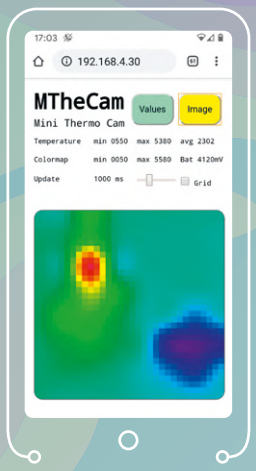
G3078 E (D) € 14,90 | CHF 27,00 | (A, B, L) € 16,00

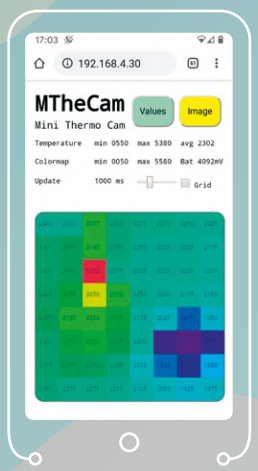
elektor MAG
design › share › sell

577 JANUAR/FEBRUAR 2021
ELEKTORMAGAZINE.DE

MTheCam

DIY-Mini-Wärmebildkamera



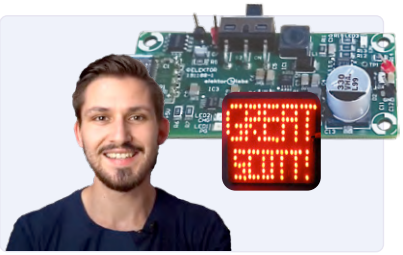


S. 8

In dieser Ausgabe

- I²S-Testsignalgenerator mit AVR-Mikrocontroller
- Sensor-Netzwerk mit RPi und RF24
- Mikrocontroller mit diskreter Logik
- Thermostat an ESPHome anschließen
- Schaltungen online simulieren
- Review: Weller Lötstation
- ESP32-Multitasking: Event Groups
- Neues LCR-Messgerät – Kalibrierung
- electronica Fast Forward 2020: Die Gewinner
- Suchmaschine für Open Hardware
- Datenanalyse und künstliche Intelligenz in Python
  Von unserer Partner-Zeitschrift Elettronica Open Source

und vieles mehr!



LiPo-Lader, -Booster und -Schutz
Mobile Stromversorgung von *GreatScott!* und Elektor
S. 6

Mehrkanal-Power-Analyzer
Drei Kanäle und Spektren auf LCD
S. 64

Parallax Propeller 2
Acht Kerne, 500 kB RAM, I/O mit bis zu 300 MHz
S. 110

4 198630 314904 10002

UNSER SORTIMENT VON TECHNIKERN FÜR TECHNIKER

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

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.

go-eCharger - die Kfz-Ladebox

Fixinstallation, 11 kW, Typ 2

Neben umfangreichen Sicherheitsfunktionen, wie Phasenerkennung, Erdungsmessung und Adaptererkennung, ist die Box mit RFID für die Zugangskontrolle sowie WLAN mit Hotspotfunktion ausgestattet und über die zugehörige App personalisierbar.

- Ausführung: 3-Phasen (16 A)
- Modell: 11 kW/ Typ 2
- Wifi, RFID
- Gigabit Ethernet für Netzwerkverbindungen
- Ladeleistung per Druckknopf und App einstellbar

PRODUKT-TIPP

Bestell-Nr.:
GO-E CH-02-18-1

666,80

GO-ECHARGER HOMEFIX 11 KW

KFW-FÖRDERUNG IN DEUTSCHLAND

Seit dem 24.11.2020 wird der go-eCharger HOMEfix 11 kW im Rahmen des deutschen KfW-Förderprogramms für private Ladeinfrastruktur gefördert. Der Charger lässt sich nicht nachträglich auf 22 kW umrüsten.

Mehr erfahren ▶

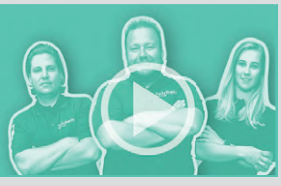
Viele weitere Produkte zu Elektromobilität finden Sie online!

Gleich entdecken ▶ www.reichelt.de/elektromobilitaet

AUF UNSEREM YOUTUBE KANAL FINDEN SIE

SPANNENDE ELEKTRONIK HOW-TO'S & UNBOXINGS

Direkt reinschauen: youtube.com/reichelt



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

www.reichelt.de

Bestellhotline: +49 (0)4422 955-333

reichelt elektronik – The best part of your project

TAGESPREISE! Preisstand: 11. 12. 2020

IMPRESSUM

52. Jahrgang, Nr. 577
Januar/Februar 2021
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. +31 (0)46 43 89444
Mobil: +31 6 380 780 29
E-Mail: margriet.debeij@elektor.com

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

Es gilt die Anzeigenpreisliste ab 01.01.2021.

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 ElektorMag

Zusammen entwickeln – jetzt erst recht!

Es sind keine leichten Zeiten für uns alle. Zum Glück können meine Kollegen und ich prima im Home-Office arbeiten. Das weggefallene Pendeln ins Büro und ein etwas lockererer Kleidungsstil sorgen sogar für ein schönes Sümmchen an gewonnener Freizeit, die man die Woche über in sein Lieblings-Hobby stecken kann. Ich habe vor, mir demnächst einmal den neuen Raspberry Pi vorzunehmen, auch das Thema Machine Learning interessiert mich (ein paar kleine Ideen, was man damit in Zusammenhang mit Musik/Video anstellen könnte, reifen schon).

Vielleicht haben auch Sie etwas mehr Zeit, und haben diese genutzt, um endlich mal eine alte Projektidee zu realisieren. Etwa mit einem der kleinen Boards, die Sie von einer der letzten (realen) Messen mitgenommen haben? Oder mit einem der Entwicklungsmodule aus unserem Shop? Wenn ja, dann zögern Sie nicht, dies uns und unsere Leser wissen zu lassen! Auf unserer Plattform *www.elektormagazine.de/labs* können Sie noch heute ein Projekt anlegen und in ein paar Zeilen beschreiben, was Sie planen oder schon entwickelt haben. Dort wird weder gutes Englisch noch eine vollständige Dokumentation verlangt – Material kann man später nachreichen. Ich bin mir fast sicher, dass sich schnell ein Gleichgesinnter meldet, der eine Frage zu Ihrem Projekt hat oder Ihnen schreibt, dass er schon etwas Ähnliches in Angriff genommen hat (und wo eventuell Fallstricke lauern). Gemeinsam entwickelt es sich besser, und das Netz der Netze macht es ja möglich, dass das auch während eines Lockdowns funktioniert!

Auch wir Redakteure schauen uns das regelmäßig an – keines der Labs-Projekte wird vergessen. Wir haben uns für 2021 zum Ziel gesetzt, dass wir schnell aussuchen, was sich dafür eignen könnte, in unserer Zeitschrift veröffentlicht zu werden. Das gilt natürlich auch für Einsendungen, die Sie uns per Mail zukommen lassen! Ein Redakteur aus unserem Team wird sich dann mit Ihnen in Verbindung setzen und Ihren Artikelvorschlag betreuen.

In diesem Sinne: Lassen Sie uns zusammen entwickeln und bleiben Sie gesund!

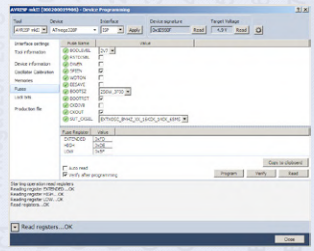
Unser Team

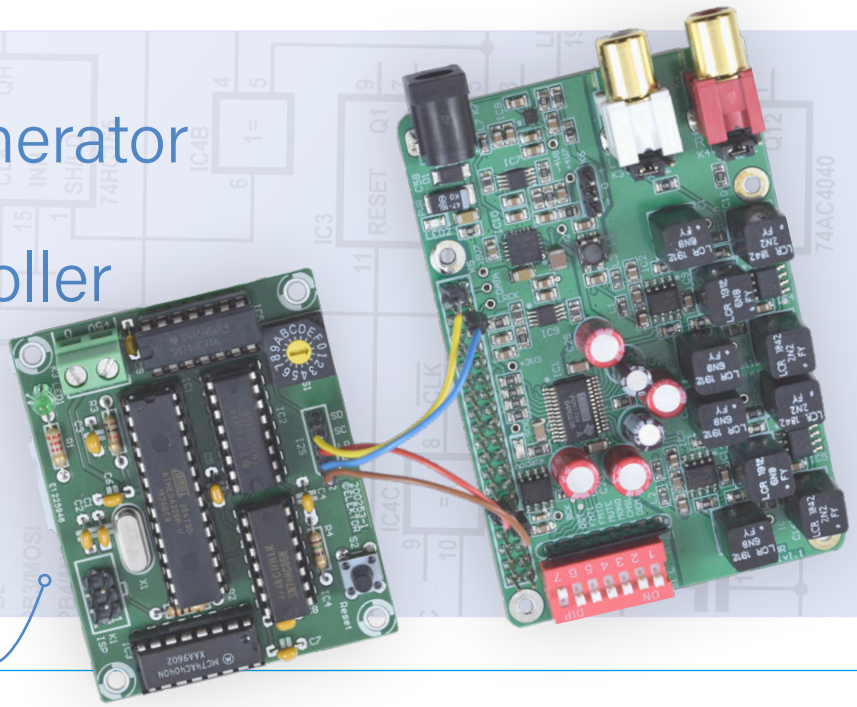
INHALT

Labor-Projekt

I²S-Testsignalgenerator mit AVR-Mikrocontroller

22





Rubriken

3 **Impressum**

20 **electronica Fast Forward 2020**
Die Gewinner

36 **Aus dem Leben gegriffen**
Der schmale Grat zwischen Ordnung und Chaos

38 **Aller Anfang …**
Basiskurs für Einsteiger

50 **Von Entwicklern für Entwickler**
Tipps & Tricks, Best Practices und andere nützliche Infos

53 **Bemerkenswerte Bauteile**
Das Dekatron

84 **Projekt 2.0**
Korrekturen, Updates und Leserbriefe

85 **Kleine Schaltungen**

96 **Elektor Ethics**
Das Open Hardware Observatory

114 **Hexadoku**
Sudoku für Elektroniker

Hintergrund

17 **Review: Lötstation WE 1010 von Weller**

32 **Schaltungen online simulieren**

60 **Praktisches ESP32-Multitasking (6)**
Event Groups

72 **Design analoger Filter**
Teil 3: Passive Filter

81 **Review: Funk-Messmodul JOY-iT VAX-1030**

94 **Fehleranalyse**
Tipps zu Spannungsregler-Schaltungen, Platinendesign und mehr

98 **Java auf dem Raspberry Pi**
Ein Interview mit Buch-Autor Frank Delporte

102 **Datenanalyse und künstliche Intelligenz in Python**
Interpretation realer Daten mit Numpy, Pandas und Scikit-Learn

110 **Parallax Propeller 2**
Teil 1: Kurz vorgestellt

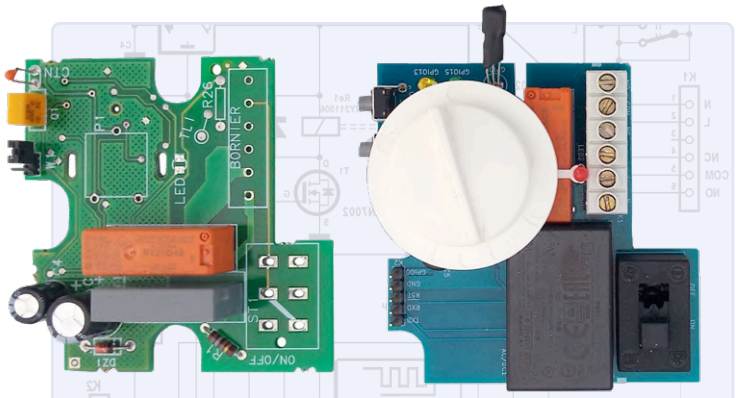
Projekte

6 **DIY LiPo Supercharger Bundle**
LiPo-Lader, -Booster und -Schutz von GreatScott! und Elektor

8 **MTheCam – Die Mini-Thermo-Cam**
Lokalisierung von Hot- und Cold-Spots

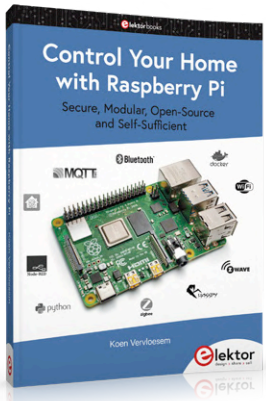
22 **I²S-Testsignalgenerator mit AVR-Mikrocontroller**
Digitales Sinus-Testsignal mit 32 Bit Auflösung, fs von 192 kHz und einstellbarem Pegel von 0 bis -110 dB

27 **Steuern Sie Ihr Zuhause mit dem Raspberry Pi**
Der RPi und das ISM-Band 433,92 MHz



Thermostat mit ESPHome verbinden 44

Schaltungen online simulieren 32



RPi empfängt auf 433,92 MHz 27

41 Mein Projekt
Ein diskret aufgebauter Mikrocontroller

44 Ein Thermostat im ESPHome
Hausautomatisierung weiter ausgebaut

54 Raspberry Pi Full Stack
RPi und RF24 als Herzstück eines Sensornetzwerks

64 Mehrkanal-Power-Analyzer
Bis zu drei Kanäle, mit grafischer und alphanumerischer Anzeige

88 Neues LCR-Messgerät 50 Hz bis 2 MHz
Teil 2: Betrieb, Kalibrierung und Firmware-Programmierung

Vorschau

Elektor März/April 2021

Die nächste Ausgabe wird von Ingenieuren aus Europa und den USA gemeinsam produziert. Das Elektor-Team arbeitet dabei mit Gastredakteuren von SparkFun Electronics zusammen, um für Sie eine Mischung aus erstaunlichen Projekten und interessanten Technik-Tutorials zusammenzustellen. Im Folgenden sind nur einige der geplanten Artikel aufgeführt:

Aus dem Inhalt:

- Eigene elektronische Geräte entwickeln - Tipps und Tricks
- GNSS-Referenzstation selbst gebaut
- Programmierung eines FPGAs
- Flexible Entwicklungsplattform MicroMod - erste Schritte
- Entwicklungsboards im Griff
- ClockClock: Ein FPGA-Demoprojekt
- FreeRTOS auf RED-V implementieren
- Must-Haves für Ihren Elektronik-Arbeitsplatz

Und vieles mehr!

Elektor März/April 2021 erscheint am 11. März 2021.
Änderungen vorbehalten.

DIY LiPo Supercharger Bundle

LiPo-Lader, -Booster und -Schutz von GreatScott! und Elektor

Von **Mathias Claußen** (Elektor)

Benötigen Sie eine wiederaufladbare LiPo-Stromversorgung mit 5-V- und 12-V-Ausgang? Möchten Sie das SMD-Löten üben? Das können Sie mit ein wenig Hilfe von GreatScott! (ein YouTuber mit mehr als einer Million Abonnenten) und Elektor. In diesem Artikel beschreiben wir sowohl die praktische mobile Stromversorgung als auch die Hindernisse, die wir während der Entwicklung überwinden mussten.

*Bild 1. Das **DIY LiPo Supercharger Bundle**, noch „in the box“.*

Beim Experimentieren mit Elektronik und um Prototypen zum Laufen zu bringen, setzen wir üblicherweise auf unser zuverlässiges Netzgerät auf dem Labortisch. Soll das von uns gebaute Gerät aber tragbar sein und sich im Labor bewegen, werden die Zuleitungen manchmal zu einem kleinen Handicap. Abhilfe schafft dann ein Bündel von Batterien, die sorgfältig mit Klebeband, Heißkleber und einem billigen DC/DC-Wandler zu einer Art tragbarem Akkupack verarbeitet werden. Das funktioniert bei einem Prototyp, aber es ist nicht so schön, besonders wenn Sie 5 V und 12 V für Ihr Gerät benötigen. Das kann man besser machen, und man macht es auch! Bei einem Treffen auf der Messe productronica 2019 in München beschlossen Elektor und der populäre Ingenieur GreatScott!, gemeinsam einen praktischen DIY-Bausatz zu entwickeln, speziell für Sie.

Wer ist GreatScott!? Great Scott! ist der Name eines Youtube-Kanals [1], der 2013 ins Leben gerufen wurde und mehr als einer Million Abonnenten elektronische Projekte und Elektronikwissen präsentiert. Viele Videos stellen Selbstbau-Projekte vor, die die Zuschauer nachbauen können, andere Videos bieten Inspiration und klare Lösungen für technische Probleme. Da es bereits auch etliche Videos mit Elektor-Produkten gibt, entstand die Idee, einen nützliches Do-it-yourself-Kit zu präsentieren, mit dem zudem jeder Interessierte sein Wissen und sein Können erweitern kann.

GreatScott! [2] hat den groben Schaltplan eines Netzteils skizziert, das auf wiederaufladbaren LiPo-Akkus basiert und vom Benutzer, also von Ihnen, mit SMD-Bauteilen aufgebaut werden soll. Alle Bauteile und sonstigen Bestandteile sehen auf den ersten Blick nicht so kompliziert aus, ein Lade-IC für LiPo-Akkus, ein DC/DC-Wandler, der 5 V und 12 V bereitstellt und ein Akku-Schutz-IC. Alle anderen Bauteile besitzen – soweit möglich – die Baugröße 1206, um auch Elektronik-Einsteigern zu ermöglichen, SMD-Bauteile ordnungsgemäß auf die Platine zu bekommen und hoffentlich zu zeigen, dass das Löten von SMDs keine schwarze Magie ist, die nur gut ausgebildete Zauberkünstler beherrschen. Einige kritische Bauteile sind sogar bereits auf der Platine vorbestückt, um Ihnen den Einstieg in die SMD-Bestückung etwas leichter zu machen und um den Umgang mit den kleinen Pins und Pads unter den ICs zu vermeiden.

Zum Aufladen der Akkus können Sie entweder 5 V über ein Kabel zuführen oder vorzugsweise den USB-C-Anschluss auf der Zusatzplatine verwenden. Vorzugsweise, denn bei der USB-C-Verbindung ist ein Verpolen des Akkus völlig ausgeschlossen. Die USB-Buchse

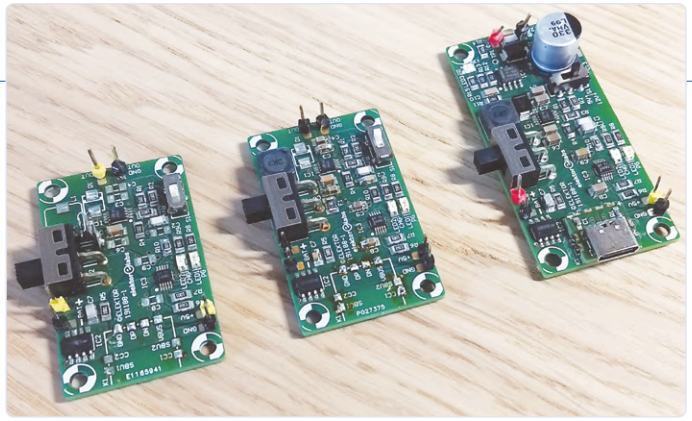


Bild 2. Die drei Platinen-Iterationen von links nach rechts.

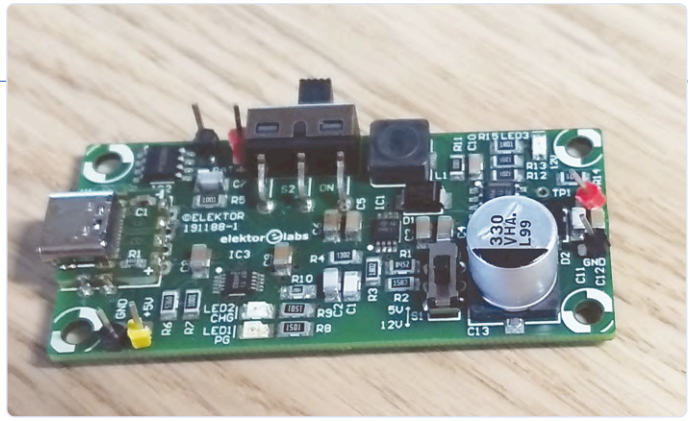


Bild 3. Die endgültige und vollständig bestückte Platinenversion.

Eigenschaften

- Eingangsspannung: 5 V ±10 %
- Ausgang: 5 V / 1,5 A oder 12 V / 0,75 A
- Einzelzellen-Lithium-Batterie

ist auf der Platine durch vier Befestigungslöcher sehr stabil befestigt. Für den sicherlich interessanten Aufbau des Netzteils gibt es eine schöne Schritt-für-Schritt-Anleitung, die im *DIY LiPo Supercharger Bundle* (**Bild 1**) enthalten ist.

Die drei ICs auf dem Board stammen von einem bekannten Hersteller. Das Batterieschutz-IC ist ein XB8089D von XySemi, das man nur bei Händlern findet, die sich auf chinesische Halbleiterprodukte spezialisiert haben. Dieser Chip bietet Überladungs-, Tiefentladungs-, Überstrom- und Verpolungsschutz in einem kleinen SOP8-Gehäuse mit Expose-Pad zu einem vernünftigen Preis. Obwohl viele Batterien, die in Produkten wie Drohnen verwendet werden, schon über einen integrierten Schutz verfügen, sollte man auf diese Schutzmaßnahme sicher nicht verzichten, wenn eine ungeschützte Lithium-Batterie eingesetzt wird.

Den Skizzen und dem groben Schaltplan, die GreatScott zur Verfügung stellte, fügte Elektor seine Erfahrung bei der Entwicklung und der Erstellung von Platinen hinzu. Man könnte meinen, dies sei das Ende der Geschichte, Sie haben eine Platine und alles ist in Ordnung, aber es gilt die Faustregel, dass es normalerweise drei Iterationen bis zum fertigen Produkt dauert, wie Sie in **Bild 2** sehen können. Der erste Prototyp links funktionierte fast wie erwartet, abgesehen von ein paar kleineren Problemen wie, dass die Batterie nicht geladen wurde oder der DC/DC-Wandler das Zeitliche segnete, wenn der Laststrom ein bestimmtes Maß überschritt. Das Batterieschutz-IC verhinderte zwar, dass die Lithium-Batterie dabei beschädigt wurde, aber die Stromgrenze liegt dann bei 10 A (entsprechend 37 W), während sich der DC/DC-Wandler schon bei etwa 15 W in Wohlgefallen auflöste. Das Problem mit dem Nicht-Aufladen wurde auch bald gelöst, es lag an einem Tippfehler, der aus einem gewünschten 10-kΩ-Widerstand einen 100-kΩ-Widerstand machte. Aber um ehrlich zu sein, wir wissen aus

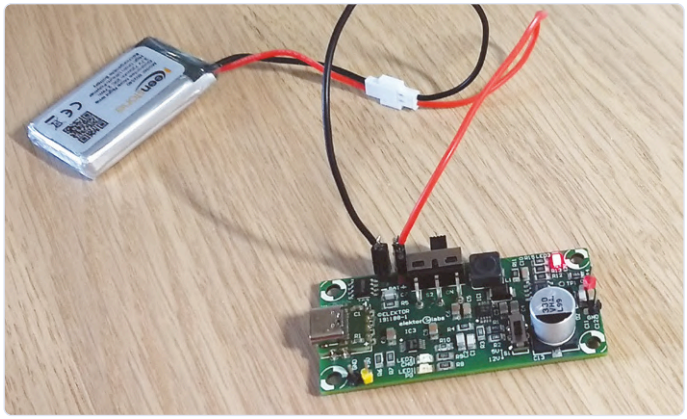


Bild 4. Die Schaltung in Aktion.

Erfahrung von vielen anderen Projekten, wo wir suchen müssen, wenn es um den falschen Widerstand für den Ladestrom geht. Das gewählte Schutz-IC BQ24092 stammt aus der BQ2409x-Familie, und hier müssen Sie das Datenblatt sorgfältig prüfen. Je nach Chip-Variante muss der erwähnte Widerstand nämlich 10 kΩ oder 100 kΩ groß sein. Einmal erkannt, war das Wechseln des Widerstands das geringste Problem. Da wir drei Iterationen durchgeführt haben, haben wir viel gelernt. Die Modifikation des DC/DC-Wandlers ist ganz einfach, wenn man weiß, welche Teile man auswählen kann. Auch bei der Zusammenstellung des Baupakets haben wir ein paar Lektionen gelernt, von denen Sie vielleicht profitieren. In der Zwischenzeit können Sie einen Blick auf den Youtube-Kanal von GreatScott! werfen und die Baugruppe und das Netzteil in Aktion sehen. Details und eine ausführliche Beschreibung der Schaltung folgen in der nächsten Elektor-Ausgabe. Die groben Spezifikationen sind im nebenstehenden Textkasten aufgeführt. Die endgültig überarbeitete Platine sieht wie in **Bild 3** aus, während **Bild 4** sie in Aktion zeigt.

191188-03

WEBLINKS

[1] **Youtube-Kanal von GreatScott!**: www.youtube.com/c/greatscottlab/

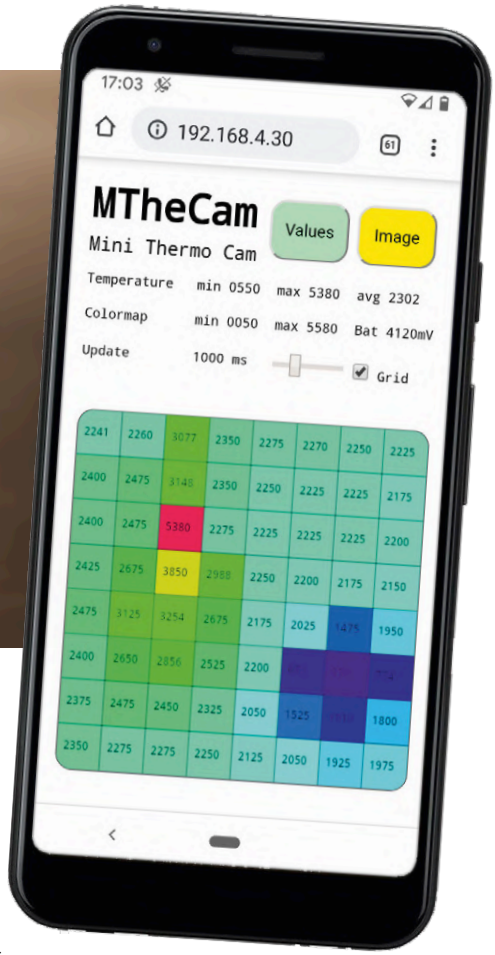
[2] **GreatScott! baut ein LoRa-Alarmsystem**: www.elektormagazine.de/magazine/elektor-150/58723

MTheCam
Die Mini-Thermo-Cam

Einfache Thermocam zur Lokalisierung von Hot- und Cold-Spots

Von **Olaf Mertens**

Ist die Kerze ausgepustet? Und die Herdplatte ausgeschaltet? Wer hat nicht schon mal das ungute Gefühl beim Verlassen seines Hauses gehabt, irgendeine Hitzequelle könnte unbeaufsichtigt großen Schaden anrichten? Ab jetzt kann man die MTheCam per Smartphone abfragen. Doch das hier vorgestellte Projekt auf Basis eines 8x8-Pixel-Sensors von Panasonic eignet sich natürlich noch für viele weitere Anwendungen.



Auch wenn man bei Hot-Spots eher an öffentliches Internet denkt, werden mit dem Begriff eigentlich Punkte bezeichnet, die gegenüber ihrer Umgebung deutlich wärmer sind (oder kälter bei Cold-Spots). Sie deuten auf Flammen, überhitzte Stellen und Kurzschlüsse sowie Wärmebrücken und undichte Stellen hin. Wenn sie nicht gerade mit offener Flamme leuchten oder glühen, sind sie für Menschen nicht sichtbar. Um sie aufzuspüren, braucht man einen passenden Sehhilfe-Sensor – MTheCam.

Die Lokalisierung von Hot-Spots ist nicht auf so etwas wie die vergessene Herdplatte beschränkt. In elektronischen Schaltungen zum Beispiel zeigen manche überforderten Komponenten schon früh durch Hitzeentwicklung ihr baldiges Ableben an. In Maschinen überhitzen verschlissene Lager und schlecht geschmierte Reibflächen – eine rechtzeitige Wartung rettet hier Laufzeiten. Selbst Personen lassen sich als Hot-Spots erkennen, man kann sie so zählen und ihre Bewegungen verfolgen.

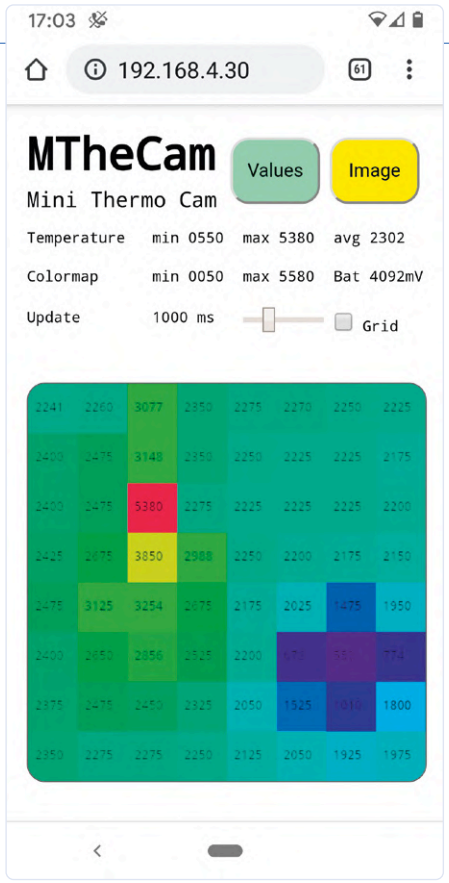


*Bild 1. Screenshot der Temperaturwerte mit 8*8 Pixeln.*

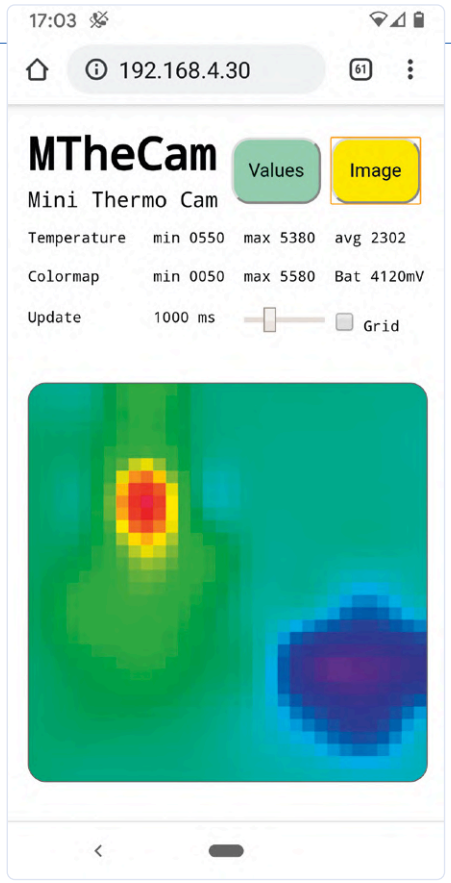


*Bild 2. Screenshot der auf 32*32 Pixel interpolierten Werte.*

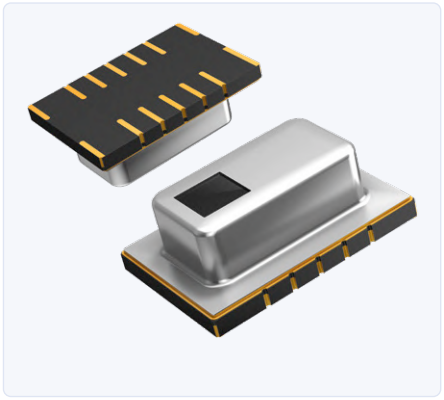


Bild 3. Der AMG88xx-Sensor von Panasonic.

Neben Messgeräten, die durch direkten Kontakt Temperaturen messen, gibt es berührungslose Sensoren, die in einiger Entfernung noch die IR-Strahlung von Objekten aufnehmen und daraus die Durchschnittstemperatur des Messpunktfeldes berechnen. Dazu wird der pyroelektrische Effekt genutzt, bei dem sich das elektrische Potenzial eines in Elektroden eingebetteten polarisierten Kristalls ändert, wenn er einer Wärmestrahlung ausgesetzt ist. Dieser Effekt lässt sich natürlich elektronisch nutzen [1].

Das Projekt

MTheCam misst bis zu fünfmal pro Sekunde gleichzeitig 64 Punkte in je acht Reihen und Spalten, die wie in einer optischen Kamera mit 60° Bildwinkel angeordnet sind. Jeder Punkt zeigt die Temperatur zwischen 0 Grad und 80 Grad Celsius (alternativ -20...100 °C) an. Den einzelnen Messwerten werden individuelle Farbwerte eines Farbverlaufs zugewiesen, so dass sich ein recht niedrig aufgelöstes Bild ergibt, das dann per WLAN als Webseite zum Beispiel auf einem Smartphone dargestellt werden kann. Ästhetisch ist dieser bunte Riesenpixelhaufen sicher nichts für verwöhnte HDTV-Augen, er zeigt aber gut Hot- oder Cold-Spots mit gegenüber dem Umfeld deutlich abweichenden Farbflächen. Zusätzlich zum Falschfarbenbild werden die Messwerte je Pixel auch als Temperaturen in Grad Celsius dargestellt, was eine präzisere Analyse zulässt (**Bild 1**).

Die 64 Messwerte sind darüber hinaus als JSON-Datensatz abrufbar, der dann auch als Datenlieferant für andere Anwendungen dient. Für die Illusion einer höheren Auflösung als die 8x8-Punktematrix wird über das mathematische Verfahren der bikubischen Interpolation [2] ein „gelogenes" Bild mit 32x32 Farbpunkten erzeugt, was nicht nur netter anzusehen ist, sondern in dem man Objekte auch leichter erkennen kann. Das Spektrum der Farbskala wird zudem auf den Bereich von minimal und maximal gemessenen Temperaturen plus etwas „Spiel" eingepasst, was dann wie eine Lupe auf die Temperaturmesswerte wirkt (**Bild 2**).

Der Thermosensor AMG88xx

Für die Aufnahme eines Thermobildes hat Panasonic ein leistungsfähiges MEMS (**Bild 3**) in zwei Varianten (sprich: Messbereichen) entwickelt, das die Optik, die thermoelektrische Umwandlung, die analoge Messung und weitere Dinge zur Datenaufbereitung in seinem recht kleinen Gehäuse vereinigt [3]. Die Messdaten im Bereich 0...80 °C (AMG8853) beziehungsweise -20...100 °C (AMG8854), die einen maximalen Fehler von +/-2,5 K (+/-3,0) aufweisen, werden über eine I²C-Schnittstelle ausgegeben. Die absolute Genauigkeit ist zwar nicht überragend, für die qualitative Beurteilung relativer Werte aber dennoch gut geeignet. Mit ein paar Zeilen Software kann man zudem eine Kalibrierung vornehmen, was die Genauigkeit erhöht und das Rauschen vermindert. Der Panasonic-Sensor ist in einem Platz sparenden SMD-Gehäuse untergebracht, das für Reflow-Löten vorgesehen ist. Für unsere Anwendung bringen wir es auf einem kleinen selbstgefertigten Breakout-Board unter, das einige notwendige passive Bauteile (im gut handlötbaren 0805-Format) zur Entkopplung der Versorgungsspannung und als Pullups für den I²C-Bus umfasst. Neben der Betriebsspannung von +5 V und GND liegen drei wichtige Leitungen (INT und den I²C-Bus) auf den Pads am Rand der Platine, in die eine (gerade oder gewinkelte) Stiftleiste führt. **Bild 4** zeigt die Schaltung dieses Breakout-Boards und zur Information verrät es auch

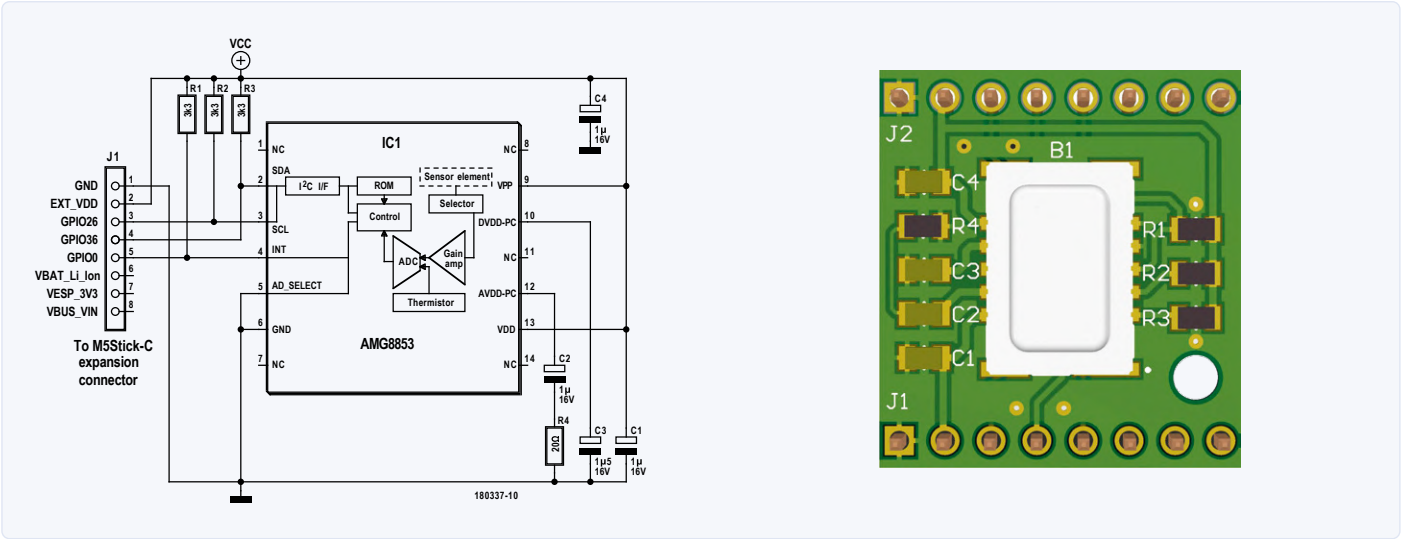


Bild 4. Innen- und Peripherieschaltung des AMG88xx-Sensors auf dem Breakout-Board.

ein wenig über die Innenschaltung des Sensors. Im Projektdownload [9] befinden sich Layouts der beiden Platinenlagen und der Bestückungsplan zum freien Download. Wer es sich ganz bequem machen möchte, kann das fertig bestückte Modul auch beim Autor erwerben.

M5StickC, die eierlegende Wollmilchsau

Jedes Smartphone ist WLAN-kompatibel und besitzt einen Touchscreen für Ein- und Ausgaben; es stellt so ein ideales Benutzerinterface für unsere MTheCam dar. Es bietet sich insofern an, per WLAN mit dem Sensor zu kommunizieren. Dafür braucht der Sensor aber einen smarten Datensender: ein SoC (System-on-chip) mit Controller und WLAN-Modul.

Die Wahl fiel auf den ESP32 von Espressif [4]. Dieses Modul besitzt alles, was unser Entwicklerherz begehrt, ist preisgünstig und energiesparend und darüber hinaus leicht über das Arduino-Ökosystem zu programmieren. An Elektronik ist neben Sensor und ESP-Modul nur noch eine (5-V-)Stromversorgung erforderlich, die über USB oder im Mobileinsatz über einen Akku erfolgen kann. Beim Blick in den Elektor-Shop fällt ein Produkt in leuchtend orangem Outfit auf, das neben dem ESP32 noch ein paar interessante Dinge drauf hat: Der M5StickC [5] passt perfekt!

Das mit etwa 50x26x14 mm^3 recht platzsparende Gerätchen hat so einiges im Kasten. Neben dem ESP-32Pico mit 4 MB Flash und 520 kB SRAM besitzt es ein 0,96"-Display mit 80x160 Pixeln, einen 6-Achsen-Bewegungssensor, eine Real-Time-Clock, eine Power-Management-Unit, eine rote LED, eine IR-LED für die Fernbedienung, ein MEMS-Mikrofon, einen 80-mAh-Akku, USB-C, Grove-Stecker (PWR, GND und zwei I²C-Ports) und eine 8er-Buchsenleiste mit drei Ports plus Spannungsversorgung. Das lässt selbst den abgebrühtesten Elektroniker aufhorchen!

Die kleine Größe, das Display und die mobile Akku-Stromversorgung lassen den M5StickC auch als schicke Smartwatch auftreten: Halterung und ein knalloranges Armband liegen sogar bei. Damit fällt man garantiert auf, wie **Bild 5** zeigt! Und wer braucht dann noch den langweiligen Armschmuck eines gewissen kalifornischen Apfelproduzenten, an den man noch nicht einmal etwas anstöpseln kann?

In diesem Projekt werden zunächst nur der ESP32, das Display und der Akku genutzt. All die übrigen Funktionen sind aber nicht verloren, sondern finden sicher in zukünftigen Anwendungen ein zu lösendes Problem.

Das Display des M5StickC ist recht klein, zeigt aber ein scharfes farbiges Bild. Gestalter von Benutzeroberflächen sind hier gefordert, die Informationen auf kleinster Fläche unterzubringen. Das Wärmebild passt gut, ein paar Zeilen für Messwerte gehen auch noch. Die Display-Library gibt viel Raum für graphische Spielereien, Animationen und Farbspektakel – Maker haben damit reichlich Gelegenheit, sich kreativ auszutoben.

Mit 80 mAh kann der LiIon-Akku den M5StickC immerhin knapp eine Stunde unter Volldampf versorgen. Geladen wird er über eine USB-C-Buchse mit 5 V / 500 mA, die das mitgelieferte Kabel aus einer normalen USB-A Schnittstelle holt. Ein kleiner Knopf an der Seite schaltet den M5StickC ein, ein Sechs-Sekunden-Druck wieder aus.

Kooperation

Das Breakout-Board des Sensors wird auf die Buchsenleiste für externe Erweiterungen gesteckt, mit einer gewinkelten Stiftleiste wie ein Rucksack oder mit gerader Stiftleiste für die Messung auf der Sichtachse parallel zur Gehäuselängskante, so dass sozusagen gerade „geschossen" wird (**Bild 6**). Mit einem 3D-Drucker kann man für beide Varianten leicht hübsche Gehäuse zaubern.

Der Erweiterungsanschluss des M5StickC besitzt acht Pole, von denen neben 5V-Out und GND die beiden I²C-Leitungen für die Daten- und Clock-Signale (SDA auf GPIO26 und SCL auf GPIO0) sowie das Interrupt-Signal INT (auf GPIO36) über die Stifte belegt sind. Die Anschlüsse BAT, 3V3 und 5V-In bleiben unverbunden. GPIO36 kann übrigens nur als Eingang genutzt werden; dieses Wissen spart so einiges an unnötiger Fehlersucherei.

Die Interrupt-Möglichkeit wird in der Software zunächst nicht berücksichtigt. Damit kann man den Sensor-Chip beispielsweise so programmieren, dass er bei Unter- oder Überschreiten einer Schwelle ein INT-Signal wirft – und das für jedes einzelne Pixel! Der Chip kann also ganz alleine eine bestimmte Position seines Blickfeldes überwachen und den ESP32 erst dann wecken, wenn ein Schwellwert überschritten wird. Das spart Energie! Den Überwachungsprozess kann natürlich später auch die App für das Smartphone selbst erledigen, indem das

ganze Bild ausgewertet wird.

Eine Frage der Software

Für die Firmware-Erstellung kann neben der proprietären Espressif-Entwicklungsumgebung auch das Arduino-Ökosystem genutzt werden, was sehr praktisch wegen des hocheffektiven globalen Community-Supports ist. Die Original-Arduino-IDE ist für den Start gut geeignet, für anspruchsvolle Aufgaben aber schnell überfordert. Der Autor empfiehlt daher den freien Visual-Studio-Code-Editor von Microsoft, der auch eine Arduino-Unterstützung als Extension mitbringt [6]. *M5Stick-C (esp32)* wird hier über den Board-Manager eingebunden. Die Firmware für MTheCam wurde vom Autor mit diesen Mitteln in C++ erstellt. Neben der Source-Datei *MTheCam_LT.ino* sind ein paar .h- und .cpp-Dateien (*Mxxx.cpp/h*) eingebunden, dazu noch die sehr empfehlenswerte *ArduinoJson*-Library (mindestens V6, V5 läuft hier nicht!) für die JSON-Behandlung [7]. Die Hardware-Spezialitäten erledigt die *M5StickC*-Bibliothek [8] ohne viel eigenes Zutun. Sie müssen über den Bibliotheksmanager (*F1 – Arduino: Library-Manager*) hinzugeladen werden.
Im Gegensatz zur erstklassigen Dokumentation der *ArduinoJson*-Lib lässt die Beschreibung der *M5StickC*-Bibliothek einiges zu wünschen übrig. Um sie richtig nutzen zu können, ist mühsames und sorgfältiges Lesen des Quellcodes nötig. Wenn einem so viel Verwirrendes widerfährt, das ist schon einen separaten Artikel wert!

Schauen wir uns nun die Softwareschnipsel für die einzelnen Funktionen in ihrer logischen Reihenfolge an!

Sensorabfrage

Die aktuellen Messwerte werden vom Sensor zehn Mal pro Sekunde in Ein-Byte-Registern bereitgestellt, die die Software laufend über die I²C-Schnittstelle *Wire* abfragt. Pro Pixel müssen (wegen der Auflösung von 12 bit) zwei Bytes abgefragt werden. Das macht *Wire* in `MTC_readReg()` so:

```
#define BUF_LENRX 128
int reg = 0x80;
byte _rxBuf[BUF_LENRX];
. . .
Wire.beginTransmission(devAddr); // Chip-Address @
   datasheet
Wire.write(reg); // 0x80 -> read 128bytes of 64
   pixels @12bit
Wire.endTransmission();
if (Wire.readTransmission(devAddr, rxBuf, BUF_
   LENRX) == I2C_ERROR_OK) { success = true; }
Wire.endTransmission();
```

Als Rückgabewert gibt `Wire.readTransmission()` unter Umständen Fehlercodes aus, die, falls das Auslesen nicht funktionieren sollte, bei der Fehlersuche nützlich sein könnten.
Für das Auslesen des Sensors und die Berechnung der Temperaturwerte ist der Abschnitt in **Listing 1** zuständig. Aus jeweils zwei `rxBuf[]`-Array-Einträgen lässt sich mit etwas Bitschieberei der Temperaturwert – hier als Integer in 100stel Grad Celsius – berechnen: Aus 21,35 °C wird also der Wert 2135. Der Sensor ist so initialisiert, dass er intern den gleitenden Durchschnitt zweier Bilder bildet und damit das doch etwas kräftige Rauschen reduziert. Es werden alle Register in einem Rutsch ausgelesen, damit garantiert alle Werte zu einem Messfenster gehören. Die weiteren Berechnungen von Pixel- und Chip-Temperatur

Bild 5. Der M5StickC als knallorange Smartwatch (Quelle: m5stack.com)

Bild 6. Gewinkelt oder gerade: zwei Befestigungsmöglichkeiten des Sensors am M5StickC.

(diese wird hier nicht verwendet) unterscheiden sich in einem Faktor (siehe Datenblatt [3]). Zur höheren Genauigkeit der Berechnung dient die Multiplikation mit 10000, die dann später durch 100 geteilt die 100stel Grad ergibt.

Berechnung der Farbwerte

Für die Farbdarstellung wird das HSL-Farbschema [10] (**Bild 7**) verwendet. In der M5Stick-C-Sammlung ist dafür eine entsprechende TFT-Bibliothek enthalten. Von den Parametern Farbwinkel (Hue, H), Sättigung (Saturation, S) und Hellwert (Lightness, L) wird nur der H-Parame-

Listing 1. Sensor auslesen und Temperatur berechnen.

```
. . .
 // read one frame with 64 temp-values à 12bit -> join 2 bytes to 1 int
 // result is int[] with calculated values -> _frame[]
 int  _frame[FRAMESIZE];          // filled by MTC_getFrame()
. . .
 int* MTC_getFrame()
  {
    byte _rxBuf[BUF_LENRX];
    byte dL[FRAME_SIZE];  // low byte
    byte dH[FRAME_SIZE];  // high byte
    int reg = 0x80;       // start with this reg for next 128 bytes to read
    . . .
    // sensor-read 128 bytes - low + high
    boolean ok = MTC_readReg(MTCADDRESS, reg, _rxBuf, 128);  // readings now in _rxBuf
    if (ok)
    {
        int k = 0;
        for (int i = 0; i < FRAME_SIZE; i++)
        {
            int index   = i;
            int d1 = _rxBuf[k++];                // low byte here
            int d2 = _rxBuf[k++];                // high byte next: includes sign!
            // sensor on PCB is upside down, start from the end
            index  = (FRAME_SIZE - i - 1);
            _frame[index] = MTC_calcTemp(d1, d2); // 2 bytes calc'd to one int
        }
     }
    return _frame;
 }
. . .
 // make one int from low/hight bytes d1/d2
 int MTC_calcTemp(uint8_t d1, uint8_t d2)        // return temp in deg Celsius * 100
  {
    int factor = (int) (0.25 * 10000);           // pixel-temp, for device-temp use 0.0625
    if (d2 & 0b00001000) d2 |= 0b11111000;       // handle sign-bit #4 in high-byte
    int valRaw = (d2 << 8) | d1;                 // merge to one int …
    int valCalc = (int)(valRaw * factor / 100);  // … with this factor
    return valCalc;
 }
. . .
```

Listing 2. Berechnung der Farbwerte.

```
#define MC_COLORWHEEL_LOW  315     // degrees on colorwheel
#define MC_COLORWHEEL_HIGH  15
#define MC_DIFFCW           60     // diff in degrees high to low
. . .
int MC_getColorHSL(int value, int vmin, int vmax, int vavg)
{
    // do not use 255 - 315 deg = 60 deg -> it's not clearly 'felt' as cold or hot
    // vmin/vmax for minimum spread of spectrum -> less color flicker if no spots seen
    if((vavg - vmin) < MC_TMIN_SPREAD2AVG)  { vmin = vavg-MC_TMIN_SPREAD2AVG; }
    if((vmax - vavg) < MC_TMIN_SPREAD2AVG)  { vmax = vavg+MC_TMIN_SPREAD2AVG; }
    // map range vmin -> 315, vmax -> 15 degree 'left turn' - exclude 315...15 = 60 deg
    int colInd = map(value, vmin, vmax, MC_COLORWHEEL_LOW, MC_COLORWHEEL_HIGH); // 315/15
    colInd = colInd - MC_DIFFCW;   // turn - 60: icecold->DEEPBLUE=255, hot->DARKRED=315
    if (colInd < 0) colInd += 360; // no neg. values!
    return colInd;
}
```

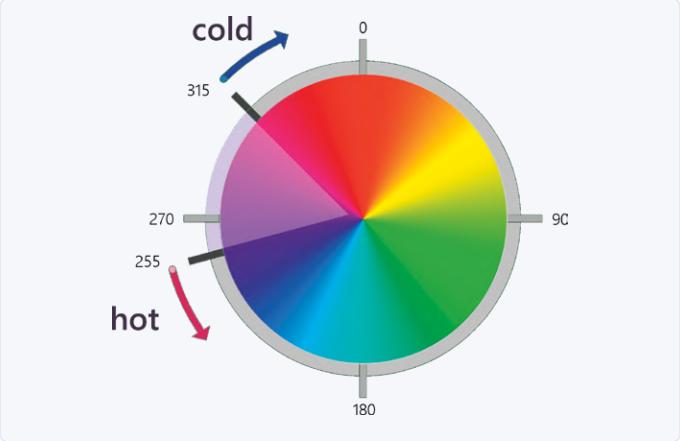


Bild 7. HSL-Modell: Farbrad und genutzter Bereich.

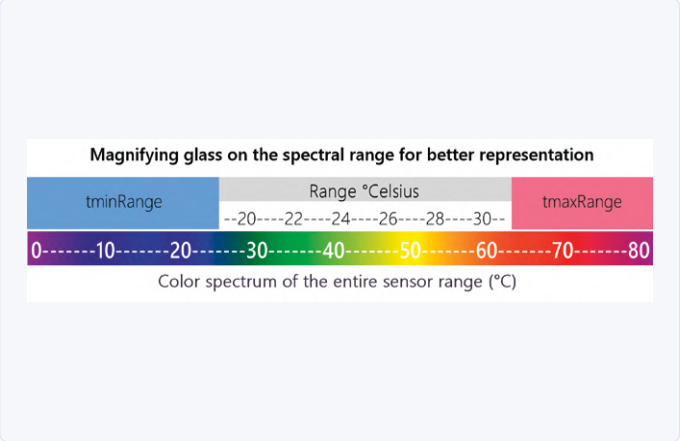


Bild 8. Der Messbereich wird über den gewünschten Ausschnitt des Spektrums gespannt.

ter genutzt, S und L bleiben unberücksichtigt (konstant). Zur späteren Darstellung im Display werden die HSL- in RGB-Werte umgerechnet. Da wir zur numerischen Anzeige den Bereich 0...80 °C in 8000 Messwerte (100-stel Kelvin) auflösen, müssen diese zunächst auf die möglichen Farbwerte 0...359 normiert werden. Nicht ohne Grund assoziieren wir die Farbe blau mit kalt und rot mit heiß, daher scheint der Kreisabschnitt beginnend mit 255 Grad (tMin = blaugrün) links herum bis 315 Grad (tMax = dunkelrot) optimal für die Darstellung zu sein. Da wir violett in der Regel nicht mit einem Temperaturempfinden in Verbindung setzen, werden die violetten Farbwerte nicht genutzt. Von den möglichen 360 Farbwerten werden also 300 verwendet und 60 nicht.

Die praktische `map()`-Funktion in **Listing 2** passt die Temperaturwerte in den 300-Grad-Winkel des Farbraums gegen den Uhrzeigersinn drehend von 315 Grad bis 15 Grad ein. Dann dreht sie alles nochmals um 60 Grad nach links (Farbwinkel -60), um schließlich die Eckwerte 255 und 315 zu erreichen. Um bei gleichmäßig warmen Flächen ohne Spots das recht kräftige Farbrauschen zu mindern, wird ein Mindestabstand von tMin beziehungsweise tMax zur Durchschnittstemperatur von je 500 Einheiten festgelegt, das heißt, das Anzeigespektrum (die „Lupe") besitzt eine gewisse Mindestbreite von 1000 Einheiten entsprechend 10 Grad, um nicht jedes Rauschpixelchen unnötig als völlig andere Farbe aufleuchten zu lassen.

Damit steht der Farbbalken mit dem nutzbaren Spektralbereich fest: Die minimale Temperatur entspricht 255 Grad das Farbrades, die maximale 315 Grad, nicht zu vergessen entgegen dem Uhrzeigersinn!

Darstellung auf dem Display

Für eine Art Lupenfunktion, die den tatsächlichen Anzeigebereich in diesem möglichen Spektralbereich verdichtet, dienen die Werte `tminRange` und `tmaxRange` als Parameter der Farbermittlungsfunktion (**Bild 8**). Das ist nur eine äußerst einfache Möglichkeit, eine ansprechende Farbdarstellung zu erzeugen. Einen speziell konstruierten Farbverlauf hinzuzuziehen und die Farblupe noch cleverer zu berechnen wäre es zum Beispiel, die Range-Werte adaptiv über mehrere Messungen zu ermitteln, um Farbsprünge zu vermeiden, oder die Dynamik für Hotspot und Hintergrund getrennt zu behandeln, um Spots deutlicher hervorzuheben. Machine-Learning Experten dürften großen Spaß daran haben, KI zur Mustererkennung, Rauschminderung und Datenanalyse einzusetzen.

Bild 9. Das Temperaturfeld auf dem ersten Prototypen.

Das TFT-Display des M5StickC zeigt, wie in **Bild 9** zu sehen, in der oberen Hälfte das Original-Thermobild mit dem verwendeten Farbspektrum, darunter die Messwerte für die minimale und maximale gemessene Temperatur aller Pixel des Bildes (tMin und tMax) sowie der daraus berechnete Durchschnittswert (tAvg). Diese Angaben sind in °C * 100 angegeben, also ohne Kommata.

WLAN und Webserver

Bisher funktioniert alles lokal auf dem M5StickC und ohne WLAN. Sollen aber die Messwerte auf einem entfernten mobilen Gerät angezeigt oder gar weiterverarbeitet werden, muss sich der M5StickC mit einem WLAN verbinden und als Webserver eine Webseite mit den Messwerten zur Verfügung stellen können.

In der Arduino-Software schaffen die Bibliotheken `WiFi.h` und `WiFiClient.h` einen solchen Webserver, der nur noch mit den

Listing 3. WLAN und Webserver.

```
. . .
    const char *ssid     = "YourSSIDHere";      // change…
    const char *password = "YourPasswordHere";  // … to your WLAN
    WebServer webServer(80);                    // listening on port 80 = standard

    WiFi.mode(WIFI_STA);                        // switch to ESP station mode
    WiFi.begin(ssid, password);                 // look for known WLAN & try to connect
    while (WiFi.status() != WL_CONNECTED)       // wait for connection
    {
       delay(500);                              // 500ms delay
       Serial.print(".");                       // still alive
    }
    Serial.println("\n[Setup] WIFI connected!");
    // what to do if browser request is coming in
    webServer.on("/", MW_index);                // call MW_index() for / or /index.html
    webServer.on("/index.html", MW_index);
    webServer.on("/frame", MW_frameJS);         // /frame - output as JSON-Data
    webServer.onNotFound(MW_notFound);
    webServer.begin();                          // webserver starts here
. . .
// Webserver callback
void MW_index()     // send HTML-Doc from literally defined PROGMEM var
 {
    webServer.send_P(200, "text/html", _uidoc); // send_P: use PROGMEM-Var
 }
. . .
// main loop
void loop()
 {
                    // give webserver a chance to handle requests. No delay() in LOOP!
      webServer.handleClient();
     . . .
 }
. . .
```

Zugangsdaten für Ihr Netzwerk (`ssid` und `password`) parametriert werden muss. Nach einem Reset beziehungsweise PowerOn zeigt das Display den Versuch an, sich dem Netzwerk anzuschließen. Bei Erfolg wird die erhaltene IP-Adresse angezeigt. Wenn nicht: Sind die Zugangsdaten korrekt? Ist ein Access-Point ausreichend nah? Die Reichweite ist übrigens überraschend gut für das kleine Gerätchen – sie kann gut mit der von Smartphones mithalten.

Der Webserver horcht dann unter seiner vom Router zugewiesenen lokalen IP-Adresse auf die Requests `/`, `/index.html` und `/frame`, also zum Beispiel `http://192.168.10.1/`.

Nach dem Aufbau der WLAN-Verbindung wird dem Webserver mitgeteilt, was zu tun ist, wenn er einen bestimmten Request vom Browser-Client erhält. Es wird dann die zugeordnete Funktion aufgerufen – also etwa bei `/` (identisch mit `/index.html`) die Funktion `MW_index()`, die das Dokument zum Client ausliefert:

`webServer.send_P(200, “text/html”, _uidoc);`

`Send_P` wird verwendet, weil das HTML-Dokument `_uidoc` im Flashspeicher als `PROGMEM` gehalten wird, um RAM-Speicher freizuhalten. Damit kann MTheCam auf jedem Smartphone bunte Wärmebilder anzeigen. WLAN- und Webserver-Programmteile sind im **Listing 3** zu sehen und dokumentiert.

HTML und JSON

Die Requests `/` und `/index.html` erzeugen eine Webseite mit 8x8 Farbflächen und in die Flächen eingeblendete Temperaturwerte. Außerdem werden die minimale (tmin), die maximale (tmax) und die Durchschnittstemperatur (tavg) einer Messung angegeben. Jeder Farbton entspricht einer Temperatur, wobei der Farbverlauf innerhalb des MIN- und MAX-Wertbereiches (`tminrange/tmaxrange`) dynamisch eingepasst wird. Die Schaltfläche *image* schaltet das Thermokino auf ein „hochauflösendes“ Falschfarbenbild interpolierter Werte um, der Knopf *values* wieder zurück. Jede Sekunde wird eine neue Messung angezeigt, dass Anzeigeintervall kann über einen Slider geändert werden. Die Checkbox *grid* legt ein Raster über das Bild.

Der Aufruf `/frame` liefert die Messwerte als JSON-Datensatz zur weiteren Verarbeitung durch andere Programme – bitte mit jeweils mindestens 200 ms Pause, damit der Sensor-Chip neu messen kann, was etwa 100 ms dauert. Ein solcher Datensatz könnte notfalls mit String-Basteleien erzeugt werden, aber die sehr praktische Bibliothek `ArduinoJson.h` macht das viel schlanker und kann auch noch solche Daten einlesen. Das Ergebnis sieht dann beispielsweise so aus:

```
{"device": "MTheCam","ver": "1.0","rowsize":
   8,"batvolt": 3.39,
"tavg": 2317,"tmin": 2050,"tmax": 2750,"tminrange":
   1817,"tmaxrange": 2817,
"frame": [2075, 2050, 2100, 2275, 2600, 2475, 2100,
   2050, , ..... ]} // 64 values °C * 100 // + 2
   arrays with 'HSL'-colors of 8*8 and 32*32 tiles.
```

Listing 4 zeigt die interessanten Stellen des HTML-Textes. Das Dokument mit CSS und HTML-Tags wird als fixer String definiert und

Listing 4. HTML-Dokument mit Messwerten erstellen.

```
. . .
/* Canvas-Area 400 px with 8*8/32*32 tiles à 50/12 px */
<div class="row">
      <div class="column">
        <canvas id="canvas"></canvas>    /* size see resizeCanvas() */
      </div>
</div>
. . .
/* resize canvas for value or image mode */
function resizeCanvas () {
 . . .
     canvasSize = rectSize * gridSize    /* gridSize: 8 or 32 @rectSize: 12 or 50 */
     canvas.width  = canvasSize
     canvas.height = canvasSize
. . .
/* update image,  rate = 200…3000 ms */
function refresh (rate = 1000) {
 . . .
refresher = setTimeout(function () {
        getData()
        }, rate)
. . .
/* request temperature-frame from MTC via /frame - call  */
function getData () {
     fetch(url)
        .then(response => response.json())
        .then(jsonData => {
          if (mode == 1) {
            // tempvalues, colors (interpolated) in framecolinter array
            paint(jsonData.frame, jsonData.framecolinter);
          } else {
            // tempvalues, colors (original) in framecol array
            paint(jsonData.frame, jsonData.framecol)
          }
          infoText(jsonData)             /* show min/max etc. values */
        }).catch(err => {
          errorText.innerHTML = 'Error: $'
      }).then(function () {
        refresh()                        /* start timer for next ride */
     })
}
. . .
/* paint tiles on canvas */
const ctx  = canvas.getContext("2d")

function paint(values, colors) {
  . . .
   ctx.clearRect(0, 0, canvas.width, canvas.height);
   colors.foreach(color, index) ….  {
     . . .   /* set x/y of next tile: x+=rectSize … */ . . .
     /* paint tile */
     ctx.beginPath()
     ctx.rect(x, y, rectSize, rectSize)           /* paint square */
     ctx.fillStyle = 'hsl($, 100%, 50%)'   /* set color of square */
     if (grid) { ctx.stroke() }                    /* show grid optional */
     ctx.fill()                                   /* do fill */
     ctx.closePath()
     if (mode == 0) {                             /* values as text over */
         // show tempvalues @mode=0 -> lowres 8*8 only
         drawText(values[index], x + 7, y + rectSize / 2)
       }  }
 . . .
```

an den Browser-Client gesendet, wobei sich das Thermo-Bild aus einem CANVAS mit 8*8- (values) bzw. 32*32- (image) Kacheln aufbaut. Eine Script-Abteilung ruft danach timergesteuert die Sensordaten ab (*/frame*), die dann per JSON-Packet geliefert werden. Javascript und JSON verstehen sich gut, so dass die Nutzdaten sehr einfach aus dem Paket herausgezogen werden können: *jsonData.frame* sowie *jsonData.framecol* stellen direkt die Arrays mit den Werten und Farben bereit, aus denen die originalen Temperatur- sowie die interpolierten Farbverlaufswerte gezogen, die zugehörigen Kachelflächen passend eingefärbt und die Wärme-Werte als Zahlen darübergelegt. Das Update-Intervall für diese Anzeige kann von 200 ms bis 3 s eingestellt werden.

Gute Basis für weitere Projekte

In einer erweiterten Firmwareversion soll ein eigener Access-Point über ESP32 betrieben werden, der über sein eigenes WLAN Daten liefert, solange keine Verbindung zu einem bestehenden Funk-Netzwerk verfügbar ist und die komfortable Eingabe der Zugangsdaten zu anderen WLANs ermöglicht. Die Farbdarstellung und Farblupe für eine optimale Darstellung werden anpassbar sein, Filter-Mathematik reduziert das Rauschen und Flackern. Eine Android-App kann mehrere MTheCam-Geräte verwalten, einrichten und gleichzeitig live anzeigen. Alarmzustände und -Meldungen können definiert und ausgelöst werden, wobei Node-Red über MQTT-Server die Datennutzung recht bequem macht. Es lohnt sich also, ab und an die Projektseite zu beobachten [9]. Die Grundlage für clevere praktische Anwendungen ist gelegt, Ideen sind gefragt! Besonderer Dank geht an Daniel Zelosko für die Entwicklung des Prototyps.

180337-01

PASSENDE PRODUKTE

M5StickC
www.elektor.de/m5stack-m5stickc-esp32-pico-mini-iot-development-board

Sie haben Fragen oder Kommentare?
Haben Sie Fragen oder Anmerkungen zu diesem Artikel? Senden Sie eine E-Mail an den Autor (mtc@micom.de) oder an die Elektor-Redaktion (redaktion@elektor.com).

Ein Beitrag von:
Schaltung, Software und Text: **Olaf Mertens (Micom)**
Redaktion: **Rolf Gerstendorf**
Layout: **Giel Dols**

WEBLINKS

[1] **Pyrometer**: https://de.wikipedia.org/wiki/Pyrometer
[2] **Interpolation in Fotos**: https://de.wikipedia.org/wiki/Interpolation_(Fotografie)
[3] **AMG88x**: https://eu.industrial.panasonic.com/products/sensors-optical-devices/sensors-automotive-and-industrial-applications/infrared-array
[4] **ESP-Doku**: www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
[5] **M5StickC**: https://www.elektor.de/m5stack-m5stickc-esp32-pico-mini-iot-development-board
[6] **VisualStudio**: https://code.visualstudio.com/
[7] **Arduino-JSON** : https://arduinojson.org/
[8] **M5StickC-Bibliothek**: https://github.com/m5stack/M5StickC
[9] **Projektseite des Autors**: https://www.micom.de/lab/mthecam
[10] **HSV-Farbraum**: https://de.wikipedia.org/wiki/HSV-Farbraum

Lötstation WE 1010
von Weller

Von **Harry Baggen**
(Niederlande)

Die Auswahl an Lötkolben und Lötstationen ist riesig, aber vielen Elektronikern wird sofort die Marke Weller in den Sinn kommen. Die Lötstationen von Weller sind beliebt, aber die Preise recht ordentlich. In dieser Besprechung geht es mit dem Modell WE 1010 um die billigste Lötstation der Profi-Serie. Mit knapp über 150 € ist sie auch für das Hobby-Labor interessant.

Meine erste Weller-Lötstation war vom Typ WTCP – die mit den bekannten Magnastat-Lötkolben. Die Temperatur wurde durch einen Magneten in der Spitze gesteuert, der ab einer bestimmten Temperatur seinen Magnetismus verlor (Curie-Effekt). Spitzen konnten für verschiedene Temperaturen gekauft werden, so dass man eine bestimmte Temperatur durch Wechsel der Spitze „einstellen“ konnte. Meine Lötstation (**Bild 1**) ist mehr als 40 Jahre alt, funktioniert aber immer noch prächtig. Jetzt, da ich zu Hause ein bisschen mehr Elektronik bastele, dachte ich, es sei an der Zeit, eine modernere Lötstation zu kaufen, mit einem Display, das die Temperatur anzeigt, und einem Bolzen, der etwas handlicher ist als die alte Magnastat.

Heutzutage kann man sehr schöne Lötstationen für wenig Geld kaufen, die hauptsächlich in China hergestellt werden. Aber ich habe mich zunächst bei Weller umgesehen, ob es auch dort etwas für akzeptable Summen zu kaufen gibt. Neben preiswerteren Hobby-Modellen hat Weller in seiner Professional-Serie eine Basisversion für etwas über 150 € im Programm. Das lag auch in meinem Budget. Die Vorteile eines Weller-Geräts im Vergleich zu chinesischen Fabrikaten: Zuverlässigkeit (hoffentlich), eine gescheite Halterung und ein sehr flexibles und hitzebeständiges Kabel zum Lötkolben. Ich hoffe nämlich, mit meiner neuen Lötstation noch einige Freude zu haben.

Bild 1. Bei meiner alten Weller-Lötstation steht „10-1978“ als Produktionsdatum auf dem Typenschild.

Bild 2. Das Display zeigt die Temperatur der Spitze und darunter etwas kleiner den Sollwert.

Bild 3. Die Spitzen lassen sich leicht auswechseln.

Gekauft!

Also bestellte ich eine WE 1010. Das Paket besteht aus einer Basisstation mit LCD und einem 70-W-Lötkolben. Obwohl es sich um einen Kolben mit indirekter Temperaturregelung handelt (Heizung und Sensor stecken nicht in der Spitze, sondern dahinter), halte ich das nicht für sehr wichtig (dazu später mehr). Der Lieferumfang besteht aus einer Basisstation mit passendem Netzkabel, einem Lötkolben WEP 70, einem ziemlich einfachen Ständer aus einem Formsockel mit Spiralhalter sowie einem in Letzterem abzulegenden Schwamm. Es sieht alles ziemlich einfach aus, doch dafür ordentlich gemacht. Die Station verfügt über einen Ein-/Ausschalter, einen LCD-Bildschirm und einige Steuertasten.

Leider ist das Display nicht hintergrundbeleuchtet. Doch die Anzeige ist auch bei wenig Umgebungslicht sehr klar und gut lesbar. Der Netzschalter hat auch keine eingebaute Beleuchtung. Mann sieht also nur dadurch ob die Station eingeschaltet ist, weil etwas auf dem Display zu sehen ist und am Netzschalter eine rote Linie sichtbar wird.

Die Basisstation ist ziemlich schwer (fast 2 kg) und fühlt sich solide an. Das 1,5 m lange Kabel (mit Silikonmantel) zum Lötkolben ist sehr geschmeidig, und der Bolzen liegt mit seinem Überzug aus einer Art gepresstem Schaumstoff gut in der Hand. Das Kabel des Lötkolbens ist am anderen Ende mit einem fünfpoligen Stecker versehen, der in die Buchse an der Basisstation gesteckt und mit einer Drehung verriegelt wird. Der Metallteil des Kolbens ist mit der geerdeten Steckdose des Netzkabels verbunden. Es gibt keine separate oder schaltbare Erdverbindung.

Bedienung und Betrieb

Die Bedienung der Lötstation wurde sehr einfach gehalten. Mit einer Auf- und Abwärtstaste kann man die Temperatur höher oder niedriger einstellen. Bei Auslieferung ist sie auf 350 °C voreingestellt. Außerdem gibt es eine Menütaste, mit der man folgende Punkte erreicht: Standby-Zeit (die Zeit, nach der der Lötkolben auf eine niedrigere Temperatur zurückgeschaltet wird; max. 99 Min.), Offset (zur Temperaturkorrektur, nur sinnvoll, wenn man die Temperatur der Spitze messen kann; max. ±40 °C), Umschaltung zwischen °C und °F und schließlich die Möglichkeit, einen Sperrcode einzustellen.

