1)
- Controllerplatine (Elektor-Store 150199-2).

Beide Platinen werden aber in einem einzigen geschlossenen, aus transparentem Hartkunststoff gefertigten formschönen Gehäuse untergebracht. Nur ein Netzspannungsstecker, eine Buchse für den Motoranschluss und die Welle des Drehgebers schauen aus dem Gehäuse heraus. Im Foto sind die beiden Netzspannung führenden Anschlüsse im VDE-06320-C13/C14-Stil (bekannt unter der Bezeichnung Kaltgerätestecker) zu sehen. Wir warnen alle Elektronikanfänger, das Projekt ohne die Hilfe oder Aufsicht eines erfahrenen Ingenieurs zu realisieren, einem Fachmann, der mit Elektroinstallation vertraut ist und die Gefahren bei Arbeiten am 230-V-Lichtnetz kennt!

Platine für den Leistungstreiber

Neben der Stückliste sind ein Foto sowie Leiterbahn- und Bestückungsseite der Platine für den Leistungstreiber zu sehen. Diese Platine ist geräumig und zur allgemeinen Freude **einseitig** und ausschließlich mit Bohrungen für **Durchsteckbauteile** versehen. Beginnen Sie die Lötarbeiten mit den „niederen“ Bauteilen wie Widerständen und Dioden. Da R1, R5 und R6 warm werden können, sollte man sie mit 2 mm Abstand zur Platinenoberfläche montieren. Die Bauteilkontur „HS1“ zeigt den speziellen Clamp-on-Kühlkörper für MOSFET T2. Zwischen dem Kühl-

körper und der leitenden metallischen Rückseite des MOSFETs sollte ein keramischer oder ein dicker Glimmerisolator eingesetzt werden. „HS2“ bezeichnet den Kühlkörper auf dem Brückengleichrichter, der mit einem thermisch leitenden Klebepad aufgeklebt wird (**Bild 3**).
Tr1 ist entscheidend für die elektrische Sicherheit des Projekts und darf niemals durch etwas ersetzt werden, was sich nicht auf der Platine befindet. Es handelt sich um einen kleinen, voll gekapselten Netztrafo mit zwei getrennten 12-V-Sekundärwicklungen (á 2x208 mA entsprechend 2x2,5 VA) und einer durchgängigen 230-V-Primärwicklung. Hier ist der Typ 44237 von Myrra angegeben, an dessen Abmessungen und Pinbelegung (**Bild 4**) man leicht erkennen kann, dass es sich um ein Standardmodell handelt, das es auch von vielen anderen Herstellern zu kaufen gibt.
Die Netzspannungsleitungen (L und N) sowie die Motorleitungen werden an stabilen Platinen-Schraubklemmen K2 und K4 mit einem sicheren Pinabstand von 0,3 Zoll (7,62 mm) angeschlossen. Im Interesse Ihres Wohlbefindens und eines lang währenden Vergnügens an zukünftigen Elektor-Projekten sollten Sie hier nicht pfuschen, indem Sie etwa die Kabel direkt auf die Platine löten! Verwenden Sie auf jeden Fall die von uns empfohlenen Schraubklemmen.
Der Halterung für die Sicherung F1 sollte mit einer Kunststoffkappe ausgestattet sein, die die Sicherung samt der Halteranschlüsse vollständig abdeckt.

Mikrocontrollerplatine

Auch hier haben wir es ausschließlich mit Durchsteckbauteilen zu tun. Dies

STÜCKLISTE

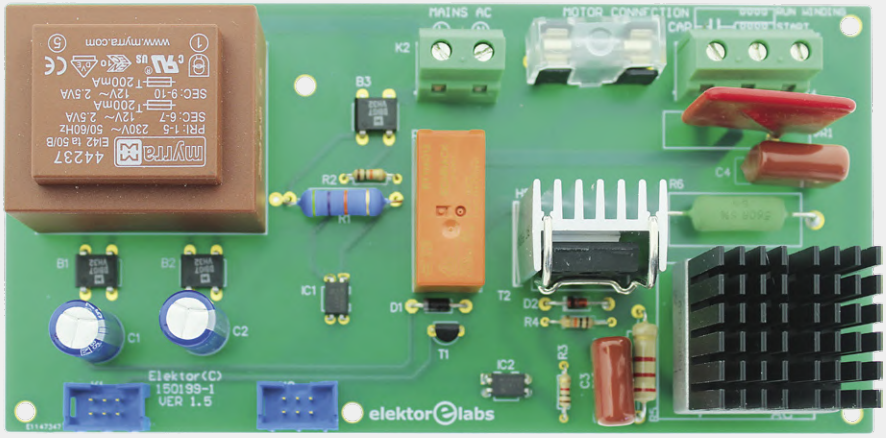
Leistungsteil (150199-1)

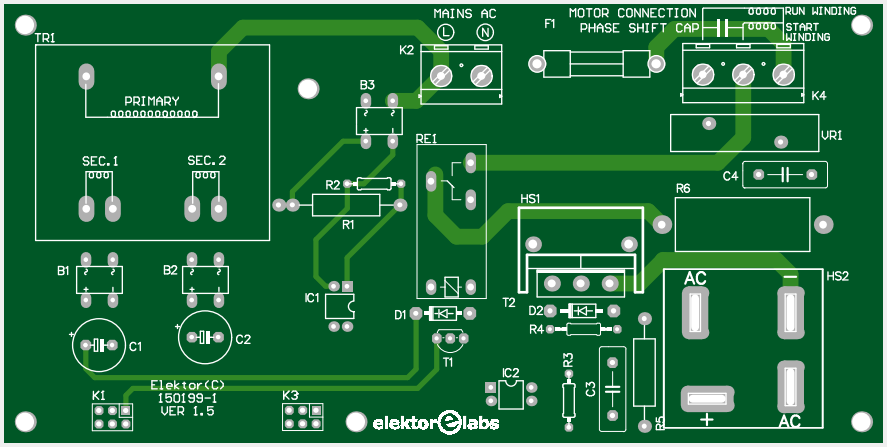
Widerstände:
R1 = 47 k, 5%, 3 W
R2,R4 = 10 k, 5%, 250 mW, 250 V
R3 = 1 k, 5%, 250 mW, 250 V
R5 = 2k2, 5%, 2 W
R6 = 560 Ω, 5%, 5 W

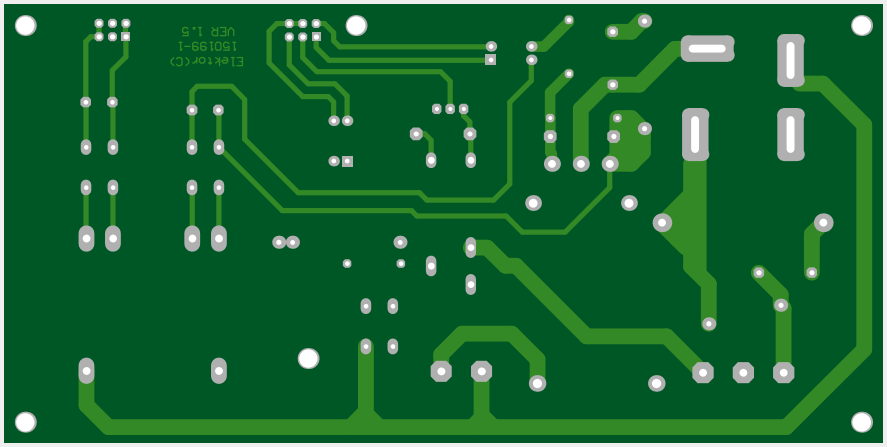
Kondensatoren:
C1,C2 = 470 µ, 50 V
C3,C4 = 0µ22, 630 V_{DC}, MPET

Halbleiter:
B1,B2,B3 = Brückengleichrichter DB107, 1000 V, 1 A
B4 = Brückengleichrichter KBPC3510, 1 kV, 35 A
D1 = 1N4007-T
D2 = Z-Diode 1N4742A, 12 V, 1 W
IC1,IC2 = Optokoppler PC817X3NSZ1B
T1 = BC547B
T2 =N-Kanal MOSFET STW26NM60N, 20 A, 600 V

Außerdem:
K1,K3 = 2x3-polige Stiftleiste (Boxheader)
K2 = 2-polige Platinenanschlussklemme, Raster 7,62 mm (0,3")
K4 = 3-polige Platinenanschlussklemme, Raster 7,62 mm (0,3")
HS1 = TO-247-Kühlkörper WV-T247-101E
HS2 = Kühlkörper Typ 658-60ABT1E (inkl. Klebefolie)
F1 = 6 A träge, 5x20 mm
Sicherungshalter für Platinenmontage, 5x20 mm, mit Abdeckung
RE1 = G2R-14-DC12 (Omron)
TR1 = Netztrafo, 2 x 12V, 200 mA (Myrra 44327, Farnell 1214601
VR1 = Varistor V25S275P (275 V_{AC} 470J, Klemmspannung 700 V, Raster 25 mm)
Platine 150199-1 v.1.5 im Elektor-Store







70% of real size

STÜCKLISTE

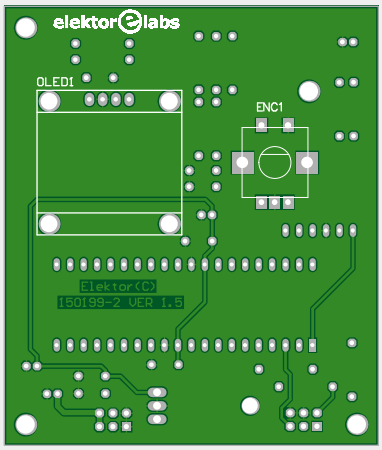
Controllerplatine (150199-2)

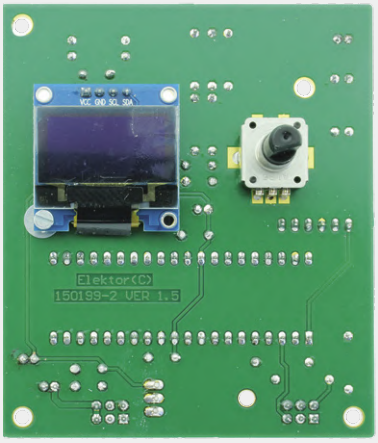
Widerstände:
R1...R7 = 4k7, 5%, 250 mW, 250 V
R8,R9,R10 = 1 k, 5%, 250 mW, 250 V
R11 = 3k3, 5%, 250 mW, 250 V
R12 = 10 k, 5%, 250 mW, 250 V

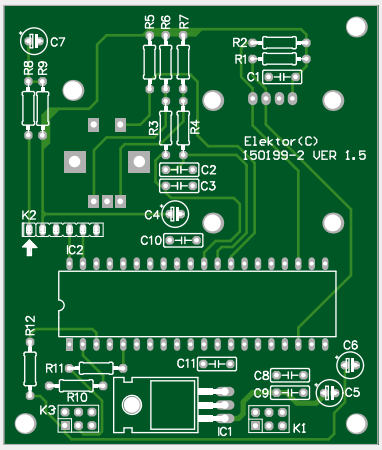
Kondensatoren:
C1,C2,C3,C8...C11 = 0µ1, 50 V
C4,C5,C6 = 10 µ 50 V, Elko
C7 = 1 µF, 50 V, Elko

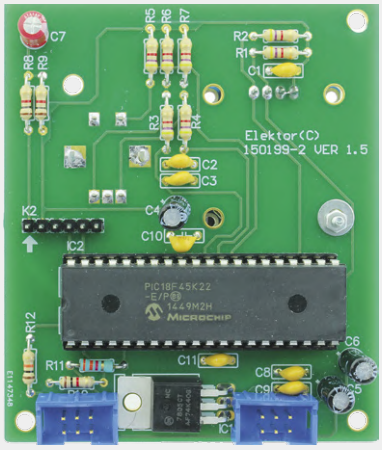
Halbleiter:
IC1 = MC7805CTG
IC2 = PIC18F45K22-E/P (programmiert. 150199-41 im Elektor-Store)

Außerdem:
K1,K3 = 2x3-polige Stiftleiste (Box)
K2 = 1x6-polige Stiftleiste, Raster 0,1"
LCD1 = OLED-Modul (0,96", I²C), (SKU 18747 im Elektor-Store)
ENC1 = Drehgeber mit Drucktaste, Alps EC12E242424407
2x20-polige IC-Fassung (DIP 0,6")
6-poliges Flachkabel und 4 IDC-Verbinder für K1,K3
Platine 150199-2 v. 1.5 im Elektor-Store









70% der wahren Größe

trifft sogar auf den Mikrocontroller zu, dem man eine klassische 2x20-polige DIL-Buchse gönnen sollte. Um das OLED-Display wie im Foto in der Stückliste zu sehen **auf der Platinenrückseite der Platine** zu montieren, setzen Sie zunächst eine 1x4-polige Stiftleiste mit den langen Pins mit dem richtigen Abstand zur Platinenoberfläche ein. Befestigen Sie dann das Display mit vier Schrauben und Muttern, wobei Sie Muttern so als Abstandshalter verwenden, dass die kurzen Pins der Stiftleiste gut lötbar durch die Bohrungen in der Displayplatine schauen. Wenn Sie erst am Schluss das Display festlöten, sitzt alles ohne mechanische Spannungen an Ort und Stelle. Verwenden Sie auf jeden Fall das in der Stückliste genannte Display, es gibt ähnliche Displays, die aber eine andere Anschlussbelegung aufweisen. Montieren Sie auch den Drehgeber auf der Rückseite der Platine.

Weblink

[1] Projektseite: www.elektormagazine.de/191142-03

IM ELEKTOR-STORE

→ **Einstellbarer 1-kW-AC-Motortreiber, Leerplatine Leistungsteil v.1.5**
www.elektor.de/150199-1

→ **Einstellbarer 1-kW-AC-Motortreiber, Controller-Leerplatine v.1.5**
www.elektor.de/150199-2

→ **Einstellbarer 1-kW-AC-Motortreiber, programmierter PIC18F45K22-E/P**
www.elektor.de/150199-41

→ **OLED-Display**
www.elektor.de/blue-0-96-oled-display-i2c-4-pin

Sicherheit geht vor!

Die fertigen und miteinander verbundenen Platinen müssen in dem vollständig isolierten, berührsicheren und transparenten Gehäuse montiert werden, das verhindert, dass man irgendein Teil der Schaltung, mit Ausnahme natürlich der Drehgeberwelle und der Netzstecker, berühren kann. Für den Netzspannungs- ein- und Motorausgang sollte man die vorgeschlagenen Kaltgeräte-Steckverbinder verwenden. Die Gehäuseausbrüche müssen etwas „Spiel“ gewähren, damit die Steckverbinder bequem hineinpassen. Stellen Sie diese Steckverbinder niemals selbst her und denken Sie nicht mal im Traum daran, mit etwas aus der Bastelkiste zu „improvisieren“. Neue Stecker kosten nicht viel und bieten Sicherheit. Verwenden Sie ausschließlich zugelassene Netzspannungskabel, um das Gerät

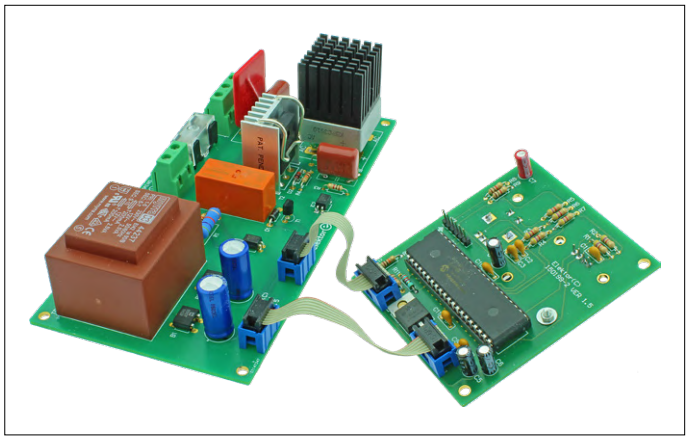


Bild 6. Die Controllerplatine und die Leistungselektronik sind mit zwei 6-poligen Flachbandkabeln verbunden.

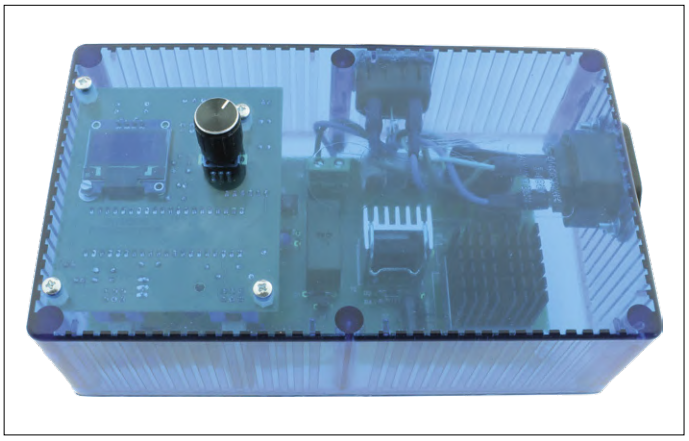


Bild 7. Geprüft, zugelassen und in seinem blau-transparenten Gehäuse gut geschützt: der 1-kW-AC-Motortreiber mit seinen drei Betriebsarten.

mit Strom zu versorgen und den Motor (oder eine andere Last) anzuschließen. Die Motortreiber- und die MCU-Platine müssen in dem Kunststoffgehäuse elektrisch voneinander isoliert befestigt werden. In unserem Prototyp haben wir deshalb Abstandshalter aus Nylon verwendet, mit Innengewinde an einem und Außengewinde am anderen Ende. So ist eine Isolierung gewährleistet, auch wenn man die Abstandshalter mit Metallschrauben an der Gehäusewand befestigt. Überprüfen Sie zum Abschluss Ihre Konstruktion gründlich und bitten Sie bei Bedarf einen Experten um Rat.

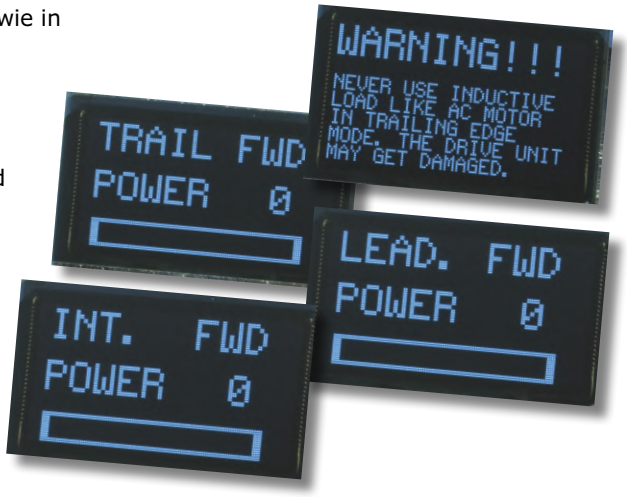
Prüfung

Ist das Gerät nicht im Gehäuse eingebaut und intern mit den Netzspannungsverbindern verkabelt, sollte es nur von einem erfahrenen Techniker mit einem zugelassenen, vollständig isolierenden Trenntransformator geprüft werden. Als Last kann ein 200...500 W starker 230-V-Einphasen-Wechselstrommotor zum Einsatz kommen. Verwenden Sie ausschließlich geeignete Kabel! Gehen Sie bei der Prüfung des Geräts auf Ihrer aufgeräumten (!) und sauberen (!!) Werkbank wie folgt vor (und brechen Sie den Test ab, wenn einer der Schritte zu falschen oder unklaren Ergebnissen führt):

- Verbinden Sie die Platinen mit den beiden 6-adrigen, mit passenden Scheidklemmverbindern ausgestatteten Flachbandkabeln (**Bild 6**).
- Ziehen Sie (wenn eingesteckt) den Mikrocontroller aus seiner Fassung.
- Schließen Sie die isolierte (!) 230-V_{AC}-Versorgung so an, dass sie die Klemme K2 auf der Leistungstreiber-Platine erreicht.
- Überprüfen Sie auf der Controllerplatine die Spannung an Pin 1 des Festspannungsreglers IC1 (Eingang des 7805). Sie sollte 14...16 V_{DC} betragen. An Pin 3, dem Ausgang des 7805, sollte 5 V_{DC} zu messen sein.
- Überprüfen Sie auf der Leistungstreiber-Platine die Spannung an den Anschlüssen „+“ und „-“ des Gleichrichters B2. Sie sollte 14...16 V_{DC} betragen.
- Schalten Sie die isolierte 230-V_{AC}-Versorgung wieder ab und setzen den Mikrocontroller (richtig herum) wieder in seine Fassung.
- Überspringen Sie diesen Schritt, wenn Sie einen vorprogrammierten Mikrocontroller von Elektor erhalten haben. Andernfalls programmieren Sie den PIC mit der bei [1] erhältlichen Hex-Datei mit einem PIC-Programmer wie dem PICKIT3. Die Stromversorgung muss beim Programmieren eingeschaltet sein. Nach dem Programmiervorgang schalten Sie die Versorgungsspannung wieder ab und ziehen den ICSP-Stecker des Programmers von der Controllerplatine.
- Schließen Sie den AC-Motor wie in **Bild 5** dargestellt an K4 an.
- Schließen Sie die isolierte 230-V_{AC}-Versorgung an. Auf dem OLED-Display erscheint die Begrüßungsnachricht und bietet als erste Betriebsart „Int.“ an, was für Schwingungspaketsteuerung steht. Die Motorleistung sollte 0% betragen. Die Motordrehrichtung wird im Display rechts oben angezeigt.
- Wenn der Motor unbelastet heftig dreht oder nicht von seiner normalen Halterung fixiert wird, sollte man Vorkehrungen gegen Rucken, Schläge und Vibrationen treffen.
- Drehen Sie am Encoder, um die Motorleistung bis auf 100% zu steigern. Ein kurzer Druck auf die Encoderwelle sollte den Motor auf Null abbremsen und die Drehrichtung anschließend umkehren.
- Trennen Sie den Motor von der Schaltung.
- Überprüfen Sie die Betriebsarten des Geräts durch langes Drücken des Encoders. Das Display sollte zeigen: [Int. Lead. Trail.] (Schwingungspaketsteuerung, Phasenanschnitt, Phasenabschnitt).

Herzlichen Glückwunsch, wenn alles klappt. Bauen Sie das Gerät endgültig zusammen und schließen Sie die gesamte innere Verkabelung wieder an. Wir haben es im Elektor-Labor genauso gemacht und dann das Foto in **Bild 7** geschossen. ⏮

191142-03



Schnelles 3,5"-Touch-Display für RPi

Mehr Leistung ohne Aufpreis

Von **Mathias Claußen** (Elektor-Labor)

Das 3,5“-Touchscreen-LCD von Waveshare ist nicht nur günstig, sondern dank seiner 50 fps eine besondere Alternative zu bisherigen kleinen Displays für den Raspberry Pi. Es ist genau da ideal, wo andere Displays dieser Preisklasse wegen geringer Frameraten nur eine Slide-Show liefern würden.

Waveshare hat seine Produktpalette aktualisiert und ein neues 3,5“-Display für den Raspberry Pi vorgestellt. Die Anbindung der meisten preiswerten RPi-Displays erfolgt seriell per SPI. Die maximale Taktfrequenz liegt dann bei 20 MHz und gelegentlich bei 26 MHz für das Schreiben von Daten. Das begrenzt die Update-Rate des Bildschirms deutlich.

Geschwindigkeit

Soll ein kompletter Bildschirminhalt an das Display übertragen werden, fallen 480 * 320 Pixel bei 16 Bit für die Farbe pro Pixel an. Insgesamt müssen also 307,2 kB für den Aufbau eines komplett neuen Bildes transferiert werden. Hinzu kommt noch für jeden Transfer ein Overhead von ein paar Bytes. Unter idealen Bedingungen benötigt dies bei 20 MHz SPI-Takt immerhin 125 ms Transferzeit, was nur für knapp acht Bilder pro Sekunde ausreichen würde, wenn das Display permanent neu beschrieben wird. Für ein Video ist das nicht mehr akzeptabel und selbst für die Bedienung einiger Programme nicht sehr angenehm, da die träge Bildschirmausgabe das genaue Anklicken von Schaltflächen erschwert - von Spielen, die eine schnelle Reaktion benötigen, oder der erschwerten Positionierung einer Videokamera erst gar nicht zu reden.

Doch wenn man nach einem schnelleren Display sucht, das einfach auf einen RPi gesteckt werden kann, dann wird man nun fündig. Das neue 3,5“-Display von Waveshare bietet nämlich die üblichen 480 * 320 Pixel mit 16 Bit Farbe bei einem besonders hohen SPI-Takt von 125 MHz! Da der Overhead relativ klein ausfällt, kann man die resultierenden 302,7 kB pro Bild flotte 50 Mal pro Sekunde übertragen. Das ermöglicht einen wirklich flüssigen Bildaufbau und macht das Display daher ideal für Video, Spiele oder die ruckelfreie Bedienung eines Browsers.

Unterschiede

Oberflächlich betrachtet gleicht das neue Display für RPi von Waveshare (**Bild 1**) den konventionellen Exemplaren anderer Hersteller. Doch ein Blick auf die Rückseite (**Bild 2**) verrät elementare Unterschiede: Es sind nur noch zwei ICs, ein Spannungsregler und ein paar Kondensatoren verbaut. Normalerweise kann man auf der Display-Rückseite (**Bild 3**) ein

Bild 1. Beim neuen Display von Waveshare ist kaum ein Unterschied zu „normalen“ Displays anderer Hersteller zu erkennen.

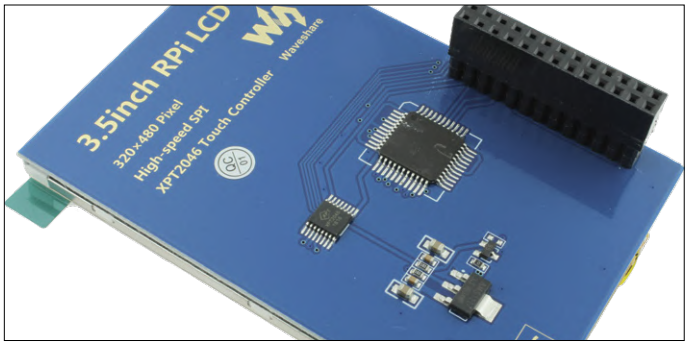


Bild 2. Rückseite des Waveshare-Displays. Es sind nur zwei ICs, ein Spannungsregler und ein paar Kondensatoren zu sehen.

Weblinks

[1] Display-Treiber: www.waveshare.com/w/upload/1/1c/LCD-show-181201.tar.gz

[2] Film „Big Buck Bunny“: http://bbb3d.renderfarming.net/download.html

[3] „Big Buck Bunny“ und Doom auf dem Waveshare-Display: www.elektormagazine.de/190271-01

Schieberegister (74HC4094), einen Zähler (74HC4040) und einen Hex-Inverter (74HC04) finden, die als SPI/Parallel-Konverter fungieren. Diese ICs wurden also durch einen neuen Chip ersetzt, dessen Beschriftung abgeschliffen wurde – um was für ein Stück Silizium es sich handelt, bleibt somit ein Firmengeheimnis. Auf jeden Fall dürfte klar sein, dass dieser Chip den Betrieb mit einem SPI-Takt von 125 MHz ermöglicht. Man könnte nun detektivisch unterwegs sein, einen Logik-Analyzer anschließen und damit die Signale vom IC Richtung Display untersuchen. Diese Art von Reverse Engineering würde aber den Rahmen dieser Besprechung sprengen und möglicherweise juristische Probleme verursachen.

Einrichtung

Genug von den technischen Details – jetzt kommt ein kurzer Test in der Praxis: Hierzu wird zunächst das Display auf den RPi gesteckt sowie Tastatur, Maus und Monitor angeschlossen. Für die Einrichtung wird ein aktuelles Raspbian mit Updates verwendet und für aktuelle Firmware gesorgt. Den Display-Treiber findet man auf der Webseite von Waveshare. Die aktuelle Version kann man unter dem Direkt-Link [1] herunterladen. In den Disziplinen Terminal oder Konsole Geübte können die Datei auch per *wget* auf den RPi bekommen oder aber einen Download per Browser starten. Die heruntergeladene Datei muss zuerst entpackt werden. Dann muss man den Installer (ein Shell-Script für das Display) ausführbar machen. Am einfachsten geht das über die GUI, in dem man den Inhalt des Archivs in das Homeverzeichnis entpackt, und dort die Datei *LCD35C-show* ausführbar macht. Ein Start des Installers richtet dann alles Nötige für den Betrieb des Displays ein, so dass jetzt auch die Wiedergabe von Videos durch die schnelle Display-Hardware ohne weiteres gelingt.

Für die Installation braucht es einen Internetzugang, damit einige Pakete und Tools nachgeladen werden können. Der Treiber wird via Terminal installiert. Dann wechselt man in den Ordner des gerade entpackten Treibers. Dort macht man den Installer mit dem Befehl `chmod 777 LCD35C-show` ausführbar und führt ihn danach mit `sudo ./LCD35C-show` aus. Wenn das Script zur Einrichtung durchgelaufen ist, startet man den RPi einmal neu.

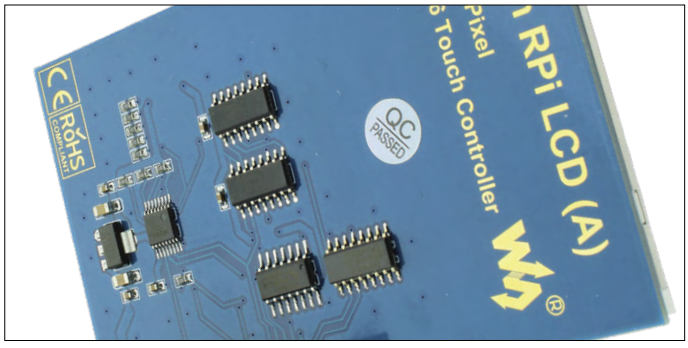


Bild 3. So sieht die Rückseite konventioneller RPi-Displays aus.

Erste Tests

Ist alles wie erwartet gelaufen, sollte kurz nach dem Booten die grafische Ausgabe starten und der Desktop auf dem Display dargestellt werden. Zum Test des Displays „unter Last“ diente das Video „Big Buck Bunny“ [2] in der 1080p-Auflösung mit 30 Bildern pro Sekunde. Abgespielt wird das Video in FullHD (1920 * 1080 Pixel) bei 30 fps per OMXPlayer. Damit Sie das Ergebnis selbst sehen können, haben wir ein Testvideo vorbereitet, das auf unserer Webseite unter [3] zur Verfügung steht. Sie werden sehen, dass das Display keine Probleme mit der Wiedergabe von Videos hat. Auch eine Runde DOOM ist kein Problem. Was bei per SPI angebundenen Displays grundsätzlich problematisch bleibt, ist die Verwendung von OpenGL-Software wie SuperTuxKart oder andere Anwendungen, die von 3D-Beschleuningung Gebrauch machen. Sobald das Overlay für den Treiber *vc4-KMS-overlay.dtb* in */boot/config.txt* angegeben ist, erfolgt die Ausgabe zwangsweise per HDMI.
Die interessanteste Frage dürfte die sein, ob die schnelle Variante von Waveshare teurer ist als die anderen in unserem Shop gelisteten Exemplare. Glücklicherweise ist sie nicht teurer. Den Vorteil hoher Geschwindigkeit erkauft man sich allerdings mit einer etwas geringeren Blickwinkelstabilität und Helligkeit als bei den Displays anderer Hersteller. In Büroumgebungen mit direktem Licht auf das Display kann das relevant sein.

Fazit

Das Resümee fällt kurz und bündig aus: Bei diesem Preis gibt es kaum einen Grund ein anderes Display zu verwenden, das langsamer als die von Waveshare gebotenen 50 Bilder pro Sekunde ist. Das „kaum“ bezieht sich auf den Einsatz unter schwierigen Lichtverhältnissen, denn hier sind dann andere Displays die bessere Wahl. Wir von Elektor würden das neue 3,5“-Waveshare-Display für fast alle Fälle empfehlen, wo ein Display direkt mit einem RPi zu einem kompakten Sandwich verbaut werden soll. Nur wer auf OpenGL angewiesene Anwendungen einsetzt, sollte sich besser anderweitig orientieren, um später keine Überraschungen zu erleben. Hier würde sich ein Blick auf das 5“-Touch-Display von JOY-iT rentieren. Ansonsten gilt: Wieso sollte man bei einem 3,5“-Display auf Tempo verzichten, wenn man diesen Luxus ohne Aufpreis bekommt? ⏮

190271-01

Zutritt für Unbefugte verboten!

