G3078 E (D) € 14,90 | CHF 27,00 | (A, B, L) € 16,00

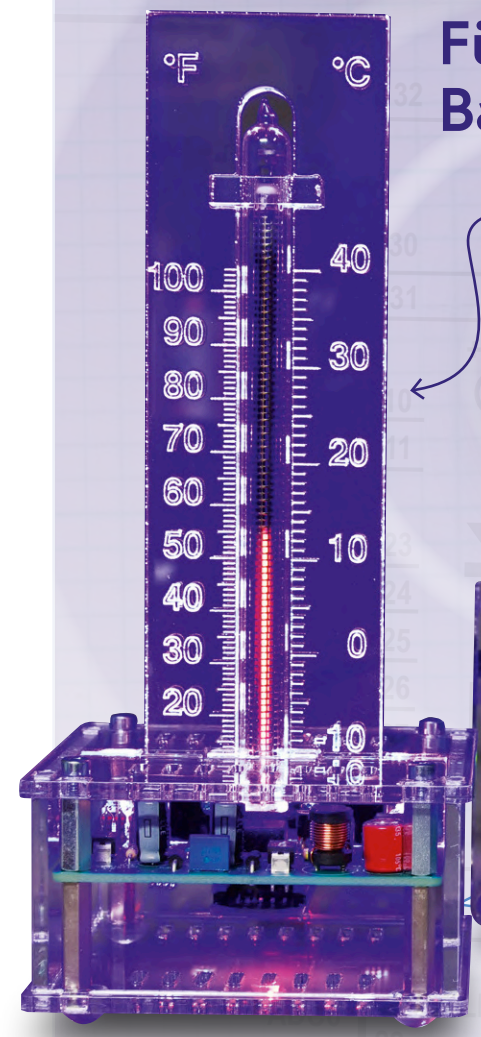
elektor MAG
design › share › sell
575
SEPTEMBER/OKTOBER 2020
ELEKTORMAGAZINE.DE

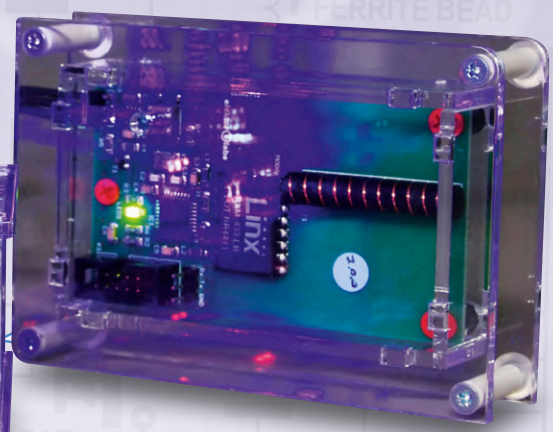
50 Jahre Elektor

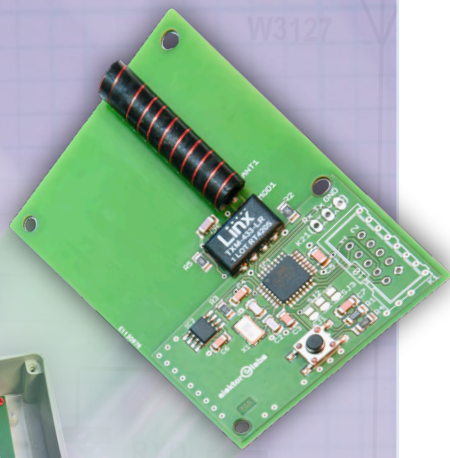
Drahtloser Temperatursensor

Für das Nixie-Bargraph-Thermometer

16

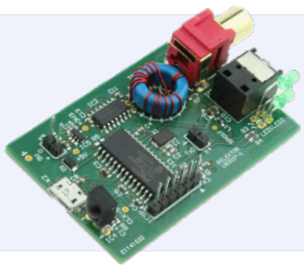






In dieser Ausgabe

und vieles mehr!



4 198630 314904 00010

UNSER SORTIMENT VON TECHNIKERN FÜR TECHNIKER

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

Nur das Beste für Sie – von über 900 Markenherstellern.

Unsere Produktmanager sind seit vielen Jahren bei reichelt tätig und kennen die Anforderungen unserer Kunden. Sie stellen ein breites Spektrum an Qualitätsprodukten zusammen, optimal auf den Bedarf in Forschung & Entwicklung, Instandhaltung, IT-Infrastruktur und Kleinserienproduktion sowie auf Maker zugeschnitten.

Grafikmultimeter mit Bluetooth®

zur Datenübertragung auf Smartphone oder Tablet per Android oder iOS App

PeakTech®

Das beleuchtete TFT-Farbdisplay stellt die verschiedenen Menüs der grafischen Benutzeroberfläche, die Messpunkte der Datenloggererfassung sowie die Messwerte der Multimeterfunktion zuverlässig und hochpräzise dar.

- Spannungsmessung bis 1.000 V AC/DC
- Strommessung bis 10 A AC/DC
- Datenlogger-Funktion mit Kurvendarstellung
- 49.999 Counts
- Lieferumfang: Tasche, Prüfleitungen, Typ-K-Temperaturfühler, 7,4 V Li-Ion Akku, Ladegerät, Bluetooth-Stick und Anleitung

NEU

Bestell-Nr.: PEAKTECH 3440

175,95 (151,68)

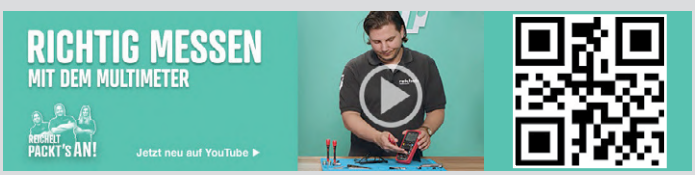
BACK LIGHT | APP | TRMS | EN 61010-1 CAT III 1000 V | EN 61010-1 CAT IV 600 V

Available on the App Store

ANDROID APP ON Google play

Viele weitere Multimeter für jeden Einsatzzweck finden Sie online!

Gleich entdecken ▶ www.reichelt.de/multimeter



- 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

IMPRESSUM

51. Jahrgang, Nr. 575
September/Oktober 2020
ISSN 0932-5468

Erscheinungsweise: 9x jährlich
(6x Elektor-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

Büsra Kas
Tel. 0241 95509178
E-Mail: busra.kas@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)

EDITORIAL

Von Jens Nickel

Chefredakteur Elektor

Die Mischung macht's

Elektor wird nicht nur wegen seiner Mischung aus Theorie und Praxis weltweit von seinen Lesern geschätzt. Wir haben uns auch einen Mix von in die Tiefe gehenden Artikeln und Einsteiger-Inhalten auf die Fahnen geschrieben. Aus vielen Lesermails und Gesprächen weiß ich, dass auch diese Artikel nicht nur von „Anfängern" gelesen werden. Auch Fortgeschrittene nutzen sie, um Wissen aufzufrischen und abzurunden. In diesem Heft finden Sie neben unseren beliebten kleinen Schaltungen und der lehrbuchartigen Serie „Aller Anfang..." auch einen Artikel, der erklärt, wie man als Elektronik-Frischling zu ersten, mit dem Computer gezeichneten Schaltplänen kommt. Die Schritt-für-Schritt-Anleitung für das einfache CAD-Programm EasyEDA stammt aus unserem neuesten Special „Einstieg in die Elektronik mit Arduino" von Florian Schäffer.

Sie haben es sicher schon gemerkt, dass unser Heft ein neues Layout bekommen hat. Auch unsere Schwester-Zeitschrift Elektor Industry, die Elektor-Bücher und die Webseiten haben eine Auffrischung erhalten. Wie finden Sie den neuen Look? Unser neuer Art Director Harmen Heida und ich würden uns freuen, wenn Sie uns per Mail Feedback geben!

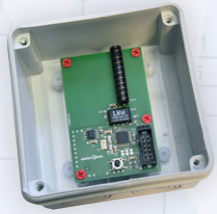
Darüber hinaus habe ich noch eine gute Nachricht für alle Elektronik-Profis, besonders jene, die sich beruflich oder privat für die allerneuesten Entwicklungen interessieren. Sollte nichts ganz Unvorhergesehenes dazwischenkommen, dann kann die Messe electronica am 10. November ihre Tore öffnen. Damit wird auch der Fast Forward Award, bei dem die besten Elektronik-Start-ups des Jahres gegeneinander antreten, wie geplant stattfinden können (www.elektormagazine.com/fastforward). Wir freuen uns schon darauf, auch viele unserer Leser begrüßen zu können!

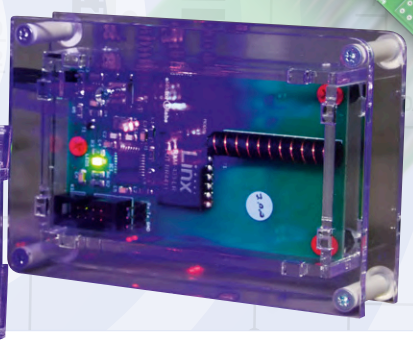
Unser Team

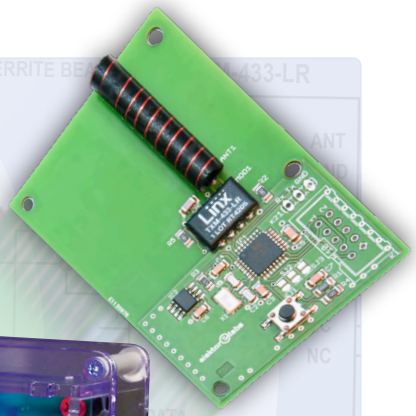
INHALT

Labor-Projekt

Drahtloser Temperatursensor







16

Rubriken

Hintergrund

Projekte

IKEA-Hack
Mit NeoPixel-LEDs und WLAN

56

Review

Elektronische Last

89

Multitasking mit Raspberry Pi

23

Vorschau

Elektor November/Dezember 2020

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

Aus dem Inhalt:
- Präzises LCR-Meter
- LoRa-Tracker
- Stromtastkopf
- Power-Analyzer
- Nixie-Armbanduhr
- Hausautomatisierung mit Home Assistant (2)
- FFT auf dem Maixduino
- Teensy 4.0 - Geschwindigkeit ausreizen
- Grundlagen Batteriemanagement

Und vieles mehr!

Änderungen vorbehalten.
Elektor November/Dezember 2020 erscheint am 5. November 2020.

190365-E-04

Besuchen Sie die neue Labs-Website

USB-S/PDIF-Schnittstelle

Digitaler Audioausgang für Computer, Laptop, Tablet oder Smartphone

Von **Stephan Lück**

Die meisten PCs, Laptops, Tablets und Smartphones verfügen über keinen „echten“ Audioausgang. In der Regel stehen nur ein Multifunktions-USB-Ausgang und ein minderwertiger Kopfhörerausgang zur Verfügung. Dies führt zu Problemen, wenn Sie eines dieser Geräte an einen hochwertigen Verstärker oder einen AV-Receiver anschließen möchten. Um dieses Problem ein für alle Mal zu beseitigen, stellen wir hier eine angemessen gute USB-S/PDIF-Schnittstelle vor.

INFOS ZUM PROJEKT

Tags
digitales Audio, PC, Laptop, Tablet, Smartphone

Niveau
Einsteiger – Fortgeschrittene – Experte

Zeit
circa 4 Stunden

Werkzeuge
Lötwerkzeug (SMD und bedrahtet), mechanisches Werkzeug

Kosten
ca. 35...40 €

EIGENSCHAFTEN

- Betriebsspannung +5 V
- Stromaufnahme 36...43 mA
- Mikrocontroller PIC32MX27F256B-I
- Unterstützt drei S/PDIF-Frequenzen: 44,1 kHz, 48 kHz und 96 kHz
- Unterstützt 16-, 20- und 24-Bit-Audio
- Optischer und elektrischer S/PDIF-Ausgang
- IR-Fernbedienung (nach aktueller RC5-Firmware)
- Unterstützt von Betriebssystemen Windows 7/10, Linux, Android und Raspbian/Raspberry Pi OS

S/PDIF (oder S/P-DIF) steht für „Sony/Philips Digital Interface Format“ und ist eine Schnittstelle zur Übertragung von digitalen Audiodaten über geringe Entfernungen zwischen Modulen oder Systemen wie Heimkino- oder HiFi-Geräte. Es handelt sich um eine unidirektionale Schnittstelle für serielle Daten ohne separates Taktsignal. Der Takt wird aus dem Signal selbst wiedergewonnen. Die zugrunde liegende Idee besteht darin, Audiodaten so lange wie möglich in der digitalen Domäne zu belassen und erst im letzten Moment in ein analoges Signal umzuwandeln.

S/PDIF [1] basiert auf dem professionellen AES3-Standard. Auf Protokollebene sind beide Standards kompatibel, aber die elektrischen und mechanischen Eigenschaften sind unterschiedlich. Eine typische professionelle AES3-Schnittstelle verwendet 3-polige XLR-Verbinder und ein abgeschirmtes symmetrisches Twisted-Pair-Kabel mit einer Impedanz von 110 Ω. Der Signalpegel beträgt 3...10 V_{SS} beziehungsweise 2...7 V_{SS} nach AES/EBU. Seltener werden BNC-Steckverbinder mit 75-Ω-Koaxialkabel verwendet. Beim symmetrischen AES3-Standard mit XLR-Steckverbindern (abgeschirmtes twisted pair) kann eine maximale Entfernung von 1000 m überbrückt werden, bei der Koax-Version sind es nur 100 m.

Für S/PDIF wurden zwei Verbindungsarten definiert. Die einfachste Version (die bei den meisten „besseren“ Audiogeräten verwendet wird) besteht aus einem 75-Ω-Koaxialkabel mit (meist orangefarbenen) Cinch-Steckern. Der Signalpegel beträgt dabei etwa 0,5 V_{SS}.

Die zweite Variante heißt Toslink (von Toshiba Link). Dabei handelt es sich um einen genormten Lichtleiter mit einer roten LED (659 nm) als Sender. Der Lichtleiter ist durchweg ein einfaches POF-Kabel (Plastic Optical Fibre). Damit können Entfernungen von bis zu 10 m überbrückt werden. Mit hochwertigen Glasfasern kann eine Entfernung von bis zu 30 m überbrückt werden, ohne dass ein Repeater dazwischen geschaltet werden müsste. Die optischen Signale sind logisch genauso aufgebaut wie die elektrischen, da das elektrische Signal die Sende-LED nur ein- und ausschaltet.

Das Projekt

Die Schaltung, die wir hier beschreiben, ist ein USB-zu-S/PDIF-Konverter. Der Konverter kann einfach an der USB-Buchse des PCs & Co. angeschlossen werden. Der S/PDIF-Konverter ermöglicht den Anschluss eines AV-Receivers, eines High-End- (Vor-)Verstärkers oder eines Stand-alone-Audio-DACs. Die Schaltung bietet einen elektrischen und einen optischen Ausgang und lässt sich dank eines IR-Empfängers fernsteuern. **Bild 1** zeigt die kompakte, fertig aufgebaute Schaltung.

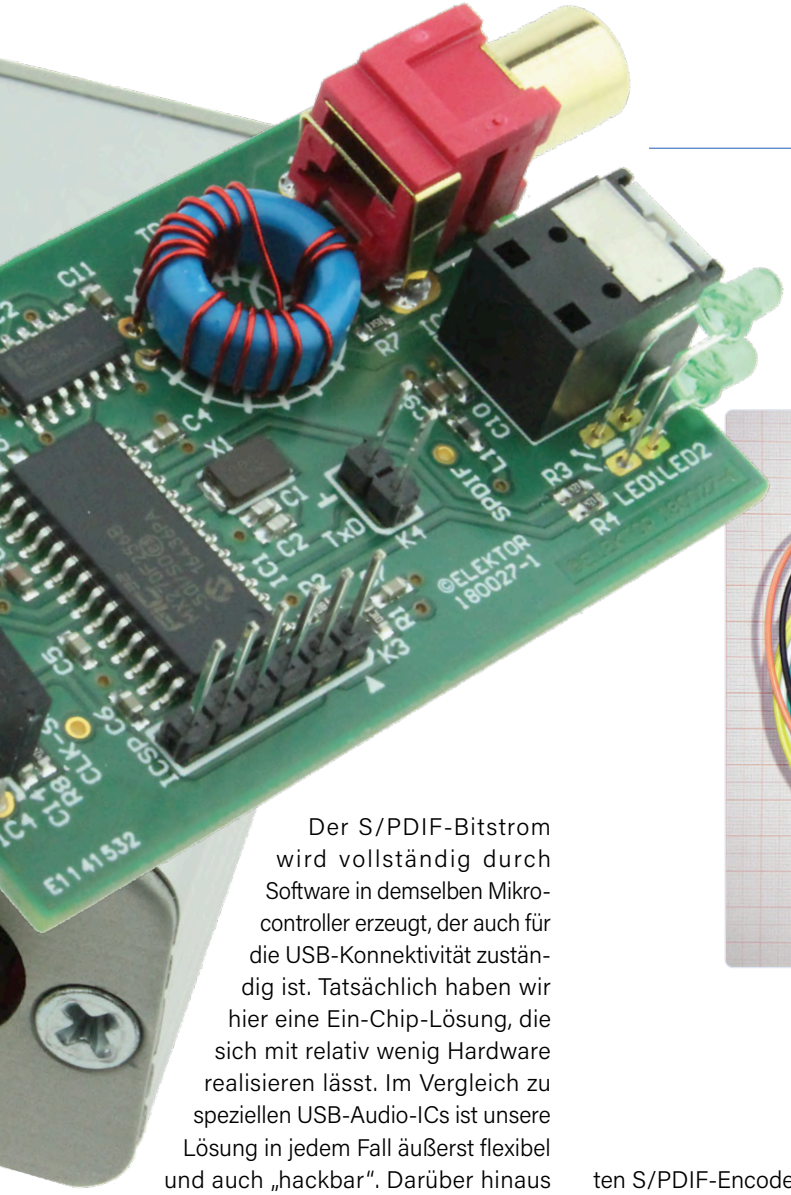


Bild 1. Die USB-S/PDIF-Schnittstelle ist schön kompakt.

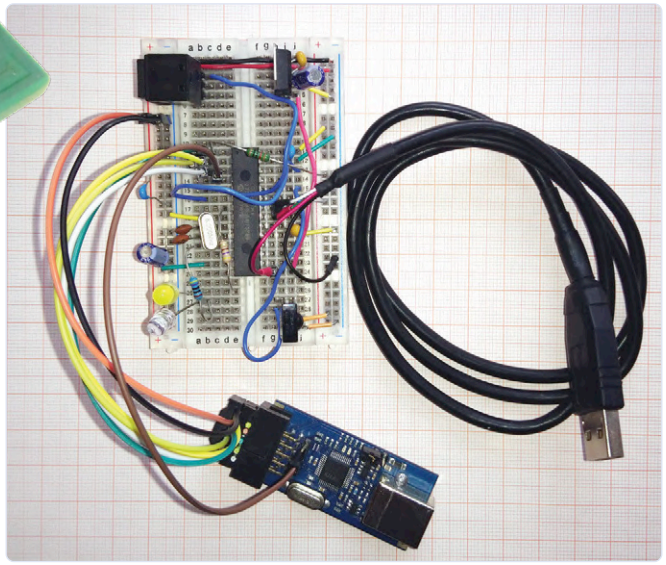


Bild 2. Der Prototyp auf dem Steckbrett.

Der S/PDIF-Bitstrom wird vollständig durch Software in demselben Mikrocontroller erzeugt, der auch für die USB-Konnektivität zuständig ist. Tatsächlich haben wir hier eine Ein-Chip-Lösung, die sich mit relativ wenig Hardware realisieren lässt. Im Vergleich zu speziellen USB-Audio-ICs ist unsere Lösung in jedem Fall äußerst flexibel und auch „hackbar". Darüber hinaus ist die Struktur nicht extrem kritisch - es ist sogar möglich, die Schaltung auf einer Lochrasterplatine aufzubauen und zum Laufen zu bringen (siehe **Bild 2**).

Für dieses Projekt haben wir uns für einen Mikrocontroller vom Typ PIC32MX270 entschieden: Dieser hat die Peripherie für USB-Audioanwendungen schon an Bord und verfügt über genügend RAM, um S/PDIF-Frames zu speichern. Darüber hinaus sind Versionen dieses Controllers mit relativ wenigen Beinchen erhältlich, was das Platinenlayout vereinfacht.

Für das Projekt wurde eine kleine doppelseitige Platine entworfen, auf der nicht nur der Mikrocontroller, sondern auch die Stromversorgung, die optischen und elektrischen S/PDIF-Ausgänge sowie ein Empfänger für die IR-Fernbedienung und zwei LEDs (diese signalisieren die Audio-Abtastfrequenz und einen aktiven Ausgang) untergebracht sind. Die Platine ist so bemessen und bestückt, dass sie in ein kleines Hammond-Gehäuse passt.

Die Audio-Sektion

Der Audioteil der Schaltung umfasst eine USB-Schnittstelle gemäß USB-Audioklasse [2][3], den in die Software implementierten S/PDIF-Encoder und den S/PDIF-Ausgang, für den der SPI-Ausgang des Mikrocontrollers verwendet wird. Ein DMA-Kanal kopiert die S/PDIF-Frames kontinuierlich von einem Ringpuffer auf das SPI. **Bild 3** zeigt die Software- und Hardwarekomponenten in einem Blockdiagramm.

Da die Standard-USB-Audio-Spezifikation schon implementiert ist, müssen auf dem Host keine speziellen Treiber für die USB-S/PDIF-Schnittstelle installiert werden.

Die Spezifikation der Geräteklasse ist recht komplex. Deshalb hat das USB-IF (*USB Implementers Forum*) die *USB Audio Device Class Specification for Basic Audio Devices* [4] veröffentlicht, die eine kleine Zusammenfassung der ursprünglichen *Audio Device Class*-Spezifikation enthält. Die in unserem Projekt implementierte USB-Audioschnittstelle ist der in [4] beschriebenen Kopfhöreranwendung sehr ähnlich. Wir haben jedoch einige zusätzliche Abtastfrequenzen hinzugefügt, der Lautstärkeregler wurde weggelassen und der USB-Deskriptor wurde geändert, um einen S/PDIF-Ausgang anstelle eines Lautsprechers zu beschreiben. Unsere USB-Audio-Schnittstelle hat die folgenden Merkmale:

- Unterstützung für drei Abtastfrequenzen (44,1 kHz, 48 kHz und 96 kHz);
- ein Audioformat: zwei Kanäle mit 3 Byte (24 Bit) pro Kanal (S24_3LE);
- eine Möglichkeit, um das Audiosignal stumm zu schalten.

Diese Merkmale sind im USB-Konfigurationsdeskriptor angegeben (das ist die Datei **usb_descriptors.c** im Softwarepaket). Weitere Informationen sind in [3], [5] und [6] zu finden.

Da dieses Projekt nicht formal USB-konform sein soll, haben wir Hersteller- und Produkt-IDs gewählt, die ohne das Risiko eines Konflikts mit offiziellen USB-Produkten verwendet werden können.

S/PDIF-Kodierer und serielle Ausgabe

Die USB-Schnittstelle verwendet den isochronen USB-Endpunkt 1, um Audiosamples an die S/PDIF-Schnittstelle zu übertragen. Dies bedeutet, dass in einem USB-Frame von 1 ms ein USB-Datenpaket übertragen wird. Jedes Mal, wenn die USB-Hardware ein neues isochrones Paket vom Mikrocontroller empfangen hat, wird eine Interrupt-Routine aufgerufen, die die einzelnen PCM-Frames aus dem USB-Datenpaket in die entsprechen-

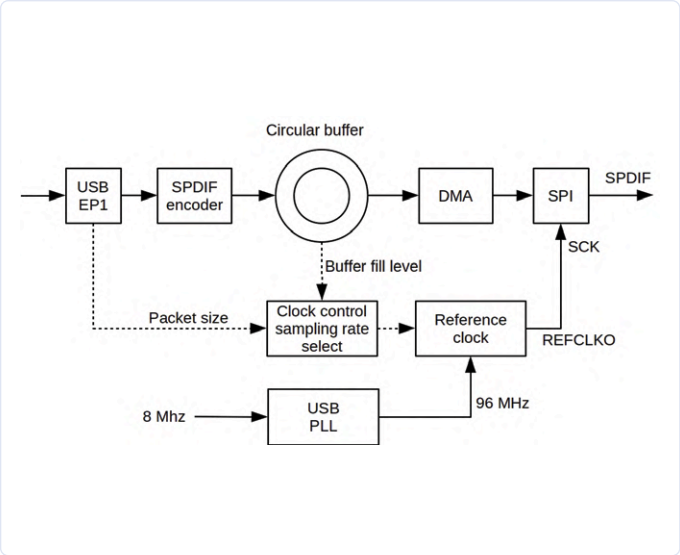


Bild 3. Die Schnittstelle im Blockschaltbild.

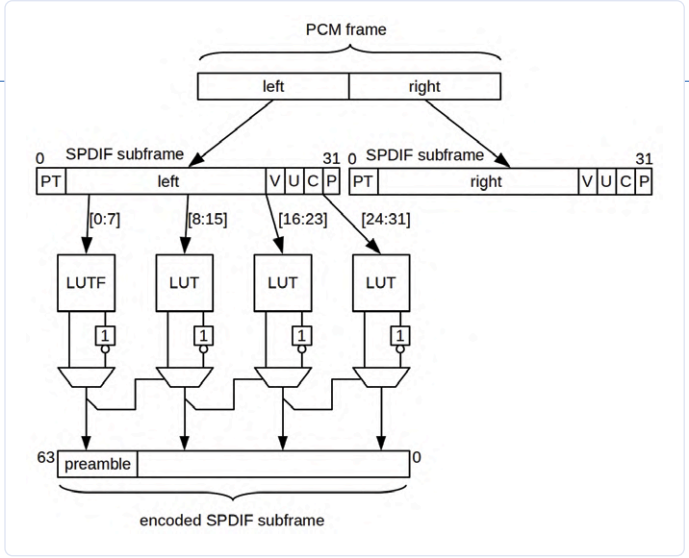


Bild 4. Grundsätzlicher Aufbau des Software-S/PDIF-Kodierers.

den S/PDIF-Frames konvertiert. Weitere Informationen zu S/PDIF finden Sie in [7] und [8]. In **Bild 4** haben wir schematisch dargestellt, wie der Software-S/PDIF-Encoder funktioniert.

Oben in der Abbildung sehen wir einen 48-Bit-PCM-Frame. Jeder Frame enthält zwei Abtastwerte - einen für den linken und einen für den rechten Kanal. Diese Samples werden in vorläufige 32-Bit-S/PDIF-Subframes umgewandelt, indem ein 4-Bit-Präambel-Tag hinzugefügt wird, das den Typ der S/PDIF-Präambel (X, Y oder Z) angibt. Diese Präambel wird bei der endgültigen Kodierung des S/PDIF-Subframes benötigt. Am Ende werden die vier Bits V, U, C und P hinzugefügt. Die Bits V und U entsprechen dem Validity- und dem User Data Bit und sind in unserer Anwendung immer null. Die C-Bits enthalten die Channel-Status-Data und die P-Bits sind Paritätsbits für die einzelnen Subframes. Diese Paritätsbits werden für jeden Teilrahmen mit dem in [9] beschriebenen optimierten Algorithmus berechnet.

Zwei aufeinander folgende S/PDIF-Subframes bilden einen S/PDIF-Frame. 192 aufeinanderfolgende S/PDIF-Frames bilden zusammen einen S/PDIF-Block. Der Kanalstatus wird durch Sequenzierung der C-Bits eines Blocks ermittelt - der Kanalstatusblock hat also eine Länge von 192 Bit (24 Byte). Hier verwenden wir die Kanalstatusbits, um die aktuelle Abtastfrequenz an den S/PDIF-Ausgang angeschlossenen Empfänger weiterzuleiten.

S/PDIF verwendet die Biphase-Mark-Kodierung (auch differentielle Manchester-Kodierung genannt), um die Audiodaten und die Präambelbits zu kodieren. Da zwei Codebits einem Datenbit entsprechen, besteht ein kodierter S/PDIF-Unterrahmen aus 64 Bits. Bei der einfachsten Implementierung der Biphase-Mark-Kodierung müsste jedes Bit eines S/PDIF-Unterrahmens separat kodiert werden, was natürlich wertvolle Prozessorzeit „verschlingt". Aus diesem Grund verwenden wir zwei verschiedene Look-up-Tabellen (LUTs), um jeweils ein Byte zu kodieren. Die LUTF-Look-up-Tabelle wird zur Kodierung des ersten Bytes eines S/PDIF-Unterrahmens verwendet, während die LUT-Look-up-Tabelle zur Kodierung der restlichen drei Bytes des Unterrahmens herangezogen wird. Die LUTF-Tabelle enthält die Präambel-Bitmuster. In Bild 4 ist die LUT-Lookup-Tabelle dreimal eingezeichnet, da sie dreimal zur Kodierung eines S/PDIF-Subframes verwendet wird, in Wirklichkeit gibt es natürlich nur eine LUT-Tabelle im Speicher. Die Look-up-Tabellen werden durch eine Initialisierungsfunktion erzeugt, die einmalig nach einem Reset des Mikrocontrollers ausgeführt wird.

Beim Biphase-Mark-Code muss es in jedem Paar von Codebits, die ein Datenbit repräsentieren, einen Übergang (Flanke) geben. Dazu werden die 16-Bit-Codewörter aus den Lookup-Tabellen invertiert oder nicht, je nach dem letzten Bit des vorhergehenden Codeworts.

Die Tabellen haben noch eine zweite Funktion: die Umkehrung der Bitfolge. Im S/PDIF-Standard werden die Subframes beginnend mit dem LSB gesendet, aber das SPI macht das Gegenteil (MSB zuerst). Daher liefern die LUTs die Codebits in umgekehrter Reihenfolge, so dass sie von der SPI-Hardware korrekt gesendet werden.

Die so gewonnenen 64-Bit-S/PDIF-Codewörter werden in einem Ringpuffer (im SRAM-Speicher des Controllers) abgelegt und per DMA über SPI ausgesendet. Der SPI-Ausgang ist direkt mit den elektrischen und optischen S/PDIF-Ausgängen verbunden.

Der S/PDIF-Bit-Takt

Ein programmierbarer fraktionaler Frequenzteiler, der in den Mikrocontroller integriert ist, leitet den S/PDIF-Bittakt aus einem internen 96-MHz-Taktsignal ab. Die erforderliche S/PDIF-Taktfrequenz ist

$$f_{SPDIF} = 128 * f_S$$

mit f_S als Audio-Abtastfrequenz.

Die USB-Audiospezifikation gibt die vom Host verwendete f_S-Abtastrate nicht explizit an das USB-Gerät weiter. Das bedeutet, dass die Controllersoftware diese Frequenz anhand der Anzahl der PCM-Frames in den isosynchronen USB-Paketen erkennen muss. Der USB-Konfigurationsdeskriptor enthält drei zulässige Abtastfrequenzen 44,1 kHz, 48 kHz oder 96 kHz. Andere Frequenzen dürfen nicht verwendet werden. Ein Vorteil davon ist, dass der Mikrocontroller die vom Host gewählte Abtastfrequenz anhand der Anzahl der PCM-Frames in einem USB-Paket leicht „erraten" kann.

Über den richtigen Takt

In Übereinstimmung mit der Spezifikation für USB-Basis-Audiogeräte [4] wurde für den isochronen Audio-Endpunkt der Synchronisierungstyp „synchron" gewählt. Anders ausgedrückt, hängt die Abtastrate des Audiostroms von der USB-Taktdomäne ab, das heißt, von der Frequenz der vom Host ausgeführten SOF-Token. Im Gegensatz dazu wird der S/PDIF-Bittakt vom lokalen Quarzoszillator abgeleitet. Um einen Überlauf oder eine Entleerung des Ringpuffers zu verhindern, passt die Software regelmäßig den Bruchteil des Frequenzteilfaktors an, basierend auf dem durchschnittlichen Füllstand des Ringpuffers.

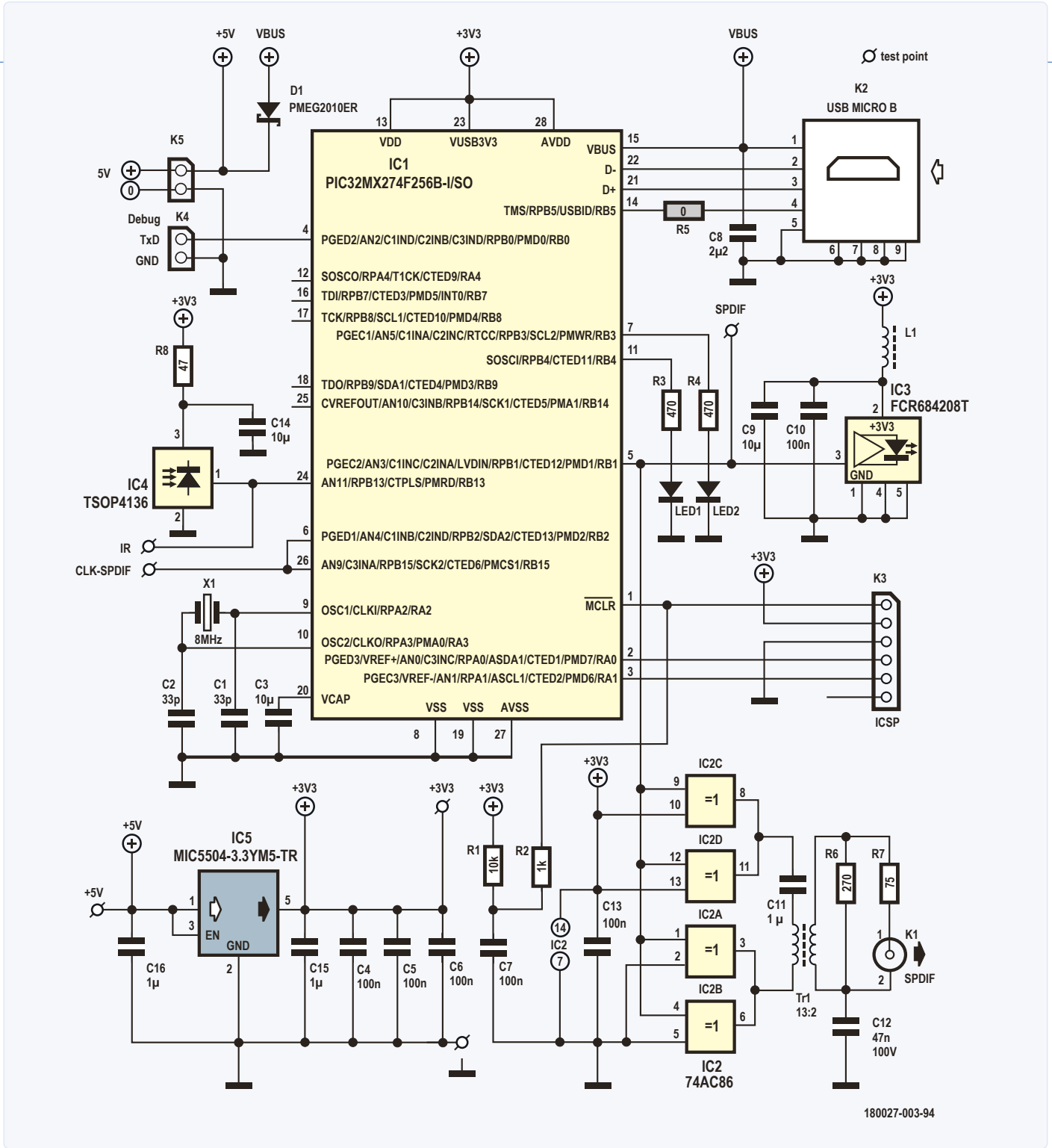


Bild 5. Detailliertes Schaltbild der USB-S/PDIF-Schnittstelle.

Obwohl Microchip in seiner Application Note AN1422 [10] ausdrücklich erwähnt, dass der Referenztakt auch während des Betriebs eingestellt werden könne, sind wir bei der Einstellung des Teilfrequenzfaktors für kleine Werte (wenn eine Abtastfrequenz von 96 kHz gewählt wurde) auf Probleme gestoßen. Jedes Mal, wenn der Teilfaktor angepasst wurde, kam es zu einer kurzen Unterbrechung des Taktsignals, was zu hörbaren Artefakten führte.

Um dieses Problem zu lösen, haben wir eine externe Verbindung zwischen dem Ausgang des Referenztaktoszillators REFCLKO (Pin 6) und dem SPI-Takteingang SCK (Pin 26) hergestellt. Auf diese Weise umgehen wir den Baudratengenerator des SPI.

IR-Fernbedienung

Der IR-Fernbedienungsempfänger ist völlig unabhängig vom Audioteil der Schaltung. Mit anderen Worten: Alles, was der Empfänger empfängt, wird direkt an den Host übertragen, ohne den Audioteil zu beeinflussen.

Zur Dekodierung der IR-Signale vieler verschiedener Fernbedienungen wird die Open-Source-IRMP-Bibliothek [11] verwendet. Die Controllersoftware enthält eine USB-HID-Implementierung [12], mit der die im Dokument *HID Usage Tables* [13] angegebenen Codes verarbeitet werden können.

STÜCKLISTE

Widerstände:
(alle SMD 0603, 1 %, 0,1 W)
R1 = 10 k
R2 = 1 k
R3,R4 = 470 Ω
R5 = 0 Ω (nicht einsetzen)
R6 = 270 Ω
R7 = 75 Ω
R8 = 47 Ω

Kondensatoren:
(alle SMD 0603)
C1,C2 = 33 p, 50 V, 5%, C0G/NP0
C3,C9,C14 = 10 µ, 16 V, 20%, X5R
C4....C7,C10,C13 = 100 n, 50 V, 10%, X7R
C8 = 2,2 µ, 10 V, 10%, X7R
C11,C15,C16 = 1 µ, 50 V, 10%, X5R
C12 = 47 n, 100 V, 10%, X7R

Induktivitäten:
L1 = 600 Ω @ 100 MHz, 0,15 Ω, 1,3 A, SMD 0603 (Murata, BLM18KG601SN1D)
TR1 = Ringkern, 12,5x5mm, Material T38 (Epcos B64290L0044X038)

Halbleiter:
LED1,LED2 = LED, grün, 3 mm, bedrahtet
D1 = PMEG2010ER, SMD SOD-123
IC1 = PIC32MX274F256B-I/SO, SMD SO-28
IC2 = 74AC86, SMD SO-14
IC3 = FCR684208T, Toslink
IC4 = TSOP4136
IC5 = MIC5504-3.3YM5-TR, SMD SOT-23-5

Außerdem:
K1 = Cinch-Buchse für Platinenmontage, bedrahtet (Pro Signal PSG01545)

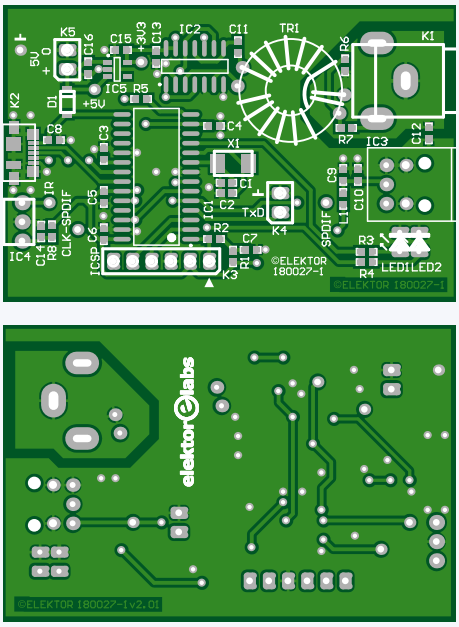


Bild 6. Für die Schaltung wurde eine kompakte Platine entworfen.

K2 = Micro-USB-Buchse Typ B, SMD (Molex 47346-0001)
K3 = 1x6-polige Stiftleiste, vertikal, Raster 2,54 mm, bedrahtet
K4,K5 = 1x2-polige Stiftleiste, vertikal, Raster 2,54 mm, bedrahtet
X1 = Quarz 8 MHz, 5x3,2 mm, SMD (Abracon, ABM3-8.000MHZ-D2Y-T)
TR1 = 0,3 m Kupferlackdraht (CuL), Ø 0,5 mm
Gehäuse Hammond 1455D601, 60 x 42,5 x 23 (mm)

Platine 180027-1 v2.01 (Elektor-Store)

Eine Keymap-Tabelle, die die vom IRMP erkannten Codes mit den Nutzungs-IDs gemäß [13] verknüpft, schafft die Verbindung zwischen IRMP und der HID-Implementierung. Wir haben beschlossen, zusätzlich Codes nach dem RC5-Standard hinzuzufügen, obwohl dieses Format immer weniger genutzt wird, aber immer noch auf vielen Universalfernbedienungen verfügbar ist. Wir glauben, dass dies in vielen Fällen funktionieren wird, ohne andere Geräte wie Fernseher und dergleichen zu stören. In [14] sind einige der RC5-Codes aufgeführt. Darüber hinaus enthält die Tastaturbelegungstabelle Codes für eine Denon-Fernbedienung.
Die Tastaturbelegung befindet sich in der Quelldatei `irhid.c` unter dem Namen `irhid_keymap` und kann leicht angepasst werden, indem man Zeilen hinzufügt oder entfernt und dann die Software neu kompiliert. Um einen Schlüssel hinzuzufügen, müssen Sie die IRMP-Codes für das entsprechende Protokoll, die Adresse und den Befehl finden. Dies kann durch Anschließen eines Terminalemulators (115200 Bit/s) an den S/PDIF-Ausgang und Drücken der entsprechenden Taste erfolgen. Sie sollten dann die Codes auf dem Terminal-Emulator sehen. Wenn das nicht funktioniert, müssen Sie möglicherweise die Unterstützung für den entsprechenden Fernsteuerungscode aktivieren, indem Sie die Datei `irmpconfig.h` ändern. Darüber hinaus benötigen Sie die entsprechende USB-HID-Nutzungs-ID, an die die Taste angeschlossen werden muss. Sie können diese ID in [13] nachschlagen.

Aufbau

Nach all diesen umfassenden Erläuterungen können wir noch einen schnellen Blick auf das detaillierte **Schaltbild 5** werfen. IC1 ist das Herz der Schaltung, der Mikrocontroller. Es kann in-circuit über den ICSP-Verbinder K3 programmiert werden. Wenn Sie keine Lust dazu haben, können Sie auch einen programmierten Controller im Elektor-Shop bestellen. Die Energieversorgung der Schaltung erfordert eine externe +5-V-Spannung an K5.
Das USB-Signal kommt am K2 an und wird direkt zum Controller geführt. Der 0-Ohm-Widerstand R5 muss nicht montiert werden, da USBID in der aktuellen Softwareversion nicht verwendet wird.
IC4 ist ein Standard-IR-Empfänger; IC3 stellt den optischen (Toslink-)Ausgang dar. Über den schnellen Vierfachpuffer IC2 geht das elektrische Ausgangssignal an K1. Damit kommen wir zum „merkwürdigsten" Teil dieser Schaltung, den Ringkerntransformator TR1. Er soll nicht primär für eine galvanische Trennung sorgen, sondern Erdschleifen verhindern. Um das HF-Rauschen zu minimieren, muss der Ausgang entkoppelt werden - daher das Vorhandensein von C12. Die EXOR-Ports von IC2 fungieren als eine Differenzverstärker und sorgen für eine stabilere Primärspannung und eine bessere Kopplung an die Sekundärwicklung des Ringkerntrafos. Sein Wicklungsverhältnis beträgt 13:2, was zu einer Spannung von fast genau 0,5 V_{tt} an einer Last von 75 Ω führt.
Für die Schaltung wurde die (doppelseitige) Platine in **Bild 6** entworfen. Obwohl die meisten Bauteile SMDs sind, stellt der Aufbau für einen auch nur einigermaßen erfahrenen Löter kein allzu großes Problem dar.
Die Platine passt exakt in ein Hammond-Aluminiumgehäuse 1455D601 (60 x 42,5 x 23 mm^3). Sie kann, wie **Bild 7** zeigt, einfach an ihren Platz geschoben werden. Je nach Fertigungstoleranzen kann es notwendig sein, die Kanten der Platine einen Hauch abzufeilen.
Wenn Sie das Hammond-Gehäuses aus der Stückliste verwenden, sollten Sie die beiden schwarzen Kunststoff-Zierringe nicht montieren. Vor allem der Mikro-USB-Stecker muss so tief in das Gehäuse eindringen können, dass er guten Kontakt mit der entsprechenden Buchse erhält. Natürlich müssen geeignete Öffnungen für die Steckverbinder und die LEDs in die Seitenwände gebohrt und gesägt werden. Apropos LEDs: Die Anschlussdrähte werden so gebogen, dass die LEDs übereinander aus dem Gehäuse schauen. **Bild 7** zeigt deutlich, was gemeint ist.
Noch ein Wort zum Transformator: Man kann ihm nicht entkommen! Sie müssen ihn selbst auf den in der Stückliste genannten Ringkern wickeln. Wie in **Bild 8** dargestellt, ist die Primärwicklung (13 Wicklungen) sozusagen

in zwei Hälften auf den Ringkern gewickelt. Dies hat den Vorteil, dass sich die Anschlüsse der Primär- und Sekundärwicklungen gegenüberliegen. Die Sekundärwicklung besteht aus nur zwei Windungen. Zum Wickeln verwenden Sie 0,5 mm starken Kupferlackdraht, etwa 30 cm dürften reichen.

Einige praktische Anmerkungen

Natürlich haben wir die USB-S/PDIF-Schnittstelle in unserem Labor getestet und sie mit verschiedenen Betriebssystemen ausprobiert. Wir hatten mit Windows 7 und Windows 10 keine Probleme und auch unter Linux (Lubuntu und Kubuntu), Android und sogar Raspbian auf einem Raspberry Pi 3 Modell B+ (mit dem 2019-09-26-raspbian-buster-full-Image) hatten wir kaum Probleme.

Windows

Windows erkennt die Schnittstelle automatisch. Die Abtastrate kann in der Systemsteuerung unter der Option Sound eingestellt werden. Wählen Sie „SPDIF Interface Properties"; auf dem Advanced-Tab können Sie das Format wählen, wie in **Bild 9** dargestellt.

Lubuntu

Unter Linux (wir haben Lubuntu verwendet) hängen die Abtastfrequenzen von der Datei ab. Beim einfachen GNOME MPlayer erscheint eine 44,1-kHz-Datei mit 44,1 kHz am S/PDIF-Ausgang. Aber wir mussten jedes Mal *USB_SPDIF Stereo (IEC958) (PulseAudio)* als Audio-Ausgang wählen - auch wenn wir das schon einmal gemacht hatten. Um das zu umgehen, können Sie alle anderen Audiogeräte ausschalten (hier bei: *Audio & Video – PulseAudio Volume Control - Configuration*). Eine 48-kHz-Datei und eine 96-kHz-Datei werden mit 48 kHz wiedergegeben. Eine andere Möglichkeit ist die direkte Verwendung von ALSA ohne PulseAudio in einem Terminal. Wenn nichts zu hören ist, starten Sie zuerst den Alsamixer in einem Terminal (Strg+Alt+T und `alsamixer`). Drücken Sie <F6>, um die Soundkarte *USB_SPDIF* auszuwählen (es erscheint nur ein kleiner Regler, der 00 und PCM anzeigt, da die Schnittstelle keinen Lautstärkeregler hat). Verlassen Sie dann den Mixer und geben Sie

```
aplay -D plughw:CARD=USBSPDIF
   //.../
```

ein, wobei `//.../` für den Pfad zur Audiodatei steht. Damit können Sie aber nur unkomprimierte wav-Dateien abspielen. Um MP3-Dateien von der Kommandozeile aus abzuspielen, können Sie zum Beispiel mpg123 verwenden, aber es gibt noch viele andere Möglichkeiten. Obwohl es USB-Audio-DACs schon seit vielen Jahren gibt, muss die standardmäßige Art und Weise, wie Linux mit Audiodateien umgeht, noch verbessert werden. Die Einstellung der Ausgangsabtastfrequenz ist auch durch Modifizierung der Konfigurationsdatei von PulseAudio (`/etc/pulse/daemon.conf`) möglich. Informationen dazu finden Sie online. Machen Sie jedoch vorher eine Sicherheitskopie der Originaldatei - ein Unfall ist schnell passiert...

Android

Verbinden Sie die Schnittstelle mit einem OTG-Kabel. Die Installation einer Anwendung namens „USB Audio Player Pro" ist wahrscheinlich der beste Weg, unsere Schnittstelle zu nutzen. Es umgeht die Audio-Einschränkungen von Android. Alle drei Abtastfrequenzen der Schnittstelle werden unterstützt. Nachdem die Anwendung gestartet wurde, wird die Schnittstelle sofort erkannt.

Raspbian

Schließlich haben wir die Schnittstelle in Kombination mit Raspbian ausprobiert. Wie Lubuntu ist Raspbian ein freies Betriebssystem, das ebenfalls auf Debian basiert, daher sollte es nicht überraschen, dass der Test dem gleichen Muster folgte. Auf diese Weise spielt das System unkomprimierte wav-Dateien ab:

```
aplay -D plughw:CARD=USBSPDIF
   //.../
```

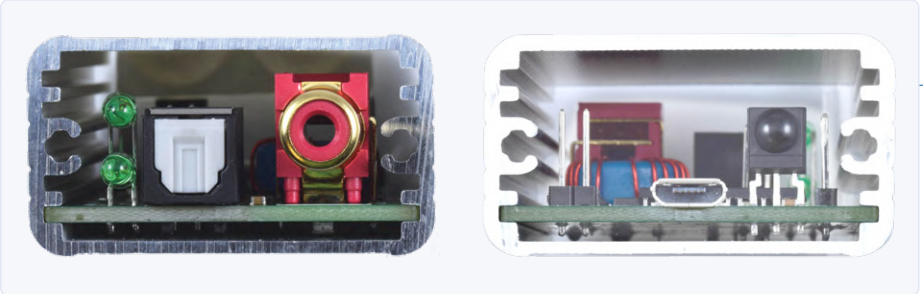


Bild 7. Die Platine kann in das Aluminiumgehäuse eingeschoben werden.

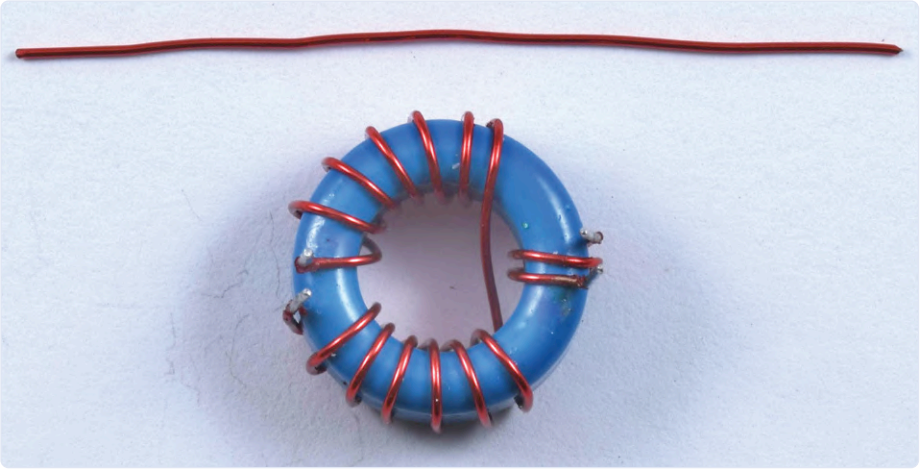


Bild 8. So wird der Ringkerntransformator gewickelt. Die Primärwicklung ist in zwei Teile aufgeteilt.

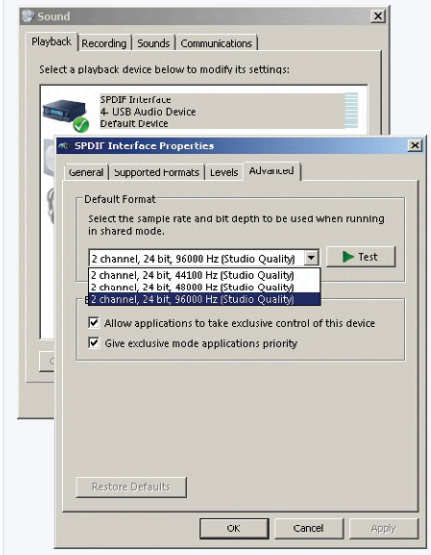


Bild 9. Einstellungen unter Windows 7.

IM ELEKTOR-STORE

- **USB-S/PDIF-Interface, unbestückte Platine:** www.elektor.de/180027-1
- **USB-S/PDIF-Interface, programmierter Controller:** www.elektor.de/180027-41

Um die Wiedergabe zu stoppen, geben Sie Strg+c ein.

Beim Abspielen von MP3-Dateien von der Kommandozeile aus funktionierte mpg123 (falls nicht installiert: `sudo apt-get install mpg123`) erst mit unserer frischen Raspbian-Installation, nachdem PulseAudio und seine Lautstärkeregelung installiert waren. So installieren Sie PulseAudio:

```
sudo apt-get install pulseaudio
```

Installation der passenden Lautstärkeregelung:

```
sudo apt-get install pavucontrol
   paprefs
```

Starten Sie das System nach der Installation neu.
Wie bei Lubuntu müssen Sie Onboard-Audio in Sound & Video unter *PulseAudio Volume Control - Configuration* ausschalten.
Der vorinstallierte VLC Media Player produzierte erst einen Ton, nachdem auch PulseAudio installiert war. Eine mit 32 kHz abgetastete Datei wird mit 96 kHz wiedergegeben.
Natürlich haben wir auch die Audioleistung gemessen. Für eine Beschreibung dieser Messungen und der Ergebnisse verweisen wir auf die Elektor-Labs-Webseite dieses Projekts [15].
Die aktuelle Software kann von der Projektseite dieses Artikels [16] oder von GitHub [17] heruntergeladen werden.
Es bleibt nur noch, Ihnen viel Bau- und Hörvergnügen zu wünschen! Wie immer sind Ihre Kommentare willkommen.

180027-02

WEBLINKS

[1] **Über Toslink**: https://kompendium.infotip.de/spdif_toslink.html

[2] **Universal Serial Bus Specification Revision 2.0**: www.usb.org/document-library/usb-20-specification

[3] **Universal Serial Bus Device Class Definition for Audio Devices, Release 1.0**: www.usb.org/document-library/audio-device-document-10

[4] **Universal Serial Bus Audio Device Class Specification for Basic Audio Devices, Release 1.0**: www.usb.org/document-library/audio-device-class-spec-basic-audio-devices-v10-and-adopters-agreement

[5] **Universal Serial Bus Device Class Definition for Audio Data Formats, Release 1.0**: www.usb.org/document-library/audio-data-formats-10

[6] **Universal Serial Bus Device Class Definition for Terminal Types, Release 1.0**: www.usb.org/document-library/audio-terminal-types-10

[7] **AES3-2003**: AES standard for digital audio – Digital input-output interfacing – Serial transmission format for two-channel linearly represented digital audio data

[8] **Crystal Application None AN22: „Overview Of Digital Audio Interface Data Structures"**: https://statics.cirrus.com/pubs/appNote/an22.pdf

[9] **Sean Eron Anderson: „Bit Twiddling Hacks", section „Compute parity in parallel"**: http://graphics.stanford.edu/~seander/bithacks.html#ParityParallel

[10] **Microchip AN1422: „High-Quality Audio Applications Using the PIC32"**: http://ww1.microchip.com/downloads/en/AppNotes/01422A.pdf

[11] **Infrarot-Multiprotokolldekoder (IRMP)**: www.mikrocontroller.net/articles/IRMP

[12] **USB Device Class Definition for Human Interface Devices (HID), Version 1.11**: www.usb.org/document-library/device-class-definition-hid-111

[13] **USB HID Usage Tables, Version 1.12**: https://www.usb.org/document-library/hid-usage-tables-112

[14] **IR-Fernbedienungsprotokolle (Elektor März 2001)**: https://www.elektormagazine.de/magazine/elektor-200103/1118

[15] **Projektseite auf Elektor Labs**: https://www.elektormagazine.de/labs/usb-spdif-interface-180027

[16] **Projektseite zu diesem Artikel**: https://www.elektormagazine.de/180027-02

[17] **Software-Download auf GitHub**: https://github.com/kiffie/usb-spdif

Praktisches ESP32-Multitasking (4)

Binäre Semaphore

Von **Warren Gay** (Kanada)

Beim Multitasking in FreeRTOS besteht oft die Notwendigkeit, zwischen Tasks zu synchronisieren. Eine solche Synchronisationsmöglichkeit sind binäre Semaphore.

Ein Semaphor ist eine Funktion, die es dem Aufrufer erlaubt, die weitere Ausführung zu blockieren, bis die Erlaubnis zum Fortfahren erteilt (give) wird. Mit anderen Worten, der Aufrufer versucht, das Semaphor einzunehmen (take), aber wenn das Semaphor bereits eingenommen ist, wartet die aufrufende Aufgabe (block). Das ist ähnlich wie bei Jungen, der beim Tanz ein beliebtes Mädchen auffordern möchten. Wenn das Mädchen gerade tanzt (taken), dann müssen die anderen interessierten Jungen warten, bis sie wieder bereit ist.
Dies führt uns dazu, eine der Eigenschaften eines Semaphors zu definieren: Ein *binäres* Semaphor kennt zwei Zustände, *taken* and *given*.
Ein Semaphor selbst kann keinen Zugriff auf Ressourcen steuern. Nur nach Vereinbarung (agreement) kann der Ressourcenschutz über ein Semaphor implementiert werden. Wenn die Jungen beim Tanz das Konzept des „Nehmens" und „Gebens" nicht respektierten, dann könnten mehrere Jungen das Mädchen gleichzeitig an den Armen packen. Das Semaphor funktioniert nach einem Protokoll: Der Zugreifende erklärt sich bereit, irgendeine Ressource nur dann zu nutzen, wenn es ihm gelingt, das Semaphor zu nehmen. Er wird die Ressource auch erst dann wieder verwenden, wenn er das nächste verfügbare Semaphor genommen hat.

Zeitüberschreitungen beim Nehmen

Ein Junge, der beim Tanz warten muss, hat vielleicht nur begrenzt Geduld. Er kann beschließen, dass er sich nach zehn Minuten Wartezeit für ein anderes Mädchen entscheidet, mit dem er tanzen möchte. Binäre Semaphoren erlauben es dem Aufrufer auch, eine solche Timeout-Periode (in System-Ticks) anzugeben. Wenn ein Versuch, ein Semaphor zu nehmen, länger als die angegebene Anzahl von Ticks dauert, kehrt der Aufruf mit einem Fehlercode zurück. Alternativ kann FreeRTOS angewiesen werden, unbegrenzt zu warten, bis das Semaphor gegeben wird (ähnlich wie bei einem Jungen, der völlig verliebt ist). Schließlich gibt es noch die Option, überhaupt kein Timeout anzugeben, wobei der Versuch sofort fehlschlägt, wenn das Semaphor nicht genommen werden kann.
Zusammenfassend lässt sich sagen, dass Semaphore auf drei Arten mit der Zeit arbeiten können:

- Für immer blockieren, bis das Semaphor vom Aufrufer genommen (taken) werden kann.
- Für eine definierte Zeitspanne blockieren, wobei der Versuch fehlschlägt, wenn die Operation mit der Zeit ausläuft.
- Scheitert sofort, wenn das Semaphor gerade genommen (taken) ist.

Besitzen und Freigeben

Wenn ein Semaphor genommen wird, sagt man, dass der Task es „besitzt" (own). Der Junge, der gerade mit dem Mädchen tanzt, besitzt sie dabei tatsächlich (wenn auch vielleicht nicht ihr Herz). Wenn er mit ihr fertig getanzt hat, kann er sie sofort jemand anderem übergeben oder einfach ihre Hände loslassen und sie so freigeben: Es ist keine Wartezeit erforderlich. Ebenso ist beim Geben eines Semaphors kein Timeout-Argument erforderlich; das Freigeben eines besessenen Semaphors geschieht unmittelbar.
Der Akt des „Freigebens" eines Semaphors bedeutet nicht unbedingt, dass es sofort von einem anderen Task „genommen" wird. Sobald der Junge den Tanz mit dem Mädchen beendet hat, wird sie verfügbar, aber das bedeutet nicht, dass es zwangsläufig andere interessierte Tanzpartner gibt. Die anderen Jungen haben vielleicht aufgegeben und andere Tanzpartner gefunden.

Ausgangszustand

Lassen Sie uns diese Analogie ein wenig weiter spannen - das Mädchen wird im „genommenen" Zustand erst von ihren Eltern erschaffen. Erst wenn ihre Eltern ihr erlauben, dem Tanz beizuwohnen, ist sie „gegeben" und wird verfügbar. Auf die gleiche Weise wird das binäre FreeRTOS-Semaphor zunächst im „genommenen" Zustand erzeugt. Dies unterscheidet sich von Mutexen in Linux oder Windows, da sie in einem „gegebenen" Zustand erstellt werden. Das FreeRTOS-Mutex wird in einem späteren Abschnitt behandelt.

xSemaphoreBinaryCreate():

Das Erzeugen eines binären Semaphors in FreeRTOS ist einfach:

```
SemaphoreHandle_t hMySemaphore;

hMySemaphore = xSemaphoreCreateBinary();
assert(hMySemaphore)
```

Es sind keine Argumente einzutragen und es wird ein Handle auf das erzeugte binäre Semaphor zurückgegeben. Es ist möglich, dass das Handle als `nullptr (NULL)` zurückgegeben wird, wenn der verfügbare Speicher erschöpft ist. Es ist empfehlenswert, den zurückgegebenen Wert zu überprüfen (hier das `assert()`-Makro aus *assert.h*), um sicherzustellen, dass das Semaphor erstellt wurde.

xSemaphoreGive()

Ein Semaphor zu freizugeben, ist auch einfach:

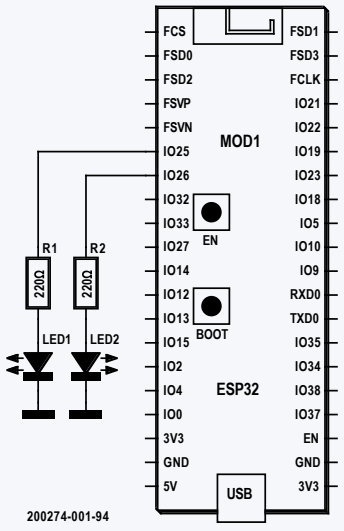


Bild 1. Demo-Schaltung für **semaphores.ino**.

```
SemaphoreHandle_t hMySemaphore;
BaseType_t rc; // return code
...
rc = xSemaphoreGive(hMySemaphore);
assert(rc == pdPASS);
```

Die „give"-Operation kann nur aus folgenden Gründen fehlschlagen und `pdFAIL` zurückgeben:
Das Handle (`hMySemaphore`) ist ungültig.
Das Semaphor wurde bereits „gegeben".

xSemaphoreTake()

Das „Nehmen" eines Semaphors ist potenziell eine blockierende Operation für den aufrufenden Task:

```
SemaphoreHandle_t hMySemaphore;
TickType_t wait = 30; // ticks
BaseType_t rc; // return code
...
rc = xSemaphoreTake(hMySemaphore, wait);
// Returns pdPASS when taken,
// otherwise returns pdFAIL if it times out
```

LISTING 1. DAS DEMOPROGRAMM SEMAPHORES.INO [1].

```
0001: // semaphores.ino
0002: // Practical ESP32 Multitasking
0003: // Binary Semaphores
0004:
0005: #define LED1_GPIO   25
0006: #define LED2_GPIO   26
0007:
0008: static SemaphoreHandle_t hsem;
0009:
0010: void led_task(void *argp) {
0011:   int led = (int)argp;
0012:   BaseType_t rc;
0013:
0014:   pinMode(led,OUTPUT);
0015:   digitalWrite(led,0);
0016:
0017:   for (;;) {
0018:     // First gain control of hsem
0019:     rc = xSemaphoreTake(hsem,portMAX_DELAY);
0020:     assert(rc == pdPASS);
0021:
0022:     for ( int x=0; x<6; ++x ) {
0023:       digitalWrite(led,digitalRead(led)^1);
0024:       delay(500);
0025:     }
0026:
0027:     rc = xSemaphoreGive(hsem);
0028:     assert(rc == pdPASS);
0029:   }
0030: }
0031:
0032: void setup() {
0033:   int app_cpu = xPortGetCoreID();
0034:   BaseType_t rc;  // Return code
0035:
0036:   hsem = xSemaphoreCreateBinary();
0037:   assert(hsem);
0038:
0039:   rc = xTaskCreatePinnedToCore(
0040:     led_task,   // Function
0041:     "led1task", // Task name
0042:     3000,       // Stack size
0043:     (void*)LED1_GPIO, // arg
0044:     1,          // Priority
0045:     nullptr,    // No handle returned
0046:     app_cpu);   // CPU
0047:   assert(rc == pdPASS);
0048:
0049:   // Allow led1task to start first
0050:   rc = xSemaphoreGive(hsem);
0051:   assert(rc == pdPASS);
0052:
0053:   rc = xTaskCreatePinnedToCore(
0054:     led_task,   // Function
0055:     "led2task", // Task name
0056:     3000,       // Stack size
0057:     (void*)LED2_GPIO, // argument
0058:     1,          // Priority
0059:     nullptr,    // No handle returned
0060:     app_cpu);   // CPU
0061:   assert(rc == pdPASS);
0062: }
0063:
0064: // Not used
0065: void loop() {
0066:   vTaskDelete(nullptr);
0067: }
```

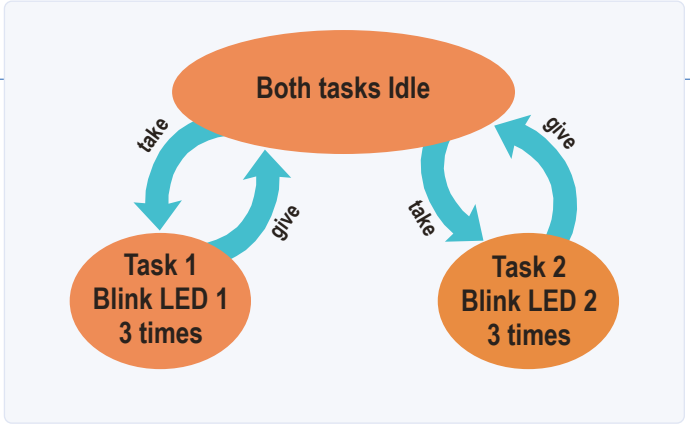


*Bild 2. Die Zustände der ausführenden Aufgaben im Programm **semaphores.ino**.*

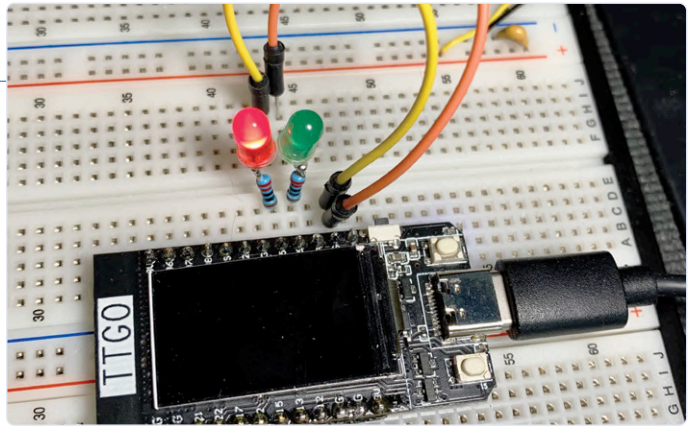


*Bild 3. Das Programm **semaphores.ino** treibt zwei LEDs am ESP32-TTGO-Board.*

Die „Take"-Operation kann fehlschlagen, wenn das Handle ungültig ist oder der Aufruf eine Zeitüberschreitung nach sich zieht (Zeitraum vom `wait`-Argument definiert). Andernfalls blockiert der Aufruf die Ausführung des aufrufenden Tasks, bis er das Semaphor „übernommen" hat. Der `wait`-Parameter kann je nach den Anforderungen des Callers einen von drei verschiedenen Werten haben:

- Makrowert `portMAX_DELAY` - ewiges Warten, bis das Semaphor „übernommen" wurde.
- Irgendein positiver Wert größer null - warten auf entsprechend viele Ticks.
- Null - schlägt sofort fehl, wenn das Semaphor nicht sofort „genommen" werden kann.

Demo

Im Demonstrationsprogramm lassen zwei Tasks jeweils eine LED blinken (siehe **Listing 1**). Durch die gemeinsame Nutzung eines binären Semaphors kann aber jeweils nur einer der beiden Tasks seine LED blinken lassen. Der nicht laufende Task muss warten, bis das Semaphor von dem anderen laufenden Task „gegeben" wurde, so dass das Semaphor auch „genommen" werden kann. **Bild 1** zeigt den Schaltplan für das Beispielprogramm für jedes beliebige ESP32-Modul, das Sie verwenden möchten.

Bild 2 stellt die Zustände des Task-Paares dar, die im Demoprogramm ausgeführt werden. Wenn ein Task das Semaphor übernimmt, kann dieser Task seine LED ansteuern und blinken lassen, während der andere Task blockiert ist und wartet. Nur wenn der besitzende Task das Semaphor zurückgibt, kann der andere Task es übernehmen (und dann weitere Dinge ausführen). **Bild 3** schließlich zeigt ein Beispiel, wie die Schaltung auf einem Entwicklungsboard aufgebaut werden kann.

In der `setup()`-Routine erzeugt Zeile 36 das Semaphor und weist das Handle der statischen globalen Variablen `hsem` zu. Die Zeilen 39...46 erzeugen den ersten Task, der die LED1 blinken lässt. Beachten Sie das Argument in Zeile 43 der Task-Erzeugung. Der zu verwendende GPIO-Pin wird als `void`-Pointer übergeben. Dieser wird dann in der Task-Funktion `led_task()` (Zeile 11) auf einen `int` GPIO-Wert zurückgeführt. Der Pointer wird quasi „missbraucht", um einen Wert zu übergeben, aber das funktioniert, so lange der Wert die gleiche Anzahl von Bits aufweist wie eine Zeigeradresse.

Nachdem der erste Task erstellt wurde, „übergeben" wir in Zeile 50 das Semaphor. Weil wir dies *vor* der Erstellung des zweiten Tasks tun, ist es wahrscheinlich, dass der erste Task das Semaphor auch zuerst erhält. Die Zeilen 53...60 erzeugen dann den zweiten Task. Beide Tasks führen die gleiche Funktion `led_task()` aus, jedoch mit unterschiedlichen Argumentwerten entsprechend den verwendenden LED-GPIO-Pins.

Die Zeilen 14...15 konfigurieren die in der Task verwendete LED. Der Task tritt dann in eine Endlosschleife ein, die bei Zeile 17 beginnt. Der erste Schritt, der innerhalb dieser Schleife ausgeführt wird, ist die „Übernahme" des Semaphors (Zeile 19). Dies gelingt jeweils nur einem Task zur gleichen Zeit.

Der Task, dem es gelingt, das Semaphor zu „nehmen", setzt seine Tätigkeit in den Zeilen 22...25 fort. In dieser Schleife blinkt die LED des Tasks dreimal. Danach in Zeile 27 wird das Semaphor „freigegeben", so dass der andere Task das Semaphor „übernehmen" kann. Auf diese Weise übergibt jeder Task abwechselnd die Kontrolle an den anderen Task.

Zusammenfassung

Das in dieser Demonstration verwendete binäre Semaphor wurde auf zwei verschiedene Arten verwendet. Vor der ersten „Give"-Operation in der `setup()`-Funktion kann keine der beiden Tasks fortfahren. Dies führte dazu, dass sich das Semaphor wie eine Barriere verhielt. Nach dieser ersten „Geben"-Operation „übernimmt" einer der beiden Tasks das Semaphor, lässt die LED blinken und „gibt" dann das Semaphor zurück. In dieser Beziehung scheint sich das Semaphor wie ein Mutex zu verhalten. Beide Tasks versuchen ständig, ihren Code auszuführen, aber durch diesen gegenseitigen Ausschluss des Semaphors ist es immer nur einem Task erlaubt, seine LED blinken zu lassen.

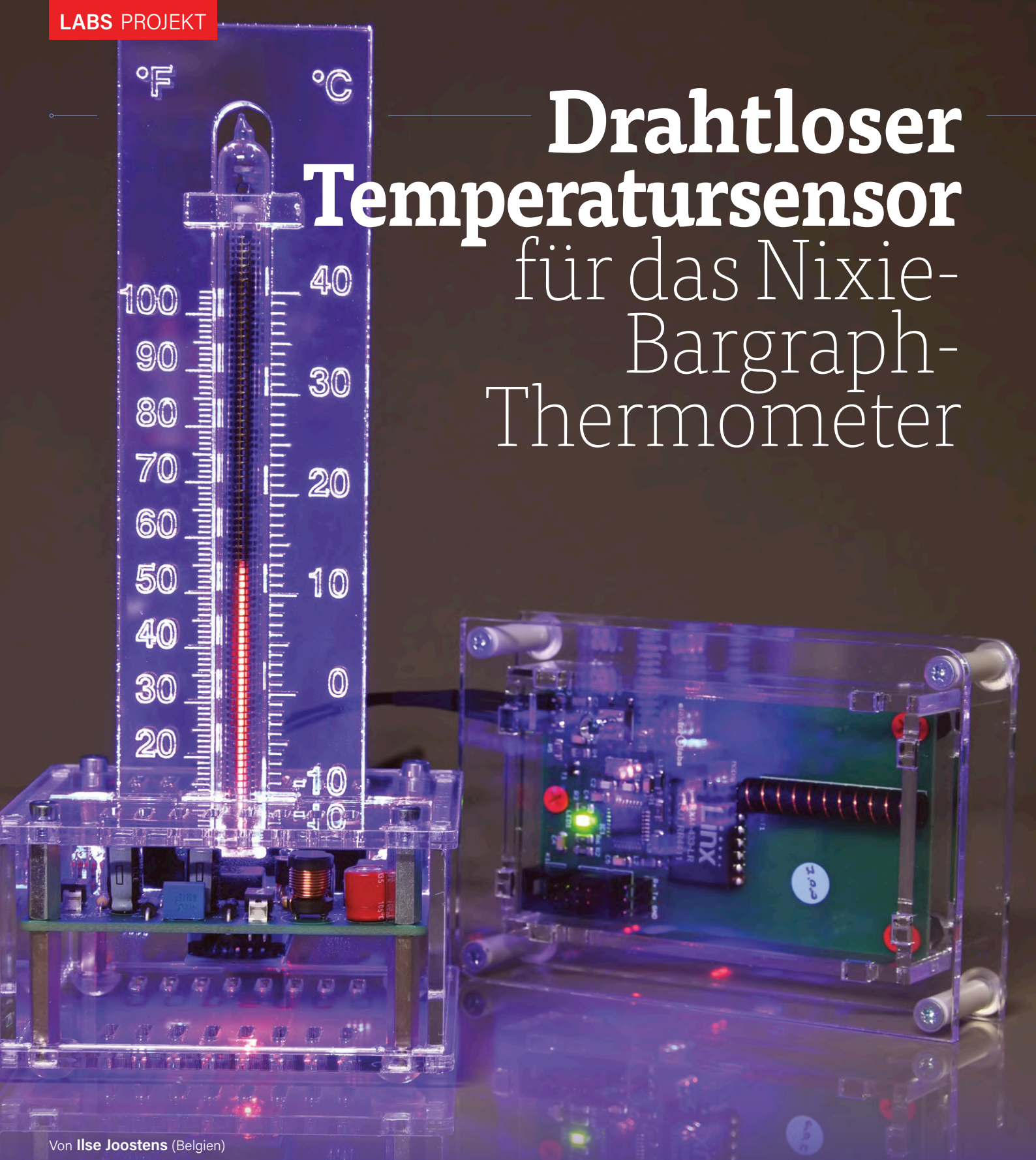
Es ist wichtig, sich zu merken, dass die binären Semaphore im „genommenen" Zustand erzeugt werden (im Gegensatz zu einem Mutex). Wäre das Semaphor in dieser Demonstration nicht ursprünglich in der `setup()`-Routine (Zeile 50) „gegeben" worden, würden beide Tasks für alle Ewigkeit steckenbleiben. In FreeRTOS sind auch andere Synchronisierungsprimitive verfügbar (einschließlich Mutex), aber das simple binäre Semaphor ist manchmal alles, was wirklich benötigt wird.