Das Display zeigt die aktuelle Temperatur der Lötspitze in großen Ziffern an und darunter den eingestellten Zielwert etwas kleiner (**Bild 2**). Zusätzlich zum Sollwert erscheint ein Heizsymbol, wenn der Lotkolben gerade aufheizt. Auch wenn die Standby-Funktion aktiv ist oder wenn die Sperrfunktion verwendet wird, werden entsprechende Symbole angezeigt.

Aufgrund seines Gewichts steht der Lötkolbenhalter gut auf dem Tisch. Er hat eine Reihe von Löchern zur Aufnahme von Ersatzspitzen. Um die Spitze sauber zu wischen, gibt es den bekannten Schwamm. Es gibt aber keinen Platz für Messing-Wolle. Wer lieber so ein Bündel Metallwolle zum Abwischen der Spitze verwendet, muss sich einen extra Behälter mit Metallwolle kaufen.

Der Lötkolben selbst (**Bild 3**) ist ziemlich schlank und das flexible Kabel schön lang. Es ist darüber hinaus so flexibel, dass man es beim Löten kaum bemerkt. Die Lötspitzen können ganz einfach durch Abschrauben der Metallhülse des Vorderteils über eine Kunststoffmutter gewechselt werden. Gerade weil dieser Kolben wie bereits erwähnt indirekt beheizt wird, besteht sein großer Vorteil darin, dass Ersatzspitzen sehr preiswert sind – sogar die originalen von Weller. Sie sind fast überall erhältlich, und ich habe bereits einige mit anderen Spitzenformen gekauft (**Bild 4**).

Anfeuern!

Nach dem Einschalten dauerte es etwa 35 Sekunden, bis die Lötspitze eine Temperatur von 350 °C erreicht hat, Die für bleifreies Lot geeigneten 380 °C benötigen etwa 5 s mehr. Das Löten mit dem Lötkolben WE P70 klappt reibungslos. Bei durchschnittlichen Bauteilen und Kupfer-Pads gibt es keinerlei Probleme. Bei großen Lötflächen merkt man, dass die Temperatur etwas absinkt und die Spitze einige Sekunden braucht, um sich wieder auf den Sollwert zu erwärmen (ein aktiver Lötkolben reagiert in einem solchen Fall viel schneller), aber das dauert dank seiner 70 W Heizleistung nur eine kurze Weile. In solchen Fällen ist es ratsam, eine kurze Lötspitze mit einer großen, angeschrägten Spitze zu verwenden, damit die

Bild 4. Mehr als 15 verschiedene Spitzen sind verfügbar. Hier die von mir angeschafften Spitzen.

Bild 5. Wenn der Kolben eine bestimmte Zeit lang nicht verwendet wird, geht er auf Standby und die Spitze bleibt auf 180 °C.

Wärme gut übertragen werden kann. Die mitgelieferte Spitze ist hierfür eher zu schmal und besser für kleine Bauteile geeignet. Für die Standby-Funktion wird kein Bewegungsmelder im Kolben verwendet, sondern die Station überwacht, wie viel Wärme dem Kolben zugeführt wird. Wenn die Heizleistung für eine bestimmte eingestellte (Standby-)Zeit gering bleibt, wird die Temperatur automatisch auf 180 °C reduziert (**Bild 5**).

Die Spitzentemperatur wird automatisch wieder hochgefahren, wenn der Spitze viel Wärme entzogen wird, indem man sie z.B. über den nassen Schwamm streift. Direkter

geht es, wenn man kurz eine der Tasten an der Basisstation betätigt. Für mich sind längere Heizdauern besser. 30 Minuten reichen, oder man schaltet den Standby gleich ganz aus. Ich habe bisher kein Verzundern der Spitzen bemerken können. Falls das doch jemals passiert: Die Spitzen sind preiswert.

Fazit

Mit der WE 1010 als Nachfolgerin meiner alten Weller-Lötstation bin ich sehr zufrieden. Der Lötkolben liegt bequem in der Hand und lötet gut, die Einstellmöglichkeiten der Basisstation sind begrenzt, aber für den normalen Gebrauch ausreichend. Was mir fehlt, ist eine bessere optische Einschaltanzeige. So muss man auf das Display schauen oder den roten Strich des Netzschalters detektieren. Hätte ein beleuchteter Netzschalter oder eine extra LED wirklich so viel mehr gekostet? Das wäre mir viel Wert gewesen. Und diese Standby-Funktion? Sie ist nicht optimal, aber man kann damit leben. Trotz dieser kleinen Mängel würde ich die WE 1010 als Qualitätsprodukt auf jeden Fall wieder kaufen, wenn ich eine Lötstation in dieser Preisklasse suchen würde.

200572-03

Sie haben Fragen oder Kommentare?
Sie haben Fragen oder Anmerkungen zu diesem Artikel? Dann senden Sie eine E-Mail an den Autor, über die Redaktion von Elektor (redactie@elektor.com).

Ein Beitrag von
Text: **Harry Baggen**
Fotos: **Patrick Wielders**
Redaktion: **Eric Bogers**
Layout: **Giel Dols**

PASSENDE PRODUKTE

› **Weller WE 1010 Digitale Lötstation**
www.elektor.de/weller-we-1010-digital-soldering-station-70-w

VIRTUAL ELECTRONICA FAST FORWARD 2020

DIE GEWINNER

electronica fast forward
the startup platform

powered by elektor

Von **Clemens Valens** (Elektor)

electronica Fast Forward, der Start-up-Wettbewerb powered by Elektor, wurde im Jahr 2020 virtuell. Von Januar bis Oktober 2020 präsentierten junge, auf Elektronik spezialisierte Unternehmen aus der ganzen Welt ihre Elevator Pitches, Geschäftspläne und Produkte. Wir gratulieren allen talentierten Teilnehmern! Hier sind die Gewinner.

In den letzten Jahren hat Elektor den *Fast Forward-Start-up*-Wettbewerb organisiert, der zeitgleich mit den Messen *electronica* und *productronica* stattfand. Aber im Jahr 2020 war alles ein bisschen anders, und wir wissen alle, warum. Anstatt die Veranstaltung abzusagen, haben Elektor und die electronica (https://electronica.de/) den *electronica Fast Forward* (e-ffwd) online durchgeführt. Unternehmen aus Deutschland, den Vereinigten Staaten, Ungarn und Frankreich (um nur einige zu nennen) stellten ihre Elevator Pitches, Geschäftspläne und Produkte vor.
Obwohl der e-ffwd-Wettbewerb für jedes Start-up-Unternehmen offen war, wurden nicht alle zur Endrunde zugelassen. Nur die Teilnehmer, die einen Businessplan, ein Firmenprofil und ein kurzes Video als Ersatz für die Elevator-Speeches vor Publikum und Juroren einreichten, wurden der Jury vorgestellt. Statt Live-Publikum beurteilten die Besucher in einer öffentlichen Abstimmung die Start-up-Profile durch die Vergabe von Sternen.

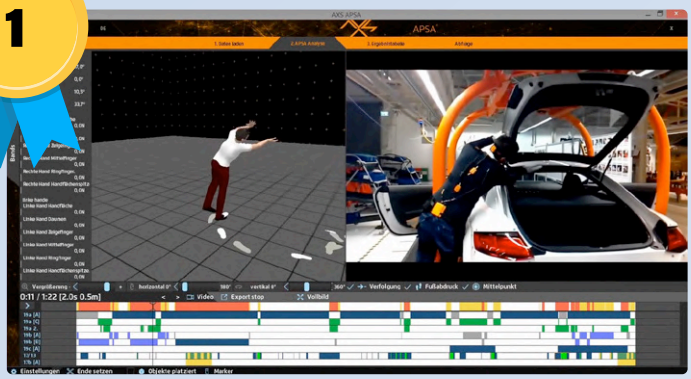
Gewinner des Wettbewerbs electronica Fast Forward

Nach Prüfung der verschiedenen Start-ups und ihrer Technologien wählte die offizielle e-ffwd-Jury unter dem Vorsitz von Professor Rik De Doncker von der RWTH Aachen (siehe Kasten) zusammen mit Elektor-Vertretern aus dem redaktionellen Bereich die drei ersten Gewinner aufgrund ihres hohen kommerziellen Potenzials und der Qualität des Teams aus. Zwei wichtige Kriterien für ein Start-up, um zu einem erfolgreichen Unternehmen zu wachsen. Herzlichen Glückwunsch an AXS Motionsystem [1], Micropack3D [2] und e-Fásli [3]!

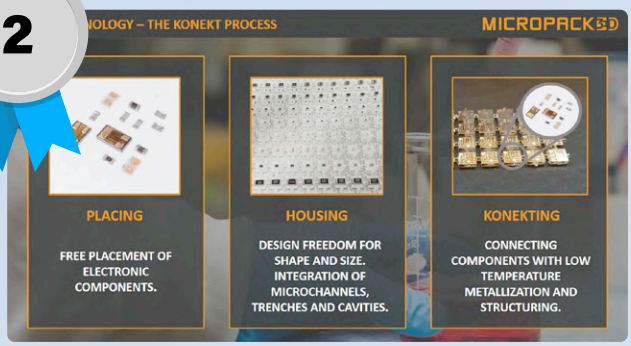
Erster Preis: AXS Motionsystem Ltd (Ungarn)
AXS Motionsystem Ltd. wurde 2014 gegründet und entwickelt ein ergonomisches Expertensystem zur Bewertung und Qualifizierung von Arbeitsplätzen. AXS Motionsystem erfasst digital die Bewegungen und die Handgriffe eines Arbeiters während des Arbeitsablaufs. Basierend auf den aufgezeichneten Daten erfolgt die Bewertung nahezu vollautomatisch nach verschiedenen Ergonomiekriterien. Für seine Leistung erhielt AXS Motionsystem ein Marketingbudget von 75.000 € von Elektor und einen Starter-Messestand auf der electronica 2022.

Zweiter Preis: Micropack-3D (Deutschland)
Micropack-3D ist ein vor kurzem gegründetes Start-up-Unternehmen, das adaptives Electronic Packaging entwickelt, das heißt, die Unterbringung und Kontaktierung von integrierten Schaltungen in kundenspezifischen Gehäusen. Ziel ist es, neue Generationen hoch angepasster und spezialisierter Elektronik zu ermöglichen und die kostbare Entwicklungszeit zu verkürzen. Das Unternehmen hat als zweiten Preis ein Marketingbudget von 50.000 € von Elektor gewonnen.

Dritter Preis: e-Fásli Ltd (Ungarn)
e-Fásli ist das erste selbstlernende Wärmetherapiegerät. Es kann am Körper befestigt werden und stundenlang eine Temperatur zwischen 25 °C und 45 °C aufrechterhalten. Patienten- und Krankheitsdaten werden von einer mobilen App erfasst und dann anony-



Der erste Preisträger AXS Motionsystem entwickelt ein ergonomisches Expertensystem zur Bewertung und Qualifizierung von Arbeitsplätzen.



Das zweitplatzierte Start-up Micropack3D bietet innovative Verpackungsoptionen für elektronische Schaltungen.

Eine Wiedergabe des vierten Prototyps von e-Fásli's selbstlernendem Wärmetherapiegerät.

misiert an den Server gesendet, wo sie verarbeitet werden, um die optimalen und personalisierten therapeutischen Parameter zu bestimmen. Dann wird das Gerät über ein WLAN entsprechend gesteuert. Für diese Leistung erhielt e-Fásli Ltd. ein Marketingbudget von 25.000 € von Elektor.

Bis zum nächsten Jahr?

Der virtuelle Wettbewerb electronica Fast Forward 2020 [4] war dank seines breiten Teilnehmerfeldes auf hohem Niveau ein Erfolg. Besuchen Sie die e-ffwd-Seite (www.elektormagazine.com/effwd-2020), um mehr über alle Finalisten zu erfahren. Die Technologien:

- Kabelloses Laden
- Edge Computing
- Elektrische Fahrzeuge
- Energy Harvesting
- Lösungen für MEMS-Spiegel und Laserscanning
- Aktor-Technologien
- Steuerung und Überwachung in Echtzeit
- und noch viele mehr...

Hoffentlich werden die Dinge bald wieder zur Normalität zurückkehren, so dass wir uns bei der nächsten Veranstaltung sehen können. Auge in Auge!

200584-02

WEBLINKS

[1] **AXS Motionsystem Ltd**: www.elektormagazine.com/labs/axs-motionsystem-ltd

[2] **MicroPack3D**:www.elektormagazine.com/labs/micropack3d

[3] **e-Fásli Ltd**: www.elektormagazine.com/labs/e-fasli-ltd

[4] **Teilnehmerprofile e-ffwd 2020**: www.elektormagazine.de/effwd-2020

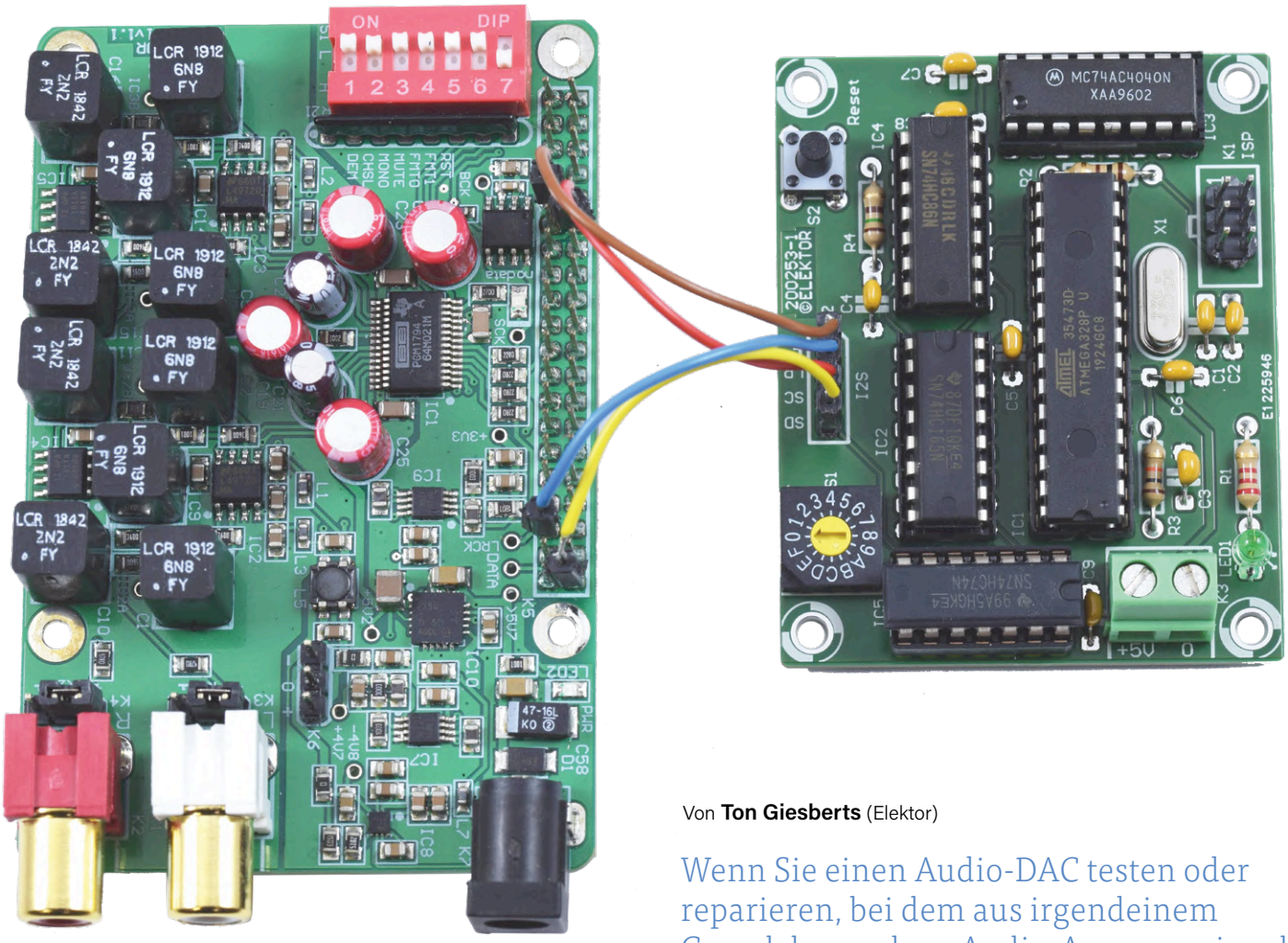
Professor Rik De Doncker

Die virtuelle Jury des Fast Forward Award wurde von Professor Rik W. De Doncker von der RWTH Aachen geleitet.

Rik W. De Doncker (M'87 SM'99 F'01) erhielt 1981 sein Diplom in Elektromaschinenbau und 1986 seinen Doktortitel in Elektrotechnik von der belgischen KU Löwen. Im Jahr 1987 wurde er zum außerordentlichen Gastprofessor an der University of Wisconsin in Madison, USA, ernannt. Im Jahr 1988 trat er in das GE Corporate Research and Development Center, Schenectady, New York, ein. Im November 1994 kam er als Vice President Technology zur Silicon Power Corporation (ehemals GE-SPCO), wo er den weltweit ersten statischen Mittelspannungs-Transferschalter entwickelte. Seit Oktober 1996 ist er Professor an der RWTH Aachen, wo er das Institut für Stromrichtertechnik und elektrische Antriebe (ISEA) leitet. Im Oktober 2006 wurde er zum Direktor des E.ON Energy Research Center an der RWTH Aachen ernannt, wo er auch das Institute for Power Generation and Storage Systems (PGS) gründete. Er ist Direktor des Clusters Nachhaltige Energie des RWTH-CAMPUS und leitet den BMBF-Forschungscampus Flexible Elektrische Netze (FEN) der Bundesregierung. Er hält einen Doktortitel honoris causa der TU Riga, Lettland.

I²S-Testsignalgenerator mit AVR-Mikrocontroller

Digitales Sinus-Testsignal mit 32 Bit Auflösung, fs von 192 kHz und einstellbarem Pegel von 0 bis -110 dB



Von **Ton Giesberts** (Elektor)

Wenn Sie einen Audio-DAC testen oder reparieren, bei dem aus irgendeinem Grund das analoge Audio-Ausgangssignal fehlt oder verzerrt ist und es unsicher ist, ob die Signalquelle (Software und/oder Hardware) das Problem darstellt oder die DAC-Schaltung selbst, kann dieses kleine Projekt die Antwort liefern. Sein hochgenaues digitales Sinus-Testsignal ist auch ideal, um die analoge Leistung des DAC zu messen.

INFOS ZUM PROJEKT

Tags
Digitales Audiosignal, Raspberry Pi, DAC, I²S

Level
Einsteiger – Fortgeschrittene – Experte

Zeit
Etwa 4 Stunden

Werkzeuge
Lötwerkzeug (für bedrahtete Bauteile), AVRISP

Kosten
Ungefähr 15 €

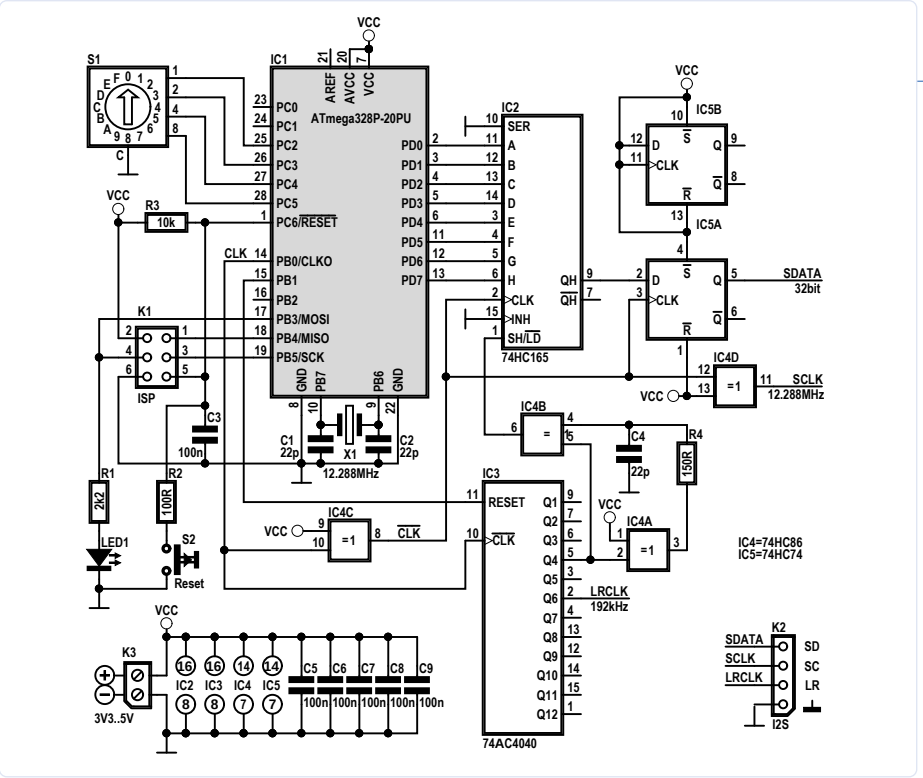


Bild 1. Schaltung des I²S-Signalgenerators.

Seit seiner Einführung im Jahr 1986 ist der I²S-Bus (Inter-Integrated Circuit Sound) der De-facto-Standard für die Übertragung serieller digitaler Audiosignale. Während der Entwicklung und des Tests unseres *Audio-DAC für den Raspberry Pi* [1] kamen wir auf die Idee, eine Schaltung zu entwerfen, die ein I²S-Signal erzeugt, um den DAC zu testen, ohne den RPi als Signalquelle anschließen zu müssen. Diese Schaltung kann natürlich auch zum Testen anderer Audio-DACs mit I²S-Eingängen verwendet werden, sofern diese eine Abtastfrequenz von 192 kHz unterstützen und 24-Bit- oder 32-Bit-Audiodaten verarbeiten können.

Entwicklungsoptionen

Eine Option, einen I²S-Testsignalgenerator zu entwerfen, wäre es, einen analogen Sinussignal-Generator an einen 24-Bit-ADC mit I²S-Ausgängen anzuschließen. Um jedoch zu prüfen, ob die analogen Ausgangssignale des DACs tatsächlich fehlerfrei sind und um korrekte Verzerrungsmessungen durchführen zu können, darf keine minderwertige Analogquelle und kein schlechter ADC eingesetzt werden. Der Sinus im I²S-Signal muss perfekt sein!
Alternativ kann zur Erzeugung des I²S-Signals ein Mikrocontroller verwendet werden, der eine Tabelle mit genau berechneten 32-Bit-Abtastwerten verwendet, um die Qualität der Audiodaten zu gewährleisten. So lässt sich ein Signal erzeugen, das perfekt für Verzerrungsmessungen geeignet ist, in diesem Fall eine Sinuswelle mit einer Frequenz von 1 kHz und einer Abtastrate von 192 kHz.

Es wäre zwar naheliegend, für diese Aufgabe einen Mikrocontroller zu verwenden, der I²S unterstützt, aber wir haben den enorm populären ATmega328P ausgewählt, der I²S nicht unterstützt. Es war zwar eine ziemliche Herausforderung, einen digitalen Sinusgenerator mit I²S-Ausgang mit diesem Mikrocontroller (und einiger zusätzlicher Hardware) zu bauen, aber dieses Projekt zeigt, dass es möglich ist. Die Firmware des ATmega wurde in BASCOM-AVR entwickelt.

Zusätzliche Hardware

Ziel ist es, ein mit 32 Bit aufgelöstes I²S-Signal bei einer Abtastrate von 192 kHz zu erzeugen, was nahe an der maximalen Abtastfrequenz des PCM1794A liegt, der in unserem RPi-Audio-DAC verwendet wird. Der serielle Takt (SCK oder SCLK) muss dazu 12,288 MHz (2 Kanäle * 192 kHz * 32 Bit) betragen. Da aber die maximale Taktfrequenz des Mikrocontrollers 20 MHz beträgt, besteht die einzige Möglichkeit, serielle Daten (SD oder SDATA) schneller auszugeben, darin, ein externes Parallel-in-Serial-out-Schieberegister zu verwenden und den Takt des Mikrocontrollers auch zum Takten des Schieberegisters zu verwenden. PB0 muss deshalb beim Setzen der Fuses des ATmega328 als CLKO (Clock

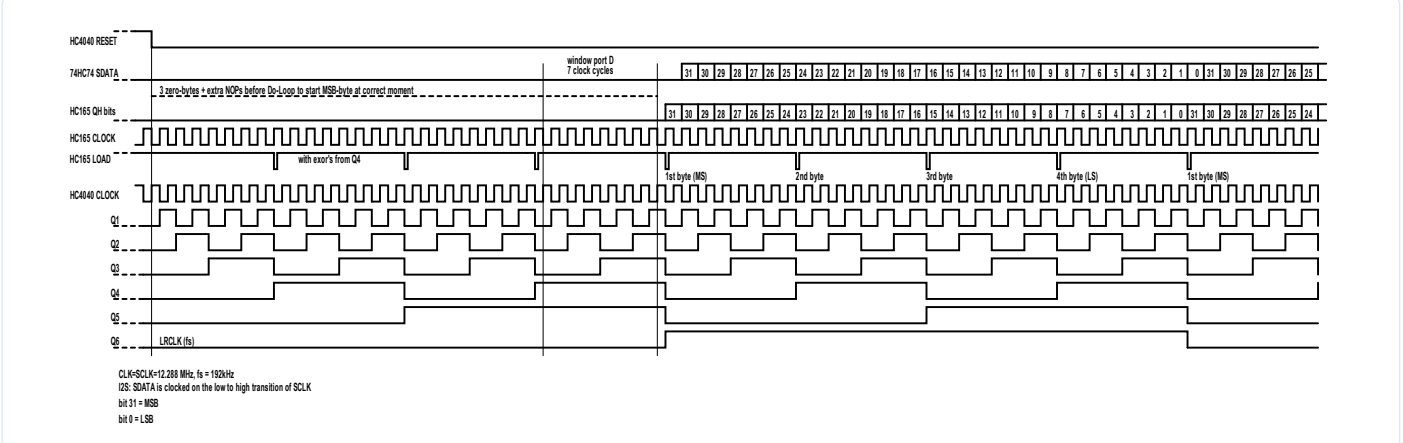


Bild 2. Timing des I²S-Signals.

STÜCKLISTE

Widerstände:

R1 = 2k2
R2 = 100 Ω
R3 = 10 k
R4 = 150 Ω

Kondensatoren:

C1,C2,C4 = 22 p, C0G/NP0, Rastermaß 5 mm
C3,C5...C9 = 100 n, X7R, Rastermaß 5 mm

Halbleiter:

LED1 = LED, grün, 3 mm
IC1 = ATmega328P-PU, 20 MHz, DIP28
IC2 = 74HC165
IC3 = 74AC4040 (keine HC-Logik !!!)
IC4 = 74HC86
IC5 = 74HC74

Außerdem:

K1 = 2x3-polige Stiftleiste, vertikal, Raster 2,54 mm
K2 = 1x4-polige Stiftleiste, vertikal, Raster 2,54 mm
K3 = 1x2-polige Platinenklemmen, Raster 5,08 mm, 630 V
S1 = Drehkodierschalter, hexadezimal, Real Code, THT (z.B. Nidec Copal Electronics SD-1010)
S2 = Tastschalter, 24 V, 50 mA, 6x6 mm
X1 = Quarz 12,288 MHz, C-Last 18 p, 10 ppm, HC-49S

Platine 200253-1 v1.1

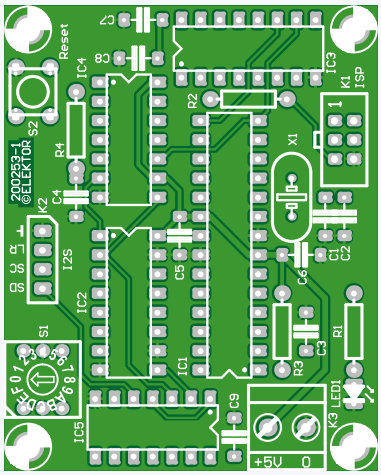


Bild 3. Das Platinenlayout.

Out) konfiguriert werden.
Der vollständige Schaltplan des Signalgenerators ist in **Bild 1** zu sehen. Neben dem Schieberegister IC2 werden ein zwölfstufiger Binärzähler IC3, ein Flipflop IC5A und einige Exklusiv-ODER-Gatter IC4 hinzugefügt, um den Mikrocontroller von anderen (Timing-) Aufgaben, als zur Ausgabe der Bytes der Abtastwerte nötig sind, zu befreien.
Alle anderen Software-Aufgaben, etwa die Berechnung der 32-Bit-Sinuswelle und die Erstellung eines Arrays von Sample-Bytes, müssen vor dem Start der Hauptschleife des Programms erledigt sein. Die 32-Bit-Abtastwerte werden in vier Bytes aufgeteilt. Das bedeutet, dass die Größe des Arrays 4 * 192 = 368 Bytes betragen muss. Linker und rechter Kanal verwenden dasselbe Signal, so dass jede Gruppe von vier Bytes pro Abtastwert wiederholt werden muss. Die Anzahl der Zeilen, die dieser Teil der Hauptschleife beansprucht, kann berechnet werden:

2 Kanäle × 4 Bytes × 192 Abtastwerte × 4 Zeilen - 3 = 6141 Programmzeilen

Nach dem letzten Sample-Byte in der Schleife werden die drei NOPs weggelassen, da der Neustart der Schleife ohnehin drei Taktzyklen dauert, was „-3" in dieser Formel erklärt.

Die Hardware im Detail

Das Timing-Diagramm in **Bild 2** kann bei der Auswahl der externen Komponenten hilfreich sein: Der I²S-Bus besteht aus drei Signalen: Die seriellen Daten (SDATA) werden mit der steigenden Flanke (Low-to-High) des seriellen Taktsignals (SCLK) getaktet, die Word-Select-Leitung (WS oder LRCLK) gibt Auskunft über den Audiokanal (0 für links, 1 für rechts). Die LRCLK-Frequenz ist gleich der Abtastfrequenz des digitalen Audiosignals (192 kHz) und kann aus dem seriellen Takt abgeleitet werden. Sowohl die serielle Daten- als auch die LRCLK-Leitung müssen bei der fallenden Flanke (High-to-Low) des seriellen Taktsignals wechseln. HC-Logik ist prinzipiell schnell genug, um hier als externe Komponenten verwendet zu werden, obwohl die Durchlaufzeit (propagation delay) der Gatter von 12 ns fast 15 % einer Taktperiode des Mikrocontrollers entspricht.
Für das korrekte Timing der I²S-Signale wird an einer Stelle aber ein deutlich schnellerer 74**AC**4040 verwendet. Dieser Zähler schreitet mit der fallenden Flanke des Takteingangs voran. Er verfügt außerdem über einen Master-Reset (Pin 11), mit dem er mit dem Mikrocontroller synchronisiert werden kann. Das sechste Flip-Flop (Q6, Pin 2) teilt 12,288 MHz durch 64 und gibt genau 192 kHz aus, die Frequenz, die wir für die LRCLK-Leitung benötigen.
Beim Start des Programms wird der Reset des Zählers aktiviert (high) und erst kurz vor dem Beginn der Hauptschleife deaktiviert. In dieser etwa eine Viertelsekunde währenden Wartezeit nach dem Einschalten kann das Array in aller Ruhe aufgebaut werden. Dieses Reset-Signal synchronisiert die Daten am Ausgang des Schieberegisters und der LRCLK-Leitung. Die ersten drei Samples mit dem Wert 0 und ein paar NOPs sorgen dafür, dass das erste Byte zum richtigen Zeitpunkt am Ausgang von SDATA liegt. Es gibt ein Fenster von sieben Taktzyklen, in dem das erste MS-Byte in das Schieberegister eingetaktet wird. Jedes Byte an Port D wird nach acht Taktzyklen durch das nächste ersetzt. Mit anderen Worten, der Zeitpunkt, zu dem sich das erste MS-Byte an Port D relativ zum Load-Impuls des Schieberegisters ändern kann, darf um sieben Taktzyklen variieren, wie auch das Timing-Diagramm angibt. Die Daten am Ausgang von Port D ändern sich nur wenige Nanosekunden nach der steigenden Flanke von CLKO (PB0).
Zur Umwandlung der parallelen Controllerdaten in ein serielles Signal wird ein 8-Bit-Schieberegister 74HC165 (IC2) verwendet. Das „Laden" der parallelen Daten und das Weiterschieben im Register erfolgt nach Maßgabe des SH/$\overline{\text{LD}}$-Anschlusses (Pin 1). Der Ladeimpuls ($\overline{\text{LD}}$, activ low) wird vom Zählerausgang Q4 abgeleitet. Das dort erscheinende Signal wird zwei Exclusive-OR-Gattern des 74HC86 (IC4) zugeführt. Es trifft an einem Eingang von IC4B (Pin 5) unmittelbar und am anderen Eingang (Pin 4) durch IC4A nicht nur invertiert, sondern auch dank der Durchlaufzeit des Gatters und zusätzlich durch R4/C5 um einige Nanosekunden verzögert ein. Aufgrund der Exklusiv-ODER-Eigenschaft führt also jede Änderung von Q4 zu einem kurzen aktiven Low-Impuls am Ausgang (Pin 6) von IC4B. Der Impuls ist lang genug, um die neuen Daten in das Schieberegister zu laden, aber kurz genug, um *vor* der steigenden Flanke des Taktes wieder inaktiv zu sein. Um die Daten von Port D zum richtigen Zeitpunkt in das Schieberegister zu laden, muss der Impuls unmittelbar nach der steigenden Taktflanke (Pin 1) aktiv sein. Dies bedeutet, dass der Takt des Schieberegisters invertiert werden muss, was von IC4C erledigt wird.
Der HC165 besitzt weiterhin Eingänge für einen Taktgeber (Pin 2) und Inhibit (Pin 15, Freigabe bei low), die die gleiche Funktionalität haben, da sie intern mit einem ODER-Gatter verknüpft sind. Je nach Bauteilplatzierung kann durch Vertauschen der Anschlüsse das Routing der Platine vereinfacht werden. Die Daten werden zur steigenden Flanke des Taktgebers verschoben. Der serielle Eingang (Pin 10) wird nicht benutzt und ist mit Masse verbunden.
Das höchstwertige Bit der seriellen Daten des I²S-Signals befindet sich leider eine Taktperi-

ode *hinter* der Änderung des LRCLK-Wechsels, so dass ein zusätzliches D-Flip-Flop eingesetzt wird. Es handelt sich um IC5, ein duales D-Flip-Flop mit Set und Reset und Triggerung zur positiven Flanke, kurz gesagt, ein 74HC74. Dadurch wird auch das SDATA-Signal verzögert. Um dies zu kompensieren, wird das von IC4C invertierte Taktsignal von IC4D nochmals invertiert, so dass der serielle Takt mit den seriellen Daten übereinstimmt.

Einstellung des Ausgangspegels

Anstatt nur eine Sinuswelle von 1 kHz mit einem festen Pegel zu erzeugen, werden vier Eingänge von Port C (mit internen Pull-ups) mit einem hexadezimal codierten Drehschalter (SD-1010) verbunden, um den Ausgangspegel einzustellen. Diese Funktion kann zur Überprüfung der Linearität des DACs verwendet werden. Ein Vierwege-DIP-Schalter hätte auch verwendet werden können, aber das Ändern des Pegels ist mit dem Drehschalter komfortabler und ähnlich wie bei einem analogen Lautstärkesteller. Es gibt Drehcodierschalter der Typen „Real Code" und „Complementary Code". In diesem Projekt wird eine Real-Code-Variante verwendet, aber Sie können auch den anderen Typ einsetzen, wenn Sie die Software entsprechend ändern. In Stellung „0" sind alle vier Schalter offen.
Um den Ausgangspegel für das digitale Audiosignal zu ändern, muss der Mikrocontroller mit Taster S2 zurückgesetzt werden, damit die Software alle Werte im Sample-Array neu berechnen kann. In einer vom Skalierungsfaktor U (Variablentyp Double) abhängigen Select-Case-Anweisung werden für jede Pegeleinstellung die richtigen Werte gesetzt. Alle Werte für alle Skalierungsfaktoren sind im Programm als konstante Werte definiert, nicht nur, um zusätzliche Berechnungen zu vermeiden, sondern auch, weil Berechnungen mit der LOG-Funktion von BASCOM im Programm nicht genau genug sind.

Berechnung des Sinus

Das Programm berechnet die Sinus-Sampletabelle unmittelbar nach dem Einschalten oder einem Reset unter Berücksichtigung der Pegeleinstellung durch den Drehschalter S1. Der Zweck dieses Projekts bestand ja nicht nur darin, „irgendeinen" I²S-Signalgenerator zu bauen, sondern auch ein perfektes 1-kHz-Sinus-Testsignal mit 32-Bit-Genauigkeit zu erzeugen. Zuerst wurde die BASCOM-AVR-Anweisung für `SIN(x)` in den Berechnungen verwendet, aber das war nicht genau genug, wie die folgenden Beispiele zeigen:

```
DIM Pi,A,X As Single
Pi = 3.14159265358979323846264433
X = Pi / 2
A = SIN(X)
Print „Sin(Pi/2) = „ ; A
```

Aus diesem Stück Code ergibt sich für Sin(Pi/2) = 0,9999999332, wobei das Ergebnis genau 1 sein sollte. Und für X = Pi/6 ergibt sich 0,49999993796, wobei das Ergebnis genau 0,5 sein sollte. Das ist für die Berechnung einer extrem genauen Sinuskurve einfach nicht gut genug! Die einzige Möglichkeit besteht darin, die Sinuskurve mit dem Tylor-Polynom für SIN(X) mit ausreichenden Termen zu berechnen.

```
SIN(X) = X-(X^3)/3!+(X^5)/5!-
         (X^7)/7!+(X^9)/9!-(X^11)/11!+(X^1
         3)/13!-(X^15)/15!
```

In BASCOM-AVR können Berechnungen nur auf zwei Operanden durchgeführt werden, daher wird, wie im Quellcode dieses Projekts ersichtlich, das Polynom in vielen Zeilen des Quellcodes berechnet. Um die Angelegenheit zu beschleunigen, werden die Fakultätsberechnungen vermieden und stattdessen Konstanten verwendet (F3 = 6 bis F15 = 1307674368000). Alle Variablen in den Berechnungen sind vom Typ *Double*, vorzeichenbehaftete 64-Bit-Binärzahlen (8 Bytes, 5 x 10^-324 bis 3,4 x 10^308). Mit dem Taylor-Polynom erhalten wir die folgenden viel genaueren Ergebnisse:

Sin(Pi/6) = 500E-3
Sin(Pi/2) = 999,9999999999993977E-3

Um das Ergebnis der Berechnung von `SIN(X)` in vier Bytes aufzuteilen, muss es zunächst in eine Variable vom Typ *Long*, vorzeichenbehaftete 32-Bit-Binärzahlen (-2147483648 bis 2147483647) umgewandelt werden. Das Ergebnis der Berechnung von `SIN(X)` liegt im Bereich von -1 bis 1. Wenn wir wollen, dass die 32-Bit-Daten ganzzahlig sind, muss der Wert von `SIN(X)` mit (2^32)/2 - 1 multipliziert werden:

```
SINX = SINX * U
```
mit U = 2147483647

Die Umwandlung in *Long* kann in nur einer Anweisung erfolgen, eine Long-Variable erhält den Wert einer Double-Variablen. Nun steht das Ergebnis der Berechnung als vorzeichenbehafteter 32-Bit-Wert zur Verfügung:

```
SINXlong = SINX
```

Um diesen Wert in vier Bytes aufzuteilen, müssen lediglich die Bits von `SINXlong` verschoben und in vier Variablen vom Typ *Byte* gespeichert werden. Ein Unterprogramm namens `SampleX` führt all diese Berechnungen für jeden beliebigen Wert von X durch. Aber die Berechnung ist nur für X = -π/2 bis X = +π/2 wirklich genau. Zuerst wird also die Sinuskurve von -π/2 bis +π/2 berechnet, wobei 97 Samples mit jeweils vier Bytes genommen werden. Der Rest des Arrays wird vervollständigt, indem die Elemente des ersten Arrays in Gruppen von vier Bytes gespiegelt werden, wobei 95 Samples genommen werden. Sample 193 ist das gleiche wie das erste des Arrays (vier Bytes). Die `Do`-Schleife beginnt dann dort erneut.
Es wäre auch möglich gewesen, ein Array nur mit Sinus-Samples von -π/2 bis +π/2 zu erstellen und die richtigen Array-Elemente in der Hauptschleife auszuwählen, um die Daten für eine volle Sinusperiode an Port D zu vervollständigen. Aber anstelle eines simplen Sinus könnte auch ein komplexeres, unsymmetrisches Signal für andere Zwecke interessant sein. Und wenn ein vollständiges Array verwendet wird, ist die Hauptschleife leichter zu lesen und zu modifizieren.

WEBLINKS

[1] **Audio-DAC für Raspberry Pi**: www.elektormagazine.de/labs/audio-dac-for-rpi-networked-audio-player-using-volumio
[2] **Download Gerber-Dateien und Software**: www.elektormagazine.de/200253-03
[3] **Projektseite auf Elektor Labs**: www.elektormagazine.com/labs/32-bit-i2s-sine-wave-generator-200253

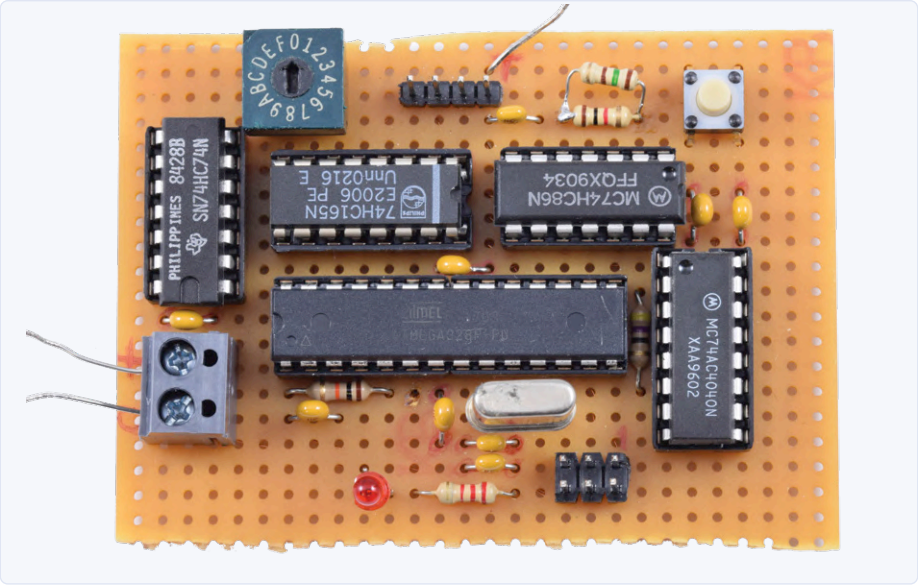


Bild 4. Der Prototyp des Generators auf einer Lochrasterplatine.

Durch die tätige Mithilfe des Schieberegisters hat der Mikrocontroller acht Taktzyklen Zeit, um jedes Byte zu verarbeiten. In BASCOM-AVR nimmt der Befehl `PORTD = A(n)` in der `Do`-Schleife, wobei `A(n)` ein Element eines Byte-Arrays ist, nur fünf Taktzyklen in Anspruch. Damit jedes Byte wie gewünscht acht Taktzyklen benötigt, müssen nach jedem Byte drei `NOP`s hinzugefügt werden. Es dauert drei Taktzyklen, bis die `Do`-Schleife wieder startet, was perfekt ist, um eine volle Periode von 192 Samples einer Sinuswelle von 1 kHz zu erzeugen. Das Einzige, was die Hauptschleife des Programms also aktiv tun muss, ist, die Bytes des Arrays an Port D auszugeben.

Aufbau der Hardware

Wir haben die Platine in **Bild 3** für dieses Projekt entworfen; das Layout und die Gerber-Dateien stehen zum Download [2] zur Verfügung. Alternativ können Sie ein Stück Lochrasterplatine verwenden, wie es in der ersten Prototyping-Phase auch gemacht wurde (**Bild 4**).

Wenn Sie keinen Lieferanten für den 74AC4040 finden können, ist es möglich, die Verbindung zwischen Pin 13 und Pin 14 von IC4 zu kappen. Durch Verbinden von Pin 13 mit Masse mit einem kurzen Drähtchen kann auch ein 74HC4040 verwendet werden, aber das Timing ist bei weitem nicht optimal. Diese Modifikation funktioniert mit unserem RPi-DAC, könnte jedoch Probleme mit anderen zu testenden DACs hervorrufen. Die Anschlüsse vom K2 des Generators zum DAC sollten so kurz wie möglich gehalten werden.
Die Versorgungsspannung dieser Schaltung muss mit der Spannung des zu testenden DACs übereinstimmen. Bei V_{CC} = 3,3 V liegt die Stromaufnahme knapp über 20 mA.

Erweiterungen in der Software...

Das Programm nutzt 77 % des Flash-Speicherplatzes. Für die Berechnung der Sinuswelle werden, wie oben beschrieben, etwa 10 % benötigt. Es bleibt also noch etwas Platz im Programmspeicher, um den Code zu erweitern und zusätzliche Funktionen hinzuzufügen. Beachten Sie, dass die kostenlose Demoversion von BASCOM-AVR aufgrund des begrenzten Programmspeichers nicht zum Kompilieren der Software verwendet werden kann; Sie müssen eine registrierte Version besitzen oder erwerben, wenn Sie das Programm ändern möchten.
Der Software-Download für dieses Projekt [2] enthält die HEX-Datei für die Original-Firmware, so dass dieses Projekt ohne eine lizenzierte BASCOM-AVR-Version gebaut werden kann, lediglich ein AVR-ISP-Programmieradapter und Programmiersoftware wie Atmel AVR-Studio oder AVRDUDE bringen Ihren ATmega128 zum Laufen.

...oder in der Hardware

Es sind noch einige I/O-Pins des Mikrocontrollers unbenutzt, die zur Steuerung zusätzlicher Funktionen wie Änderung der Wellenform oder Frequenz implementiert werden können. Natürlich müssen Sie auch die Software ändern, um solche Funktionen zu unterstützen. Die Mikrocontroller-Pins PB2, PC0 und PC1 können problemlos verwendet werden, während PB4 und PB5 dedizierte Pins für die In-System-Programmierung (ISP) sind, und es muss darauf geachtet werden, dass zusätzliche angeschlossene Bauteile an dieser Programmierschnittstelle nicht stören. LED1 wurde hinzugefügt, um anzuzeigen, dass der Prozessor mit der Berechnung der Audio-Sample-Daten beschäftigt ist, aber da dies in nur einer Viertelsekunde erledigt ist, kann PB3 auch für andere Funktionen verwendet werden. In diesem Fall kann man R1 direkt an die Stromversorgung anschließen, um weiterhin eine Einschaltkontrolle zu behalten.

Falls und wenn Sie diesem I²S-Signalgenerator eine neue Funktionalität hinzufügen, wenn Sie weitere Anmerkungen oder Vorschläge zu diesem Projekt haben, vergessen Sie nicht, sie uns und anderen Lesern auf der Elektor Labs-Seite [3] mitzuteilen!

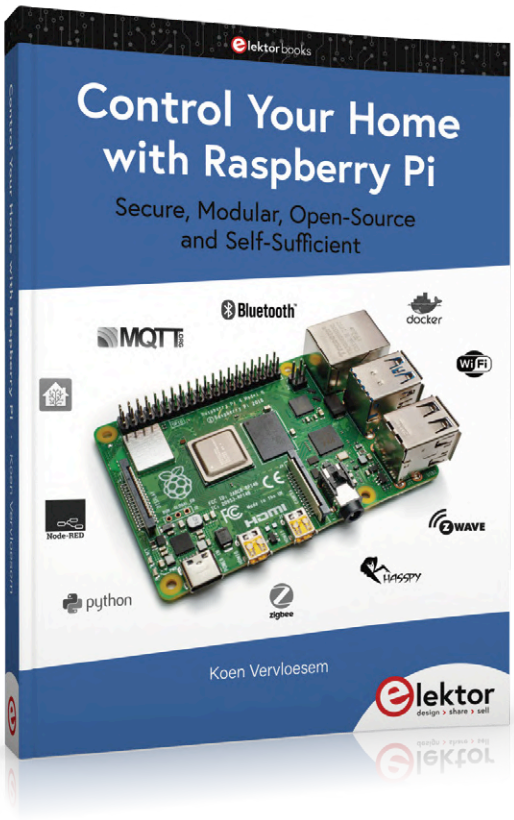
200253-03

Sie haben Fragen oder Kommentare?
Haben Sie Fragen oder Anmerkungen zu diesem Artikel? Senden Sie eine E-Mail an den Autor unter ton.giesberts@elektor.com.

Ein Beitrag von
Idee, Entwurf, Text: **Ton Giesberts**
Schaltbilder und Illustrationen: **Ton Giesberts, Patrick Wielders**
Redaktion: **Luc Lemmens**
Übersetzung: **Rolf Gerstendorf**
Layout: **Giel Dols**

Steuern Sie Ihr Zuhause
mit dem Raspberry Pi

Der RPi und das ISM-Band 433,92 MHz



Von **Koen Vervloesem** (Belgien)

Im *Elektor-Bücherregal* steht seit kurzer Zeit das Buch *Control Your Home with Raspberry Pi* von Koen Vervloesem, das wir Ihnen in Auszügen vorstellen möchten. Koen erläutert, wie Sie Ihren RPi in einen leistungsfähigen Empfänger und Decoder für die meisten, wenn nicht sogar für alle Fernsteuerungen und Sensor-Signalerfassungen verwandeln können, die auf der bekanntesten und am weitest verbreiteten ISM-Frequenz 433,92 MHz basieren.

In der europäischen DIY-Community läuten bei der Frequenz 433,92 Hz alle (Alarm?-) Glocken. Viele billige drahtlose Geräte wie Garagentoröffner, Klimasensoren und Türklingeln nutzen nämlich diese Frequenz. Und die Hardware, die mit solchen Geräten kommunizieren kann, ist ebenso preiswert.
Zunächst eine Warnung: Der Nachteil dieser Geräte ist, dass die meisten eine unverschlüsselte oder nur mit schwachen und/oder proprietären Algorithmen gesicherte Funkkommunikation verwenden. Aus Sicherheitsgründen nutze ich deshalb nur 433,92-MHz-Temperatursensoren, einen in fast jedem Raum meines Hauses und auch draußen. Die Temperatur in meinem Schlafzimmer dürfte für Außenstehende wohl kaum interessant sein und ich würde 433,92-MHz-Geräte nicht für kritischere Aufgaben verwenden. Doch es sind so viele dieser Geräte verfügbar und sie sind so billig, dass man sie einfach nicht ignorieren kann.
In diesem Kapitel zeige ich Ihnen, wie Sie die Messwerte dieser drahtlosen Temperatursensoren lesen und wie Sie sie an Ihren MQTT-Broker zur weiteren Integration in Ihr Hausautomatisierungssystem weiterleiten können.
In diesem Buch spreche ich von 433,92 MHz, aber je nachdem, wo Sie wohnen, müssen Sie diese durch eine andere Frequenz ersetzen. In Nord- und Südamerika zum Beispiel ist die entsprechende Frequenz 915 MHz. Achten Sie nur darauf, dass Sie die richtigen Geräte für Ihr Land kaufen!

433,92-MHz-Protokolle

Geräte, die die Frequenz 433,92 MHz nutzen, arbeiten im nicht lizenzierten industriellen, wissenschaftlichen und medizinischen ISM-Frequenzband. Aber die Frequenz ist eine Sache, das Protokoll, das sie verwenden, eine andere. Es gibt kein Standardprotokoll für diese Frequenz, es ist kein Z-Wave oder Zigbee. Viele Protokolle dieser Geräte lassen sich jedoch im Reverse-Engineering-Verfahren entschlüsseln, so dass Sie mit ihnen kommunizieren können, wenn Sie einen Transceiver für das Frequenzband um 433,92 MHz und die richtige Software zum Dekodieren und/oder Kodieren des Protokolls besitzen.
Einige interessante Geräte sind:

- Temperatur-, Feuchtigkeits- und Klimasensoren von Alecto, Cresta, La Crosse und Oregon Scientific
- Tür-/Fenstersensoren mit Hallsensoren
- Schalter und Dimmer von Energenie, KlikAanKlikUit und LightwaveRF
- Türklingeln von Byron und Chacon

Bei AliExpress und Banggood finden Sie auch viele noch billigere Geräte, die die gleichen Protokolle unterstützen. Und es gibt sogar kleine Platinen wie den STX882-Sender, die Sie an einen Mikrocontroller oder ein Arduino-Board anschließen können, um Ihre

Bild 1. Für rund 5 € finden Sie einen Temperatur- und Feuchtigkeitssensor von DANIU, der seine Werte über 433,92 MHz überträgt und auf einem übersichtlichen Display anzeigt.

Bild 2. Der RTL-SDR dekodiert eine Vielzahl von Funksignalen, darunter auch Klimasensoren, die auf 433,92 MHz senden.

Bild 3. Mit der Stativhalterung, der Dipolfassung und den Teleskopantennen aus dem RTL-SDR-Kit haben Sie alles, was Sie brauchen, um Messungen von allen 433,92-MHz-Sensoren zu empfangen.

eigenen drahtlosen Sensorplatinen zu erstellen.
In diesem Kapitel will ich mich auf die erste Art von Geräten konzentrieren, Temperatur- und Feuchtigkeitssensoren. Diese gibt es für weniger als 10 € (**Bild 1**) bei den genannten chinesischen Online-Warenhäusern. Ihre Funkreichweite ist recht gut: Ich kann Sensoren in meinem ganzen Haus auslesen, auch in meinem Kühl- und Gefrierschrank und sogar auf meiner Terrasse im Freien.

Anforderungen an die Hardware

Damit Ihr Raspberry Pi Sensormesswerte über das ISM-Frequenzband empfangen kann, benötigen Sie einen Empfänger und eine Antenne.

Empfänger

Ein beliebter Empfängertyp für 433,92-MHz-Projekte ist ein DVB-Dongle mit einem Herz aus RTL2832. Ja, das haben Sie richtig gelesen, DVB wie in Digital Video Broadcasting. Wie sich herausstellt, kann der RTL2832-Chip von Realtek, der in vielen dieser Dongles steckt, weit mehr als nur digitale Videosignale dekodieren. Mit der richtigen Software können Sie damit sogar ein echtes softwaredefiniertes Radio (SDR) bauen!
Wenn Sie also einen alten DVB-Dongle in Ihrer Restekiste finden, stehen die Chancen gut, dass Sie mit ihm Signale von Ihren Klimasensoren empfangen können. Ansonsten ist der RTL-SDR [1] eine gute Wahl (**Bild 2**). Sie finden Varianten dieses Sticks für 25 € und in einem Kit mit Antenne und anderem Zubehör für 40…45 € [2]. Sie können so etwas Nützliches sogar noch billiger finden: Ich habe über 7-€-DVB-Dongles bei AliExpress gelesen, die perfekt für diesen Zweck funktionieren sollten, aber ich habe noch keine Erfahrung mit ihnen.

Antenne

Als Nächstes benötigen Sie eine gute Antenne. Es gibt ganze Bücher über Antennentheorie und ich will mich hier nicht in dieses riesige Thema ausbreiten, weil ich halt kein Antennenspezialist bin.
Eine Sache, die Sie für die Wahl Ihrer Antenne kennen sollten, ist deren Länge, die von der Wellenlänge des Signals abhängt. Die Wellenlänge ist gleich der Geschwindigkeit (in m/s) geteilt durch die Frequenz (in Hz) und wird in Metern (m) gemessen. Lassen Sie uns die Mathematik für die Kommunikation bei 433,92 MHz erledigen. In frischer Luft ist die Geschwindigkeit der Welle praktisch die Lichtgeschwindigkeit. Also beträgt die Wellenlänge:

299.792.458 m/s / 433.920.000 Hz = 0,69 m.

Die volle Wellenlänge beträgt also 69 cm, die Halbwellenlänge 34,5 cm und die Viertelwellenlänge 17,25 cm. Dies sind die theoretisch optimalen Längen für eine Antenne zum Empfang von 433,92-MHz-Signalen. In der Praxis beeinflussen aber verschiedene Faktoren die Eigenschaften der Antenne, einschließlich der Positionierung, und es gibt die Faustregel, von dieser theoretischen Länge 5 % abzuziehen.
Auch hier handelt es sich nicht um ein Antennentheoriebuch, daher werde ich nicht auf die verschiedenen Arten von Antennen eingehen. Außerdem ist es für das Auslesen von Sensorwerten in Ihrem Haus gar nicht so wichtig, welche Qualität Ihre Antenne hat. Sie könnten damit experimentieren, aber sie dürfte ohne weiteres Zutun funktionieren, wenn Sie eine „433-MHz-Antenne“ für ein paar Euro im Internet kaufen. Wenn Sie kein Risiko eingehen möchten, verwenden Sie die Antenne, die in einem Bausatz mit dem RTL-SDR enthalten ist. Der offizielle Antennenbausatz (**Bild 3**) enthält Teleskop-Dipolantennen, die Sie von 5 cm bis 1 m ausziehen können und die die optimale Wellenlänge für die Frequenz 433,92 MHz abdecken [3].

Empfangen von Sensorwerten mit rtl_433

Auf der Softwareseite ist eine beliebte Wahl zum Lesen von 433,92-MHz-Signalen das Programm *rtl_433* [4], das trotz seines Namens ein generischer Datenempfänger ist, hauptsächlich für die ISM-Bänder 433,92 MHz, 868 MHz (SRD), 315 MHz, 345 MHz und 915 MHz.

Jeder DVB-Dongle mit Realtek RTL2832 sollte mit rtl_433 funktionieren, auch der offizielle RTL-SDR-Dongle. Ich verwende den RTL-SDR-Dongle mit einer Dipolantenne aus dem RTL-SDR-Kit. Schließen Sie einfach die Antenne an den RTL-SDR an und stecken Sie den RTL-SDR in einen USB-Port Ihres Raspberry Pi.

Warnung: Der RTL-SDR wird im Betrieb ordentlich warm. Achten Sie darauf, wo Sie ihn positionieren.

Bild 4. Der Befehl rtl_433 findet automatisch den RTL-SDR-Empfänger und beginnt mit der Anzeige der empfangenen Sensormesswerte.

Installation von rtl_433

Das rtl_433-Programm wird aktiv entwickelt und gepflegt und verfügt über mehr als 150 Protokolldekoder für verschiedene Geräte, die auf 433,92 MHz senden. Darüber hinaus kann es die empfangenen Werte an einen MQTT-Broker senden. Glücklicherweise hat jemand genau für diesen Zweck einen Docker-Container mit rtl_433 geschrieben [5].

Erstellen Sie zunächst ein Verzeichnis für den Container:

```
mkdir -p /home/pi/containers/rtl433tomqtt
```

Fügen Sie dann die Containerdefinition zu Ihrer Datei *docker-compose.yml* hinzu:

```
version: '3.7'
 service:
 mosquitto:
 # mosquitto config
 rtl433tomqtt:
 image: bademux/rtl_433tomqtt:latest
 container_name: rtl433tomqtt
 restart: always
 volumes:
 - ./containers/rtl433tomqtt:/home/user/.config/
rtl_433:ro
 - /etc/localtime:/etc/localtime:ro
 devices:
 - /dev/bus/usb:/dev/bus/usb
```

Geben Sie die Konfiguration des Docker-Containers für Mosquitto anstelle von `#mosquitto config` ein. Der Container benötigt Zugriff auf den USB-Bus, um vom RTL-SDR-Gerät lesen zu können. Beachten Sie auch, dass das von Ihnen erstellte Verzeichnis als ein Volume gemountet wird. Sie müssen noch keine Konfigurationsdatei erstellen.

Erstellen Sie zunächst eine `udev`-Regel, um dem USB-Gerät die nötigen Berechtigungen zu erteilen:

```
sudo nano /etc/udev/rules.d/20.rtl-sdr.rules
```

Geben Sie folgende Zeile ein:

```
SUBSYSTEM==“usb“, ATTRS{idVendor}==“0bda“, ATTRS{idProduct}==“2838“,OWNER=“pi“, MODE=“0660“
```

Speichern Sie die Datei mit <Strg+o> und beenden Sie nano mit <Strg+x>. Ziehen Sie dann den RTL-SDR ab und schließen Sie ihn wieder an. Schauen Sie sich mit `lsusb` nun die Liste der angeschlossenen USB-Geräte an und suchen Sie in dieser Liste nach einer Zeile wie dieser:

```
Bus 001 Device 008: ID 0bda:2838 Realtek Semiconductor Corp. RTL2838 DVB-T
```

Notieren Sie sich die Bus- und Gerätenummer: 001 und 008. Sehen Sie sich nun die Gerätedateien für den Bus 001 an:

```
ls -l /dev/bus/usb/001
```

Sie sollten eine Zeile wie diese sehen:

```
crw-rw---- 1 pi root 189, 10 May 7 20:22 008
```

Dies zeigt, dass das Gerät dem Benutzer *pi* gehört, der über Lese- und Schreibrechte verfügt.

Danach erstellen Sie den Container mit:

```
docker-compose up -d
```

Nachdem der Container erstellt wurde, sehen Sie sich seine Protokolle an:

```
docker logs -f rtl433tomqtt
```

Wie in **Bild 4** dargestellt, sollten Sie einige Meldungen sehen, dass das Programm *rtl_433* an einigen Stellen versucht, eine Konfigurationsdatei zu finden, und dass es mehr als 120 Dekodierprotokolle registriert und ein Empfangsgerät gefunden hat. Dann sollten Sie die Meldung *Tuned to 433.920MHz* sehen. Verläuft dies alles gut, so trudeln nun die Sensorwerte ein. Haben Sie etwas Geduld, denn viele dieser Sensoren senden nur einmal pro Minute.

Konfigurieren von rtl_433

Am Anfang des Protokolls sahen Sie, dass *rtl_433* versuchte, einige Konfigurationsdateien zu finden. Es fand keine, also benutzte es einfach eine Standardkonfiguration, die zum Testen zwar in Ordnung war, aber kein MQTT verwendete. Nun wollen wir eine Beispielkonfigurationsdatei in einen Pfad kopieren, in dem *rtl_433* sucht:

Bild 5. Der Befehl rtl_433 kann die Sensorwerte in vielen Formaten, auch in JSON-Notation anzeigen.

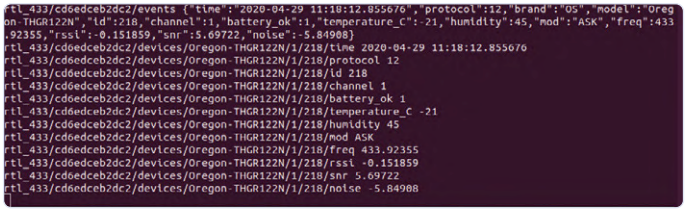


Bild 6. Der Befehl rtl_433 sendet die empfangenen Sensorwerte an Ihren MQTT-Broker.

```
docker cp rtl433tomqtt:/usr/local/etc/rtl_433/rtl_433.
example.conf /home/pi/containers/rtl433tomqtt/rtl_433.
conf
```

Wenn Sie das Verzeichnis *containers/rtl433tomqtt* im pi-User-Home-Verzeichnis unter */home/user/.config/rtl_433* im Container aufsuchen, können Sie diese Konfigurationsdatei bearbeiten und den Container neu starten, damit diese Konfiguration verwendet wird. Sie können zum Beispiel Protokolle deaktivieren, die Sie nicht benötigen, oder Protokolle aktivieren, die standardmäßig deaktiviert sind. Die Konfigurationsdatei ist gut kommentiert, was Ihnen helfen sollte, herauszufinden, was zu ändern ist. Außerdem finden Sie viele Informationen in der Online-Dokumentation von *rtl_433* [6], einschließlich der zu unternehmenden Schritte, um Unterstützung für einen Sensor hinzuzufügen, der (noch) nicht unterstützt wird. Starten Sie nach den Änderungen an Ihrer Konfigurationsdatei den Container neu:

```
docker restart rtl433tomqtt
```

Wenn alles gut geht, sollte das Protokoll zeigen, dass *rtl_433* nach der zweiten Datei aufhört, weiter nach einer Konfigurationsdatei zu suchen:

```
Reading conf from „/home/user/.config/rtl_433/rtl_433.
conf“.
```

Mit der Beispielkonfigurationsdatei zeigt *rtl_433* nun die Ausgabe jedes Sensors als JSON-Dictionary an (**Bild 5**).
Dies ist schon einmal ein großer Schritt voran, um Ihre 433,92-MHz-Sensoren in die Hausautomatisierung zu integrieren, aber der letzte Schritt ist die Veröffentlichung dieser Werte über Ihren MQTT-Broker.

433,92-MHz-Sensorwerte via MQTT veröffentlichen

Das Programm rtl_433 bietet bereits Unterstützung für das Senden der empfangenen Sensorwerte an einen MQTT-Broker. Öffnen Sie die Konfigurationsdatei (*/home/pi/pi/containers/rtl433tomqtt/rtl_433.conf*) und suchen Sie die Zeile, in der es heißt:

```
output json
```

Sie können diese Zeile hier beibehalten, weil Sie mehrere Ausgaben angeben können, oder sie in `output kv` ändern, wenn Sie die menschenfreundlichere Standardausgabe bevorzugen. Fügen Sie die folgende Zeile hinzu, um eine MQTT-Ausgabe zu definieren:

```
output mqtt://mosquitto:1883,user=home,pass=PASSWORD
```

Stellen Sie sicher, dass Sie den richtigen Benutzernamen und das richtige Passwort für Ihren MQTT-Broker eingeben.

Warnung: Das Programm rtl_433 unterstützt kein MQTT über TLS. Wenn Sie den rtl_433toMQTT-Container auf demselben Raspberry Pi wie Ihren Mosquitto-Container ausführen, ist dies natürlich kein Problem, aber wenn Sie Ihren RTL-SDR-Empfänger auf einem anderen Rechner als den MQTT-Broker betreiben, um zum Beispiel eine bessere Abdeckung zu erreichen, empfehle ich Ihnen, einen Mosquitto-Container neben dem rtl_433toMQTT-Container laufen zu lassen und ihn als Brücke zu Ihrem Haupt-MQTT-Broker über eine verschlüsselte Verbindung zu konfigurieren. Einzelheiten finden Sie im Anhang am Ende des Buches.
Nach einem Neustart Ihres Containers sollten Sie für jeden Sensorwert eine MQTT-Nachricht sehen, die unter dem Hauptthema rtl_433 veröffentlicht wird.

```
mosquitto_sub -t 'rtl_433/#' -v
```

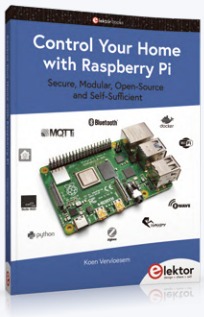
Zum Beispiel sehe ich jedes Mal, wenn der Temperatur- und Feuchtigkeitssensor in meinem Gefrierschrank eine Nachricht sendet, Daten wie in **Bild 6**. Jetzt können Sie noch einige Dinge an der MQTT-Konfiguration ändern. Einzelheiten erfahren Sie aus den Kommentaren in der Beispielkonfigurationsdatei. Diese Konfiguration ändert zum Beispiel die MQTT-Topics in etwas Kürzeres:

```
output
 mqtt://mosquitto:1883,user=home,pass=PASSWORD,de-
vices=rtl433/[model]/
 [channel]/[id]
```

Und wenn Sie nicht an den Low-Level-Metadaten über die Funkverbindung interessiert sind (Modulation, Frequenz, RSSI und so weiter), entfernen Sie einfach die Zeile `report_meta level` in der Konfigurationsdatei oder kommentieren Sie sie aus.
So, jetzt haben wir alles beisammen: Alle Messungen Ihrer 433,92-MHz-Sensoren werden an Ihren MQTT-Broker geschickt, und Sie können deren Topics in Ihren Python-Skripten oder Ihrer anderen Heimautomatisierungssoftware abonnieren.

Fazit und mehr

Drahtlose Geräte im 433,92-MHz-Frequenzband sind in der DIY-Community sehr beliebt, sie sind billig und leicht erhältlich.



CONTROL YOUR HOME WITH RASPBERRY PI

Dieses Buch zeigt, wie Sie Ihr Haus mit einem Raspberry Pi automatisieren können. Sie erfahren, wie Sie verschiedene drahtlose Protokolle für die Hausautomation wie Bluetooth, 433,92 MHz-Funk, Z-Wave und Zigbee verwenden können. Schnell werden Sie sogar in der Lage sein, Ihr Zuhause mit Python, Node-RED und Home Assistant zu automatisieren und mit Ihrem Hausautomationssystem zu kommunizieren. All dies geschieht sicher, in einem modularen System, vollständig quelloffen, ohne auf Dienste Dritter angewiesen zu sein. Sie haben die volle Kontrolle über Ihr Zuhause und niemand sonst.
Am Ende des Buches sind Sie in der Lage, Ihren Raspberry Pi als hochflexibles Hausautomatisierungs-Gateway für Protokolle Ihrer Wahl zu installieren, zu konfigurieren und verschiedene Dienste mit MQTT zu verbinden.

- Machen Sie Ihren Raspberry Pi zu einem zuverlässigen Gateway für verschiedene Hausautomatisierungsprotokolle.
- Machen Sie Ihre Hausautomatisierung mit Docker Compose reproduzierbar.
- Sichern Sie Ihre gesamte Netzwerkkommunikation mit TLS.
- Erstellen Sie ein Videoüberwachungssystem für Ihr Zuhause.
- Automatisieren Sie Ihr Heim mit Python, Node-RED, Home Assistant und AppDaemon.
- Greifen Sie von entfernten Standorten aus sicher auf Ihr Smart-Home-Dashboard zu.
- Verwenden Sie vollständig offline Sprachbefehle in Ihrer eigenen Sprache.

Die Software und die Errata für das Buch sind auf GitHub verfügbar [8].

Buch:
www.elektor.de/control-your-home-with-raspberry-pi

e-Buch:
www.elektor.de/control-your-home-with-raspberry-pi-e-book

Besonders interessant sind Temperatur- und Feuchtigkeitssensoren, mit denen Sie Ihren Kühl- und Gefrierschrank oder Ihr Gewächshaus überwachen können.

Sie brauchen dazu nur einen billigen DVB-Dongle und eine Antenne. Das Programm *rtl_433* unterstützt mehr als 150 Protokolle von 433,92-MHz-Geräten und ermöglicht es Ihnen, die empfangenen Signale als Nachrichten an Ihren MQTT-Broker weiterzuleiten, so dass andere Hausautomatisierungssoftware darauf reagieren kann. Es gibt immer noch viele interessante Themen, die ich kaum berührt habe, etwa die optimale Antennenwahl und -platzierung für eine bessere Abdeckung. Es ist auch spannend zu versuchen, *rtl_433*-Unterstützung für ein bisher nicht unterstütztes Gerät hinzuzufügen. Das Projekt verfügt über eine detaillierte Dokumentation darüber, wie man die Rohsignale erfasst und ihr Protokoll durch Reverse Engineering nachvollzieht.

