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

elektor MAG

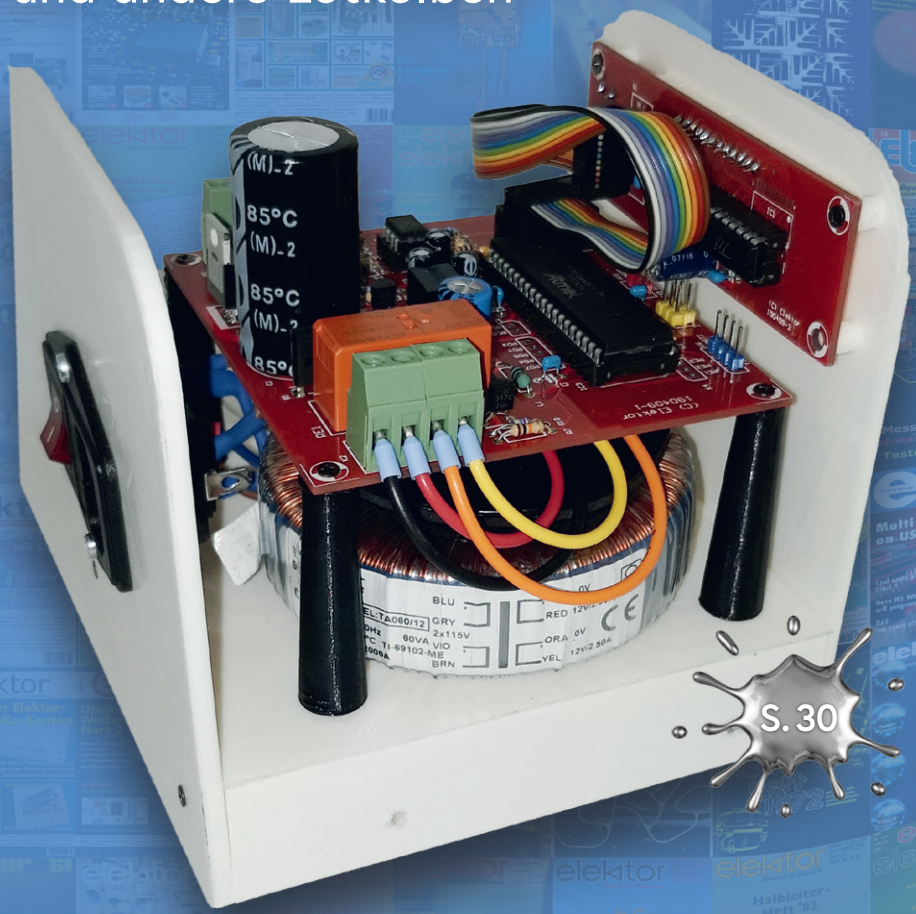
design › share › sell

579
* SEIT 1961 *

MAI/JUNI 2021
ELEKTORMAGAZINE.DE

Temperaturgesteuerte Lötstation 2021

Neuer Entwurf für Weller RT und andere Lötkolben



60 years young

In dieser Ausgabe

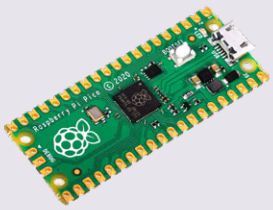
- **Elektor wird 60! Rückblick auf sechs Jahrzehnte Elektronik**
- **Java auf dem Raspberry Pi**
- **WLAN-Schalter**
- **MicroPython für Mikrocontroller**
- **Wärmebild-Kamera**
- **Einführung in die objektorientierte Programmierung**
- **Parallax Propeller 2: Entwicklungsumgebung**
- **Retronik: Micro-Professor**
- **Aller Anfang: Kondensatoren**
- **Recht auf Reparatur gewinnt an Schwung**

und vieles mehr!

S. 30

Wiederaufladbare LiPo-Akku-Stromversorgung S. 06
GreatScott! und Elektor präsentieren das LiPo-Supercharger-Kit

Zeiterfassung mit dem IoT-Button S. 48
ESP32-basiertes M5Stack und Toggl-Webservice



Raspberry Pi Pico und der RP2040 S. 57
Brücke zwischen Einplatinen-computern und Mikrocontrollern

4 198630 314904 10006

60 elektor MAG

sixty › years › young

In diesem Mai wird Elektor 60 Jahre jung. In den nächsten 12 Monaten werden wir unser Gründungserbe würdigen und der Gemeinschaft unsere große Anerkennung aussprechen! Vor allem aber sind wir stolz ein Teil der Elektronik-Revolution gewesen zu sein und das seit *1961!*

1961 1971 1981 1991 2001 2011 2021

IMPRESSUM

52. Jahrgang, Nr. 579
Mai/Juni 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
Raoul Morreau (Leitung)
Mobil: +31 6 440 399 07
E-Mail: raoul.morreau@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)

Der Herausgeber ist nicht verpflichtet, unverlangt eingesandte Manuskripte oder Geräte zurückzusenden. Auch wird für diese Gegenstände keine Haftung übernommen. Nimmt der Herausgeber einen Beitrag zur Veröffentlichung an, so erwirbt er gleichzeitig das Nachdruckrecht für alle ausländischen Ausgaben inklusive Lizenzen. Die in dieser Zeitschrift veröffentlichten Beiträge, insbesondere alle Aufsätze und Artikel sowie alle Entwürfe, Pläne, Zeichnungen einschließlich Platinen sind urheberrechtlich geschützt. Ihre auch teilweise Vervielfältigung und Verbreitung ist grundsätzlich nur mit vorheriger schriftlicher Zustimmung des Herausgebers gestattet. Die veröffentlichten Schaltungen können unter Patent- oder Gebrauchsmusterschutz stehen. Herstellen, Feilhalten, Inverkehrbringen und gewerblicher Gebrauch der Beiträge sind nur mit Zustimmung des Verlages und ggf. des Schutzrechtsinhabers zulässig. Nur der private Gebrauch ist frei. Bei den benutzten Warenbezeichnungen kann es sich um geschützte Warenzeichen handeln, die nur mit Zustimmung ihrer Inhaber warenzeichengemäß benutzt werden dürfen. Die geltenden gesetzlichen Bestimmungen hinsichtlich Bau, Erwerb und Betrieb von Sende- und Empfangseinrichtungen und der elektrischen Sicherheit sind unbedingt zu beachten. Eine Haftung des Herausgebers für die Richtigkeit und Brauchbarkeit der veröffentlichten Schaltungen und sonstigen Anordnungen sowie für die Richtigkeit des technischen Inhalts der veröffentlichten Aufsätze und sonstigen Beiträge ist ausgeschlossen.

© 2021 elektor international media b.v.

EDITORIAL

Von Jens Nickel

Chefredakteur ElektorMag

Bleiben Sie mit uns jung!

Es ist soweit: Wir sind 60! Im April 1961 brachte Bob van der Horst die erste Ausgabe seiner neuen Zeitschrift *Elektronika Wereld* (Welt der Elektronik) heraus, die später in Elektuur und dann Elektor umbenannt wurde. Der Verleger witterte hier die Marktlücke. Das Elektronik-Wissen dieser Zeit verbarg sich zum Großteil in akademischer Fachliteratur und unzähligen Datenblättern, in schwer verdaulichen Abhandlungen und drögen Diagrammen. Van der Horst sah voraus, dass sich eine Menge Leute so wie er daran begeistern könnte, elektronische Schaltungen aufzubauen. Und - was noch spannender war - sie kreativ für eigene Zwecke anzupassen oder gar von Grund auf neu zu entwickeln. Hierfür benötigt man das Wissen, und das vermittelt man am besten dadurch, dass man seinen Lesern Schritt für Schritt zu einem Erfolgserlebnis verhilft und das Lernen ganz „nebenbei" passiert.
Auch 60 Jahre später bleiben wir diesem erfolgreichen Modell treu. Das Programmieren von Mehrkern-Prozessoren, das Entwickeln mit MicroPython und das Anbinden von eigener Hardware an Web-Plattformen sind nur drei der Themenbereiche, in die wir unsere Leser Schritt für Schritt einführen möchten - mit der bewährten Mischung aus Theorie und Praxis. Als Chefredakteur kommt mir zugute, dass unsere Redakteure und unsere Autoren allesamt vom DIY-Virus befallen sind. Daher bekomme ich fast nie einen Artikel auf den Tisch, der zu trocken und theoretisch ist. Meist muss ich unsere kreativen Köpfe sogar bremsen, damit das Projekt nicht zu umfangreich wird (denn Sie möchten ja auch immer pünktlich eine neue Ausgabe auf den Tisch).
Das Schönste dabei ist, dass wir dadurch immer am Puls der Zeit bleiben. Das hält uns jung und unsere Zeitschrift frisch. Ich weiß nicht, über was Elektor in ein paar Jahrzehnten berichten wird, doch ganz sicher werden unsere Nachfolger in der Redaktionsversammlung überlegen, mit welchem Demoprojekt sich das Ganze am besten vermitteln lässt!

ELEKTOR WIRD 60: ES KOMMT NOCH MEHR!

Der Elektor-Verlag feiert 60 Jahre Elektronik-Innovation mit einigen spannenden Sonderprojekten. Wir arbeiten an einem Jubiläumsbuch, einem Film, einem Live-Event (World Ethical Electronics Forum) und dem „Elektor-Mobilheimlabor". Wir wetten, Sie sind jetzt schon neugierig. Bleiben Sie dran und erfahren Sie mehr in den nächsten Wochen. Sie haben Ideen für weitere besondere Projekte? Schicken Sie mir Ihre Gedanken: denise.bodrone@elektor.com!

Denise Bodrone, „Elektor 60" Festkomitee-Koordinatorin

Unser Team

Chefredakteur: Jens Nickel (v.i.S.d.P.) (redaktion@elektor.de)
Redaktion: Eric Bogers, Jan Buiting, Stuart Cording, Rolf Gerstendorf, Dr. Thomas Scherer, Clemens Valens
Leserservice: Ralf Schmiedel
Elektor-Labor: Mathias Claußen, Ton Giesberts, Luc Lemmens, Clemens Valens
Grafik & Layout: Giel Dols, Harmen Heida
Herausgeber: Don Akkermans

Elektor ist Mitglied des 1929 gegründeten VDZ (Verband Deutscher Zeitschriftenverleger), der „die gemeinsamen Interessen von 500 deutschen Consumer- und B2B-Verlagen vertritt."

Elektor ist Mitglied von FIPP, einer Organisation, die „über fast 100 Jahre gewachsen ist und Medienbesitzer und Content-Ersteller aus der ganzen Welt umfasst".

INHALT

52. Jahrgang
Elektor Nr. 579 - Mai/Juni 2021

Rubriken

3 **Impressum**

45 **Von Entwicklern für Entwickler**
Keine Angst vor dem Mobilfunkmodul

88 **Mein Labor**
Heimlabor und ein Analog-Synthesizer von Kurt Diedrich

96 **Bemerkenswerte Bauteile**
DIP-Schalter

98 **Interaktiv**
Korrekturen und Updates || Fragen und Antworten

100 **Elektor Ethics**
Das Recht auf Reparatur gewinnt an Schwung

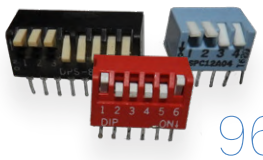
102 **Aus dem Leben gegriffen**
Das große Buch der Patzer

104 **Retronik**
Assembler lernen auf dem Z80

109 **Aller Anfang ...**
Kondensatoren

114 **Hexadoku**
Sudoko für Elektroniker

Bemerkenswerte Bauteile



96

Hintergrund

14 **Elektor wird 60!**
Rückblick auf sechs Jahrzehnte Elektronik

23 **Review**
Siglent-Tischmultimeter SDM3045X

37 **Parallax Propeller 2**
Teil 2: Entwicklungsumgebung und Code

57 **Raspberry Pi Pico und RP2040**
Was man über das Pico-Board und seinen Controller wissen sollte

76 **Der Hitze auf der Spur**
Thermo-Kamera Seek Shot Pro

80 **Objektorientierte Programmierung**
Eine kurze Einführung unter C++

90 **Java auf dem Raspberry Pi**
Teil 1: GPIOs

Projekte

6 **DIY-LiPo Supercharger-Kit**
Vom Prototyp zum Massenmarkt

26 **DC-Stromzange selbst gebaut**
Hall-Sensor + Ferrit-Kerne + Arduino

30 **Temperaturgesteuerte Lötstation 2021**
Einfach selbstgebaut!

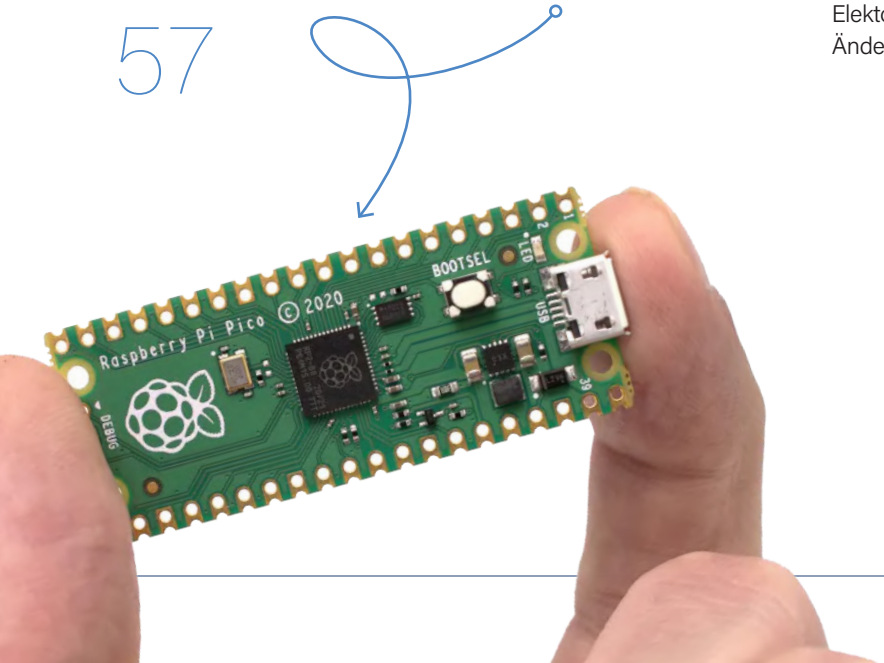
40 **WLAN- statt LoRa-Schalter**
Integriert in Home Assistant mit ESPHome

48 **Zeiterfassung mit ESP32 und Toggl**
Home-Office mit dem M5Stack

64 **Python für Mikrocontroller**
Bildschirme im Kleinformat: Displaytechnik

70 **Gleichspannungswandler 12 V auf 200 V**
Sichere Hochspannung für Röhrenverstärker

Kennengelernt: **DER RASPBERRY PI PICO**
57



Vorschau

Elektor Juli/August 2021
Das nächste Heft ist wie immer randvoll gefüllt mit Schaltungsprojekten, Grundlagen sowie Tipps und Tricks für Elektroniker.

Aus dem Inhalt:
- Magnetische Levitation
- LoRa-Modul mit dem Raspberry Pi Pico
- Electronische Last für DC und AC
- Symmetrische Breadboard-Spannungsversorgung über Micro-USB
- DIY-Kamerasystem für den Raspberry Pi
- Elektronischer Kompass mit GY-271
- MicroPython auf dem ESP32 – Schritt für Schritt

Und vieles mehr!