200274-02

WEBLINK

[1] **Demo-Programm**: https://github.com/ve3wwg/esp32_freertos/blob/master/semaphores/semaphores.ino

Drahtloser Temperatursensor für das Nixie-Bargraph-Thermometer



Von **Ilse Joostens** (Belgien)

Nachdem er den Elektor-Artikel über das Nixie-Bargraph-Thermometer verdaut hatte, wies ein Leser aus dem Mittleren Westen der USA freundlichst darauf hin, dass die Temperaturen in seiner Heimat seit mehr als sechs Monaten leicht über 30 °C lagen. Unter diesen Umständen war die ursprüngliche Skalierung unseres Thermometers von 10...30 °C natürlich ziemlich nutzlos. Als wir die Skala auf -10...+40 °C erweiterten, kam uns der Gedanke, dass wir das Thermometer in ein drahtloses Gerät verwandeln könnten...

INFOS ZUM PROJEKT

Tags
433 MHz, Linx, LPR, 1-Wire, Manchester, Nixie, Sensorik

Niveau
Einsteiger – Fortgeschrittene – Experte

Zeit
etwa 1,5 Stunden

Tools
Laborwerkzeuge, Lötkolben

Kosten
etwa 75 €

EIGENSCHAFTEN

- Versorgung Empfänger: 3,3...5 V
- Versorgung Sender: 3 V (CR2450-Batterie)
- Lizenzfreie Funkverbindung 433,92 MHz
- Sendeintervall circa 2 Minuten, einstellbar
- Vorgefertigte Halbwellen-Wendelantenne
- Temperaturfühler DS2438Z+
- Nutzbarer Temperaturbereich: -20 ...+70 °C (Fehler ±2 °C)
- TX- und OOK/ASK-RX-Modul von Linx Technologies
- RX emuliert 1-Wire-Temperatursensor DS18B20, jedoch ab 2,4 V
- Vorbestückte Sender- und Empfängerplatine

Ursprünglich dachten wir, dass es schwieriger sein würde, die Skala des Nixie-Bargraph-Thermometers [1] zu erweitern, da die Bargraph-Röhre IN-9 ziemlich ungenau ist. In der Praxis erweist sie sich jedoch als genau genug, um jede Außentemperatur in einer Region mit gemäßigtem Klima anzuzeigen. Daher könnte ein wetterfester Außensensor mit Funkverbindung sehr nützlich sein.

Drahtlose Module für Sender und Empfänger

Heutzutage sind viele preisgünstige Kombinationen von drahtlosen Empfängermodulen (RX) und Sendermodulen (TX) erhältlich. Der Einfachheit halber und um eine vernünftige Reichweite zu erzielen, haben wir uns für 433-MHz-Module entschieden, die bauartgenehmigt und lizenzfrei sind. Um den Preis des Endprodukts so niedrig wie möglich zu halten, testeten wir zuerst die berüchtigten chinesischen Module FS1000A und MX-RM-5V. Wir kauften drei Paare von Sender- und Empfängermodulen bei eBay und fertigten dann Prototypen von Platinen für einen Testaufbau an. Letztendlich funktionierten die Module, aber die Reichweite war recht enttäuschend. Die Hauptprobleme waren:

- Der Sender benötigt eine Betriebsspannung von 9...12 V, um genügend Leistung an die Antenne zu liefern;
- Das Empfängermodul war extrem empfindlich gegenüber Störungen auf der Versorgungsschiene.

Wir untersuchten dann andere (chinesische) Funkmodule und stellten fest, dass die meisten von ihnen mit einfacher ASK- oder OOK-Modulation (amplitude shift keying, on-off keying) von undurchsichtigen Herstellern stammten, ohne jegliche Garantie bezüglich der zukünftigen Verfügbarkeit. Sollten wir vielleicht Transceiver-Module mit SPI-Schnittstelle verwenden? Das hätte aber eine völlige Neufassung der Firmware bedeutet, die zu diesem Zeitpunkt fast fertig war.

Nach weiteren Fehlversuchen mit 433-MHz-FM-Modulen, die zwar prima funktionierten, sich aber (nach Lieferung und Bezahlung natürlich) als „not recommended for new designs" erwiesen, stießen wir auf die Kombi TXM-433-LR und RXM-433-LR von Linx Technologies. Ok, es sind nicht die billigsten Module, aber sie sind leicht, längerfristig und in größeren Stückzahlen erhältlich. Und sie sparen im Gebrauch eine Menge Geld, da insbesondere der Sender sehr effizient mit der Batterie umgeht. Die Linx-Funkmodule sind auch in 315-MHz- und 418-MHz-Versionen für überseeische Anwendungen erhältlich, in denen das 433-MHz-ISM-Band anderen Einsatzgebieten zugeordnet ist.

Die Hardware des Senders

Im Zentrum des Sender-Schaltplans in **Bild 1** sitzt ein Mikrocontroller ATmega328P-AU, der von einem 8-MHz-Quarz getaktet wird. Es handelt sich um genau den gleichen Mikrocontroller, der häufig auf Arduino-Uno- und Nano-Boards verwendet wird. Zuerst wollten wir den überaus bekannten 1-Wire-Temperatursensor DS18B20 einsetzen, aber angesichts einer minimalen Versorgungsspannung von 3 V (Datenblatt [2]) war das kein guter Gedanke für ein von einer 3-V-Batterie betriebenes Gerät. Das es aber viele andere Temperatursensoren gibt, die bei Versorgungsspannungen unter 3 V problemlos funktionieren, haben wir uns für einen unbekannteren Sensor entschieden, den DS2438Z+. Auch dieser Sensor verwendet die Onewire-Architektur und besitzt eine eindeutige 64-Bit-Seriennummer, die zur eindeutigen Identifikation des Senders verwendet werden kann. Deshalb kann der Empfänger nur mit diesem bestimmten Sender gepaart werden. Der DS2438Z+ ist übrigens kein „richtiger" Temperatursensor, sondern wurde zur Überwachung von Batteriespannungen (bis hinunter zu 2,4 V) entworfen, besitzt aber einen integrierten Temperatursensor, den wir für unsere Zwecke nutzen können.

Zwar ist die Temperaturmessung des DS2438Z+ mit einem maximalen Fehler von ±2 °C weniger genau als die des DS18B20, aber in der Praxis durchaus akzeptabel.

Das Sender-Funkmodul ist der bereits erwähnte Typ TXM-433-LR von Linx Technologies [3]. Intern besteht es aus einem VCO, der von einem Frequenzsynthesizer gesperrt wird, der seine Referenz von einem hochpräzisen Quarz bezieht. Der VCO-Ausgang wird von einem internen Leistungsverstärker gepuffert, der in der Lage ist, +10 dBm, also 10 mW, an eine 50-Ω-Last abzugeben. Bei voller Leistung und einem Tastverhältnis von 50% beträgt die Stromaufnahme etwa 5 mA, was im Vergleich zu ähnlichen Funkmodulen gering ist. Mit Ausnahme einer Antenne sind keine externen HF-Bauteile erforderlich. Die Antenne muss nicht selber gestaltet werden, es kommt eine fertige, spiralförmig gewickelte Halbwellenantenne zum Einsatz, die senkrecht zur Platine angebracht wird. Die abgestrahlte Leistung ist über den Widerstand R1 einstellbar. Standardmäßig ist für maximale Ausgangsleistung ein Widerstand von 0 Ω montiert, es kann jedoch erforderlich sein, die HF-Leistung zu reduzieren, um den örtlichen Vorschriften zu entsprechen. Weitere Informationen entnehmen Sie bitte dem Datenblatt des TXM-433-LR.

Zu den anderen Bauteilen auf der Senderplatine gehören

- eine Reset-Taste,
- der Programmieranschluss K1,
- ein (serieller) Debug-Verbinder K2,
- die Lötjumper SJ1, SJ2, SJ3 und

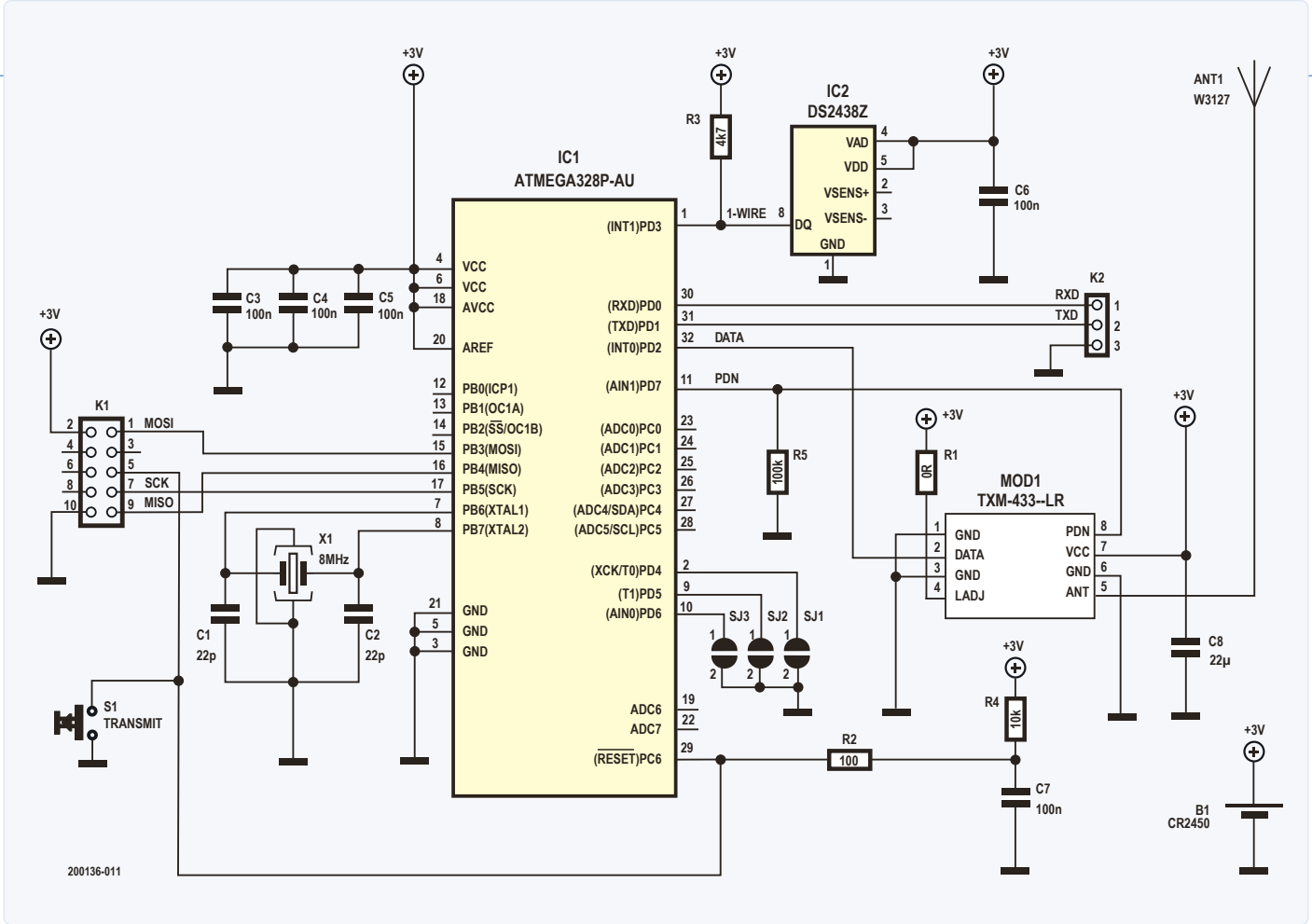


Bild 1. Schaltplan des Temperatur-Senders. Ein Sensor DS2438Z+ ersetzt einen DS18B20, damit die Schaltung bis hinunter zu 2,4 V arbeiten kann.

› eine CR2450-Batterie zur Stromversorgung der Schaltung.

Die Steckverbinder K1 und K2 werden im normalen Betrieb nicht gebraucht und müssen nicht montiert werden. Mit den Lötbrücken SJ1, SJ2, SJ3 kann das Übertragungsintervall zwischen 128 s und 184 s eingestellt werden. Dadurch wird das Risiko reduziert, dass mehrere Sender gleichzeitig Daten senden und es dadurch zu Datenkollisionen kommt. Zur Stromversorgung haben wir uns für eine Lithiumbatterie vom Typ CR2450 entschieden, da diese eine höhere Kapazität als die häufiger anzutreffende CR2032 besitzt, und dies zu einem nur geringfügig höheren Preis. Bei unserem Prototyp lag die Batteriespannung nach sechs Monaten Dauerbetrieb (mit einem Übertragungsintervall von 144 s) immer noch bei 2,991 V. Während einer Datenübertragung knickte die Spannung kurzzeitig auf etwa 2,93 V ein.

Software für den Sender

Die meiste Zeit befindet sich der Mikrocontroller im Power-Down-Modus, wobei der Watchdog-Timer läuft. Der Watchdog-Timer ist auf sein maximales Abschalt-Intervall von 8 s eingestellt. Wenn die Timeout-Periode verstrichen ist, generiert der Watchdog-Timer einen Interrupt, um den Mikrocontroller zu wecken. Ein Zähler im RAM wird dann dekrementiert. Solange der Zählwert noch nicht null erreicht hat, wird der Watchdog-Timer erneut gestartet und der Mikrocontroller wieder in den Power-Down-Modus versetzt. Dieser Vorgang dauert etwa 20 Instruktionen. Wenn der Zähler nach einem Watchdog-Timer-Interrupt null erreicht, werden die Temperaturregister und die 64-Bit-Seriennummer des DS2438Z+ gelesen. Schlägt dies fehl, versucht die Software vier weitere Male, den DS2438Z+ zu lesen.

Die Software unterstützt übrigens auch die Temperatursensoren DS18B20, DS18S20 und DS1822, die eingesetzt werden könnten, wenn der Sender von einer 3,3-V-Spannungsquelle statt von einer Lithiumbatterie versorgt wird. Ist der DS2438Z+ eingesetzt, wird dessen 1-Wire-Familiencode im Programm so geändert, dass er dem Familiencode eines DS18B20 entspricht. Der CRC wird natürlich ebenfalls neu berechnet.

Die 64-Bit-Seriennummer mit dem neuen Familiencode, der Temperaturwert und der CRC werden zweimal mit einer vorangestellten Präambel zum Funkmodul übertragen. Die Funkmodule verwenden die OOK/ASK-Modulation, was bedeutet, dass der Träger zusammen mit den jeweiligen Daten ein- und ausgeschaltet wird.

Wenn Sie einfache serielle Daten in das Sendemodul einspeisen, können Sie nicht erwarten, dass sie fehlerfrei beim Empfänger ankommen. Ohne einen anwesenden 433-MHz-Träger würde der Empfänger Hintergrundrauschen oder Störungen von anderen Geräten aufnehmen, die ebenfalls auf 433 MHz arbeiten. Um diese Phänomene zu unterdrücken, müssen die Daten zunächst in etwas kodiert werden, das für die Funkübertragung besser geeignet ist. Die „Präambel" ermöglicht es dem Empfänger, ein bekanntes Muster zu erkennen und sich mit diesem zu synchronisieren, bevor die Daten dekodiert werden. Zu den bekannteren Standards gehören die Manchester-Kodierung [4] und die der RadioHead-Bibliothek für Arduino [5].

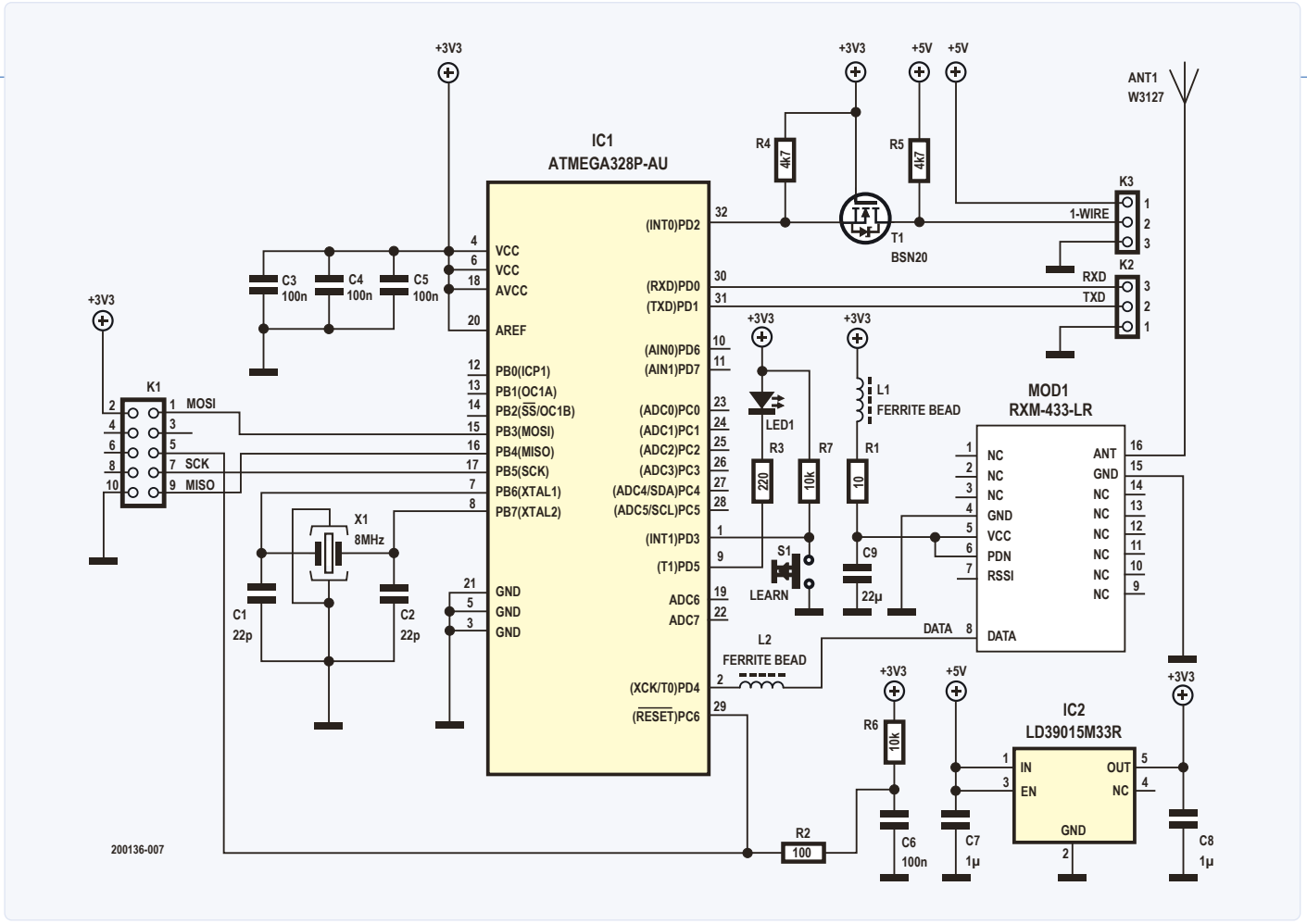


Bild 2. Schaltung des Temperatur-Empfängers. Zu den Hauptaufgaben der Mikrocontroller-Firmware gehören die Verarbeitung demodulierter Funksignale und die Interpretation der 1-Wire-Befehle.

Die Daten werden während der Übertragung nicht verschlüsselt, da die Außentemperatur wohl kaum als sensible Information zu betrachten ist.

Nach der Datenübertragung wird der Status der Lötjumper SJ1...SJ3 gelesen und der Zähler für die Anzahl der Sleep-Zyklen im RAM gesetzt. Die Anzahl ergibt sich aus dem Wert von 16 plus der durch die Lötjumper repräsentierten Binärzahl (0...7), so dass die Gesamtzahl der Sleep-Zyklen zwischen 16 und 23 liegt. Dann wird der Watchdog-Timer gestartet, der Mikrocontroller geht erneut in den Power-Down-Modus und der Zyklus wiederholt sich. Nach einem Reset wird auch der gesamte Zyklus neu gestartet. Durch Drücken des Reset-Knopfes wird so eine drahtlose Datenübertragung eingeleitet, was beim Einrichten einer Übertragungsstrecke nützlich ist.

Die Hardware des Empfängers

Der Schaltplan des Empfängers in **Bild 2** ist dem des Senders sehr ähnlich und basiert ebenfalls auf einem ATmega328P-Mikrocontroller. Die einzigen Verbindungen zur Außenwelt sind Vcc, GND und DQ (1-Wire-Daten). Der Überlagerungsempfänger RXM-433-LR von Linx Technologies [6] verwendet den selben Antennentyp wie der Sender. Dieses AM/OOK-Empfängermodul zeichnet sich, so das Datenblatt, durch „eine fortschrittliche Single-Conversion-Superheterodyn-Architektur aus, die zu einer außergewöhnlichen Empfindlichkeit und einer hervorragenden Reichweitenleistung führt".

Rauschen und Welligkeit der Stromversorgung können die Empfängerempfindlichkeit aber erheblich beeinträchtigen. Um sicherzustellen, dass die Versorgungsspannung für das Empfängermodul so sauber wie möglich ist, wurden die Ferritperle L1 und der 10-Ω-Widerstand R1 in Reihe mit dem Vcc-Pin geschaltet und der 22-µF-Entkopplungskondensator C9 hinzugefügt. Die Ferritperle L2 in der Datenleitung blockiert Störungen durch den Mikrocontroller selbst (zugegeben: Perlenform weisen diese Entstörinduktivitäten schon lange nicht mehr auf).

Da der RXM-433-LR einen Versorgungsspannungsbereich von 2,7...3,6 V besitzt, haben wir uns dafür entschieden, die Schaltung mit einer Versorgungsspannung von 3,3 V zu betreiben. Da das Nixie-Bargraph-Thermometer jedoch mit 5 V arbeitet, haben wir für eine einfache Verbindung zwischen Empfängerschaltung und Thermometer mit dem LD39015M33R (IC2) einen Spannungsregler eingesetzt, der einen abnorm niedrigen Spannungsabfall und auch noch ein sehr geringes Rauschen aufweist. Der Spannungsregler wird mit dem MOSFET-Transistor T1 wie ein bidirektionaler Logikpegelwandler eingesetzt.

Wenn der Empfänger mit einem 3,3-V-One-wire-Bus (wie bei der VFD-Röhren-Uhr [7]) verwendet wird, ist die Spannung „hinter" dem Regler immer noch hoch genug, damit die Schaltung ordnungsgemäß funktioniert. Neben einem (normalerweise nicht montierten) Programmier- und Debugging-Verbinder sind eine LED und ein Taster mit dem Mikrocontroller verbunden. Jedes Mal, wenn ein drahtloses Datenpaket empfangen wird, ändert die LED ihren Status. Auf diese Weise lässt sich überprüfen, ob der Sender noch funktionsfähig ist. Die Taste dient dazu, den Empfänger mit einem Sender zu koppeln.

STÜCKLISTE

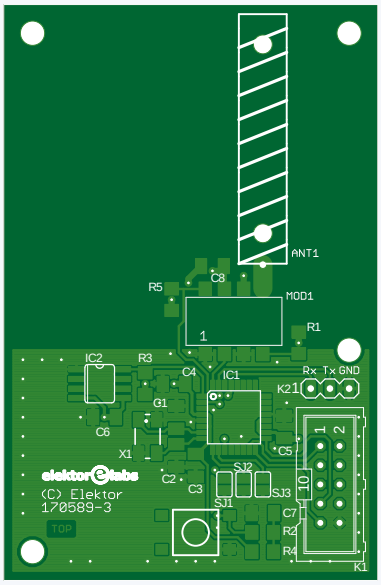
Sender

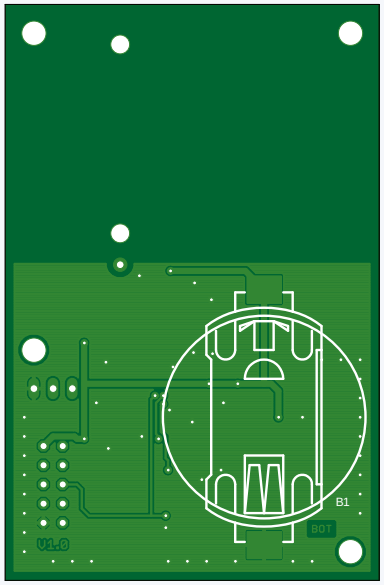
Widerstände
(alle 1206)
R1 = 0 Ω*
R2 = 100 Ω
R3 = 4k7
R4 = 10 k
R5 = 100 k

Kondensatoren
(alle 1206)
C1,C2 = 22 p, C0G/NP0
C3…C7 = 100 n, X7R
C8= 22 µF, X5R, 10V
(Kemet C1206C226M8PACTU)

Halbleiter
IC1 = ATmega328P-AU, programmiert
IC2 = DS2438Z+
MOD1 = TXM-433-LR, Linx Technologies

Außerdem
ANT1 = 433-MHz- Wendelantenne
(Pulselektronik W3127)
B1 = Batteriehalter Renata SMTU 2450N-1-LF
für und mit Lithium-Knopfzelle CR2450





K1 = 2x5-polige Stiftleiste mit Rahmen,
Raster 2,54 mm (optional)
K2 = 3-polige Stiftleiste, vertikal,
Raster 2,54 mm (optional)
S1 = Tastschalter, SMD,
TE Connectivity FSM6JSMA
X1 = 8-MHz-Quarz,
Abracon Typ ABMM2-8.000MHZ-E2-T oder
Würth Elektronik # 830055663

* siehe Text

Hinweis: Ein Bauteilsatz für den Sender ist im Elektor-Store erhältlich.

STÜCKLISTE

Empfänger

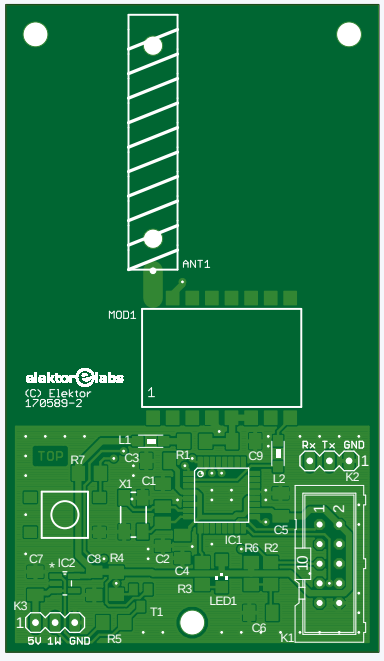
Widerstände
(alle 1206)
R1 = 10 Ω
R2 = 100 Ω
R3 = 220 Ω
R4,R5 = 4k7
R6,R7 = 10 k

Kondensatoren
(alle 1206)
C1,C2 = 22 p, C0G/NP0
C3-C6 = 100 n, X7R
C7,C8 = 1 µ, X7R
C9 = 22 µ, X5R, 10V
(Kemet C1206C226M8PACTU)

Halbleiter
LED1 = grün, 1206, Broadcom HSMG-C150
IC1 = ATmega328P-AU, programmiert
IC2 = LD39015M33R
MOD1 = RXM-433-LR, Linx Technologies
T1 = BSN20

Außerdem
ANT1 = 433-MHz- Wendelantenne
(Pulselektronik W3127)
K1 = 2x5-polige Stiftleiste mit Rahmen,
Raster 2,54 mm (optional)
K1 = 3-polige Stiftleiste, vertikal,
Raster 2,54 mm (optional)
K3 = 3-polige Stiftleiste, gewinkelt,
Raster 2,54 mm mit
Buchsenklemmengehäuse # 61900311621
und 3 vorgecrimpte Kabel # 619100126015
(Würth Elektronik)
L1,L2 = Ferritperle, Fair-rite 1206 2512067007Y3
S1 = Tastschalter, SMD,
TE Connectivity FSM6JSMA
X1 = Quarz 8 MHz,
Abracon ABMM2-8.000MHZ-E2-T oder
Würth Elektronik # 830055663

Hinweis: Ein Bauteilsatz für den Empfänger ist im Elektor-Store erhältlich.



Die Software des Empfängers

Bis vor kurzem war der 1-Wire-Bus von Dallas Semiconductors (jetzt Maxim) ein sehr beliebtes Kommunikationsbussystem zum Aufbau von MicroLAN-Netzwerken, zum Beispiel für den Einsatz in Hausautomatisierungssystemen. Aufgrund des zunehmenden Einsatzes von drahtlosen Lösungen kommt Onewire aber immer mehr aus der Mode. Dennoch halten sich einige 1-Wire-Bausteine wie der DS18B20 und ähnliche Temperatursensoren hartnäckig, weil sie billig sind, genau, einfach mit einem Mikrocontroller zu verbinden und in leicht zu lötenden TO-92-Gehäusen angeboten werden. Dies waren die Hauptgründe für den Einsatz im Elektor-Nixie-Bargraph-Thermometer (und im IV-22-VFD-Uhrenprojekt) sowie für die Simulation eines DS18B20 mit unserem Empfängermodul. Da die ursprünglichen 1-Wire-Patente ausgelaufen sind, ist dies auch in der sengenden Hitze des Mittleren Westens völlig legal.

Die Software horcht sowohl auf das Empfänger-Funkmodul als auch auf den 1-Wire-Bus. Wenn eine Datenübertragung beim Empfänger eintrifft, werden die kodierten Daten dekodiert und auf ihre Gültigkeit überprüft. Wenn ein gültiges Datenpaket empfangen wird, wird die empfangene 1-Wire-Seriennummer anschließend mit der im EEPROM des Mikrocontrollers gespeicherten 1-Wire-Seriennummer verglichen. Wenn beide übereinstimmen, wird das Scratchpad (das unter anderem die Temperaturwert-Bytes enthält) des simulierten DS18B20-Temperatursensors aktualisiert, wenn nicht, wird das Datenpaket verworfen.

Der simulierte Onewire-Slave unterstützt die folgenden 1-Wire-Befehle (ROM- und DS18B20-Funktionsbefehle):

- `Read Rom` [33h]
- `Match Rom` [55h]
- `Search Rom` [F0h]
- `Convert T` [44h]
- `Write Scratchpad` [4Eh]
- `Read Scratchpad` [BEh]
- `Copy Scratchpad` [48h]
- `Recall E2` [B8h]
- `Read Power Supply` [B4h]

Es ist nicht möglich, das Empfängermodul im parasitären Modus zu betreiben. Der Befehl `Read Power Supply` gibt immer den Status „externally powered" zurück.

Die 1-Wire-Slave-Funktionen des Empfängermoduls haben stets Vorrang vor den Funkfunktionen. Wenn also eine 1-Draht-Bus-Kommunikation initiiert wird, während ein Datenpaket empfangen und dekodiert wird, wird der Datenempfang über Funk abgebrochen. Da sowohl das Thermometer als auch die VFD-Uhr den Temperaturwert nur einmal pro Minute lesen, ist die Wahrscheinlichkeit, dass dies häufig geschieht, relativ gering. Das ist auch der Grund, warum das kürzeste Sendeintervall des Senders 128 s und nicht glatte 120 s beträgt.

Konstruktion und Einsatz in der Praxis

Obwohl das Sende- als auch das Empfangsmodul werden als vormontierte und getestete Platinen im Elektor-Store angeboten werden, finden Sie hier die Bestückungspläne für die Platinen und die zugehörigen Stücklisten, falls Sie die Elektronik in Eigenregie erstellen möchten. Ebenso können die Gerber-Dateien für die Platinen und die ATmega-Firmware für Sender und Empfänger von der Projektseite [8] heruntergeladen werden.

Die TX- und RX-Boards sind zwar vorbestückt, damit sie funktionieren, muss aber noch ein wenig Hand angelegt werden. K3, eine 3-polige, gewinkelte 0,1-Zoll-Stiftleiste muss angebracht und die drei vorgecrimpten Drähte von hinten in die 3-Weg-Buchse geschoben werden, bis diese mit einem zarten „Klick" einrasten. Führen Sie die Drähte durch das Loch in der Seitenwand des Gehäuses und löten Sie die anderen Enden an eine Buchsenleiste, die auf die Stifte der Thermometerplatine passt. Bevor Sie die Verbindungen mit Schrumpfschlauchstückchen sichern, sollten Sie sich vergewissern, dass die Anschlüsse GND, DQ und 5 V auch richtig belegt sind.

Montieren Sie nun den Empfänger in sein Gehäuse, schließen Sie ihn an das Thermometer an und versorgen Sie das Thermometer mit Strom. Das Thermometer zeigt nun den vollen Skalenendwert an, da der simulierte DS18B20 einen Rückstelltemperaturwert von 85 °C hat, genau wie der „echte" DS18B20.

Bild 3. Die für den Einsatz im Sender vorgeschlagene 3-V-Lithium-Knopfzelle CR2450-3V ist etwas teurer als die CR2032, bietet aber mehr Kapazität.

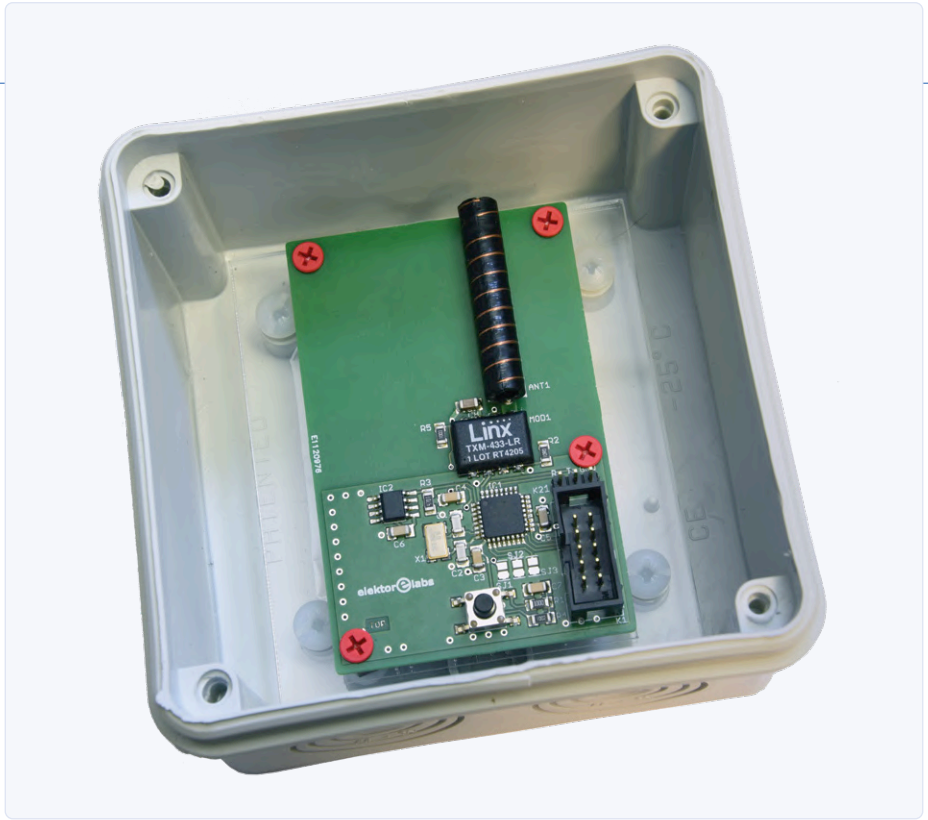


Bild 4. Um den Betrieb im Freien zu ermöglichen, wurde der Sender in eine Verteilerdose (IP65) eingepasst.

Wenn der Empfänger noch nie mit einem Sender gepaart war, enthält das EEPROM keine 1-Wire-Seriennummer. In diesem Fall werden die 1-Wire-Slave-Funktionen ausgesetzt, was zu keiner Anzeige auf der IN-9-Röhre führt.

Setzen Sie nun eine CR2450-Lithium-Knopfzelle in den Batteriehalter auf der Rückseite des Senders ein (**Bild 3**). Positionieren Sie den Sender einige Meter vom Empfänger entfernt und drücken Sie die Taste am Empfänger. Die grüne LED beginnt zu blinken, was anzeigt, dass sich der Empfänger im Lernmodus befindet. Drücken Sie jetzt die Taste am Sender. Wenn alles gut ist, beendet die LED am Empfänger ihr Blinken, was anzeigt, dass der Empfänger jetzt mit dem Sender gekoppelt ist und ein gültiger Temperaturwert empfangen wird. Da das Thermometer den 1-Wire-Sensor nur einmal pro Minute liest, kann es einige Zeit dauern, bis der tatsächliche Temperaturwert angezeigt wird.

Montieren Sie den Sender für den Einsatz im Freien in einer wasserdichten Box (billig und gut ist eine Verteilerdose der Schutzart IP65), zusammen mit einem kleinen Beutel Kieselgel (Silikagel), um die Elektronik trocken zu halten (**Bild 4**).

Jedes Mal, wenn Sie das Thermometer einschalten, kann es einige Minuten dauern, bis der Temperaturwert empfangen und angezeigt wird. Während dieser Zeit zeigt das Thermometer seinen Skalenendwert an. Es ist möglich, mehrere Sender und Empfänger gleichzeitig zu verwenden. Da der Sender den Familiencode des DS2438Z+ in den eines DS18B20 ändert, besteht eine winzige Chance, dass die Seriennummer des simulierten DS18B20 nicht eindeutig ist, wenn ein oder mehrere Empfänger im 1-Draht-Netzwerk mit anderen (echten) DS18B20-Temperatursensoren verwendet werden.

Schließlich stellten wir im praktischen Einsatz fest, dass einige Arduino-Nano-Boards mehr Störungen (RFI) verursachen als andere. Diese Interferenzen (aber auch Hindernisse und Metallstrukturen in der Funkstrecke) können die effektive Reichweite beeinträchtigen.

200136-02

IM ELEKTOR-STORE

- **Empfängermodul mit überarbeiteter Acrylskala und Kabel**
  www.elektor.de/170589-72
- **Sendermodul mit Batterie und wasserdichtem Gehäuse**
  www.elektor.de/170589-73
- **Bausatz Nixie-Bargraph-Thermometer (Originalausführung)**
  www.elektor.de/nixie-bargraph-thermometer-170589-71

WEBLINKS

[1] **Nixie-Bargraph-Thermometer, Elektor Juli/August 2018**: www.elektormagazine.de/magazine/elektor-201807/41680

[2] **Datenblatt Smart Battery Monitor DS2438**: https://datasheets.maximintegrated.com/en/ds/DS2438.pdf

[3] **Linx LR Series Transmitter Module Data Guide**: www.linxtechnologies.com/wp/wp-content/uploads/txm-fff-lr.pdf

[4] **Manchester-Datenkodierung für die Funkkommunikation**: www.maximintegrated.com/en/design/technical-documents/app-notes/3/3435.html

[5] **Bibliothek RadioHead Packet Radio für eingebettete Mikrocontroller**: www.airspayce.com/mikem/arduino/RadioHead/

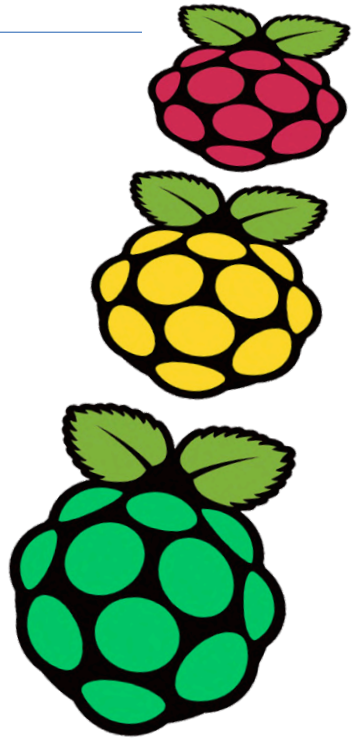
[6] **Linx LR Series Receiver Module Data Guide**: www.linxtechnologies.com/wp/wp-content/uploads/rxm-fff-lr.pdf

[7] **VFD-Röhren-Uhr mit ESP32, Elektor Mai/Juni 2018**: www.elektormagazine.de/magazine/elektor-201805/41489

[8] **Projektseite**: www.elektormagazine.de/200136-02

Multitasking mit Raspberry Pi

Am Beispiel einer Ampelsteuerung



Von **Dogan Ibrahim**

Zielgruppe des neu erschienenen Buchs „Multitasking with RPi“ sind Studenten, praktizierende Ingenieure und fortgeschrittene „Freizeit-Elektroniker“, die an der Entwicklung von Multitasking-Projekten mit der Programmiersprache Python 3 auf dem Raspberry Pi-Computer interessiert sind. Hier ein beispielhaftes Projekt aus dem Buch.

Multitasking ist zu einem der wichtigsten Themen bei Mikrocontroller-Systemen geworden, nämlich bei Automatisierungsanwendungen. Mit zunehmender Komplexität der Projekte wird den Systemen mehr Funktionalität abverlangt. Solche Projekte umfassen den Einsatz mehrerer zusammenhängender Aufgaben, die auf demselben System laufen und sich die CPU (oder mehrere CPUs) teilen, um die erforderlichen Operationen zu implementieren. Aus diesem Grund hat die Bedeutung des Multitasking-Betriebs von auf Mikrocontrollern basierenden Anwendungen in den letzten Jahren stetig zugenommen. Viele komplexe Automatisierungsprojekte verwenden heutzutage irgendeine Form eines Multitasking-Kernels. In dem Buch wird die Programmiersprache Python 3 auf dem Raspberry Pi 4 verwendet, aber auch ältere Modelle des Raspberry Pi können ohne Änderung des Codes eingesetzt werden.

Das Buch ist in Projekte eingeteilt. Sein Hauptziel ist es, die grundlegenden Funktionen von Multitasking mit Python auf dem Raspberry Pi zu vermitteln. In dem Buch werden viele vollständig getestete Projekte vorgestellt, in denen das Multitasking-Modul von Python verwendet wird. Jedes Projekt wird ausführlich beschrieben und diskutiert und jedes Projekt ist mit vollständigen Programmlistings versehen. Die Leser sollten in der Lage sein, die Projekte so zu verwenden, wie sie sind, oder sie so zu modifizieren, dass sie ihren eigenen Bedürfnissen entsprechen.

Beispiel: Eine Ampelsteuerung

In diesem Projekt wird eine einfache Ampelsteuerung für die Kreuzung der East Street mit der North Street entworfen. Es gibt Ampeln an jeder Ecke der Kreuzung. In der Nähe der Ampeln auf der North Street befinden sich Zebrastreifen und beidseitig der Straße Tasten für Fußgänger. Durch Drücken eines Tasters werden alle Ampeln am Ende ihres Zyklus auf Rot geschaltet und verbleiben für eine Zeit in diesem Zustand. Zudem ertönt ein Summer, der akustisch anzeigt, dass die Fußgänger die Straße sicher überqueren können. Außerdem ist eine LC-Anzeige an das System angeschlossen, auf der angegeben wird, ob die Kreuzung momentan für Fußgänger oder für den Autoverkehr freigegeben ist. **Bild 1** zeigt die Anordnung der „Hardware“ an der Kreuzung.

In diesem Projekt sind für jeden Ampelzyklus und auch für die Dauer der Fußgänger-Freigabe die folgenden festen Zeiten angegeben. Der Einfachheit halber wird angenommen, dass die Ampeln beider Straßen der Kreuzung die gleichen Intervalle aufweisen:

- Zeit für Fußgänger: 10 Sekunden
- Rot: 19 Sekunden
- Rot+Gelb: 2 Sekunden
- Grün: 15 Sekunden
- Gelb: 2 Sekunden

Die Gesamtzykluszeit der Ampeln in diesem Beispielprojekt ist also auf 38 Sekunden plus

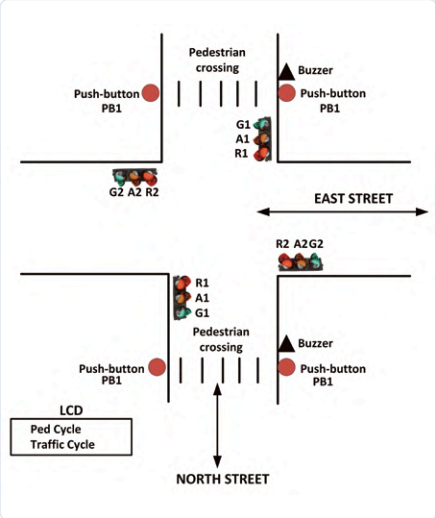


Bild 1. Anordnung der Ampeln an der Kreuzung.

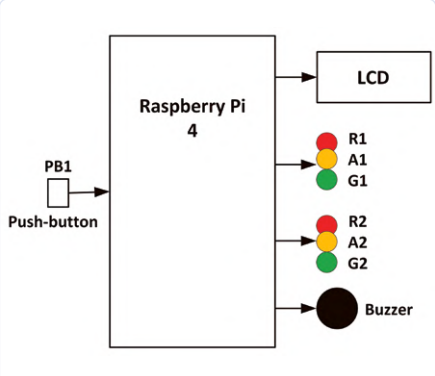


Bild 2. Blockdiagramm des Projekts.

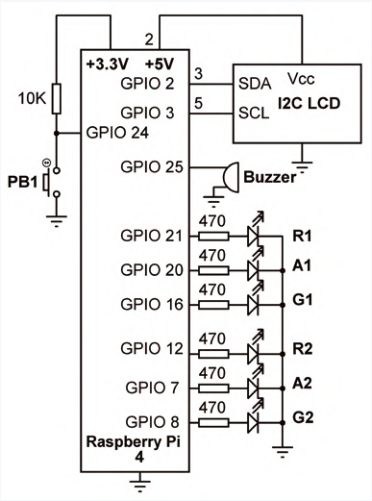


Bild 3. Schaltplan des Projekts.

10 Sekunden, wenn die Fußgängertaste gedrückt wurde, eingestellt.
Die Abfolge der einzelnen Ampelphasen ist wie gewohnt: Rot…Rot+Gelb…Grün…Gelb…Rot. **Bild 2** zeigt das Blockdiagramm des Projekts, **Bild 3** den Schaltplan. In diesem Projekt werden rote (R), gelbe (A) und grüne (G) LEDs verwendet, um die realen Ampeln darzustellen. Die folgenden Verbindungen werden zwischen dem Raspberry Pi und der Ampelanlage gelegt:

Raspberry Pi	Ampel-Hardware
GPIO 21	LED R1
GPIO 20	LED A1
GPIO 16	LED G1
GPIO 12	LED R2
GPIO 7	LED A2
GPIO 8	LED G2
GPIO 25	Summer
GPIO 2	LCD SDA
GPIO 3	LCD SCL
GPIO 24	PB1 (Drucktaster)

Zu Beginn des Programms traffic.py in **Listing 1** (Download von [1]) werden die Module mit den Namen *RPi, time, I2C LCD driver* und *multiprocessing* in das Programm importiert und zwei Queues mit den Namen `pedq` und `lcdq` erstellt.
Im Programm werden zwei Funktionen mit den Namen `ONOF` und `CONF_OUT` definiert. Die Funktion `ONOF` hat zwei Argumente: `port` und `state`. Diese Funktion sendet den Status (0 oder 1) an den angegebenen GPIO-Port. Die

Listing 1. Das Programm traffic.py

```
#--------------------------------------------------------------------
#                    TRAFFIC LIGHTS CONTROLLER
#                    =========================
#
# This is a traffic lights controller project controlling lights
# at a junction. 6 LEDS are used to represent the traffic lights.
# Additionally a button is used for pedestrian crossing, and an
# LCD shows the state of the traffic lights at any time
#
# Author: Dogan Ibrahim
# File  : traffic.py
# Date  : May 2020
#--------------------------------------------------------------------
import RPi.GPIO as GPIO                         # Import RPi
import multiprocessing                          # Import multiprocessing
import time                                     # Import time
import RPi_I2C_driver                           # I2C library
LCD = RPi_I2C_driver.lcd()                      # Import LCD
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)                          # GPIO mode BCM
pedq = multiprocessing.Queue()                  # Create queue
lcdq = multiprocessing.Queue()                  # Create queue

#
# This function sends data 'state (0 or 1)' to specified port
#
def ONOF(port, state):
   GPIO.output(port, state)

#
# This function configures the specified port as output
#
def CONF_OUT(port):
   GPIO.setup(port, GPIO.OUT)

#
# Process to control the lights
#
def Lights():                                   # Process Lights
   R1=21; A1=20; G1=16                          # LED connections
   R2=12; A2=7;  G2=8                           # LED conenctions
   Buzzer=25                                    # Buzzer connection
   CONF_OUT(R1); CONF_OUT(A1); CONF_OUT(G1)     # Configure
   CONF_OUT(R2); CONF_OUT(A2); CONF_OUT(G2)     # Configure
   CONF_OUT(Buzzer)                             # Configure
   ONOF(R1,0); ONOF(A1,0); ONOF(G1,0); ONOF(R2,0); ONOF(A2,0); ONOF(G2,0)
   ONOF(Buzzer, 0)

   RedDuration = 15
   GreenDuration = 15
   AmberDuration = 2

#
# Control the traffic light sequence
#
   while True:          # Do forever
      ONOF(R1,0); ONOF(A1,0); ONOF(G1,1); ONOF(R2,1); ONOF(A2,0);
   ONOF(G2,0)
      time.sleep(RedDuration)
      ONOF(G1,0); ONOF(A1,1)
      time.sleep(AmberDuration)
      ONOF(A1,0); ONOF(R1,1); ONOF(A2,1)
      time.sleep(AmberDuration)
      ONOF(A2,0); ONOF(R2,0); ONOF(G2,1)
      time.sleep(GreenDuration)
```

```
      ONOF(G2,0); ONOF(A2,1)
      time.sleep(AmberDuration)
      ONOF(A2,0); ONOF(A1,1); ONOF(R2,1)
      time.sleep(AmberDuration)

      while not pedq.empty():                # If ped request
         lcdq.put(1)
         ONOF(G1,0); ONOF(R1,1); ONOF(A1,0)  # Only RED ON
         ONOF(G2,0); ONOF(R2,1); ONOF(A2,0)  # Only RED ON
         d = pedq.get()                      # Clear ledq
         ONOF(Buzzer, 1)                     # Buzzer ON
         time.sleep(10)                      # Wait 10 secs
         ONOF(Buzzer, 0)                     # Buzzer OFF
         d = lcdq.get()                      # Clear lcdq


def Pedestrian():                            # Process Pedestrian
   PB1 = 24
   GPIO.setup(PB1, GPIO.IN)                  # PB1 is input

   while True:         # Do forever
      while GPIO.input(PB1) == 1:            # PB1 not pressed
         pass
      pedq.put(1)                            # Send to Ped queue
      while GPIO.input(PB1) == 0:            # PB1 not released
         pass

#
# Create the processes
#
p = multiprocessing.Process(target = Lights, args = ())
q = multiprocessing.Process(target = Pedestrian, args = ())
p.start()
q.start()

#
# LCD Display control. Display 'Ped Cycle' or 'Traffic Cycle'
#
LCD.lcd_clear()          # Clear LCD
LCD.lcd_display_string("TRAFFIC CONTROL", 1) # Heading

while True:        # DO forever
   if not lcdq.empty():
      LCD.lcd_display_string("Ped Cycle    ", 2)
   else:
      LCD.lcd_display_string("Traffic Cycle", 2)
   time.sleep(1)
```

Bild 4. Beispiel für die Anzeige auf dem LCD.

Funktion `CONF_OUT` besitzt einen Parameter namens `port`. Diese Funktion konfiguriert den angegebenen GPIO-Port als Ausgang.
Es gibt zwei Prozesse im Programm: `Lights` und `Pedestrian`. Zu Beginn des `Lights`-Prozesses werden die Verbindungen zwischen dem Raspberry Pi und den Ampel-LEDs sowie dem Summer definiert und diese Ports als Ausgänge konfiguriert. Zusätzlich werden all diese Portausgänge auf „0" gesetzt, so dass alle LEDs und der Summer ausgeschaltet (OFF) sind. Außerdem werden die LEDs in der richtigen Reihenfolge und mit dem richtigen Timing festgelegt. Am Ende der Funktion wird geprüft, ob die Fußgängertaste gedrückt wurde, was der Fall ist, wenn der Queue `pedq` nicht leer ist. Während des Fussgängerzyklus werden alle Ampeln auf Rot geschaltet, um den Autoverkehr zu stoppen und den Fussgängern eine ungefährliche Überquerung der Straßen zu ermöglichen.
Außerdem wird während des Fussgängerzyklus der Summer für 10 Sekunden aktiviert, um die Fussgänger darüber zu informieren, dass es jetzt sicher ist, die Straße zu überqueren.
Der Fussgängerprozess überwacht kontinuierlich die Taste an PB1. Wenn sie gedrückt wird, wird eine „1" an den Queue `pedq` gesendet, so dass der Prozess `Lights` diese Aktion leicht erkennen und den Fussgängerzyklus starten kann.
Das Hauptprogramm steuert das LCD. Wenn das Programm gestartet wird, wird die Meldung „TRAFFIC CONTROL" in der ersten Zeile des LCDs angezeigt. In der zweiten Zeile erscheint, wie in **Bild 4** zu sehen, der Zustand des Queues `lcdq` in Form der Meldungen „Ped Cycle" beziehungsweise „Traffic Cycle".

200381-02

WEBLINK

[1] **Download des Programms Traffic.py**: www.elektormagazine.de/200381-02

Timer für Kopfhörerverstärker

Von **James Glyn**

Schon wieder war die Batterie meines batteriebetriebenen Röhrenkopfhörerverstärkers leer, weil ich vergessen hatte, ihn auszuschalten. Nachdem mir das zum x-ten Mal passiert war, beschloss ich, eine Zeitschaltuhr zu bauen, die den Verstärker automatisch abschaltet. Bei der Überprüfung meiner Hörgewohnheiten schien mir eine Zeitspanne von 90 Minuten optimal zu sein. Eine Oszillatorschaltung wurde wegen der potentiellen hörbaren Interferenzen und des schlechten Stromverbrauchs schnell verworfen. Stattdessen bot sich für das Abschalten ein Widerstands-Kondensator-Netzwerk zusammen mit einem Schmitt-Trigger und einem MOSFET als besonders stromsparende Möglichkeit an.

Die Schaltung in **Bild 1** ist mit einem CMOS-IC des Typs 4093 aufgebaut, einem vierfachen NAND-Gatter mit Schmitt-Trigger (IC1). Dieser Baustein kann Versorgungsspannungen von bis zu 15 V standhalten, mehr als genug für meine Zwecke. Jeweils zwei der Gatter steuern parallel geschaltet die Gates von IRL1404-Leistungs-MOSFETs (Q1, Q2) an. Am Eingang der Gates IC1A und IC1B befindet sich besagtes RC-Netzwerk mit dem sehr hochohmigen Widerstand R3 und dem „dicken" 1-mF-Kondensator C1. Die Werte sind so gewählt, dass bei einer Betriebsspannung von 12 V eine Zeitkonstante von ungefähr 100 Minuten erreicht wird.

Der Timer wird über einen gefederten SPDT-Wechseltaster mit Mittelstellung bedient (zwei getrennte Taster könnten die Batterie kurzschließen, wenn man sie gleichzeitig drückt). Im Ruhezustand liegen die Eingänge der Gatter IC1A und IC1B auf high; die Ausgänge sind low und der Transistor sperrt. Damit bleibt die Last CN2 (der Verstärker) abgeschaltet. Drückt man den Taster nun in Richtung R2, liegt low am Schmitt-Trigger, dessen Ausgang auf high kippt und über den Transistor den Verstärker an die Batteriespannung legt. Gleichzeitig lädt sich C1 dank des niedrigen Wertes von R2 sehr rasch auf. Lässt man den Taster los, sorgt die Spannung über C1 dafür, dass der Verstärker an der Batterie verbleibt. C1 entlädt sich, aber wegen des hohen Wertes von R3 nur sehr langsam. Nach etwa 100 Minuten (wegen der hohen Toleranz des Elkos ist der Timer nicht besonders präzise) ist die Schwelle erreicht, bei der der Schmitt-Trigger wieder zurück kippt, so dass der Transistor die Last wieder von der Batterie trennt. Der Abschaltvorgang lässt sich manuell auf einen Wimpernschlag beschleunigen, wenn man den Taster kurz in die andere Richtung gen R1 antippt.

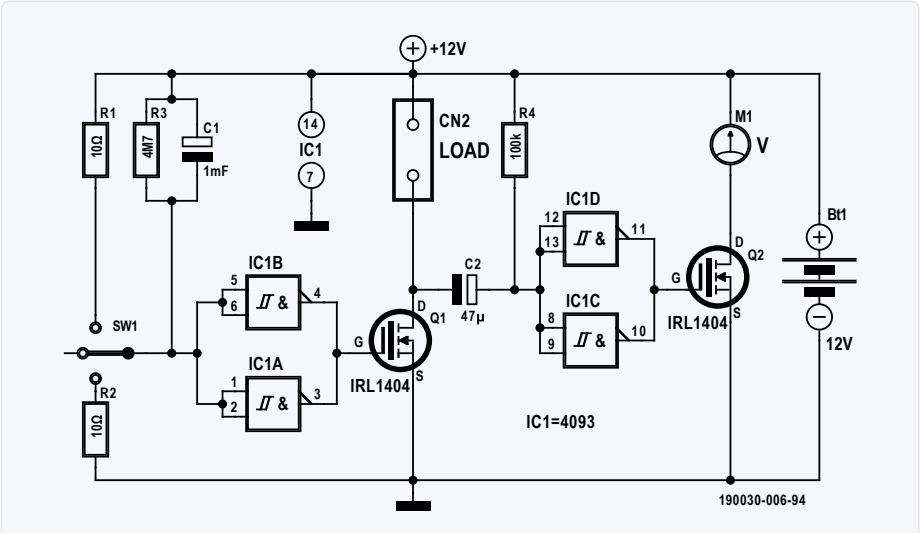


Bild 1. Die Einschaltzeit dieses Timers liegt bei etwa 100 Minuten, wenn er an einer 12-V-Batterie betrieben wird. Der zweite Timer zeigt die Batteriespannung für etwa 3 Sekunden an.

Bild 2. Der auf einer Platine aufgebaute Abschalttimer mit dem EIN-AUS-EIN-Taster und dem gewählten Frontplatten-Voltmeter.

Um die verbleibenden NAND-Gatter (IC1C/D) nicht zu verschwenden, beschloss ich, ein Voltmeter hinzuzufügen, das den Batteriestatus anzeigt. Dies geschieht aus Energiespargründen aber nur für 2...3 s nach dem Einschalten des Verstärkers. C2 und R4 liefern das Timing für diese Funktion. Wenn Q1 eingeschaltet wird, lädt sich C2 über R4 auf und aktiviert das Voltmeter über Q2. Sobald C2 vollständig aufgeladen ist, schaltet sich das Voltmeter wieder aus. C2 entlädt sich über die Last (den Verstärker) und R4, wenn die Last ausgeschaltet wird.
Das Voltmeter (**Bild 2**) ist ein Exemplar mit Miniatur-Sieben-Segment-LED-Anzeige, wie man es in vielen Online-Shops zu niedrigen Kosten erhält. Ich habe verschiedene Typen getestet und eine Stromaufnahme von nur 6 mA bis 16 mA festgestellt. Bei der kurzen Anzeigedauer hat das Voltmeter also nur geringe Auswirkungen auf die Standzeit der Batterie. Das gewählte Voltmeter besitzt eine runde schwarze Kunststoffblende, ideal für eine einfache Frontplattenmontage.
Der gewählte MOSFET kann problemlos mit allen möglichen Stromanforderungen von Batterieschaltungen umgehen, da er sich laut Datenblatt für Ströme bis zu 100 A einsetzen lässt. Er kann Gate-Source-Spannungen bis 20 V verarbeiten, wobei die Gate-Schwellenspannung ($V_{GS(th)}$) zwischen 1,0 V und 3,0 V liegt. Der Schmitt-Trigger-Eingang in den NAND-Gattern sorgt dafür, dass beide MOSFETs entweder vollständig ein- oder ausgeschaltet werden, was eine minimale Stromaufnahme garantiert. Mein Multimeter kann Ströme bis hinunter zu 100 nA messen, aber Tests ergaben, dass die Stromaufnahme im ausgeschalteten Zustand unter diesem Wert lag.
Der Drain-Source-Einschaltwiderstand der MOSFETs beträgt weniger als 6 mΩ, was bedeutet, dass die Schaltung wenig Einfluss auf den Verstärker hat und dass die Anzeige im Voltmeter tatsächlich die wahre Batteriespannung widerspiegelt. Die MOSFETs stecken Ströme bis 800 mA kaltlächelnd weg, so dass ich keinen Kühlkörper verwenden musste. Sollten größere Lasten an die Schaltung angeschlossen werden, könnte dies aber sehr wohl nötig sein. Ich sah auch keinen Bedarf für eine Diode in meiner Anwendung, aber wenn man eine induktive Last versorgen möchte, sollte man eventuell zwischen Drain und Source eine 1N4007 „falsch herum" einbauen, um Q1 vor Rück-EMF zu schützen.

190030-02

Open-Network-Wetterstation Mark 2

Teil 2: Software

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

In der Ausgabe Mai/Juni 2020 [1] haben wir die neue Open-Source-Wetterstation aus dem Elektor-Labor vorgestellt. Das System besteht aus einem im Elektor-Shop erhältlichen Mechanik-Kit (Wind- und Regensensoren, Gehäuse) sowie Elektronik rund um einen ESP32, mit dem sich die erhobenen Daten zu Clouddiensten wie openSenseMap und ThingSpeak senden lassen. Doch Hardware ist natürlich nur ein Teil des Projekts. In diesem Artikel erfahren Sie, wie die Funktionsblöcke der modular programmierten Software zusammenspielen. Dank eines intelligenten Mappers lassen sich Sensordaten flexibel mit Kommunikationskanälen verknüpfen. Auch der Webserver, der im Projekt eine zentrale Rolle spielt, wird im Artikel ausführlich beleuchtet.

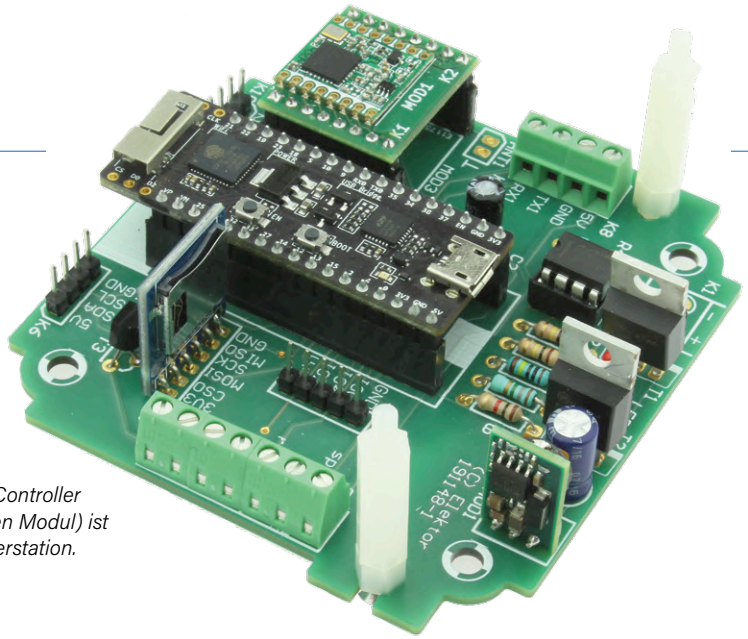


Bild 1. Der ESP32-Controller (auf dem schwarzen Modul) ist das Herz der Wetterstation.

Wir im Elektor-Labor freuen uns, dass Leser uns immer wieder Anregungen geben, wie wir unsere Projekte noch verbessern oder erweitern könnten. Daher achten wir darauf, Software von vorne herein so zu programmieren, dass nachträgliche Erweiterungen schnell realisiert werden können. Bei unserer Wetterstation [1] lässt sich übrigens auch die Hardware leicht erweitern. Der ESP32 (**Bild 1**) bietet durch seine I/O-Matrix eine sehr flexible Möglichkeit, benötigte Funktionen an fast allen Pins bereitzustellen, sei es SPI, I²C oder einfach nur ein digitaler Eingang für einen Taster.
Doch welche Module umfasst unsere Software und warum ist sie so flexibel? Schauen wir uns zuerst die Funktionen an, die durch die Software [2][3] realisiert werden müssen.

Grundfunktionen

Sensoren

Bei den Sensoren werden momentan die drei Basisgrößen Windrichtung, Windgeschwindigkeit und Regenmenge durch GPIO-Pins erfasst. Darüber hinaus steht ein I²C-Bus bereit, der unter anderem den BME280 (Temperatur, Feuchtigkeit und Luftdruck) steuert und ausliest. Daneben werden noch folgende Sensoren von der aktuell vorliegenden Software unterstützt:

- VEML6070 (UVA)
- VEML6075 (UVA/UVB)
- TSL2561 (Licht)
- TSL2591 (Licht)
- WSEN-PADS (Luftdruck)

Für die Messung von Feinstaubkonzentrationen kann die Software per UART den SDS011 oder den Honeywell HPMA115SXXX ansprechen.
Weitere Sensoren lassen sich ebenfalls anschließen, brauchen aber dann eine Anpassung in der Firmware.
Beim Booten sucht die Firmware automatisch nach angeschlossenen Sensoren und versucht automatisch den passenden Treiber auszuwählen.

Clouddienste

Clouddienste stellen eine Möglichkeit dar, die Daten der Wetterstation in regelmäßigen Abständen speichern zu lassen, sofern eine Verbindung zu ihnen besteht. Als Dienste werden hier ThingSpeak und openSenseMap unterstützt, sowie eine Anbindung an einen MQTT-Broker für die lokale Cloud. Lässt man einen MQTT-Broker wie Mosquitto auf einem Raspberry Pi laufen, dann wäre Node-RED [4] eine Möglichkeit, die Daten zu verarbeiten und zu speichern.

Für jeden Dienst kann ein individuelles Intervall für den Upload der Daten eingestellt werden, sowie eine individuelle Auswahl, welche Daten an die Dienste übertragen werden sollen.

SD-Karte, Display und Taster

Damit Daten auch ohne aktive WiFi-Verbindung gespeichert werden können, ist in der Wetterstation ein SD-Karten-Steckplatz vorgesehen, in denen sich auch Karten mit mehr als 4 GB Größe verwenden lassen. Die Messwerte aller Sensoren werden in in einem vom Benutzer definierbaren Intervall auf der SD-Karte abgespeichert, und zwar im CSV-Format. Die Daten werden direkt in das Hauptverzeichnis geschrieben, jeweils eine Datei pro Tag wird dort angelegt.

Das Display der Station wird für die Anzeige von Statusmeldungen genutzt (**Bild 2**). Nach dem Booten erscheint hier der WiFi-Status und der Hinweis, ob die SD-Karte durch die Software eingebunden wurde, oder diese sicher entfernt werden kann.
Wenn das Display aktiv ist, sorgt ein Tastendruck für das Auswerfen oder Einbinden der SD-Karte. Wenn die Taste für 30 Sekunden nicht gedrückt ist, wird das Display abgeschaltet, um Energie zu sparen, ein erneuter Druck auf die Taste schaltet das Display wieder an. Neben der Anzeige für den Status der SD-Karte können wir dem Display entnehmen, ob eine Verbindung zu einem WLAN-Netzwerk besteht und wenn, mit welcher Signalstärke. Außerdem wird die IP-Adresse der Station angezeigt.

Eine Besonderheit beim Taster: Er ist an den gleichen Pins angeschlossen wie der Boot-Taster des ESP32. Nach dem Booten steht dieser der Anwendersoftware zur Verfügung (siehe unten).

Bild 2. Das Display zeigt Statusmeldungen an. „NOSD" bedeutet, dass keine Speicherkarte eingelegt ist.

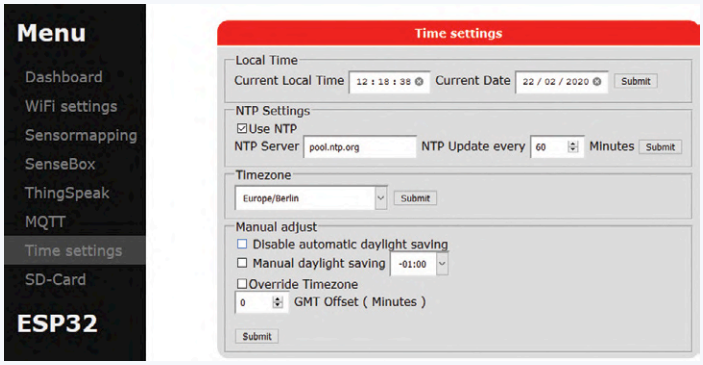


Bild 3. Eines der zentralen Elemente der Software ist der Webserver, der die Webseiten für die Konfiguration der Wetterstation ausliefert.

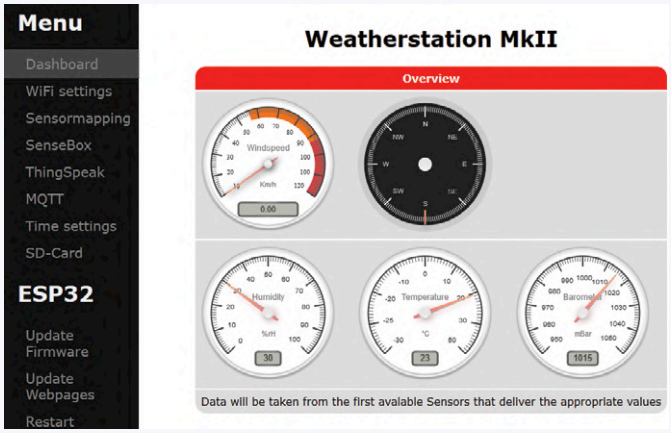


Bild 4. Der Webserver stellt auch die wichtigsten Messwerte bereit, in einem Browser (etwa auf dem Smartphone) werden diese dann als schicke Grafik dargestellt.

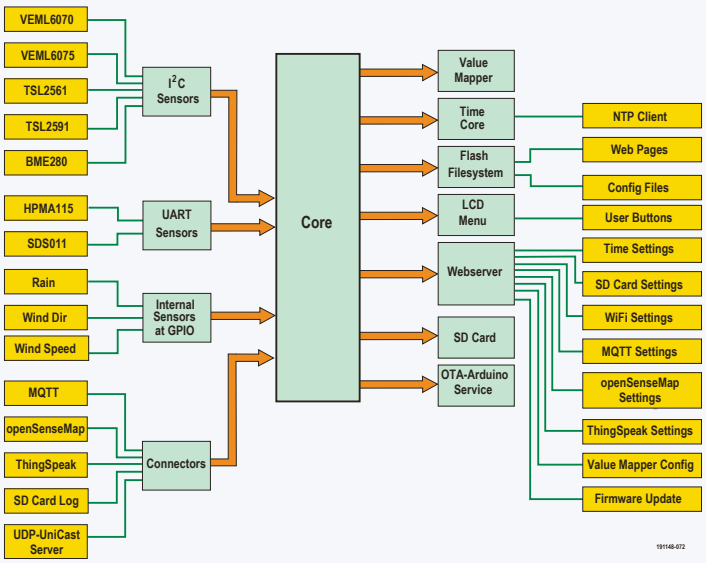


Bild 5. Gesamtüberblick der Software-Module.

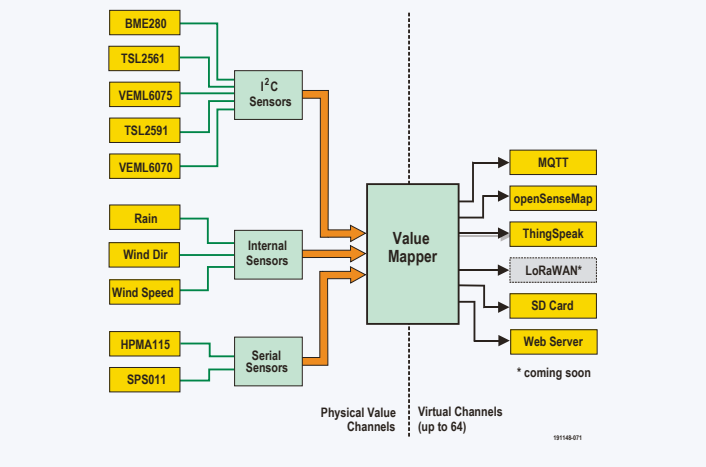


Bild 6. Der Mapper verbindet die Datenquellen (Sensoren) mit den Datensenken (Konnektoren zu den Cloudplattformen und dem Webserver für die Darstellung). Er stellt im System 64 virtuelle Messkanäle bereit.

Webserver und Webseite

Wie bei vielen Elektor-Projekten kommt auch hier ein Webserver mit Konfigurationsmöglichkeit zum Einsatz. In **Bild 3** sieht man zum Beispiel die Einstellung der Zeit und des Datums. Damit die Daten in einem Browser dargestellt werden können, benötigt es mehrere ineinander greifende Komponenten. Unser ESP32-Webserver liefert nicht nur Webseiten aus, die in einem Browser dargestellt werden, er schickt auch Daten on-demand zum Browser. Damit die Daten vom Webbrowser einfach zu parsen sind, wird JSON verwendet. Die Webseiten bestehen hierzu nicht nur aus HTML, sondern auch aus etlichen Zeilen JavaScript.