Sie können auch andere Empfänger ausprobieren, zum Beispiel die RFXtrx-Gerätefamilie, die einen Transceiver enthält, mit dem Sie RTS-Rollläden von Somfy steuern können. Es gibt sogar eine Python-Bibliothek mit Namen pyRFXtrx [7], um mit Ihren 433,92 MHz-Geräten über einen RFXtrx-Transceiver zu kommunizieren. Home Assistant (siehe Kapitel 10) verwendet diese Bibliothek zur Unterstützung von 433,92-MHz-Geräten. Der RFXtrx-Transceiver kostet jedoch viel mehr als ein RTL-SDR. Am anderen Ende des Preisspektrums sind der Sender STX882 und der Empfänger SRX887 gute Ausgangspunkte zum Ausprobieren, was Sie mit den billigsten Geräten alles anstellen können.

200534-02

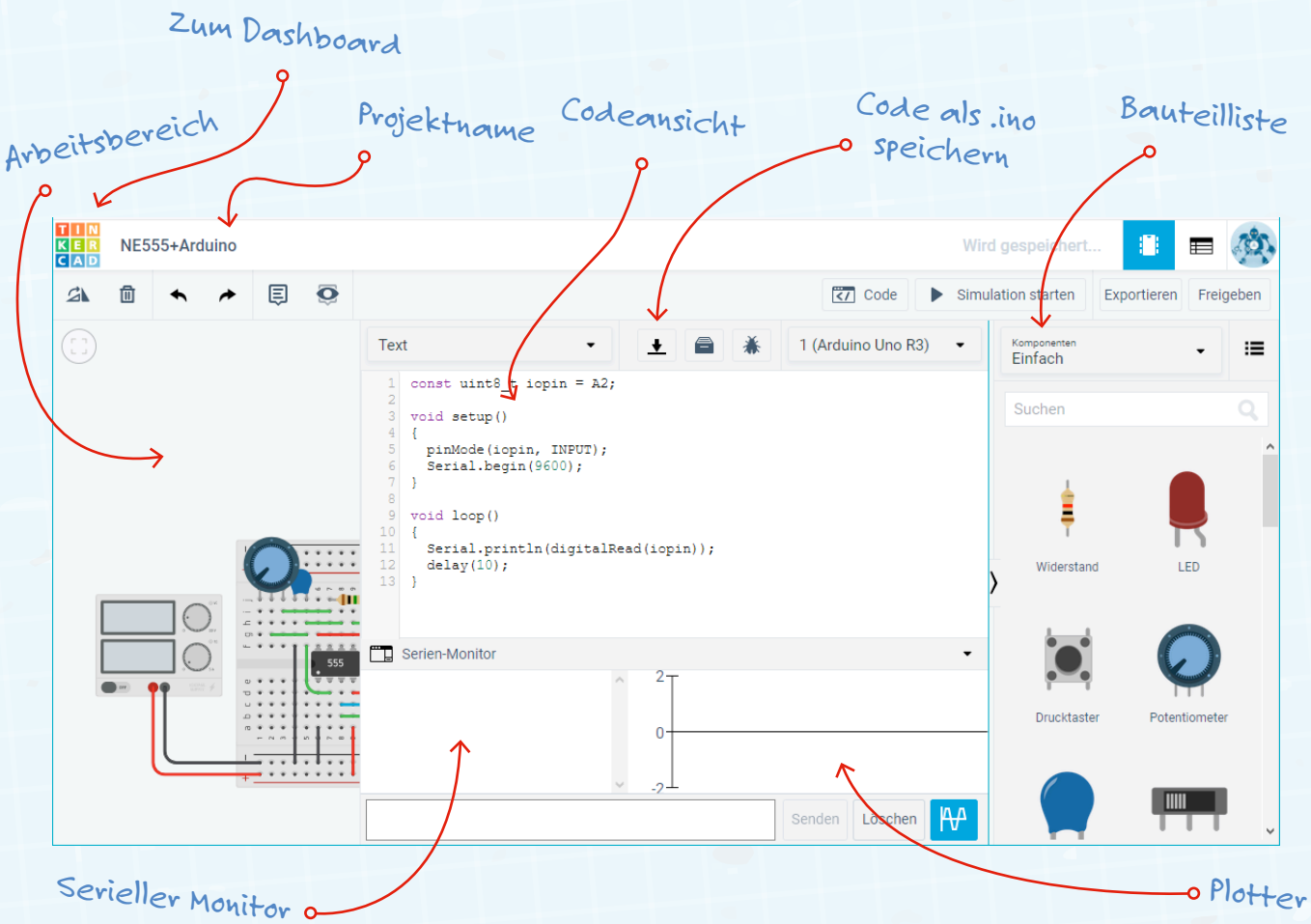
Erratum
Kurz bevor dieses Buch fertiggestellt wurde, kündigte der Entwickler an, dass er des Docker-Image (das mehr als 100.000-mal „gepullt“ wurde) nicht mehr aktiv weiterentwickeln würde, so dass keine neuen Funktionen zu erwarten sind. Es bleibt abzuwarten, ob ein anderes Image genau so populär wird. Auch wenn Sie ein anderes Image finden (wollen oder müssen), sollten nur einige kleine Änderungen nötig sein, um es zu verwenden: Die Hauptkonfigurationsdatei von rtl_433 bleibt die gleiche.

Sie haben Fragen oder Kommentare?
Wenn Sie Fragen zu diesem Artikel haben, können Sie sich gerne per E-Mail an den Autor wenden: koen@vervloesem.eu.

Ein Beitrag von
Autor: **Koen Vervloesem**
Illustrationen: **Koen Vervloesem**
Redaktion: **Jan Buiting**
Übersetzung: **Rolf Gerstendorf**
Layout: **Giel Dols**

LINKS & LITERATUR

[1] **RTL-SDR**: www.rtl-sdr.com
[2] **Elektor RTL-SDR-Bausatz mit Antennen, Halterungen und Verlängerungskabel**: www.elektor.de/rtl-sdr-software-defined-radio-with-dipole-antenna-kit
[3] **Dipolantennen-Kit am RTL-SDR**: www.rtl-sdr.com/using-our-new-dipole-antenna-kit/
[4] **SDR-Programm rtl_433**: https://github.com/merbanan/rtl_433
[5] **Docker Container von Bademux**: https://github.com/bademux/rtl_433toMQTT
[6] **Online-Dokumentation rtl_433**: https://triq.org/rtl_433/
[7] **Python-Bibliothek**: https://github.com/Danielhiversen/pyRFXtrx
[8] **Buch-Software auf Github**: https://github.com/koenvervloesem/raspberry-pi-home-automation

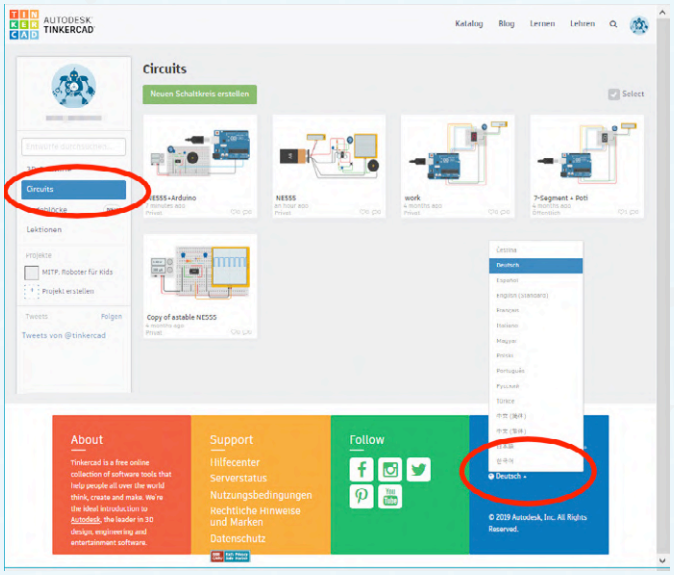


Schaltungen online simulieren

Von **Florian Schäffer**

Haben Sie (noch) keine Bauteile, um die Schaltungen praktisch zu erproben oder wollen Sie nichts kaputt machen? Dann ist eine Schaltungssimulation hilfreich, die sogar Arduino-Code ausführen kann!

SPICE (Simulation Program with Integrated Circuit Emphasis) ist die wohl bekannteste Software zur Simulation analoger und digitaler elektrischer Schaltungen. Allerdings ist die Nutzung oft mühsam und erfordert einen geeigneten Schaltplan. Wie gut, dass es inzwischen eine grafische Benutzeroberfläche gibt, die Sie online und ohne Installation von Software benutzen können und in der Sie die Schaltung wie auf einem Breadboard aufbauen. Tinkercad Circuits bietet sogar die Integration von ATmega- (Arduino Uno) und ATtiny-Prozessoren für die Sie Programme in C/C++ sowie der grafischen Sprache Scratch erzeugen und ausführen können.



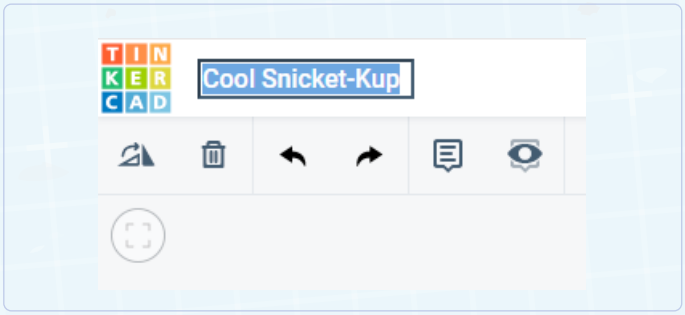
Wechseln Sie im Dashboard in den Bereich Circuits. Am unteren Rand können Sie gegebenenfalls die Sprache umstellen.

Melden Sie sich auf der Seite https://www.tinkercad.com an. Dazu können Sie ein neues Konto erstellen oder sich über einen Drittanbieter wie Google anmelden. Wechseln Sie im Dashboard in den Bereich CIRCUITS – als neuer Nutzer werden noch keine Schaltungsentwürfe zu sehen sein. Am unteren Rand können Sie gegebenenfalls die Sprache auf DEUTSCH umstellen.

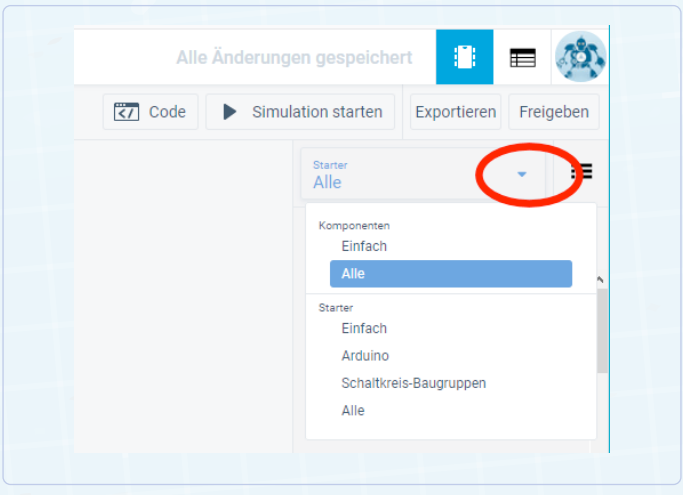
Das folgende Projekt finden Sie auch online unter der Adresse https://www.tinkercad.com/things/az1gBrXUclZ.

1. Klicken Sie auf NEUEN SCHALTKREIS ERSTELLEN. Die Breadboardansicht ähnelt an vielen Stellen der von Fritzing, so dass es vorteilhaft ist, wenn Sie sich vorab damit vertraut machen und den Artikel dazu lesen (siehe letzte Ausgabe).

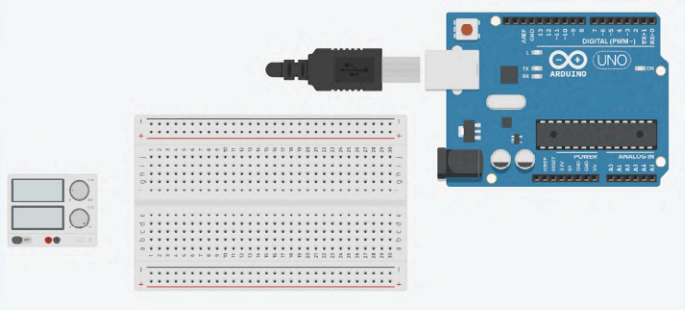
2. Ihr Entwurf wird permanent in der Cloud gespeichert. Oben links neben dem Logo von Tinkercad wurde dazu bereits ein phantasievoller Name festgelegt. Sie können auf den Text klicken und einen eigenen Projektnamen vergeben.



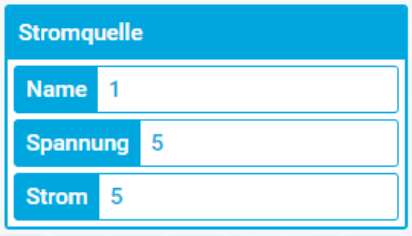
3. Am rechten Rand können Sie eine Vorauswahl treffen, welche Bauteile und Beispielprojekte angezeigt werden. In der Auswahlliste befinden sich unterhalb von STARTER einige Beispiele, die Sie ausprobieren können. Damit Sie ein neues Projekt von Grund auf selbst erstellen, wählen Sie jetzt bei KOMPONENTEN ALLE aus.



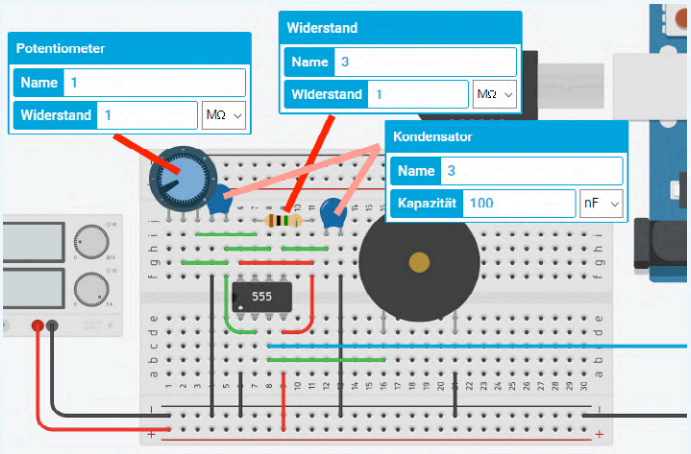
4. Suchen Sie aus der Bauteilauswahl die STROMQUELLE (Labornetzteil), den ARDUINO UNO R3 und die KLEINE STECKPLATINE. Klicken Sie das jeweilige Symbol an und legen Sie das Bauteil dann im Arbeitsbereich ab.



5. Für viele Bauteile können Sie Parameter wie Farbe, Typ oder Dimensionierung festlegen, wenn Sie das Bauteil im Arbeitsbereich anklicken. Es öffnet sich dann ein kleines Dialogfenster mit den Einstellmöglichkeiten. Klicken Sie auf die Stromquelle und tragen Sie bei Spannung „5" ein. Bei allen Werten beachten Sie, dass Sie die US-Schreibweise bei Zahlen mit Dezimalpunkt statt –Komma benutzen.



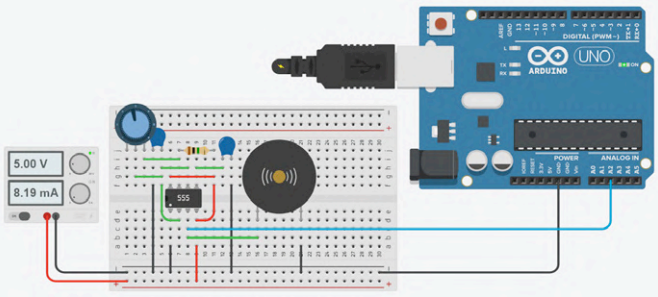
6. Platzieren Sie die Bauteile und Leitungen auf dem Breadboard. In der Grafik sehen Sie, welche Werte Sie für die Bauteile einstellen müssen (soweit möglich).



7. Zusätzlich ist eine Verbindung zu einem GND-Pin am Arduino notwendig (schwarze Linie nach rechts) und die blaue Verbindung zu Pin A2.

8. Haben Sie die Schaltung fertig aufgebaut, können Sie die elektrische Funktion testen und simulieren, in dem Sie oben rechts auf SIMULATION STARTEN klicken. Der USB-Stecker wird in einer kurzen Animation in den Arduino gesteckt, um zu zeigen, dass dieser jetzt mit Strom versorgt wird (zudem leuchtet die LED „ON"). Allerdings gibt es noch kein Programm, das auf ihm läuft. Zusätzlich werden auf dem Labornetzteil die Spannung und der momentan aufgenommene Strom angezeigt.

9. Sie können mit gedrückter Maustaste an den Drehknöpfen des Netzteils drehen. Die Schaltung an sich verträgt bis etwa 18 V – allerdings nicht der Arduino, der über die blaue Leitung verbunden ist. In der Praxis würden Sie Ihren Mikrocontroller damit zerstören – hier schadet es nicht. Der Piezosummer gibt einen knatternden Ton von sich, wenn Sie an Ihrem PC einen Lautsprecher angeschlossen haben. Mit der Maus können Sie das blaue Potentiometer oben links auf dem Steckboard durch drehen verstellen und dadurch den Ton ändern.



10. Klicken Sie auf SIMULATION STOPPEN und dann auf CODE (links daneben), um das Eingabefenster für den Quellcode zu öffnen. Vom rechten Rand schiebt sich die Codeansicht ins Bild. Diese zeigt zuerst den Editor für Scratch. Stellen Sie oben über das Auswahlfeld auf TEXT um. Es wird eine Warnung angezeigt, die Sie mit WEITER bestätigen.

11. Es gibt bereits ein kleines Programm im Codefenster. Löschen Sie den gesamten Inhalt und ersetzen Sie ihn durch folgende Befehle:

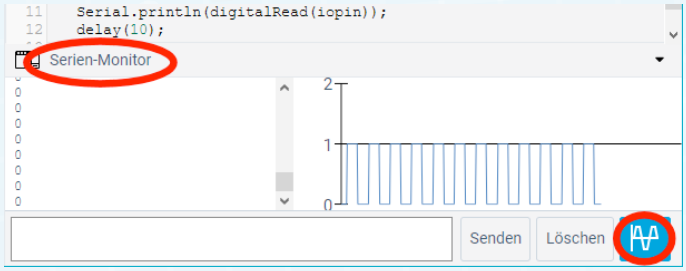
```
const uint8_t iopin = A2;

void setup()
{
  pinMode(iopin, INPUT);
  Serial.begin(9600);
}

void loop()
{
  Serial.println(digitalRead(iopin));
  delay(10);
}
```

12. Wenn Sie jetzt wieder auf SIMULATION STARTEN klicken, wird auch der Programmcode ausgeführt. Sollte Ihr Programm Fehler enthalten, dann werden diese wie auch in der Arduino-IDE hervorgehoben und müssen erst beseitigt werden.

13. Der Arduino soll in diesem Beispiel wie ein einfaches Oszilloskop arbeiten und den Signalpegel an seinem I/O-Pin ausgeben, der dem Signal entspricht, das den Piezosummer antreibt. Klicken Sie dazu am unteren Rand der Codeansicht auf SERIELLER MONITOR, so dass Sie die Folge von Nullen und Einsen sehen. Ganz unten rechts gibt es ein Symbol mit einem Kurvenverlauf: Klicken Sie darauf, um die grafische Ausgabe zu öffnen.



14. Drehen Sie wieder am Poti, um den Ton zu ändern und Sie sehen, wie sich auch die gezeichnete Kurve ändert.

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.**

200225-C-01

Aus dem Leben gegriffen

Der schmale Grat zwischen Ordnung und Chaos

Von **Ilse Joostens** (Belgien)

Ab und zu komme ich zu der Erkenntnis, dass der Schreibtisch eines durchschnittlichen Elektronikers in der Regel in scharfem Kontrast zum Schreibtisch eines durchschnittlichen Buchhalters oder Finanzbeamten steht. Während ich bei letzteren vermute, dass sie einen ungesunden, fast sadistischen Wunsch haben, die Welt zu organisieren, herrscht bei vielen Elektronikern blankes Chaos, und nicht selten scheint ihr Arbeitsplatz auf ein Entseuchungsteam zu warten. In dieser Hinsicht ist der Arbeitsplatz des verstorbenen Bob Pease, des Erfinders des Breadboards, berüchtigt [1]. Natürlich wissen wir es besser, aber dem ahnungslosen Laien dürfte es schwer fallen, den Unterschied zu einem Foto aus einem Katastrophengebiet nach dem Durchzug eines tropischen Wirbelsturms zu erkennen…

Der klassische Erbsenzähler aka Buchhalter.

Chaos ist nicht nur eine Theorie

Obwohl ich noch viel von Bob Pease lernen kann, muss ich Ihnen mit Schamröte auf den Wangen gestehen, dass auch ich eine unverbesserliche Chaotin bin. Mit meiner unaufhaltsamen Neigung zur „Haufenbildung“ habe ich es geschafft, meine Eltern, so manchen ehemaligen Arbeitgeber und sogar meinen Partner zur puren Verzweiflung zu treiben.

Vor Jahren war ich hauptsächlich mit SMD-Bauteilen beschäftigt und wir alle wissen, dass man die besser in den Tütchen lassen sollte, wenn man verhindern will, dass sie verloren gehen oder durcheinander kommen. Wie üblich stapelten sich die Tütchen mit den Komponenten, und nach einer Weile hatte ich mehrere Stapel von Sichtlagerkästen mit verschiedenen Bauteilen gefüllt. Das Ergebnis lässt sich erahnen: Jedes Mal, wenn ich eine kleine Reihe von Platinen anfertigen wollte, verbrachte ich mehr Zeit damit, die Teile zusammen zu suchen, als die Leiterplatten selbst zu bestücken und im Reflow-Verfahren aufzuschmelzen. Wie oft habe ich Teile bestellt, die ich noch auf Lager hatte, aber nicht sofort finden konnte, oder - schlimmer noch - Teile, die dringend benötigt wurden, bewusst nicht bestellt, weil ich irrtümlich dachte, es seien noch genug da. Da war also noch Verbesserungspotenzial…

Viele neigen dazu, ihre Bauteile schön in Ablagefächern oder Schubladenschränken nach Typ und Wert zu sortieren, mit dem heimlichen Hintergedanken, möglichst viele komplette Wertesätze zu sammeln, zum Beispiel die gesamte E24-Widerstandsreihe. Dies ist ein relativ gutes System für Maker oder sogar Elektronikentwickler und -reparateure, aber es ist weit davon entfernt, auch bei der Produktion flexibel zu sein.

Vor allem das SMD-Streugut, auch Hühnerfutter genannt, und ihre bedrahteten Äquivalente kaufe ich in großen Mengen, je nachdem, was ich für meine Produkte benötige. Dabei ist es nicht vernünftig, tausende von Widerständen jeden Wertes in der E24-Serie (um nur ein Beispiel zu nennen) zu lagern, wenn man mehr als drei Viertel dieser Werte fast nie verwendet, ganz zu schweigen davon, dass man sie schön sortiert in Schubladen räumen kann.

Dank einiger Industriespionage an einem Tag der offenen Tür bei einem unserer Distributoren gelang es mir, ein besseres System zu entdecken. Ich verwende jetzt Regale mit Schachteln, bei denen jede Schachtel ihren eigenen Code hat, der die Position angibt, zum Beispiel „A5K0“. Dies steht für Regal „A“, Regalbrett „5“, Fach „K“ und dort von vorne nach hinten die erste Schachtel „0“. Um

Dieses System funktioniert in der Praxis sehr gut.

herauszufinden, wo sich ein Bauteil befindet, verwende ich eine Tabelle. Der Vorteil dieser Methode besteht darin, dass die Bauteile selbst nicht sortiert werden müssen. So kann eine Schachtel mit Mikrocontrollern auch neben einer Schachtel mit Schrauben stehen. Man kann auch leicht Dosen verschieben und leere Positionen können für etwas anderes wiederverwendet werden.

Nur diese Tabelle macht mich auch nicht sehr glücklich, aber leider habe ich noch keine erschwingliche und brauchbarere Software dafür finden können. Irgendwas ist immer...

Und na ja, jetzt verbringe ich eigentlich mehr Zeit damit, Bauteile ein- und auszutragen und aus den Schachteln zu nehmen, als damit meine Leiterplatten zu bestücken. Natürlich vergesse ich manchmal, die Tabelle zu aktualisieren, wodurch manche Bestellungen immer noch vermasselt werden. Vielleicht war das Chaos gar nicht so schlimm. Allerdings können wir nun die Bestandsdaten am Jahresende schneller an den Korinthenkacker (sorry) in unserer Buchhaltung übermitteln.

Bauteile bestellen für Dummies

Für eine fortgeschrittene Haufenbildung benötigen Sie natürlich Bauelemente und vorzugsweise viele verschiedene in großen Mengen. Um ehrlich zu sein, war ich schon seit Jahren nicht mehr in einem konventionellen Elektronikgeschäft und heute bestelle ich alles online bei den bekannten Großhändlern. Der Preis ist natürlich wichtig, aber ich empfehle trotzdem, bei kritischen Bauteilen und Halbleitern einen großen Bogen um Onkel Baba zu machen.

Aus mir unbekannten Gründen ist das Angebot an hellen LEDs bei den bekannten Anbietern eher entmutigend, weshalb ich gelegentlich über eBay bestelle.

Normalerweise öffne ich die Webseiten von etwa vier Anbietern in verschiedenen Tabs meines Browsers und wähle denjenigen, der eine Komponente in der von mir gewünschten Menge am günstigsten anbietet. Da in der Regel nicht jeder Lieferant alles vorrätig hat, bestelle ich meistens bei mindestens drei Lieferanten, was dazu führt, dass ein paar Tage später ein Defilee aus Lieferwagen vor meiner Haustür steht. Und dann bekomme ich natürlich eifersüchtige Blicke des Nachbarjungen, der glaubt, hier sei fast jeden Tag Weihnachten, wobei ich wirklich nicht wüsste, was er denn mit ein paar tausend MOSFET-Transistoren anfangen sollte.

Es lohnt sich auch, einen genauen Blick auf die Staffelpreise zu werfen. Normalerweise bewegen die Rabatte einen dazu, viel mehr zu kaufen, als eigentlich geplant war, was dachten Sie denn? Doch manchmal ist etwas sehr merkwürdig daran. Ich habe schon mehrmals erlebt, dass hundert Stück eines ICs exakt den gleichen Gesamtpreis haben wie fünfzig Stück. Kostenlose Bauteile also, ja, es gibt sie wirklich!

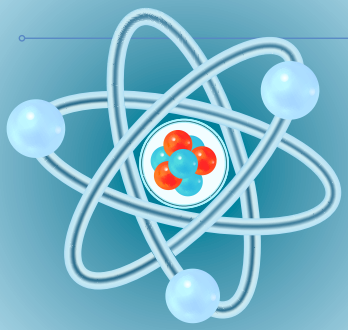
200556-03

Sie haben Fragen oder Kommentare?
Sie haben Fragen oder Anmerkungen zu diesem Artikel? Senden Sie eine E-Mail an den Autor oder an die Redaktion von Elektor über redaktion@elektor.de.

Ein Beitrag von
Text und Bilder: **Ilse Joostens**
Redaktion: **Eric Bogers**
Übersetzung: **Sophia Gerstendorf**
Entwurf: **Giel Dols**

WEBLINK

[1] **Bob Pease**: www.electronicdesign.com/technologies/analog/article/21805320/whats-all-this-messy-office-stuff-anyhow



Aller Anfang …

muss nicht schwer sein!

Von **Eric Bogers**

Wie in der vorhergehenden Folge dieser Serie versprochen, werden wir uns nun mit „greifbaren“ Bauteilen befassen - mit Widerständen, um genau zu sein. Es gibt Widerstände, solche Widerstände, andere Widerstände, spezielle Widerstände, merkwürdige Widerstände und dann noch diese Widerstände…

Widerstände

Widerstände werden zu den passiven Bauelementen gezählt, da durch sie (im Gegensatz zu aktiven Bauelementen) keine Signale verstärkt werden können. Passive Bauelemente sind Widerstände, Kondensatoren und Spulen, aktive Bauelemente sind Transistoren, Triacs - und ja, auch Dioden werden zu den aktiven Bauelementen gezählt. Eigentlich ist das nicht wahr (schließlich verstärkt eine Diode nicht), aber weil wir traditionell alle Halbleiter in einen Topf werfen, werden Dioden auch als „aktiv“ bezeichnet.

Doch sehen wir uns zunächst einmal an, wie Widerstände tatsächlich aussehen. In **Bild 1** haben wir die bekanntesten Vertreter abgelichtet. Ganz unten sehen Sie einen Metallschichtwiderstand und darüber einen Kohleschichtwiderstand, beide mit einer Belastbarkeit von 0,25 W. Unter Belastbarkeit versteht man die maximale Leistung, die im Widerstand in Wärme umgewandelt werden kann, ohne dass dieser Widerstand den Hitzetod stirbt. Das sollte man im Auge behalten! Bei Widerständen gibt es nicht nur eine maximale Belastbarkeit, sondern auch eine maximale Spannung: Wird diese überschritten, kann es zu einem Überschlag kommen und der Widerstand ist auf jedem Fall zerstört. Bei den Feld-, Wald- und Wiesenwiderständen, die üblicherweise in Hobbyprojekten eingesetzt werden, liegt die maximale Spannung in der Regel bei 250 V. In der Praxis müssen wir uns also nicht so viele Gedanken machen.

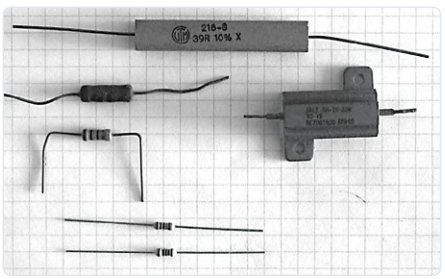


Bild 1. Einige Widerstände.

Zurück zu Bild 1: Links in der Mitte sind zwei Widerstände mit höherer Belastbarkeit (0,5 W und 1 W) und ganz oben ein zementierter 39-Ω-Widerstand zu sehen, der 11 W verkraftet. Der Leistungswiderstand in der Mitte rechts ist ein Exemplar, das auf einen Kühlkörper geschraubt wird und dann nicht weniger als 25 W verarbeiten kann.

Normale Widerstände für den täglichen Gebrauch – wie die Exemplare unten - sind so klein, dass es schier unmöglich ist, ihren Wert in lesbarer Form aufzudrucken. Heutzutage werden sogar noch viel kleinere Bauteile eingesetzt, so genannte SMDs (Surface Mounted Devices), aber da diese Artikelserie für den Elektronik-Einsteiger gedacht ist und SMDs nun wirklich nicht für eine erste Bekanntschaft geeignet sind, werden wir darauf nicht eingehen. Wir halten uns an die bekannten „bedrahteten“ und deshalb einfach zu handhabenden Bauteile, bei denen der Wert mit farbigen Ringen angezeigt wird.

Der Farbcode

Es gibt Widerstände mit vier und mit fünf Farbringen. Die ersten zwei beziehungsweise drei Ringe geben den Wert des Widerstandes an, der noch mit einem (dezimalen) Faktor multipliziert werden muss, der mit dem folgenden Ring symbolisiert wird. Schließlich gibt der letzte Ring die Toleranz an. Toleranz? Ja: Wenn wir eine Charge von 1000-Ω-Widerständen nehmen und sie mit einem sehr genauen Ohmmeter messen, werden wir feststellen, dass keine zwei Widerstände exakt den gleichen Wert aufweisen. Bei einem messen wir 1001,3 Ω, bei einem anderen 998,6 Ω und so weiter. Dies ist auf unvermeidliche kleine Abweichungen im Produktionsvorgang zurückzuführen. Die Toleranz gibt an, zwischen welchen Extremwerten der tatsächliche Wert eines Widerstandes liegt. Für einen 1000-Ω-Widerstand mit einer Toleranz von 5 % liegt der tatsächliche Wert zwischen 950 Ω und 1050 Ω; für einen 1 %-Widerstand liegt der tatsächliche Wert zwischen 990 Ω und 1010 Ω. Metallschichtwiderstände sind in der Regel enger toleriert als Kohleschichtwiderstände. Es kommt auch mal, aber selten, ein Ausreißer vor, dessen Wert außerhalb der Toleranzgrenzen liegt.

Übrigens: Für die meisten Anwendungen sind 5 %-Kohleschichtwiderstände gut genug; die (teureren) Metallschichtwiderstände verwenden wir nur, wenn es wirklich sehr genau sein muss, zum Beispiel in Filterschaltungen) und/oder wenn die Schaltung möglichst rauscharm aufgebaut werden soll, etwa bei einem High-End-Audioverstärker.

Tabelle 1 zeigt den Farbcode von Widerständen. Ein 1%iger Metallschichtwiderstand von 1 kΩ trägt den Farbcode braun, schwarz, schwarz, braun, braun. Leider ist in diesem Fall auf den ersten Blick nicht klar, in welche Richtung der Farbcode gelesen werden soll, und umgekehrt könnte es sich um einen Widerstand von 110 Ω handeln. Meist weist der Toleranzring einen klein wenig größeren Abstand zum benachbartem Faktor-Ring auf. Ein weiterer Nachteil der Farbringe ist, dass die Farben rot und orange manchmal schwer zu unterscheiden sind. In allen Zweifelsfällen löst das Ohmmeter das Problem!

Widerstände werden (aus offensichtlichen Gründen) nicht in allen denkbaren Werten, sondern in genormten E-Reihen hergestellt. Dabei ist das Verhältnis zweier aufeinander folgenden Werte innerhalb einer Dekade mehr

Tabelle 1. Farbcode von Widerständen.

Wert				Faktor	Toleranz
4 Ringe:	**1. Ring**	**2. Ring**	**–**	**3. Ring**	**4. Ring**
5 Ringe:	**1. Ring**	**2. Ring**	**3. Ring**	**4. Ring**	**5. Ring**
schwarz	–	0	0	1	
braun	1	1	1	10	±1%
rot	2	2	2	100	±2%
orange	3	3	3	1 k	
gelb	4	4	4	10 k	
grün	5	5	5	100 k	±0,5%
blau	6	6	6	1 M	
violett	7	7	7	10 M	
grau	8	8	8	100 M	
weiß	9	9	9	1 G	
gold	–	–	–	0,1	±5%
silber	–	–	–	0,01	±10%

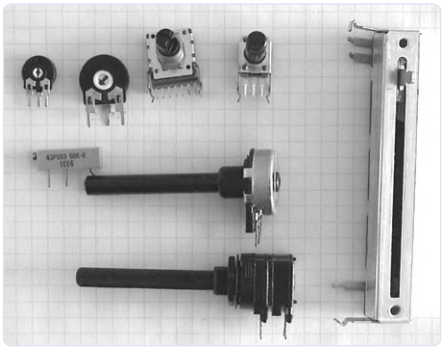


Bild 2. Potentiometer und Einstellpotentiometer.

oder weniger gleich groß. Die E-Werte werden wiederum in verschiedenen Dekadierungen produziert (zum Beispiel 2,2 Ω, 22 Ω, 220 Ω, 2,2 kΩ, 22 kΩ, 220 kΩ und 2,2 MΩ). **Tabelle 2** zeigt die am weitest verbreiteten E-Reihen E3, E6, E12 und E24, wobei die Ziffer die Anzahl der Werte pro Dekade bezeichnet. Für Präzisionsanwendungen gibt es die Reihen E48, E96 und E192, also insgesamt sieben genormte E-Reihen.

Es gibt übrigens auch so genannte R-Reihen, aber die Wahrscheinlichkeit, dass Sie in der Elektronik damit zu tun bekommen, ist sehr gering. R-Reihen finden Sie in der Elektrotechnik höchstens bei Sicherungen beziehungsweise Leitungsschutzschaltern. Wenn Sie einmal einen extrem genauen und gleichzeitig extrem krummen Wert benötigen, der in keiner der E-Reihen vorkommt, dann kann er mit einer entsprechend gewählten Parallel- und/oder Reihenschaltung von E24-Widerständen realisiert werden. Oder Sie können natürlich auch einen variablen Widerstand verwenden...

Variable Widerstände

Potentiometer (Potis) und Einstellpotentiometer (Trimmpotis) sind Widerstände, deren Werte geändert werden können. Potentiometer sind meist mit einer Achse ausgestattet, die durch ein Loch im Gerätegehäuse ragt und vom Benutzer gedreht werden kann. Das bekannteste Beispiel ist natürlich der Lautstärkesteller eines Verstärkers (obwohl im heutigen digitalen Zeitalter dafür oft Impulsgeber - Enkoder - verwendet werden). Trimmpotis hingegen befinden sich im Gerät, normalerweise auf der Platine, und sind für eine einmalige Einstellung mit einem kleinen Schraubendreher oder einem speziellen Trimmschlüssel vorgesehen.

Bild 2 zeigt einige Beispiele von (Einstell-) Potentiometern. Auf der rechten Seite sehen Sie einen Schieberegler oder Fader, der hauptsächlich in Mischpulten verwendet wird, in der Mitte zwei gewöhnliche Potentiometer in Mono- und darunter in Stereo-Ausführung. Ein solches Stereopotentiometer besteht eigentlich aus zwei Mono-Potis mit einer gemeinsamen Achse.

Ein Drehpotentiometer besitzt einen mit der Achse verbundenen Schleppkontakt, der über eine Kohlenstoffbahn hin und her bewegt wird. Aufgrund von Alterung und Verschmutzung beginnt ein solches Poti irgendwann zu „krachen", eine besonders bei Verstärkern höchst unerwünschte Eigenschaft. Wenn ein solches Poti nicht hermetisch verschlossen ist, kann ein spezielles Spray eine *vorübergehende* Lösung bieten. Ansonsten kann man versuchen, die Welle einige Male kräftig von einem Endpunkt zum anderen und zurück zu drehen - aber auch das bringt bestenfalls nur eine kurze Erleichterung.

Cermet-Potentiometer reagieren weniger empfindlich auf diese Alterungsphänomene (und sind daher teurer). In Bild 2 sind oben in der Mitte zwei solche Exemplare (rechts Mono und links Stereo) zu sehen. Schließlich finden sich in der oberen linken Ecke zwei Einstellpotentiometer (die geschlossene Version ist unempfindlicher gegen Staub und Schmutz).

Linear oder logarithmisch?

In Stücklisten findet sich oft die Abkürzung *log* oder *lin* hinter dem Potentiometerwert. Dies steht für logarithmisch oder linear und gibt an, wie sich der Widerstandswert in Abhängigkeit vom Drehwinkel ändert. Logarithmische Potentiometer werden hauptsächlich als

Tabelle 2.
Genormte E-Widerstandsreihen.

E3	E6	E12	E24
1	1	1	1
			1,1
		1,2	1,2
			1,3
	1,5	1,5	1,5
			1,6
		1,8	1,8
			2,0
2,2	2,2	2,2	2,2
			2,4
		2,7	2,7
			3,0
	3,3	3,3	3,3
			3,6
		3,9	3,9
			4,3
4,7	4,7	4,7	4,7
			5,1
		5,6	5,6
			6,2
	6,8	6,8	6,8
			7,5
		8,2	8,2
			9,1

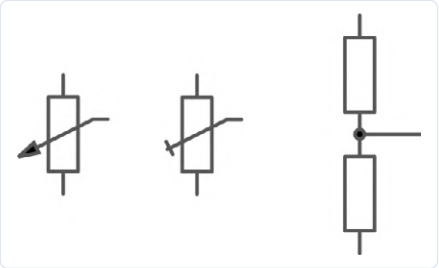


Bild 3. Schaltplansymbole für Potentiometer.

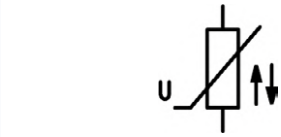


Bild 4. Spannungsabhängiger Widerstand (VDR).

Sie haben Fragen oder Kommentare?
Haben Sie Fragen oder Anmerkungen zu diesem Artikel? Senden Sie eine E-Mail an den Autor über die Elektor-Redaktion redaktion@elektor.de.

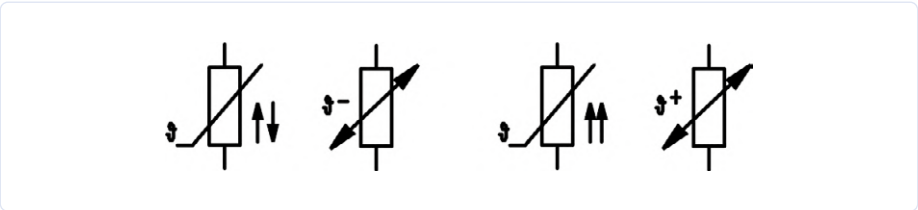


Bild 5. NTC- und PTC-Widerstände.

Ein Beitrag von
Idee und Illustrationen: **Michel Ebner**
Text und Redaktion: **Eric Bogers**
Übersetzung: **Rolf Gerstendorf**
Entwurf: **Giel Dols**

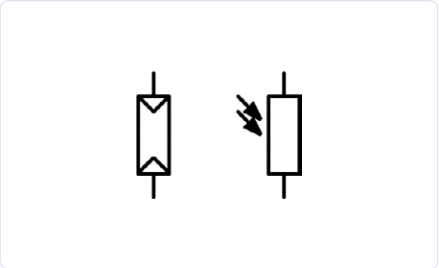


Bild 6. Lichtempfindlicher Widerstand (LDR).

Lautstärkeregler von Verstärkern verwendet, da auch unser Gehör eine mehr oder weniger logarithmische Charakteristik aufweist.
Bild 3 zeigt die Schaltsymbole für Potentiometer: links ein Potentiometer, in der Mitte ein Einstellpotentiometer und rechts das Ersatzschaltbild. Abhängig von der Position des Schleiferkontakts (der mittlere Anschluss im Schaltsymbol) variiert der Wert der beiden Teilwiderstände.
Wie bei normalen Widerständen gilt auch für Potentiometer eine maximale Belastbarkeit.

Spezielle Widerstände

Es gibt eine ganze Reihe von Widerständen, deren Wert in Abhängigkeit von einer bestimmten physikalischen Größe variiert. Die häufigsten werden im Folgenden kurz besprochen.
Bild 4 zeigt das Schaltzeichen eines spannungsabhängigen Widerstandes (Voltage Dependent Resistor, VDR). Sobald eine bestimmte Spannung überschritten wird, wird ein VDR sehr niederohmig. Das Bauelement kann so andere Komponenten vor Überspannung schützen. VDRs werden häufig verwendet, um Netztransformatoren vor Überspannung zu schützen (was bei einem falsch angeschlossenen Neutralleiter passieren kann). In diesem Fall fließt so viel Strom durch den VDR, dass die Netzsicherung anspricht und die Gefahr verschwindet. Nach einem solchen Störfall muss der VDR aber in der Regel ersetzt werden.
In **Bild 5** sehen wir einige Vertreter der Gattung „temperaturempfindliche Widerstände". Die beiden Symbole (griechisches kleines Theta) auf der linken Seite stehen für einen NTC-Widerstand (negativer Temperaturkoeffizient), die auf der rechten Seite für einen PTC-Widerstand (positiver Temperaturkoeffizient). Bei einem NTC-Widerstand nimmt der Widerstandswert ab, wenn der Widerstand wärmer wird, bei einem PTC-Widerstand ist es natürlich genau umgekehrt. NTC-Widerstände werden manchmal zur Begrenzung von Einschaltströmen verwendet. Herkömmliche Leistungsverstärker enthalten in der Regel schwere Netztransformatoren und dicke Glättungskondensatoren, die beim Einschalten noch ungeladen sind. Unmittelbar nach dem Einschalten fließt ein extrem hoher Strom, der im schlimmsten Fall ein Ansprechen des Leitungsschutzschalters verursachen kann. Ein NTC kann dieses Problem lösen, indem er zunächst den Strom begrenzt und dann, wenn er als Folge des durch ihn fließenden Stroms wärmer wird, den Einschaltstrom ansteigen lässt, bis er schließlich den Nennstrom erreicht. In der Regel werden diese Strombegrenzungswiderstände nach einigen Sekunden durch ein (elektromechanisches) Relais überbrückt, so dass im Betrieb im NTC-Widerstand keine Leistung mehr verloren geht.
Bild 6 schließlich zeigt einen lichtabhängigen Widerstand (Light Dependent Resistor, LDR). Je mehr Licht auf diesen LDR fällt, desto mehr nimmt sein Widerstandswert ab. In der Vergangenheit wurden diese Widerstände häufig in Lichtschranken eingesetzt, ihr Nachteil war und ist es jedoch, dass sie relativ langsam auf eine Lichtänderung reagieren. Heutzutage werden dafür vorzugsweise sogenannte Fotodioden oder Fototransistoren verwendet.
Damit sind wir am Ende unserer Diskussion über Widerstände. Das nächste Mal werden wir uns den Kondensatoren widmen.

200551-03

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

PASSENDE PRODUKTE

- **Basiskurs Elektronik (PDF):**
  www.elektor.de/basiskurs-elektronik-pdf

Zutriff für Unbefugte verboten!

Ein Blick ins Allerheiligste aller Elektroniker

Von **Dennis Kuschel** (Deutschland)

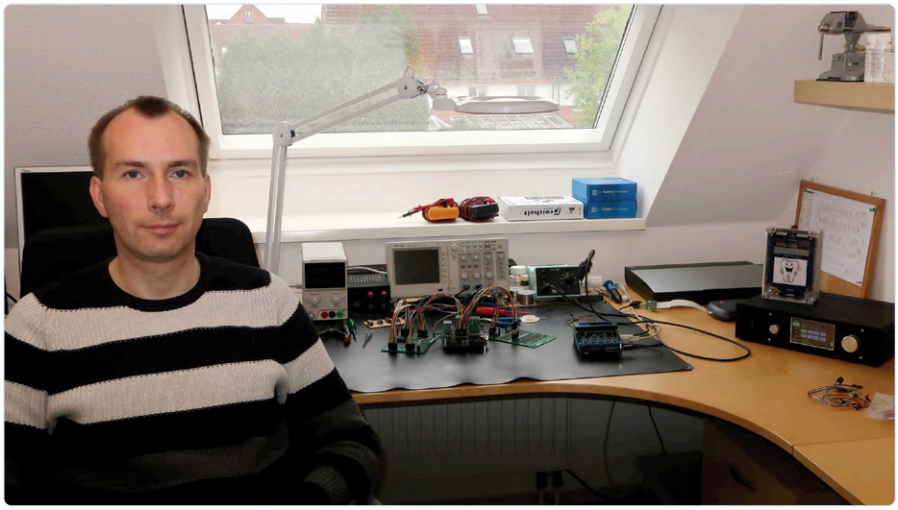


Bild 1. Dennis Kuschel in seinem Heimlabor. Die Einrichtung kann als quasi-spartanisch bezeichnet werden.

Moderne Mikrocontroller ermöglichen es, sehr viel Funktionalität in einem extrem kleinen Volumen unterzubringen - Dinge, für die wir früher einen ganzen Beutel voller Chips brauchten, können jetzt mit einem einzigen IC realisiert werden. Natürlich muss ein solcher Mikrocontroller noch programmiert werden, und das geschieht in der Regel in einer höheren Programmiersprache. Das hat aber den Nachteil, dass der Entwickler den Kontakt zu dem verlieren kann, was in einem solchen Controller auf Port- oder Transistorebene tatsächlich passiert.

Das war auch die Idee von Dennis Kuschel, als er sich anschickte, einen Mikrocontroller diskret mit „losen" logischen ICs zu bauen - eigentlich sollten wir das „Makrocontroller" nennen. Doch lassen wir Dennis zu Wort kommen:

„**Bild 1** zeigt meinen Arbeitsplatz zu Hause. Ich besitze nur wenige Messgeräte - ein Oszilloskop und zwei digitale Multimeter. Das ist alles, was ich brauche, und eigentlich fehlt mir nichts. Auf dem Schreibtisch sehen Sie ein Exemplar des MyNOR-Computers, für den ich zehn ICs durch diskrete Transistorschaltungen (die fröhlich bunten Drahtbrücken) ersetzt habe.

1989 bekam ich zu Weihnachten meinen ersten eigenen Computer: einen Commodore 64. Schon bald wollte ich mehr tun als nur Spiele spielen - ich wollte meine eigenen Programme schreiben und wissen, wie so ein Teil eigentlich funktioniert. Vier Jahre später (ich war damals 17 Jahre alt) baute ich einen neuen Computer mit den Teilen eines kaputten C64, den ich in Assembler programmierte. Während meines Studiums der Elektrotechnik, in den 90er Jahren, kam mir die Idee, eine CPU mit diskreten Logikgattern und ICs zu bauen. Einige Jahre später wurde MyCPU [1] geboren, ein Computer, der aus dutzenden logischen ICs der 74-CMOS-Serie aufgebaut war und auf dem eine C64-Basic-Version läuft.

Auf Computerfestivals und Hobbymessen stieß diese MyCPU auf großes Interesse, aber viele Interessierte schreckten vor einem Nachbau zurück, weil die komplette Schaltung plus Peripherie mal eben rund 1000 € kosten sollte. Deshalb entschied ich mich (nach nunmehr 20 Jahren) zu einem neuen Projekt: einen möglichst einfachen und kostengünstigen Computer, natürlich wieder völlig diskret aufgebaut. Dieser musste zwei Bedingungen erfüllen. Zunächst einmal musste er ohne die ALU (Arithmetic Logic Unit) auskommen - aus dem einfachen Grund, dass der früher weit

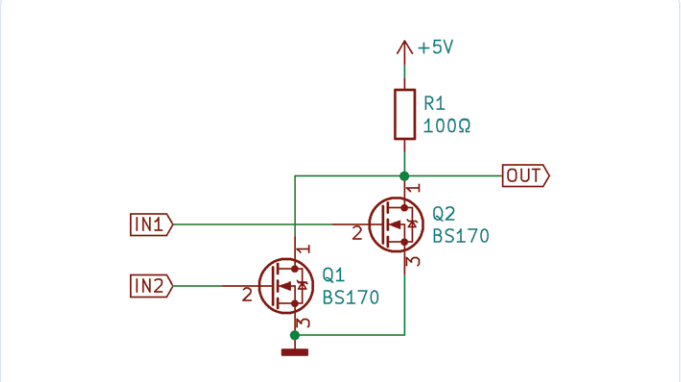


Bild 2. Dies ist das Herzstück des MyNOR-Computers: ein einzelnes, diskret aufgebautes NOR-Gatter.

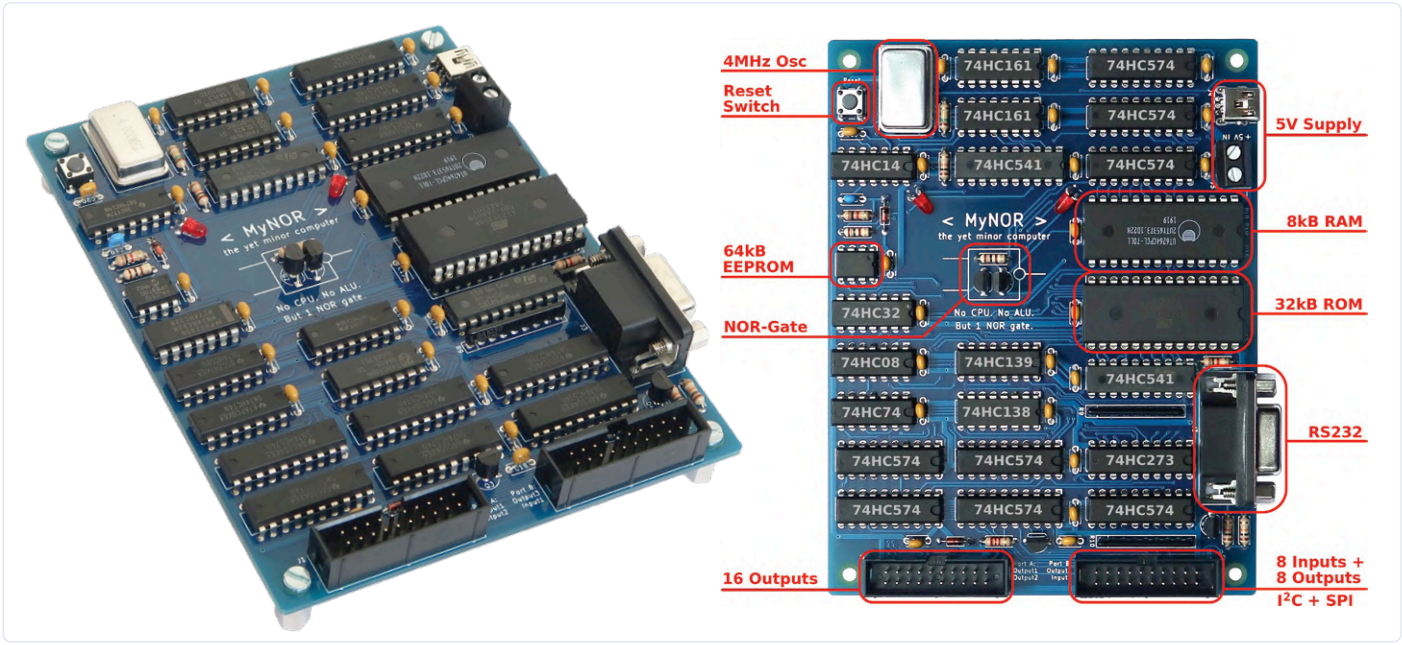


Bild 3. Die endgültige Version von MyNOR. Die Übersicht auf der rechten Seite zeigt, dass auch die notwendige Peripherie vorhanden ist.

verbreitete ALU-Chip 74LC181 nicht mehr erhältlich ist und Alternativen zu kompliziert werden. Und zweitens musste er sich mit einem einzigen programmierbaren Speicher-IC (EPROM) zufrieden geben.

Mein neuer Computer verwendet nur ein einziges NOR-Gatter zum Rechnen. Eine sehr einfache Logikeinheit, die ich da gewählt habe, weil sie sich leicht mit zwei MOSFET-Transistoren und einem Widerstand aufbauen lässt (**Bild 2**). Alle arithmetischen Operationen (wie UND, ODER, EXOR, Addition und Subtraktion) werden von der Software aus vielen einzelnen NOR-Operationen zusammengesetzt. Wegen des zentralen NOR-Gatters habe ich meinen neuen Computer MyNOR [2] getauft. Die Gesamtzahl der Teile ist so gering, dass sie auf eine Platine von 10 x 13 cm passt. **Bild 3** zeigt auf der linken Seite ein Porträt von MyNOR, während das Bild rechts zeigt, was sich wo auf der Platine befindet. Es enthält auch die notwendige Peripherie: ein 64kB-EEPROM als Massespeicher, während acht digitale Eingänge und 24 digitale Ausgänge die Kommunikation mit der Außenwelt sicherstellen. Über diese 32 I/Os werden auch die üblichen Schnittstellen realisiert (softwarebasiert): RS232, I²C und SPI.

Bild 4. Links der Taschenrechner und rechts der MyNOR mit Tastatur und Display.

Mit diesem kleinen Einplatinencomputer kann erstaunlich viel erreicht werden. Ich habe zum Beispiel einen einfachen Taschenrechner in das Betriebssystem eingebaut, der über die RS232-Schnittstelle betrieben werden kann. Dieser Rechner dient hauptsächlich als Beweis dafür, dass Gleitkommaberechnungen mit einem einzigen NOR-Gatter durchgeführt werden können! Darüber hinaus enthält das Betriebssystem ein einfaches Monitorprogramm, mit dem Assembler-Befehle eingegeben werden können. Zum Beispiel kann man eigene Assembler-Programme C64-like direkt in das EEPROM von MyNOR programmieren. Dies geschieht über eine Textdatei mit einem speziell formatierten Binärprogramm. Etwas Geduld ist erforderlich, da die Verbindung bei 2400 Baud nicht gerade atemberaubend schnell ist.

Richtig spannend wird es, wenn dieser Einplatinencomputer auch ohne die Nabelschnur zu einem PC verwendet wird. Zu diesem Zweck habe ich zwei Erweiterungskarten entwickelt. Die erste enthält nur 20 Drucktaster, eine LED, acht 7-Segment-Anzeigen und die zu ihrer Ansteuerung erforderlichen Transistoren. Anzeigen und Drucktaster sind in herkömmlicher Weise gemultiplext. Ich benutze einen auf diese Weise konstruierten „Taschenrechner" jeden Tag bei der Arbeit.

Die zweite Erweiterungskarte macht MyNOR zu einem richtigen kleinen Computer. Neben den üblichen Drucktastern enthält diese Karte auch ein 4x20-LCD, einen kleinen Lautsprecher, eine batteriegepufferte Echtzeituhr und einen Temperatursensor. Wenn MyNOR mit dieser Karte erweitert wird, ergeben sich endlose Möglichkeiten. Ich habe bereits einige Spiele (Minesweeper, Tetris), einen Küchenwecker, eine Musikbox und einen I²C-Bus-Scanner programmiert.

Bild 5. Dieser Zahnputztimer besteht aus MyNOR v1.0 plus der 7-Segment-Platine v1.0. Bei dem Gerät unter dem Timer handelt es sich übrigens um einen High-End-Empfänger (FM und DAB+) mit integriertem 11-Band-Equalizer und Touch-Control, den Dennis im vergangenen Jahr gebaut hat.

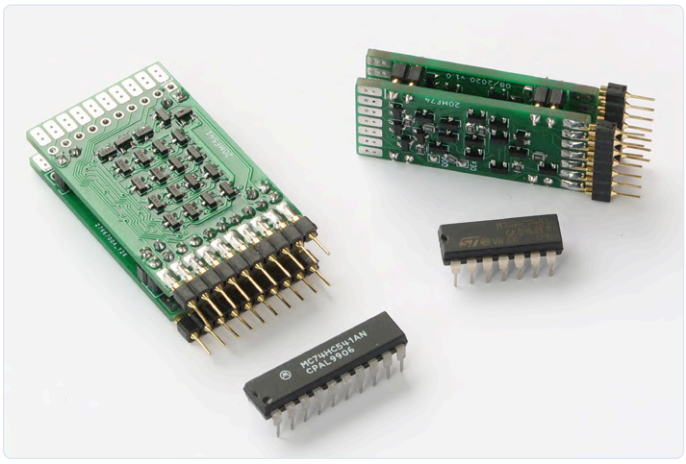


Bild 6. Zwei mit diskreten Transistoren nachgebaute ICs. Die ursprünglichen ICs liegen zum Vergleich daneben (74HC541 und 74HC74).

FORTSETZUNG FOLGT?

Möchten Sie mehr über diese oder andere selbst gebaute CPUs wissen? Bitte teilen Sie uns dies über redaktion@elektor.de mit; wir werden dann später diesem Thema einen ausführlicheren Artikel widmen.

Sie haben Fragen oder Kommentare?
Haben Sie Fragen oder Anmerkungen zu diesem Artikel? Senden Sie eine E-Mail an den Autor oder an die Redaktion von Elektor über redaktion@elektor.de.

Beide Erweiterungskarten sind in **Bild 4** dargestellt.
Die erste Version von MyNOR auf einer echten Platine war noch nicht ganz perfekt. Dennoch hat dieses Exemplar (einschließlich der 7-Segment-Erweiterung) noch lange nicht ausgedient. Ich habe MyNOR v1.0 in einen Zahnputztimer für meine Kinder umfunktioniert (**Bild 5**).
Im Moment arbeite ich an einer noch weitergehende „Diskretisierung" des Computers. Da der gesamte MyNOR-Computer aus 22 ICs besteht, ist es eine beachtliche Herausforderung, die 19 Logik-ICs durch diskrete Transistorschaltungen zu ersetzen. Erste Experimente in dieser Richtung waren aber sehr ermutigend.
Übrigens bin ich nicht der Einzige, der sich mit selbstgebauten CPUs beschäftigt. Viele haben sich im „Homebrew CPU Webring" zusammengeschlossen; ein Besuch der Webseite [3] lohnt sich auf jeden Fall.

200552-03

Über den Autor
Dennis Kuschel kam zum ersten Mal mit Elektronik in Berührung, als er im Alter von sechs Jahren mit einem Klassenkameraden eine Tüte Bonbons gegen ein defektes Transistorradio tauschte. Zur Verblüffung seiner Eltern brachte er das Radio kurze Zeit später wieder zum Laufen. Dennis arbeitet derzeit als Hardware-Entwickler bei einem deutschen Kamerahersteller. Dort entwirft er Schaltungen und 10-lagige Starrflex-HDI-Platinen und programmiert komplexe IP-Kernel für FPGAs. Privat geht er gerne in die andere Richtung: „Keep it simple" ist sein Credo - je weniger Elektronik, desto weniger kann kaputtgehen...

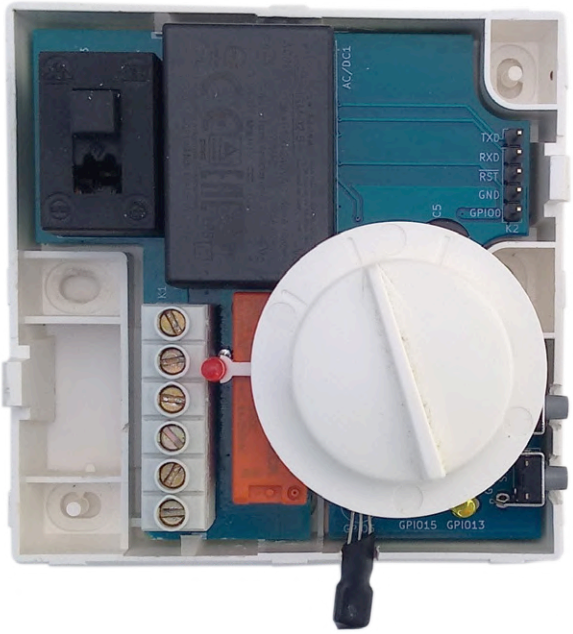
Ein Beitrag von
Text und Bilder: **Dennis Kuschel**
Übersetzung: **Rolf Gerstendorf**
Redaktion: **Eric Bogers**
Entwurf: **Giel Dols**

WEBLINKS

[1] **MyCPU**: http://www.mycpu.eu
[2] **MyNOR**: http://www.mynor.org
[3] **Selbstgebaute CPU**: http://www.homebrewcpuring.org

Ein Thermostat im ESPHome

Hausautomatisierung weiter ausgebaut



Von **Clemens Valens** (Elektor)

Eine richtig ausgeführte Hausautomation ist wie eine unsichtbare Hand, die Sie sanft einen Hügel hinaufschiebt. Wenn sie da ist, macht sie das Leben ein wenig komfortabler; wenn sie nicht da ist, können Sie den Hügel immer noch erklimmen. In diesem Artikel geht es darum, einen Thermostat für ein solches Hausautomationssystem zu entwerfen. Doch ob automatisiert oder nicht, mit seiner traditionellen Benutzeroberfläche haben Sie immer die volle Kontrolle über die Raumtemperatur.

Vor etwa einem Jahr beschloss ich, mich an der Automatisierung meines Heims zu versuchen. Mein erster Meilenstein war die Automatisierung des Thermostats in unserem Wohnzimmer. Dazu ersetzte ich den vorhandenen Wandthermostat durch den WLAN-Thermostat (Elektor-Projekt 160269, veröffentlicht in der Februarausgabe 2018 [1] [2]), den ich mit ESPHome-basierter Firmware umprogrammiert hatte. Die neue Firmware „virtualisierte" alle Bedienelemente der Desktop-variante (in Worten: ein Relais, zwei Drucktasten und drei LEDs), so dass sie von einer Hausautomatisierungssteuerung wie dem Home Assistant bedient werden konnte.
In diesem System spielt der Home Assistant auf einem Raspberry Pi die Rolle des Thermostaten, das heißt, er entscheidet, wann die Heizung ein- oder ausgeschaltet wird. Der Tischthermostat selbst ist zu einem einfachen ferngesteuerten Relais mit einigen LEDs degradiert worden.

Ganz nett, aber...

Das System funktionierte gut und half uns bequem durch den Winter 2019/20, doch brachte die Wintersonne einige Unzulänglichkeiten an den Tag:

- WLAN-Netzwerk erforderlich
- Home Assistant erforderlich
- unästhetisch!

Zukunftssicher? Aber sicher!

Als ich den Thermostat testete, war das erste Problem kein großes Problem, da die meisten von uns ein WLAN im Haus haben. Wenn es nicht mehr funktioniert, wird es einfach durch einen Neustart des Routers (oder was auch immer) wieder in Gang gesetzt. Doch wie vieles im Leben kommen und gehen Technologien, und es gibt keine Garantie, dass es immer ein WLAN geben wird. Unsere Wohnungen werden aber auch in dreißig oder mehr Jahren noch da sein. Mit anderen Worten, ein einigermaßen zukunftssicheres Design wäre wünschenswert.

Rückwärtsgewandt? Auch das!

Der zweite Punkt hängt mit dem ersten zusammen, da auch Home Assistant und Raspberry Pi eines Tages verschwinden könnten. Aber es ist noch schlimmer: Ich weiß immer, wie mit dem Home Assistant auf dem Raspberry Pi umzugehen ist, aber die meisten Leute, die ich kenne, haben keine Ahnung davon. Damit auch andere Leute meinen automatischen Thermostat benutzen können, muss er auch „past-proof" sein. Er sollte wie ein klassischer Wandthermostat aussehen und sich auch wie einer verhalten. Die Automatisierung darf nicht aufgezwungen werden, sondern es sollte eine optionale Funktion bleiben. Die Automatisierung ist für diejenigen, die sie benutzen wollen, aber wer das nicht will, soll nicht davon gehindert werden.

Reine Geschmackssache

Die dritte Frage ist etwas subjektiv. Der automatische Tischthermostat baumelte am Ende eines Drahtstücks, das mit einem Loch in der Wand verbunden war, wo sich der alte Thermostat befand (**Bild 1**). Ein paar unbenutzte Netzkabel, die aus der Wand ragten, wurden mit Isolierbandstücken vor neugierigen Fingern geschützt. Da der Tischthermostat eine 5-V-Stromversorgung über einen USB-Anschluss benötigt, wurde er von einem Telefonladegerät versorgt, das an einer nahe gelegenen Steckdose angeschlossen war. Es gab also mehrere äußerst sichtbare Drähte, während der ursprüngliche Thermostat keine hatte. Obwohl es ein ausgezeichneter Aufhänger für einen Smalltalk war, fanden die meisten Besucher das System nicht unbedingt schön (sondern stattdessen „hässlich", „seltsam" und „gefährlich", Banausen halt).

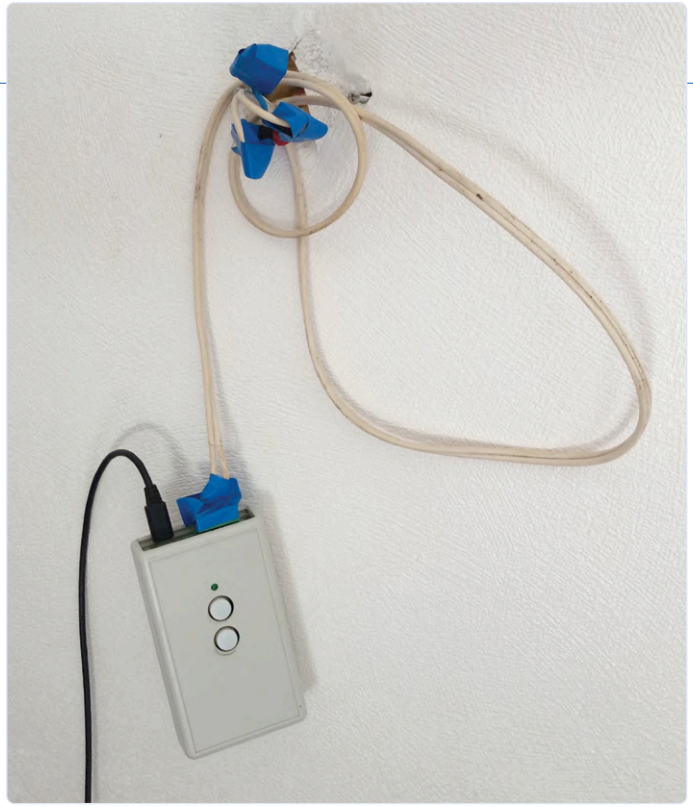


Bild 1. Einige Leute empfanden diese Installation dieses intelligenten Thermostats in einem Wohnzimmer unverständlicherweise als „unästhetisch".

Zurück zum Zeichenbrett

Die Unzulänglichkeiten (und der Spott meiner Freunde) führten zu der Entscheidung, den ursprünglichen Tischthermostat umzugestalten. Dies führte zu folgendem „Pflichtenheft":

- Lokale Einstellung der Soll-Temperatur am Gerät.
- Lokale Steuerung setzt die Automatisierung außer Kraft.
- Netzspannungsbetrieb
- Saubere Anschlüsse und ein professionell wirkendes Gehäuse.