Elektor Juli/August 2020 erscheint am 1. Juli 2021.
Änderungen vorbehalten.

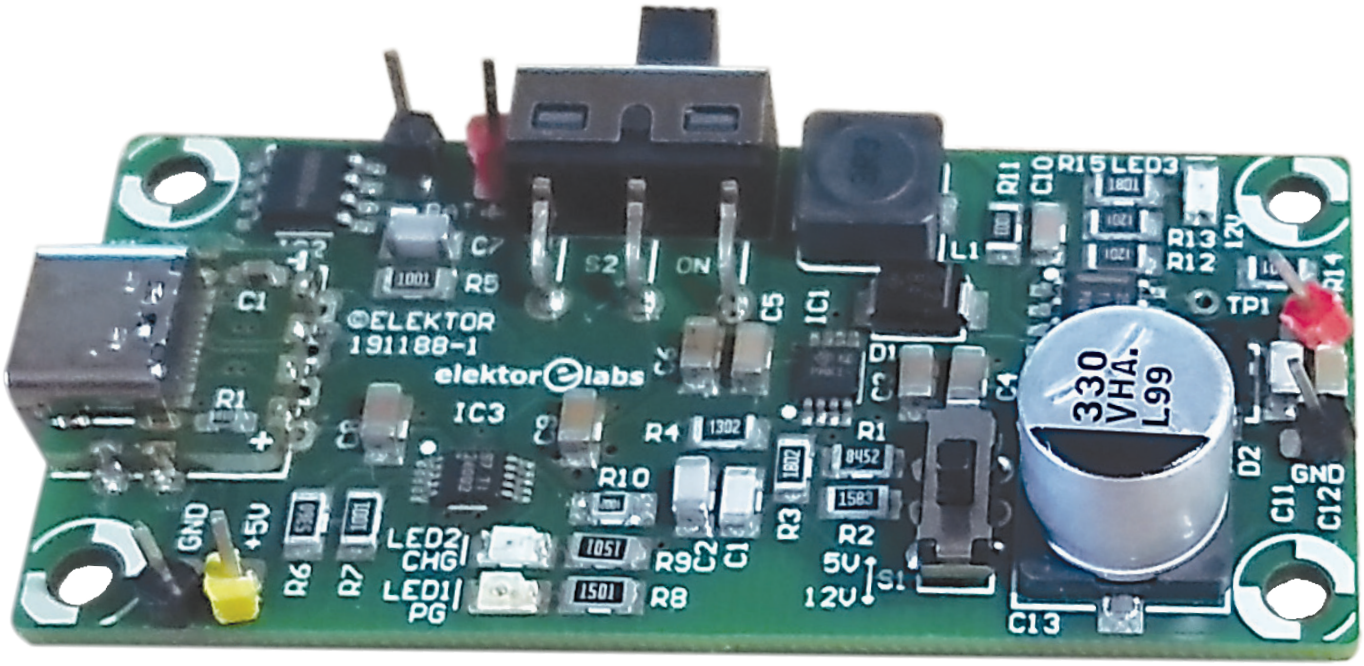
elektor MAG
design › share › sell

DIY-
LiPo-Supercharger-Kit

Vom Prototyp zum Massenmarkt

Von **Mathias Claußen** (Elektor)

GreatScott!, ein auf Selbstbauelektronik fokussierter YouTube-Kanal mit mehr als 1,4 Millionen Abonnenten, und Elektor präsentieren eine LiPo-Akku-Stromversorgung zum Selberbauen. Der handliche Bausatz führt mit sorgfältig ausgewählten 1206-Bauteilen und einer teilbestückten Platine auch in die Welt des SMD-Lötens ein. Bei der Entwicklung des Bausatzes vom Prototyp zum Massenprodukt gab es einige interessante Herausforderungen, von denen wir Ihnen gerne berichten wollen.



SPEZIFIKATIONEN

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

GreatScott! und Elektor trafen sich auf der Messe productronica 2019, um ein neues Projekt zu besprechen - eine Selbstbau-Energiequelle mit wiederaufladbarem Lithium-Akku. Die meisten von uns verwenden ihr vertrautes Labornetzteil für ihre Projekte. Aber wenn Sie an etwas arbeiten, das irgendwann die Werkbank verlassen soll wie ein autonomes Modellauto oder ein Outdoor-Sensor, sind lange Kabel nicht wirklich eine Lösung. Solche Projekte enden in der Regel mit einem mit Gaffertape zusammengeklebten Haufen Batterien, der an einen preiswerten DC/DC-Wandler befestigt ist, als mobile Stromversorgung. Um ein robusteres System mit zusätzlichem Schutz zu erhalten, haben sich GreatScott! und Elektor zusammengetan und eine batteriebetriebene, wiederaufladbare Stromversorgung entwickelt, die weit verbreitete Lithium-Akkus verwendet. Von Anfang an war klar, dass der Bausatz SMD-Bauteile verwenden sollte, um SMD-Lötkenntnisse zu vermitteln und zu zeigen, dass das Löten von SMD keine Zauberei ist, die nur gut ausgebildete E-Zauberer beherrschen. Während des Meetings erfuhren wir, dass GreatScott! bereits Entwürfe für Schaltpläne

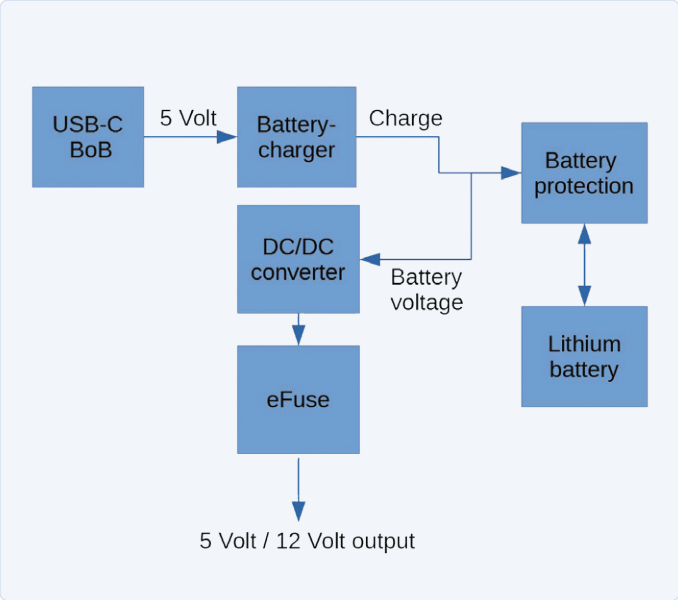


Bild 1. Die Funktionsblöcke des Superchargers.

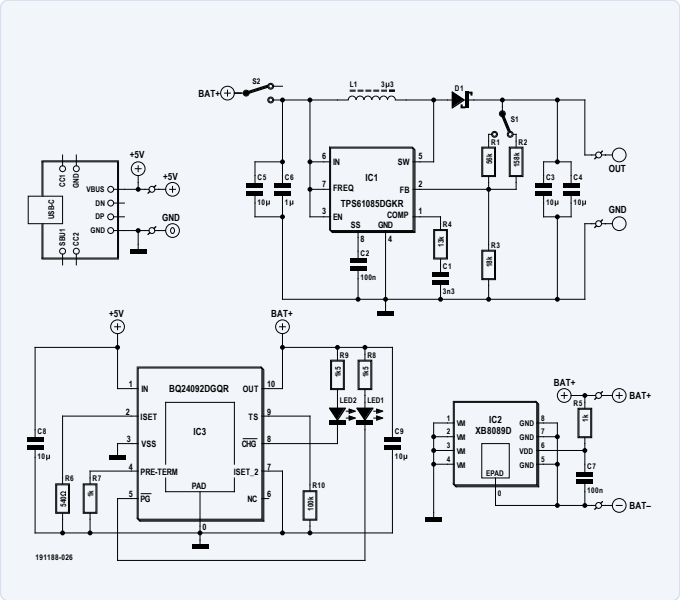


Bild 2. Schaltplan für die erste Revision.

und einige Berechnungen vorbereitet hatte. Der Unterschied zu anderen „Powerpacks" sollte die Möglichkeit sein, die Ausgangsspannung zwischen 5 V und 12 V umzuschalten.

Funktionsblöcke

Um eine Lithium-Batterie-Stromversorgung mit Ladefunktion zu bauen, müssen Sie die Funktionsblöcke in der richtigen Weise verbinden (**Bild 1**).

Der erste Block ist ein USB-Anschluss, damit ein simples 5-V-Steckernetzteil mit USB-Buchse zur Stromversorgung herangezogen werden kann. Die kleine Platine *USB-C BoB*, die nur mit einer USB-C-Buchse und zwei passiven Bauteilen bestückt ist, stellt an ihren Rändern die wichtigsten Signale der 24-poligen Verbindung in „handhabbarer" Form zur Verfügung, so dass sie später leicht auf die Hauptplatine gesetzt und auch für andere Projekte mit USB-C wiederverwertet werden kann. Daran schließt sich die Ladeschaltung *Battery charger* an, die die Eingangsspannung in eine passende(n) Ladespannung/strom verwandelt.

Um Schäden am angeschlossenen Akku zu vermeiden, haben wir eine Schutzschaltung (*Battery protection*) hinzugefügt, um Überstrom, Überladung, Tiefentladung und falsche Polarität zu verhindern. In der Mitte sehen Sie den Gleichspannungswandler (*DC/DC converter*), der 5 V oder 12 V ausgibt. Der anschließende und letzte Block *eFuse* ist eine elektronische Sicherung, die den Ausgangsstrom begrenzt und damit den Gleichspannungswandler und die Batterie vor Schäden schützt. Verbindet man diese Blöcke in der richtigen Weise, erhält man eine portable wiederaufladbare Stromquelle.

Komponenten von der Stange

Die Stückliste wurde für die Beschaffung der Bauteile in China optimiert, was im Dezember 2019 noch sinnvoll zu sein schien, da Waren vom und zum chinesischen Festland in großen Mengen zu einem niedrigen Preis transferiert werden konnten. Die meisten Teile aus der Stückliste konnten aber auch von europäischen Herstellern/Lieferanten bezogen werden, zum Beispiel von Texas Instruments: Zum Laden von Lithium-Ionen- und Lithium-Polymer-basierten Akkus wird ein BQ24092DGQR verwendet, als 5/12-V-Aufwärtswandler ein TPS61085DGKR. Andere Bauteile wie Kondensatoren, Widerstände und Induktivitäten waren natürlich bei europäischen Distributoren erhältlich. Lediglich ein IC war und ist nur über chinesische Distributoren zu beziehen, aber das scheint kein Problem zu sein. Generell wurden für den Entwurf Bauteile von der Stange verwendet. Ein Ingenieur bei Elektor übernahm den Schaltplan (**Bild 2**), die Stückliste und erste Berechnungen. Mit diesem Input war das Layout einer Platine schnell erstellt und das erste Muster wurde geordert (**Bild 3**). Kurz vor Weihnachten 2019 war die Platine bestückt und bereit zum Testen. Der erste Test, der von GreatScott! durchgeführt wurde, sah auch recht vielversprechend aus. Doch damit war die Geschichte nicht zu Ende, im Gegenteil, sie fing erst an, kompliziert zu werden: COVID-19 startete die Reise um die Welt und begann, die Lieferketten zu beeinflussen, wobei Bauteile von der Stange mehr und mehr aus den Regalen verschwanden. Und zu allem Überfluss hinaus tauchten zwei große Fehler im Entwurf auf.

Ein falscher Widerstand und sich auflösende ICs

Ein erster Test ergab, dass der Akku zwar die geforderte Spannung lieferte, das Laden jedoch nicht funktionierte. Da der

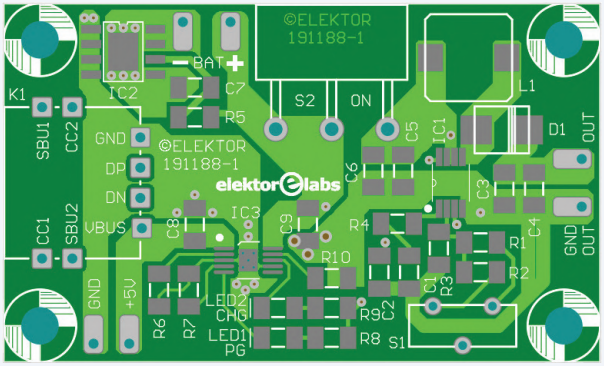


Bild 3. Das erste Platinenlayout.

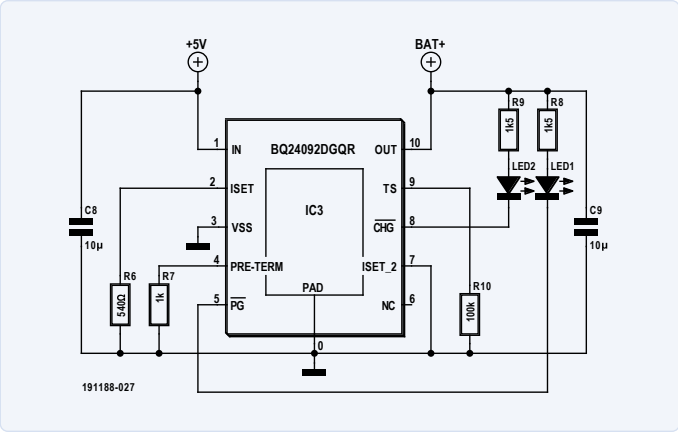


Bild 4. Die Ladeschaltung.

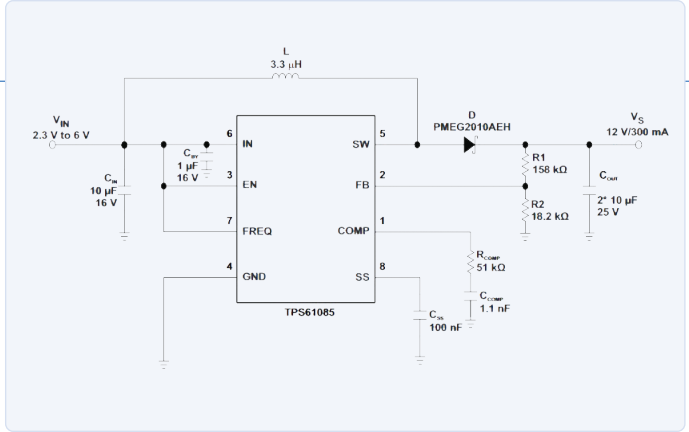


Bild 5. Standardbeschaltung des Aufwärtswandlers.

BQ24092DGQR schon häufiger in Prototypen eingesetzt wurde, war uns dieses Verhalten keinesfalls neu. **Bild 4** zeigt den Schaltplan für das Laden des Akkus mit R10 = 100 kΩ. Der Widerstand R10 wird benötigt, um diese Chips bei Akkupacks ohne einen thermischen Widerstand einsetzen zu können. Je nach Ausführung des BQ2409x kann er 10 kΩ oder 100 kΩ betragen, und hier ist er 10-mal höher als das, was der Typ -2DGQR erwartet. Dies führt dazu, dass der Akku nicht geladen wird, da das Lade-IC fälschlicherweise eine Temperatur des Akkus außerhalb des zulässigen Bereichs annimmt. Die Behebung des Fehlers ist recht einfach und erfordert nur, dass für R10 ein 10-kΩ-Widerstand eingesetzt wird.