Damit kann die für die Anzeige nötige Rechenleistung vom ESP32 teilweise in den Browser verlagert werden. Ein Beispiel ist das Dashboard (**Bild 4**), mit dem die Messwerte der Wetterstation in einem Browser der Wahl angezeigt werden können (zum Beispiel auf dem Smartphone). Die Anzeigen werden nicht als Grafik übertragen, sondern im Browser mit Javascript gezeichnet.

Interna der ESP32-Firmware

All die angeschnittenen Funktionen benötigen Software. Daher folgt nun ein Überblick der einzelnen Teile, die am Ende eine Wetterstation ergeben. Eine Übersicht der Software-Module findet man in **Bild 5**.

Zuerst schauen wir uns die Verarbeitung der Messwerte an. Wir haben hier drei Hauptteile: einmal die Treiber zum Ansprechen der Sensoren, dazu die Konnektoren, um Daten der Station an Clouddienste zu senden und auf die SD-Karte zu schreiben, und schließlich als „Vermittler" ein Mapper.

Mapper

Der Mapper stellt im System 64 virtuelle Messkanäle bereit, die von den Konnektoren abgefragt werden können. Ein virtueller Kanal im Mapper kann dem Messwert eines Sensors zugeordnet werden; man beachte, dass Sensoren mitunter mehrere Messwerte ausgeben (z.B. erzeugt der BME280 drei Werte: Temperatur, Luftfeuchtigkeit und Luftdruck) . Wenn nun einer der Konnektoren den Wert für den virtuellen Kanal 0 haben möchte, führt der Mapper gemäß einer internen Tabelle eine Übersetzung durch und holt über den passenden Sensortreiber den entsprechenden Wert. Die Zusammenhänge sind in **Bild 6** veranschaulicht.

Dies mag im ersten Moment etwas kompliziert klingen. Ein sehr großer Vorteil dabei ist aber, das der Konnektor selber nicht wissen

TASK-PROGRAMMIERUNG

Die Konnektoren sind in der Software als eigene Task implementiert, so dass diese unabhängig voneinander Sensordaten auslesen und weiterverarbeiten können. Das verursacht aber auch ein paar Probleme, die zu lösen sind. Eines davon ist ein möglicherweise simultaner Zugriff auf den Mapper und damit die Sensoren auf dem Bus. Als Annahme seien zwei Konnektoren aktiv, Tom und Jerry, die jeweils jede Minute Daten vom Mapper holen und damit letztendlich auf Sensoren zugreifen. So soll Tom als erstes anfangen, Daten einzulesen. Tom muss danach aber eventuell warten bis aktuelle Messwerte bereit stehen oder gewandelt wurden, und so gibt dieser Rechenzeit und damit die CPU ab, so dass ein anderer Task aktiv sein kann. In unserem Beispiel ist das nun Jerry. Jerry muss nun auch Daten vom Mapper holen - und damit gegebenenfalls von den gleichen Sensoren, die an einem Bus angeschlossen sind. Ist der I²C-Bus im Spiel, dann kann es nun sein, dass Tom gerade die Datenwandlung von einem Sensor gestartet hatte und wartet bis der Sensor fertig ist; wenn Jerry nun auch auf dem I²C-Bus anfängt, Daten zu lesen, vor allem vom gleichen Sensor, erzeugen wir ein Durcheinander.

Im Betriebsystem des ESP32, dem FreeRTOS, gibt es zur Lösung dieses Problems mehrere Mechanismen, die genutzt werden können. Die einfachste hier ist ein Mutex. Man kann sich einen Mutex wie einen Schlüssel zu einem abgesperrten Bereich vorstellen, von dem es nur einen gibt. Wer in diesen Bereich will, braucht einen Schlüssel, öffnet den Zugang und betritt den Bereich, der sich nach dem Betreten wieder verschließt. Andere die den Bereich betreten wollen, müssen warten bis der Schlüssel wieder aus dem Bereich vom aktuellen Besitzer hinausgetragen wurde. Erst dann kann der Nächste den Schlüssel erhalten. Bei FreeRTOS muss der Entwickler dafür Sorge tragen, dass erstens der Bereich durch einen Mutex (Schlüssel) geschützt ist und dass nach dem Verlassen des geschützten Bereichs der Schlüssel (also der Mutex) wieder zurückgelegt wird. Passiert dieses nicht, und so etwas unterläuft Entwicklern häufiger als ihnen lieb ist, kommt es zu einem Stillstand im System, da der Schlüssel „verlegt“ wurde.

Damit ist der erste Punkt, der geregelte Zugriff auf die Daten, abgearbeitet. Wir müssen uns aber auch darum kümmern, dass Rechenzeit abgegeben wird, wenn ein Task keine Aufgaben hat. Während bei einer Software ohne Betriebssystem alles in einem Superloop läuft und man hier sich Mechanismen ausdenken muss, wie man die Zeit und Zeiteinteilung sinnig gestaltet, so kann bei einem RTOS ein Task schlafen gelegt werden. In der Zeit sucht sich das Betriebssystem andere Tasks, die nicht schlafen und führt diese aus. Bei den Konnektoren (z.B. bei demjenigen für openSenseMap) wird die Zeit zwischen zwei Sendeintervallen nicht in einer Schleife mit `delay()` Anweisungen verbracht, sondern es wird ein weiterer Mechanismus verwendet, der dem Mutex sehr ähnlich ist, eine binäre Semaphore. Ein Mutex und eine binäre Semaphore haben bei FreeRTOS eine sehr ähnliche Funktion mit kleinen Unterschieden, mehr darüber kann in der FreeRTOS-Artikelserie [5] nachgelesen werden. Innerhalb von FreeRTOS lässt sich die Wartezeit für eine Semaphore angeben; wird während der Wartezeit eine Semaphore frei, so erhält man aus dem Betriebssystem eine passende Rückmeldung. Auf diese Weise lässt sich eine definierte Wartezeit oder ein definiertes Intervall erstellen und automatisch Rechenzeit abgeben. Doch warum eine Semaphore nutzen und nicht einfach dem Betriebsystem mit dem Befehl `vTaskDelay()` sagen, nach welcher Zeitspanne der Task weiter ausgeführt werden soll? Die Antwort liegt in der hier verwendeten Methode, wie dem Task Änderungen der Konfiguration mitgeteilt werden, die der Benutzer durch das Webinterface einstellen kann. Wenn es beim Warten auf eine Semaphore einen Time-out gab, dann wurde die Konfiguration nicht geändert. Wenn das Warten auf eine Semaphore ohne Fehler erfolgt, so liegen neue Einstellungsdaten vor. Die Einstellungen werden dann komplett neu geladen und angewendet, das Warten beginnt danach wieder von vorne.

muss wie der Sensor anzusprechen ist. All dieses wird durch den Mapper verborgen. Falls später ein weiterer Bus oder anderer Sensortyp hinzukommt, müssen nur der Mapper und die Treiber angepasst werden, die Konnektoren selber arbeiten weiter wie gehabt.

Der Mapper ist natürlich auch an den Webserver angebunden. Dieser fragt nicht nur die Messwerte der einzelnen virtuellen Kanäle ab, um selbst Daten darzustellen, sondern benötigt auch Informationen über angeschlossene, unterstützte und fehlende Sensoren.

Jeder Sensorwert hat eine definierte Adresse innerhalb der Software. Diese setzt sich aus dem Bus, dem Wertetyp und einer ID zusammen, um den passenden Sensor für den Zugriff zu identifizieren. Zum Beispiel bedeutet „*BME280.0.Pressure.I2C*“: *BME280* ist der Bezeichner des Sensors und soll den Namen für Menschen besser lesbar machen, *0* ist die ID des Sensorwertes, *Pressure* der Typ des Messwertes. Der Ausdruck *I2C* zeigt an, dass der Sensor an den I²C-Bus des ESP32 angebunden ist. Daraus kann der Mapper entnehmen, dass es sich um ein I²C-Device handelt und nach dem passenden Treiber für den Zugriff gesucht werden muss.

Treiber

Der Treiber erhält vom Mapper die Informationen um welchen Wertetypen es sich handelt und welche ID der Wert hat. Daraus ermittelt dieser, welcher Sensor anzusprechen ist und welcher Wert von diesem Sensor geholt werden muss.

Damit dies zum Beispiel für den I²C-Bus klappt, wird nach dem Boot durch den I²C-Treiber einmal der Bus nach allen bekannten Sensoren abgesucht. Dies kann durch einen *START* auf dem Bus geschehen und das Prüfen, ob ein Device reagiert. Wenn ein Device reagiert, wird dieses als vorhanden markiert und ein *STOP* gesendet. Diese Sequenz wird für alle bekannten Sensoren wiederholt. Am Ende entsteht so eine Liste dessen, was am Bus angeschlossen und aktiv ist. Mit dieser Liste kann der Treiber entscheiden ob bei einem Zugriff auf einen Sensor direkt ein Fehler (nicht vorhandener Sensor) zurückgegeben wird, oder aber, ob ein Zugriffsversuch für den Sensor erfolgt. Wenn der Wert erfolgreich vom Sensor gelesen wurde, wird er an den Mapper zurückgegeben und ansonsten ein passender Fehler.

Konnektoren

Ein Konnektor ist das Bindeglied zwischen den Messwerten und einem Ort an dem Daten abgelegt werden sollen (Datensenke). Die Konnektoren sind innerhalb der ESP32-Firm-

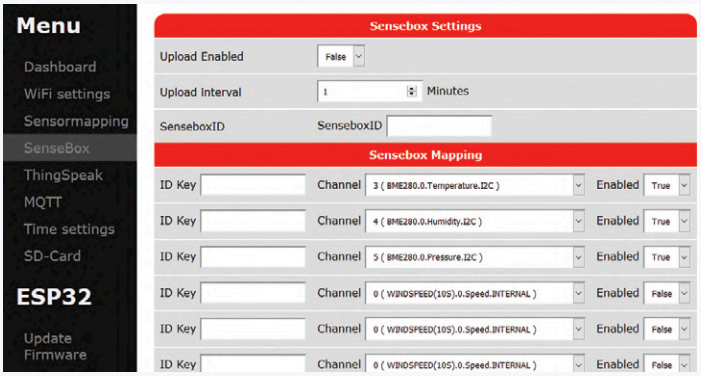


Bild 7. Konfiguration des Zugangs zu openSenseMap, …

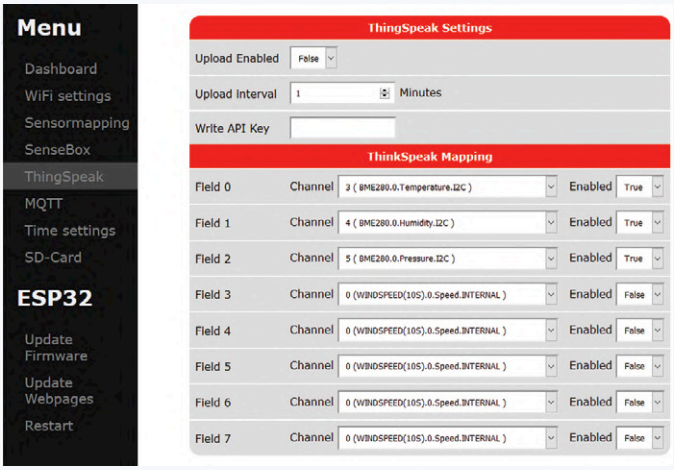


Bild 8. … zu ThingSpeak und …

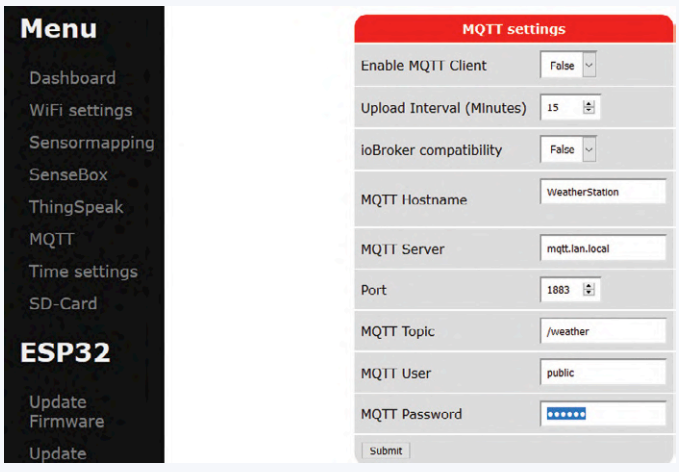


Bild 9. … zum MQTT-Broker. Man sieht dass jede Konfigurationsseite andere Kontrollelemente aufweist.

ware als eigene Task gestaltet, so das diese unabhängig voneinander Sensordaten auslesen und weiterverarbeiten können. Grundsätzlich verfügt jeder Konnektor über eine Liste an virtuellen Kanälen, die ausgelesen werden müssen, und eine Logik, die dafür sorgt, dass die Werte, z.B. für ThingSpeak, passend aufbereitet werden.
Durch die Task-Programmierung sind die Konnektoren zwar voneinander unabhängig, verursachen aber auch ein paar Probleme, die zu lösen sind (siehe **Kasten „Task-Programmierung"**).

Webinterface

Nachdem wir nun wissen, wie Daten von den Sensoren zu den Clouddiensten gelangen, kommen wir nun zum Benutzer-Interface der Wetterstation. Das Webinterface selbst ist in mehrere Teile zergliedert. Für den Mapper und jeden Konnektor gibt es einen eigenen modularen Teil im Webinterface, so dass neue Konnektoren recht einfach an den bestehenden Webserver angeschlossen werden können. Der Webserver stellt die Webseiten und Daten bereit, die der Browser braucht, um das Webinterface rendern zu können. Die Dateien hierfür befinden sich in einem Teil des ESP32-Flash, dem SPIFFS. Die meisten Leser werden diesen Speicher in eigenen Programmen bisher nur lesend genutzt haben, um Webseiten durch einen ESP32 ausliefern zu lassen. Auf das SPIFFS können jedoch, wie auf eine SD-Karte, auch Dateien geschrieben werden, die der Webserver direkt ausliefern kann, z.B. eine JSON-Datei. Jedoch hat diese Methode einen Nachteil. Log-Files sollten nicht in das SPIFFS geschrieben werden, da Flash-Speicher nur eine begrenzte Lebensdauer hat und nicht so einfach getauscht werden kann wie eine SD-Karte.
Die Firmware nutzt einen Teil des SPIFFS für einen Teil der Konfigurationsfiles, primär für die Konnektoren. In **Bild 7, 8** und **9** sieht man zum Beispiel jene Webseiten, mit denen der User die Konnektoren zu den Cloudplattformen ThingSpeak und openSenseMap sowie den MQTT-Client konfigurieren kann. Man erkennt, dass die Seiten verschiedene Kontrollelemente aufweisen. Die Seiten müssen je nach Konnektor also dynamisch generiert werden. Einen solchen Mechanismus haben wir im Elektor-Labor auch schon für andere ESP32-Webserver verwendet. Je nach URL, die der Webbrowser zum Webserver sendet, führt dieser eine eigene Funktion aus. Diese wird verwendet, um Konfigurationen und Parameter aus dem System an den Webserver zu übertragen.

ESP32 UND JAVASCRIPT

Moderne Browser fangen von Haus aus an, mehrere Dateien gleichzeitig zu laden, je nach Browser können es 4, 6, 8 oder noch mehr sein. Damit der Anwender nicht warten muss, bis alle Dateien geladen sind, beginnt der Browser schon mal, Skripte auszuführen, zum Beispiel für die Anzeige der Werte, Tabellen und Anzeigeelemente der Seite. Sollten nun die Skripte in mehreren Dateien verteilt sein, um den Code etwas übersichtlicher zu gestalten, so kann es sein, dass bei der Ausführung Teile des Codes noch nicht geladen sind und die Ausführung mit einem Fehler abbricht. Durch ein Skript wurde daher aller Code in eine Datei gepackt, um so das Problem mit teilweise geladenem Code zu umgehen. Dieses ist auch erst auf dem ESP32 aufgetreten, da der Webserver auf dem Entwicklungssystem die Daten schnell genug ausliefern konnte.
Die andere Hürde war das Laden mehrerer Dateien innerhalb von JavaScript. Diese erfolgen nebenläufig zur Skriptausführung und melden durch einen Callback, ob die Datei geladen wurde oder nicht. Werden nun an einer Stelle mehrere Daten benötigt, müsste dafür das Laden verkettet werden (also Datei1 laden lassen und ein Callback zum Laden für Datei2 mitgeben, in dem dann Datei3 geladen wird). Um das zu umgehen wurde ein kleiner Hilfsteil geschrieben der ein Array an URLs und Speicher für die Antworten bereitstellt und diese Aufgabe für uns automatisiert. Innerhalb der Datei *basic.js* sieht man, wie die Funktion `loadMultipleData` arbeitet.
Das Zusammenspiel zwischen Webinterface, Webserver und Firmware ist etwas, das bei eigenen Projekten nicht unterschätzt werden sollte, Stolpersteine lauern nicht nur in der Firmware, sondern auch in den Browsern, die auf den unterschiedlichen Geräten laufen. Jeder Browser verhält sich etwas anders und braucht eventuell angepasste Skripte, damit die Seite auch sauber angezeigt wird.

JavaScript

Doch am Webinterface ist nicht nur der Webserver beteilig, sondern auch der Webbrowser. Es kann Arbeit vom Webserver in den Browser verlagert werden, was den ESP32 entlastet. Unsere Webseite macht davon Gebrauch, in dem z.B. die Rundanzeigen des Dashboards nicht als Grafik übertragen werden. Vielmehr werden diese im Browser durch Skripte (JavaScript) zur Laufzeit gezeichnet. Auch das Aufbereiten von Daten und Tabellen wird nicht mehr im ESP32 durchgeführt sondern direkt durch die Skripte in der Webseite. Damit das möglich ist muss wiederum vom ESP32 ein Weg geschaffen werden, dass die Webseite im Browser auf Daten zugreifen kann. An diesem Punkt kommen die dynamisch generierten Dateien zum Zuge.

Die folgenden dynamischen Dateien können vom Webbrowser angefordert werden:

/setWiFiSettings
/getSSIDList
/mapping/mappingdata.json
/devices/supportedsensors.json
/devices/connectedsensors.json
/mapping/{}/value
/mqtt/settings
/sdlog/settings.json
/sdlog/sd/status
/sensebox/settings.json
/sensebox/mapping/{}
/sensebox/mapping.json
/thingspeak/settings.json
/thingspeak/mapping/{}
/thingspeak/mapping.json
/timesettings

Als Antwort wird immer eine JSON-Datei geliefert, so dass das im Browser ausgeführte JavaScript diese sehr einfach parsen und weiterverarbeiten kann.

Der Ausdruck *{}* wird als Parameter für eine dynamische Funktion verwendet. */mapping/0/value* liefert z.B. den Sensorwert des ersten logischen Kanals zurück, */mapping/1/value* den Wert des zweiten logischen Kanals. */sensebox/mapping/0* gibt den ersten logischen Kanal zurück, dessen Wert zu openSenseMap übertragen wird . Ein Blick in den Quelltext zeigt, wie das Ganze umgesetzt wurde. Zuerst einmal der Befehl, der die Funktion im Webserver zuweist:

```
server->on("/sensebox/
   mapping/{}", HTTP_
   GET, GetSenseboxChMapping );
```

Damit wird dem Webserver mitgeteilt, dass bei einem Zugriff auf *sensebox/mapping/* der letzte Teil der URL als Parameter zu sehen ist und die Funktion `GetSenseboxChMapping` aufgerufen werden muss. In der Funktion selbst wird dann mit `String IDs = server->pathArg(0);` der Parameter als String weiterverarbeitet.
Ein Problem, das bei der Kombination ESP32 und JavaScript auftreten kann, zeigen wir im **Kasten „ESP32 und JavaScript"**.

Start des Systems

Ein LCD für die Anzeige von Daten ist manchmal sehr hilfreich, man braucht aber zum Bedienen noch ein paar Tasten. In diesem Fall beschränkt sich die Menge der Tasten auf eine, was die Menge an Löchern im Gehäuse der Wetterstation verringert. Daher hier eine kleine Übersicht über die Anzeige des Displays und die Funktion der Boot-Taste. Durch Drücken des Boot-Pins kann der ESP32 beim Start in den Bootloader-Modus versetzt werden, danach steht dieser der Software zur Verfügung.
Nach dem Starten der Firmware und dem Drücken des Tasters wechselt die Station in den AP-Modus (im Display wird ein Hinweis eingeblendet). Es erscheint ein neues WLAN-Netzwerk mit dem Namen *ESP32-xx-xx-xx*, das mit einem Gerät verbunden werden kann. Ist die Verbindung erfolgreich, so kann unter der URL *http://192.168.4.1* die Konfigurationsseite der Wetterstation aufgerufen werden. Das Layout der Seite sollte vertraut sein, da es schon in anderen Projekten verwendet worden ist. Sobald das Wifi-Netz des eigenen Routers und die Zugangsdaten eingegeben worden sind, startet der ESP32 neu und sollte sich mit dem WiFi-Netz verbinden. Damit eingeschätzt werden kann, wie gut die Verbindung ist, wird zusätzlich der RSSI-Wert ausgegeben.
Das Display wird mit der gleichen Taste bedient. Um Energie zu sparen, wird nach 30 Sekunden das Display abgeschaltet. Um dieses wieder zu aktivieren muss die Taste einmal kurz gedrückt werden, und die Hintergrundbeleuchtung und die Statusinformationen werden wieder angezeigt. Ist das Display nun aktiv, kann durch erneutes Drücken der Taste die SD-Karte sicher entfernt oder eingebunden werden, je nach aktuell angezeigtem Zustand. In der ersten Zeile des Displays ist ein „_SD_" zu sehen wenn die SD-Karte ins System eingebunden ist, oder ein „NOSD" wenn keine Karte eingebunden ist oder erkannt wurde.
In der Software musste noch das Entprellen

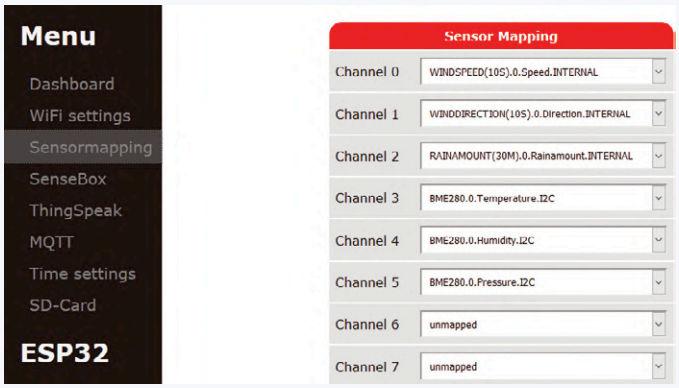


Bild 10. Zuordnung der Sensorwerte zu den virtuellen Kanälen.

der Taste gelöst werden. Hier erzeugt die Taste einen Interrupt, und zwar bei steigender und fallender Flanke den gleichen (bedingt durch die Architektur des ESP32 und des GPIO-Controllers). Das bedeutet aber auch, dass innerhalb des Interrupts ermittelt werden muss, ob es sich um eine steigende oder fallende Flanke gehandelt hat. Das Entprellen selbst erfolgt über das Messen der Zeit zwischen dem Drücken der Taste und dem Loslassen; ist diese unter 100 ms, wird der Tastendruck verworfen. Das Signalisieren eines Tastendrucks wird aus dem Interrupt heraus durch eine Semaphore realisiert, so dass ein eigener Task diese abfragen kann.
Ähnlich ist auch die Anzeige des LCD realisiert. Ein eigener Task aktualisiert jede Sekunde das Display für 30 Sekunden, legt sich nach 30 Sekunden schlafen und wartet dann auf einen Tastendruck.

Einstellungen

Nach der Installation der Firmware erstellt die Wetterstation eine Basiskonfiguration, die von einem angeschlossenen Regenmesser, Windmesser und einem BME280 ausgeht. Ist kein BME280 angeschlossen, ist das kein Problem, die entsprechenden Kanäle geben dann keinen Wert aus.

Beim Setup müssen zuerst die virtuellen Kanäle eingestellt werden, eine Zuordnung eines Sensorwertes zu einem virtuellen Kanal (siehe **Bild 10**).
Diese Kanäle werden später durch die Konnektoren abgefragt, die die Anbindung an Datensenken wie openSenseMap, ThingSpeak oder die SD-Karte verwirklichen.
Die Einstellungen selber haben auch Einfluss auf das Dashboard. Dieses verwendet aus den virtuellen Kanälen die Werte für Windrichtung, -Geschwindigkeit, Temperatur, Luftfeuchtigkeit und Luftdruck, und zwar jeweils den ersten dieser Werte, die in den virtuellen Kanälen zu finden sind.
Die Konnektoren sind nicht nur das Bindeglied zwischen der Wetterstation und den Cloud-Diensten. Auch das Speichern auf

WEBLINKS

[1] **Open-Network-Wetterstation Mark 2 (Teil 1), Elektor Mai/Juni 2020**: www.elektormagazine.de/191148-03
[2] **Elektor-Labs-Webseite zum Projekt**: www.elektormagazine.com/labs/remake-elektor-weather-station
[3] **Projektseite zu diesem Beitrag**: www.elektormagazine.de/191148-B-01
[4] **Einführung in Node-RED**: www.elektormagazine.de/articles/einstieg-in-nodered
[5] **Artikelserie FreeRTOS auf dem ESP32**: www.elektormagazine.de/magazine/elektor-138/56969

die SD-Karte wird durch einen Konnektor geregelt. Für die SD-Karte selber sind die Einstellungen recht einfach: das Intervall in dem die Daten geschrieben werden sollen und ob das Schreiben überhaupt aktiv sein soll. Auf der Webseite kann auch die SD-Karte sicher aus dem System entfernt werden (oder eingebunden). Als Info wird die Größe und der verwendete Platz der SD-Karte ausgegeben. Die Ablage der Daten selbst erfolgt als CSV-Datei im Wurzelverzeichnis der SD-Karte, für jeden Tag in einer eigenen Datei. Damit die Daten mit der korrekten Zeit versehen werden, muss diese unter dem Punkt *Time settings* eingestellt werden (Bild 2).

Für die Cloud-Dienste sind etwas mehr Einstellungen möglich und nötig. Für SenseBox und damit openSenseMap werden die *Sensebox ID* und für jeden Sensor noch einmal ein eigener Key benötigt (Bild 6). Diese können einfach aus dem Webinterface von openSenseMap entnommen werden. Jedem dieser Sensor-Keys kann dann ein virtueller Kanal zugewiesen werden. Insgesamt lassen sich bis zu 16 Werte an openSenseMap übertragen.
Für ThingSpeak (Bild 7) sind die Einstellungen ähnlich, wobei hier nur ein *API Key* für das Schreiben notwendig ist und die zu übertragenden Felder wieder den virtuellen Kanälen zugewiesen werden müssen.

MQTT bietet hier deutlich weniger Einstellungen; man braucht aber für das Weiterverarbeiten der Daten mehr Informationen für den Benutzer (Bild 8). Neben dem Intervall mit dem die Daten gesendet werden (und der Entscheidung ob Daten gesendet werden), gibt es noch den Punkt *ioBroker compatibility*. Wer die Daten z.B. mit Node-RED verarbeitet braucht diesen Modus nicht zu nutzen. In diesem Fall werden die Daten so per MQTT transportiert dass der Broker die Daten daraus wieder entnehmen kann. Nachfolgend wird von einem generischen Transport per MQTT ausgegangen, an dessen Ende sich ein System befindet, das einen JSON-String verarbeiten kann, wie Node-RED [4].

```
// the JSON produced by the
   Station looks like this:

{
 Data:[
   {channel:0, value:0.0,
   Name:"WINDSPEED(10S).0.Speed.
   INTERNAL"},
    …
   {channel:63, value:0.0,
   Name:"WINDSPEED(6d0S).0.Speed.
   INTERNAL"}
       ]
}
```

Innerhalb des Arrays `Data` werden nur Kanäle übertragen, die auch einem Sensorwert zugewiesen sind. Kanäle die keine Zuweisung haben, tauchen nicht auf.

Mittelwerte

Bei der Ermittlung der drei Messwerte von Regenmenge, Windrichtung und Windgeschwindigkeit werden Mittelwerte gebildet; bei der Windrichtung und Windgeschwindigkeit über 10 Sekunden, 60 Sekunden und eine Stunde. Bei der Regenmenge werden die Werte für 30 Minuten, 60 Minuten, 720 Minuten und einen Tag (1440 Minuten) ausgegeben. Bei diesen Werten handelt es sich um laufenden Mittelwerte über die vergangene Zeit. Diese Funktion ist aktuell noch in den Treibern der Werte fest hinterlegt, könnte aber durch den Kern selber später in einen „Recorder" verlagert werden, der Daten als Konnektor entgegennimmt und diese Daten wiederum in virtuellen Kanälen bereitstellt. Mit diesem Prinzip ließen sich auch andere mathematische Funktionen verwirklichen.

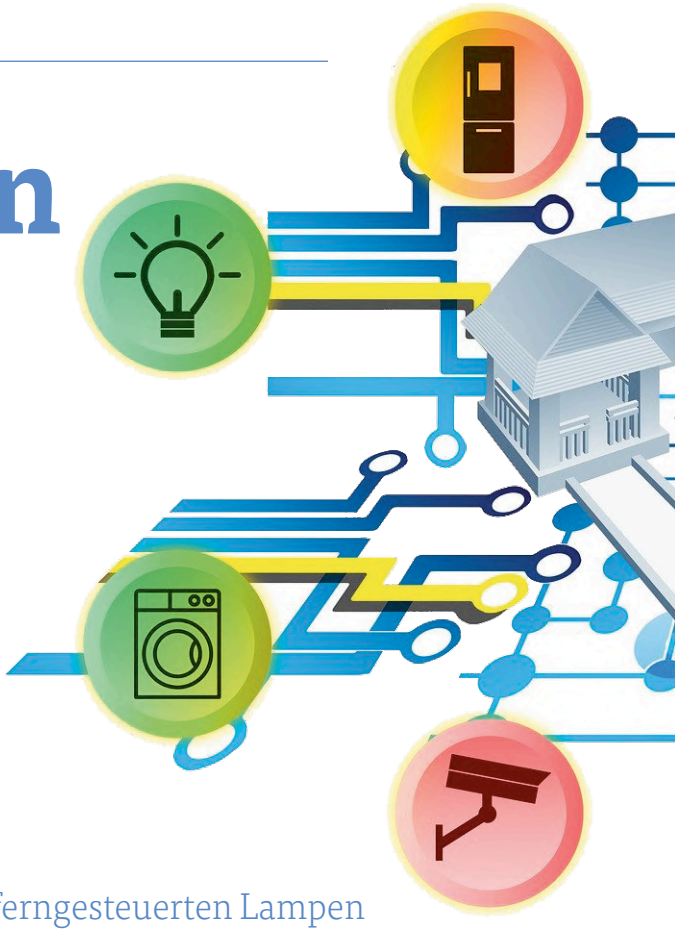
Raum für Erweiterungen

Auch wenn die Wetterstation als solche einsatzbereit ist, wird sicher noch der eine oder andere Wunsch aufkommen, die Funktionalität zu erweitern. So gibt es auch in unserem Labor schon eine paar Ideen, wie die Unterstützung von One-Wire-Sensoren. Interessant wäre ein Filebrowser für das Webinterface, um direkt Daten von der SD-Karte zu löschen oder herunterzuladen. Wir freuen uns über weitere Anregungen, die Sie gerne als Kommentar auf der Elektor-Labs-Seite zu diesem Projekt [2] hinterlassen können!

191148-B-01

Hausautomation leicht gemacht

Mit ESPHome, Home Assistant und MySensors



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

Wer hat noch nie davon geträumt, seine Wohnung mit ferngesteuerten Lampen oder mit Fensterläden, die sich automatisch öffnen und schließen, auszustatten? Die Zauberformel lautet Hausautomation!

Müssen Träume Schäume bleiben?

Ich hatte sicherlich viele Träume, aber oft sind es Träume geblieben. Sobald ich anfing, einen Traum zu verwirklichen, stieß ich unweigerlich auf jene Hürden in der Praxis, die meine Motivation anscheinend nicht überwinden konnte.

Der Bau eines drahtlosen, batteriebetriebenen Temperatursensors zur Messung der Temperatur im Wohnzimmer ist nicht sonderlich kompliziert, und auch das Hinzufügen eines ferngesteuerten Relais zum Ein- und Ausschalten einer Heizung ist machbar. Aber da fängt die Komplexität erst an. Es wird eine Art Regler benötigt, der die Erstellung von Grundregeln wie *„An Wochentagen die Heizung morgens um 7 Uhr einschalten; am Wochenende erst um 9 Uhr"* ermöglicht. Und es muss eine Art manuelles Überschreiben möglich sein, so dass jeder berechtigte Hausbewohner die Temperatur einstellen kann, ohne vorher Python lernen zu müssen. Bis zu diesem Punkt können wir noch nicht einmal von Hausautomation sprechen, da nur ein etwas ausgefallenerer, programmierbarer Thermostat beschrieben wurde. Es ist weit mehr nötig, um aus einer ferngesteuerten Lampe ein vollwertiges, erweiterbares Hausautomatisierungssystem zu machen. Aus diesem Grund bin ich nie über meine Träume hinausgekommen.

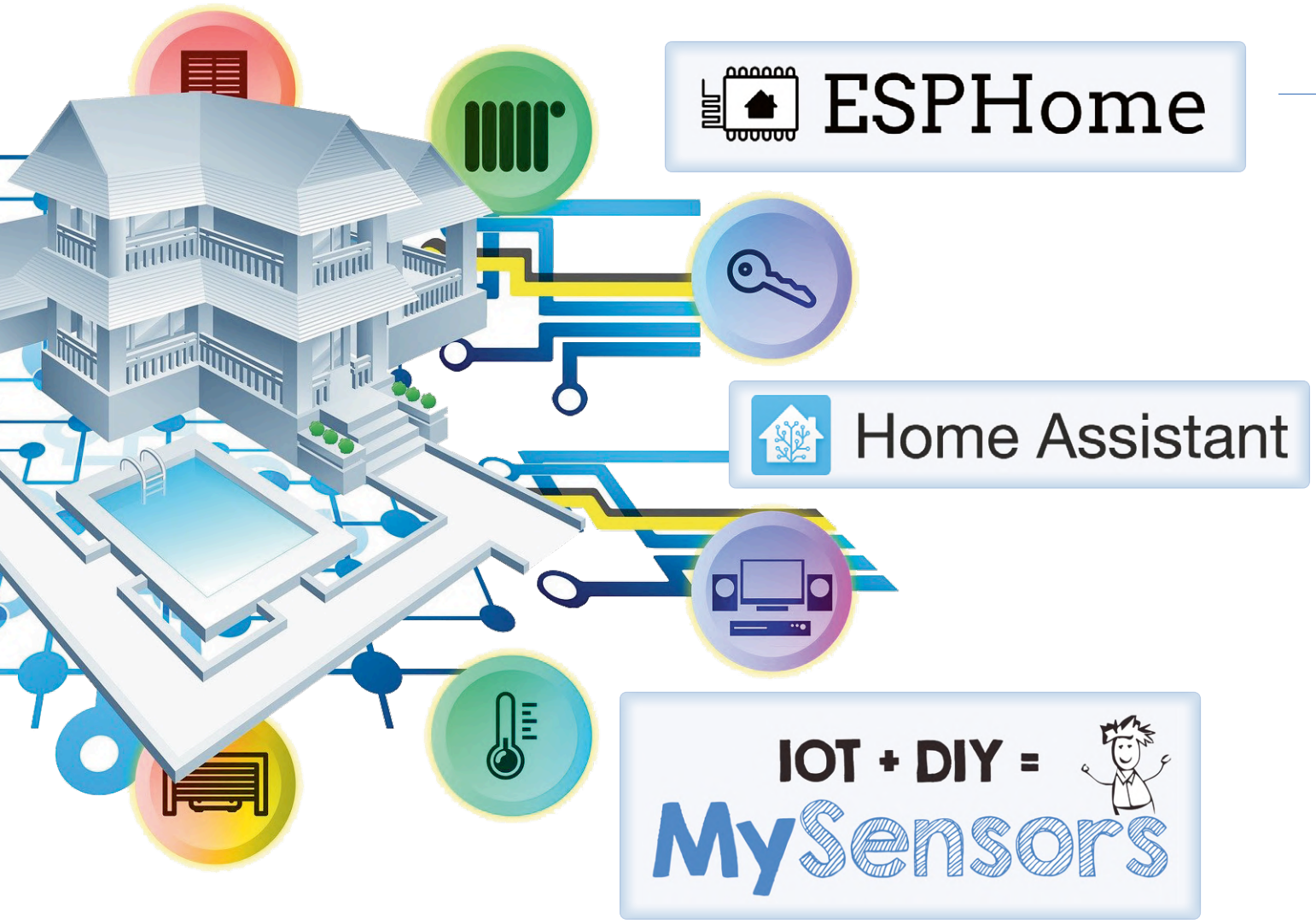
Espurna und ESPHome

Bis vor Kurzem! Ich stieß bei der Online-Suche nach dem Handbuch eines WLAN-gesteuerten Netzsteckers, den ich vor ein paar Jahren gekauft, aber nie benutzt habe, auf das Espurna-Projekt [1]. Auch wenn ich es am Ende nicht genutzt habe, erwähne ich es hier, da es ein ausgezeichnetes Open-Source-Projekt ist, das es verdient, bekannter zu werden. Espurna ermöglicht die einfache Programmierung von WLAN-basierten Sensoren und Controllern (hauptsächlich ESP8266), die miteinander kommunizieren können. ESPHome [2] (**Bild 1**) ist ein Projekt, das Espurna sehr ähnlich ist.

Und Sie dachten, nur Arduino sei einfach?

Espurna und ESPHome bieten beide einfache Möglichkeiten, ESP8266- und ESP32-Boards von Espressif zu programmieren. Tatsächlich

Bild 1. Die recht umfangreiche ESPHome-Homepage zeigt alle vorgefertigten Komponenten, die Teil eines ESPHome-Geräts sein können.



ist dies so einfach, dass sie den Arduino wie Raketenwissenschaft aussehen lassen. Nach der Installation der Software schreiben Sie – kurz gesagt – eine einfache Textdatei, die angibt, welche Art von Sensor an welche(n) Pin(s) Ihres ESP-Moduls angeschlossen ist. Nach dem Kompilieren und Flashen der Firmware erhalten Sie ein intelligentes Gerät mit einer Webschnittstelle, Over-the-Air (OTA)-Programmierung, MQTT-Kommunikation und was nicht alles. Beeindruckend, nicht wahr? Ich denke schon, aber das ist noch nicht alles.

Übernehmen Sie die Kontrolle über Ihre WLAN-Smart-Stecker!

Das Internet quillt über von billigen intelligenten Steckern und Relaiskarten mit integriertem WLAN und anderen WLAN-verbundenen Geräten. Viele davon sind mit einem ESP8266 aufgebaut (**Bild 2**). Gemeinsam ist ihnen aber auch, dass sie einen Zugang zu einem Cloud-Service irgendwo im Internet benötigen, der über (s)eine App auf Ihrem Smartphone gesteuert wird. Dies macht die Nutzung sehr umständlich, so dass diese Geräte oft in einer Schublade oder, noch schlimmer, im Elektroschrott landen.

Das muss nicht sein, denn Espurna und ESPHome erlauben es Ihnen, diese Dinger mit einer Firmware umzuprogrammieren, die Sie nach Belieben verändern können. Keine Internetverbindung mehr, Schluss mit miserablen englischsprachigen Apps und dubiosen Cloud-Diensten! Einfach das Gerät neu flashen und in ihm ein von *you@home* gesteuertes Heimautomatisierungssystem integrieren.

Von der Fernsteuerung zur Automatisierung

ESPHome und Espurna ermöglichen eine Automatisierung, die es den Benutzern ermöglicht, Regeln für das Ein- und Ausschalten von Geräten auf Grund von Ereignissen und/oder Sensordaten festzulegen. Damit kann man ein ziemlich ausgefallenes Heimautomatisierungssystem aus handelsüblicher, billiger WLAN-fähiger Hardware oder selbst gebauten Geräten erstellen.

Dieser Artikel würde hier enden, wenn ich den Einstieg in die Automa-

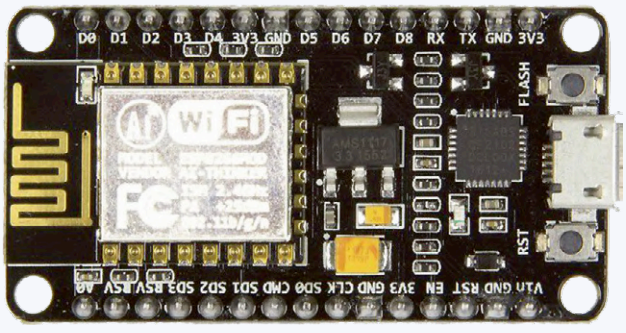


Bild 2. ESPHome läuft auf ESP8266-basierten Modulen wie NodeMCU, das weit verbreitet und kostengünstig ist. Es können auch ESP32-Module verwendet werden.

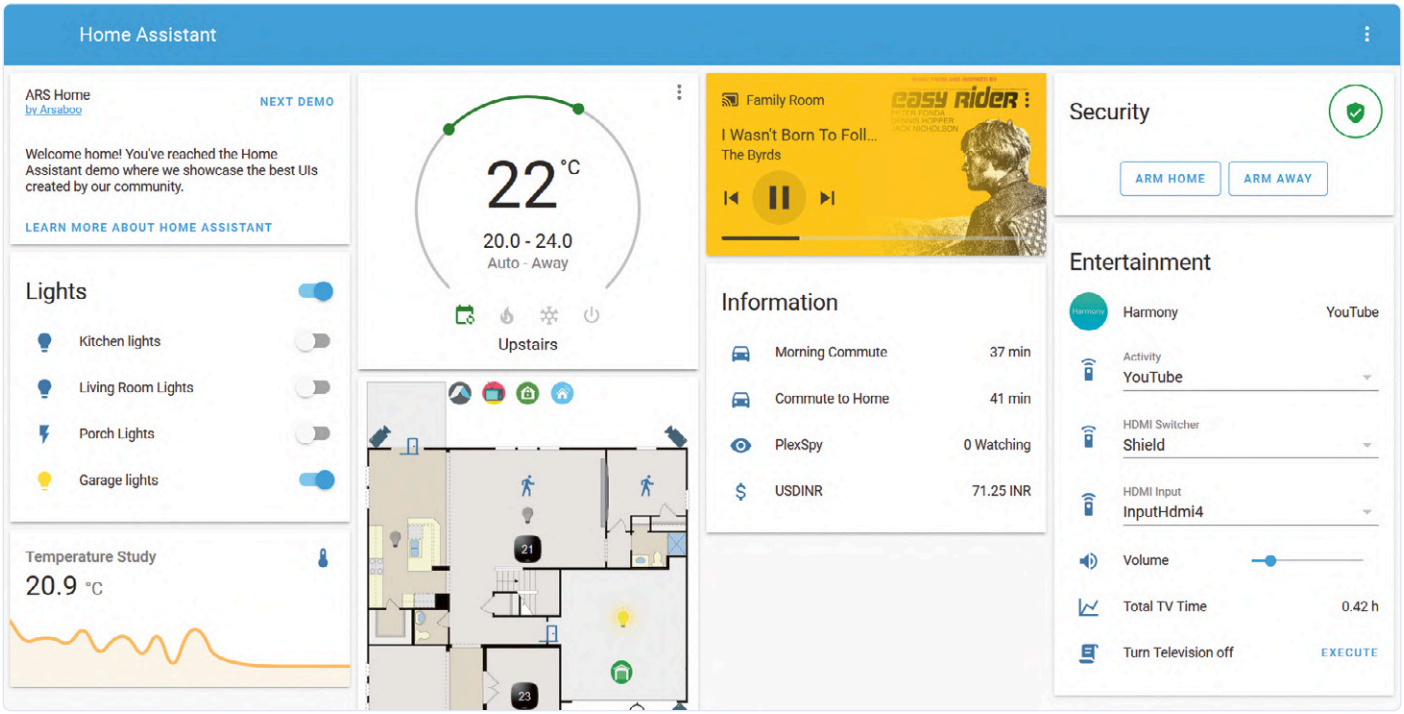


Bild 3. Das Dashboard des Home Assistant ist vollständig anpassbar und kann so ausgefallen oder so spartanisch sein, wie Sie es möchten.

tisierung mit Espurna und ESPHome nicht für zu kompliziert halten würde. Und der Mangel an guter Dokumentation für diese Projekte ist ebenfalls nicht hilfreich. Es scheint zwar ziemlich viel davon zu geben, aber ziemlich verstreut im Netz und dann auch noch wenig deutlich.

Hausassistent

Sowohl ESPHome als auch Espurna ermöglichen eine Integration von Home Assistant [3], einem so genannten *Home Automation Hub* oder *Controller*. Dabei handelt es sich um ein Mittel, mit dem der Benutzer Sensoren und Aktoren von verschiedenen Herstellern in einem einzigen System kombinieren kann. Dabei sind „Sensoren und Aktoren" in einem recht weiten Sinne zu verstehen, denn sie umfassen auch Geräte wie Thermometer über GPS und Internetdienste bis hin zu Motorsteuerung und Textverarbeitung. Home Assistant (auch HA, Hass oder Hass.io genannt) ist kompatibel mit - um nur einige bekannte Namen zu nennen - Alexa und OK Google, Ikeas Trådfri, Philips' Hue, Z-Wave und Zigbee Home Automation sowie Sonoff von iTead [4]. Als dieser Artikel entstand, listete HA 1574 Integrationen auf und geht weit über die Möglichkeiten von Espurna und ESPHome hinaus.

Mit HA können die Regeln für die Geräte in und um Ihr Haus herum definiert werden. *„Wenn die Sonne in 20 Minuten untergeht und das Smartphone von Bewohner A im Haus ist und es seit drei Wochen nicht mehr geregnet hat, dann schalten Sie den dritten Rasensprenger von rechts ein".* Dies ist eine (unsinnige, aber auch) typische (und sogar grundlegende) HA-Automatisierung und funktioniert, wenn denn die betreffende Hardware installiert ist. HA bietet auch eine vollständig anpassbare grafische Benutzeroberfläche (GUI) oder ein Dashboard für das System (**Bild 3**).

HA ist Open Source, in Python geschrieben und läuft auf einem Raspberry Pi. Es funktioniert auch auf anderen Betriebssystemen, ich habe es halt nur auf einem Raspberry Pi 3 installiert, weil das so einfach war. Sie haben es inzwischen gemerkt, ich liebe die Einfachheit.

Integration in den Home Assistant

Home Assistant war auch der Grund, warum ich bei ESPHome statt bei Espurna geblieben bin. ESPHome hat ein Plugin (**Bild 4**), mit dem es so in den Home Assistant integriert werden kann, dass ESPHome-basierte Geräte automatisch von HA erkannt werden. Flash'n'Play, was wollen Sie mehr? Die Integration ist so weit fortgeschritten, dass Sie nicht einmal mehr einen Entwicklungscomputer benötigen. Sensoren und Aktoren können von Ihrem Smartphone aus (neu) programmiert und (neu) konfiguriert werden, während Sie bequem in Ihrem Sessel sitzen.

IoT-Geräte mit geringem Stromverbrauch

WLAN ist großartig, um Geräte schnell an ein Netzwerk anzuschließen, aber es ist ein bisschen, nein, ziemlich stromhungrig. Daher ist es keine Lösung für Sensorknoten mit niedrigem Stromverbrauch, die viele Jahre lang mit einer einzigen Knopfzelle betrieben werden müssen oder vom Energy Harvesting leben. Solche Geräte verbringen die meiste Zeit im Schlaf, und wenn sie wieder aufwachen, spucken sie ihre Daten so schnell wie möglich aus, da sie nicht die Energie für lange Handshake-Protokolle haben.

Eine großartige Lösung für diese Art von Geräten ist MySensors [5], ein Open-Source-Heimautomations- und IoT-Projekt, das auf Funkgeräten für das ISM-Band basiert, insbesondere dem nRF24 von Nordic Semiconductor (**Bild 5**) und dem RFM69 von HopeRF. Die neuere nRF5-Plattform, wie sie auf dem BBC micro:bit zu finden ist, kann ebenfalls verwendet werden.

Die MySensors-Website ist ein wenig unordentlich, aber wenn Sie da erst einmal durchblicken, werden Sie feststellen, dass es ziemlich gutes Material gibt.

MySensors verwendet hauptsächlich (aber nicht nur) Arduino als Mikrocontroller-Plattform und erstellt und und unterhält ein Baum- (oder Stern-) Netzwerk ganz von selbst. Wie ESPHome integriert es sich nahtlos, aber nicht ganz so nahtlos wie ESPHome in den Home Assistant.

Das Projekt besteht aus einer Arduino-Bibliothek, die im Bibliotheksmanager der Arduino-IDE enthalten ist. Nachdem Sie sie installiert haben, können Sie Ihre Anwendung auf einem der Beispiele aufbauen. In vielen Fällen bedeutet dies nur die Änderung der Pin-Nummer(n) der angeschlossenen Peripheriegeräte.

Nun aber in die Hände gespuckt!

Nach dieser langen Einführung möchten Sie vielleicht etwas Praktisches tun, und dafür schlage ich als erstes die Einrichtung eines Home Assistant auf einem Raspberry Pi vor. Eine gute Schritt-für-Schritt-Anleitung finden Sie unter [3] unter dem Tab *Getting Started*. Dort wird ein Rpi 4 vorgeschlagen, aber bei mir läuft HA klaglos auf einem Rpi 3. Eine microSD-Karte mit 32 GB oder größer wird empfohlen. Beachten Sie bitte, dass der RPi nicht mit exFAT-formatierten SDXC-Karten (das heißt, SD-Karten größer als 32 GB) umgehen kann. Wenn Sie sich also für eine 64-GB-microSD-Karte (oder noch größer) entscheiden, stellen Sie sicher, dass diese als FAT32 (neu) formatiert ist. Damit das System robust ist, sollten Sie in Erwägung ziehen, statt der empfindlichen microSD-Karte eine SSD zu verwenden.

Multicast-DNS

Home Assistant verlässt sich auf das Multicast-DNS-Protokoll (mDNS), um Geräte zu finden und mit ihnen zu kommunizieren, aber dieses Protokoll wird von Android und Windows nicht besonders gut unterstützt (siehe Kasten). Apple-Geräte und Raspberry Pis funktionieren gut, und ich nehme an, dass Computer mit einem anderen Linux auch funktionieren. Aus diesem Grund sollten Sie vielleicht eine statische IP für HA wählen. Mein HA-System verwendet DHCP, und ich habe Verbindungsprobleme mit der HA-Anwendung für Android erlebt, als der HA-Computer vom DHCP-Server eine neue IP erhielt.
Beachten Sie, dass das HA-Installationsprogramm schnell einen Webserver startet, auf dem Sie den Installationsfortschritt überwachen können. Das Anschließen eines Displays an den Pi ist hier nicht sinnvoll, nur um die IP-Adresse dieses Servers herauszufinden. Schauen Sie stattdessen in der GUI des Routers Ihres Netzwerks nach.

Zugriff auf die Konfigurationsdatei

Der zweite Installationsschritt ist die Konfiguration des Home Assistant. Hierfür gibt es einen Assistenten, daher brauche ich hier nicht ins Detail gehen. Wichtiger ist die Installation einiger Add-ons nach der Konfiguration von HA. Dies geschieht über das Menü *Supervisor* auf der Registerkarte *Add-on Store*. Aus irgendeinem Grund erlaubt Ihnen die Standard-Konfiguration von HA nicht, die Hauptkonfigurationsdatei (*configuration.yaml*) zu bearbeiten, dennoch werden Sie dies regelmäßig tun müssen, besonders wenn Sie mit dem System experimentieren. Deshalb habe ich das Add-on *File Editor* (früher bekannt als *Configurator*) und auch *Samba share* installiert. Ersteres erlaubt es Ihnen, Low-Level-Dateien direkt in HA zu bearbeiten, das zweite stellt einige HA-Ordner im Netzwerk zur Verfügung, so dass Sie zum Beispiel Dateien mit Ihrem Lieblings-Texteditor bearbeiten können. Aus Sicherheitsgründen möchten Sie statt *Samba share* vielleicht das Add-on *Terminal & SSH* installieren.
Samba ermöglicht Ihnen auch die Verwendung von HA als Dateiserver, wenn Sie einen *www*-Ordner im *config*-Ordner erstellen.
Die Installation von Add-Ons ist einfach, klicken Sie einfach auf die Karte des Add-Ons, um es zu öffnen, und klicken Sie dann auf installieren. Eine geöffnete Karte enthält Anweisungen zur Konfiguration des Add-ons.

Installieren des ESPHome-Add-ons

Das bringt uns zur Installation des ESPHome-Add-ons. Dazu fügen Sie im Home Assistant zunächst die URL des Repositorys im Add-on-Store hinzu. Suchen Sie das Kästchen *Add new repositiory by URL* und fügen Sie die folgende URL dort ein:

`https://github.com/esphome/hassio`

Jetzt können Sie das ESPHome-Zusatzmodul installieren. Es gibt die drei Versionen Plain, Beta und Dev, von denen wir die Plain-Vanilla-Version verwenden. Die einzige Konfiguration, die ich für dieses Add-on vorgenommen habe, war die Aktivierung der Optionen *Start on boot*, *Auto update* und *Show in sidebar*. Letztere Option ist praktisch, da sie den Zugriff auf das Add-on viel einfacher macht.

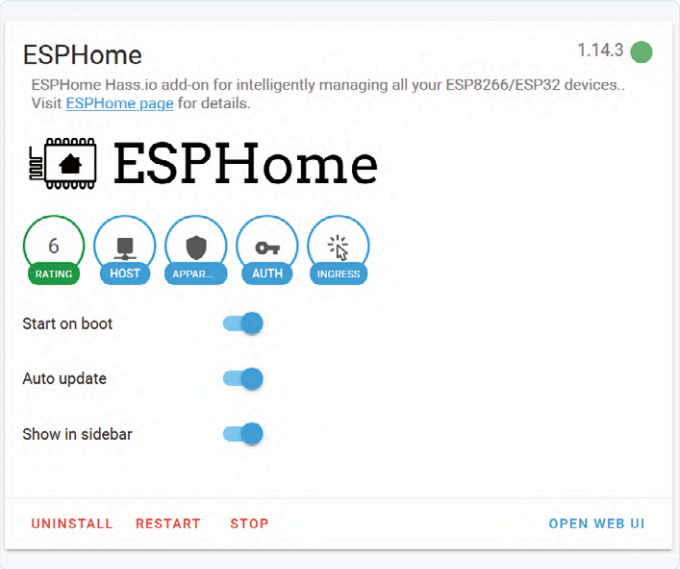


Bild 4. So ist die Option **Show in sidebar** *für das ESPHome-Add-on im Home Assistant aktiviert.*

Bild 5. nRF24-Module gibt es in verschiedenen Formen. Der 8-polige Typ scheint der gebräuchlichste zu sein; 10-polige Typen sind bis auf die Pinbelegung identisch.

AKTIVIEREN VON **MDNS** UNTER WINDOWS

Im Falle einer DHCP-Netzwerkkonfiguration (wenn das Netzwerk den angeschlossenen Geräten IP-Adressen zuweist) können Windows-Benutzer Schwierigkeiten haben, eine Verbindung zu HA herzustellen, weil das Multicast-DNS-Protokoll (mDNS) nicht läuft. Wenn mDNS läuft, können Sie eine Verbindung zu HA über die Adresse *http://hassio.local:8123* herstellen (unter [3] sind andere URLs angegeben, aber sie haben bei mir nicht funktioniert). Im Netz habe ich einen Weg gefunden, dies zu beheben:

1. Öffnen Sie den Registrierungseditor (Windowstaste + R, dann *regedit* eingeben) und navigieren Sie zu `Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows NT\DNSClient`
   Klicken Sie mit der rechten Maustaste darauf und wählen Sie *Neu*, um ein neues 32-Bit-DWORD namens *EnableMulticast* mit dem Wert 0 (Null) zu erstellen.
2. Laden Sie iTunes von der Apple-Website (https://www.apple.com/itunes/) herunter, aber installieren Sie es nicht. Dies erfordert ein wenig Sucherei. Das letzte Mal, als ich nachgesehen habe, musste man *iTunes for Windows* auswählen und es dann nicht aus dem Microsoft Store holen, sondern stattdessen nach unten nach *Looking for other versions?* scrollen. Dieser Link führt Sie zu der gewünschten Datei. Nachdem der Download abgeschlossen ist, ändern Sie die Extension .EXE in .ZIP und extrahieren Sie daraus die Datei *bonjour.msi* (oder *bonjour64.msi*, je nachdem, welche iTunes-Version Sie heruntergeladen haben).
3. Installieren Sie *bonjour* und starten Sie Ihren Computer neu.

EIN WORT ZUR **YAML-EINRÜCKUNG**

Sowohl Home Assistant als auch ESPHome verwenden YAML (Yet Another Markup Language) für ihre Konfigurationsdateien. Wie Python und alte Assemblersprachen verwendet dieses Format Einrückungen, um Informationen zu strukturieren (eine schreckliche Art, solche Dinge zu tun, wenn Sie mich fragen). Die Einrückung muss für alle Zeilen auf der gleichen Ebene gleich sein, aber jede Ebene kann eine andere Einrückung haben. Bei den YAML-Schnipseln in diesem Artikel besteht jede Ebene der Einrückung aus zwei Leerzeichen.

Integration von MySensors

Als ich diesen Artikel verfasste, gab es noch kein Home-Assistant-Add-on für MySensors. Um die Integration in HA zu ermöglichen, mussten der Datei *HA configuration.yaml* einige Zeilen hinzugefügt werden (siehe auch Kasten):

```
mysensors:
   gateways:
   - device: '192.168.1.100'
     persistence_file: 'mysensors/wifi_gateway.json'
     tcp_port: 5003
     optimistic: false
     persistence: true
     retain: true
     version: '2.0'
```

Wie angegeben, müssen Sie eine statische IP für das MySensors-Gateway wählen, das Sie erstellen wollen (ja, das wollen Sie, siehe unten). Sie müssen auch einen Namen und einen Speicherort für die JSON-Datei wählen, in der HA die MySensors-Netzwerkinformationen speichern kann.

Wichtig: Jedes Mal, wenn die Datei *HA configuration.yaml* geändert wird, benötigt das System einen Neustart, damit die Änderungen wirksam werden. Dies kann über die Registerkarte *System* des Supervisors erfolgen.
Wenn Sie die Konfigurationsdatei bearbeiten, können Sie auch direkt Folgendes hinzufügen (Speichern bei Neustart):

```
binary_sensor:
   - platform: workday
     country: [country code]
```

Diese Zeilen ermöglichen das Schreiben von Automatisierungsregeln, die nur an Werktagen oder an Wochenenden feuern. Ersetzen Sie `[country code]` durch den Code des Landes, in dem das System arbeitet. Diesen Code (und andere nützliche Informationen) finden Sie auf der Hilfeseite des *Workday Binary Sensor*.

Mein erstes ESPHome-Gerät

Wenn Sie alles wie oben beschrieben eingerichtet haben, können Sie nun ESP8266- und ESP32-basierte Geräte programmieren. Da ein neues (jungfräuliches) Gerät noch nicht mit ESPHome kompatibel ist, muss es zunächst über die serielle Schnittstelle programmiert werden. Je nach Gerät ist möglicherweise zuvor die Installation einen USB-zu-seriell-Port-Treibers auf dem HA-Computer erforderlich. Die von mir verwendeten NodeMCU-Boards mit einem CP2102-USB-Chip von Silabs (Silicon Laboratories) funktionierten aber ohne weitere Maßnahmen.
Nachdem Sie das Gerät an den Computer angeschlossen haben, auf dem HA läuft, öffnen Sie das ESPHome-Dashboard entweder über die Sidebar (wenn Sie diese Option aktiviert haben) oder in der Supervisor-Dashboard-Ansicht durch einen Klick auf *Open Web UI* von der Add-on-Karte aus. Vergewissern Sie sich, dass die serielle Schnittstelle aus der Dropdown-Liste in der oberen rechten Ecke verfügbar ist, die standardmäßig auf *OTA (Over-The-Air)* eingestellt ist. Sollte dies nicht der Fall sein, starten Sie das ESPHome-Add-on neu, indem Sie zur ESPHome-Karte im Supervisor-Dashboard zurückgehen. Das sollte reichen.
Im nächsten Schritt wird die rosafarbene +-Schaltfläche auf dem ESPHome-Dashboard angeklickt und damit ein Assistent geöffnet, der Sie durch den ersten Teil führt. Ich habe an keiner Stelle irgendein Zugangspasswort für irgendein Gerät angegeben, aber vielleicht bin ich unverantwortlich leichtsinnig. Wählen Sie für den Upload den seriellen Port, an dem das Gerät angeschlossen ist.

Bearbeiten der YAML-Datei

Der Assistent ist nun fertig und eine Karte für Ihr Gerät erstellt worden. Zu diesem Zeitpunkt funktionierte (bei mir) ein Klick auf die Schaltfläche *Edit* erst nach dem Neuladen der Seite. Klicken Sie also auf *Edit* und überprüfen Sie die Anmeldedaten für Ihr WLAN-Netzwerk. Auch sollten die Zeilen *api:* und *ota:* vorhanden sein. Wenn Sie diese Konfiguration auf Ihr Gerät hochladen und es neu starten, wird sie vom Home

Assistant erkannt. Es kann nun auch vom Computer getrennt werden, weil die Over-the-Air-Programmierung aktiviert wurde. Das Gerät wird jedoch nichts tun, weil Sie seine Peripherie nicht konfiguriert haben. Lesen Sie deshalb weiter, bevor Sie eine Konfiguration hochladen.
Wenn Sie ein NodeMCU-Modul verwenden, können Sie es in Betrieb nehmen, indem Sie vor dem Hochladen der Konfigurationsdatei die untenstehenden Zeilen am Ende der Konfigurationsdatei (der Übersichtlichkeit halber unterhalb von *ota:*, aber die Position in der Datei spielt eigentlich keine Rolle) anfügen. Dadurch erhält HA Zugriff auf die On-Board-LED und die *Flash*-Taste.

```
output:
   - platform: gpio
     id: „led“
     pin:
       number: GPIO16
       inverted: True

 light:
   - platform: binary
     name: „LED“
     output: „led“

 binary_sensor:
   - platform: gpio
     name: „Flash pushbutton“
     pin:
       number: GPIO0
       inverted: True
```

Beachten Sie, dass in diesem Programmschnipsel jede Einrückungsebene aus zwei Leerzeichen besteht, was bedeutet, dass die Zeilen `number: GPIOx` und `inverted: True` mit sechs Leerzeichen beginnen, also drei Ebenen eingerückt sind (siehe Kasten).
Diese Konfiguration sollte auch mit einem ESP-01-Modul funktionieren, wenn Sie GPIO16 in GPIO3 ändern und eine LED in Reihe mit einem 470-Ω-Widerstand zwischen GPIO3 (RXD) und Masse schalten. Schließen Sie einen Taster zwischen GPIO0 und GND und einen 10-kΩ-Pull-up-Widerstand zwischen GPIO0 und 3V3 an. Drücken Sie diesen Taster nicht beim Booten des Geräts. Laden Sie diese Konfiguration auf das ESPHome-Gerät hoch und starten Sie es neu. Der Home Assistant sollte die Konfiguration erkennen - das ESPHome-Dashboard zeigt sie als *Online* an - und, wenn Sie es zulassen, erstellt er eine Steuerung für eine Lampe und eine Anzeige für den Drucktaster. Wenn Sie zuerst die leere Konfiguration hochgeladen haben, dann müssen Sie möglicherweise in das HA-Menü *Configuration* gehen, um sie in der *Devices*-Liste zu sehen. Sie können nun Boards für die Steuerungen zur *Overview* in HA hinzufügen (klicken Sie auf die drei Punkte und dann auf *Configure UI*) und erstellen Sie im Konfigurationsmenü (*Automations*) automatische Vorgänge im *Automation Editor*. Ich überlasse es Ihnen, dies zu ergründen, da es ziemlich selbsterklärend und zudem ein guter Anlass ist, sich im Home Assistant umzuschauen. Außerdem wartet MySensors auf uns.

Ein MySensors-WLAN-Gateway

Ein Gateway ist sowohl erforderlich, um ein MySensors Netzwerk aufzubauen, als auch, um Verbindung zu anderen (WLAN-)Netzwerken aufzunehmen. Hierfür habe ich eine ESP8266-basierte NodeMCU verwendet, weil sie einen nutzbaren SPI-Port zur Verfügung stellt. Die Verwendung eines ESP32-Moduls ist ebenfalls eine Option.

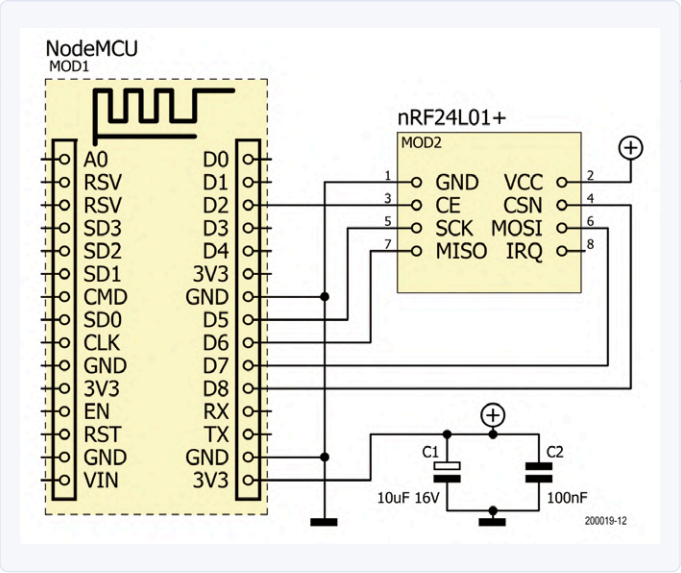


Bild 6. Mit einem NodeMCU-Modul kann ein MySensors-WLAN-Gateway aufgebaut werden.

Schließen Sie den SPI-Port an ein nRF24L01+-Modul an (**Bild 6**). Es wird empfohlen, auch einen Elektrolytkondensator (4,7...47 µF) zusammen mit einem Keramikkondensator (etwa 100 nF) zwischen den VCC- und GND-Pins des nRF24-Moduls zu schalten. Das ist alles, was Sie an „Elektronik“ benötigen.
Auf der Softwareseite ist ein Computer mit installierter Arduino-IDE erforderlich. Fügen Sie der IDE den ESP8266- (oder ESP32-)Kern für Arduino hinzu - alle Details finden Sie unter [6] - und binden Sie die MySensors-Bibliothek ein (Sketch Bibliothek einbinden Bibliotheken verwalten). Laden Sie das Beispiel *GatewayESP8266* (Datei Beispiele MySensors) und geben Sie die korrekte SSID, das Passwort und die statische IP ein, die Sie zuvor in der HA-Datei *configuration.yaml* angegeben haben. Laden Sie den Sketch auf Ihr Gerät hoch und schon ist Ihr Gateway betriebsbereit.
Bitte denken Sie daran, dass ein Gateway (und auch Repeater-Knoten, die in diesem Artikel nicht behandelt werden) immer mit Strom versorgt werden muss und niemals schlafen dürfen. Sensorknoten dagegen können tun, was immer sie wollen.

Mein erster MySensors-Knoten

Das Erstellen eines MySensors-Knotens ist dem Aufbau eines Gateways nicht unähnlich, außer dass das ESP-Modul durch eine Arduino-kompatible Platine ersetzt wird, an der Sensoren und Aktoren angeschlossen werden. Schließen Sie ein nRF24L01+-Modul an den SPI-Port des Boards an und fügen Sie die oben erwähnten Kondensatoren hinzu (**Bild 7**).
Laden Sie einen Beispielsketch aus der MySensors-Beispielbibliothek, vorzugsweise einen, der dem ähnelt, was Sie zu erreichen versuchen. Sollten Sie keinen finden: Es gibt weitere Beispiele auf der MySensors-Website. Im Sketch müssen die Pin-Nummer(n) an Ihre Peripheriegeräte angepasst werden, aber im Grunde ist das alles, was zu tun ist. Laden Sie den Sketch auf Ihr Gerät hoch. Wenn Sie es booten, tritt es automatisch dem MySensors-Netzwerk bei (wenn das Gateway eingeschaltet ist).

Vorsicht bei der Protokollversion!

An diesem Punkt stieß ich auf das Problem, dass mein Gerät, ein ferngesteuertes Relais, im Home Assistant nicht auftauchte. Schließlich entdeckte ich, dass dies etwas mit der verwendeten Version der MySensors-API zu tun hatte. Bei der Durchsicht der Persistenz-JSON-Datei (siehe Abschnitt „Integration von MySensors“) bemerkte ich, dass das Gateway als API- oder Protokollversion 2.3.2 aufgeführt war. Die Seite

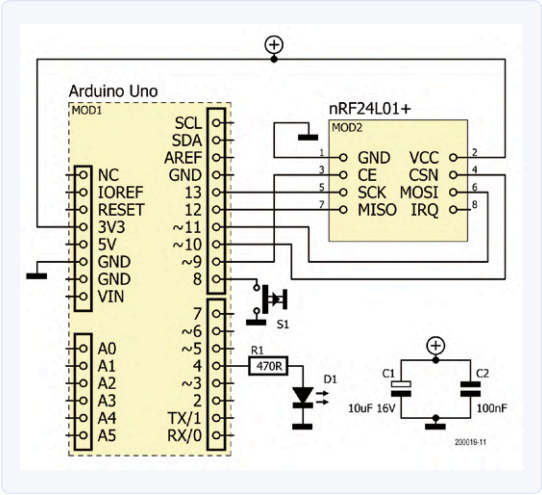


Bild 7. Der Schaltplan eines MySensors-Knotens mit einem Taster für einen binären Sensor und einer LED für einen Aktor.

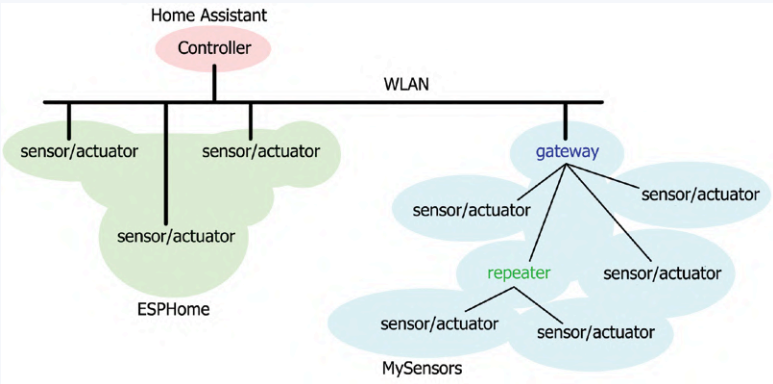


Bild 8. Wenn Sie alle in diesem Artikel beschriebenen Elemente kombinieren, erhalten Sie am Ende ein Hausautomatisierungssystem wie dieses.

MySensors unter [7] enthält ein Beispiel, das für die Verwendung mit API-Versionen 2.x vorgesehen ist. Als ich das ausprobierte, erschien mein Knoten in der Liste *Entities* im HA-Menü *Configuration* und ich konnte dafür UI-Karten erstellen.
Gemäß [7] muss ein V2.x-basierter Knoten, damit er in HA funktioniert, einen Anfangswert von der Funktion loop() senden. Die Beispielsketches tun dies nicht. Bei näherer Betrachtung kam mir der Verdacht, dass ein Aktor-Knoten einen Initialwert aus HA anfordern (und vielleicht auch senden) muss, um erkannt zu werden. Ein Sensorknoten wird gemeldet, sobald er mit dem Senden von Daten beginnt. Die Anforderung muss nicht über `loop()` erfolgen, sondern einfach irgendwo während des Bootvorgangs.

Spezielle Funktionen *before()* und *presentation()*

Um V2.x-Sketches älteren Typs auszumachen, suchen Sie nach der Funktion `presentation()`. Wenn diese Funktion vorhanden ist, so handelt es sich um V2 oder höher. Umgekehrt trifft dies jedoch nicht unbedingt zu, da `presentation()` weggelassen werden kann. Die `present`-Anweisungen, die normalerweise in dieser Funktion zu finden sind, dürfen aber nicht ausgelassen werden und müssen in eine andere Funktion verschoben werden, zum Beispiel in `setup()`.
MySensors-Sketches können auch eine Funktion `before()` enthalten. Sowohl `before()` als auch `presentation()` werden vor `setup()` aufgerufen, und zwar in der Reihenfolge `before()`, `presentation()`, `setup()` und schließlich `loop()`.
Die Anpassung von vorhandenem Code an Ihre Hardware ist jetzt ganz einfach. Viele Beispiele für alle Arten von gängigen Sensoren sind entweder in der Arduino-Bibliothek oder online verfügbar, also schauen Sie sich um, bevor Sie (zu) tief in die Materie eintauchen.

Ich hoffe, Sie fanden es interessant...

Hier endet die kleine Lektion über einfache Heimautomatisierung (**Bild 8**). Sie ist sicher nicht vollständig und es gibt bestimmt noch bessere oder einfachere Möglichkeiten und auch viele gänzlich andere Heimautomatisierungsprojekte. Einige sind vielleicht besser, andere sehen vielleicht schöner aus, aber sie alle haben ihre Stärken und Schwächen.
Da es so viele gibt, ist es schwer, sich für eine Variante zu entscheiden. In diesem Artikel habe ich einige vorgestellt, die ich ausprobiert und beibehalten habe, um mein Hausautomationssystem darauf aufzubauen. Ich verwende sie immer noch, und ich bin immer noch erstaunt über die endlosen Möglichkeiten, die sie bieten. Bleiben Sie dran, denn ich habe nicht vor, hier aufzuhören.
Bei Elektor Labs finden Sie Beispielkonfigurationen für ESPHome [8] und MySensors-Geräte [9] und auch einige sehenswerte Videos.