Ein Blick ins Allerheiligste aller Elektroniker

Von **Eric Bogers**

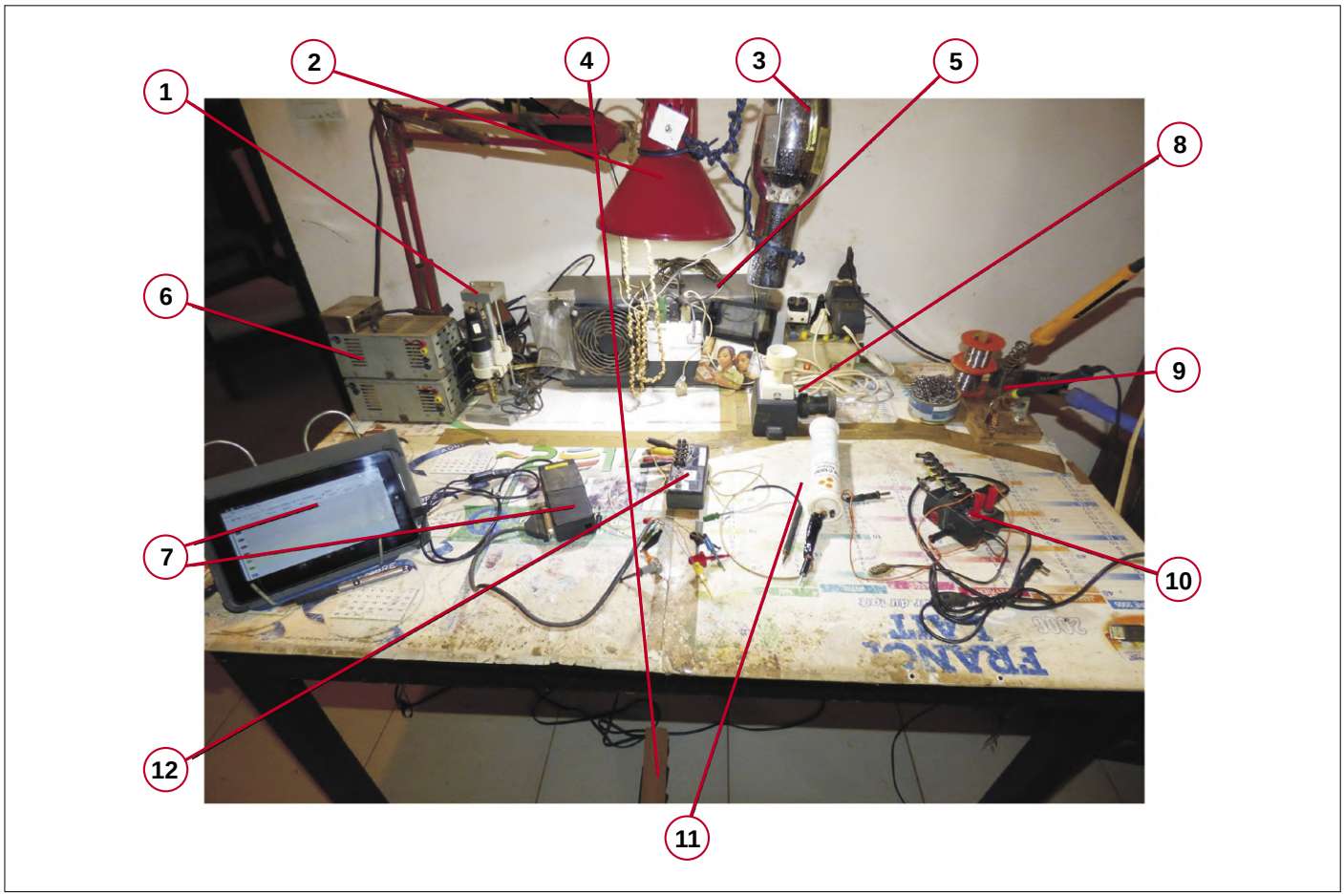
Im Juli 2019 organisierte Elektor Labs einen Wettbewerb, um herauszufinden, wer das schönste/ unordentlichste/ interessanteste/merkwürdigste Heimlabor hat. In den kommenden Elektor-Ausgaben werden wir Ihnen die denkwürdigsten Einsendungen präsentieren.

Wir denken – zu Unrecht - nicht wirklich darüber nach, aber für einen Elektronik-Hobbyisten ist das Leben in (West-)Europa fast paradiesisch. Zwar stirbt der „Elektronikladen um die Ecke“ mit seinen endlosen Reihen kleiner Schubladen, mit jemanden, der zuhörte, diskutierte und gut beriet, aber die großen Versandhäuser liefern schnell und für relativ wenig Geld, was wir brauchen.

Die Situation sieht ganz anders aus für André Aguila, der in Ouagadougou in Burkina Faso lebt und arbeitet. Es ist nicht gerade einfach, dort erschwingliche Werkzeuge oder Bauteile zu finden - und wenn Versandhäuser überhaupt dorthin liefern, ist es teuer und es braucht Zeit. André schreibt:

„Ich habe in meiner Studienzeit die Elektronik als Hobby entdeckt; sie danach für lange Zeit links liegen lassen und erst vor ein paar Jahren den Faden wieder aufgenommen. Ich kann nicht besonders viel Geld für mein Hobby ausgeben, also habe ich viele Werkzeuge und Hilfsmittel selbst hergestellt. Die folgenden Bilder geben einen Einblick.
Im Moment baue ich meinen eigenen Reflow-Ofen, der in einem ausgeschlachteten Gehäuse eines 5¼“-Diskettenlaufwerks untergebracht wird. Zwei Halogenlampen sollen die nötige Wärmeenergie liefern.“

Hier bei Elektor würden wir gerne sehen, wie sich dieses Projekt weiter entwickelt!



Andrés Kommentare zu den Fotos:

1 und 2: Die Mini-Bohrmaschine und die Lampe stammen aus meiner Studienzeit; die Muschelkette war ein Geschenk eines Kommilitonen aus Tahiti.

3: Das ist kein Föhn (mehr) - er saugt jetzt schädliche Dämpfe ab. Der Luftstrom wurde natürlich umgekehrt; auf der Rückseite befindet sich ein Kohlefilter.

4: Mit diesem Pedal schalte ich den Absauger ein und aus.

5: Das ist keine USV oder so, es ist ein Trenntrafo. Tatsächlich handelt es sich um zwei 220/110-V-Transformatoren, deren 110-V-Wicklungen miteinander verbunden sind. Nur einer von ihnen besitzt eine „echte“ galvanische Trennung.

6: Netzteile, ausgeschlachtet aus 5¼“-Diskettenlaufwerken.

7: Preiswerter Tablet-PC mit einem vielbenutzten USB und angeschlossenem Logikanalysator (8 Kanäle mit 24 MHz oder 16 Kanäle mit 12 MHz).

8: Glühlampe in Reihe mit der Netzspannung als Sicherung zum Schutz der angeschlossenen Schaltung im Falle eines Kurzschlusses.

9: Selbstgebauter Ständer für den Lötkolben.

10: Einfacher Multi-Adapter für Testzwecke. Ich habe ihn auf meiner Elektor-Labs-Seite [1] beschrieben.

11: Eines meiner ersten Messgeräte: ein einfacher Durchgangsprüfer, wahrscheinlich ein Elektor-Entwurf.

12: Ein einfacher Transistor-Tester, der 2014 in Elektor [2] beschrieben wurde.

13: Preiswertes USB-Oszilloskop, angeschlossen an ein preiswertes Tablet und von einer Powerbank mit Strom versorgt. Die Android-Software war nicht völlig kostenlos, aber jeden Cent wert, den ich dafür bezahlt habe.

14: Mein Hobby-Laptop, Baujahr 2011! Ich habe eine Menge Elektronik- und Programmiersoftware darauf. Das Bild zeigt den Visual Analyzer als Signalgenerator (leider nur für Audiofrequenzen....).

15: Dieser „Verstärker“ erhöht den Pegel der vom Laptop erzeugten Audiosignale. Damit habe ich unter anderem einen selbstgebauten Class-A-HiFi-Verstärker getestet.

16: Günstige externe USB-Soundkarte.

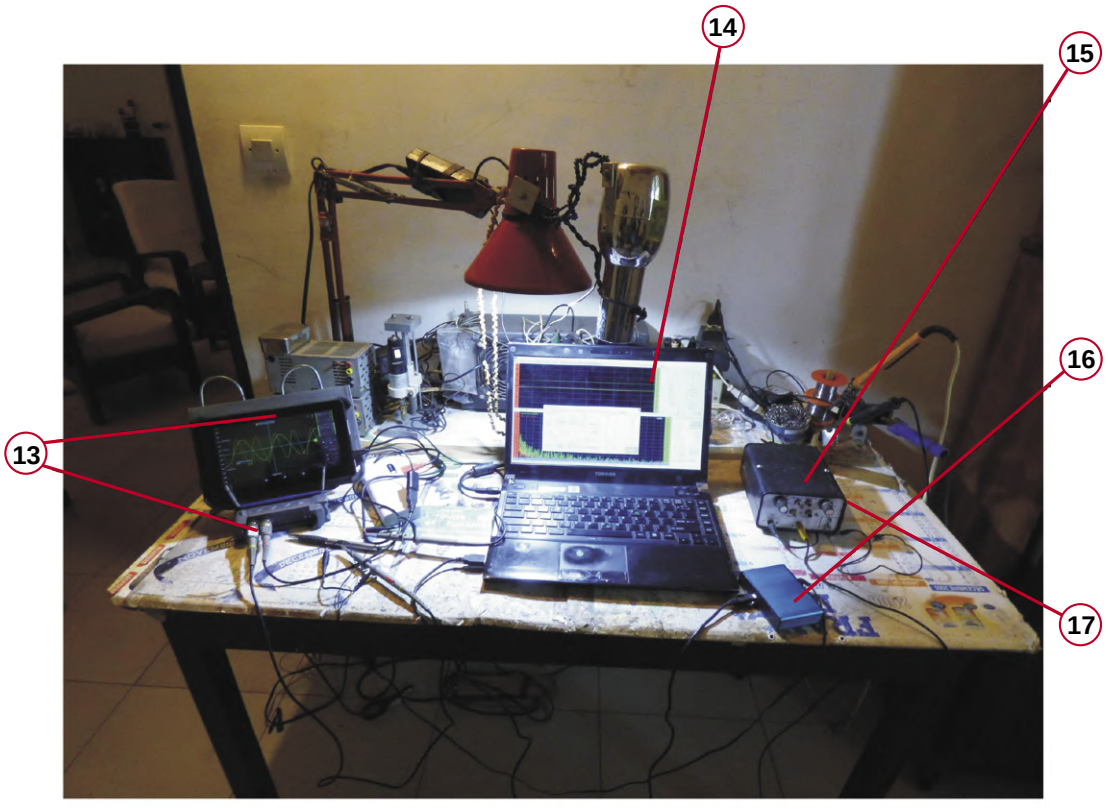
17: Auf der Rückseite dieses Verstärkers befindet sich ein Abschwächer (auch von Elektor), um gegebenenfalls Eingangssignale zu dämpfen. ⏮

191139-04

Weblinks

[1] Multi-Adapter: www.elektormagazine.de/labs/comfortable-and-inexpensive-v-a-controller-civac-1-1

[2] Transistortester: www.elektormagazine.de/magazine/elektor-201801/41214

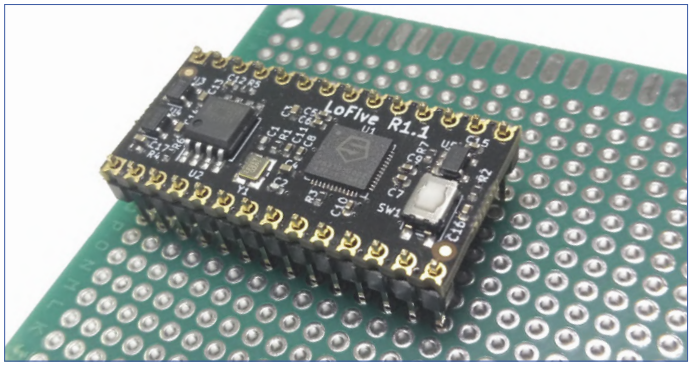


Erste Schritte mit RISC-V

LoFive-Board ausprobiert

Von **Tam Hanna**

Nach ARM könnte RISC-V das nächste große Ding auf dem Prozessormarkt werden. Die von einem großen Konsortium [1] vorangetriebene Befehlssatz-Architektur ist lizenzkostenfrei nutzbar und bietet sich sowohl für Embedded- als auch Computerprozessoren an. Noch gibt es allerdings nur wenige Mikrocontroller und -Boards zu kaufen. Elektor-Autor Tam Hanna hat das günstige „LoFive“-Board ausprobiert.



Während die ARM-Architektur als Ganzes streng patentiert ist, sorgt die permanente Weiterentwicklung der x86-Architektur dafür, dass ein Anbieter eines aktuellen x86-Cores ebenfalls jede Menge Lizenzgebühren bezahlen muss.
Die an der Universität zu Berkeley entwickelte RISC-V-Architektur hat das explizite Ziel, sowohl im Embedded- als auch im „größeren“ Bereich zu wildern. Das mit Abstand wichtigste Argument der Anbieter ist dabei die Lizenzkostenfreiheit - **Bild 1** stammt von der unter [2] bereitstehenden Webseite und informiert, dass die Verwendung der Architektur kostenlos ist.

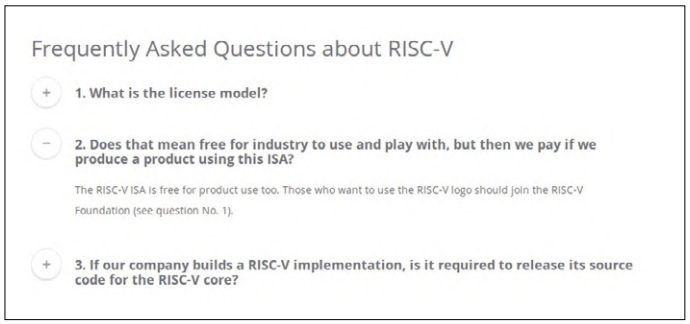


Bild 1. Die RISC-V-Architektur ist kostenfrei verwendbar und bringt keine viral-lästigen Lizenzen mit.

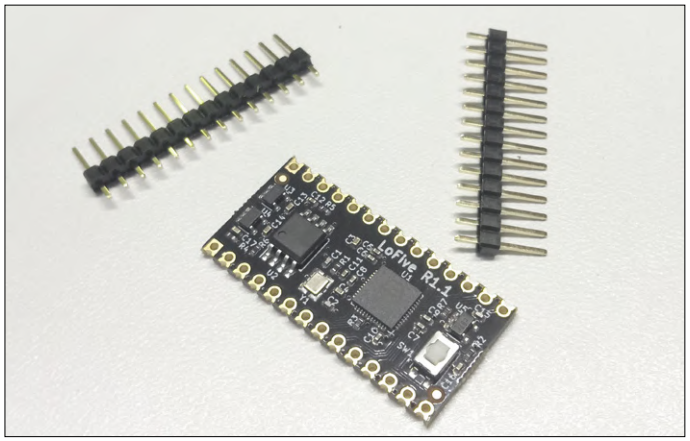


Bild 2. Die Platine lässt sich auch als Surface-Mount-Modul verwenden.

Während man seit vielen Jahren an der RISC-V-Architektur arbeitet, ist reale Hardware erst seit vergleichsweise kurzer Zeit verfügbar. Über GroupGets ist ein günstiges Evalboard erhältlich, das auf den Namen LoFive hört und einen RISC-V-Prozessor Freedom E310 des Herstellers SiFive enthält [3].
Ob der rapiden Weiterentwicklung der Architektur gibt es mittlerweile zwei Versionen davon - wir wollen hier mit der aktuelleren Variante arbeiten, die auf den Namen R1 bzw 1.1 hört. Beachten Sie, dass sich die ältere Version technisch stark von von ihrem Nachfolger unterscheidet, und sich die hier besprochenen Anweisungen nicht 1:1 umsetzen lassen.

Vorbereitungshandlungen

Wer den LoFive auspackt, sieht - wie in **Bild 2** gezeigt - eine Gruppe exponierter Header. Der Hersteller setzt auf „kombinatorische“ Pinouts, die neben dem Einlöten von 2,54-mm-Headern auch das direkte Verlöten der Platine auf einer anderen Planare ermöglichen. Der Autor wird in den folgenden Schritten - schon aus Gründen der Bequemlichkeit - mit einer klassischen Steckplatine arbeiten.

Das LoFive-Board ist aktuell für rund 25 € erhältlich, dies wird unter anderem dadurch erreicht, dass der Anbieter auf die Integration eines Programmier-Chips verzichtet. Sie müssen stattdessen ein USB/Seriell-Adapterboard FT2232H-56Q aus dem Hause FTDI beschaffen; die korrekten Verbindungen zwischen den Platinen sind in der **Tabelle 1** zusammengefasst.
Wer sich ernsthaft mit dem LoFive auseinandersetzt, sollte an dieser Stelle ein „Trägerboard“ bauen. Für erste Experimente reicht es allerdings aus, Dupont-Kabel zu verwenden (**Bild 3**). Im EMI-technisch ob des Vorhandenseins vieler LED-Lampen eher unruhigen Labor des Autors funktionierten zehn Zentimeter lange Drähte den Gutteil der Zeit problemlos, manchmal auftretende Probleme bei der Erkennung des Ziels waren beim nächsten Durchlauf verschwunden.

Tabelle 1. Verbindung von LoFive und Programmieradapter.	
LoFive Pin	**FTDI Breakout Pin**
+5Vin	VBS
GND	GND
TRSTN	AD5
TCK	AD0
TDO	AD2
TMS	AD3
TDI	AD1
UART0.TX	BD1
UART0.RX	BD0

Beginn der Entwicklungsarbeiten

Auf der mit Ubuntu 18.04 laufenden Workstation des Autors waren die meisten der benötigten Elemente schon an Bord. Geben Sie das folgende Kommando ein, um die Pakete zu laden:

```
sudo apt-get install autoconf automake libmpc-dev
   libmpfr-dev libgmp-dev gawk bison flex texinfo
   libtool libusb-1.0-0-dev make g++ pkg-config
   libexpat1-dev zlib1g-dev
```

Windows- und Mac-OS-Anwender schauen an dieser Stelle übrigens in die Röhre. Linux hat sich in der Welt der RISC-V-Entwicklung mehr oder weniger als Standard durchgesetzt; insbesondere bei der Arbeit mit wenig verbreiteten Boards hat man mit Windows nur wenig Chancen. Das Eclipse-basierte Freedom Studio, vom Prozessorhersteller SiFive entwickelt, funktioniert mit dem „offiziellen“ HiFive-Entwicklungsboard. Der LoFive findet in der IDE zum Zeitpunkt der Drucklegung dieses Hefts keine Liebe.
Im nächsten Schritt können Sie jedenfalls das Herunterladen des SDKs über den bekannten git-Client befehligen:

```
tamhan@TAMHAN18:~$ cd riscvneu/
tamhan@TAMHAN18:~/riscvneu$ git clone --recursive
   https://github.com/mwelling/freedom-e-sdk.git
. . .
tamhan@TAMHAN18:~/riscvneu$ cd freedom-e-sdk
tamhan@TAMHAN18:~/riscvneu/freedom-e-sdk$ git
   checkout lofive-r1
M            freedom-devicetree-tools
. . .
```

Neuartig ist hier vor allem, dass wir nach dem Herunterladen des Haupt-Archivs noch ein Submodul laden müssen. Es ist für die Bereitstellung der LoFive-R1-spezifischen Module erforderlich.
Im nächsten Schritt müssen wir Github eine Synchronisation befehlen, um danach ein weiteres Untermodul herunterzuladen:

```
tamhan@TAMHAN18:~/riscvneu/freedom-e-sdk$ git
   submodule sync
Synchronizing submodule url for 'doc/html'
. . .
tamhan@TAMHAN18:~/riscvneu/freedom-e-sdk$ git
   submodule update --init --recursive
. . .
Submodule path 'freedom-devicetree-tools': checked
   out '4f25a7512696f0b41c17e517d39d097499b931a7'
```

Der Aufruf von `sync` ist dabei nicht unbedingt erforderlich - in Tests des Autors kam es allerdings zu Fehlermeldungen, weshalb im Zweifelsfall eine „vorsichtigere“ Vorgehensweise vernünftig ist.
Die RISC-V-Toolchain setzt im Allgemeinen auf den GCC-Compiler und OpenOCD; einige andere Evaluationsboards verwenden stattdessen ein Seggersches Programmier-Interface.
Da die Kompilation der Toolchain auf der Workstation vergleichsweise viel Zeit in Anspruch nimmt, stellt SiFive mittlerweile ein mehr oder weniger fertiges Paket zur Verfügung. Öffnen Sie im ersten Schritt die URL *https://www.sifive.com/boards* im Browser ihrer Wahl, und scrollen Sie danach - wie in **Bild 4** gezeigt - bis zur Passage *Prebuilt RISC-V GCC Toolchain and Emulator* nach unten.

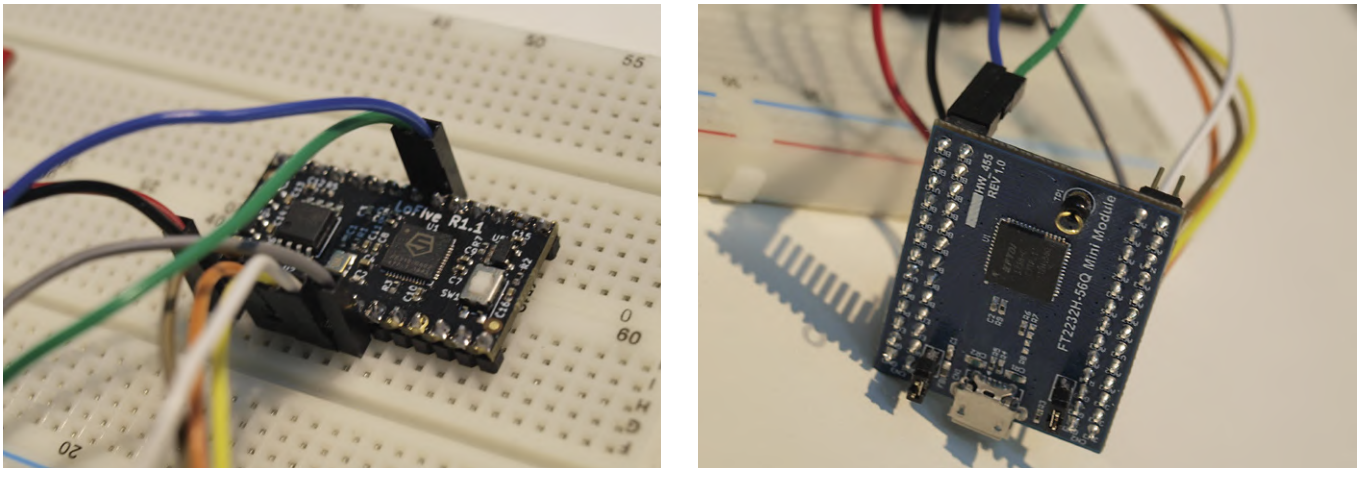


Bild 3. Zum Programmieren wird ein USB/Seriell-Modul von FTDI benutzt.

Für die folgenden Schritte benötigen wir jedenfalls die Versionen *GNU Embedded Toolchain — v2019.08.0* und *OpenOCD — v2019.08.2*. Klicken Sie die beiden Links an, und extrahieren Sie Ihre Inhalte danach in Unterverzeichnisse des Arbeitsverzeichnisses. Der Autor arbeitet in den folgenden Schritten mit dem Verzeichnisnamen *riscvneu*; die Ausgabe des `dir`-Befehls präsentiert sich folgendermaßen:

```
tamhan@TAMHAN18:~/riscvneu/$ dir
freedom-e-sdk/
riscv64-unknown-elf-gcc-8.3.0-2019.08.0-x86_64-linux-
   ubuntu14/
riscv-openocd-0.10.0-2019.08.2-x86_64-linux-ubuntu14/
```

Im Interesse der Bequemlichkeit geht die Toolchain davon aus, Compiler und Co. über Umgebungsvariablen zu finden - der Sinn davon ist, dass das Installieren einer neuen Toolchain durch eine kleine Änderung der Umgebungsvariable effektiv wird. Da Änderungen am PATH-Parameter im Allgemeinen in Arbeit ausarten, wollen wir in den folgenden Schritten mit temporär festgelegten Pfaden arbeiten:

```
tamhan@TAMHAN18:~/riscvneu/freedom-e-sdk$ export
   RISCV_OPENOCD_PATH=/home/tamhan/riscvneu/riscv-
   openocd-0.10.0-2019.08.2-x86_64-linux-ubuntu14/
tamhan@TAMHAN18:~/riscvneu/freedom-e-sdk$ export
   RISCV_PATH=/home/tamhan/riscvneu/riscv64-unknown-
   elf-gcc-8.3.0-2019.08.0-x86_64-linux-ubuntu14/
```

Beachten Sie, dass die Export-Deklaration nur in dem Terminalfenster Gültigkeit hat, in dem sie eingegeben wurde. Möchten Sie ein anderes Fenster verwenden oder schließen Sie das Fenster versehentlich, so müssen Sie den Tanz von vorne beginnen. Dass die abgedruckten Verzeichnis-Werte nur auf der Workstation des Autors gelten, sei ebenfalls angemerkt – es ist sehr unwahrscheinlich, dass Ihr Benutzername ausgesprochen *tamhan* lautet.

Konfiguration von OpenOCD

OpenOCD ist ein vergleichsweise kompliziertes Werkzeug der Linux-Embedded-Programmierung. Es nimmt über verschiedene, auch als *Probes* bezeichnete Systeme Kontakt zu einem Zielsystem auf, um daraufhin über einen Netzwerk-Port auf der Workstation Befehle entgegenzunehmen. Normalerweise tritt OpenOCD „im Team“ mit dem GDB-Debugger auf.
Problematisch ist an OpenOCD, dass die in .conf-Dateien erfolgende Konfiguration sehr „streng“ ausgelegt wird. Da der Entwickler, der die Toolchain an den RISC-V-Prozessor angepasst hat, nicht unter Ubuntu 18.04 arbeitet, kommt es bei der Ausführung mitunter zu Fehlern der folgenden Bauart:

```
tamhan@TAMHAN18:~/riscvspace/freedom-e-sdk$ sudo make
   upload PROGRAM=led_fade BOARD=freedom-e300-lofive
[sudo] password for tamhan:
. . .

Error: unable to open ftdi device with vid 0403, pid
   6010, description 'FT2232H-56Q MiniModule', serial
   '*' at bus location '*'
```

Wenn Ihr System den hier gezeigten Fehler wirft, müssen Sie im ersten Schritt feststellen, welchen *Description*-Text der Kerneltreiber dem angeschlossenen FTDI-Modul zuweist. Dies lässt sich am bequemsten unter Verwendung von `lsusb` bewerkstelligen:

```
tamhan@TAMHAN18:~$ lsusb
. . .
Bus 001 Device 009: ID 0403:6010 Future Technology
   Devices International, Ltd FT2232C Dual USB-UART/
   FIFO IC
```

Im nächsten Schritt müssen wir nach der für Probleme sorgenden Beschreibung suchen. Dies lässt sich am bequemsten durch den Kommandozeilen-Oldie *Grep* bewerkstelligen; die Ausgabe präsentiert sich folgendermaßen:

```
root@TAMHAN18:~/riscvneu/freedom-e-sdk#  grep -r
   FT2232 *
bsp/lofive-r1-bootloader/openocd.cfg:ftdi_device_desc
   "FT2232H-56Q MiniModule"
bsp/lofive-r1/openocd.cfg:ftdi_device_desc "FT2232H-
   56Q MiniModule"
```

Öffnen Sie die beiden Dateien danach mit einem Editor Ihrer Wahl, um den folgenden Deklarationsblock zu finden:

```
interface ftdi
#ftdi_device_desc "Dual RS232-HS"
ftdi_device_desc "FT2232C Dual USB-UART/FIFO IC"
ftdi_vid_pid 0x0403 0x6010
```

Dies informiert OpenOCD im ersten Schritt darüber, dass er es mit einem FTDI-Interface zu tun bekommt. Im nächsten Schritt folgen neben den beiden numerischen IDs auch ein Description-Text, der zusammen das Finden des Zielgeräts ermöglichen soll. Da in unserem Fall die Kombination aus VID und PID ausreicht, können wir die Konfiguration vereinfachen:

```
interface ftdi
ftdi_vid_pid 0x0403 0x6010
```

Und jetzt starten

Ein Problem, mit dem man sich seit dem Aufkommen des ersten Evaluation-Boards herumschlagen muss, ist mit Sicherheit veraltete Firmware. Die ist logisch, als die Boards nicht direkt vom Hersteller auf den Schreibtisch des Entwicklers kommen - sie liegen einige Zeit einem Distributor, wo die enthaltene Software Staub ansetzt. Unsere erste Amtshandlung besteht deshalb darin, den Bootloader zu aktualisieren. Hierzu müssen wir im ersten Schritt die Konfigurationsumgebung bereinigen, da sie Artefakte von der Maschine des Anbieters mitbringt:

```
root@TAMHAN18:~/riscvneu/freedom-e-sdk# make
   PROGRAM=lofive-boot TARGET=lofive-r1-bootloader
   clean
make -C /home/tamhan/riscvneu/freedom-e-sdk/software/
   lofive-boot PORT_DIR= clean
make[1]: Entering directory '/home/tamhan/riscvneu/
   freedom-e-sdk/software/lofive-boot'
. . .
```

Wer über das Freedom-SDK mit einem RISC-V-Board interagiert, tut dies im Allgemeinen über das Make-Werkzeug. Es erwartet neben den beiden Variablen `PROGRAM` und `TARGET` einen Befehl, der ausgeführt wird.
Die eigentliche Auslieferung des Bootloader-Codes erfolgt dann folgendermaßen:

```
root@TAMHAN18:~/riscvneu/freedom-e-sdk# make
   PROGRAM=lofive-boot TARGET=lofive-r1-bootloader
   upload
cd /home/tamhan/riscvneu/freedom-e-sdk/bsp/lofive-r1-
   bootloader/build/debug/ && \
```

Der Autor ruft die diversen Make-Kommandos aus Gründen der Bequemlichkeit aus einer Root-Shell auf. Wer dies nicht möchte, kann die diversen Berechtigungen auch von Hand setzen.
Nach der erfolgreichen Auslieferung des Bootloaders können wir uns mit einem ersten Programm beschäftigen:

```
root@TAMHAN18:~/riscvneu/freedom-e-sdk# make
   PROGRAM=sifive-welcome TARGET=lofive-r1 upload
```

Nach dem Durchlaufen dieses Kommandos sehen Sie an den PWM-Ausgängen eine - zugegebenermaßen langsame - Wellenform. Sofern Sie diese auf ihrem Oszilloskop (Roll-Mode erforderlich bzw. empfehlenswert!) sehen, haben wir die Konfiguration erfolgreich abgeschlossen. Unser Beispielprogramm speit während seiner Abarbeitung zudem Statusinformationen aus, die sich über das Screen-Kommando oder einen beliebigen anderen Terminal-Emulator abernten lassen:

```
tamhan@TAMHAN18:~$ screen /dev/ttyUSB1 115200
[screen is terminating]
tamhan@TAMHAN18:~$
```

Beachten Sie bei der Verwendung dieses Kommandos, dass Sie Screen unbedingt vor dem eigentlichen Programm anwerfen müssen. Das Screen-Terminal lässt sich übrigens nicht ohne weiteres beenden - der einfachste Weg ist das Drücken von *Steuerung* plus *A* und danach der Taste *K*. Screen fragt sodann, ob Sie das Fenster wirklich beenden möchten. Die korrekte Antwort ist hier ein Druck auf *Y*.

Wo liegt der Code?

Zu guter Letzt möchten wir uns in diesem Artikel noch kurz der Frage zuwenden, wo der „zu verarbeitende“ Code liegt. Die Antwort darauf ist das Software-Verzeichnis; im Fall unseres Projektbeispiels präsentiert sich der Inhalt folgendermaßen:

```
root@TAMHAN18:~/riscvneu/freedom-e-sdk/software/
   sifive-welcome# ls
debug  LICENSE  LICENSE.Apache2  LICENSE.MIT
   Makefile  README.md  sifive-welcome.c
```

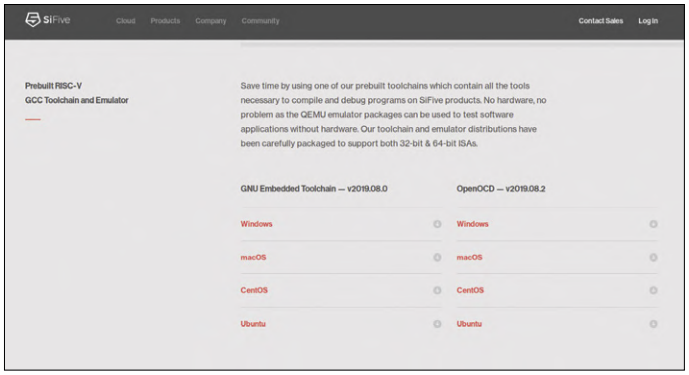


Bild 4. Die Download-Option für die RISC-V-Toolchain ist gut versteckt.