Der zweite Punkt betrifft die Software, die drei anderen haben (nur) etwas mit der Elektronik des neuen Thermostats zu tun.
Pflicht Nr. 1 bedeutet eine intuitive Benutzerschnittstelle zur Einstellung der Soll-Temperatur. Der Tischthermostat verfügt zwar über zwei Drucktasten zur Auf- und Abwärtsregelung der Zieltemperatur, es gibt aber kein Display, das diese Temperatur anzeigen könnte. Da es (nicht unmöglich, aber) recht kompliziert ist, die I/O-Pins für das Display bei einem kleinen WLAN-Modul zur Verfügung zu stellen, scheint ein einfaches Potentiometer mit einer kalibrierten Skala der bessere Weg. Und einen bisher ungenutzten analogen Eingang besitzt das WLAN-Modul dann auch!
Auch die dritte Anforderung will wohlüberlegt sein. Für eine WLAN-Verbindung und zur Versorgung des Relais wird etwa 1 W Leistung benötigt. Der Bau einer kleinen Stromversorgung mit Transformator ist zwar möglich, aber es ist schwierig, sie klein genug zu halten, denn schließlich wollen wir keine riesige Kiste an der Wand. Der ursprüngliche Thermostat besaß ein transformatorloses Netzteil, das für relativ konstante Lasten gut funktioniert, aber bei den variablen Lastströmen einer WLAN-Verbindung vielleicht oder eher wohl nicht. Meine Absicht war es deshalb, ein kleines AC/DC-Wandlermodul zu verwenden.
Nun noch ein geeignetes Gehäuse. In den entsprechenden Katalogen konnte ich nichts Passendes finden, und auch, wenn im Zeitalter von FabLabs, 3D-Druck und Laserschneiden die Produktion eines (!) kundenspezifisches Gehäuses kein unüberwindbares Hindernis darstellt, habe ich es mir noch einfacher gemacht und mich entschlossen, das ursprüngliche Thermostatgehäuse weiterhin verwenden, da es bereits alles enthielt, was ich brauchte: ein Potentiometer mit einer Skala, eine LED und einen Netzschalter. Außerdem besaß es an den richtigen Stellen Befestigungslöcher (ich brauchte keine neuen Löcher in die Wand bohren) und stellte eine gute Möglichkeit dar, eine Platine durch den Gehäuseboden hindurch an die Netzspannung anzuschließen.
Meine Modifikationen liefen also darauf hinaus, den neu gestalteten Desktop-Thermostat so auf eine Platine zu packen, dass alles in das bestehende Gehäuse passte, wobei das Potentiometer, die LED, der Netzschalter und der Netzstecker in genau den gleichen Positionen wie beim ursprünglichen Gerät angeordnet sein mussten.
Die Änderungen der Schaltung des Tischthermostats war einfach genug (**Bild 2**). Ich ersetzte die USB-Stromversorgung durch ein 5-V-AC/DC-Modul und fügte ein Potentiometer hinzu. Ein Widerstand zur Spannungsbegrenzung war ebenfalls nötig, da das WLAN-Modul

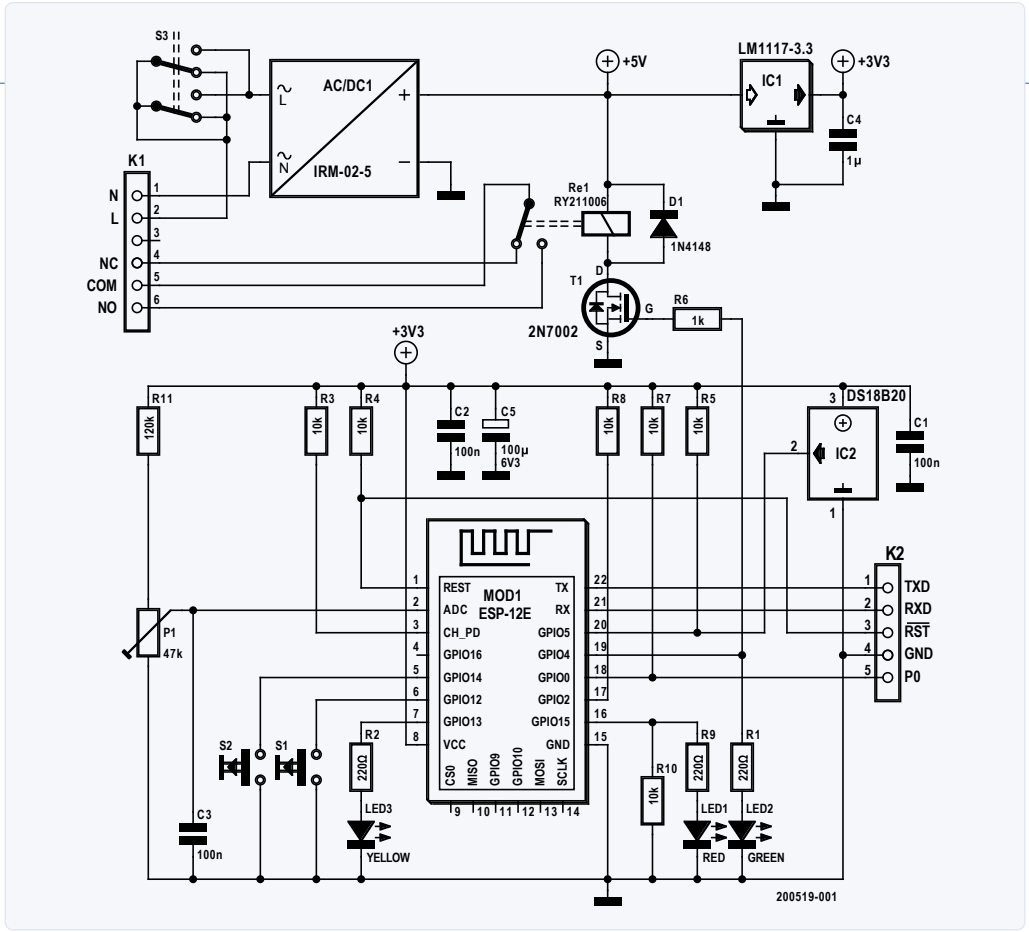


Bild 2. Der neue, diskret angeschlossene Thermostat ist im Grunde genommen der Elektor-Tischthermostat von 2018 mit einem neuen Netzteil. Außerdem wurde ein Potentiometer zur Einstellung der Zieltemperatur hinzugefügt.

keine Spannungen über 1,1 V verarbeiten kann. Die beiden Drucktasten und die drei LEDs blieben erhalten, da sie sich irgendwann als nützlich erweisen könnten.
Es war ein glücklicher Zufall, dass das AC/DC-Modul gerade so klein war, dass es unter den großen Plastikknopf des Potis passte. Poti, Netzschalter und Klemmenleiste wurden aus dem alten Thermostat recycelt (**Bild 3**). Ich musste das alte 48-V-Relais durch eine 5-V-Version ersetzen. Das Relais aus dem Tischthermostaten wollte partout nicht passen, aber zum Glück war das Relais des alten Thermostaten Mitglied einer altehrwürdigen Industriestandardfamilie, die immer noch erhältlich ist, auch in einer 5-V-Ausführung.

Alles auf eine Platine auzuordnen, die in das Originalgehäuse passte, erforderte eine Menge Arbeit mit der Schieblehre, aber am Ende gelang es. Alle SMDs einschließlich des WLAN-Moduls wurden auf der Unterseite der Platine untergebracht (**Bild 4**), während alle bedrahteten Bauteile auf der Oberseite Platz fanden. Ein wenig zusätzliche Platinenfläche wurde dadurch gewonnen, dass einige ungenutzte Montagelöcher des Gehäuses verdeckt und einige hinderliche Kunststoffteile im Inneren des Gehäuses kurzerhand abgeschnitten wurden. Um alle Leiterbahnen zu routen, war ein „flexibler Ansatz" der empfohlenen Isolationsvorgaben leider unvermeidlich.

Software

Die ursprüngliche ESPHome-Firmware musste ebenfalls überdacht werden. Anstatt einfach alle Sensoren und Aktuatoren des Thermostaten freizulegen und den Home Assistant den Rest erledigen zu lassen, musste ich nun Automatisierungen innerhalb von ESPHome vornehmen. Die Programmierung und Konfiguration erfolgt in der YAML-Datei des ESPHome-Projekts des Thermostats (siehe *Hausautomation leicht gemacht* [3]).

Raumtemperatur messen

Für den eingesetzten Raumtemperatursensor DS18B20 (ursprünglich von Dallas, jetzt Maxim und sogar Analog Devices) an GPIO-Pin 5 besitzt ESPHome eine spezielle Dallas-Komponente. Daraus ergibt sich folgender Eintrag:

```
dallas:
  - pin: GPIO5

sensor:
  - platform: dallas
    address: 0x6D00000C24013928
    name: "Measured temperature"
    id: t_room
    filters:
      - offset: 0.0
```

Die erste Zeile veranlasst ESPHome, sein Dallas-1-Draht-Kommunikationsmodul einzubeziehen, die zweite, es an GPIO5 anzuschließen. Der Sensor ist von der `dallas`-Plattform, `address` ist optional. Wenn Sie sie die eindeutige Adresse angeben, muss sie natürlich korrekt sein. Sie können sie im ESPHome-Log nachschlagen (verwenden Sie nicht meine!). Die Angabe der `id: t_room` ist hier erforderlich, da wir an anderer Stelle der YAML-Datei auf sie verweisen müssen (siehe unten). Mit den `filters` könnte man die gemessenen Werte in irgendeiner Weise korrigieren.

Soll-Temperatur

Die Soll-Temperatur wird mit dem Potentiometer eingestellt. Dazu wird in den `sensor`-Bereich der YAML-Datei ein Sensor des Plattform-Typs `adc` mit einem Spannungswert definiert. Mit den so genannten Sensor-

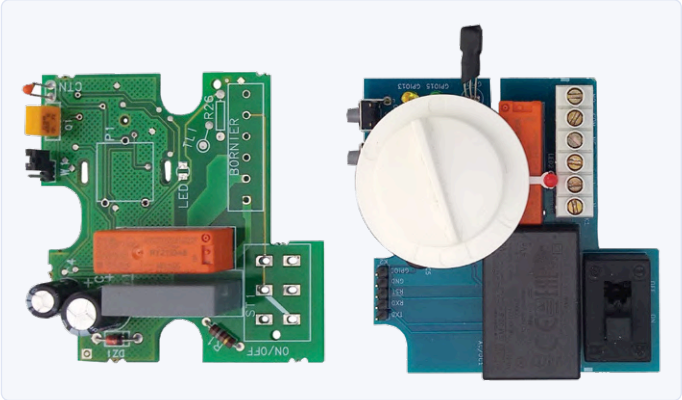


Bild 3. Einige Teile des alten Thermostats wurden in der neuen Konstruktion wiederverwendet. Dazu gehörte das Gehäuse, was die seltsame Form der neuen Platine erklärt. Es brachte auch Einschränkungen bei der Positionierung bestimmter Bauteile mit sich.

Bild 5. Dank einer geringfügigen Anpassung des alten Thermostatgehäuses (Abschneiden von etwas Kunststoff) passt die neue Platine gut.

filtern wird die Sensorspannung in den auf das Gehäuse gedruckten Temperaturwert umgerechnet. Die Gleichung $T_{target} = 25 \cdot V_{in} + 6.75$ passt in meinem Fall recht gut. Daraus ergibt sich ein `multiply`-Filter mit dem Wert 25 und ein `offset`-Filter mit dem Wert 6,75. Wenn man die Einheiten in `°C` angibt, behandelt Home Assistant diese Daten automatisch als Temperatur.

```
- platform: adc
    name: "Target temperature"
    id: t_target
    icon: "mdi:temperature-celsius"
    pin: A0
    update_interval: 5s
    # Convert potentiometer scale to °C (min=6.75°C,
max=31.75°C)
    filters:
      - multiply: 25.0
      - offset: 6.75
    unit_of_measurement: "°C"
    on_value:
      then:
      - lambda: |-
          auto call = id(t_controller).make_call();
          call.set_target_temperature(x);
          call.perform();
```

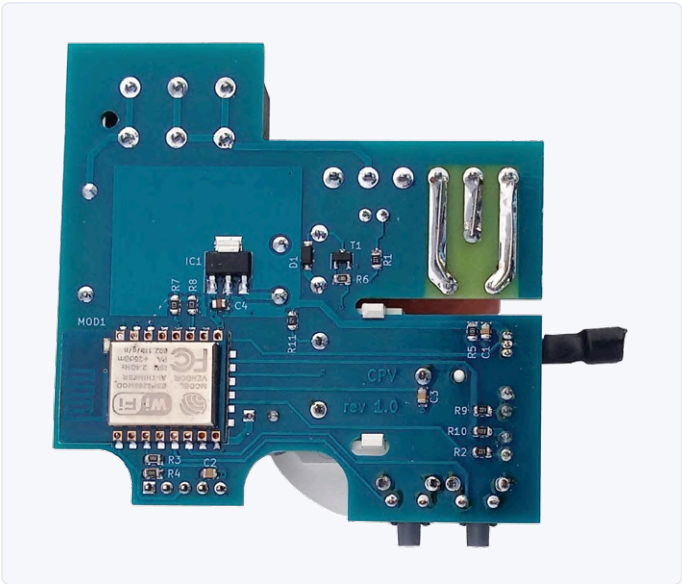


Bild 4. Alle SMDs sind auf der Rückseite der Platine montiert. Die mit der Heizung verbundenen Leiterbahnen (rechte obere Ecke) wurden mit zusätzlichem Lot verstärkt, damit sie große Ströme ohne Überhitzung leiten können. Der Schlitz rechts darunter sorgt für eine galvanische Trennung zwischen den Hoch- und Niederspannungsteilen. Das schwarze Gebilde, das rechts herausragt, ist der Temperatursensor, der durch einen Schrumpfschlauch geschützt ist. Alle Wärme erzeugenden Teile befinden sich auf der linken Seite der Platine und sollten nach dem Einbau des Thermostats nach oben zeigen.

Bild 6. Mission accomplished! Die beiden Druckknöpfe sind durch zwei Ausschnitte (links) zugänglich und der Sensor ragt aus dem Boden heraus. Die grüne und die gelbe LED sind nur durch die Lüftungsschlitze sichtbar, um eine Illumination des Raumes bei Nacht zu vermeiden. Durch die unterschiedlichen Farben ist aber leicht zu erkennen, welche eingeschaltet ist.

Der Teil nach `on_value:` ist eine Automatisierung und wird weiter unten erläutert. Zuerst sehen wir uns das Relais an.

Schalten der Heizung

Dies ist ziemlich einfach, da das Relais nur ein Schalter an GPIO 4 und daher Teil der `gpio`-Plattform ist. Wie der Temperatursensor benötigt es eine `Id (heater)`, um für andere Teile der YAML-Datei zugänglich zu sein.

```
switch:
   - platform: gpio
     pin: GPIO4
     name: "Heater"
     id: heater
```

Volle Klimakontrolle

Wir haben bisher einen Sensor für die Raumtemperatur (`t_room`), ein Poti für die Soll-Temperatur (`t_target`) und ein Relais, um die Heizung ein- und auszuschalten (`heater`).
ESPHome verfügt über eine `climate`-Komponente zur Steuerung von Heiz- und Kühlgeräten. Ein Thermostat ist eine solche Klimakomponente. Dies erspart einiges an Arbeit und stellt zudem ein schönes grafisches Steuerungs-Widget in der Benutzerschnittstelle des Home Assistant zur Verfügung.

```
climate:
   - platform: thermostat
     name: "Thermostat"
     id: t_controller
     sensor: t_room
     default_target_temperature_low: 20 °C
     heat_action:
       - switch.turn_on: heater
     idle_action:
       - switch.turn_off: heater
     hysteresis: 0.5
     away_config:
       default_target_temperature_low: 15 °C
```

Der Thermostat kühlt nicht, sondern kann nur heizen, was es zum Mitglied der ESPHome-`thermostat`-Plattform macht. Es benötigt eine `id`, damit es innerhalb der YAML-Datei gesteuert werden kann. Ich habe die `id` auf `t_controller` gesetzt.

Die Klimakomponente besitzt einen Eingang für den Temperatursensor, den wir mit `t_room` verbinden. Sie muss auch an eine Heizung angeschlossen werden, was wir dank des Heizungsschalters, den wir zuvor definiert haben, tun können.

Man könnte erwarten, dass auch eine Klimakomponente einen Eingang für die Soll-Temperatur besitzt. Dies ist aber nicht der Fall. Vielleicht wird dies ja in einer zukünftigen Version von ESPHome hinzugefügt? Stattdessen gibt es eine Steuerung der Soll-Temperatur, ähnlich wie das Poti des Thermostats. Glücklicherweise gibt es einen Weg, diese Beschränkung zu umgehen, und zwar die so genannten Lambda-Blöcke.

Lambda-Blöcke

Das Konzept der Lambda-Blöcke ist großartig und schrecklich zugleich. Es ist großartig, weil man mit ihnen (fast) alles machen kann, was man will, und sie sind schrecklich, weil sie dem ganzen Konzept widersprechen, Geräte mit einer einfachen YAML-Datei zu konfigurieren, ohne dass man C++-Kenntnisse haben müsste.

Einfach ausgedrückt: Ein Lambda-Block ist C++-Code, der in das ESPHome-Projekt eingebaut wird. Um zu einer brauchbaren YAML-zu-C++-Schnittstelle zu gelangen, mussten die Entwickler einiges an Verrenkungen durchführen, mit dem Ergebnis, dass Lambda-Code komplizierter ausfällt, als es bei normalem C++ der Fall gewesen wäre. Eigentlich sollten Sie darüber nachdenken, statt Lambda-Blöcke eine eigene Komponente zu erstellen. ESPHome unterstützt benutzerdefinierte Komponenten für fast alles. Ich war kurz davor, aber schließlich entschied ich mich, dieses Thema für einen weiteren Artikel aufzusparen.

Die C++-Schnittstelle der Klimakomponente besitzt eine Funktion zur Einstellung der Soll-Temperatur. Der Lambda-Block im `on_value`-Abschnitt des Potentiometersensors zeigt, wie sie zu verwenden ist. Jedes Mal, wenn ein neuer Wert verfügbar ist, wird die Methode `set_target_temperature` der Klimakomponente `t_controller` aufgerufen:

```
    on_value:
       then:
       - lambda: |-
           auto call = id(t_controller).make_call();
           call.set_target_temperature(x);
           call.perform();
```

In normalem C/C++ wäre dies in etwa so (einfacher) gewesen:

```
t_controller.set_target_temperature(x);
```

Hätte die Klimakomponente einen Eingang für die Soll-Temperatur gehabt, hätten die Dinge schön einfach sein können, wie zum Beispiel:

```
climate:
   - platform: thermostat
     name: "Thermostat"
     id: t_controller
     sensor: t_room
     target: t_target
     …
```

Hätte hätte, Sie wissen schon…

Leider ist diese Art der Automatisierung von `on_value` zu einfach, da sie die Fernsteuerung der Soll-Temperatur vom Home Assistant außer Kraft setzt. Um dies zu lösen, sollte die Automatisierung nur dann laufen dürfen, wenn sich `t_target` ändert, also jemand am Poti dreht. Da ESPHome keinen `on_value_changed`-Mechanismus für Sensoren kennt, können wir dies im Lambda-Block erledigen, in etwa so:

```
    on_value:
       then:
       - lambda: |-
           static float x_last = 0.0;
           if (x<x_last-0.1 || x>x_last+0.1)
           {
             x_last = x;
```

```
    auto call = id(t_controller).
                make_call();
    call.set_target_temperature(x);
    call.perform();
  }
```

Die kleine Hysterese x von ±0,1 verhindert zu schnelles Umschalten aufgrund des Sensorrauschens.

Das Delta-Filter

Eine andere und schönere Lösung ist die Verwendung des Delta-Filters auf `t_target`. Dieses Filter lässt einen neuen Wert nur dann passieren, wenn er sich vom vorherigen Wert um einen Betrag von ±Delta unterscheidet. Wenn also delta = 1 und der letzte Wert gleich 20 war, dann wird der nächste Wert nur dann durchgelassen, wenn er entweder kleiner als 19 oder größer als 21 ist.

```
- platform: adc
  name: "Target temperature"
  id: t_target
  …
  filters:
    - multiply: 25.0
    - offset: 6.75
    - delta: 0.2
  …
```

Die Filter werden in der Reihenfolge ausgeführt, in der sie erscheinen: Das Deltafilter arbeitet mit dem in Grad Celsius umgerechneten Wert und nicht direkt mit der Eingangsspannung. Der Deltawert sollte klein sein, da es sonst schwierig ist, die Thermostatschwelle nur ein wenig nach oben oder unten zu schieben, was den ganzen Unterschied im Benutzerkomfort ausmacht.

Fertigstellung des Geräts

Ist die YAML-Konfigurationsdatei fertig, kann die Firmware auf das WLAN-Modul hochgeladen werden. Bei einem Modul, auf dem noch keine ESPHome-kompatible Software läuft, muss dazu die serielle Schnittstelle verwendet werden (verfügbar auf Header K2), wie es [3] genau beschreibt. Sobald das Modul ESPHome mit aktivierter Over-the-Air-Programmierung (OTA) ausführt (wenn die YAML-Datei die Zeile `ota:` enthält), kann der Thermostat umprogrammiert werden, ohne physisch mit dem Entwicklungssystem verbunden zu sein.

Der Thermostat sollte so montiert werden, dass der Temperaturfühler nicht durch das WLAN-Modul und die Stromversorgung aufgeheizt wird.

Fazit

Der Artikel zeigt, wie ein vorhandener „dummer" Thermostat durch einen selbstgebauten, intelligenten und vernetzten Thermostat als Teil der Hausautomatisierung ersetzt werden kann. Es war nicht allzu schwierig, einen Prototyp zu schustern, der in etwa das tut, was er tun soll, und von jedem jederzeit benutzt werden kann. Damit er den ästhetischen Ansprüchen meiner Freund:innen genügt, war etwas mehr Aufwand nötig. Zweifellos wird das neue Thermostat noch etwas Feinabstimmung benötigen, aber da er fernprogrammiert werden kann (in-the-field, wie man sagt), ist dies einfach zu bewerkstelligen.

200519-03

Sie haben Fragen oder Kommentare?
Haben Sie Fragen oder Anmerkungen zu dem Artikel? Schicken Sie eine E-Mail an den Autor unter clemens.valens@elektor.com oder kontaktieren Sie Elektor unter editor@elektor.com.

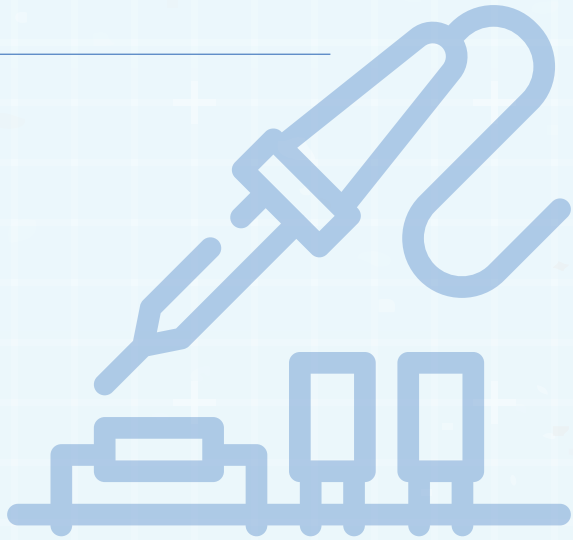
Ein Beitrag von
Idee, Entwicklung, Text und Fotografien: **Clemens Valens**
Schaltplan: **Patrick Wielders**
Redaktion: **C. J. Abate**
Übersetzung: **Rolf Gerstendorf**
Gestaltung: **Giel Dols**

WEBLINKS

[1] **R. Aarts & C. Valens, „WLAN-Thermostat", Elektor Februar 2018**: www.elektormagazine.de/magazine/elektor-201801/41213
[2] **C. Valens, „Wi-Fi controlled thermostat / timer / switch", Elektor Labs 2018**: http://bit.ly/wifi-thermostat
[3] **C. Valens, „Hausautomation leicht gemacht", Elektor September 2020**: www.elektormagazine.de/magazine/elektor-154/58936

Von Entwicklern für Entwickler

Tipps & Tricks, Best Practices und andere nützliche Infos



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

High-Side- oder Low-Side-Schalten?

In High-Side-Systemen wird der Schalter zwischen die positive Versorgungsleitung und die Last eingefügt, bei einer der Low-Side-Schaltung dagegen wird die Last mit Masse verbunden (**Bild 1**). Die Funktionsprinzipien beider Schaltungsvarianten sind leicht verständlich, aber welche Methode ist vorzugsweise anzuwenden? Es hängt wie so oft von der Anwendung ab!

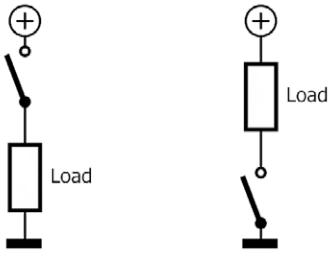


Bild 1. Schalten Sie eine Last gegen die Versorgungsleitung (links) oder gegen Masse (rechts)?

Vermeiden Sie gefährliche Situationen!

Das High-Side-Schalten (**Bild 2**) ist die bevorzugte Schalttechnik in Situationen, in denen Kurzschlüsse nach Masse wahrscheinlicher sind als Kurzschlüsse zur positiven Versorgungsleitung. Denken Sie zum Beispiel an Autos oder Maschinen, bei denen der größte Teil der Struktur mit Masse verbunden ist. In solchen Fällen ist es sicherer, die Last vom Pluspol der Batterie als von der Masse zu trennen. Außerdem führt dies in feuchten Umgebungen normalerweise zu weniger Korrosion bei Steckern, da die Last im ausgeschalteten Zustand keine Spannung führt.

Leistung besser schalten mit N-Transistoren

Da N-Transistoren im Allgemeinen mehr Strom führen können ihre P-Brüder, sind sie zum Schalten schwerer Lasten vorzuziehen. Das Low-Side-Schalten mit N-Transistoren ist einfacher als das High-Side-Schalten und kann oft durch Mikrocontroller-Ports erfolgen, ohne dass spezielle Treiber erforderlich sind.
High-Side-Schalten mit N-Transistoren ist zwar auch möglich, erfordert jedoch eine Steuerspannung, die höher ist als die an Source oder Emitter angeschlossene (Last-)Spannung. Um das Gate oder die Basis des Transistors über seine Source- oder Emitterspannung zu ziehen, ist eine Art Ladungspumpe oder eine zusätzliche Versorgung erforderlich. Dies verkompliziert den Schaltungsentwurf, macht ihn teurer und erhöht auch die Empfindlichkeit gegenüber Rauschen und Interferenzen.
Die Ansteuerung eines solchen High-Side-Schalters mit einem PWM-Signal zur Steuerung zum Beispiel der Geschwindigkeit eines

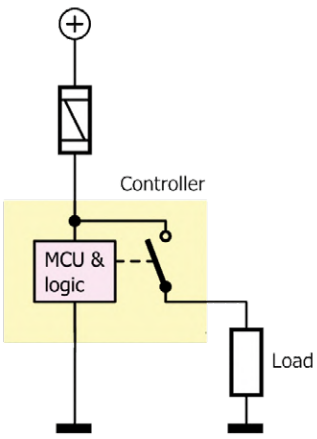


Bild 2. Die High-Side-Schaltung wird bevorzugt, wenn ein Kurzschlusses nach Masse wahrscheinlicher ist als ein Kurzschluss zur Versorgung.

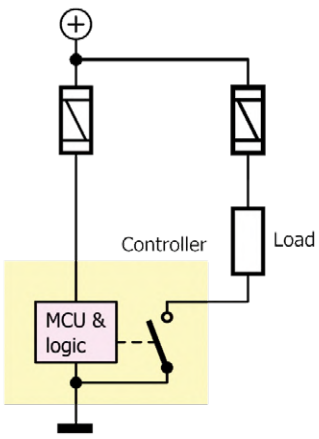


Bild 3. Low-Side-Switching ist billig und einfach. Wenn jedoch eine Sicherung hinzugefügt werden muss, weil der Abstand zwischen der Last und dem Controller zu groß ist (rechts), kann diese Konfiguration teurer werden als eine High-Side-Schaltung.

Motors oder der Helligkeit einer LED kann wegen der Ladungspumpe problematisch sein.

Eine Sicherung macht den Unterschied

Daher ist Low-Side-Switching tendenziell billiger als High-Side-Switching. Wenn jedoch die Last und ihr Controller sich nicht nahe sind, kann eine effiziente Systemverkabelung statt nur einer Sicherung, die beides schützt, separate Sicherungen für die Last und den Controller erfordern (**Bild 3**). Für eine High-Side-Schaltung wäre auch in diesem Fall nur eine einzige Sicherung erforderlich. Dies mag wenig wichtig erscheinen, aber wenn man die Verkabelung und den Arbeitsaufwand berücksichtigt, der erforderlich ist, um eine Sicherung in einem Sicherungskasten zugänglich zu machen, können getrennte Sicherungen den Kostenvorteil der Low-Side-Schaltung leicht zunichte machen.

Erdung gut - alles gut

Die Low-Side-Schaltung hat einen Erdungsanschluss sowohl für die Last als auch für den Controller. Dadurch werden Massepotenzial-Differenzen zwischen den beiden vermieden, wenn die Ströme hoch sind und Controller und Last weit voneinander entfernt sind. Die Low-Side-Schaltung ist daher robuster gegenüber solchen Massestörungen als die High-Side-Schaltung.
Auf der anderen Seite fügt ein Schalter, wenn er geschlossen ist, einen kleinen Widerstand in Reihe mit der Last ein. Dies führt zu einem kleinen Spannungsabfall, was bedeutet, dass die Masse der Last etwas über der Masse des Controllers liegt.

Was muss ich also tun?

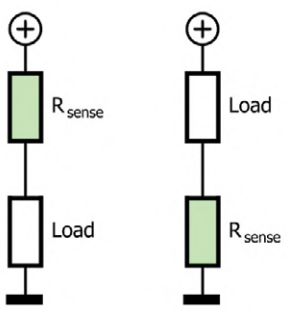
In Fällen, in denen eine (schwere) Last nur ein- oder ausgeschaltet werden muss, ist die High-Side-Schaltung die bevorzugte Methode. Wenn die Leistung einer Last durch (relativ) schnelle PWM-Signale gesteuert werden muss, beispielsweise in einem Beleuchtungs- oder Heizsystem, ist die Low-Side-Schaltung zu empfehlen.
Und dann gibt es halbe H-Brücken, die sowohl einen High-Side-Schalter als auch einen Low-Side-Schalter erfordern ... und PWM. Vergewissern Sie sich daher wie immer, bevor Sie sich für eine Technik entscheiden, ob sie für Ihre Anwendung geeignet ist.

190369-F-03

Sie haben Fragen oder Kommentare?
Haben Sie Fragen oder Anmerkungen zu dem Artikel? Schicken Sie uns eine E-Mail (clemens.valens@elektor.com) oder kontaktieren Sie die Elektor-Redaktion unter editor@elektor.com.

HIGH-SIDE- VERSUS LOW-SIDE-STROMMESSUNG?

Ein ähnliches Problem ergibt sich bei der Strommessung. Es gibt verschiedene Techniken zur Erfassung von Strömen in einer Schaltung, aber die wahrscheinlich häufigste besteht darin, einen kleinen Widerstand mit einem bekannten Wert in Reihe mit der Last zu schalten. Da der Strom durch die Last und den Reihenwiderstand derselbe ist, lässt sich der Strom nach dem Ohmschen Gesetz berechnen, indem zunächst die Spannung über dem Reihenwiderstand gemessen und dann durch den Wert des Widerstandes geteilt wird. Die Frage, die sich nun stellt, lautet: Wo soll dieser Widerstand angeschlossen werden? Vor oder nach der Last? High-Side oder Low-Side?



High-side versus low-side current sensing.

Low-Side-Strommessung ist preiswert
Im Fall der Low-Side-Strommessung wird die typischerweise kleine Spannung über dem Messwiderstand auf Masse bezogen und kann mit kostengünstigen Niederspannungs-Operationsverstärkern verstärkt werden, bevor sie digitalisiert und von einem Mikrocontroller weiterverarbeitet wird. Wie der Schalter bei der Low-Side-Lastschaltung führt der Messwiderstand jedoch einen kleinen Spannungsabfall ein, der die Last von der Systemmasse anhebt, was zu Rauschen und Masseproblemen führen kann.
Außerdem kann die Low-Side-Strommessung nicht erkennen, ob die Last gegen Masse intern oder an einem ihrer Terminals kurzgeschlossen ist.

Hohe Gleichtaktspannungen handhaben
Die High-Side-Strommessung leidet nicht unter diesen Mängeln, aber sie hat ihre eigenen Probleme. Die Spannungsdifferenz über dem Strommesswiderstand ist zum Beispiel nicht auf Masse bezogen. Eine erneute Referenzierung auf Masse erfordert eine zusätzliche Schaltung, wodurch das System teurer und komplexer wird.
Darüber hinaus muss der Verstärker in der Lage sein, eine Gleichtaktspannung bis zur positiven Versorgungsspannung (und sogar noch höher bei Transienten) zu verkraften, die bei bestimmten Anwendungen Hunderte von Volt betragen

kann. Er muss entweder eine hohe Versorgungsspannung unterstützen oder Eingänge haben, die mit solch hohen Gleichtaktspannungen umgehen können. Um Gleichtaktspannungsfehler gering zu halten, müssen außerdem teure Präzisionsbauteile verwendet werden.
Um dem Schaltungsentwickler das Leben ein wenig leichter zu machen, haben mehrere Halbleiterhersteller dedizierte High-Side-Stromerfassungs-ICs in ihr Angebot aufgenommen.

Schlussfolgerung?
Welche Strommessmethode zu wählen ist, hängt (wiederum) von der Anwendung ab. Die High-Side-Strommessung kann erkennen, ob die Last kurzgeschlossen oder offen ist, wobei die Last auf Masse bezogen bleibt. Aufgrund der Gleichtaktanforderungen ist sie jedoch komplexer und kostspieliger als die Low-Side-Strommessung. Die Entscheidung für ein dediziertes Strommess-IC könnte der beste Weg sein.

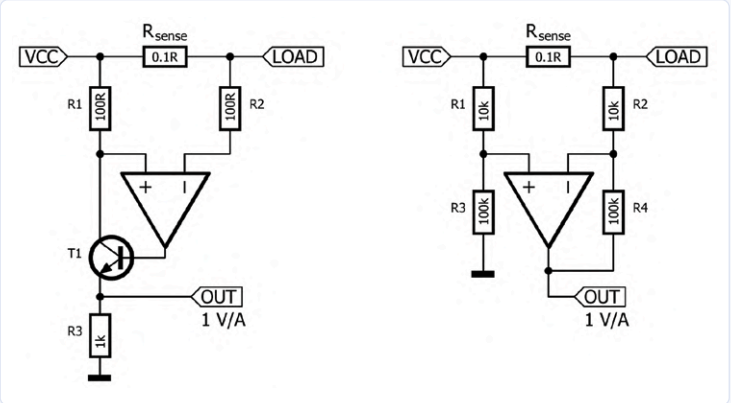
ZWEI METHODEN ZUR HIGH-SIDE-STROMMESSUNG

Es gibt mehrere Methoden, um High-Side-Strommessungen durchzuführen. Der hier gezeigte Strommessverstärker und der Differenzverstärker sind zwei davon.

Grundsätzlich teilt der Strommessverstärker (links) den Strom durch R_{sense} durch einen Faktor von $R1/R_{sense}$ (= 1.000) und zwingt ihn durch R3. Die Spannung über R3 ist daher proportional zum Strom durch R_{sense}. Ein Strom von 1 A durch R_{sense} erzeugt einen Strom von 1 mA durch R3 (1 kΩ), was zu einer Spannung von 1 V über R3 führt.

Der Differenzverstärker (rechts) multipliziert die Spannung über R_{sense} mit R4/R2 (= 10). Ein Strom von 1 A durch R_{sense} erzeugt eine Spannung von 0,1 V über R_{sense}, die, um den Faktor 10 verstärkt, 1 V am Ausgang des Operationsverstärkers ergibt.

Beide Schaltungen erzeugen zwar die gleiche Ausgangsspannung, sind aber nicht gleichwertig.



Bandbreite
Erstens ist die Bandbreite der beiden Schaltkreise nicht dieselbe. Der Differenzverstärker hat in der Regel eine (viel) geringere Bandbreite als der Strommessverstärker und eignet sich daher besser zur Messung des „durchschnittlichen" Stroms, der durch eine Last fließt. Der Strommessverstärker hingegen ist schnell und kann den momentanen Laststrom bei hohen Frequenzen messen.

CMR
Ein weiterer wichtiger Faktor, den es zu berücksichtigen gilt, ist die Gleichtaktunterdrückung (Common Mode Rejection, CMR). Da die Gleichtaktspannung an den Eingängen hoch ist, führt selbst eine kleine Asymmetrie zwischen den beiden Eingängen zu einem Fehler am Ausgang des Verstärkers. Der Differenzverstärker benötigt daher hochpräzise Widerstände, um den CMR-Fehler so gering wie möglich zu halten. Im Stromsensor-Verstärker ist es hauptsächlich der Operationsverstärker, der den CMR-Fehler bestimmt, ein Parameter, für den der Hersteller des ICs verantwortlich ist.

Robustheit
Auf der anderen Seite kann ein Differenzverstärker dank seiner externen Widerstände leichter an sehr hohe Gleichtaktspannungen angepasst werden als der Strommessverstärker. Damit sind wir beim Thema Robustheit der Strommessschaltung angelangt. Wenn Transienten vorhanden sein können, die über den Sensorwiderstand unzulässige Spannungen erzeugen würden, schützen die hohen Widerstände den Differenzverstärker.

In solchen Fällen muss der Strommessverstärker hauptsächlich auf die im Operationsverstärker eingebaute Eingangsüberspannungsschutzschaltung zurückgreifen. In ähnlicher Weise ist in der Regel ein Differenzverstärker auch im Falle einer invertierten Versorgung robuster.

Stromaufnahme
Schließlich gibt es noch die Stromaufnahme zu beachten, besonders wichtig bei (Ultra-)Low-Power-Anwendungen. Wenn der Differenzverstärker direkt an die Stromversorgung angeschlossen wird, „zieht" er wegen seiner Widerstände immer ein wenig Strom, auch wenn der Operationsverstärker selbst nicht mit Strom versorgt wird. Der Strommessverstärker weist eine viel höhere Eingangsimpedanz auf, was dazu beiträgt, dass Batterien länger halten.

Spezielle ICs
Wegen all dieser Feinheiten verkaufen Halbleiterhersteller spezielle Strommess-ICs, die auf unterschiedlichen Topologien basieren, wodurch es für die Entwickler viel einfacher wird, die Strommessung in ihre Anwendungen zu integrieren.

Das Dekatron

Bemerkenswerte Bauteile

Von **Neil Gruending** (Kanada)

Heute sind integrierte Schaltkreise für uns selbstverständlich, aber sie waren nicht immer so leicht verfügbar. In der Ära der Vakuumröhren wurden die zunehmende Komplexität der Schaltungen und die Anzahl der erforderlichen Röhren zu einem echten Problem, so dass die Unternehmen nach Möglichkeiten suchten, komplexe Schaltungen in einer einzigen Röhre zu realisieren. Ein Beispiel dafür ist der geniale Polykathodenzähler, besser bekannt unter Ericssons Handelsnamen Dekatron.

Bild 2. Foto einer Dekatron-Röhre mit seinen 30 Stiften [3].

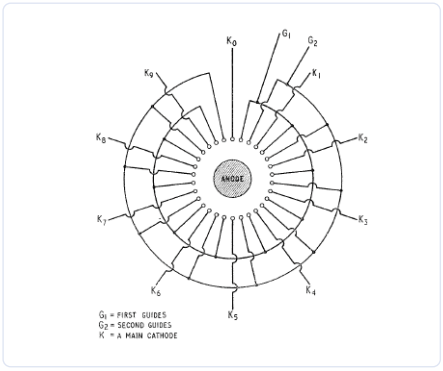


Bild 3. Schaltskizze einer Dekatron mit Gates zum Aufwärts- und Abwärtszählen [1].

Normalerweise würde für jeden Schritt in einem Zähler eine Röhre benötigt, so dass für das Zählen bis 10 zehn Röhren erforderlich gewesen wären. Das Dekatron (**Bild 1**) wurde entwickelt, um all diese Röhren durch nur eine einzige zu ersetzen und gleichzeitig die Möglichkeit zu bieten, aufwärts und abwärts zu zählen. Aber das Interessanteste daran ist ihre Konstruktion. Sie besitzt eine zentrale Anode, die symmetrisch von 30 kleinen, kreisförmig angeordneten Stäbchen umgeben ist, wie in **Bild 2** zu sehen. Jede „Zählerstand-Kathode“ (Kx) ist von zwei Transferelektroden umgeben, die an G_1 und G_2 miteinander verbunden sind.

Normalerweise sind diese Zählerstand-Kathoden geerdet, so dass beim Anlegen von 400 V an die Anode eine beliebige Zählkathode zufällig zündet und zu glühen beginnt. Sobald dies geschieht, wird das Spannungspotential zwischen der Anode und den Kathoden reduziert und die anderen Kathoden bleiben ausgeschaltet. Ein Zählvorgang wird durchgeführt, indem zunächst ein negativer Impuls an eine der Transferelektroden angelegt wird, der deren Zündspannung reduziert. Ist die Zündspannung ausreichend reduziert, zündet sie und schaltet die Zählkathode aus. Dann wird ein Zündimpuls an die nächste Transferelektrode gelegt, wodurch am Ende des Impulses die nächste Zählkathode zündet. Die Zählrichtung wird so durch die Impulsfolge zwischen G1 und G2 bestimmt. Wenn G1 zuerst gepulst wird, erhöht sich der Zählerstand, wenn G2 zuerst gepulst wird, verringert sich der Zählerstand (**Bild 3**).

Es gab verschiedene Arten von Dekatrons. In der Zählervariante ist das Carry/Borrow-Bit mit einem eigenen Pin verdrahtet und die übrigen Zählkathoden sind mit einem einzigen Pin verbunden. Bei der Selektorkonfiguration in Bild 1 sind alle Zählkathoden auf separate Pins herausgeführt. Die meisten Dekatrons sind mit Wasserstoff gefüllt, der ihre Zählgeschwindigkeit auf etwa 4 kHz begrenzt, wobei in einigen Fällen auch eine maximale Geschwindigkeit von 10 kHz möglich ist.

Bild 1. Foto einer Dekatron-Röhre mit sichtbaren Einbauten [4] (Quelle: Wikicommons).

Ein Dekatron war nicht wirklich als Display gedacht, aber im Betrieb erzeugt es während der Zählung kreisförmig glühende Muster, die den Betrachter in ihren Bann ziehen. Leider ist es viel wahrscheinlicher, dass man sie in Schaltkreisen wie den frühen Taschenrechnern wie der Anita Mk8 [2], wo sie in der Tastatur-Abtastlogik verwendet werden, versteckt hatte und man diese deshalb nicht findet. Es kann also etwas schwierig sein, ihnen auf die Spur zu kommen, aber wenn Sie sie finden, werden Sie wissen, wozu diese kleinen sich drehenden Lichtkreise da sind. Und sich daran erfreuen…

200561-03

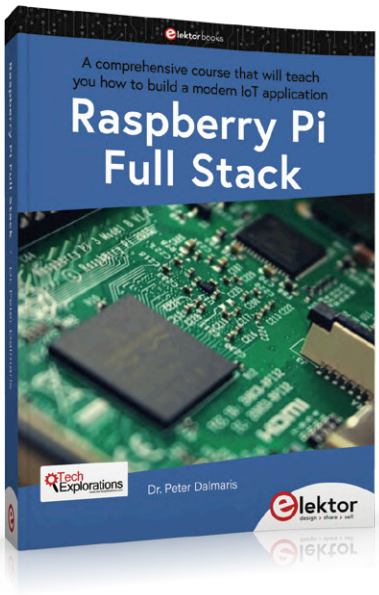
Ein Beitrag von
Idee, Text und Bilder: **Neil Gruending**
Redaktion: **Stuart Cording, C. J. Abate**
Layout: **Giel Dols**

WEBLINKS

[1] **J. B. Dance, „Electronic Counting Circuits,” London Iliffe Books Ltd., 1967.**: https://amzn.to/329Eikd
[2] **N. Tout, „Calculator Electronics,” Vintage Calculators Web Museum, 2002.**: https://bit.ly/38f4z46
[3] **Foto von Hellbus, CC BY-SA 3.0 license.**: https://bit.ly/2Go4ShC
[4] **Foto von Hellbus, CC BY-SA 3.0 license.**: https://bit.ly/2HZbBzi

Raspberry Pi Full Stack

RPi und RF24 als Herzstück eines Sensornetzwerks



Von **Dr. Peter Dalmaris** (Australien)

Das kürzlich erschienene Buch Raspberry Pi Full Stack von Peter Dalmaris richtet sich an wissbegierige RPi-Anwender, die auf ihrer Learning-by-Doing-Reise Hard- und Software miteinander verbinden möchten. Wir präsentieren daraus zwei Kapitel, in denen sehr detailliert die Kombination eines RPi mit einem RF24-Modul zum Auslesen entfernter Sensoren, die in ein kleines Netzwerk eingebunden sind, beschrieben wird.

Die Raspberry Pi fungiert als Basisknoten des RF24-Netzwerks und ist für die Verarbeitung der Sensorwerte verantwortlich, die er von dem oder den Arduino-Knoten erhält [der Arduino-Knoten wird in den vorhergehenden Kapiteln des Buches beschrieben]. In diesem Kapitel zeige ich Ihnen, wie Sie das RF24-Modul mit dem Raspberry Pi verbinden. Sie haben die Wahl, die Verdrahtung auf einer Lochrasterplatine vorzunehmen oder einen speziellen HAT zu verwenden, den ich zu diesem Zweck entworfen habe.

Ich habe mein Raspberry-Pi-RF24-Modul mit dem HAT-Breakout ausgestattet. In **Bild 1** können Sie es in Aktion sehen. Die HAT beherbergt das nRF24-Transceivermodul, den DHT22, einen Taster und zwei LEDs (zur Betriebs- und Aktivitätsanzeige). Sie können die Gerber-Dateien für die HAT-Platine bei TechExplorations herunterladen [1] und selber anfertigen oder (fertig, aber unbestückt) bei PCBWay [2] bestellen.

Wenn Sie es vorziehen, diese Schaltung als Prototyp auf einem Breadboard aufzubauen, nehmen Sie bitte die Verbindungstabelle in **Bild 2** und den Schaltplan der HAT-Platine in **Bild 3** zur Hilfe. Eine hochauflösende Version des HAT-Schaltplans erhalten Sie ebenfalls bei TechExplorations [3].

In der folgenden Liste habe ich alle Teile aufgeführt, die Sie für die Schaltung in Bild 2 benötigen. Einige dieser Teile, wie den Taster und den DHT22-Sensor, haben Sie bereits in früheren Kapiteln dieses Projekts kennengelernt. Dann sind die einzigen Neuhei-

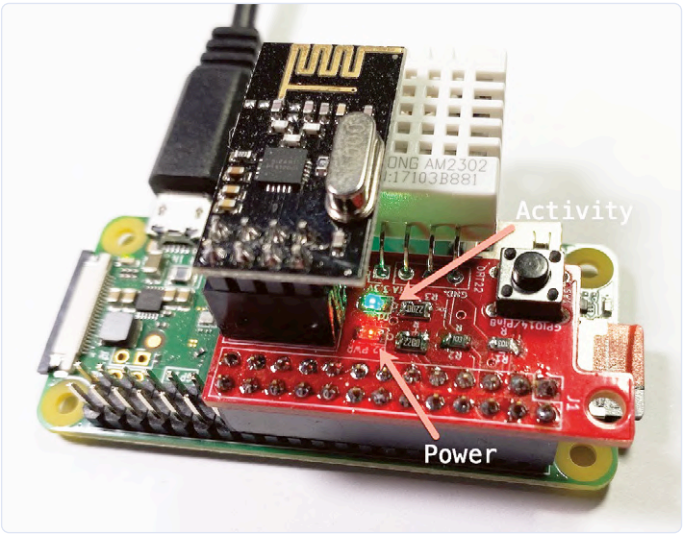


Bild 1. Die HAT-Platine auf einem Raspberry Pi Zero W.

Raspberry Pi - RF24 - DHT22

Raspberry Pi	nRF24	DHT22	Comments
3.3V	Vcc	Pin 1 - Power	220uF elec. cap pin "+".
GND	GND	Pin 4- GND	220uF elec. cap pin "-".
GPIO 11	SCK		
GPIO 9	MISO		
GPIO 10	MOSI		
GPIO 8	CSN		
GPIO 7	CE		
GPIO4 (Pin 7)		Pin 2 - Data	Add a 10KOhm pull up resistor between Data and GND.

Raspberry Pi Full Stack TechExplorations

Bild 2. Anschlussdetails für die Module nRF24 und DHT22.

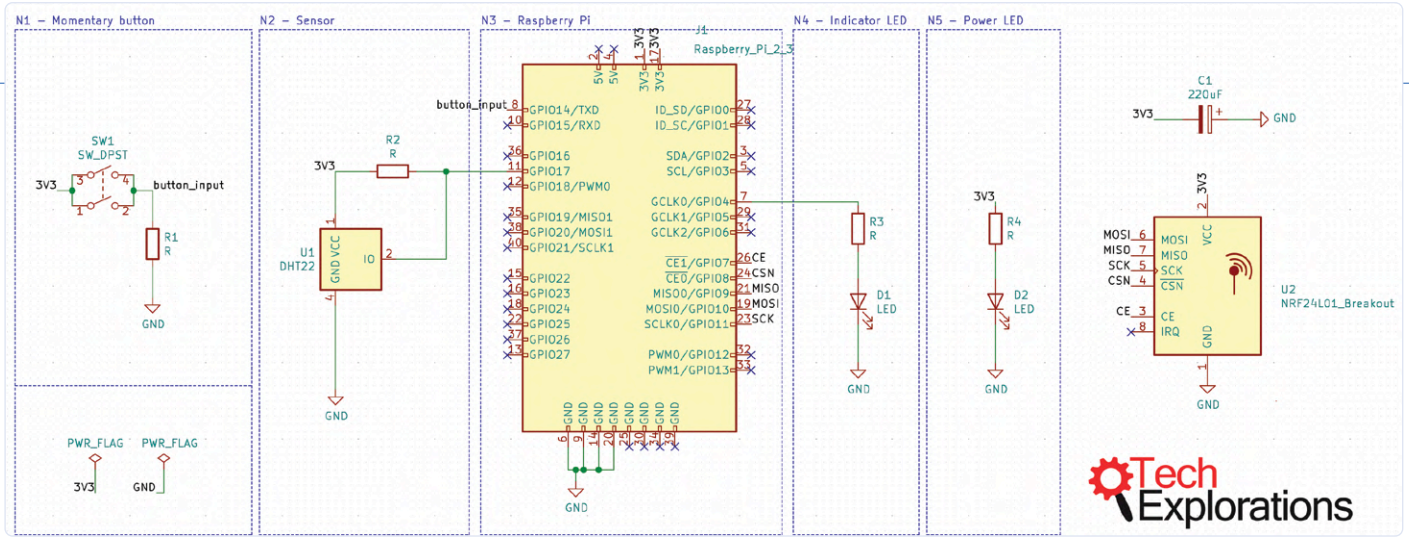


Bild 3. Das Schaltbild des Raspberry-Pi-HATs für die nRF24- und das DHT22-Breakout-Boards.

ten für Sie der nRF24-Transceiver und sein Bypass-Kondensator. Benötigte Bauteile:

- ein Sensor DHT22
- ein nRF24-Breakout-Board
- zwei Widerstände á 10 kΩ;
- zwei Widerstände á 330 Ω
- eine LED, rot (Betriebsanzeige)
- eine LED, blau (Aktivitätsanzeige)
- ein Elektrolytkondensator, etwa 220 µF.

Das Receiver-Script des nRF24

In diesem Abschnitt werde ich die Funktionalität des Python-Receiver-Scripts erklären, das auf dem Raspberry Pi läuft und die nRF24-Kommunikation übernimmt. Beachten Sie, dass dieses Script nicht „out of the box" funktioniert. Es hängt von den in C geschriebenen Treibern für RF24 und RF24Network und den Python-Wrappern ab, deren Einrichtung ich Ihnen im nächsten Kapitel zeigen werde. Für den Moment konzentrieren wir uns auf das Empfänger-Python-Script. Sie können den vollständigen Quellcode dieses Scripts herunterladen und dann im Projekt-Repository [4] einsehen. Ich habe dieses Script so geschrieben, dass es als ein von `systemd` gesteuerter Hintergrundprozess laufen kann. Ich werde Ihnen später zeigen, wie man das macht, denn zuerst müssen wir sicher sein, dass das Script in der Befehlszeile richtig läuft.
Der Inhalt der Funktion `log_values()` ist fast eine Kopie der gleichen Funktion im Script *env_log.py*, das bereits auf eine Ausführung nach einem Cron-Schedule eingestellt ist. Diese Funktion empfängt einfach Sensorwerte als Parameter und speichert sie in der lokalen Datenbank und in Google Sheet in der Cloud.
Im Folgenden liste ich ausgewählte Elemente des Scripts auf und erläutere vor allem die, die sich auf die RPi-nRF24-Kommunikation beziehen.

- Unmittelbar nach der Definition der Funktion `log_values()` richtet das Script das Modul nRF24 ein. Es initialisiert die Variable `radio` mit dem RF24-Konstruktor. Dieser Konstruktor ist Teil der RF24-Python-Wrapper-Bibliothek, die es uns ermöglicht, den RF24-C-Treiber aus unserem Python-Script heraus zu verwenden. Sie können sich über diese Funktion im Quellcode des Treibers informieren [5].
- Sobald das `radio`-Objekt erstellt und initialisiert ist, erzeugt das Script das `network`-Objekt mit Hilfe des RF24Network-Konstruktors. Dieses Objekt ermöglicht es dem Raspberry Pi, eine Sendung vom Arduino-Knoten zu empfangen. Der einzige Parameter, der zum Erstellen des `network`-Objekts benötigt wird, ist das `radio`-Objekt, das wir in der vorherigen Zeile erstellt haben. Weitere Informationen über den RF24Network-Konstruktor finden Sie im Quellcode von RF24Network.h, Zeile 373 [6].
- In der nächsten Zeile erzeugen wir mit `octlit = lambda n: int(n, 8)` eine Lambda-Funktion, die wir später zur Umwandlung einer Dezimal- in eine Oktalzahl verwenden. Dies ist notwendig, weil der RF24-Netzwerktreiber für die Knotenadressen das Oktalsystem verwendet. In Python ähnelt eine `lambda`-Funktion [7] einer regulären Funktion (bei der das Schlüsselwort `def` verwendet wird), hat aber keinen Namen. Sie sind bequem zu verwenden, wenn Sie Dinge tun möchten wie die Auswertung eines einzelnen Ausdrucks, genau wie es hier der Fall ist: Wir übergeben eine Dezimalzahl an `octlit` und die Lambda-Funktion gibt ihr Oktal-Äquivalent mit der Python-Funktion `int()` [8] zurück.
- In `this_node = octlit("00")` verwenden wir das `octlit`-Lambda, um das oktal-basierte Äquivalent von `00` zu erhalten, und speichern den Wert in der Variablen `this_node`. Dies ist die RF24-Netzwerkadresse des Raspberry Pi.
- In den nächsten sechs Zeilen bis zum `while`-Block macht das Script folgendes:
  - das RF24-Modul starten
  - 0,1 Sekunden warten, bis es bereit ist
  - das Netzwerk auf Kanal 90 starten
  - die HF- und Netzwerkkonfiguration auf die Konsole drucken
  - den Zähler `packets_sent` auf null zurücksetzen
  - und den Zähler `last_sent` auf null zurücksetzen
- Jetzt, wo das HF-Modul und das Netzwerk bereit sind, tritt das Script in eine Endlosschleife ein, in der es auf eine Sendung von einem Arduino-Knoten wartet.
- Zu Beginn jeder Schleife ruft es die `update()`-Funktion [9] des Netzwerkobjekts auf. Dieses prüft auf Ihre Nachrichten.
- Wenn es keine neue Nachricht gibt, schläft das Script für 1 s ein, dann startet die Schleife erneut.
- Wenn es eine neue Nachricht gibt, verwendet das Script die `read()`-Funktion [10], um deren 12 Byte breite Nutzlast zu lesen und sie in der Variablen `payload` zu speichern. Die

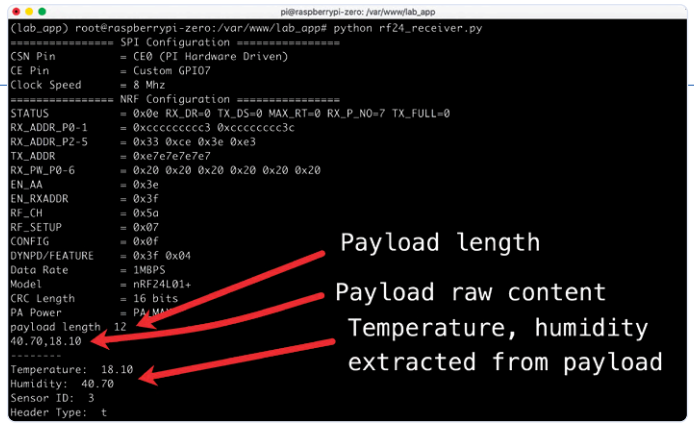


Bild 4. Beispiel für die Ausgabe des RF24-Empfängers.

Read-Funktion holt auch den Header der Nachricht und übergibt ihn an die `header`-Variable.

- In den nächsten beiden Zeilen benutze ich `print`, um die Nutzlast auf der Konsole zu drucken. Die Payload-Daten werden im folgenden `try`-Block in Zahlen dekodiert.
- Die `payload`-Variable enthält die Zeichenfolge `51.23,23.09`. Das Script verwendet `decode()` [11], um sie in eine UTF-8-Kodierung umzuwandeln, und die `split()`-Funktion [12], um die Temperatur- und Feuchtigkeitswerte in ein Array aus zwei Substrings zu verschieben, wobei das Komma als Trennzeichen verwendet wird. Die beiden Werte werden in der Variablen `values` (einem String-Array) gespeichert.
- Im `try`-Block verwendet das Script die Funktion `float()` [13], um die im Payload-Zeichenarray gespeicherten Zahlen zu extrahieren und in eine Gleitkommazahl umzuwandeln. Die Funktion `float()` empfängt die Zeichenfolge, und wenn sie in eine Zahl umgewandelt werden kann, gibt sie diese zurück.
- Mit `float(values[1][0:5])` nimmt das Script die ersten fünf Zeichen der Zeichenfolge, die im Index 1 des `value`-Arrays gespeichert ist, und wandelt sie mit der Funktion `float()` in eine Zahl um. Dies ist der in der Variablen `temperature` gespeicherte numerische Wert.
- Dasselbe geschieht beim Feuchtigkeitswert mit `float(values[0][0:5])`.
- Wenn eine dieser beiden Umwandlungen fehlschlägt, wird der `try`-Block beendet und eine Fehlermeldung auf der Konsole ausgegeben. Eine Konvertierung kann fehlschlagen, wenn die Nutzlastzeichenfolge vom Arduino nicht richtig formatiert oder die Nutzlast während der Übertragung und des Empfangs zum Beispiel durch Störungen beschädigt wird.

Nehmen Sie sich so viel Zeit wie nötig, um diesen Code zu studieren, damit Sie damit vertraut sind. Wenn Sie dann fertig sind, kopieren Sie ihn in Ihr Anwendungsverzeichnis. Kopieren Sie den Code aus dem Projekt-Repository [14] in den Zwischenspeicher und erzeugen Sie eine neue Datei mit dem Namen *rf24_receiver.py* in Vim. Bevor Sie die RF24-Kommunikation testen können, müssen Sie die C-Treiber von RF24 und RF24Network und die Python-Wrapper kompilieren. Sie werden dies im nächsten Kapitel tun. Aber zuvor möchte ich Ihnen zeigen, wie das Script, das Sie gerade kennengelernt haben, bei der Ausführung aussieht (**Bild 4**). Ich habe diesen Screenshot mit Anmerkungen versehen, damit Sie die in das Script eingebetteten Ausdrucke sehen können. Bauen Sie die Schaltung nun auf und fahren im nächsten Kapitel mit der Einrichtung der RF24- und RF24Network-Treiber und der Implementierung des Python-Scripts fort, das die nRF24-Kommunikation regelt.

So installieren Sie die nRF24-Module auf dem Raspberry Pi

In diesem Kapitel zeige ich Ihnen, wie Sie die C-Treiber und die relevanten Python-Wrapper für das nRF24-Modul kompilieren und installieren. Es sind zwei Module beteiligt, das Hauptmodul RF24 und das RF24Network-Submodul. Es gibt mehrere Forks des RF24-Projekts, aber die für den RPi optimierte „Zinke“, die Sie verwenden werden, wird von TMRh20 [15] verwaltet.
Den Quellcode für die beiden Module können Sie aus den jeweiligen Github-Repositories beziehen:

- RF24-Repository: *https://github.com/nRF24/RF24*;
- RF24Network-Repository: *https://github.com/nRF24/RF24Network*;

Für jedes Modul umfasst der Installationsprozess folgende Schritte:

- Herunterladen des Quellcodes des Moduls von Github
- Kompilieren und Installieren des C-Treibers
- Kompilieren und Installieren des Python-Wrappers

Bevor Sie beginnen, stellen Sie sicher, dass die SPI-Schnittstelle des Raspberry Pi aktiviert ist. Das Modul nRF24 verwendet SPI zur Kommunikation mit dem Raspberry Pi. Um dies zu überprüfen (und um SPI gegebenenfalls zu aktivieren), melden Sie sich bei Ihrem Raspberry Pi an und geben Sie in die Befehlszeile ein:

```
sudo raspi-config
```

Verwenden Sie die Eingabe- und die Pfeiltasten, um zu den Schnittstellenoptionen und dann zu SPI zu navigieren. Aktivieren Sie SPI, falls erforderlich, und beenden Sie raspi-config. Fahren Sie mit der Aktualisierung von Raspbian fort:

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

Sie werden nun den Quellcode der Module in Ihrem Anwendungsverzeichnis herunterladen. Sie werden auch die Python-Wrapper mit aktivierter virtueller Python-Umgebung Ihrer Anwendung kompilieren, damit die Python-Scripte Ihrer Anwendung zur Verfügung stehen. Bereiten Sie Ihre Arbeit mit diesen Befehlen vor:

```
$ sudo su
# cd /var/www/lab_app/
# . bin/activate
# mkdir rf24libs
# cd rf24libs
```

Sie sind nun bereit, mit der Kompilierung und der Installation der beiden Module zu beginnen.

Kompilierung und Installation der RF24-Treiber

Erstellen Sie einen Klon des RF24-Quellcodes aus dem Git unter *https://github com/tmrh20/RF24*.

```
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
```

```
   rf24libs# git clone
https://github.com/tmrh20/RF24.git RF24
Cloning into 'RF24'...
remote: Enumerating objects: 46, done.
remote: Counting objects: 100% (46/46), done.
remote: Compressing objects: 100% (41/41), done.
remote: Total 3545 (delta 16), reused 15 (delta 4),
   pack-reused 3499
Receiving objects: 100% (3545/3545), 1.54 MiB |
   679.00 KiB/s, done.
Resolving deltas: 100% (2119/2119), done.
```

In Ihrem Quellcode-Verzeichnis haben Sie jetzt ein neues Verzeichnis namens RF24.

```
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
   rf24libs# ls -al
total 12
drwxr-xr-x 3 root root 4096 May 13 01:38 .
drwxr-xr-x 14 root root 4096 May 13 01:39 ..
drwxr-xr-x 9 root root 4096 May 13 01:38 RF24
```

Gehen Sie in das Verzeichnis RF24 und installieren Sie dort den Treiber:

```
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
   rf24libs# cd RF24/
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
   rf24libs/RF24# make install
[Running configure]
[SECTION] Detecting arm compilation environment.
[OK] arm-linux-gnueabihf-gcc detected.
[OK] arm-linux-gnueabihf-g++ detected.
… (omitted output)…
[Installing Libs to /usr/local/lib]
[Installing Headers to /usr/local/include/RF24]
```

Der C-Treiber ist jetzt installiert, nun ist der Python-Wrapper an der Reihe. Dieser Vorgang kann auf dem langsameren Raspberry Pi Zero W mehrere Minuten dauern.

```
(lab_app) root@rpi4:/var/www/lab_app/rf24libs/RF24/
   pyRF24# cd pyRF24/
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
   rf24libs/RF24/pyRF24#
python setup.py install
running install
running bdist_egg
running egg_info
creating RF24.egg-info
… (omitted output)…
Installed /var/www/lab_app/lib/python3.8/
   site-packages/RF24-1.3.4-py3.8-
linux-armv6l.egg
Processing dependencies for RF24==1.3.4
Finished processing dependencies for RF24==1.3.4
```

Das Modul RF24 ist jetzt installiert und der Python-Wrapper einsatzbereit. Fahren Sie mit RF24Network fort..

```
RF24Network
```

Gehen Sie zurück zum Root des *rf23libs*-Verzeichnisses und erstellen Sie einen Klon des RF24Network-Quellcodes von *https://github.com/nRF24/RF24Network.git*.

```
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
   rf24libs/RF24/pyRF24# cd
../..
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
   rf24libs# git clone
https://github.com/nRF24/RF24Network.git
Cloning into 'RF24Network'...
remote: Enumerating objects: 2249, done.
remote: Total 2249 (delta 0), reused 0 (delta 0),
   pack-reused 2249
Receiving objects: 100% (2249/2249), 1.60 MiB |
   733.00 KiB/s, done.
Resolving deltas: 100% (1342/1342), done.
```

Der Klonvorgang hat ein neues Verzeichnis „*RF24Network*“ erstellt. Wechseln Sie in dieses neue Verzeichnis und kompilieren Sie den Quellcode:

```
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
   rf24libs# cd RF24Network/
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
   rf24libs/RF24Network# make
install
g++ -Wall -fPIC -c RF24Network.cpp
g++ -shared -Wl,-soname,librf24network.so.1 -o
   librf24network.so.1.0
RF24Network.o -lrf24-bcm
[Install]
[Installing Headers]
```

Der Treiber ist ebenfalls installiert. Es fehlt nur noch der Python-Wrapper für RF24Network.

```
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
   rf24libs/RF24Network# cd
../RF24/pyRF24/pyRF24Network/
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
   rf24libs/RF24/pyRF24/
pyRF24Network# python setup.py install
running install
running build
running build_ext
building 'RF24Network' extension
… (omitted output)…
running install_egg_info
Removing /var/www/lab_app/lib/python3.8/
   site-packages/
RF24Network-1.0-py3.8.egg-info
Writing /var/www/lab_app/lib/python3.8/site-packages/
RF24Network-1.0-py3.8.egg-info
```

Der Python-Wrapper für RF24Network ist installiert. Lassen Sie uns die Angelegenheit testen.

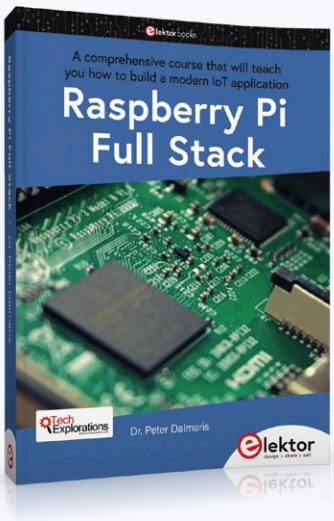
Funktionstest

Der einfachste Weg, um zu testen, ob die Module RF24 und RF24Network ordnungsgemäß funktionieren, ist die Ausführung der

RASPBERRY PI FULL STACK

Das Buch *Raspberry Pi Full Stack* nimmt Sie mit auf eine Wirbelwind-Tour durch die Entwicklung von Full-Stack-Webanwendungen mit dem Raspberry Pi. Sie lernen nicht nur, wie man eine Anwendung von Grund auf entwickelt, sondern sammeln auch Erfahrungen mit folgenden Technologien:

- Das Linux-Betriebssystem und die Befehlszeile
- Die Programmiersprache Python
- Der General Purpose Input Output-Anschluss (GPIO) des Raspberry Pi
- Der Nginx-Webserver
- Das Flask-Python-Microframework für Webanwendungen
- JQuery und CSS zur Erstellung von Benutzeroberflächen
- Umgang mit Zeitzonen
- Erstellen von Diagrammen mit Plotly und Google Charts
- Datenprotokollierung mit Google Sheet
- Entwickeln von Applets mit IFTTT
- Sichern Sie Ihre Anwendung mit SSL
- Empfang von SMS-Benachrichtigungen auf Ihrem Telefon mit Twilio



In diesem Buch erfahren Sie auch, wie Sie einen entfernten drahtlosen Arduino-Sensorknoten einrichten und Daten von diesem erfassen. Ihre Webanwendung auf dem Raspberry Pi wird in der Lage sein, die Daten des Arduino-Knotens auf die gleiche Weise zu verarbeiten wie die Daten eines Onboard-Sensors.

Raspberry Pi Full Stack wird Ihnen viele Fähigkeiten vermitteln, die für die Erstellung von Web- und Internet-of-Things-Anwendungen unerlässlich sind. Die Anwendung, die Sie in diesem Projekt erstellen werden, ist eine erweiterbare Plattform und nur der Anfang dessen, was Sie mit einem Raspberry Pi und den Software- und Hardwarekomponenten, die Sie kennenlernen, tun können. Dieses Buch wird vom Autor mit einem speziellen Diskussionsforum unterstützt.

- **Druckversion:** *www.elektor.de/raspberry-pi-full-stack*
- **E-Book:** *www.elektor.de/raspberry-pi-full-stack-e-book*

Python-Beispielprogramme, die mit dem Quellcode verbunden sind. Sie finden diese Beispiele im examples-Verzeichnis von pyRF24Network. Führen Sie das „rx"-Beispiel wie folgt aus:

```
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
   rf24libs/RF24/pyRF24/
pyRF24Network# cd examples/
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
   rf24libs/RF24/pyRF24/
pyRF24Network/examples# ls
helloworld_rx.py helloworld_tx.py
(lab_app) root@raspberrypi-zero:/var/www/lab_app/
   rf24libs/RF24/pyRF24/
pyRF24Network/examples# python helloworld_rx.py
================ SPI Configuration ================
CSN Pin = CE0 (PI Hardware Driven)
CE Pin = Custom GPIO22
Clock Speed = 8 Mhz
================ NRF Configuration ================
```

WEBLINKS

[1] **Gerber-Dateien für das RF24-HAT-Board**: https://techexplorations.com/parts/rpifs-parts/

[2] **HAT-Leerplatine von PCBWay**: www.pcbway.com/project/shareproject/Raspberry_Pi_Full_Stack_RF24_and_DHT22_HAT.html

[3] **High-Resolution-Version des Schaltplans**: https://techexplorations.com/parts/rpifs-parts/

[4] **Quellcode des Python-Scripts rf24_receiver.py**: https://github.com/futureshocked/RaspberryPiFullStack_Raspbian/blob/master/rf24_receiver.py

[5] **Siehe Zeile 137 von RF24.h**: https://github.com/nRF24/RF24/blob/master/RF24.h#L137

[6] **Siehe Zeile 373 von RF24.h**: https://github.com/nRF24/RF24Network/blob/master/RF24Network.h#L373

[7] **Lernen Sie mehr über „Lambda Expressions"**: https://docs.python.org/3/tutorial/controlflow.html#lambda-expressions

[8] **Erfahren Sie mehr über „int()"**: https://docs.python.org/3/library/functions.html#int

[9] **Weitere Informationen unter**: https://github.com/nRF24/RF24Network/blob/master/RF24Network.h#L413

[10] **Weitere Informationen unter**: https://github.com/nRF24/RF24Network/blob/master/RF24Network.h#L467

[11] **Erfahren Sie mehr über „decode()"**: https://docs.python.org/3/library/stdtypes.html#bytes.decode

[12] **Erfahren Sie mehr über „split()"**: https://docs.python.org/3/library/stdtypes.html#str.split

[13] **Erfahren Sie mehr über „float()"**: https://docs.python.org/3/library/functions.html#float

[14] **Neueste Version von rf24_receiver.py**: https://github.com/futureshocked/RaspberryPiFullStack_Raspbian/blob/master/rf24_receiver.py

[15] **Die „Heimat" des Projekts**: https://tmrh20.github.io/RF24/

```
STATUS = 0x0e RX_DR=0 TX_DS=0 MAX_RT=0 RX_P_NO=7
   TX_FULL=0
RX_ADDR_P0-1 = 0xcccccccccc3 0xcccccccc3c
RX_ADDR_P2-5 = 0x33 0xce 0x3e 0xe3
TX_ADDR = 0xe7e7e7e7e7
RX_PW_P0-6 = 0x20 0x20 0x20 0x20 0x20 0x20
EN_AA = 0x3e
EN_RXADDR = 0x3f
RF_CH = 0x5a
RF_SETUP = 0x07
CONFIG = 0x0f
DYNPD/FEATURE = 0x3f 0x04
Data Rate = 1MBPS
Model = nRF24L01+
CRC Length = 16 bits
PA Power = PA_MAX
```

Beachten Sie, dass die Ausgabe Statistiken des RF24-Moduls enthält, die während des Betriebs angefallen sind. Sie enthalten gültige Werte für den RX, eine Adresse (Werte ungleich Null) sowie den Rest der NRF-Konfiguration. Wenn die Module RF24 und RF24Network nicht richtig installiert wurden oder wenn der Raspberry Pi nicht mit dem nRF24-Modul über SPI kommunizieren konnte, erhalten Sie entweder eine Fehlermeldung vom Beispielprogramm: Es startete überhaupt nicht oder die Konfigurationswerte sind leer. Wenn die RF24-Module ordnungsgemäß installiert sind, sollte das installierte Python-Script funktionieren.