200019-03

WEBLINKS

[1] **Espurna**: https://github.com/xoseperez/espurna
[2] **ESPHome**: https://esphome.io/
[3] **Home Assistant**: https://www.home-assistant.io/
[4] **Sonoff**: https://www.itead.cc/
[5] **MySensors**: https://www.mysensors.org/
[6] **ESP8266 core for Arduino**: https://github.com/esp8266/Arduino
[7] **MySensors in Home Assistant**: https://www.home-assistant.io/integrations/mysensors
[8] **ESPHome bei Elektor Labs**: www.elektormagazine.de/labs/how-to-home-assistant-esphome
[9] **MySensors bei Elektor Labs**: www.elektormagazine.de/labs/mysensors-home-assistant-howto

Die Speicher-Kathodenstrahlröhre

Bemerkenswerte Bauteile

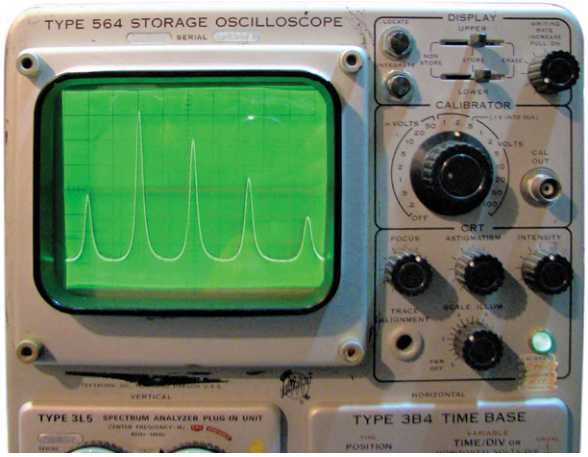


Image: TekWiki

Von **Neil Gruending** (Kanada)

Herkömmliche Kathodenstrahlröhren-Oszilloskope (CRT-Oszilloskope) sind ein hervorragendes Mittel zur Beobachtung sich wiederholender Signale, aber langsame Sweeps und nur sporadisch auftretende Signale sind eine Herausforderung. Moderne Digitaloszilloskope lösen diese Probleme, indem sie die Wellenformen in digitaler Form in einem Speicher aufbewahren, aber frühe Analogoszilloskope benötigten eine andere Lösung. Ein beliebter Ansatz war es, die Kathodenstrahlröhre selbst als Speicher zu verwenden. Diese Röhren waren als Speicher- oder bistabile CRTs bekannt.

Eine Speicher-Kathodenstrahlröhre, wie sie **Bild 1** zeigt, macht sich die Tatsache zunutze, dass die Geschwindigkeit der zwischen zwei Oberflächen fließenden Elektronen durch die Spannungsdifferenz zwischen ihnen gesteuert wird. Sie besitzt zwei Glühkathoden, eine zum Schreiben und eine zum „Fluten". Die Schreibkathode ist die übliche Kathode, die die Wellenform auf den phosphorbeschichteten Bildschirm wirft. Der Unterschied zu einem herkömmlichen Oszilloskop besteht aber darin, dass die Steuerspannung hoch genug ist, um eine positive Ladung im Phosphor zu erzeugen. Diese positive Ladung entsteht, wenn sich die Elektronen so schnell bewegen, dass beim Auftreffen auf den Phosphor mehr Elektronen freigesetzt werden als einschlagen.

Sobald die Wellenform geschrieben ist, wird sie mit Hilfe der Flutkathode dauerhaft sichtbar gemacht. Dabei trifft ein Strom von niederenergetischen Elektronen auf die gesamte Phoshor-Oberfläche. Die Elektronen sind zu langsam, um auf der unbeschriebenen Oberfläche viele Elektronen freizusetzen, so dass diese Bereiche leicht negativ geladen sind und der Stromfluss stoppt. Aber die positiv geladenen Bereiche, in denen die Wellenform geschrieben wurde, ziehen die Elektronen an und beschleunigen sie. Wenn sie dann auf die Phosphoroberfläche treffen, sind sie schnell genug, dass sie in diesen Bereichen die gleiche Anzahl von Elektronen freisetzen. Dadurch bleibt die positive Ladung erhalten und es entsteht ein Stromfluss, der die Wellenform weiter leuchten lässt.

Es gab jedoch ein Problem bei dieser Methode, da sich die gespeicherte Wellenform mit der Zeit ausbreiten würde wie Tinte auf nassem Papier. Das Hughes Memoscope 104 verwendete mehrere Maschengitter auf der Phosphorfläche, um den Effekt in Schranken zu halten, aber es fehlte dem Oszilloskop an Kontrast und Schärfe. Und es war auch sehr teuer in der Herstellung. Dann, im Jahr 1959, begann Bob Anderson für Tektronix nach einer kostengünstigeren Alternative zu suchen. Da er wusste, dass der Phosphor eine diskontinuierliche oder gebrochene Zieloberfläche haben musste, versuchte er zunächst, Punktmuster im Phosphor einzusetzen. Leider war die präzise Herstellung sehr schwierig, aber dieser Ansatz bewog ihn, eine poröse Schicht aus gestreuten Phosphorpartikeln auf der inneren Bildschirmoberfläche als Speichermedium zu entwickeln (**Bild 2**).

Diese neue Röhre wurde erstmals im Jahr 1963 im Oszilloskop Tek 564 verwendet. Diesem Modell folgte schnell das Oszilloskop Tek 549, das über einen geteilten Bildschirm verfügte, der zwei verschiedene Wellenformen speichern konnte.

190383-E-03

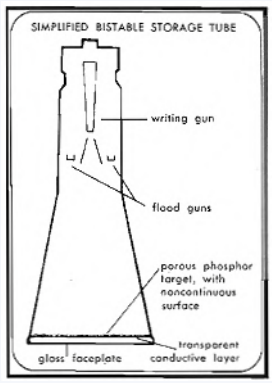


Bild 1. Vereinfachtes Diagramm einer Speicherröhre [1].

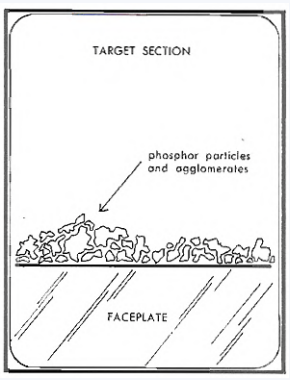
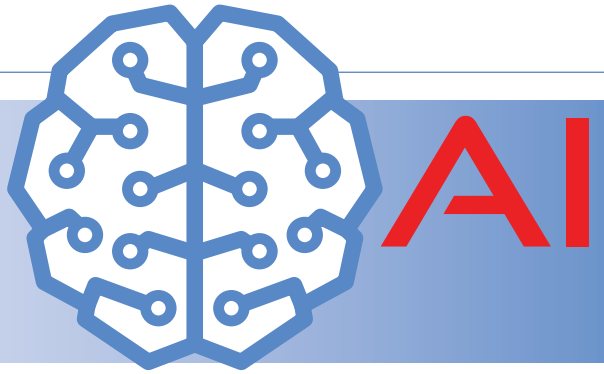


Bild 2. Poröse Phosphorschicht [1].

WEBLINK

[1] **Vintage Tek - The Storage Story**: https://vintagetek.org/wp-content/uploads/2011/10/The-Storage-Story3.pdf

KI für Einsteiger (3)



Ein eigenes Neuronales Netz

Von Walter Trojan

In den beiden vorangegangenen Teilen dieser Serie wurde das Maixduino-Board vorgestellt und gezeigt, wie man mit MicroPython ein vorgefertigtes Neuronales Netz in Betrieb nehmen kann, um auf Bildern Gesichter zu erkennen. In dieser dritten und letzten Folge soll ein eigenes Neuronales Netz entwickelt werden. Außerdem geht es um den ESP32 auf dem Board, der als Co-Prozessor Kommunikationsaufgaben übernehmen kann.

Bisher habe ich gezeigt, wie man bekannte KI-Demos mit bereits trainierten NNs auf dem Maixduino ausführt. Will man eigene Neuronale Netze implementieren, muss man derzeit ebenfalls noch die Entwicklung und das Training auf einem PC oder einem Cloud-Service durchführen und dann das trainierte NN auf den Maixduino transferieren. Dieses Verfahren muss kein Nachteil sein, denn ein Training erfordert meistens große Datenmengen und eine hohe Verarbeitungsleistung. Und das fertige NN kann dann mit geringem Energieverbrauch auf einer mobilen Plattform, z.B. bei einer Robotersteuerung, zum Einsatz kommen.
Nachfolgend beschreibe ich den Weg von der Entwicklung eines NNs bis zur Ausführung auf dem Maixduino. Zuerst ist dafür eine Entwicklungsumgebung auf dem Linux-PC erforderlich, die bei vielen Entwicklern eine empfehlenswerte Struktur aufweist, die in **Bild 1** zu sehen ist.

KI-Baukasten

Die Schnittstelle zum Entwickler bildet das KI-Framework Keras, das auch als komfortabler „Legobaukasten für KI" bezeichnet wird. Es setzt auf dem sehr verbreiteten Framework TensorFlow auf, das die eigentliche Arbeit verrichtet, oder auf einem alternativen System wie Theano. Verfügt der PC über eine Nvidia-Grafikkarte, kann man die vielen Grafikprozessor-Kerne über die CUDA-Bibliothek nutzen, ansonsten setzt man die PC-CPUs über BLAS in Aktion.

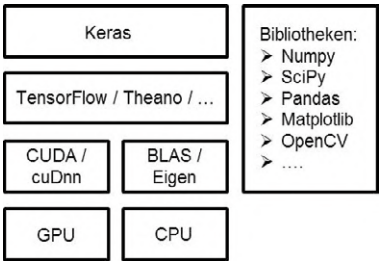


Bild 1. Struktur einer populären KI-Entwicklungsumgebung.

Hier noch einmal die wichtigsten Bausteine in Kürze:
- *Keras*: KI-Framework, setzt auf TensorFlow oder anderen Frameworks auf.
- *TensorFlow*: Leistungsstarkes und vielfach eingesetztes KI-Framework, von Google entwickelt und bereitgestellt.
- *CUDA / cuDnn*: Bibliothek zur Unterstützung von Nvidia-Grafikprozessoren.
- *BLAS / Eigen*: Programmbibliothek mit elementaren Operationen der linearen Algebra wie Vektor- und Matrixmultiplikationen.

Bibliotheken werden bei Bedarf eingebunden:
- *Numpy*: Stellt hocheffiziente Funktionen, insbesondere für Matrizenoperationen zur Verfügung.
- *SciPy*: Auf *Numpy* aufbauende Bibliothek mit zusätzlichen Funktionen, z.B. Differentialgleichungen.
- *Pandas*: Eine im Programm nutzbare Tabellenkalkulation.
- *Matplotlib*: Zeichnet Diagramme und Images.
- *OpenCV*: Leistungsstarke Bildverarbeitung.
- *HDF5*: Bibliothek zur Verarbeitung und Speicherung von heterogenen Datenbeständen.
- *Graphviz*: Zur Visualisierung von strukturierten Informationen.
- *pydot-ng*: Hilfsprogramm für Graphviz.
- *Jupyter*: Entwicklungsumgebung für Python, Programm-Abschnitte werden in Zellen strukturiert, die auch einzeln ausführbar sind, von Profis bevorzugt.

Darüber hinaus sind noch weitere KI-Frameworks verfügbar, z.B. Theano, Torch, Caffe, Pytorch, Yolo und natürlich noch zusätzliche Bibliotheken. Da zeigt sich die Reichhaltigkeit der Linux/Python-Welt und motiviert zum Googeln. Die in Bild 1 gezeigte Umgebung kann auf einem Linux-PC mit folgenden Terminalkommandos eingerichtet werden:

Aktualisierung:

```
sudo apt-get update
sudo apt-get upgrade
```

Python und pip:
```
sudo apt-get install python3-pip python3-dev
pip install --upgrade pip
```

OpenBlas:
```
sudo apt-get install build-essential cmake git unzip
   pkg-config libopenblas-dev liblapack-dev
```

SciPy, Numpy:
```
sudo apt-get install python-numpy python-scipy
   python-yaml
```

matplotlib:
```
sudo pip3 install matplotlib
```

HDF5:
```
sudo apt-get install libhdf5-serial-dev python-h5py
```

Graphviz, pydot-ng:
```
sudo apt-get install graphviz
sudo pip3 install pydot-ng
```

OpenCV:
```
sudo apt-get install python3-opencv
# ist Version 3, Version 4 mit aufwändiger Prozedur
```

Tensorflow 2.0:
```
pip3 install tensorflow==2.0.0b1
```

Keras:
```
sudo pip3 install keras
```

Pandas:
```
sudo apt-get install python3-pandas
```

Thonny:
```
sudo apt-get install thonny
```

Jupyter:
```
pip3 install jupyter
```
Aufruf:
`jupyter notebook` (Terminal-Kommando)

Damit stehen die wesentlichen Bausteine zur Verfügung. Sind projektspezifisch zusätzliche Komponenten erforderlich, installiert man diese mittels `pip` oder `apt-get`. Der gewählte Python-Editor Thonny bietet zwar nur minimale, aber ausreichende Funktionalität, Profis nutzen andere wie z.B. Jupyter, der eine Programm-Strukturierung in Zellen erlaubt, die auch einzeln ausführbar sind.

Damit sind alle Vorbereitungen getroffen, um ein NN von der Pike auf zu entwickeln und auf den Maixduino zu transferieren. Als Beispiel dafür habe ich die Erkennung von handgeschriebenen Ziffern ausgewählt. Basis hierfür ist die MNIST-Datenbank (Modified National Institute of Standards and Technology database), in der 60.000 Beispielbilder zum Training und 10.000 zum Test zur freien Verfügung stehen. Alle darin enthaltenen Bilder im Format 28 x 28 Pixel besitzen ein Label mit dem korrekten Ziffernwert (**Bild 2**).

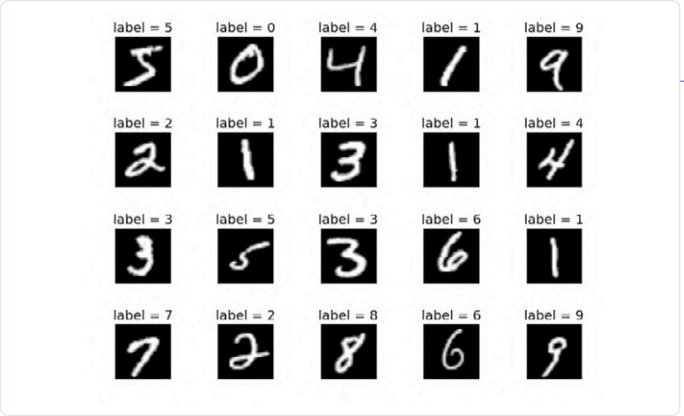


Bild 2. Auszug aus der MNIST-Datenbank.

Schritt für Schritt zum NN

Das mittels Keras/Tensorflow zu erstellende NN soll ihm noch nicht bekannte handgeschriebene Ziffern richtig klassifizieren. Grundsätzlich läuft die Entwicklung eines NNs in folgenden Phasen ab:

- **Datenaufbereitung**: Trainings- und Testdaten in ein passendes Format konvertieren.
- **Definition**: Festlegung der NN-Struktur mit Art und Anzahl der Schichten.
- **Kompilierung**: Übersetzung der Keras-Instruktionen in Tensorflow-Befehle.
- **Training**: Wiederholtes Vorspielen der Trainingsdaten zum Abgleich der NN-Gewichte.
- **Test**: Überprüfung der Vorhersagequalität mit den Testdaten.
- **Erkennung**: Vorlage neuer Ziffern und deren Klassifizierung.

Datenaufbereitung

Die Daten von MNIST stehen bereits in der Keras-Bibliothek zur Verfügung und werden im passenden Format importiert.

Definition

Das folgende von Profis entwickelte Modell verfügt bereits über eine exzellente Erkennungsrate von über 99%.

```
model = Sequential()
   # sequenzielles Modell, es gibt auch funktionale
model.add(Conv2D(32, kernel_size=(3, 3),
   # erste zweidimensionale Convolutions-Schicht mit
   32
                    Ausgängen
                activation='relu',
   # Aktivierungsfunktion
                input_shape=input_shape))
   # Eingangsformat 28*28
model.add(Conv2D(64, (3, 3), activation='relu'))
   # zweite Convolutions-Schicht mit 64 Ausgängen und
                    relu-Aktivierungsfunktion
model.add(MaxPooling2D(pool_size=(2, 2)))
   # Verdichtung
model.add(Dropout(0.25))
   # Verstärkung der Robustheit
model.add(Flatten())
   # Von zwei auf eine Dimension umwandeln
model.add(Dense(128, activation='relu'))
   # Dritte NN-Schicht mit 128 Ausgängen und
                     relu-Aktivierung
model.add(Dropout(0.5))
```

ESP32 - HELFER BEI SPEZIELLEN AUFGABEN

Dem ansonsten reichhaltig ausgestatteten KI-Chip K210 fehlen analoge Eingänge und er kann nicht über WiFi und Bluetooth kommunizieren. In diese Bresche springt der ESP32, der außer diesen Funktionen noch einen Doppelprozessor mit einer Taktfrequenz von 240 MHz einbringt. Die Kopplung zwischen K210 und ESP32 erfolgt über eine schnelle SPI-Verbindung; sechs Analogeingänge sind auf die arduino-typische Buchsen-Leiste herausgeführt. Eine serielle Verbindung zum ESP32 ist über den Port *ttyUSB1* möglich und programmiert wird der ESP32 mittels der MaixPy-IDE über den K210.

Dazu ein erstes Beispiel für die Erfassung analoger Signale:

```
import network
   # import necessary modules
import utime
from Maix import GPIO
from fpioa_manager import *

#iomap at MaixDuino
   # register GPIOs for ESP32 interface
fm.register(25,fm.fpioa.GPIOHS10)   # cs
fm.register(8,fm.fpioa.GPIOHS11)    # rst
fm.register(9,fm.fpioa.GPIOHS12)    # rdy
fm.register(28,fm.fpioa.GPIOHS13)   # mosi
fm.register(26,fm.fpioa.GPIOHS14)   # miso
fm.register(27,fm.fpioa.GPIOHS15)   # sclk

# definition of network interface
nic = network.ESP32_SPI(cs=fm.fpioa.GPIOHS10,rst=fm.fpioa.GPIOHS11,rdy=fm.fpioa.GPIOHS12,
mosi=fm.fpioa.GPIOHS13,miso=fm.fpioa.GPIOHS14,sclk=fm.fpioa.GPIOHS15)

adc = nic.adc( (0,1,2) )
    # get ADC0 ADC1 ADC2
print('ADC 0,1,2')
    # print results of ADC 0-2
print(adc)
print()
print('ADC 0,1,2,3,4,5')

while True:
    try:
        adc = nic.adc()
    # get ADC0-5
    except Exception as e:
        print(e)
           # in case of error print reason
        continue
    for v in adc:
        print("%04d" %(v), end=" ")
           # print results of ADC 0-5 formatted
    print()
```

```
   # Verstärkung der Robustheit
model.add(Dense(num_classes, activation='softmax')) #
   Vierte NN-Schicht mit 10 Ausgängen und
                      softmax-Aktivierung
```

Das NN verfügt über vier Schichten, davon zwei Convolutional-Layer und zwei klassische Schichten (Dense) mit einer übersichtlichen Zahl von Knoten (32/64/128/10). Die Aktivierung der letzten Schicht mit `softmax` begrenzt die Summe der Ausgangswerte auf die Wahrscheinlichkeit 1. In den beiden ersten Schichten werden in Feldern von jeweils 3 x 3 Pixeln Details wie Linien, Bögen und Punkte herausgefiltert. Zwischen den Schichten sind noch Filter mit folgenden Funktionen eingefügt:

`MaxPooling` verdichtet die Ausgabe einer Schicht und übergibt nur Durchschnittswerte eines Feldes (hier 2 x 2). `Dropout` vernachlässigt beim Training zufällig gewählte Knoten und ordnet damit den Nachbarn mehr Bedeutung zu, macht dadurch das NN robuster. `Flatten` verwandelt eine mehrdimensionale CNN-Ausgabe in eine eindimensionale.

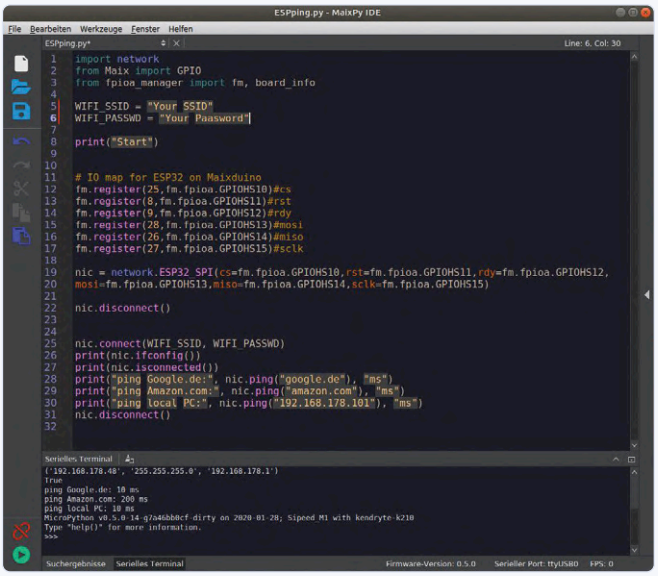
Kompilierung

In dieser Phase erfolgt die Umwandlung der Keras-Befehle in Tensorflow-Instruktionen.

```
model.compile(loss=keras.losses.categorical_
   crossentropy,      # Zuordnung der Verlustfunktion
```

```
    utime.sleep_ms(1000)
          # pause 1000 ms
```

Zum Programmbeginn erfolgt der Import der benötigten Python-Module, danach werden die GPIOs für die Kommunikation mit dem ESP32 registriert und dem Objekt `nic` zugeordnet. Zur Erfassung der analogen Eingänge erhält die Funktion `nic.adc` ein Tuple mit den gewünschten Kanälen, hier ADC0 bis ADC2. Nach dem Display dieser Resultate werden in der `while`-Schleife ohne Nennung der Kanalnummern sogar alle sechs verfügbaren Eingänge abgefragt. Falls Fehler bei der Erfassung auftreten, wird über die Ausnahmefunktion der Fehlercode ausgegeben.
Zur Funktionsprüfung der Analogerfassung habe ich dem Kanal ADC0 über ein Potentiometer einen variablen Eingangspegel zugeleitet. Dabei ist zu beachten, dass der native Analogeingang nur den Bereich 0 bis 1,0 V überstreicht. Die vom Terminal dargestellten Werte bestätigen die korrekte Arbeitsweise:




```
ADC 0,1,2
(0, 1786, 73)

ADC 0,1,2,3,4,5
0000 1469 0082 0521 0120 0001
    # Poti auf 0 V
0000 1813 0160 0606 0291 0000
2622 2135 0210 0630 0323 0000
    # Poti Mittelstellung
4095 2421 0259 0575 0261 0000
4095 2472 0274 0615 0315 0000
    # Poti auf 1.0 V
```

Die anderen Kanäle waren nicht angeschlossen und zeigen in den Spalten 1-5 zufällige Werte.

In einem zweiten Beispiel wird die Kommunikation über das Internet verdeutlicht, dabei werden zwei bekannte Internetportale Google und Amazon sowie ein lokaler Rechner angepingt (siehe Screenshot).
Der Import von Modulen sowie die Registrierung von GPIO-Ports und die Definition des `nic`-Objekts erfolgen wie beim ersten Beispiel. Zur Nutzung des Internets sind natürlich die Zugangsdaten zum WiFi-Router erforderlich. Nach der Verbindung mit dem Netz kann man mit `nic.ifconfig()` die IP-Adresse des ESP32 erfragen und mit `nic.isconnected()` wird der Status der Verbindung angezeigt. Beim Befehl `nic.ping()` ist lediglich die Zieladresse anzugeben. Neben den Internetadressen können auch lokale Rechner mittels Angabe ihrer IP-Adresse angepingt werden.
Es lassen sich noch weitere Anwendungsbeispiele finden, z.B. ein Webserver zur Übermittlung der KI-Analyse, eine Erfassung von Objekten aus dem Internet, und vieles mehr. Lassen Sie sich von den Beispielen unter dem Weblink [5] inspirieren und zu Experimenten einladen. Bitte beachten Sie, dass der Maixduino noch eine junge Konstruktion ist und sich noch in einer eher stürmischen Entwicklungsphase befindet, d.h. neben der Firmware des K210 ist auch für die Anbindung des ESP32 hin und wieder ein Update erforderlich. Dieses lässt sich mit folgender Befehlssequenz nach dem Download der unten genannten Binärdatei von Link [4] erledigen:

```
pip install esptool
esptool.py --chip esp32 --port /dev/ttyUSB1 erase_flash
esptool.py --chip esp32 --port /dev/ttyUSB1 --baud 1500000 write_flash -z 0x0000
   maixduino_esp32_firmware_v1.4.0.bin
```

```
            optimizer=keras.optimizers.
  Adadelta(),       # Auswahl des Optimierers
            metrics=['accuracy'])
                 # Definition der Genauigkeit
```

Training
Jetzt wird das NN mit 60.000 Ziffernbildern trainiert und dabei in 12 Durchläufen (Epochen) durchgespielt.

```
model.fit(x_train, y_train,
  # x_train enthält Bilder, y_train enthält Labels
        batch_size=batch_size,
  # Training erfolgt stapelweise, z.B. in 16er
  Paketen
        epochs=epochs,
  # Anzahl der Epochen, Durchläufe mit allen Daten
        verbose=1,
  # Ausgabe des Trainingsverlaufs
        validation_data=(x_test, y_test))
  # Validierung des Modells mit den Testdaten
```

Test
Abschließend wird die Erkennungsrate mit den 10.000 Testdaten festgestellt.

```
score = model.evaluate(x_test, y_test, verbose=0)
model.save("Mnist3.h5")
```

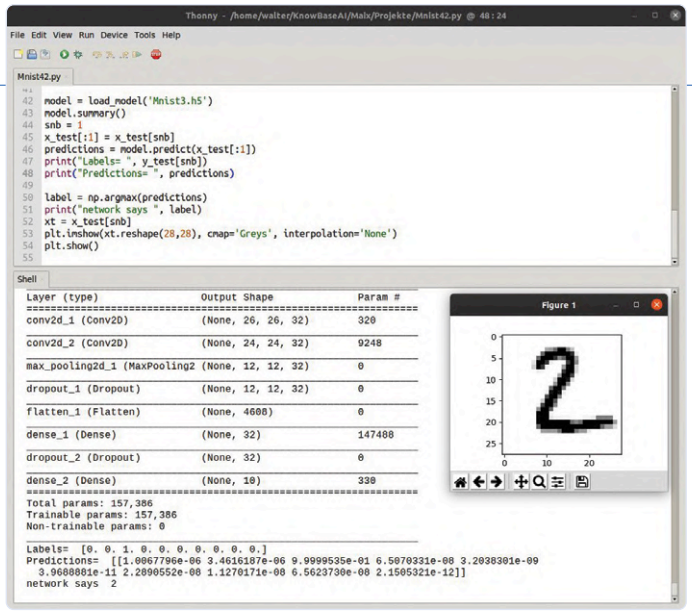


Bild 3. Erkennung der Ziffer 2.

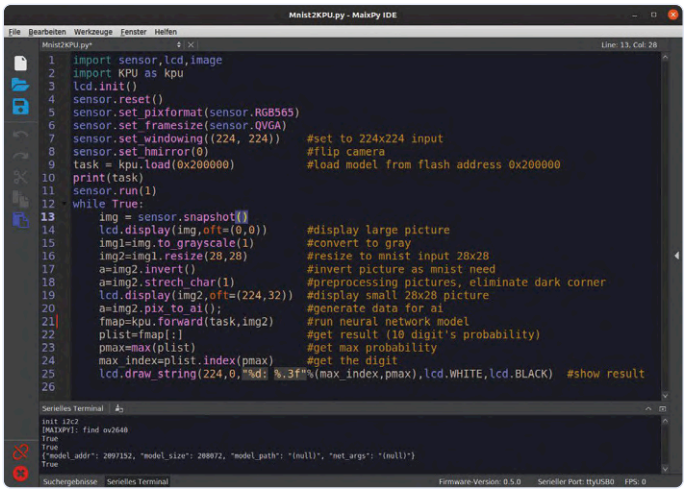


Bild 4. Ziffernerkennung mit MicroPython.

Das Modell kann nun mit Struktur und trainierten Gewichten im *h5*-Format abgespeichert und später – auch auf einer anderen Plattform - zur Erkennung genutzt werden.

Erkennung

Diese soll nach dem Transfer auf den Maixduino lokal auf diesem Board erfolgen. Man kann dieses aber auch auf dem Linux-PC mit dem Editor Thonny bewerkstelligen, wie **Bild 3** zeigt.

Zur Erkennung neuer handgeschriebener Ziffern wird das bereits trainierte NN geladen; anschließend erfolgt die Ausgabe der Modellstruktur. Dem aufmerksamen Leser wird nicht entgangen sein, dass in einigen Schichten die Knotenanzahl verringert wurde. Dieses war erforderlich, damit das NN in den kleineren Hauptspeicher (max. 5,9 MB für das Modell) hineinpasst. Danach erfolgt die Erkennung des zweiten Elements der Testdaten, zufällig eine „2", was die Label zu diesem Datensatz bestätigen. Das NN sieht das ebenso, denn seine Erkennung weist für diesen Index eine Wahrscheinlichkeit von 0,9999 aus, während alle anderen Ziffern vernachlässigbare Werte erhalten. Zur verständlichen Auswertung trägt *numpy* mit der `argmax`-Funktion bei, die nur den größten Wert des Arrays ausgibt. Mittels der Bibliothek *matplotlib* wird zusätzlich das Eingabebild gezeichnet. Das Modell ist nun auf dem Linux-PC trainiert und getestet. Was jetzt noch fehlt ist der Transfer auf den Maixduino.

Diese Übertragung besteht aus folgenden Schritten:

- Erstellung von Modell mit Training und Test mittels Keras, Ergebnis ist ein KI-Modell im *h5*-Format (wie gerade gezeigt).
- Konvertierung des *h5*-Modells nach TensorflowLite in das *tflite*-Format.
- Kompilation des *tflite*-Modells mittels nncase-Compiler in ein Format für die KPU, das *kmodel*-Format.
- Dieses Modell mittels Kflash auf die Adresse 0x200000 in den Maixduino flashen.
- Mittels MaixPy-IDE ein Python-Script erstellen und auf den Maixduino downloaden sowie dort ausführen und testen.

Weiterführende Dokumentation darüber finden Sie im Elektor-Projektordner [3], in dem sämtliche im Artikel verwendeten Programme und Dateien mit Hinweisen bereitgestellt sind.

In **Bild 4** sieht man die Anwendung auf dem Maixduino.
Nach dem Import benötigter Module werden Kamera und LCD initialisiert und eingestellt. Danach wird das NN von Adresse 0x200000 geladen und steht mit dem Objekt `task` zur Verfügung. Innerhalb der `while`-Schleife erfolgt zuerst die Aufnahme eines Bildes mit anschließender Darstellung auf dem LCD. Danach wird das Bild für das NN aufbereitet: Umwandlung auf grau, Konvertierung auf 28 x 28 Pixel und Invertierung für MNIST. Nach der Anpassung der Pixelwerte in ein von KI verwertbares Format (meistens Teilung durch 255) erfolgt die Übermittlung des Bildes zur Erkennung an das NN, das sein Ergebnis in der Variablen `fmap` zur Verfügung stellt. Abschließend erscheint die erkannte Ziffer inklusive ihrer Trefferwahrscheinlichkeit auf dem Display. Kurz und knapp: Ziffernerkennung in 25 Befehlen!
Der Praxistest (**Bild 5**) bestätigt die korrekte Arbeitsweise des NNs und mit der Erkennung von einigen Bildern pro Sekunde die Leistungsfähigkeit der KPU.

WEBLINKS

[1] **KI für Einsteiger (1), Elektor Mai/Juni 2020**: www.elektormagazine.de/200023-01

[2] **KI für Einsteiger (2), Elektor Juli/August 2020**: www.elektormagazine.de/200023-B-01

[3] **Projektseite zu diesem Beitrag**: www.elektormagazine.de/200023-C-01

[4] **ESP32-Firmware**: https://github.com/sipeed/Maixduino_esp32_fimware/releases/tag/v1.4.0

[5] **ESP-Demoprogramme**: https://github.com/sipeed/MaixPy_scripts/tree/master/network

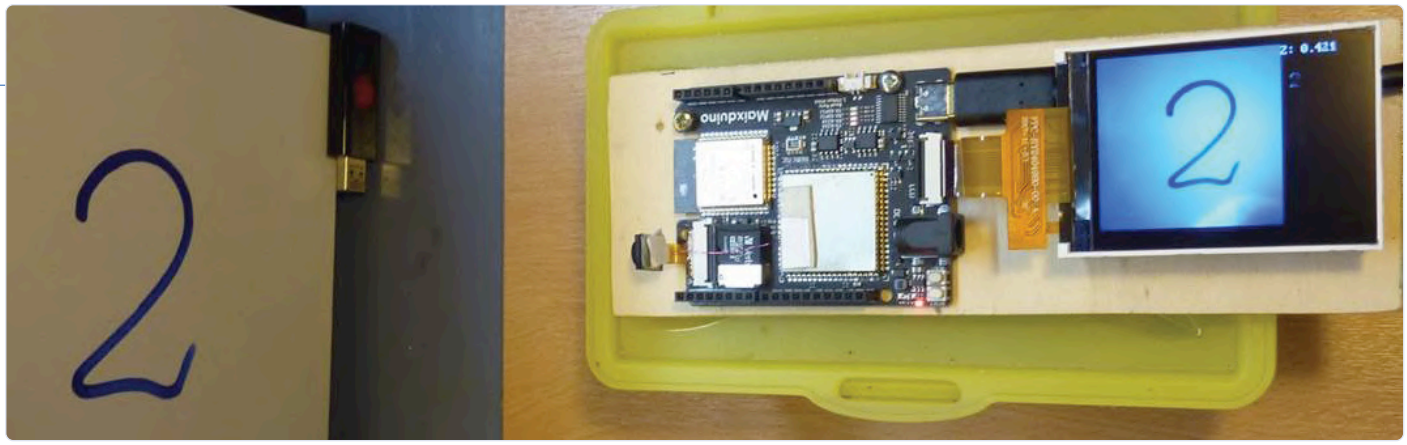


Bild 5. Testaufbau der Ziffernerkennung mit Maixduino.

Und weiter geht´s!

Die leistungsstarke Hardware und die bereits jetzt verfügbare Software-Umgebung zeigen, dass der Maixduino für den Einstieg in die Künstliche Intelligenz gut geeignet ist. Er bietet sich aufgrund des geringen Stromverbrauchs für den Einsatz in mobilen Geräten mit bereits trainierten Neuronalen Netzen an. Will man Entwicklung, Training und Ausführung auf einer Plattform durchführen, eignen sich größere Systeme mit installiertem Linux, z.B. der Nvidia Jetson Nano mit guter Softwareversorgung und hilfsbereitem Support. Zusätzlich kann man sich Neuerscheinungen wie den Rock Pi N10 und andere ansehen, sollte aber vor einem Kauf sicherstellen, dass Software und Dokumentation in guter Qualität verfügbar sind. Und ein Linux-PC mit oder ohne GTX-Grafikkarte tut es für den Einstieg ebenfalls.

Es bewegt sich viel bei der KI, denn die Welle kommt ins Rollen. Ich konnte Ihnen in dieser kleinen Serie lediglich die Spitze des KI-Eisbergs präsentieren. Denn neben dem hier vorgestellten Verfahren des *Supervised Learning*, bei dem ein Neuronales Netz mit bekannten, also „gelabelten" Daten trainiert wird, existieren weitere KI-Lösungen. Beim *Unsupervised Learning* werden in das Neuronale Netz nicht bekannte Daten eingespeist und diese anhand ihrer Charakteristika in Cluster, d.h. einander ähnliche Gruppen, aufgeteilt. Diese Methode wird z.B. für die Analyse von großen Datenbeständen genutzt, um bestimmte Gemeinsamkeiten herauszuarbeiten. Oder bei *Autoencodern* zur Rauschunterdrückung bei der Schrifterkennung.

Und wie können NNs auf bestimmten Gebieten besser sein als der Mensch, z.B. beim Sieg über den besten Go-Spieler, wenn doch die Trainingsdaten von Menschen bereitgestellt werden? Um das zu erreichen kommt eine weitere Spielart der KI zum Einsatz, das *Reinforced Learning*. Hierbei wird das NN durch ein Belohnungssystem motiviert, bei dem mittels der Kombination mehrerer Analyseschritte die größte Belohnung (Ertrag) herausgearbeitet wird. Dieses Verfahren wird u.a. in der Finanzwirtschaft und bei Computerspielen eingesetzt.

Ein Schwachpunkt der KI ist zurzeit die „Black-Box"-Architektur, d.h. nach der Dateneingabe erfolgt eine Klassifikation ohne Begründung. Es bleibt z.B. unklar, warum die KI einer Personalberatung diesen und nicht einen anderen Bewerber vorschlägt. So hat sich beispielsweise herausgestellt, dass ein solches System vorzugsweise Männer auswählte, einfach weil bei den Trainingsdaten mehr Männer als Frauen enthalten waren. An dieser Schwachstelle wird im Rahmen der *Explainable AI*, also der „Erklärenden KI", intensiv geforscht, damit NNs zukünftig ihre Entscheidungen auch begründen und damit für die Nutzer nachvollziehbar machen.

Machen Sie mit und werden Sie Teil dieser Entwicklung, nutzen Sie dabei die Unterstützung durch Videos, Tutorials und Fachliteratur und fügen Sie ihren Projekten eine gute Portion Intelligenz hinzu. Und wenn es manchmal aufwändig und schwierig wird, dann halten Sie es mit Winston Churchill: „Never give up!"

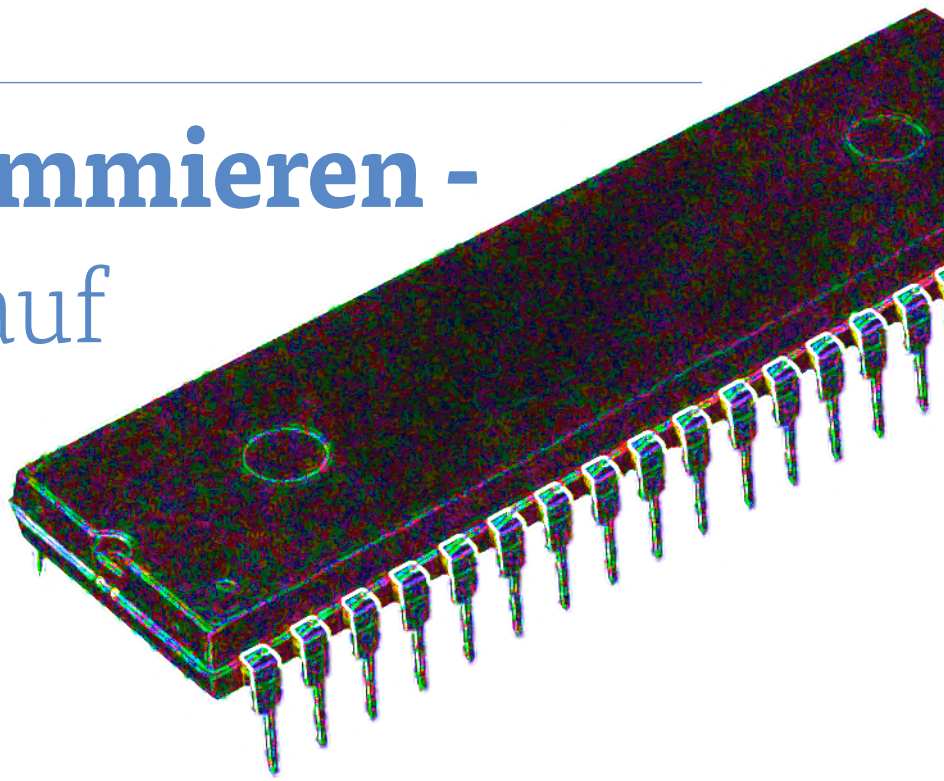
200023-C-01

IM ELEKTOR-STORE

› **Sipeed MAix BiT Kit for RISC-V AI+IoT**
www.elektor.de/sipeed-maix-bit-kit-for-risc-v-ai-iot

PICs programmieren - von der Pike auf

Sinusschwingung mit Assembler



Von **Tam Hanna**

Von 8-Bit-Controllern lernen heißt siegen lernen! Denn wer es versteht, diese Controller von der Pike auf zu programmieren, dem erschließt sich auch die Architektur komplexerer Bausteine. Als Sprache der Wahl bietet sich hier Assembler an. In diesem Artikel wollen wir eine Sinusfunktion über den ADC ausgeben.

Als IDE kommt MPLAB X zum Einsatz, als Zielobjekt ein PIC16F18877. Die Programmierung erfolgt über die ICSP-Schnittstelle, und beim Entwicklungsboard haben Sie die freie Auswahl (auch eine Steckplatine genügt).

Ein virtueller Analog-Digital-Wandler soll mit 32 diskreten Werten pro Schwingung „gefüttert" werden, die wir vorher ausrechnen. Da die in Excel enthaltene Funktion `sin()` aber mit Werten von 0...31 nichts anfangen kann, müssen wir diese zunächst in ein brauchbares Format umwandeln. Die Generierung der Werte erledigen wir in Excel und bauen dabei einen kleinen Fehler ein. Ganz links in der Excel-Tabelle in **Bild 1** finden wir eine gewöhnliche Laufvariable (Schritt). Zur Berechnung dient die Excel-Formel `=A2*((2*PI())/32)`. Die Laufwerte reichen nun von 0 bis inklusive 2*π. In der dritten Sinuswert-Spalte wird mit `=SIN(B2)` der Sinus der Laufwerte berechnet. Leider verläuft diese Sinusfunktion im Wertebereich -1...+1, der DAC arbeitet dagegen nur mit positiven Werten. Wir führen mit `=1+C2` eine Werttransformation durch, um den Wertebereich von -1...+1 nach 0...2 zu verschieben. Die bereinigten Werte werden schließlich mit `D3*(255/2)` in Richtung der DAC-Werte 0...255 transformiert und, um allzu krumme Werte zu vermeiden, mit `ABRUNDEN(E10)` abgerundet.

Tabellen aus dem Datenspeicher

Da sich die Werte nicht ändern, können wir sie im Datenspeicher ablegen. Für Datentabellen wird das RETLW-Mnemonic genutzt, das nach dem RETURN einen bestimmten Wert in W zurücklässt.

Aus technischen Gründen ist es empfehlenswert, dass Tabellen an bestimmten Adressen beginnen. Den Grund werden wir später erkennen - akzeptieren wir im Moment, dass die betreffende Code-Sektion mit einer Zieladresse (hier `0x200`) versehen werden sollte:

```
TABLE_VECT  CODE  0x0200
  dt  127, 152, 176, 198, 217, 233, 245, 252, 255,
   252, 245, 233, 217, 198, 176, 152, 127, 102, 78,
   56, 37, 21, 9, 2, 0, 2, 9, 21, 37, 56, 78, 102
END
```

`dt` nimmt eine Liste von Werten entgegen. Jeder Wert wird zu einem Eintrag im Programmspeicher. Das Programm ist schon kompilierbar, denn der MPLAB-Assembler betrachtet viele kritische Situationen nicht als Fehler. Wer die Ausgabe betrachtet, sieht Warnungen:

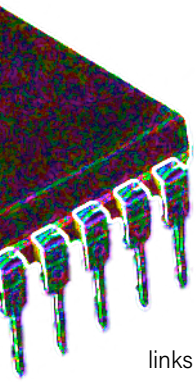
```
Warning[202] C:\USERS\TAMHA\MPLABXPROJECTS\CH6-
   DEMO1.X\NEWPIC_8B_SIMPLE.ASM 72 : Argument out of
   range.  Least significant bits used.
```

Solche Fehler sollte man ernst nehmen. Die Tabelle funktioniert nämlich nicht, aber genau dies liefert erfreulicherweise Möglichkeiten für interessante Experimente.

Exkurs: Disassemblage und Konstanten

Abgesehen von Makros besteht ein direkter Zusammenhang zwischen Mnemonics und Maschinencode. Maschinencode entsteht durch Assemblage der Mnemonics – der umgekehrte Weg heisst Disassemblage.

Doppelklicken Sie im unten rechts eingeblendeten Dashboard-Tab den Eintrag *Usage Symbols Disabled* (**Bild 2**). Die IDE zeigt ein Einstellungsfenster. Markieren Sie die Checkbox *Load Symbols when programming or building for production*, um die Analysewerkzeuge in den Kompilationsprozess einzubinden.



Nach dem Anklicken von *Apply* kompilieren Sie neu - die Speicherverbrauchsanzeigen aktualisieren sich. Zudem steht die Option *Window > Debugging > Output > Disassembly Listing File Project* zur Verfügung, die die Disassemblage-Ansicht zeigt (**Bild 3**).

Die Ausgabe besteht aus sechs Spalten: Die Zahlen ganz links beschreiben die „logische" Adresse des Worts im Programmspeicher des PICs. Die nächste Spalte zeigt die dezimale Beschreibung des Befehls. In der dritten Spalte findet sich die Disassemblage, die aus dem Hex-File entsteht (so werden beispielsweise Konstantennamen nicht aufgelöst). Die vierte Spalte gibt die Zeilennummer in der .asm-Datei an, nach dem Doppelpunkt findet sich die jeweilige Zeile, die für das weiter links befindliche Binärwort verantwortlich zeichnet. Der zu unserer Datentabelle gehörende Code sieht folgendermaßen aus. Wer sich mit Hexadezimalwerten auskennt, sieht aber sofort, dass der Gutteil der verwendeten Zahlen viel zu klein ist:

```
0200  3427  RETLW 0x27     73:     dt  127, 152, 176,
   198, 217, 233, 245, 252, 255, 252, . . .
0201  3452  RETLW 0x52
0202  3476  RETLW 0x76
0203  3498  RETLW 0x98
0204  3417  RETLW 0x17
0205  3433  RETLW 0x33
. . .
```

Die Ursache für das seltsame Verhalten ist eine Eigenart des Assemblers. Er kennt fünf als Radix bezeichnete Arten, um Zahlen anzuschreiben. Statt die Zahlen „einfach so" zu schreiben, müssen vor unsere dezimale Konstanten Punkte gesetzt werden, um den Assembler zu einem korrekten Verhalten zu zwingen.

```
TABLE_VECT  CODE  0x0200 dt  .127, .152, .176, .198,
   .217, .233, .245, .252, .255, .252, .245, .233,
   .217, .198, .176, .152, .127, .102, .78, .56, .37,
   .21, .9, .2, .0, .2, .9, .21, .37, .56, .78, .102
END
```

Tabellenzugriff

Zum Lesen der in den Programmspeicher geschriebenen Informationen muss der RETLW-Block angesprungen werden. Die Return-Anweisungen sorgen dafür, dass das W-Register (Omega) mit dem jeweiligen Parameter befüllt wird. Zum Verständnis der Operation muss die hexadezimale Startadresse 0x0200 in einen binären Wert umgewandelt werden - das Resultat lautet 0000 0000 0000 0010 0000 0000. Bewegliche Aufrufe von Zielen im Programmspeicher erfolgen über das CALLW-Mnemonic, dessen Deklaration **Bild 4** zeigt.
Wegen der Größe des Programmspeichers des PICs von 32768 Worten muss der Adresszeiger 15 Bit lang sein. Omega ist nur 8 Bit lang, weshalb wir zur Vervollständigung einer Adresse zusätzliche Bits aus dem Register PCLATH benötigen.

Zur Vermeidung von Timing-Konflikten greift der PIC auf den Wert von PCLATH nur dann zu, wenn dies von Mnemonics verlangt wird. Wir können den Wert zusammenbauen, bevor wir den Sprungbefehl abfeuern. Schreibvorgänge in PCLATH lassen den Programmzeiger unbeeindruckt. Die Initialisierung des Programms beginnt mit dem Inkrementieren des Laufwerts:

Schritt	Laufwert	Sinuswert	Bereinigt	DAC-Wert	Abgerundet
0	0,0000	0,0000	1,0000	127,5000	127
1	0,1963	0,1951	1,1951	152,3740	152
2	0,3927	0,3827	1,3827	176,2921	176
3	0,5890	0,5556	1,5556	198,3352	198
4	0,7854	0,7071	1,7071	217,6561	217
5	0,9817	0,8315	1,8315	233,5124	233
6	1,1781	0,9239	1,9239	245,2946	245
7	1,3744	0,9808	1,9808	252,5501	252
8	1,5708	1,0000	2,0000	255,0000	255
9	1,7671	0,9808	1,9808	252,5501	252
10	1,9635	0,9239	1,9239	245,2946	245
11	2,1598	0,8315	1,8315	233,5124	233
12	2,3562	0,7071	1,7071	217,6561	217
13	2,5525	0,5556	1,5556	198,3352	198
14	2,7489	0,3827	1,3827	176,2921	176
15	2,9452	0,1951	1,1951	152,3740	152
16	3,1416	0,0000	1,0000	127,5000	127
17	3,3379	-0,1951	0,8049	102,6260	102
18	3,5343	-0,3827	0,6173	78,7079	78
19	3,7306	-0,5556	0,4444	56,6648	56
20	3,9270	-0,7071	0,2929	37,3439	37
21	4,1233	-0,8315	0,1685	21,4876	21
22	4,3197	-0,9239	0,0761	9,7054	9
23	4,5160	-0,9808	0,0192	2,4499	2
24	4,7124	-1,0000	0,0000	0,0000	0
25	4,9087	-0,9808	0,0192	2,4499	2
26	5,1051	-0,9239	0,0761	9,7054	9
27	5,3014	-0,8315	0,1685	21,4876	21
28	5,4978	-0,7071	0,2929	37,3439	37
29	5,6941	-0,5556	0,4444	56,6648	56
30	5,8905	-0,3827	0,6173	78,7079	78
31	6,0868	-0,1951	0,8049	102,6260	102

Bild 1. So ist die Datentabelle einsatzbereit.

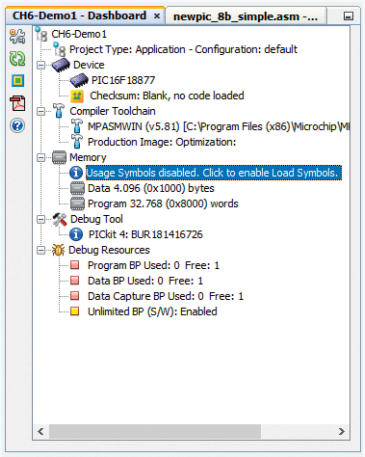


Bild 2. Klicken Sie diese Zeile doppelt an, um das Optionsfenster zu öffnen.

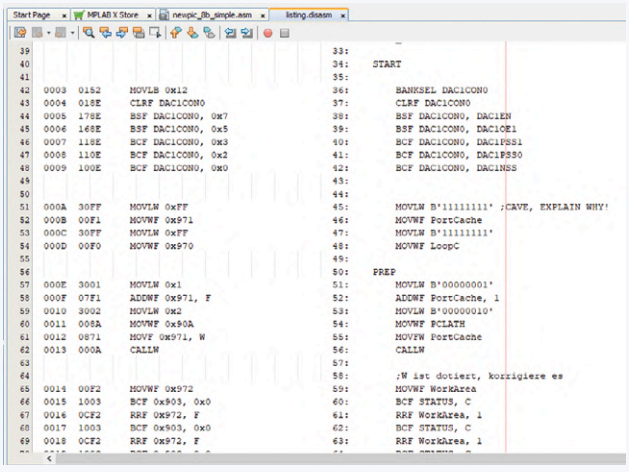


Bild 3. Die Disassemblage-Ansicht informiert über die Inhalte des Maschinencodes.

CALLW	Subroutine Call With W
Syntax:	[*label*] CALLW
Operands:	None
Operation:	(PC) +1 → TOS, (W) → PC<7:0>, (PCLATH<6:0>) → PC<14:8>
Status Affected:	None
Description:	Subroutine call with W. First, the return address (PC + 1) is pushed onto the return stack. Then, the contents of W is loaded into PC<7:0>, and the contents of PCLATH into PC<14:8>. `CALLW` is a 2-cycle instruction.

Bild. 4. Das CallW-Mnemonic berechnet die Adresse mit einem vergleichsweise komplizierten Verfahren.

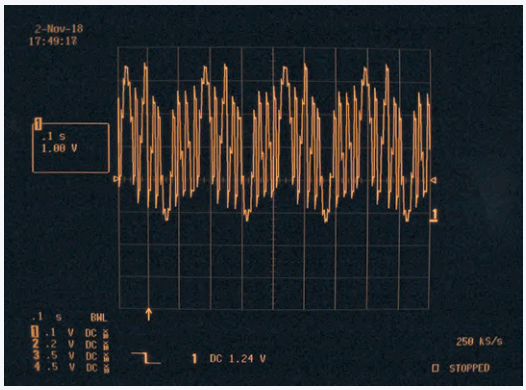


Bild 5. Das sieht wenig nach einer Sinusschwingung aus…

```
WORK
   BANKSEL DAC1CON1
   MOVLW B'00000001'
   ADDWF PortCache, 1
```

Angesichts der obigen Adresse der Tabelle weiß man, wie der obere Teil des Binärworts aussieht. Man muss über das Omega-Register in Richtung von PCLATH wandern:

```
   MOVLW B'00000010'
   MOVWF PCLATH
```

Damit sind wir für den eigentlichen Sprung bereit. CALLW erwartet in Omega den Zielwert. Nach dem Aufruf finden wir in Omega den Rückgabewert aus der Tabelle, der in Richtung des DACs wandert:

```
   MOVFW PortCache
   CALLW
   MOVWF DAC1CON1
   CALL WAIT
```

Wer das Programm im vorliegenden Zustand ausführt, erlebt eine Überraschung. Der PIC „dreht durch", bis MPLAB die Ausführung irgendwann anhält.
Die Ursache für dieses Verhalten ist, dass unsere Tabelle nur 32 Werte aufweist, wir aber 255 Werte abarbeiten. Die darauffolgenden Speicherstellen sind beliebige Instruktionen, die aus dem letzten Programmierdurchlauf liegengeblieben sind. Im Rahmen des Anwachsens des Sprungwerts finden wir uns im „unbeschriebenen" Bereich wieder.
Der folgende Code beschränkt den Wert von `PortCache` mit CLRF:

```
   MOVFW PortCache
   SUBLW .31
   BTFSC STATUS, Z
   CLRF PortCache
```

Zu guter Letzt springen wir nach oben, um den nächsten Durchlauf zu starten:

```
   GOTO WORK        ; loop forever
```

An dieser Stelle können wir unser Programm ausführen. Doch die Wellenform in **Bild 5** hat wenig mit einer Sinusschwingung gemein. Der Grund: Die in das DAC-Register geschriebenen Werte laufen von null bis 255. Der DAC kann allerdings nur mit Werten von null bis 31 etwas anfangen, was zu Chaos führt.

Tabellen aus dem RAM

Zwar könnten wir zu in Excel zurückkehren, Faktoren anpassen und eine lauffähige Tabelle erzeugen, doch diese Vorgehensweise ist nicht ratsam. Wir wollen die Anpassung der Parameter lieber direkt am Controller vorführen.
Das „Linksschieben" eines Werts multipliziert diesen mit dem Faktor zwei, eine Rechtsschiebeoperation um ein Bit entspricht einer Division durch zwei. Da wir von 256 auf 32 kommen müssen, entspricht dies einer Division durch acht. Das lässt sich auch als Folge von drei Divisionen durch zwei beschreiben.
Beim „Kopieren" der Werte müssen wir Zieladressen im Arbeitsspeicher berechnen. Dazu werfen wir einen Blick auf die Core-Register. PCLATH ist bekannt, nun interessieren uns INDF und FSR.

Unser PIC besitzt zwei Paare aus Pointer-und Adressregister – ein Entgegenkommen an Entwickler, die an zwei Stellen gleichzeitig arbeiten. Die INDF-Register (steht für INDirect File) sind nicht physikalisch implementiert. Sie verweisen auf die Adresse, die in den zu ihnen gehörenden FSRs (File Select Register) eingestellt wird. Stellen Sie dort eine Adresse im Arbeitsspeicher ein, so können Sie über INDF sowohl lesen als auch schreiben. Verweisen die FSR-Register in den Programmspeicher, so können sie (normalerweise) nur lesen.
Eine Besonderheit des PIC ist, dass die Auswahl zwischen Programm- und Arbeitsspeicher über das siebente Bit des oberen Adressregisters erfolgt. Ist es gesetzt, so wird der Rest der Adresse auf den Programmspeicher bezogen, wenn nicht, so adressiert es den Datenspeicher.
Wir beginnen abermals mit dem Anlegen einer Variablen. Diesmal sind insgesamt 32 Byte Speicher erforderlich - eine Allokationsgröße, die den „geteilten" Speicherbereich der MCU überfordern würde.

Wir greifen uns Speicher in einer von Assembler auszuwählenden Bank und arbeiten mit relativer Adressierung – MPASM gewährt wegen des Fehlens von Zusatzparametern freie Wahl bei der Platzierung von `DataBuffer`:

```
udata_shr
   LoopC res 1
   PortCache res 1
udata
   DataBuffer res 32
```

Fraglich bleiben die hohen und die niedrigen Teile der Adresse. Erfreulicherweise macht uns der Linker mit zwei wenig bekannten Operatoren die Arbeit einfach:

```
START
  MOVLW high DataBuffer
  MOVLW low DataBuffer
```

Mit diesem Wissen können wir Informationen aus dem Programmspeicher in den Arbeitsspeicher kopieren. Da Schiebebefehle nicht direkt in W arbeiten, ist eine zusätzliche Variable erforderlich:

```
udata_shr
   LoopC res 1
   PortCache res 1
   WorkArea res 1
```

Bei der Arbeit mit Datentabellen ist die Anfangsbedingung wichtig. Wir laden PortCache mit 1111.1111 vor, weil die Schleife *vor* ihrer Verwendung inkrementiert. Der erste Durchlauf erfolgt so mit 0 - würden wir 0 eingeben, so schriebe der erste Durchlauf in die Speicherstelle 1:

```
START
. . .
   MOVLW B'11111111'
   MOVWF PortCache
   MOVLW B'11111111'
   MOVWF LoopC
```

Unser Programm besitzt zwei Schleifen. Die PREP-Schleife ist für die Vorbereitung der Datentabelle verantwortlich, während `Work` die Werte in Richtung des DAC kopiert. PREP beginnt mit einem Aufruf der Tabelle:

```
PREP
   MOVLW B'00000001'
   ADDWF PortCache, 1
   MOVLW B'00000010'
   MOVWF PCLATH
   MOVFW PortCache
   CALLW
```

Danach müssen wir mit drei Aufrufen von RRF Werte verschieben. Da die Schiebeoperation nur in einem F-Register erfolgen kann, müssen wir den Wert vorher in eine Ablage schreiben:

```
   MOVWF WorkArea
   BCF STATUS, Z
   RRF WorkArea, 1
   BCF STATUS, Z
   RRF WorkArea, 1
   BCF STATUS, Z
   RRF WorkArea, 1
```

Um das Einsammeln von Fehlern aus dem Carry-Bit zu vermeiden, wird vor jedem RRF-Durchlauf `BCF STATUS, Z` aufgerufen. In diesem Listing befinden sich noch zwei kleine Fehler, und zwar beabsichtigt, denn die Behebung ist interessant!
Danach müssen wir INDF auf die richtige Speicherstelle zeigen lassen. Dazu werden die Register FSR0H und FSR0L mit Adressdaten beladen. Das H-Register (High) nimmt dabei den höherwertigeren Teil auf, das L-Register (Low) den niederen Teil:

```
   MOVLW high DataBuffer
   MOVWF FSR0H
   MOVLW low DataBuffer
   MOVWF FSR0L
```

INDF0 zeigt nun auf den Beginn des Speicherbereichs. Wir müssen den Offset hinzu addieren, der die jeweilige Speicherstelle identifiziert. Dabei ist nicht sicher, dass der Beginn des Feldes am Beginn einer Seite liegt. Kommt es beim Addieren des Offsets im L-Teil zu einem Überlauf, so würde der H-Teil dies nicht mitbekommen. Als Lösung dieses Problems wird der Wert des C-Bits überprüft und der Wert von FSR0H inkrementiert, wenn ein Überlauf vorliegt:

```
   MOVFW PortCache
   CLRC
   ADDWF FSR0L
   BTFSC STATUS, C
   INCF FSR0H
```

Das Kommando CLRC (Clear Carry) ist ein Makro, das das C-Bit im Statusregister löscht. Damit ist INDF0 korrekt konfiguriert. Wir müssen den in der Work Area zwischengespeicherten Wert laden und herausschreiben:

```
   MOVFW WorkArea
   MOVLW INDF0
```

Zu guter Letzt müssen wir sicherstellen, dass die Schleife weiterläuft.

```
   MOVFW PortCache
   SUBLW .31
   BTFSS STATUS, Z
   GOTO PREP
   CLRF PortCache
```

Da der Code recht komplex ist, wäre es wünschenswert, wenn wir uns von der Korrektheit der Ausgabe überzeugen könnten.

Fehlersuche mit Assembler

Das Platzieren von Breakpoints ist theoretisch einfach. Klicken Sie in der IDE auf die Zeilennummern, um die roten Stopp-Symbole zu platzieren. Da aber der Mikrocontroller nur einen einzigen Hardware-Breakpoint verwalten kann, erscheint eine Fehlermeldung.
MPLAB verbraucht Debuggingressourcen, um das stufenweise Ausführen zu ermöglichen. Die Emulation von Breakpoints in der Software ist im Moment nicht erwünscht, weshalb wir die Fehlermeldung mit einem Klick auf *No* beseitigen. Da der PIC Software-Breakpoints unterstützt, können Sie dies bejahen (*Yes*): Sobald Sie mehr als einen Breakpoint in der Assemblerdatei platzieren, aktiviert Microchip die Funktion automatisch.

Da unser PIC nur einen Hardware-Breakpoint unterstützt, klicken Sie in der Toolbar auf den nach unten zeigenden Pfeil neben dem Debugger-Symbol und wählen Sie die Option *Debug main project*. MPLAB öffnet ein Disassemblagefenster, das wir sofort wieder schließen. Nach

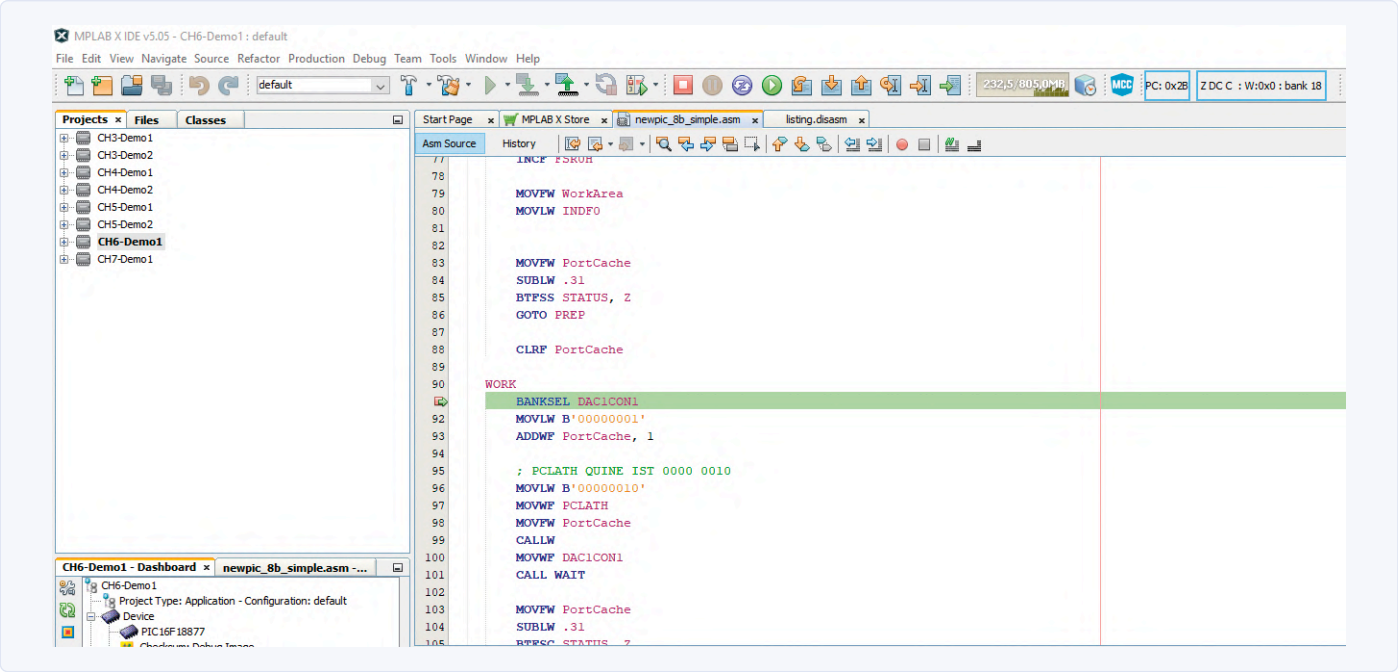


Bild 6. Der Debugger hat die Programmausführung unterbrochen.

dem Erreichen des Haltepunkts präsentiert sich die IDE wie in **Bild 6**. Die grün hinterlegte Zeile mit dem Pfeil links ist die aktuelle Instruktion. Stopp- und Sprungsymbole in der Toolbar erlauben die Interaktion mit dem Programmzustand. *Window → Target Memory Views → File Registers* öffnet ein Fenster, das den Speicher des PIC zeigt. Legen Sie Cursor wie Mauszeiger über die Deklaration von DataBuffer, um ein Tooltip-Fenster mit der Adresse des ersten Bytes und seinem Wert zu öffnen. Auf der Workstation des Autors lautete die Position `0xD20`. Bequemer ist es, im Fenster *File Registers* auf den nach unten zeigenden blauen Pfeil (*GoTo*) zu klicken. Wählen Sie im *GoTo-What*-Feld *Symbol* aus, und wählen Sie *DataBuffer*. Schließen Sie das Popup nach dem Anklicken des *Go To*-Buttons, um sich am in **Bild 7** gezeigten Resultat zu erfreuen. Das rot hervorgehobene Kästchen ist das erste Byte der Allokation. Es ist offensichtlich, dass das vorliegende Programm nicht funktioniert, denn Werte wie FC liegen außerhalb des erlaubten Wertebereichs.

Zur Suche ist es empfehlenswert, den Speicherbereich mit einem leicht erkennbaren Pattern auszufüllen. Ein guter Versuch würde das Literal 1111.1111 in das indirekte Register schreiben:

```
MOVFW PortCache
CLRZ
ADDWF FSR0L
BTFSC STATUS, C
INCF FSR0H
MOVFW B'11111111'
MOVLW INDF0
```

Wer den Wertebereich im Debugger öffnet, sieht eine Sequenz desselben Werts. In den meisten Fällen dürfte er nicht FF lauten. Der Code weist einen kleinen Fehler auf. Wir nutzen den Befehl MOVLW, der den Wert einer Speicherstelle ins Omega-Register lädt:

```
MOVFW B'11111111'
MOVLW INDF0
```

Aus Sicht von MPLAB ist das Literal INDF0 eine Zahl: Nach der Kompilation sind auch Werte wie PORTA nur Zahlen. Unser Programm kopiert also die Adresse des Registers in alle 32 Speicherstellen. Trotzdem sind wir einen Schritt weiter, da wir schon das Ansprechen der Adressdaten verifiziert haben. Eine korrigierte Version des Programms sieht folgendermaßen aus:

```
MOVLW B'11111111'
MOVWF INDF0
```

Die Berechnung der Speicheradressen funktioniert, wir können den eigentlichen Rechenfehler ausmerzen. Das erste Problem war die Verwendung von MOVLW statt MOVWF, was das Schreiben in INDF außer Gefecht setzte:

```
MOVFW WorkArea
MOVWF INDF0
```

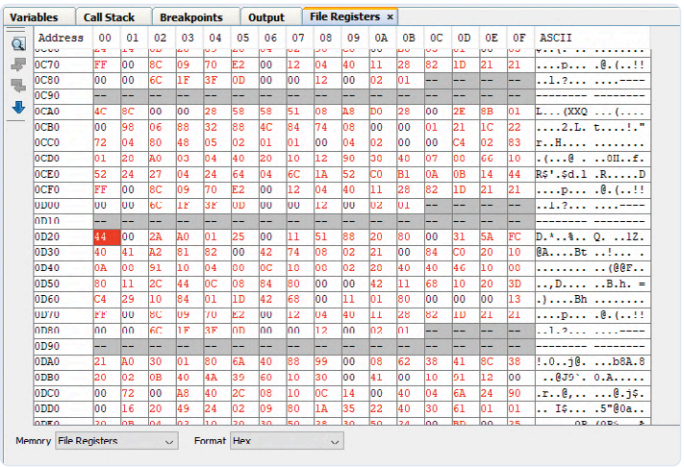


Bild 7. Hier ist etwas faul.

```
MOVFW PortCache
SUBLW .31
BTFSS STATUS, Z
GOTO PREP
```

Bei Betrachtung der Ausgabe stellen wir fest, dass bei sehr kleinen Werten Einsen auftauchen. Die Ursache dieses Problems ist, dass das RRF-Mnemonic mit dem Carry-Bit arbeitet. Unser Code hat aber bisher das Z-Bit gelöscht, was zu dieser kleinen Änderung führt:

```
MOVWF WorkArea
BCF STATUS, C
RRF WorkArea, 1
BCF STATUS, C
RRF WorkArea, 1
BCF STATUS, C
RRF WorkArea, 1
```

Damit ist das Programm einsatzbereit, die Sinustabelle erscheint im Debuggerfenster. Zur Fertigstellung müssen wir in der Arbeitsschleife dafür sorgen, dass die Werte aus dem Datenspeicher entnommen werden. Dazu ist eine Inkrementierung der Laufvariablen erforderlich, um einen fortlaufenden Index zu erzeugen:

```
WORK
   BANKSEL DAC1CON1
   MOVLW B'00000001'
   ADDWF PortCache, 1
```

Die indirekte Adressierung ist zum Lesen und zum Schreiben geeignet. Wir laden die beiden Teile der Adresse des Puffers in FSR0H und FSR0L. Danach addieren wir den Fehlbetrag und prüfen auf einen Überlauf: Tritt ein Überlauf auf, so inkrementieren wir das höhere Register:

```
MOVLW high DataBuffer
MOVWF FSR0H
MOVLW low DataBuffer
MOVWF FSR0L
MOVFW PortCache
CLRZ
ADDWF FSR0L
BTFSC STATUS, C
INCF FSR0H
```

Neu ist, dass wir aus dem Register INDF0 lesen. Der Wert wandert in das Ausgaberegister des DAC:

```
MOVFW INDF0
MOVWF DAC1CON1
CALL WAIT
```

Der Rest des Programms ist eine gewöhnliche Schleife, die unter anderem die Inkrementierung vornimmt:

```
MOVFW PortCache
SUBLW .31
BTFSC STATUS, Z
CLRF PortCache
GOTO WORK
```

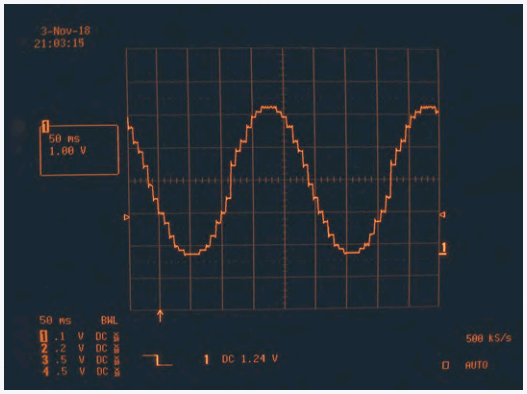


Bild 8. Unser Sinus erscheint (im Rahmen der Möglichkeiten des DACs) am Oszilloskop.

Damit ist das Programm fertig – die Ausgabe präsentiert sich wie in **Bild 8**.

Fazit

Der Artikel zeigt, dass man auf Achtbittern interessante Experimente mit Assemblerbefehlen durchführen kann. Mehr dazu finden Sie unter anderem in meinem neuen Lehrbuch „Mikrocontroller-Basics mit PIC“. Der Autor freut sich über Feedback!

200154-01

IKEA-Hack

Tuning einer preiswerten Ikea-Lampe mit NeoPixel-LEDs und WLAN

Von **Hans Henrik Skovgaard**

Man kann fernsteuerbare Lampen mit eingebauter Anbindung ans heimische WLAN mittlerweile von etlichen Herstellern zu vertretbaren Preisen kaufen. Man kann sich so eine Lampe aber noch preiswerter selber bauen. Interessanter und individueller ist das allemal, wie dieses Projekt beweist!

Als Ingenieur ist es für mich nicht ungewöhnlich, dass ich mit unvorhergesehenen Problemen/Herausforderungen konfrontiert bin, mit denen ich irgendwie umgehen muss. Das passierte mir aber vor einiger Zeit auch privat, denn mein Sohn überreichte mir ein „NeoPixel Jewel 7" [1] von Adafruit, das er aus Versehen für sein Universitätsprojekt gekauft hatte. Dabei handelt es sich um die in **Bild 1** zu sehende kleine, runde Platine, die mit sieben „intelligenten" NeoPixel-RBG-LEDs im SMD-Format bestückt ist. Die große Frage: Was könnte ich damit Nützliches machen?

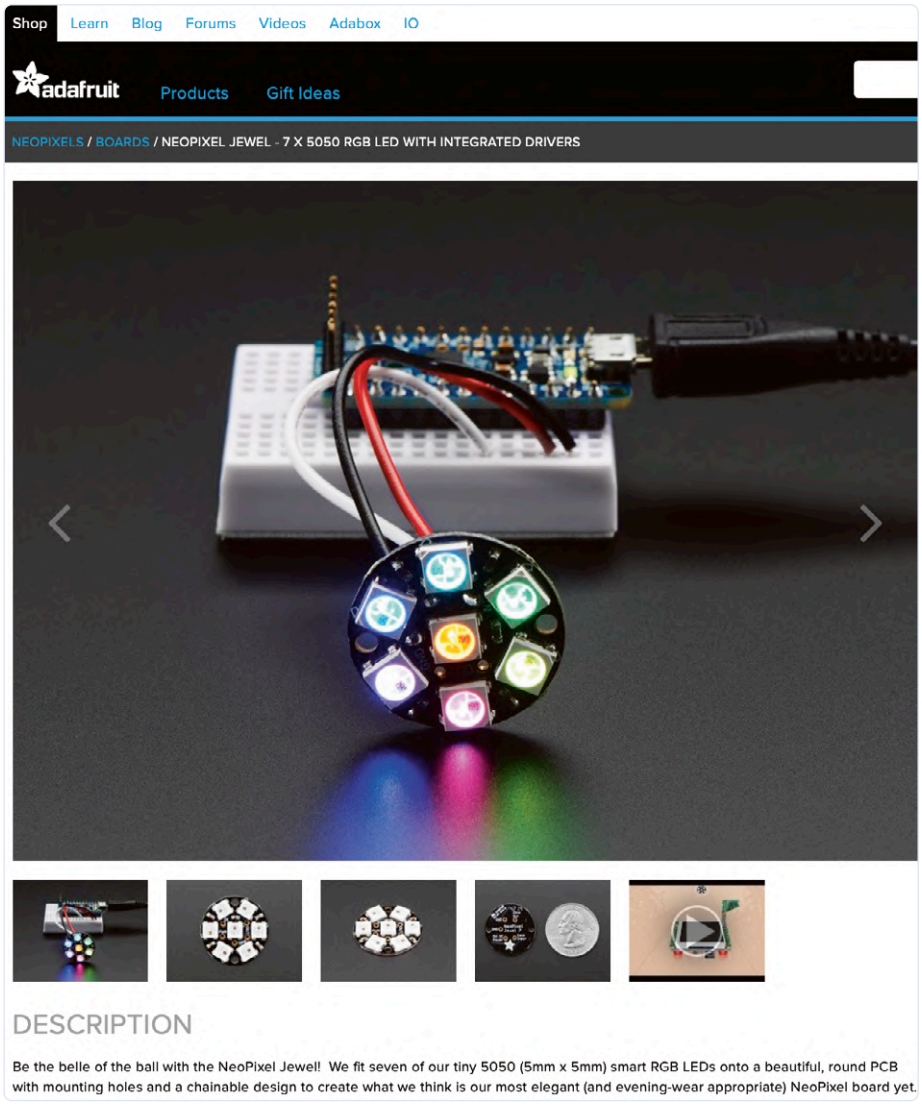


Bild 1. NeoPixel Jewel 7, angeschlossen an ein Arduino-Board [1].