Die zweite Panne war etwas schwerwiegender. Während des Tests löste sich der DC/DC-Wandler auf, wenn er einen hohen Strom leiferte. Da in der Blockschaltung in Bild 1 nur einen Batterieschutz aufweist, die den Strom aus der Batterie auf 10 A begrenzt, bedeutet dies etwa 37 W, die der Aufwärtsregler leisten müsste. Bei 5 V Ausgangsspannung könnte dem Spannungswandler theoretisch unter idealen Bedingungen ein Strom von 7,4 A entnommen werden. Da wir aber keine idealen Bedingungen haben, bedeutet dies, dass der DC/DC-Wandler überlastet und zerstört werden kann. Deshalb sollte auch für den Wandler ein Schutz integriert werden. Zunächst ein paar Worte zur Funktionsweise eines Aufwärtswandlers.

Funktionsweise eines DC/DC-Wandlers

Der Zweck eines Gleichspannungswandlers ist es, eine Ausgangsgleichspannung zu erzeugen, die niedriger oder höher liegt als die Eingangsgleichspannung. Bei Eingangswechselspannungen ist dies recht einfach, da man einen Transformator verwenden kann, dessen Ausgangsspannung vom Wicklungsverhältnis abhängt. Gleichspannungswandler dagegen müssen vor der Wandlung zunächst aus der Eingangsgleich- in eine Art Wechselspannung erzeugen. Erstaunlicherweise gab es solche Wandler schon vor dem Silizium-Zeitalter, meistens mechanisch und zigmal größer als das, was man in der modernen Elektronik finden kann. Eine kurze Beschreibung ist auf der Wikipedia-Website [1] zu finden.

In der modernen Elektronik benötigen Gleichspannungswandler nur wenige externe Bauteile, da die meisten Funktionen in einem winzigen IC integriert sind. Wird auf eine niedrigere Ausgangsspannung umgeschaltet, spricht man von dem am häufigsten verwendeten Abwärtswandler oder Buck-Regler oder Tiefsetzsteller. Soll eine niedrige auf eine höhere Spannung angehoben werden, erfolgt dies durch einen Aufwärts- oder Boost-Regler oder Hochsetzsteller. Der hier verwendete TPS61085DGKR ist ein Aufwärtswandler, der 5 V oder 12 V aus einer Eingangsspannung im Bereich 3...4,2 V erzeugt. **Bild 5** zeigt, wie das funktioniert. Vom Eingang V_{IN} führt eine Induktivität L in Reihe mit einer Diode D zum Ausgang V_S. Am Ausgang ist ein Kondensator C_{out} gegen Masse angeschlossen. Zwischen Induktivität und Diode ist der IC-Pin SW angeschlossen, der intern einen Schalter nach Masse darstellt.

Wenn der Schalter geschlossen ist, liegt die Eingangsspannung U_{IN} über der Induktivität. Der Strom I_L über den SW-Schalter nach Masse steigt an und erzeugt ein Magnetfeld in der Induktivität. Diode D verhindert, dass sich der Kondensator C_{out} während der Zeit, in der der Schalter geschlossen ist, entlädt. Wird der Schalter geöffnet, ist die Induktivität bestrebt, die im Magnetfeld gespeicherte Energie wieder loszuwerden. Die Spannung an SW steigt schnell an, über die Eingangsspannung hinaus (die Polarität der Induktivität kehrt sich also um), bis sie höher ist als die des Ausgangskondensators und die Diode leitend wird. Der Strom fließt also weiter Richtung Last und lädt den Kondensator C_{out} auf die Summe der Induktivitäts- plus der Versorgungsspannung auf, bis die Energie des Magnetfelds erschöpft ist. Nun kann der Vorgang erneut beginnen, indem der Schalter geschlossen wird. Wiederholt man diesen Vorgang dauerhaft, ergibt sich eine Ausgangsspannung, die größer ist als die Eingangsspannung.

Um das für den Boost-Betrieb erforderliche Timing zu steuern und eine stabile Ausgangsspannung zu erhalten, werfen wir einen Blick auf das interne Blockschaltbild (**Bild 6**) des Gleichspannungswandler-ICs. Der integrierte Schalter - hier in Form eines FETs - verbindet den „Ausgang" der Induktivität mit Masse. Um das passende Timing für den internen FET zu erzeugen, wird die Ausgangsspannung mit einer Referenzspannung V_{ref} verglichen, die bei diesem IC 1,238 V beträgt. Dazu berechnen wir einen Spannungsteiler, der mit dem Rückkopplungseingang FB verbunden ist, dessen Teilerspannung 1,238 V beträgt, wenn der Ausgang bei 12 V oder 5 V liegt. Wenn die Spannung am Feedback-Pin höher als 1,238 V ist, sorgt die interne Logik dafür, die erzeugte Ausgangsspannung so weit zu senken, bis der Feedback-Pin wieder bei 1,238 V liegt. Intern gibt es Schutzmaßnahmen, die den Strom durch den internen FET begrenzen sollen, eine Unterspannungsabschaltung bieten und den Gleichspannungswandler völlig abschalten, wenn es dem IC zu heiß wird.

Überstrom und Schutz

Wenn der Ausgang des Gleichspannungswandlers hohe Lasten treibt oder gar kurzgeschlossen ist, würde dies zu einer Überlastung des FETs und damit zum plötzlichen Tod des ICs führen. Um dies zu verhindern, gibt es zwei Stellen, an denen ein Schutz angebracht werden kann: vor und hinter dem Wandler. Als schnelle Lösung mit wenigen Bauteilen bietet es sich an, den Strom und die Leistung, die dem Wandler-IC zugeführt werden, zu begrenzen. Um den Strom vor dem Wandler

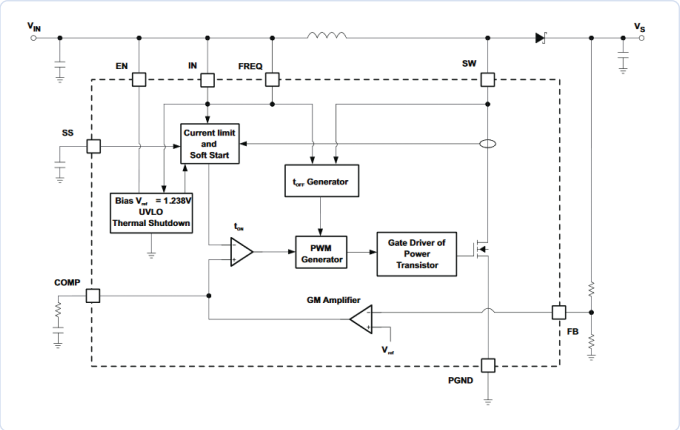


Bild 6. Interne Blöcke des Gleichspannungswandlers.

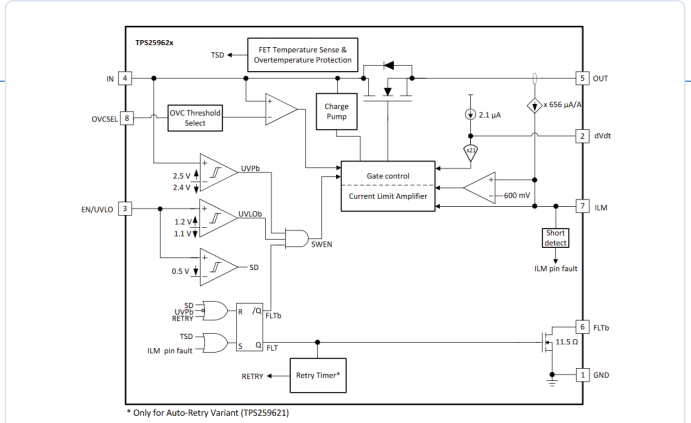


Bild 7. Innenschaltung der eFuse.

zu begrenzen, könnte als Schutzelement ein polymeres Bauteil mit positivem Temperaturkoeffizienten oder kurz rücksetzbare Polysicherung eingesetzt werden. Dies mag als schnelle, simple und billige Lösung erscheinen, aber hätte wegen der Prinzipien, nach denen ein Gleichspannungswandler funktioniert, neue unerwünschte Nebenwirkungen. Wir haben im ersten Versuch eine rücksetzbare Polysicherung auf der Platine eingesetzt. Sie schützte den Wandler zwar vor kurzfristigen Schäden, erzeugte aber unerwünschte Schwingungen. Um die Ursache dafür herauszufinden, sind keine Messungen oder gar ein Testaufbau erforderlich, sondern nur sorgfältiges Nachdenken. Eine rücksetzbare Polysicherung begrenzt den Stromfluss, indem ihr Innenwiderstand mit steigender Temperatur zunimmt. Da die Sicherung selbst einen definierten Widerstand hat, erhöht der Stromfluss durch sie ihre Innentemperatur. Je höher also der Strom durch die Sicherung ist, desto höher wird die Temperatur und der daraus resultierende Widerstand im Inneren der Sicherung. Dieser Widerstand begrenzt aber letztendlich die Strommenge, die durch die Sicherung fließen kann.

Angenommen, der Wandler arbeitet mit einer hohen Ausgangsbelastung, führt dies auch zu einem hohen Eingangsstrom. Die Polysicherung wird warm und erhöht ihren Widerstand, wodurch der Stromfluss begrenzt wird. Der erhöhte Widerstand der Polysicherung führt auch zu einem höheren Spannungsabfall und damit zu einer niedrigeren Eingangsspannung für den Wandler. Der Strom durch den Wandler ist zwar jetzt begrenzt und das IC wird nicht zerstört, allerdings muss er nun die Ausgangsspannung mit einer noch niedrigeren Eingangsspannung aufrechterhalten, was wiederum zu einem höheren Strombedarf führt. Dies führt zu einem weiter erhöhten Widerstand in der Polysicherung und zu einem Anstieg des Spannungsabfalls und zu einer niedrigeren Eingangsspannung für den Wandler und ... irgendwann dazu, dass der Schwellwert der Eingangsspannung unterschritten ist, die Unterspannungsabschaltung aktiv wird und den Spannungsregler abschaltet.

Wenn dann weniger Strom fließt, kühlt die Polysicherung ab und verringert ihren Widerstand. Der Spannungsabfall über der Polysicherung wird geringer und die Eingangsspannung für den Wandler höher. Sie können sich denken, was passiert: Der Wandler läuft wieder an und das Spiel beginnt erneut! Nun könnte man denken: Was geht mich das an, wenn doch nur die Ausgangsspannung stabil bleibt? Doch die Oszillation wirkt sich auch auf den Ausgang aus, weil Spikes und andere unschöne Komponenten die Ausgangsspannung verunreinigen.

Am Eingang des Wandlers ist eine Polysicherung also nicht das Gelbe vom Ei, doch wie sieht es mit dem Ausgang aus? Ja, dies wäre ein einfacher Ansatz für nur eine Spannung und nur eine Stromgrenze. Bei zwei Ausgangsspannungen und zwei unterschiedlichen Stromgrenzen würde eine einzelne Polysicherung keinen definierten Schutz für beide Spannungen und Ströme bieten. Eine vielseitigere Lösung für dieses Problem, insbesondere bei wechselnden Spannungen und Strömen, kann eine ***eFuse*** sein, die bereits in einem Elektor-Online-Artikel vorgestellt wurde [2]. Nachdem wir uns nach eFuses umgesehen haben, die für 5 V bis 12 V Eingangsspannung geeignet sind und 0,5 A bis 2 A verkraften können, fiel die Wahl auf die eFuse TPS259621DDAR von Texas Instruments. Wie Sie in **Bild 7** sehen können, bietet die eFuse mehr als nur einen einfachen Überstromschutz, sondern kann auch für den Über- und Unterspannungsschutz verwendet werden. Bild 2 zeigt, dass wir ja schon einen Widerstands-Schalter zum Ändern der

Anzeige

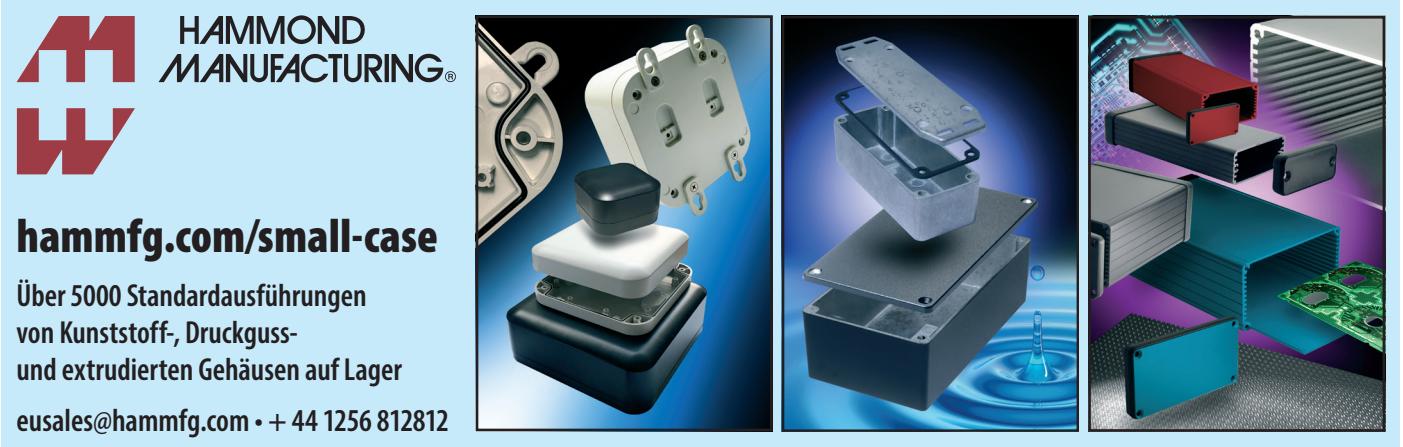


Bild 8. Aus flach mach rund: beschädigte Lithiumakkus.

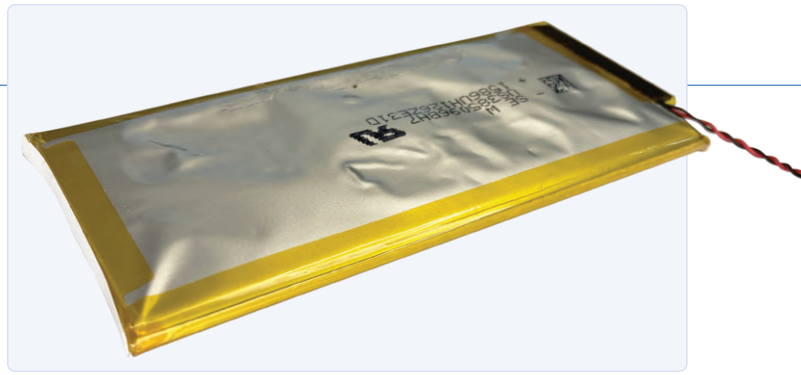


Bild 9. Ein beschädigter Handy-Akku.