200517-02

Über den Autor

Dr. Peter Dalmaris ist Pädagoge, Elektroingenieur und Konstrukteur sowie der Urheber von Online-Videokursen über DIY-Elektronik und Autor mehrerer Fachbücher. Als Chief Tech Explorer bei Tech Explorations seit 2013, dem von ihm in Sydney (Australien) gegründeten Unternehmen, hat Peter Dalmaris sich zur Aufgabe gemacht, die Technik zu erforschen und zur Fortbildung von Interessierten beizutragen.

Ein Beitrag von

Autor: **Dr. Peter Dalmaris**
Illustrationen: **Dr. Peter Dalmaris**
Redaktion: **Jan Buiting**
Übersetzung: **Rolf Gerstendorf**
Layout: **Giel Dols**

Sie haben Fragen oder Kommentare?

Haben Sie Fragen oder Anmerkungen zum Buch? Schicken Sie eine E-Mail an den Autor unter peter@txplore.com.

Praktisches
ESP32-Multitasking (6)
Event Groups

Von Warren Gay (Kanada)

In Anwendungen mit FreeRTOS ist oft eine Synchronisierung zwischen mehreren Tasks erforderlich. Bisher haben wir Semaphore und Task-Benachrichtigungen kennengelernt, die aber nur in der Lage sind, ein Event eines Tasks oder einer Interrupt Service Routine (ISR) an einen einzelnen empfangenden Task zu senden. Wenn Sie ein Ereignis aber an mehrere Tasks senden müssen, sind Event Groups die gesuchte Lösung. In diesem letzten Artikel der Reihe untersuchen wir die Fähigkeiten, die diese Funktion Embedded-Entwicklern bietet.

Schauen wir uns einen typischen Anwendungsfall an. Ein MIDI-Drumcomputer soll auf Grundlage eines über seinen seriellen Kanal empfangenen Eingabebefehls vier Sounds gleichzeitig erzeugen. Um eine Kakophonie zu vermeiden, sollten alle klangerzeugenden Tasks natürlich genau zur gleichen Zeit beginnen. Dies kann zwar auch durch vier separate Semaphore erreicht werden, wenn man sich auf die Geschwindigkeit der CPU verlässt, so dass alles recht zeitgleich erledigt wird. Doch dieser Ansatz wäre ungeschickt und skaliert nicht gut, wenn die Zahl der empfangenden Tasks zunimmt. Die bevorzugte Methode in FreeRTOS, um mehrere Tasks von einer einzigen Eventquelle aus zu koordinieren, sind Event Groups.

Event Flags

FreeRTOS definiert 24 Event-Flags für jedes erzeugte Event-Group-Objekt (**Bild 1**). Diese Flags werden im C-Datentyp `EventBits_t` dargestellt, einem 32 Bit breiten Datentyp, von dem 24 Bit zur Verfügung des Programmierers stehen. Bit 0 ist das niederstwertige Bit (LSB) dieser verfügbaren Flags. Die höchstwertigen acht Bits sind für den internen Gebrauch von FreeRTOS reserviert.

Wie diese 24 Bits von der Anwendung zugewiesen und verwendet werden, bleibt dem Programmierer überlassen. In der folgenden Demo erzeugt der `loop()`-Task jede Sekunde ein Ereignis, das dazu führt, dass zwei Tasks synchron unterschiedliche Blinkmuster ihrer zugehörigen LEDs starten. Bit 0 der Event Group (Wert `0b0001`/Dezimalwert 1) benachrichtigt einen Task namens `blink2()`, der die LED1 zweimal blinken lässt. Bit 1 der Event Group (Wert `0b0010`/Dezimalwert 2) wird angewiesen, einen Task namens `blink3()` zu benachrichtigen, der die LED2 dreimal blinken lässt.

Demo-Programm

Bevor wir in den Code in **Listing 1** [1] eintauchen, lassen Sie uns noch einmal überprüfen, was das Demoprogramm zu erreichen hofft. Die Schaltung in **Bild 2** zeigt zwei aktiv high geschaltete LEDs an den GPIO-Pins 25 und 26 (Zeilen 5/6). Diese GPIOs werden in der `setup()`-Funktion als Ausgänge konfiguriert (Zeilen 62...65). Task `blink2()` steuert LED1, die zweimal blinken soll, bevor auf das nächste Ereignis gewartet wird (Zeilen 19...25). Task `blink3()` steuert LED2 an, die dreimal blinken und dann auf das nächste Ereignis warten soll (Zeilen 41...47). Beide Tasks sind identisch mit Ausnahme der Anzahl, wie oft die Blinkschleife (Zeilen 27/49) durchlaufen wird, und der vorgegebenen Verzögerungszeiten.

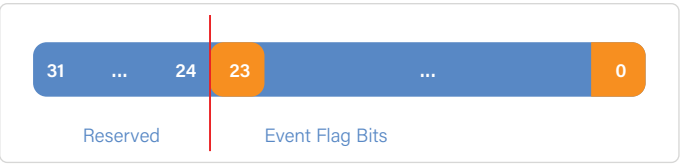


Bild 1. Event-Flags innerhalb des C-Datentyps EventBits_t.

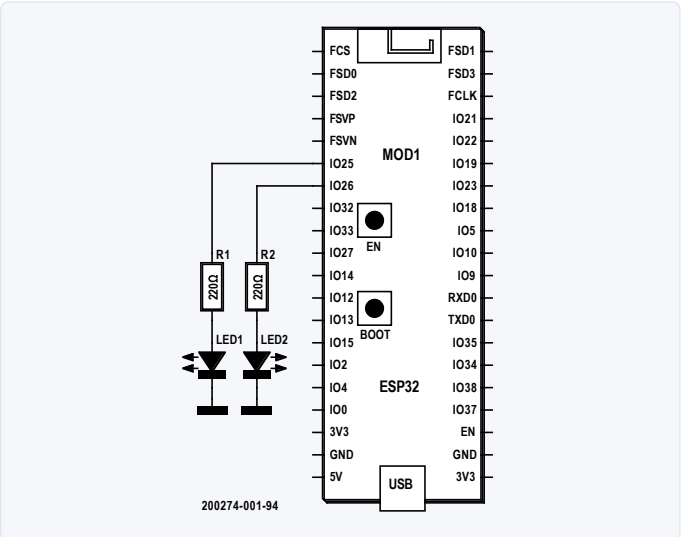


*Bild 2. Die Schaltung für das Demoprogramm **evtgrp.ino**.*

Listing 1. Das Arduino-Programm evtgrp.ino mit Event Groups [1].

```
0001: // evtgrp.ino
0002: // MIT License (see file LICENSE)
0003:
0004: // LED is active high
0005: #define GPIO_LED1     25
0006: #define GPIO_LED2     26
0007:
0008: #define EVBLK2        0b0001
0009: #define EVBLK3        0b0010
0010: #define EVALL         0b0011
0011:
0012: static EventGroupHandle_t hevt;
0013:
0014: void blink2(void *arg) {
0015:
0016:   for (;;) {
0017:     // Call blocks until EVBLK2 bit set,
0018:     // and same bit is cleared upon return
0019:     xEventGroupWaitBits(
0020:       hevt,
0021:       EVBLK2,
0022:       pdTRUE,
0023:       pdFALSE,
0024:       portMAX_DELAY
0025:     );
0026:     // Blink 2 times
0027:     for ( int x=0; x < 2; ++x ) {
0028:       digitalWrite(GPIO_LED1,HIGH);
0029:       delay(120);
0030:       digitalWrite(GPIO_LED1,LOW);
0031:       delay(120);
0032:     }
0033:   }
0034: }
0035:
0036: void blink3(void *arg) {
0037:
0038:   for (;;) {
0039:     // Call blocks until EVBLK3 bit set,
0040:     // and same bit is cleared upon return
0041:     xEventGroupWaitBits(
0042:       hevt,
0043:       EVBLK3,
0044:       pdTRUE,
0045:       pdFALSE,
0046:       portMAX_DELAY
0047:     );
0048:     // Blink 3 times
0049:     for ( int x=0; x < 3; ++x ) {
0050:       digitalWrite(GPIO_LED2,HIGH);
0051:       delay(75);
0052:       digitalWrite(GPIO_LED2,LOW);
0053:       delay(75);
0054:     }
0055:   }
0056: }
0057:
0058: void setup() {
0059:   int app_cpu = xPortGetCoreID();
0060:   BaseType_t rc;
0061:
0062:   pinMode(GPIO_LED1,OUTPUT);
0063:   pinMode(GPIO_LED2,OUTPUT);
0064:   digitalWrite(GPIO_LED1,LOW);
0065:   digitalWrite(GPIO_LED2,LOW);
0066:
0067:   delay(2000);
0068:
0069:   hevt = xEventGroupCreate();
0070:   assert(hevt);
0071:
0072:   rc = xTaskCreatePinnedToCore(
0073:     blink2,   // func
0074:     "blink2", // name
0075:     1024,     // stack bytes
0076:     nullptr,  // arg ptr
0077:     1,        // priority
0078:     nullptr,  // ptr to task handle
0079:     app_cpu   // cpu#
0080:   );
0081:   assert(rc == pdPASS);
0082:
0083:   rc = xTaskCreatePinnedToCore(
0084:     blink3,   // func
0085:     "blink3", // name
0086:     1024,     // stack bytes
0087:     nullptr,  // arg ptr
0088:     1,        // priority
0089:     nullptr,  // ptr to task handle
0090:     app_cpu   // cpu#
0091:   );
0092:   assert(rc == pdPASS);
0093: }
0094:
0095: void loop() {
0096:
0097:   delay(1000);
0098:   xEventGroupSetBits(
0099:     hevt,
0100:     EVBLK2|EVBLK3
0101:   );
0102: }
0103:
0104: // End evtgrp.ino
```

Synchronisierung

Die Synchronisation wird erreicht, indem die Task-Funktionen in einem Aufruf von `xEventGroupWaitBits()` blockiert werden. Bis das Event ausgelöst wird, bleibt die Ausführung innerhalb dieses Funktionsaufrufs hängen.

Sobald die Event Group benachrichtigt wird, können die blockierten Tasks durch Rückkehr von der aufgerufenen Funktion wieder ausgeführt werden. Das Ereignis wird bei uns jede Sekunde durch einen entsprechenden Aufruf von `xEventGroupSetBits()` im `loop()`-Task ausgelöst (Zeilen 98..101).

Erstellen von Event Groups

Die Event Group wird durch den Aufruf von `xEventGroupCreate()` in Zeile 69 erstellt. Es müssen keine Argumente angegeben werden. Die Funktion gibt ein Handle auf das zugewiesene Event-Group-Objekt zurück. Der zurückgegebene Datentyp des Handle ist `EventGroupHandle_t`. Wenn der zurückgegebene Wert null ist, ist Ihr Heap-Speicher ausgeschöpft (daher die Assertion-Prüfung in Zeile 70).

```
0069:   hevt = xEventGroupCreate();
0070:   assert(hevt);
```

Benachrichtigung

Es gibt verschiedene Möglichkeiten, Event Groups zu manipulieren. In diesem Artikel verwenden wir eine der einfacheren Funktionen namens `xEventGroupSetBits()`.

```
EventBits_t xEventGroupSetBits(
  EventGroupHandle_t xEventGroup,    // Handle
  const EventBits_t uxBitsToSet      // Bits to set
);
```

Das Handle `xEventGroup` gibt die Event Group an, mit der gearbeitet werden soll, während das Argument `uxBitsToSet` die Event-Bits angibt, die Sie atomar setzen wollen. Der zurückgegebene Wert sind die Event-Bits, die *nach* der Rückkehr des Aufrufs von `xEventGroupSetBits()` gültig waren. Sie müssen sich jedoch darüber im Klaren sein, dass der zurückgegebene Wert nicht immer die Bits enthält, die Sie gerade gesetzt haben, da die empfangenden Tasks sie beim Empfang möglicherweise bereits gelöscht haben.
Der Task `loop()` ruft die Funktion `xEventGroupSetBits()` auf und setzt Bit 0 und Bit 1 mit dem Makroausdruck `EVBLK2|EVBLK3`.

```
0008: #define EVBLK2        0b0001
0009: #define EVBLK3        0b0010

0098:   xEventGroupSetBits(
0099:     hevt,
0100:     EVBLK2|EVBLK3
0101:   );
```

Es ist ersichtlich, dass durch diesen Aufruf Bit 0 und Bit 1 atomar gesetzt wurden.

Der loop()-Task

Der Task `loop()` verzögert um eine Sekunde (Zeile 97) und sendet dann eine Event-Benachrichtigung (Zeilen 98…101). Dann wird, der Arduino-Tradition folgend, die `loop()`-Funktion beendet und wiederholt. Dies führt dazu, dass im Intervall von etwa einer Sekunde Events ausgelöst werden.

Empfangende Tasks

Die Tasks `blink2()` und `blink3()` werden von der Event-Gruppe mit dem gespeicherten Handle (Zeilen 12/69) benachrichtigt. So wird die Ausführung von `blink2()` in den Zeilen 19…25 verhindert, bis ein Event gesendet wird, genau so wie die Ausführung von `blink3()` in den Zeilen 41…47. Da die Events atomar durch die Zeilen 98…101 ausgelöst werden, nehmen beide Tasks gleichzeitig wieder ihre Arbeit auf. Wenn die beiden Tasks derselben CPU zugeordnet sind, werden sie nacheinander zur Ausführung geplant, aber wenn sie verschiedenen CPUs zugeordnet sind, ist es tatsächlich möglich, dass sie gleichzeitig fortgesetzt und ausgeführt werden.
Sobald die Tasks ihre Tätigkeit aufnehmen, lassen sie ihre LEDs auf ihre eigene Weise blinken. Wenn das Blinken beendet ist, kehren die Tasks zum Anfang der Taskschleife zurück und warten auf die Ankunft des nächsten Events.

Event Clearing

Der Funktionsaufruf `xEventGroupWaitBits()` ist aufgrund seiner großen Flexibilität ein klein wenig tricky. Lassen Sie ihn uns also aufschlüsseln:

```
EventBits_t xEventGroupWaitBits(
  const EventGroupHandle_t xEventGroup, // Event
   group handle
  const EventBits_t uxBitsToWaitFor,
  const BaseType_t xClearOnExit,
  const BaseType_t xWaitForAllBits,
  TickType_t xTicksToWait
);
```

Das erste Argument ist das Handle auf die Event Group, auf die verwiesen wird. Dies ist recht einfach. Das zweite Argument gibt die Event-Bits an, auf die die Funktion warten will. Im Demoprogramm haben wir das Makro `EVBLK2` (Zeile 21) für den Task `blink2()` und das Makro `EVBLK3` für den Task `blink3()` angegeben. Der Grund für den Gebrauch separater Bits wird gleich deutlich werden.
Das dritte Argument `xClearOnExit` ist eine C-Sprachversion eines boolschen Wertes (`pdTRUE` wurde in den Zeilen 22/44 angegeben). Damit wird die Funktion informiert, die empfangenen Event-Bits vor der Rückkehr zu löschen. Der Task `blink2()` wartet also darauf, dass Bit 0 gesetzt wird (Makro `EVBLK2`), und der Wert des dritten Arguments `pdTRUE` bestimmt, dass dieses Bit bei Empfang und Return gelöscht wird. Diese Wait-and-Clear-Operation wird atomar ausgeführt.
In unserem speziellen Fall ist das vierte Argument nicht nötig, aber lassen Sie es uns dennoch untersuchen. Das Argument `xWaitForAllBits` ist ebenfalls boolsch und gibt an, wann die Funktion zurückkehren soll:

- wenn eines der Bits, auf die gewartet wird, gesetzt ist (`pdFALSE`), oder
- erst dann, wenn alle Bits, auf die gewartet wird, gesetzt sind (`pdTRUE`).

In unserem Beispiel wird nur ein einzelnes Bit erwartet (Bit 0 für `blink2()` und Bit 1 für `blink3()`), daher hat die Angabe eines oder aller Bits keine wesentliche Auswirkung. Wenn Sie jedoch auf eine Kombination von Bits warten möchten, kann diese Fähigkeit recht nützlich sein.

Event-Steuerung

Der Grund, warum separate Event-Bits benötigt wurden, ist jetzt vielleicht offensichtlich. Jeder empfangende Task wartet auf sein eigenes Event-Bit und löscht es dann mit einem Aufruf von

WEBLINKS

[1] **Code für evtgrp.ino: https**://bit.ly/35YrjCN
[2] **W. Gay, FreeRTOS for ESP32-Arduino, Elektor 2020**: www.elektor.de/freertos-for-esp32-arduino
[3] **FreeRTOS-Dokumentation**: https://bit.ly/386HZL6

`xEventGroupWaitBits()`. Daher muss es für jeden Task ein eigenes Bit geben. In der `loop()`-Funktion werden beide Tasks gleichzeitig benachrichtigt, indem ihre spezifischen Ereignisbits gleichzeitig gesetzt werden (Zeile 100).
Wie zu sehen, können diese Bits verwendet werden, um sehr komplexen und kreativen Applikationscode zu erstellen. Weitere Einzelheiten zu den zusätzlichen Funktionen finden Sie im Buch *FreeRTOS for ESP32-Arduino* [2] und in der FreeRTOS-Dokumentation [3].

Ausführen der Demo

Das Demoprogramm benutzt nicht den seriellen Monitor und kann auf fast jedem ESP32 betrieben werden, der GPIO 25 und GPIO 26 zur Verfügung stellt (oder Sie ändern für anderen Ausgänge die Zeilen 5/6 des Listings). Die beiden LEDs werden über 220-Ω-Strombegrenzungswiderstände angeschlossen.
Flashen Sie das Programm *evtgrp.ino* in den ESP32. Sobald die Ausführung des Programms beginnt, sollten die LEDs einmal pro Sekunde blinken – LED1 zweimal und LED2 dreimal und dies schnell. Danach warten beide Tasks, bis der Vorgang nach einer Sekunde wiederholt wird.

Synchronisation abgeschlossen

Diese Demo veranschaulicht, wie die beiden völlig unabhängigen Tasks `blink2()` und `blink3()` mit einem einzigen Event-Group-Objekt synchronisiert werden können. Der Task `loop()` überträgt sein Event in den Zeilen 98...101. Dieser Ansatz kann bei Bedarf auf bis zu 24 Tasks erweitert werden. Diese Grenze wird durch die Anzahl der in einer einzigen Event Group verfügbaren Ereignis-Flags gesetzt. Dieser Artikel hat nur eine Möglichkeit zur Verwendung von Event Groups gezeigt; es gibt mit anderen FreeRTOS-Funktionen eine ganze Reihe anderer, ausgefeilterer Möglichkeiten zur Koordinierung von Events.

200528-02

Sie haben Fragen oder Kommentare?
Haben Sie Fragen oder Anmerkungen zu diesem Artikel? Schicken Sie eine E-Mail an den Autor unter ve3wwg@gmail.com oder kontaktieren Sie Elektor unter editor@elektor.com.

Ein Beitrag von
Idee und Text: **Warren Gay**
Schaltbild: **Patrick Wielders**
Grafik: **Jack Jamar**
Redaktion: **Stuart Cording**
Übersetzung: **Rolf Gerstendorf**
Gestaltung: **Giel Dols**

PASSENDE PRODUKTE

› **W. Gay, *FreeRTOS for ESP32-Arduino*, Elektor, 2020.**
www.elektor.de/freertos-for-esp32-arduino

Mehrkanal-Power-Analyzer

Bis zu drei Kanäle, mit grafischer und alphanumerischer Anzeige

Von **Wil Dijkman** (Niederlande)

Bei der Durchführung von Spannungs-, Strom- und Leistungsmessungen an wechselstrombetriebenen Geräten werden die Dinge komplizierter als bei der Messung von Gleichstrom, da hier Wellenform und Phasenverschiebung zwischen Spannung und Strom eine wichtige Rolle spielen. Dieses Gerät misst und berechnet nicht nur Größen, sondern zeigt auch Wellenformen und Spektren von Wechselstromsignalen auf einem grafischen LCD an.

Dieses Projekt wurde durch den AC/DC-Leistungsmesser inspiriert, der im September 2015 in Elektor veröffentlicht wurde [1]. Zuerst wollte ich das Gerät bauen, fand aber dann im Entwurf einige Einschränkungen und Nachteile, so dass ich beschloss, das Design weiter zu entwickeln. Die Dinge, die ich verbessern wollte, waren:

- Die Eingangsschaltung: Strom und Spannung sollen unabhängig voneinander zu messen sein. Der Strom durch die Messleitungen soll die Spannungsanzeige nicht beeinflussen.
- Die Abtastrate: Durch die niedrige Abtastrate konnten nur die ersten acht Harmonischen eines 50-Hz-Signals gemessen werden (mit perfektem Filter). Dies soll auf die ersten 40 Harmonischen erhöht werden.
- Die Bereichsumschaltung und Offset-Korrektur soll automatisch erfolgen.

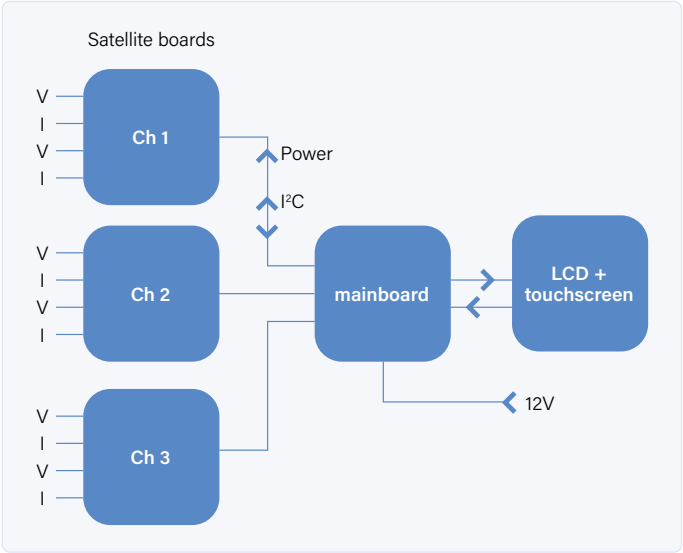


Bild 1. Blockschaltung des Power Analyzers.

Dies bedeutet allerdings, dass die Eingangs- und Verstärkerschaltung komplett neu gestaltet werden mussten. Außerdem wurde ein Mikrocontroller in der Messelektronik auf der Satellitenplatine benötigt, um die automatische Offset-Korrektur und das Auto-Ranging zu ermöglichen. Die Blockschaltung in **Bild 1** zeigt eine Hauptplatine und (bis zu) drei Satellitenplatinen.

Die Hauptplatine

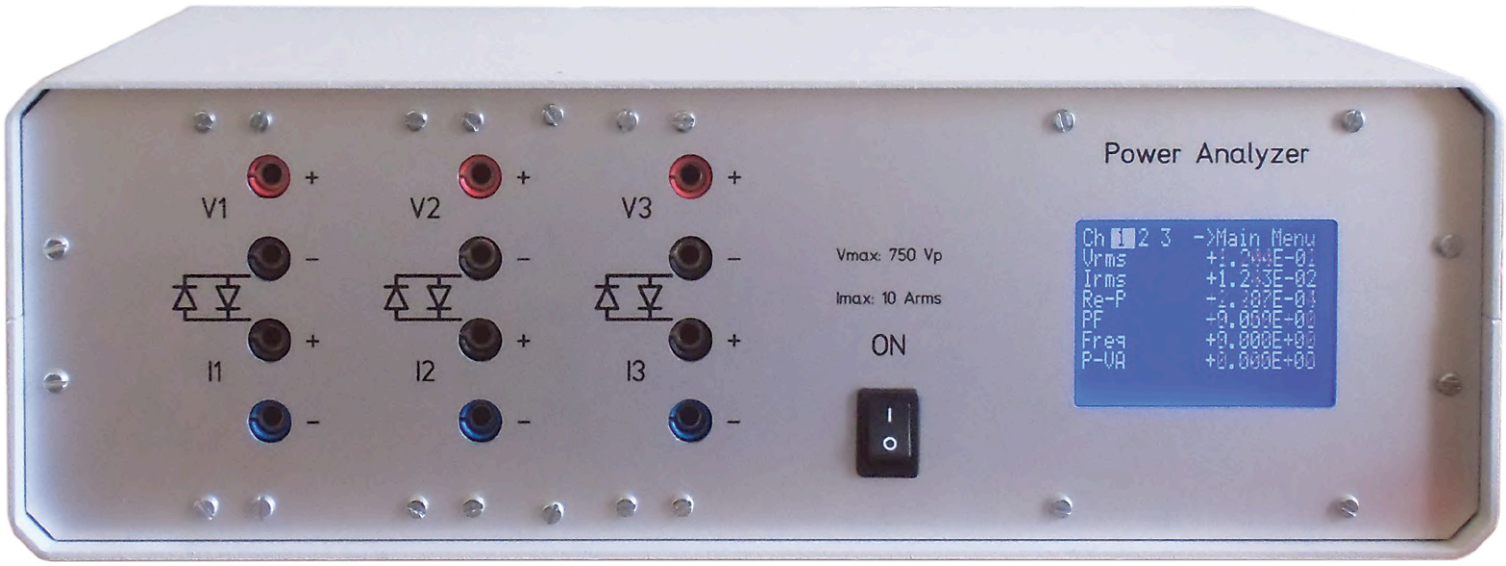
Der Schaltplan in **Bild 2** ist weitgehend eine Kopie des EasyPICV7-Boards mit einem grafischen LCD-Touchscreen als Benutzerschnittstelle. Ein Rechteck-Oszillator (Schaltung um IC1) erzeugt ein Rechtecksignal mit einer Amplitude von 12 V_{pp} und einer Frequenz von etwa 150 kHz zur Versorgung der Satellitenplatinen. Die Datenkommunikation mit den Satellitenplatinen erfolgt über I²C, wobei der Mikrocontroller auf der Hauptplatine der Master ist und die auf den Satellitenplatinen seine Slaves darstellen.

Der Stromversorgungsteil der Hauptplatine erlaubt es, zur Versorgung des Messgeräts entweder eine stabilisierte 12-V- oder eine unstabilisierte 15...18-V-Versorgung zu verwenden. Die Auswahl erfolgt durch den „Lötjumper" SJ1. Mit den beiden weiteren Lötbrücken/Jumpern MAX1 und MAX2 kann die Anzahl der Kanäle für den Master eingestellt werden: MAX1 zu schließen bedeutet nur einen, MAX2 zu schließen bedeutet zwei und beide zu schließen bedeutet drei Kanäle (sprich: Satellitenplatinen). Einige Drahtbrücken (in den Schaltplänen und auf dem Hauptplatinen-Layout mit Jx gekennzeichnet) helfen, die Notwendigkeit einer mehrlagigen Leiterplatte zu vermeiden.

Die Satellitenplatine

Auf den Satellitenplatinen (wer glaubt, hier die einfachere Schaltung zu finden, sieht sich durch **Bild 3** widerlegt) wollte ich Spannung und Strom unabhängig voneinander messen. Dies ist nicht vollständig gelungen, aber der Niederspannungseingang und der Stromeingang können aufgrund der Dioden D3, D4, D7, D8 immerhin nur um ±1 V gegeneinander floaten.

Ohne Schutzmaßnahmen könnte die Schaltung durch einen falschen Anschluss, ein hohes Potential am Niederspannungseingang und auch



am Stromeingang, beschädigt werden. Um dies zu verhindern, wurden T10...T14 und das Reed-Relais K1 hinzugefügt. Für eine erste Strombegrenzung sorgt schon R11: Bei 750 V Eingangsspannung beträgt der maximale Strom 0,5 A. Wenn der Strom durch R11 größer als einige 10 mA bis 20 mA ist, fließt ein kleiner, aber ausreichender Strom durch R34, um T11 und T13 umzuschalten. Dadurch wird der Thyristorkreis mit T10 und T12 getriggert. Nach dem Triggern verbleibt der Thyristorkreis in diesem Zustand und schaltet über T14 das Relais K1 ab. Das Signal OC (über R39) zeigt dem Mikrocontroller, dass der Schutz ausgelöst ist. Wenn keine Spannung vorhanden ist, wird K1 auch abgeschaltet (V-LOW wird getrennt), so dass der Schaltkreis sicher ist. Um den Betrieb wiederherzustellen, muss die Stromversorgung der Schaltung aus- und wieder eingeschaltet werden.

Der Spannungsteiler (R2/4/7/12 gegen R22) liefert 100 mV Ausgangsspannung bei der maximalen Eingangsspannung von 750 V (DC), die maximale Wechselspannung kann etwa 500 V betragen. Der Spannungsabfall am 6,5-mΩ-Strommesswiderstand R62 beträgt etwa 100 mV bei 15 A_{DC} (etwa 10 A_{AC}). Die Verstärker für Spannung und Strom sind identisch, daher beschreibe ich hier nur den Spannungsverstärker. T1, T2 und T4, T6 dienen als Schalter, um den Differenzverstärker entweder mit dem Signal aus dem Eingangskreis zu verbinden oder den Eingang kurzzuschließen, um eine Offset-Messung zu ermöglichen. Der Wert des Offsets kann gespeichert und später von der Messung subtrahiert/addiert werden, um eine „saubere Messung" zu ermöglichen. Der Verstärkungsfaktor des üblichen Instrumentenverstärkers IC2/IC6 wird von T3 und T5 entweder auf 1x oder 25x geschaltet. IC5 asymmetriert das Differenzsignal, anschließend erfolgt eine weitere Verstärkung durch IC4, dessen Verstärkungsfaktor je nach T8 und T9 1x oder 5x beträgt. Insgesamt sind damit vier Verstärkungen geben: 1x, 5x, 25x oder 125x. Die Einstellung des Verstärkungsfaktors erfolgt durch den Mikrocontroller IC13.

Der Verstärker erzeugt ein positives oder negatives Signal in Bezug auf COMV. Dieses Signal ist das 0,5-fache von VREF, das von der Schaltung um IC3 erzeugt wird. VREF beträgt 4,5 V, so dass COMV, COMI und CINV 2,25 V betragen. VREF und CINV sind mit dem A/D-Wandler des PIC-Controllers PIC18F26K80 verbunden. Dieser Wandler kann

WARNUNG. Das Arbeiten mit Hochspannung kann tödlich sein. Der Bau der hier beschriebenen Schaltung ist nichts für Anfänger. Verwenden Sie sie nur, wenn Sie im Umgang mit Hochspannung erfahren sind!

eine Auflösung von 12 Bit plus Vorzeichenbit auf 13 Bit konvertieren. Bei 750 V ergibt dies eine Auflösung von etwa 200 mV pro Bit, bei 6 V etwa 1,6 mV/Bit. Ich halte dies für ausreichend für unsere Zecke. Die I²C-Adresse einer Satellitenplatine wird durch SJ1 und SJ2 auf der Platine eingestellt. Durch Schließen von SJ1 wird die Platine zu Kanal 1, durch Schließen von SJ2 zu Kanal 2 und durch Schließen beider Lötjumper zu Kanal 3.

Galvanische Trennung

Das von der Hauptplatine stammende Rechtecksignal wird dem Transformator auf der rechten Seite zugeführt (Verhältnis 1:1). Die Gleichrichter liefern dann eine Ausgangsspannung von etwas weniger als 6 V, die dann von den Parallelreglern auf etwa 5 V geregelt wird. Die Wicklungen des Transformators haben einen Isolationswert von 900 V, so dass die Gesamtisolation 1800 V_{Spitze} beträgt, vorausgesetzt, der Rest der Konstruktion ist in Ordnung. Das I²C-Signal verläuft über IC14, das einen Isolationswert von 4 kV_{Spitze} hat. Die Satellitenplatinen sind also in Bezug auf die Hauptplatine und in Bezug zueinander vollständig potentialfrei.

Korrektur der Satellitenplatine

Nachdem der erste Prototyp gebaut war, zeigten Tests, dass einige Korrekturen vorgenommen werden mussten. Das Übersprechen vom Schalt- auf das Messsignal in den Offset-Schaltern musste durch C41/R102 und C40/R103 reduziert werden. Die Schutzschaltung sprach auch auf Spikes an, zum Beispiel, wenn eine Messleitung eingesteckt wurde. Die Empfindlichkeit für solch Spikes wurde durch C33 verringert. D14

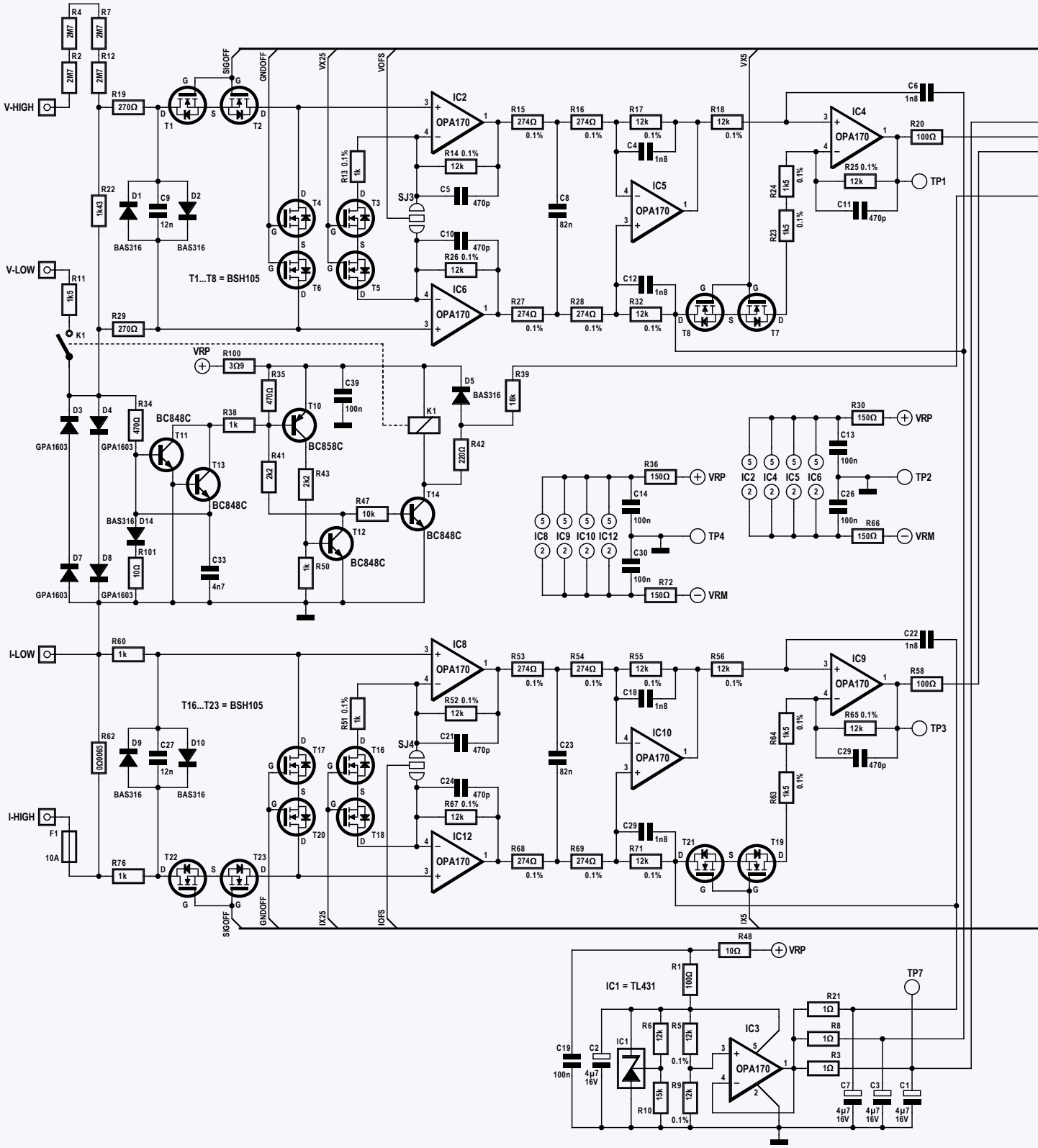
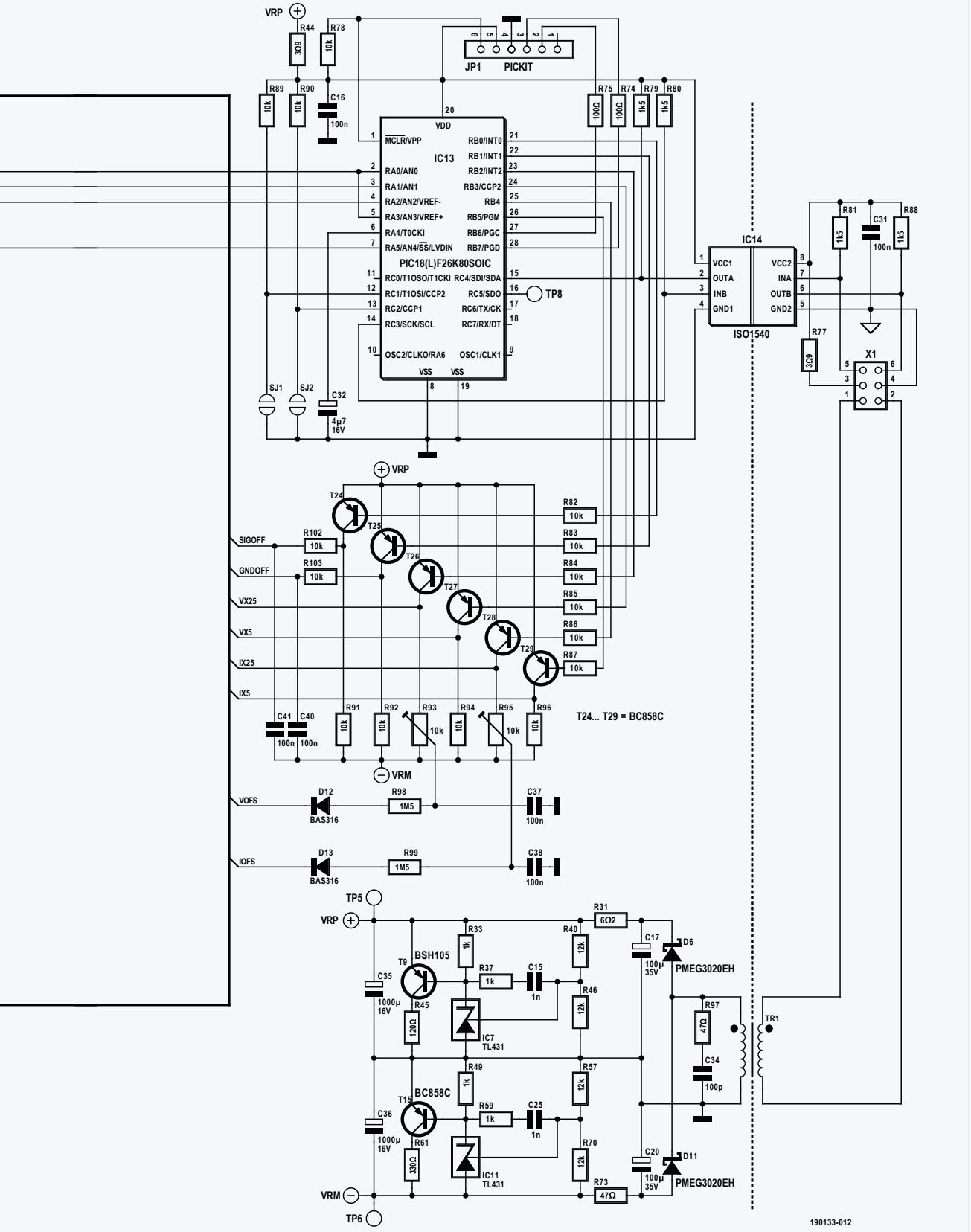


Bild 2. Schaltpläne der Hauptplatine.


VRP
R44
3Ω9
R78
10k
JP1
PICKIT
R89
R90
C16
100n
R75
R74
R79
R80
100Ω
1k5
IC13
PIC18(L)F26K80SOIC
MCLR/VPP
RA0/AN0
RA1/AN1
RA2/AN2/VREF-
RA3/AN3/VREF+
RA4/T0CKI
RA5/AN4/SS/LVDIN
RC0/T1OSO/T1CKI
RC1/T1OSI/CCP2
RC2/CCP1
RC3/SCK/SCL
OSC2/CLKO/RA6
RB0/INT0
RB1/INT1
RB2/INT2
RB3/CCP2
RB4
RB5/PGM
RB6/PGC
RB7/PGD
RC4/SDI/SDA
RC5/SDO
RC6/TX/CK
RC7/RX/DT
OSC1/CLK1
VDD
VSS
TP8
SJ1
SJ2
C32
4µ7
16V
IC14
ISO1540
VCC1
OUTA
INB
GND1
VCC2
INA
OUTB
GND2
R81
R88
C31
100n
R77
X1
VRP
T24
T25
T26
T27
T28
T29
R82
R83
R84
R85
R86
R87
R102
R103
SIGOFF
GNDOFF
VX25
VX5
IX25
IX5
R91
R92
R93
R94
R95
R96
C41
C40
T24... T29 = BC858C
VRM
D12
BAS316
R98
1M5
C37
VOFS
D13
R99
C38
IOFS
TP5
TP6
R31
6Ω2
R33
1k
R40
12k
C17
100µ
35V
D6
PMEG3020EH
T9
BSH105
C35
1000µ
16V
R37
C15
1n
R45
120Ω
R46
IC7
TL431
R97
47Ω
TR1
C34
100p
R49
R57
T15
BC858C
C36
R59
C25
R61
330Ω
R70
C20
D11
R73
IC11
190133-012

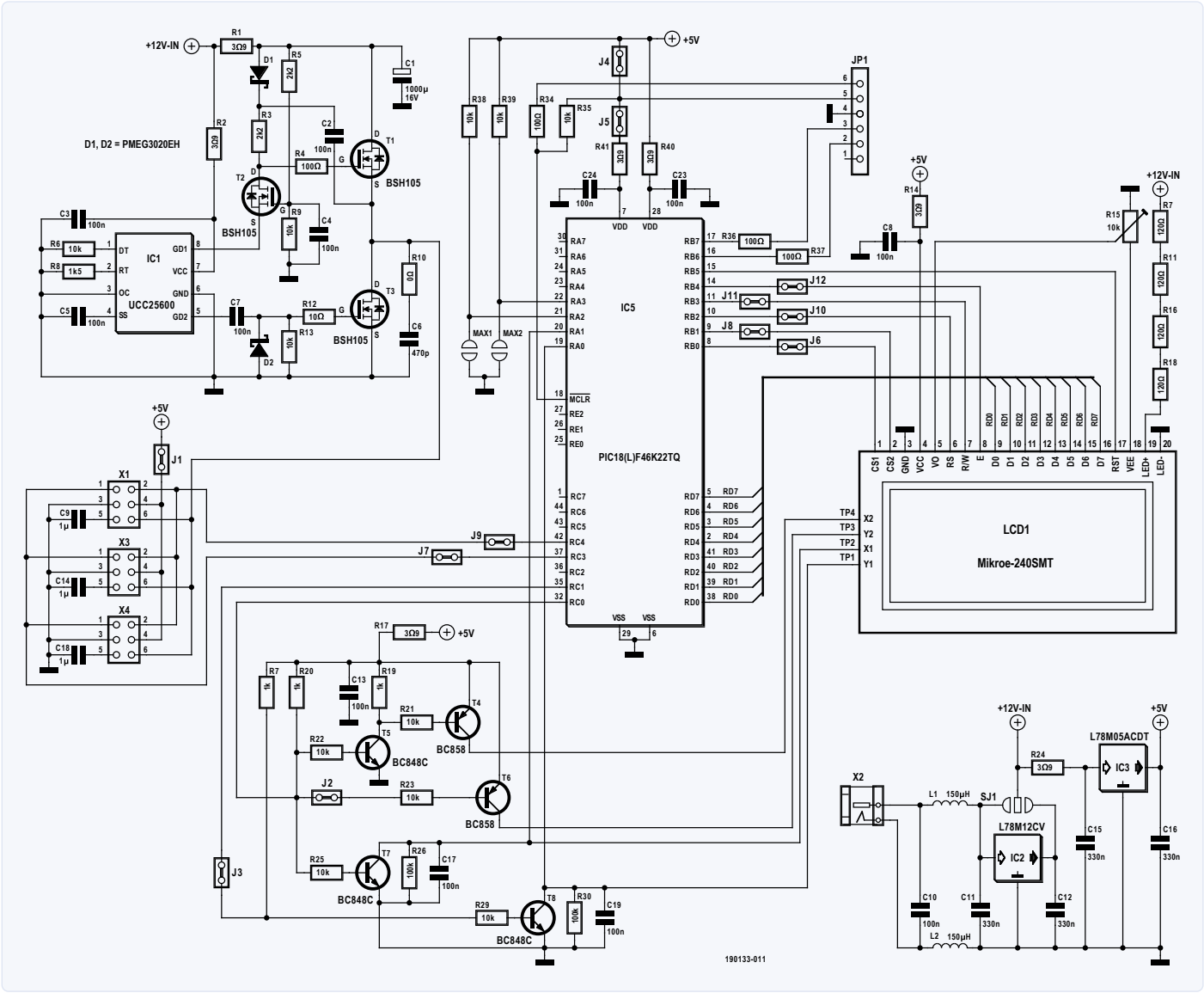


Bild 3. Schaltpläne der Satellitenplatine.

und R101 begrenzen auch die Störempfindlichkeit für positive Spitzen. Wenn die Verstärker auf die höchste Verstärkung von etwa 2750x eingestellt sind, ist eine Offset-Regelung unabdingbar. Dieser Regler wird bei niedriger(er) Verstärkung abgeschaltet. Abhängig vom Vorzeichen des Offsets muss mit den Lötbrücken SJ3/SJ4 eine Auswahl getroffen werden.

Leider hatte ich auch einige Entkopplungskondensatoren im Versorgungsteil vergessen, diese mussten ebenfalls hinzugefügt werden. Die Stückliste zu diesem Projekt im Open Office Format kann unter [2] heruntergeladen werden. Bestellnummern für Farnell sind angegeben, aber natürlich können Sie jeden anderen Lieferanten wählen.

Software für die Satellitenplatine

Das Programm ist in der relativ leicht verständlichen Programmiersprache mikroBasic für PIC (V5.6.1) von MikroElektronika geschrieben. Ich wollte, dass ein Frequenzbereich von (etwas) weniger als 50 Hz bis zu einigen hundert Hertz verarbeitet werden kann. Daher musste das Signal mindestens 30 ms lang abgetastet werden. Dann gibt es bei 50 Hz mindestens drei Nulldurchgänge, so dass man stets eine Periode erfassen könnte, wenn diese Nulldurchgänge äquidistant sind. Bei einem Tastverhältnis ungleich 50 % bei 50 Hz reichen 30 ms aber nicht aus, deshalb habe ich mich für 35 ms entschieden. Bei einem Tastverhältnis von weniger als 25 % kann es deshalb sein, dass die Frequenz bei 50 Hz nicht korrekt bestimmt wird, bei höheren Frequenzen gibt es dagegen kein Problem. Signale mit weniger als drei Nulldurchgängen während der Abtastperiode werden als DC behandelt.

Die zweite Frage ist, wie viele Abtastungen in diesen 35 ms vorgenommen werden. Ideal wäre eine Abtastfrequenz von 10 kHz oder höher. Ich habe mich für 357 Abtastwerte bei 10,2 ksamples/s entschieden. Dies führt zu einem ganzzahligen Vielfachen der Abtastwerte für 50-Hz und 60-Hz-Signale. Diese Frequenzen können mit höchster Genauigkeit analysiert werden.

Bild 4a. Die Hauptplatine und das an der Frontplatte angebrachte Display.

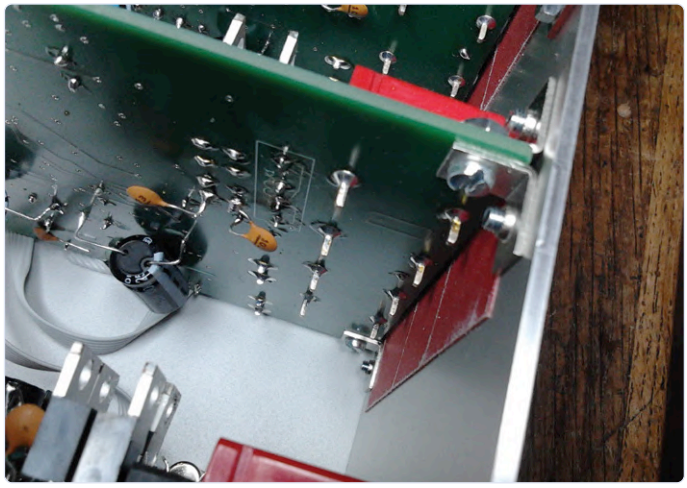


Bild 4b. Eine an der Vorderseite befestigte Satellitenplatine. Beachten Sie die zusätzliche Isolierung (dickes rotes Klebeband) zwischen der Platine und dem Panel.

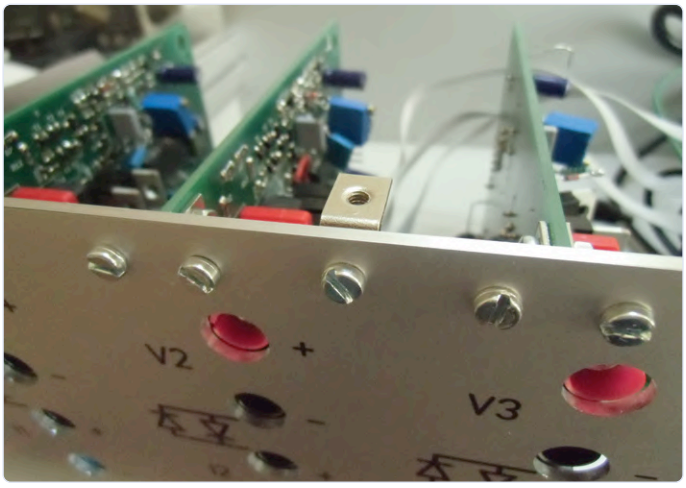


Bild 4c. Eine der zusätzlichen Stabilisierungswinkel zwischen Frontplatte und Abdeckung.

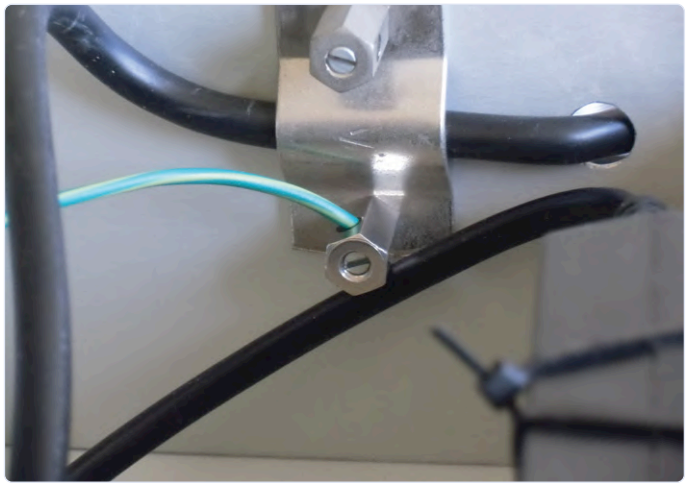


Bild 4d. PE-Verbindung

Es gibt eine Bibliothek mit I²C-Befehlen in mikroBasic, allerdings leider nur für einen I²C-Master. Für einen I²C-Slave musste ich eigene Prozeduren schreiben, um die Kommunikation mit dem I²C-Master auf der Hauptplatine einzurichten und zu steuern. Der Master fordert lediglich Informationen von den Satellitenplatinen an und zeigt sie an. (Auto-)Ranging und Offset, die gesamte Signalverarbeitung einschließlich Abtastung und Filterung wird auf der/den Satellitenplatine(n) durchgeführt.
Schnelle Fourier-Transformationen werden ebenfalls auf den Satellitenplatinen berechnet. Die Berechnungen sind nicht von mir erfunden, sondern von [3] „entlehnt". Die digitale Filterung basiert auf Kapitel 16 des *The Scientist and Engineer's Guide to Digital Signal Processing* [4]. Es wird ein Blackman-Fenster verwendet und ein Filter mit 13 Koeffizienten (M=12). Die Wahl der Länge des Filters ist ein Kompromiss zwischen Unterdrückung von Signalen über 2,55 kHz und Dämpfung von Signalen unter 2,55 kHz und auch zwischen Rechenzeit, Speicherbedarf und Leistungsanforderung an das Filter.

Montage

Es wurde eine Frontplatte entworfen (für eine Dreikanalversion), die bei Schaeffer [5] bestellt werden kann. Ich habe ein Metallgehäuse verwendet (Metcase M5503110 bei Farnell 1510827).

Bild 4 zeigt einige detaillierte Abbildungen der eingebauten Hardware, die einen Eindruck davon vermitteln, wie der Leistungsmesser aussehen kann und wie die Einheiten im Gehäuse befestigt sind. Beachten Sie das vorgefertigte Standard-Netzteil, das für die Stromversorgung des Power Analyzers verwendet wird.
In **Bild 4c** ist eine von zwei zusätzlichen kleinen Winkeln zu sehen, die ein Verbiegen der Frontplatte beim Ein- und Ausstecken der Messleitungen verhindern. Zur Befestigung der Halterungen muss also je ein zusätzliches Loch in die obere und untere Abdeckung gebohrt werden. Ich habe ein Metallgehäuse verwendet, so dass das Gehäuse an den Schutzleiter angeschlossen werden muss (**Bild 4d**).

WIE WIRD WAS WO ANGESCHLOSSEN?

Es besteht ein Unterschied zwischen dem Anschluss von Messleitungen zur Messung der Leistungsaufnahme einer Last oder der Messung der von einer Quelle gelieferten Leistung, wie in **Bild 5** zu sehen ist. Bei den angegebenen Methoden wird der Spannungsabfall der Anschlussleitungen und der Strommessstrecke eliminiert. Die Spannungsdifferenz zwischen den I-Anschlüssen und dem V-Anschluss muss unter 0,5 V bleiben, da sonst wird die Schutzschaltung auslöst und die Stromversorgung neu eingeschaltet werden muss.

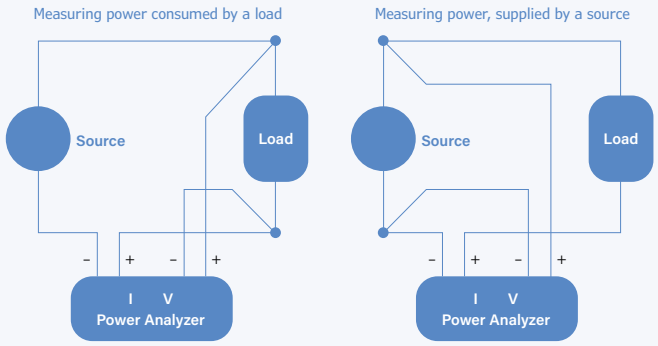


Bild 5. Anschluss der Messleitungen.

Bedienung

Nach dem Einschalten erscheint der Messbildschirm. Er zeigt einige Einheiten und deren Werte für den gewählten Kanal. Oben können Sie weitere Kanäle auswählen (falls verfügbar). Der Bildschirm kann maximal sieben Messwerte anzeigen, die im Konfigurationsbildschirm ausgewählt wurden. Das funktioniert so: Gehen Sie zum *Main Menu* (oben rechts auf dem Display) und wählen Sie *Configuration*, dann erscheint der Konfigurationsbildschirm mit 16 Werten zur Auswahl. Die gewählten Werte, die angezeigt werden sollen, werden hervorgehoben. Einige dieser Werte sind selbsterklärend, andere bedürfen einer Erläuterung. Vdis und Idis (Verzerrung) werden berechnet mit:

$$V_{dis} = \frac{\sqrt{\sum_{n=2}^{40} V_n^2}}{V_1}$$

wobei V_N für die Amplitude der Harmonischen N des Signals steht. Re-P steht für die Wirkleistung, P-VA für Scheinleistung (Vrms x Irms), Im-P für die Blindleistung und PF für den Leistungsfaktor, also Re-P / P-VA.

Wenn Sie *Eff->* wählen, gelangen Sie zum Effizienz-Bildschirm. Hier können Sie die Formel wählen, die das von Ihnen untersuchte Leistungsverhältnis beschreibt. Dann kehren Sie zum Konfigurationsbildschirm und schließlich zum Messbild zurück, so dass die gewählte Formel mit dem berechneten Wert für alle Kanäle angezeigt wird.

Im Konfigurationsbildschirm können Sie auch *Graphs->* wählen. Hier können Sie wiederum zwischen *Scope* oder *FFT* wählen. Ein Druck auf *Scope* bringt Sie zur Auswahl der Messkurven. Es wird immer Trace1 angezeigt. Eine ausgewählte Trace wird hervorgehoben dargestellt. Alle Messkurven können mit einem Kanal und mit V, I oder P dieses Kanals verknüpft werden. Die Anzeige wird von einem positiven Nulldurchgang von Trace1 „getriggert". Die *Scope*-Anzeige gibt Ihnen einen Eindruck von der Wellenform der Signale und ihrer Phasenlage. Es werden zwei Perioden des Signals angezeigt. Damit man die Signale voneinander unterscheiden kann, werden Spannungssignale mit der maximal darstellbaren Amplitude, Stromsignale mit 80 % und Leistungssignale mit 60 % angezeigt. **Bild 6** zeigt dies am Beispiel der Wellenformen eines Transformators, der in die Sättigung geht. Die im Display gezeigte Amplitude sagt also nichts über die Absolutwerte aus!

Zurück im *Graphs*-Screen wählen Sie *FFT*, um die FFT-Komponenten eines ausgewählten Signals anzuzeigen. Oben können Sie den Kanal, die LINeare oder LOGarithmische Anzeige und entweder V oder I wählen. Die größte Harmonische (normaler-, aber nicht notwendigerweise die erste) wird bei 100 % oder 0 dB angezeigt. Die horizontale Skala gibt die Zahl der Harmonischen an, nicht ihre Frequenz.

In der *Logging*-Funktion müssen Sie zunächst die Zeitperiode auswählen, die Sie anwenden möchten. Mögliche Werte liegen zwischen 0,25 h und 128 h, wobei sich die Schritte verdoppeln. Dann wählen Sie aus, wie viele Messkurven Sie protokollieren möchten (maximal drei). Jede Kurve kann mit einem Kanal verknüpft werden, und von diesem Kanal aus können Sie entweder V, I oder P wählen. Beim Druck auf *Continue* wird die Aufzeichnung sofort gestartet. Es wird ein alphanumerischer Bildschirm angezeigt, auf dem Sie den Durchschnitts-, Maximal- und Minimalwert sowie die verstrichene Zeit verfolgen können. Sie können auch zum *Graphs*-Bildschirm wechseln. Oben können Sie dort eine andere Messkurve auswählen, falls diese eingeschaltet ist. Wenn Sie *Return* drücken, gelangen Sie zurück in den Hauptbildschirm, die gesammelten Daten gehen dann verloren.

Auf dem *Configuration*-Bildschirm können Sie den Scheitel- oder Crest-Faktor (*V-CF* oder *I-CF*) wählen. Er dividiert einfach den höchsten Spitzenwert (positiv oder negativ) durch den RMS-Wert des Signals. Denken Sie daran, dass die Bandbreite der Messverstärker 2,5 kHz beträgt, so dass diese Funktion nicht für Audiosignale geeignet ist, sondern nur für Frequenzen bis etwa 250 Hz.

Wie man kalibriert und justiert

Vom Hauptbildschirm aus erreichen Sie den Kalibrierungsbildschirm und wählen dort zwischen Kalibrierung des Touchscreens und der

BAU DES TRAFOS

Zutaten

1 Spulenkörper für EF20
Draht (900 V Isolation und max. 1,3 mm Durchmesser)
2 Kernhälften für EF20
2 Clips für EF20

Man nehme

- den Spulenkörper und schneide die Stifte 2, 3, 4, 7, 8 und 9 ab.
- 9 Windungen des Drahts und wickle sie zwischen Stift 1 und Stift 10. Der Draht sollte genau eine Lage füllen.
- 9 Windungen des Drahts und wickle sie zwischen Stift 5 und Stift 6. Der Draht sollen genau eine Lage füllen.
- die Kernhälften und setze sie ein.
- die Clips und stecke sie auf. Es gibt keinen Luftspalt.

Wenn alles gut gegangen ist, hat man einen Transformator mit einer gleichen Primär- und Sekundärinduktivität von etwa 125 µH. Die Kapazität zwischen Primär- und Sekundärwicklung dürfte im Bereich von 30 pF liegen.

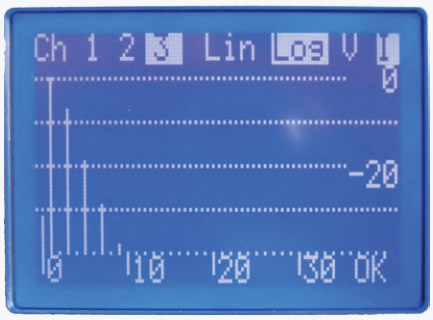


Bild 6a. Messung eines Transformators in Sättigung, FFT des Stroms.

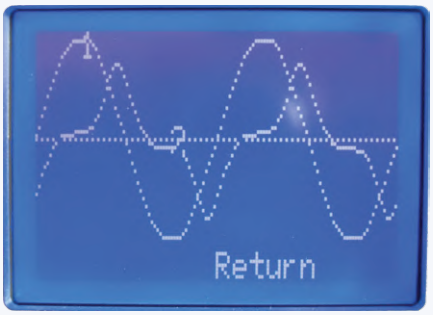


Bild 6b. Messung eines Transformators in Sättigung, Spannung und Strom.

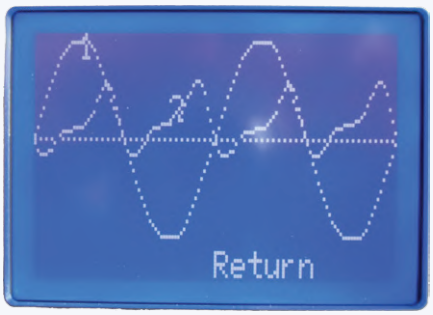


Bild 6c. Messung eines Transformators in Sättigung, Spannung und Leistung.

Kanäle. Beim ersten Start des Geräts wird automatisch die Touchscreen-Kalibrierung angezeigt. Verwenden Sie zur Kalibrierung des Touchs einen weichen (Blei-)Stift.

Bei der Kanalkalibrierung soll zunächst ein Kanal ausgewählt werden. Dann gibt die Software einen Bildschirm vor, in dem Spannung, Strom und Frequenz gemessen und eingestellt werden können. In diesem Bildschirm ist die automatische Bereichswahl für V und I deaktiviert, so dass die Kalibrierung für jeden Bereich leicht vorgenommen werden kann. Mit den Pfeilen (< und >) können Sie die Bereiche ändern und den Messwert anpassen.

Die Frequenzmessung wird am besten mit einem Digitaloszilloskop an TP8 (Signal) und TP2 (GND) auf der Satellitenplatine des gewählten Kanals vorgenommen. Stellen Sie die sichtbare Impulsbreite mit den Pfeilen auf der Frequenzlinie auf 35,00 ms ein. Die zweitbeste Methode zur Einstellung der Frequenz besteht darin, eine Signalquelle mit einer kalibrierten Frequenz von 50 Hz oder 60 Hz zu verwenden und den Messwert auf 50,00 Hz oder 60,00 Hz einzustellen. Der Messwert darf um ±0,5 % variieren.

Wenn die Kalibrierung eines Kanals abgeschlossen ist, drücken Sie *OK*, so dass die Kalibrierdaten auf der Satellitenkarte des gewählten Kanals gespeichert werden.

Soweit es mich betrifft, ist dieses Projekt abgeschlossen, es wird keine größeren Ergänzungen oder Änderungen mehr geben. Aber natürlich werde ich jede Frage beantworten.

Dieser Artikel basiert auf den Informationen, die auf der Elektor Labs-Projektseite unter [2] präsentiert werden. Auf dieser Seite können detailliertere Informationen zu diesem Power Analyzer einschließlich Software, PCB-Design-Dateien und Stückliste gefunden und heruntergeladen werden. Die Eagle-CAD-Dateien auf Elektor Labs sind stets auf dem neusten Stand der letzten Änderungen, aber neuere Versionen der Platinen wurden nicht gebaut und/oder getestet!

190133-B-03

Ein Beitrag von
Idee, Gestaltung, Text und Illustrationen: **Wil Dijkman**
Schaltbilder: **Patrick Wielders**
Redaktion: **Luc Lemmens, CJ Abate**
Übersetzung: **Rolf Gerstendorf**
Layout: **Giel Dols**

Sie haben Fragen oder Kommentare?
Haben Sie Fragen oder Anmerkungen zu diesem Artikel? Schicken Sie eine E-Mail an den Autor unter w.j.dijkman@onsbrabantnet.nl oder kontaktieren Sie Elektor unter editor@elektor.com.

WEBLINKS

[1] **AC/DC-Leistungsmesser**: www.elektormagazine.de/magazine/elektor-201509/27998
[2] **Power Analyzer auf Labs**: www.elektormagazine.com/labs/power-analyzer
[3] **FFT-Berechnung**: www.nicholson.com/dsp.fft1.html
[4] **The Scientist and Engineer's Guide to Digital Signal Processing, by Steven W. Smith, Ph.D.**: www.dspguide.com/pdfbook.htm

Design analoger Filter (Teil 3)

Passive Filter

Von **Alfred Rosenkränzer**

Im dritten und letzten Teil dieser Grundlagen-Reihe geht es um die Feinheiten passiver Filter. Da hier nur rein passive Bauteile wie Widerstände, Spulen und Kondensatoren vorkommen, ist keine Verstärkung machbar. Somit lässt sich auch keine hohe Eingangs- und niedrige Ausgangsimpedanz verwirklichen. Dafür sind hohe Frequenzen im dreistelligen MHz-Bereich und darüber wenig problematisch.

Bei aktiven Filtern kann die Eingangsimpedanz durch einen Puffer sehr hoch ausfallen und ein Puffer am Ende die Ausgangsimpedanz sehr niederohmig machen. Folglich muss man sich nicht um den korrekten „Abschluss“ bzw. eine adäquate Anpassung der Impedanzen an Ein- und Ausgang kümmern. Bei sehr hohen Frequenzen aber sind nach wie vor passive Filter üblich. Diese

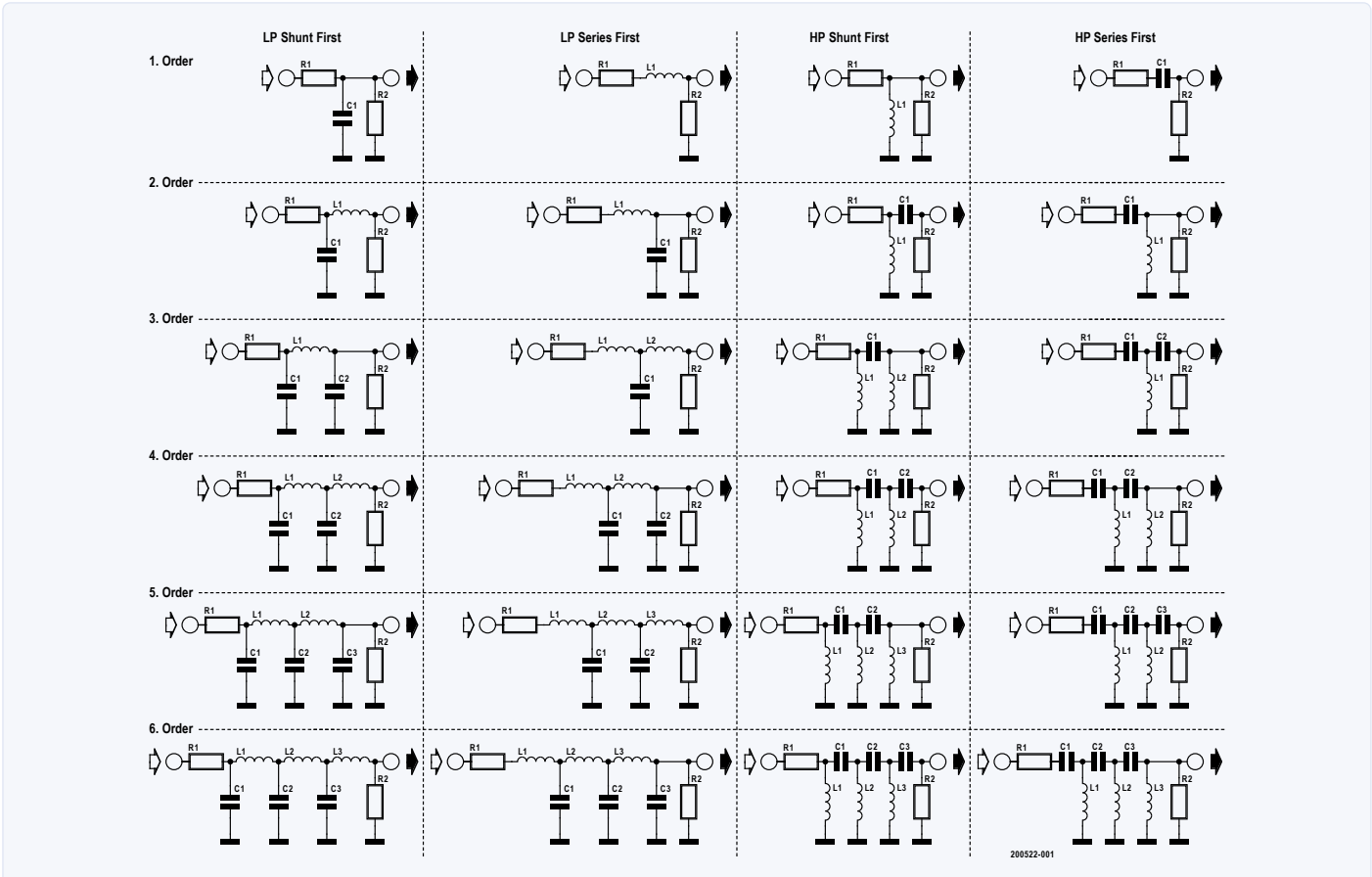


Bild 1. Grundschaltungen passiver Tief- und Hochpässe in PI-und T-Anordnung 1. bis 6. Ordnung. Nicht nur die Grenzfrequenz ist von den Werten der Bauteile abhängig, sondern auch die Filtercharakteristik. Die abgebildeten Strukturen eignen sich für Bessel-, Butterworth- oder Tschebyscheff-Filter.