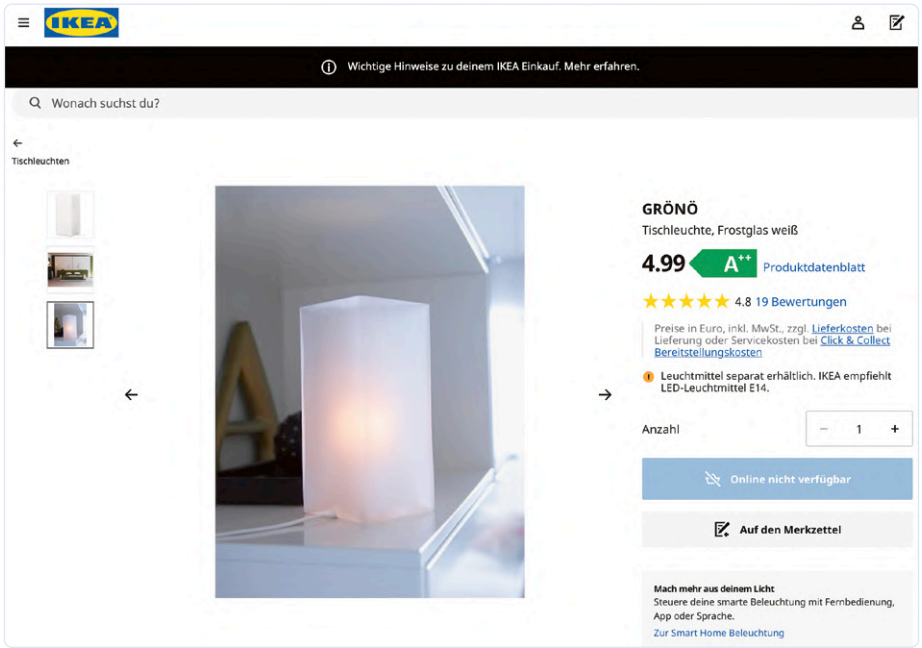


Bild 2. Die preiswerte Lampe „Gröno" [4].

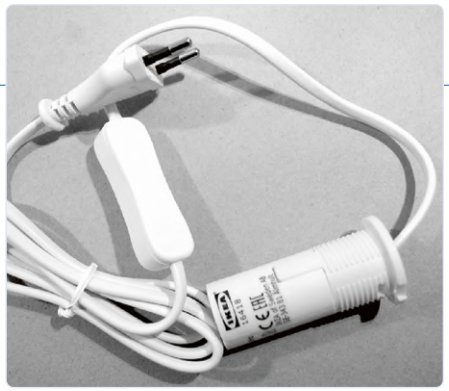


Bild 3. Das noch unmodifizierte Innenleben der IKEA-Lampe.

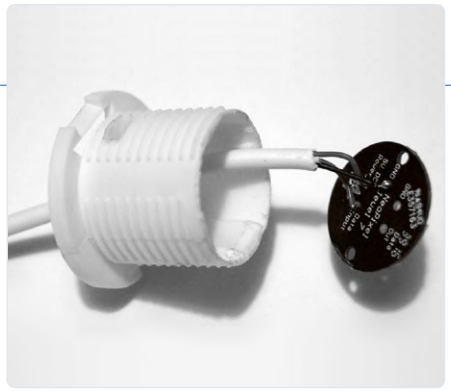


Bild 4. Die LED-Platine kommt oben auf die demontierte Lampenfassung.

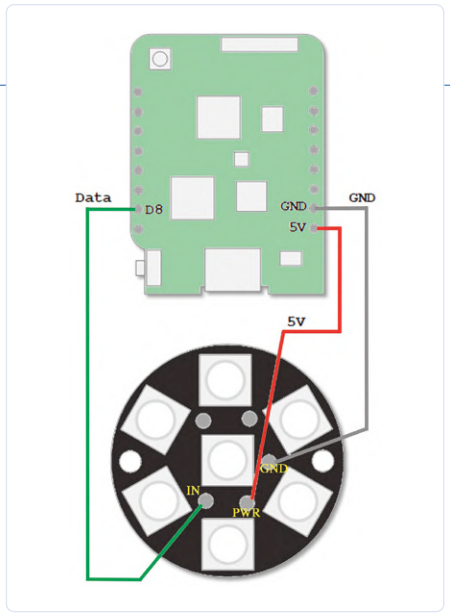


Bild 5. Die „Schaltung" meines Hacks ist sehr simpel: Mikrocontroller-Board + LED-Platine.

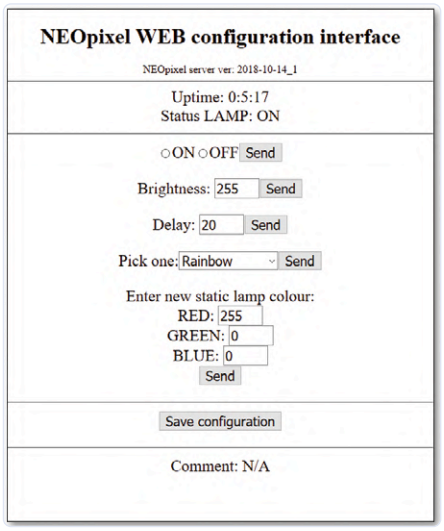


Bild 6. Die einfache Konfigurationsoberfläche des Webservers.

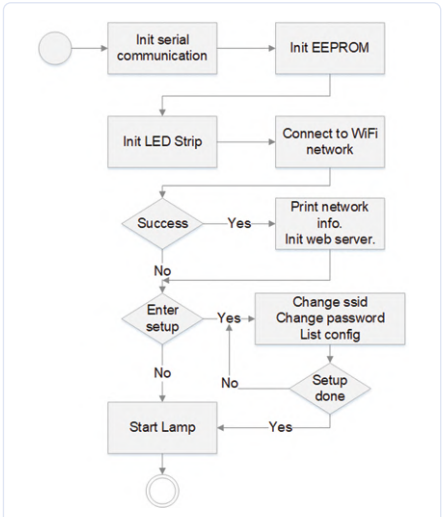


Bild 7. Ablaufdiagram für den Setup-Prozess.

NeoPixel + Arduino

Die Firma WorldSemi hat schon eine ganze Zeit RGB-LEDs mit integriertem Controller im Programm, die einfach und effektiv seriell angesteuert werden können. Man findet Massen an Angeboten aus Fernost, wenn man die Begriffe „NeoPixel" oder „WS2812B" bei eBay eingibt. Da ich bereits eine ganze Reihe von Arduino-Projekten entwickelt hatte, war es nicht schwer, diese LEDs mit Hilfe der Library von Adafruit [2] zum Leuchten zu bringen. Resultat meiner Experimente war eine Reihe von weißen Punkten in meinem Blickfeld, denn die LEDs sind wirklich sehr hell. Seien Sie also vorsichtiger als ich, wenn Sie solche LEDs ausprobieren!

Arduino ist das Eine. Wenn man aber gerne eine Internet-Anbindung haben will, nimmt man doch besser etwas anderes als Basis. Zur gleichen Zeit beschäftigte ich mich damals mit der ESP8266-MCU von Espressif in Form des Boards „D1 mini Pro" von Wemos [3]. Für mich ergab dieses Board mit der NeoPixel-Platine eine gelungene Kombination. Zur Erhöhung des WAF (Wife Acceptance Factor ;-)) musste ich dieses Stück Technik allerdings in etwas Schönem (aber nicht zu Teurem) verkleiden.

Lampen-Tuning

Meine Lösung bestand darin, eine fertige Lampe von IKEA namens Gröno [4] quasi als Gehäuse zu verwenden. Ich bin zwar nicht auf die Produkte von IKEA fixiert, aber diese in **Bild 2** zu sehende Lampe kostet in Dänemark nur 6,50 € (in D sogar nur 4,99 €). Passiert der Lampe aufgrund meiner Umbaumaßnahmen eine destruktive Katastrophe, würde mich das vermutlich nicht ruinieren – ein nicht ganz unwichtiger Aspekt beim Hacken von Dingen. **Bild 3** zeigt „die Elektrik" bzw. das Innenleben der Lampe: Eine E14-Lampenfassung nebst Netzkabel und Schnurschalter. Da drei Adern zum Anschluss der NeoPixel-Platine ausreichen, wurde einfach die Lampenfassung demontiert und die Platine obendrauf geklebt sowie die Zuleitung durch ein normales dreiadriges Kabel ersetzt (**Bild 4**).
Die Schaltung der einzubauenden Elektronik von **Bild 5** ist mehr als einfach. Neben der Neopixel-Platine und einem ESP8266 benötigt man nur noch 5 V mit 500 mA Belastbarkeit via USB. Der Rest ist reine Software. Die Neopixel-Platine beherbergt sieben RGB-LEDs mit integriertem Controller im SMD-Format 5050. Man kann diese LEDs auch einzeln unter der Bezeichnung WS2812B [5] kaufen und damit fast beliebig komplexe Anordnungen realisieren. Sie benötigen bis zu 60 mA Strom und leuchten maximal mit rund 20 Lumen. Man kann sie in Serie schalten und trotzdem einzeln adressieren, indem man die RGB-Helligkeitsdaten an einem Ende seriell in die Kette hineinschiebt. Die Ansteuerung ist also denkbar einfach.

Software

Ein wichtiger Aspekt des Boards D1 mini Pro von Wemos ist, dass es mit der Arduino-IDE programmiert werden kann. Die Verwendung der Arduino-IDE hat Vor- und Nachteile – besonders relevant für mich ist aber, dass man damit Projekte schnell und einfach zum Laufen bringen kann. Die Erstellung von Software für diesen IKEA-Hack braucht folgende Vorarbeiten:

- Installation der Arduino-IDE [6].
- Hinzufügen der Unterstützung für die ESP8266-MCU [7].
- Installation der Bibliothek für Adafruit-NeoPixel (in der Arduino-IDE) [2].
- Installation des Arduino ESP8266-Dateisystem-Uploaders (für SPI-Flash) [8].

Die Software [12] mag angesichts der einfachen Aufgabe des Steuerns dieser LEDs etwas aufwändig erscheinen, doch der Grund für die Komplexität liegt darin, dass ein ESP8266 über WLAN-Unterstützung verfügt. Und wenn wir diese nutzen wollen, benötigen wir schon etwas Code. Die Software hat folgende Aufgaben:

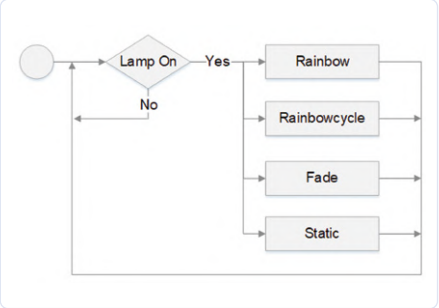


Bild 8. Ablaufdiagram für die Modus-Auswahl.

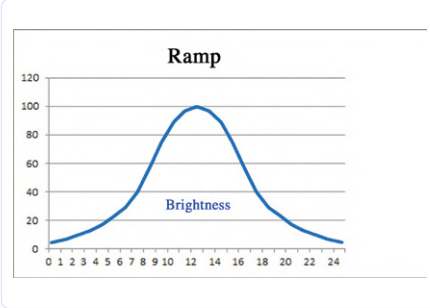


Bild 9. Optimierter Helligkeitsverlauf beim Fading.

- Erzeugung spezieller Lichteffekte.
- Internet-Anbindung per WLAN.
- Webserver zur Konfiguration der Lampe.

Die Lampe kann auch „offline" bzw. im Standalone-Modus betrieben werden, in dem sie z.B. die Regenbogenfarben in einem definierten Tempo durchläuft (z.B. 20 ms für jede der 8x8x8 Farbstufen). Wenn man die Lampe während des Betriebs konfigurieren möchte, muss man aber zwingend WLAN aktivieren, damit die Neopixel-Lampe eine Verbindung zum lokalen Netz herstellen kann. Die nötigen Parameter für WLAN kann man entweder fix im Sourcecode mit anschließender Neukompilierung der Wemos-Firmware oder beim „Booten" der Lampe einstellen. Hat die Lampe Verbindung mit dem WLAN, ist folgendes möglich:

- Ein- und Ausschalten der Lampe.
- Einstellung der Helligkeit.
- Änderung der Verzögerungszeit zwischen den Farbzyklen.
- Auswahl der Farbzyklen (aktuell: Regenbogen, Regenbogenzyklen, statisch, Kerzenflackern und Fading).
- Statische Einstellung der Farbe.

Die Einstellungen kann man über eine (sehr) einfache Web-Oberfläche (siehe **Bild 6**) vornehmen. Der HTML-Code für die Webseite wird mit der Funktion

```
getPage(string str)
```

generiert.

Das alles ist in der Software fest vorgegeben und nicht leicht zu ändern - aber es funktioniert. Die Web-Konfigurationsschnittstelle aktualisiert die NeoPixel-Lampe per HTTP-POST-Requests. Das prinzipielle Vorgehen ist in meinem Buch „IoT Home Hacks with ESP8266" [9] beschrieben. Insgesamt ist die Software ähnlich wie bei anderen Projekten von mir gestrickt. **Bild 7** zeigt den Ablaufplan der Software beim Setup und **Bild 8** beim Betrieb bzw. der Modus-Auswahl.
Recht bald entdeckte ich, dass das Fading (An- und Abschwellen der Helligkeit) zusätzliche Überlegungen erforderte. Wenn man das nämlich mit einer linearen Rampe löst, sieht es nicht gut aus. Wie in **Bild 9** dargestellt, ist eine eher sanfte, an eine Sinuskurve angenäherte Rampe erforderlich. Dieser Helligkeitsverlauf wurde in Excel simuliert und als Array `byte fadeInterpolation[]` implementiert – er kann bei Bedarf beliebig verändert werden. Der Artikel „LED-Dimmer" in Elektor 9/2018 [10] beschreibt die Zusammenhänge. Die zugrundeliegende Theorie basiert auf dem psycho-physikalischen Weber-Fechner-Gesetz [11].

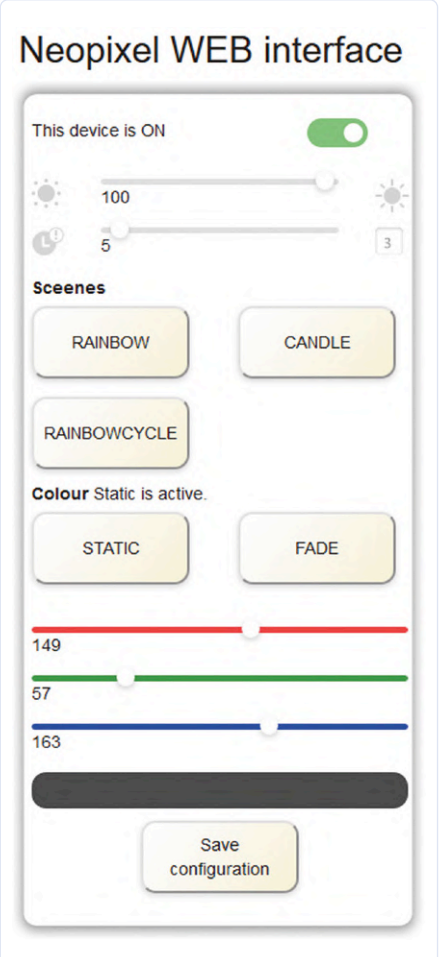


Bild 10. Entwurf des verbesserten Web-Interfaces.

Alltagsbetrieb

Sobald das mit der Firmware versehene Wemos-Board an eine 5-V-Stromversorgung angeschlossen wird, bootet es und durchläuft dabei einen „Einschaltzyklus".

WLAN-Verbindung (blaues Licht)
Zunächst wird versucht, anhand der eingestellten Konfiguration eine Verbindung mit dem WLAN herzustellen. Während dieses Vorgangs blinken die LEDs der NeoPixel-Platine nacheinander blau.

Warten (rotes Licht)
Nach einer erfolgreichen oder fehlgeschlagenen WLAN-Verbindung wartet die Elektronik auf eine Benutzerinteraktion am USB-Anschluss. Während dieses Vorgangs blinken die LEDs nacheinander rot auf. Wenn man mit der Konfiguration der NeoPixel-Lampe beginnt, erlöschen alle LEDs. Wenn zehn Sekunden lang kein Benutzer mit der Lampe interagiert, startet sie die vorkonfigurierte Beleuchtung. Nach dem Einschalten über

SHOPPING LIST

- **Buch „IoT Home Hacks with ESP8266"**
  www.elektor.de/iot-home-hacks-with-esp8266-e-book
- **WeMos D1 mini Pro**
  www.elektor.de/wemos-d1-mini-pro-esp8266-based-wifi-module
- **ESP8266-Webserver für NeoPixel-LEDs**
  www.elektor.de/esp8266-serveur-led-rgb-bare-pcb-160487-1

eine aktive WLAN-Verbindung ist es möglich, den Lichteffekt bzw. den Betriebsmodus der Neopixel-Lampe zu verändern.

Neues Web-Interface

Nachdem ich so weit gekommen war, zeigte ich das Ergebnis meinem Sohn. Er hat einen Master-Abschluss in Interaktionsdesign und zeigte sich von meiner Gestaltung der Web-Schnittstelle nicht sehr beeindruckt. Also entwickelte er ein neues Design (**Bild 10**), das nicht in einfach „hart" per Code realisiert werden konnte, da die Pflege zu schwierig geworden wäre.

Stattdessen wurde ein HTML-, CSS- und Javascript-Design erstellt. An dieser Stelle kommt das SPI-Flash-Datei-System ins Spiel. Eine ESP8266-MCU stellt immerhin 14 MB per Datei-System SPIFFS nutzbaren Flash-Speicher zur Verfügung. In meinem Buch [9] findet man die Informationen zum:

- Hochladen von Dateien ins SPIFFS.
- Ablageort der Dateien auf dem PC, um sie hochladen zu können.
- Installation der nötigen Software für die Arduino-IDE.

Name	Date modified	Type	Size
data	25/12/2019 22.54	File folder	
New_IKEAHack_20191221_1.ino	08/02/2020 21.38	Arduino file	48 KB

Bild 11. So sollte das Arduino-Software-Verzeichnis aussehen.

Name	Date modified	Type	Size
brightnessHigh.svg	19/12/2019 15.39	SVG Document	3 KB
brightnessLow.svg	19/12/2019 15.40	SVG Document	5 KB
delayLeft.svg	20/12/2019 22.16	SVG Document	2 KB
delayRight.svg	20/12/2019 22.17	SVG Document	7 KB
main_background.svg	25/12/2019 22.07	SVG Document	13 KB
NEOPixel_new_20191222.css	25/12/2019 22.52	Cascading Style Sheet Doc	2 KB
NEOPixel_new_20191222.html	18/01/2020 22.32	Firefox HTML Document	8 KB
NEOPixel_new_20191222.js	26/12/2019 23.29	JavaScript File	4 KB
sliderCircle.svg	18/12/2019 20.19	SVG Document	2 KB

*Bild 12. Der Inhalt des Verzeichnisses **Data**.*

Wenn dies korrekt durchgeführt wurde, sollte Ihr Arduino-Software-Verzeichnis wie in **Bild 11** aussehen. In **Bild 12** sind die im Ordner *Data* enthaltenen Dateien für das Web-Interface aufgeführt. Die aktualisierte Software finden Sie zum kostenlosen Download auf der Elektor-Webseite zu diesem Artikel [13]. Wenn Sie in das Verzeichnis *Data* schauen, werden Sie feststellen, dass keine Datei *NEOPixel_new_20191222_load.js* enthalten ist, denn diese Funktionen sind fest in der ESP8266-Software kodiert. Auf diese Weise kann die ESP8266-MCU das Web-Interface beim Einschalten mit gespeicherten EEPROM-Werten konfigurieren.

Das Web-Interface ist von vorneherein für die Hochkant-Darstellung der Bildschirme von Smartphones konzipiert. Da die NeoPixel-Lampe durch den Empfang von HTTP-POST-Requests konfiguriert werden kann, ist es auch möglich, die Lampe über die Heimautomatisierungslösung OpenHAB zu steuern - aber das ist eine andere Geschichte. Diese bescheidene NeoPixel-Jewel-Platine hat mich in viele verschiedene neue Bereiche geführt, die mir vor Beginn des Projekts nicht bekannt waren. Und das hat sich gelohnt, denn dieser IKEA-Hack bzw. diese Neo-Pixel-Lampe ist seit mehr als einem Jahr fester Bestandteil meines Zuhauses.

200165-01

WEBLINKS

[1] **NeoPixel Jewel 7**: http://www.adafruit.com/product/2226
[2] **Anleitung Library**: https://learn.adafruit.com/adafruit-neopixel-uberguide/arduino-library-use
[3] **Wemos D1 mini Pro**: http://www.wemos.cc/en/latest/d1/d1_mini_pro.html
[4] **Lampe Gröno von Ikea**: http://www.ikea.com/de/de/p/groenoe-tischleuchte-frostglas-weiss-20373225/
[5] **Datenblatt WS2812B**: http://www.world-semi.com/DownLoadFile/111
[6] **Arduino-IDE**: http://www.arduino.cc/en/Main/Software
[7] **ESP8266-Unterstützung**: https://arduino.esp8266.com/stable/package_esp8266com_index.json
[8] **Datei-System-Uploader**: https://github.com/esp8266/arduino-esp8266fs-plugin#arduino-esp8266-filesystem-uploader
[9] **Buch „IoT Home Hacks with ESP8266"**: http://www.elektor.de/iot-home-hacks-with-esp8266-e-book
[10] **Artikel LED-Dimmer**: http://www.elektormagazine.de/magazine/elektor-59/41891/
[11] **Weber-Fechner-Gesetz**: https://de.wikipedia.org/wiki/Weber-Fechner-Gesetz
[12] **Software**: http://www.elektor.de/amfile/file/download/file/2134/product/9476/
[13] **Webseite zu diesem Artikel (neuere Software)**: http://www.elektormagazine.de/200165-01

Lassen Sie Ihr Hobbyprojekt nicht in der Ecke verstauben

Angebotsorientiertes Zeitmanagement und spiralförmige Entwicklung

Von **Tessel Renzenbrink** (Niederlande)

Wir alle kennen es: Sie starten mit Begeisterung ein Projekt, machen einen Entwurf, bestellen die Bauteile und Ihr Projekt erhält einen prominenten Platz auf der Werkbank. Ein paar Monate später räumen Sie Ihre Elektronik-Ecke auf. Inzwischen liegt Ihr Projekt in einer Ecke und setzt Staub an. Mit der Absicht, weiter daran zu arbeiten, wenn der Winter kommt, legen Sie es mit einem Seufzer in die Schublade zu Ihren anderen unvollendeten Projekten. Eine Möglichkeit, die Erfolgschancen zu erhöhen und mehr aus Ihrem Projekt herauszuholen, ist die Kombination aus zwei Methoden: angebotsorientiertes Zeitmanagement (supply-side time management) und spiralförmige Entwicklung (spiral development).

Angebotsorientiertes Zeitmanagement

Sie können bei der Zeitplanung eines Projekts von der „Nachfrage“ ausgehen. Sie denken sich ein grandioses und ehrgeiziges Projekt aus und überlegen, wie viel Zeit es in Anspruch nehmen dürfte. Sie schätzen, dass Sie etwa sieben bis acht freie Sonntage dafür brauchen werden. Das ist großartig, in zwei Monaten wird Ihr Projekt abgeschlossen sein. Doch - in der zweiten Woche ist das Wetter fantastisch, und Sie wären ein Dieb Ihres eigenen Glücks, wenn Sie nicht diesen Waldspaziergang machen würden. Kein Problem, Sie hatten eine zusätzliche Woche einkalkuliert, so dass Sie Ihre Frist noch problemlos einhalten können. In der vierten Woche stoßen Sie auf ein unvorhergesehenes Problem. Ihre Platine funktioniert nicht, und nach einem Tag der Fehlersuche sind Sie noch keinen Schritt weiter. Es macht keinen Spaß und in der fünften Woche haben Sie keine Lust mehr, noch weiterzumachen. Ihr Enthusiasmus ist gedämpft und das erste greifbare Ergebnis rückt immer weiter in die Zukunft. Und so führt der Weg unweigerlich in die Schublade der halbfertigen Projekte.

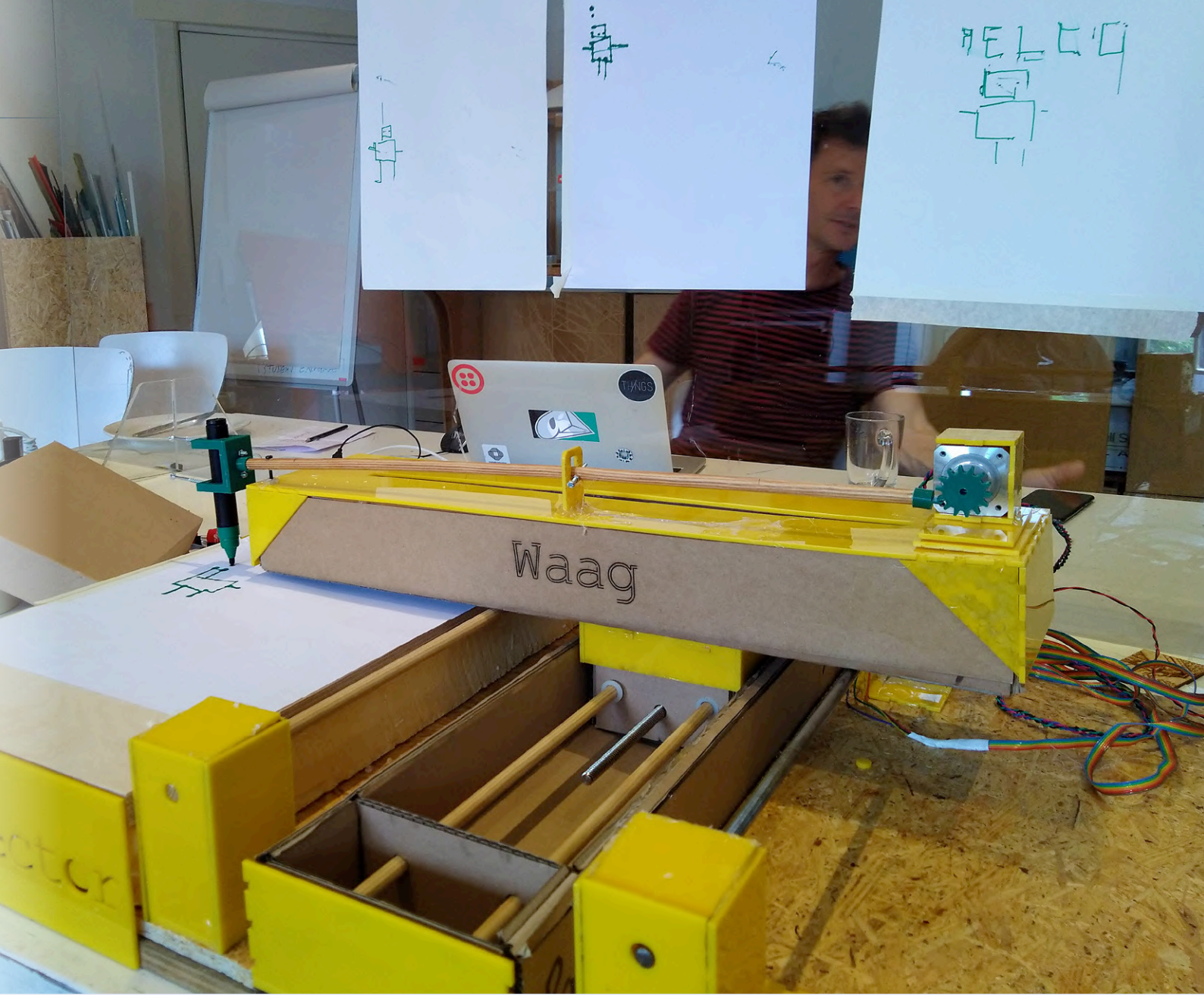
Dieses Beispiel veranschaulicht, wie nachfrage-orientiertes Zeitmanagement (*demand-side time management*) funktioniert. Sie lassen Ihr Projekt bestimmen, wie viel Zeit es braucht, und machen sich auf die Suche nach freien Stunden dafür. Das Gegenteil davon ist das Zeitmanagement auf der Angebotsseite. Sie schauen, wie viel Zeit Ihnen zur Verfügung steht und passen Ihr Projekt entsprechend an. Sie sind sich sicher, dass Sie am nächsten Sonntag basteln wollen, also fangen Sie ein Projekt an, das Sie an einem Sonntag abschließen können. Sie teilen Ihren Sonntag für jede Phase des Projekts in Abschnitte auf. Zwei Stunden für Design, drei Stunden für Prototyping und so weiter. Der Trick dabei ist, sich an den Zeitplan zu halten. Auf diese Weise vermeiden Sie, den ganzen Sonntag in der Entwurfsphase stecken zu bleiben, weil Sie dieses eine Detail nicht richtig hinbekommen.

Spiralförmige Entwicklung

Eine Planung darauf zu basieren, wie viel Zeit Ihnen zur Verfügung steht, bedeutet nicht, dass Sie nur kurzfristige Projekte in Angriff nehmen können. Hier kommt das Spiralmodell ins Spiel. Dabei teilen Sie Ihr Projekt in „Kreise“ ein, die aufeinander aufbauen. Jeder Kreis ist ein vollständig abgeschlossenes Projekt, vom Entwurf bis zum funktionsfähigen Prototyp. Jeder nächste Kreis beginnt dort, wo der letzte endete. Dadurch entsteht eine Art Spirale, in der Ihr Projekt mit jedem Kreis, den Sie hinzufügen, komplexer wird. Einer der Vorteile der spiralförmigen Entwicklung besteht darin, dass Sie viel öfter ein Endziel erreichen. Bei der seriellen Entwicklung erhalten Sie dieses Gefühl der Genugtuung erst, wenn Sie das komplette Projekt abgeschlossen haben. Bei der Spiralentwicklung haben Sie nach jedem Kreis etwas Greifbares in der Hand. Und wenn man es mit angebotsorientierter Zeitwirtschaft kombiniert, hat man jeden Sonntag etwas zu feiern.
Ein zweiter Vorteil ist, dass Sie mit jedem Kreis testen, ob Ihr Projekt funktioniert. Dadurch wird verhindert, dass Sie am Ende eines langfristigen Projekts feststellen, dass das Ding nicht funktioniert. Auch ist das Debugging eines kleinen Projekts viel einfacher als das Debugging eines großen und komplexen Projekts.

Bau einer Maschine in sieben Tagen

Wir verwendeten diese Methoden, als meine Klassenkameraden und ich in einer einzigen Woche eine Maschine bauen mussten. Wir haben die FabAcademy besucht, in der man im Affenzahn computergesteuerte Herstellungsverfahren wie den 3D-Druck und die Elektronikproduktion erlernt. Für die Gruppenarbeit haben wir eine ferngesteuerte Zeichenmaschine gebaut. Unsere Kommilitonin in Frankreich kann jetzt Katzen und Roboter zeichnen, die in unserem Labor in Amsterdam auf Papier erscheinen. Um dies in sieben Tagen zu erreichen, wandten wir angebotsorientiertes Zeitmanagement und Spiralentwicklung an.

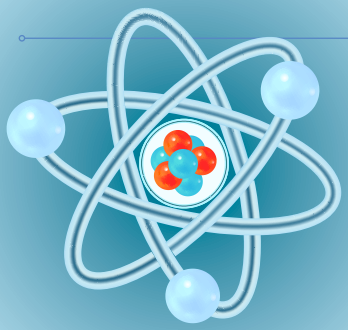


Die Zeichenmaschine, die über das Internet gesteuert werden kann (Foto: Tessel Renzenbrink).

Der Zeitdruck zeigte sich von Anfang an von einer positiven Seite. Wenn man an einem Projekt mit fünf Personen arbeitet, kann viel Zeit für Besprechungen verschwendet werden - Uneinigkeit darüber, was man baut oder was die Anforderungen sein sollen. Aber durch das unerbittliche Ticken der Uhr kamen wir schnell zu einer Einigung. Die erste Spirale bestand darin, die Schrittmotoren lokal über den seriellen Monitor der Arduino-IDE ansteuern zu können und ein einfaches Gehäuse zur Führung der X- und Y-Achse zu fertigen.

Diese erste Phase dauerte länger als von uns veranschlagt. Um dennoch innerhalb der Zeit zu bleiben, wendeten wir „Triage“ an. Dieser Begriff, den wir alle aus traurigem Corona-Anlass kennenlernen mussten, ist eine Art der Prioritätensetzung, die zuerst in Feldlazaretten angewandt wurde. Aufgrund der begrenzten Behandlungskapazität werden die Patienten in drei Kategorien eingeteilt: „unbehandelbar“, „direkt behandeln“ und „kann noch etwas warten“. Wir taten dasselbe in unserem Projekt. Der Patient, der dem Untergang geweiht war, war das Anbringen der Endschalter für die Schrittmotoren. Es schien eine einfache Aufgabe zu sein, aber nach stundenlangem Durchsuchen von Internetforen und im Anschauen von Youtube-Tutorials (im Zeitraffer-Modus) waren wir noch keinen Schritt weiter gekommen. Der Trick besteht dann darin, sich nicht zu verbeißen, sondern loszulassen. Man kann die Anforderung loslassen oder in eine spätere Spirale aufschieben. Die Zahl der Abkürzungen und „kreativen“ Lösungen nahm zu, je näher der Abgabetermin kam. Wo wir am Anfang schöne Fingerzinkenverbindungen vorsahen, wurde am Ende die Klebepistole hervorgekramt. Die Endphase einer Spirale ist aufregend, weil man in kurzer Zeit viel erreicht. Durch die Arbeit mit angebotsorientiertem Zeitmanagement und dem Spiralmodell machen Projekte mehr Spaß, weil man gezwungen ist, weiterzumachen. Es ist weniger frustrierend, weil man keine Zeit hat, sich in etwas zu verstricken, das nicht klappt. Das bedeutet nicht, dass Sie Ihr Projekt übers Knie brechen müssen, aber es hilft Ihnen, Prioritäten zu setzen. Für unseren Zweck, die Vorführung einer funktionsfähigen Maschine während des Unterrichts, waren diese Endschalter nicht entscheidend. Will man die Zeichenmaschine auch für die Nutzung durch andere öffnen, ist es ratsam, die Schalter hinzuzufügen. Aber das kann in einer folgenden Spirale eingeplant werden.

200300-02



Aller Anfang...

muss nicht schwer sein!

Von **Eric Bogers**

Wo waren wir stehengeblieben, bevor Corona unser Leben auf den Kopf stellte und wir Sie mit einer speziellen Sommerausgabe versorgt hatten? Genau. Reihen- und Parallelschaltung von Widerständen. Wir beendeten die Episode aus der Maiausgabe mit einer Denksportaufgabe: der H-Schaltung.

Das Hauptthema dieser Folge ist die Stern-Dreieck-Transformation (Sie können das auch umgekehrt lesen: Dreieck-Stern-Transformation), deren Prinzip in **Bild 1** dargestellt ist.

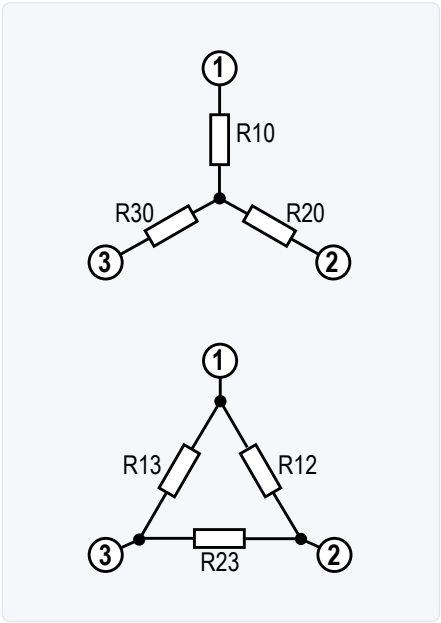


Bild 1. Die Stern-Dreieck-Transformation.

Es geht darum, die Sternschaltung oben in die Dreiecksschaltung unten umzuwandeln - und zwar so, dass sich beide für die „Außenwelt" an den Punkten 1, 2 und 3 absolut identisch verhalten.
Für die Umrechnung von Stern zu Dreieck gelten die untenstehenden Formeln (mit deren Herleitung wir Sie nicht belästigen wollen, denn schließlich ist dies eine Zeitschrift für Elektronik und keine für Mathematik):

$$R12 = \frac{R10 \cdot R20}{R30} + R10 + R20$$

$$R23 = \frac{R20 \cdot R30}{R10} + R20 + R30$$

$$R13 = \frac{R10 \cdot R30}{R20} + R10 + R30$$

Und für die Umrechnung in die andere Richtung:

$$R10 = \frac{R12 \cdot R13}{R12 + R23 + R13}$$

$$R20 = \frac{R12 \cdot R23}{R12 + R23 + R13}$$

$$R30 = \frac{R23 \cdot R13}{R12 + R23 + R13}$$

Achten Sie bitte genau auf die Bezeichnungen der verschiedenen Widerstände!
Nun können wir uns um die H-Schaltung (**Bild 2**) kümmern. Wenn wir genau hinsehen, sehen wir eine Dreieckschaltung mit zwei zusätzlichen Widerständen (R4 und R5) an der Basis. Wir unterziehen das Dreieck von R13, R12 und R23 der Transformation. Damit die Sache anschaulicher wird, setzen wir die folgenden Widerstandswerte ein: R13 = 10 Ω, R12 = 20 Ω, R23 = 30 Ω, R4 = 40 Ω und R5 = 50 Ω.
Wenn wir die Transformation anwenden, erhalten wir:

$$R10 = \frac{R12 \cdot R13}{R13 + R12 + R23} = \frac{200}{60}\Omega = 3{,}3\Omega$$

$$R20 = \frac{R12 \cdot R23}{R13 + R12 + R23} = \frac{600}{60}\Omega = 10\Omega$$

$$R30 = \frac{R23 \cdot R13}{R13 + R12 + R23} = \frac{300}{60}\Omega = 5\Omega$$

Was bleibt, ist eine einfache Kombination von Reihen- und Parallelschaltungen; wir überlassen es Ihnen, weiter zu rechnen (nebenbei, das Ergebnis lautet 29,05 Ω).

Über den Daumen gepeilt

Oft können wir uns viel von dieser Rechenarbeit sparen. Wie das? Indem man zwei „extreme" Situationen betrachtet und ein bisschen darüber nachdenkt.
In unserer H-Schaltung in Bild 2 entfernen wir den Widerstand R23 und setzen nichts an seine Stelle. Der Gesamtwiderstand des Netzwerks muss dann zunehmen (zur Erinnerung: Wenn wir einen Widerstand in einem Netzwerk mit ohmschen Widerständen erhöhen, nimmt der Gesamtwiderstand nie ab - und hier erhöhen wir R23 auf unendlich). Der Gesamtwiderstand des Netzes ist dann sehr leicht zu berechnen; wenn Sie das tun, erhalten Sie (mit den gleichen Widerstandswerten wie im Beispiel) 29,16 Ω.
Nun ersetzen wir R23 durch das andere Extrem: eine Drahtbrücke (also reduzieren wir den Widerstand auf 0 Ω). Erinnern wir uns noch einmal: Wenn wir einen Widerstand in einem Netzwerk von ohmschen Widerständen reduzieren, erhöht sich der Gesamtwiderstand dieses Netzwerks niemals. Auch hier ist die Berechnung einfach, das Ergebnis ist 28,8 Ω. Unabhängig vom tatsächlichen Wert von R23 muss der Gesamtersatzwiderstand unserer H-Schaltung zwischen diesen beiden Extremwerten liegen. Und wenn (wie in diesem Beispiel) diese Werte nicht viel mehr als etwa ein Prozent voneinander abweichen, können wir in der Praxis mit dem Mittel der beiden Extremwerte rechnen. Homines perfectici

würden das natürlich alles bis zur letzten Dezimalstelle berechnen, aber das ist oft nicht notwendig, weil der daraus resultierende Fehler ohnehin durch die Bauteiltoleranzen „verdeckt" wird.
In anderen Fällen kann die Abweichung tatsächlich zu groß sein, um den Gesamtwiderstand so einfach über den Daumen zu peilen. Dann muss gerechnet werden!

Leitungswiderstand

Bevor wir (endlich!) zum Wechselstrom kommen, müssen wir kurz über den Leitungswiderstand sprechen. Bislang haben wir den Widerstand von Verbindungskabeln eigentlich vernachlässigt - aber jedes Kabel und jede Leiterbahn auf der Platine hat einen bestimmten Widerstand. Und der sollte unter keinen Umständen vernachlässigt werden!
Dasselbe gilt für den Widerstand eines Kabels:

$$R = \frac{l \cdot \rho}{A} = \frac{l}{A \cdot \gamma}$$

Hier ist ρ ist der spezifische Widerstand und γ die spezifische Leitfähigkeit (Konduktivität). Dies sind beides Materialkonstanten. Bei den üblichen Leitern (Kupfer, Aluminium, Silber) ist es etwas einfacher, die spezifische Leitfähigkeit zu berechnen.

Spezifischer Widerstand und Leitfähigkeit

Material	ρ (Ω·mm²/m)	γ (m/Ω·mm²)
Kupfer	0,017	56
Aluminium	0,0287	34,8
Silber	0,016	62

Die Formel für den spezifischen Leiter enthält zwei weitere Variablen: die Länge *l* des Leiters und seinen Querschnitt *A*.
Berechnen wir das für eine 50-Meter-Kabelrolle mit einem Querschnitt von 1,5 mm². Das Kabel hat zwei Adern, wir müssen also mit der doppelten Länge rechnen:

$$R = \frac{l}{A \cdot \gamma} = \frac{2 \cdot 50m}{1{,}5mm^2 \cdot 56m/\Omega^2} = 1{,}19\Omega$$

Jetzt jagen wir einen Strom von 16 A durch dieses Kabel. Über dem Kabel fällt eine Spannung ab von

$U = R * I$ = 1,19 Ω * 16 A = 19 V

Das bedeutet nicht, dass wir so schon fertig sind. Der spezifische Widerstand eines Materials wie Kupfer ist nämlich temperaturabhängig - mit steigender Temperatur nimmt der spezifische Widerstand zu. Es gilt:

$$\rho_T = \rho_{20}\left(1 + \alpha\left(T - 20°\right)\right)$$

Hier ist α der Temperaturkoeffizient, der für Kupfer 0,0038 beträgt, und *T* ist die Temperatur in °C. Der Wert von ρ wird immer mit 20 C° angegeben.

Es kann gefährlich werden...

Angenommen, wir benutzen unsere Kabeltrommel an einem heißen Sommertag. Die Isolierung wird in der Sonne schön warm, und ein Strom von 16 A ist auch nicht nichts. Gehen wir vorsichtig davon aus, dass der Kupferleiter eine Temperatur von 60°C erreicht. Wenn wir dies berechnen, kommen wir auf einen Spannungsverlust von nicht weniger als 23,3 V (glauben Sie nicht? Rechnen Sie!).
Bei normaler Raumtemperatur führt das Kabel bereits eine beträchtliche Menge an Energie (in Wärme umgewandelt) ab:

$P_{Verlust} = I2 * R$ = (16 A)² * 1,19 Ω = 304,6 W

Das ist viel, und das positive Vorzeichen des Temperaturkoeffizienten wirkt sich hier für uns negativ aus: Ein stark belastetes Kabel wird heiß, was den spezifischen Widerstand erhöht, was den Leistungswiderstand erhöht, was die Verlustleistung erhöht und das Kabel noch wärmer macht, und so fort: ein sich selbst verstärkender Effekt, der leicht zu Überhitzung und Feuer führen kann. Deshalb sollen Kabeltrommeln im Gebrauch auch stets vollständig abgewickelt sein. Sie sind gewarnt worden!

200320-02

Die Artikelserie „Aller Anfang..." basiert auf dem Buch „Basiskurs Elektronik" von Michael Ebner, erschienen im Elektor-Verlag.

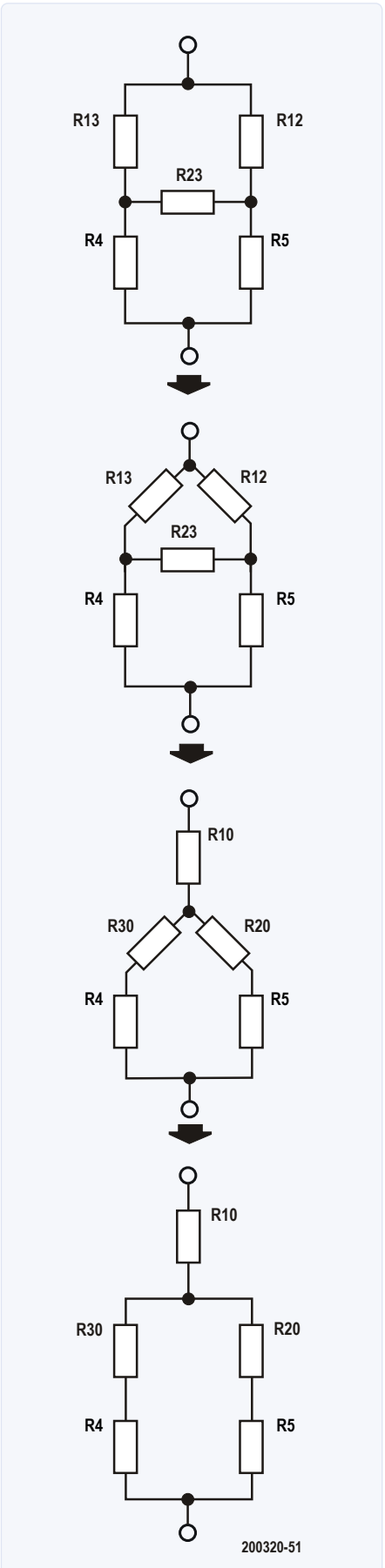


Bild 2. So behandeln wir eine H-Schaltung.

Zutriff für Unbefugte verboten!

Ein Blick ins Allerheiligste aller Elektroniker

Von **Eric Bogers**

In der heutigen Epoche der digitalen Audiosignale und hochwertigen CDs ist die analoge Schallplatte nicht eines stillen Todes gestorben - im Gegenteil, die Zahl der Liebhaber von Plattenspielern und Vinyl-Platten nimmt immer mehr zu. Und nicht nur das: Der wahre Liebhaber kauft nicht nur LPs, um sie abzuspielen, sondern nimmt selbst Musik auf Vinyl auf!

Für treue Elektor-Leser ist Quentin Therond kein Unbekannter - unter anderem gewann er 2018 den ESP32-Wettbewerb mit seinem Entwurf eines web-basierten Cocktailmixers [1]. Aber er liebt auch die gute alte Vinyl-Schallplatte - so sehr, dass er die Geräte zum Schneiden seiner eigenen Platten selbst baut. Solche Apparate werden auch fertig zum Verkauf angeboten [2], aber die sind teuer und Selberbauen macht natürlich viel mehr Spaß.

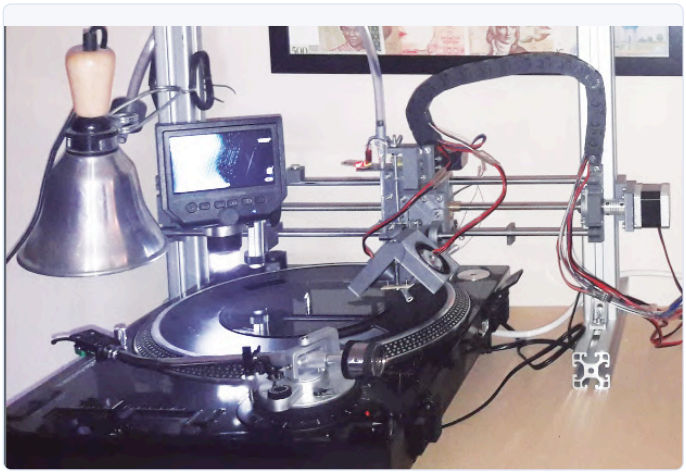


Bild 1. Das Herzstück der Anlage ist ein hochwertiger Schallplattenspieler.

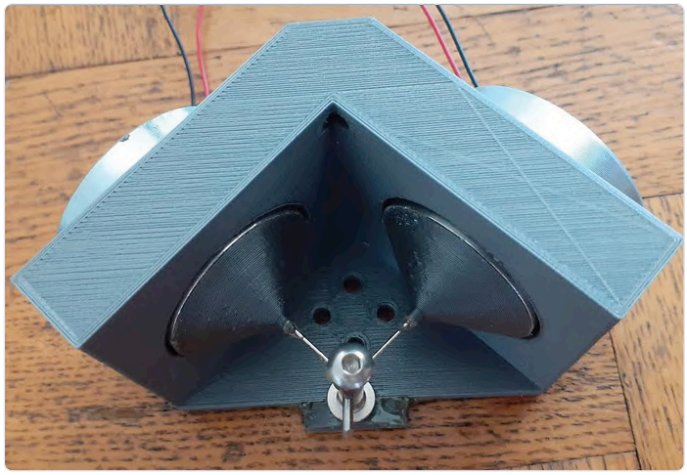


Bild 2. Selbstgebauter Prägestempel.

In dem Film „O Brother Where Art Thou“ (Eine Mississippi-Odyssee) aus dem Jahr 2000 taucht in einer Szene ein Presto-Rekorder auf - eine (nach Meinung von Enthusiasten) wundervolle Maschine, um Musik oder Sprache direkt auf einer Schallplatte aufzunehmen. Dies war Quentins erste Erfahrung mit dem Direktschnitt (direct-to-disc recording).
Bild 1 zeigt die Anlage, die Quentin baute, um seine eigenen Platten zu schneiden. Lassen wir Quentin zu Wort kommen:

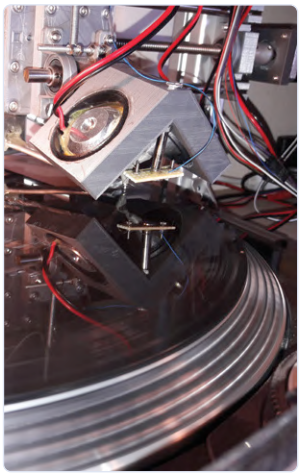
„Das Ziel besteht darin, das Audiosignal in Form einer Rille in eine Matrix zu gravieren. Dies geschieht in Echtzeit: Der Schneidstichel oder die Nadel ist mechanisch mit einem Lautsprecher oder Wandler verbunden, der das aufzunehmende Signal wiedergibt. Die Schwingungen der Membran werden durch den Taststift in die Matrix gedrückt oder eingeschnitten; die Rille stellt somit theoretisch genau das ursprüngliche Signal dar.

Zuerst müssen die Frequenzeigenschaften des Audiosignals angepasst werden - während der Aufnahme müssen die hohen Frequenzen angehoben und die tiefen Frequenzen abgeschwächt werden. Bei der Wiedergabe geschieht genau das Gegenteil. Dies ist die bekannte Schneidkennlinie [3], die verhindert, dass die tiefen Töne den Schneidstichel zu sehr auslenken und die hohen Töne zu wenig.“

Pressen oder Schneiden?
„Es gibt zwei Möglichkeiten, Ton auf eine Schallplatte (Matrix) aufzunehmen: Pressen und Schneiden. Bei der ersten Methode werden in mehreren Zwischenschritte zwei extrem hart verchromte Pressmatrizen gefertigt, zwischen denen das weiche Vinyl zusammengepresst wird, um das Signal abzubilden. Für kleine und Einzelstückzahlen bietet sich aber das Direktschnittverfahren an, bei dem die Rille mit einer Saphir- oder Diamantnadel in eine Lackschicht geschnitten wird. Dies ist zwar teurer, ergibt aber die beste Tonqualität.“

Bild 2 zeigt den von Quentin gebauten Prägestempel (mit Wandlern und Nadel) und **Bild 3** einen selbst gebauten Schneidkopf mit Lautsprechern und einer Diamantnadel.

Mono oder Stereo?



„Mein Schneidkopf ist für Stereo gebaut: zwei Lautsprecher, die in einem Winkel von 90° montiert sind. Die Diamantnadel ist an ihnen befestigt. Auf diese Weise entspricht die horizontale Verschiebung der Rille der Summe des linken und des rechten Kanals, und die Variation der Tiefe der Rille entspricht der Differenz der beiden Signale.“

Wie viele Umdrehungen?
„Je nach Rotationsgeschwindigkeit der Platte (33 oder 45 Umdrehungen pro Minute) muss der Schneidkopf mehr oder weniger schnell von außen nach innen bewegt werden. Für eine gute Qualität sollte diese Verschiebung etwa 0,1 mm pro Umdrehung betragen. Mit einer 30-cm-Langspielplatte bei 33 U/min können wir dann etwa 20 Minuten pro Seite aufzeichnen. Um den Schneidkopf mit dieser Präzision bewegen zu können, benötigt man einen hochpräzisen Schlitten (**Bild 4**) und für den Antrieb einen „gebremsten“ Motor oder einen Schrittmotor, der möglichst wenig Vibrationen und Geräusche verursacht. Ich selbst habe einen Schrittmotor verwendet, aber er vibriert eigentlich zu stark. Es gibt also immer noch Raum für Verbesserungen.“

200321-02

WEBLINKS

[1] **Cocktailmixer**: www.elektormagazine.de/magazine/elektor-59/41894
[2] **Vinyl-Recorder**: https://phonocut.com/
[4] **Schneidkennlinie**: https://de.wikipedia.org/wiki/Schneidkennlinie

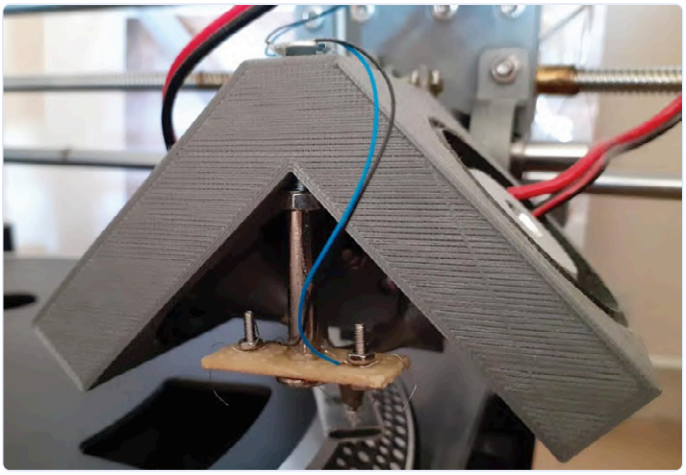


Bild 3. Schneidkopf der Marke Eigenbau.

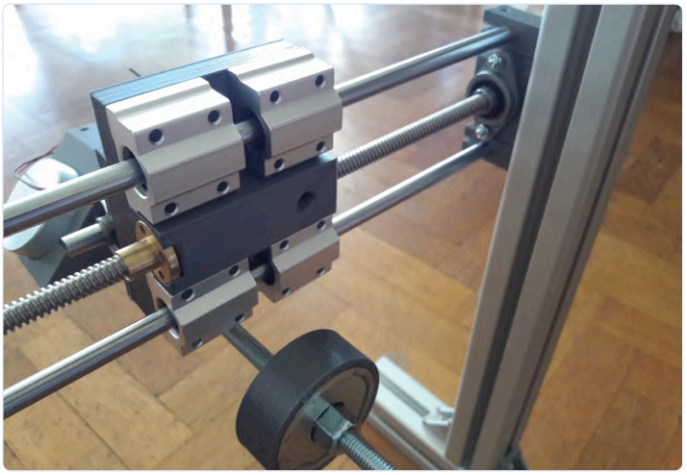


Bild 4. Schlitten zum Bewegen des Schneidkopfes.

8-bit-Mikrocontroller und darüber hinaus

Interview mit Tam Hanna

Will man IoT-Anwendungen erstellen, muss man zwingend 8-bit-Mikrocontroller verstehen. Tam Hanna, Autor von „Mikrocontroller-Basics mit PIC“ (Elektor 2020), sprach kürzlich über sein Buch, seine Sicht auf 8-bit-Mikrocontroller und einige seiner aktuellen Elektronikprojekte.

Elektor: Ihr neues Buch „Mikrocontroller-Basics mit PIC“ soll Lesern helfen, einen 8-bit-Mikrocontroller zu verstehen und zu programmieren. Sagen Sie uns, warum Sie sich auf dieses Thema konzentriert haben?
Tam Hanna: Weil 8-bit-Controller ein faszinierendes Fenster in die Welt der MCUs und allgemein in die Mikroprozessor-Elektronik sind. Aus technischer Sicht bedeutet dies schlicht und einfach: Wenn man versteht, was in einer 8-bit-MCU passiert, ist die 400 Seiten lange Architekturbeschreibung einer RISC-V-MCU viel leichter zu verstehen.

Elektor: Welche Rolle spielen Hardwarekenntnisse für den angehenden Ingenieur – damals und heute?
Tam Hanna: Zu Anfang meines Berufslebens als Ingenieur stand ich quasi in Feindesland. Damals hat mir die Einsteigerreihe „Elektronik - gar nicht schwer“ von Elektor sehr geholfen, da sie mir die Fähigkeit gab, schneller zu lernen. Jetzt, da ich dazu in der Lage bin, ist es mir eine große Ehre, Wissen wieder zurückzugeben.

In vielerlei Hinsicht sind junge Ingenieure heute mit einer anderen Situation konfrontiert. Die Verfügbarkeit von Sprachen wie Python macht es einfach, funktionierende Systeme auf die Beine zu stellen. Arduino - ob man ihn liebt oder hasst – isoliert von der zugrunde liegenden Hardware. In der Praxis begegne ich jedoch immer wieder Menschen, die gegen eine Wand laufen. Ihr Problem liegt häufig im Fehlen des Verständnisses der grundlegenden Konzepte der Elektronik. Ein primitiver 8-bit-Mikrocontroller kann auf Baugruppenebene vollständig verstanden werden. Dieses Wissen kann dann auf fortgeschrittene Controller-Architekturen übertragen werden. Angesichts der Tatsache, dass ich den PIC immer noch viel kommerziell nutze, hatte ich das Gefühl, dass es jetzt an der Zeit ist, darüber ein Buch zu schreiben.

Elektor: Wer ist die Zielgruppe für dieses Buch? Muss ein Leser Experte für C-Programmierung sein?
Tam Hanna: Um ehrlich zu sein, mache ich mir keine Gedanken über C-Programmierkenntnisse. C hat einen schlechten Ruf, weil es leicht ist, damit Chaos zu produzieren. Aber wie bei einer TU-22 tauchen Probleme nur dann auf, wenn man dumme oder aggressive Dinge tut (der Überschallbomber TU-22 neigte zur Steuerungsumkehr, wenn er aggressiv geflogen wurde).

Ich mache mir mehr Sorgen über das Niveau der elektronischen Kenntnisse. Wenn man nicht zumindest die Grundlagen der digitalen Elektronik kennt und das Ohmsche Gesetz versteht, wird man Probleme haben. Ein Elektroniker sollte zumindest die Bücher über Gleichspannung und digitale Elektronik aus der oben angeführten Serie gelesen und verstanden haben.

Elektor: Was hat Ihnen beim Schreiben des Buches am meisten gefallen?
Tam Hanna: Als ich mich vor vielen Jahren bei SourceForge anmeldete, bat mich die Seite, einen Fragebogen zur Selbsteinschätzung durchzugehen. Einer der höchsten Stufen war, ein Buch über diese Themen zu schreiben. In gewisser Weise gab mir das Schreiben dieses Buches die Möglichkeit, das Wissen zu sortieren, das ich in meiner Karriere gesammelt habe. Und natürlich auch die Fähigkeit, etwas von meiner Erfahrung zurückzugeben, um anderen zu helfen. Elektrotechnik war schon immer meine Leidenschaft, daher war es eine interessante Herausforderung, an dem Text zu arbeiten und coole Beispiele zu finden. Es ist eine große Ehre, helfen zu können, insbesondere mit einem großartigen Team.

Elektor: Was war die größte Herausforderung beim Schreiben des Buches?
Tam Hanna: Verzeihen Sie mir, wenn das kitschig klingt, aber dies ist eine sehr persönliche Frage, und ich werde eine persönliche Antwort geben. Mein größtes Problem war, herauszufinden, wann ich aufhören musste. Während ich dieses Buch schrieb, spürte ich das Gewicht der nächsten Generation von Studenten auf meinem Rücken. Ich dachte an eine jüngere Version von mir selbst, die im Labor sitzt und versucht, mehr zu lernen, um schließlich diesen Job zu bekommen, der den Start ermöglicht. Es gibt so viele hilfreiche Dinge, die ich ihm erzählen könnte, nicht nur über Mikrocontroller, sondern auch über allgemeine Elektronik und Messgeräte – doch hatte ich nur eine begrenzte Zeit dafür. Deshalb muss ich meinem Herausgeber wirklich danken. Er hat großartige Arbeit geleistet, meine Arbeit einzuschränken und dafür zu sorgen, dass dem Leser tatsächlich eine Version des Buches zur Verfügung steht, wovon er profitieren kann. Und natürlich gibt es immer die Hoffnung auf eine zweite erweiterte Auflage.

Elektor: Intel hat vor vier Jahrzehnten den MCS-51 eingeführt. Warum ist der 8-bit-Mikrocontroller immer noch relevant?
Tam Hanna: Auf die Gefahr hin, wie ein Maschinenstürmer zu klingen: Es gibt besonders unter jungen Ingenieuren ein eigenartiges Gefühl von Technikverliebtheit. Als ich anfing, hatte ich das Vergnügen, den legendären Sir Bilal Musa von der Generali-Versicherung kennenzulernen. Er schockierte mich, indem er mir sagte, dass er niemals die erste Generation eines Produkts – ob Flugzeug oder IT-System – verwenden würde. Dieses Prinzip hat sich in vielen Fällen als nützlich erwiesen, ein Beispiel dafür ist die Yakovlev Yak-42, ein sowjetisches Passagierflugzeug, das erst nach der Behebung zahlreicher Kinderkrankheiten (mit fatalen Folgen) zum robusten Arbeitstier wurde.

Ich weiß, dass 32-bit-Controller, beispielsweise der GD32E230 bei GigaDevice, fast von Minute zu Minute billiger werden. Aber braucht es wirklich immer 32-bit-Leistung? Schauen Sie sich an, was bei der Umstellung von Java auf 64 bit passierte: In vielen Fällen führten die längeren Pointer zu schlechterer Gesamtsystemleistung. Besonders bei der Arbeit an einem System mit niedrigem Stromverbrauch macht es einen Unterschied, wenn man vier Speicherzellen statt acht Bytes benötigt, um einen Pointer zu speichern.

Der portable Computer Palm VII mit weniger als 2 MB Gesamtspeicher!

Elektor: Der 8-bit-Mikrocontroller ist also nach wie vor relevant. Können Sie uns sagen, was Ihnen an der PIC-Mikrocontroller-Familie gefällt?
Tam Hanna: Ein Wort: Einfachheit! Es ist kein Zufall, dass Massimo Banzi einen AVR-Chip für den Arduino Uno ausgewählt hat. Dieser 8-bit-Mikrocontroller wurde allerdings für C-Compiler optimiert. PICs hingegen wurden von Anfang an als ein Produkt entwickelt, das „von Hand" programmiert werden sollte. Dadurch eignet sich seine interne Architektur ideal zum Lernen.
Durch die extrem weite Verbreitung dieser MCUs ist ihr Ökosystem sehr ausgereift: Programmiergeräte sind spottbillig und die meisten PICs sind im DIP-Gehäuse erhältlich. Hinzu kommt: Als PCs immer schneller und die Compilertechnik immer fortschrittlicher wurden, hat Microchip sich viel Mühe gemacht, die C-Programmierung zu perfektionieren.

Elektor: Einige Ingenieure meinen, dass 8-bit-Mikrocontroller schwierig in C zu programmieren seinen. Stimmen Sie mit dieser Meinung überein?
Tam Hanna: Einige Leute meinen, dass der Transport von Fracht mit einer Tu-144 schwierig sei, und dass das dieses Überschallflugzeug zu einer schlechten und völlig nutzlosen Konstruktion macht. Kein Tool kann alle zufriedenstellen. Wenn Sie einen komplexen Algorithmus ausführen wollen, ist ein 8-bit-Mikrocontroller wahrscheinlich nicht ideal. Wenn Sie gerne mit großen Mengen an Speicher spielen, werden Sie damit ebenfalls Probleme bekommen.
Aber seien wir doch ehrlich: Bei vielen, wenn nicht sogar den meisten Anwendungen sind die zu erledigenden Jobs doch recht einfach. Und hier ist eine kleine C-Routine einfacher zu handhaben als Assembler. Wenn Sie ständig mit dem Kopf gegen eine Wand stoßen, sollte der Wechsel zu einem leistungsfähigeren Controller oder die Erstellung eines kombinatorischen Prozessrechners der Schritt der Wahl sein.
Auf der anderen Seite ist meine Sichtweise natürlich nicht unvoreingenommen. Sehen Sie sich nur den Palm VII an. Das Ding habe ich tatsächlich programmiert.

Für jede Aufgabe ein Oszilloskop.

Hygrometer der nächsten Generation.

Elektor: Ok, wir haben bislang über 8-bit-Mikrocontroller gesprochen. Nun zu Ihrem Hintergrund und Ihren Interessen: Wann haben Sie sich zum ersten Mal für Elektronik interessiert? Wurden Sie von jemandem inspiriert? Oder entwickelte sich Ihre Liebe zur Elektronik durch eigene Bastelei?

Tam Hanna: In den 1990er und 2000er Jahren wurden an Technik interessierte Menschen automatisch als internetsüchtige Soziopathen geschmäht. Die Professionalisierung meines Interesses an Elektronik und IT war ein Weg, dem zu entkommen. Mikrocontroller und PDAs kümmern sich nicht viel darum, wer sie bedient. Solange der Quellcode oder die Schaltungskonfiguration stimmt, funktioniert das System. Dieses leistungsbasierte System war ein willkommener Ausweg für mich und hat mich seither glücklich und gewinnbringend bei der Arbeit gehalten.

Ich habe mir selbst eine Menge Sachen beigebracht, indem ich experimentierte. Dabei will ich aber die Verdienste der Ingenieurschule TGM in Wien nicht abwerten. Damals habe ich den Zugang zur Werkstatt nicht besonders geschätzt, aber im Nachhinein betrachtet, habe ich da viele Dinge erfahren, die ich anderweitig nicht gesehen hätte.

Es wäre unfair, wenn ich mich nicht bei all den Leuten bedanken würde, die mich auf meiner Bahn als Ingenieur begleitet haben – ob es nun um den Kauf von Apps für PDAs, Handy-Anwendungen, Bücher oder meine vielen, vielen Beratungskunden ging. Vielen Dank an Sie alle. Danke auch an all die Autoren von Büchern, die es mir ermöglicht haben, von ihnen zu lernen. Und last but not least danke ich Ihnen, dass Sie dieses Interview gelesen haben und den Kauf meines Buches in Erwägung ziehen. Es würde mich sehr freuen, wenn mein Buch Ihnen dabei hilft, Ihre Ziele zu erreichen.

Elektor: Mit was beschäftigen Sie sich gerade?

Tam Hanna: Das wird eine lange Geschichte. Eines der großen Dinge im Leben eines Elektro-Ingenieurs ist, dass es ihm einfach nie langweilig wird. Es ist einfach so viel los.

Sie interviewen mich in einer etwas komischen Zeit. Meine Firma befindet sich derzeit freiwillig in Quarantäne, wie auch die meisten unserer Lieferanten. Das ist aber nicht so schlimm. Vor kurzem habe ich meine Werkstatt gegen ein großes unterirdisches Labor in Ungarn eingetauscht. Ich sitze dort und arbeite an allen möglichen Ingenieurprojekten. Bevor ich auf einige Einzelheiten eingehe, möchte ich Sie auf meinen Instagram-Account aufmerksam machen. Damit bleiben Sie immer auf dem Laufenden, was das „Crazy Electronics Lab“ gerade so macht.

Dank der Bevorratung kann meine Firma normal weiterarbeiten. Der Schwerpunkt liegt auf der technischen Beratung unserer Kunden, damit diese ihre Prozesse am Laufen halten können. Zusätzlich zu den unten besprochenen neuen Produkten nutze ich die Gelegenheit, um einige Wartungsarbeiten durchzuführen. Vielleicht sollte ich auch mein Solartron-Tischmultimeter überholen, verstopfte Extruderköpfe von 3D-Druckern reparieren, Innenrenovierungen durchführen und überfällige Dinge in der Küche installieren. Schließlich könnte auch in meinem Labor endlich einmal der Boden gewischt werden.

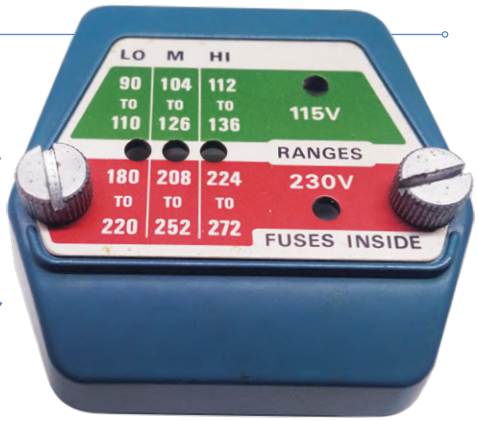
Elektor: Haben Sie irgendwelche technischen Tipps oder Ratschläge für Elektor-Leser?

Tam Hanna: Auf die Gefahr hin, mir Feinde zu machen – mein Labor sieht aus, als ob es in Havanna (Kuba) beheimatet wäre. Mein Rat: Zögern Sie nicht, gebrauchte Dinge zu kaufen. Ein eigener Vector-Network-Analyzer oder ein eigener Spectrum-Analyzer ist Gold wert. Sie brauchen solche Geräte genau dann, wenn Sie sie nicht haben und Ihr Freund in Urlaub ist.

Darüber hinaus möchte ich den alten, aber klugen Ratschlag wiederholen, Aufgaben in kleinere Problemeinheiten aufzuteilen. In vielen Fällen ist es effizienter, Schritt für Schritt vorzugehen. Wie ich schon in meinem Buch sage: Sie sollten sich nicht scheuen, zwischen der Hardware und der Software zu wechseln. Und haben Sie keine Hemmungen, Systeme zum Debugging zu missbrauchen. Ein wirklich gutes Beispiel für diesen Ansatz war das Debugging einer Kinect-Anwendung. Es war beabsichtigt, ein Bild stückweise auf verschiedene Threads aufzuteilen. Also behandelte Thread 1 den ganz linken Teil, Thread 2 den mittleren und Thread 3 den ganz rechten Teil. Wir lösten das Problem, indem wir jeden Thread vorübergehend dazu veranlassten, jedes erfasste Pixel auch zu zeichnen. Als wir den modifizierten Code ausgeführt hatten, wurde klar, dass die Bildgrenzen der Threads nicht korrekt waren - dies zu beheben war dann ein Kinderspiel.

Haben Sie keine Angst vor „großen Maschinen“ und vor Mechanik.

Etliche meiner Geräte waren im schlimmen Zustand, als ich sie bekam. Jetzt aber sind sie in einem mehr als anständigen Zustand. Meine Erfahrungen mit mechanischen Arbeiten vom Zusammenbau und der Anpassung von Möbeln bis hin zum Anbringen von Entlüftungsöffnungen an meinem alten Radiator ließen sich direkt auf meine Arbeit mit Elektronik übertragen.



Klein, aber wichtig (und unersetzlich).

Elektor: Ingenieure und Maker wissen, dass es wichtig ist, aus Fehlern zu lernen. Was war der lehrreichste technische Fehler, den Sie je gemacht haben?

Tam Hanna: Das dürfte sich schräg anhören. Normalerweise plädiere ich dafür, einfach loszulegen. Aber mein dümmster Fehler in der jüngeren Vergangenheit war der Versuch, eine Reparatur an einem Abschwächer des Typs TDS754D alleine durchführen zu wollen, obwohl ich das aufgrund eines Tremors zu der Zeit offensichtlich nicht gut tun konnte. Wenn man um Hilfe bittet oder eine Aufgabe an einen Kollegen weitergibt, kann das oft Zeit und Geld sparen.

Ein weiterer Fehler bestand darin, schwer zu ersetzende Teile bei einer Reparatur nicht sklavisch mit Klebeband am Gehäuse zu fixieren. Vor kurzem verbrachte ich eine Woche damit, eine kleine Messingabdeckung für einen 576 (Kennlinienschreiber von Tektronix) zu suchen - ohne diese konnte die Netzsicherung nicht installiert werden.

Schließlich sollte man keine Scheu vor echtem Neuland haben. 3D-Drucke bei Consumer-Anwendungen wurden bisher nicht praktikabel angesehen. Ich bin ziemlich zuversichtlich, dass Besitzer von Humidoren (Zigarrenkisten mit Feuchtereglung) es bald zu schätzen wissen werden, dass sie die Farbe ihres Hygrometers auswählen können.

Elektor: Wie geht es weiter? Haben Sie ein neues Buch, Produkt oder Projekt in Arbeit?

Hanna: Was als nächstes für mich ansteht? Ich werde mir eine Zigarre anzünden. Und während ich das tue, betrachte ich eines meiner zukünftigen Produkte; den HygroSage. Es ist das erste Humidor-Hygrometer, das überhaupt keine Kalibrierung benötigt. Es verfügt über ein Farbdisplay und hat eine garantierte Genauigkeit von 2% über die gesamte Lebensdauer.