Das für die Steuerung der diversen Kompilationsprozesse verantwortliche Makefile beschränkt sich dabei im Allgemeinen darauf, ein von SiFive vorgegebenes Makefile zu inkludieren – es kümmert sich dann darum, die diversen Kommandos wie Upload zu implementieren.
Schon aus Platzgründen müssen wir unsere Besprechung an dieser Stelle beenden - angemerkt sei nur, dass die Toolchain zum Zeitpunkt der Drucklegung dieses Hefts mit C++ noch nichts anzufangen weiß.

Lohnt es sich?

Wer heute auf die Jagd nach einem preiswerten Mikrocontroller geht, wird bei ST, Microchip und den diversen chinesischen Anbietern mit Sicherheit einfacher fündig als bei RISC-V. Denn auch eine „offene“ Architektur sorgt nicht dafür, dass das Silizium kostenlos ist: Ob der geringen Anzahl an verkauften Exemplaren ist es im Moment sogar so, dass die Skalierungseffekte ARM und andere proprietäre Architekturen unterm Strich günstiger machen.
Die Beschäftigung mit RISC-V lohnt sich schon jetzt für Personen, die entweder „akademisches“ oder hackerisches Interesse an der RISC-V-Plattform haben. Doch schon aufgrund der immens breiten Unterstützung durch viele beteiligte Firmen ist davon auszugehen, dass die Plattform im Embedded-Bereich zumindest Achtungserfolge erzielen kann. ⏮

191138-01

Weblinks

[1] RISC-V Foundation: https://riscv.org/
[2] FAQ zu RISC V: https://riscv.org/faq/
[3] LoFive bei GroupGets: https://groupgets.com/manufacturers/qwerty-embedded-design/products/lofive-risc-v

LoRa-Tracker als Herausforderung

Probleme und Lösungen bei der Elektronik-Entwicklung

Von **Mathias Claußen** (Elektor-Labor)

Ein mit LoRa-Funk ausgestatter, mobiler GPS-Tracker ist eine feine und vielseitig verwendbare Sache, wie aus den vielen Zuschriften von Elektor-Lesern zu folgern ist. Also hat sich das Elektor-Labor vorgenommen, diesen Wunsch zu erfüllen. Lesen Sie hier, welche Schwierigkeiten bei der Entwicklung auftauchten!

Aktuell ist im Elektor-Labor ein LoRa-GPS-Tracker in Arbeit, der schon mehrere Platinen-Iterationen durchlaufen hat und sich langsam der Produktionsreife nähert. Die Idee hinter dem LoRa-Tracker ist recht simpel: Ein mobiles, durch Akku versorgtes Gerät, das in einem definierten Intervall seine Position per LoRa-Funk-WAN sendet. Das Ganze soll dabei in ein Gehäuse 1551K von Hammond passen und von wiederaufladbaren Zellen gespeist sein, die durch den Benutzer austauschbar sind. Batterien scheiden hier zwecks Müllvermeidung aus. Also kommen nur NiMh- oder Lithium-Akkus in Frage.

Versorgung

Die Entscheidung fiel zugunsten austauschbarer Lithium-Akkus im Format 10440 bzw. AAA aus, denn mit ihrer typischen Spannung von etwa 3,6 V kann die Schaltung dann einfach via LDO-Spannungsregler betrieben werden. Da Lithium-Akkus durch eine Tiefentladung geschädigt werden, wurde auch an eine Unterspannungsabschaltung gedacht. Damit kann dann nichts mehr schiefgehen. Die Schaltung ist in **Bild 1** zu sehen.

Sobald die Elektronik anläuft, wird kurzfristig ein erhöhter Strom für das Laden der Kondensatoren hinter dem Spannungsregler fließen. Dieser kann so hoch ausfallen, dass die Spannung des Akkus kurzzeitig auf unter 2,95 V einbricht. Dann würde von IC1 ein „Under Voltage Lockout“ ausgelöst und IC2 abgeschaltet. Abhilfe schaffen hier mit C16 ein 470-µF-Elko parallel zum 100-nF-Kondensator C15 vor dem Spannungsregler, die den kurzfristig erhöhten Strom abpuffern und so die Spannung vor dem Regler über der Schwelle von 3 V halten. Damit wird IC1 nur noch dann aktiv, wenn der Akku wirklich leer ist. Bei der ersten Platine wurde daran noch nicht gedacht. Sie wurde dann mit der Kombination aus C16 und C15 nachgerüstet, indem diese huckepack aufeinander gelötet wurden. Der modifizierte Schaltplan ist in **Bild 2** zu sehen.

Hindernisse …

Anschließend wurde eingeschaltet und die MCU mit einem Debugger verbunden. Sie hat sich auch brav gemeldet. Also war bis hier alles in Ordnung. Jetzt kam das Spannende: Das

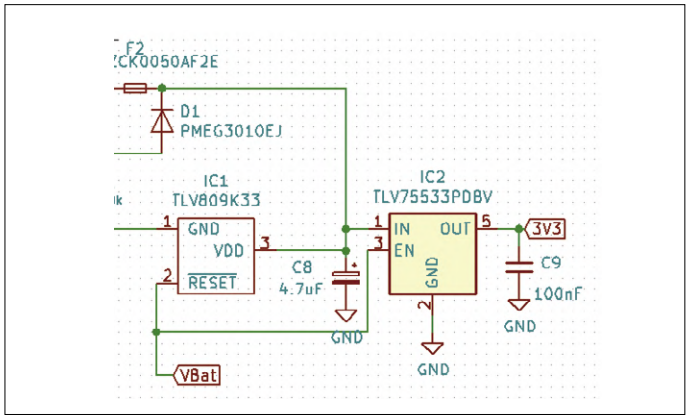


Bild 1. Schaltungsausschnitt mit der Stromversorgung.

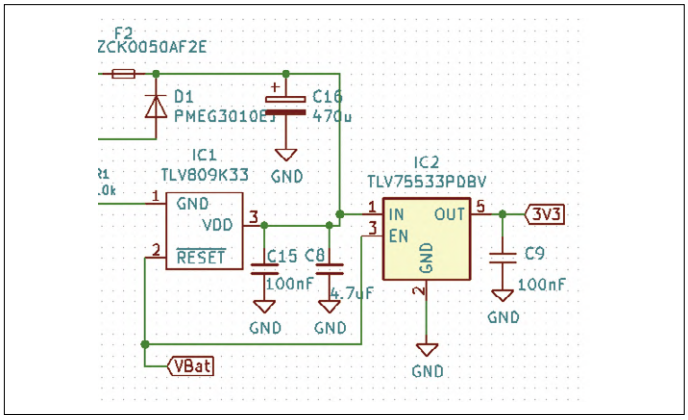


Bild 2. Modifizierte Schaltung mit Puffer-Elko.

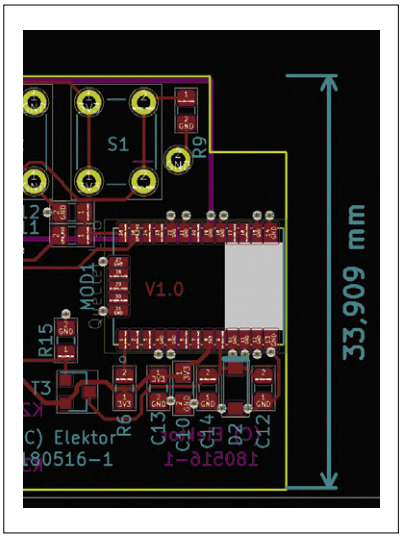


Bild 3. Bei dieser Platinenversion wurde rund um das GPS-Modul zu wenig Platz für einen guten Empfang gelassen.

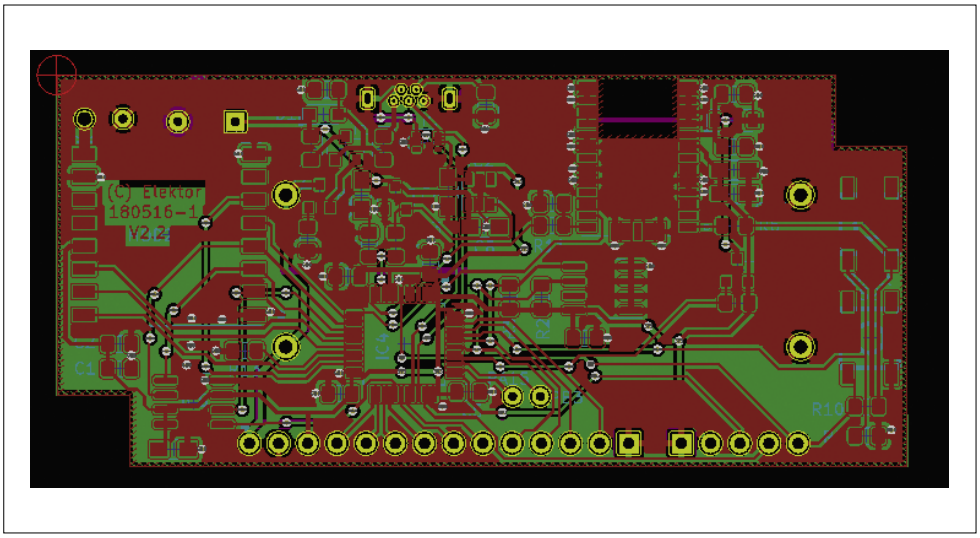


Bild 4. Die andere Position an der Längsseite und die Groundplane ermöglichen bei der verbesserten Platinenversion guten Empfang.

GPS-Modul musste getestet werden. Leider wurde bei der Platine unter der Antenne des Moduls das Kupfer nicht entfernt, so dass der Empfang zunächst nicht möglich war. Nach dem Entfernen des Kupfers solle eigentlich das Modul GPS-Daten empfangen. Doch schon in den Design-Notes wird auf bestimmte Mindestabmessungen verwiesen, die zum Empfang erforderlich sind.

Vorgeschrieben sind mindestens 45 mm Breite + 10 mm links und rechts sowie 20 mm in der Tiefe. Doch wie aus dem Layout der Platine **Bild 3** zu ersehen ist, stehen leider nur 34 mm zur Verfügung. Auch eine durchgehende Groundplane möglichst ohne Bauteile war bei den angestrebten kompakten Abmessungen nicht realisierbar. So war also kein Empfang möglich und eine weiter verbesserte Version der Platine wurde notwendig. Das GPS-Modul selbst ist ja nur 14 * 10 mm groß und somit die On-Board-Antenne nur ein paar Millimeter lang und breit. So schön wie man Halbleiter auch miniaturisieren kann geht das bei Antennen leider nicht. Je kleiner sie sind, desto größer muss das „Gegengewicht“ der Groundplane ausfallen. Bei der verbesserten Platine von **Bild 4** wurden die Abmessungen und die Positionierung so geändert, dass die Vorgaben des Datenblatts eingehalten wurden. Beim erneuten Test hat das Modul dann auch mehr als zwei Satelliten empfangen. Nach den üblichen zehn bis zwölf Minuten war ein GPS-Fix erreicht.

… und Probleme

Für den Test wurde das Modul mit 5 V aus dem USB-Port eines Notebooks gespeist. Das ist schon grundsätzlich keine gute Idee für die Versorgung eines GPS-Empfängers, da über das Kabel auch HF-Reste transportiert werden, die in den Empfänger einkoppeln können und diesen stören. Bei der Versorgung aus einem Labornetzteil hat sich der Empfang weiter verbessert, war allerdings immer noch nicht optimal. Er bot kaum eine Schlechtwetter-Reserve für den Einsatz im Feld. Zusätzlich zeigte sich der Aufbau mit dem verwendeten GPS-Modul nicht ganz einfach. Mit etwas Kopfzerbrechen fanden wir aber eine Lösung, die besser zu löten ist, besseren Empfang bietet und auch noch weniger kostet. Es gab auch noch weitere Überraschungen bezüglich MCU, Software und Mechanik. Mehr darüber in den kommenden Ausgaben und aktuell auf Elektor Labs [3]!

191155-01

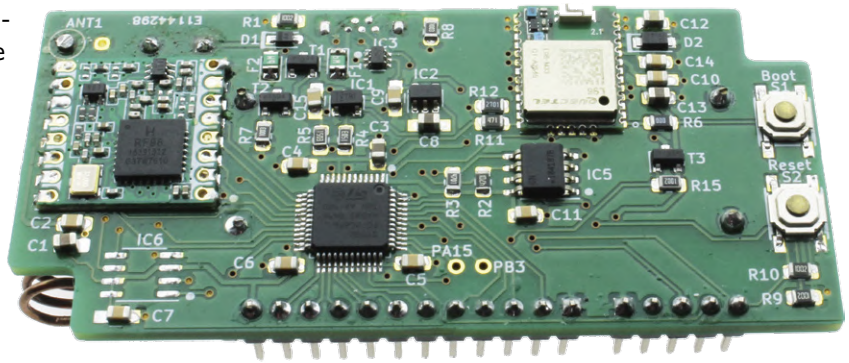


Bild 5. So sieht der getestete Prototyp aus.

Weblinks

[1] GPS Module (L96) Design Guide: www.quectel.com/UploadImage/Downlad/Quectel_L96_Hardware_Design_V1.2.pdf

[2] GPS Module (L96) Datasheet: www.quectel.com/UploadFile/Product/Quectel_L96_GNSS_Specification_V1.1.pdf

[3] www.elektormagazine.de/labs/lorawan-node-experimental-platform

Mit dem Fuchs ins IoT (2)

Anmeldung im Sigfox-Netz

Von **Frank Schleking** und **Bernd vom Berg**

In diesem Teil der kleinen Sigfox-Serie erläutern wir zunächst die grundlegende Struktur des Sigfox-Netzwerkes und den Aufbau des Sigfox-Backends. Dann integrieren wir unser MKR FOX 1200-Board in dieses weltweite Kommunikationsnetzwerk.

Bild 1 zeigt die grundlegende Struktur des Sigfox-Netzwerkes. Die Sigfox-**Objects** (auch Geräte = **Devices** genannt) senden ihre Telegramme als Broadcast über das lizenzfreie 868-MHz-ISM-Band aus. In jedem Telegramm befindet sich neben der Stationsidentifizierung **(Stations-ID)** ein Nutzdatenfeld, **Payload** genannt, von maximal 12 Byte Größe. Das bedeutet, dass der Anwender mit jeder Aussendung maximal 12 Bytes Mess- oder Zustands- oder andere Daten übermitteln kann. Das hört sich zunächst wenig an, da Sigfox aber auf der Feldebene als 0G-Netzwerk seinen Einsatzzweck findet, ist das mehr als ausreichend. So können zum Beispiel zwölf Messwerte á 1 Byte oder drei Messwerte á 2 Byte zusammen mit einem GPS-Datensatz von 6 Byte übermittelt werden.

Da das Sigfox-Netzwerk im lizenzfreien ISM-Band arbeitet, sind aufgrund der gesetzlichen Bestimmungen maximal **nur 140 Aussendungen pro Gerät pro Tag** erlaubt. Das bedeutet, dass unser MKR FOX 1200-Board offiziell alle elf Minuten ein Telegramm senden darf.

Diese Stationsaussendungen werden dann, je nach Abdeckung, von ein bis drei Sigfox-Basisstationen (**Sigfox stations**) empfangen. Das alles ist vergleichbar mit der Funktionsweise der Mobilfunknetze, nur dass über das Sigfoxnetzwerk reine Mess-, Zustands- oder Positionsdaten übertragen werden und keine Sprache, Musik, Bilder oder Filme. Um ganz Deutschland abzudecken, sind etwa 1.200 solcher Sigfox-Basisstationen notwen-

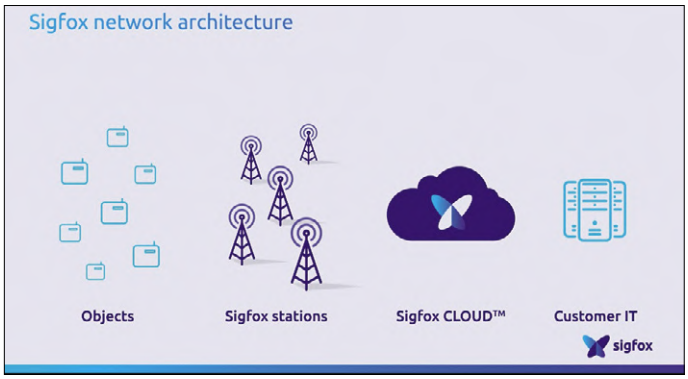


Bild 1. Grundlegende Struktur des Sigfox-Netzwerkes (Quelle: Sigfox).

dig, wobei die Abdeckung im Moment bereits circa 85 % beträgt. Jede Sigfox-Basisstation kann Aussendungen von bis zu einer Million Stationen empfangen, verarbeiten und weiterleiten.
Die Basisstationen senden alle empfangenen Daten via Internet oder GSM-Verbindung in die **Sigfox Cloud**, von der sich der Anwender seine Daten abholen und in seiner EDV (**Customer IT**) weiter auswerten kann. Innerhalb der Cloud werden die Daten automatisch an den entsprechenden Benutzeraccount weitergeleitet, in dem das sendende Gerät registriert wurde. Die Konfigurationsoberfläche der Benutzeraccounts nennt sich **Sigfox Backend.** Hierin werden die Sigfox-Knoten registriert, Gruppen zugeordnet und die Datenweiterleitungen an die Customer IT (Callbacks) eingerichtet.
Man sieht, dass zwischen der Sigfox-Station und dem Anwender jede Menge komplexer Sigfox-Hard- und Software vorhanden ist. Aber das alles braucht uns hier gar nicht zu interessieren, denn der Einsatz einer Sigfox-Station im Netz ist sehr einfach mit wenigen Schritten zu realisieren:

- Man benötigt neben dem Sigfox-Modem für seinen Mikrocontroller eine geeignete Sigfox-Kommunikationssoftware. Das ist in unserem Fall gar kein Problem, denn es gibt eine fertige Arduino-Bibliothek, die einfach nur zum Sketch hinzugebunden werden muss.
- Als Nächstes muss man sich selber, via Internet, beim Sigfox-Backend als Sigfox-Anwender anmelden, was mit wenigen Mausklicks erledigt ist. Beim Kauf eines MKR FOX 1200-Boards erhält man gleichzeitig ein Abonnement für ein Jahr kostenfreien Betrieb im Sigfox-Netz.
- Dann muss man seine Station beim Sigfox-Backend anmelden. Auch das kostet nur wenige Mausklicks.
- Danach entwickelt man mit den fertigen Funktionen aus der Sigfox-Bibliothek seine Sigfox-Anwendung (den Arduino-Sketch) und sendet die Daten zum Sigfox-Backend.
- Vom Sigfox-Backend kann man sich die Daten dann weltweit via Internet sehr einfach „weiterleiten“ und auf seinem Rechner (zum Beispiel mit einem Freeware-Dashboardprogramm) weiterverarbeiten.

Die ersten drei Schritte werden in diesem Artikelteil behandelt; im nächsten Teil widmen wir uns dem vierten Punkt, während der letzte Teil ganz dem fünften Punkt vorbehalten ist.
Da Sigfox ein funkbasiertes Datenübertragungssystem ist, sollten Sie sich vorab darüber informieren, wie gut die Sigfox-Abdeckung an Ihrem Standort aussieht. Das können Sie sehr einfach mit Hilfe der Sigfox-Abdeckungskarte machen [1].

Das Sigfox-Backend

Bild 2 zeigt die allgemeine Struktur, die der Anwender auf dem Sigfox-Backend für seine Sigfox-Systeme erstellen muss. Der Endpunkt jeder Datenübertragung von den einzelnen Sigfox-Stationen ist die Sigfox-Cloud mit dem Backend-Rechner im jeweiligen Land. Von dort aus kann sich dann der Anwender via Internet die Daten auf seinen Rechner herunterladen und entsprechend weiterverarbeiten. Wir erläutern zunächst, wie die Daten von unserem MKR FOX 1200-Board im Backend-Rechner strukturiert abgelegt werden.
Bei der ersten Anmeldung als Anwender im Sigfox-Backend muss man unter anderem seinen (beliebigen) **Business Name** (Geschäftsnamen) angeben. Daraus erzeugt das Backend auto-

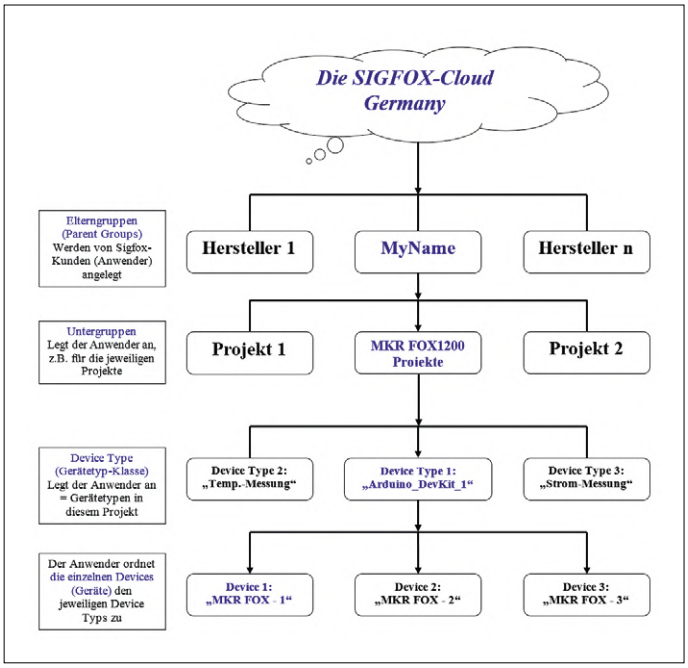


Bild 2. Grundlegende Struktur auf dem Sigfox-Backend.

matisch die Sigfox-**Parent Group** gleichen Namens, hier beispielsweise *MyName*. Damit ist man als Anwender im Sigfox Netzwerk registriert. Der Name kann nachträglich nicht mehr geändert werden.
Nun kann man innerhalb (unterhalb) dieser Gruppe so genannte **Untergruppen** anlegen und so seine ganzen Sigfox-Anwendungen etwa nach einzelnen Projekten individuell strukturieren. Man könnte also eine Untergruppe namens *„MKR FOX 1200 Projekte“* anlegen, was aber wenig Sinn ergibt, wenn wie hier nur ein einziges MKR FOX 1200-Board betrieben wird. Wir verzichten daher auf das Anlegen von Untergruppen.
In der nächsten darunter liegenden Ebene lassen sich nun die eingesetzten Geräte zu einem **Device Type** (Gerätetyp-Klasse) zusammenfassen. In einem Device Type befinden sich alle Geräte gleicher Art, die gleich aufgebaut sind, die gleichen Funktionen haben und somit auch die gleichen Datenarten und die gleichen Datenmengen versenden. Im Allgemeinen sind dies identische Geräte eines Herstellers, etwa die gleichen Geräte zur Temperaturmessung, zur Druckmessung, zur Mengenmessung und so weiter. Der Sinn dahinter ist, dass die Datensätze einer Gerätetyp-Klasse gleich behandelt werden können, nur eben mit unterschiedlichen Inhalten.
In unserem Fall ist es wieder sehr einfach, da wir nur ein Gerät des einzigen Gerätetyps haben, nämlich unser MKR FOX 1200-Board. Wenn wir also später das Board beim Sigfox-Backend anmelden, so legt das Backend automatisch den Device Type *„Arduino_DevKit_1“* an. Dieser Name kann später geändert werden, ebenso wie weitere Device Types individuell angelegt und wieder gelöscht werden können.
In der letzten Stufe werden nun die einzelnen eingesetzten Geräte (**Device**) konkret einem Device Type zugeordnet, und das bedeutet: Bei der Anmeldung unseres MKR FOX 1200-Boards ordnet das Backend dieses Board automatisch dem Device Type *„Arduino_DevKit_1“* zu und vergibt dafür auch automatisch den gleichlautenden Device-Namen *Arduino_DevKit_1_device*. Diesen nicht sehr originellen Namen ändern

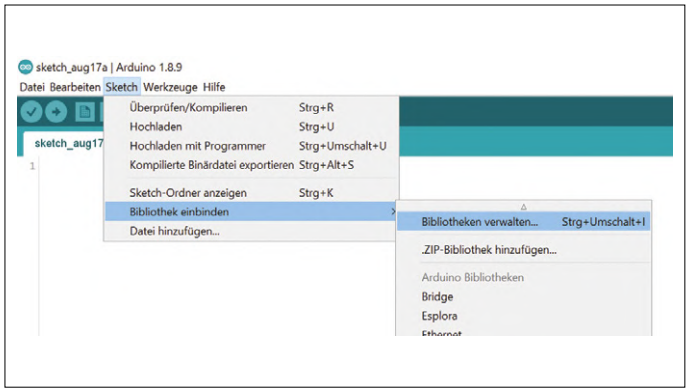


Bild 3. Aufrufen des Bibliotheksverwalters.

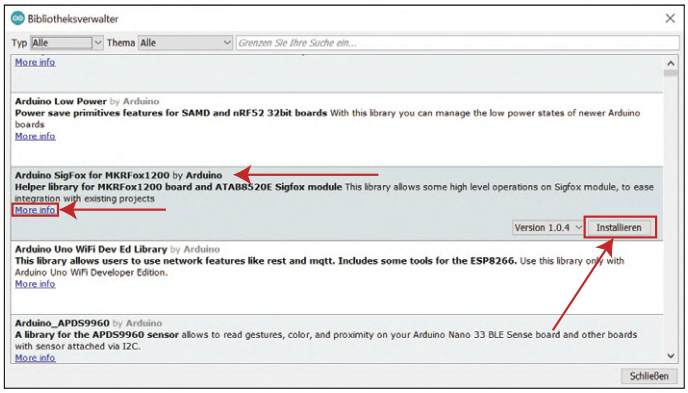


Bild 4. Liste der bereits vorhandenen originalen Arduino-Bibliotheken.

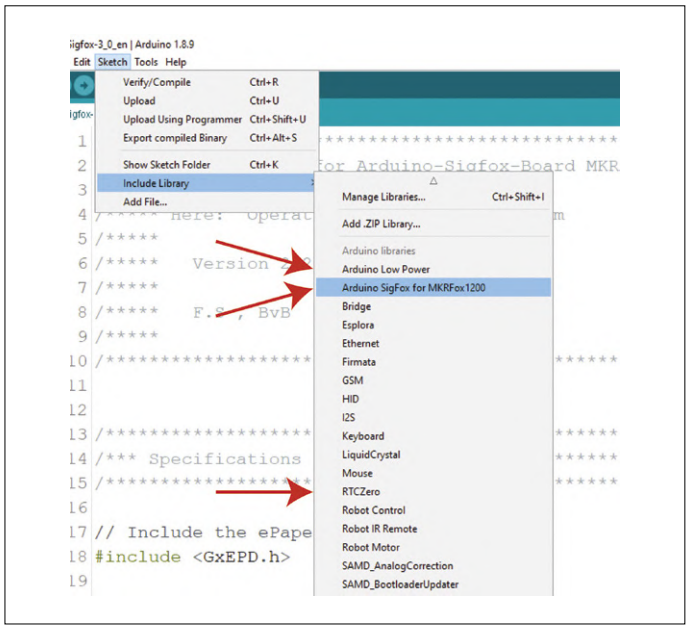


Bild 5. Die drei neuen Bibliotheken stehen zur Verfügung.

wir später in *MKR FOX – 1*. Dabei ist folgendes sehr wichtig: Alle Devices eines Device Type senden später gleich strukturierte Datensätze (Callbacks) aus dem Sigfox-Backend an den Anwender-Rechner, nur eben mit individuell verschiedenen Inhalten! Unterschieden werden die einzelnen Geräte dann

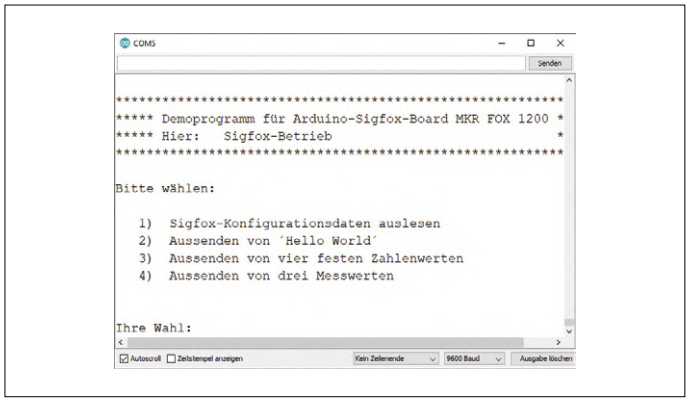


Bild 6. Hauptmenü des Sigfox-Sketches *Sigfox-3_0.ino*.

anhand ihrer eindeutigen Geräte-Identifikationsnummer (ID). Da wir nur ein einziges MKR FOX1200-Board einsetzen, reduziert sich der Strukturbaum einfach auf:

- `Group:` `MyName`
- `Untergruppe:` entfällt
- `Device Type:` `Arduino_DevKit_1`
- `Device Name:` `MKR FOX - 1`
- `Geräte ID:` `47110815` (beispielsweise)

Unser MKR FOX 1200-Board sendet seine Nachrichten an das Sigfox-Backend. Dort werden die Daten intern zugeordnet und können dann an den Anwender weitergeleitet werden. In welcher Form dies geschieht, wird in den so genannten **Callbacks** festgelegt (die wir in den folgenden Teilen des Artikels besprechen wollen).

Installation der Arduino-Sigfox-Bibliothek

Bevor man mit dem MKR FOX1200 im Sigfox-Netzwerk arbeiten kann, muss die passende Sigfox-Bibliothek *Arduino Sigfox for MKRFox1200* mit den erforderlichen Sigfox-Kommunikationsroutinen installiert werden. Dazu ruft man in der Arduino-IDE den Bibliotheksverwalter mit *Sketch\Bibliothek einbinden\Bibliothek verwalten* auf (**Bild 3**). Es öffnet sich eine (sehr umfangreiche) Liste mit den originalen Arduino-Bibliotheken, die bereits in der Arduino-IDE enthalten sind und bei Bedarf, je nach Anwendung, nur noch hinzuinstalliert werden müssen (**Bild 4**). Dazu gehören (in neueren Versionen der IDE) auch die drei von uns benötigten Bibliotheken. Laden und Installieren Sie jetzt die Bibliothek durch einen Klick auf *Installieren*. Auf die gleiche Art und Weise werden die beiden originalen Arduino-Bibliotheken *Arduino LowPower* und *RTCZero* installiert.
Solche Original-Arduino-Bibliotheken aus der IDE haben den sehr großen Vorteil, dass sie gut und ausführlich dokumentiert sind. Wenn Sie auf den Schaltbutton *More info* (unten links in jedem Bibliotheksfeld) klicken, so gelangen Sie zu einer detaillierten Beschreibung der Funktionen dieser Bibliothek und ihrer Anwendung. Wenn Sie erneut *Sketch\Bibliothek einbinden* anwählen, können Sie sehen, dass die drei neuen Bibliotheken jetzt in der IDE zur Verfügung stehen (**Bild 5**).
In einem neuen Sketch bindet man lediglich die Sigfox-Bibliothek durch Klicken auf den Bibliotheksnamen ein (die anderen beiden Bibliotheken werden automatisch mit eingebunden). Die IDE fügt daraufhin selbstständig die entsprechende `#include`-

Anweisung in den Sketch ein:

```
#include <SigFox.h> //Einbinden der Sigfox-Bibliothek
```

Alle Funktionen aus den drei Bibliotheken werden damit verfügbar und können sofort verwendet werden.
Tabelle 1 zeigt eine Kurzbeschreibung der wichtigsten Sigfox-Kernfunktionen, die wir nachfolgend anwenden werden.
In unserem Softwarepaket [2] finden Sie den Sketch *Sigfox-3_0.ino*, in dem alle nachfolgend beschriebenen Programmsequenzen sofort lauffähig enthalten sind. Laden Sie diesen Sketch, starten Sie ihn und auch den Seriellen Monitor. Im Seriellen Monitor erscheint das Hauptmenü unseres Sketches (**Bild 6**).