Spannungen verwenden. Da die Strombegrenzung ebenfalls mit Widerstandswerten erfolgt und mit den eingestellten Spannungen geändert wird, können wir mit diesem Schalter zwei unabhängige Kanäle umschalten und die Ausgangsspannung mit der Strombegrenzung gleichzeitig einstellen. Die Schalter, die wir für dieses Projekt verwenden, stellen also eine moderne Lösung für einen einstellbaren Überstromschutz dar.

Lithium-Batterien: leicht aufladbar, leicht brennbar

Die meisten portablen elektronischen Geräte verwenden heute Batterien auf Lithiumbasis. Dank der Fortschritte in der Batterietechnik haben diese Zellen eine höhere Leistungsdichte pro Volumen und pro Gewicht. Während ältere Technologien wie NiCd und NiMh komplexere Schaltungen benötigen, um zu laden und das Ende des Ladevorgangs zu erkennen, sind Lithium-Akkus einfacher zu handhaben. Wenn die Zelle nicht vollständig entladen ist und eine Restspannung von rund 2,5 V aufweist, erfolgt das Aufladen mit einem CC/CV-Ansatz - zuerst konstanter Strom mit einer Ladeschlussspannung und Beenden des Ladens, wenn eine definierte Spannung erreicht wird. Für Lithium-Ionen- und Lithium-Polymer ist dies typischerweise 4,15 V. Der BQ24092DGQR tut genau dies, indem er einen begrenzten Strom von bis zu 1 A anlegt, bis die Zellspannung 4,15 V erreicht ist.

Ein zusätzlicher, aber optionaler Schutz von Lade-ICs wäre die thermische Überwachung. Hochwertige Ladegeräte und Akkupacks enthalten eine thermische Überwachung, meist einen thermischen Widerstand, um eine Überhitzung der Akkus während des Ladevorgangs zu verhindern. Wegen der mannigfaltigen hier einsetzbaren Lithium-Akkutypen und -marken ist es nicht sinnvoll, diesen Widerstand auszuwerten. Statt dessen wird hier ein Widerstand genutzt, mit dem das Ladegerät ohne thermische Überwachung arbeiten kann. Achten Sie nur darauf, dass Sie den Widerstand richtig auslegen; andernfalls wird kein Ladevorgang stattfinden. Der Anschluss inkompatibler Akkus auf Lithiumbasis, wie etwa Lithium-Ionen-Phosphat (LiFePo), kann dazu führen, dass der Akku Feuer fängt! Achten Sie darauf, qualitativ hochwertige Zellen zu verwenden, da eine beschädigte Zelle sich auszudehnen kann oder sogar heiß wird und möglicherweise gar abbrennt. **Bild 8** zeigt beschädigte Lithium-Batterien, die sich von ihrer ursprünglich flachen zu einer fast runden Form ausgedehnt haben, **Bild 9** eine beschädigte Handy-Zelle,

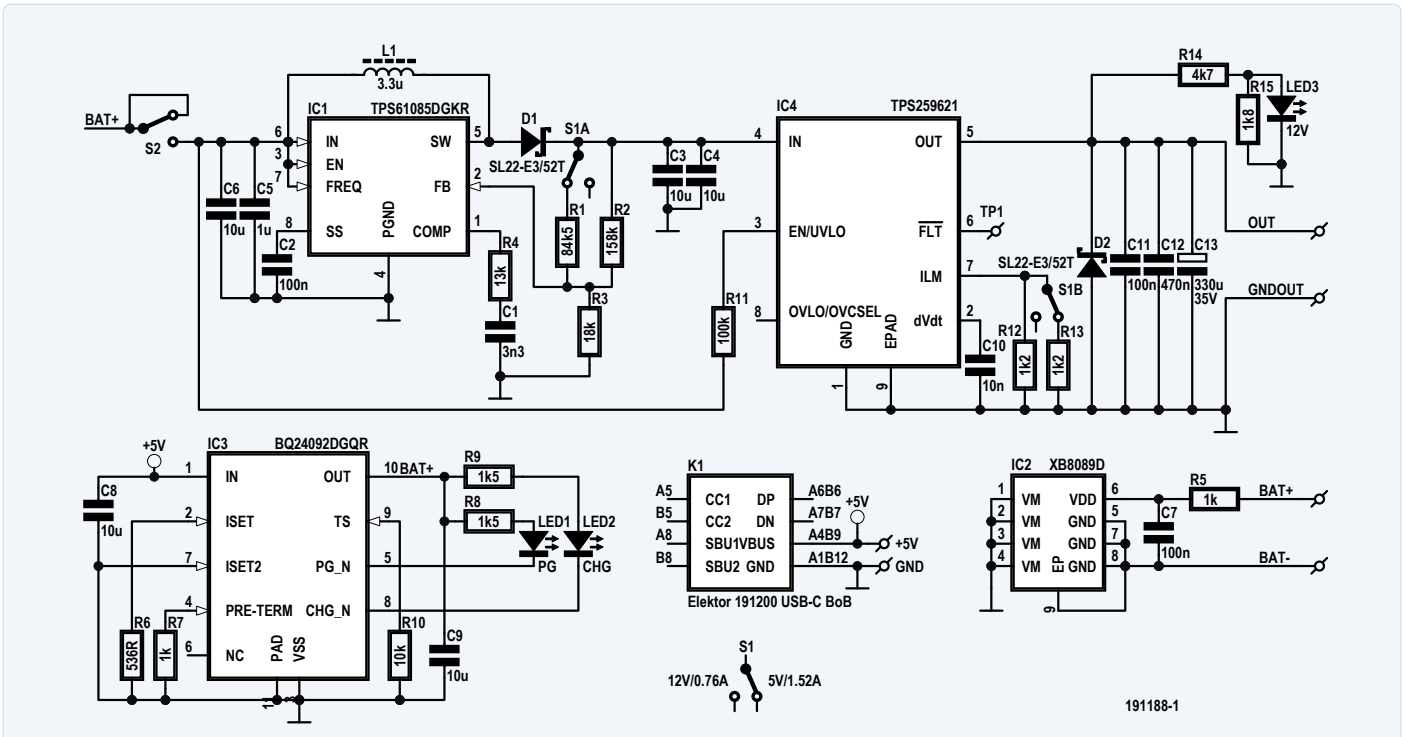


Bild 10. Schaltung des DIY-LiPo-Supercharger-Kits.

die sich ebenfalls leicht ausdehnte und dabei genug Kraft aufbrachte, um ein verklebtes Handy-Display aus seinem Gehäuse zu sprengen.

Endgültige Spezifikationen

Nun müssen nur noch die Parameter für die eFuse festgelegt werden. Die eingestellten Werte von 1,5 A bei 5 V und 0,75 A bei 12 V garantieren einen sicheren Betrieb des Gleichspannungswandlers. Nun noch zur Einstellung des Ladestroms für den angeschlossenen Akku. Der BQ24092DGQR bietet bis zu zwei Ladestromeinstellungen, abhängig von den Pins ISET und ISET2. Wenn ISET2 mit Masse verbunden ist, bedeutet dies, dass der Chip die Stromeinstellung an ISET verwendet, was bei einem Widerstand von R6 = 536 Ω zu einer Ladestrombegrenzung von 1 A führt. Achten Sie darauf, dass der Akku, den Sie verwenden, diesen Ladestrom gut verträgt.

Ein wenig Feedback und Fertigstellung der Platine

Ein letztes i-Tüpfelchen des Entwurfs ist eine optische Anzeige, ob die Ausgangsspannung auf 5 V oder 12 V eingestellt ist. GreatScott! und Elektor kamen zu dem Schluss, dass eine zusätzliche LED als Indikator die Benutzerfreundlichkeit des Produkts erheblich verbessern und einem versehentlichen Anschluss von 12 V an ein 5-V-System vorbeugen würde. Die zusätzliche 12-V-Anzeige besteht aus der LED3 und zwei Widerständen. Die LED leuchtet nur, wenn 12 V am Ausgang anliegen.

Die zweite Platinenrevision funktionierte wie erwartet. Nun blieb nur noch, den Entwurf abzurunden, die letzten Änderungen an Text und Logo vorzunehmen, Referenzen richtig zu platzieren und das gesamte Material an die Produktion zu übergeben. **Bild 10** zeigt den endgültigen Schaltplan des DIY-LiPo-Supercharger-Kits.

Vom Prototyp zur Marktreife

Die Entwicklung von Produkten und der Bau von Prototypen im eigenen Haus ist etwas anderes als die Produktion durch automatisierte Anlagen und Dienstleister. Was ich noch nicht erwähnt habe, ist das kleine USB-C-BoB, dessen Schaltplan in **Bild 11** zu sehen ist und das auf der Hauptplatine angebracht wird. Da ein USB-C-Anschluss für Strom und Daten aber nicht nur für dieses Produkt sinnvoll sein kann, wurde die kleine Platine (**Bild 12**) mit einer eigenen Elektor-Projektnummer versehen. Alle wichtigen Anschlüsse für Datenübertragung in USB2.0-Geschwindigkeit und Stromversorgung sind als Halbloch-Kantenkontakte an drei Kanten des Boards ausgeführt. Ein solches BoB ist sehr nützlich und bequem anwendbar, die Massenproduktion bereitete allerdings Schwierigkeiten. Die Platinen sind nämlich so klein, dass die Maschinen Schwierigkeiten hatten, sie zu verarbeiten. Um dieses Problem zu lösen, musste die Herstellerfirma einige technische Maßnahmen (= Kosten) ergreifen. Und auch die Halbloch-Kantenkontakte sind beileibe kein „Standard", es musste Rücksprache mit dem Hersteller gehalten und ein Kostenvoranschlag für zusätzliche Fertigungskosten eingeholt werden.

Wie sollen die Bauteile eines SMD-DIY-Bausatzes verpackt oder organisiert werden? Umständliche Verpackung bedeutet manuelle Arbeit bedeutet zusätzliche Kosten. SMD-Bauteile in der richtigen Menge in kleine Tüten zu verpacken ist etwas, was Sie im Vorfeld mit Ihrem Fertigungs- und Serviceanbieter besprechen sollten. Diese Aufgaben sind arbeitsintensiv und können unerwartete Kosten zu Ihrem Projekt hinzufügen. Es kann am Ende sogar günstiger sein, eine fix und fertig bestückte Platine anzubieten als eine Leerplatine mit verpackten Bauteilen. Doch da es uns ja auch um eine SMD-Lötübung geht, haben wir nur die ICs mit nicht für den dicken Lötkolben zu bewältigenden Footprints vormontiert.

Die Übergabe von Gerberdateien und Pick-and-Place-Daten an einen Hersteller kann zu Überraschungen im Prozess führen. In der beinahe letzten Phase des Produktionsprozesses erhielten wir ein Muster des endgültigen Bausatzes. Während Widerstände, Induktivität, LEDs und alle anderen mechanischen Teile ordentlich in Tütchen verpackt waren, kam die Platine mit einer kleinen Überraschung an. Die vorbestückten Bauteile waren wie erwartet vorbestückt, aber es schien ein Problem mit der Lötpaste zu geben. Der Lötpastenlayer wurde nämlich so exportiert, als ob alle Bauteile zu platzieren wären. Die Lötpaste war auf alle Pads der Platine aufgetragen, was das Löten der Bauteile unnötig erschweren und komplizieren würde. **Bild 13** zeigt das Ergebnis und gibt einen Eindruck von den unerwarteten Schwierigkeiten. Wir mussten einen neuen Satz von Gerberdateien anfertigen mit einem angepassten Lötpastenlayer, so dass während der Fertigung nur die Pads für die vormontierten Bauteile mit Lötpaste belegt werden.

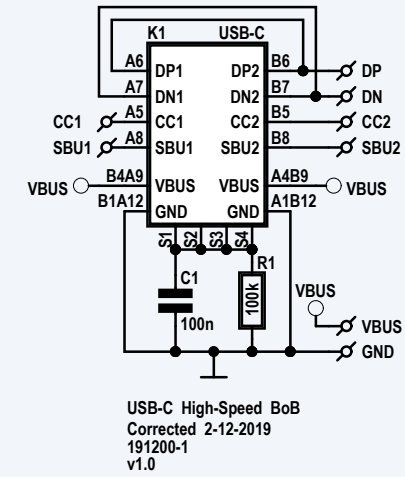


Bild 11. Schaltung der kleinen USB-C-Platine.

Bild 12. So präsentiert sich das USB-C-BoB.

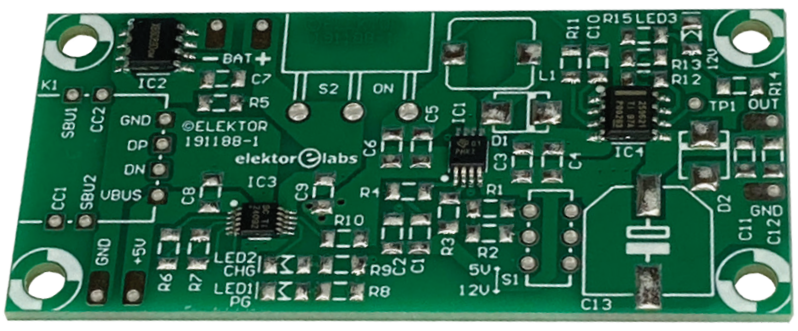


Bild 13. Platine mit falsch aufgebrachter Lötpaste.

Bild 14. Verkaufsverpackung des DIY-LiPo-Supercharger-Kits.

Bild 15. Sorgsam verpackte Einzelteile des DIY-LiPo-Supercharger-Kits.

USB-C-PD UND DIE WIDERSTÄNDE

Wenn Sie an ein klassisches 5-V-1-A-Steckernetzteil ein normales USB-A-nach-USB-C-Kabel anschließen, funktioniert das Laden tadellos. Dies ist bei USB-C-PD (Power delivery) nicht der Fall. Bei Micro-USB werden einfach VCC und Ground zur Stromversorgung herangezogen, bei USB-C-PD werden dazu jedoch zwei zusätzliche Widerstände für die Stromeinstellung benötigt. Das DIY-LiPo-Supercharger-Kit von GreatScott! Setzt auf die bewährten Micro-USB-Verbindungen, was bedeutet, dass USB-C-PD-Adapter in der Regel die Stromversorgung des Boards verweigern.