Projekt Nummer 2 ist die Stinkely-Serie von Ersatz-Displays. Der Konzern Danaher verdient ein Heidengeld mit dem Verkauf von Kathodenstrahlröhren für die 57x-Kennlinienschreiber von Tektronix. Da werde ich mich einmischen. Außerdem gibt mir die Rettung dieser Geräte vor der Mülldeponie immer ein warmes und gutes Gefühl.

Schließlich arbeite ich für ein amerikanisches Modeunternehmen, das sich aus rechtlichen Gründen immer noch im Stealth-Modus befindet. Das USPTO (United States Patent and Trademark Office) ist eine seltsame und faszinierende Institution, aber glauben Sie mir, wenn ich behaupte, dass wir Sonnenbrille und Aschenbecher neu definieren werden. Wenn Sie derzeit an die Anschaffung eines Aschenbechers oder einer Sonnenbrille im Preisbereich von 200$ denken, warten Sie besser ab und lassen Sie sich überraschen!

Arbeiten Sie an einem 8-bit-Mikrocontroller-Projekt? Mit einem kostenlosen Elektor-Labs-Account können Sie Details austauschen und mit anderen Ingenieuren und Makern zusammenarbeiten. Melden Sie sich an – noch heute: www.elektormagazine.com/labs!

200305-03

IM ELEKTOR-STORE

- **Tam Hanna: Mikrocontroller-Basics mit PIC:**
  www.elektor.de/mikrocontroller-basics-mit-pic
- **E-Book (pdf):**
  www.elektor.de/mikrocontroller-basics-mit-pic-pdf

Von der Pike auf gelernt

Aus der Elektor-Ideenkiste

Zusammengestellt von **Eric Bogers**

Für einen verregneten Sonntagnachmittag oder für einen dieser Lockdown-Tage, an denen einem die Decke auf den Kopf zu fallen scheint, sind kleine Schaltungen, die man rasch auf einem Stückchen Lochraster aufbauen kann, ein willkommener Zeitvertreib. Hier stellen wir einige weitere vor.

Idee: Petre Petrov (Bulgarien)

Dreiweg-Audioverstärker

Manchmal brauchen wir einen einfachen „LowFi"-Audioverstärker für Experimente im Labor oder für Anwendungen, bei denen es nicht auf Klangqualität ankommt. Hier ist einer, der sicherlich in die Kategorie „LowFi" fällt, aber dennoch einige interessante Möglichkeiten bietet, wie das Schalt**bild 1** zeigt.

Es war beabsichtigt, für diesen Verstärker eines dieser externen Netzteile für Laptops oder Notebooks zu verwenden, die man, wenn man sucht, für ganz wenig Geld erwerben kann. Diese Netzteile liefern zwar in der Regel eine Ausgangsspannung von 19 V_{DC}, haben aber üblicherweise den Nachteil, dass die Ausgangsspannung mit allen möglichen Störungen verseucht ist. Deshalb wurde ein LC-Filter verwendet, um diese Störungen und Rauschen so weit wie möglich zu unterdrücken. Als eigentlicher Verstärker wird der bekannte LM380 (oder LM384) verwendet, und zwar nicht weniger als drei Mal, um einen Drei-Wege-Verstärker mit getrennten Signalwegen für tiefe, mittlere und hohe Frequenzen aufzubauen. Die notwendigen Filter können auf Wunsch mit Schaltern (S3...S8) überbrückt werden. Der Verstärker verfügt auch über separate Eingänge für Tiefton, Mittelton und Hochton, die jedoch mit den Schaltern S1 und S2 kombiniert werden können.

Die Schaltung ist nicht besonders kritisch und kann relativ einfach auf einer Lochrasterplatine aufgebaut werden.

Ein paar Anmerkungen: Die Verstärker-ICs können recht heiß werden, so dass ein geeigneter Kühlkörper zwingend erforderlich ist. Abhängig von der Impedanz der verwendeten Lautsprecher muss der Wert des Ausgangskondensators wie in der Schaltung angegeben angepasst werden.

Sowohl der LM380 als auch der LM384 weisen einen internen Verstärkungsfaktor von 50 auf, bei einer Eingangsimpedanz von 150 kΩ. Bei einer Versorgungsspannung von 19 V beträgt die maximale Ausgangsleistung (bei einem Klirrfaktor von 10%) etwa 3 W.

Ein besser lesbarer Schaltplan kann von der Projektseite [1] heruntergeladen werden.

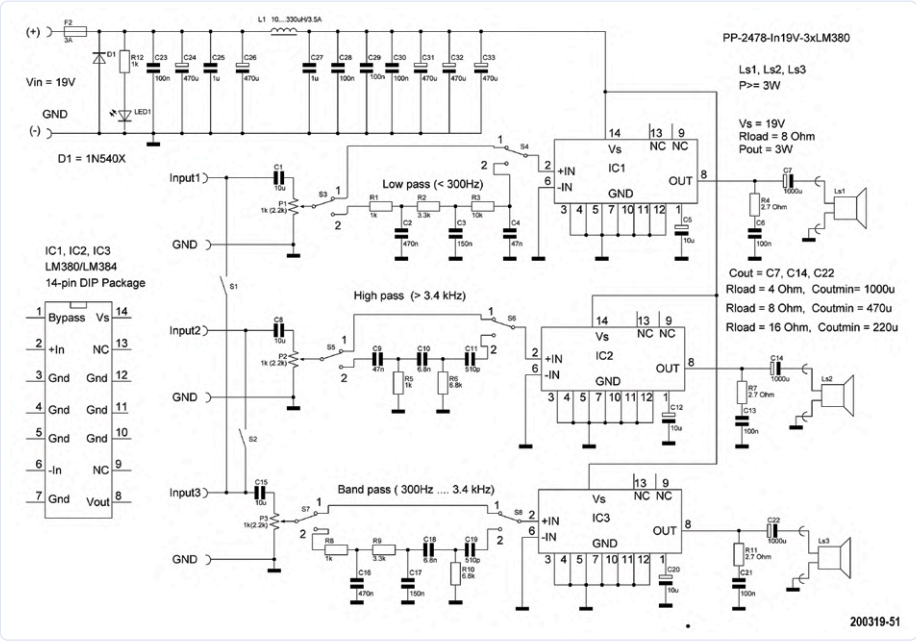


Bild 1. Das Diagramm sieht kompliziert aus, aber glücklicherweise trügt der Schein.

Idee: Elex-Team

Einfaches Ionometer

Es gibt zwei Gründe, warum wir Ihnen diese supereinfache Schaltung präsentieren: Zum einen ist in Home-Office-Zeiten das Ionenklima in Innenräumen angeblich wichtig für das Wohlbefinden, zum anderen können Sie dieses antike Drehspulinstrument aus der Krabbelkiste endlich für etwas gebrauchen...

Angeblich haben negative Ionen in der Umgebungsluft einen positiven Einfluss auf Gesundheit und Stimmung von Menschen. Zugegeben: Die positive Wirkung solcher heilsamer, negativer Ionen wird vor allem von den Herstellern sogenannter Ionisatoren behauptet. Wissenschaftlich fundierte Studien, welche positive gesundheitliche Auswirkungen belegen? Fehlanzeige. Dennoch: Man kann sich immerhin besser fühlen, wenn man annimmt, dass das stimmt. Außerdem hat man in den Zeiten des globalen Lockdowns Zeit, etwas für die Verbesserung des Raumklimas zu tun. Mit Hilfe der hier vorgestellten Schaltung können Sie dann selbst testen, ob die Luft um Sie herum genügend negative Ionen enthält.

Die Schaltung in **Bild 2** beweist, dass zum Nachweis negativer Ionen nicht viel Aufwand erforderlich ist. Die Ionen werden mit einer kleinen Metallplatte (die schraffierte Fläche links

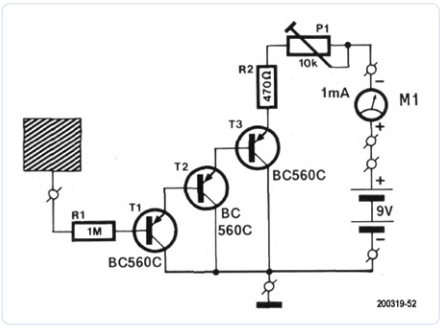


Bild 2. Das Ionometer besteht aus ein paar Teilen, die Sie garantiert auf Lager haben...

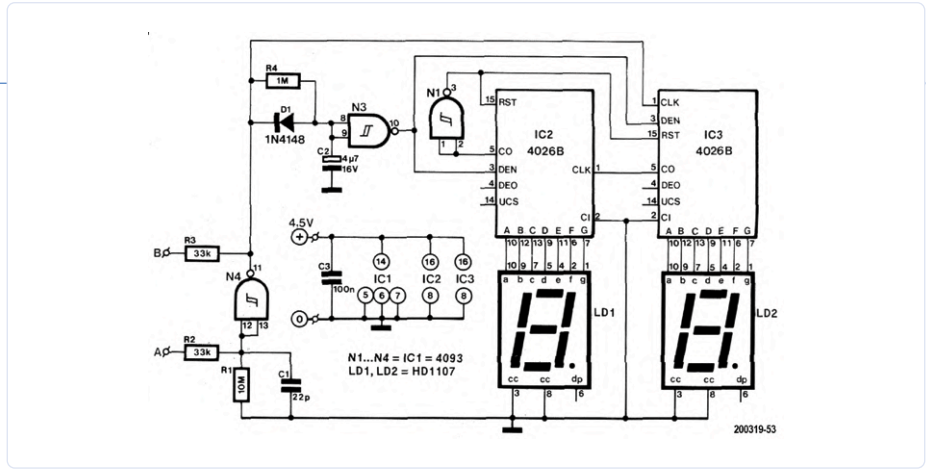


Bild 3. Dieser Lottozahlengenerator kann recht einfach auf einem Stück Lochraster aufgebaut werden.

in der Schaltung) „eingefangen" (oder gesammelt). Wenn es sich um positive (und somit angeblich schädliche) Ionen handelt, passiert nichts. Wenn jedoch die guten alten negativen Ionen in der Mehrheit sind, dann wird die Basis des PNP-Transistors T1 gegenüber dem Emitter negativ und damit der ganze Transistor leitend. Dasselbe gilt dann dank der Stromverstärkung von T1 um so mehr für T2 und erst recht für T3. Diese drei Transistoren bilden einen dreistufigen Super-Darlington - die Gesamtverstärkung entspricht dem Produkt der einzelnen Verstärkungsfaktoren und ist daher enorm groß. Bereits bei einer geringen negativen Ladung auf der Metallplatte beginnt T3 ausreichend zu leiten, und der Strom durch T3 wird durch das Drehspulinstrument M1 sichtbar gemacht. Man kann also nicht nur sehen, ob überhaupt negative Ionen vorhanden sind, sondern dank Zeigerausschlag die Konzentration grob abschätzen.

Der Aufbau auf einem Stück Lochrasterplatine ist nicht kritisch. Man muss lediglich auf eine möglichst kurze Verbindung zwischen der Metallplatte und R1 achten. Die Empfindlichkeit der Schaltung kann man mit dem Trimmpoti P1 einstellen. Sie können die Funktion übrigens gut im Badezimmer testen: Wenn Sie die Dusche aufdrehen, werden viele negative Ionen freigesetzt. Da stellt sich doch gleich die Frage: Ist eine schöne heiße Dusche gerade deshalb so entspannend?

Idee: Elex-Team

Lottozahlen-Generator

Sagen Sie ehrlich: Natürlich haben Sie schon einmal darüber fantasiert, was Sie tun würden, wenn Sie im Lotto gewännen - ein Haus oder ein neues Auto kaufen, oder eine Weltreise machen, oder dem nervigen Chef die Meinung geigen (und dann Ihren Job schnell kündigen)... Aber das Ausfüllen eines solchen Lottoformulars, das Ankreuzen der hoffentlich gewinnbringenden Zahlen, kann enervierend sein und nach der Ziehung zur Selbstzerfleischung führen (hätte ich doch nur die 33 angekreuzt...)

Hier kann die Elektronik helfen. Wenn wir davon ausgehen, dass alle Zahlenkombinationen gleich wahrscheinlich sind, können wir mit Hilfe eines Zufallsgenerators und einer Anzeige echte Zufallszahlen (in diesem Fall zwischen 0 und 49) erzeugen.

Die Schaltung des Generators ist in **Bild 3** dargestellt. Das Herzstück bildet ein Oszillator, der um N4 herum gebaut ist. An den Punkten A und B sind zwei kleine Metallflächen (Platinenabschnitte, Reißzwecken) als „Tipptasten" angeschlossen. Solange die Tasten nicht berührt werden, steht der Oszillator still und sein Ausgang (Pin 11 von N4) ist high. Infolgedessen ist auch die Spannung am Kondensator C2 hoch und der Ausgang von N3 low. Dieser Ausgang steuert die Freigabeeingänge der Dekadenzähler IC2 und IC3, und auch die Siebensegmentanzeigen werden gelöscht.

Kurzes Intermezzo: In dem verwendeten Dekadenzähler-Typ CD4026B steckt ein 7-Segment-Decoder plus Treiber, so dass die Anzeigen mit gemeinsamer Kathode LD1 und LD2 direkt an die Ausgänge der Zähler-ICs angeschlossen werden können.

Sobald wir die Kontaktflächen mit einem Finger berühren, beginnt der Oszillator bei einer Frequenz von einigen kHz zu arbeiten. Am Ausgang befindet sich ein Rechtecksignal, das als Taktsignal für IC3 verwendet wird. Der Überlauf-Ausgang (Carry Out, CO) dieses ICs liefert das Taktsignal für IC2.

Wenn der Oszillator läuft, wird C2 jedes Mal schnell entladen, wenn der Ausgang von N4 low ist. Das Aufladen dieses Kondensators über R4 dauert wesentlich länger. Dies bedeutet, dass der Ausgang von N3 high ist, solange der Oszillator läuft, und dass die Anzeigen aktiviert sind. Da der Zähler sehr schnell läuft, scheint es, dass alle Segmente gleichzeitig aufleuchten (und man keinesfalls irgendeine Ziffer erkennt).

In dem Moment, in dem die Tipptaste losgelassen wird, stoppt der Oszillator und die Anzeigen zeigen die aktuelle Zählerposition deutlich an - für einige Sekunden, bis C2 wieder ausreichend geladen ist. Dann verlöschen die Anzeigen und die nächste Zahl kann generiert werden.

Der Carry-Out-Ausgang der Zähler-ICs ist bei den Zählerpositionen 0...4 hoch und kippt auf low, sobald der Zähler die Position 5 erreicht. Diese Eigenschaft nutzen wir, damit der Generator nicht weiter als 49 zählt. Sobald der Zählwert 50 erreicht ist, werden beide Zähler über N1 zurückgesetzt.

Die Schaltung hat zwei Schönheitsfehler: Es ist möglich, dass der Zählwert 00 angezeigt wird. Das ist natürlich ungültig. Und die Schaltung prüft nicht, ob die gleiche Zahl zweimal gezogen wird. Angesichts der Einfachheit der Elektronik lässt sich damit jedoch leben.

200319-02

WEBLINK

[1] **Projektseite:** www.elektormagazine.de/200319-02

Bluetooth-Multimeter
Owon OW18E

Von **Harry Baggen**

Da heute ja scheinbar alles irgendwie „Verbindung“ haben muss, kommt auch ein Multimeter nicht umhin, seine Messresultate per Funk auf ein Smartphone oder Tablet zu übertragen. Dank der Bluetooth-Funktion und einer entsprechenden App klappt das auch. Hier geht es um das neue Multimeter Owon OW18E, das nicht nur über Bluetooth verfügt, sondern auch viele weitere Funktionen und hohe Genauigkeit zu einem bescheidenen Preis bietet.

Die OW18-Serie birgt die aktuellen tragbaren Multimeter von Owon. Ursprünglich bestand sie nur aus den Modellen OW18A und OW18B, doch jetzt sind noch die beiden Versionen OW18D und OW18E hinzugekommen. Im Vergleich zu den A- und B-Versionen bieten die Typen D und E eine höhere Genauigkeit und eine höhere Auflösung des Displays (4½ im Vergleich zu 5⅝ Digit). Wie schon die B-Version ist auch der Typ E mit Bluetooth ausgestattet.

Hardware

Der Typ OW18E sieht genauso aus wie alle Geräte dieser Serie. Das relativ große Gehäuse fühlt sich robust an und ist mit einer blauen (abnehmbaren) Schutzhülle aus Weichplastik versehen. Der Drehschalter funktioniert ziemlich geschmeidig. Das große Display zeigt den Wert vierstellig mit unterdrückter 1 an, so dass der maximale Messwert 19999 beträgt. Es gibt mehrere Indikatoren, die je nach Einstellung sichtbar sind. Das Display ist recht gut ablesbar, nur bei Sicht von oben nimmt der Kontrast schnell

ab. Auf der Rückseite befinden sich eine klappbare Stütze und das Batteriefach für eine 9-V-Batterie.
Es gibt vier Tasten zum Umschalten auf die zweite Funktion einer Schalterstellung, zur manuellen Bereichseinstellung, zur Displaybeleuchtung, für Bluetooth, Hold, Relative und Duty-Cycle.
Das Messgerät wird mit einem Satz Messleitungen mit passenden Krokodilklemmen, einem Kabel mit einem Temperatursensor (K-Typ), einer Batterie, einem Handbuch, einem Datenblatt, einer Karte mit Anweisungen zum Herunterladen der Software und schließlich einem Schraubendreher zum Öffnen des Batteriefachs geliefert.

Der Lieferumfang des OW18E.

Messoptionen

Die Anzahl der Messmöglichkeiten ist recht groß. Die am häufigsten verwendeten Bereiche sind natürlich Volt und Ampere, bei Wechselspannungen wird der „echte Effektivwert“ gemessen. Es gibt einen besonders empfindlichen V-Bereich von 20/200 mV und einen besonders empfindlichen Strombereich von 200 µA. Strom kann bis zu 20 A gemessen werden (für maximal 10 s!) – bemerkenswert, weil sonst eher 10 A geboten werden.
Natürlich verfügt das Messgerät auch über eine Widerstands/Dioden/Durchgangsmessung. Der Kapazitätsmessbereich geht bis 20.000 µF. Frequenzen kann man bis 20 MHz erfassen – echt brauchbar für diese Preisklasse! In diesem Modus kann man auch das Tastverhältnis messen. Die Temperaturmessung mit dem mitgelieferten Sensor reicht bis zu 400 °C. Schließlich gibt es einen NCV-Modus, mit dem man kontaktlos erkennen kann, ob ein Stecker, eine Leitung oder eine Steckdose Netzspannung führt. Oberhalb der Anzeige befindet sich eine LED, die zu blinken beginnt, wenn sich Netzspannung in der Nähe befindet. Diese Anzeige leuchtet auch beim Betrieb der meisten Funktionen. Zusätzlich zum NCV-Sensor auf der Stirnseite gibt es eine weiße LED, die als Taschenlampe dient. Das Messgerät ist (natürlich) mit einer Auto-Power-Off-Funktion (APO) ausgestattet, die es nach 30 Minuten Inaktivität abschaltet.

An der Vorderseite befinden sich der NCV-Sensor und eine Beleuchtungs-LED.

Messergebnisse

Zunächst ein Wort zur Sicherheit. Das Messgerät entspricht nach Angaben des Herstellers der Messkategorie CAT III 1000 V / CAT IV 600 V. Das bedeutet eine recht gute Isolierung. Die Grundgenauigkeit des OW18E ist mit 0,1%+2 Digit für den 2-V-Bereich und höher spezifiziert. Im Vergleich zu einem Messgerät mit zehnmal höherer Genauigkeit erwies sich der Owon als sehr genau: Die Abweichung betrug weniger als 2 Digit. Bei DC-Messungen waren die Ergebnisse also vergleichbar.
Bei AC-Messungen betrug die Abweichung einige Millivolt bei einem gemessenen Wert von fast 2 V. Auch das ist in Ordnung, denn angegeben sind 0,5 %+10 Digit. Für AC-Messungen wird ein Frequenzbereich von 40 bis 1.000 Hz angegeben. Bei den Testmessungen blieb die Anzeige bis etwa 1,5 kHz ziemlich genau.
Auch die Widerstandsmessung war viel genauer als die versprochenen 0,3 %, bei 1 kΩ kam ich auf eine Abweichung von 0,1...0,2 %. Auch bei der Kapazitätsmessung wich der gemessene Wert nur wenige Digits vom Referenzmessgerät ab. Dies sind ausgezeichnete Ergebnisse für ein Multimeter dieser Preisklasse.

Praxis

Und wie gefällt dieses Messgerät im täglichen Gebrauch? Das Display ist übersichtlich und seine großen Ziffern sind aus fast allen Winkeln gut lesbar. Die Autoranging-Funktion ist nicht so schnell, aber arbeitet zuverlässig. Ein nettes Detail: Auf dem Display ist die Bereichsanzeige unterhalb des Dezimalpunktes in kleinen Ziffern dargestellt. Die Funktionen der verschiedenen Tasten sind schnell verstanden. Bei ihrer Betätigung ertönt der eingebaute Piepser und die LED leuchtet kurz auf. Der Durchgangs-Piepser funktioniert sehr gut. Man hört den Piepser oder sieht die leuchtende LED oder das reagierende Display. Auch die Messung von Kondensatoren funktioniert gut, selbst bei größeren Werten. Bei einem Elko mit 2.200 µF benötigte das Messgerät etwa 8 s zur Messung der Kapazität.
Die Beleuchtung des Displays sorgt für eine gute Ablesbarkeit in dunkler Umgebung, gleichzeitig leuchtet auch die Taschenlampen-LED auf. Letzteres wäre für mich nicht notwendig, aber man kann sie leider nicht getrennt einschalten. Mit der NCV-Funktion war ich nicht zufrieden, denn sie ist ziemlich unsensibel. Selbst wenn sich der Sensor direkt oben auf einem Netzstecker befindet, zeigt der Detektor dennoch manchmal nichts an. Ich würde mich also nicht darauf verlassen.

Die 9-V-Batterie befindet sich in einem separaten Halter, der in das Messgerät eingeklickt und angeschraubt wird.

Schönes Detail: Unterhalb des Dezimalpunktes wird der Messbereich klein dargestellt.

Bluetooth

Neben der für diese Preisklasse hohen Genauigkeit ist die Bluetooth-Funktion das Hauptmerkmal des OW18E. Verbaut ist ein BLE4.0-Modul, das eine stabile Verbindung und einen niedrigen Energieverbrauch gewährleistet. Ich habe die Bluetooth-Funktion des Messgeräts für mehrere Stunden aktiviert, ohne dass die Batterie aufgegeben hat. Die entsprechende Multimeter-BLE-App ist für Android und iOS verfügbar.

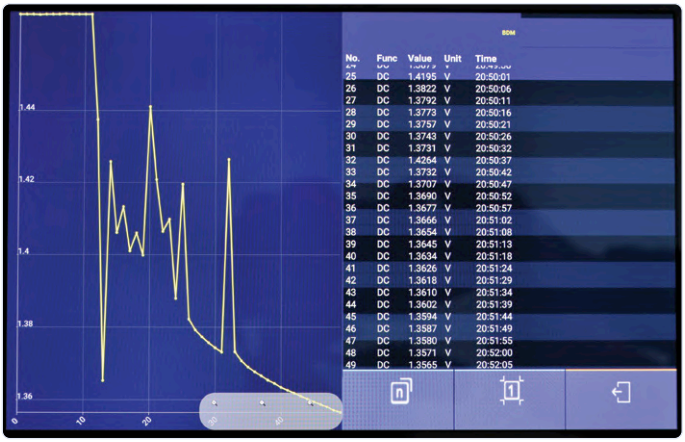
Nachdem die App installiert und gestartet ist, schaltet man die Bluetooth-Funktion des Messgeräts ein, woraufhin es in der App als „BDM“ erscheint. Nachdem dies ausgewählt wurde, verbindet sich die App und der Messwert erscheint auf dem Display. Die Funktionen der Drucktasten können auch durch die App bedient werden. Der Drehschalter muss jedoch manuell bedient werden. Die App bietet auch eine Datenlogger-Funktion mit einer Reihe von Einstellmöglichkeiten, wie das Aufzeichnungsintervall. Die protokollierten Werte können in einer Datei gespeichert und versendet werden. Via App kann man das Messgerät auch selbstständig protokollieren lassen, woraufhin die Bluetooth-Verbindung getrennt werden kann. Später können dann die Messwerte durch die App abgerufen werden. Die App bietet auch die Möglichkeit, sich gleichzeitig mit zwei Geräten zu verbinden.

Es ist etwas schade, dass die App nur das Querformat bietet und beim Start standardmäßig den Bildschirm von zwei Geräten zeigt, denn die meisten Benutzer dürften nur über ein Messgerät verfügen. Ansonsten funktioniert die App reibungslos: Die Verbindung ist stabil und die Daten werden schnell vom Gerät zur App übertragen. Bluetooth ist sehr praktisch, wenn man nicht die ganze Zeit neben dem Gerät sitzen will oder kann. Die Informationen in der App lassen gegenüber dem Display des Geräts nichts vermissen. Das Messgerät bietet daher eine gute Alternative zum beliebten, aber nicht mehr erhältlichen Mooshimeter.

Fazit

Für weniger als 70 € ist das Owon OW18E ein ausgezeichneter Kauf. Für dieses Geld bekommt man ein solides Gerät, das nicht nur viele Messmöglichkeiten bietet, sondern auch überraschend genau ist. Hinzu kommt Bluetooth, was Fernmessungen und Datenprotokollierung ermöglicht. Mehr kann man für sein Geld wirklich nicht erwarten.

200322-03



Der Bildschirm mit protokollierten Werten in der App.

Interaktiv

Korrekturen & Upadates || Fragen & Antworten

Zusammengestellt von **Clemens Valens** (Elektor-Labor)

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

Neue Firmware für ein altes Projekt

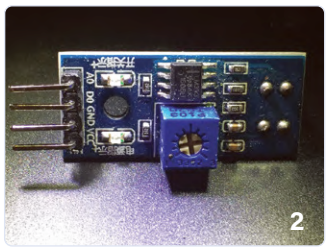
Der Artikel „Universeller PWM-Treiber“ aus dem Halbleiterheft 2010 weckte mein Interesse, weil er einen Drehgeber zur Auswahl der Optionen verwendete, die gleichzeitig in einem LCD-Menü dargestellt wurden. Ich habe mehrmals versucht, einen PIC16F628A [den im Projekt eingesetzten Controller, Redaktion] zu programmieren, aber die Software funktionierte einfach nicht. Sie wurde, glaube ich, ursprünglich in C/C++ entwickelt, wahrscheinlich mit einem CCS-Compiler. Da ich ein EasyPICv7-Board und eine mikroC-Lizenz besaß, portierte ich die Originalsoftware, aber nur, um festzustellen, dass sie immer noch nicht funktionierte. Daher analysierte ich den Quellcode im Detail, modifizierte ein paar Zeilen und programmierte einen PIC16F88 mit der daraus resultierenden HEX-Datei (dem ursprünglichen PIC16F628A fehlte RAM, die neue LCD-Bibliothek verbraucht wahrscheinlich mehr Ressourcen als die vom Entwickler Alexander Ziemek verwendete). Auf diese Weise gelang es mir, ein voll funktionsfähiges System zu entwickeln. Der Quellcode wurde zudem mit vielen Kommentaren versehen, wodurch er leichter zu lesen und zu verstehen ist. Der Code kann, da ich beileibe kein professioneller Software-Entwickler bin, sicherlich noch optimiert werden. Ich biete Ihnen hiermit meine Arbeit [1] als ZIP-Archiv zusammen mit dem leicht modifizierten Schaltplan an. Der R/W-Pin des LCDs wird nicht genutzt, sondern fest auf Masse gelegt.Dadurch wird der RA2-Portpin des Prozessors frei. Die Idee hinter der Schaltung, die Funktionen und Merkmale des ursprünglichen Designs bleiben jedoch unverändert.

Philippe Le Guen

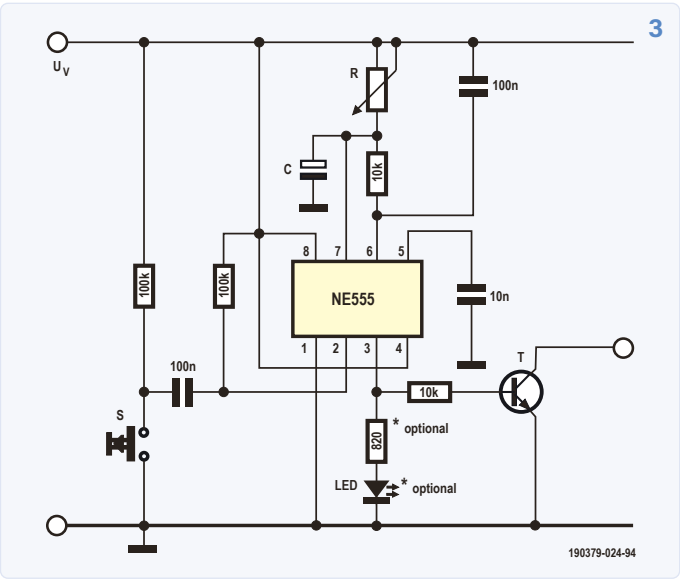
[1] www.elektormagazine.de/190379-D-02

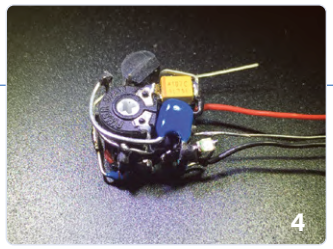
Verbesserter Sensor für Wasserverbrauchsüberwachung

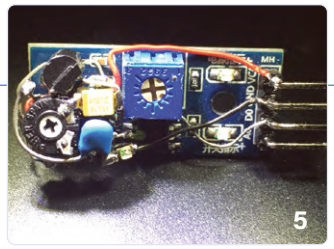
Nachdem ich den Artikel „Wasserverbrauchsüberwachung mit ESP32“ von Denis Lafourcade in der Elektor-Ausgabe Juli/August 2019 gelesen hatte, wurde mir sofort klar, dass ein solches System für mich sehr nützlich sein würde. Doch der erste Schritt erwies sich als schwierig: Ich konnte keinen geeigneten Sensor für meinen Wasserzähler finden. Eine Anfrage an den Hersteller blieb unbeantwortet und so musste ich mich auf eine DIY-Lösung verlassen.

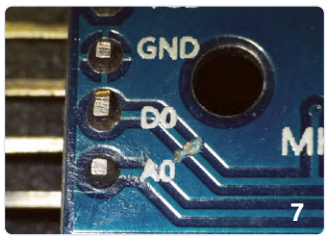


Ein erster Test mit einem Hallsensor war erfolglos, da der Wasserzähler keine magnetischen Teile enthält (um betrügerische Handlungen zu verhindern, wie ich später herausfand). Ein halbkreisförmiges, glänzendes Plättchen auf dem Literzeiger schien eine mögliche Lösung zu bieten (Bild 1). Vielleicht könnte ich es dazu verwenden, die Drehungen des Zeigers zu erkennen? Da ich vor einiger Zeit einige optische Infrarot-Reflexionssensoren bestellt hatte (**Bild 2**), suchte ich nach Befestigungsmöglichkeiten für einen dieser Sensoren am Wasserzähler. Von Thingiverse [1] lud ich 3D-Modelldateien für ein geeignetes Sensorgehäuse herunter, druckte eines aus und setzte einen Sensor darin ein. Diese Baugruppe wurde mit Saugnäpfen an der Wasseruhr befestigt.









Auch der Rest des Wasserverbrauchsmonitors war schnell gebaut. Der erste Test ergab enttäuschende Ergebnisse: Die Messwerte des Monitors waren viel zu hoch. Ein Blick auf den Arduino-Sketch und die anschließende Analyse des Ausgangssignals des Sensors zeigten den Grund dafür. Der Sketch ging von 500-ms-Impulsen aus, aber mein Infrarotsensor gab ein anderes Signal aus, denn die Länge der Impulse hing nicht nur von der Wasserdurchflussrate ab. Auch kam es recht häufig vor, dass das glänzende Plättchen direkt vor dem Sensor Halt machte, was zu einem permanenten High-Pegel führte. Was tun? Sofort dachte ich an den guten, alten Zeitgeber NE555, aber ein Mikrocontroller wäre auch möglich gewesen. Da ich aber alles analog halten wollte und da ich „zufällig" ein paar 555er herumliegen hatte, entschied ich mich für die erste Lösung. Der 555 wurde als nicht retriggerbarer, monostabiler Multivibrator (MMV) geschaltet (**Bild 3**). Dieser MMV kann nur ausgelöst werden, wenn er sich im „Leerlauf" befindet. Dann erzeugt er an seinem Ausgang einen Impuls von etwa 500 ms Länge. Diese Schaltung arbeitete perfekt, ja, sie verhinderte sogar eine Fehlauslösung beim Einschalten der Schaltung.

Der Aufbau der Schaltung war nicht einfach, da alle Teile, einschließlich des 100-µF-Elkos, in das kleine Sensorgehäuse passen mussten. Aus diesem Grund habe ich einige SMD-Bauteile verwendet (**Bilder 4…6**).

Um die Schaltung mit dem Sensor zu verbinden, mussten die beiden Leiterbahn durchtrennt werden, die vom LM393 auf der Sensorplatine zum Ausgang am Steckverbinder D0 führen (**Bild 7**). Pin 7 des LM393 wird an den Eingang des MMVs gelötet, der Ausgang des MMVs wird mit dem Steckverbinder (D0) verbunden.

Ich habe der Projektseite auf Elektor Labs [2] auch eine kurze Anleitung zur Einrichtung des IFTTT-Dienstes beigefügt.

Hans Schneider

[1] www.thingiverse.com/thing:2749407

[2] www.elektormagazine.com/labs/waterflow-monitor

Bauen Sie einen „wearable" Vibrationsmonitor

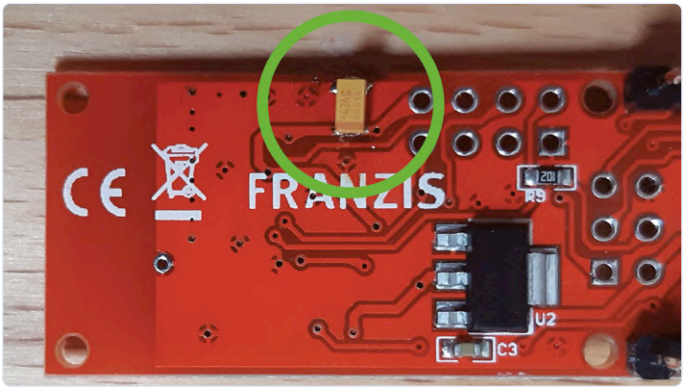
Wenn Sie Ihren Körper oder Teile davon langfristig Vibrationen aussetzen, kann dies zu Verletzungen führen. Aber wie misst man das Risiko schädlicher Vibrationen? Elektor-Labs-Mitglied „sixbacon", Physiker und Ingenieur mit Interesse an Gesundheit und Arbeitsschutz, wollte der Sache auf den Grund gehen. Er wollte auch Heimwerker-Werkzeuge beurteilen, die meist unreguliert sind. Da sixbacon auch gerne Dinge baut, sah er die Herstellung eines Vibrationsmonitors als eine interessante Herausforderung an. Der Vibrationsmonitor besteht aus einer hölzernen Halterung (ein 3D-druckbares Design lässt sich sicherlich leicht erstellen) in Form eines einfachen „H", die zwischen die Finger einer behandschuhten Hand passt. Ein LIS331-Beschleunigungssensor mit einem ESP32-Modul samt OLED-Display, auf dem der Handwerker die Vibrationen ablesen kann, wird wie im Foto zu sehen auf der Oberseite des „H" befestigt (das Display im Vordergrund ist in

das Foto hineinmontiert). Der untere Schenkel des „H" sorgt für eine Ankopplung des Sensors an die Maschine. Ein kleines USB-Batteriepack versorgt das Gerät mit Strom und kann während des Einsatzes in die Tasche des Arbeitsanzuges des Bedieners gepackt werden. Obwohl das Gerät ursprünglich für die Messung von Vibrationen in Hand und Arm des Benutzers konzipiert wurde, die durch Handheld-Elektrowerkzeuge wie Bohrmaschinen oder Schleifgeräte verursacht werden, gibt es wahrscheinlich noch viele andere Anwendungsbereiche. Alles, was Sie tun müssen, ist den Sensor an der richtigen Stelle zu platzieren.

www.elektor-labs.com/1879

Verbesserung des Pretzel-Boards

Vor einiger Zeit habe ich beim erfolglosen Herumspielen mit dem Pretzel-Board [1] eine „Kinderkrankheit" ausgemacht. Nach einiger Online-Suche fand ich heraus, dass die Stromversorgung des WLAN-Teils (auch auf aktuellen Boards) ziemlich instabil sein kann. Als Abhilfe wurde in Foren vorgeschlagen, einen großen Pufferkondensator von 100 µF auf dem Steckverbinder des Boards anzubringen, um die Stabilität des WLAN-Teils zu verbessern. Das löst zwar



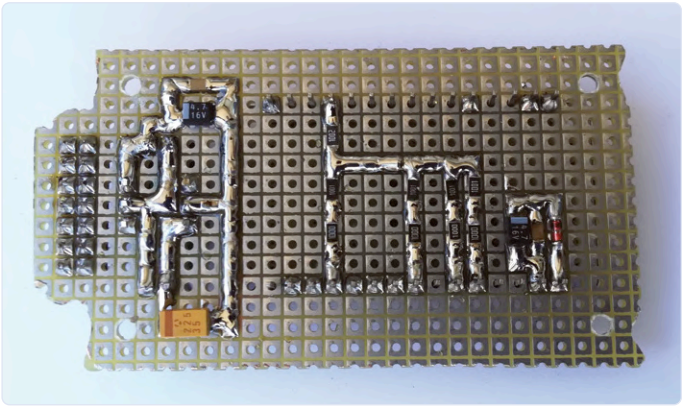
das Problem, aber ich zog eine elegantere Methode vor, als den Steckverbinder mit einem großen, dicken, hässlichen Kondensator zu blockieren.
Sie können das Ergebnis auf dem Foto sehen. Der Elko im SMD-Format mit einer Kapazität von 47 µF ist unauffällig an der Unterseite der Platine angebracht. Dazu brauchte ich bloß an der richtigen Stelle vorsichtig etwas Lötstopmaske von zwei Leiterbahnen abkratzen und den Kondensator mit einer geeigneten Lötspitze verlöten. Ein 100-µF-Typ hätte auch gepasst, aber ich hatte gerade keinen in meinen Sortierkästen.

Uwe Werner

[1] www.elektor.com/pretzel-board-iot-wifi-board

Wie man SMDs von Hand lötet

Heutzutage ist es fast unmöglich geworden, Elektronikprojekte ohne diese winzigen oberflächenmontierbaren Bauteile, besser bekannt als SMDs, zu verwirklichen. Auch wenn sie angeblich schwer zu löten sind, haben sich viele Leute Wege ausgedacht,



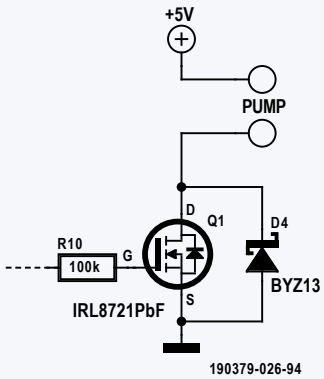
sie trotzdem für den Prototypenbau auf Lochraster zu verwenden. Hier ist ein Trick, der uns von Antoni Gendrau vorgeschlagen wurde. Wenn Sie einen weiteren haben, können Sie ihn gerne als Kommentar zum Projekt hinzufügen und die Elektronikwelt darüber informieren.

www.elektor-labs.com/1913

Upgrade für Dürrealarm

Elektor 07/2020, p. 39 (200213)

In der Ausgabe Juli/August 2020 von Elektor beschrieben wir einen „Dürrealarm“, eine kleine Schaltung, die akustisch und optisch warnt, wenn der Boden einer Pflanze zu trocken ist. Nützlich, aber noch nützlicher wäre es, wenn die Pflanze automatisch bewässert würde. Das war genau das, was der Konstrukteur auch dachte und mit einer Pumpe von SOS SOLUTIONS und ein paar zusätzlichen Teilen auch



erreichte (siehe Schaltung). Die Pumpe wird durch einen MOSFET (IRLB8721PbF) gesteuert, der wiederum über einen 100-kΩ-Widerstand (anstelle des ursprünglichen Summers) mit dem Ausgang der Schaltung verbunden ist. Eine Zenerdiode schützt den Transistor. Die Dauer des Warnimpulses von 100 ms muss zu einem „Bewässerimpuls“ von 1 s verlängert werden, indem man C4 auf 100 nF erhöht, je nach Wasserbedarf der Pflanze. Außerdem muss die Periodenzeit des astabilen Multivibrators auf zum Beispiel drei Sekunden erhöht werden, indem für C1 ein Kondensator von 2,7 µF oder 3,3 µF verwendet wird.
Mit dieser Modifikation und einem geeigneten Wasserreservoir kann die Pflanze monatelang unbewacht überleben.

www.sossolutions.nl/1150-peristaltic-liquid-pump-with-silicone-tubing

Stargate-NeoPixel-Uhr

Elektor 07/2020, S. 76 (200208)

In der Beschreibung von Peter Neufelds Heimatlabor und der Stargate-NeoPixel-Uhr, die er dort baute (Juli/August 2020), schrieben wir, dass das hölzerne Innere viele Stunden fachmännischer Laubsägearbeit darstellte. Zuviel der Ehre, wie Herr Neufeld uns erzählte: Das Innere ist ein Bausatz der Firma UGears - man muss nur die lasergeschnittenen Teile aus der Holzplatte herauspressen und zusammenbauen (ohne Klebstoff!).

https://ugearsmodels.com/de/date-navigator.html

Einfache Einschaltstrombegrenzung

Elektor 03/2020, S. 34 (200004)

Der Entwickler dieser handlichen kleinen Schaltung, die in der Ausgabe März/April 2020 veröffentlicht wurde, ist Kurt Kühni aus der Schweiz und nicht jemand aus dem Elektor-Labor (auch wenn wir selbst gerne darauf gekommen wären). Wir bitten Herrn Kühni um Entschuldigung!

190379-D-02

Aus dem Leben gegriffen

Bleifreies Löten und das europäische Regelwerk

Von **Ilse Joostens**

In meiner frühen Kindheit, Anfang der 1970er Jahre, lebten wir in einem Reihenhaus mit bleihaltigen Wasserleitungen. Als ich etwa fünf Jahre alt war, zogen wir mit unserer Familie in ein neues Haus, ohne Bleirohre, aber diesmal mit Asbestfußboden und mit Pentachlorphenol behandelten Fenstern. Letzteres klingt ziemlich unheimlich, muss ich sagen, und angesichts der derzeitigen Weltuntergangsszenarien scheint es fast ein Wunder zu sein, dass ich noch am Leben bin und dass Sie deshalb diese Kolumne lesen können. Seitdem ist die Welt offenbar viel gefährlicher geworden, und insbesondere die Europäische Union hat es für nützlich befunden, mit der Einführung der RoHS-Verordnung im Jahr 2006 einen regelrechten Kreuzzug gegen die Verwendung einer Reihe gefährlicher Stoffe in der Unterhaltungselektronik zu starten. Aus irgendeinem obskuren Grund ist vor allem Blei der Hauptverantwortliche und soll nun für Munition und Gewichte zum Fischen verboten werden.

Bleifreies Löten

Wenn Sie hobbymäßig löten, Reparaturen durchführen oder Prototypen bauen, lesen Sie nicht weiter und seien Sie den Göttern auf den Knien dankbar, dass Sie bleihaltiges Lot verwenden dürfen. Wenn Sie, wie ich, kommerzielle Produkte zusammenstellen, dann haben Sie großes Pech, denn Sie müssen bleifreies Lot verwenden.

Streng genommen könnte ich für Prototypen immer noch bleihaltiges Lot verwenden, aber um das Risiko einer Kontamination auszuschließen, löte ich alles bleifrei.

Die Zeit kurz nach dem Inkrafttreten der RoHS-I-Gesetzgebung (*Restriction of Hazardous Substances*) im Jahr 2006 war für die Elektronikindustrie recht intensiv. Produktionsverfahren mit bleifreien Lot waren noch nicht gänzlich ausgereift und die Erfahrungen mit den neuen Techniken waren äußerst begrenzt. In dieser Zeit bin ich auch mehr als einmal schweißgebadet aufgewacht, nachdem ich geträumt hatte, dass zwischen den Beinchen von SMD-Bauteilen immer messerscharfe Blechhaare wachsen, die schließlich zu Kurzschlüssen führen.

Was das maschinelle Löten betrifft, so sind die meisten Probleme gelöst, und auch mit den Zinnhaaren und anderen möglichen Unannehmlichkeiten ist es in der Praxis gar nicht so schlimm, obwohl ich den Eindruck habe, dass die Unterhaltungselektronik im Vergleich zu früher schneller und häufiger ausfällt. Es könnte natürlich auch an der Qualität der verwendeten Teile liegen, die mit den zunehmenden Billigimporten aus dem Fernen Osten systematisch gesunken ist.

Leider ist das manuelle bleifreie Löten noch immer eine kleine Herausforderung, und wenn man den unzähligen Bildern glauben darf, die im Internet kursieren, dann ist das Löten auf Mainboards von Computern sehr kompliziert und wird durch die höhere Schmelztemperatur des bleifreien Lotes nicht besser. Ja, suchen Sie mal im Netz die Bilder unter der Rubrik „Lötfehler“ und halten Sie Brandsalbe griffbereit, um ganz sicher zu gehen…

Mit meinen alten Lötstationen war das Löten von größeren bedrahteten Bauteilen vor allem auf Platinen mit großen Masseflächen und Lötinseln alles andere als einfach. Normalerweise hatte ein Lötkolben eine Temperatur von 420…450 °C, und selbst dann brauchte ich manchmal zwei Lötkolben gleichzeitig, um das Lot flüssig zu machen. Es sollte Sie nicht überraschen, dass die Lötarbeiten mit hoher Geschwindigkeit durchgeführt werden mussten. Mit modernen Lötstationen mit austauschbaren aktiven Spitzen ist das Arbeiten mit bleifreiem Lot zwar viel besser, aber das Löten auf Kupferflächen verläuft meiner Meinung nach nicht reibungslos. Das Entlöten von bedrahteten Bauteilen und die SMD-Nachbearbeitung ist nach wie vor eine recht heikle Aufgabe, die zu mehr Ausfällen von Bauteilen und Baugruppen führt. Wenn ich ausnahmsweise einmal wieder mit bleihaltigem Lot arbeite, ist das immer eine Erleichterung.

Wenn Sie vorhaben, mit der Elektronik zu beginnen oder wenn Sie ab und zu ein bisschen hobbymäßig löten, kann ich Ihnen nur abraten, bleifreies Lot zu verwenden. Die besseren Lötstationen mit aktiven Spitzen sind teuer, und eigentlich sehe ich keine spezifischen Gesundheitsrisiken bei der Arbeit mit bleihaltigem Lot. Natürlich sollten Sie es nicht essen. Bleifreies Lot dagegen enthält normalerweise ein etwas aggressives Flussmittel und seine Dämpfe sind nicht sehr gesund. Letzteres soll kein Freibrief sein, dass ab jetzt das bleihaltige Zeugs rauchen mag wie ein Schornstein - so viel sollte klar sein.

Recycling

Ich finde es auch fraglich, ob ein Verbot von Blei in der Elektronik wirklich so viel besser für die Umwelt ist. Statt der heutigen billigen Einweg-Elektronik wäre es besser gewesen, sich auf qualitativ hochwertigere und nachhaltigere Produkte zu konzentrieren. Zudem gibt es in den meisten europäischen Ländern Firmen, die sich dem Recycling von ausrangierten Geräten und wiederverwertbaren Abfällen widmen. Für Belgien seien die Organisationen Recupel, Bebat und Fost Plus genannt. Ich unterstütze den Recycling-Gedanken voll und ganz, aber der Beitritt zu den oben genannten Organisationen ist nicht nur teuer, sondern auch mit einem hohen Verwaltungsaufwand verbunden. Solange ich keine komplett fertigen Produkte oder Batterien verkaufe und die Verwendung von Kunststoffverpackungen unter Kontrolle halte, muss ich mich glücklicherweise nicht allzu sehr damit befassen. Nach dem, was ich von den Kollegen in Deutschland höre, ist es in Belgien nicht allzu schlimm. Die deutsche Stiftung EAR (Elektro-Altgeräte Register) ist nicht zimperlich - hier und da wird schon ein saftiges Bußgeld verhängt.

Trotz aller Regeln und Verbote sind zum Beispiel Quecksilberschalter und Fotowiderstände mit Cadmiumsulfid (Schauder) aus dem Fernen Osten immer noch leicht erhältlich, ganz zu schweigen von den teilweise lebensbedrohlichen Produkten, die zwar keine Normen erfüllen, aber trotzdem angeboten werden. Ist das nicht irgendwie merkwürdig?

200082-04

Ein vielversprechendes Projekt:

Neues LCR-Messgerät für 50 Hz bis 2 MHz

Messgenauigkeit und Komfort

Von **Jean-Jacques Aubry** (Frankreich)

Vor etwas mehr als sieben Jahren veröffentlichte Elektor eine Reihe von Artikeln über mein LCR-Messgerät mit 0,05% Genauigkeit. Mehr als fünfhundert Leser haben dieses Gerät mit dem von Elektor angebotenen Bausatz gebaut. Motiviert von diesem Erfolg beschloss ich, ein neues LCR-Messgerät zu entwickeln, das vor allem im Hinblick auf den Benutzerkomfort (einfachere Messung und erweiterte Funktionen) noch besser sein sollte. Hier ein Überblick des Projekts, das in der nächsten Ausgabe ausführlich vorgestellt wird.

Elektor Kickstarter?

Dieses Messgerät gehört zu den ersten Schaltungen, die wir in der neuen Kickstarter-Abteilung unserer Webseite *Elektor Labs* präsentieren. Dies soll uns dabei helfen, mehr über das Interesse unserer Leser an bestimmten Schaltungsprojekten zu lernen und ob sich bei ausreichend positivem Feedback die Produktion eines kompletten Bausatzes lohnt.

So funktioniert es

1. Schauen Sie sich das Projekt auf der Webseite Elektor Labs an (www.elektormagazine.de/labs/remake-lcr-meter)
2. Klicken Sie auf „Read the full project“.
3. Wir informieren Sie über (begrenzte, aktuell geltende) Preis- und Produktdetails.
4. Wenn Sie zu einem späteren Zeitpunkt kaufen möchten: Klicken Sie auf „Back This Project“.
5. Geben Sie Ihre E-Mail-Adresse ein
6. Fertig

Sie müssen nicht im Voraus bezahlen und es sind keine Bedingungen an Ihre Zusage geknüpft: Sie können jederzeit kündigen, wenn Sie nicht mehr interessiert sind. Wir informieren Sie per E-Mail über den Status des Projektes, die Produktion und das Erstverkaufsdatum.

Nicht vergessen

Alle unsere Unterstützer erhalten einen Rabatt, wenn das Produkt zur Verfügung steht!

Besserer Name

Elektor Kickstarter? Wir haben einen Wettbewerb gestartet, damit Sie uns bei der Suche nach einem besseren Namen helfen können:
elektor.com/kickstarter

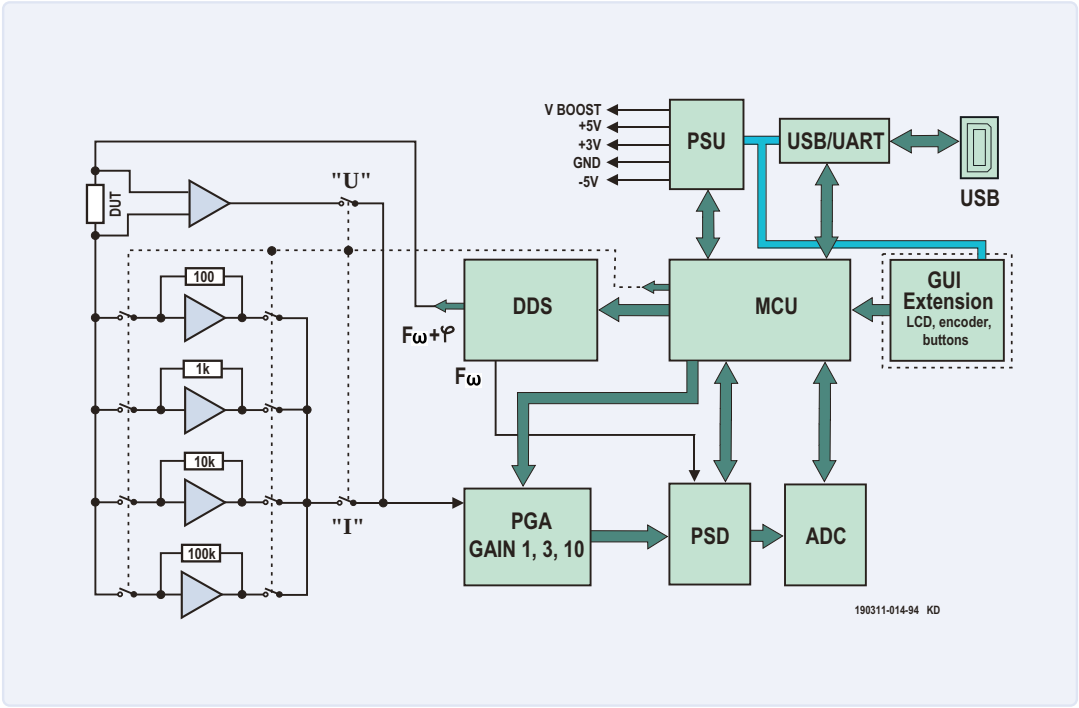


Bild 1. Schematische Darstellung des neuen LCR-Messgeräts. Alle Funktionen befinden sich auf derselben Platine, mit Ausnahme der Anzeige, der Bedienelemente und der möglichen Erweiterungen, die auf einer zweiten Karte zusammengefasst sind (GUI-Erweiterungsblock, gestrichelt umrandet).

Die Herausforderung, der ich mich stellte, war der Komfort der Messung und der erweiterten Funktionen:

- Testsignal von 50 Hz bis 2 MHz
- Wahl zwischen vier Spannungsbereichen: 100 mV, 200 mV, 500 mV oder 1 V
- zusätzliche DC-Polarisationsspannung von 5 V für Kapazitäts- und 50 mA für Induktivitäts-Messungen.

Um dem Anwender ein Gerät dieser Klasse schmackhaft zu machen, habe ich besonders auf Benutzerfreundlichkeit (die Kalibrierungen sind automatisiert) und auf Ergonomie geachtet: Ein Drehschalter und ein grafisches LCD-Display mit 240 x 128 Pixeln ermöglichen in Verbindung mit fünf Multifunktionstasten eine einfache und intuitive Navigation durch die Menüs, um zum Beispiel die Frequenz zu ändern oder die Messergebnisse anzuzeigen.

Ausgereizte Grenzen

Zur Messung der Werte passiver elektronischer Bauteile (Widerstand, Kondensator, Induktivität) ist die Impedanz (Z) ein wichtiger Parameter. Um diesen zu bestimmen, sind mindestens zwei Werte (Betrag und Phase) erforderlich. Normalerweise handelt es sich dabei um die Spannung über dem Bauelement und um den durch das Bauelement fließenden Strom. Wie sein Vorgänger aus dem Jahre 2013 verwendet das neue LCR-Messgerät die selbstabgleichende Brückenmethode, bei der der Strom-Spannungs-Wandler durch einen einfachen Operationsverstärker realisiert wird. Die Methode ist einfach, die Genauigkeit gut, die Kosten sind angemessen. Ihr Hauptnachteil besteht in der Regel in der Begrenzung des Frequenzbereichs durch den Operationsverstärker. Die meisten vergleichbaren Messinstrumente funktionieren nur bis 100 oder 200 kHz. Es ist mir gelungen, die Grenze auf 2 MHz zu erhöhen, ohne dass es dadurch teurer oder komplizierter wurde. Das Projekt wird in der nächsten Ausgabe ausführlich beschrieben und besteht aus zwei Platinen.
Erstens: **Die Hauptplatine**, deren Funktionen in **Bild 1** schematisch dargestellt werden:

- Eingangsschaltung: Die zu prüfende Komponente (Device Under Test = DUT) erhält ein Testsignal. Um den Einfluss der Kabel zu reduzieren, wird die Messung mit einer Konfiguration mit fünf Verbindungen durchgeführt.
- Sinusgenerator: Das Verfahren (Direkte Frequenzsynthese), das zur Erzeugung der Testfrequenz verwendet wird, liefert eine beliebige Frequenz im Bereich von 50 Hz bis 2 MHz, die auch mit variabler relativer Phase für den Synchrondetektor zur Verfügung steht.
- Programmable Gain Amplifier (PGA): 1-, 3- oder 10-fache Verstärkung, um den Phasendetektor unter optimalen Bedingungen anzusteuern.
- Phasendetektor: Erfordert ein perfektes Rechteck-Signal mit der gleichen Frequenz wie der Generator, dessen relative Phase jedoch variiert werden kann. Diese Phasenverschiebung des Schaltsignals relativ zur Sinuskurve ermöglicht die Messung der Signalanteile, die sich mit dem Eingangssignal in gleicher Phase befinden oder die dessen Quadratur-Anteile bilden.

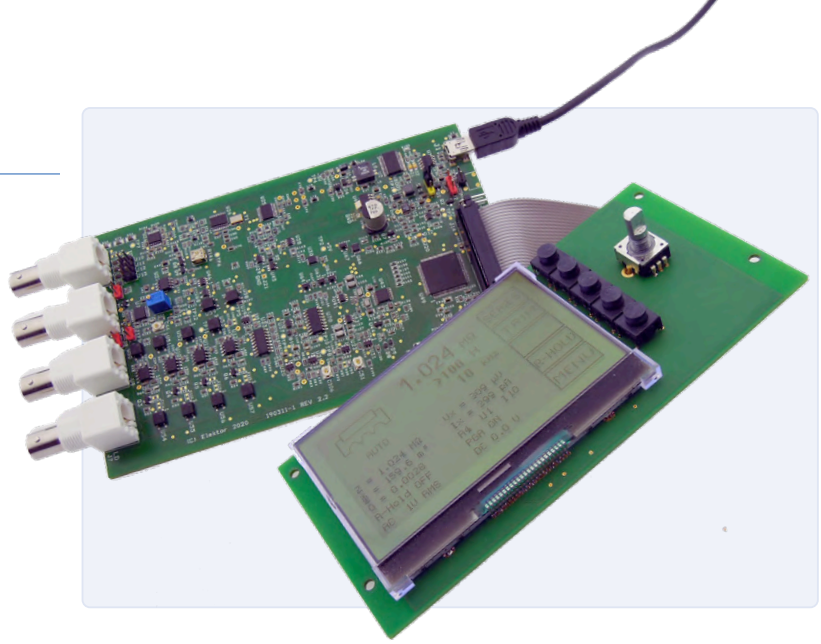


Bild 2. Die Kombination von Hauptplatine und Erweiterung ermöglicht einen unabhängigen Aufbau des Systems. Es ist auch möglich, die Hauptplatine mit einem PC zu verwenden.

- Ein Mikrocontroller und eine USB-Schnittstelle sind ebenfalls auf der Hauptplatine vorhanden.

Zweitens: Eine **Anzeige- und Schnittstellenkarte**:
- Grafische Anzeige
- Fünf Multifunktionstasten (Funktion auf dem Bildschirm) und ein Drehgeber

Die Merkmale des LCR-Messgeräts finden Sie in **Tabelle 1** weiter unten zusammengefasst:
Ich empfehle Ihnen, mein Projekt auf Elektor Labs [1] zu verfolgen.

Formlose Zusage mit Belohnung

Vor Beginn der Produktion des Bausatzes startet Elektor eine Kampagne zur Unterstützung des Projektes: Wenn Sie an dessen Realisierung interessiert sind und Sie den Bausatz erwerben möchten, können Sie sich ganz formlos registrieren. Die Produktion des Bausatzes wird ab 150 Zusagen gestartet. Als Gegenleistung für Ihr Engagement wird Ihnen das Kit zu einem reduzierten Preis angeboten.

Lieferumfang des Kits:

- Vorbestückte Hauptplatine mit allen aufgelöteten SMDs
- Vorbestückte Display-Karte mit allen aufgelöteten SMDs
- Verbindungselemente für beide Leiterplatten (hintergrundbeleuchtete LC-Anzeige, Anschlüsse, Drucktasten, Drehschalter, Taste)
- Flachbandkabel zur Verbindung von Anzeige und Hauptplatine
- Mini-USB-Kabel für PC-Verbindung und Firmware-Updates
- Hammond-Aluminiumgehäuse, gebohrte und gefräste Front
- Schrauben
- Kelvin-Clip mit 4-poligem BNC-Testkabel
- Handbuch

200309-02

Tabelle 1. Technische Daten

Anzeige	• Parameterwerte (primär und sekundär) • Äquivalente Schaltung: seriell oder parallel • Frequenz • |Z| • ϕ • Q oder D • DUT-Spannung und -Strom (V_x und I_x) • Prüfspannung (AC) und DC-Polarisationsspannungswert (DC) • Bereich einfrieren (R_Hold) • (rechts) Beschriftung der variablen Funktionstasten	
Messbereich	**Parameter**	**Wert**
	L	10 nH ➞ 100 H
	C	1 pF ➞ 100 mF
	R, |Z|	10 mΩ ➞ 100 MΩ
	Q	0 – 5000 für die Anzeige
	Φ	– 90,00 ° / +90,00 °
Testfrequenz	50 Hz bis 2 MHz in 54 vordefinierten Schritten + jede Frequenz im Bereich	
Verbrauch	Mit Tastatur- und Anzeige-Erweiterung	5 V / 420 mA
Software PC	für Windows, MacOSX	

Testbedingung:

Leerlauf-Testspannung	4 Werte: 0,1 V_{eff}, 0,2 V_{eff}, 0,5 V_{eff}, 1 V_{eff} ± 5 %
Polarisations-Spannung	Für C: 0 bis 5 V | Für L: 0 bis 50 mA

Genauigkeit der Hauptparameter (R, L, C) :

Bedingungen*	Nach Kalibrierung und Verwendung einer Testfrequenz in Übereinstimmung mit dem DUT (bei hohen Frequenzen spielen die parasitären Komponenten eine wichtige Rolle)
Fehler	Bis ± 0,1 % ±1 der letzten Ziffer

Sonstiges:

Messverbindungen	Vier Kelvin-Kabel mit BNC-Steckern
Versorgung	5 V_{DC} ± 5 %, über Mini-USB

Fehleranalyse

Tipps zu FMEA, hohen Strömen und mehr

Zusammengestellt von **C.J. Abate**

Übermäßige elektromagnetische Störungen, eine durchgebrannte Leiterplatte, eine explodierte Batterie und fehlerhafte Sensormesswerte: Dies alles kann auf einen einfachen Konstruktionsfehler zurückzuführen sein. Um sich als professioneller Ingenieur oder seriöser Elektroniker zu verbessern, müssen Sie aus Ihren Konstruktionsfehlern lernen, aber auch aus den Fehlern anderer. In einer neuen Serie mit dem Titel „Fehleranalyse“ tauschen wir Erkenntnisse von Mitgliedern der Elektor-Community über ihre Konstruktionsfehler aus und was sie aus den Erfahrungen gelernt haben. Und natürlich stellen wir auch Tipps und Tricks vor, die Ihnen helfen, Ihre Entwurfs-, Test- und Programmierkenntnisse zu verbessern.

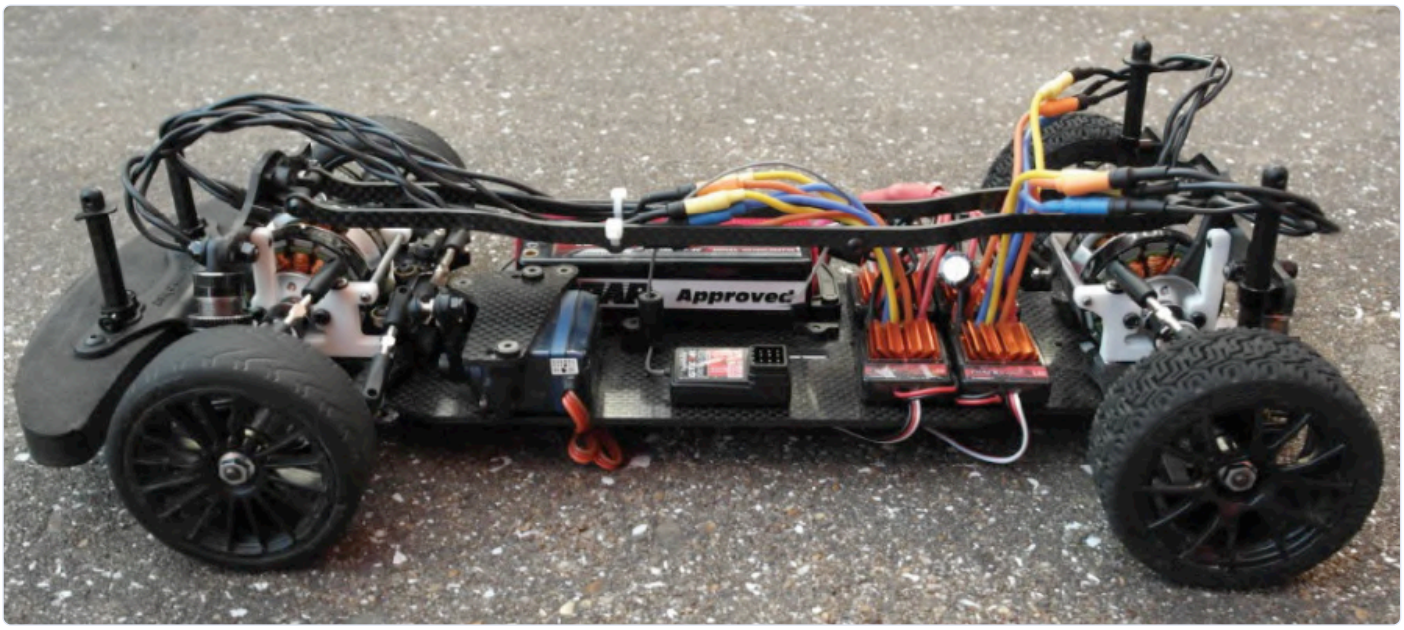


Bild 1. Das RC-Projekt von Rajesh Nakarja.

Verbessern Sie die Zuverlässigkeit Ihres Projekts mit FMEA

Unabhängig davon, wie kompliziert Ihr Projekt ist, kann die Anwendung der Fehlermöglichkeits- und -einflussanalyse (FMEA) den Unterschied zwischen Erfolg und Katastrophe ausmachen. Der in Stockholm ansässige schwedische Ingenieur Rajesh Nakarja weiß davon ein Lied zu singen. Hier erzählt er die Geschichte eines problematischen RC-Designs auf Arduino-Basis und natürlich auch, was er aus dieser Erfahrung gelernt hat:

„Als Student liebte ich RC-Modellautos. Mein damaliger Abschluss in Embedded Control führte mich zur Entwicklung eines kundenspezifischen DSP-Controllers für die Drehmomentverteilung von RC-Fahrzeugen im Maßstab 1:10. Das mit vier unabhängig voneinander angetriebenen Rädern und verschiedenen Sensoren ausgestattete Fahrzeug war in der Lage, extreme Geschwindigkeiten zu erreichen, die normalerweise nur bei Wettbewerbsrennen auf einer Rennstrecke vorkommen (**Bild 1**). Mit all der Echtzeitverarbeitung war das Fahrzeug auch bei voller Leistung stabil, da es in der Lage war, Traktionsverluste zu erkennen und sich mit Hilfe von Gyroskopen selbst zu korrigieren. Das Programm wurde vollständig mit dem Simulink-Coder entwickelt und auf einem kundenspezifischen STM32F4-Board implementiert. Es wies zahlreiche Sicherheitsvorkehrungen auf: Über-/Unterläufe,

> Moderne Werkzeuge und Arbeitsabläufe wie das modellbasierte Design ermöglichen eine automatisierte Testabdeckung während der Entwicklung.

Verlust von Funk- oder Sensordaten, all diese Dinge würde der Code sicher auffangen und das Fahrzeug zum Stillstand bringen. Ich war zuversichtlich, dass das Modell tatsächlich sicher fahren würde.
Obwohl das Modell nur ein paar Kilogramm leicht war, bedeuteten die Geschwindigkeit und die Leistung von vier 40-A-BLDC-Motoren, dass ein Aufprall immer noch ziemlich verheerend für jeden und alles sein konnte. Daher war ich sehr vorsichtig, als ich das Modell draußen im öffentlichen Verkehrsraum benutzte.
Um das Auto zu steuern, hatte ich einen Xbox-Controller mit einem Arduino-ZigBee verbunden, der das Auto in Echtzeit steuern und Gas geben konnte. Während der Auto-Controller selbst zuverlässig und robust konstruiert wurde, war der Arduino-Controller eher ein fragi-

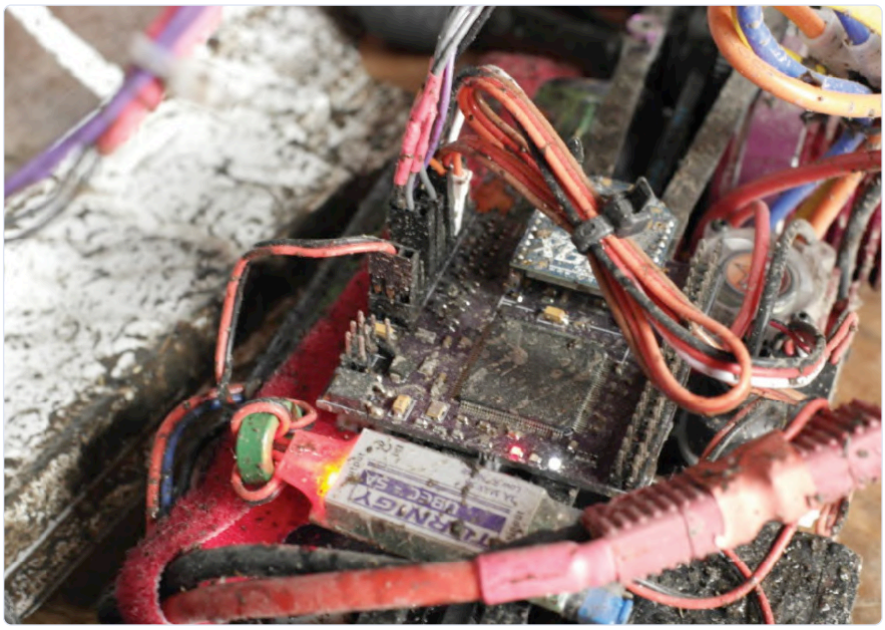


Bild 2. Eine Katastrophe kann jederzeit passieren!

ler Hack, und die einzige zusätzliche Sicherheitsmaßnahme bestand darin, das Auto im Falle eines Funkausfalls abzuschalten.
Eines Nachmittags, während eines Tests in einem offenen Park, gelang es dem USB-Controller, kurzzeitig die Verbindung mit dem Arduino/ZigBee-Gerät zu verlieren. Dadurch stürzte der USB-Treiber ab und gab einen Vollgas-Befehl an das Funkgerät, das weiter sendete. Das Fahrzeug fuhr dann mit voller Geschwindigkeit geradewegs auf eine überfüllte Hauptstraße zu.
Die ZigBee-Module, die ich verwendet hatte, besaßen eine große Reichweite, was bedeutete, dass das Fahrzeug weit außer Sichtweite sein musste, bis die Sicherheitsmaßnahme wirkte. Als ich hastig die Batterien vom Arduino entfernte, flog das Fahrzeug an einer verdutzten Menschenmenge vorbei und krachte mit einer Geschwindigkeit von vielleicht 50 oder 60 Sachen gegen einen Bordstein. Zum Glück wurde niemand verletzt, und beim Aufprall riss die Lithium-Batterie aus dem Chassis, so dass der Strom ausfiel. Peinlich berührt rannte ich unter Entschuldigungen zu dem jämmerlichen Haufen von Kohlefaserteilen, klaubte ihn auf und ging mit meinem zerstörten Projekt nach Hause (**Bild 3**).
An diesem Tag habe ich gelernt, dass selbst einfachste Fehler an allen Sicherheitsmaßnahmen vorbei schlüpfen können, ganz gleich, wie sie beschaffen waren, es reicht ein winziger Bug, um potentiell eine Katastrophe auszulösen. Seither nehme ich die Analyse der Auswirkungen eines Fehlers viel ernster und wende sie prinzipiell bei jedem Projekt, an dem ich arbeite, von Anfang an an.
Moderne Werkzeuge und Arbeitsabläufe, auch beim beschriebenen RC-Modell, ermöglichen eine automatisierte Testabdeckung während der gesamten Entwicklung. Zusammen mit einem sich entwickelnden Release-Plan bedeutet dies, dass Fehler frühzeitig vermieden oder schnell im Release-Prozess abgefangen werden. Auch wenn gelegentlich noch Dinge schief gehen können, bedeutet ein guter Testplan, dass ein Fehler (fast) nie mehr als einmal geschehen kann. Die Bilder zeigen das RC-Fahrzeug und die Steuerplatine. Die gesamte Elektronik wurde übrigens vollständig gecoatet, um Schäden durch Schmutz oder Wasser zu verhindern. Ein Video auf YouTube beschreibt das Projekt ausführlicher [1]“.

Hohe Ströme wollen gesteuert werden

Lars Krüger ist ein in Lehrer aus Potsdam, der seit acht Jahren Elektor liest und seit 1992 Elektrofahrräder entwirft. Wie Sie sich vorstellen können, hat er im Laufe der Jahre viel über die Steuerung hoher Ströme, über das Testen von Schaltungen und über E-Bikes gelernt.

„Früher, als ich jünger war, habe ich versucht, alles in sehr kurzer Zeit durchzuziehen. Eine Idee am Morgen musste am Abend fertig sein. Einige Problemchen haben mich nicht sonderlich gestört, wenn der Rest in Ordnung war und lief. Aber das führte zu vielen defekten und durchgebrannten Halbleitern, da es ja um Spannungen jenseits von 5 V und/oder Ströme ging! 1992 habe ich mein erstes Elektrofahrrad gebaut. Nachdem ich nur ein Viertel eines Buches gelesen hatte, baute ich eine PWM, die mit etwa 100 Hz läuft. Hinzu kam ein Satz von zehn BUZ-MOSFETs, eine große 100-Ah-Autobatterie und der 2-kW-Anlasser eines Citroen CX 2.5D.