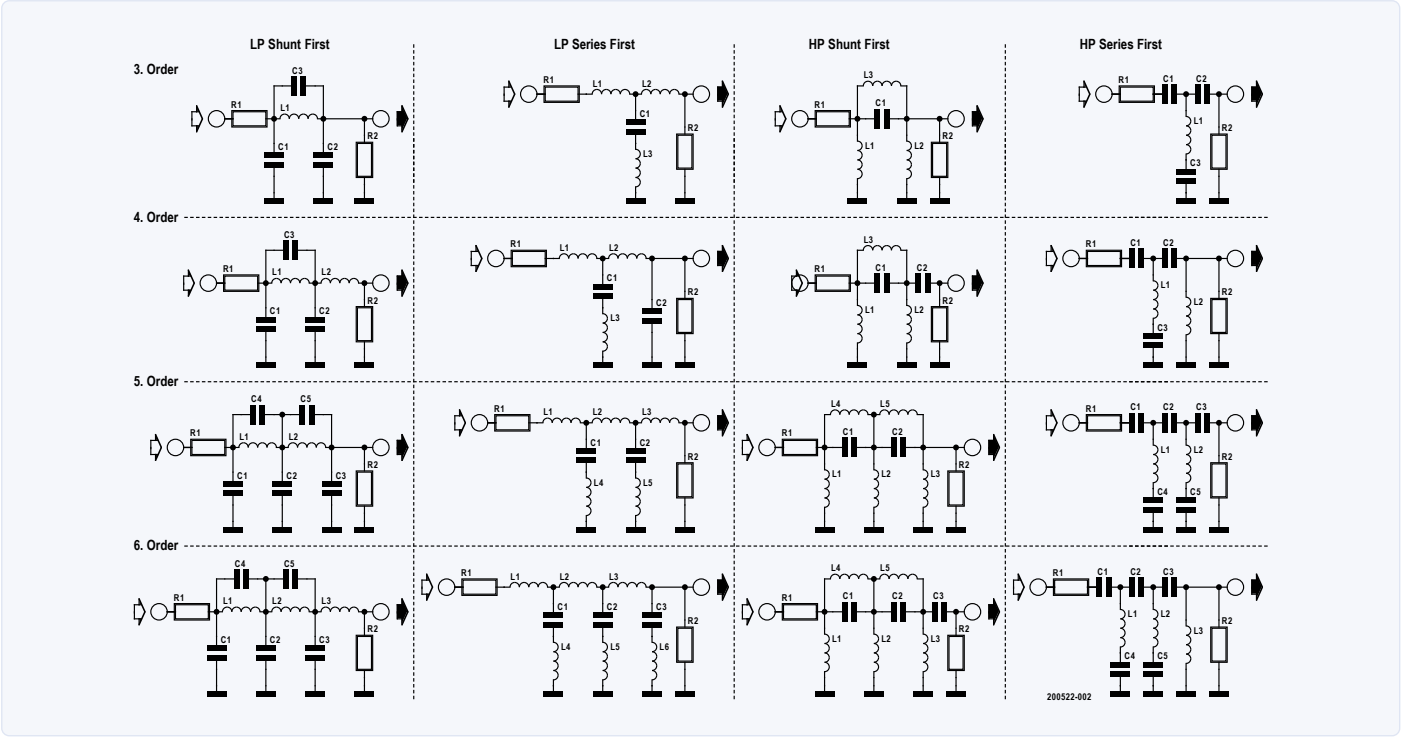


Bild 2. Grundschaltungen von Tief- und Hochpässen der 3. bis 6. Ordnung für Tief- und Hochpässe mit Cauer- oder Invers-Tschebyscheff-Charakteristik in PI- und T-Anordnung.

Filter werden für eine bestimmte Eingangs- und Ausgangsimpedanz (die nicht gleich sein muss) entworfen. Eine Abweichung von der angenommenen Quell- und Abschlussimpedanz kann die Filtereigenschaften daher deutlich beeinflussen. Nachfolgend ist beschrieben, welche Arten von passiven Filtern gebräuchlich sind und worauf es dabei ankommt.

Strukturen

Bild 1 zeigt die grundsätzlichen Strukturen von Tief- und Hochpässen der 1. bis 6. Ordnung. Durch geeignete Dimensionierung der Bauteile sind mit diesen Strukturen Filter mit Bessel-, Butterworth- oder Tschebyscheff-Charakteristik möglich. Man kann dabei wählen, ob ein Filter mit einem Bauteil in Serie zum Rest beginnt (T-Struktur) oder mit einem Bauteil, das gegen Masse geschaltet ist (PI-Struktur).

Die linke Spalte von Bild 1 zeigt PI-Tiefpässe, die alle mit einem Kondensator gegen Masse am Eingang beginnen. Rechts daneben sind T-Tiefpässe, die daher mit einer Spule „in Serie“ beginnen.

Bei den Hochpässen ist es umgekehrt: In der dritten Spalte sind PI-Hochpässe mit einer Spule nach Masse und in der rechten Spalte sind T-Hochpässe mit einem Serien-Kondensator zu sehen.

Für höhere Ordnungen werden abwechselnd Bauteile „längs“ bzw. seriell oder „quer“ bzw. gegen Masse geschaltet hinzugefügt. Dabei ändern sich aber die Werte der weiteren Bauteile. Ob man eine T- oder eine PI-Struktur wählt, ist für die Funktion unerheblich. Da Spulen bei Elektronikern nicht so beliebt sind, wird oft versucht, mit so wenig Induktivitäten wie möglich auszukommen.

In **Bild 2** sind die grundsätzlichen Strukturen von Cauer- (Eliptic) und Invers-Tschebyscheff-Filtern (der sogenannte Typ 2) zu sehen. Spulen bzw. Kondensatoren sind durch Parallel- oder Serienschwingkreise ersetzt. Dargestellt sind Tief- und Hochpässe in PI- und T-Anordnung der 3. bis 6. Ordnung.

In **Bild 3** sind die grundsätzlichen Strukturen von Bandpässen und Bandsperren mit Bessel-, Butterworth- oder Tschebyscheff-Charakteristik (3. bis 7. Ordnung) und in **Bild 4** entsprechend komplexer mit Cauer und Invers-Tschebyscheff Charakteristik (5. und 7. Ordnung) dargestellt. Auch hier gibt es PI- und T-Varianten.

Dimensionierung

Ein Bessel- oder Butterworth-Filter ist vollständig durch die Filterapproximation, die -3-dB-Grenz-Frequenz, die gewählte Struktur, die Ordnung sowie Ein- und Ausgangsimpedanz definiert. Im Gegensatz zu aktiven Filtern gibt es bei passiven keine Freiheitsgrade zur Änderung von Bauteilwerten. Wenn man z.B. den Wert einer Spule auf einen Standardwert eine E-Reihe setzen will, muss man hierfür bei gegebener Impedanz die Grenzfrequenz des Filters leicht verändern. Das kann man natürlich nur machen, wenn es die Applikation zulässt.

Wie sich die Eigenschaften eines Butterworth-Filters durch die Wahl verschiedener Ordnungen verändern, wurde schon in Teil 1 dieser Serie [1] gezeigt. Nun ist es interessant zu beobachten, wie sich die Bauteilewerte durch Variation der Grenzfrequenz verändern. **Bild 5** zeigt die fertig dimensionierte Schaltung eines Butterworth-Tiefpasses 5. Ordnung in PI-Struktur mit einer Grenzfrequenz von 1 MHz. Will man die Grenzfrequenz auf 2 MHz verdoppeln, muss man lediglich die Werte der Spulen und Kondensatoren halbieren. Das ist auch kein Wunder, denn die Grenzfrequenz ist proportional $1 / \sqrt{(L * C)}$. Daher war es auch schon vor dem Zeitalter von Taschenrechnern oder PCs leicht möglich, eine Grundschaltung mit gegebener Grenzfrequenz auf die gewünschte Zielfrequenz

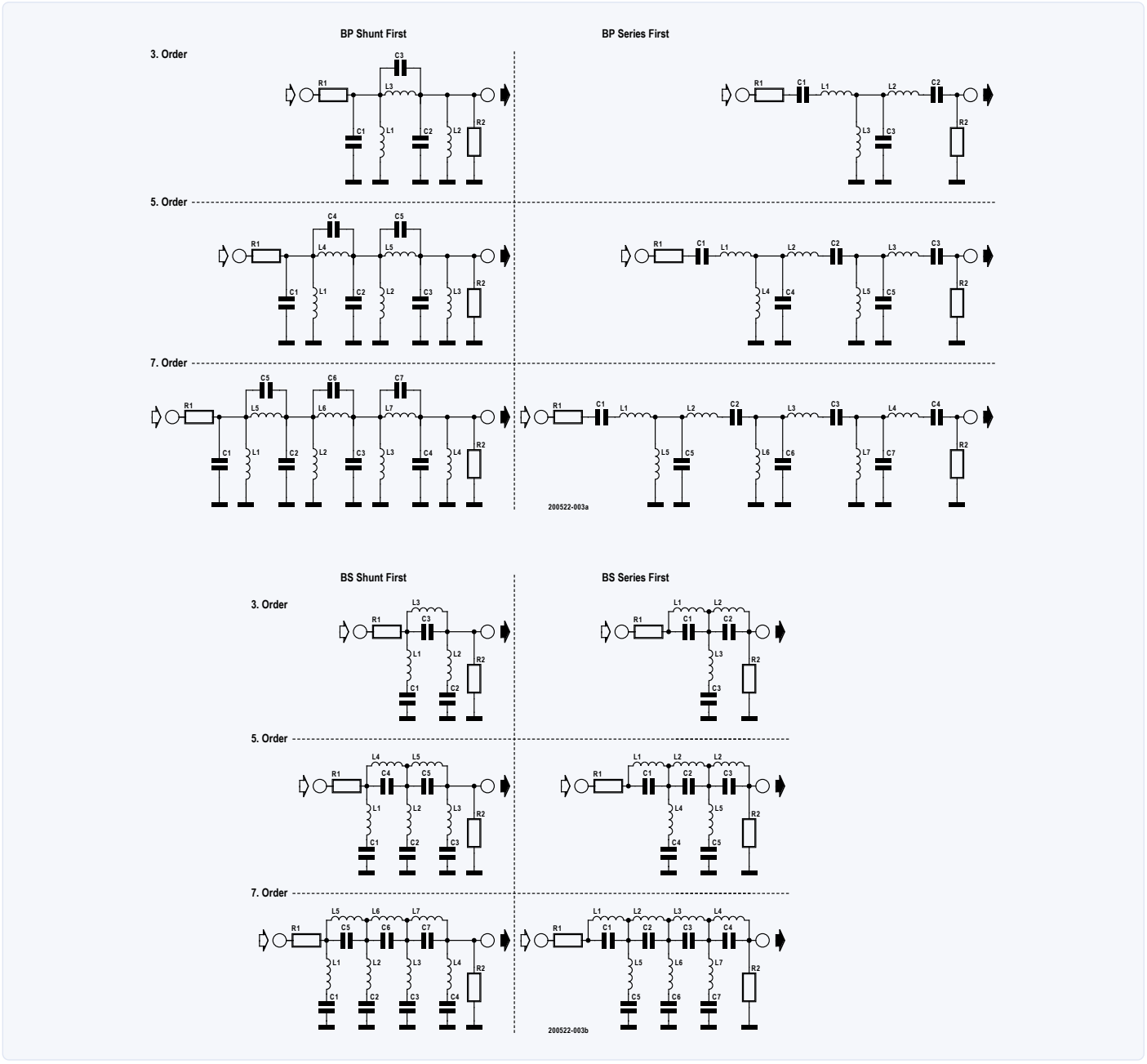


Bild 3. Grundschaltungen von Bandpässen und Bandsperren der 3., 5. und 7. Ordnung mit Bessel-, Butterworth- oder Tschebyscheff-Charakteristik in PI- und T-Anordnung.

umzurechnen. Bei gleich bleibender Standard-Impedanz von 50 Ω bleiben selbstverständlich die Widerstandswerte gleich.

Interessant ist auch, wie sich die Bauteilewerte verhalten, wenn man ein Filter für andere Impedanzen dimensionieren will. Auch das ist ohne den Einsatz von Großrechnern möglich: Will man die Impedanzen von den 50 Ω des 1-MHz-Tiefpasses in Bild 5 auf 100 Ω verdoppeln, dann verdoppeln sich logischerweise nur die Spulenwerte, doch die Kondensatorwerte halbieren sich. Auch das ist kein Wunder, denn Z ist proportional zu √(L/C).

Da es sich gerade anbietet möchte ich hier zeigen, welche Auswirkung ein falscher Quell- und Abschlusswiderstand haben kann. Das für eine Impedanz von 100 Ω dimensionierte Filter wurde exemplarisch „falsch" mit einer Quelle mit 50 Ω angesteuert und auch am Ausgang mit 50 Ω abgeschlossen. **Bild 6** zeigt den Frequenzgang im Durchlassbereich.

Zurück zur Dimensionierung: Bei einem Tschebyscheff-Filter kommt die Welligkeit (Ripple) im Durchlassbereich als weiterer Parameter hinzu. Wie sich das auf den Frequenzgang auswirkt zeigt **Bild 7**. Dargestellt sind die Amplitudenverläufe von 1-MHz-Tschebyscheff-Tiefpässen 7. Ordnung mit einem Ripple im Durchlassbereich von 0,1 dB, 0,5 dB, 1 dB und 3 dB. Je mehr Ripple man zulässt, desto steiler verläuft die Kurve in den Sperrbereich. Wie viel Ripple man in Kauf nehmen kann, muss die Anwendung entscheiden.

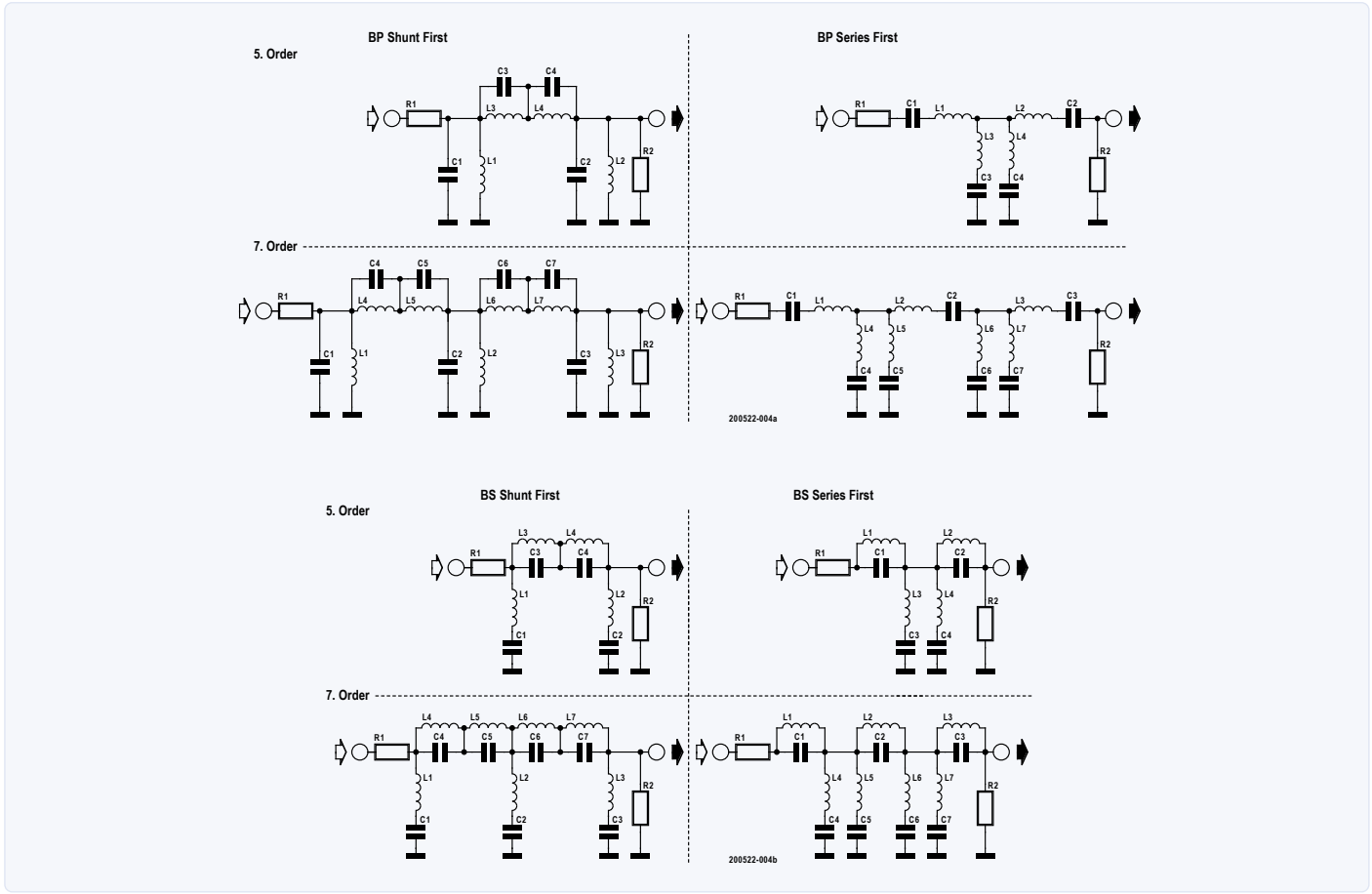


Bild 4. Grundschaltungen von Bandpässen und Bandsperren der 5. und 7. Ordnung mit Cauer- oder Invers-Tschebyscheff-Charakteristik in PI- und T-Anordnung.

Bild 8 zeigt eine vergrößerte Version von Bild 7 im Durchlassbereich. Hier sieht man, dass die Grenzfrequenz nicht wie bei anderen Filtern der -3-dB-Punkt ist, sondern der Punkt der Kurve, an dem der definierte Ripple unterschritten wird. Möchte man die Kurven zur besseren Vergleichbarkeit auf den -3-dB-Punkt normieren, muss man die Grenzfrequenz entsprechend anpassen.
Beim Cauer-Filter kommt noch die minimale Dämpfung im Sperrbereich als weiterer Parameter hinzu. Durch die Notches

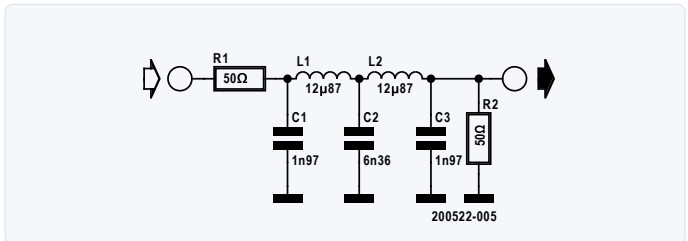


Bild 5 Butterworth-Tiefpass 5. Ordnung in PI-Struktur mit einer Grenzfrequenz von 1 MHz.

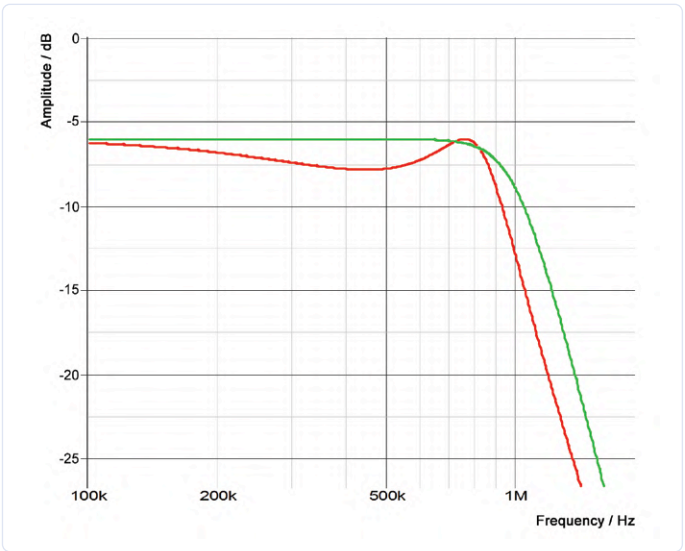


Bild 6. Frequenzgang eines Butterworth-Tiefpasse 5. Ordnung im Durchlassbereich. Die grüne Line entspricht dem korrekten Verlauf und die rote Linie ergibt sich bei Fehlanpassung eines 100-Ω-Filters mit 50-Ω-Abschlüssen.

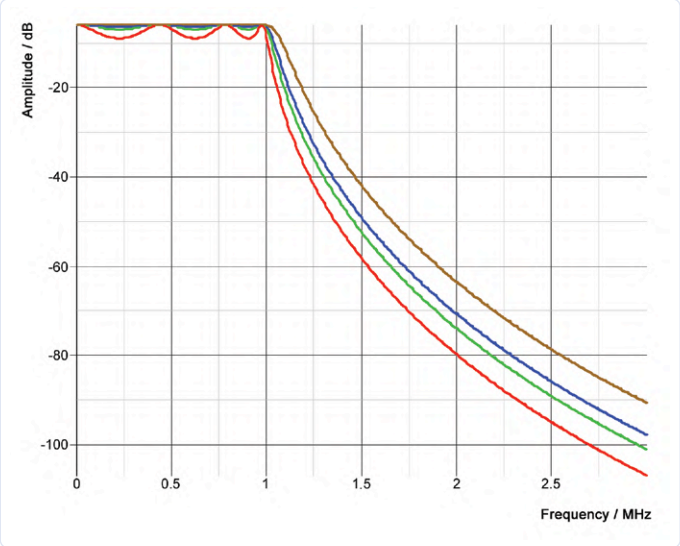


Bild 7. Frequenzgänge von 1-MHz-Tschebyscheff-Tiefpässen 7. Ordnung mit einem Ripple im Durchlassbereich von 0,1 dB (braun), 0,5 dB (blau), 1 dB (grün) und 3 dB (rot).

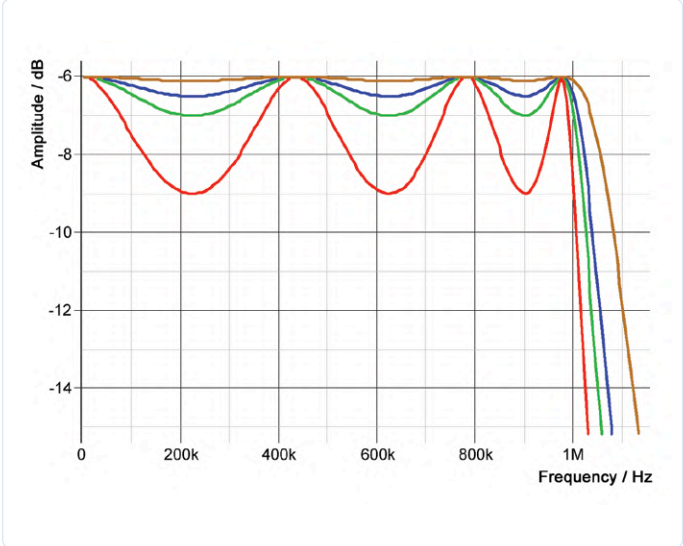


Bild 8. Eine vergrößerte Version von Bild 7 im Durchlassbereich. Die Grenzfrequenz entspricht dem Punkt der Kurve, an dem der definierte Ripple unterschritten wird. Ripple: 0,1 dB (braun), 0,5 dB (blau), 1 dB (grün) und 3 dB (rot).

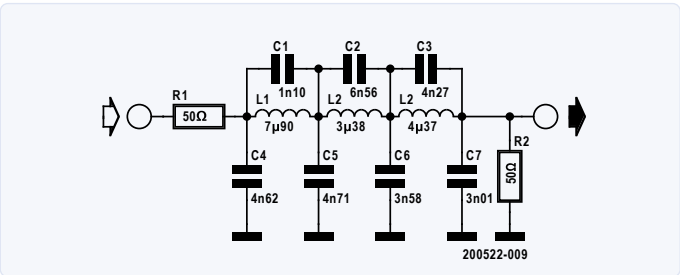


Bild 9. Cauer-Tiefpass 7. Ordnung mit einem Ripple von 0,5 dB im Durchlassbereich und 40 dB Mindestdämpfung. Die Bauteilewerte für weitere Dämpfungen finden sich in Tabelle 1.

wird zwar noch ein steilerer Übergang in den Sperrbereich erreicht, doch landet die Kurve im weiteren Verlauf wieder bei niedrigeren Dämpfungen. Die Anzahl der Notches stimmt mit der Anzahl der Schwingkreise im Schaltplan überein (sofern sie verschiedene Frequenzen haben). **Bild 9** zeigt einen Cauer-Tiefpass 7. Ordnung mit einer Mindestdämpfung von 40 dB. **Tabelle 1 Bauteilewerte für Bild 9** enthält die Werte für die Mindestdämpfungen 40, 50, 60 und 70 dB. Die Höhe der „Hügel" entspricht der minimalen Dämpfung.

Bild 10 zeigt die Frequenzgänge alle vier Cauer-Tiefpässe. Die Höhe der „Hügel" entspricht der minimalen Dämpfung. Man sieht die verschiedene Steilheit der Kurven im Übergangsbereich und die

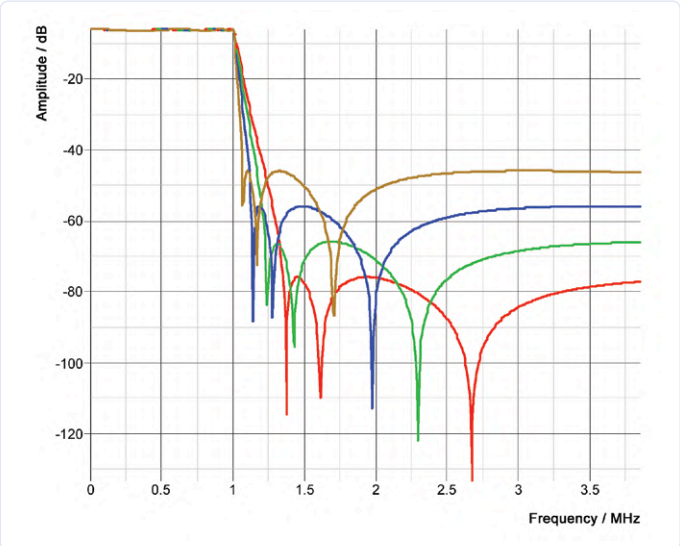


Bild 10. Frequenzgänge von 1-MHz-Cauer-Tiefpässen 7. Ordnung mit den Mindestdämpfungen 40, 50, 60 und 70 dB. Die Farben sind selbsterklärend.

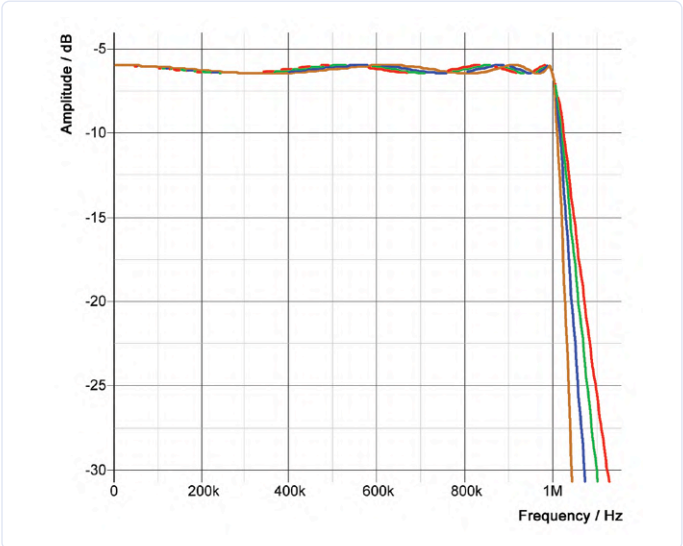


Bild 11. Vergrößerung von Bild 10 im Durchlassbereich mit den Mindestdämpfungen 40 dB (braun), 50 dB (blau), 60 dB (grün) und 70 dB (rot).

Tabelle 1. Bauteilewerte für Bild 9.

Dämpfung	C1	C2	C3	C4	C5	C6	C7	L1	L2	L3
40 dB	1n10	6n56	4n27	4n62	4n71	3n58	3n01	7µ90	3µ38	4µ37
50 dB	766p	4n10	2n77	4n90	5n48	4n52	3n61	8µ50	4µ78	5µ61
60 dB	541p	2n74	1n88	5n07	6n14	5n36	4n09	8µ92	6µ04	6µ65
70 dB	385p	1n90	1n31	5n20	6n68	6n07	4n45	9µ22	7µ11	7µ49

damit verbundenen Mindestdämpfungen. Damit verändern sich aber auch die Frequenzen der Notches. Hat man eine feste Störfrequenz im zu filternden Spektrum, kann man einen Notch entweder über die Anpassung der Grenzfrequenz oder der Mindestdämpfung darauf schieben. In diesem Fall sollte man aber die Bauteiletoleranzen und damit die genaue Lage der Notches im Auge behalten. Der vergrößerte Ausschnitt der Frequenzgänge in **Bild 11** beweist, dass der Ripple bei allen Filtern 0,5 dB beträgt.

Die letzte Filterapproximation ist die Invers-Tschebyscheff-Charakteristik (IT). Sie ähnelt im Durchlassbereich einem Butterworth-Filter und hat also keinen Ripple, weshalb dieser Parameter entfällt. Im Sperrbereich hingegen entspricht die IT-Charakteristik mehr einem Cauer-Filter mit Notches und entsprechender Mindestdämpfung. Die Grundschaltung ist die gleiche wie beim Cauer-Filter (Bild 9), lediglich die Dimensionierung ist anders. **Bild 12** zeigt die Frequenzgänge von Invers-Tschebyscheff-Filtern mit den Mindestdämpfungen 50, 60 und 70 dB. **Bild 13** ist die im Durchlassbereich gezoomte Version von Bild 12.

Allpässe

In der analogen Fernsehtechnik wurden (einst) steile Filter gebraucht, doch die starken Überschwinger der Sprungantwort störten erheblich. Daher wurde der Verlauf der Gruppenlaufzeit durch Allpässe „aufgefüttert“. Der Mehraufwand dafür war recht groß, und die Filter mussten manuell mit Hilfe eines Netzwerkanalysators abgeglichen werden. Es gibt nur wenige Filterprogramme, die das Gesamtfilter berechnen können. Durch die zunehmende Digitalisierung auch der Videotechnik mit höheren Taktraten und Oversampling in DACs und ADCs sind die Ansprüche an analoge Filter deutlich zurückgegangen, so dass man heute auf den Ausgleich der Gruppenlaufzeit verzichten kann. **Bild 14** zeigt die Schaltung eines passiven Allpasses 2. Ordnung und **Bild 15** den zugehörigen Verlauf der Gruppenlaufzeit.

Nun eine kurze Stippvisite in die Praxis: In **Bild 16** ist ein selbstgebauter Video-Filter mit Neosid-Spulen (die kupferfarbenen Quader mit abstimmbaren Kern) zu sehen. Die beiden Spulen links sind Bestandteil eines Cauer-Tiefpasses 5. Ordnung. Der Block aus den sechs restlichen Spulen gehört zu einem Allpass. Es sind immer mehrere Kondensatoren parallel geschaltet, um die notwendigen „krummen“ Werte zu erreichen. Die runden blauen Exemplare sind KP-Kondensatoren im nF-Bereich mit einer Toleranz von 2 %.

Besondere Filter

Die bisher betrachteten Filter waren „single ended“ – ihre Signale waren immer auf Masse bezogen. Wie nachfolgend deutlich wird, kann man aber auch mit passiven Bauteilen differentielle Signale filtern.

Differentielle Filter

Moderne ADCs und DACs für höhere Signalfrequenzen haben differentielle Ein- bzw. Ausgänge. Es ist daher naheliegend, erforderliche

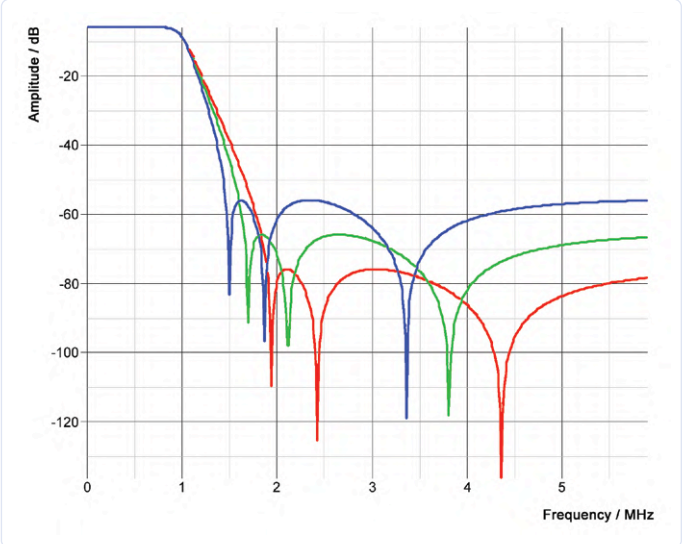


Bild 12. Frequenzgänge der Invers-Tschebyscheff-Filter mit 50, 60 und 70 dB. Die Farben sind selbsterklärend.

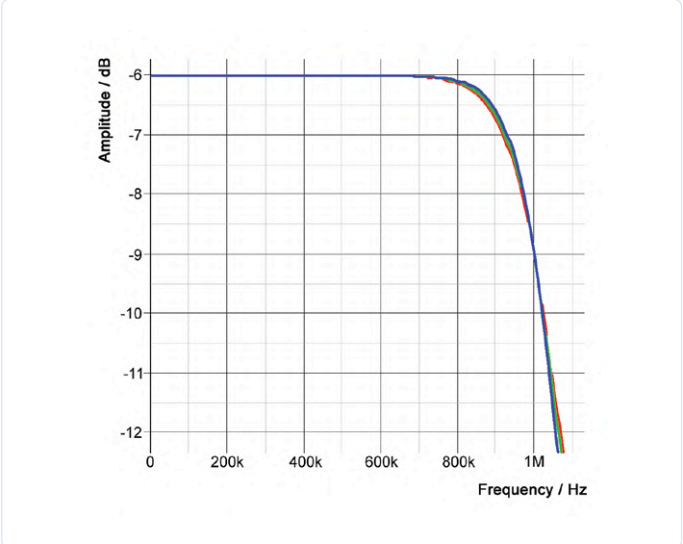


Bild 13. Vergrößerung von Bild 12 im Durchlassbereich mit den Mindestdämpfungen 50 dB (blau), 60 dB (grün) und 70 dB (rot).

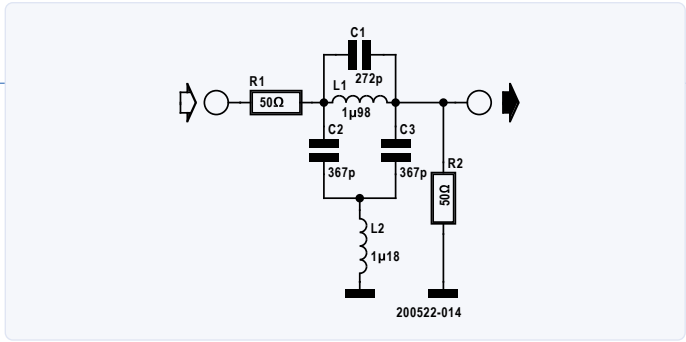


Bild 14. Schaltung eines passiven Allpasses 2. Ordnung.

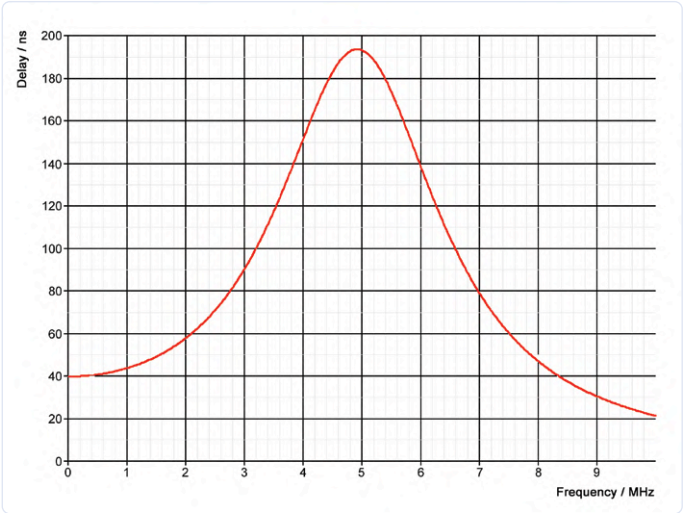


Bild 15. Verlauf der Gruppenlaufzeit des passiven Allpasses 2. Ordnung von Bild 14.

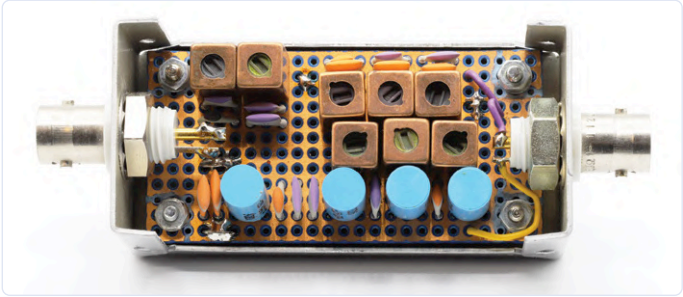


Bild 16. Selbstgebauter Video-Filter mit acht Neosid-Spulen.

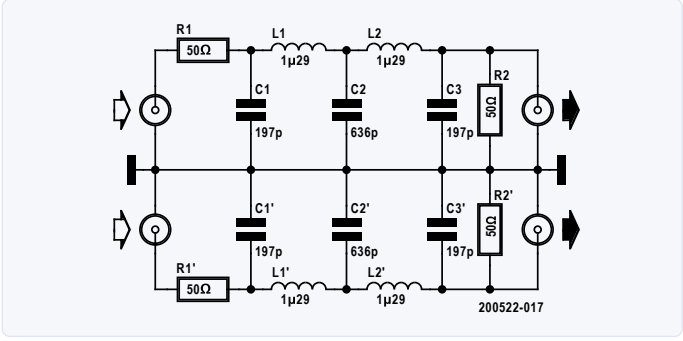


Bild 17. Schaltung zweier „normaler" Butterworth-Tiefpässe zur Filterung eines differentiellen Signals.

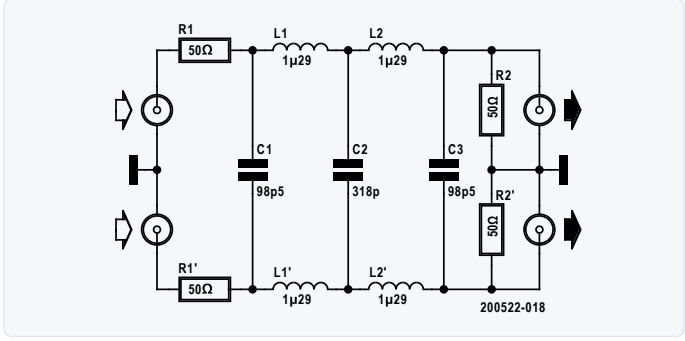


Bild 18. Diese Schaltungsvariante fasst je zwei gegen Masse geschaltete Kondensatoren Cx und Cx' von Bild 17 zu jeweils einem Kondensator mit halber Kapazität der Einzelkondensatoren zusammen.

Tiefpässe direkt am DAC-Ausgang und sogar das Anti-Aliasing-Filter vor dem ADC-Eingang gleich differentiell auszuführen. Bei der Schaltung von **Bild 17** handelt es sich schlicht und einfach um zwei identische Butterworth-Tiefpässe 5. Ordnung für 10 MHz in konventioneller Single-Ended-Ausführung. Sie haben kein gemeinsames Bauteil (außer der Masse), weshalb Bauteiltoleranzen unterschiedliches Verhalten der beiden „Kanäle" bewirken können.

Die zweite Variante von **Bild 18** hingegen geht schon mehr in Richtung eines echten differentiellen Filters, denn hier wurden je zwei der bei Bild 17 gegen Masse geschalteten Kondensatoren zu jeweils einem zusammengefasst. Sie beeinflussen nun beide Pfade gleichermaßen. Unterschiede können hier nur noch bei den Spulen auftreten.
Für differentielle bzw. Gegentakt-Signale ergibt sich mit beiden Schaltungen die gleiche Filterwirkung. Für Gleichtakt-Signale ergibt sich allerdings ein unterschiedliches Filterverhalten. **Bild 19** zeigt, dass beim kombinierten Filter von Bild 18 lediglich noch die Spulen den Frequenzgang beeinflussen und daher für Gleichtaktsignale nur eine schwache Filterwirkung resultiert. Obwohl Gleichtaktsignale durch DACs vom folgenden Differenzverstärker verringert werden, ist dies suboptimal, denn die Gleichtaktunterdrückung von HF-Verstärkern wird mit steigender Frequenz schlechter. Jede Lösung hat immer ihre Vor- und Nachteile. Um möglichst identische Filterzweige zu realisieren kann man z.B. fertige Filterbausteine aus demselben Fertigungslos verbauen, da so die Varianz der Filter kleiner ausfällt.

Doppelt-Invers-Tschebyscheff-Filter 4. Ordnung

Dieses Filter ist mir beim Reverse Engineering eines gekauften Filters aufgefallen. Es ist aus zwei zusammengesetzten Filtern 4. Ordnung gebaut (siehe **Bild 20**) und hat die schöne Eigenschaft, dass man es gut mit Bauteilen der klassischen E-Reihen realisieren kann und weniger krumme Werte benötigt, was den Bau solcher Filter erleichtert. Möchte man die Grenzfrequenz des Filters ändern so verschiebt man die Werte innerhalb der E-Reihen. Dieser Vorzug wird allerdings durch eine kleine Delle im Durchlassbereich erkauft. **Bild 21** zeigt den Frequenzgang dieses Filters, das im Sperrbereich eine beeindruckende Mindestdämpfung von 70 dB erzielt. Beide Notches haben dieselbe Frequenz und verstärken den Effekt. In **Bild 22** ist der vergrößerte Frequenzgang im Durchlassbereich samt der typischen kleinen Delle bei 75 MHz zu sehen.

Bauteilauswahl

Wie schon am Anfang erwähnt, gibt es bei der Dimensionierung passiver Filter keine Freiheitsgrade. Für die Berechnung dieser doch recht anspruchsvollen Filter ist spezialisierte Software unerlässlich. Unter [2, 3 und 4] findet sich eine Auswahl geeigneter Filterprogramme. Die so errechneten Werte der Spulen und Kondensatoren werden in den meisten Fällen nicht verfügbar sein. Bei Kondensatoren ist die Sache noch recht einfach zu lösen: Man schaltet schlicht zwei oder drei Kondensatoren parallel, um dem geforderten, krummen Wert möglichst nahe zu kommen, denn Kondensatoren sind klein und auch nicht ganz so teuer. Trotzdem sollte man dabei die Toleranzen im Auge behalten.

In einer Simulation [5] sollte man überprüfen ob der Frequenzgang des Filters mit den real gefundenen Werten nahe genug am Ideal ist. Beim Bau eines Prototypen sollte man den Aufwand nicht scheuen und die Kondensatoren vor der Bestückung messen. Passive Filter verhalten sich da nicht wirklich anders als aktive Filter. Für beide gilt: Jedes Filter ist nur so gut wie seine Kondensatoren. SMD-Kondensatoren des Typs X7R etc. sind für Filterzwecke daher strikt zu vermeiden. Gute Kondensatoren sind (leider) meist groß und nicht ganz preiswert. Das Platinen-Layout sollte man erst angehen, wenn die Schaltung steht und man genau weiß, welche Bauteile genau verwendet werden.

Das Thema Spulen ist leider noch ein paar Grade schwieriger. Sie parallel zu schalten sollte man gar nicht erst versuchen. Auch eine Reihenschaltung funktioniert nur, solange die Spulen nicht magnetisch gekoppelt sind. Zur Vermeidung magnetischer Koppelungen versetzt man sie möglichst um 90° und ordnet sie in einer Art Zick-Zack-Muster auf der Platine an. Bei dem fertigen Filter von **Bild 23** kann man schön den Winkelversatz der Spulen bewundern, auch wenn die dritte Spule mit ihrem 45°-Winkel etwas aus der Reihe tanzt. Beim Fertigfilter von **Bild 24** sind weder die kleinen blauen Spulen in Serie noch die Kondensatoren parallelgeschaltet. Man kann also berechtigt davon ausgehen, dass der Hersteller sich diese Bauteile passgenau herstellen ließ.

Ein kleiner Ausweg aus der Spulenproblematik sind abstimmbare Exemplare. Hier kann man durch Verdrehen eines Ferritdeckels oder Kerns die Induktivität in gewissen Grenzen ändern. Bei den in Bild 23 verwendeten Neosid-Spulen ist das zum Beispiel möglich. Noch etwas gilt es bei Spulen zu beachten: Spulen haben einen Wicklungswiderstand, der häufig nicht zu vernachlässigen ist. Man sollte ihn messen oder aus dem Datenblatt entnehmen und ihn mit in die Simulation aufnehmen bzw. bei der Dimensionierung der Filter berücksichtigen. Das nichtideale Verhalten von Spulen dank des Widerstandes ihrer Wicklung führt im Durchlassbereich unter Umständen zu einem abfallenden Frequenzgang, den man eventuell in der nachfolgenden Schaltung korrigieren oder zumindest berücksichtigen sollte.

Generell gilt: Spulen mit größeren Induktivitäten verwenden Ferritkerne quasi als „Verstärkung" der Induktivität der reinen Wicklung. Je nach Größe des Kerns, der Wicklungszahl und der auftretenden Ströme kann der Kern in die magnetische Sättigung gehen und so massive Verzerrungen verursachen. Misst man am Ausgang eines Filters mehr Oberwellen als am Eingang, muss man entweder den Pegel verringern oder Spulen mit größeren Ferritkernen einsetzen.

Layout

Neben dem bereits erwähnten Zick-Zack-Muster für die Spulenanordnung sollte man auf kurze, direkte Verbindung zur Masse

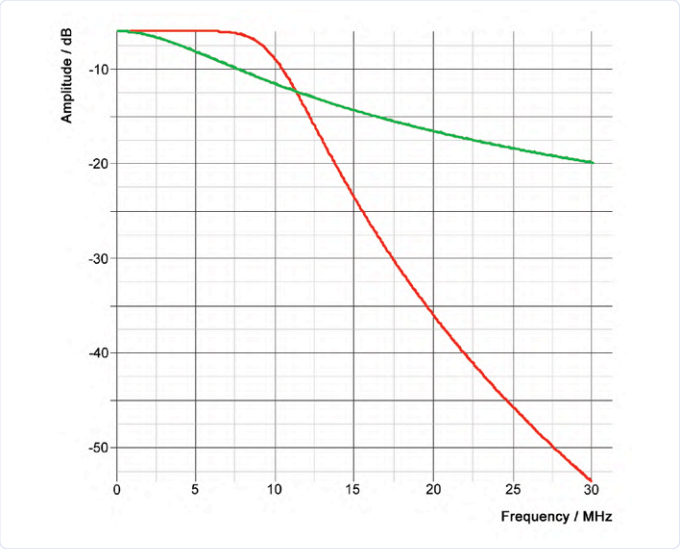


Bild 19. Gleichtakt-Frequenzgänge des doppelten SE-Filters von Bild 17 (rot) und des kombinierten Filters von Bild 18 (grün).

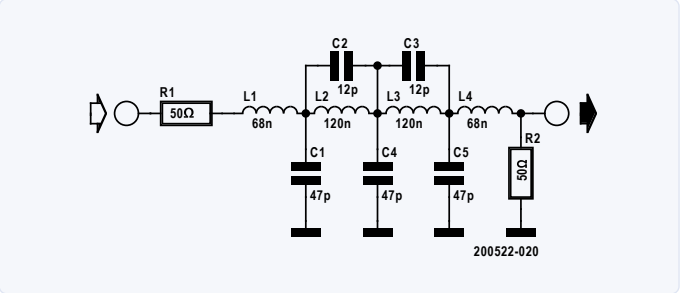


Bild 20. Doppelt-Invers-Tschebyscheff-Filter 4. Ordnung.

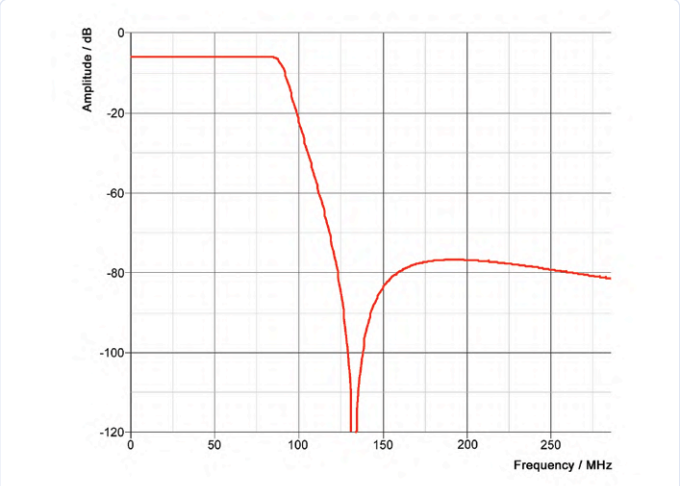


Bild 21. Frequenzgang eines Doppelt-Invers-Tschebyscheff-Filters 4. Ordnung.

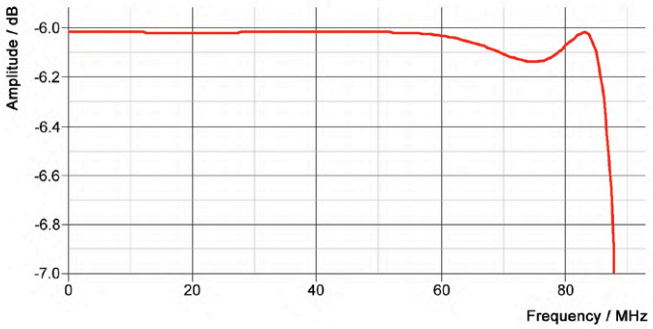


Bild 22. Vergrößerter Frequenzgang des Filters von Bild 20 im Durchlassbereich. Die kleine Delle bei 75 MHz ist typisch.

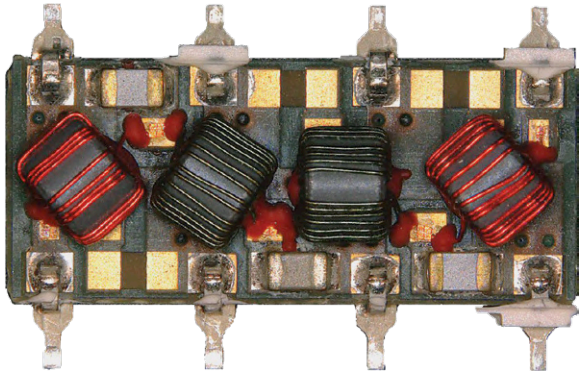


Bild 23. Innenleben eines Fertigfilters. Man achte auf den Winkelversatz der Spulen.

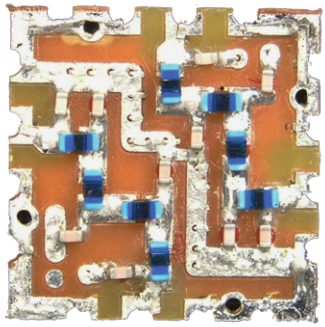


Bild 24. Ein fertiger Filter in SMD-Technik mit Spulen im Winkel von 90°. Auffällig: Weder gibt es bei den blauen Spulen eine Serien- noch bei den Kondensatoren eine Parallelschaltung.

achten. Doppelseitige Platinen mit einem Layer für die Signale und einem gegenüberliegenden Layer nur für die Masse erleichtern dieses Vorhaben. Gerade bei hohen Frequenzen und demgemäß kleinen Kapazitäten der Kondensatoren sollte man nicht außer Acht lassen, dass die Pads gegenüber Masse selbst schon als winzige Kondensatoren fungieren. Falls relevant muss man diese „parasitäre“ Kapazität bei der Wahl des Kondensatorwerts berücksichtigen.

200522-01

Sie haben Fragen oder Kommentare?
Gerne können Sie sich an den Autor wenden unter der E-Mail-Adresse: alfred_rosenkraenzer@gmx.de.

Ein Beitrag von
Idee, Bilder und Text: **Alfred Rosenkränzer**
Redaktion: **Dr. Thomas Scherer**
Layout: **Giel Dols**

PASSENDE PRODUKTE

- **OWON SDS1102 2-Kanal-Digital-Oszilloskop (100 MHz)**
  www.elektor.de/owon-sds1102-2-ch-digital-oscilloscope-100-mhz
- **Siglent SDG2042X Arbitrary Waveform Generator (40 MHz)**
  www.elektor.com/siglent-sdg2042x-arbitrary-waveform-generator-40-mhz
- **OWON XSA1015-TG Spectrum Analyser (9 kHz – 1.5 GHz)**
  www.elektor.de/owon-xsa1015-tg-spectrum-analyser-9-khz-1-5-ghz

WEBLINKS

[1] **Design analoger Filter (Teil 1)**: www.elektormagazine.de/magazine/elektor-154/58927
[2] **LC Filter Design Tool**: https://rf-tools.com/lc-filter/
[3] **AADE**: https://getwinpcsoft.com/Filter-Design-179557/download/
[4] **Quickfil 5.1 (DOS-Programm)**: www.omicron-lab.com/products/vector-network-analysis/quickfil/
[5] **Simetrix**: www.simetrix.co.uk

Funk-Messmodul
JOY-iT VAX-1030

Von **Harry Baggen**

Nicht gerade selten will man eine Schaltung oder ein vorhandenes Gerät mit einer Anzeige von Spannung, Strom und vielleicht noch mehr ausstatten. Geeignete Messmodule gibt es in allen möglichen Formen und Größen und das zu Preisen, für die man so etwas nicht mehr selbst bauen muss. Das JOY-iT VAX-1030 ist ein solches Modul. Es bietet für etwa 40 € eine Menge Extras und verfügt sogar über eine Funk-Anbindung.

Was können Sie für 40 € erwarten? Eigentlich hatte ich mir nicht allzu viel erhofft, als das Paket vor mir stand. Aber ich war doch überrascht, als ich den Inhalt inspiziert habe. Enthalten war ein Messmodul mit vielen Anschlüssen, ein separates Anzeigemodul mit Touch-Tasten, ein USB-Kabel, ein Kabel mit Temperatursensor und schließlich ein separates Kabel für die Stromversorgung der Anzeige (**Bild 1**).

Bild 1. Diese Hardware steckte im Paket.

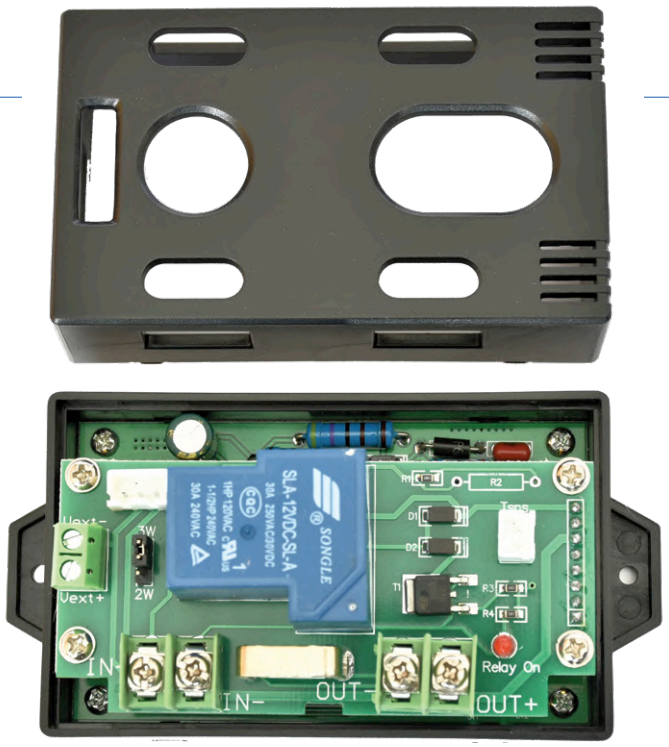


Bild 2. Das Messmodul mit abgenommenem Oberteil.

Viel Strom

Die hier besprochene Version ist für Gleichspannungen bis 100 V und Ströme bis 30 A geeignet. Es gibt auch eine Version für Ströme bis 100 A, aber solange man keine Autobatterien oder ähnliches messen will, dürften 30 A genug sein. Das Messmodul besteht aus einer Kunststoff-Box, deren Oberteil viele Aussparungen enthält. So kann man einfach Kabel und Stromversorgung an die Schraubklemmen in der Box anschließen.

Im Modul befinden sich zwei übereinander montierte Platinen (**Bild 2**). Die obere birgt alle Schraubklemmen und ein großes Relais, und die untere enthält die eigentliche Elektronik und ein 2,4-GHz-Transceivermodul für die drahtlose Kommunikation mit dem Anzeigemodul (sieht einem ESP8266-WLAN-Modul sehr ähnlich, aber der Chip ist unter einer Versiegelung verborgen). Die obere Gehäusehälfte lässt sich leicht abknicken, so dass man die Schraubklemmen richtig erreichen kann.

Dieses Modul ist zum Einbau an einer Stelle gedacht, an der man die Spannung und den Strom messen kann und will. Das Messmodul benötigt eine eigene Versorgungsspannung von 12 V. Der Temperatursensor kann bei Bedarf an einen Steckverbinder auf der Platine angeschlossen werden.

Display

Das Anzeigemodul kann auf zwei Arten an das Messmodul angeschlossen werden: Über das mitgelieferte USB-Kabel direkt an das Messmodul oder via eingebauter Funkverbindung. Letztere wird automatisch eingerichtet, wenn die USB-Verbindung getrennt wird. Es ist möglich, mehrere Module gleichzeitig zu verwenden. Laut Hersteller können 26 Funk-Kanäle gleichzeitig genutzt werden. Im Falle einer Funkverbindung benötigt das Anzeigemodul jedoch eine eigene Stromversorgung. 5 V kann man über die Mikro-USB-Buchse oder 8...16 V über einen zweipoligen JST-Stecker zuführen, für den ein geeignetes Kabel mitgeliefert wird.

Das ca. 3 x 2,5 cm große Display ist sehr hell und kann auch in heller Umgebung gut abgelesen werden. Neben dem Display befinden sich drei Touch-Steuertasten. Standardmäßig stellt das Display die gemessene Spannung, den Strom und die Zeit ab dem Zeitpunkt dar, an dem die Messung gestartet wurde. Auf der linken Seite kann ein Batteriesymbol angezeigt werden, das die verbleibende Batteriekapazität nach Eingabe der nominalen Batteriekapazität anzeigt. Oben sitzen die Anschlüsse der Stromversorgung und des Temperatursensors sowie das Relais.

Ein Druck auf eine Taste bringt weitere Informationen. Statt Spannung und Strom kann man die zugeführte oder aufgenommene Leistung anzeigen lassen. In einem kleineren Bereich darunter sieht man die von der Versorgung oder einem Akku gelieferten Amperestunden, die Wattstunden und die Verbrauchszeit. Auf der rechten Seite befindet sich eine Liste mit einer Reihe von Menüpunkten, durch die man mit den Auf- und Ab-Pfeilen navigieren kann. Auf einige dieser Punkte wird später noch eingegangen.

Anwendungsmöglichkeiten

Diese Kombination aus Anzeige- und Messmodul eignet sich für verschiedene Zwecke. Man kann damit den Strom messen, den ein Netzteil an eine Last liefert. Bei einer Geräteversorgung durch

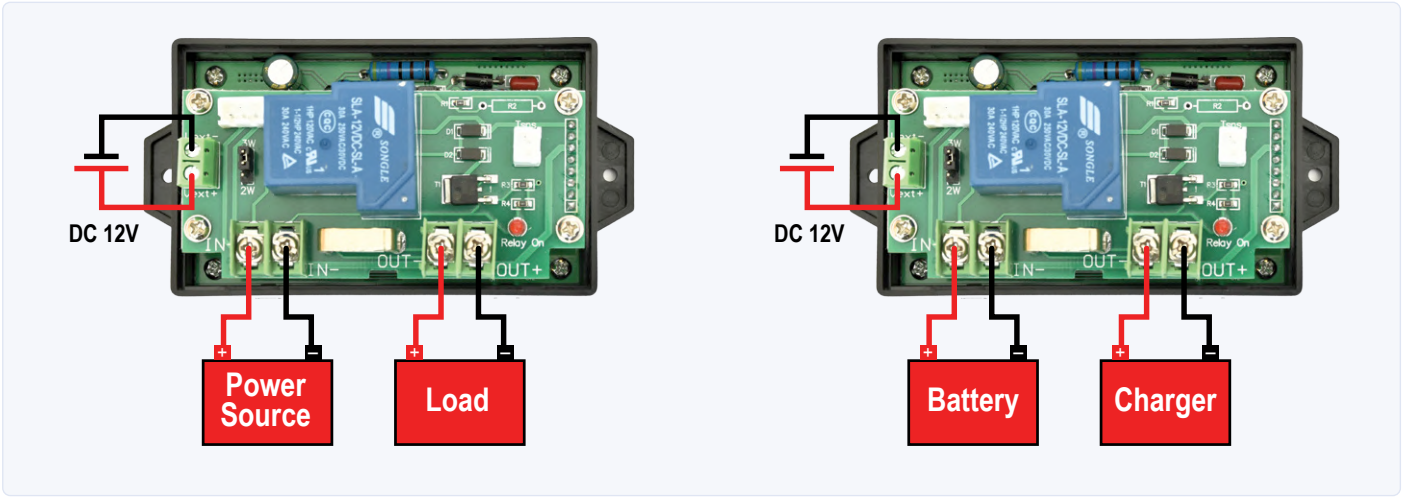


Bild 3. Man kann auch die Werte eines Akkus am Ladegerät in beiden Richtungen erfassen.

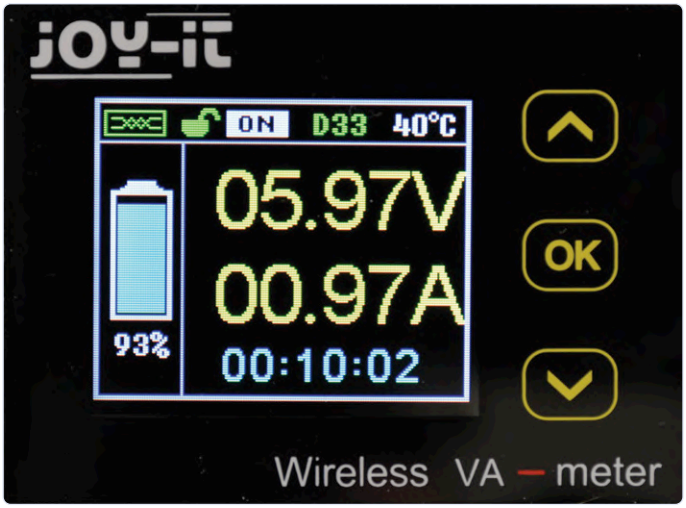


Bild 4. Die Standardanzeige.

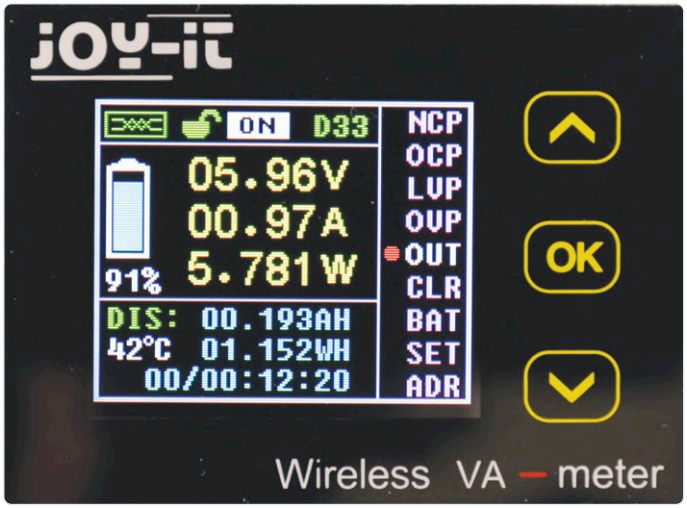


Bild 5. Nach Berühren einer Schaltfläche erscheint eine umfangreichere Anzeige samt Menü.

eine Batterie oder einen Akku werden auch die gelieferten Ah gemessen. Aber es geht auch andersherum, wenn nämlich die Last durch ein Ladegerät ersetzt wird, kann man messen, mit wie viel Energie der Akku geladen wird (**Bild 3**).
Speziell für die Verwendung mit Akkus gibt es einige interessante Menüeinstellungen. Man kann einen unteren und oberen Schwellenwert festlegen, bei deren Überschreitung das Relais abschaltet. Dadurch wird verhindert, dass der Akku überladen oder tiefentladen wird. Man kann auch Schwellwerte für die maximalen Ströme getrennt für Stromquelle -> Last und umgekehrt für Ladegerät -> Akku einstellen, oberhalb derer das Relais abschaltet. Man kann also auch gut verfolgen, wie viel Strom eine angeschlossene Schaltung verbraucht oder wie viel ein Akku während des Ladevorgangs aufnimmt, ohne einen Taschenrechner bemühen zu müssen. Übrigens kann das Batteriesymbol auch abgeschaltet werden, wenn es in einer bestimmten Anwendung nicht erforderlich ist.

Praxis

Obwohl das zugehörige Handbuch genügend Informationen enthält um den Einstieg zu erleichtern, empfehle ich Ihnen zum Kennenlernen aller Möglichkeiten, die verschiedenen Menüpunkte selbst auszuprobieren. Sie werden dann schnell herausfinden, welche Funktion oder Bedienung was macht.
Die gemessenen Spannungs- und Stromwerte werden auf dem Display mit zwei Ziffern vor und zwei Ziffern nach dem Dezimalpunkt angezeigt (**Bild 4** und **5**). Laut Handbuch beträgt die Genauigkeit bei der Spannung ±2% und beim Strom ±5%. Dies erweist sich im Vergleich zu einem Multimeter als akzeptabel, aber es ist schade, dass das Anzeigemodul auch ohne Eingangsspannung noch einen niedrigen Wert anzeigt. Er ist kleiner als 2% des Maximalwerts, aber es sieht doch merkwürdig aus.
Auf der Platine des Messmoduls befindet sich eine Steckbrücke, die in die Positionen 2*W* und 3*W* gesteckt werden kann. Dieser Jumper wird im Handbuch nicht erwähnt, aber nach einigem Suchen stellt sich heraus, dass er für die Auswahl der separaten 12-V-Versorgungsspannung verwendet wird. In Position 3*W* sind extra 12 V erforderlich. In der Position 2*W* kann sich das Modul diese Spannung von der angeschlossenen Spannungsquelle borgen, sofern diese zwischen 10 und 30 V liefert. Bei mir hat das Modul schon ab 8 V funktioniert. Die Funkverbindung ist sehr praktisch, um etwas aus der Ferne im Auge zu behalten. Ich versorgte das Anzeigemodul über ein kleines Netzteil und konnte damit im Haus herumlaufen. Die Reichweite (max. 10 m) nimmt aber schnell ab, wenn sich zwischen Mess- und Anzeigemodul einige Wände befinden. Aber das lässt sich anhand der Feldstärkeanzeige auf dem Display leicht verfolgen.

Fazit

Für die Messung von Strom und Spannung in allen Arten von Schaltungen ist diese Kombination aus Mess- und Anzeigemodul recht komfortabel und preisgünstig. Dank des großen Messbereichs von 30 A eignet es sich gut für Situationen, in denen größere Ströme fließen. Die Möglichkeit, das Anzeigeteil per Funk anzubinden und die Daten an einem anderen Ort auszulesen, ist ein tolles Extra, das man bei diesem Preis nicht wirklich erwarten würde. Diese Kombination ist mit Sicherheit ihren Preis von etwa 40 € wert!

200571-01

PASSENDE PRODUKTE

› **JOY-iT VAX-1030 Drahtloses Multimeter**
www.elektor.de/joy-it-vax-1030-wireless-multifunction-meter

Projekt 2.0

Korrekturen, Updates und Leserbriefe

Redaktion: **Ralf Schmiedel** und **Jens Nickel**

USB-S/PDIF-Schnittstelle

Heft 09-10/2020, S. 6 (180027)

Das Schaltbild in Bild 5 und die Stückliste zeigen beide die falsche Typennummer für IC1. Es muss sich um einen PIC32MX27**0**F256B-50I/SO und nicht um einen PIC32MX27**4**F256B-I/SO handeln. Die Leiterplatte ist korrekt.

Der ewige Blinker

Heft 07-08/2020, S. 65 (200200)

Es gibt einen Fehler im Schaltplan: T1...T3 sind als BC548C (NPN) aufgeführt, aber T3 ist als PNP gezeichnet. Letzteres ist richtig! T3 sollte ein PNP-Transistor sein, z.B. ein BC558.

BalBot: Ein selbstbalancierender Roboter

Heft 05-06/2020, S. 59 (200074)

Zum selbst balancierenden BalBot hat uns der Autor Dr. Günter Spanner auf ein Video hingewiesen, das diesen in Aktion zeigt: *www.youtube.com/watch?v=UlKyHn1llxw*

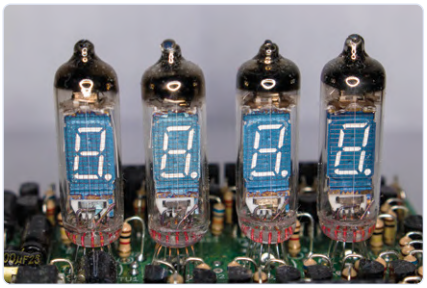
Möchten Sie selbst etwas beitragen?
Gerne können Sie sich an die Elektor-Redaktion wenden unter der E-Mail-Adresse: redaktion@elektor.de.

Wie macht man (schöne) Fotos von Platinen...

Heft 07-08/2020, S. 44 (200186)

Mit großem Interesse habe ich den Artikel „Wie macht man (schöne) Fotos ...“ gelesen. Den Aufwand kann man allerdings etwas minimieren und die Linse in der Lupenlampe kann bleiben, ohne dass die Qualität leidet. Es gibt sehr preisgünstige Ringblitze für alle gängigen SLR-Kameras. Trotz des günstigen Preises bieten sie eine gute Qualität. Für meine Fotos benutze ich den Ringblitz von Fositan TYP RF-550D. Dieser Blitz ist über Amazon für ca. 34 Euro erhältlich. Mit dem Ringblitz wird eine große Anzahl Adapter für viele unterschiedliche Objektivdurchmesser mitgeliefert. Wenn man kein Makroobjektiv sein Eigen nennt, sollte man sich noch eine Nahlinse beschaffen. Mit +2 Dioptrien ist man auf der sicheren Seite. Bei der Arbeit mit dem Ringblitz muss die Kamera im manuellen Modus arbeiten, Blende und Zeit müssen von Hand eingestellt werden. Bei der Standardhelligkeit des Blitzes (Einstellung 0) und ISO 100 habe ich mit Blende 11 und 1/60 gute Erfahrungen gemacht. Sollten die Fotos zu dunkel werden, kann man den ISO-Wert auch auf 200 erhöhen. Die Blende sollte nicht zu weit geöffnet werden, da der Bereich der Schärfe sonst zu klein wird. So kann man auch von Fall zu Fall ohne Stativ arbeiten. Der Blitz arbeitet sehr gut mit meiner Kamera (Canon 80D mit Zoomobjektiv 18-55 mm) zusammen. Besondere Einstellungen an der Kamera müssen nicht gemacht werden. Der passende Adapterring wird auf das Objektiv aufgeschraubt, das Steuergerät auf die Blitzschiene gesteckt. Der Blitz kann dann am Adapterring befestigt werden.

Volker Schmidt



190381-E-01

Von der Pike auf gelernt

Neues aus der Elektor-Ideenkiste

Zusamengestelt von **Eric Bogers**

Nicht in der Stimmung für etwas Großes? Das ist prima, denn hier stellen wir ein paar kleine Projekte vor, die ohne große Probleme gebaut werden können. Ideal, um dem Januar-Blues entgegenzuwirken…

Idee: Peter Neufeld (Deutschland)

Tranquili-T

Die „großen" Ideen und Projekte eines Hobby-Elektronikers werden von Familie und Freunden oft nicht wahrgenommen - ein selbstgebauter Oszillator oder eine Servosteuerung ist einfach kein Publikumsrenner. Herr Neufelds Tranquili-T hingegen hat viel Zustimmung erfahren. Er hat bereits viele Exemplare gebaut und an Freunde und Bekannte verschenkt.

Die Schaltung (**Bild 1**) ist um einen billigen MP3-Player herum aufgebaut, der mit beruhigender Musik oder Brandungs- oder Vogelgeräuschen aufgeladen wird. Die beruhigende Wirkung wird durch eine Regenbogen-LED verstärkt, die in einem dunklen Raum für gedämpfte Beleuchtung sorgt. Ein Video des Autors zeigt ein Tranquili-T in Aktion [1].