Sehr wichtig: Spätestens jetzt sollte die Sigfox-Antenne an das Modul angeschlossen sein, damit bei (unbeabsichtigten) Sendeversuchen die HF-Endstufe des Moduls nicht beschädigt wird! Und denken Sie daran: Offiziell dürfen pro Sigfox-Gerät nur 140 Nachrichten pro Tag gesendet werden!

Tabelle 1. Die wichtigsten Sigfox-Kernfunktionen aus der Sigfox-Bibliothek.

Funktion	Beschreibung	Syntax	Parameter	Returns
`SigFox.begin()`	Initialisiert die Sigfox-Bibliothek und das Modul.	`SigFox.begin();`	keine	liefert ein `true`, wenn alles in Ordnung, sonst ein `false`
`SigFox.beginPacket()`	Vorbereitung, ein Paket zu senden	`SigFox.beginPacket();`	keine	keine

`SigFox.print()`	Sendet Werte als Zeichenfolgen an das Sigfox-Backend.	`SigFox.print(val);`
		`SigFox.print(str);`
		`SigFox.print(buf,len);`
			val: ein Zahlenwert als Zeichenfolge
			str: ein String
			buf: ein Array, das Werte als Zeichenfolge enthält
			len: Länge des Buffers
				keine

`SigFox.write()`	Sendet binäre Daten an das Sigfox-Backend (Byte oder Serie von Bytes).	`SigFox.write(val);`
		`SigFox.write(str);`
		`SigFox.write(buf,len);`
			val: ein Zahlenwert als Byte
			str: ein String als Folge von Bytes
			buf: ein Array, das Bytes enthält
			len: Länge des Buffers
				keine

`SigFox.endPacket()`	Beendet den Prozess der Übertragung, die durch `SigFox.beginPacket()` gestartet wurde.	`SigFox.endPacket();`	keine	Liefert als integer eine 0, wenn Übertragung erfolgreich war und eine 1 bei einem Fehler
`SigFox.debug()`	Hierdurch wird der Debug-Modus aktiviert. Alle Energiespar-Funktionen werden deaktiviert, die auf dem Board befindliche LED bei Datenübertragungen aktiviert.	`SigFox.debug();`	keine	keine
`SigFox.SigVersion()`	Liefert die Firmwareversion des Sigfox-Moduls.	`SigFox.SigVersion();`	keine	String aus 2 Bytes
`SigFox.ID()`	Liefert die eindeutige Modul-ID, die vom Hersteller fest gespeichert wurde.	`SigFox.ID();`	keine	String aus 4 Bytes
`SigFox.PAC()`	Liefert den eindeutigen PAC, der fest zur ID gehört.	`SigFox.PAC();`	keine	String aus 16 Bytes
`SigFox.end()`	Deinitialisiert die Sigfox-Bibliothek und das Modul.	`SigFox.end();`	keine	keine

Listing 1. Auslesen der notwendigen Sigfox-Stationsparameter.

```
// Sigfox-Modem/Bib. aktivieren und
initialisieren
  if (!SigFox.begin())
  {
    Serial.println("Sigfox-Modem nicht gefunden !
    - Weiter mit RESET !");
    while (1);  // In diesem Fall: Endlosschleife
  }
  else
  {
    Serial.println("Sigfox-Modem OK !\n");
  }


  // Firmware-Version, ID, PAC und Temp. des
  // Modems auslesen und abspeichern
  String version = SigFox.SigVersion();
  String ID = SigFox.ID();
  String PAC = SigFox.PAC();
  float temp = SigFox.internalTemperature();

  // Modeminformationen an ser. Monitor senden
  ...

  SigFox.end(); // schicke das Modem in den
                // Tiefschlaf
```

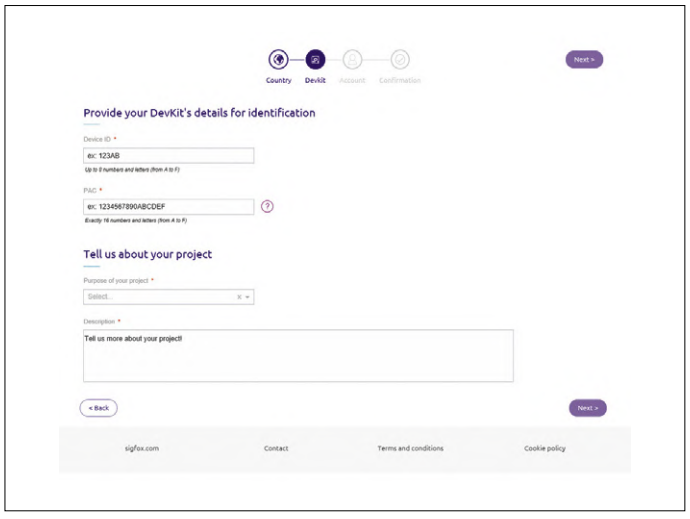


Bild 7. Registrierung des MKR FOX 1200 beim Sigfox-Backend (I).

Bild 8. ID und PAC sind in Ordnung und weiter geht´s …

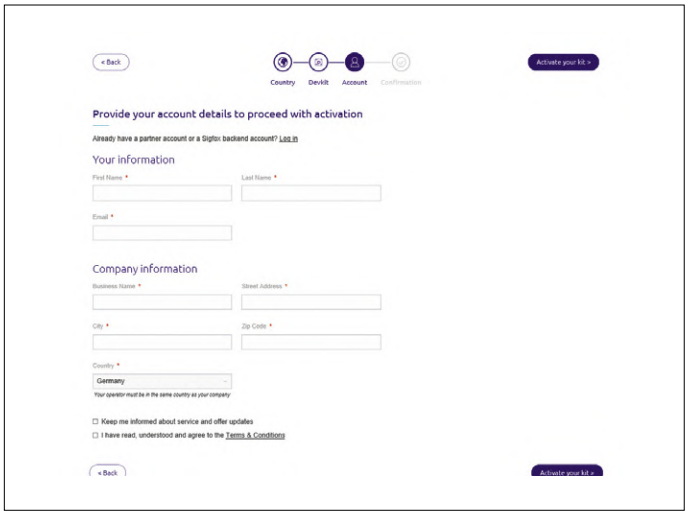


Bild 9. Registrierung des MKR FOX 1200 beim Sigfox-Backend (II).

Auslesen der Konfigurationsdaten des Sigfox-Modems

Wir erläutern nachfolgend nur die eigentlichen Kernteile des Sigfox-Betriebes aus unserem gut dokumentierten Sketch. Im ersten Schritt ist die Anmeldung (Registrierung) des MKR FOX 1200 im Sigfox-Netzwerk, genauer beim Sigfox-Backend, notwendig. Dazu benötigt man die **ID-** und die **PAC-Nummer** des Sigfox-Modems. Diese beiden Informationen sind für jede Sigfox-Einheit einmalig und eindeutig vorhanden (im Sigfox-Modem fest abgelegt) und können einfach ausgelesen werden.
Laden Sie nun unseren Demo-Sketch und wählen Sie den Menü-Punkt 1, dessen Kernfunktion in **Listing 1** deutlich wird. Als Erstes werden das Sigfox-Modem und die Sigfox-Bibliothek mit `Sigfox.begin()` aktiviert und initialisiert. Ein dabei eventuell auftretender Fehler wird lediglich angezeigt. Danach werden die benötigten Informationen mit den entsprechenden Sigfox-Funktionen gelesen, in Variablen gespeichert und schließlich mit geeigneten `Serial.println`-Anweisungen auf dem seriellen Monitor ausgegeben. Das Auslesen der Firmware-Version und der internen Modem-Temperatur ist interessant, aber nicht erforderlich. Mit dem letzten Funktionsaufruf `SigFox.end();` wird die Sigfox-Bibliothek deinstalliert, das Sigfox-Modem deaktiviert und in den Stromsparmodus geschaltet.
Die so ermittelten und zu notierenden ID- und PAC-Nummern sind beispielsweise:

`ID  = 00123456` (immer 8-stellig)
`PAC  = 1234567890abcdef` (immer 16-stellig)

Anmeldung des MKR FOX 1200-Boards beim Sigfox-Backend

Nun soll unser MKR FOX 1200-Board beim Sigfox-Backend unter [3] angemeldet werden. Auf der Startseite füllt man die notwendigen Felder wie in **Bild 7** aus und klickt unten rechts auf den *Next*-Button. Die ID-und PAC-Daten werden überprüft. Ist alles in Ordnung, erhält man eine freundliche Rückmeldung (**Bild 8**).
Danach klickt man wieder auf den *Next*-Button und gelangt zum zweiten Aktivierungsfenster (**Bild 9**). Wieder füllt man

Bild 10. Registrierung des MKR FOX 1200 beim Sigfox-Backend (III).

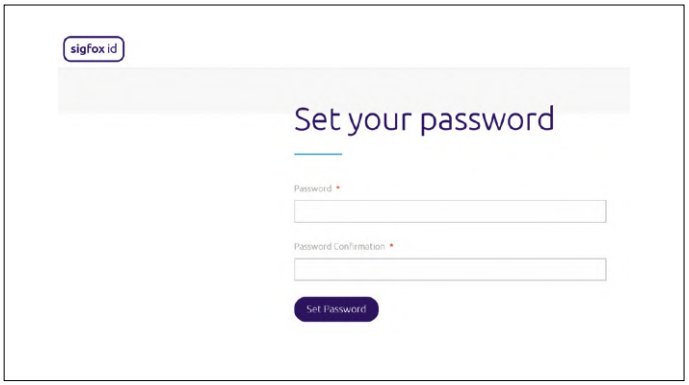


Bild 12. Vergabe des Passworts.

Bild 11. Bestätigungs-E-Mail.

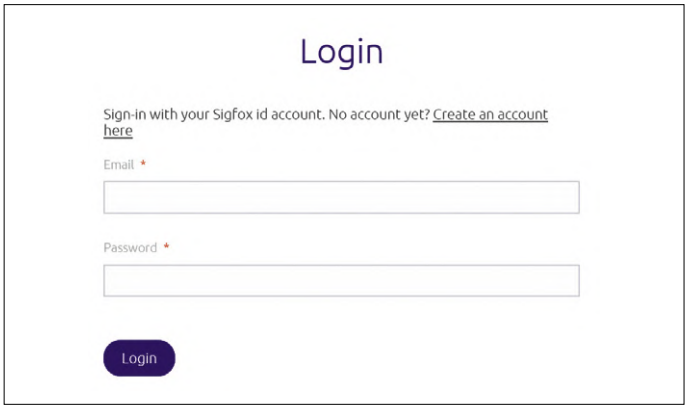


Bild 13. Eingabe der Login-Daten.

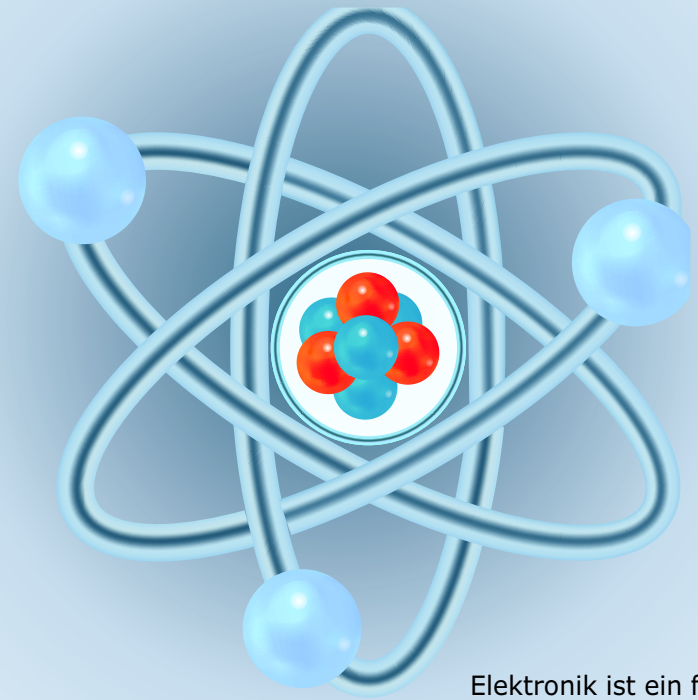
die Felder entsprechend aus und klickt dann rechts unten auf den Button *Activate your kit.* Wenn alles korrekt abgelaufen ist, erhalten Sie die Bestätigung Ihrer Registrierung (**Bild 10**). Wie man hier erkennen kann, wird die Aktivierung des MKR FOX 1200-Board erst dann endgültig abgeschlossen, wenn das Board seine erste Nachricht an das Sigfox-Backend gesendet hat (worum wir uns erst im nächsten Artikelteil kümmern). Erst danach ist das Gerät im Sigfox-Netzwerk bekannt und aktiviert und man kann ein Jahr lang kostenfrei 140 Nachrichten á 12 Nutzbytes pro Tag im Upload versenden (vier Nachrichten á 8 Nutzbytes pro Tag können als Download empfangen werden, zum Beispiel zur Konfiguration aus der Ferne). Bevor Sie die Seite schließen, sollten Sie sich noch die Einführungsvideos zu Sigfox auf dieser Seite ansehen. Sie können das aber auch später nachholen, denn man findet diese Filme auch bei Youtube [4][5][6]. Mittlerweile sollten Sie auch eine Bestätigung über die angegebene E-Mail-Adresse erhalten haben (**Bild 11**). Über diese Mail müssen Sie noch ein Passwort für die Anmeldung am Sigfox-Backend vergeben. Klicken Sie dazu auf das Feld *SET YOUR SIGFOX ID PASSWORD*, so dass die Eingabemaske in **Bild 12** erscheint.

Füllen Sie die Felder wie gewohnt aus und geben im nächsten Fenster Ihre Login-Daten ein (**Bild 13**). Nach dem Klicken auf *Login* wird das angelegte Profil angezeigt. Die Anmeldeprozedur beim Sigfox-Backend ist erledigt! Nun loggen Sie sich rechts oben im Fenster über *Profile&Settings* aus und üben Sie sich in Geduld. Im nächsten Teil werden wir den Arduino programmieren und die ersten Gehversuche im Sigfox-Netz unternehmen! ⏮

190281-B-01

Weblinks

[1] Sigfox-Abdeckung: www.sigfox.com/en/coverage
[2] Projektseite: www.elektormagazine.de/190281-B-01
[3] Sigfox-Anmeldung: https://buy.sigfox.com/activate/devkit/DE
[4] Einführungsvideo zu Sigfox: www.youtube.com/watch?v=6ZBGDtmDGRU
[5] Einführungsvideo zu Sigfox: www.youtube.com/watch?v=7gTwFbiiJwE
[6] Einführungsvideo zu Sigfox: www.youtube.com/watch?v=dDNY-xAxECE



Aller Anfang…

muss nicht schwer sein!

Von **Eric Bogers**

Elektronik ist ein faszinierendes Hobby, aber... es gibt immer ein Aber. Um mit dem Hobby auch erfolgreich zu sein, benötigt man ein gewisses Maß an Wissen, sonst hängt der Lötkolben bald wieder am Nagel. Natürlich kann man irgendeinen Bausatz erwerben, ihn mit einem Lötkolben aus dem Baumarkt zusammenbraten und dann beten, dass die Elektronik-Götter einem hold sind und die Schaltung funktioniert. Aber wenn doch etwas schief geht ...

... dann ist es nur gut, dass Sie diese Elektor-Ausgabe erworben, geliehen oder anderweitig in die Finger bekommen haben. Denn von diesem Augenblick an werden wir in jeder Ausgabe Grundlagen der Elektronik behandeln, um Ihrem Hobby ein solides Fundament zu verleihen. Und an die „alten Hasen“: Sie haben doch auch mal klein angefangen, nicht wahr? Aber wenn Sie denken, dass Sie bereits alles (besser) wissen, können Sie diese beiden Seiten ja ruhig überspringen.

Definitionen

Zuerst einmal sollte klar sein, worüber wir überhaupt reden. Deshalb sind einige klare Definitionen unerlässlich. Das mag ein bisschen langweilig und trocken klingen, aber beim nächsten Mal wird es deutlich spannender, versprochen!

Elementarladung

In der Elektronik und der Elektrotechnik dreht sich alles um Elektronen - daher der Name. Alle chemischen Elemente bestehen aus Atomen, die wiederum aus einem Kern (in dem fast die gesamte Masse versammelt ist) und darum kreisenden Elektronen. Die Elektronen bestimmen viele chemische Eigenschaften des betreffenden Elements. Elektronen haben eine (unteilbare) negative Ladung, die Elementarladung e, für die gilt:

$e = 1{,}6021892 \cdot 10^{-19}$ C

Ladung

Das *Einheitensymbol* C in dieser Definition steht für den *Einheitennamen* Coulomb, die Einheit der *Physikalischen Größe* Ladungsmenge. Eine Größe, zum Beispiel eine Länge, ist etwas, das in irgendeiner Weise gemessen werden kann. Die *Maßeinheit* oder kurz *Einheit* verleiht einer bestimmten Menge einer Größe einen wohldefinierten Wert, bei der Länge zum Beispiel das Meter. Die Einheit für die elektrische Ladung ist als das Coulomb C definiert:

$1\ C = 6{,}2414601 \cdot 10^{18}$ e

Dieser „krumme“ Wert stammt aus einer Zeit, in der die Elementarladung noch nicht gemessen werden konnte. Damals wurde willkürlich vereinbart, dass ein Coulomb gleich einer Amperesekunde ist: Wenn ein Strom von einem Ampere eine Sekunde lang fließt, dann wurde eine Ladung von einem Coulomb transportiert.

Nebenbei bemerkt: Einheitennamen werden stets groß und die Einheitensymbole klein geschrieben, es sei denn, dass der Name der Einheit von dem Eigennamen einer Person abgeleitet oder es anders festgelegt wurde, um Verwechslungen zweier Einheitssymbole zu vermeiden. Also heißt es Meter und 1 m, aber Coulomb und 1 C.

Strom

Die offizielle Definition des elektrischen Stroms (Einheit Ampere, Symbol A) ist ein Relikt aus ferner Vergangenheit, mit dem wir Sie nicht langweilen wollen. Wir leiten den elektrischen Strom aus der elektrischen Ladung ab:

1 A = 1 C/s

Man könnte auch sagen: Ein Ampere ist ein Strom von $6{,}2414601 \cdot 10^{18}$ Elektronen pro Sekunde.

Spannung

Es fließt also ein Strom - aber warum tut er das? Stellen wir einen Vergleich mit Wasser an. Wasser fließt dank des „Antriebs“ einer Druckdifferenz, verursacht durch eine Pumpe oder eine Höhendifferenz. Beim elektrischen Strom ist die treibende Kraft eine Ladungs- oder Potentialdifferenz zwischen zwei Punkten. Diese Ladungsdifferenz als Spannung mit dem Einheitsnamen Volt und dem Einheitssymbol V bezeichnet. Wichtig: Eine Spannung wird immer zwischen zwei Punkten gemessen.

Die Spannung wird über einen Umweg definiert: Fließt in einem Leiter ein Strom von einem Ampere und wird dabei eine Leistung von einem Watt in Wärme umgewandelt, liegt eine Spannung von einem Volt über diesem Leiter:

$1\ W = 1\ V \cdot 1\ A$

Leistung

Das bringt uns zur letzten Definition für heute, die der Leistung. Dabei müssen wir zwischen zwei Konzepten unterscheiden, die oft verwechselt werden: Leistung und Arbeit. Um beispielsweise ein Ei zu kochen, muss eine gewisse „Arbeit“ geleistet werden, um das Wasser zum Kochen zu bringen und diese Temperatur acht Minuten zu halten, bis das Ei hart ist. Das ist Arbeit, die während der gesamten Zeitspanne geleistet wird. Die Leistung dagegen ist diese Arbeit pro Zeiteinheit, zum Beispiel pro Sekunde.

Wenn wir über Elektrizität und Elektronen sprechen, ist Leistung definiert als das Produkt aus Strom und Spannung. Die Einheit der Leistung ist das Watt (W).

Formelzeichen

Natürlich ist es nicht sinnvoll, wenn wir immer umständlich beschreiben müssten, welchen Strom, welche Spannung oder welche Leistung wir meinen. Deshalb verwenden wir (kursiv geschriebene) *Formelzeichen*. Für die Ladung beispielsweise ist das der Großbuchstabe *Q*, Strom wird durch den Großbuchstaben *I* dargestellt, für die Spannung verwenden wir den Großbuchstaben *U* (im englischen Sprachraum *V*) und für die Leistung den Großbuchstaben *P*. Lassen wir uns in einer Tabelle zusammenfassen, was wir bisher über die grundlegenden Einheiten, die in der Elektr(on)ik eine Rolle spielen, gelernt haben.

Größe	Einheit	Formelzeichen
Ladung	Coulomb (C)	Q
Strom	Ampere (A)	I
Spannung	Volt (V)	U
Leistung	Watt (W)	P

Soviel zur trockenen Theorie. In der nächsten Ausgabe wird es interessanter, denn dann wird gerechnet!

191166-04

Diese Artikelserie „Aller Anfang…“ basiert auf dem bei Elektor erschienenen Buch „Basiskurs Elektronik“ von Michael Ebner.

Entwicklung analoger Elektronik

Fall Nr. 1 — MEMS-Mikrofon, Test 1-2-3 !

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

In dieser neuen Artikelserie zeigen Experten einem zunehmend „digitalen“ Publikum, dass analoge Elektronik kein Hexenwerk ist. Wir starten direkt mit einem seltsamen Bauteil, dem MEMS-Mikrofon. Extrem klein ist es und es wurde bislang wenig in DIY-Projekten verwendet, aber es ist ein wichtiger Bestandteil des erfolgreichen Fledermaus-Detektors, der 2016 im Elektor-Halbleiterheft veröffentlicht wurde. Ein MEMS-Mikrofon ist kein „drop-in-and-forget-Bauteil“, denn es hat Auswirkungen sowohl auf die eigene Beschaltung als auch auf den nachfolgenden Verstärker. Hier einige Details, mit Dank an die Abteilung für die „gute alte Elektronik“!

MEMS ist ein Akronym von *Micro Electro-Mechanical Systems*. Dabei werden mechanische Teile wie zusätzliche Substrate und hochwertige Mikro-Mechaniken in der Nähe von oder auf einem Halbleiterchip platziert. MEMS sind SMDs und können in der Produktion unter anderem mit Pick & Place-Maschinen bestückt werden. Die Einsatzmöglichkeiten sind vielfältig; MEMS-Bauteile werden in fast allen Marktsegmenten wie Unterhaltungs- und Medizinelektronik, Kommunikation oder Automotive eingesetzt. Anwendungen finden sich in der Bildstabilisierung für Kameras, Festplattensicherung, Herzschrittmachern, Insulinpumpen, medizinischen Drucksensoren, Mobiltelefonen, Airbags und vielem mehr. Kurz gesagt: MEMS sind aus der heutigen Welt nicht mehr wegzudenken.

Das MEMS-Mikrofon

Der vielleicht wichtigste Teil des Fledermaus-Detektors [1] ist das Mikrofon, das den Ultraschall der Fledermäuse aufnimmt. Es gibt zwar einige Arten, die im hörbaren Frequenzbereich zwischen 10 kHz und 20 kHz „arbeiten“, aber die meisten erzeugen doch Signale im Ultraschall-Bereich. Wie bei einem Elektretmikrofon verursacht eine MEMS-Membran, die der eigentliche Sensor für die Geräusche ist, minimale Kapazitätsunterschiede, die in eine Spannungsänderung umgewandelt werden können. Die meisten Elektretmikrofone sind mit einem JFET als Puffer ausgestattet, bei einem MEMS-Mikrofon jedoch ist die Membran direkt neben einem Halbleiterchip (ASIC) montiert. Dies ermöglicht die Umwandlung der analogen Spannung direkt in ein digitales (meist PWM- oder PDM-)Signal.

Bild 1 zeigt verschiedene Arten von MEMS-Mikrofonen, mit der akustischen Kopplung oben (ein einfaches Loch im Metalldeckel) oder unten zwischen den Pads, was ein Loch in der Platine erfordert. MEMS-Mikrofone sind im Allgemeinen sehr unempfindlich gegen Spannungsrauschen, Stöße, Vibrationen, Feuchtigkeit und Kondensation.

Natürlich hat auch der Einsatz eines MEMS-Mikrofons auf einer Platine Nachteile. Damit das Mikrofon nicht beschädigt wird,

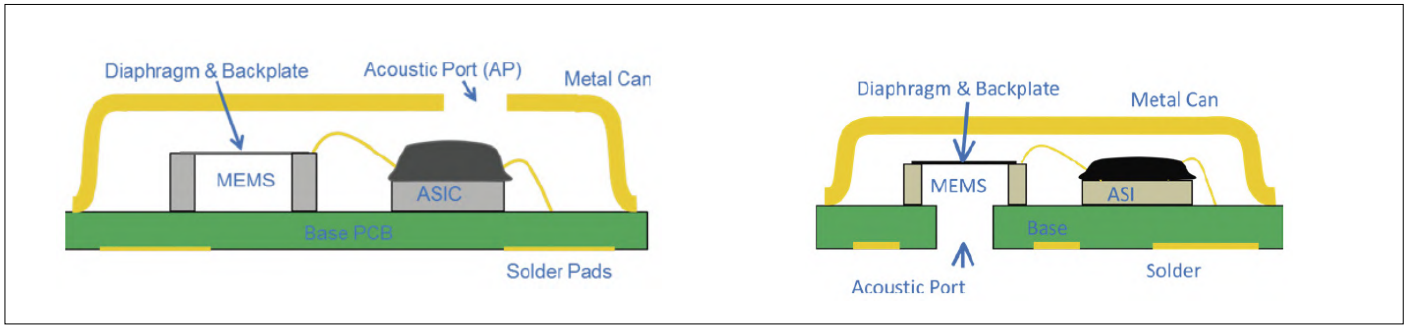


Bild 1. Querschnitte von MEMS-Mikrofonen mit oberem Loch (links) und unterem Loch (rechts). Quelle: Applikationsschrift „Sisonic Design Guide“ von Knowles [2].

darf die Platine nach dem Reflow-Prozess weder mit noch ohne Lösungsmittel gewaschen oder gebürstet werden. Anweisungen zur Bestückung finden Sie üblicherweise im Datenblatt und den Designrichtlinien des Herstellers.
Da die Membran im MEMS-Mikrofon sehr klein ist, besitzt es eine hohe Bandbreite. Die Empfindlichkeit ist oberhalb von 10 kHz um einige Dezibel höher. MEMS-Mikrofone eignen sich nicht nur für Sprachsignale, sondern auch für Einbruchmeldeanlagen, Bewegungserkennung und Ähnliches. Der Hersteller des im Fledermausdetektor verbauten Mikrofons bietet Modelle an, die bis zu 80 kHz (!) spezifiziert sind. Sofern nicht anders angegeben, kann davon ausgegangen werden, dass die Empfindlichkeit im Ultraschallbereich und die Bandbreite dieser Mikrofone auch für allgemeine Zwecke geeignet sind. Bei dem linken Mikrofon in Bild 1 allerdings befindet sich das Loch nicht genau über der Membran, was den Frequenzgang beeinflusst. Das rechte Modell wäre für unseren Zweck also besser geeignet. Der Frequenzgang des von uns bevorzugten Mikrofons SPH0641LU4H-1 ähnelt der oberen roten Kurve in **Bild 2**, ein deutliches Zeichen, dass es zur zweiten Kategorie gehört.

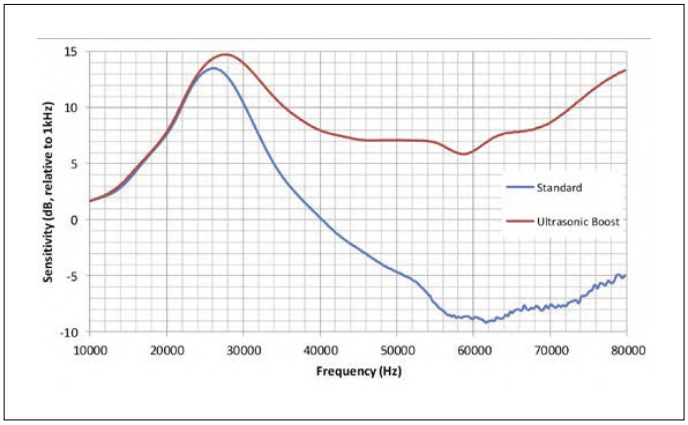


Bild 2. Frequenzgänge von MEMS-Mikrofonen. Typen mit oberem Loch oder „MEMS-on-Lid"-SiSonic-Mikrofone von Knowles sind für Ultraschall im Bereich von 20...80 kHz sehr geeignet. Diverse Modelle besitzen für diesen Bereich eine zusätzliche Verstärkung. Quelle: Applikationsschrift AN24 „Sisonic Design Guide" von Knowles [2].

Die Mikrofonplatine

Damit man das MEMS-Mikrofon auch mit manuellen Mitteln, sprich, einem Lötkolben, bestücken kann, gibt es im Elektor-Store eine kleine, mit dem Mikro vorbestückte Platine für eine marginale Schaltung (**Bild 3**). Dieses Breakout-Board ist universell einsetzbar. Falls die Versorgungsspannung rauscht, Störungen auftreten oder man die Platine über ein langes 3-adriges Kabel anschließt, sollte ein Entkopplungskondensator - am besten ein 1206-SMD mit 100 nF und C0G(NP0)-Dielektrikum - an der Unterseite der Platine angebracht werden. 1206 ist recht groß und daher immer noch leicht von Hand zu löten. Im Fledermausdetektor wird die Versorgungsspannung zusätzlich von einer 3V-Z-Diode stabilisiert, so dass aufgrund des geringen Abstands (die Leiterplatte ist in einem 3,5-mm-Audio-Klinkenstecker untergebracht) eine weitere Entkopplung nicht erforderlich ist. Mit Mikrophonie, von Bewegungen durch Vibrationen und/oder Stößen hervorgerufenen Spannungsschwankungen, hat ein solcher C0G(NP0)-Kondensator keine Probleme, ganz im Gegensatz zu den Keramikkondensatoren, die man oft als Kopplungs- oder Filterbauteile in Audioverstärkern findet und deren Kapazität von der angelegten Spannung abhängt.
Wie bei BGA-Gehäusen für SMD-ICs befinden sich die Anschlüsse des MEMS-Mikrofons auf der Unterseite des Bauteils. Es ist deshalb am besten, sie mit einem Reflow-Ofen auf die Platine zu löten. Das für das Projekt gewählte MEMS-Mikrofon besitzt ein Loch an der Oberseite, aber bei Typen mit einem Loch auf dem Boden wird ein passendes Loch in der Platine benötigt. Dieses beeinflusst wiederum den Frequenzgang je nach Durchmesser und Platinendicke. Die Platine spielt aber auch bei Mikrofonen mit oberem Loch eine Rolle, weil Reflexionen an ebenen Flächen in unmittelbarer Nähe des Mikrofons den endgültigen Frequenzgang beeinflussen. Um die Öffnung im Gerätegehäuse mit der Schallöffnung des Mikrofons zu verbinden, wird zudem eine Dichtung verwendet, die richtig sitzen muss, um Undichtigkeiten und Luftspalte zu vermeiden.

Anpassungen an den MEMS-Mikrofon-Analogverstärker

Bevor der Mikrofonvorverstärker entworfen werden kann, muss man sich über die Amplituden des Eingangs- und des Ausgangssignals im Klaren sein, unabhängig davon, ob ein MEMS-, Elektret- oder sogar ein dynamisches Mikrofon verwendet wird. Das eigentliche Mikrofon muss im Prinzip nur richtig angepasst und mit der richtigen Betriebsspannung versorgt werden. Häufig wird vom Hersteller eine Eingangsimpedanz angegeben (und natürlich der für ein Elektret erforderliche Pull-up-Widerstand). Im Datenblatt findet man auch meist eine „Typische Anwendungsschaltung", zumindest wird der Drainstrom des internen JFETs angegeben, so dass bei entsprechender Anpassung des Widerstandes auch eine höhere Versorgungsspannung für das Elektretmikrofon möglich ist. Achten Sie dabei aber auf die Maximalspannung des JFETs.
Die Ausgangsimpedanz eines Elektretmikrofons ist aufgrund des Pull-Up-Widerstands höher als die eines MEMS und kann daher mehr Rauschen erzeugen. Bei einem Elektretmikrofon bestimmt der JFET-Biaswiderstand zwischen Versorgung und Ausgang den Ausgangswiderstand. Der typische Wert beträgt 2,2 kΩ. Um die Signalspannung nicht übermäßig zu belasten, muss der Eingangswiderstand des Verstärkers 5 bis 10 Mal höher sein. Bei einem nicht-invertierenden Verstärker ist (am

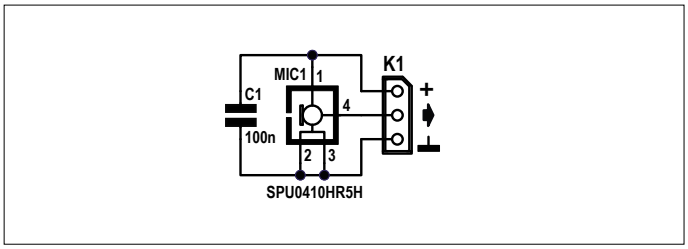


Bild 3. Schaltplan des SPH0641LU4H-BoBs.