Bild 3. Lars Krüger lernte bei seinen E-Bike-Projekten das Steuern hoher Ströme. Sehen Sie sich sein E-Bike von 1994 an.

TEILEN SIE IHRE FEHLERANALYSE

Möchten auch Sie über einen speziellen elektronischen Fehlerteufel berichten? Dann füllen Sie einfach unser Formular „Fehleranalyse“ aus:
www.elektormagazine.com/pages/error-analysis-submission

Nach dem ersten „Vollgas“ war der Motor richtig an: Die MOSFETs waren durchgebrannt und das Motorrad, das eine Geschwindigkeit von über 60 km/h erreichte, war nicht mehr zu stoppen. Natürlich war das Abschalten der Versorgungsspannung der PWM nutzlos, da die durchgebrannten FETs sich nicht mehr um die Gate-Spannung kümmerten. Ich hatte Glück, dass ich eine Sicherung eingebaut habe, die bei starkem Bremsen durchbrannte. Als nächsten Schritt baute ich einen großen Schalter aus dickem Kupfer und einer Feder ein. Dieser wurde anstelle des früheren Potentiometers an den Bowdenzug angeschlossen. Das Rad hat mit diesem puristischen Setup - 10 m in die Pedale treten und dann „ssss“ ab gehts - einfach Spaß gemacht. In den engen Gassen meiner Heimatstadt war das Gefährt zwar unkontrollierbar, doch ich lächelte von einem Ohr zum anderen. Aus dieser Geschichte habe ich gelernt, dass die Steuerung (Begrenzung) des Stroms eine ziemlich notwendige Sache ist, wenn man es mit einem starken Motor zu tun hat. Diesen Aspekt darf man weder ignorieren noch vergessen. Außerdem muss die Steuerung schnell genug sein muss, um die gewünschten Stromanstiegsgeschwindigkeiten zu bewältigen. Ansonsten gibt es unerwünschte Schwingungen im Regelkreis. Später studierte ich und lernte einiges über Regeltechnik. Theoretisch. Der Stoff hatte leider wenig mit dem wirklichen Leben zu tun. Mehr gelernt habe ich durch Simulations-Tools wie PSpice oder seinem Nachfolger SIMetrix. Von da an hatte ich viel weniger Probleme mit der Steuerung hoher Ströme. Aber natürlich ist das Durchbrennen von MOSFETs in meinem Heimlabor nicht vollständig vorbei. Manchmal rutscht eine Sonde ab und produziert während der Messung einen Kurzen. Gut, wenn man eine Schutzbrille trägt und das Stromnetz des Raumes von der Schaltung getrennt ist, an dem man testet. Das Foto zeigt übrigens mein zweites E-Bike von 1994. Das erste war wohl so schnell, dass ich keine Zeit hatte, ein Foto davon zu machen.“

MOSFETs, Bipolartransistoren und PLL-Steuerung

Robert Owen ist ein in Stenløse, Dänemark, ansässiger Senior System Engineer mit langjähriger Erfahrung in verschiedenen Bereichen der Elektronik. Hier zeigt er, was er einst bei der Entwicklung der Video-Sync-Separation und der Strahlablenkungssteuerung für auf Kathodenstrahlröhren aufbauende Videoprojektoren gelernt hat. Mit einem MOSFET-Transistor in der Schaltung geriet er aber bald auf Kriegsfuß. Glücklicherweise löste er das Problem ohne große Umstände und lernte dabei ein wenig über MOSFETs, BJTs und PLL-Steuerung.

„Vor vielen Jahren, in einem anderen Leben, war ich an der Entwicklung der Sync-Trennung und der Strahlablenkungssteuerung für CRT-basierte Videoprojektoren beteiligt. Um Benutzern entgegenzukommen, die mit ihrem 200-Dollar-Videorekorder ein perfektes Bild auf unseren 10.000-Dollar-Projektoren erzeugen wollten, mussten wir die PLL-Steuerschleife während des Schaltintervalls des Kopfes vom Videorekorder aus öffnen. Der Steuerchip verfügte über einen Eingang für die Phasenregelschleife, der für diesen Zweck geeignet schien. Also habe ich einen MOSFET-Transistor als variablen Widerstand eingesetzt und eine neue Platine entworfen. Die Schaltung öffnete jedoch nicht nur die Schleife nicht, sondern schien auch zu versuchen, dem Synchronsignal noch enger zu folgen, was zu einem Bildabriss am oberen Bildschirmrand führte. (Unsere Projektoren zeigten alle Zeilen des Bildes, im Gegensatz zu einem Fernsehgerät, das an jedem Ende zehn oder mehr Zeilen abschneidet.) Was ich entdeckte, war, dass der Steuereingang empfindlich auf die „Stromeinspeisung“ durch die riesige Gate-Source-Kapazität reagierte. Ich konnte den MOSFET durch einen JFET oder einen einfachen Bipolartransistor ersetzten und damit das Problem lösen, ohne dass Änderungen am Layout erforderlich waren.“

200303-02

WEBLINK

[1] **Independent Wheel Drive 1/10 Scale RC Car - Project Aelith, SiliconWitch Couture, YouTube, 2014**: https://youtu.be/ET1HEyqEQJQ

Von Entwicklern für Entwickler

Tipps & Tricks, Best Practices und andere nützliche Infos

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

VON DER IDEE ZUM PRODUKT – TEIL 6

Bisher wurde die Elektronik unseres unseres neuen Produkts zur Produktionsreife gebracht und Sie haben ein Päckchen mit fertigen Platinen erhalten. Nun ist es an der Zeit zu prüfen, ob sie auch wie vorgesehen funktionieren, was Tests voraussetzt. Aber wie wollen Sie testen? Glücklicherweise haben Sie über dieses schwierige Thema bereits vor dem Start der Produktion nachgedacht und sind deshalb gut vorbereitet.

WAS MUSS GETESTET WERDEN?

Idealerweise muss jede Funktion oder jedes Merkmal der Schaltung getestet werden. Wie machen Sie das? Wie definieren Sie einen Test? Testen Sie jedes Board? Automatisieren Sie das Testen? Kann der Testvorgang überhaupt automatisiert werden? Wäre automatisiertes Testen möglich gewesen, wenn Sie in einer früheren Phase des Produktdesigns nur daran gedacht hätten? Autsch! Entschuldigung, hat das wehgetan?

EINGEBAUTE TESTS

Die ideale Schaltung kann sich dank eingebauter Tests (build-in tests, BITs) ganz von selbst vollständig testen. Aber selbst wenn er dies kann, müssen das Testverfahren aktiviert und die Ergebnisse interpretiert werden, und irgendwie müssen die fehlerhaften Platinen von den guten getrennt werden. Ein eingebauter Test kann in Software implementiert werden (vergessen Sie nicht, den Test zu testen), aber nicht jede Schaltung ist software-basiert, und Software kann auch nicht alles testen. Eine Testroutine, die alle Pixel oder Segmente einer Anzeige durchläuft, ist schön, aber ohne einen Beobachter ist sie nutzlos.

ZURÜCK ZUM ZEICHENBRETT!

Eine Sache kehrt in dieser Artikelserie häufig wieder: Jedes Mal, wenn Sie glauben, einen Schritt vorwärts zu machen, kommen Sie am Ende wieder zur Phase der Produktspezifikation zurück. Hier ist es genauso. Das Testen der Platinen sollte Teil der Designspezifikationen sein. Wenn Sie von Anfang an über das Testen nachdenken, dann können Sie das Design so anpassen, dass Testen so einfach und so umfassend wie möglich durchgeführt werden kann. Denken Sie immer daran, dass das Testen letztendlich von Menschen oder Maschinen durchgeführt wird, die keinerlei Kenntnisse über das zu testende Gerät (device under test, DUT) haben. Deshalb gilt: je einfacher das Testverfahren, desto besser!

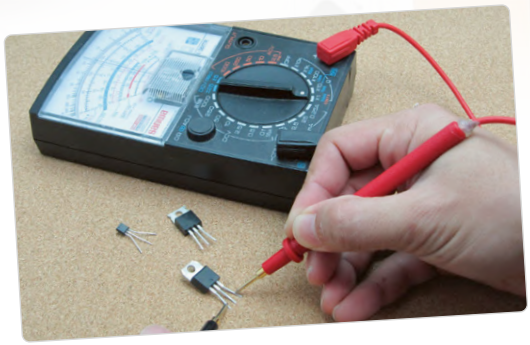
TESTVERBINDER UND TESTPUNKTE

Das Hinzufügen eines Testverbinders in die Schaltung ist eine mögliche Lösung. JTAG zum Beispiel ist eine beliebte Methode, um die Verbindungen von Logikbausteinen wie Mikrocontrollern und FPGAs mit der Leiterplatte und auch zwischen den Bausteinen zu testen. Ein Test-Steckverbinder kann analoge und/oder digitale Ein- und Ausgänge kontaktieren, die von einem Tester, sei es Mensch oder Maschine, angeregt und gelesen werden können. Der Tester kann Tastendrücke ausführen und ein entsprechendes Ausgangssignal überwachen. Durch das Eingeben eines Tastendrucks erfahren Sie jedoch nicht, ob die Taste selbst vorhanden ist und funktioniert. Außerdem kostet ein Testverbinder Platz auf der Platine und Geld, selbst wenn man die zusätzlichen Komponenten nicht mitzählt, die erforderlich sind, damit er funktioniert. Um dies zu umgehen, verwenden Designer oft Testpunkte. Oszilloskope und ähnliche Instrumente können programmiert werden, um zu prüfen, ob (komplexe) Signale an den Testpunkten innerhalb der erlaubten Grenzen bleiben. Der Vorteil liegt eindeutig in der geringen Größe der Testpunkte und einer entsprechenden Freiheit in der Platzierung, aber für den Anschluss an diese Testpunkte ist eine spezielle Testvorrichtung erforderlich. Außerdem müssen die Testinstrumente programmiert und Testvorrichtungen entwickelt und getestet werden, und all dies kostet natürlich Zeit und Geld.

Die Qualität Ihrer Testverfahren kann den Unterschied ausmachen, ob Sie sich auf einer tropischen Insel entspannen oder aus dem Geschäft katapultiert werden.

ENTWURF EINES UMFANGREICHEN TESTVERFAHRENS

Unabhängig davon, wie die Tests durchgeführt werden, muss ein klares Testverfahren definiert werden, das zu einer zuverlässigen Pass/Fail-Entscheidung führt. Das Verfahren sollte jeden Aspekt der Schaltung abdecken und muss so einfach wie möglich zu implementieren, zu verstehen und auszuführen sein, und dies umso mehr, wenn die Produktionsmengen steigen. Die Entwicklung eines solchen Verfahrens sollte nicht auf die leichte Schulter genommen werden. Wie bereits erwähnt, sollte der Tester letztendlich keine oder nur wenig Schulung benötigen, nicht nur, um die Kosten niedrig zu halten, sondern auch, um das Risiko von Fehlern und Irrtümern beim Testen zu minimieren.



Die Testmethode hängt nicht nur von der Anzahl der zu testenden Produkte ab.

ANDAUERNDE TESTS

Funktionstests sind schön, aber vielleicht nicht gut genug. Denken Sie auch über Ausdauertests nach. Sie müssen nicht jede Platine testen, aber es ist sinnvoll, von jeder Charge einige Proben einer gründlichen Sichtprüfung zu unterziehen und sie in eine Klimakammer für Temperatur- und Feuchtigkeitstests zu legen. Überleben die Schaltungen? Abhängig von der Anwendung des Endprodukts können auch andere Belastungstests wie Fall- und Schocktests erforderlich sein.

AUF FEEDBACK HÖREN

Es ist auch eine gute Praxis, auf das Feedback der Personen zu hören, die die von Ihnen angegebenen Tests durchführen, da dies zur Verbesserung der Boards beitragen kann, zu einer besseren Qualität und Endbenutzererfahrung, weniger Rückgaben, einer billigeren Produktion und/oder schnelleren Tests.

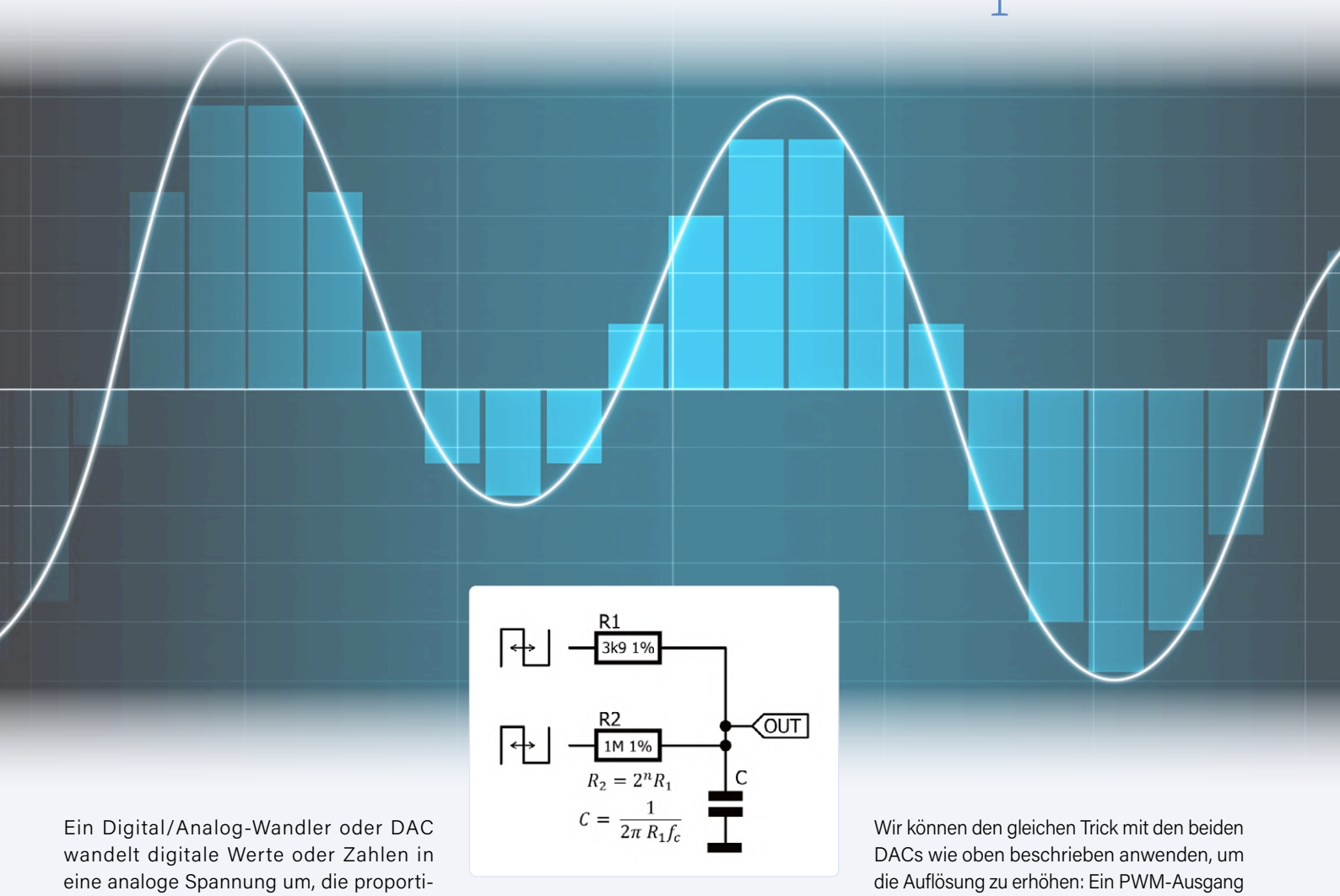
KONFORMITÄTSPRÜFUNG

Es gibt auch Konformitätsprüfungen, um die man sich kümmern und auf die man sich vorbereiten muss. Dies ist insofern einfacher, als dass Sie sich keine Testverfahren ausdenken müssen, da diese von den Behörden festgelegt wurden (auch wenn Sie die Behörde für die Konformitätsprüfung möglicherweise anweisen müssen, Ihr Produkt korrekt zu behandeln). Doch der schwierige Teil kann darin bestehen, Ihr Produkt konform zu machen. Auch dies kostet Zeit und Geld. Sie müssen nicht nur die Konformitätsprüfung selbst bezahlen, sondern auch jemanden, der herausfindet, welche Normen von Ihrem Produkt in welchem Land betroffen sind und welche Auswirkungen dies auf Ihr Produkt hat. Es könnte interessant sein, in einige Pre-Compliance-Testgeräte zu investieren, damit Sie potenzielle Probleme frühzeitig erkennen und korrigieren können. Und, noch einmal, auch die Vorbereitung auf die Konformitätsprüfung sollte erfolgt sein, bevor Sie anfangen, echtes Geld für Ihr Produkt auszugeben.

TESTEN KOSTET GELD

Da die Vorbereitung, das Implementieren und das Ausführen guter Testverfahren ein integraler Bestandteil des Produktdesigns und der Produktherstellung ist, erfordert es Zeit und Aufwand und kostet damit Geld. Wenn Sie nicht in einer perfekten Welt leben, wird das Testen auch eine Menge von fehlerhaften Platinen entdecken. Diese können Sie einfach wegwerfen oder versuchen, sie zu reparieren. Wofür auch immer Sie sich entscheiden, auch hier ist ein Preis zu zahlen. Die einzige Möglichkeit, diese Kosten zu senken, besteht darin, die Qualität Ihrer Platine und die Art und Weise, wie sie produziert wird, zu verbessern. Hier kommen Qualitätsingenieure ins Spiel. Aber wenn Sie versuchen, die Anzahl der ausgefallenen Geräte durch eine Vereinfachung des Testverfahrens zu reduzieren, können Sie damit rechnen, später vom After-Sales-Service gebissen zu werden. Und genau das ist ja der Sinn des Produkttests: unzufriedene Kunden vermeiden, die sich über ihre Produkte beschweren und sie zurückschicken, und Ihnen einen schlechten Ruf verschaffen. Letztlich ist der After-Sales-Service viel teurer, als es von Anfang an richtig zu machen. Es kann Sie sogar Ihr Geschäft kosten.

Dual PWM
verbessert die 8-Bit-Audioqualität



Ein Digital/Analog-Wandler oder DAC wandelt digitale Werte oder Zahlen in eine analoge Spannung um, die proportional zum digitalen Eingangswert ist. Die Ausgangsauflösung wird durch die Anzahl der Bits bestimmt, die der DAC verarbeiten kann, sozusagen seine Wortbreite. Um die Auflösung zu erhöhen, kann man einfach einen DAC mit einer größeren Wortbreite nehmen. Eine andere Lösung besteht darin, zwei DACs mit niedriger Auflösung parallel zu schalten. Wenn wir einen DAC verwenden, um einen groben Wert zu erzeugen, kann ein zweiter DAC ihn dann „feinabstimmen", wenn wir seine Ausgangswerte richtig skalieren. Auf diese Weise kann man theoretisch mit zwei 8-Bit-DACs eine 16-Bit-Auflösung erreichen. In der Praxis ist die Auflösung jedoch geringer, da die Skalierung unvollkommen ist, wenn keine sehr präzise Elektronik verwendet wird. Dies wäre aufwendiger und teurer als einfach einen 16-Bit-DACs zu verwenden, und daher wird diese Technik nicht sehr oft genutzt.

Die meisten Mikrocontroller verfügen über keinen dezidierten DAC. Statt dessen ist es gängige Praxis, zur Erzeugung analoger Signale die Pulsweitenmodulation (PWM) zu verwenden. Die Tiefpassfilterung eines PWM-Signals führt zu einer Spannung, die proportional zur Pulsbreite ist. Die Auflösung wird nun durch die Auflösung des PWM-Signals bestimmt. Arduino verfügt beispielsweise über die Funktion `analogWrite()`, um PWM-Signale mit einer Auflösung von 8 Bit zu erzeugen. Der Arduino Uno kann bis zu sechs dieser Signale gleichzeitig ausgeben (an den mit einer Tilde „~" beschrifteten Pins).

Wir können den gleichen Trick mit den beiden DACs wie oben beschrieben anwenden, um die Auflösung zu erhöhen: Ein PWM-Ausgang erzeugt den groben Wert, ein zweiter sorgt für die Feinabstimmung. Die Skalierung kann mit den Widerständen vorgenommen werden, die Teil eines Tiefpassfilters sind. Die resultierende Schaltung ist einfach. Natürlich ist durch die Toleranz der Widerstände eine echte 16-Bit-Auflösung schwer zu erreichen, aber mit 1%-Typen dürfte die nutzbare Auflösung fast 14 Bit betragen.

Aber, so höre ich Sie sagen, der Controller des Arduino Uno kann auch 16-Bit-PWM, warum also nicht stattdessen diese verwenden? Der Grund ist die Geschwindigkeit. Bei gleicher Taktfrequenz besitzt die 16-Bit-PWM nur die halbe Ausgaberate der 8-Bit-PWM. Eine hohe Ausgangsfrequenz oder Abtastrate ist zum Beispiel für Audio-Anwendungen interessant.

www.openmusiclabs.com/learning/digital/pwm-dac/dual-pwm-circuits/

Elektronische Last
Siglent SDL1020X-E

Von **Harry Baggen**

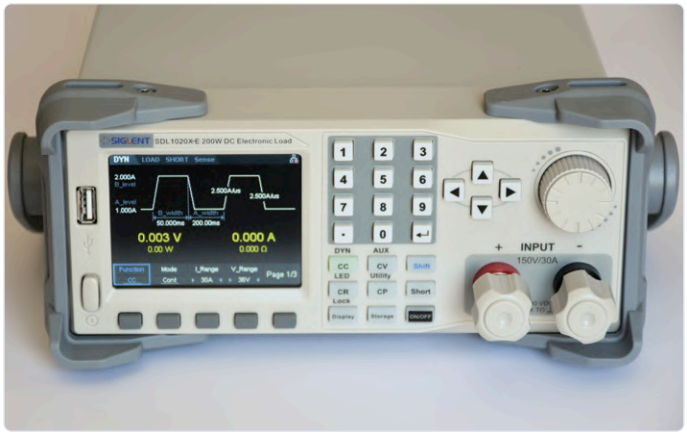
Für den Test von Stromversorgungen, Batterien und Akkus ist eine elektronische Last ein sehr nützliches Tool. Eine solche Last imitiert nicht nur einen simplen Festwiderstand, sondern kann auch als Konstantstrom- oder -spannungslast fungieren oder zwischen verschiedenen Lastwerten umschalten. Das Siglent SDL1020X-E bietet viele solcher Möglichkeiten und eignet sich für Spannungen bis 150 V und Ströme bis 30 A bei einer maximalen Verlustleistung von 200 W.

Bei der Entwicklung oder dem Test von Stromversorgungen ist es wichtig, dass sie unter verschiedenen Lastarten stabil bleiben, schnell auf Laständerungen reagieren und störungsarm sind. In den letzten Jahren scheint es einen zunehmenden Bedarf an Geräten zu geben, die eine Spannungsquelle auf unterschiedliche Weise belasten können. Besonders bei chinesischen Messgeräte-Herstellern sind neuerdings recht viele „elektronische Lasten" im Angebot. Auch Siglent hat in den letzten Monaten eine Reihe von elektronischen Lasten in das Programm aufgenommen. Davon habe ich das Modell das SDL1020X-E getestet. Es handelt sich dabei um die kleinste Version der Reihe mit einer maximalen Verlustleistung von 200 W. Es gibt auch eine 300-W-Version und in beiden Leistungsklassen gibt es Typen ohne Suffix „-E", die eine höhere Ablesegenauigkeit bieten.

Stabiles Gehäuse

Bei Siglent-Geräten habe ich schon einmal darauf hingewiesen, dass sie in der Regel mit einem robusten Gehäuse ausgestattet sind – das ist auch hier der Fall. Was noch auffällt: Das Gerät ist mit fast 40 cm ziemlich tief. Wenn Sie das Gerät in ein Regal stellen wollen, sollten Sie genügend Platz dafür haben.
Die Frontplatte enthält alle Steuertasten, zwei dicke Anschluss-

Das Gerät ist viel tiefer als andere Geräten von Siglent.



Das Display zeigt die Einstellung für dynamische Wellenformen.

Die Rückseite enthält eine ganze Reihe von Anschlüssen und Kühlschlitze.

Die großen Schraubklemmen sind zwar robust, erlauben aber leider nicht die Verwendung von Bananensteckern.

klemmen und ein 3,5“-LC-Display, auf dem alle Informationen angezeigt werden. Unterhalb des Bildschirms befinden sich fünf Schaltflächen, deren jeweils aktuelle Funktion auf dem Bildschirm angezeigt wird.
Auf der Rückseite befindet sich der Netzanschluss sowie Buchsen für USB, LAN, RS232 und zweimal BNC für Messsignale (Spannung und Strom der Last, zum Anschluss zum Beispiel an ein Oszilloskop). Am unteren Ende befindet sich ein Klemmenblock mit einer ganzen Reihe von Anschlüssen. Die wichtigsten sind der Sense- und der Trigger-Eingang. Auf der linken Seite befinden sich Lüftungsschlitze zur Kühlung des großen Kühltunnels, der innen fast über die gesamte Tiefe des Geräts verläuft. An der Vorderseite des Tunnels befindet sich ein temperaturgeregelter Lüfter. Bei kleineren Lasten ist das SDL1020X-E daher praktisch geräuschlos.

Möglichkeiten

Das SDL1020X-E bietet so viele Möglichkeiten, dass es dieses Review sprengen würde, wollte ich auf alle Details eingehen, so dass ich mich auf die wichtigsten Aspekte beschränken muss. Wer dieses Gerät kaufen will, sollte daher unbedingt auch das Datenblatt oder das Handbuch durchlesen.
Zunächst die grundlegenden Möglichkeiten. Sie können einen konstanten Strom (CC), eine konstante Spannung (CV), eine konstante Leistung (CP) oder einen konstanten Widerstand (CR) vorwählen. Das Display zeigt dabei immer die gemessenen Werte für Spannung, Strom, Leistung und Widerstand. Es gibt zwei Strombereiche (5/30 A) und zwei Spannungsbereiche (36/150 V), die sich hauptsächlich auf die Messgenauigkeit auswirken. Eine separate Batterietestfunktion ermöglicht unter anderem die Entladung eines Akkus mit konstanten Werten für Strom, Widerstand oder Leistung bis zu einer eingestellten, unteren Schwellenspannung, wobei das Display die Entladekapazität und die aktuelle Leistung anzeigt.
Am interessantesten sind die dynamischen Testmöglichkeiten, denn diese sind mit einfachen Mitteln wie Festwiderständen nicht so leicht zu imitieren. Wenn die dynamische Testfunktion (DYN) eingeschaltet ist, erscheint eine rechteckförmige Kurve auf dem Bildschirm. Das Gerät schaltet dynamisch zwischen zwei Werten um, wobei alle Parameter einstellbar sind: die unteren und oberen

Der Testaufbau.

Lastwerte für CC/CV/CR/CP, die Zeiten für die unteren und oberen Werte sowie die Steilheit der ansteigenden und abfallenden Flanke. Die Schaltfrequenz kann bis zu 25 kHz betragen, was für die meisten Tests sicherlich ausreicht. Der dynamische Test kann auch mit einem externen oder manuellen Triggersignal gesteuert werden. Mit Hilfe der Listen- und Programmfunktionen kann man komplexere Belastungen in mehreren Schritten zusammenstellen. Dieser Modus ist sehr mächtig. Bei der Listenfunktion (maximal 100 Schritte pro Kurvenform) ist die Belastungsart (zum Beispiel Konstantstrom) bei allen Schritten identisch. Bei der Programmfunktion (maximal 50 Schritte pro Kurvenform) kann bei jedem Schritt auf eine andere Belastungsart umgeschaltet werden (etwa von konstantem Strom auf Widerstand). Für beide Funktionen lassen sich alle Parameter pro Schritt in einer Tabelle eingeben.
Standardmäßig werden alle gemessenen Werte auf dem Display angezeigt. Im dynamischen Modus erscheint eine Rechteckwelle mit allen eingestellten Werten. Im Modus DISPLAY wird der zeitliche Verlauf eines ausgewählten Parameters grafisch dargestellt.

Einsatz

Nachdem ich eine Laborstromversorgung, ein Oszilloskop und ein genaues Multimeter angeschlossen hatte, konnte ich mit dem Test dieser elektronischen Last beginnen. Die Genauigkeit der angezeigten Werte war sehr gut und lag in der Regel innerhalb von ±1 mA/mV. Die großen Anschlussklemmen bieten genügend Platz, um Kabel direkt oder per Gabelschuh fest anzuschließen. Ich vermisste allerdings die Möglichkeit, Bananenstecker zu verwenden, was bei Strömen bis zu einigen Ampere einfacher ist.
Nachdem dem Ausprobieren einiger einfachen Dinge schmökerte ich zunächst doch im Handbuch, da es einfach sehr viele Einstellungen und Optionen gibt. Durch das Handbuch fand ich etwa heraus, dass die beiden Monitorausgänge auf der Rückseite erst in einem Untermenü aktiviert werden müssen, bevor sie Messsignale liefern. Leider wird diese Einstellung beim Ausschalten des Geräts vergessen und muss beim nächsten Mal wieder aktiviert werden. Ich entdeckte auch, dass die maximale Flankensteilheit im dynamischen Testmodus vom gewählten Strombereich abhängt: Im 30-A-Bereich sind 2,5 A/µs und im 5-A-Bereich nur 0,5 A/µs möglich. Abgesehen von diesen Kleinigkeiten (die wohl mit einem Firmware-Update angepasst werden könnten) ist das SDL1020X-E ein nettes und praktisches Gerät in der von Siglent mittlerweile gewohnten Qualität. Es ist auch möglich, die elektronische Last über einen PC zu steuern, doch ist dies derzeit nur über SCPI-Befehle oder einen LabView-Treiber möglich. Siglent scheint wohl an einem Windows-Programm zu arbeiten, doch es ist unklar, wann es verfügbar sein wird.

Fazit

Wenn Sie öfter eine elektronische Last zum Testen von Netzteilen, Akkus und ähnlichen Geräten benötigen, ist das SDL1020X-E eine ideale Ergänzung zu Ihren Labor-Equipment. Es bietet wirklich sehr viele Testmöglichkeiten. Das übersichtliche Display zeigt alle relevanten Informationen an und macht in vielen Fällen den Anschluss eines separaten Oszilloskops oder Multimeters überflüssig. Alles in Allem ist es ein gut durchdachtes Gerät zu einem erschwinglichen Preis!

200323-02

SHOPPING LIST

› **Siglent SDL1020X-E Elektronische Last**
www.elektor.de/siglent-sdl1020x-e-programmable-dc-electronic-load-200-w

Hochspannungsnetzteil mit Kennlinienschreiber

Spannungen bis 400 V einstellen und Kennlinien für Röhren und Transistoren erstellen

Von **Rainer Schuster**

Dieses universelle Hochspannungsnetzteil liefert nicht nur die Versorgungsspannungen für Röhrenschaltungen, sondern kann auch als Kennlinienschreiber für Röhren und Halbleiter verwendet werden. Die Steuerung übernimmt ein Raspberry Pi zusammen mit einem 7"-Touchscreen.

Ein Labornetzteil hat eigentlich jeder Elektroniker zur Verfügung, denn es kommt auf der Rangliste unentbehrlicher Tools sicherlich gleich nach Lötkolben und Multimeter. Üblicherweise hat man es dabei aber mit einer „Kleinspannung" von ≤ 60 V und Strömen im einstelligen Amperebereich zu tun. Das reicht für den Löwenanteil der Fälle, bei denen man im Elektronik-Labor eine Schaltung versorgen muss. Aber da es nicht nur Löwenartiges in der Elektronikwelt gibt, benötigt man gelegentlich auch deutlich höhere Spannungen. Dies gilt vor allem dann, wenn es um Röhren geht.

Wenn man schon ein spezielles Netzteil für größere Spannungen konstruiert, kann man mit moderner Technik ein paar interessante

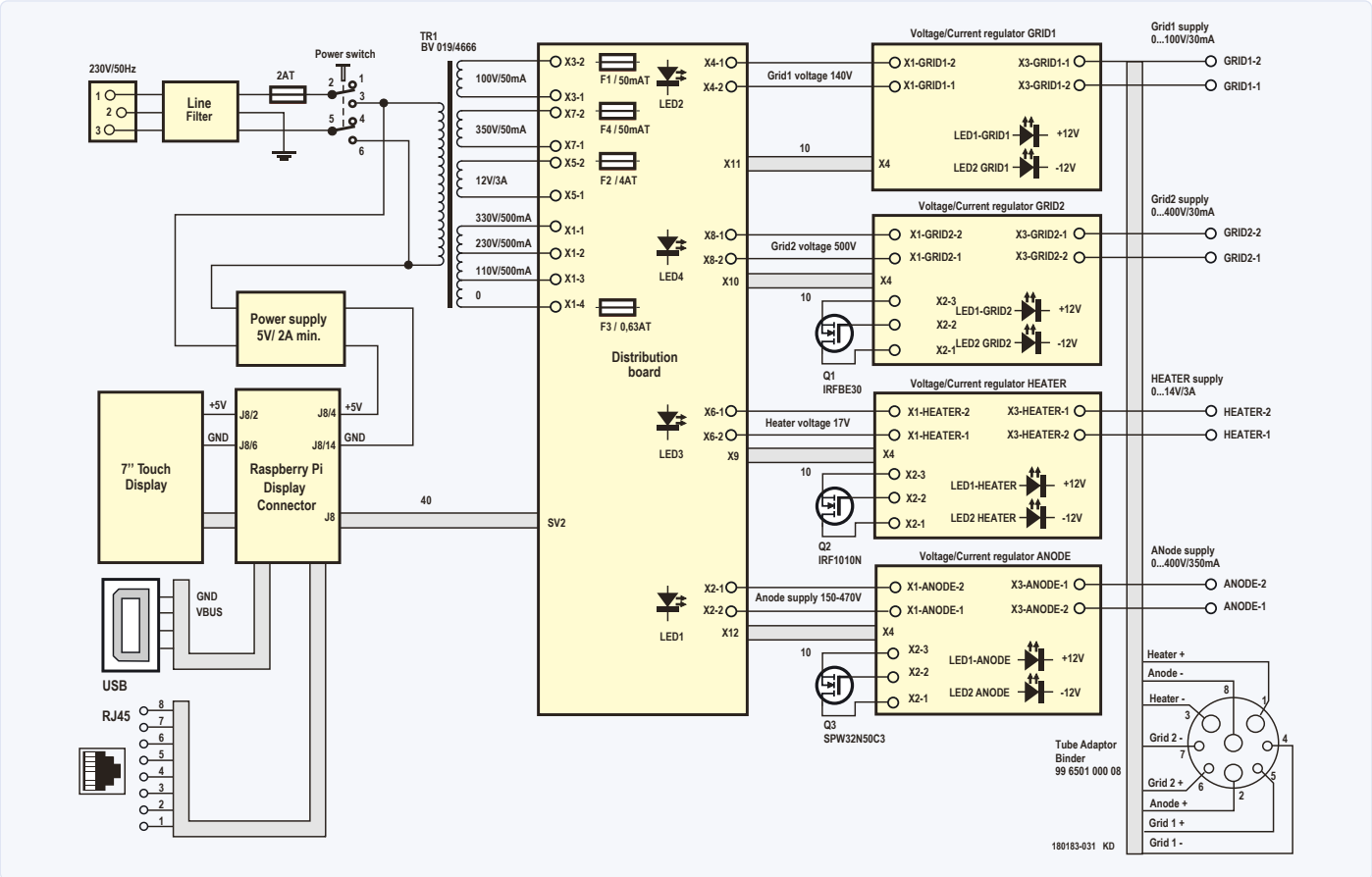


Bild 1. Die Gesamtschaltung des Hochspannungsnetzteils gibt einen Überblick über die Verbindung der einzelnen Bestandteile.

Zusatzfunktionen einbauen. Mit moderner Technik ist die Intelligenz eines RPis gemeint. Da die Ansprüche an die Rechenleistung nicht hoch sind, muss es nicht zwingend die aktuellste Version 4 sein – auch ein RPi 3 oder 2 reichen völlig aus. Dank eines 7"-Touchscreens ist auch die Bedienung und die Anzeige auf der Höhe der Zeit. Nützliche Zusatzfunktionen wie ein Kennlinienschreiber für Röhren und Halbleiter sind per Software leicht implementiert.

Netzteil-Hardware

Das Netzteil erzeugt natürlich mehr als nur eine „Hochspannung", die definitionsgemäß ja eigentlich immer noch Niederspannung ist, da sie deutlich kleiner als 1,5 kV Gleichspannung ist. Möchte man Röhren versorgen, braucht es neben der Anoden-Spannung schon mal eine weitere Spannung für die Heizung. Und wenn es um Kennlinien geht, so sind extra Spannungen für Steuer- und Schirmgitter unverzichtbar. All diese Spannungen und Ströme sind einstellbar.

Die Hardware zur Regelung von Spannung und Strom für die verschiedenen Subnetzteile besteht aus vier unabhängigen und galvanisch voneinander getrennten Einheiten, die folgende Spannungen bzw. Ströme liefern können:

- Anode: 0...400 V; 0...300 mA
- Steuergitter: 0...100 V; 0...30 mA
- Schirmgitter: 0...400 V; 0...30 mA
- Heizung: 0...14 V; 0...3 A

Bild 1 zeigt die Gesamtschaltung des Netzteils bzw. die Verdrahtung seiner einzelnen Teile. Die diversen Funktionen des Netzteils sind auf dem Distribution-Board und den unabhängigen Subnetzteilen untergebracht.

Distribution-Board

Das Distribution-Board ist mit den Gleichrichtern, Siebelkos und Sicherungen für die verschiedenen Regler-Boards bestückt. Der 40-polige Steckverbinder SV2 stellt die Verbindung zu den IO-Ports des RPi her und verteilt die SPI-Signale via X9...X12 auf die Regler-Boards.

Bei der Anoden-Spannung wird je nach Ausgangsspannung zur Verringerung der Verlustleistung des Längstransistors mit den Relais K1 und K2 auf die Transformatorwicklungen 110, 220 oder 330 V umgeschaltet. Die Umschaltung erfolgt per Software mittels der GPIO-Pins 20 und 21 des RPi. **Bild 2** zeigt die Schaltung des Distribution-Boards.

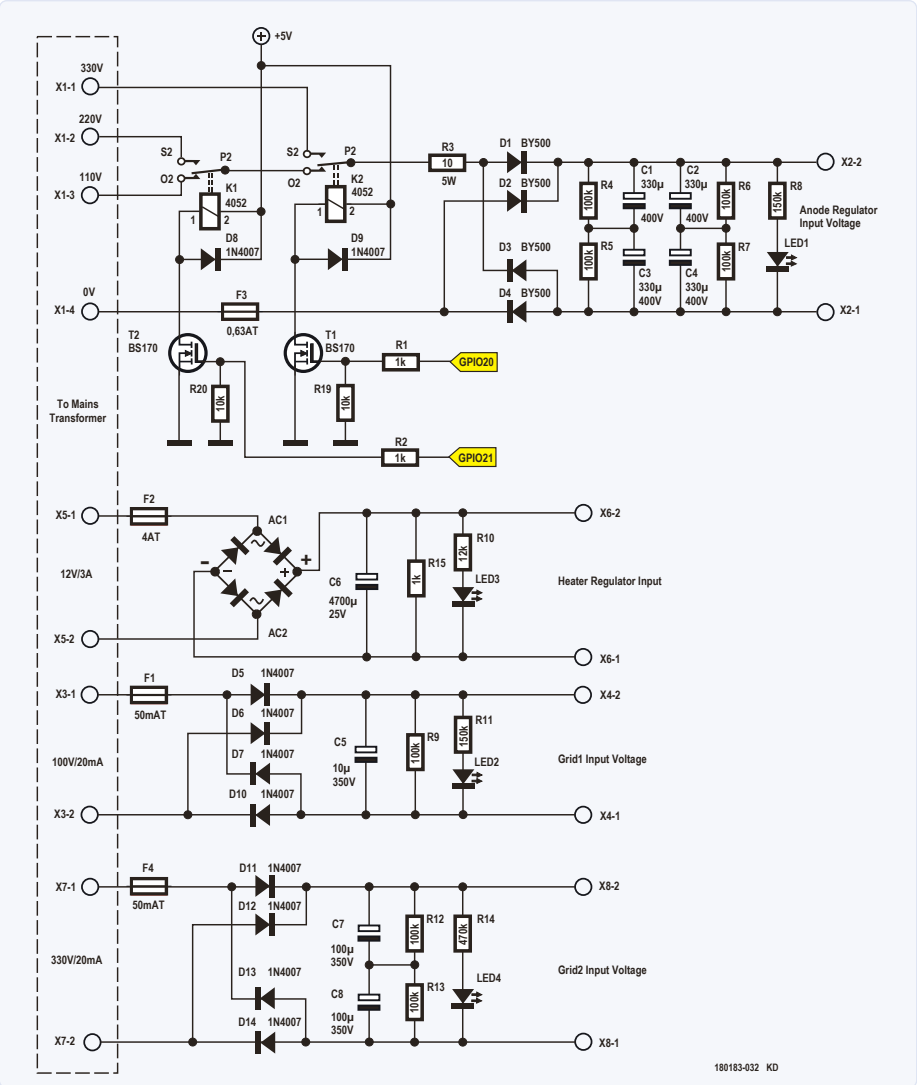


Bild 2. Die Schaltung des Distribution-Boards.

Regler-Boards

Jedes Regler-Board enthält A/D- und D/A-Konverter zur Einstellung und Messung von Spannung und Strom, welche per SPI-Schnittstelle mit dem RPi kommunizieren. Auch die SPI-Signale sind galvanisch vom RPi getrennt.

Schaltungsprinzip

Bild 3 zeigt die verwendete Prinzipschaltung. Zur Spannungsregelung dient der als Längsregler geschaltete MOSFET T2. Damit T2 bei den hier vorkommenden hohen Spannungen von normalen Opamps angesteuert werden kann, ist der positive Pol der Ausgangsspannung mit der Masse (GND) der Opamp-Versorgung verbunden, die hierfür symmetrische ±12 V liefert.

Zur Einstellung des Sollwertes für Strom und Spannung wird mit dem Typ MCP4812 ein 2-Kanal-DAC mit einer Auflösung von 10 bit verwendet. Dieser DAC besitzt eine interne Referenzspannung von 2,048 V, die per SPI-Befehl auf 4,096 V verdoppelt werden kann. Die Ausgangsspannung wird über R17/R18 geteilt und die Spannung über R18 als Ist-Wert digitalisiert durch den 12-bit-A/D-Konverter vom Typ LTC1298 an den RPi gesendet.

Zur Stromregelung wird der Spannungsabfall über R1 von einem Opamp fünffach verstärkt. Dessen Spannung gelangt dann als Ist-Wert des Stroms über den zweiten Kanal des A/D-Konverters an den RPi.

Zur galvanischen Trennung der SPI-Signale MOSI, MISO, SCLK und LDAC sowie der Chip-Select-Leitungen für den A/D und D/A-Wandler werden Highspeed–Optokoppler vom Typ 6N137 verwendet, die von einem 74HC04 getrieben werden.

Alle Regler-Boards benötigen natürlich ebenfalls galvanisch getrennte Stromversorgungen. Hierzu versorgt der 5-V-Ausgang des RPi pro Board einen DC/DC-Wandler des Typs TMA0512D, der die erforderlichen ±12V für die Opamps generiert.

Die Ist-Werte von Spannung und Strom liegen nicht nur an den Eingängen des A/D-Konver-

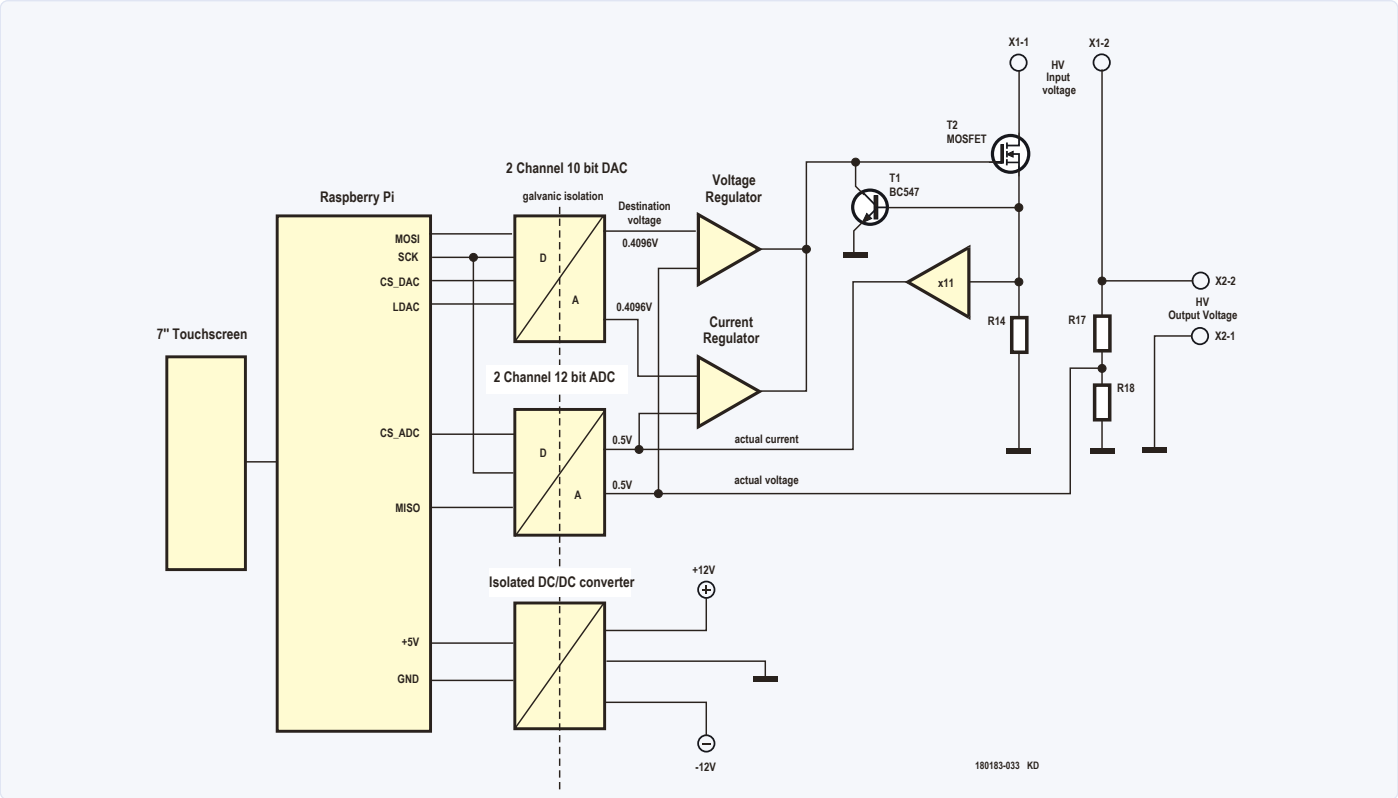


Bild 3. Die Prinzipschaltung des Hochspannungsnetzteils.

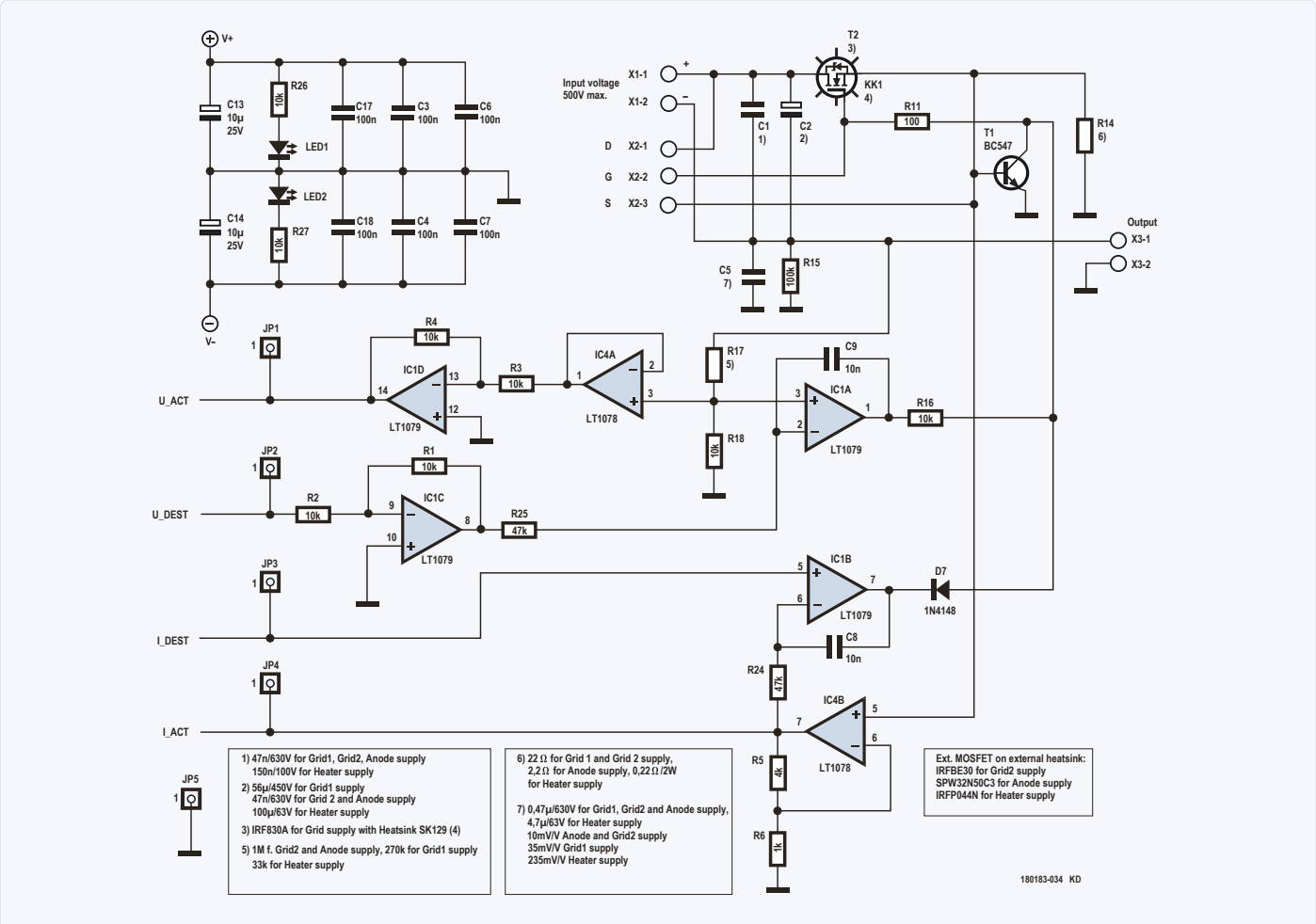


Bild 4a. Schaltung der eigentlichen Regelung des Regler-Boards.

ters, sondern auch als Feedback an den beiden Opamps für die Spannungs- und Stromregelung. Diese Opamps vergleichen die Istwerte mit den jeweiligen Sollwerten der D/A-Konverter. Das sich so ergebende Korrektursignal des Spannungsregler-Opamps steuert dann das Gate von T2. Die Stromregelung greift ein, wenn der gemessene Strom höher ist als der eingestellte. Es handelt sich also um eine Konstantspannungsregelung mit einstellbarer Strombegrenzung.

Durch das I-Verhalten der Regler wird der Anfangsstrom bei einem „harten" Kurzschluss am Ausgang kurzfristig den Sollwert überschreiten. Dabei wird beim Board für die Anoden-Spannung eventuell die so genannte SOA (Safe Operating Area) von T2 verlassen. Um dies zu verhindern, begrenzt T1 hier den Ausgangsstrom schnell und radikal auf maximal 0,7 V / R14 = 0,31 A. Die Verlustleistung beträgt dann kurzzeitig bis zu 465 V * 0,31 A = 144 W, bis die Eingangsspannung von der Software auf 110 V umgeschaltet wird. Laut Datenblatt verträgt T2 fast die doppelte Verlustleistung.

Bild 4 zeigt die konkrete Schaltung der Regler-Boards. Außerdem: Das Regelverhalten der Schaltung wurde mit LTSpice simuliert. Für Interessierte: Die Datei *HVRegulator.asc* befindet sich im kostenlosen Download-Archiv der Elektor-Webseite zu diesem Artikel [1]. Alle Regler-Boards haben zwar das gleiche Layout, doch sind sie für die unterschiedlichen Spannungen und Ströme auch unterschiedlich bestückt (siehe **Tabelle 1**).

Raspberry Pi

Wie schon erwähnt, wird zur Steuerung ein RPi mit 7"-Touchscreen eingesetzt. Zur Stromversorgung des Raspberry eignet sich jedes kleine Open-Frame-Schaltnetzteil mit 5 V

Tabelle 1.

Board	C1	C2	C5	T2	R14	R17
Anode	47 nF 630 V	47 nF 630 V	0,47 uF 630 V	SPW32N50C3 (KK ≤0,7 k/W)	2,2 Ω 1 W	1 MΩ
Steuergitter	47 nF 630 V	56 uF 450 V	47 nF 630 V	IRF830A (KK SK129)	22 Ω	270 k
Schirmgitter	47 nF 630 V	47 nF 630 V	47 nF 630 V	IRFBE30 (KK ≤0,7 k/W)	22 Ω	1 M
Heizung	150 nF 100 V	100 nF 63 V	4,7 uF 63 V	IRFP044 (KK ≤0,7 k/W)	0,22 Ω 2 W	33 k

* R14 = 4,096 V / (5 * I_{max})
** R17 = U_{max} * 2,441 kΩ / V – 10 kΩ

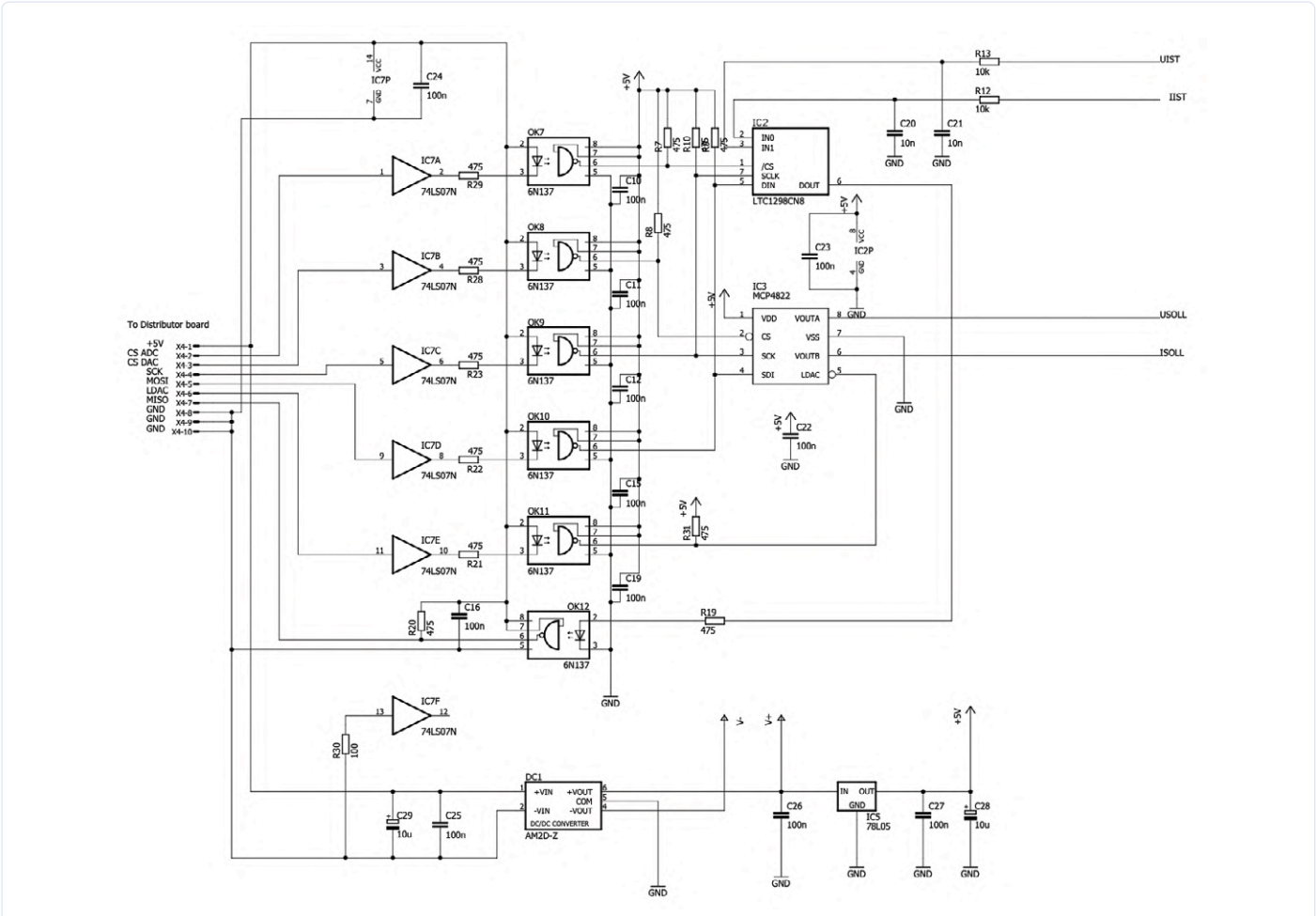


Bild 4b. Anbindung des Regler-Boards ans Distribution Bord.

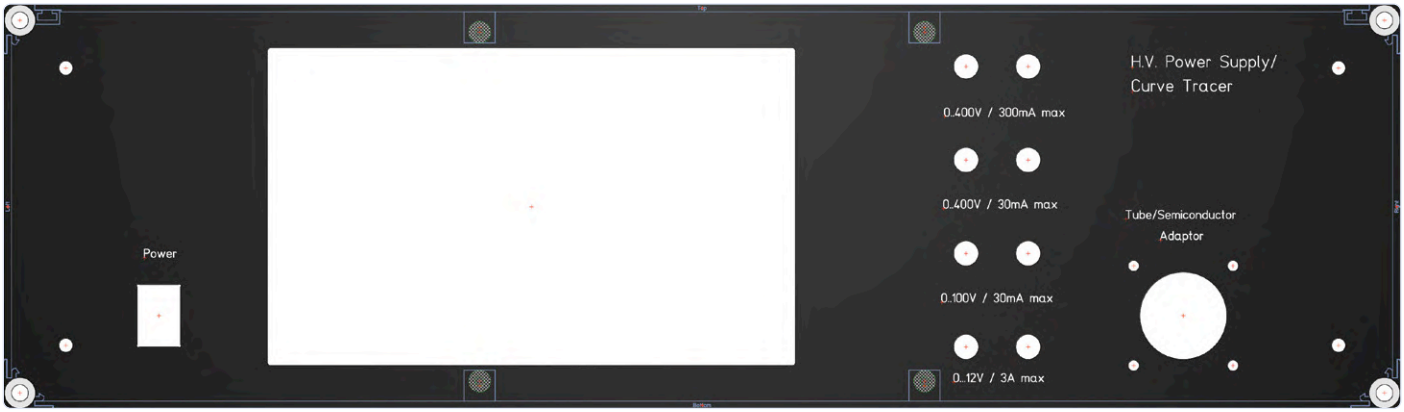


Bild 5. Die Frontplatte gezeichnet mit dem Programm FrontDesign.

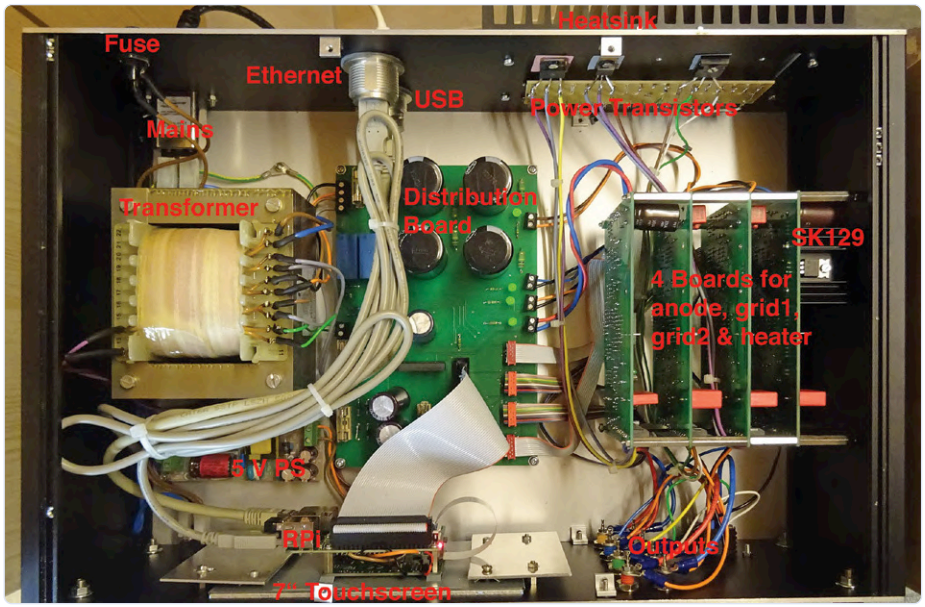


Bild 6. Innerer Aufbau des Prototypen.

bei 2 A (3 A bei RPi 4). Die Ausgangsspannung dieses Netzteils sollte aber direkt am Verbinder J8 angeschlossen werden, da der Spannungsabfall am Micro-USB-Connector zu hoch ist.

Aufbau

Das Gerät wurde in ein Profilgehäuse eingebaut, das mit dem Programm FrontDesign gezeichnet wurde, das kostenlos von der Firma Schäffer [2] für Windows, MacOS und Linux erhältlich ist. **Bild 5** zeigt den Entwurf der Frontplatte. Die zugehörigen Dateien befinden sich ebenfalls im Elektor-Download [1]. **Bild 6** zeigt den Aufbau des Prototyps von oben. Man sieht deutlich die vier gleichartigen Regler-Boards auf der rechten Seite.

Software

Die RPi-Software wurde in C++ für Visual Studio von Microsoft [3] erstellt. Damit der Code auch auf dem RPi läuft, ist der Cross-Compiler VisualGDB [4] erforderlich, für den es eine 30 Tage lauffähige Testversion gibt. Zur Erstellung der grafischen Oberfläche wurde Qt-Designer [5] genutzt.

Die beigefügte Projekt-Datei *HV_PowerSupply.sln* im Elektor-Download [1] enthält bereits die erforderlichen Einstellungen für VisualGDB (**Bild 7**). Hier muss man einstellen, dass es sich um ein Linux-Projekt handelt. Außerdem müssen der Projektname und das Projektverzeichnis angegeben werden.

Im nächsten Schritt wird eingestellt, dass es sich um ein Qt5-basierendes System handelt (**Bild 8**). Nun folgt die Einstellung, ob die Software auf dem lokalen PC, oder auf dem Zielsystem (RPi) erstellt wird (**Bild 9**). Ich habe den lokalen PC gewählt, da der RPi sonst während des Compiler/Linker-Laufs eingeschaltet und mit dem PC verbunden sein

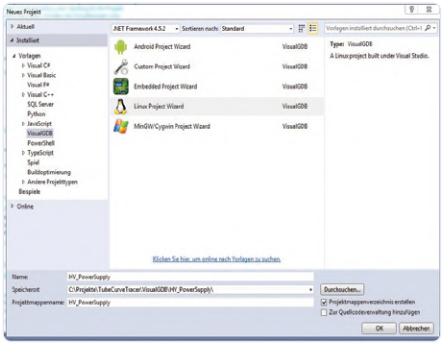


Bild 7. Einstellung für VisualGDB als Linux-Projekt.

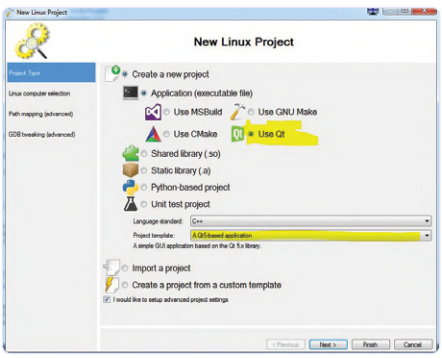


Bild 8. Es handelt sich um ein Qt5-basierendes System.

muss. Da die Library WiringPi [6] verwendet wird, muss diese ebenfalls angegeben werden (**Bild 10**).

Nach der erfolgreichen Erstellung der Software befindet *HV_PowerSupply* als auf dem RPi lauffähige Datei im Debug-Ordner des Projekts. Die Datei muss nun z.B. mit WinSCP [7] in das Verzeichnis */home/pi/HV_PowerSupply* des RPi kopiert werden. Auch das HV-Powersupply-Icon *Tube.png* muss noch in dieses Verzeichnis kopiert werden.

Um die Applikation vom Desktop zu starten, muss die Datei *HV_PowerSupply.desktop* nach */home/pi/desktop* kopiert sein. Soll das Programm nach dem Einschalten automatisch starten, muss sich diese Datei auch im Verzeichnis */home/pi/.config/autostart* befinden.

Zur korrekten Anzeige der Up/Down-Symbole auf den verwendeten Tasten ist es erforderlich, die Dateien *collapse-arrow.png* und *expand-arrow.png* in das Verzeichnis */home/pi* zu kopieren. Möchte man das Röhren-Icon in der Taskleiste anzeigen, so muss sich auch die Datei *Tube.png* in */home/pi* befinden.

Wer die Software unverändert verwenden möchte, braucht natürlich nur die fertigen Dateien des Elektor-Downloads wie beschrieben auf den RPi kopieren.

Source-Dateien

Wer die Software modifizieren möchte, muss die zuvor beschriebenen Software-Pakete installieren und die Source-Dateien dieses Projekts in das Verzeichnis */VisualGDB/HV_PowerSupply/HV_PowerSupply* kopieren. Das Projekt besteht aus folgenden Source-Dateien:

- *HV_PowerSupply.cpp*: Hauptprogramm (*MainWindow.cpp* wird gestartet).
- *MainWindow.cpp*, *MainWindow.h*, *ui_MainWindow.h*: Source-Dateien für die Stromversorgung
- *CurveTracerWindow.cpp*, *CurveTracerWindow.h*, *ui_CurveTracerWindow.h*: Source-Dateien für den Röhren-Kennlinienschreiber
- *TransistorCurveTracerWindow.cpp*, *TransistorCurveTracerWindow.h*, *ui_TransistorCurveTracerWindow.h*: Source-Dateien für den Halbleiter-Kennlinienschreiber
- *Hardware.cpp*, *hardware.h*: Gemeinsam genutzte Funktionen zur Steuerung der D/A- und A/D-Konverter, verwendet die Bibliothek *wiringPi*
- *Qcustomplot.cpp*, *qcustomplot.h*: Bibliothek zur grafischen Darstellung der Kennlinien. Damit diese Bibliothek fehlerfrei übersetzt wird, muss die Zeile

```
QT += core gui xml printsupport
```

in der Datei *debug.pro* ergänzt werden.

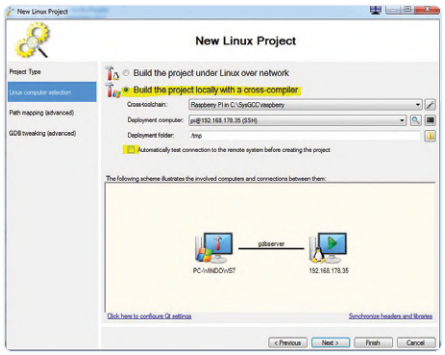


Bild 9. Software für PC oder das Zielsystem (RPi)?

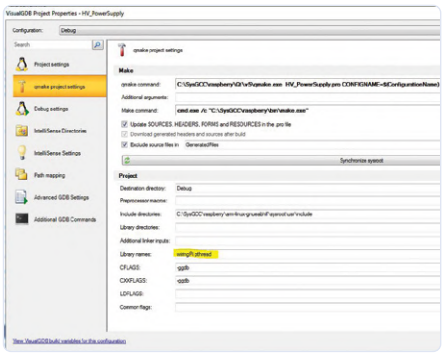


Bild 10. Die Library WiringPi wird verwendet.

Die Dateien *ui_XXXX.h* werden von Qt-Designer generiert, nachdem die grafischen Oberflächen für die einzelnen Screens gezeichnet wurden. **Bild 11** zeigt den Bildschirm für die Stromversorgung. Durch Auswahl von *Formular-> Code erzeugen* im Menü wird der Sourcecode generiert. Er muss dann nur noch im Source-Verzeichnis abgelegt werden.

Programmablauf

Beim Programmstart wird der Stromversorgungs-Bildschirm angezeigt und die Hardware initialisiert. Alle Spannungen und Ströme werden zunächst auf 0 gesetzt. Jetzt können die Spannungen und Ströme für die einzelnen Ausgänge eingestellt werden. Dies kann durch Berühren der Auf/Ab-Tasten auf dem Touchscreen, durch eine externe Tastatur/Maus-Kombination via USB oder durch ein virtuelles Keyboard wie z.B. dem Matchbox-Keyboard [9] geschehen (**Bild 12**).

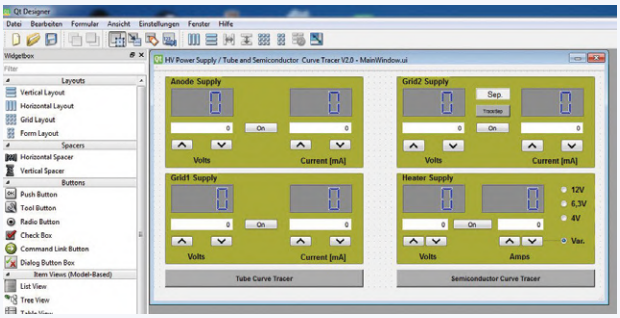


Bild 11. Bildschirm des Hochspannungsnetzteils erstellt in QtDesigner.

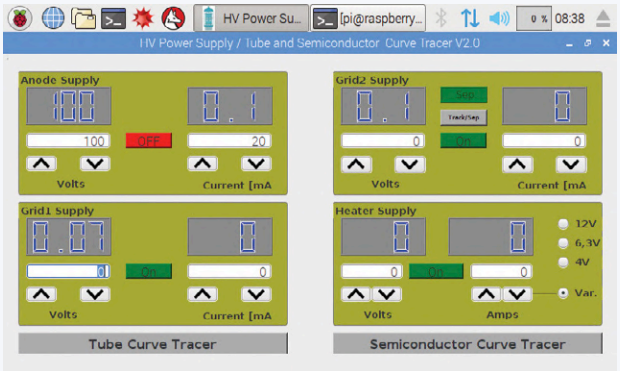


Bild 12. Aktive Oberfläche des Hochspannungsnetzteils.

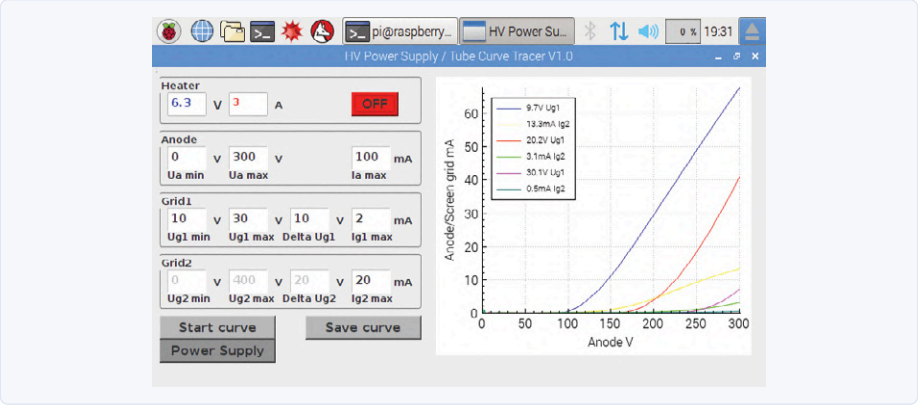


Bild 13. Oberfläche des Kennlinienschreibers für Röhren.

Wenn die Schirmgitter-Spannung der Anoden-Spannung folgen soll, wird mit der Taste *Track/Sep* der Tracking-Mode aktiviert. Die Spannung für die Heizung kann entweder von 0...14 V via *Var* eingestellt oder auf die vordefinierten Werte 4/6,3/12 V gesetzt werden. Mit den Tasten *On/Off* können die einzelnen Ausgänge individuell ein- bzw. ausgeschaltet werden.

Röhren-Kennlinienschreiber

Mit der Taste *Tube Curve Tracer* wird der Bildschirm des Röhren-Kennlinienschreibers aktiviert. Dabei werden zunächst werden alle Spannungen und Ströme auf 0 gesetzt.

In diesem Modus werden Kennlinien für Röhren aufgenommen und ggf. als Tabellen mit der Extension *.csv* abgespeichert, damit sie direkt z.B. von Excel gelesen werden können. Für die Kennlinie können Minimal- und Maximalwerte für Anoden- und Steuerspannung angegeben werden. Für Anoden-, Steuergitter- und Schirmgitter-Strom lassen sich Maximalwerte einstellen. Die Schirmgitter-Spannung folgt der Anoden-Spannung. Es können maximal fünf Kennlinien angezeigt werden.

Bild 13 zeigt die aufgenommenen Kennlinien einer Röhre vom Typ EL34. Die Kennlinien enthalten den Anoden-Strom als Funktion der Anoden-Spannung mit der Steuergitter-Spannung als Parameter. Ebenfalls aufgezeichnet werden die dazugehörigen Schirmgitterströme.

Die Kennliniendaten kann man sichern, indem man die Taste *Save Curve* betätigt. Dann wird ein Dialogfenster zum Sichern der Kennlinien-Datei geöffnet und man kann sie als Datei mit der Endung *.csv* sichern. Die Dateien können mit z.B. WinSCP zum PC kopiert und mit Excel weiterverarbeitet werden. Im Elektor-Download [1] ist die Beispieldatei *EL34.csv* enthalten.

Zum Testen von EL34-Endstufenröhren habe ich einen Adapter gebaut, der sich einfach über ein Kabel mit dem Kennlinienschreiber verbinden lässt (**Bild 14**). Die simple Schaltung dieses Adapters ist in **Bild 15** zu sehen. Durch Verwendung anderer Röhrensockel können nach dem gleichen Prinzip auch Adapter für andere Röhrentypen gebaut werden. Man kann die zu testende Röhre aber auch einfach direkt an die Klemmen des Netzteils anschliessen.

Halbleiter-Kennlinienschreiber

Mit der Taste *Semiconductor Curve Tracer* wird der Bildschirm für den Halbleiter-Kennlinienschreiber angezeigt. **Bild 16** zeigt exemplarisch das Verhalten eines gesperrten, bipolaren Transistors bei offener Basis.