Über die Stromversorgung können wir uns kurz fassen: Alles, was 4,5…5 VDC liefert, ist brauchbar. Drei 1,5-V-Batterien sind eine Möglichkeit, aber vergessen Sie nicht, dass die LED etwa 20 mA und der Player bei durchschnittlicher Lautstärke um die 50 mA zieht. Eine USB-Stromversorgung oder eine Power-Bank kann gute Dienste leisten; dann müssen Sie sich keine Sorgen über leere Batterien machen.

Aus Gründen der Einfachheit sind MP3-Player-Module mit integriertem Klasse-D-Verstärker vorzuziehen. Der Autor hat gute Erfahrungen mit dem JQ6500 und dem DFplayer Mini gemacht, die man für wenig Geld bei den üblichen Anbietern aus dem Fernen Osten erhält. Im Schaltplan sind zwei Möglichkeiten der Lautstärkeregelung eingezeichnet. Die Variante mit S2 und S3 verwendet zwei digitale Eingänge, die mit S4 und S5 nutzt einen analogen Eingang in Kombination mit zwei verschiedenen Widerständen. Die eingezeichneten Werte passen für den JQ6500, bei anderen Playern sind wahrscheinlich einige Experimente erforderlich - unterstützt durch Informationen aus dem Datenblatt und dem Internet.

Was den Lautsprecher betrifft: Verwenden Sie ein möglichst kleines und flaches Exemplar mit einer Belastbarkeit von mindestens 0,5 W und einer Impedanz von 8…32 Ω. Bei einem 4-Ω-Exemplar stellte sich heraus, dass die Lautstärke insgesamt zu hoch war.

Bild 2 zeigt die „Eingeweide" eines DFPlayer Mini und **Bild 3** ein Tranquili-T in seiner vollen Pracht.

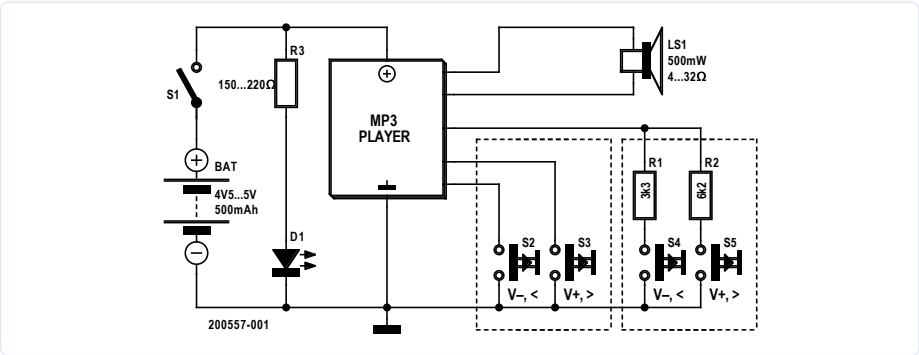


Bild 1. Das Herzstück der Schaltung ist ein MP3-Player-Modul.

Bild 2. Die Elektronik kann kompakt auf einem Stück Lochrasterplatine aufgebaut werden.

Bild 3. Das Tranquili-T in seiner ganzen Pracht.

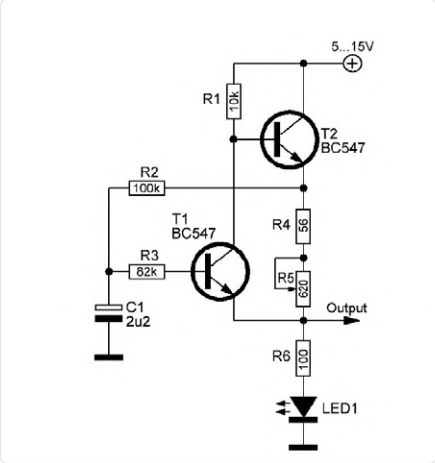


Bild 4. Grundschaltung eines Impulsoszillators auf der Basis eines modifizierten Schmitt-Triggers.

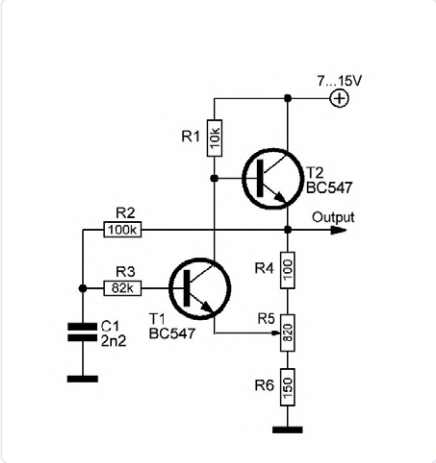


Bild 5. Eine Variante des Impulsoszillators von Bild 4.

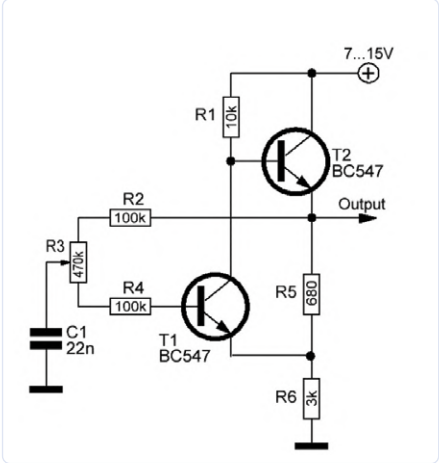


Bild 6. Und eine weitere Variante, bei der die Pulsdauer eingestellt werden kann.

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

Three Shades of … Schmitt-Trigger-Oszillator

Mit Schmitt- oder asymmetrischen, emittergekoppelten Triggern können einfache Impuls-Oszillatoren mit weitem Frequenzbereich aufgebaut werden. Um einen Schmitt-Trigger in einen Oszillator zu verwandeln, sollte der Ausgang des Triggers über einen Widerstand mit dem Eingang verbunden werden und ein Kondensator zwischen den Eingang des Triggers und der „gemeinsamen" Versorgungsschiene (hier: Masse) geschaltet werden.

Die drei Bilder zeigen praktische Schaltungen von Impulsoszillatoren, basierend auf dem modifizierten Schmitt-Trigger. Es handelt sich um eine Brückenschaltung, die aus einem Widerstandsteiler, dem Emitter des Eingangstransistors, der am Mittelpunkt des Widerstandsteilers angeschlossen ist, und einem zeitbestimmenden RC-Glied besteht, dessen zusätzliche Elemente dazu dienen, den Trigger in einen repetitiven Impulsoszillator zu verwandeln.

Der p-n-Übergang des Transistors liegt zwischen dem diagonal gegenüberliegenden Paar von Verbindungen der Brücke. Wenn der Kondensator C1 entladen ist, sperrt der Transistor T1 und der Transistor T2 leitet. Wird der Kondensator auf ein paar Volt in Bezug auf den Mittelpunkt des Widerstandsteilers beziehungsweise den Emitter von T1 geladen, so leitet der Eingangstransistor T1 und T2 sperrt. Folglich wird der Widerstandsteiler abgeschaltet und der zeitbestimmende Kondensator C1 wird wieder entladen. Der Transistor T1 wird aufgrund der Entladung des Kondensators C1 wieder gesperrt, während Transistor T2 dann wieder leitet. Der Ausgangszustand ist erreicht und das Spiel beginnt von neuem.

Die Frequenz des Impulsoszillators in **Bild 4** kann proportional durch Anpassung der Kapazität von Kondensator C1 geändert werden. Mit dem Potentiometer R5 wird ein Frequenzbereich größer als 1:10 abgedeckt. LED1 dient zur visuellen Kontrolle der Frequenzabstimmung: Die Leuchtintensität ist am Anfang des Bereichs maximal und am Ende des Bereichs minimal. Der Oszillator arbeitet im Frequenzbereich von 3...30 Hz. Der Strom durch die LED1 variiert zwischen 2 mA und 20 mA bei einer Versorgungsspannung von 9 V für die in der Schaltung gezeigten Werte. Wenn die Versorgungsspannung von 5 V auf 15 V erhöht wird, ändert sich die Oszillationsfrequenz um nicht mehr als 10%.

Die Impulsoszillatoren in **Bild 5** und **Bild 6** arbeiten bei einer Versorgungsspannung von 9 V nominal im Frequenzbereich von 0,8...10 kHz respektive 0,35...2,8 kHz. Der Oszillator in Bild 2 wird durch Änderung des Verhältnisses im Spannungsteiler R4, R5, R6 (die rechte Hälfte der Brückenschaltung) gesteuert. Beim Oszillator in Bild 3 ist es die Widerstandskette R2, R3, R4, die die Entlade/Ladevorgänge in der linken Hälfte der Brückenschaltung steuert.

Idee: Elektor-Labor

Wanderndes Mono-Signal

Bevor die digitale Signalverarbeitung die Bühne eroberte, mussten Musiker großformatige elektromechanische Vorrichtungen einsetzen, um einen ganz bestimmten Klangeffekt zu erzeugen.

Nehmen Sie zum Beispiel die Hammondorgel, einen (zugegebenermaßen etwas verkürzt ausgedrückt) elektromechanischen Vorläufer des Synthesizers – ein Instrument, dem Organisten wie Walter Wanderley reizvolle Klänge entlocken konnten. Mit einem Nachteil allerdings: Eine solche Orgel war prinzipiell monofon, was den Live-Klang auf der Bühne für das Publikum „dünn" wirken ließ.

Auftritt Donald Leslie: Er entwickelte eine sehr clevere Lösung für dieses Problem, denn er kam auf die Idee, dass ein rotierender Lautsprecher dem Mono-Tonen Leben einhauchen kann. Es entsteht ein viel breiteres Klangbild. Nun ist ein rotierender Lautsprecher elektrisch nur schwer zuverlässig hinzubekommen. Aus diesem Grund besteht der Leslie-Lautsprecher in der Regel aus einem horizontal montierten, statischen Lautsprecher, der über einen rotierenden Zylinder mit einem darin angebrachten Schallloch abstrahlt. Das Ganze sieht etwa so aus wie in **Bild 7**.

Hier zeigen wir Ihnen in **Bild 8** ein (sehr einfaches) elektronisches Äquivalent, das zwar die Qualität eines originalen und sehr teuren Leslie-Lautsprechers nicht erreichen kann, aber dennoch interessant genug für eigene

Experimente ist.
Links ist ein diskret aufgebauter astabiler Multivibrator (AMV) zu sehen, dessen Frequenz mit P1 zwischen etwa 1 Hz und 8 Hz eingestellt werden kann. Das lineare Stereopotentiometer P1 ist wie in der Mitte gezeigt verdrahtet. Die beiden Glühbirnchen dienen in erster Linie dazu, den Widerstand der LDRs R3 und R4 zu variieren und fungieren in zweiter Linie als Kollektorwiderstände für beide Transistoren.
Nun zur rechten Seite der Schaltung: Ein Mono- und kein Stereosignal (!) trifft an C3 ein und wird durch P2 (das als eine Art Balance-Regler fungiert) in zwei Zweige aufgeteilt, die zu den beiden Eingängen eines Stereoverstärkers führen. Wegen der lichtempfindlichen Widerstände wird für jeden Kanal ein Teil des Monosignals nach Masse abgeleitet. Der resultierende Pegel eines Kanals hängt vom eingestellten Widerstandswerts von P2 und dem aktuellen Widerstandswert des jeweiligen LDR ab, der wiederum von der auf ihn fallenden Lichtmenge abhängt. Und diese Lichtmenge variiert dank des astabilen Multivibrators periodisch.

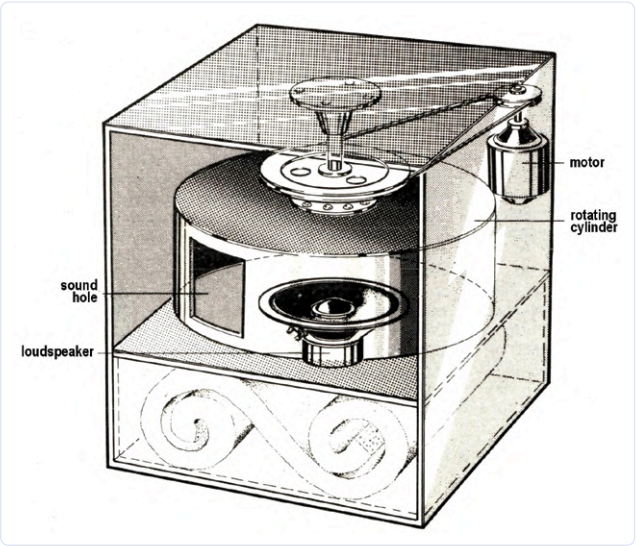


Bild 7. Die Leslie-Box, ein horizontal montierter Lautsprecher mit einem rotierenden Zylinder, in den ein Schallloch eingefügt wurde.

Nun leuchtet ein, warum hier Glühlämpchen und keine LEDs zum Einsatz kommen: Eine Glühlampe weist eine gewisse Trägheit auf, die den Wechsel des Widerstandes (und damit den Wechsel des Signals von links nach rechts und zurück) etwas weicher macht.
Die Konstruktion unserer Pseudo-Leslie ist nicht kritisch, nur die Glühbirnchen müssen in der Nähe der entsprechenden LDRs angebracht werden, so dass sie sich nicht gegenseitig beeinflussen können und das Umgebungslicht ausgeblendet ist - in lichtdichten Röhren oder etwas Ähnlichem.

200557-03

Sie haben Fragen oder Anmerkungen?
Wenn Sie Fragen oder Kommentare zu diesem Artikel haben, senden Sie bitte eine E-Mail an den Autor an die Elektor-Redaktion (redaktion@elektor.de).

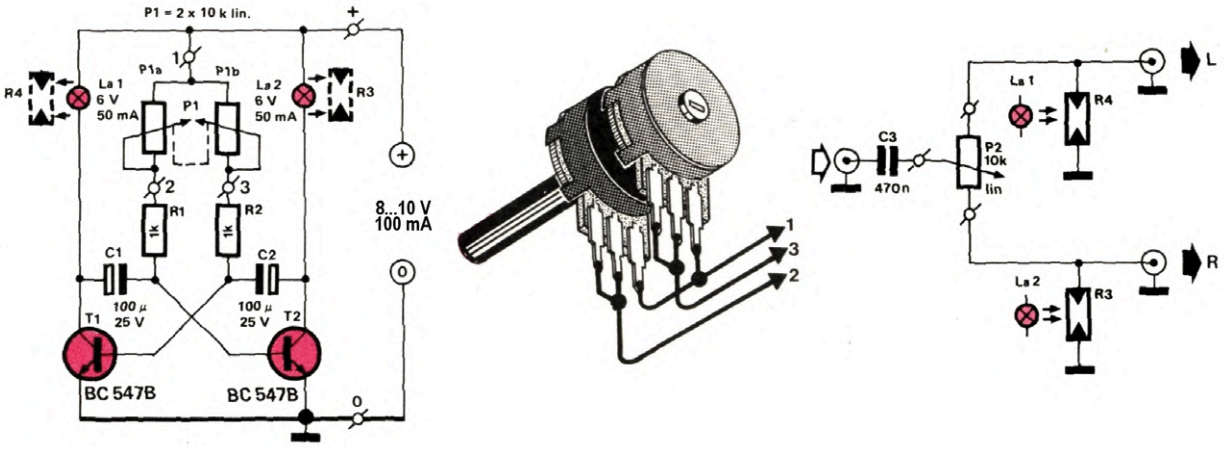


Bild 8. Die Schaltung (links) ist relativ einfach. In der Mitte sehen Sie, wie das Stereopotentiometer verdrahtet und rechts, wie es an einen Verstärker angeschlossen wird.

WEBLINKS

[1] **Tranquili-T in Aktion**: https://vimeo.com/272785420
[2] **Donald Leslie**: https://de.wikipedia.org/wiki/Donald_Leslie

PASSENDE PRODUKTE

- **Englisches Buch „Electronic Circuits For All"**
  www.elektor.de/electronic-circuits-for-all

Neues LCR-Messgerät 50 Hz bis 2 MHz

Teil 2: Betrieb, Kalibrierung und Firmware-Programmierung

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

Das Messprinzip des neuen LCR-Messgeräts und die Hardware wurde im ersten Teil des Artikels besprochen. In diesem zweiten und letzten Teil werden die Benutzerschnittstelle, die Kalibrierung und die Firmware-Programmierung des AU2019 behandelt.

Im ersten Teil erfuhren wir, wie der AU2019 funktioniert, wie er Impedanzen misst. Darüber hinaus lernten wir die eingesetzten Kalibrier- und Kompensationsprinzipien sowie die erforderliche Hardware kennen. Jetzt ist es an der Zeit, einen Blick auf dieses LCR-Messgerät aus der Sicht des Benutzers zu werfen.

Stand-alone-Betrieb

Der AU2019 kann über eine USB-Verbindung von einem Computer aus bedient werden, aber mit der LC-Display-Erweiterung, dem Drehgeber und den Tasten ist es auch als eigenständiges Messgerät unabhängig und ohne Computer zu betreiben. In diesem Modus startet es beim allerersten Einschalten automatisch, wenn eine externe

Tabelle 1.

							50	60				*Hz*
100	120	150	200	250	300	400	500	600	700	800	900	*Hz*
1,0	1,2	1,5	2,0	2,5	3,0	4,0	5,0	6,0	7,0	8,0	9,0	*kHz*
10	12	15	20	25	30	40	50	60	70	80	90	*kHz*
100	120	150	200	250	300	400	500	600	700	800	900	*kHz*
1,0	1,2	1,5	2,0									*MHz*

5-V-USB-Stromversorgung verwendet wird. Wenn das Gerät über einen PC mit Strom versorgt wird und Sie möchten trotzdem im Standalone-Modus arbeiten, dann drücken Sie beim Einschalten die obere Taste auf der Frontplatte. Der neue Modus wird so gespeichert und beim nächsten Start automatisch gewählt. Dieselbe Aktion ist für die Rückkehr in den PC-Modus erforderlich. Wenn der Messbildschirm angezeigt wird, lässt sich mit dem Drehencoder die Messfrequenz auf einen der 54 vordefinierten Werte ändern (**Tabelle 1**). Nach einem langen Tastendruck auf den Drehencoder können Sie eine benutzerdefinierte Frequenz eingeben. Die zu ändernde Ziffer wird mit den beiden oberen Tasten ausgewählt; der Drehencoder ändert ihren Wert. Nach der Bestätigung wird dieser neue Wert invers dargestellt und auch verwendet. Er kann nicht mehr mit dem Drehknopf geändert werden. Ein erneuter langer Tastendruck auf den Drehknopf ist erforderlich, um diesen Modus zu verlassen. Wenn der Wert gesichert ist, wird er bei der nächsten Abfrage angeboten.

Die aktuelle Belegung der Tasten wird auf dem Display angezeigt und ändert sich entsprechend der von Ihnen gewählten Einstellungen. Bei einigen Tasten hängt die Aktion davon ab, ob sie kurz oder länger gedrückt werden. Im letzteren Fall blinkt der Bildschirm kurz auf, um zu signalisieren, dass dieser Zustand erkannt wurde.

- Die erste (obere) Schaltfläche schaltet die Komponentendarstellung um: Von *AUTO* (Standardmodus) auf *SERIES* oder *PARALLEL*.
- Die zweite Taste *TRIM* leitet die TRIM-Prozedur für die aktuelle Frequenz ein (*OPEN* oder *SHORT* entsprechend der gemessenen Impedanz). Ein langer Druck löst die Prozedur zur Speicherung aller seit dem letzten Start durchgeführten TRIM-Aktionen aus.
- Mit der dritten Schaltfläche *AVG S* oder *AVG F* können Sie die Anzahl der für die Anzeige verwendeten Messungen (Mittelwert) und damit das Ansprechverhalten des Geräts ändern. Sie dient als Umschalter zwischen langsamer Anzeige (*Average SLOW* wird angezeigt) und schneller Anzeige (*Average FAST*). Diese Funktion wird allerdings erst ab Firmware-Version 2.0.0 implementiert, die es noch nicht gab, als die meisten der hier gezeigten Screenshots geschossen wurden.
- Mit der vierten Taste *R-HOLD* können Sie den aktuellen Bereich einfrieren. Ein langer Druck schaltet die Anzeige der Einstellungen des Messgeräts um: den Bereich (R1…R4), die PGA-Verstärkung bei Spannungs- (Ux) beziehungsweise Strommessung (Ix) mit x = 1, 3 oder 10.
- Die unterste Taste *MENU*: Kurzer Tastendruck für den Zugriff auf das Benutzermenü (Messparameter), ein längerer Tastendruck öffnet das Kalibrierungsmenü.

Bild 1. Das User-Menü.

User-Menü (**Bild 1**)

- *Language (1a):* Auswahl der in den Menüs verwendeten Sprache. Die aktuelle Version unterstützt Englisch, Französisch, Deutsch und Niederländisch.
- *Q limit for secondary display (1b):* Wählen Sie den Wert von Q (= 1/D), über dem die sekundäre Einstellung nicht angezeigt wird. Mögliche Werte sind: *100*, *200*, *500*, *1000*, *2000*, *5000* und *no limit*.
- *Sorting Parameters (1c):* In diesem Menü können Sie die Parameter (Toleranz und Wert) auswählen, die für die Sortierung der Bauteile verwendet werden. Der vorgeschlagene Wert ist derjenige des Primärparameters des angeschlossenen Referenzbauteils, bevor in den Menümodus gewechselt wird. Diese Parameter werden gespeichert. Im ersten Schritt (1c) können Sie die Standardtoleranz wählen, von der Reihe E6 (20 %) bis zur Reihe E96 (1 %), im zweiten Schritt den Sollwert in dieser Reihe (1d).
- *Sort (1e)*: In diesem Menü wird verglichen, ob der Ist-Wert des Bauteils innerhalb der erlaubten Toleranzgrenzen liegt und ob der Test ein *PASS* oder ein *FAIL* ergibt.

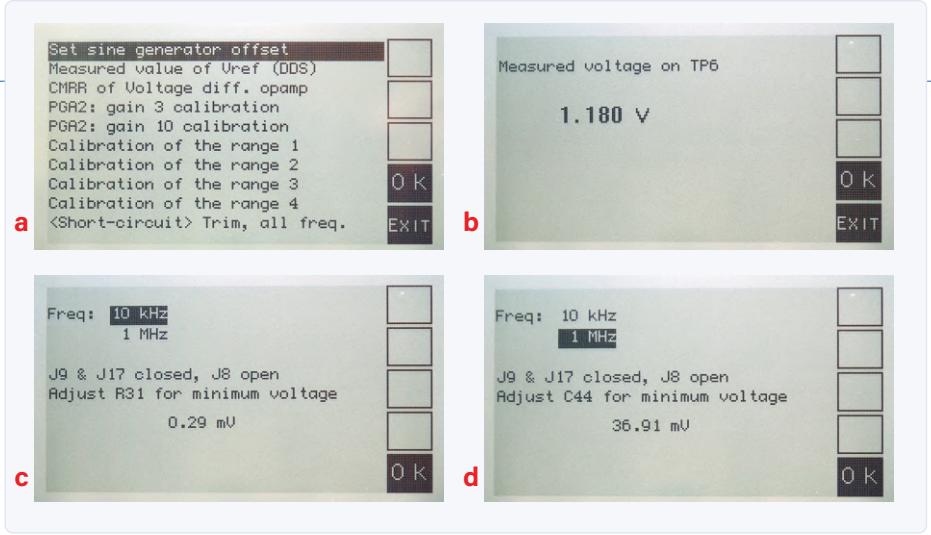


Bild 2. Das Kalibrierungsmenü.

- *AC level (1f):* Dieses Menü wählt die Amplitude des Testsignals ohne Last aus. Mögliche Werte sind: 100 mV, 200 mV, 500 mV, 1 V (RMS).
- *DC Bias (1g/1h)*: Dieses Menü stellt den Wert der Spannung (für Kondensatoren) oder des Stroms (für Induktivitäten) von DC-Bias ein. Mögliche Werte sind: 0...5,0 V in 1-V- oder 0,1-V-Schritten beziehungsweise 0...50 mA in 10-mA- oder 1-mA-Schritten. Durch Drücken des Drehencoders wird die Schrittweite geändert. Bei Induktivitäten wird der Istwert des Stroms erst nach der Bestätigung und einigen Sekunden Wartezeit zur Stabilisierung unter der Bezeichnung *<DC>* angezeigt, da er vom Widerstand des Prüflings abhängt. Die Beschriftung der zweiten Taste ändert sich, da bei DC-Bias die TRIM-Funktion nicht mehr möglich ist. Sie wechselt auf BIAS 0 und ein Tastendruck schaltet den Bias ab.

Kalibriermenü (Bild 2)

Alle Kalibrierungsschritte des Geräts sind über das Menü zugänglich. Wenn die Erstkalibrierungen durchgeführt werden, empfiehlt es sich, diese auch in der Reihenfolge auszuführen, wie sie im Menü erscheinen. Verwenden Sie Jumper mit vergoldeten Kontakten, um den parasitären Widerstand zu minimieren, und trennen Sie alle Kabel von den BNC-Buchsen. Das vollständige Kalibrierverfahren ist in der herunterladbaren Dokumentation [1] beschrieben. Es ist überhaupt nicht schwierig und erfordert nur ein Multimeter, Konfigurations-Jumper, ein kleines Trimmwerkzeug ... und ein wenig Geduld, denn bei einigen Schritten muss man mehrere Minuten warten. Wir geben Ihnen hier nur einen kurzen Überblick.

- *Set sine generator offset (2a)*
- *Measured value of V_{Ref} (DDS) (2b)*
- *CMRR of voltage diff.(erential) opamp (2c/d)*

Diese Einstellungen sind schnell erledigt, aber die folgenden Schritte nehmen einige Zeit in Anspruch. Am Ende jedes Schrittes muss die Speicherung der neuen Werte bestätigt werden. Beachten Sie, dass ältere Parameter gelöscht werden, auch wenn die neuen Werte NICHT gespeichert werden!

- *PGA2: gain 3 calibration* und *gain 10 calibration*
- *Calibration of the range 1* bis *Calibration of the range 4*
- *<Open-circuit> Trim, all freq.*
- *<Short-circuit> Trim, all freq.*

Am Ende der Kalibrierungen bitte alle Jumper entfernen und die Kelvin-Kabel oder die Testhalterung an die BNC-Stecker anschließen. Wiederholen Sie dann die Kalibrierungsprozedur für Leerlauf und Kurzschluss, um die neuen parasitären Komponenten zu berücksichtigen.
Vor einem Test, bei einer gegebenen Frequenz, ist es möglich und sogar empfehlenswert, die beiden Korrekturen TRIM durch Drücken der entsprechenden (zweiten) Taste erneut vorzunehmen. Neue Einstellungen (wie auch solche bei anderen Frequenzen) können durch einen langen Tastendruck auf diesen Knopf gesichert werden.

Fehlerprüfung

Während der Startsequenz der Firmware werden einige Prüfungen durchgeführt.

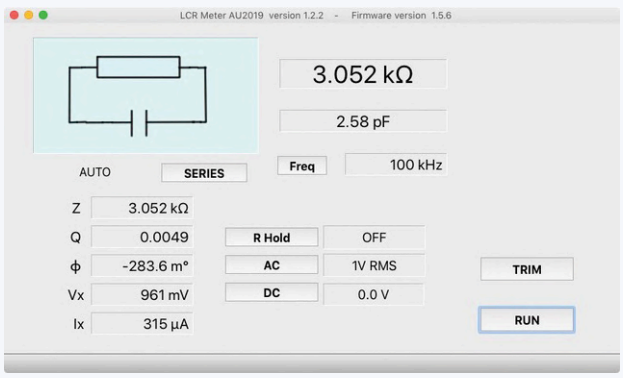


Bild 3. Steuerung des LCR-Messgeräts über die AU2019-App.

Tabelle 2.

Bit-Number	Ux	Bit ist 0 bei Messspannung
1	-5 V	-5,3 V < Ux < -4,7 V
2	+5 V	4,7 V < Ux < 5,3 V
3	+3 V	2,85 V < Ux < 3,15 V
4	+6.5 V	6,29 V < Ux < 6,68 V
5	+7.5 V	7,30 V < Ux < 7,60 V

Zuerst testet der Bootloader die Integrität der Firmware im Speicher. Dann prüft die Firmware auf gültige Sprachdaten im Speicher und verifiziert die Versorgungsspannungen (**Tabelle 2**). Wenn die Prüfung der Stromversorgung fehlschlägt, wird die Meldung *Power supply Test Error, code A* angezeigt, und das Programm schaltet eine Sequenz ein, bei der die LED D12 für 0,5 s ein- und für 1 s ausgeschaltet ist. Jedes Bit des binären Äquivalents von A zeigt den Zustand einer der verifizierten Versorgungsspannungen an: 0, wenn die Spannung korrekt ist, andernfalls 1.

PC-Modus

Starten Sie das Gerät im PC-Modus (*Waiting for the GUI...* wird angezeigt) und starten Sie einige Sekunden später das Programm AU2019 auf dem PC (**Bild 3**). Wählen Sie im Menü *Port* den richtigen seriellen Anschluss und klicken Sie dann auf die Schaltfläche *Open port*. Sobald die Kommunikation hergestellt ist, wird die Firmware-Version rechts neben der Anwendungsversion des LCR-Messgeräts AU2019 angezeigt. Nach kurzer Zeit werden alle Parameter des DUT angezeigt. Die Schaltflächen verhalten sich wie im Standalone-Modus.

Einige Menüs sind nur im PC-Modus vorhanden, andere dagegen sind identisch mit denen des Stand-Alone-Modus. All dies wird in der Dokumentation erläutert. Es gibt auch Anweisungen für Firmware- und Menütext-Updates. Eine vollständige Beschreibung ist in der Bedienungsanleitung [1] zu finden.

Wie man Komponenten misst

Die verwendeten Prüfadapter haben einen großen Einfluss bei Messungen am Ende des Messbereichs und/oder bei hohen Frequenzen. Besonderes Augenmerk wird deshalb auf die TRIM-Verfahren gelegt. Damit die Messung genau ist, wenn es sich bei dem Messobjekt um eine kleine Induktivität oder einen kleinen Widerstand handelt, ist es wichtig, dass die TRIM-Kurzschlusskompensation kurz vor der Messung durchgeführt wird. Dasselbe gilt für kleine Kondensatoren, nur ist in diesem Fall die TRIM-Leerlaufkompensation gefragt.

VERGLEICH MIT KOMMERZIELLEN MESSGERÄTEN

Natürlich ist es eine gute Idee, dieses LCR-Messgerät AU2019 mit kommerziell erhältlichen Geräten zu vergleichen. Leider hatten weder der Konstrukteur noch das Elektor-Labor Geräte zur Verfügung, um einen Vergleichstest durchzuführen, und noch schlimmer: Es ist sehr schwierig, überhaupt ein LCR-Messgerät mit Messfrequenzen von bis zu 2 MHz zu finden. Mit der freundlichen Hilfe und Unterstützung von Alfred Rosenkränzer wurde das AU2019 mit einem Hameg HM8118, einem HP/Agilent 4263B und einem Handheld-Keysight U1732A, wobei das Hameg-Gerät die höchste Messfrequenz von 200 kHz besaß, verglichen. Alfred Rosenkränzers allgemeine Schlussfolgerungen lauten:
„Das AU2019 hat sich zu einem recht guten Messgerät entwickelt. An einigen Punkten hat es Schwierigkeiten, zum Beispiel bei kleinen Spulen bei 100 Hz Messfrequenz. Zwar ist dies sicher nicht die ideale Frequenz, um so kleine Spulen zu messen, Hameg und HP lieferten dennoch recht sinnvolle Werte.
Es ist zwar schön, dass man die Kelvin-Kabel und das 4-Terminal-Testgerät TH26001A sehr günstig bei Ebay kaufen kann, aber es ist schwer zu sagen, wie gut die Qualität ist. Eine gute Verbindung zwischen Bauteil und LCR-Meter ist aber der Schlüssel zu einer stabilen und glaubwürdigen Messung. Die Prüfhalterung Keysight 16047E (derselbe Gerätetyp wie der TH26001A) sollte da vertrauenswürdiger sein.
Man kann sagen, dass der Gesamtfehler des AU2019 1 % beträgt, sogar bis zu 0,1 % für Werte in der Mitte des Messbereichs, wenn man die richtigen Messbedingungen in Bezug auf den Typ und den Wert des Bauteils auswählt."

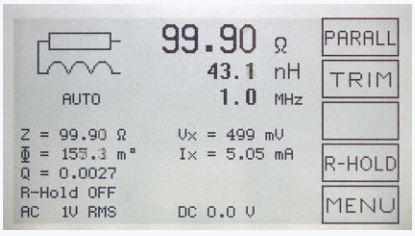
a 

Bild 4a. Messung eines niederohmigen Widerstandes bei hoher Frequenz.

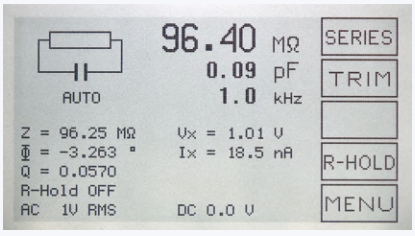
b 

Bild 4b. Messung eines hochohmigen Widerstandes bei niedriger Frequenz.

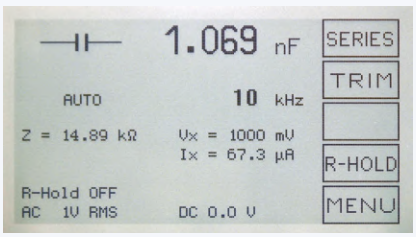
a 

Bild 5a. Messung eines kleinen Kondensators bei höherer Frequenz.

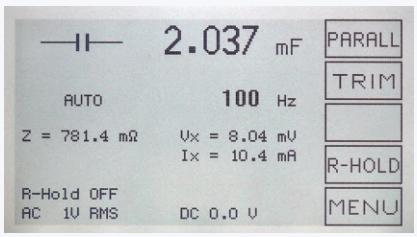
b 

Bild 5b. Messung eines großen Kondensators bei niedriger Frequenz.

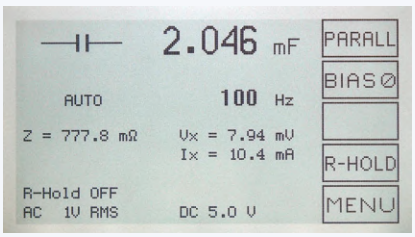
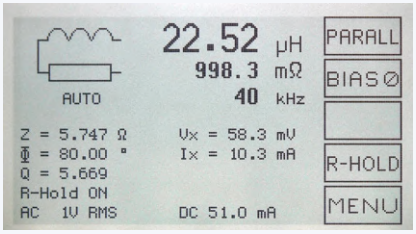
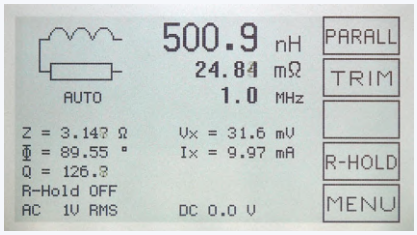
c 

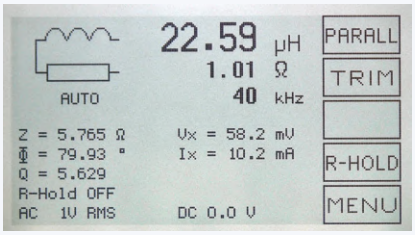
Bild 5c. Messung eines großen Kondensators bei niedriger Frequenz und 5 V_{DC} Bias.



a



b



c

Bild 6a. Messung einer Induktivität mit Bias-Strom.

Bild 6b. Messung einer Induktivität bei höherer Frequenz.

Bild 6c. Messung einer Induktivität ohne Bias-Strom.

ERSTER START DES GERÄTS MIT EINEM LEEREN MIKROCONTROLLER

Beim ersten Einschalten benötigen Sie einen PC mit Windows und den USB-Debug-Adapter von Silicon Labs (bei RS Componets Nr. 757-0297) zum Laden der Mikrocontroller-Firmware über die JTAG-Schnittstelle (J15). Sie müssen die Datei *MergedLCR6.hex* bereithalten, die sowohl den Bootloader (*boot_LCR6.hex*) als auch die Firmware *LCR6.hex* enthält. Wie das funktioniert, wird detailliert in der Bedienungsanleitung [1] beschrieben.

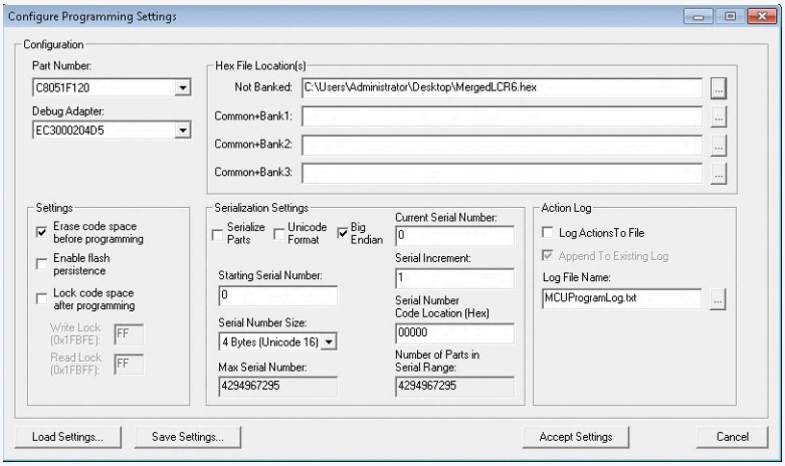


*Bild 7a. Programmiereinstellungen im **MCU Production Programmer** von Silicon Labs.*

ERSTMALIGES LADEN DER FIRMWARE

- Schließen Sie das USB-Debug-Adapterkabel an J15 auf der Platine und am PC an und setzen Sie SW1 auf ON. Starten Sie den *MCU Production Programmer* [3].
- Gehen Sie zu *Program Menu /Configure Programming Information...* und übernehmen Sie alle Einstellungen aus dem Screenshot, dann wählen Sie *Accept Settings*. Sie können diese Einstellungen mit *Save Settings...* speichern (**Bild 7a/7b**).
- Klicken Sie auf *Program Device*, um die Programmierung zu starten.
- Nach Abschluss der Programmierung erscheint (hoffentlich) die Meldung *Device Programmed and Verified* im *Status Log*-Fenster
- Dann muss die Sprachdatei für die Displaytexte gemäß der Betriebsanleitung (§4.2.5) hochgeladen werden.

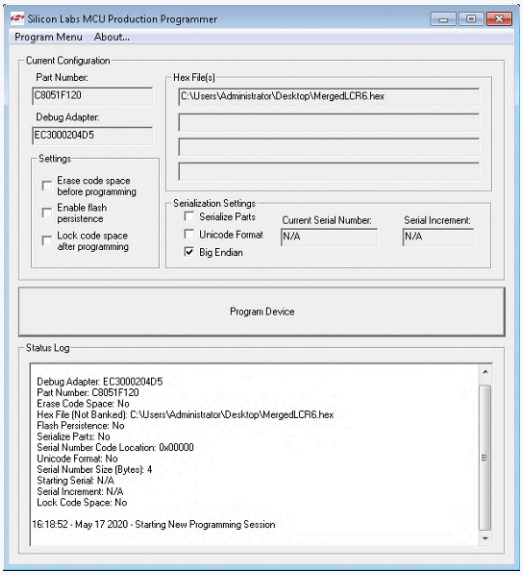


*Bild 7b. Programmier-Screen des **MCU Production Programmer** von Silicon Labs.*

WEBLINKS

[1] **Dokumentation and Software-Download**: www.elektormagazine.de/190311-B-02

[2] **Neue Projektseite zum LCR-Messgerät auf Elektor Labs**: www.elektormagazine.com/labs/remake-lcr-meter

[3] **Silicon Labs Production Programmer**: www.silabs.com/documents/login/software/MCUProductionProgrammer.zip

Resistor (Bild 4)
In den meisten Fällen können Widerstände leichter mit einem guten Multimeter gemessen werden, doch andererseits kann es interessant sein, die parasitäre Reiheninduktivität eines niederohmigen Widerstandes bei hohen Frequenzen oder die parasitäre Kapazität eines hochohmigen Widerstandes zu kennen.

Capacitor (Bild 5)
Bei einem Kondensator ist es möglich, dem Testsignal eine Gleichspannung zwischen 0,0 V und 5,0 V zu überlagern (siehe Abschnitt über DC-Bias). Die positiven Anschlüsse für diese Vorspannung sind J4/J5. Die Kapazität muss der dominierende Teil der Impedanz sein, sonst schaltet das LCR-Meter die DC-Biasspannung automatisch ab! Für die Messung von hochwertigen Aluminium-Elektrolytkondensatoren ist es wichtig, eine Testfrequenz zu wählen, die dieser Technologie entspricht und kleiner als ihre Resonanzfrequenz ist.

Inductor (Bild 6)
Bei einer Induktivität ist es möglich, dem Prüfsignal einen Gleichstrom zwischen 0 mA und 50 mA zu überlagern (siehe Abschnitt 3.1.5 in der Bedienungsanleitung). Es ist darauf zu achten, dass die durch den Bias-Strom am Innenwiderstand abfallende Gleichspannung kleiner als 0,3 V ist, da sonst die automatische Offsetschaltung des Differenzverstärkers U7 nicht mehr richtig arbeiten kann. Bei kleinen HF-Induktivitäten ist darauf zu achten, dass die Messung bei ausreichend hoher Frequenz durchgeführt wird, so dass der dominante Teil der Impedanz induktiv ist.
Ein Biasstrom 50 mA erhöht die Stromaufnahme des Messgeräts auf etwa 210 mA. Dies sollte ihre Stromquelle liefern können!

190311-B-02

Verfolgen Sie dieses Projekt auch auf der Elektor Labs-Projektseite [2], wo der Autor auch auf Fragen und Kommentare antwortet.

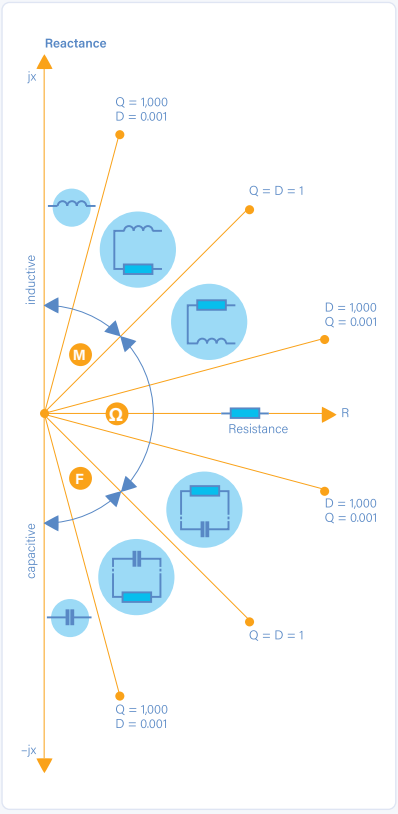


Bild 8. Äquivalenzschaltung basierend auf Qualitätsfaktor und reaktiver Komponente (Quelle: Fluke).

BESTIMMUNG VON BAUTEILTYP UND ERSATZSCHALTUNG

Dieses Gerät ist für die Messung von drei Arten von Bauelementen (Widerstände, Kondensatoren, Induktivitäten) ausgelegt. Für jeden dieser Typen sind die Messbedingungen entsprechend seinem Wert und seiner Verwendung zu wählen. Zusätzlich zum Wert des Hauptparameters, der den Bauelementetyp charakterisiert, gibt es auch einen sekundären Parameter, der hauptsächlich von der Testfrequenz abhängt. Das Gerät bewertet die Art der Komponente auf der Grundlage der Äquivalenzgleichung:

$$Z = Rs + jXs$$

aus dem sie den Qualitätsfaktor berechnet:

$$Q = |Xs|/Rs$$

Im AUTO-Modus hängt die Definition des Hauptparameters und der Ersatzschaltung vom Wert von Q (= 1/D) und des Vorzeichens des Blindanteils Xs ab. **Bild 8** zeigt die Beziehung zwischen Q, Xs und Ersatzschaltung.

Sie haben Fragen oder Kommentare?
Haben Sie Fragen oder Anmerkungen zu dem Artikel? Senden Sie eine E-Mail an den Autor unter jjacques.aubry@free.fr.

PASSENDE PRODUKTE

- **Elektor „Kickstarter-Projekt“**
  Bausatz mit Hauptplatine + Display-Erweiterungsplatine + alle Teile
  www.elektor.com/lcr

Ein Beitrag von
Idee, Gestaltung, Text und Illustrationen: **Jean-Jacques Aubry**
Redaktion: **Luc Lemmens, Denis Meyer**
Schaltbilder: **Kurt Diedrich, Patrick Wielders**
Übersetzung: **Rolf Gerstendorf**
Vergleichende Messungen: **Alfred Rosenkränzer**
Layout: **Giel Dols**

Fehleranalyse

Tipps zu Spannungsregler-Schaltungen, Platinendesign und mehr

Zusammengestellt von **C. J. Abate** (Elektor)

Ingenieure, Maker und Studenten lernen sowohl aus ihren eigenen Konstruktionsfehlern als auch aus den Fehlern ihrer Kollegen. In unserer Reihe „Fehleranalyse“ teilen wir und die Mitglieder der Elektor-Community ihre ingenieurtechnischen Fehler und Erfahrungen.

Ein Fall für U

Haben Sie schon einmal Stunden mit einem Elektronikprojekt zugebracht, nur um es aufgrund eines einfachen Fehlers in Rauch aufgehen zu sehen? Scott Coppersmith, ein leitender Forschungsingenieur an der Uni von Notre Dame in South Bend (USA), lernte so etwas kennen und teilte uns kürzlich seine Erfahrungen – und gab einen Tipp.

„Ich habe einmal für eine Beleuchtung meines Bootes LEDs in durchsichtigen Rohren aneinandergereiht. Ich rechnete mit einer empirisch bei Betrieb des Bootsmotors ermittelten Spannung von 13 VDC. Dann machte ich den Fehler, den Motor zu starten, ohne dass die Batterie angeschlossen war, und alle LEDs brannten fröhlich ab, als der Ausgang der Lichtmaschine auf 21 V sprang. Stunden und Stunden von Arbeit - und puff. Auto- und Schiffsbordnetze können je nach Ladezustand und Belastung der Batterie zwischen 8 V und 18 V oder mehr stark variieren. Es ist deshalb am besten, wenn Sie für solche Anwendungen eigene Spannungsregler vorsehen“.
Scott Coppersmith

Mehr zu Spannungsregler-Schaltungen, Stromversorgungen und LEDs

Suchen Sie nach weiteren Informationen über Stromversorgungssysteme, LEDs und Spannungsregler? Schauen Sie sich diese nützlichen Elektor-Artikel und -Videos an:

- Spannungsregler: www.elektormagazine.de/tags/Spannungsregler
- Spannungsversorgung: www.elektormagazine.de/tags/Spannungsversorgung
- Stromversorgungen: www.elektormagazine.de/tags/Stromversorgung
- C. Valens, „How to Calculate an LED Resistor“, Elektor August 2019: http://bit.ly/LED-resistor
- M. Heine, „LED-Booster für Mikrocontroller“, Elektor Mai 2020: www.elektormagazine.de/articles/ledbooster-fr-mikrocontroller
- C. Valens, „Video: Linearer LED-Treiber“, Elektor September 2020: www.elektormagazine.de/news/video-linearer-led-treiber

Alle Bauteile sind verdächtig

Suchen Sie nach Tipps zum Aufbau von Schaltungen? Chris Clapham ist ein in Auckland, Neuseeland, ansässiger Hardware-Ingenieur mit über 30 Jahren Erfahrung im Umgang mit Platinen. Er gibt hier einige großartige Ratschläge: Überprüfen Sie immer Ihre Bauteile, denn selbst neue können fehlerhaft sein.

„Auch brandneue Bauteile können verdächtig sein! Als ich vor über 30 Jahren für ein Recycling-Unternehmen fehlerhafte TV-Platinen nicht nur reparierte, sondern sie auch überholte, Elkos ersetzte, kalte Lötstellen heiß machte und alle irgendwie gestresst aussehende Bauteile austauschte, stieß ich bisweilen auf brandneue, aber dennoch fehlerhafte Komponenten. Dazu gehörten Transistoren, die sich als komplementär zur aufgedruckten Typenbezeichnung entpuppten (das heißt, ein PNP mit NPN-Teilenummer oder umgekehrt), Dioden, die sich fast so verhielten, als wären sie in Ordnung – was sie aber nicht waren, so dass die Stromversorgung sehr heiß wurde. Wir hatten sogar eine Diode, bei der Anode und Kathode vertauscht waren: Bei dem (wohl preiswert eingekauften) Bauteil, das der Fernsehherhersteller verwendet hatte, war die Kathode mit der Metallfahne des TO-220-Gehäuses verbunden, und diese war (ohne Isolation) auf den Kühlkörper geschraubt, und dieser war wiederum nicht geerdet. Bei dem neuen Ersatzbauteil war, wie es sich gehört, die Anode mit der Metallfahne verbunden, so dass sie beim Einbau einen Kurzschluss verursachte. Techniker vor Ort

ersetzten das kurzgeschlossene Originalteil und konnten dann nicht herausfinden, warum die Platine immer noch defekt war. Wenn sie die neue Diode herausnahmen, war die Messung gut. Und es dauerte lange, bis man herausfand, dass es nur eine Glimmerscheibe brauchte, damit es funktionierte.

Ein anderer Vorfall betraf eine Diode im DO-201-Gehäuse. Die kurzgeschlossene alte Diode auf der Platine wurde ersetzt, doch die Platine wollte nicht funktionieren. Nachdem wir alles mehrmals überprüft und weise Ratschläge von Arbeitskollegen eingeholt hatten, stellten wir fest, dass im Vergleich zu einer funktionsfähigen Platine die Anschlüsse vertauscht werden mussten, damit die Diode leitend wurde. Die Diode war zwar genauso ausgerichtet wie das alte Modell auf der Platine, aber die innere Verbindung im Bauteil war vertauscht. Jetzt weiß ich also, dass ich nicht nur überprüfen muss, ob eine neue Diode in die eine Richtung leitet und in die andere nicht, sondern dass sie auch korrekt ausgerichtet eingebaut werden muss, egal, was der Aufdruck auf dem Bauteil oder der Platine besagt."

> „Auch brandneue Bauteile können verdächtig sein!"

Chris Clapham

200562-02

Teilen Sie Ihre Fehleranalyse

Sind Sie professioneller Elektroingenieur, DIY-Konstrukteur oder E-Technik-Student? Möchten Sie Einzelheiten über Ihre Fehler und Erkenntnisse im Zusammenhang mit Elektronik austauschen? Dann füllen Sie einfach unser Formular „Fehleranalyse" aus.
www.elektormagazine.com/news/error-analysis-submission

Mehr zu Platinen, Platinendesigns und Prototypen

Interessiert an Tipps zu und Einblicke in Platinenentwurf und Prototyping?

- ElektorTV, „PCB-Layout with Pcbnew", 5.6.2020: https://youtu.be/-_nZZLuwYd0
- C. Valens, „More 3D Libraries for KiCad EDA", Elektor 10.7.2019: www.elektormagazine.com/news/kicad-3d-library
- „ElektorPCB4Makers: preiswerter und umweltfreundlicher Platinen-Service", Elektor 29.1.2020: www.elektormagazine.de/news/elektorpcb4makers-your-affordable-environmentally-friendly-pcb-service
- „KiCad-Plugins und -Add-Ons", Elektor November 2020
- „Erste Schritte mit KiCad - Schaltpläne zeichnen mit Eeschema", Elektor Juni 2020: www.elektormagazine.de/news/erste-schritte-mit-kicad-schaltplane-zeichnen-mit-eeschema

Das Open Hardware Observatory

Community-basierte Bewertung von Open-Source-Hardware

Von **Tessel Renzenbrink** (Niederlande)

Das Open Hardware Observatory (OHO) ist eine Initiative, die sich zum Ziel gemacht hat, Offene Hardware zugänglicher zu machen. Sie stellt eine Suchmaschine zur Verfügung, die Ihnen hilft, Open-Hardware-Projekte zu finden, die irgendwo im Web veröffentlicht sind. Und sie entwickelt derzeit eine community-basierte Plattform zur Bewertung von Open-Source-Hardware-Dokumentation auf der Grundlage der DIN SPEC 3105. Das Projekt wurde vom gemeinnützigen Verein Open Source Ecology Germany e.V. und der TU Berlin gegründet. Lukas Schattenhofer arbeitet derzeit - zusammen mit Consuelo Alberca Susano, Mehrdad Mansouri und Nils Weiher – am Aufbau der Community rund um das OHO. Schattenhofer erklärt, wie das funktioniert und lädt die Elektor-Leser ein, sich der Gemeinschaft anzuschließen.

Open-Source-Hardware (OSH) ist Hardware, deren Entwurf öffentlich zugänglich gemacht wird, so dass jede Person den Entwurf oder die auf diesem Entwurf basierende Hardware studieren, modifizieren, verbreiten, herstellen und verkaufen kann [1]. „Um die Vorteile von Open Hardware zu verstehen, müssten wir unsere Perspektive auf Technologie ändern," sagt Lukas Schattenhofer. „Für die meisten Menschen ist Technologie derzeit etwas, das man eher konsumiert als baut und instand hält. Wenn ein Teil ihres Handys kaputt geht, kaufen sie ein neues, anstatt es zu reparieren. Heute ist die meiste Technologie verschlossen. Das bedeutet, dass die Informationen, die man für Reparaturen benötigt, nicht zugänglich sind. Aber wenn die gesamte Dokumentation verfügbar wäre, bräuchten Sie nicht alle zwei Jahre ein neues Handy zu kaufen. Sie wären in der Lage, das Gerät zu reparieren und sogar zu bauen. Außerdem würde sich dadurch die Art und Weise ändern, wie Produkte entworfen werden. Sie würden modularer werden, so dass Sie Teile austauschen und Produkte miteinander kombinieren könnten."

„Die Dokumentation ist die Quelle"

Doch obwohl es eine florierende Open-Hardware-Community gibt, ist OSH immer noch viel weniger bekannt als ihr Gegenstück, die Open-Source-Software. Das Open Hardware Observatory will OSH sichtbarer machen, indem es ein Online-Dokumentationsarchiv freier Projekte zur Verfügung stellt [2]. Zu diesem Zweck hat das OHO eine Suchmaschine entwickelt, die das Web nach Projekten durchforstet und diese mit Fotos und einer kurzen Beschreibung aufgelistet. Aber nicht alle Projekte im Archiv sind wirklich frei. Hier kommt der zweite Zweig des OHO ins Spiel - die Bewertungsplattform. Sie hilft Makern, ihr Projekt so zu präsentieren, dass es den Standards für OSH entspricht.

Es gibt zwei Hauptkriterien dafür, dass ein Projekt frei ist. Das erste sind die Lizenzen: Code oder Firmware ist eigentlich automatisch urheberrechtlich geschützt. Wenn ein Maker Software in sein Projekt aufnimmt, muss er dieser aktiv eine offene Lizenz wie die GNU General Public License oder die MIT-Lizenz zuweisen [3]. Eine Produktdokumentation, insbesondere wenn sie umfangreich ist, fällt oft auch unter das Urheberrecht. Nicht allen Makern ist es bewusst, dass sie explizit eine offene Lizenz zuweisen müssen, um anderen Usern die legale Wiederverwendung des Codes oder die Verbreitung der Dokumentation zu ermöglichen.

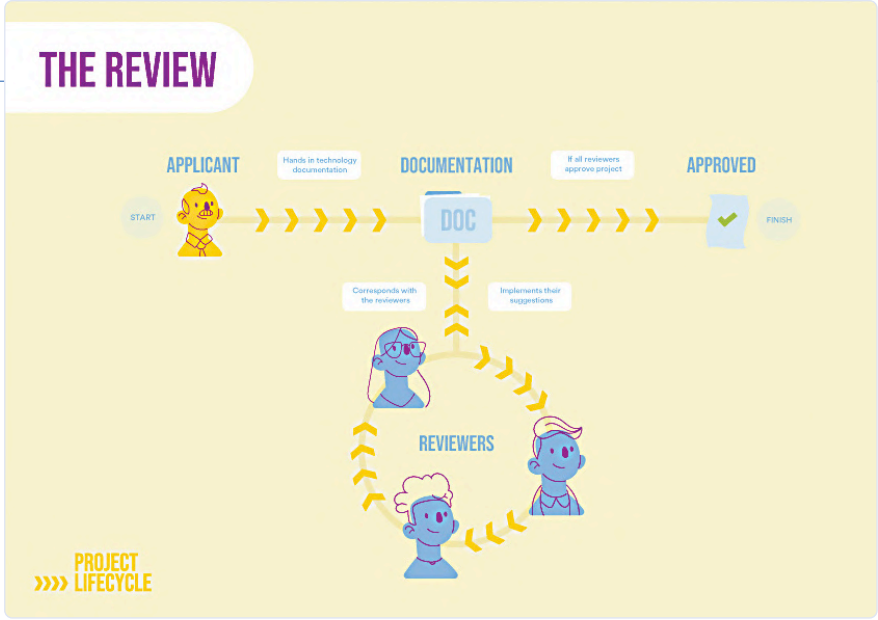
Das zweite Kriterium betrifft die Dokumentation. Schattenhofer: „Die Dokumentation ist wichtig. Bei OSH geht es darum, Menschen dazu zu befähigen, die Hardware zu studieren, zu modifizieren und herzustellen. Und dafür braucht man die Entwürfe. Man kann den Quellcode nicht herunterladen, wie man es bei Software tut. Bei Hardware *ist* die Dokumentation die Quelle."

Daher muss die Dokumentation ein Mindestmaß an Anforderungen erfüllen. Ein begeisterter Maker, der ein Youtube-Video seines selbstgebauten Tropfenbewässerungssystems hochgeladen hat, möchte es wahrscheinlich mit anderen teilen. Wenn das Video jedoch nicht die für den Nachbau des Projekts erforderlichen Informationen enthält, ist es nicht wirklich Open Source.

Plattform zur Bewertung der Dokumentation

Schattenhofer und das Team arbeiten derzeit an der Bewertungsplattform. Der Überprüfungsprozess funktioniert wie folgt. Ein Maker lädt ein Projekt hoch. Drei Gutachter beurteilen die Dokumentation. Sie können Kommentare hinzufügen, um darauf hinzuweisen, welche Teile fehlen oder verbessert werden sollten. Wenn die Dokumenta-

tion alle Anforderungen erfüllt, erhält das Projekt ein Zertifikat.
Die Anforderungen an die Dokumentation sind in der DIN SPEC 3105 festgelegt. Dabei handelt es sich um eine öffentlich verfügbare Technische Regel, die durch das Deutsche Institut für Normung (DIN) veröffentlicht wird. Sie legt konkret messbare Kriterien für eine OSH-konforme Dokumentation fest. Die DIN SPEC selbst ist ein Ergebnis eines kollaborativen Prozesses der OSH-Community. Und in Übereinstimmung mit den Open-Source-Prinzipien kann jeder über die Gitlab-Seite [4] zur Weiterentwicklung der DIN SPEC 3105 beitragen.
Es gibt bereits einen OSH-Zertifizierungsprozess, der von der Open Source Hardware Association (OSHWA) [5] bereitgestellt wird. Auch er ist das Ergebnis des Bedürfnisses der OSH-Community, über klare Richtlinien zu verfügen, um festzustellen, ob ein Projekt wirklich frei ist. Der OSHWA-Zertifizierungsprozess konzentriert sich jedoch hauptsächlich auf den lizenzrechtlichen Aspekt von Open-Source. Was noch fehlte, waren Anforderungen hinsichtlich der Dokumentation von OSH, mit anderen Worten, eine klare Definition dessen, was eine Quelle ausmacht. Die Bewertungsplattform von OHO, die auf den in DIN SPEC 3105 festgelegten Anforderungen basiert, strebt an, diese Lücke zu schließen.



Schematischer Überblick über den community-basierten Überprüfungsprozess einer Open-Source-Hardware-Dokumentation (Bild: Mit freundlicher Genehmigung des Open Hardware Observatory).

> Das Open Hardware Observatory (OHO) will OSH sichtbarer machen, indem es ein Online-Dokumentationsarchiv freier Projekte zur Verfügung stellt.

Der Community beitreten

Die Bewertungsplattform ist bereits in Betrieb, befindet sich aber dennoch in der Entwicklung. Die Entwickler erhielten Rückmeldungen aus der Community mit dem Wunsch nach einem intuitiveren Arbeitsablauf. Daher arbeiten sie derzeit an der Verbesserung der Benutzererfahrung. Schattenhofer und das Team laden die Elektor-Leser ein, der OHO-Community beizutreten. Sie können teilnehmen, indem Sie Ihr Projekt zur Bewertung hochladen. Oder Sie können Ihr technisches Wissen als Gutachter einsetzen. Aber, so warnt Schattenhofer, Sie sollten bereit sein, sich mit einer Website auseinanderzusetzen, die noch nicht ganz fertig ist. Es gibt also eine dritte Rolle, die Sie im OHO-Projekt übernehmen können. Sie können sich als Tester der Plattform engagieren und mithelfen, eine bessere Benutzeroberfläche und einen besseren Workflow zu schaffen.
Schattenhofer: „Im Moment kennen noch nicht viele Leute die DIN SPEC 3105 und die community-basierte Bewertungsplattform. Aber wir wollen eine aktive Community rund um das Open Hardware Observatory aufbauen, um die Verbreitung von Open-Source-Hardware voranzutreiben. Menschen, die jetzt der OHO-Community beitreten, können wirklich Teil des Entwicklungsprozesses sein. Wenn Sie sich jetzt beteiligen, können Sie Ihre eigenen Ideen einbringen. Wenn Sie daran interessiert sind, dem OHO beizutreten, kontaktieren Sie uns bitte unter info@oho.wiki."

200560-03

Hinweis: OHO ist eine Kooperation des gemeinnützigen Vereins Open Source Ecology Germany e.V. (OSEG) und des deutsch-französischen Forschungsprojekts OPEN! Methods and tools for community-based product development [6].

WEBLINKS

[1] Diese weit verbreitete Definition wurde von der Open Source Hardware Association aufgestellt (keine Verbindung zum OHO).
[2] **Open Hardware Observatory**: https://de.oho.wiki/wiki/Offene_Hardware
[3] **Liste der offenen Lizenzen**: https://opensource.org/licenses
[4] **DIN SPEC3105**: https://gitlab.com/OSEGermany/OHS/
[5] **Open Source Hardware Association Certification**: https://certification.oshwa.org/
[6] **Gründungspartner**: https://de.oho.wiki/wiki/Gr%C3%BCndungspartner

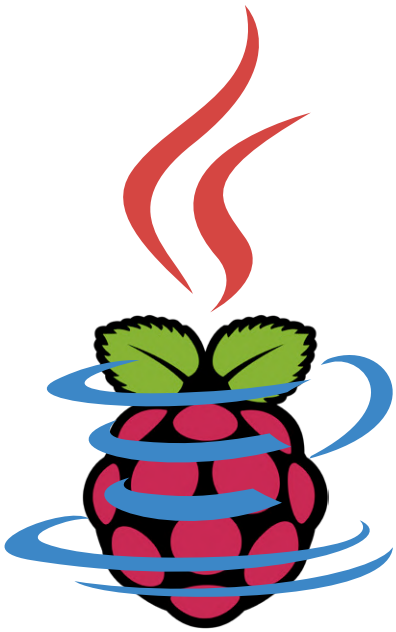
Java auf dem Raspberry Pi

Ein Interview mit Buch-Autor Frank Delporte

Von **C. J. Abate**

Möchten Sie Java auf einem Raspberry Pi laufen lassen? Frank Delporte kann helfen. Sein neues Buch „Getting Started with Java on the Raspberry Pi“ ist eine ausgezeichnete Ressource sowohl für professionelle Programmierer als auch für Maker, die experimentieren und lernen wollen

Möchten Sie Java auf der Raspberry Pi laufen lassen? Frank Delporte aus Belgien kann Ihnen dabei helfen. Sein neues Buch „Getting Started with Java on the Raspberry Pi“ [1] ist eine ausgezeichnete Ressource sowohl für professionelle Programmierer als auch für Maker, die experimentieren und lernen wollen. In diesem Interview spricht Frank Delporte mit C.J. Abate über die Vorteile der Kombination von Java und Raspberry Pi sowie über seine Erfahrungen als Programmierer.



Programmieren, Entwickeln und Schreiben

Abate*: *Herzlichen Glückwunsch zur Veröffentlichung Ihres Buches Getting Started with Java on the Raspberry Pi (Elektor 2020). Ich werde Sie gleich mehr zum Buch fragen. Aber zuerst: Sind Sie zufrieden, wo Sie jetzt das ganze Schreiben und Redigieren hinter sich haben? Oder gefällt Ihnen diese Art von Arbeit?

***Delporte*: Ich schreibe wirklich gerne, wie Sie auch auf meinem Blog [2] sehen können. Aber ich muss zugeben, dass so ein Buch eine Menge Arbeit ist. Man muss nicht nur schreiben, sondern auch alle erforderlichen Informationen sammeln, recherchieren, Experimente durchführen, Schaltungen zeichnen, Andere interviewen, redigieren usw. Aber das Gefühl, ein eigenes, fertig gedrucktes Buch aus Papier in der Hand zu halten, ist einmalig!. Es war die ganze harte Arbeit wirklich wert.**

Abate*: *Apropos Schreib- und Redaktionsarbeit: Im Mai 2020 wurde Ihr Heimarbeitsplatz bzw. Ihr Elektronik-Labor auf unserer englischen Website vorgestellt [3]. Arbeiten Sie aufgrund der COVID-19-Situation immer noch von zu Hause aus?

***Delporte*: Die Situation hat sich mehr oder weniger wieder normalisiert. Ich arbeite immer noch von zu Hause aus, aber nicht wirklich wegen Covid-19. Auf der Arbeit können wir genug Abstand halten, aber von Zeit zu Zeit ist es immer noch besser, zu Hause zu arbeiten, damit man sich auf eine bestimmte Aufgabe besser konzentrieren kann.**

Abate*: *Sie haben einen interessanten Background, der die Arbeit als Software-Entwickler, technischer Leiter, Autor und Video-Produzent umfasst. Erzählen Sie uns bitte mehr darüber.

***Delporte*: Ich habe mich schon immer für Technik interessiert und dafür, wie Dinge funktionieren. Ich war die Art von Kind, das alles auseinander nahmen: jede Kaffeemaschine und jedes Radio oder welches Gerät auch immer kaputt war. Ich konnte nicht alle reparieren, aber ich habe jedes Mal etwas Neues gelernt! Als Teenager hatte ich eine eigene Radiosendung als DJ bei einem lokalen Sender, was mir die Möglichkeit gab, mehr mit Elektronik zu experimentieren. Deshalb beschloss ich, an einer (technischen) Filmhochschule zu studieren. Dort lernte ich, wie Filme, Radio und Fernsehen produziert werden, wie man Kameras kalibriert und alle Geräte und Aufnahmen miteinander verbindet. Nach meinem Abschluss hat sich die Videoproduktion mit der Einführung des Computer-Schnitts stark verändert, und so bin ich wieder ins Programmieren eingestiegen, als Kunden ihr Firmenvideo auf CD-ROM und im Internet haben wollten.**

Abate*: *In Ihrer Biographie erwähnen Sie den Commodore 64. Erinnern Sie sich an einige Ihrer frühen Erfahrungen mit ihm?

Bild 1. Das neue Buch von Delporte.

Bild 2. Die gesteuerte LED-Leiste.

***Delporte*:** Ich hatte nur ein Spiel auf meinem C64, weil ich mich immer mehr für das Programmieren interessiert habe. Dank Elektor fand ich ein Buch (muss um 1987 herum sein) mit einer Elektronikplatine mit acht Relais, die man mit dem C64 und Basic steuern konnte. Ich benutzte es, um meinen Lego-Zug zu steuern und benutzte Magnetschalter, um die Position dieses Zuges via Joystick-Ports zu erkennen. Das war das erste Mal, dass es mir gelang, Software und Hardware zu kombinieren. Heutzutage wäre ein solches Projekt mit Arduino oder einem Raspberry Pi und den vielen tollen Erweiterungsplatinen viel einfacher (und billiger).

***Abate*:** *Was waren Ihre beruflichen Ziele 1994, als Sie das NARAFI (Nationaal Radio en Filmtechnisch Instituut) in Belgien verließen?*
***Delporte*:** Mein erster Job war Videoschnitt bei einem lokalen Fernsehsender. Nach einigen Jahren begann ich, das als Freiberufler anzubieten. Ich hatte nie ein klares Karriereziel, sondern rollte von Job zu Job und lernte dabei neue Dinge. So wandelte ich mich vom Videoproduzent zum Multimedia-Entwickler und erstellte Firmenpräsentationen auf DVD und CD-ROM. Als das Internet zu wachsen begann, wechselte ich zur Web-Entwicklung, da meine Kunden die gleichen Informationen auf Webseiten mit einem Content-Management-System teilen wollten. Und mit diesem Wissen wuchs ich in die Java-Entwicklung hinein und übernahm die technische Leitung bei der Produktentwicklung mit dieser Technologie.

Unterricht im Programmieren

***Abate*:** *Wann haben Sie begonnen, CoderDojo-Sitzungen zu organisieren? Und welche Art von Kursen unterrichten Sie?*
***Delporte*:** In jedem Unternehmen, in dem ich gearbeitet habe, war es eine Herausforderung, Kollegen mit gutem technischen Know-How zu finden. Das Ingenieurwesen (und auf jeden Fall die IT) ist immer noch ein sehr männliches Gebiet. Für mich ist das Ingenieurwesen pure Magie. Mit ein paar Zeilen Code oder ein paar elektronischen Bauteilen kann man „Sachen" bauen. Ich fragte mich, warum Kinder - die ja gerne bauen und experimentieren - plötzlich damit aufhören und kein Studium wählen, wo sie weiterhin „Dinge bauen" können.
CoderDojo [4] ist ein kostenloser Club, in dem Kinder von 7 bis 18 Jahren freiwillig lernen, mit „digitalem Zeug" zu experimentieren. Wir verwenden das blockbasierte Scratch zur Programmierung, bauen Welten in Minecraft mit JavaScript, steuern Elektronik mit Arduino, bauen Roboter mit Lego und vieles mehr. Im Jahr 2013 habe ich einen solchen Club in den Orten Ieper und Roeselare gegründet und leite immer noch den in Ieper - außer jetzt zu Corona-Zeiten, da es schwierig ist, solche informellen Clubs zu organisieren, wenn man einen Sicherheitsabstand einhalten muss und sich nicht mit verschiedenen Leuten um einen PC versammeln kann. Dank des CoderDojo und anderer STEM-Initiativen (Science, Technology, Engineering and Mathematics) sehen wir einen langsamen Anstieg der Zahl der Ingenieurstudenten – übrigens Jungen und Mädchen!

***Abate*:** *An welchem Punkt haben Sie erkannt: „Hey, ich bin ein guter Ausbilder/Lehrer, und ich denke, ich kann anderen helfen, die sich für Java interessieren"? Oder hat Sie ein Freund oder Kollege in diese Richtung geführt?*
***Delporte*:** Ich mag es, Dinge zu erklären und glaube fest an „Lernen durch Lehren". Das ist es, was ich im CoderDojo mit Kindern, aber auch in meinem Blog und bei der Arbeit tue. Um ein Thema vollständig zu verstehen, muss man in der Lage sein, es zu erklären - oder umgekehrt. Die Artikel, die ich für meinen Blog schreibe, sind immer das Ergebnis von etwas, das ich ausprobieren möchte, aber (noch) nicht wirklich weiß, wie ich es machen soll. Während des Ausprobierens schreibe ich auf, welche Schritte ich gemacht habe und was funktioniert hat und was nicht. So lerne ich neue Dinge und kann dieses Wissen mit anderen teilen.