Ein Karton und eine Anleitung

Es ist nun an der Zeit, einen Blick auf das Endprodukt zu werfen. **Bild 14** zeigt den verpackten Bausatz. Doch es reicht nicht aus, alle Teile in eine Plastiktüte zu werfen und eine kurze Anleitung hinzuzufügen. Wenn Sie vom handgefertigten Prototypen zur Massenproduktion übergehen, müssen Sie sich auch Gedanken über die Verpackung machen und darüber, wie Sie Ihren Kunden eine Anleitung zur Verfügung stellen. Abhängig von den Ländern, in die Sie Produkte versenden, gelten bestimmte Vorschriften für die Gestaltung von Handbüchern in den entsprechenden Sprachen. Im *DIY LiPo Supercharger Kit* sind nicht nur alle benötigten Bauteile (vorsortiert in beschrifteten Tüten) enthalten, sondern auch ein QR-Code mit einem Link zu einer ausführlichen Anleitung. Dadurch konnten wir Ressourcen einsparen und dazu beitragen, den Planeten ein bisschen grüner zu halten. In **Bild 15** sehen Sie, was im Bausatz enthalten ist. Sie können sich ein Bauvideo von GreatScott! auf YouTube

STÜCKLISTE DIY LIPO SUPERCHARGER KIT

Widerstände:

R1 = 84k5, 1 %, 0,25 W, SMD 1206
R2 = 158 k, 1 %, 0,25 W, SMD 1206
R3 = 18 k, 1 %, 0,25 W, SMD 1206
R4 = 13 k, 1 %, 0,25 W, SMD 1206
R5,R7 = 1 k, 1 %, 0,25 W, SMD 1206
R6 = 536 Ω, 1 %, 0,25 W, SMD 1206
R8,R9 = 1k5, 1 %, 0,25 W, SMD 1206
R10 = 10 k, 1 %, 0,25 W, SMD 1206
R11 = 100 k, 1 %, 0,25 W, SMD 1206
R12,R13 = 1k2, 1 %, 0,25 W, SMD 1206
R14 = 4k7, 1 %, 0,25 W, SMD 1206
R15 = 1k8, 1 %, 0,25 W, SMD 1206

Kondensatoren:

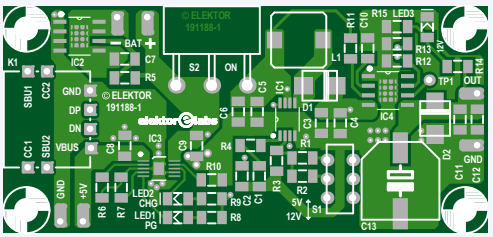
C1 = 3n3, 5 %, 50 V, NP0, SMD 1206
C2,C7,C11 = 100 n, 5 %, 50 V, C0G, SMD 1206
C3,C4,C6,C8,C9 = 10 µ, 10 %, 25 V, X7R, SMD 1206
C5 = 1 µ, 10 %, 50 V, X7R, SMD 1206
C10 = 10 n, 5 %, 50 V, X7R, SMD 1206
C12 = 470 n, 10 %, 50 V, X7R, SMD 1206
C13 = 330 µ, 20 %, 35 V, SMD 10,3x10,3

Induktivität:

L1 = 3,3 µ, 20 %, 3,5 A, 35 mΩ, SMD 7,3x7,3x4,5 mm³

Halbleiter:

D1,D2 = SL22-E3/52T, SMD SMB
LED1 = 11-21/GPC-AM2P1/2T, grün, SMD 1206
LED2,LED3 = 15-21SDRC/S530-A2/TR8, rot, SMD 1206
IC1 = TPS61085DGKR, SMD VSSOP-8
IC2 = XB8089D, SMD SOIC-8-EP
IC3 = BQ24092DGQR, SMD MSOP-10-EP
IC4 = TPS259621DDAR, SMD SO-PowerPad-8



Außerdem:

PC1..PC6 = Stiftleiste, Einzelpins
S1 = Schalter DPDT, THT, 9,1x3,5 mm² (K2-2235D-F1)
S2 = Schalter SPDT, 250 VAC, 3 A (XKB, SS-12D06L5)
K1 = USB-C-BoB, Elektor 191200-1
Platine 191188-1 v2.1

STÜCKLISTE USB-C-BOB

Widerstand:

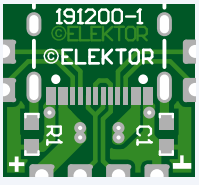
R1 = 100 k, 1 %, 100 mW, SMD 0603

Kondensator:

C1 = 100 n, 10 %, 100 V, X7R, SMD 0603

Außerdem:

K1 = USB - Typ C, Buchse, Nennstrom 3 A, SMD/THT
Platine 11200-1 v1.0



200% der tatsächlichen Größe

[3] ansehen. Und wenn Sie Zweifel an Ihren Fähigkeiten haben, SMD-Bauteilen löten zu können, schauen Sie sich sein Video-Tutorial an [4]. Eine wiederaufladbare Batterie ist im Bausatz nicht enthalten. Sie können sich aber eine passende aus einem beliebigen Online-Shop besorgen, oder eine, den Sie gerade zur Hand haben, verwenden. Achten Sie darauf, dass der Akku 1 A Ladestrom verträgt und es ein einzelliger Lithium-Ionen- oder Lithium-Polymer-Akku ist.

Für den Zusammenbau sollten Sie sich Zeit nehmen und geduldig sein. SMD-Bauteile, auch wenn sie groß wie 1206 sind, haben Ninja-ähnliche Tarnfähigkeiten, wenn sie von der Werkbank fallen oder sich hinter/unter Teilen auf Ihrem Schreibtisch verstecken. Nachdem Sie den Bau und die Tests abgeschlossen haben, ist der DIY-LiPo-Supercharger einsatzbereit. Der Schalter S2 auf der Platine dient dazu, den Akku mit dem Wandler zu verbinden beziehungsweise von ihm zu trennen. Damit kann man den Ausgang abschalten und außerdem Strom sparen, denn auch ein unbelasteter Wandler kann einige Milliampere Strom ziehen. Nach dem Anschließen eines Akkus muss dieser zunächst vollständig aufgeladen werden. Beim DIY-LiPo-Supercharger zeigt LED1 an, dass die vom USB-C-Anschluss gelieferte Spannung in einem gültigen Bereich liegt. Während des Ladevorgangs können Sie sehen, dass LED2 leuchtet. Sie leuchtet nicht (mehr), wenn der Akku vollständig geladen ist. Dies bedeutet, dass Ihre portable Energiequelle nun für Ihre Projekte einsatzbereit ist.

Der Weg ist das Ziel

Eine tragbare Energiequelle zur Hand zu haben, insbesondere mit umschaltbaren Ausgangsspannungen, ist äußerst nützlich. **Bild 16** zeigt einen Raspberry Pi Zero, der mit dem DIY-LiPo-Supercharger betrieben wird. Wenn Sie ein knappes Budget haben und nichts löten wollen, können Sie sich an Heißkleber, Klebeband, fliegende Drähte und billige chinesische Fertigkomponenten halten. Mit diesem Kit aber bauen Sie alles selbst, und das SMD-Löten macht es besonders reizvoll. Eine Lösung von der Stange zu haben, die einfach aus einer Plastiktüte kommt und einsatzbereit ist, macht viel weniger Spaß! Der Weg ist die Belohnung, und wenn Sie das Ziel erreicht haben, besitzen Sie darüber hinaus eine praktische portable Stromquelle für Ihre nächsten Projekte. Suchen Sie nach Schaltplänen? Die finden Sie auf der Elektor-Labs-Webseite dieses Projekts [5]. Wenn Sie Unterstützung benötigen, können Sie dort auch einen Kommentar hinterlassen.

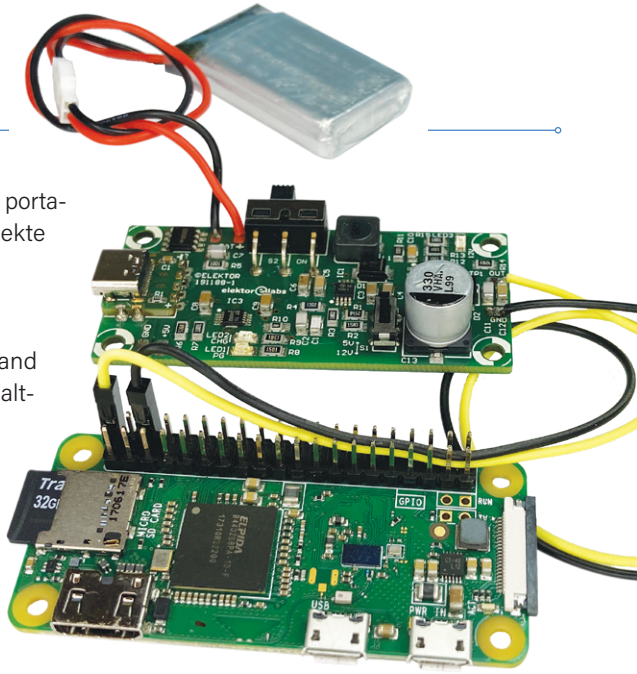


Bild 16. Ein Raspberry Pi Zero, der vom Supercharger versorgt wird.

191188-B-02

Ein Beitrag von:
Idee: **GreatScott!**
Entwurf: **Ton Giesberts**
Autor: **Mathias Claußen**
Redakteure: **Jens Nickel** und **C. J. Abate**
Illustrationen: **Patrick Wielders**
Übersetzung: **Rolf Gerstendorf**
Layout: **Giel Dols**

Haben Sie Fragen oder Kommentare

Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Schreiben Sie dem Autor eine E-Mail an mathias.claussen@elektor.com oder kontaktieren Sie Elektor unter editor@elektor.com.

PASSENDE PRODUKTE

- **GreatScott! – DIY LiPo Supercharger Kit**
  www.elektor.de/diy-lipo-supercharger-kit-by-greatscott
- **Weller WE 1010 Digitale Lötstation (Education Kit)**
  www.elektor.de/weller-we-1010-digital-soldering-station-education-kit
- **Fumetractor mit LED-Lampe**
  www.elektor.de/fumetractor-with-led-light

WEBLINKS

[1] **Zerhacker**: https://de.wikipedia.org/wiki/Zerhacker_(Elektrotechnik)

[2] **S. Cording, „The Modern Fuse“, ElektorMagazin.com, 12.8.2020**: www.elektormagazine.com/articles/modern-fuse

[3] **GreatScott!, „DIY LiPo Supercharger (Charge, Protect, 5V/12V Boost V2)“, 6.12.2020**: https://youtu.be/6LxRnf6sQNQ

[4] **GreatScott!, „How to Solder properly: Through-hole (THT) & Surface-Mount (SMD)“ 3.9.2017**: www.youtube.com/watch?v=VxMV6wGS3NY

[5] **DIY-LiPo-Supercharger auf Elektor Labs**: www.elektormagazine.de/labs/diy-lipo-supercharger-kit-by-greatscott

60 Jahre

So sah das erste Cover von Electronica Wereld aus. Später wurde der Name in Elektuur geändert, was so viel wie „E-Lektüre" bedeutet. Im Jahr 1970 erschien dann die erste deutsche Ausgabe unter dem Namen Elektor.

Von **Jens Nickel** (Elektor)

Sechzig Jahre Elektor! Das ist eine ziemlich lange Zeit für eine Zeitschrift, die sich der Elektronik widmet, einem Thema, das sich so schnell weiterentwickelt, dass es schwer sein kann, mit all den Entwicklungen Schritt zu halten. Seit 1961 berichten wir nicht nur über die neuesten Technologien, Geräte, Platinen und Bauteile (mit unserer typischen Mischung aus Theorie und Praxis), sondern teilen auch unsere Ideen und Projekte mit professionellen Elektroingenieuren und Makern. Anlässlich unseres Jubiläums haben wir kürzlich einige unserer Autoren und Redakteure gebeten, interessante Projekte und Artikel aus sechs Jahrzehnten Elektor auszuwählen!

Elektor

Unse wohlbekannte Zeitschrift wurde im April 1961 in den Niederlanden ins Leben gerufen. Es gibt wohl niemanden, der besser darüber berichten kann als Harry Baggen, der jahrzehntelang Chefredakteur der niederländischen Ausgabe war und auch viele Jahre als internationaler Chefredakteur des Magazins gearbeitet hat.

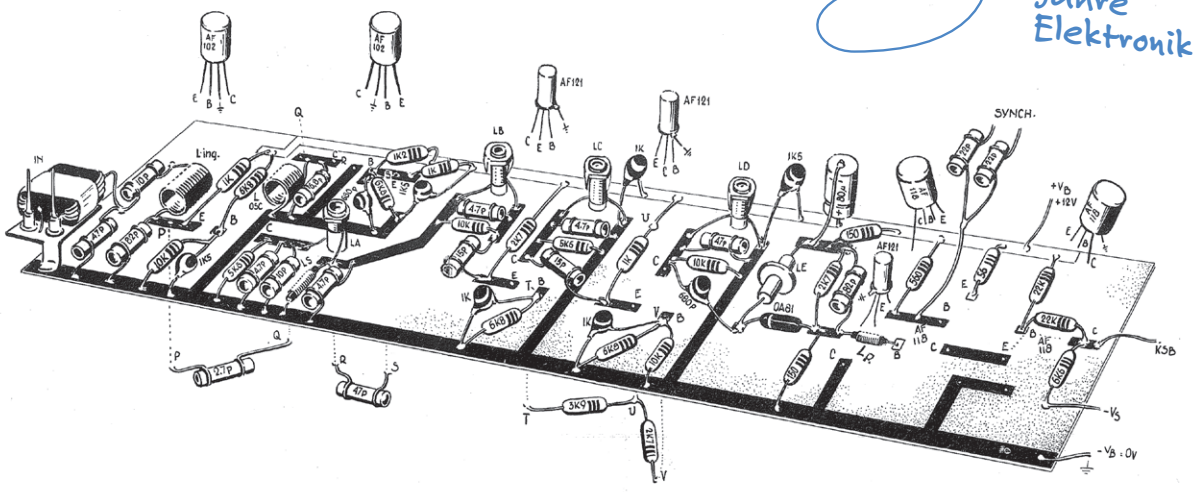
„Als Bob W. van der Horst sein eigenes Elektronikmagazin unter dem Namen Electronica Wereld *startete, hatte er sich zum Ziel gesetzt, ein besseres Magazin zu machen, als es die bestehenden Zeitschriften waren. Er wollte den mystischen Schleier um das Thema entfernen und hauptsächlich die praktische Seite zeigen. Anfangs bestand die Zeitschrift vor allem aus einer Sammlung von Produktbesprechungen und technischen Hintergrundartikeln, doch nach einigen Jahren setzte sich die rein praktische Seite durch. Und so ist es auch bis heute geblieben. Keine endlosen theoretischen Abhandlungen, nein: praktische Schaltungen mit zeitgemäßen Bauteilen, das war und ist das* Ziel. *Bob wollte auch immer an vorderster Front bei neuen Bauteilen dabei sein. In den ersten Jahren waren es noch hauptsächlich Röhrenschaltungen, die in der Zeitschrift erschienen, doch bald wurde ihr Platz von Halbleitern übergenommen. Neben Transistoren wurden häufig auch FETs verwendet. Elektor galt bald als eigensinnig, anders denkend, innovativ: genau das, was Van der Horst erreichen wollte!“*

Die erste Ausgabe enthielt mehr als 50 Schaltungen, zum Beispiel einen Verzerrungsmesser und einen Eintransistor-Sender (kostenloser Download unter www.elektormagazine.de/210025-02). Und es kamen noch viele, viele weitere Projekte dazu! Eine schöne Auswahl an Projekten finden Sie auf den nächsten Seiten, zusammen mit einigen Kommentaren von Experten und Autoren.

Elektor-Mitglieder können diese Artikel über die angegebenen Weblinks herunterladen. Zum 60-jährigen Bestehen von Elektor stellen wir sieben dieser „Editors’ Picks“ bis zum 30. Juni 2021 frei zur Verfügung. Viel Spaß damit!

210025-01

60 Jahre Elektronik



Die Leidenschaft für einen praxisnahen Umgang mit Elektronik liegt in der DNA von Elektor. Vom ersten Tag an wollten wir unseren Lesern die Möglichkeit geben, zu bauen und gleichzeitig zu lernen. Daher haben die Mitglieder der Elektor-Community unsere Projekte immer geliebt.