Weblinks

[1] Fledermaus-DetektorPLUS: www.elektormagazine.de/magazine/elektor-201607/29063

[2] Knowles Sisonic Design Guide: www.knowles.com/subdepartment/application-notes/dpt-microphones/subdpt-sisonic-surface-mount-mems

[3] Elektor MEMS-Mikrofonplatine: www.elektor.de/bat-detector-with-amplitude-recovery-mems

Surfen und Lernen

Weblinks für weitere Informationen, auf die im Artikel nicht verwiesen wird

1. MEMS Microphones, the Future for Hearing Aids www.analog.com/en/analog-dialogue/articles/mems-microphones-future-for-hearing-aids.html
2. Digikey: www.digikey.de/de/articles/techzone/2019/feb/mems-vs-ecm-comparing-microphone-technologies
3. MEMS-Einführung: www.allaboutcircuits.com/technical-articles/improving-on-the-electret-an-introduction-to-mems-microphones/
4. Fledermausdetektoren: www.skillbank.co.uk/bat_detectors/afd1.html
5. TDK InvenSense-ICs: www.mouser.de/new/tdk/tdk-invensense-ics-40740/#TextLinks-4

Plus-Eingang) dieser Widerstand sehr viel höher. Er hat viel geringere Auswirkungen auf die Rauschbildung als die Widerstände in der Rückkopplung, die aber deutlich niedriger gewählt werden können, da sie das Mikrofonsignal nicht belasten. Nur wenn die Eckfrequenz am Eingang relativ hoch ist, entsteht um diese Frequenz herum eine höhere Eingangsimpedanz. Ansonsten wird die Impedanz nur durch den Ausgangswiderstand des Mikrofons bestimmt.

Wenn wir aber einen invertierenden Verstärker nehmen, bedeutet ein zu hoher Widerstand zusätzliches Eingangsrauschen. Dann muss man bei der Wahl des Eingangswiderstands einen Kompromiss eingehen, da er Teil der Rückkopplung ist. Nehmen wir zum Beispiel R4 = 7,5 kΩ im Verstärker des Fledermausdetektors. Das Rauschen eines Widerstandes nimmt mit der Quadratwurzel einer Widerstandsvergrößerung zu, also 3 dB zusätzliches Rauschen für eine Verdopplung (6 dB) des Widerstandswertes.

Wenn ein Mikrofon für die Aufnahme eines Konzerts oder einer Rede vorgesehen ist, kann man mit einem Signal in bestimmten Grenzen rechnen. Aber wenn es sich um ein Ultraschallsignal handelt, die Charakteristik des Mikrofons nicht spezifiziert ist und zudem der Abstand zur Schallquelle wie bei einer Fledermaus variiert, liegen die Dinge etwas komplizierter. Über eine bestimmte Entfernung in der Luft werden höhere Frequenzen stärker gedämpft als niedrigere, und diese Auswirkungen sind bei Ultraschall und unterschiedlichen Entfernungen erheblich. Wie die Kurven in Bild 2 zeigen, sind die Kennlinien alles andere als gerade, insbesondere bei der blauen Standard-Kurve.

Zuerst können Sie die gewünschte Verstärkung bestimmen. Im Falle des Fledermausdetektors muss sie garantieren, dass der Komparator (IC3B) richtig arbeitet, und auch den Signalpegel, den der Gleichrichter IC4A benötigt, um die Modulationswiederherstellung durchzuführen. Wahrscheinlich muss diese Verstärkung empirisch ermittelt und/oder an das verwendete Mikrofon angepasst werden. Besitzt der Verstärker eine feste Einstellung, erschwert dies die Verwendung verschiedener Mikrofone. Darüber hinaus ist es natürlich notwendig, den Frequenzbereich eines Verstärkers auf das Objekt der Begierde, in unserem Fall die Fledermäuse einzustellen. Der Mikrofonverstärker muss für eine Bandbreite von 10...120 kHz ausgelegt werden.

In der nächsten Folge entwickeln wie den Mikrofonvorverstärker des Fledermausdetektors weiter und suchen nach Lösungen, um ihn für die Verwendung mit dem vorgeschlagenen MEMS-Mikrofon zu optimieren. ⏮

191143-03

Machen Sie mit!

Wenn Sie ein Experte für analoge Elektronikentwicklung sind und gerne zu dieser Serie beitragen möchten, melden Sie sich beim für diese Reihe zuständigen Redakteur Jan Buiting, E-Mail jan.buiting@elektor.com.

elektor Schaltungs-Sonderheft 2020

JETZT AM KIOSK!

Dieses Schaltungs-Sonderheft enthält mehr als 90 kleine Schaltungen, Tipps und Tricks. Der Inhalt wurde aus veröffentlichten Elektor-Büchern und -Zeitschriften der letzten 10 Jahre ausgewählt. Bei der Auswahl der Artikel wurde darauf geachtet, dass die Schaltungen mit Standardkomponenten nachbaubar sind.

Aus dem Inhalt:

- Drehgeber und Motordrehzahlanzeige mit RPi Zero W
- Eisenloser Kopfhörerverstärker mit 4x EL504
- 10-Volt-Referenzspannungsquelle
- Fotodiode misst Gammastrahlung
- Rechteckgenerator 125 Hz bis 4 MHz
- GPS-Außenantenne
- Diebstahlschutz über OBD
- 4-A-Solarlader
- Joule Robbin' Hood
- Motorregelung mit MCP3002 ADC und RPi

Oder FREI HAUS bestellen unter
www.elektor.de/19140

Von der Pike auf gelernt

Neues aus der Elektor-Ideenkiste

Von **Eric Bogers**

Wir Elektor-Redakteure erhalten viele E-Mails, mal mit Lob, mal mit Kritik. Ein Kritikpunkt der letzten Jahre war, dass wir nach dem Ende des Halbleiterhefts nicht mehr genügend „kleine" Schaltungen, Tipps und Tricks veröffentlichen, die sich besonders für Hobby-Elektroniker eignen. Das ändert sich hier und jetzt!

Energiesparendes Relais

Idee von **Michael A. Shustov** (Russland)
und **Andrey M. Shustov** (Deutschland)

Diese Relaisschaltungen können in Reihe mit einer Last (Lampe) und einer oder mehreren, normal geschlossenen Drucktasten in Reihe geschaltet werden. Wenn einer dieser Schalter gedrückt wird, leuchtet die Lampe für eine bestimmte Zeit auf, bis nach Ablauf dieser Zeit die Lampe automatisch verlischt.
Das hört sich ganz nach einer Hotelschaltung [1] oder Treppenhausschaltung [2] an, bei der man die Beleuchtung mit einem Knopfdruck ein- und mit einem Knopfdruck an anderer Stelle wieder ausschalten kann. Der Unterschied: Das hier vorgestellte Relais schaltet das Licht automatisch aus – kann also erheblich zum Energiesparen beitragen.

Variante 1
Die erste Variante (**Bild 1**) ist nicht schwer zu ergründen. Die Schaltung wird von einer Gleichspannung von 12 VDC (Batterie, Akku, Netzteil) versorgt - und kann daher sogar von einem Noob sicher nachgebaut werden. Unmittelbar nach Anschluss der Stromversorgung wird der Kondensator C1 über die in Reihe geschalteten, normal geschlossenen Tasten S1...Sn, die Lampe LA1 und die Diode D1 aufgeladen. Es sind drei Taster eingezeichnet, aber es können beliebig viele sein. Die Lampe leuchtet kurz auf und LED1 zeigt an, dass die Schaltung „scharf" ist. Zur Vollständigkeit: Der Diodenstrom, der ja auch durch die Lampe fließt, ist viel zu niedrig, als dass diese leuchten könnte. Sobald eine der Tasten betätigt wird, kann kein Strom mehr durch D1 zu C1 fließen. Die Basis des Transistors T1, der im Ruhezustand über die Lampe am Pluspol der Spannungsquelle liegt, wird nun über den Widerstand R1 und die LED1 auf Masse gezogen. Dies hat zur Folge, dass der Transistor durchschaltet und der Kondensator C1 sich über die Relaisspule nach Masse entlädt. Das Relais zieht an. Über den Relaiskontakt kann jetzt ausreichend Strom fließen, so dass die Lampe leuchtet. Außerdem fließt kein Strom mehr, der den Kondensator C1 aufladen könnte. Nach einiger Zeit ist der Kondensator so weit entladen, dass das Relais nicht mehr aktiviert bleiben kann. Der Relaiskontakt öffnet sich und die Ausgangssituation ist wiederhergestellt (wobei wir stillschweigend davon ausgehen, dass die Taste nicht gedrückt gehalten wird). Die Zeit, in der die Lampe eingeschaltet bleibt, wird ungefähr durch die Beziehung

$$t = 0{,}67 \times R_{Relais} \times C1 \text{ [Sekunden, Ohm, Farad]}$$

bestimmt. Das in der Versuchsschaltung eingesetzte Relais besitzt einen Spulenwiderstand von 1050 Ω, bei einem anderen Typ sind es 1200 Ω. Mit dem erstgenannten Relais und einem Kondensator von 68.000 µF beläuft sich das Intervall, in dem die Lampe leuchtet, auf etwa 40 s.
Der Widerstand R1 muss so gewählt werden, dass der durch ihn fließende Strom gerade so ausreicht, dass die LED1 leicht aufleuchtet. Der Schaltplanwert reicht in der Regel aus, lässt aber Raum für Experimente. Abschließend noch ein deutliches Wort der Warnung: Diese Schaltung darf NICHT für Lampen am Lichtnetz verwendet werden!

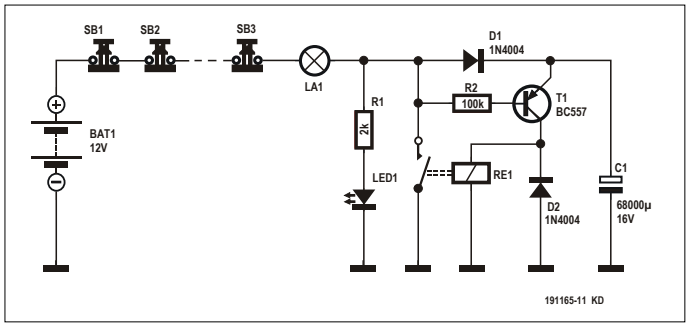


Bild 1. Die einfachste Version mit einem elektromechanischen Relais.

Variante 2
In dieser Variante (**Bild 2)** wurde das elektromechanische Relais aus Bild 1 durch einen MOSFET (2N7075 oder 2N7085) ersetzt, der als Schaltelement fungiert. Was hier nicht unwichtig ist: Im Gegensatz zur Relaisversion kann hier die Betriebsdauer der Lampe mit dem Potentiometer R3 bequem eingestellt werden. Grob gesagt entspricht 1 kΩ einer Leuchtdauer von 1 s, so dass mit einem 50-kΩ-Poti ein Einstellbereich von 1...50 s erreicht wird.
Der große Vorteil dieser Variante ist, dass für den zeitbestimmenden Kondensator C1 ein deutlich kleineres und preiswerteres Exemplar verwendet werden kann. Der maximale Schaltstrom beträgt (bei ausreichender Kühlung) etwa 30 A bei einem 2N7075 und 20 A, wenn man einen 2N7085 einsetzt. Auch hier gilt, dass sich diese Schaltung nur für Kleinspannungsanwendungen eignet und NIEMALS an das Lichtnetz angeschlossen werden darf!

Variante 3

In der Praxis dürfte es jedoch häufiger vorkommen, dass Sie Lampen schalten wollen, die an das Lichtnetz mit 230 V /50 Hz Wechselspannung angeschlossen sind. Nicht nur die Tatsache, dass es sich um Wechselspannung handelt, macht eine komplette Überarbeitung der Schaltung erforderlich, auch die Höhe der Spannung erfordert eine sorgfältige Auswahl der Bauteile und eine noch sorgfältigere Montage!

Wechselspannungen können nicht mit einem normalen Halbleiterschalter (Transistor oder MOSFET) geschaltet werden, man benötigt entweder wieder ein elektromechanisches Relais oder ein spezielles Bauteil namens Triac [3]. Kurz gesagt, können wir uns einen Triac als zwei antiparallele Thyristoren vorstellen [4]. Und ein Thyristor wiederum kann als eine Diode betrachtet werden, die über ihren Gate-Anschluss mit einem Zündimpuls in den (unidirektional) leitenden Zustand gebracht wird. Der Thyristor bleibt leitend, auch wenn der Zündimpuls vergangen ist, bis der Strom durch ihn unter ein gewisses Minimum fällt. Mit diesem Wissen versteht man auch leicht die Variante in **Bild 3**. Im Ruhezustand (keiner der Schalter gedrückt) fließt ein kleiner Strom durch die Lampe, den Kondensator C1, den Widerstand R2, die Diode D1 und den Widerstand R1, um LED1 ein wenig aufleuchten zu lassen und den Transistor T1 zu sperren. Der Kondensator C3 wird über D3 vollständig auf etwa die Z-Spannung von D2 (13 V) abzüglich der Schwellspannungen von D1 und D3 geladen.

Da sich der Triac nicht im leitenden Zustand befindet, leuchtet die Lampe nicht (auch hier ist der Strom durch R1 und die LED viel zu klein).

Das ändert sich schlagartig, sobald einer der Schalter kurz (!) gedrückt wird. Die Basis von T1 wird nach „unten“ gezogen, der Transistor leitet und der Kondensator C3 kann sich über T1 und R6 am Gate (Steuereingang) des Triacs entladen. Mit C3 = 10.000 µF bleibt der Triac für etwa 27 s im leitenden Zustand (etwas weniger als 3 s pro 1000 µF). Nachdem der Gate-Strom dann zu niedrig geworden ist, um den Triac leitend zu halten, erlischt die Lampe und der Ruhezustand ist nach ein paar Sekunden, in denen C3 wieder aufgeladen wird, erneut hergestellt.

Der Triac wird in dieser Schaltung nicht optimal genutzt, von seiner Fähigkeit, auch ohne dauernde Zündspannung in Leitung zu verbleiben, wird kein Gebrauch gemacht. Stattdessen wird das Gate des Triacs für die gesamte Leuchtdauer der Lampe kontinuierlich angesteuert, was die Verlustleistung im Triac natürlich erhöht.

Zwar ist die Zeit zwischen den Nulldurchgängen der Netzspannung (Stromminimum) sehr kurz, aber viele Schaltungen schicken kurz nach jedem Nulldurchgang nur einen kurzen Spannungsimpuls zum Gate, so dass der Triac die restliche Halbwelle leitet. Durch Veränderung der Zeitdauer zwischen Nulldurchgang und Zündung kann sogar die Helligkeit der Lampe eingestellt werden – tatsächlich, wir haben es mit einem Phasenanschnittsdimmer zu tun.

Konstruktion

Die beiden ungefährlichen Niederspannungsvarianten können einfach auf einer Lochraster/streifenplatine aufgebaut werden. Beim dritten Modell ist das anders. Am besten wäre eine echte „gedruckte“ Platine, bei der der Abstand zwischen allen Leiterbahnen mindestens 6 mm beträgt. Eine Experimentierplatine ist auch möglich, aber nur dann, wenn Sie schön Abstand zwischen den Lötstellen wahren und die nicht benötigten Lötpads dazwischen abkratzen, um einen sicheren Abstand einzuhalten.

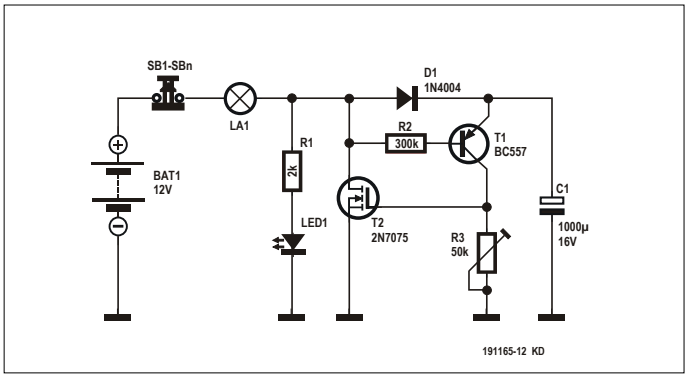


Bild 2. Die Gleichspannungsvariante in „pure silicon“.

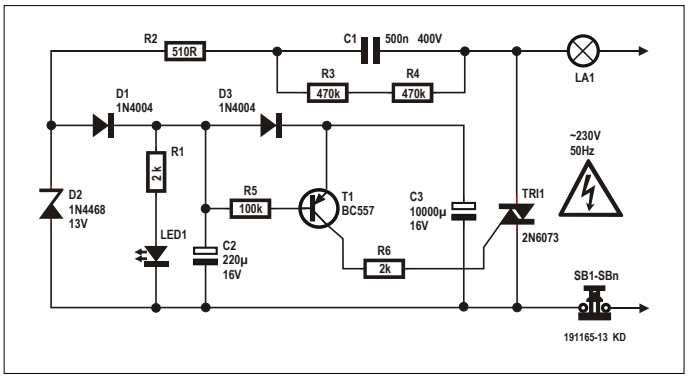


Bild 3. Die Netzspannungsvariante muss berührungssicher montiert werden!

Und bitte beachten Sie: Bei der Hochspannungsversion liegt an **allen** Bauteilen Netzspannung, so dass die Schaltung in ein berührsicheres Gehäuse verpackt werden muss!

Wenn Sie keine LED wünschen, die anzeigt, dass die Schaltung scharfgeschaltet ist, können Sie die LED in allen drei Varianten getrost weglassen. Der Widerstand R1 sollte jedoch erhalten bleiben, also ersetzen Sie die LED im Hinterkopf und in der Schaltung durch eine Drahtbrücke. Ohne die LED darf der Wert für R1 2...20 kΩ betragen. ⏮

(191165-04)

Links und Literatur

[1] Multi-Schalter für die Lichtsteuerung: www.elektormagazine.de/magazine/elektor-201405/26420

[2] Wechselschaltung: https://de.wikipedia.org/wiki/Wechselschaltung

[3] Triac: https://de.wikipedia.org/wiki/Triac

[4] Thyristor: https://de.wikipedia.org/wiki/Thyristor

Kurzgefasst: Texte für Mikrocontroller

Speicher sparen durch Kompression

Von **Thomas Beck**

Viele Mikrocontroller-Projekte geben Menü- und Hilfetexte, Fehler- oder Diagnosemeldungen über eine grafische Benutzer- oder eine Diagnoseschnittstelle aus. Ein großer Textumfang führt allerdings zu großem Speicherplatzbedarf. Mit einer geeigneten Kompression lässt sich aber so viel Text im Mikrocontrollerspeicher unterbringen, dass externe Datenspeicher meist vermieden werden können. Dieser Artikel zeigt eine mögliche Lösung durch Wörterbuchkompression.

Die Voraussetzung jeder erfolgreichen Kompression sind in den zu komprimierenden Daten enthaltene Redundanzen. Bei Texten entstehen Redundanzen durch Zeichen, die häufiger vorkommen als andere Zeichen, oder Zeichenfolgen, die mehrfach auftreten. So könnte man sich einer Entropiekodierung, zum Beispiel der Huffman-Kodierung [4] bedienen, um in einem Text häufiger auftretende Buchstaben mit kürzeren Bitfolgen zu kodieren als weniger häufige Buchstaben. Oder man könnte sich einem Substitutionsverfahren wie dem LZ77-Verfahren [5], auch Wörterbuchkompression genannt, bedienen, um sich wiederholende Zeichenfolgen in einem Wörterbuch abzuspeichern. Im Text wird die Zeichenfolge dann durch einen Verweis auf den Wörterbucheintrag ersetzt. Beim LZ77-Verfahren ist dafür interessanterweise nicht einmal das Abspeichern eines separaten Wörterbuchs in den komprimierten Daten nötig. Dort bildet ein Ausschnitt aus den dekomprimierten Daten ein Wörterbuch, das mit fortschreitender Dekompression langsam gefüllt und beim Erreichen der maximalen Wörterbuchgröße über ein Sliding-Window-Verfahren verschoben wird. Kompressionsprogramme wie 7z oder Zip verwenden eine Kombination beider Verfahren, Wörterbuchkompression mit nachfolgender Entropiekodierung, um eine bessere Kompression zu erreichen. Für kurze Texte ist die mögliche Einsparung bei beiden Verfahren jedoch gering. Außerdem ist es nicht möglich, einzelne Sätze aus einem komprimierten Text zu extrahieren, ohne den gesamten Text zu dekomprimieren.

Analyse des zu komprimierenden Texts

Wenn man sich die Texte der konkreten Anwendung in **Listing 1** anschaut, zeigt sich, dass eine Wörterbuchkompression, die Redundanzen ganzer Wörter und Wortfolgen ausnutzt, erfolgversprechend ist. Das in jedem Text häufig auftretende Leerzeichen wird als Worttrenner verwendet. Für das Leerzeichen erzeugt der Komprimierer entweder einen eigenen Wörterbucheintrag oder der Dekomprimierer fügt die Leerzeichen automatisch zwischen jedem Wort wieder ein.

Letzteres funktioniert nur dann sehr einfach, wenn es keine Sonderbehandlungen für Sonderzeichen gibt, die ohne Leerzeichen vor oder hinter einem Wort stehen, also wenn zum Beispiel *(Gauge)* ein Wörterbucheintrag ist und nicht durch die drei Einträge *(*, *Gauge* und *)* dargestellt wird.

- Der gesamte Text liegt in einem oder mehreren C-String-Arrays vor ➞ Einzelne Texte können gezielt über Array-Indizes angesprochen werden, was bei den komprimierten Texten ebenfalls möglich sein muss.
- Der Text enthält viele Wörter oder Wortfolgen, die häufig vorkommen ➞ Redundanzen sind die Voraussetzung jeder Kompression.
- Der Text liegt in englischer Sprache vor ➞ Der Text verwendet nur 7-Bit-ASCII-Codes (0...127). Die Codes ≥128 bleiben frei. Das vereinfacht später den Schritt 5 des vorgestellten Verfahrens.
- Die meisten Wörter im Text beginnen mit einem Großbuchstaben ➞ *Engine* und *engine* wären zwei verschiedene Einträge im Wörterbuch. Die Kompression kann verbessert werden, wenn alle Wörter mit Großbuchstaben beginnen. Aus Gründen der Lesbarkeit sollten Präpositionen und Artikel klein geschrieben werden.
- Der Text enthält wenige Sonderzeichen, zum Beispiel *( ) # / - ,* ➞ Bei der Generierung des Wörterbuchs entstehen deshalb nur wenige Einträge wie *(Gauge)*, die sich von einem möglichen zweiten Eintrag *Gauge* unterscheiden.

Listing 1. Ausschnitt des zu komprimierenden Texts als C-String-Array.

```
const char * const text[] = {
  /*   0 */ "Fuel System 1 Status",
  /*   1 */ "Fuel System 2 Status",
  /*   2 */ "Calculated Load Value",
  /*   3 */ "Engine Coolant Temperature",
  /*   4 */ "Short Term Fuel Trim Bank 1",
  /*   5 */ "Short Term Fuel Trim Bank 3",
  /*   6 */ "Long Term Fuel Trim Bank 1",
  /*   7 */ "Long Term Fuel Trim Bank 3",
  /*   8 */ "Short Term Fuel Trim Bank 2",
  /*   9 */ "Short Term Fuel Trim Bank 4",
  /*  10 */ "Long Term Fuel Trim Bank 2",
  /*  11 */ "Long Term Fuel Trim Bank 4",
  /*  12 */ "Fuel Pressure (Gauge)",
  ...
  /*  46 */ "OBD Requirements to Which Vehicle or
   Engine is Certified", // len=57
  ...
  /* 149 */ "Commanded Boost Pressure A",
  /* 150 */ "Boost Pressure Sensor A",
  /* 151 */ "Commanded Boost Pressure B",
  /* 152 */ "Boost Pressure Sensor B",
  /* 153 */ "Boost Pressure A Control Status",
  /* 154 */ "Boost Pressure B Control Status",
  ...
  /* 177 */ "Manifold Surface Temperature",
  /* 178 */ "Intake Manifold Absolute Pressure A",
  /* 179 */ "Intake Manifold Absolute Pressure B",
  ...
  /* 188 */ "Exhaust Gas Temperature Bank 2 -
   Sensor 7",
  /* 189 */ "Exhaust Gas Temperature Bank 2 -
   Sensor 8"
};
```

Wie alles begann...

Der Auslöser für die Entwicklung einer verlustlosen Textkompression waren meine Elektor OBD2-Fahrzeugdiagnose-Projekte [1][2][3]. Wie bei professionellen Diagnosetestern sollten für den Benutzer mehrere tausend Beschreibungen für Diagnosefehlercodes (Diagnostic Trouble Codes, DTC) bereitgestellt werden. Der Gesamtumfang dieser Texte beträgt etwa 211 KB, der durch das hier vorgestellte Kompressionsverfahren auf rund 35 KB reduziert werden konnte. Dadurch passen die Texte in den im Projekt [1] verwendeten 8-Bit-AVR-Mikrocontroller AT90CAN128 mit 128 KB Flash-Speicher.
Im Arduino Projekt [3] sollte auch der kleine Arduino Uno R3 und der Elektor Uno R4 mit nur 32 KB Flash-Speicher unterstützt werden. Natürlich ist dieser Speicherplatz auch für die komprimierten Fehlertexte zu klein, aber allein die etwa 7 KB umfassenden Beschreibungen für die bisher in den Projekten unterstützten OBD2-Parameter-Identifier (PID), meistens Beschreibungen der Sensordaten, konnten auf rund 2 KB komprimiert werden. Das von mir entwickelte Kompressionsverfahren soll am Beispiel der aktuell 190 kurzen PID-Texte in Listing 1 vorgestellt werden.
Für einen 8-bit-AVR-Mikrocontroller AT90CAN128 und den avr-gcc-Compiler 4.9.2 (Optimierung -Os eingeschaltet) wurden die folgenden Werte für den Speicherplatzbedarf der Dekomprimierroutine ermittelt (die restlichen Werte sind berechnet):

	Ursprungstext [Bytes]	Komprimierter Text [Bytes]	Dekomprimierer [Bytes]	RAM-Puffer für Dekompression [Bytes]
DTC-Texte	216311	35340	566	106
PID-Texte	7273	1820	314	57

- Der längste Text des gesamten Texts ist mit Stringendezeichen 57 Zeichen lang ➞ Der Dekomprimierer benötigt einen Puffer von mindestens 57 Bytes RAM.

Wörterbuchkompression

Schritt 1: Erzeugen des Wörterbuchs

Dieser Schritt zerlegt den zu komprimierenden Text in seine einzelnen Wörter. Als Trennzeichen wird das Leerzeichen verwendet, was später vom Dekomprimierer automatisch wieder eingefügt wird. Nebenbei wird die Häufigkeit und der Speicherplatzbedarf (Stringlänge plus Stringendezeichen) jedes Wortes berechnet. Am Ende wird das erzeugte Wörterbuch nach absteigenden Häufigkeiten sortiert. Bei gleicher Häufigkeit entscheidet die größere Länge. **Listing 2** zeigt bereits das im Schritt 2 optimierte Wörterbuch. Häufigkeit und Länge sind dort als Kommentar angegeben.

Listing 2. Ausschnitt des optimierten Wörterbuchs als C-String-Array.

```
const char * const dictionary[] = {
   /*   0 */ "Sensor", // cnt=116 len=7
   /*   1 */ "Bank", // cnt=104 len=5
   /*   2 */ "-", // cnt=89 len=2
   /*   3 */ "2", // cnt=67 len=2
   /*   4 */ "1", // cnt=66 len=2
   /*   5 */ "Fuel", // cnt=45 len=5
   /*   6 */ "Temperature", // cnt=43 len=12
   ...
   /*  16 */ "Exhaust Gas", // cnt=16 len=12
   ...
   /*  30 */ "Status", // cnt=9 len=7
   ...
   /*  45 */ "System", // cnt=4 len=7
   ...
   /*  77 */ "8", // cnt=2 len=2
   /*  78 */ "Currently Being Utilized by the",
   // cnt=1 len=32
   /*  79 */ "Advance for #1 Cylinder",
   // cnt=1 len=24
   ...
   /* 125 */ "F", // cnt=1 len=2
   /* 126 */ "G" // cnt=1 len=2
};
```

Listing 3. Array für Indexpaare (berechnet für 8-Bit-Indizes).

```
const dict_index_t pair[][2] = {
   /*  0 => 127 */ {   2,   0 },
     // "-", "Sensor", cnt=78
   /*  1 => 128 */ {   1,   4 },
     // "Bank", "1", cnt=40
   /*  2 => 129 */ {   1,   3 },
     // "Bank", "2", cnt=40
   /*  3 => 130 */ { 128, 127 },
     // "Bank 1", "- Sensor", cnt=31
   ...
   /* 57 => 184 */ {  15,  51 },
     // "Air", "Flow", cnt=4
   ...
   /* 67 => 194 */ { 184,   0 },
     // "Air Flow", "Sensor", cnt=3
   /* 68 => 195 */ {  56, 194 }
     // "Mass", "Air Flow Sensor", cnt=3
};
```

Schritt 2: Optimierung des Wörterbuchs

Falls im Text Wörter gleicher Häufigkeit vorkommen, die für jedes Auftreten benachbart sind, zum Beispiel *Exhaust* und *Gas*, können diese Wörterbucheinträge zu einem Eintrag *Exhaust Gas* zusammengefasst werden. Dazu muss im neuen Eintrag zwar das Leerzeichen zwischen den Wörtern wieder eingefügt werden, aber dafür entfällt eines von zwei Stringendezeichen. Da nach der Zusammenfassung zweier Einträge der zusammengefasste Eintrag wieder die obigen Bedingungen mit einem weiteren benachbarten Eintrag erfüllen kann, können bei wiederholter Anwendung Einträge entstehen, die mehr als zwei Wörter enthalten.

Da jeder Eintrag im Wörterbuch (C-String-Array) zusätzlich zum Speicherplatz für den String auch Speicher für einen Pointer auf diesen String erfordert, wird pro zusammengefasstem Eintrag ein Pointer eingespart. Die tatsächliche Einsparung hängt vom Speicherplatzbedarf eines Pointers auf dem Zielsystem ab. Zusätzlich wird das im Schritt 3 erzeugte Array `dictIndex[]` um die Häufigkeit des zusammengefassten Eintrags `* sizeof(dict_index_t)` kleiner.

Schritt 3: Kodierung des Originaltexts

Wenn das Wörterbuch erstellt ist, kann der Text mit Hilfe der Array-Indizes der Wörterbucheinträge kodiert werden.
Hier eine beispielhafte Kodierung des ersten Texts aus Listing 1:

text[0] = *Fuel System 1 Status*

- *Fuel* = Wörterbucheintrag dictionary[5]. Ersetze *Fuel* durch Index 5.
- *System* = Wörterbucheintrag dictionary[45]. Ersetze *System* durch Index 45.
- *1* = Wörterbucheintrag dictionary[4]. Ersetze 1 durch Index 4 (keine Kompression).
- *Status* = Wörterbucheintrag dictionary[30]. Ersetze *Status* durch Index 30.

Die Leerzeichen entfallen. Dann ergibt sich daraus folgendes Array:

```
const dict_index_t dictIndex[] = {
   /*   0 */ 5, 45, 4, 30, /* Fuel System 1 Status */
   /*   1 */ 5, 45, 3, 30, /* Fuel System 2 Status */
   ...
   /* 189 */ 16, 6, 1, 3, 2, 0, 77 /* Exhaust Gas
   Temperature Bank 2 - Sensor 8 */
};
```

Da die Anzahl der Indizes pro komprimiertem Text variabel ist, wird eine zweite Tabelle benötigt, die für jeden Text aus Listing 1 einen Verweis auf seinen ersten Index enthält. So kann jeder Text unabhängig von vorhergehenden Texten dekomprimiert werden.

```
const start_index_t startIndex[] = {
   /*   0 */    0, /* first index of text[0] is
   dictIndex[0] */
   /*   1 */    4, /* first index of text[1] is
   dictIndex[4] */
   ...
   /* 188 */ 1207, /* first index of text[189] is
   dictIndex[1207] */
   /* 189 */ 1213  /* first index of non-existent
   text[190] is dictIndex[1213] */
};
```

Der letzte Eintrag im Array `startIndex[]` führt dazu, dass der Dekomprimierer für alle Texte, auch den letzten, die Anzahl der Indizes einfach berechnen kann. Die Anzahl der Indizes für text[n] ist `startIndex[n+1] - startIndex[n]`.