Ein Schwerpunkt auf Java

***Abate*:** *Erzählen Sie uns bitte von Ihren Erfahrungen mit Java. Haben Sie es zuerst aus Neugierde gelernt? Oder war es für einen Kurs? Oder für die Arbeit?*
***Delporte*:** Als ich begann, Multimedia-Anwendungen zu entwickeln, musste ich ActionScript lernen (und davor sogar Lingo). Später wechselte ich zu C# und SQL für die Webanwendungen. Wie Sie jetzt vielleicht schon wissen, lerne ich durch Experimentieren. Aber auch durch das Lesen von Büchern und kurzen (Online-) Kursen. Als ich 2010 bei Televic Rail anfing, schloss ich mich einem Team an, das bereits Java benutzte. Der Wechsel von C# zu Java war sehr einfach. Nach all diesen Jahren der Programmierung muss ich sagen, dass ich am meisten von Kollegen lerne! Das Teilen der Arbeit in Präsentationen mit anderen, den Code mit Pull-Anfragen zu verbessern, Kommentare zu akzeptieren, wie man Dinge besser machen kann, sind die besten Möglichkeiten, von anderen zu lernen.
Seit ich angefangen habe, mit Java auf Raspberry Pi zu experimentieren, habe ich mich an einigen Open-Source-Projekten und -Diskussionen beteiligt, und das

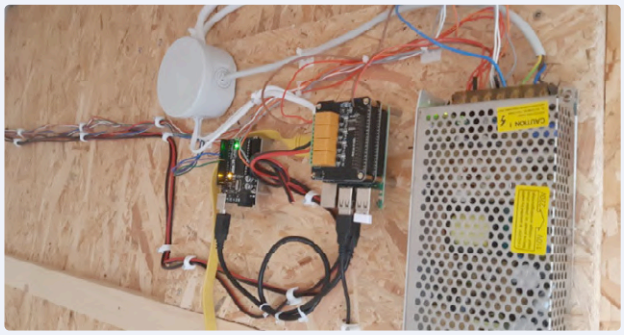


Bild 3. Installiertes System.

Bild 4. Regenbogeneffekt mit LED-Streifen.

ist eine völlig neue Welt für mich, in der ich viele sehr kluge Leute treffe, die auch bereit sind, ihr Wissen und ihre Erfahrung zu teilen. Noch immer bin ich jeden Tag aufs Neue erstaunt, was man von diesen Projekten und Menschen lernen kann. Man muss keinen Code beisteuern, aber man kann sich auch an einem solchen Projekt beteiligen, indem man Pull-Anfragen prüft, beim Testen hilft oder den Code dokumentiert.

Abate: *Haben Sie etwas gegen Python oder gegen C? Ich vermute: eher nicht, aber ich muss Sie das fragen.*
Delporte: Definitiv nicht! Ich hasse Hasser. Es gibt keine schlechten Programmiersprachen! Ich habe einmal ein Zitat gelesen, in dem es heißt: „Das beste Werkzeug für die Arbeit ist das, das man am besten kennt". In meinem Fall ist das Java und JavaFX, wenn ich eine Anwendung mit einer schönen Benutzeroberfläche erstellen will. In meinem Buch habe ich aber auch Python verwendet, um eine LED-Zahlenanzeige und einen Arduino mit LED-Streifen zu steuern. Für jedes Projekt muss man eine Entscheidung treffen, was das beste Werkzeug, die beste Programmiersprache oder das beste Framework ist. Und wenn man sich entschieden hat, legt man los! Vielleicht merkt man erst später, dass man nicht die beste Wahl getroffen hat, aber zumindest hat man dabei dann neue Dinge gelernt.

Arbeiten mit Java auf dem Raspberry Pi

Abate: *Sie bloggen seit 2007 über Technologie. Ihr erster Beitrag zum Raspberry Pi war wohl „Pong on a Raspberry Pi" [5] (Dezember 2017). Können Sie uns etwas über Ihre ersten Erfahrungen und Projekte mit RPi berichten? Wann haben Sie damit angefangen?*
Delporte: Als ich mit CoderDojo begann, gab es einige Trainer, die bereits Erfahrung mit Arduino und Raspberry Pi hatten. Sie brachten ihre Bausätze mit in den Club, und ich war wirklich erstaunt über die Leistungsfähigkeit dieser preiswerten Boards und was man damit in Kombination mit kleinen elektronischen Bauteilen erreichen kann.
Ich habe schon seit einiger Zeit gebloggt, aber mein erstes „öffentliches" Raspberry Pi-Projekt war tatsächlich dieses Pong-Spiel, das wir bei einigen Aktivitäten der Schule meines Sohnes verwendeten. Für die Benutzeroberfläche verwendete ich Python; aber ich muss ehrlich sein, ich habe es nicht sehr gerne programmiert. Für diese Art von Anwendung bevorzuge ich wirklich JavaFX, für das es sogar ein sehr schönes Gaming-Framework gibt: FXGL [6].

Abate: *Haben Sie irgendwelche RPi-basierten Designs oder Anwendungen, die zu Hause oder in Ihrem Arbeitsbereich laufen?*
Delporte: Ich habe mit Java auf der Raspberry Pi begonnen, um für meinen Sohn einen Schlagzeug-Controller [7] zu bauen. Es handelt sich dabei um eine Touchscreen-Benutzeroberfläche zur Steuerung mehrerer Lichter mit einer Relaiskarte und LED-Streifen, die von Arduino gesteuert werden.
Auf diese Weise habe ich gelernt, serielle Kommunikation zwischen den beiden Platinen und I^2C zur Steuerung der Relais zu nutzen. In meinem Buch habe ich dies weiter ausgebaut und eine Mosquitto-Warteschlange verwendet, um Nachrichten zwischen weiteren Platinen und PCs auszutauschen.

Abate: *Woran arbeiten Sie zurzeit? An neuen Projekten, Programmen oder Büchern?*
Delporte: Ich experimentiere natürlich weiter mit Java auf dem Raspberry Pi. Ich habe weitere Blog-Beiträge zu diesem Thema geschrieben und auch mit anderen Java-Technologien (Quarkus [8], Spring, GraalVM) und 64-Bit-Betriebssystemen auf dem Board experimentiert.
Ich habe mich auch dem Pi4J-Team angeschlossen. Pi4J ist ein Framework und eine Bibliothek, um Java-Anwendungen mit den vollen Möglichkeiten der RPi-GPIOs zu kombinieren. Dieses Projekt wurde von Robert Savage ins Leben gerufen. Er war auf der Suche nach zusätzlichen Teammitgliedern, um dieses Projekt auf eine neue Version zu bringen, die Java 11+ und RPi 4 mit Java-Modulen und einer leicht erweiterbaren Architektur voll unterstützt. Ich bin sehr gespannt auf die zweite Version dieses Frameworks, die wir hoffentlich bald veröffentlichen werden.

Abate: *Zurück zum Buch „Getting Started with Java on the Raspberry Pi". Warum haben Sie es geschrieben?*
Delporte: Als ich mit dem Schlagzeug-Projekt begann, musste ich herausfinden, wie man Java auf dem Raspberry Pi benutzt, wie man die richtige Version von JavaFX installiert, wie man GPIOs und einen Arduino steuert, usw. Zu diesem Zeitpunkt schrieb ich meinen ersten Artikel [9], der in MagPi veröffentlicht wurde (Juli 2019, niederländische [10] und französische [11] Ausgabe). Elektor kontaktierte mich und fragte, ob dies ein Thema für ein Buches sein könnte. Da ich kein aktuelles Buch zu diesem Thema finden konnte und Java in den letzten Jahren einige große Veränderungen erfahren hat, packte mich diese Frage richtig, und am nächsten Tag begann ich mit dem Schreiben. Ich brauchte mehr als sechs Monate und viele Abende und Nächte, aber ich hatte wirklich Spaß beim Schreiben und Experimentieren. Und natürlich hoffe ich, dass es genauso viel

Spaß macht, das Buch zu lesen und die Projekte auszuprobieren, die ich darin aufgenommen habe.

***Abate*:** *Haben Sie irgendwelche Ratschläge für Ingenieure oder Maker, die darüber nachdenken, Java für ihre auf RPi basierenden Designs zu verwenden?*
***Delporte*:** Probieren Sie es aus, wirklich! Java ist immer noch eine der Top-Programmiersprachen. Ob Sie ein erfahrener Java-Entwickler sind oder neu anfangen – es gibt eine Menge zu lernen und zu experimentieren, wenn Sie Java mit einer RPi und elektronischen Bauteilen kombinieren.
Die Beispiele in meinem Buch verwenden sehr billige Teile wie LEDs, Taster, LCDs, LED-Displays usw., so dass Sie diese vielleicht sowieso schon haben (oder in jedem Starter-Kit finden). Die Beispiele aus dem Buch können Sie zu Ihrem Traumprojekt kombinieren. Der Schlagzeug-Controller, den ich für meinen Sohn gebaut habe, ist eine Kombination aus mehreren dieser Beispiele.

***Abate*:** *Wie ist das Feedback bisher ausgefallen?*
***Delporte*:** Obwohl Python eine Standardsprache für RPi ist (daher kommt schließlich das „Pi“) und einige immer noch glauben, dass Python die einzig richtige Wahl ist, habe ich eine Menge positives Feedback und Fragen zu diesem Thema erhalten. Ich konnte darüber sogar einen Beitrag für Oracles Java Magazine schreiben, das viele Leser angesprochen hat! Es besteht ein klares Interesse an diesem Thema, und die zukünftige neue Generation von Pi4J wird es noch einfacher machen, leistungsfähige Anwendungen zu erstellen.

***Abate*:** *Gibt es eine Programmiersprache, die Sie nicht kennen, die Sie aber lernen möchten? Gibt es eine Hardware, die Sie auszuprobieren gedenken?*
***Delporte*:** Java ist nicht nur eine Programmiersprache, sondern auch eine virtuelle Maschine, die den Java-Code ausführt. Auf derselben VM können Sie auch Scala, Kotlin und viele andere Sprachen ausführen. Es gibt also noch viel zu erforschen in dieser Welt. Für das Pi4J-Projekt möchte ich den Beispielcode und die Dokumentations-Webseite erweitern, so dass ich mehrere kleine Hardware-Beispiele einrichten und selbst eine Menge neues Material lernen muss.

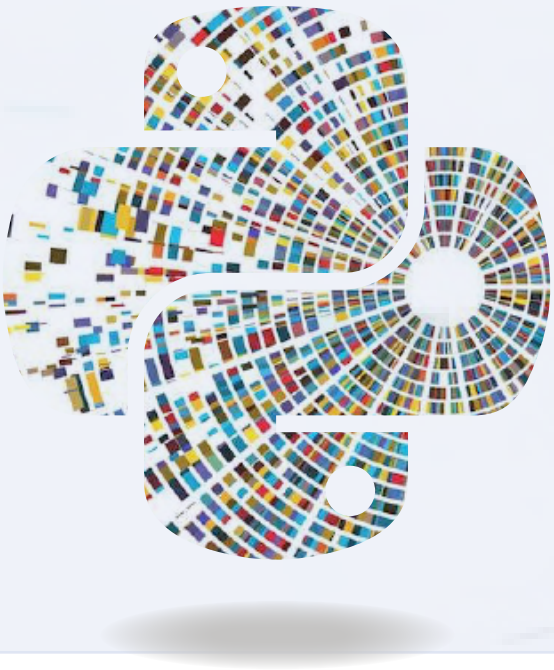
Erfolgreiches Programmieren

***Abate*:** *Lassen Sie uns mit Ihrem größten Erfolg als Ingenieur oder Programmierer abschließen. Gibt es ein bestimmtes Projekt (Software oder Hardware), das herausragt? Was war an diesem Projekt schwierig? Was haben Sie gelernt?*
***Delporte*:** In meinem Job bei Televic verwenden wir eine Kombination aus Java und Embedded Programming, um mehrere Server und Datenquellen miteinander zu verbinden und Fahrgastinformationen in Echtzeit auf die Bildschirme ganzer Zugverbände zu bringen. Es ist sehr befriedigend, diese technische Herausforderung zu lösen und schließlich durch einen fahrenden Zug mit 100 Bildschirmen zu gehen, auf denen die abfahrenden Züge an der nächsten Station mit Echtzeit-Verspätungen und Bahnsteigwechseln angezeigt werden.
Der Datenfluss, der für die Übertragung all dieser Daten über unzuverlässige drahtlose Verbindungen erforderlich ist (Mobilfunk-Signale sind nicht wirklich für schnelle Fahrzeuge ausgelegt), ist ein wahres Meisterwerk, auf das ich sehr stolz bin, dass wir es mit einem kleinen Team realisieren konnten. Aber ich bin ebenso beeindruckt von den Kindern im CoderDojo, die es schaffen, ihr erstes Flappy-Bird-Spiel in Scratch zu konstruieren oder mit Arduino eine LED blinken zu lassen!

200503-02

WEBLINKS

[1] www.elektor.de/getting-started-with-java-on-the-raspberry-pi
[2] http://webtechie.be/
[3] www.elektormagazine.com/news/ electronics-workspace-software-developers-space
[4] http://coderdojo.com/
[5] http://webtechie.be/post/2017-12-20-pong-on-a-raspberry-pi/
[6] http://webtechie.be/post/2020-05-07-getting-started-with-fxgl/
[7] http://webtechie.be/post/2020-03-30-drumbooth-controller-with-java-javafx-raspberrypi-arduino/
[8] http://webtechie.be/post/2020-07-28-spring-versus-quarkus-rest-h2-db-on-raspberry-pi/
[9] http://webtechie.be/articles/
[10] www.magpi.nl/
[11] www.magpi.fr/
[12] http://blogs.oracle.com/javamagazine/getting-started-with-javafx-on-raspberry-pi



Datenanalyse und künstliche Intelligenz in Python

Interpretation realer Daten mit Numpy, Pandas und Scikit-Learn

Das Analysieren und Interpretieren von Daten aus realen Prozessen ist ein sehr interessantes Thema. Solche Daten gehören heute zu unserem täglichen Leben: Denken Sie an die verfügbaren Informationen über Klimaprozesse oder an alle Informationen, die bei intelligenten Fertigungsprozessen gewonnen werden. Diese Datenmengen erlauben es theoretisch, jedes Phänomen zu charakterisieren; der Umgang damit erfordert jedoch die Beherrschung verschiedener theoretischer und praktischer Begriffe. Lassen Sie uns gemeinsam einige dieser Techniken entdecken und Python bei der Analyse eines realistischen Szenarios verwenden.

Von **Angelo Cardellicchio**

Begriffe wie *Big Data* und *Künstliche Intelligenz* sind inzwischen zu einem festen Bestandteil der Alltagssprache geworden. Dies ist vor allem auf zwei Faktoren zurückzuführen: Zum einen hat die zunehmende und weit verbreitete Verbreitung von Datenerfassungssystemen die Schaffung praktisch endloser *Repositories* des Wissens ermöglicht, zum anderen hat die Zunahme der Rechenkapazitäten, die auch auf den weit verbreiteten Einsatz von GPUs [1] zurückzuführen ist, es ermöglicht, Probleme zu bewältigen, deren Lösung einst unmöglich war.

Nehmen wir ein Beispiel, das wir in diesem Artikel als Anwendungsszenario verwenden werden. Stellen wir uns vor, wir müssten eine ganze Produktionskette überwachen (unabhängig vom Endprodukt, das in diesem Zusammenhang keine Rolle spielt). Wir haben die Möglichkeit, Daten aus verschiedenen Quellen zu erfassen: Wir könnten zum Beispiel Sensoren entlang der gesamten Produktionslinie einsetzen oder *Kontextinformationen* verwenden, die sich auf das Alter und den Typ jeder Maschine beziehen. Dieser Datensatz (*Dataset*) kann für verschiedene Zwecke verwendet werden: Die vorausschauende Wartung ermöglicht uns beispielsweise, den Beginn anormaler Situationen zu bewerten und vorherzusagen, um einen Austausch oder eine Reparatur *vor* dem Ausfall zu planen, was zu Einsparungen und erhöhter Produktivität führt. Darüber hinaus ermöglicht die Kenntnis der Datenhistorie eine Korrelation der von jedem Sensor gemessenen Größen, wodurch mögliche Ursache-Wirkungs-Beziehungen aufgezeigt werden. Wenn beispielsweise auf einen plötzlichen Anstieg der Temperatur und der Luftfeuchtigkeit in einem Raum ein Rückgang der Anzahl der gefertigten Teile folgt, kann es erforderlich sein, die Klimaanlage des Raumes zu verbessern, um konstante klimatische Bedingungen aufrechtzuerhalten.

Die Implementierung eines solchen Systems ist sicherlich nicht für jeden erreichbar, wird aber durch die von der Open-Source-Community zur Verfügung gestellten Werkzeuge vereinfacht. Für den Anfang brauchen Sie also nur einen PC (oder alternativ auch unseren treuen Raspberry Pi, wenn die zu verarbeitende Datenmenge nicht zu groß ist), Kenntnisse in Python (die Sie durch Tutorials [2] vertiefen können) und natürlich die Tools, das heißt, die Konzepte und grundlegenden Werkzeuge. Lassen Sie uns diese gemeinsam entdecken!

Die Tools

Unnötig zu sagen, dass wir zunächst in der Lage sein müssen, Programme zu erstellen, die in der Sprache Python geschrieben sind. Dazu müssen wir den Interpreter installieren, der auf der offiziellen Website von Python [3] zu finden ist. Im weiteren Verlauf dieses Artikels gehen wir davon aus, dass Sie Python bereits installiert und zu den Umgebungsvariablen des Systems hinzugefügt haben.

Die virtuelle Umgebung

Sobald die Einrichtung von Python abgeschlossen ist, wird es an der Zeit sein, unsere *virtuelle Umgebung* einzurichten, eine Art

„Container“, der vom Rest unseres Systems abgeschottet ist und in dem wir die von uns verwendeten Bibliotheken installieren werden. Der Grund, warum eine virtuelle Umgebung einer globalen Installation von Bibliotheken vorzuziehen ist, hängt vor allem mit der raschen Entwicklung in der Python-Welt zusammen. Sehr oft gibt es erhebliche Unterschiede zwischen den Releases des Interpreters, was dazu führt, dass Bibliotheken (und damit Programme) inkompatibel werden. Eine *deterministische* Umgebung zu haben, von der wir die Versionen jeder einzelnen installierten Bibliothek im Detail kennen, stellt eine Art „Garantie“ für das Funktionieren unserer Programme dar: Es reicht tatsächlich aus, die Konfiguration der virtuellen Umgebung exakt nachzubilden.
Zur Verwaltung unserer virtuellen Umgebungen verwenden wir ein Paket namens `virtualenvwwrapper`, das von der Shell mit dem Paketmanager pip installiert wird:

```
$ pip install virtualenvwrapper
```

Sobald die Installation abgeschlossen ist, erstellen wir eine neue virtuelle Umgebung:

```
$ mkvirtualenv ml-python
```

`ml-python` ist dabei der Name der im Beispielszenario gewählten virtuellen Umgebung. Natürlich kann der Name auch anders sein und Sie können den Namen wählen, der Ihnen am besten gefällt.
Die virtuelle Umgebung wird nun aktiviert:

```
$ workon ml-python
```

Wir sind nun zur Installation der für den Rest des Artikels notwendigen Bibliotheken bereit.

Die Bibliotheken

Die Bibliotheken, die wir beschreiben und verwenden werden, gehören zu den fünf am häufigsten für die **Datenanalyse in Python** eingesetzten.
Die erste und vielleicht berühmteste ist *Numpy*, eine Art *Portierung* von MATLAB in Python. Numpy ist eine Bibliothek vor allem für algebraische und Matrix-Berechnungen. Wer gewöhnlich MATLAB verwendet, dürfte viele Ähnlichkeiten finden, sowohl in Bezug auf die Syntax als auch auf die Optimierung. Die Verwendung algebraischer Berechnungen in Numpy ist effizienter als verschachtelte Zyklen, gerade mit MATLAB (in dem Tutorial [4] erfahren Sie mehr darüber). Der Datentyp, der einer Numpy-Berechnung zugrunde liegt, ist das *Array*, nicht zu verwechseln mit dem bei Computerberechnungen üblichen *Vektor*, sondern im algebraischen und geometrischen Sinne als *Matrix* zu verstehen. Da die Datenanalyse auf algebraischen und Matrix-Operationen basiert, ist Numpy auch die „Basis“ für zwei der am häufigsten verwendeten Frameworks in diesem Bereich, nämlich *Scikit-Learn* (dazu gleich mehr) und *TensorFlow*.
Eine natürliche Ergänzung zu Numpy ist *Pandas*, eine Bibliothek, die zur Verwaltung und zum Lesen von Daten aus verschiedenartigen Quellen wie Excel-Tabellen, CSV-Dateien oder sogar JSON- und SQL-Datenbanken geeignet ist. Pandas ist extrem flexibel und leistungsstark und ermöglicht es Ihnen, Daten in Strukturen, so genannten *Dataframes*, zu organisieren, die nach Belieben manipuliert und problemlos direkt in Numpy-Arrays übernommen werden können.
Die dritte Bibliothek, die wir verwenden werden, ist *Scikit-Learn*. Scikit-Learn, das aus einem akademischen Projekt hervorgegangen ist, ist ein Framework, das die meisten der heute verwendeten Algorithmen des maschinellen Lernens implementiert und eine *gemeinsame* Schnittstelle bietet. Dieses Konzept ist genau das der objektorientierten Programmierung: Es ist möglich, durch die Methode *fit_transform* praktisch jeden von Scikit-Learn angebotenen Algorithmus zu verwenden, an den mindestens zwei Parameter übergeben werden, das bedeutet, die zu analysierenden Daten und die mit ihnen verbundenen Labels.
Die letzten beiden verwendeten Bibliotheken sind *Matplotlib* und *Jupyter*. Die erste ist zusammen mit dem Addon *Seaborn* notwendig, um die Ergebnisse unserer Experimente in grafischer Form zu visualisieren; die zweite bietet uns die Möglichkeit, so genannte *Notebooks* zu verwenden, das heißt, interaktive Umgebungen zur einfachen und unmittelbaren Verwendung, die dem Datenanalytiker die Möglichkeit geben, Teile des Codes unabhängig von anderen zu schreiben und auszuführen.
Bevor wir jedoch fortfahren, wollen wir einige theoretische Konzepte vorstellen, die uns helfen, eine gemeinsame Basis für den Diskurs zu schaffen.

Bild 1. Der Startbildschirm für die Verwaltung von Notebooks in Jupyter.

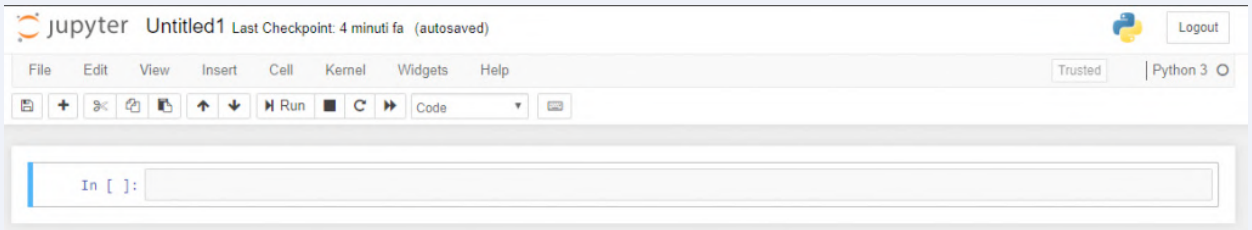


Bild 2. Ein leeres Notebook.

Die Konzepte

Das erste Konzept, das man kennen muss, ist das oft als selbstverständlich angesehene Konzept des *Datasets*, eines Satzes von *Samples*, von denen jedes durch eine bestimmte Anzahl von Variablen oder Merkmalen gekennzeichnet ist, die das beobachtete Phänomen beschreiben. Der Einfachheit halber können wir uns einen Datensatz wie ein Excel-Sheet vorstellen: In den Zeilen haben wir die *Samples*, also die einzelnen Beobachtungen des Phänomens, während in den Spalten die *Features*, die Werte stehen, die jeden der Aspekte des Prozesses charakterisieren. Um auf das Beispiel der intelligenten Fertigung zurückzukommen: Die Zeilen repräsentieren die Bedingungen der Produktionskette zu einem bestimmten Zeitpunkt, während in den Spalten die Messwerte der Sensoren dargestellt werden.

Bei Scikit-Learn spielt das Konzept der *Labels* oder der *Klasse* eine Rolle. Das Vorhandensein oder Fehlen von Labels ermöglicht es, zwischen überwachten und nicht überwachten Algorithmen zu unterscheiden. Der Unterschied ist, zumindest im Prinzip, ganz einfach: *Überwachte Algorithmen* benötigen von vornherein Kenntnisse über die Klasse jedes Samples im Datensatz, während bei *unüberwachten Algorithmen* dies offensichtlich nicht erforderlich ist. In der Praxis ist es für die Verwendung eines überwachten Algorithmus notwendig, dass ein „Experte" die Zugehörigkeitsklasse jedes Samples kennt. Bei einem intelligenten Fertigungsprozess könnte dieser Experte feststellen, ob der Datensatz der Messwerte zu einem bestimmten Augenblick eine anomale Situation darstellt oder nicht, und somit die einzelnen Samples eine von zwei möglichen Klassen zuordnen. Für unüberwachte Algorithmen ist dies natürlich nicht notwendig. Weiterhin muss unterschieden werden zwischen Prozessen mit *unabhängigen* und *identisch verteilten* Daten (IID) und mit Daten in einer *zeitlichen Reihenfolge*. In diesem Fall hängt der Unterschied mit der Art des beobachteten Phänomens zusammen: Die Samples eines IID-Prozesses sind voneinander unabhängig, während in der Zeitreihe jedes Sample von einer linearen oder nichtlinearen Kombination der Werte abhängt, die der Prozess zu vorhergehenden Zeitpunkten angenommen hat.

Lassen Sie uns zur Sache kommen!

Nachdem wir die notwendigen theoretischen und praktischen Begriffe kennengelernt haben, gehen wir zur Praxis über. Wir verwenden einen geeigneten Datensatz zur Beschreibung unseres Beispiels. Dieser Datensatz ist *SECOM*, ein Akronym, das für *SEmiCOnductor Manufacturing* steht, ein Datensatz, der die Werte enthält, die von einer Reihe von Sensoren während der Überwachung eines *Halbleiterherstellungsprozesses* gelesen werden. In dem Datensatz, der von verschiedenen Quellen (wie Kaggle) heruntergeladen werden kann, sind 590 Variablen enthalten, von denen jede repräsentativ für den Messwert eines Sensors zu einem bestimmten Zeitpunkt ist. Der Datensatz enthält auch Labels, die die Fehler und Anomalien vom korrekten Funktionieren des Systems unterscheiden.

Sobald der Datensatz heruntergeladen ist, installieren wir die oben genannten Bibliotheken. Geben Sie dazu in die Befehlszeile ein:

```
$ pip install numpy pandas scikit-learn matplotlib seaborn jupyter
```

Schauen wir uns an, wie man eine einfache Pipeline für die Datenanalyse einrichtet.

Das erste Notebook

Der erste Schritt besteht darin, ein neues Notebook einzurichten. Von der Kommandozeile aus starten wir Jupyter mit der folgenden Anweisung:

```
$ jupyter-notebook
```

Es öffnet sich ein Bildschirm wie in **Bild 1**.

Wir erstellen ein Notebook mit *New* > *Python 3*. In unserem Browser wird ein Tab mit dem neu erstellten Notebook geöffnet. Machen wir uns mit der Benutzeroberfläche in **Bild 2** vertraut, die (sehr vage) einer interaktiven Befehlszeile ähnelt, mit einem Kopfmenü und mehreren Optionen.

Das erste, was auffällt, ist die so genannte *Zelle* als eines der Teile, in die das Notebook unterteilt ist. Die Ausführung einer einzelnen Zelle wird von der Schaltfläche *Run* initiiert und ist recht unabhängig von der Ausführung anderer (beachten Sie, dass das Konzept des variablen Bereichs immer gilt).

Mit den drei Schaltflächen rechts neben *Run* können Sie den *Kernel* stoppen, neu starten und zurücksetzen. Dies ist die Instanz, die Jupyter unserem Notebook zuordnet. Ein Neustart der Instanz kann insbesondere dann erforderlich sein, wenn man die mit dem Skript verbundenen lokalen und globalen Variablen zurücksetzen

	a21	a87	a88	a89	a114	a115	a116	a117	a118	a120	...	a528
count	1567.000000	1567.000000	1567.000000	1567.000000	1567.000000	1567.000000	1567.000000	1567.000000	1567.000000	1567.000000	...	1567.000000
mean	1.405054	2.401872	0.982420	1807.815021	0.945424	0.000123	747.383792	0.987130	58.625908	0.970777	...	6.395717
std	0.016737	0.037332	0.012848	53.537262	0.012133	0.001668	48.949250	0.009497	6.485174	0.008949	...	1.888698
min	1.179700	2.242500	0.774900	1627.471400	0.853400	0.000000	544.025400	0.890000	52.806800	0.841100	...	2.170000
25%	1.396500	2.376850	0.975800	1777.470300	0.938600	0.000000	721.023000	0.989500	57.978300	0.964800	...	4.895450
50%	1.406000	2.403900	0.987400	1809.249200	0.946400	0.000000	750.861400	0.990500	58.549100	0.969400	...	6.410800
75%	1.415000	2.428600	0.989700	1841.873000	0.952300	0.000000	776.781850	0.990900	59.133900	0.978300	...	7.594250
max	1.453400	2.555500	0.993500	2105.182300	0.976300	0.041400	924.531800	0.992400	311.734400	0.982700	...	14.447900

Bild 3. Die ersten fünf Zeilen des SECOM-Datensatzes.

	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	...	a582	a583	a584	a585	a586	a587	a588	a58
0	3030.93	2564	2187.7333	1411.1265	1.3602	100	97.6133	0.1242	1.5005	0.0162	...	?	0.5005	0.0118	0.0035	2.363	?	?	
1	3095.78	2465.14	2230.4222	1463.6606	0.8294	100	102.3433	0.1247	1.4966	-0.0005	...	208.2045	0.5019	0.0223	0.0055	4.4447	0.0096	0.0201	0.00
2	2932.61	2559.94	2186.4111	1698.0172	1.5102	100	95.4878	0.1241	1.4436	0.0041	...	82.8602	0.4958	0.0157	0.0039	3.1745	0.0584	0.0484	0.014
3	2988.72	2479.9	2199.0333	909.7926	1.3204	100	104.2367	0.1217	1.4882	-0.0124	...	73.8432	0.499	0.0103	0.0025	2.0544	0.0202	0.0149	0.004
4	3032.24	2502.87	2233.3667	1326.52	1.5334	100	100.3967	0.1235	1.5031	-0.0031	...	?	0.48	0.4766	0.1045	99.3032	0.0202	0.0149	0.004

5 rows × 591 columns

Bild 4. Kurze statistische Beschreibung des SECOM-Datensatzes.

möchte. Dies ist äußerst nützlich, wenn Sie mit neuen Methoden und Bibliotheken experimentieren.

Eine weitere interessante Option ist die Auswahl des Zellentyps, bei der Sie zwischen *Code* (Python-Code), *Markdown* (nützlich, um Kommentare und Beschreibungen in dem Format einzufügen, das zum Beispiel für GitHub READMEs verwendet wird), *Raw NBContent* (reinen Text einzufügen) und *Heading* (zum Einfügen von Titeln) wählen können.

Daten importieren und anzeigen

Sobald wir mit der Schnittstelle vertraut sind, können wir mit der Implementierung unseres Skripts fortfahren. Wir importieren zunächst die verwendeten Bibliotheken und Module:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from ipywidgets import interact
from sklearn.ensemble import RandomForestClassifier
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix,
accuracy_score
from sklearn.utils import resample
```

Dabei ist die Anweisung `%matplotlib inline` hervorzuheben, die nützlich ist, um die von der Matplotlib erzeugten Diagramme korrekt anzuzeigen.

Dann lesen wir die Daten aus dem SECOM-Datensatz mit der Funktion `read_csv` von Pandas. Eine kurze Anmerkung: In diesem Beispiel ist der relative Pfad zu der Datei der Einfachheit halber fest angegeben, es ist aber in der Praxis ratsam, das Python-Paket *os* zu verwenden, damit unser Programm diesen Pfad immer frisch ermitteln kann.

```
data = pd.read_csv('data/secom.csv')
```

Mit dieser Anweisung lesen wir die in der Datei *secom.csv* enthaltenen Daten und ordnen sie in ein Dataframe namens `data` an. In **Bild 3** sind die ersten fünf Zeilen des Datenrahmens (plus `head()`-Anweisung) dargestellt: `data.head()`

Die Visualisierung der ersten Zeilen des Datenrahmens kann nützlich sein, um einen ersten Überblick über die zu analysierenden Daten zu erhalten: Hier bemerken wir zum Beispiel sofort, dass einige Werte gleich „?“ (vermutliche Null-Werte) vorhanden sind. Darüber hinaus ist es offensichtlich, dass sich die Werte in recht verschiedenen Größenordnungen bewegen, ein Faktor, den wir im Auge behalten müssen. Wir können auch die Funktion `describe()` verwenden, um einen schnellen Überblick über die statistischen Merkmale der einzelnen Variablen zu erhalten:

```
data.describe()
```

Die statistische Analyse kann prinzipiell eine Nicht-Normalität (Daten mit nicht-parametrischer Verteilung) oder das Vorhandensein von Anomalien hervorheben. Um ein Beispiel zu nennen: Wir stellen fest, dass die Standardabweichung der Variablen *a116* und *a118* verhältnismäßig hoch ist, so dass man erwarten kann, dass diese Daten eine große Bedeutung in der Analyse haben, andererseits haben Variablen wie *a114* eine so geringe Varianz, so dass man sie als nicht besonders beachtenswert verwerfen kann.

Sobald das Thema Laden und Anzeigen von Dataframes abgeschlossen ist, können wir zu einem grundlegenden Teil der Pipeline übergehen: dem *Preprocessing*.

PData Preprocessing

In einem ersten Schritt zeigen wir die Anzahl der mit jeder Klasse verknüpften Samples an. Dazu verwenden wir die Funktion `value_counts()` in der Spalte `classvalue`, die die den einzelnen Samples zugeordneten Bezeichnungen enthält.

```
data['classvalue'].value_counts()
```

Wir können sehen, dass 1463 Samples in einer „normalen“ Betriebssituation (Klasse -1) und 104 Samples in einer Fehlersituation (Klasse 1) gesammelt wurden. Der Datensatz ist daher stark unausgewogen, und es sollten Vorkehrungen getroffen werden, um die Verteilung der Samples auf die verschiedenen Klassen „einheitlicher“ zu gestalten. Dies hängt mit der intrinsischen Funktionsweise von Algorithmen des maschinellen Lernens zusammen, die auf Grundlage der ihnen zur Verfügung stehenden Daten lernen. In diesem speziellen Fall lernt der Algorithmus, eine Situation als Standardverhalten zu charakterisieren, ist sich aber bei der

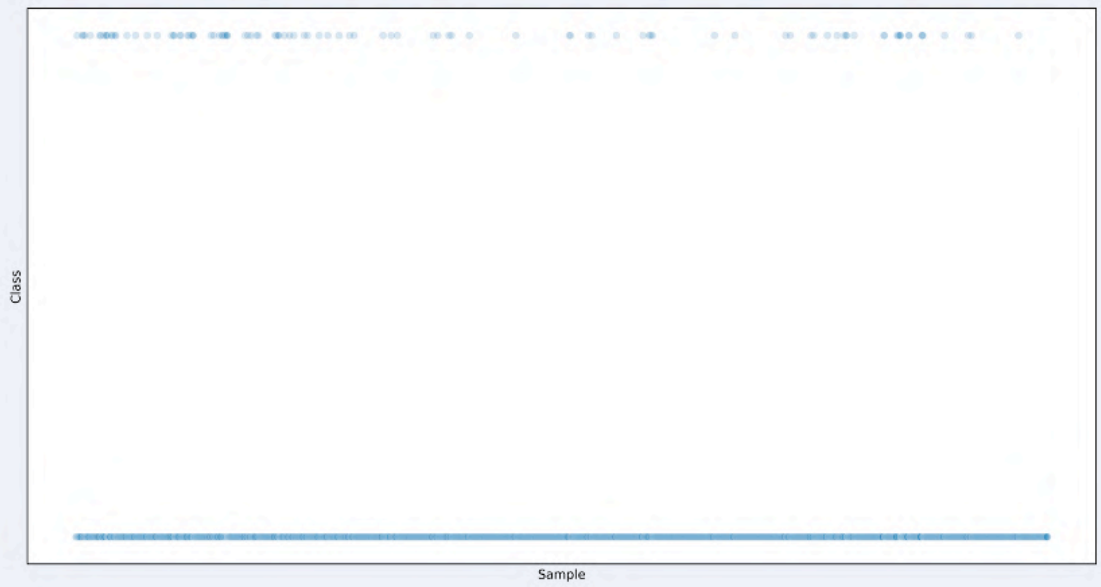


Bild 5. Anzahl der Samples pro Klasse im SECOM-Datensatz.

Charakterisierung anomaler Situationen unsicher. Die Unausgewogenheit wird noch deutlicher, wenn man ein Streudiagramm verwendet (siehe **Bild 5**):

```
sns.scatterplot(data.index, data['classvalue'],
alpha=0.2)
plt.show()
```

Behalten wir dieses Ungleichgewicht im Hinterkopf (wir werden später darauf zurückkommen) und „trennen" nun die Labels von den Daten:

```
labels = data['classvalue']
data.drop('classvalue', axis=1, inplace=True)
```

Beachten Sie die Verwendung des *Achsenparameters* `axis` in der `Drop`-Funktion, mit dem man festlegen kann, dass die Funktion mit den Spalten des Datenrahmens arbeiten muss (standardmäßig arbeiten Pandas-Funktionen mit Zeilen).

Eine weitere Information, die aus der Datensatzanalyse gezogen werden kann, ist, dass in dieser speziellen Version der SECOM-Datei viele Spalten unterschiedliche Datentypen enthalten (Strings und numerische Werte). Deshalb ist Pandas nicht in der Lage, den Datentyp eindeutig zu bestimmen und überlässt die Definition dieser Daten dem Benutzer („?"). Um alle Daten in ein numerisches Format zu bringen, werden deshalb drei Pandas-Funktionen genutzt.

Die erste verwendete Funktion ist `replace()`, mit der wir alle Fragezeichen durch den konstanten Wert *numpy.nan* ersetzen können, ein Platzhalter, der zur Behandlung von Nullwerten in Numpy-Arrays verwendet wird.

```
data = data.replace('?', np.nan, regex=False)
```

Die Syntax der Funktion ist selbsterklärend: Der erste Parameter ist der zu ersetzende Wert, der zweite ist der ersetzende Wert und der dritte ist ein Flag, das angibt, ob der erste Parameter ein regulärer Ausdruck ist oder nicht. Wir könnten auch eine alternative Syntax verwenden, indem wir den `inplace`-Parameter auf `True` setzen:

```
data.replace('?', np.nan, regex=False, inplace=True)
```

Die beiden anderen Funktionen zur Lösung der oben genannten Probleme sind `apply()` beziehungsweise `to_numeric()`. Mit der ersten können Sie eine bestimmte Funktion auf alle Spalten (oder Zeilen) eines Datenrahmens anzuwenden, während die zweite eine einzelne Spalte in numerische Werte umwandelt. Durch ihre Kombination erhalten wir ein eindeutiges Datenformat und entfernen alle für Numpy und Scikit-Learn nicht nachvollziehbaren Werte.

```
data.apply(pd.to_numeric)
```

Lassen Sie uns nun bewerten, welche der im Datensatz enthaltenen Funktionen nützlich sind. Normalerweise werden in der Praxis (mehr oder weniger komplexe) *Feature-Selection*-Techniken verwendet, um Redundanzen und den Umfang des zu behandelnden Problems zu reduzieren. Wegen der offensichtlichen Vorteile in Bezug auf Verarbeitungszeit und die Leistung des Algorithmus werden wir in unserem Fall eine weniger komplexe Technik benutzen, die auf die Eliminierung von Merkmalen mit geringer (und daher, wie oben erwähnt, nicht sehr signifikanter) Varianz baut. Wir erstellen daher ein interaktives Steuerelement, das es uns ermöglicht, die Verteilung der Daten für jedes Merkmal in Form eines Histogramms anzuzeigen:

```
@interact(col=(0, len(df.columns) - 1)
def show_hist(col=1):
    data['a' + str(col)].value_counts().hist(grid=-
False, figsize=(8, 6))
```

Die Interaktivität wird durch den *Decorator* `@interact` gewährleistet, dessen Referenzwert (`col`) zwischen 0 und der Anzahl der im Datensatz vorhandenen Merkmale variiert. Wenn wir die angezeigten Daten über das Widget untersuchen, werden wir feststellen, wie viele Merkmale nur einen einzigen Wert annehmen und damit bei der Analyse keine Rolle spielen. Wir können sie deshalb eliminieren:

```
single_val_cols = data.columns[len(data)/data.nunique()
< 2]
secom = data.drop(single_val_cols, axis=1)
```

Natürlich gibt es relevantere und raffiniertere Techniken zur Feature Selection, die beispielsweise auf statistischen Parametern basieren. Einen vollständigen Überblick bietet die Dokumentation bei Scikit-Learn [5].

Der letzte Schritt besteht darin, sich mit Nullwerten zu befassen (die wir zuvor durch *np.nan* ersetzt haben). Wir untersuchen Datensatz, um zu sehen, wie viele es davon gibt. Dazu verwenden wir eine Heatmap wie in **Bild 6,** wobei die hellen Stellen die Nullwerte darstellen.

```
sns.heatmap(secom.isnull(), cbar=False)
```

Es ist offensichtlich, dass viele Samples einen hohen Prozentsatz von Nullwerten aufweisen, so das sie nicht berücksichtigt werden sollten, um einen *Bias*- oder Verzerrungseffekt zu vermeiden.

```
na_cols = [col for col in secom.columns if secom[col].
isnull().sum() / len(secom) > 0.4]
secom_clean = secom.drop(na_cols, axis=1)
secom_clean.head()
```

Dank der obigen Anweisungen erstellen wir eine „list comprehension", um alle Features mit mehr als 40% Nullwerten zu isolieren und sie später aus dem Datensatz zu entfernen.

Nur die Features mit weniger als 40% Nullwerten müssen noch behandelt werden. Dazu verwenden wir unser erstes Scikit-Learn-Objekt `SimpleImputer`, das allen NaNs auf der Grundlage einer benutzerdefinierten Strategie Werte zuweist. Hier kommt die *Mittelwertstrategie* (`mean`) zum Einsatz. Jedem NaN wird der vom Feature angenommene Mittelwert zugeordnet.

```
imputer = SimpleImputer(strategy='mean')
secom_imputed = pd.DataFrame(imputer.
fit_transform(secom_clean))
secom_imputed.columns = secom_clean.columns
```

Zur Übung können wir eine weitere Heatmap erstellen, die – vorhersehbar - eine einheitlich dunkle Farbe annehmen wird, um zu bestätigen, dass es keinen Nullwert im Datensatz mehr gibt. Damit kommen wir zur eigentlichen Datenverarbeitung.

Datenverarbeitung

Wir teilen unseren Datensatz in die beiden Untergruppen *Training* und *Test* auf. Diese Unterteilung ist notwendig, um das Phänomen der Überanpassung abzuschwächen, das den Algorithmus „zu sehr an den Daten haften" lässt (weitere Informationen unter [6]). Dadurch wird sichergestellt, dass das gefundene Modell auch auf andere Fälle als den, an dem es trainiert wurde, anwendbar ist. Dazu verwenden wir die Funktion `train_test_split`:

```
X_train, X_test, y_train, y_test = train_test_split(se-
com_imputed, labels, test_size=0.3)
```

Mit dem Parameter `test_size` können wir den Prozentsatz der für den Test reservierten Daten angeben. Die Standardwerte für diesen Parameter liegen normalerweise zwischen 0,2 und 0,3.

Eine weitere wichtige Aufgabe ist die *Normalisierung* der Daten. Wir haben bereits bemerkt, dass bei einigen Features Werte in viel größeren Bereichen annehmen als andere, was ihnen auch mehr Gewicht in der Analyse verleihen würde. Eine Normalisierung erlaubt es, alle Features mit einem einheitlichen Wertebereich zu versehen, so dass es keine Ungleichgewichte aufgrund von anfänglichen Offsets gibt. Dazu verwenden wir einen `StandardScaler`:

```
scaler = StandardScaler()
X_train = pd.DataFrame(scaler.fit_transform(X_train),
index=X_train.index, columns=X_train.columns)
X_test = pd.DataFrame(scaler.fit_transform(X_test),
index=X_test.index, columns=X_test.columns)
```

Hier macht sich die von Scikit-Learn angebotene gemeinsame Schnittstelle nützlich: Sowohl `Scaler` als auch `Imputer` verwenden die Methode `fit_transform`, um Daten zu verarbeiten, was in komplexen Pipelines das Schreiben von Code und das Verständnis der Bibliothek stark vereinfacht.

Wir sind nun endlich bereit, die Daten zu klassifizieren. Wir verwenden einen *Random Forest* [7], um nach dem Training ein Modell zu erhalten, das in der Lage ist, zwischen normalen und anormalen Situationen zu unterscheiden. Wir überprüfen die Leistungsfähigkeit des ausgearbeiteten Modells auf zwei Arten, zum einen durch den *Accuracy Score*, dem Prozentsatz der Samples im Testsatz, die durch den Algorithmus korrekt klassifiziert wurden, zum anderen durch die Wahrheits- oder *Konfusionsmatrix* [8], die es erlaubt, die Anzahl der falsch positiven und falsch negativen Ergebnisse zu unterscheiden.

Zuerst erstellen wir den Klassifikator:

```
clf = RandomForestClassifier(n_estimators=500,
max_depth=4)
```

Wir verwenden einen Random Forest mit 500 *Estimators* (die Anzahl der Bäume im Wald) mit einer maximalen Tiefe von vier Ebenen.

Jetzt können wir unser Modell anhand der Trainingsdaten trainieren:

```
clf.fit(X_train, y_train)
```

Sobald das Training abgeschlossen ist, lässt sich das trainierte Modell zur Klassifizierung der Testsamples verwenden:

```
y_pred = clf.predict(X_test)
```

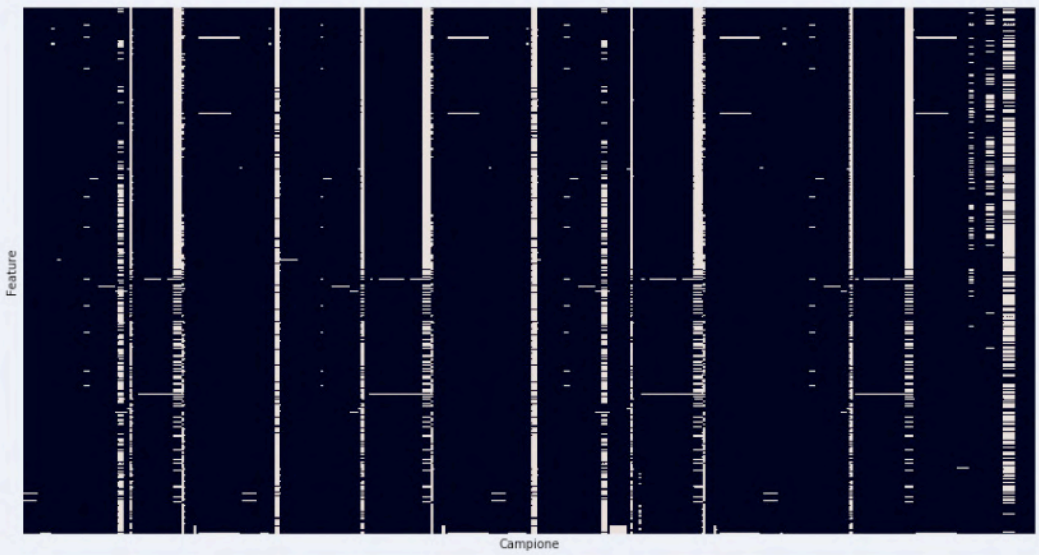


Bild 6. Anzeige der Nullwerte im SECOM-Datensatz.

Wir verfügen dann für jedes Sample der Testmenge über zwei Labels: Das erste (zu `y_test` gehörende) Label repräsentiert die „Wahrheit", während das zweite (zu `y_pred` gehörende) Label den vom Algorithmus vorhergesagten Wert darstellt. Wenn wir sie vergleichen, erhalten wir sowohl die `Accuracy` als auch die `Confusion_matrix`.

```
accuracy = accuracy_score(y_test, y_pred))
cf = confusion_matrix(y_test, y_pred))
```

Im Beispieldurchlauf haben wir die folgenden Ergebnisse erhalten:

```
Die Modellgenauigkeit des Testsatzes beträgt:
0.9341825902335457
Die Konfusionsmatrix des Modells ist:
[[440 0]
[ 31 0]]
```

Die Genauigkeit von etwa 93% ist sehr gut, so dass das Modell in Ordnung zu sein scheint. Wir stellen jedoch fest, dass das Modell zwar bei der Klassifizierung der Samples der tonangebenden Klasse sehr genau ist, aber ebenso ungenau bei der Klassifizierung der Samples, die der „Minderheitenklasse" angehören. Es ist offensichtlich eine Verzerrung vorhanden.

Wir sollten und können eine Strategie finden, um diese Situation zu verbessern. Dazu setzen wir ein *Upsampling* der Daten fort, die der Minderheitenklasse angehören, um den Datensatz zumindest teilweise auszugleichen. Dazu eignet sich die `Resample`-Funktion von Pandas.

```
normals = data[data['classvalue'] == -1]
anomalies = data[data['classvalue'] == 1]
anomalies_upsampled = resample(anomalies, repla-
ce=True, n_samples=len(normals))
```

Wir erhöhen den Umfang des Datensatzes, um die Anzahl der normalen Samples an die Anzahl der anomalen Samples anzupassen. Daher müssen sowohl X als auch Y neu definiert werden:

```
upsampled = pd.concat([normals, anomalies_upsampled])
X_upsampled = upsampled.drop('classvalue', axis=1)
y_upsampled = upsampled['classvalue']
```

Durch erneutes Durchführen des Trainings (und natürlich durch Wiederholung der Split- und der Normalisierungsprozedur) erhalten wir in unserem Beispieldurchlauf die folgenden Ergebnisse:

```
Die Modellgenauigkeit des Testsatzes beträgt:
0,8631921824104235
Die Konfusionsmatrix des Modells ist:
[[276 41]
[ 43 254]]
```

Wir stellen sofort fest, dass die Genauigkeit des Modells abgenommen hat, was vermutlich auf die nun größere Heterogenität des Datensatzes zurückzuführen ist. Doch wenn man die Konfusionsmatrix betrachtet, fällt auf, dass das Modell seine Verallgemeinerungsfähigkeiten tatsächlich verbessert hat und es gelingt, auch Samples, die zu anomalen Situationen gehören, korrekt zu klassifizieren.

Schlussfolgerungen und Hinweise

In diesem Artikel haben wir eine Pipeline für die Analyse von Daten aus realen Prozessen in Python kennengelernt. Es ist jedoch klar, dass jedes der behandelten Themen in der Tat äußerst vielfältig ist und theoretische wie praktische Kenntnisse unerlässlich sind, wenn man sich ernsthaft mit der Datenanalyse beschäftigen möchte. Wir haben auch gelernt, dass man nicht beim ersten erzielten Ergebnis stehen bleiben darf, auch nicht in so komplexen Situationen wie der behandelten: Es ist vielmehr notwendig, die erzielten Ergebnisse aus verschiedenen Blickwinkeln zu interpretieren, um den Unterschied zwischen einem arbeitsfähigen und einem mehr oder weniger offensichtlich von Verzerrungen betroffenen Modell zu entdecken. Die Botschaft, die es mit nach Hause zu nehmen gilt, lautet daher: Datenanalyse ist keine mechanische Disziplin, sondern setzt eine kritische, eingehende und vielfältige Analyse des beobachteten Phänomens voraus, die sich an theoretischen Vorstellungen und praktischen Fertigkeiten orientiert. Abschließend finden Sie in den Weblinks einige Hinweise, durch die Sie einige der im Artikel angesprochenen Aspekte vertiefen können, sowie den Link zum GitLab-Repository, wo Sie den für den Artikel geschriebenen Code einsehen können.

200505-02

Wir haben diesen Beitrag der Zeitschrift/Online-Plattform *Elettronica Open Source* entnommen und mit freundlicher Genehmigung des Verlags übersetzt (https://it.emcelettronica.com).

WEBLINKS

[1] **GPU-Computing auf Mathlab**: https://de.mathworks.com/solutions/gpu-computing.html

[2] **Python-Tutorials**: https://www.python-kurs.eu/index.php

[3] **Python im Web**: https://www.python.org/

[4] **Optimierung von Matlab-Code**: https://de.mathworks.com/videos/optimizing-and-accelerating-matlab-code-1504278738913.html

[5] **Scikit-Learn-Benutzerhandbuch**: https://scikit-learn.org/stable/user_guide.html

[6] **Overfitting**: https://medium.com/@stefan.preusler/overfitting-und-underfitting-auf-deutsch-mit-python-erklärt-b028e2bea2d3

[7] **Understanding Random Forest**: https://towardsdatascience.com/understanding-random-forest-58381e0602d2

[8] **Understanding Confusion Matrix**: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

[9] **Code zum Artikel bei GitLab**: https://gitlab.com/eos-acard/machine-learning-in-python

Elektor hilft, Ihr Geschäft zu optimieren

Mit fast sechzig Jahren hat Elektor so schon einiges erlebt aber das Jahr 2020 wird wohl bleibenden Eindruck hinterlassen. Wir haben gesehen, wie große, namhafte Kunden ihre Marketingaktivitäten gekürzt haben, aber wir haben auch Unternehmen gesehen, die stark gewachsen sind wie nie zuvor. Obgleich es einer gewissen Flexibilität bedurfte, waren wir in der Lage, beiden zu helfen.

Wir alle brauchen ein gesundes Geschäft, ob das nun bedeutet, die Dinge etwas zu straffen oder sie auf die nächste Stufe zu heben. Nichtstun ist keine Option.

Mein Team ist bereit, in beiden Szenarien zu helfen. Sei es mit kompakten, aber hocheffizienten Content-Marketing-Strategien oder einer kompletten Multichannel-Kampagne, um die Vorteile dieser neuen Marktöffnung zu nutzen - Elektor Helps. In diesen Zeiten ist eine maßgeschneiderte Herangehensweise für unsere Kunden erforderlich, und wir gehen noch einen Schritt weiter. Wir sind davon überzeugt, dass wir gemeinsam mit Ihnen den Unterschied bewirken können.

Ja, das ist eine Herausforderung!"

Margriet Debeij
International Client Manager

Kontaktieren Sie uns auf www.elektor.com/helps-clients

Elektor Industry Edition

Innerhalb der Extra-Editionen von Elektor Industry, die für alle kostenlos zum Download zur Verfügung stehen, bieten wir unseren Kunden eine Reihe von Standard-Elektor-Helps-Paketen an. Da aber jeder Kunde anders ist, bitten wir jeden, der nach einer passenden Lösung sucht, sich mit unserem Kundenteam in Verbindung zu setzen. Wir sind sicher, dass unser Team unter der Leitung von Margriet Debeij immer einen Weg finden wird, um Ihnen zu helfen. Besuchen Sie elektor.com/help-clients, um mehr zu erfahren.

elektor
design › share › sell

Parallax Propeller 2

Teil 1: Kurz vorgestellt

Von **Mathias Claußen** (Elektor)

Parallax hat die neue Version 2 seines neuen Propeller-Chips mit acht Kernen, einem halben Megabyte RAM und Hochgeschwindigkeits-I/Os mit bis zu 300 MHz in Silizium gegossen. Wir haben ein Eval-Board erhalten, also lassen Sie uns einen Blick auf den Propeller 2, seine Eigenschaften und seine Fähigkeiten werfen.

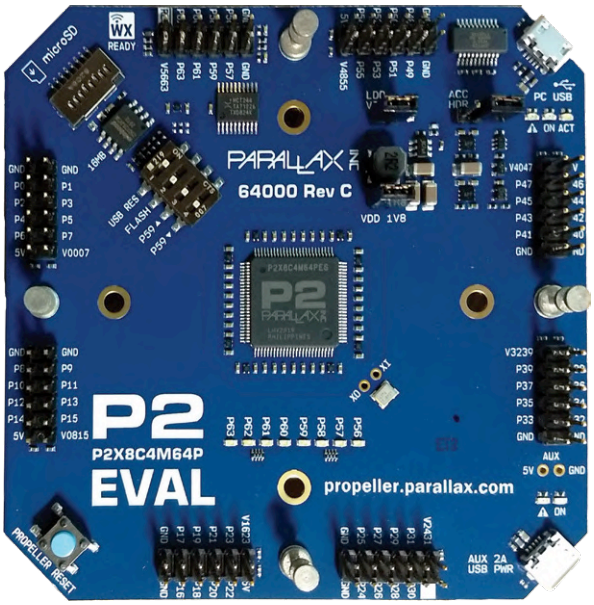


Bild 1. Parallax Propeller 2 Evaluation Board Rev. C.

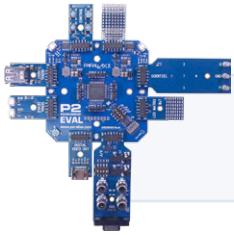
Als Parallax 2006 seinen ersten Propeller-Chip vorstellte, war das etwas völlig anderes als alles, was wir zuvor gesehen hatten. Seit einigen Monaten arbeitet Parallax an der neuen Version 2 des Chips, und zwar auf eine ungewöhnliche Art. Der neue Propeller wurde zunächst nicht als fertiger Silizium-Chip, sondern lediglich als Bitstream für ein DE10-Nano FPGA-Board von Altera eingeführt, mit der Bitte an User und Ingenieure um Feedback. Im Laufe der Zeit sammelte Parallax das Feedback und erhielt so wichtige Kriterien für den Bau des „echten" Propeller 2 in Silizium. Das Unternehmen berichtete zudem in einem Propeller-Forum fortwährend über den Prozess der Chip-Herstellung, so dass die Community immer auf dem neusten Stand blieb. Während ich dies hier schreibe, wird der Silizium-Chip in der Revision C für die Produktion freigegeben. Werfen wir also einen Blick auf den Propeller 2 und seine Eigenschaften!

Der Propeller 2

Der Chip ist offiziell für eine Taktfrequenz von 180 MHz ausgelegt, was zu 90 MIPS pro Kern führt, und jeder Befehl benötigt mindestens zwei Taktzyklen. Eine Übertaktung des Propeller 2 bis mehr als 300 MHz ist möglich, so dass eine Rechenleistung von 150 MIPS pro Kern erreicht werden kann. Normalerweise besitzt ein Mikrocontroller einen Kern oder zwei, beim Propeller 2 sind es dagegen acht unabhängige Kerne, die so genannten Cogs, die eine Menge Rechenleistung liefern. Alle Cogs teilen sich bei diesem Modell 512 KB RAM sowie zusätzliche 512 × 32 Bit-Register und 512 × 32 Bit Lookup-RAM pro Kern. Neben der Zahl der Kerne und dem RAM enthält der Chip eine ganze Reihe interessanter Peripherie und Funktionen:

- CORDIC-Solver mit Skalierungsfaktor-Korrektur
- 16 Semaphor-Bits mit atomarem Read-Modify-Write (RMW)
- Frei laufender 64-Bit-Zähler
- USB 2.0-FS-Host- und Slave-Interface
- Intelligente I/O-Pins (Einzelheiten im Kasten „Intelligente Pin-Funktionen").

Parallax war so freundlich, uns ein Evaluation-Kit (**Bild 1**) mit einem Controller der der Revision C zu überlassen, damit wir einen ersten Blick darauf werfen können. In dieser Artikelserie werden wir den Chip, seine Peripherie, Beispielcode und mehr genau unter die Lupe nehmen.
Eine kleine Warnung: Während ich dies schreibe, ist der Chip offiziell noch nicht freigegeben, so dass Sie damit rechnen müssen, dass wir auf einige „Unebenheiten" in der Software stoßen werden. Uns wurde nicht nur das blanke Evaluation-Board zur Verfügung



Der Propeller 2 hat acht unabhängige Kerne, so genannte Cogs, die viel Rechenleistung liefern.

SMART-PIN-FUNKTIONEN

- 8-bit, 120-Ω- (3 ns) und 1-kΩ-DACs mit 16-bit-Oversampling, Rauschen und high/low-digitalen Modi
- Delta-Sigma-ADC mit fünf Bereichen, zwei Quellen und VIO/GIO-Kalibrierung
- Mehrere ADC-Samplingmodi: automatische 2n SINC2, einstellbare SINC2/SINC3, Oszilloskop
- Eingangsmodi Logik-, Schmitt-, Pin-to-Pin-Komparator und 8-bit-Pegelkomparator
- 2/3/5/8-bit eindeutige Eingangsfilterung mit skalierbarer Sample-Rate
- Einbeziehung von Eingangsinformationen relativer Pins, -3 bis +3
- Negatives oder positives lokales Feedback mit oder ohne Taktung
- Getrennte Ansteuerungsmodi für hohen und niedrigen Ausgang: logic / 1,5 k / 15 k / 150 k / 1 mA / 100 µA / 10 µA / float
- Programmierbarer 32-Bit-Taktausgang, Transition-Ausgang, NCO/Duty-Ausgang
- Dreieck/Sägezahn-/SMPS-PWM-Ausgang, 16-Bit-Frame mit 16-Bit-Prescaler
- Quadratur-Dekodierung mit 32-Bit-Zähler mit Positions- und Geschwindigkeitsmodi
- 16 verschiedene 32-Bit-Messungen mit einem oder zwei Signalen
- USB Full-Speed und Low-Speed (über ungerade/gerade Pin-Paare)
- Synchrones serielles Senden und Empfangen, 1 bis 32 Bit, bis zu Takt/2-Baudrate
- Asynchrones serielles Senden und Empfangen, 1 bis 32 Bit, bis zu Takt/3-Baudrate

gestellt, sondern auch ein Stapel von Add-Ons (siehe **Bild 2**), der eine Vielzahl von Tests und Spielereien bequem gestaltete.

Wo ist die Peripherie geblieben?

Sie fragen sich, wo die SPI-, I²C- und UART-Module geblieben sind? Diese können mit Hilfe der Smart-Pins und etwas Code innerhalb der Cogs gebildet werden, und auch wenn sie nicht direkt in der Feature Map stehen sind, sind sie verfügbar. Mit diesem Ansatz werden wir (am Ende der Artikelreihe) in der Lage sein, ein HDMI-Signal direkt auf einen Monitor auszugeben und Inhalte von einem Flash-Chip anzuzeigen. Da eine Beschreibung der einzelnen Peripherieeigenschaften „am Stück" recht langwierig und langweilig wäre, wollen wir sie im Laufe der Artikelserie bei Bedarf vornehmen. Im nächsten Teil des Artikels werden wir einen kurzen Blick auf die Entwicklungsumgebung werfen und mit unserem ersten I/O-Pin eine LED ansteuern.

200479-04

Sie haben Fragen oder Kommentare?
Haben Sie Fragen oder Anmerkungen zu diesem Artikel? Schicken Sie eine E-Mail an den Autor unter mathias.claussen@elektor.com oder kontaktieren Sie Elektor unter editor@elektor.com.

Ein Beitrag von
Autor: **Mathias Claußen**
Redaktion: **Jens Nickel und C. J. Abate**
Übersetzung: **Rolf Gerstendorf**
Gestaltung: **Giel Dols**

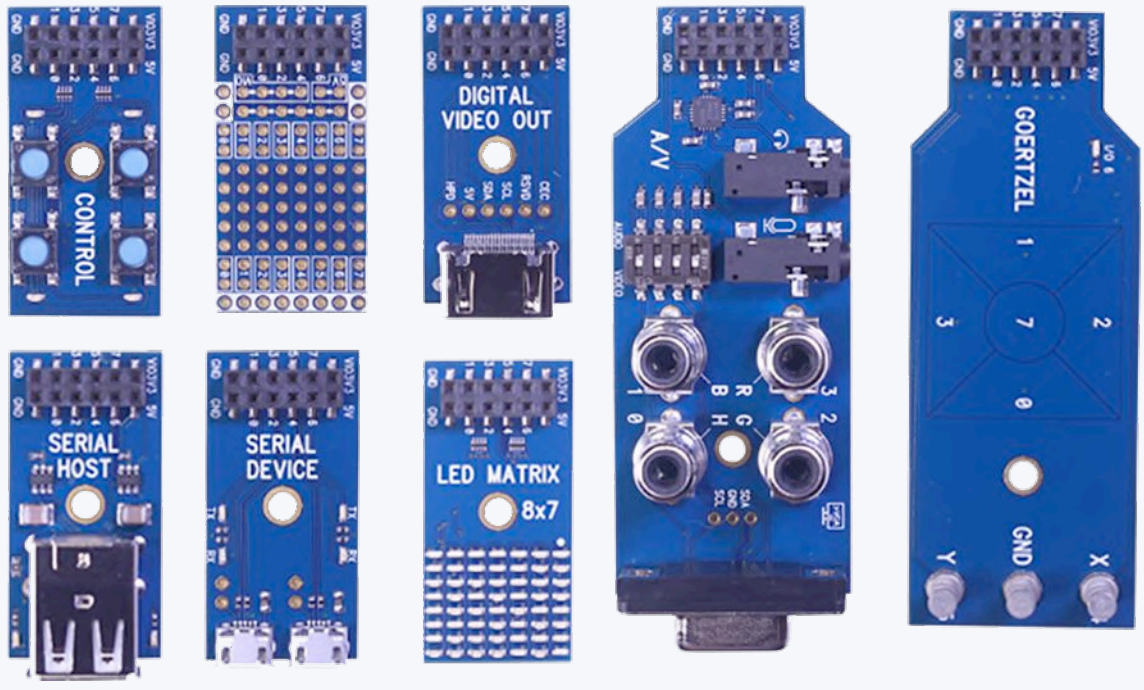


Bild 2. Add-on-Boards für das Propeller 2 Evaluation Board.

SeeedStudio DT71 Mini Digital Smart Tweezers

Preis: 79,95 €

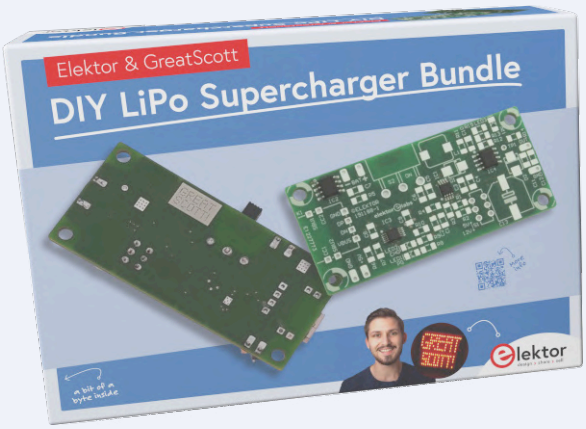
Mitgliederpreis: 71,96 €

www.elektor.de/19422

Raspberry Pi 400 – Raspberry Pi 4-basierter PC

Preis: 74,95 €

www.elektor.de/19430

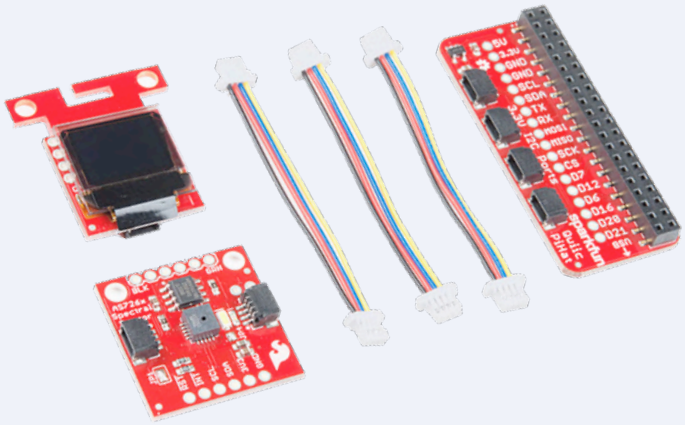


DIY LiPo Supercharger Kit (by GreatScott!)

Preis: 24,95 €

Mitgliederpreis: 22,46 €

www.elektor.de/19525



SparkFun Qwiic Starterkit für Raspberry Pi

Preis: 69,95 €

Mitgliederpreis: 62,96 €

www.elektor.de/19521

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 10. Februar 2021.

Die Gewinner des Hexadokus aus der Ausgabe November/Dezember 2020 stehen fest!

Die richtige Lösung ist: **F1235**

Einen Elektor-Wertgutschein über je 50 € haben gewonnen:

Huub Liebregts, G.D. (Joe) Young, Jean-Claude Carré, Ralf Boldt Velbert und R. Torfs.

Herzlichen Glückwunsch!

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

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

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

188 Seiten mit den besten Artikeln über Raspberry Pi im Industrie-Einsatz

Der Raspberry Pi wurde ursprünglich für die Forschung und Lehre entwickelt. Der kleine Bastler-PC hat jedoch bewiesen, dass er auch in der Industrie vielseitig einsetzbar ist. In enger Zusammenarbeit mit ELEKTRONIKPRAXIS haben wir dieses einzigartige Buch über aktuelle Lösungsansätze und Anwendungsbeispiele vom RPi in der Industrie veröffentlicht.

Raspberry Pi goes Industry

elektor

JETZT 24,90 € im Elektor Shop

Dieses Buch bietet eine anschauliche Übersicht über die Möglichkeiten der industriellen Anwendung des Raspberry Pi. Von Sensoren, Blockchain-Technologie, Industriecontroller über Automatisierung und KI, werden zahlreiche Anwendungsmöglichkeiten für den Raspberry Pi anschaulich erklärt. Außerdem werden unterschiedlichste Tools zur erweiterten Nutzung des Raspberry Pi sowie die verschiedenen Betriebssysteme vorgestellt. Die Möglichkeiten und Grenzen für den Raspberry Pi in der Industrie und konkrete Anwendungsbeispiele liefern ein realistisches Gesamtbild.

Folgende Themen werden beschrieben:
- Raspberry Pi 3 A+
- Raspberry Pi 4 B
- RPi CM3+
- 45 Betriebssysteme für RPi
- RPi und Blockchain Ethereum
- RPi im Industrieeinsatz
- und viele weitere Themen

Der Raspberry Pi wurde ursprünglich für die Forschung und Lehre entwickelt. Der kleine Bastler-PC hat jedoch bewiesen, dass er auch in der Industrie vielseitig einsetzbar ist. Aufgrund der zahlreichen Anwendungen des Raspberry Pi in privaten Hobbykellern und Hochschullabors ist er etlichen angehenden Ingenieuren bereits bekannt, daher ist es nicht verwunderlich, dass er schon in unterschiedlichsten industriellen Betrieben zum Einsatz kommt.

ELEKTRONIK PRAXIS Wissen. Impulse. Kontakte.

Bestellen Sie jetzt:
www.elektor.de/raspberry-pi-goes-industry