Harry Baggen

(ehemaliger Chefredakteur)

100-W-Edwin-Verstärker (1971)

Elektor war schon in den Anfangsjahren für seine ungewöhnlichen Selbstbauprojekte im Audiobereich bekannt, und das ist bis heute so geblieben. Ein „bahnbrechender“ Entwurf war 1970 der Edwin-Verstärker, ein für heutige Verhältnisse sehr einfacher Audio-Verstärker, der über eine ruhestromlose Endstufe verfügte. Er wurde zum Klassiker und ist einer der meistkopierten Audioverstärker des letzten Jahrhunderts.

www.elektormagazine.de/magazine/elektor-197101/54726

Elektorskop (1975)

In den siebziger Jahren waren Oszilloskope für Hobby-Elektroniker fast unerschwinglich. Das war ein guter Grund für die damaligen Ingenieure bei Elektor, ein Selbstbau-Oszilloskop zu entwickeln und zu veröffentlichen. Das war damals ein hartes Stück Arbeit, zumal die Oszilloskop-Röhre, die zugehörige Steuerung und die Hochspannung sorgfältig abgestimmt werden mussten und viele der Bauteile (vor allem die Röhre) nur sehr schwer zu bekommen waren. Dennoch ist es gelungen, eine Schaltung zu präsentieren, die sicher nachgebaut werden konnte.

www.elektormagazine.de/magazine/elektor-197503/55776

Universal-Zähler mit µP (1985)

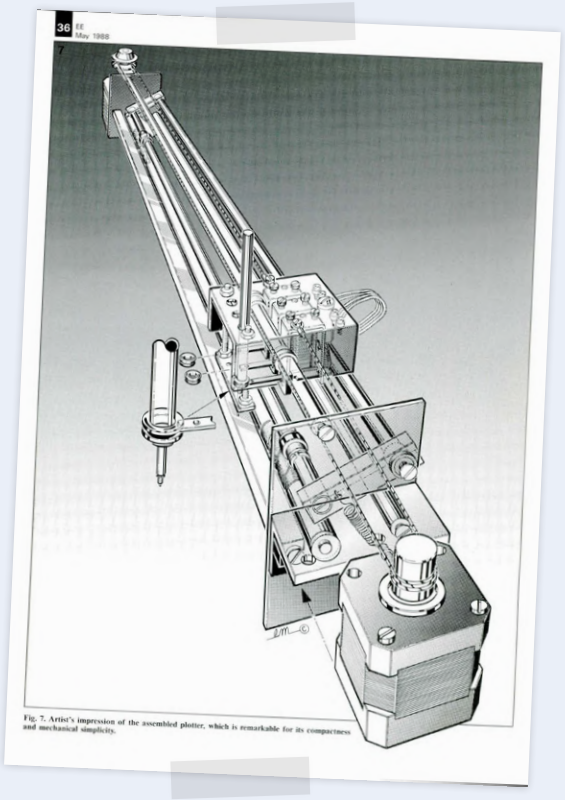
Anfang der 1980er Jahre präsentierte Elektor eine Reihe von Messgeräten, deren Highlight der mikroprozessorgesteuerte Frequenzzähler war. Bei Ausstattung und Funktionalität konnte dieses Schmuckstück locker mit professionellen Geräten mithalten. Es hatte einen Frequenzbereich bis 1,2 GHz, Autoranging, eine Folientastatur zur Bedienung und ein alphanumerisches Display.

www.elektormagazine.de/magazine/elektor-198501/47917

Selbstbauplotter Mondrian (1988)

In einer Zeit, in der Plotter nur in Labors und Ingenieurbüros zu finden waren, veröffentlichte Elektor den Entwurf eines sehr einfachen Plotters zum Selbstbau. Die Teile für den Plotter kosteten im Vergleich zu einem echten Plotter nur ein' Appel und ein Ei, und zum Zeichnen wurden Filzstifte verwendet. Später erschien sogar ein HP-GL-Treiber für den Plotter.

www.elektormagazine.de/magazine/elektor-198801/48542



Surround-Sound-Dekoder (1995)

In einer Zeit, in der Surround-Sound-Geräte gerade auf den Markt kamen, entwickelte Elektor eine Selbstbau-Schaltung, die aus Lizenzgründen ohne spezielle Dolby-ICs auskommen mussten. Dennoch gelang es dem Elektor-Labor, mit konventionellen Mitteln einen Dekoder zu entwerfen, der den vorhandenen Stereoton um einen Mitten- und Surroundkanal erweiterte. Die Platinen waren im Nu ausverkauft!

www.elektormagazine.de/magazine/elektor-199502/30672

Jan Buiting

(Buch-Redakteur, früherer Chefredakteur der Englischen Ausgabe)

Sondernorm-Dekoder (1989)

Niemals in meiner 36-jährigen Tätigkeit bei Elektor habe ich mehr Begeisterung, Lob und allgemeine Resonanz aus der DIY-Elektronik-Community erfahren und genossen als bei der Veröffentlichung des „Sondernorm-Dekoders“ im Jahr 1989. Dieses geniale Projekt machte Elektor sofort zu einer führenden Quelle für „gebildete, gewitzte und raffinierte“ Methoden, wie man dank einiger Astra- und Intelsat-Satelliten europaweit empfangbare Pay-TV-Kanäle hackt. Allein in Skandinavien waren schätzungsweise 20.000 Elektor Sondernorm-Dekoder im Einsatz. „Nur für den privaten und experimentellen Gebrauch, Jan“ – aber klar! Und: „Unsere Winter sind lang und dunkel, Jan“ - natürlich! Sowohl die Entstehung wie auch die Veröffentlichung des Artikels war für Elektor höchst ungewöhnlich. Erstens wurde aus Angst vor rechtlichen Konsequenzen das Projekt in der niederländischen Elektuur gar nicht veröffentlicht, zweitens wurde die Platine, ebenfalls aus Angst vor Anwälten am Telefon oder an der Tür des Verlagsgebäudes nicht auf den damaligen EPS-Seiten (dem Vorläufer des Elektor-Store) gelistet. Drittens wurde in der deutschen Elektor-Ausgabe das Wort „Filmnet“ geflissentlich vermieden und schließlich wurde ein gewisser „P.N.P. Wintergreen“ als Autor genannt, den Literaturkenner natürlich sofort einem Protagonisten in Joseph Hellers Satire Catch-22 zuordnen konnten.

Was von den Elektor-Verlegern als kurioser Artikel angesehen wurde, löste in der Tat bei Tausenden von Bastlern in ganz Europa den Wunsch aus, den Dekoder zu bauen. Es sprach sich schnell herum, dass Filmnet von Elektor im großen Stil gehackt worden war. Die englische Ausgabe von Elektor war zum ersten Mal seit 1974 komplett ausverkauft und ging in eine zweite Auflage. In den Niederlanden wurden Stapel von illegalen Kopien des englischsprachigen Originalartikels von Elektronikhändlern verteilt. Die Damen der Elektor-Telefonzentrale lasen wochenlang einen vorbereiteten Text vor und reagierten damit auf eine Flut von Anrufen. Leerplatinen „aus unbekannter Quelle“ wurden illegal in Geschäften und auf großen Funkamateur-Treffen in Großbritannien, Deutschland und Holland verkauft.

Die Schaltung des Dekoders bestand, wie in den 1980er Jahren üblich, aus nichts anderem als Standardbauteilen, CMOS-ICs, dem TBA120S und einer Menge PNP-Transistoren. Keine Mikrocontroller oder anderer neumodischer Kram - die glorreichen Tage der von Elektor erfundenen „TUN & TUP“ lebten wieder auf.

Filmnet änderte in den Monaten nach der Elektor-Veröffentlichung mehrmals das Bildverschlüsselungsformat, wurde aber immer wieder durch raffinierte Updates des Elektor-Dekoders besiegt. Diese Tweaks durfte ich in einigen Folgen in der Reihe „From the Satellite TV Desk“ veröffentlichen. Nach drei Updates und passenden Texten für die Damen der Telefonzentrale entwarf P.N.P. Wintergreen eine geniale „selbstlernende“ Zusatzschaltung auf Basis eines Ein-Kondensator-Sample&Hold-Gliedes. Von da an bemerkten Besitzer eines voll aufgerüsteten Sondernorm-Dekoders nicht einmal mehr Änderungen im Scrambling-Modus. Es war ein sublimer Hack eines sehr teuren Scrambling-Systems, das von den Filmnet-Besitzern als „höchst problematisch zu hacken“ verkauft wurde. Später musste sich auch das SAVE-Scrambling des BBC World TV Service und mehrere DMAC/CMAC/Irdeto-Sat-TV-Sender dem Elektor-Hacking beugen - mit ebenso glorreichen Ergebnissen, massiver Leserresonanz und einem Gütefaktor, von dem man heute nur träumen kann.

Ich kenne den Entwickler persönlich, aber er soll anonym bleiben. Er ist allerdings ein echter Elektorist. „Problematisch? – Einen solchen Eintrag gibt es nicht in meinem Elektronik-Lexikon!“

www.elektormagazine.de/magazine/elektor-198904/48803

Luc Lemmens (Elektor-Labor)

Entwicklungshelfer (8052AH BASIC Mikrocontroller) (1987)
8052-/8032-Compuboard (1991)
8051-Mikrocontroller und Assembler-Kurs (1991)

Eines der ersten größeren Projekte, die ich im Elektor-Labor betreute, war der 8032/8052-Einplatinencomputer, der im Mai 1991 veröffentlicht wurde, als Nachfolger des beliebten 8052AH-BASIC-Entwicklungshelfers von 1987. Es war nicht sehr groß, was die Hardware anging, aber diesem Artikel folgte ein viel beachteter achtteiliger Kurs über Assembler mit diesem Board. Wer Assembler nicht lernen wollten, konnte auch Intels 8052AH-BASIC-Mikrocontroller auf diesem Board verwenden. Mit einem einfachen Programm konnte dessen interner BASIC-Interpreter (sein kompletter Programmspeicher) „illegal" auf externes EPROM kopiert werden.

www.elektormagazine.de/magazine/elektor-198711/48519
www.elektormagazine.de/magazine/elektor-199104/29538
www.elektormagazine.de/magazine/elektor-199109/29711

GBDSO (Gameboy Digital Sampling Oszilloscope) (2000)

Das in Oktober 2000 vorgestellte Nintendo Gameboy Digital Sampling Oscilloscope (GBDSO) war ein weiterer Volltreffer und eines der ersten komplett bestückten Module, die im Elektor-Store erhältlich waren. Ursprünglich wollten wir nur eine Charge produzieren, aber am Ende erwies sich die Nachfrage als viel größer, als wir erwartet hatten. Ich habe aufgehört zu zählen, wie viele „allerletzte Bestellungen" aufgegeben wurden. Offenbar gab es viele Leser, die noch einen unbenutzten Gameboy herumliegen hatten und ihm ein zweites, sinnvolleres Leben als portables Messgerät geben wollten.

www.elektormagazine.de/magazine/elektor-200010/1059

Geruchsgenerator mit CD 4711 (1986)

Die berühmte jährliche Halbleiterheft, die Doppelausgabe Juli/August, war stets gefüllt mit über hundert kleinen, oft bemerkenswerten Schaltungen, Tipps und Tricks. Ein Standard-Feature war die „Witzschaltung", von denen das programmierbare Geruchsgenerator-IC (CD4711) aus dem Jahr 1986 wohl eine der berühmtesten ist. Noch Jahre später erhielten wir Anfragen von Lesern, ob wir Kontaktdaten des Herstellers Odorant Elektronik GmbH in Köln hätten.

www.elektormagazine.de/magazine/elektor-198607/48220

Jens Nickel (Chefredakteur)

Embedded Linux leicht gemacht (2012)

Ganz stolz war ich 2012 als frischgebackener deutscher Chefredakteur: Gegenüber dem Vorjahr konnte ich die frei verkaufte Auflage um 20% nach oben schrauben. Aber es war nicht mein Verdienst, sondern einem Titelprojekt geschuldet, das seiner Zeit voraus war. Unser Autor Benedikt Sauter und sein Team hatten ein kompaktes Board entwickelt, auf dem Linux lief und das mit 50 € für die damalige Zeit unverschämt günstig war!

www.elektormagazine.de/magazine/elektor-201205/4069

Hier kommt der Bus! (2011)

In meinen ersten Jahren als Elektor-Redakteur war ich der einzige, der nicht bastelte. Meine „Berührungsängste" mit praktischer Elektronik mussten dringend abgebaut werden. Der internationale Chefredakteur Wisse Hettinga ließ mich deshalb eine Kolumne über die Jungs in unserem Labor schreiben. Es war ihm allerdings nicht klar, welche Lawine er damit lostrat! Da es im Labor gerade nicht viel Neues gab, orakelte ich bald über ein Bussystem, das für alle möglichen Steuerungen genutzt werden konnte. Das Feedback war überwältigend! Ich bekam Hunderte von Mails und bald war eine Gruppe von einem Dutzend Mitstreitern zusammen, um die Spezifikationen des ElektorBus festzulegen. Doch auch Beschimpfungen („dilettantisch", „Design aus den 60ern") blieben

natürlich nicht aus.
Später basierten etliche Projekte auf dem Bus, er wurde auch in einer Diplomarbeit genutzt. Ich bekam einen Anruf einer Haustechnik-Firma, die Module entwickeln wollte. Nur die Jungs aus unserem Labor verweigerten dem Projekt ihre Zuneigung. Und so schlief das Groß-Projekt - zum Leidwesen vieler Leser - irgendwann langsam ein.

www.elektormagazine.de/magazine/elektor-201101/3721

XMEGA-Multifunktions-Board (2013)

Meine Leidenschaft für das Entwickeln war freilich geweckt; zwei Jahre später konnte ich zusammen mit ein paar Profi-Ingenieuren ein eigenes Mikrocontrollerboard konzipieren. Wir packten einen XMEGA und ordentlich Peripherie drauf, doch am Ende wurde das Ganze im Verkauf rund 100 € teuer, das Display nicht einmal eingerechnet. Die Ingenieure schenkten uns ihre Entwicklungszeit und hatten für rund 1000 € Platinen und Prototypen angefertigt. Nicht einmal diese Kosten konnten wir wieder reinholen. Ein Flop! Und ich hatte mich softwaremäßig mächtig ins Zeug gelegt, mit der Embedded Firmware Library sogar eine Bibliothek geschaffen, die nicht nur von der Controller-Hardware, sondern auch von der Board-Verdrahtung abstrahierte. Einer unser Leser und ich hatten dabei eine Menge Programmierspaß; ich bekam Rückmeldungen von Lesern, die meinen Ansatz als „genial" bezeichneten. Wieder andere hielten das Konstrukt einfach nur für überflüssig. Doch am Ende zählt, dass ich selbst eine Menge hinzugelernt und vielleicht auch der Community ein paar Impulse gegeben habe - genau das macht ja Elektor aus!

www.elektormagazine.de/magazine/elektor-201310/23474
www.elektormagazine.de/magazine/elektor-201305/20573

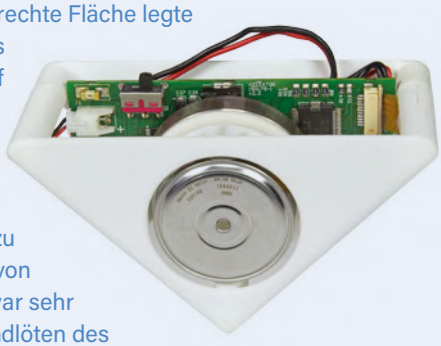
Clemens Valens (Elektor-Labor)