Schritt 4: Erweiterung des Wörterbuchs um Indexpaare

Dieser Schritt komprimiert häufiger auftretende Wortpaare (Indexpaare) und Wortfolgen (Indexfolgen) effektiver. Zuerst wird die am häufigsten auftretende Indexfolge bestehend aus zwei Indizes gesucht. Diese wird am bisherigen Ende des Wörterbuchs als neuer logischer Eintrag eingetragen. Tatsächlich wird bei der Implementierung in **Listing 3** ein neues zweidimensionales Array `pair[][2]` angelegt und um einen aus zwei Indizes bestehenden Eintrag erweitert. Wenn der bisher größte Index des Wörterbuchs 126 war, so nimmt das erste Indexpaar in `pair[0]` den logischen Index 127 an. Dann müssen im Array `dictIndex[]` alle Indexpaare aus `pair[0][0]` und `pair[0][1]` (in Listing 3 Index 2 und Index 0) durch einen einzigen Index 127 ersetzt werden. Das Array `dictIndex[]` wird also um die Häufigkeit des Indexpaars kleiner, während im Array `pair[][2]` genau ein Eintrag, der dieses Indexpaar enthält, dazu kommt. Da dieser Schritt so oft auf das Array `dictIndex[]` angewendet wird, wie Paare mit einer minimalen Auftrittshäufigkeit gefunden werden, werden nicht nur zwei benachbarte Wörter, sondern auch längere Wortfolgen gefunden. Ein Indexpaar kann dann wieder Indizes von einem oder zwei Indexpaaren enthalten (Eintrag 3, Eintrag 67 oder Eintrag 68 in Listing 3).

Die Auftrittshäufigkeit eines Paares muss folgende Bedingung erfüllen, damit tatsächlich eine Kompression erreicht wird und ein neues Paar im Array `pair[][2]` eingetragen wird: Häufigkeit Indexpaar > 2 * `sizeof(dict_index_t)`.

Schritt 5: Einführung von ASCII-Indizes

In der tatsächlichen Implementierung folgt noch ein weiterer Schritt, der aufgrund seiner vielen Sonderfälle und Bedingungen hier nicht umfassend dargestellt werden kann. Durch die Einführung von ASCII-Indizes für Wörter im Wörterbuch, die mit Häufigkeit 1 auftreten oder für kurze Wörter mit geringer Häufigkeit, die nicht in `pair[][2]` verwendet werden, lassen sich Wörter, die nicht zur Komprimierung beitragen, aus dem Wörterbuch entfernen. Diese Wörter werden direkt mit den ASCII-Codes ihrer Zeichen im Array `dictIndex[]` eingetragen und im Wörterbuch entfernt. Zuvor müssen alle anderen Indizes entsprechend zu Werten ungleich der Werte der ASCII-Codes verschoben worden sein, bei 7-Bit-ASCII-Codes zum Beispiel zu Werten ≥ 128. Details dazu finden sich im veröffentlichten Sourcecode des Komprimierers und Dekomprimierers beim Arduino Projekt [3].

Nachbetrachtung

Bei der Ausführung der einzelnen Schritte zeigt sich eine generelle Problematik der Kompression. Der Speicherplatzbedarf für einen einzelnen Index `sizeof(dict_index_t)` ist vor der Kompression noch nicht bekannt, wird aber im Verfahren benötigt, um bei niedrigen Auftrittshäufigkeiten und kurzen Wörtern entscheiden zu können, ob überhaupt ein Wörterbucheintrag oder Indexpaar erzeugt werden soll. Im vorgestellten Fall der Textkompression der PID-Texte sind 8-Bit-Indizes ausreichend, was das Verfahren und die Dekompression vereinfacht. Die Textkompression der Fehlertexte kommt nicht mehr mit nur 256 8-Bit-Indizes aus. Dort wird mit einer Entropiekodierung der Indizes, die die häufigsten Wörter mit 8-Bit-Indizes und alle anderen Wörter mit 16-Bit-Indizes kodiert, der Vergrößerung des `dictIndex[]`-Arrays entgegengewirkt.

Wenn die mit Schritt 3 oder Schritt 4 erzielte Textkompression ausreichend ist, kann das Verfahren vorzeitig beendet werden. Die Dekomprimierroutine benötigt dann weniger Speicherplatz und weist eine kürzere Laufzeit auf, als wenn man weitere Schritte unternehmen würde. ⏮

180278-01

Weblinks

[1] OBD2-Analyser NG: www.elektormagazine.com/labs/firmware-update-and-emulator-for-obd2-analyser-ng-wireless-obd2

[2] OBD2 for Raspberry Pi: www.elektormagazine.com/labs/obd2-for-raspberry-pi

[3] OBD2 for Arduino: www.elektormagazine.com/labs/obd2-for-arduino

[4] Huffman-Kodierung: https://de.wikipedia.org/wiki/Huffman-Kodierung

[5] LZ77: https://de.wikipedia.org/wiki/LZ77

Im Fokus:
Autonomes Fahren

(Quelle: Viacheslav Gromov, IAA)

Stand der Technik im Überblick

Von **Viacheslav Gromov**

Das Thema „Autonomes Fahren“ ist global und allgegenwärtig: Aus den USA erreichen uns Schlagzeilen von Unfällen mit dem Tesla-Autopilot, in Europa vereinen sich die Automobilriesen, um konkurrenzfähig zu bleiben, in China fahren schon erste autonome Linienbusse. Wo stehen wir aus technischer Sicht momentan? Welche Mobilitätskonzepte streben wir an und welche gesellschaftlichen Herausforderungen trennen uns davon?

Die Computerassistenz im Fahrzeug, wie wir sie heute in Form von Spurhalteassistenten oder Sprachsteuerung kennen, blickt auf eine lange technologische Geschichte zurück. Spannend sind schon die Anfänge. Ein erstmals 1914 vorgestellter gyroskopischer Autopilot war für die Stabilisierung von Flugzeugen gedacht und gilt als erster Meilenstein der Steuerungsautomatisierung. Später wurden autonome Fahrzeuge (kurz: AF) eher im militärischen Kontext (zum Beispiel zur Minensuche) entwickelt. In den Medien wurden aber auch gerne sehr erstaunliche Mobilitätsvisionen vorgestellt. Dazu zählt auch eine Reihe von ferngesteuerten Versuchsmodellen sowie die im **Bild 1** dargestellte Leitdrahtvision [1].

Die Versprechen

Heutzutage gibt es folgende zentrale Argumente für das autonome Fahren: Sicherheit (menschliches Versagen verursacht etwa 90 % aller Unfälle), Komfort, umfassende Mobilität für alle sowie Entlastung unserer Verkehrsräume mit besserer Umwelt-Effizienz.

Beim Stichwort Sicherheit – für viele das wichtigste Argument – geht man davon aus, dass autonome Fahrzeuge auch dank des später erwähnten C2X die relativ häufigen menschlichen Unfallursachen (Unaufmerksamkeit, Alkohol, ...) minimieren und im Gegenzug nur wenige neue Unfallszenarien zum Beispiel durch Falschberechnungen oder Hackerangriffe entstehen. Klare, quantifizierbare Nachweise für diese höhere Sicherheit gibt es allerdings noch nicht, weshalb die (auch geisteswissenschaftlich diskutierten) Faktoren oft rein spekulativ sind.

Hinsichtlich der Mobilität gilt es hinzuzufügen, dass die Führerscheinpflicht entfällt und dadurch jede bisher ausgegrenzte gesellschaftliche Gruppe ein Fahrzeug nutzen kann. Auch können solche Fahrzeuge auf Abruf (vor allem Kleinbusse) gut im

ländlichen Raum verwendet werden.
Wenn es um die Verkehrsentlastung geht, sagen Studien eine Erhöhung der Straßenauslastung um 21 % beziehungsweise 80 % bei einer 50- respektive 90 %-igen Marktabdeckung durch AF voraus [2]. Dabei kann die Kraftstoffeffizienz um bis zu knapp 40 % erhöht werden, nicht nur durch einen geregelten Verkehrsfluss mit weniger Brems- und Beschleunigungsvorgängen, sondern auch durch grundsätzlich andere Geschwindigkeiten. Nach einigen Berechnungen kommt man selbst mit langsameren Geschwindigkeiten schneller von A nach B, so dass die Gesamtgeschwindigkeit steigt.

Konzepte und Automatisierungsgrade

Eine vielversprechende (Folge-)Technologie ist die Car-to-Car- (C2C) oder Car-to-Infrastructure-Kommunikation (C2I), die (mit anderen Varianten) auch unter dem Kürzel C2X bekannt ist, wobei statt „C“ für Car oft auch „V“ für Vehicle eingesetzt wird. Dabei können Fahrzeuge miteinander kommunizieren, um beispielsweise bei zu erwartenden Staus viel früher zu reagieren, oder mit der Infrastruktur interagieren, bis hin zu einer Vision, dass kaum noch sichtbare Verkehrsschilder benötigt und die Ampelschaltzeiten viel angepasster sein oder gänzlich wegfallen werden.
Grundsätzlich werden die Automatisierungsstufen des Fahrzeugs wie im **Bild 2** definiert. Die heute üblichen Level-1-Systeme wie ABS und ESP sind teilweise für die seit vielen Jahren sinkende Zahl der Verkehrstoten verantwortlich. Bei aktuellen Fahrzeugen können schon teilautonome Systeme integriert sein, die dem Level 2 entsprechen, etwa Spurhalte- oder Stauassistent. Der in die Schlagzeilen geratene Autobahnpilot von Tesla könnte schon an der Schwelle zu Level 3 eingeordnet werden.
Neben den Automatisierungsstufen müssen noch die Use-Cases, also Mobilitätsmodelle berücksichtigt werden. Dabei sind bei den oben genannten Automatisierungsstufen viele Konzepte denkbar. Anfangen könnte man beim autonomen Valet-Parken, bei dem man beispielsweise das Auto bis zum Einkaufscenter eigenhändig steuert, aussteigt und das Auto selbstfahrend einen Parkplatz im Umkreis sucht. Die höchste Form könnte dann das Rufauto (Vehicle-on-Demand) sein - je nach Preis könnte man dann allein oder gemeinsam mit anderen vom Auto befördert werden, nachdem man den Rufbutton auf dem Smartphone gedrückt und das Fahrzeug damit geordert hat.
Bei so gut wie allen Zielkonzepten verzichtet man auf ein Individualfahrzeug. Gerade dies kann, rechnet man die oben genannten positiven Effekte mit, ein großer Vorteil der AF sein. Die autonomen Fahrzeuge müssten dann keine 23 h am Tag stillstehen, sondern wären viel flexibler einsetzbar. Dies hat auch einen weiteren ökonomischen Grund: Die verbauten redundanten Systeme wären bei Level 5 nach heutigen Schätzungen trotz Serienfertigung ziemlich teuer und nicht für jedermann zu bezahlen. Die großen Automobilkonzerne wären folglich auch Mobilitätsdienstleister.

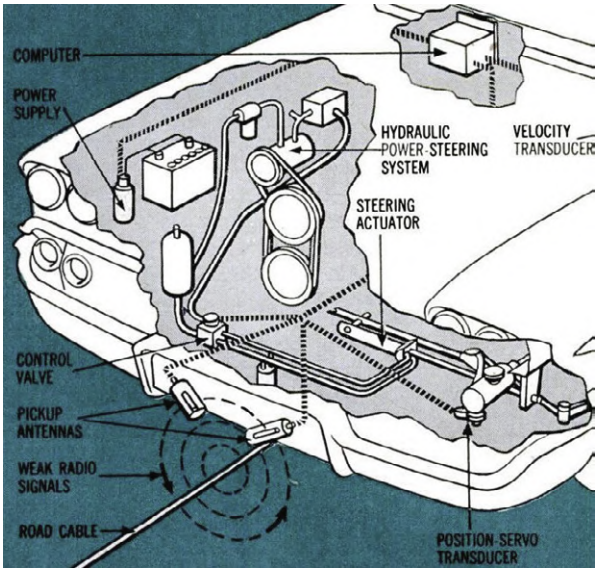


Bild 1. Ein Prototyp von General Motors, basierend auf dem 58er Chevrolet und mit zwei Antennen in frontaler Bodennähe ausgerüstet. Das Fahrzeug fuhr im Jahre 1958 „autonom“ an einer mit einem Leitdraht ausgestatteten Teststrecke (Quelle: Popular Science 05/1958).

Nun wollen wir (stark zusammengefasst und verallgemeinert) die technische Funktionsweise eines autonomen Fahrzeugs erläutern, so dass Sie eine „Ahnung“ von den Begriffen und Stichworten erhalten, die dann Grundlage für Ihre eigenen Recherchen sein können.

Ein Blick ins Innere

Der erste (und beim Menschen am Steuer kritischste Schritt) ist die **Perception**, also die Wahrnehmung. Hierbei soll die Umgebung des Fahrzeugs genauestens wahrgenommen wer-

SAE level	Name	Narrative Definition	Execution of Steering and Acceleration/ Deceleration	*Monitoring* of Driving Environment	Fallback Performance of *Dynamic Driving Task*	System Capability *(Driving Modes)*
***Human driver* monitors the driving environment**						
0	No Automation	the full-time performance by the *human driver* of all aspects of the *dynamic driving task*, even when enhanced by warning or intervention systems	Human driver	Human driver	Human driver	n/a
1	Driver Assistance	the *driving mode*-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	Human driver and system	Human driver	Human driver	Some driving modes
2	Partial Automation	the *driving mode*-specific execution by one or more driver assistance systems of both steering and acceleration/ deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	**System**	Human driver	Human driver	Some driving modes
***Automated driving system* (“system”) monitors the driving environment**						
3	Conditional Automation	the *driving mode*-specific performance by an *automated driving system* of all aspects of the dynamic driving task with the expectation that the *human driver* will respond appropriately to a *request to intervene*	System	**System**	Human driver	Some driving modes
4	High Automation	the *driving mode*-specific performance by an automated driving system of all aspects of the *dynamic driving task*, even if a *human driver* does not respond appropriately to a *request to intervene*	System	System	**System**	Some driving modes
5	Full Automation	the full-time performance by an *automated driving system* of all aspects of the *dynamic driving task* under all roadway and environmental conditions that can be managed by a *human driver*	System	System	System	**All driving modes**

Bild 2. Die fünf Automatisierungsstufen samt Kurzdefinition im Überblick (Quelle: SAE International J3016).

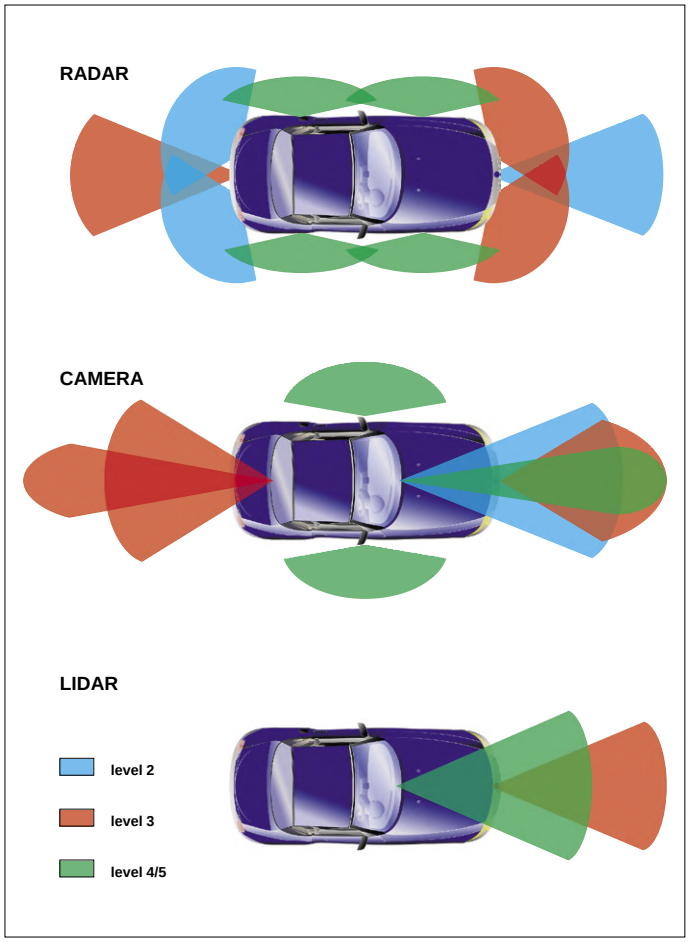


Bild 3. So könnte laut Infineon die Zunahme von Sensoren mit steigender Automatisierungsstufe aussehen. Weitere Sensoren, wie z.B. mit Ultraschall, sind hier nicht berücksichtigt (Quelle: Infineon).

den. Dabei spielen im autonomen Fahrzeug Kamera-, Radar- und Lidartechnik, deren Stärken sich ergänzen, zentrale Rollen. Ein Beispiel: Während eine Kamera die einzig gängige Möglichkeit ist, herkömmliche Farbschilder und Ampeln zu erkennen, kann ein Radar selbst bei sehr schlechter Sicht Objekte mit sehr genauer Entfernungsangabe wahrnehmen. Um aus jeder Situation das Optimum herauszuholen, nutzt man eine Sensorfusion, was bedeutet, dass man mehrere Sensordaten in einem Kontext gewichtet kombiniert, um so viele möglichst sichere Erkenntnisse über die Geschehnisse zu gewinnen.

Bild 3 zeigt ein Beispiel, wie die drei Hauptsensorarten je nach Automatisierungsstufe eingesetzt werden könnten, wobei die Angaben je nach Hersteller und Entwicklungsphase stark schwanken können. Die Sensor-Kegel von Radar und Lidar reichen dabei zweckmäßigerweise von wenigen bis zu einigen hundert Metern in Strahlrichtung. Selbstverständlich spielt dabei auch die Winkelauflösung und dadurch die Gesamtmenge der Daten eine entscheidende Rolle. Bei den häufig verbauten HD-Kameras kommen meist Weitwinkel-Brennweiten von 6…25 mm infrage. Auch der Einsatz von Stereokameras ist nicht unüblich.

In Folge der Perception müssen auch die Bewegungen der Objekte in naher Zukunft (beispielsweise die Fahrtrajektorie eines Fahrzeugs) samt allen dazu nötigen Vektoren abgeschätzt werden (**Prediction**), was eine sehr anspruchsvolle Aufgabe ist, bei der man auch wahrscheinliche Vorannahmen trifft. Ein Beispiel für den Perception-Aufbau von Apollo ist in **Bild 4** zu sehen. Apollo ist eine offene, vom chinesischen Digitalriesen Baidu ins Leben gerufene Plattform für AF, an der auch einige westliche Automotive-Größen beteiligt sind. Für eine Vertiefung der Materie ist ein Blick ins dazugehörige Github-Projekt interessant [3].

Ziel der Wahrnehmung ist es auch, Objekte richtig zu klassifi-

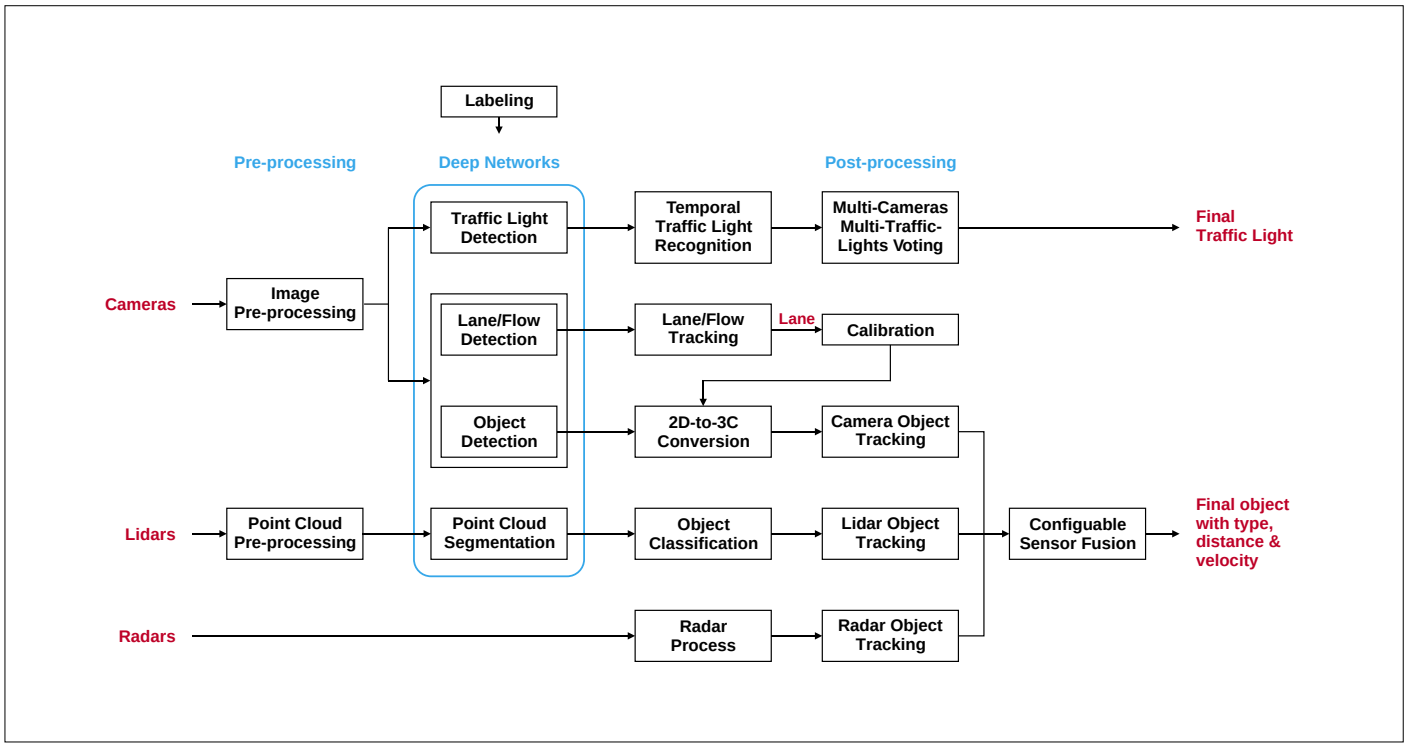


Bild 4. Der Aufbau der Perception-Einheit bei Apollo. Objekterkennung mit metrischen Daten und die Ampelerkennung laufen hier größtenteils parallel ab (Quelle: Apollo).

Bild 5. Zwei gängige Klassifizierungsarten: Links die semantische Segmentierung und rechts die Objekterkennung (Bild links mit Genehmigung von MathWorks, rechts: Symbolbild von Rolf Gerstendorf).

zieren. Objekte sind dabei sowohl Verkehrsteilnehmer als auch Schilder, Ampeln, Straßenmarkierungen, Hindernisse und vieles mehr. Eine nützliche Klassifizierungstechnik ist die semantische Segmentierung, die Flächen bestimmter Klassen entsprechend trennt (in **Bild 5** eingefärbt). Gerade bei der Fahrbahnerkennung und bei der Flächenabschätzung von anderen Objekten erweist sich die semantische Segmentierung als sehr nützlich. Für die Klassifizierung eignen sich das Maschinelle Lernen und dort vor allem Künstliche Neuronale Netze (KNN) dank Mustererkennung und hoher Robustheit hervorragend. Bei der Objektklassifizierung müssen drei Unsicherheiten berücksichtigt werden: Die Zustandsunsicherheit als Folge von Messfehlern der Sensorik, die Existenzunsicherheit mit der Frage nach der Existenz des erkannten Objekts und die Klassenunsicherheit, die die Korrektheit der zugeordneten Klasse betrifft. Gerade beim letzten Punkt ist die angenäherte Abschätzung und Berücksichtigung der Unsicherheit ziemlich kompliziert.

Eine ebenfalls der Perception folgende Aufgabe ist die **Localization**, also die Lokalisierung des Fahrzeugs in seiner Umgebung. Um auf diese „Wo bin ich?“-Frage zentimetergenau eine Antwort zu finden, werden meist die aus der Perception gewonnenen metrischen (Objekt-)Daten mit einer an Bord gespeicherten hochaufgelösten Karte (**Bild 6**) verglichen. Dabei werden unter anderem die aus der Robotik bekannten Vergleichsmethoden der gemessenen Punktewolken verwendet, stark vereinfacht gesagt also beim Überblenden der ursprünglich fahrzeug-zentrierten Sensordaten über die Karte Differenzen minimiert oder bestimmte typische Objekte (Landmarken) erkannt. Natürlich kommen auch GPS-Ortungsdaten sowie Bewegungsdaten aus der internen Inertial Measurement Unit (IMU) zum Einsatz. Zur Abschätzung des Zustands selbst bei Ungenauigkeiten oder gar einem Mangel von Sensordaten werden dabei (erweiterte) Kalmanfilter und viele andere Verfahren (auch im Zuge der Sensorfusion) eingesetzt.

Die Lösung mit der internen HD-Karte ist allerdings ein zweischneidiges Schwert. Einerseits kann dadurch schnell und effizient erkannt, lokalisiert und geplant werden. Aufgrund der aus der Karte stammenden Information können Erkennungsalgorithmen für Straßenschilder oder Ampeln effizienter eingesetzt werden, da sie dank der Erkenntnisse aus der Karte auf eine einzige Region-of-Interest (ROI) des Sensorfeldes ausgerichtet werden. Großer Nachteil ist dabei die Größe und die Aktualität der Karte. Die Karte müsste bei idealer Infrastruktur im festgelegten Radius ständig aus dem Netz in den internen Speicher geladen werden, was natürlich aufwändig ist. Und sie muss, damit das System wie vorgesehen funktioniert, sehr aktuell sein. So wird angedacht, dass Fahrzeuge bei der Durchfahrt das in der Cloud residente Kartenmodell für andere aktualisieren (ein Teilaspekt des möglichen kontinuierlichen Lernens). Vorteilhaft wäre natürlich, in Zukunft ganz auf die interne HD-Karte, zumindest in der aktuellen Form, zu verzichten. Es ist aber nicht gänzlich abzusehen, ob es irgendwann dazu kommen kann.

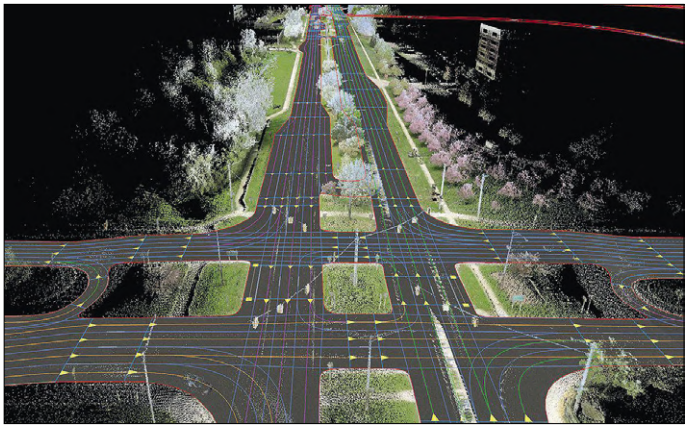


Bild 6. Eine Teilansicht der HD-Karte von Apollo. Die insbesondere für Lidar typischen Punktewolken sind hier weiß zu erkennen, während hier keine klassifizierten Objekte abgebildet sind. (Quelle: CB insights).

Wenn Position und Umgebung bekannt sind, wird die meist einige Dutzend Meter folgende Route geplant (**Planning**). Dabei wird aus einer Reihe von Wegemöglichkeiten der günstigste Weg ausgewählt, nachdem alle anderen durch Optimierung bestimmter Faktoren und unter Berücksichtigung physikalischer,

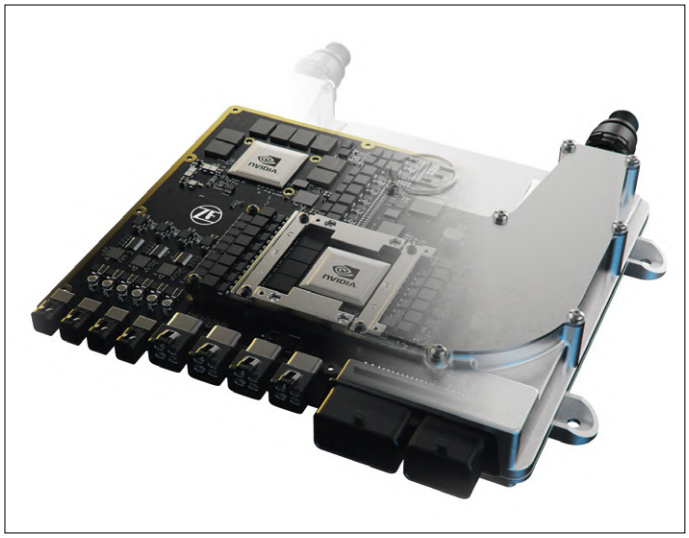


Bild 7. Der ProAI RoboThink von ZF, basierend auf NVIDIA DRIVE Xavier (oder bei Bedarf anderen) Chips. Abwärme ist bei Systemen dieser Art oft auch eine Designherausforderung - hier ist die Wasserkühlung angedeutet (Quelle: ZF).

Bild 8. Auf 3D-Engines beruhende Simulationssysteme sind beim AF üblich – hier ein Beispiel von AAI mit einigen Grundbausteinen wie Straßenschildern oder Gebäuden zum Aufbau auf der linken Seite (Symbolbild: AAI).

normativer und anderer Vorgaben (Constraints) ausgeschlossen wurden. Um dies effizient zu machen, können die Routenmöglichkeiten in Segmente, also meist Spurabschnitte, aufgeteilt werden. Selbstverständlich müssen als Grundlage der Zielort und die Gesamtstrecke bekannt sein (Gesamtnavigation).
Der letzte Schritt vor den Aktoren ist **Control**, also die Steuerung, die die zuvor geplante Bewegungs-Trajektorie in der realen Welt ausführt. Dazu muss sie auf Störgrößen in Echtzeit reagieren und dabei stets die physikalischen Möglichkeiten des Fahrzeugs berücksichtigen (zum Beispiel die Trägheit). Vom einfachen und vielen aus der Regelungstechnik bekannten Proportional-Integral-Derivative-Regler (PID) bis zu stark modellbasierten Verfahren werden dabei viele Techniken eingesetzt. Die Hauptaktoren sind natürlich Beschleunigung, Bremse und Lenkachse, wobei die dazu notwendigen mechatronischen Technologien aus den heute existierenden Drive-by-Wire-Systemen (DbW) verwendet werden können.
Die Gesamtsysteme solcher Fahrzeuge sind je nach Hersteller meist unterschiedlich auf Hard- und Softwarekomponenten aufgeteilt. Es gibt Mischkonzepte mit stark hardwaregetrennten Komponenten oder mit großen, stark redundanten Zentralelementen. Dies betrifft auch letztlich das Zusammenspiel von deliberativen und reaktiven Systemkomponenten. Besonders die Echtzeitfähigkeit und Effizienz aufgrund begrenzter Rechen- und Speicherleistung ist dabei zu berücksichtigen.
Dennoch gibt es viele grundsätzliche Ähnlichkeiten. Die Vorverarbeitung der Sensordaten bei der Perception beispielsweise wird wegen hoher Parallelisierung oft FPGAs beziehungsweise (bei Serienproduktion) ASICs überlassen. Auch kommen meist Zentralrecheneinheiten wie in **Bild 7** mit großen GPUs und sehr viel parallelisierter Rechenleistung zum Einsatz. Das hierbei abgebildete Modell, bei dem bis zu vier Einheiten zusammengeschaltet werden, zählt dabei zu den leistungsfähigsten auf dem Markt. Es lässt sich damit eine kombinierte Rechenleistung von bis zu 600 Tera-OPS (also 600 Billionen Rechenschritte pro Sekunde) erreichen!
Für die Entwicklung und Zulassung von AF-Modellen werden statt der üblichen Testkilometer im niedrigen zweistelligen Millionenbereich Schätzungen zufolge einige Milliarden Kilometer benötigt, da der Fahrer als fehlerkompensierende Komponente zunehmend entfällt und das System selber die benötigte Sicherheitsleistung erbringen muss. Hinzu kommt, dass das System um ein Vielfaches sicherer als ein heutiges Mensch-Pkw-Gespann sein sollte.
Da man diese vielen Kilometer nicht auf die Straße absolvieren kann, wird bei AF vieles auf der Softwareebene (Software-in-Loop, SiL) beziehungsweise auf dem Prüfstand (Vehicle-in-Loop, ViL) eintrainiert, simuliert und getestet. Besonders bei SiL werden meist bekannte Game- und Grafik-Engines mit virtuellen hochaufgelösten dreidimensionalen Umgebungen aufgebaut (**Bild 8**). Da man unendlich viele Fahrsituationen erzeugen könnte, arbeitet man an Methoden, die Situationen auf die sicherheitskritischsten und häufigsten zu beschränken. Grundsätzlich gelten auch für die Systeme höchste Sicherheitsvorgaben wie bei der ASIL-D-Klasse mit einer Ausfallswahrscheinlichkeit von weniger als $1/10^8$ pro Stunde. Und in der Diskussion sind viele weitere neue Sicherheitsstandards.
Zum Schluss muss noch betont werden, dass gerade bei Algorithmen, Sensoren und leistungsfähigen Halbleitern noch viele Verbesserungen zu erwarten sind - sowohl bei der Leistung als auch beim Preis.