Auch für Halbleiter sind entsprechende Adapter sehr hilfreich. **Bild 17** zeigt die Schaltung des Halbleiter-Adapters und **Bild 18** das fertige Gerät. **Bild 19** gibt einen Einblick ins Innenleben.

Steckverbinder SV1 stellt die Verbindung zum Kennlinienschreiber dar. Mit der Heizungsspannung werden die Relais K1 und K2 versorgt, welche die PNP/NPN- bzw. NMOS/PMOS-Umschaltung vornehmen. Die Steuergitter-Spannung wird für K3 verwendet. Dieses Relais schaltet den Basis-Strom bzw. die Gate-Spannung. Die Anoden-Spannung wird zur Versorgung von Kollektor/Emitter bzw. Drain/Source verwendet. Zum Testen von Z-Dioden muss die Anode mit dem Minuspol und die Kathode mit dem Pluspol der Anoden-Spannung verbunden werden.

Zum Test von bipolaren Transistoren wird die Schirmgitter-Spannung zur Erzeugung des Basis-Stroms herangezogen. Dieser wird durch R2 und R3 auf 4 mA bei 400 V begrenzt. Zum Test von MOSFETs wird R1 als Spannungsteiler zur Erzeugung der Gate-Spannung zugeschaltet. D3 und D4 begrenzen U_{GS} auf maximal 19 V.

Z-Dioden und $U_{ce\ max}$ von Bipolar-Transistoren

Nach der Auswahl *Zenerdiode* ändert sich der Text der Felder *Ucemax* und *Ic max* in *Uz* und *Pmax* (Bild 16), da bei Z-Dioden üblicherweise die maximale Leistung angegeben wird. Nach Betätigung der Taste *Start curve* wird die Anoden-Spannung von 0...Uz und maximal bis P_{max} hochgefahren (je nachdem, was zuerst erreicht wird). Beträgt der Strom durch die Z-Diode bei *Uz* weniger als 1% des maximalen Z-Stroms, wird die Fehlermeldung von **Bild 20** ausgegeben. Dann ist entweder die Z-Diode defekt, oder *Uz* muss erhöht werden. Wegen der hohen möglichen Anoden-Span-

Bild 14. Röhrenadapter für EL34.

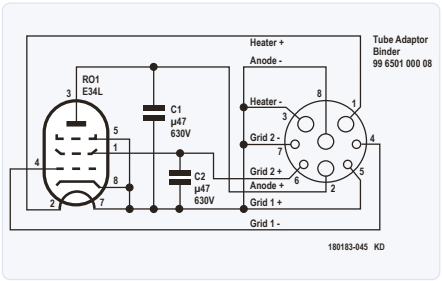


Bild 15. Schaltung des Röhrenadapters.

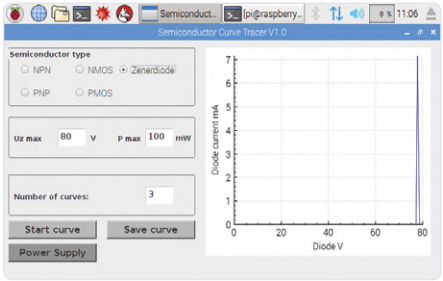


Bild 16. Maximale Kollektor-Emitter-Spannung eines getesteten BC547C.

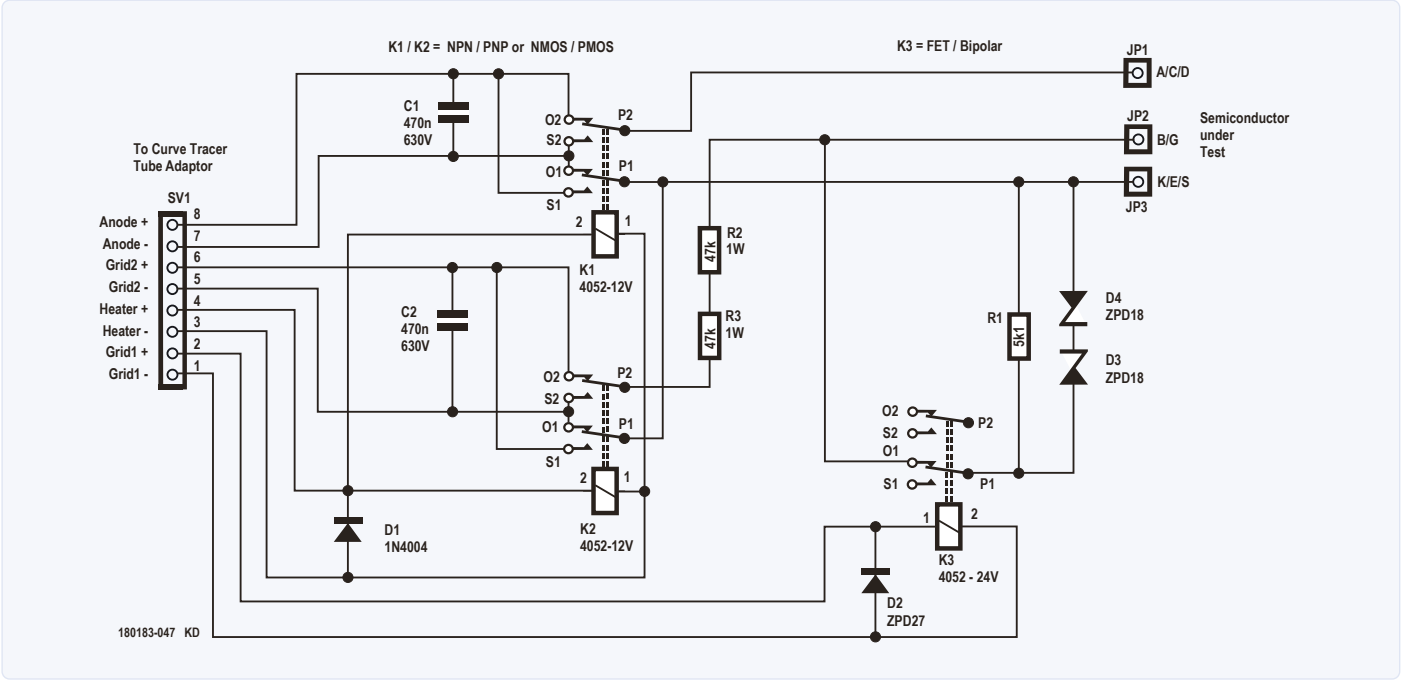


Bild 17. Schaltung des Halbleiteradapters.

nung kann auch die maximale Kollektor-Emitter-Spannung vieler Bipolar-Transistoren bei offener Basis (= $U_{ceo\ max}$) getestet werden. Die Kurve von Bild 16 ist beim Testen eines BC547C entstanden. Obwohl das Datenblatt lediglich 45 V erwarten lässt, erreichte dieser Transistor eine maximale U_{ceo} von >75 V.

Test bipolarer Transistoren

Bei bipolaren Transistoren muss entweder *NPN* oder *PNP* ausgewählt werden. $U_{ce\ max}$ und $I_{c\ max}$ sowie die Anzahl der Kennlinien können eingestellt werden. Nach dem Start prüft die Software, ob auch ohne Basis-Strom bereits ein nennenswerter Kollektor-Strom fließt. In diesem Fall wird der Test mit einer Fehlermeldung beendet. Andernfalls wird $U_{ce\ max}$ auf den gewählten Wert eingestellt und der Basisstrom solange erhöht, bis $I_{c\ max}$ erreicht ist. Kann der eingestellte Kollektorstrom beim maximalen Basis-Strom von 4 mA nicht erreicht werden, wird ebenfalls eine Fehlermeldung ausgegeben.
Angezeigt werden dann die Kennlinien $I_c = f(U_{ce})$ mit den zugehörigen Basisströmen, die in der Legende angezeigt werden. Mit der Taste *Save Curve* lassen sich die Kennliniendaten von I_c, I_b und U_{ce} als Datei mit der Endung *.csv* sichern.

Test von MOSFETs

Bei MOSFETs muss man sich zwischen *NMOS* oder *PMOS* entscheiden. Es können

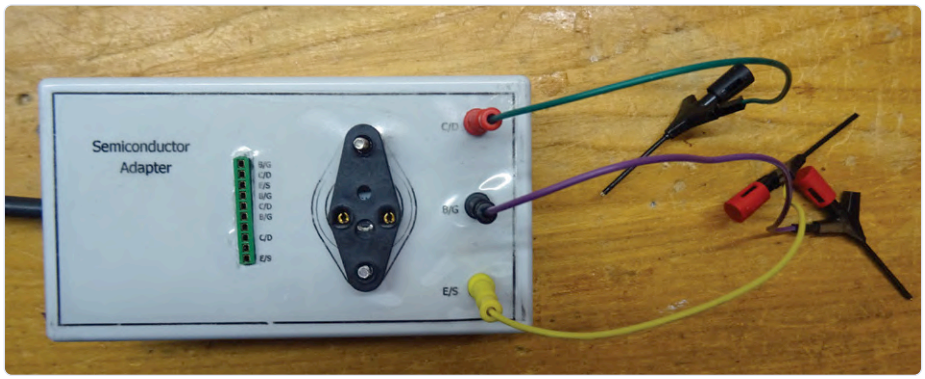


Bild 18. Halbleiteradapter von außen.

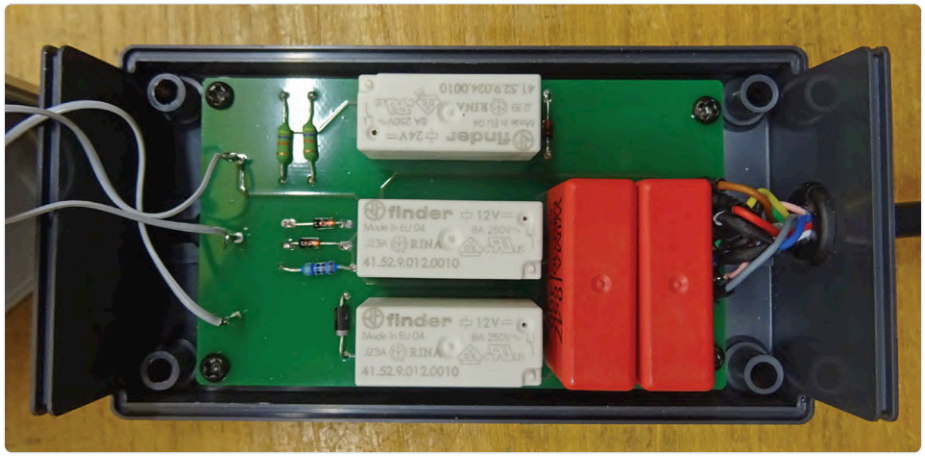


Bild 19. Halbleiteradapter von innen.

Bild 21. So sieht der Prototyp von vorne aus.

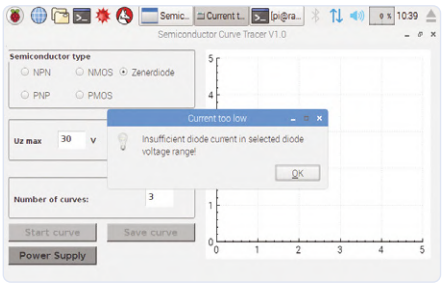


Bild 20. Fehlermeldung bei zu geringem Strom durch die Z-Diode.

die Maximalwerte für *Uds max* und *Id max* sowie die Anzahl der Kennlinien eingestellt werden. Nach dem Start prüft die Software, ob bei U_{gs} = 0 bereits ein Drain-Strom fließt. In diesem Fall wird der Test mit einer Fehlermeldung beendet. Andernfalls wird der Wert für $U_{ds\,max}$ eingestellt und U_{gs} solange erhöht, bis $I_{d\,max}$ erreicht ist. Kann der maximale Drain-Strom bei $U_{gs\,max}$ = 18 V nicht erreicht werden, zeigt sich ebenfalls eine Fehlermeldung.

Angezeigt werden die Kennlinien $I_d = f(U_{ds})$ mit den zugehörigen Gate-Spannungen, die in der Legende angezeigt werden. Mit der der Taste *Save Curve* lassen sich die Kennliniendaten von I_d, U_{gs} und U_{ds} als Datei mit der Endung *.csv* sichern.

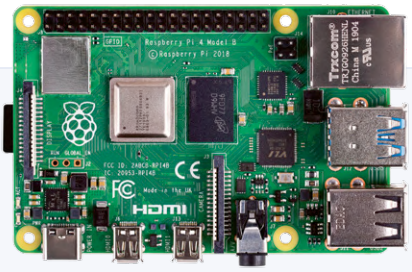
Warnung!

Vorsicht bei der Einstellung des maximalen Drain/Kollektor-Stroms und der maximalen Drain-Source- bzw. Kollektor-Emitter-Spannung! Eine Einstellung von 400 V bei einem Strom von 300 mA ergibt eine Verlustleistung des zu testenden Transistors von 120 W! Es sollten unbedingt die SOA-Angaben des Datenblatts eingehalten werden, und der „Transistor under Test" braucht eventuell eine Kühlung.

Außerdem

Im Archiv des kostenlosen Downloads [1] sind die Platinen-Dateien im Eagle-Format samt Stücklisten enthalten. Auch die Gehäusezeichnungen und LTSpice-Dateien und der komplette Sourcecode finden sich dort und sind für die private Nutzung freigegeben. **Bild 21** gibt zum Schluss einen Eindruck vom Aussehen des Prototypen.

180183-01



SHOPPING LIST

- **Raspberry Pi 4 B (2 GB RAM)**
  www.elektor.de/raspberry-pi-4-b-2-gb-ram
- **7"-Touchscreen for Raspberry Pi**
  www.elektor.de/joy-it-7-touchscreen-for-raspberry-pi
- **JOY-iT JT-RD6006 DC Power Supply Bundle**
  www.elektor.de/joy-it-jt-rd6006-dc-power-supply-bundle

WEBLINKS

[1] **Download zu diesem Artikel**: www.elektormagazine.de/180183-01
[2] **FrontDesign**: www.schaeffer-ag.de/frontplatten-designer
[3] **Visual Studio**: https://visualstudio.microsoft.com/de/downloads/
[4] **Cross-Compiler für RPi**: https://visualgdb.com/download/
[5] **Qt Designer**: https://build-system.fman.io/qt-designer-download
[6] **WiringPi**: http://wiringpi.com/
[7] **WinSCP**: https://winscp.net/eng/download.php
[8] **QCustomPlot**: https://www.qcustomplot.com/index.php/download
[9] **Matchbox-Keyboard**: https://github.com/xlab/matchbox-keyboard

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 5. Oktober 2020.

Die Gewinner des Hexadokus aus der Ausgabe Juli/August 2020 stehen fest!

Die richtige Lösung ist: **8C5B2**

Einen Elektor-Wertgutschein über je 50 € haben gewonnen:

Heinz Seitz, Kathryn Rangeley, Mihails Sehurins, Martin Poelstra und Eric Poole.

Herzlichen Glückwunsch!

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

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

Design analoger Filter (Teil 1)

Von **Alfred Rosenkränzer**

Diese Artikelserie behandelt die Entwicklung analoger Filter. Im vorliegenden ersten Teil geht es um grundlegende Theorie, allerdings mit besonderem Augenmerk auf der praktischen Anwendung. Der zweite Teil wird sich mit dem Entwurf aktiver Filter beschäftigen. Der abschließende dritte Teil wird sich dann passiven Filtern widmen.

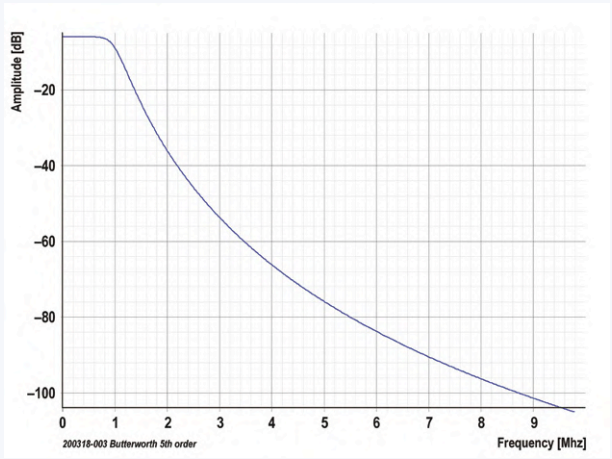


Bild 1. Frequenzgang eines 1-MHz-Tiefpassfilters mit linearer Frequenz und der Amplitude in dB.

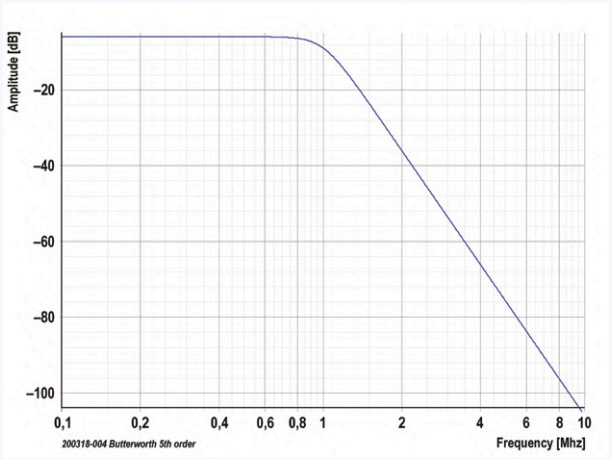


Bild 2. Frequenzgang im Audiobereich mit logarithmischer Frequenzskala und der Amplitude in dB.

Analoge Filter werden mit Hilfe von Widerständen, Spulen, Kondensatoren und Operationsverstärkern realisiert. Sie wirken auf analoge Spannungs- und Stromverläufe. Digitale Filter hingegen basieren auf digitalen Schaltungen wie Flipflops, Addierern, Multiplizierern (meist in Form eines DSP oder FPGAs) oder Software in Mikrocontrollern und SoCs und wirken auf Datenströme.

Filtereigenschaften

Die relevanten Eigenschaften eines Filters werden typischerweise in XY-Diagrammen dargestellt. Das wichtigste Diagramm zeigt die Amplitude über der Frequenz, kurz: den Frequenzgang. Das Verhältnis von Ein- und Ausgangsspannung wird auf der Y-Achse seltener linear in Prozentwerten, sondern meistens logarithmisch in dB angezeigt. Auf der X-Achse ist die Frequenz - häufig linear bei kleineren Bandbreiten (**Bild 1**) oder gerade für den Audiobereich auch oft logarithmisch (**Bild 2**) - aufgetragen. Alle folgenden Diagramme wurden mit der Simulationssoftware Simetrix [1] erstellt. Ein Filter führt aber nicht nur zu einer veränderten Amplitude sondern auch zu einer veränderten Phase des Ausgangssignals. Beim Phasengang ist die Y-Achse mit der Phasenverschiebung in Grad

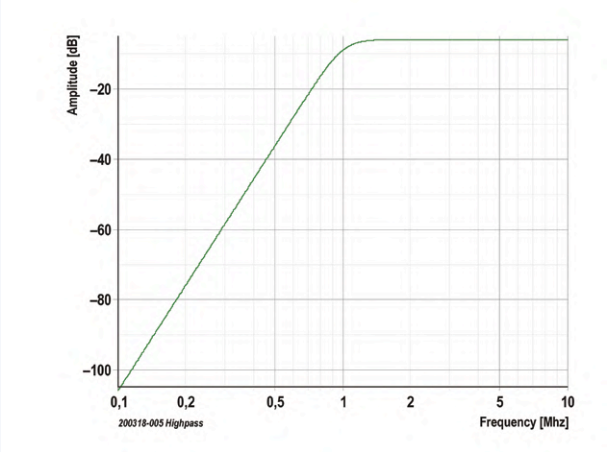


Bild 3. Frequenzgang eines 1-MHz-Hochpasses mit doppelt logarithmischer Skalierung.

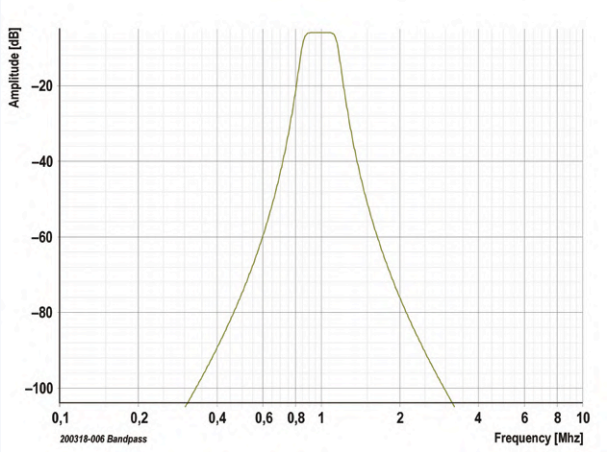


Bild 4. Frequenzgang eines 1-MHz-Bandpasses mit doppelt logarithmischer Skalierung.

bezeichnet. Auf der X-Achse ist wie zuvor die Frequenz aufgetragen. Für mich ist der Phasengang gerade im HF-Bereich nicht sehr aussagekräftig. Interessanter finde ich die Gruppenlaufzeit über der Frequenz. Dieses Diagramm entspricht der Ableitung des Phasengangs nach der Frequenz. Daraus kann man sehr einfach entnehmen, wie lange ein Signal einer bestimmten Frequenz braucht, um das Filter zu passieren. Ist der Verlauf der Gruppenlaufzeit flach, brauchen alle Frequenzen die gleiche Zeit. Warum das wichtig ist wird weiter unten mit Diagrammen genauer erläutert. Im Zeitbereich ist die Sprungantwort des Filters wichtig. Sie zeigt den linearen Amplitudenverlauf des Ausgangssignal über der Zeit bei Anregung durch eine Signalflanke – ganz ähnlich wie die Darstellung eines Oszilloskops. Auch diese Eigenschaften werden nachfolgend noch genauer beleuchtet.

Filterarten

Ein **Tiefpass** lässt tiefe Frequenzen möglichst ungehindert durch und dämpft hohe Frequenzen. Dazwischen liegt der Übergangsbereich. Der Punkt an dem das Signal eine Dämpfung um 3 dB erfährt, wird als Grenze zwischen Durchlassbereich und Übergangsbereich betrachtet. Seine Frequenz ist daher eine wichtige Filtereigenschaft. Der Sperrbereich beginnt bei einer bestimmten Dämpfung. Für diese Dämpfung gibt es keine Konventionen, da sie von der jeweiligen Anwendung abhängt. Die Bilder 1 und 2 enthalten Beispiele für den Frequenzgang von Tiefpässen. Ein Tiefpass lässt Gleichspannung passieren.

Ein **Hochpass** lässt hohe Frequenzen möglichst durch und dämpft tiefe Frequenzen. Auch hier indiziert der -3-dB-Punkt den Übergang vom Durchlass- zum Übergangsbereich (**Bild 3**). Das beim Tiefpass Gesagte gilt sinngemäß auch für den Hochpass. Ein Hochpass trennt allerdings die Gleichspannungskomponente des Eingangssignals ab.

Ein **Bandpass** lässt einen bestimmten Bereich an Frequenzen durch und dämpft sowohl tiefere als auch höhere Frequenzen. Hier gibt es zwei -3-dB-Punkte, die den Durchlassbereich definieren (**Bild 4**). Aufgrund der Hochpasskomponente werden Gleichspannungen geblockt.

Das Gegenteil des Bandpasses ist die **Bandsperre**. Hier wird ein bestimmter Frequenzbereich bedämpft, aber hohe und tiefe Frequenzen werden durchgelassen (**Bild 5**). Die Tiefpasskomponente sorgt für das Durchlassen von Gleichspannungsanteilen.

Ein **Allpass** ändert nichts an der Amplitude, hat also einen flachen Frequenzgang. Allerdings ändert sich die Gruppenlaufzeit bzw. die Phase (**Bild 6**) des Ausgangssignals über der Frequenz. Addiert man die Ausgangssignale von (mehreren) Allpässen zum Originalsignal, entsteht ein Kammfilter mit entsprechenden Notches im Amplitudenverlauf.

Filter-Approximationen

Wie kann man Filter designen und ihre Eigenschaften festlegen? Das ist vor allem für Filter höherer Ordnung keine triviale Angelegenheit. Hierfür haben berühmte Mathematiker vor vielen Jahren nach speziellen Regeln mathematische Lösungen entwickelt. Diese Regeln gelten immer noch, lediglich die Rechnerei wird uns heute durch PC und Filter-Programme erleichtert. Zu Ehren dieser Mathematiker tragen die entsprechenden Filtertypen ihre Namen. Eine Auswahl gebräuchlicher Filtertypen ohne Anspruch auf Vollständigkeit:

- Bessel
- Butterworth
- Tschebyscheff
- Cauer

Darüber hinaus sind noch viele weitere Filtervarianten denk- und machbar - man kann sogar verschiedene Charakteristiken kombinieren. Nachfolgend werden die wichtigsten Eigenschaften dieser Filtertypen beleuchtet, um die Auswahl eines Filters für eine bestimmte Anwendung zu erleichtern. In **Bild 7** sind vier unterschiedliche passive Tiefpässe dargestellt. Die zugehörigen Frequenzgänge finden sich in **Bild 8**.

Obwohl die Schaltung der oberen drei Filter bis auf die Bauteilewerte identisch ist, ergeben sich beim Frequenzgang deutliche Unterschiede. Beim Besselfilter ist der Übergang vom Durchlass- in den Sperrbereich sehr sanft. Ein Butterworthfilter hat schon einen steileren Frequenzgang. Das Tschebyschefffilter ist nochmals deutlich steiler. Das Cauerfilter generiert mit zwei zusätzlichen Kondensatoren über den Spulen Bandsperren bei zwei bestimmten Frequen-

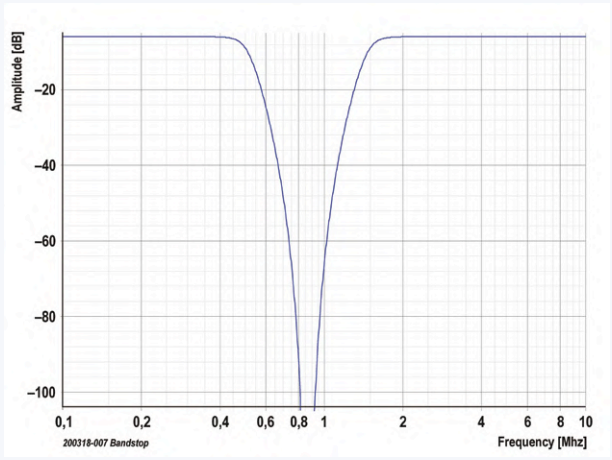


Bild 5. Frequenzgang einer Bandsperre mit doppelt logarithmischer Skalierung.

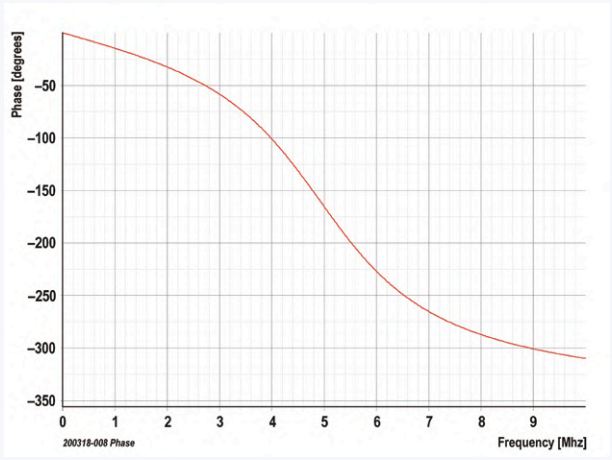


Bild 6. Phasengang eines Allpasses mit linearer Frequenzskala.

zen und erreicht so einen noch steileren Übergang in den Sperrbereich. Der Frequenzgang ist daher nicht stetig und gerade für höhere Frequenzen bleibt die Dämpfung fast konstant und niedriger als bei den anderen drei Filtertypen. Die Lage der „Notches" ergibt sich aus der Konstruktion. Bei passender Auslegung kann man mit diesen Notches gezielt bestimmte Frequenzen unterdrücken. Je nach Auslegung verändert sich dann allerdings der „Buckel" zwischen den beiden Notches und die Dämpfung bei hohen Frequenzen.

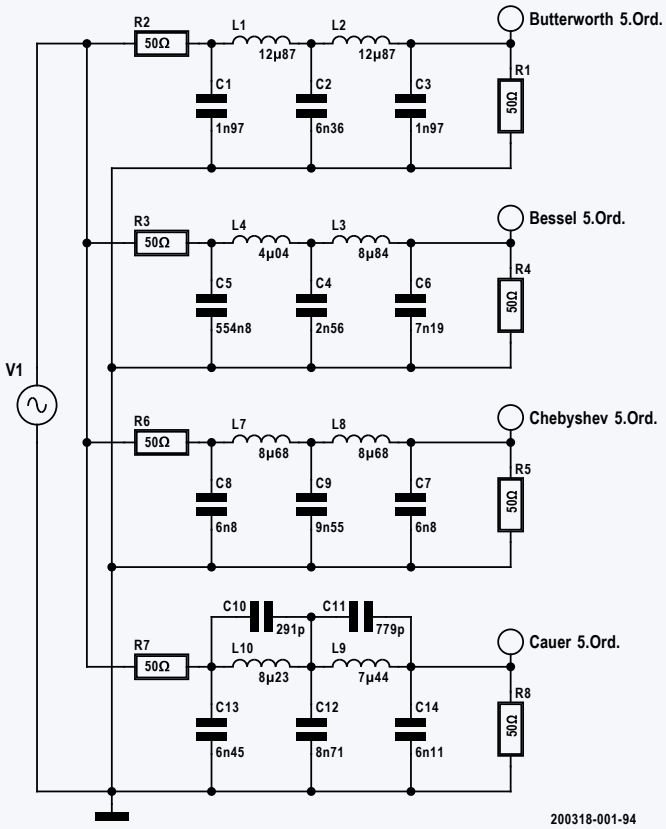


Bild 7. Schaltungen passiver 1-MHz-Tiefpässe 5. Ordnung (Butterworth, Bessel, Tschebyscheff, Cauer).

In **Bild 9** kann man die Welligkeiten im Durchlassbereich und den Amplitudenverlauf rund um den -3-dB-Punkt genauer inspizieren. Bessel dämpft schon recht früh im Durchlassbereich, dafür aber sanft. Butterworth verläuft fast ideal, ähnlich wie bei einfachen RC-Tiefpässen, aber dafür nur mit mittlerer Flankensteilheit. Tschebyscheff und Cauer zeigen eine spezifische Welligkeit im Durchlassbereich mit einem Amplitudenabfall bis zu -1 dB. Ihr Amplitudenverlauf schneidet bei der Nennfrequenz nicht den -3-dB-Pegel, sondern erreicht hier -1 dB. Gerade Letzteres gilt es beim Design in Hinterkopf zu behalten. Bei Tschebyscheff und Cauer gilt zudem: Je steiler das Filter, desto größer die Welligkeit im Durchlassbereich.

In **Bild 10** ist zu sehen, dass die Gruppenlaufzeit beim Besselfilter von der Frequenz unabhängig verläuft. Beim Butterworthfilter steigt die Gruppenlaufzeit kontinuierlich mit der Frequenz. Bei Tschebyscheff und Cauer gibt es auch eine Welligkeit beim Verlauf der Gruppenlaufzeit.

Interessant ist auch der konkrete Verlauf der Ausgangssignale, wenn man die Filter mit Flanken anregt, bzw. sie mit einem Rechtecksignal ansteuert (**Bild 11**). Die Sprungantwort des Besselfilters (**Bild 12**) zeigt nahezu kein Überschwingen. Das Butterworthfilter (blauer Graph) zeigt deutliches aber schnell abklingendes Überschwingen. Bei Tschebyscheff und

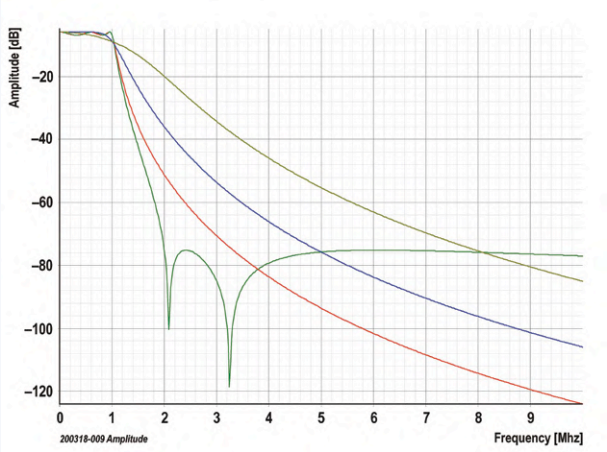


Bild 8. Frequenzgänge der vier Tiefpässe von Bild 7 mit logarithmischer Amplituden- und linearer Frequenzskala. Kurven: hellgrün = Bessel, blau = Butterworth, Tschebyscheff = rot und Cauer = dunkelgrün.

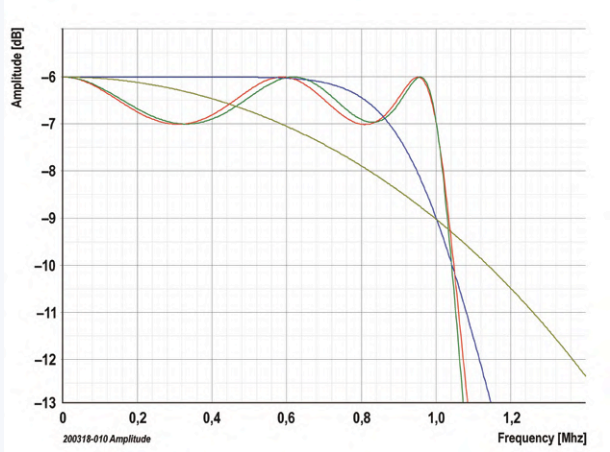


Bild 9. Die vier Frequenzgänge von Bild 8 höher aufgelöst. Man sieht den Verlauf im Durchlassbereich und um den -3-dB-Punkt genauer.

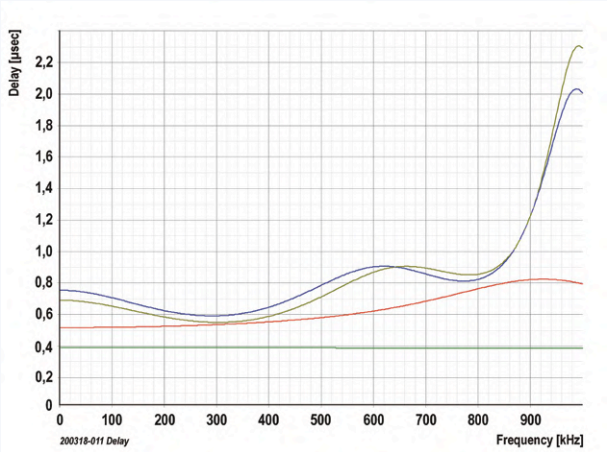


Bild 10. Gruppenlaufzeit über der Frequenz der vier Filter im Durchlassbereich. Kurven: dunkelgrün = Bessel, rot = Butterworth, hellgrün = Tschebyscheff und blau = Cauer.

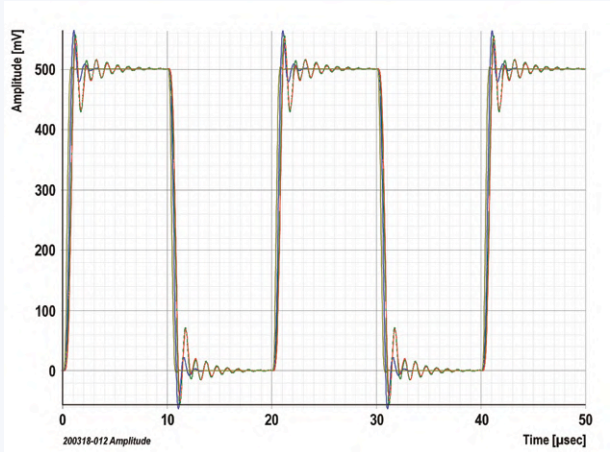


Bild 11. Verlauf der Ausgangssignale der vier Filter bei Zuführung eines Rechtecksignals. Kurven: hellgrün = Bessel, blau = Butterworth, rot = Tschebyscheff und dunkelgrün = Cauer.

Cauer kann man quasi gedämpfte Schwingungen sehen. Bei den vier Filtern sind zudem die ansteigenden Flanken zeitlich versetzt, was ja auch durch die unterschiedlichen Gruppenlaufzeiten zu erwarten ist. Neben rein elektrotechnischen Überlegungen kann man die Welligkeiten der Signalverläufe auch spektral herleiten: Ein Rechtecksignal kann man sich laut Fourier als Kombination seiner sinusförmigen Frequenzanteile vorstellen. Diese Frequenzen erfahren beim Filterdurchgang eine unterschiedliche Änderung ihrer Amplitude und gegebenenfalls auch eine unterschiedliche zeitliche Verschiebung. Kein Wunder, dass am Ausgang der Filter keine stetigen Amplitudenverläufe resultieren. Beim Besselfilter werden höhere Frequenzanteile zwar gedämpft, aber dank der konstanten Gruppenlaufzeit nicht gegenüber tieferen Frequenzen verschoben. Dadurch wird die Anstiegs- und Abfallzeit vergrößert und das Rechtecksignal erhält sozusagen runde Ecken. Durch die stark frequenzabhängige Gruppenlaufzeit bei Tschebyscheff und Cauer ergeben sich die beobachtbaren Überschwinger und Welligkeiten nach einer schnellen Änderung des Eingangssignals an einer Flanke.

Dieses Verhalten war z.B. einst in der analogen Videotechnik sehr störend. Durch Hinzufügen von Allpässen hat man dann den Verlauf der Gruppenlaufzeit nivelliert, um letztlich die Überschwinger zu verringern und um die Flanke herum zu zentrieren. Das Design solcher Allpässe zur Laufzeitkorrektur ist eine Wissenschaft für sich. Nur wenige Design-Programme bieten solche Berechnungen an. Außerdem vergrößert sich damit der Bauteileaufwand gegenüber einem reinen Tiefpass.

Heuristiken zur Filterauswahl

Im **Kasten Filter-Eigenschaften** sind die wichtigsten Features der vier gebräuchlichsten Filtertypen zusammengefasst. Nachfolgend werden daher nochmals die Besonderheiten aufgeführt:

Filter-Eigenschaften

Charakteristik	Durchlassber.	Sperrbereich		Steilheit
Filtertyp	Verlauf	Verlauf	Dämpfung	
Bessel	sanft fallend	sanft fallend	gering	gering
Butterworth	flach	stetig fallend	mittel	mittel
Chebyshev	wellig	stetig fallend	groß	groß
Cauer	wellig	wellig	groß	groß

Ein **Besselfilter** zeigt optimales Impulsverhalten und eignet sich daher zur Formung eines Impulses mit „runden Ecken“ aus einem digitalen, steilflankigen Signal. Zur Frequenzunterdrückung ist es aufgrund der geringen Steilheit kaum zu gebrauchen.
Das **Butterworthfilter** bietet einen guten Kompromiss zwischen Filtersteilheit und Impulsverhalten. Es zeigt keine Welligkeiten im Amplitudenverlauf.
Beim **Tschebyscheff-** und **Cauerfilter** erreicht man eine hohe Steilheit bei der Grenzfrequenz, erkauft sich das allerdings durch Welligkeiten im Durchlassbereich und deutliche, länger anhaltende Überschwinger im Zeitverlauf des Signals. Nachfolgend geht es darum, wie sich die Eigenschaften eines Filters mit seiner Ordnung, bzw. der Anzahl frequenzbestimmender Bauteile verändern. Als Beispiele in **Bild 13** dienen Butterworthfilter der Ordnungen 5, 6, 7 und 9. Schaut man sich die Bauteilwerte an, so erkennt man eine Symmetrie bei den ungeraden Ordnungen. In **Bild 14** ist schön zu sehen, wie sich die Steilheit der Filter mit der Ordnung erhöht. **Bild 15** zeigt die Verhältnisse rund um den -3-dB-Punkt vergrößert: Alle Filter schneiden sich genau in diesem Punkt. Filter mit höherer Ordnung schwächen erst bei höheren Frequenzen ab, dann aber um so heftiger.

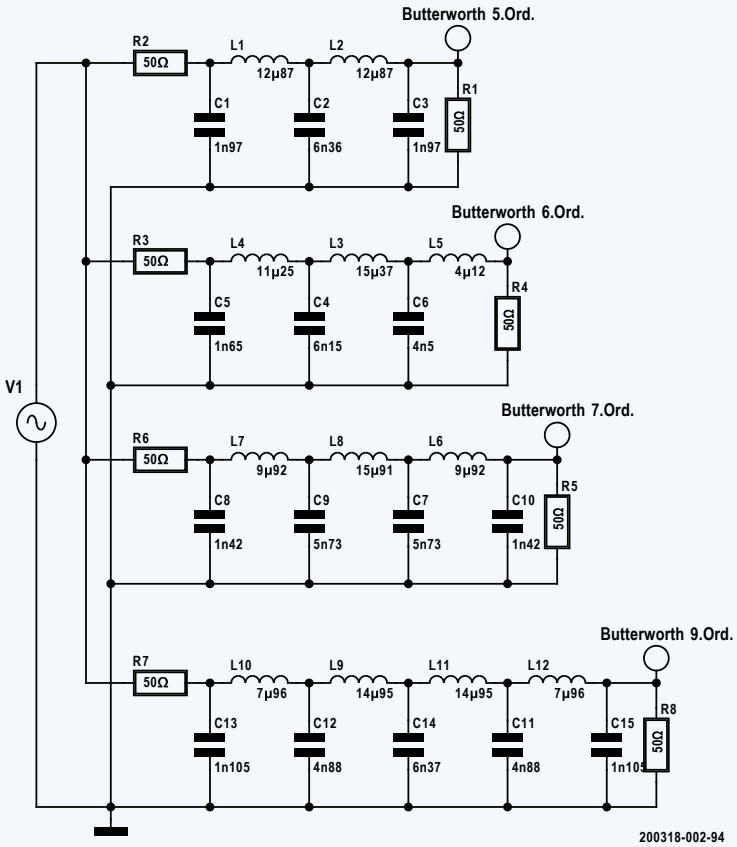


Bild 13. Schaltungen von Butterworthfiltern der Ordnungen 5, 6, 7 und 9.

Filter-Termini

Technologie	Analog passiv, analog aktiv, digital
Typ	Tiefpass, Hochpass, Bandpass, Bandsperre, Allpass
Approximation	Bessel, Butterworth, Tschebyscheff, Invers Tschebyscheff, Cauer, Lehmann, Legendre, Gauß, Raised Cosine, TBT
Impedanz	Eingangsimpedanz, Ausgangsimpedanz
Frequenzen	Grenzfrequenz, Sperrfrequenz, Durchlassbereich, Sperrbereich, Bandbreite
Dämpfung	Durchlassdämpfung, Sperrdämpfung
Weitere Begriffe	Gruppenlaufzeit, Impulsverhalten, Ordnung, Aufwand und Empfindlichkeit gegen Bauteiltoleranzen

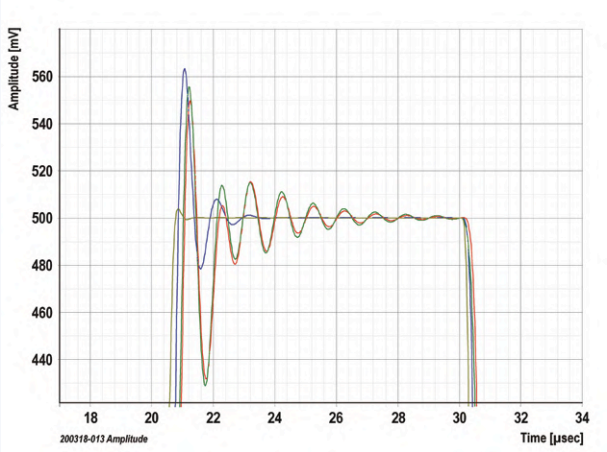


Bild 12. Die Sprungantworten sind in einer Ausschnittvergrößerung von Bild 11 gut zu erkennen. Kurven: hellgrün = Bessel, blau = Butterworth, rot = Tschebyscheff und dunkelgrün = Cauer.

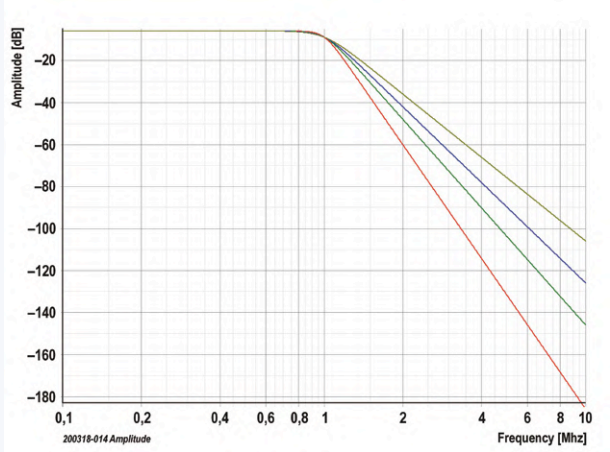


Bild 14. Frequenzgänge nach Filterordnung mit doppelt logarithmischer Skalierung. Filterordnung: 5 = hellgrün, 6 = blau, 7 = dunkelgrün und 9 = rot.

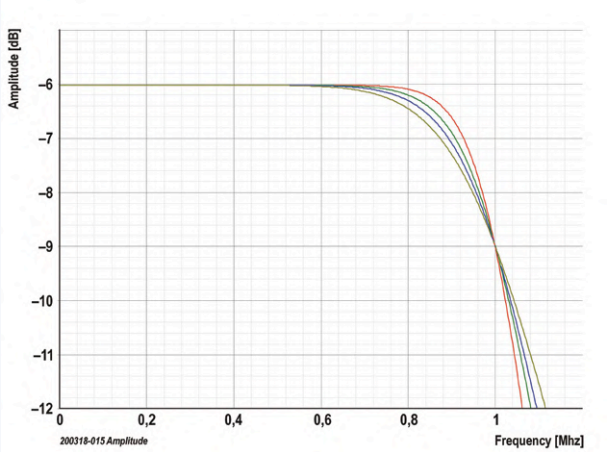


Bild 15. Frequenzgänge nach Filterordnung, gegenüber Bild 14 rund um den -3-dB-Punkt vergrößert.

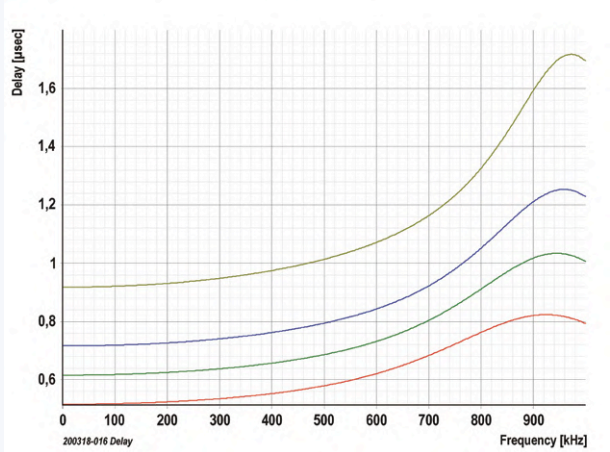


Bild 16. Gruppenlaufzeiten über der Frequenz bei vier verschiedenen Filterordnungen. Kurven: hellgrün = 9, blau = 7, dunkelgrün = 6 und rot = 5.

Je höher die Ordnung desto größer ist die Gruppenlaufzeit (**Bild 16**) und das Überschwingen (**Bild 17**) in Amplitude und Dauer. Die Steilheit eines Filters wird also sowohl von der gewählten Approximation bzw. dem Filtertyp **und** seiner Ordnung bestimmt. Durch Wahl einer hohen Ordnung und damit hohem Bauteileaufwand kann man auch ein steileres Besselfilter designen, Im **Kasten Filter-Termini** sind noch einmal die wichtigsten Begriffe zusammengefasst, die bei Filtern gebräuchlich sind.

200318-02

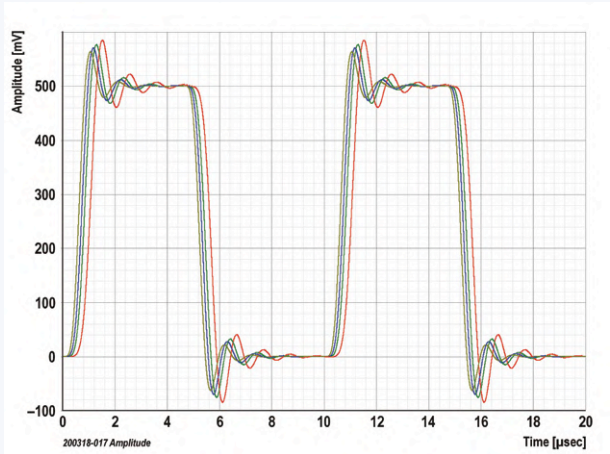


Bild 17. Sprungantworten bei vier verschiedenen Filterordnungen. Kurven: hellgrün = 5, blau = 6, dunkelgrün =7 und rot = 9.

WEBLINK

[1] Simetrix: http://www.simetrix.co.uk

Rigol DP832 3-Kanal-Netzgerät (0-30 V, 0-3 A, 195 W)

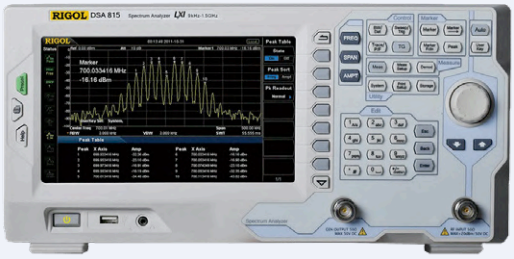
normal ~~549,00 €~~
jetzt 449,00 €

www.elektor.de/17823

Rigol DG2052 Arbiträr-Funktionsgenerator (50 MHz)

normal ~~699,00 €~~
jetzt 579,00 €

www.elektor.de/19345



Rigol DSA815-TG Spektrum-analysator (9 kHz – 1,5 GHz)

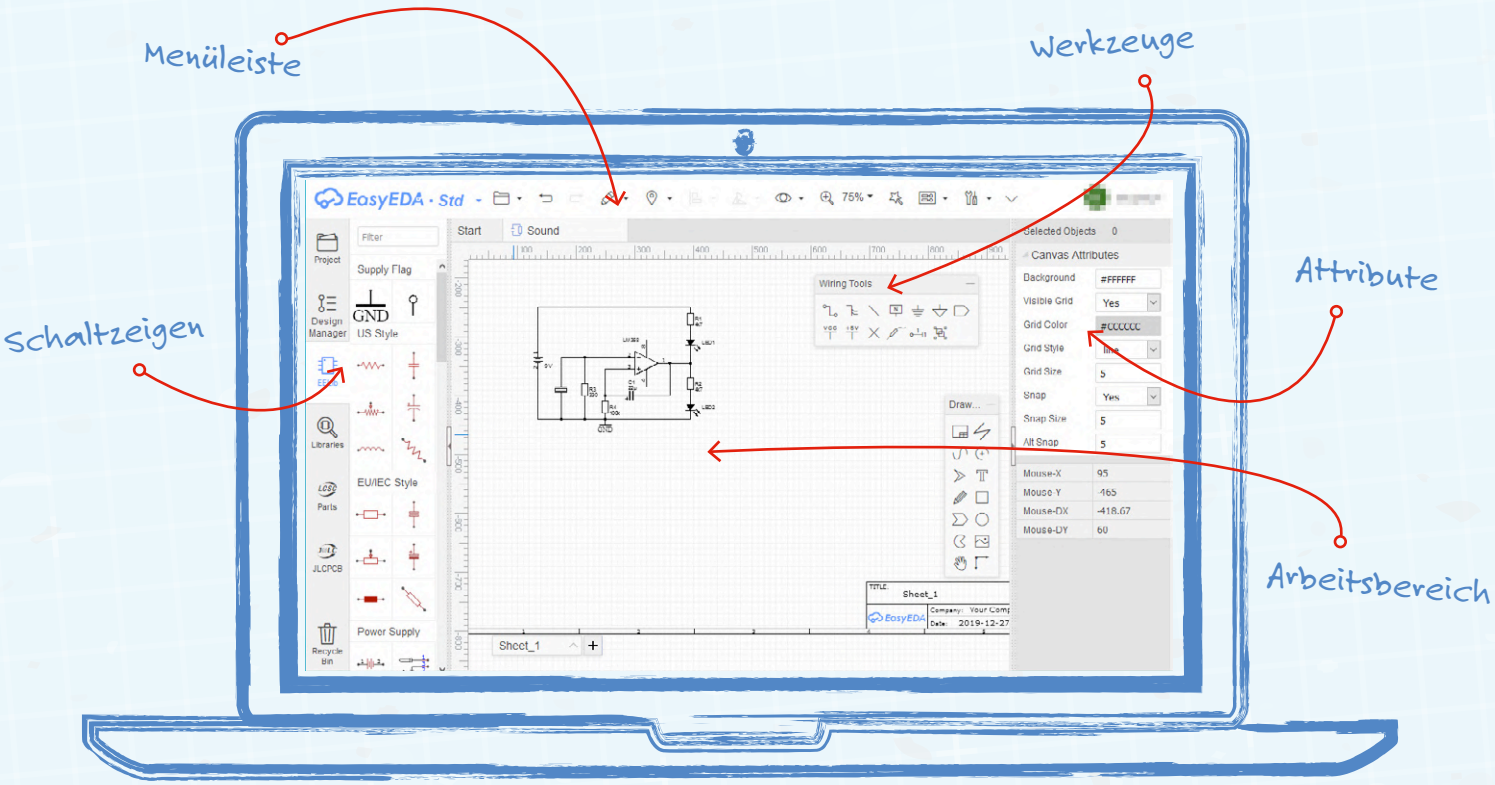
normal ~~1.549,00 €~~
jetzt 1.299,00 €

www.elektor.de/19349

Rigol DG2102 Arbiträr-Funktionsgenerator (100 MHz)

normal ~~1.149,00 €~~
jetzt 949,00 €

www.elektor.de/19347



Schaltpläne erstellen

Von **Florian Schäffer**

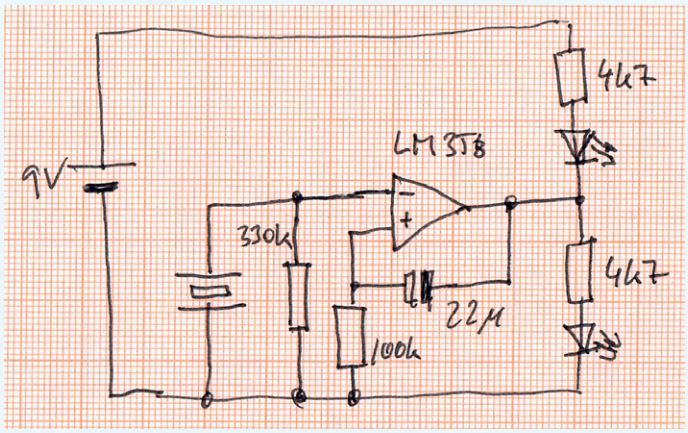
Sobald Sie eine eigene Schaltungsidee haben und diese mit anderen teilen oder für sich selber dokumentieren wollen, wird es Zeit, einen Schaltplan zu erstellen. Mit EasyEDA ist das fast einfacher, als von Hand zu zeichnen und sieht auf jeden Fall besser aus.

Prinzipiell können Sie Schaltpläne natürlich mit jeder Grafiksoftware erstellen. In dem Fall müssen Sie aber alle Symbole selbst erstellen und vor allem profitieren Sie nicht von den Funktionen, die spezielle EDA-Programme (Electronic Design Automation, zu Deutsch Entwurfsautomatisierung elektronischer Systeme) bieten. Die Programme erleichtern nicht nur den Schaltplanentwurf, in dem sie umfangreiche Bibliotheken mit Schaltzeichen und Bauteilen besitzen, sondern können teilweise auch die Schaltung auf logische Fehler und Funktion überprüfen oder sogar simulieren. Auf Basis des Schaltplans helfen sie Ihnen auch bei der Gestaltung von Leiterplatten (Platinen, englisch printed circuit board: PCB).

Welche Software ist perfekt?

Eine allgemeingültige Antwort gibt es hierzu nicht. Das hier gezeigte EasyEDA bietet den Vorteil, dass es kostenlos und im Webbrowser ohne lokale Installation benutzt werden kann. Für die gelegentliche Nutzung und als Einstieg ist das praktisch und einfach. Mit Fritzing können nicht nur die beliebten Breadboardansichten, sondern auch Schaltpläne erstellt werden. Allerdings sind diese in jeder Hinsicht unansehnlich und vor allem fehlerhaft beziehungsweise entsprechen die Skizzen nicht den europäischen Maßstäben. KiCad und gEDA sind kostenlos verfügbar, erfordern aber eine intensive Einarbeitung und die Erstnutzung ist aufgrund fehlender Installationsmechanismen umständlich. Target 3001 ist ein günstiges kommerzielles Programm, von dem es auch eine (stark eingeschränkte) Version für private Nutzer gibt. In der Profi-Liga spielen Programme wie Altium Designer und EAGLE. Von letzterem gibt es ebenfalls eine eingeschränkte Gratisversion.

In diesem kleinen Workshop lernen Sie die grundlegende Bedienung von EasyEDA kennen. Mit Hilfe des Schaltplans können Sie auf der Webseite ebenfalls eine Platine entwerfen, worauf hier aber nicht eingegangen wird.

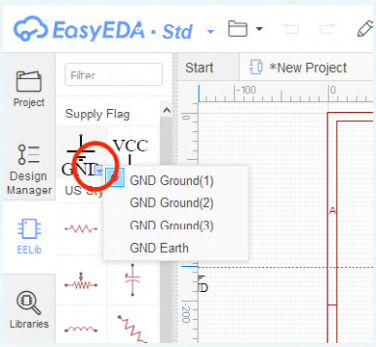


Eine einfache Schaltungsskizze soll mit einer EDA-Software nachgezeichnet werden (die Funktion ist hierfür belanglos).

1. Melden Sie sich auf der Webseite des Online Editors https://easyeda.com/editor oben rechts mit LOGIN an oder registrieren Sie bei Ihrem ersten Besuch einen neuen Zugang: REGISTER. Für das Zeichnen eines Schaltplans benötigen Sie keinen Zugang aber dann können Sie ihn nicht speichern oder exportieren.

2. Sie könnten die Sprache für die Oberfläche auf Deutsch umstellen, da die Übersetzung aber teilweise fehlt oder mangelhaft ist, belassen Sie es besser bei Englisch. Erstellen Sie einen neuen Plan mit DOCUMENT/NEW/SCHEMATIC.

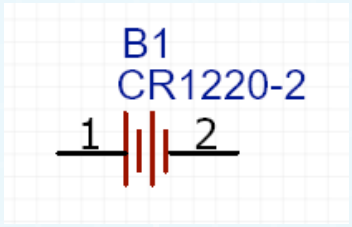
3. Am linken Rand befinden sich verschiedene Kategorien. Aktivieren Sie für die ersten Schritte EELIB, so dass daneben die gängigsten Bauteilsymbole gezeigt werden. Die Liste können Sie mit dem Mausrad scrollen. Es gibt sowohl Symbole im US-Stil, als auch im EU-Stil nach IEC (International Electrotechnical Commission). Wenn sich der Mauszeiger über einem Symbol befindet, erscheint bei den meisten Symbolen ein kleines Dreieck, hinter dem sich ein Kontextmenü befindet. Darüber können Sie spezielle Bauformen auswählen, die entweder das Aussehen des Symbols ändern oder für die Anfertigung von Leiterplatten wichtig sind – in den meisten Fällen benötigen Sie die Einstellung nicht, wenn Sie nur einen Schaltplan zeichnen.



4. Wählen Sie aus dem Bereich POWER SUPPLY das Batteriesymbol durch Anklicken an. EasyEDA bietet hier nur Knopfzellen an, aber da die Bauform des Bauteils derzeit nicht relevant ist, spielt das keine Rolle.

5. Am Mauszeiger hängt nun das Symbol und Sie können es auf den Arbeitsbereich bewegen. Mit dem Mausrad können Sie in den Arbeitsbereich hinein und heraus zoomen.

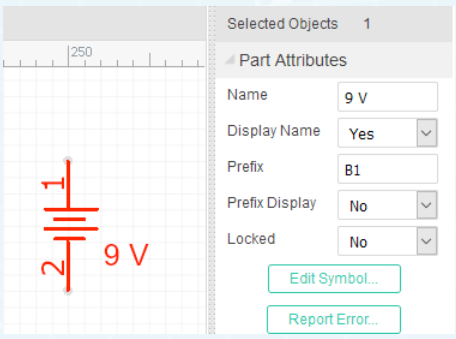
6. Klicken Sie auf den Arbeitsbereich, um das Schaltzeichen abzulegen. Durch weitere Klicks könnten Sie weitere Symbole ablegen, was aber für die Batterie nicht notwendig ist. Deshalb beenden Sie die Einfügeoperation durch drücken von Esc.



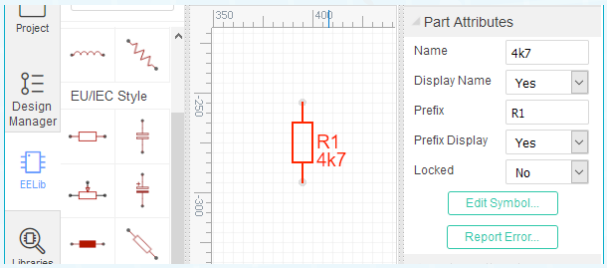
7. Sie können das Schaltzeichen jederzeit mit der Maus verschieben oder mit Entf löschen. Achten Sie dabei darauf, das Sie das Schaltzeichen anklicken und nicht nur die Beschriftung.

8. Markieren Sie das Batteriesymbol durch anklicken, um es anschließend zu drehen: Menü ROTATE AND FLIP/ROTATE RIGHT.

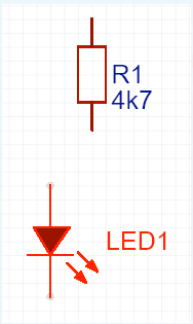
9. Am rechten Fensterrand können Sie die Attribute des Schaltzeichens anpassen, solange es ausgewählt ist. Ändern Sie die Angabe bei NAME in „9 V" und stellen Sie bei PREFIX DISPLAY den Wert NO ein.



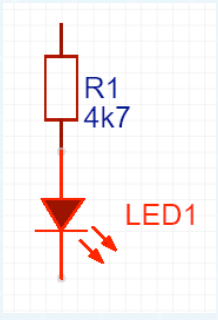
10. Fügen Sie einen Widerstand ein, drehen diesen um 90° nach rechts und ändern Sie bei NAME im Bereich der Attribute den Wert auf „4k7".



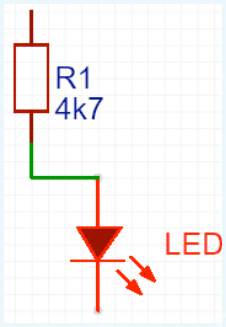
11. Fügen Sie das Symbol für Leuchtdioden (LED) in den Schaltplan in der Nähe des zuvor eingefügten Widerstands ein. Drehen Sie die LED nach rechts und wählen Sie als Attribut, dass der Name nicht zu sehen ist: NO bei DISPLAY NAME.



12. Die beiden Bauteile müssen elektrisch miteinander verbunden werden. Dazu muss eine Leitung (englisch: wire) eingezeichnet werden. Sie können dazu die beiden Zeichen einfach so zueinander schieben, dass sich die Anschlüsse berühren.

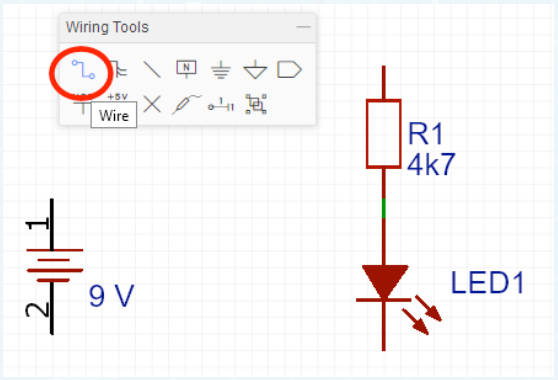


13. EasyEDA erstellt eine (nicht sichtbare) Verbindung. Sobald Sie eins der beiden Symbole wieder ein Stück wegschieben, sehen Sie, dass eine grüne Leitung existiert.

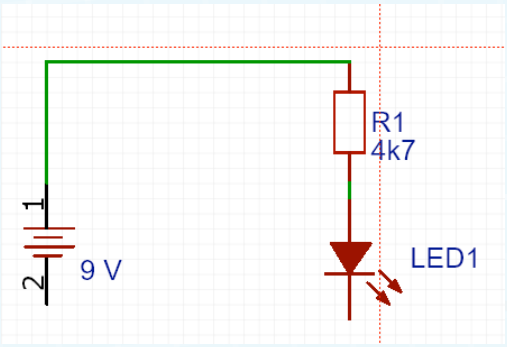


14. Auch diese Leitung können Sie bearbeiten, nachdem Sie sie angeklickt haben, um sie zu verschieben oder auch wieder zu löschen. Solange die Verbindung besteht, bleibt sie auch dann erhalten, wenn Sie die Symbole verschieben – das ist einer der großen Vorteile einer EDA-Software.

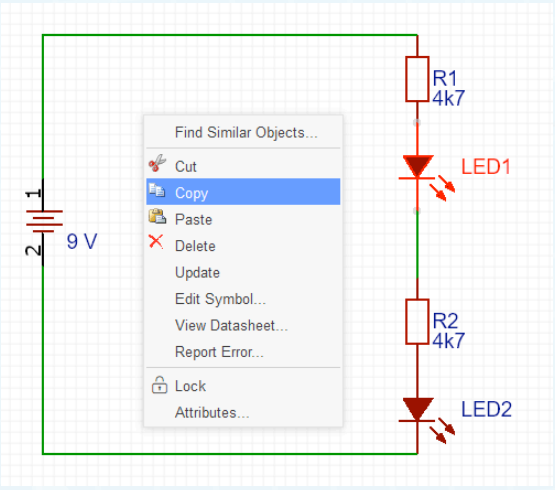
15. Sie können auch manuell Verbindungen einzeichnen, um beispielsweise den anderen Anschluss des Widerstands mit der Batterie zu verbinden. Es gibt zwei schwebende Fenster mit Werkzeugen (wenn nicht: VIEW/WIRING TOOLS). Wichtig ist, dass Sie elektrische Verbindungen immer mit WIRE und nie mit LINE aus dem Fenster DRAWING TOOLS zeichnen, denn dabei handelt es sich um einfache Grafiklinien.



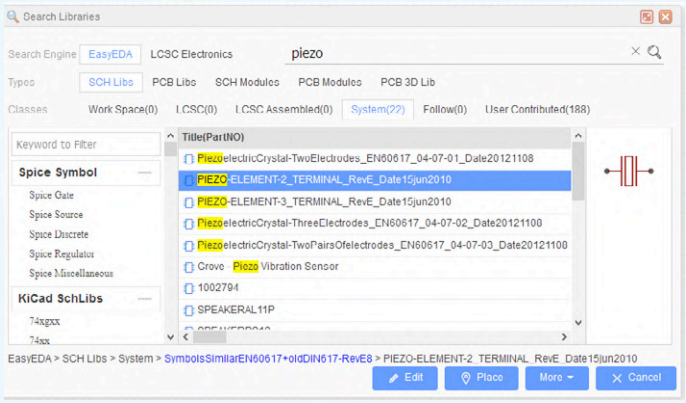
16. Aktivieren Sie WIRE und klicken Sie an den Anschluss der Batterie. Von dort wird nun eine grüne Verbindung zur Mausposition gezeichnet. Sie können die Linie direkt zum offenen Anschluss des Widerstands führen. Dabei gibt die Software einen Streckenverlauf vor. Wenn Ihnen dieser nicht zusagt, dann können Sie durch Mausklicks Zwischenpunkte setzen. Sobald Sie eine andere Verbindung oder einen Bauteilanschluss anklicken, ist die Verbindung fertiggestellt.



17. Fügen Sie die zweite LED und den zweiten Widerstand ein und verbinden Sie die Bauteile miteinander und mit der Batterie. Sie können entweder neue Symbole aus der linken Auswahl nutzen oder wie bei jedem Programm über die Zwischenablage mit COPY und PASTE arbeiten. Hierzu gibt es das Menü EDIT oder Sie nutzen das Kontextmenü, das sich bei einem Rechtsklick öffnet.



18. Zwei Schaltzeichen sind in der Standard-Liste nicht vorhanden: der Piezosummer und der Operationsverstärker. Um den letzteren einzufügen, klicken Sie ganz links auf das Symbol LIBRARIES, so dass sich das Fenster mit der Bauteilsuche öffnet. Geben Sie in das Suchfeld die Bezeichnung (Typnummer oder Beschreibung) ein: „LM358“ und klicken Sie auf das Lupensymbol rechts daneben.



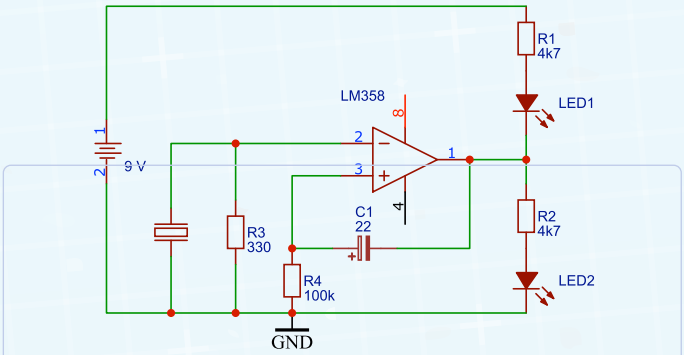
19. Nach einem Moment erscheint eine Liste der Suchergebnisse. Es gibt mehrere Einträge, die sich in der Form des Schaltzeichens (rechts am Rand) und des realen Bauteils unterscheiden. Für das Beispiel passt das Bauteil LM358APWR am besten. Dieses Modell besitzt zwei Operationsverstärker (es spielt derzeit keine Rolle, was die Aufgabe eines Operationsverstärkers ist und wie er funktioniert) in einem Gehäuse Wählen Sie den zweiten Untereintrag (LM358APWR.2) aus und klicken Sie auf PLACE: Das Fenster wird geschlossen und Sie können wie gewohnt das Symbol auf dem Arbeitsbereich einfügen.

20. Für die anderen Bauteile müssen Sie vermutlich etwas Platz schaffen und die bisherigen Symbole verschieben. Die Anschlüsse 4 und 8 für Spannungsversorgung des Operationsverstärkers bleiben einfach unbeachtet.

21. Auch nach dem Piezoelement müssen Sie suchen: Klicken Sie wieder auf Libraries und geben Sie als Suchbegriff „piezo“ ein. Der zweite Treffer in der Liste sieht ganz passend aus: Klicken Sie ihn an und fügen ihn mit PLACE in den Plan ein.

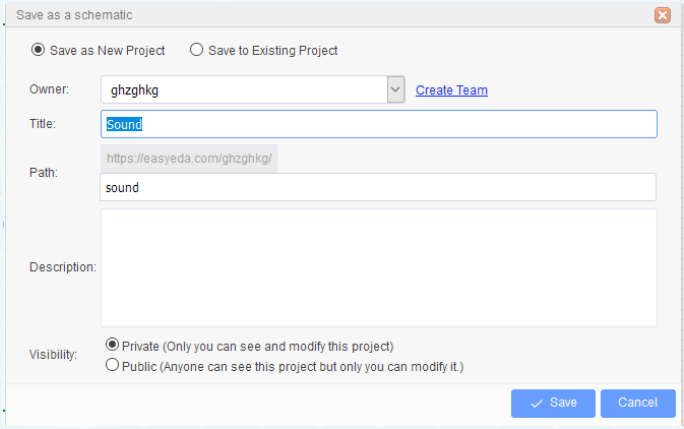
22. Die restlichen Schaltzeichen und Verbindungen sollten kein Problem mehr sein. EasyEDA beherrscht die Regeln zum Einzeichnen von Verbindungspunkten. Wenn sich zwei Leitungen einfach nur kreuzen, wird kein Punkt gezeichnet. Klicken Sie aber beim Verlegen einer Verbindung an eine andere Verbindung, wird ein Punkt hinzugefügt.

23. Der Vollständigkeit halber soll noch ein Massesymbol eingefügt werden. Wählen Sie, aus dem Bereich SUPPLY FLAG für das Symbol GND aus dem Kontextmenü den Eintrag GND GROUND(3), um das EU-Symbol für Masse zu bekommen und kein Erdungssymbol zu zeichnen.



So sollte am Ende der Schaltplan in etwa bei Ihnen aussehen.

24. Speichern Sie den Plan zum Schluss noch ab. Wählen Sie DOCUMENT/SAVE AS und geben Sie in dem sich öffnenden Fenster bei TITLE einen beliebigen Dateinamen ein. Die anderen Einstellungen belassen Sie wie sie sind. Sie könnten Ihr Projekt aber auch veröffentlichen, so dass andere Nutzer im Internet Zugriff haben, in dem Sie PUBLIC aktivieren. Klicken Sie auf SAVE zum Speichern.



25. Über DOCUMENT/EXPORT/... können Sie den Plan in verschiedenen Dateiformaten lokal speichern. Wünschen Sie eine monochrome Darstellung bei Grafikformaten, wählen Sie vorher THEME/BLACK ON WHITE.

Der Artikel wurde dem Elektor-Sonderheft „Einstieg in die Elektronik mit Arduino“ entnommen, das im Elektor-Shop bestellt werden kann: www.elektor.de/elektor-special-einstieg-in-die-elektronik-mit-arduino. Ab Oktober ist das Special auch in englischer Sprache erhältlich.

Treten Sie jetzt der Elektor Community bei!

Jetzt elektor GOLD membership Mitglied werden!

- Komplettes Webarchiv ab 1970
- 6x Elektor Doppelheft (Print)
- 9x Digital (PDF) inkl. Elektor Industry
- 10% Rabatt im Online-Shop und exklusive Angebote
- Elektor Jahrgangs-DVD
- Mit Tausenden von Mitgliedern des Online-Labors gemeinsam entwickeln mit Zugang zu über 1.000 Gerber-Dateien und direktem Kontakt zu unseren Experten!
- Veröffentlichen Sie Ihr eigenes Projekt oder verkaufen Sie direkt über unseren Shop!

Auch erhältlich

Die digitale elektor GREEN membership Mitgliedschaft!

- Zugang zu unserem Webarchiv
- 10% Rabatt in unserem Online-Shop
- 6x Elektor Doppelheft (PDF)
- Exklusive Angebote
- Zugang zu über 1.000 Gerber-Dateien

www.elektor.de/mitglied

elektor design › share › sell