EPROM-Emulator für Centronics-Schnittstelle (1989)

Als ich mit dem begann, was man heute Embedded System Design nennt, stellte ich schnell fest, dass das Umprogrammieren von EPROMs ziemlich zeitaufwendig war. Als Elektor 1989 den EPROM-Emulator veröffentlichte, baute ich daher sofort einen. Das Hochladen von Firmware auf eine Zielplatine wurde so einfach wie das Kopieren einer Datei auf den parallelen Druckerport des PCs. Viel schneller als Arduino heute und ohne Bootloader, benutzten meine Kollegen und ich den Emulator jahrelang, bis die parallele Schnittstelle des PCs verschwand und Mikrocontroller mit eingebautem Flash-Speicher das Ruder übernahmen.

www.elektormagazine.de/magazine/elektor-198911/48961

I-Pendel (2016)



Das I-Pendel war vielleicht nicht das nützlichste, aber eines der interessantesten Projekte der Elektor-Geschichte. Wenn man das Pendel auf eine waagerechte Fläche legte und einschaltete, sprang es plötzlich auf, stellte sich auf die Spitze und verblieb in diesem Zustand. Meine Aufgabe war es, den Prototyp zu bauen und den Elektor-Artikel dazu zu schreiben. Der Entwurf von Jean-Sébastien Gonsette war sehr gut gemacht, aber das Handlöten des Motortreiber-ICs war die Hölle. Irgendwie schaffte ich es dann doch und hatte am Ende ein funktionierendes Gerät. Als ich es einem Freund zeigte, fiel er vor Lachen auf den Boden. Das war eine der erfreulichsten Reaktionen, die ich je auf etwas bekommen habe, das ich selbst gebaut habe.

www.elektormagazine.de/magazine/elektor-201604/28883

Halbleiterhefte

Früher kaufte ich Elektor nur, wenn mich darin wirklich etwas interessierte, mit Ausnahme des Halbleiter-Doppelhefts, das ich jedes Jahr mit geschlossenen Augen aus dem Kioskregal ergriff. Neben den über 100 Schaltungen mochte ich auch die Anzeigen,

vor allem die katalogartigen mehrseitigen, die hunderte von Bauteilen auflisteten. Ich kaufte das Halbleiterheft blind, bis 2007, als es zum ersten (und letzten) Mal nur ein Thema hatte: Robotik. 95% der Schaltungen betrafen Robotik und ich mochte keine davon. Im Jahr darauf begann ich, anstatt die Sommerausgabe zu kaufen, für Elektor zu arbeiten.

www.elektormagazine.de/magazine/elektor-198507
www.elektormagazine.de/magazine/elektor-200707

Formant (1976)

Zu jung, als er veröffentlicht wurde, zu arm, um mich ernsthaft dafür zu interessieren, bekam ich schließlich doch einen Formant-Musiksynthesizer in die Hände, als zwei Freunde zusammen einen fast funktionierenden kauften. Er landete schnell in meinem Schlafzimmer, wo ich ihn neu verkabelte, die Hüllkurvengeneratoren durch einen Entwurf aus einer konkurrierenden Elektronikzeitschrift ersetzte und mein brandneues selbstgebautes digitales Frequenzmessgerät benutzte, um alles zu kalibrieren. Zusammen mit meinen Freunden spielten wir die wiederbelebte Maschine viele, viele Stunden lang.

www.elektormagazine.de/magazine/elektor-197612/56155

Mathias Claußen (Elektor-Labor)

Logik-Analyser für Atari ST (1989)

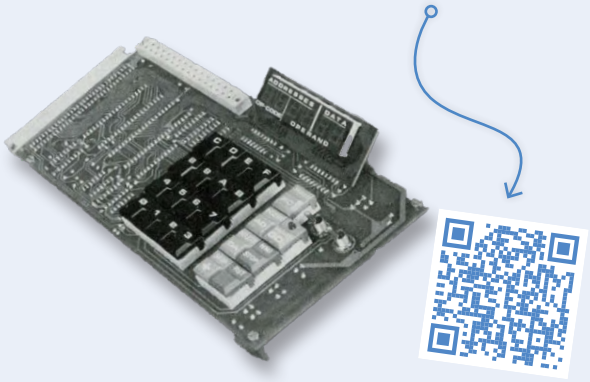
Mein erster Computer war ein Atari ST. Ich war beeindruckt zu sehen, dass Logikanalysatoren mit diesen Computern selbst gebaut werden konnten. Mit nur einem Haufen Logikchips und gut durchdachter Software wurde aus dem ST ein achtkanaliger Logikanalysator mit 2 MHz Abtastfrequenz. Dies wurde durch geschickte Nutzung der ST-Schnittstelle für externe Festplatten realisiert. Der Entwurf von 1989 wurde fast zwei Jahrzehnte später mit dem Logic 8 der Firma Saleae neu erfunden. Anstelle eines Festplatteninterfaces und einiger 74HC-Chips verwendete der Logic 8 einen Mikrocontroller und USB. Natürlich mit etwas besseren Spezifikationen für die Abtastrate und fortschrittlicher Software, aber das Prinzip blieb das gleiche.

www.elektormagazine.de/magazine/elektor-198909/48934

Junior-Computer (1980)

Um ein wenig in dieser Retro-Welle zu bleiben, mit dem MOS6502 und seinen Derivaten haben immer noch Entwickler auf der ganzen Welt Spaß und basteln an Code oder Systemen um diesen Chip herum. Der MOS6502 war die CPU vieler berühmter Systeme wie dem Nintendo NES, Atari 800, Apple II und in einer modifizierten Version im C64. 1980 veröffentlichte Elektor den Junior-Computer, ein 6502-basiertes Gerät, das den Einstieg in die Welt der Mikroprozessoren ermöglichte. Mit 1 MHz Taktfrequenz, 1 KB RAM und 1 KB ROM war er zwar langsamer und weniger leistungsfähig als ein heutiger Arduino auf ATmega-Basis, aber damals konnte man so etwas noch selbst bauen und reparieren. Drei Jahrzehnte alte 6502-basierte Systeme gibt es immer noch, und mit dem Commander X16 von David Murray ist sogar ein neuer Entwurf auf Basis dieser CPU auf dem Weg. Und wenn Sie Ihren eigenen 6502-basierten Computer bauen wollen: Die CPUs werden auch heute noch produziert!

www.elektormagazine.de/magazine/elektor-198005/46018



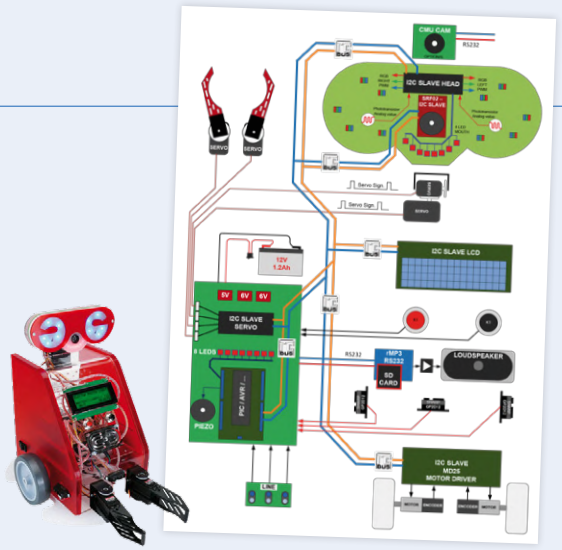
E-Lock: Der erste Chip von Elektor (2014)

Apropos Mikrochips: Auch Elektor hat einen eigenen Chip produziert. Im Jahr 2014 erschien der E-Lock-Chip, der kabelgebundenes Ethernet, ADC, I²C, DAC und einige GPIOs mit sicherer Kommunikation und Verschlüsselung kombinierte. Ausgestattet mit Hardware-Verschlüsselung und Sicherheitselementen war er für das kommende IoT gedacht. Dies ist eines der kuriosesten Elektor-Projekte, da einige Interna dieses Chips auch für uns ein Geheimnis blieben. Wir werden vielleicht nie erfahren, welcher Kern verwendet wurde und wie die internen Register funktionierten. Neuere Chips wie der ESP8266 oder einige von WIZnet haben den Platz von E-Lock eingenommen. Kabelgebundenes Ethernet hat immer noch seinen Platz, aber die Anwender verlangen mehr und mehr drahtlose IoT-Lösungen.

www.elektormagazine.de/magazine/elektor-201404/26199

Elektor Proton Roboter (2011)

Etwas, das lange im Büro herumstand und leider zu viel Staub angesetzt hat, ist der Elektor Proton, ein selbstfahrender Roboter, entwickelt als Lernplattform für Studenten. Mit einem modularen Design, das auf einem I²C-Bus basiert, sollte er einen schnellen und einfachen Einstieg in die Welt der Robotik ermöglichen, mit einer ganzen Reihe von Funktionen, die es Anfängern und Profis leicht machen sollten, ihn für eigene Anwendungen zu nutzen. Da nun ein

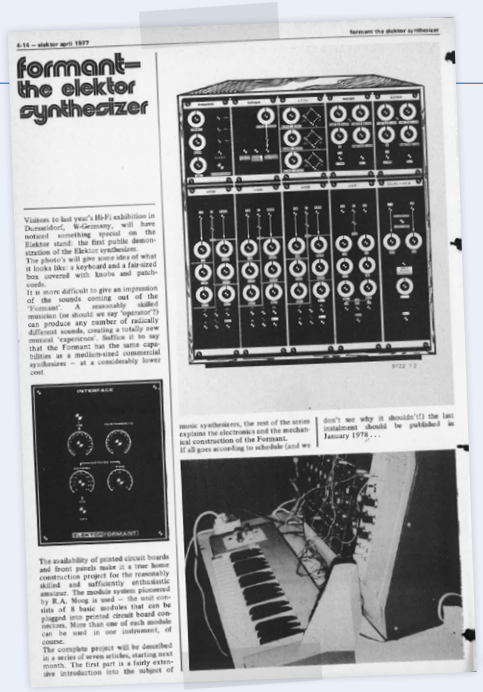


Jahrzehnt seit der ersten Veröffentlichung im Jahr 2011 vergangen ist, könnte der im Büro verbliebene Proton eine Überholung mit modernen Sensoren, Computer-Vision und einer neuen leistungsstarken, vielleicht WLAN-fähigen MCU vertragen. Dank des verwendeten I²C-Busses sollte das aber kein Problem sein.

www.elektormagazine.de/magazine/elektor-201105/3796

C. J. Abate (Content Director)

Viele von Ihnen sind wahrscheinlich schon den Tiefen von YouTube verschwunden: Sie fangen nach dem Abendessen an, das GitHub-Video von Clemens Valens (http://bit.ly/elektor-github) auf dem ElektorTV-Kanal anzuschauen und merken dann zwei Stunden später, dass Sie in ein Video über den Bau einer netzunabhängigen Hütte in Alaska vertieft sind. Aber sind Sie jemals in die Tiefen des Elektor-Archivs hinabgestiegen? Ich habe zwar nicht jedes Elektor-Heft zu Hause, aber ich habe Zugriff auf das digitale Archiv. In den letzten Jahren habe ich unzählige Stunden damit verbracht, in den PDFs zu blättern und zu klicken. Eines Abends, nachdem ich ein Video über den Elektor-Formant auf YouTube gesehen hatte, begann ich im Elektor-Archiv den ersten Artikel über dieses beliebte Musiksynthesizer-Projekt zu finden. Neunzig Minuten später fand ich mich in dem Artikel „Radar-Detektor“ vom März 1991 wieder. Wenn man Jahrzehnte an Inhalten zur Verfügung hat, ist es schwer, Favoriten auszuwählen. Hier sind also einige Artikel, die ich sehr empfehlen kann, weil sie praktische Einführungen in ein paar Themen bieten, die ich wirklich interessant finde.



Formant - Der Elektor Musiksynthesizer (1976)

Dieses Projekt wurde lange vor meiner Zeit entwickelt. Da ich aber schon öfter davon gehört hattee, habe ich mich kürzlich entschlossen, ein wenig zu recherchieren. Natürlich kann man auch heute noch Leute finden, die mit dem Formant auf YouTube Musik machen!

www.elektormagazine.de/magazine/elektor-197612/56155

Bauen Sie Ihren Chip! Leistungsstarker FPGA sicher im Griff (2012)

Vor einigen Jahren reisten der langjährige Elektor-Redakteur Jan Buiting und ich nach Kalifornien, um Xilinx zu besuchen. Während wir uns mit einigen Ingenieuren trafen und die Anlage besichtigten, wurde mir schnell klar, wie wenig ich über FPGA-Technologie wusste. Der Artikel „Bauen Sie Ihren Chip“ ist ein hilfreicher Startpunkt.

www.elektormagazine.de/magazine/elektor-201212/4213

Elektronenröhren (1984)

Wer ist nicht fasziniert von Röhren? Sie wissen schon, diese zerbrechlichen Glasdinger mit allen möglichen kompliziert aussehenden Teilen darin. Dieser Artikel ist ein Klassiker von Elektor. Egal, ob Sie ein Elektronikus der alten Schule sind oder ein Fan von modernen Entwürfen, die High-End-Elektronik mit cool aussehenden Retro-Röhren mischen, Sie werden diesen Artikel lieben.

www.elektormagazine.de/magazine/elektor-198411/47902

Michel Kuenemann (Autor)

Mitte der 80er Jahre entdeckte ich Elektor und war erstaunt über die Qualität der Projekte. Ich baute den 1985 erschienenen Frequenzzähler auf 6502-Basis, der immer noch funktioniert. Während meines Studiums der Elektrotechnik war ich in einem Amateurraketen-Verein engagiert. Wir planten, die Veränderung des atmosphärischen Drucks während des Aufstiegs der Rakete zu messen. Elektor hatte ein sehr ordentliches Projekt mit einem Drucksensor und mehreren Operationsverstärkern veröffentlicht. Die bestückte Platine wurde erfolgreich in die Rakete integriert. Nach meinem Studienabschluss gründete ich eine Firma und Elektor war über all die Jahre hinweg eine erstklassige Inspirationsquelle für mich. Dank Elektor entdeckte ich den I²C-Bus und viele andere Technologien. Später konnte ich mit mehreren Artikeln zum Thema Modellbau zur Zeitschrift beitragen, wie einen Prüfstand für Modellbau-Verbrennungsmotoren (April 2009) und einer ZigBee-basierten Fernsteuerung (Oktober 2011). Ich bin jetzt 58 Jahre alt, und Elektor hat immer noch einen großen Einfluss auf mich. Herzlichen Glückwunsch zum 60. Geburtstag von Elektor und vielen Dank an das Team und die Autoren für all die guten Sachen!