Herausforderungen

Zweifelsohne ist die gesellschaftliche Akzeptanz eine der größten Hürden dieser Technologie. Studien zufolge ist die Angst vor einem Computerfehler sogar größer als die vor einem Hackerangriff, einem aus dem Alltag bekannten Risiko.
Wohl der brisanteste Punkt ist allerdings die Ethik, die früher oder später auch in Gesetzesform gegossen werden muss. Dem Menschen bleibt bei vielen Unfällen gar keine Zeit, um moralisch fundiert zu reagieren kann, die Maschine muss es können (und dies ist keine Übermoralisierung)! In Situationen eines Dilemmas kann ein komplexes Geflecht an Maschinenethik entstehen, und bei vielen damit zusammenhängenden Fragen kommt es letztendlich auch auf grundlegende Normen der Gesellschaft (auch global!) an.
Die für das AF in Deutschland eingesetzte Ethikkommission verbot selbstverständlich das Abwägen des Lebens nach Merkmalen (Alter, Geschlecht, ...), hält aber die Schadensminimierung bei Personen für vertretbar (Regel 9) [4]. Man kann aber viel tiefer und technischer einsteigen: Welche Sensoren müssen bei solchen Entscheidungen berücksichtigt werden? Wo zieht man die rote Linie? Man könnte nämlich bis zum

Reifenzustand, Straßenzustand und darüber hinaus gehen – Größen, die heute auch oft schon sensorisch erfasst werden. Die Diskussionen führen aber noch weiter. Machen sie mal ein einfaches Gedankenspiel: Wenn das Fahrzeug nicht mehr bremsen kann und Sie haben zwischen einer Kollision mit einem Motorradfahrer mit Helm und einem ohne Helm die Wahl, wie soll das Fahrzeug handeln? Der Helmträger hätte eine höhere Überlebenschance, aber wieso sollte man als Motorradfahrer dann einen Helm und damit ein höheres Unfallrisiko tragen? Darüber hinaus gibt es eine Reihe an Diskussionen über die passive Rollenübernahme des Fahrzeugs, eine Zufallsvariable, die Festlegung des „sicheren Zustands", des Szenarienkatalogs bei der Zulassung und unzählige weitere Themen. Immer müssen bei diesen Entscheidungen auch die Redundanz/Sicherheit und Rechenzeit berücksichtigt werden.

Die Ethik könnte man aber auch viel allgemeiner abstrahieren, mit der Frage, ob ab einem gewissen Sicherheitsgewinn der Gesetzgeber die Bürger zur ausschließlichen Nutzung von AF zwingen kann. Hierzu muss man sagen, dass Mischverkehr je nach AF-Zusammenspiel den geregelten Verkehr maßgeblich stören, aber durch Anpassung technisch gut toleriert werden kann. Aber was ist dann mit Freiheit und Fahrspaß? Die Ethikkommission erteilte diesem Zwang eine klare Absage; Oldtimer-Freunde und Poser dürfen aufatmen!

Rechtlich ist das Haftungsrisiko in diesem Zusammenhang ein spannendes Thema. Auch wenn wie bisher eine Produkthaftung für Herstellerfehler gelten soll und auch der Fahrer – wenn es beispielsweise nach der weltweit ersten Gesetzgebung zu AF in Deutschland von 2017 geht – für die Einschätzung, ob er bei aktueller Verkehrssituation den autonomen Piloten eingeschaltet lassen kann oder nicht, verantwortlich ist, treten noch neue, unvorhersehbare Situationen auf. Ein Beispiel: Die Technologie des maschinellen Lernens arbeitet unter anderem mit universalisierten Wahrscheinlichkeitsanordnungen, die auf unbekannte Situationen entsprechend reagieren sollten. Was passiert aber, wenn diese „Black Box" in der individuellen Situation etwas falsch erkennt und falsch handelt?

Experten sind auch überzeugt, dass AF aufgrund des Umfangs und der gesellschaftlichen Verantwortung nur ein übergreifendes Projekt sein kann. Nicht grundlos sind deshalb Plattformen wie Apollo oder die bekannten (internen) AF-Kooperationen wie die von Daimler und BMW entstanden. So oder so sind gemeinsame (auch ausdrücklich politische) Regelungen und Vorgaben unausweichlich und ein wesentlicher Beitrag zum Technologieerfolg. Herstellerübergreifende Softwareumgebungen wie das junge, offene Projekt namens OpenADx der Eclipse Foundation in Kooperation mit Bosch und vielen weiteren Beteiligten sind schon einmal ein guter Anfang [5].

Über den Autor

Viacheslav Gromov ist Gründer, Entwickler und freier Journalist im Bereich Elektronik. Seit seiner Schulzeit schreibt er Fachartikel und Fachbücher für diverse Zeitschriften und Verlage. Dank dem Hochbegabtenprogramm der Universität Freiburg durfte er schon während der Schulzeit mit dem Studium beginnen. Seine Fachgebiete sind künstliche Intelligenz (Edge AI), (ARM-/Wireless-)MCUs und FPGAs, über die er nicht nur Lehrbücher für Hochschulen oder Halbleiterhersteller verfasst, sondern auch an internationalen Elektronikentwicklungen von Südafrika bis zum Silicon Valley beteiligt war. Vor kurzem gründete er ein Unternehmen zur Entwicklung von lokalen KI-Lösungen für Industrieanlagen sowie von (zum Patent eingereichten) normativen Hybridsystemen für das autonome Fahren (readers@gromov.de).

Viele Länder scheinen in einigen Bereichen bessere Voraussetzungen für eine rasche Entwicklung dieser Technologie zu bieten als Deutschland (mit dem bekannten Testfeld auf der A9). In den USA gibt es mehr gesetzliche Freiräume zum Testen von Fahrzeugen ganz ohne Fahrer (Steuerung / Notaus per Fernbedienung) und in China sind die benötigten Trainings-, Test- und Simulations-Daten viel einfacher zugänglich – dort sind sogar schon alle Autobahnen für das AF HD-kartiert.

Schlussendlich muss für das AF das gesamte Verkehrskonzept überdacht werden. Es entstehen damit ganz neue Herausforderungen für die Verkehrsplanung in Stadt und Land.

Trotz technischer und struktureller Herausforderungen wird diese Technologie weltweit durch aktuelle Investitionen, Förderungen und Regelungen so vorangetrieben wie noch nie. Auch an den Rahmenbedingungen wird nun fleißiger gearbeitet. Und harte Arbeit zahlt sich bekanntlich aus! ⏮

191147-01

Weblinks

[1] „Autonomes Fahren – Technische, rechtliche und gesellschaftliche Aspekte", M. Maurer et al., Springer Open, 2015: https://link.springer.com/book/10.1007/978-3-662-45854-9

[2] „Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations", D. J. Fagnant et al., 2015, Elsevier: https://www.aamva.org/WorkArea/DownloadAsset.aspx?id=6738

[3] Apollo-Projekt: http://apollo.auto/

[4] Bericht der Ethik-Kommission Automatisiertes und Vernetztes Fahren, BMVI, Juni 2017: https://www.bmvi.de/SharedDocs/DE/Publikationen/DG/bericht-der-ethik-kommission.pdf

[5] OpenADx: https://openadx.eclipse.org/

Start-up-Leader und Innovatoren sprechen in München über „Innovation 4.0“

Von **Rachit Syag** und **Udo Bormann**

Elektor war kürzlich Co-Sponsor der Veranstaltung „The Top 50 Who’s Who“, einer Nacht des Networkings und des Ideenaustauschs in München. Die anwesenden Start-up-Gründer, Ingenieure und Innovatoren nutzten die Gelegenheit, um sich kennenzulernen, Drinks zu genießen und über die Zukunft von Technologie und Unternehmertum zu diskutieren.

Top-Innovatoren aus ganz Europa trafen sich kürzlich bei „Top 50 Who’s Who: Innovation 4.0“ in München, um zu diskutieren, wie man die Lücke zwischen Hardware, KI, IoT und Software schließen könnte. Im Rahmen der Start-Up-Night auf der *Bits & Pretzels 2019* ermöglichte die von *Sourceability*, *TechFounders*, *Brinc*, *Invest in Bavaria* und *Elektor* gesponserte Veranstaltung Existenzgründern, Ingenieuren und Innovatoren Visitenkarten auszutauschen, Drinks zu genießen und über die Zukunft von Technologie und Unternehmertum zu diskutieren. Diese gelungene Mischung aus Erfahrungsaustausch und Zusammenarbeit machte die Veranstaltung zu etwas Besonderem und ließ tief in die Zukunft von Technologie und Unternehmertum blicken!
In dieser ersten Folge der neuen Rubrik *Start-up-Zone* wollen wir einen Überblick über einige der Referate und Highlights dieses Abends geben.

Das Start-up-Potenzial in Europa und darüber hinaus

Jens Gamperl, Hauptgeschäftsführer von *Sourceability*, sprach über die aktuellen Themen KI, Maschinelles Lernen, Blockchains, Autonomes Fahren und Smart Cities. Er erklärte, dass die USA offenbar mehr Optionen in Bezug auf die Finanzierung, eine Fülle von neuen innovativen Technologielösungen und Erfahrungen mit schnellem Wachstum haben. Er glaubt jedoch, dass die Fähigkeit Europas, längerfristige Entwicklungen vorherzusehen, ein besonderes Verkaufsargument ist.
„Darüber hinaus garantiert Europas hochqualifizierter Talentpool die globale Wettbewerbsfähigkeit und stärkt seine Fähigkeit, nachhaltige Geschäftsmodelle aufzubauen“, sagte Gamperl.
Jewell Sparks (Global Director for Innovation & Strategic Partners von *Surcle.io*) meinte, Europa und die USA böten gute Startbedingungen für Gründer. Nach Jewells Beobachtung engagieren sich europäische Unternehmen intensiv bei Start-ups in den Bereichen Künstliche Intelligenz, Maschinelles Lernen und Augmented Analytics. Sie sprach auch davon, dass Vielfalt und Integration ein wichtiges Thema bei Innovationen in den USA sei. Auch in Kapitalbeteiligungsunternehmen, Accelerator-Programmen und Organisationen sind die Führungskräfte für Vielfalt, Integration, innovative Technologie und Skalierbarkeit verantwortlich, sagte Jewell.
Heiko Huber (Geschäftsführer der *TechFounders UnternehmerTUM Projekt GmbH*) unterstützt Start-ups bei ihrer Zusammenarbeit. In seinem Vortrag sagte er, der Hauptvorteil für die Teilnehmer solcher Veranstaltungen sei es, neue Menschen kennenzulernen, die bei Projekten und der Geschäftsentwicklung helfen könnten. „Dies kann durch Erfahrungsaustausch, durch die Betrachtung der eigenen Ideen aus einem anderen Blickwinkel (zum Beispiel durch Anwendung einer noch nicht berücksichtigten Start-up-Idee) sowie durch eine neue Kooperation oder ein neues Investment erfolgen“, sagte er.
Bernd Wunderlich (Leiter des Digitalen Projektmanagements, Gartner, **Bild 1**) hielt einen interessanten Vortrag zum Thema „Sicherung einer neuen Grundlage für das digitale Geschäft“. Neben dem Thema „Digitale Statistik“ erläuterte er die Ergebnisse einer Umfrage, die Gartner unter Geschäftsführern im deutschsprachigen Wirtschaftsraum durchgeführt hat, wobei nach den zehn wichtigsten zukünftigen Geschäftsaktivitäten gefragt wurde. Dabei rangierten digitale Initiativen an erster Stelle, erst dann gefolgt von Aktivitäten, die sich mit Umsatz und Geschäftswachstum befassten. Interessant ist, dass die Digitalisierung für die Entscheidungsträger oberste Priorität hat, so dass schon 49% ihr Geschäftsmodell geändert haben. Von diesen 49% nutzen 28% die genannte Digitalisierung zur Skalierung des Unternehmens, während 51% aller Befragten noch nicht mit einem Digitalisierungsprozess begonnen haben.

Vorträge der Innovatoren

Sechs Unternehmen hatten bei der Veranstaltung „Top 50 Who’s Who“ die Gelegenheit, ihre Produkte und Lösungen zu präsentieren, von Wearables bis hin zur KI. Im Kasten finden

Sie Links zu den Websites dieser innovativen Unternehmen.

Teiimo - Markus Strecker, Gründer und Hauptgeschäftsführer

Um Elektronik und Textilien zu vereinen, arbeitet Teiimo im Bereich der formbaren Elektronik. Auf der Veranstaltung präsentierte Markus Strecker ein neues medizinisches Sensorhemd, das die Herzfrequenz der Patienten in Echtzeit überwacht und so dem Gesundheitswesen von hohem Nutzen sein könnte. Teiimo gewann den ersten Preis im Wettbewerb „electronica Fast Forward 2018“.

Franck.AI - Isabell Franck, Gründerin und Geschäftsführerin

Im Zeitalter von Industrie 4.0 und Künstlicher Intelligenz bietet Franck.AI innovative Softwarelösungen an. Die Firma konzentriert sich auf die Verbindung von Maschinellem Lernen und kundenspezifischem Expertenwissen, um Software zur Optimierung von Produktions- und Entwicklungsaktivitäten zu erarbeiten.

Bild 1. Bernd Wunderlich spricht über die Sicherung des neuen digitalen Geschäfts.

Smartfurniture - Jörg Sahlmann, Mitbegründer

Wie der Name schon sagt, verbindet Smartfurniture Möbel mit Elektronik und Software, um intelligente Arbeits- und Freizeitgeräte zu entwickeln. Das Unternehmen konzentriert sich bei der Herstellung intelligenter Möbel auf vier Schlüsselelemente: Design, Funktion, Qualität und Preis.

Swap Language - Nicholas Møller Walsted, Hauptgeschäftsführer

Einen innovativen Ansatz bietet Swap Language zum Erlernen von Sprachen mit Hilfe eines Muttersprachlers. Mit einer einfachen, aber effektiven Lösung hilft das Unternehmen Menschen, einen Muttersprachler für die neue Sprache zu finden und diesem im Gegenzug die eigene Muttersprache zu lehren.

Valuer.ai - Signe Andersen, Regionalmanager, Deutschland

Mit Crowdsourcing und Künstlicher Intelligenz hilft Valuer.ai Akzeleratoren und Unternehmen, aussichtsreiche Start-ups zu finden, die ihren strategischen Innovationsanforderungen genügen.

Rydies - Andreas Nelskamp, Geschäftsführer

Rydies bietet in unserer Zeit des steigenden Verkehrsaufkommens, in dem immer mehr Menschen immer mehr Zeit auf Straßen und auf der Suche nach Parkplätzen verbringen müssen, Lösungen für die Mikromobilität: Fahrräder, E-Bikes und ähnliche Fortbewegungsmittel. Das Unternehmen konzentriert sich auf ein stressfreies Reiseerlebnis auf kurzen Strecken und kämpft gleichzeitig gegen die zunehmende Verschmutzung durch Autoabgase.

Zukunftschancen

Die Veranstaltung „Top 50 Who’s Who“ in München war eine weitere Gelegenheit für Tech-Innovatoren, die Unterstützung zu erhalten, die sie benötigen und verdienen. Elektor ist nicht nur eine nützliche und zuverlässige Quelle für Elektronikingenieure aus mehr als 80 Ländern, sondern setzt sich auch für die Förderung von Innovation und Unternehmertum ein. Durch die Plattform *Elektor Labs* und globale Start-up-Wettbewerbe wie *productronica Fast Forward* stärkt Elektor nachhaltig die Innovationskraft von Start-up-Unternehmen. ⏮

191151-02

Weblinks

[1] Teiimo: https://teiimo.com/

[2] Franck.AI: https://franck.ai/

[3] Smartfurniture: https://smartfurniture.de/

[4] Swaplanguage: www.swaplanguage.com

[5] Valuer.ai: https://valuer.ai/

[6] Rydies: www.rydies.com

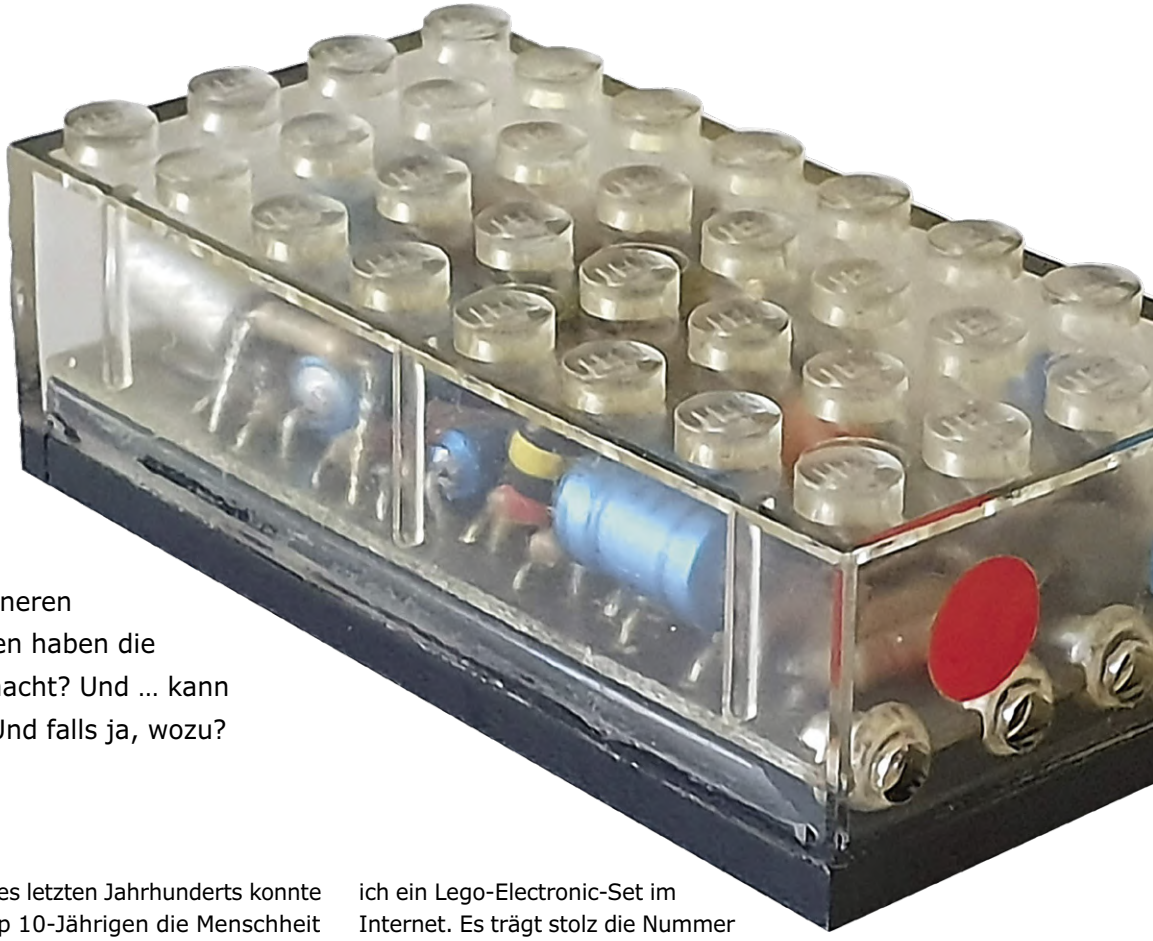
Mitarbeit

- Brinc - Vanessa Narvios. **www.brinc.io**
- Invest in Bavaria - Andreas Fischer. **www.invest-in-bavaria.com**
- Sourceability **/** Surcle.io - Jens Gamperl. **https://sourceability.com**; **www.surcle.io**
- Elektor - Udo Bormann & Rachit Syag. **www.elektormagazine.com**

Lego Electronic anno 1968

Elektronisches Spielzeug kann auch nach 50 Jahren noch faszinieren

Von **Martin Kompis**



Es war der Traum manchen Kindes und für die meisten unerreichbar: Ein Lego-Electronic-Set, wie es Ende der sechziger Jahre auf den Markt kam. Ein Pfiff, und die Lokomotive fuhr los. Und die kleinen Kästchen funktionieren heute immer noch. Doch wie sah es im Inneren aus und welche Überlegungen haben die Entwickler wohl damals gemacht? Und … kann man das auch nachbauen? Und falls ja, wozu?

Gegen Ende der sechziger Jahre des letzten Jahrhunderts konnte aus der Sicht eines damals knapp 10-Jährigen die Menschheit (also: der Teil der Menschheit, der damals ebenfalls rund 10 Jahre alt war) grob in zwei Gruppen aufgeteilt werden: Auf der einen Seite die große Mehrheit, die keinen Lego-Electronic-Block besaß und auf der anderen Seite die verschwindend kleine, aber von allen anderen beneidete Minderheit, die einen solchen ihr Eigen nannte. Nun können Sie drei Mal raten, zu welcher Gruppe der Autor damals gehörte. Gut, erraten. Nicht zur beneideten Minderheit.

Wie uns allen inzwischen klar geworden sein dürfte, bringt das Erwachsenenalter viel Unbill mit sich: Steuern zahlen, in Ermangelung geeigneter Erbonkel einer regelmäßigen Erwerbstätigkeit nachgehen, den Müll hinuntertragen, am Neujahresempfang nicht gähnen und über die richtigen Witze lachen, nur um einige gängige Beispiele zu nennen. Aber einige Vorteile hat es eben doch, erwachsen zu sein. Und einer davon ist, dass man sich nun endlich all die Spielsachen kaufen kann, die man gerade dringend benötigt.

Da das Wunschobjekt im vorliegendem Fall die Schaufenster der Spielwarengeschäfte schon lange nicht mehr ziert, ersteigerte ich ein Lego-Electronic-Set im Internet. Es trägt stolz die Nummer 139A und wird mit Mikrofon, Lego-Pfeife und sogar dem Originalkarton geliefert (**Bild 1**).

Die Funktion des Systems ist an sich ganz einfach und könnte wohl nicht besser dargestellt werden, als auf der Innenseite des Deckels in Bild 1: Pfeift man einmal auf der Lego Pfeife, setzt sich der Zug in Bewegung. Pfeift man ein zweites Mal, dann bleibt der Zug wieder stehen. Anstatt zu Pfeifen genügt es aber auch, zum Beispiel einfach in die Hände zu klatschen. Nun, das mag heute, im Zeitalter der Smartphone-gesteuerten und Kamera-besetzten Drohnen für Kinder ab drei Jahren (vielleicht auch erst ab sechs) nicht besonders aufsehenerregend klingen. Am Ende der sechziger Jahre aber, als Fernbedienungen für Fernseher theoretisch zwar schon existierten, hierzulande aber noch kaum gesichtet wurden, war dies allemal ein wirklich verlockendes Spielzeug, das stundenweise Spaß versprach.

In den Jahren 1968 und 1969 waren zwei verschiedene Versionen der Lego-Elektronik erhältlich, welche insgesamt in vier verschiedenen Lego-Sets mit den Nummern 118, 138, 139

und 139A in den Verkauf kamen. Die ältere Version aus dem Jahre 1968 findet sich in den Sets 118 und 139A. Letzteres ist in Bild 1 zu sehen und enthielt einen Elektronik-Block, ein Mikrofon in einem weißen Plastikgehäuse, zwei Paar Kabel und eine Pfeife, von der gleich noch die Rede sein wird. Das größere Lego Set Nummer 118 enthielt eine komplette Lokomotive mit Tender (**Bild 2**). Nur ein Jahr später kam dann ein neuer Elektronik-Block in die Verkaufsregale, welcher wiederum entweder einzeln unter der Nummer 139 oder komplett mit Lokomotive und Tender als Set Nummer 138 erhältlich war, letzterer diesmal mit einem neuen, kleineren Motor.

Während die ältere Version der Elektronik vollständig aus diskreten Bauteilen aufgebaut ist, enthält diejenige aus dem Jahre 1969 einen integrierten Schaltkreis mit der kryptischen Bezeichnung „211 OM“ [1]. Damit kann der Zug nicht nur vorwärts fahren, sondern, ebenfalls per Pfeife ferngesteuert, nun auch rückwärts.

Ein kurzer Blick auf das von mir ersteigerte Kästchen zeigt, dass es sich hier um die ältere, noch diskret aufgebaute Version handelt. Das ist nicht nur für die Retronik-Rubrik passender, sondern auch für die Analyse und den Nachbau der Schaltung einfacher, da das IC möglicherweise ein eigens für diese Anwendung entwickeltes (oder programmiertes?) Bauteil als eine Art Vorläufer heutiger FPGAs ist.

Vor der Elektronik kommt die Akustik

Der Elektronik-Block ist aber erst der zweite Teil der ganzen Anlage. Den ersten Teil bildet der akustische Signalgeber, also die mitgelieferte Lego-Pfeife (Bilder 1 und 2). Und die hat es in sich. Die vom Autor ersteigerte Pfeife funktioniert mit der Schaltung zusammen tadellos, ist aber subjektiv, nun ja, beunruhigend laut. Eine erste Messung zeigt denn auch, dass bereits bei einem mittleren Luftstrom in 1 m Entfernung ein Pegel von rund 95 dBA erreicht wird. Nicht ganz wenig. Vor meinem eigenen Ohr, das ja deutlich weniger als 1 m von der Pfeife entfernt ist, messe ich sogar 105 dBA. Nicht nur der Pegel, auch die Frequenz des Pfeiftons variiert leicht mit der Stärke des Luftstroms. Sie liegt bei meinem Exemplar um die 5,7 kHz. Das menschliche Ohr scheint bei Frequenzen um die 3 bis 6 kHz am anfälligsten auf akustische Überlastung zu sein [3] und 5,7 kHz scheinen bei diesen doch recht hohen Pegeln nicht besonders glücklich gewählt. Immerhin genügen für das zuverlässige Umschalten schon etwas tiefere Pegel, bei meinem Exemplar reichen um die 82 dBA.

Die Elektronik

Doch der mit Abstand spannendste und wertvollste Teil des ganzen Systems ist und bleibt natürlich der Elektronik-Block. Er ist prall gefüllt mit elektronischen Bauteilen, selbstverständlich alle älter und alle bedrahtet, da die SMD-Technologie im Jahre 1968 noch nicht eingeführt war [2]. Die ganze Oberseite des Gehäuses ist durchsichtig, worin bereits ein Teil seiner Anziehungskraft auf technikaffine Kinder bestand. Auch in der fertigen Lokomotive wird die Elektronik nicht etwa schamhaft versteckt, sondern stolz und für jeden gut sichtbar zur Schau gestellt (Bild 2).

Die Verbindung zum Motor erfolgt über zwei Metallnoppen auf der Unterseite des Elektronik-Blocks, die Verbindungen zum Mikrofon und zum Batteriekasten über zwei 2,5-mm-Buchsen-Paare auf dessen Vorderseite. Sie sind mit einem blauen

Bild 1. Das ersteigerte Lego-Electronic-Set aus den sechziger Jahren: links der eigentliche Elektronik-Block, vorne das Mikrofon und rechts vorne eine Pfeife, die es in sich hat…

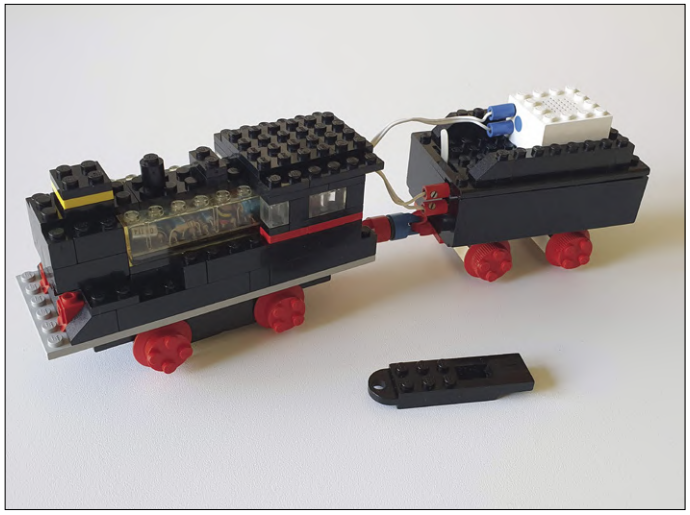


Bild 2. Nachbau (leider kein Original) der Lok Nr. 118. Stolz wird das Elektronik-Modul zur Schau gestellt.

Punkt für das Mikrofon und einem roten Punkt für die Versorgung markiert (Bild 1).

Das wirft erste Fragen bezüglich der Schaltung auf. Die Polung der Versorgungsspannung ist nirgends markiert. Dies wäre auch sinnlos, da die Polarität der Betriebsspannung mit einem kurzen Umlegen am Hebel des Batteriefachs geändert werden kann. Mechanisch sind alle Buchsen gleich, man kann also problemlos das Mikrofon dort anschließen, wo der Motor hingehört, die Batterie dort, wo das das Mikrofon hingehört und so weiter. Und das Ganze ist für experimentierfreudige Kinderhände bestimmt und sollte darum ziemlich fehlertolerant sein. Ein kurzer Versuch zeigt, dass die Schaltung zwar nur mit einer Polarität der Versorgungsspannung wie vorgesehen funktioniert, aber auch bei umgekehrter (falscher?) Polung etwas halbwegs Sinnvolles tut: Der Zug fährt rückwärts, wenn auch deutlich langsamer.

Belastet man den Motor mechanisch, fließt rasch ein Strom von immerhin rund 80 mA. Im inneren des Elektronikblocks ist

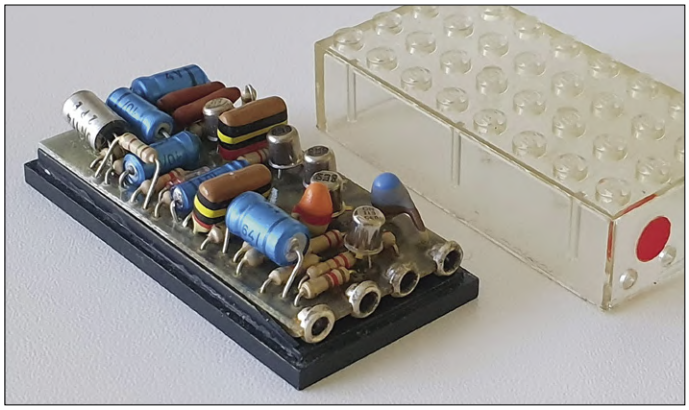


Bild 3. Die Lego-Elektronik nach dem Öffnen des Gehäuses. Ganz links hinten der Germaniumtransistor.

aber durch die durchsichtige Plastikhülle auf den ersten Blick kein Leistungstransistor zu erkennen, ebenso wenig Dioden zum Schutz der Schaltung vor falscher Polung. Wie wurde das also im Jahre 1968 gemacht?

Um diese und andere Fragen zu beantworten, wäre der Schaltplan nützlich. Im Internet werde ich (zunächst) nicht fündig und so muss ich das schöne Gehäuse öffnen. (Wie sich später herausstellen wird, habe ich einfach nicht gut genug gesucht: der Schaltplan ist sehr wohl im Internet zu finden [1], sogar in zwei Varianten. Allerdings geben beide die Polarität der Transistoren nicht richtig wieder, so dass die Schaltung wie angegeben nicht funktionieren würde.)

Der Gehäuseoberteil ist auf die Bodenplatte geklebt und ich muss mit einem Schweizer Taschenmesser zur Tat schreiten. Für diese Freveltat an einem historischen Lego-Bauteil komme ich bestimmt in die Lego-Hölle (oder sagt man Spielhölle?). Es gelingt, den oberen Teil abzunehmen, ohne dass eines der Plastikteile ernsthaft Schaden nimmt und die Elektronik liegt nun in voller Pracht vor mir (**Bild 3**). Hebt man die kleine, nur einseitig beschichtete Platine etwas an, so findet sich darunter ein vergilbter Zettel mit dem Text „SEP. 1968“. Sehr hübsch. Mit Pauspapier, Messgerät und einem scharfen Blick wird nun die Schaltung nachgezeichnet. Als eine (kleine) Herausforderung erweist sich dabei die Platzierung der Bauteile: Diese sind aus Platzgründen teilweise übereinandergelegt, so dass einige wenige Widerstände und eine Diode nur durch Wegbiegen der darüber liegenden Teile gut sichtbar werden.

Bild 4 zeigt den so entstandenen Schaltplan. Auf der kleinen Platine finden sich insgesamt sechs Transistoren in Metallgehäusen. Fünf davon in kleinen TO-18 Gehäusen sind als „ON 113“ beschriftet, der letzte im größeren TO-1 Gehäuse ist als „ON 114“ gekennzeichnet.

Die Bezeichnungen folgen keinem mir bekannten System und beide Transistortypen sucht man denn auch im Internet vergeblich. Eine Messung zeigt aber, dass es sich beim ON 114 um einen PNP-Germaniumtransistor handeln muss, bei den fünf anderen Transistoren hingegen um NPN-Typen und — für mich etwas überraschend — um Siliziumtransistoren.

Die Funktion der Schaltung ist im Grundsatz gut nachzuvollziehen. Die Transistoren T1 und T2 bilden einen Verstärker für das Mikrofonsignal. High Fidelity ist hier nicht gefragt, vielmehr soll offenbar einfach die Verstärkung hoch sein. Und so sieht das Signal am Kollektor von T2 denn auch reichlich verzerrt aus. Im Wesentlichen werden nur Teile der negativen Halbwellen verstärkt, und auch das nicht besonders sauber. Dafür erreicht das Signal aber locker peak-to-peak Werte von rund 2,5 V.

T3 bildet zusammen mit Kondensator C4 und Widerstand R9 einen Schalter, der den bistabilen Multivibrator aus T4 und T5 ansteuert. C4 ist im Ruhezustand über R9 (und damit natürlich auch über R10) fast bis auf die Betriebsspannung aufgeladen. Ertönt ein Pfiff, wird T3 bei jeder Halbwelle kurz leitend und entlädt so C5 rasch. Das Wiederaufladen von C5 erfolgt dann sehr viel langsamer, mit der durch C5 und R9 bestimmten Zeitkonstante von knapp 50 ms, nachdem das akustische Signal verstummt ist. Über C7 und C8 wird der bistabile Multivibrator angesteuert. Leitet T4, steht der Motor still. Leitet hingegen T5, so wird die CE-Strecke des Endstufentransistor T6 leitend und der Zug fährt.

Obwohl fünf der sechs Transistoren der Schaltung siliziumbasiert sind, haben die Entwickler für die Ansteuerung des Motors einen Germaniumtransistor gewählt. Dies ist deswegen sinnvoll, weil der Spannungsabfall bei durchgeschaltetem Transistor und Germaniumtypen deutlich geringer ist als bei Siliziumtransistoren. In der vorliegenden Schaltung liegt er gerade einmal bei fast unglaublichen 0,03 V. Entsprechend gering ist die Verlustleistung, die an T6 anfällt.

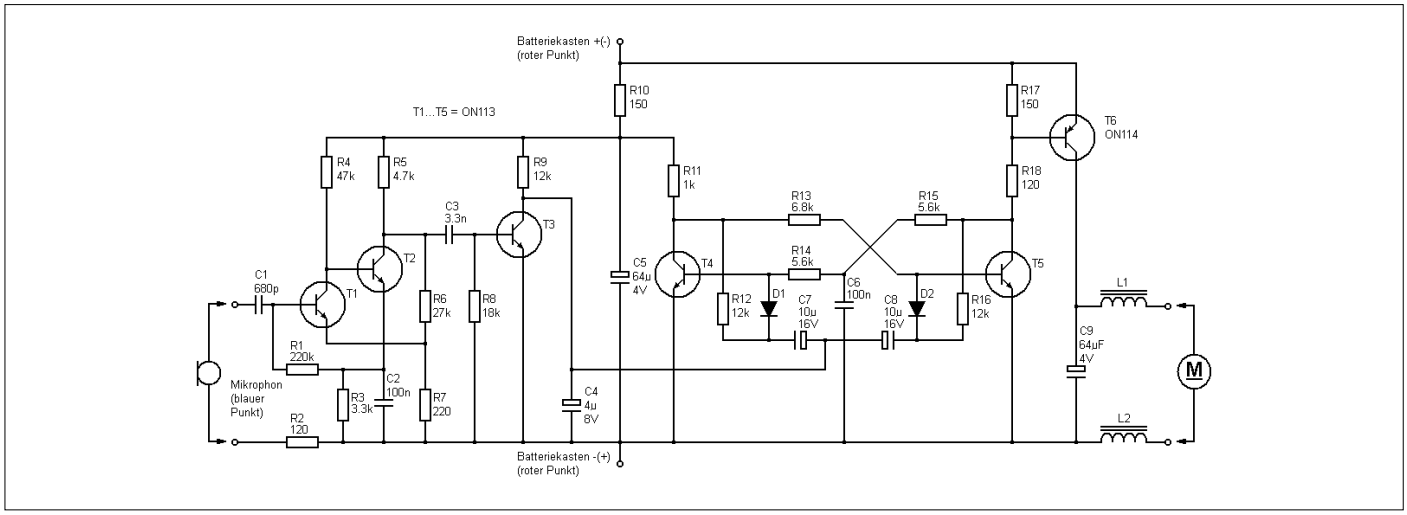


Bild 4. Das Schaltbild. Bei den Transistoren T1 bis T5 handelt es sich um Silizium-Transistoren, T6 ist ein Germaniumtransistor.

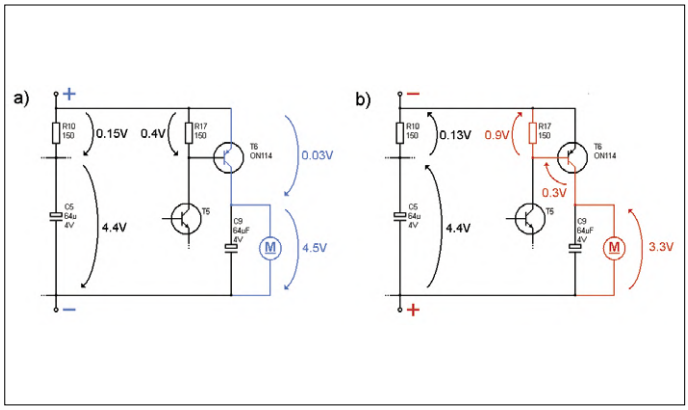


Bild 5. Einige Spannungen (a) bei der Vorwärtsfahrt und (b) bei umgekehrter Polarität der Versorgungsspannung, bei welcher der Zug langsamer wird und in die Gegenrichtung fährt.

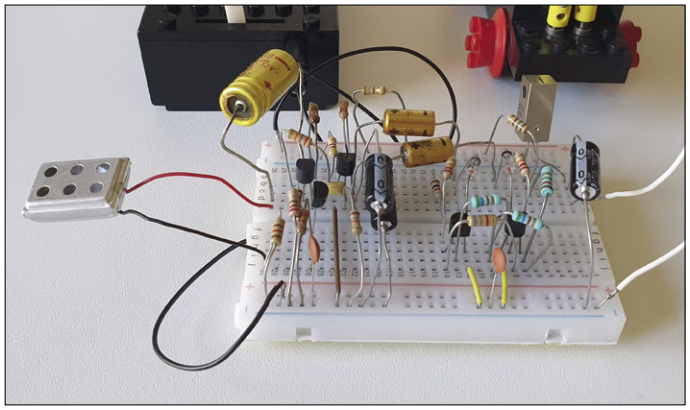


Bild 6. Der Nachbau der Lego-Elektronik. Rechts hinten der Germaniumtransistor AC153K im TO-1 Gehäuse mit Kühlklotz, der für diese Anwendung gar nicht nötig wäre.

Eine Frage der Polarität

So weit so einfach. Dennoch bleibt einiges unklar. Was geschieht bei „falscher“ Polung der Betriebsspannung? Wieso fährt der Zug und warum langsamer als nach vorne? Was geschieht mit den einzelnen Schaltungsteilen bei „falscher“ Polung? Der erste Teil der Schaltung mag durch R10 ein wenig geschützt sein, aber was geschieht mit den Elektrolytkondensatoren C5 und C9? Sind die dann nicht falsch gepolt? **Bild 5** zeigt die beiden Zustände und einige relevante Spannungen im Betrieb.

Bei richtiger Polung (**Bild 5a**) funktioniert alles genau so wie beschrieben. Im Rückwärtsgang (**Bild 5b**) hingegen wirkt die BC-Strecke von T6 als Diode mit dem erwarteten Spannungsabfall von rund 0,3 V. Der Strom, der durch den Motor fließt muss jetzt aber zusätzlich auch noch durch den 150-Ω-Widerstand R17 fließen, wo weitere rund 0,9 V abfallen. Entsprechend läuft dann der Motor deutlich langsamer, aber er läuft! Und die beiden Elektrolytkondensatoren? Nun, es geschehen keine Wunder. Fährt die Lok langsam rückwärts, so sind die beiden Elkos C5 und C9 tatsächlich falsch gepolt, und die Spannungen an Ihnen sind nicht ganz unerheblich. Das haben die Entwickler offenbar in Kauf genommen, und es scheint zu funktionieren, auch noch nach über 50 Jahren.

Die Messungen auf der engen Leiterplatte gestalten sich nicht ganz einfach. Zudem reizt es natürlich, die Schaltung nachzubauen, um sicher zu sein, dass der Schaltplan so auch stimmt. Die logische nächste Frage lautet darum: muss man das wirklich nachbauen? Wie bei allen schwierigen Lebensfragen konsultiere ich zuerst meine Familie. Die Antworten reichen diesmal von „Mach die Türe zu“ (Nachwuchs im schwierigen Alter) über „Wozu soll das gut sein?“ (liebende Ehefrau. Merke: jedes Alter kann zuweilen etwas schwierig erscheinen) bis zum krampfhaften Schließen beider Augen (Katze, Alter nicht so genau bekannt, aber kein ganz aktuelles Modell). Die Resonanz kann also im Quervergleich zu früheren Projektideen meinerseits nur als äußerst positiv gewertet werden und ich mache mich sofort an die Arbeit.

Der Nachbau

Bild 6 zeigt den Nachbau auf einem kleinen Steckbrett. An Stelle der fünf ON 113 NPN-Siliziumtransistoren T1 bis T5 wurden BC548C-Typen verwendet und anstelle des Germaniumtransistors ON 114 ein Einzelexemplar eines AC153K, der die letzten Jahrzehnte tief unten in der Bastelkiste verbracht hatte. Die Schaltung funktioniert auf Anhieb und zwar genau so, wie das über 50-jährige Original: ein Pfiff, und der Motor legt los, ein zweiter Pfiff und er hält. Legt man den Batterieschalter um, dreht der Motor in die andere Richtung, und zwar deutlich langsamer. Bingo. Als Mikrofon wurde übrigens ein kleines Kristallmikrofon verwendet. Vielleicht ist die Schaltung für jemanden nützlich, der ein nicht mehr funktionierendes Exemplar der Lego-Elektronik sein Eigen nennt. Und auch die nachgebaute Schaltung ist nicht ohne Spielwert. Viel Spaß!

190382-01

ESTD 2004

www.elektor.tv

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

Weblinks

[1] Schaltkreis „211 OM“: www.eurobricks.com/forum/index.php?/forums/topic/10989-electronic-train-118138-and-139-anyone-have-one-or-have-seen-one/&page=2

[2] SMT-Geschichte: www.all-electronics.de/eine-kleine-geschichte-der-smt/

Literaturhinweis

[3] Kompis M., „Audiologie“, Hogrefe Verlag Bern, 4. Auflage, 2016, ISBN: 978-3-456-85553-0, 4. Kapitel

Die MX3D-Brücke überwacht die Stadt

Wie Datenerfassung die Privatsphäre retten kann!

Von **Tessel Renzenbrink**

Die MX3D Brücke ist die allererste 3D-gedruckte Metallbrücke, gebaut von Industrierobotern, die über einen Zeitraum von sechs Monaten Millionen von Schweißnähten aus Edelstahl anbrachten, ein einzigartiges Ingenieursobjekt. Um die neuartige Konstruktion auf Herz und Nieren zu testen, wurde sie mit einem dauerhaften Sensornetzwerk ausgestattet. Für Alec Shuldiner, leitender Produktmanager beim Softwareunternehmen Autodesk, eröffnen die Sensordaten eine Vielzahl weiterer neuer Möglichkeiten.

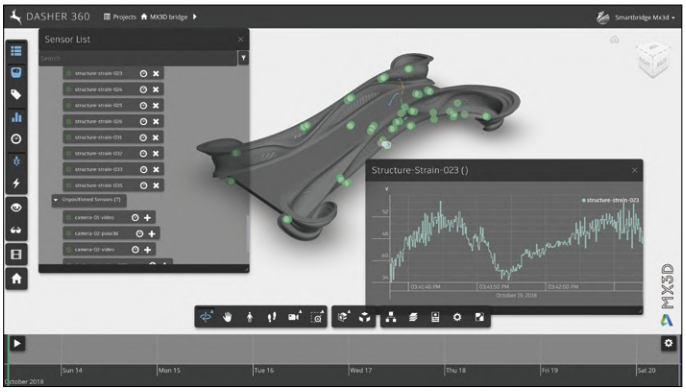


Bild 1. Das Dashboard zeigt die Sensorsignale der MX3D-Brücke.

In diesem Interview erklärt er, dass die Daten nicht nur Informationen über die Brücke selbst liefern, sondern auch über ihre Umgebung. Dies könnte dabei helfen, das Management öffentlicher Räume zu verbessern. Wenn man zum Beispiel weiß, wie viele Menschen die Brücke überqueren, findet man etwas über die Frequentierung der Umgebung heraus und trägt zur Steuerung der Verkehrsströme bei. „Was viele Menschen nicht wissen, ist, dass die Datenerfassung den Zerfall der Privatsphäre im öffentlichen Raum tatsächlich reduzieren kann“, sagt Shuldiner. „Aber um das zu erreichen, müssen wir die Struktur des Internet der Dinge überdenken.“

Das MX3D-Projekt stützt sich auf das Fachwissen vieler unterschiedlicher Beteiligter, zuallererst des Amsterdamer Unternehmens MX3D. Sie entwickelten die Technik des 3D-Drucks im Freien und vermieden dabei das hinderliche „Druckergehäuse“ und andere Stützkonstruktionen. Das Sensornetzwerk für die Fußgängerbrücke wird zusammen mit dem Imperial College of London, der Universität Twente und dem für seine AutoCAD-Anwendung bekannten US-Unternehmen Autodesk konzipiert und installiert. Autodesk liefert außerdem auch die Designsoftware und kooperiert mit dem britischen Alan Turing Institute zur Entwicklung von Algorithmen zum Maschinellen Lernen in der Datenverarbeitung. Ein weiterer Partner ist die Stadt Amsterdam als Kunde von MX3D. Geplant ist, diese neuartige Brücke über einer Gracht in De Wallen, dem ältesten Bezirk der Stadt, zu installieren.

IoT als Fenster zur Welt

„Das Hauptargument für die Anbringung von Sensoren“, so Shuldiner, „ist, dass die Bauweise der Brücke so innovativ und einzigartig ist, dass wir Sensordaten sammeln mussten, allein um zu verstehen, ob und wie die Technik funktioniert. Das Ziel liegt darin, eine Basis ingenieurtechnischen Wissens zu legen, um die Sicherheit der Brücke zu gewährleisten und Daten für zukünftige ähnliche Projekte zu sammeln. Das bleibt das Hauptinteresse für die Brücke, aber mein persönliches Interesse gilt dem Aspekt des Internets der Dinge (Internet of Things, IoT). Sie können durch die Daten, die die Sensoren sammeln, herauszufinden, was auf der und um die Brücke herum passiert.“

„Der Grund für mein Interesse ist, dass wir durch diese Erfassung der städtischen Umwelt viel besser lernen, die Stadt zu verwalten. Die Städte werden immer voller, was zu massiven Problemen beim Verkehrsfluss führt. Im Amsterdamer Viertel De Wallen, in dem sich ein Rotlichtviertel befindet, ist dieses Problem am gravierendsten. Es ist einer der überfülltesten Orte der Welt. Das ist nicht nur unangenehm, es kann auch zu einem Sicherheitsproblem werden. Deshalb versucht die Stadt seit langem herauszufinden, wie man die Menschenmassen quantifizieren kann, um sie besser zu leiten. Was wir mit dem MX3D-Projekt erreichen wollen, ist deshalb, dass die Brücke die Anzahl der dort anwesenden Personen erfasst und übermittelt.“

Kameras nur zum Training

„Die Brücke nutzt die Sensordaten, um festzustellen, wie viele Personen sich zu einem bestimmten Zeitpunkt darauf befin-

Bild 2. Sensoren an der MX3D-Brücke.

Bild 3. Alec Shuldiner vor der MX3D-Brücke.

den“, fährt Shuldiner fort. „Im Grunde genommen gibt es zwei Dinge, die die Sensoren messen. Ein Aspekt sind die Umgebungsparameter, die Temperatur und deren Änderung, der Schallpegel, die Lichtintensität und so weiter. Die wichtigsten Umgebungssensoren sind Kameras, die Aktivitäten auf der und um die Brücke herum aufzeichnen. Die anderen Sensoren erfassen die inneren Bewegungen der Brücke als Reaktion auf ihre Umgebung: Vibrationen, Frequenzen, Neigungen, Dehnungen und Verschiebungen. Die Idee dabei ist, mit den Kameras Daten über das Geschehen auf der Brücke zu sammeln und diese Informationen dann durch Maschinelles Lernen mit den physikalischen Veränderungen in der Brücke zu korrelieren. Irgendwann wird die Brücke dann in der Lage sein, uns nur anhand der registrierten physischen Parameter mitzuteilen, wie viele Personen sich auf ihr befinden. Und dann werden die Kameras, die nur zu Trainingszwecken montiert worden sind, nicht mehr gebraucht und können entfernt werden.

Derzeit sind Kameras die Standardtechnologie für diese Art von Zählungen, nicht nur in Amsterdam, sondern überall auf der Welt. Aber Kameras sind als Mittel zur Datenerfassung problematisch, sowohl in praktischer als auch in gesellschaftlicher Hinsicht. Zum einen sind es keine sehr präzisen Sensoren, denn Bilddaten sehen sehr unterschiedlich aus, wenn sich äußere Umstände wie Licht oder Wetter ändern. Außerdem ist es eine ‚schwere‘ Datenquelle, da die Verarbeitung von Videodaten sehr rechenintensiv ist. Neben solchen technischen Fragen gibt es aber auch das sehr ernste Problem, dass Kameras alle Daten liefern, die man braucht, um jemanden mit einem hohen Maß an Genauigkeit zu identifizieren. Früher bewegten wir uns im öffentlichen Raum mit der Erwartung, dass wir nicht bespitzelt würden. Aber das ist heute keine realistische Annahme mehr! Der massive Einsatz von Kameras zerstört die Privatsphäre im öffentlichen Raum.

Selbst wenn die Kameras Verschleierungstechniken verwenden, gibt es psychologische Auswirkungen. Im MX3D-Projekt ‚skelettieren‘ wir die Videodaten: Man sieht keine Menschen, sondern nur Strichmännchen. Aber die Fußgänger in der Gegend wissen das nicht. Sie sehen eine Kamera und gehen natürlich davon aus, dass ihr Bild hochauflösend aufgenommen wird. Selbst wenn eine Kamera ihre Privatsphäre objektiv nicht beeinträchtigt, beeinträchtigt sie doch ihr Gefühl der Privatsphäre, was fast genauso schlimm ist. Deshalb wollen wir die Kameras entfernen, nachdem das Training beendet ist.“

IoT-Design überdenken

Shuldiner: „Der Einsatz von Kameras deutet aber auf ein noch größeres Problem hin, das ich mit dem Internet der Dinge verbinde. Wir haben es mit der einfachen Aufgabe zu tun, Menschen zu zählen, lösen sie aber mit schwer handhabbaren und nicht gut geeigneten Sensoren, die außerdem negative Nebenwirkungen haben. Wir sammeln mit Kameras für die Fragestellung nicht gut passende Daten. Wir denken nicht weiter darüber nach, welche spezifischen Sensoren wir wirklich benötigen. Selbst bei MX3D, einem hochmodernen Projekt, waren die Sensoren nur ein nachrangiger Gedanke. Wir sollten den Prozess beim Entwurf von IoT-Objekten umkehren, mit den Fragen beginnen, die man beantwortet sehen will. Dann bestimmen Sie, welche Daten benötigt werden, und stellen den geeigneten Satz von Sensoren zusammen, um diese Informationen zu erfassen. Und erst dann kommen wir zum Entwurf des physikalischen Objekts, das diese IoT-Funktionen beherbergt, und schaffen den Platz für die Sensoren und die Verkabelung, damit das Sensornetzwerk in das Objekt passt. Was ich mir wirklich wünsche, sind Designtools, die den Entwickler veranlassen, darüber nachzudenken, welche Fragen das Objekt beantworten soll. Und dann sollten diese Überlegungen automatisch einen Satz von Sensoren vorschlagen, die in den Entwurf einbezogen werden.“ ⏮

191141-03

(Alle Bilder mit freundlicher Genehmigung von Alec Shuldiner.)

Willkommen in Ihrem E-SHOP

ELEKTOR EMPFIEHLT

Andonstar AD407 Digital-Mikroskop mit 7”-Display

Mit dem großflächigen 7“ Farb-LC-Display wird das neue Andonstar-Modell AD407 zu einem eigenständigen, unabhängigen Mikroskop. Im Elektor-Labor haben wir dem Vorgänger ADSM302 noch einen großen Bildschirm zur Seite gestellt. Beim AD407 ist ein externer Bildschirm überflüssig, der Platz auf dem Arbeitstisch wird frei. Das kann schon Grund genug sein, dem AD407 eine klare Präferenz zu geben, falls der Kauf eines neuen Geräts ansteht. Sofern das Budget es zulässt, würden wir ein vorhandenes USB-Mikroskop ADSM302 durch ein neues Modell AD407 ersetzen.

Luc Lemmens
(Elektor Labs)

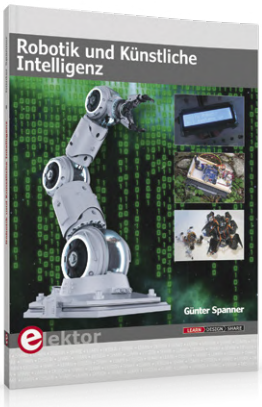
www.elektor.de/19079

Elektor-Bestseller

1. Elektor Schaltungs-Sonderheft 2020
   www.elektor.de/19140

2. Raspberry Pi 4 B
   www.elektor.de/rpi4
3. Robotik und Künstliche Intelligenz
   www.elektor.de/19120
4. Andonstar AD407 Mikroskop
   www.elektor.de/19079
5. Raspberry Pi Zero WH
   www.elektor.de/18567
6. Mendocino-Motor AR O-8
   www.elektor.de/19129

Robotik und Künstliche Intelligenz



Dieses Buch ist eine Einführung in das hochaktuelle Gebiet der Robotik. Dabei stehen praktische Anwendungsbeispiele im Vordergrund. Neben den technischen und mechanischen Grundlagen werden die elektronischen Komponenten und Module erläutert. Eine zentrale Rolle spielt dabei der Mikrocontroller.

Mitgliederpreis: 32,80 €

www.elektor.de/19120

Grafisch programmieren, messen und steuern mit Processing

Dieses Buch führt den Leser in diese visuelle Programmiersprache ein. Das Buch richtet sich an Leser, die bereits allgemeine Erfahrungen im Umgang mit Programmiersprachen besitzen und wissen, worum es sich bei Strings, Arrays oder Schleifen handelt. Der Autor zeigt anhand vieler kurzer Programmbeispiele, wie einfach es ist, mit Processing auch leistungsfähige Software zu programmieren.

Mitgliederpreis: 34,80 €

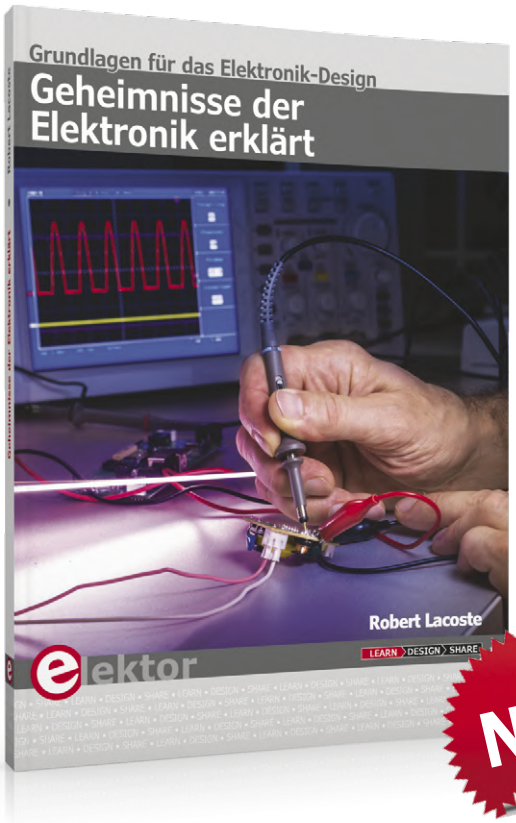
www.elektor.de/19037

Mikrocontroller-Basics mit PIC

Der Elektor-Autor Tam Hanna zeigt in diesem Buch alle wichtigen Aspekte der Mikrocontroller-Programmierung, ohne den Leser mit unnötigen oder nebensächlichen Informationen zu überladen. Am Ende der Lektüre ist der Leser in der Lage, 8-Bit-Mikrocontroller zu verstehen und zu programmieren.

Mitgliederpreis: 32,80 €

www.elektor.de/18945



Geheimnisse der Elektronik erklärt

Autor Robert Lacoste, ein hochrangiger Elektronikingenieur, hat für professionelle Elektronik-Zeitschriften eine Serie von Grundlagenartikeln geschrieben, die hier in einem Buch zusammengefasst sind. Wichtige Themen wie Taktgeber, Filter, analoge Signalverarbeitung, digitale Kommunikation und viele weitere werden verständlich erklärt.

Der Autor zeigt Ihnen dabei, wie Sie die Thematik besser verstehen und Ihr Wissen erweitern können, ohne mathematischen Ballast. Mit einfachen Worten erklärt der Autor, wie es funktioniert, und warum es manchmal nicht so funktioniert, wie man es will. Damit stoßen Sie nicht nur an Ihre eigenen Grenzen, sondern wissen auch, wo die Grenzen der von Ihnen verwendeten Geräte liegen. So wird es Ihnen ermöglicht, den tatsächlichen technischen Fortschritt von rein kommerziellen Aussagen zu trennen.

NEU

Mitgliederpreis: 34,80 €

www.elektor.de/19138

JOY-iT 3-in-1-Gerät (Oszilloskop + Signalgenerator + Multimeter)

Dieses praktische und flexibel einsetzbare 3-in-1-Gerät vereint die Funktionen eines Oszilloskops, eines Funktionsgenerators und eines Multimeters. 2 Akkus (18650 Zellen) ermöglichen einen Einsatz von bis zu einem Tag. Geladen werden die Akkus über einen USB-C Port (über den das Gerät während des Ladens auch betrieben werden kann).

Mitgliederpreis: 215,10 €

www.elektor.de/19157

Elektor SDR-Praxis-Bundle

Das Elektor SDR-Shield ist ein vielseitiger Kurzwellenempfänger bis 30 MHz. Zusammen mit einem Arduino-Board und der passenden Software lassen sich nicht nur Rundfunkstationen empfangen, sondern auch Morsesignale, SSB-Stationen und digitale Signale. Der Erfolgsautor und Amateurfunker Burkhard Kainka beschreibt in seinem Buch die moderne Praxis des Software Defined Radios mithilfe des im Kit enthaltenen Elektor SDR-Shields.

Mitgliederpreis: 49,46 €

www.elektor.de/19042

Elektronik-Grundlagen und Einsteiger-Projekte

Alles beginnt mit der analogen Elektronik. Man sollte die einfachsten Bauteile und Schaltungen genau kennen und ihr Verhalten sowie mögliche Probleme verstehen. Der beste Weg dazu sind reale Experimente, die Theorie allein reicht nicht. Dieses Buch bietet eine große Zahl praktisch nutzbarer Einsteiger-Schaltungen, mit denen jeder die nötigen Erfahrungen sammeln kann.

Mitgliederpreis: 34,80 €

www.elektor.de/19035

Hexadoku

Sudoku für Elektroniker

Wie in jeder Ausgabe finden Sie auch in diesem Heft unser ganz spezielles Sudoku. PC, Oszilloskop und Lötkolben können sich erholen, während Ihre kleinen grauen Zellen auf Hochtouren arbeiten. Wenn Sie alle Hex-Ziffern in den grauen Kästchen herausgefunden haben, sollten Sie uns diese gleich zumailen – denn hier warten fünf Elektor-Gutscheine!

Die Regeln dieses Rätsels sind ganz einfach zu verstehen: Bei einem Hexadoku werden die Hexadezimalzahlen 0 bis F verwendet, was für Elektroniker und Programmierer ja durchaus passend ist. Füllen Sie das Diagramm mit seinen 16 x 16 Kästchen so aus, dass alle Hexadezimalzahlen von 0 bis F (also 0 bis 9 und A bis F) in jeder Reihe, jeder Spalte und in jedem Fach mit 4 x 4 Kästchen (markiert durch die dickeren schwarzen Linien) **genau einmal** vorkommen. Einige Zahlen sind bereits eingetragen, was die Ausgangssituation des Rätsels bestimmt. Wer das Rätsel löst – sprich die Zahlen in den grauen Kästchen herausfindet – kann einen von fünf Gutscheinen im Wert von 50 Euro gewinnen!

Einsenden

Schicken Sie die Lösung (die Zahlen in den grauen Kästchen) per E-Mail, Fax oder Post an:

Elektor Redaktion
Kackertstr. 10
52072 Aachen

Fax: 0241 / 955 09-013
E-Mail: hexadoku@elektor.de

Als Betreff bitte nur die Ziffern der Lösung angeben!

Einsendeschluss ist der 27. Januar 2020.

Die Gewinner des Hexadokus aus der Ausgabe November/Dezember 2019 stehen fest!

Die richtige Lösung ist: **DB072.**

Einen Elektor-Wertgutschein über je 50 € haben gewonnen: Martin Bratko, Volker Treiber, Arjen de Rijke, Jean-Pierre Demangeon und Alexandr Papazyan.

Herzlichen Glückwunsch!

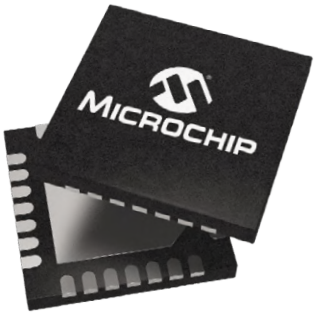
E			A		3	6			0	8		F			D
B	5					D			9					A	1
8	0	6	7									9	C	E	3
9	E	A	1	6	0					5	7	4	B	D	2
2	B	7	F	9	D					3	4	0	E	6	5
C	6	4	0	1	A					D	E	3	F	7	8
5	3	8	D	E	2	4			F	0	6	1	9	C	A
3	F	1	E	7	5	C			B	9	0	A	D	2	4
6	7	0	5	2	8					4	A	C	3	F	9
4	2	C	B	D	9					1	3	6	0	8	E
A	D	9	8	3	E					C	F	5	1	B	7
7	9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

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

Die Verbindung herstellen

Mit der Welt vernetzen – mit und ohne Kabel

Sie sind tagsüber genug gefordert. Microchip ist sich dessen bewusst, weshalb wir die Datenanbindung Ihres Designs einfacher machen. Ob Sie eine robuste und zuverlässige Kabelverbindung oder die Mobilität und den Komfort einer Funkverbindung benötigen – das umfangreiche Angebot von Microchip hilft Ihnen, die Verbindung herzustellen.



Unsere MCUs und MPUs sind so ausgelegt, dass sie zu unseren kabel- und funkbasierten Bausteinen kompatibel sind. Mit unseren zertifizierten Modulen und produktionsfertigen Protokoll-Stacks können Sie Ihre Produkte schnell auf den Markt bringen.

Vernetzen Sie sich mit Microchip und erfahren Sie, wie Sie eine sichere Verbindung mit der Welt um Sie herum herstellen können.

SMART | CONNECTED | SECURE

Verbindung herstellen unter
www.microchip.com/Connected

MICROCHIP