Artikel von Michel Kuenemann:
www.elektormagazine.de/authors/171/michel-kuenemann

Andrew Pratt (Autor)

Ich habe Elektor erstmals in den 1970er Jahren gelesen, lange vor dem Internet. Technische Zeitschriften vermittelten Bildung und enthielten nützliche Werbung über Bauteile und Geräte. Heutzutage bietet das Internet fast unbegrenzte Informationen, wozu brauchen wir also noch Zeitschriften? Websites sind bequem, aber physische Medien haben viele Vorteile. Elektor kann beides! Die Website ordnet die Informationen an einem Ort an, so dass

Sie finden können, was Sie brauchen. Ich habe drei Bücher bei Elektor veröffentlicht. Ein Buch oder ein Artikel zusammen mit einigen praktischen Übungen ist ein guter Weg, um zu lernen, wie die Dinge funktionieren. Man kann immer experimentieren, auch wenn man kein vollständiges Projekt plant. Es gibt auch die Möglichkeit für jeden, ein Projekt auf Elektor-Labs hochzuladen, damit andere Ihre Arbeit kopieren oder verbessern können. Das Prinzip der Open-Source-Software war ein spektakulärer Erfolg, Elektor macht dies auch für die Elektronik-Hardware.

Bücher von Andrew Pratt:
www.elektor.de/catalogsearch/result/?q=Andrew%20Pratt

GreatScott! (Autor, YouTuber)

Bei mir hat es eine ganze Weile gedauert, bis ich Elektor entdeckte. Nachdem ich bereits viele Jahre YouTube-Videos über Elektronik produziert hatte, fiel mir eines Tages eine Empfehlung eines Zuschauers ins Auge. Dabei handelte es sich um ein DIY-Kit von Elektor. Durch den Weblink kam ich auf die Elektor-Website, wo ich mich mehrere Stunden umsah und bemerkte, dass die angesprochenen Themen mit meinen eigenen Elektronik-Videos einige Gemeinsamkeiten aufwiesen. Wenige Monate später wurde ich von Elektor kontaktiert, ob wir zusammen nicht einige Videoproduktionen umsetzen können - der Rest ist Geschichte. Ich wurde Elektor-Leser und produziere in Zusammenarbeit mit Elektor Videos über ganz verschiedene Elektronik-Themen. Eine Win-Win-Situation für alle!

Freier Online-Artikel über das DIY-LiPo-Supercharger-Kit von GreatScott!:
www.elektormagazine.com/articles/diy-lipo-supercharger-kit

Tischmultimeter SDM3045X von Siglent

Genauer, mehr Stellen und besser bedienbar

Von **Harry Baggen**

Jeder Elektroniker besitzt ein oder mehrere Multimeter. Aber haben Sie schon einmal darüber nachgedacht, ein Tischmultimeter anstelle eines Handmultimeters zu kaufen? Diese bieten nämlich mehr Funktionen und eine bessere Anzeige. Das Siglent SDM3045X ist ein 4½-stelliges Tischmultimeter mit einer Grundgenauigkeit von 0,02% und vielen Schnittstellen. Hier mein Eindruck, nachdem ich es ein paar Monate lang getestet habe.

Jeder, der etwas mit Elektronik zu tun hat, hat mindestens eines – oft auch mehrere Multimeter. Man bekommt sie für wenig Geld, doch für größere Robustheit, bessere Sicherheit, mehr Funktionen und höhere Genauigkeit steigen die Beträge schnell auf einige hundert Euro. Vor allem die Genauigkeit hat einen großen Einfluss auf den Preis.

Zu Hause hatte ich immer ein einfaches Multimeter. Letztes Jahr wurde es nach jahrzehntelangem, treuem Dienst endlich durch eine bessere Version mit Dual-Display und vor allem einer besseren Genauigkeit ersetzt. Ich war sehr zufrieden damit. Beim Test eines Tischmultimeters von Siglent kam ich auf die Idee, es mit meinem Handmultimeter zu vergleichen. Das Siglent ist in der Anschaffung zwar teurer, aber vielleicht ist es dieses Geld auch wert. Lesen Sie, welches Messgerät einfacher zu bedienen ist und was die Vor- und Nachteile der einzelnen Typen sind.

Das Messgerät

Dies ist das „kleinste“ Tischmultimeter von Siglent aus einer Serie von drei Geräten, aber abgesehen von einer geringeren Anzahl von Ziffern und einer geringeren Grundgenauigkeit bietet das Modell SDM3045X immer noch den gleichen Funktionsumfang wie die teureren Versionen. Wenn seine Genauigkeit ausreicht, bekommt man damit das beste Preis-/Leistungsverhältnis.

Das Modell SDM3045X hat ein stabiles Metallgehäuse mit Schutzpuffern aus Kunststoff an den Ecken. Der Tragegriff lässt sich in verschiedene Positionen drehen, so dass man die Vorderseite des Messgeräts schräg nach oben kippen kann. Das Gehäuse hat die gleichen Abmessungen wie viele der anderen Geräte von Siglent (z.B. die Signalgeneratoren), so dass sie sich leicht stapeln lassen.

Vorne befindet sich ein übersichtliches 4,3“-Farb-Display. Darunter liegen sechs Softmenü-Tasten, deren Funktionen im Display angezeigt werden. Auf der rechten Seite sind die üblichen Funktionstasten (von denen die meisten eine sekundäre Funktion haben) sowie eine Reihe von Cursor-Tasten für die Navigation durch die Menüs und die Einstellung von Werten.

Es gibt fünf Eingangsbuchsen (**Bild 1**): zwei für die normalen Eingänge, zwei Sense-Eingänge für Vierdraht-Widerstandsmessungen und einen Eingang für Strommessungen. Auf der Rückseite befinden sich eine IEC-Netzbuchse, ein USB-Anschluss und ein LAN-Anschluss für die Verbindung mit einem Computer

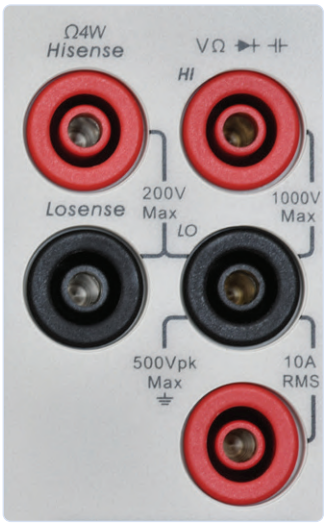


Bild 1. Es gibt Eingänge für Vierdraht-Widerstandsmessungen.

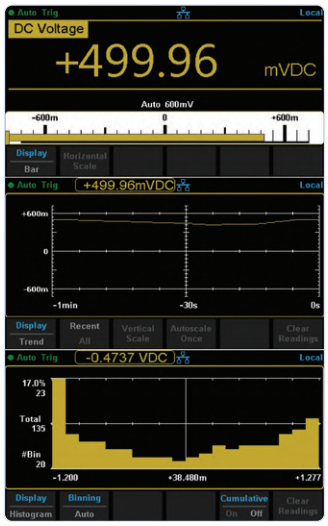


Bild 2. Eine Auswahl von Anzeigemöglichkeiten.

Bild 3. Das Messgerät kann über USB oder LAN mit einem PC verbunden werden.

oder die Integration in ein LAN. Hinzu kommen zwei BNC-Buchsen für die externe Triggerung. Außerdem gibt es einen Sicherungshalter für die Strommessung.

Messfunktionen

Das SDM3045X bietet natürlich alle Funktionen, die man auch bei einem guten Handmultimeter findet: Spannungs- und Strombereiche, Widerstand, Kapazität, Frequenz, dB(m), Temperatur- und Dioden- sowie Durchgangsmessung, Auto-Ranging und - wie bei den besseren Multimetern inzwischen üblich – eine Möglichkeit zur gleichzeitigen Anzeige von zwei Messwerten (z.B. Wechselspannung und Frequenz). Das Display zeigt Werte bis maximal 66.000 an. Die Grundgenauigkeit beträgt 0,02 % bei Gleichspannung. Es kann bis zu 150 Messungen/s durchführen, ist aber nicht für den

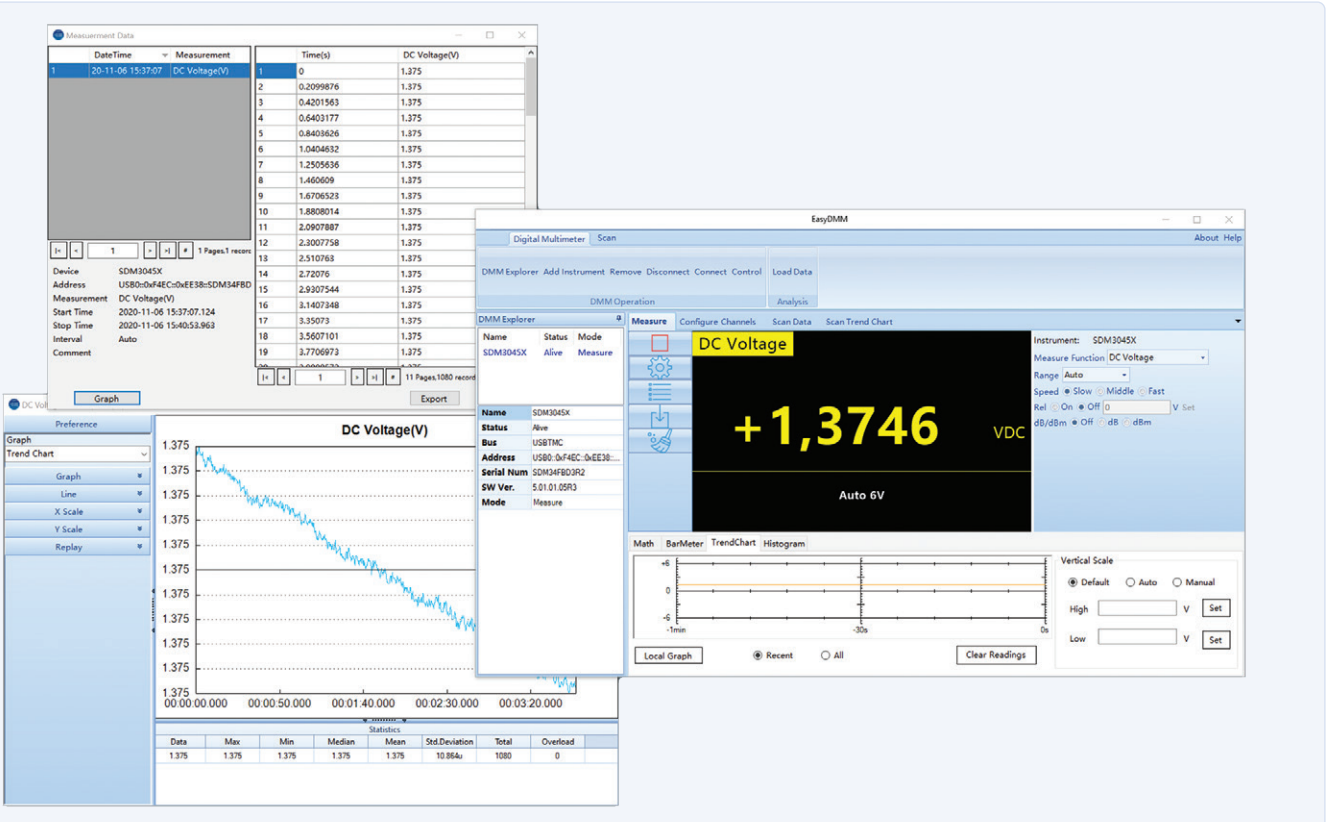


Bild 4. Die kostenlose Software EasyDMM zur Steuerung des Messgeräts und zum Auslesen der Daten am PC.

Einsatz bei hohen Spannungen (CAT I / 10.000 V bzw. CAT II / 300 V) ausgelegt. Für diese Zwecke sollte man ein Tischmessgerät in einem Elektroniklabor aber ohnehin nicht verwenden. Das Multimeter kann per LAN oder USB an einen PC angeschlossen werden, der dann die Messergebnisse speichert.
Bei Spannungsmessungen gibt es verschiedene Darstellungsmöglichkeiten wie z.B. eine zusätzlichen Balkenanzeige unterhalb des digitalen Wertes, eine Trendkurve oder ein Histogramm (**Bild 2**). Außerdem stehen verschiedene mathematische Funktionen zur Verfügung. Für viele Messgerätefunktionen hat man Optionen zur Auswahl, die sich in diesem Review nicht alle aufzählen lassen. Ein paar Beispiele: Im 600-mV-Bereich lässt sich zwischen einer Eingangsimpedanz von 10 MΩ oder 10 GΩ wählen, beim Diodentest lässt sich die Prüfspannung zwischen 0 V und 4 V einstellen, bei Temperaturmessungen kann man je nach Art des verwendeten Temperatursensors aus verschiedenen Konfigurationsmöglichkeiten wählen. Das Messgerät lässt sich manuell oder extern triggern, so dass es eine bestimmte Anzahl von Messungen durchführt. Außerdem können obere und untere Schwellwerte festgelegt werden, deren Über- bzw. Unterschreitung geprüft wird. Der interne Flash-Speicher von 1 GB fasst bis zu 10.000 Messergebnisse. Über den USB-Anschluss an der Vorderseite lassen sich Konfigurationseinstellungen und Messergebnisse auf einem USB-Stick speichern. Die Verbindung des Messgeräts mit einem PC ist über USB oder ein LAN möglich (**Bild 3**). Die mitgelieferte Software EasyDMM (**Bild 4**) ist für Windows gedacht, was für die meisten Elektroniker kein Problem darstellen dürfte. Diese Software unterstützt die Steuerung des Messgeräts und auch das Abspeichern von Messsequenzen. Wie die meisten anderen Siglent-Messgeräte verwendet auch dieses Standard-SCPI-Befehle, mit denen sich eigene Messabläufe zusammenstellen lassen.

In der Praxis

Der größte Unterschied zwischen diesem Tischmultimeter und einem Handmultimeter besteht wohl darin, dass ersteres vom Stromnetz versorgt wird. Bei stationärer Verwendung schützt das vor leeren Batterien. Platziert man das SDM3045X an geeigneter Stelle auf einem Labortisch, bleibt es normalerweise einfach dort, neben einem Oszilloskop, dem Labornetzteil und anderen Messgeräten. Ich habe überlegt, wie oft ich mein Handmultimeter an einen anderen Ort getragen habe. Tatsächlich war das nur sehr selten der Fall. Die meiste Zeit lag es auf meinem Labortisch, da ich genau da ein Multimeter brauche. Seit das SDM3045X auf meinem Labortisch stand (**Bild 5**), habe ich mein Handmultimeter kaum noch benutzt (außer wenn ich zwei Messungen gleichzeitig machen wollte). Das 3045X steht viel stabiler auf dem Tisch als ein Handmultimeter und mit seiner ausklappbaren Halterung fand ich die Bedienung der Drucktasten viel bequemer als einen Drehschalter.
Es dauert ein wenig, bis man ich mich an die Bedienung und alle Funktionen gewöhnt hatte, vor allem weil es so viele Konfigurationsmöglichkeiten gibt. Und wenn man es einschaltet, muss man ein wenig Geduld haben, bis der „Computer“ gebootet hat - deutlich anders als bei einem Handmultimeter. Ansonsten gibt es nur Vorteile: Ein großes und übersichtliches Display und so ziemlich alle Funktionen, die man sich für ein Multimeter vorstellen kann. Die Genauigkeit ist fantastisch und bei Wechselspannungsmessungen ist auch der große Frequenzbereich bis 100 kHz ein Bonus, denn die meisten Handmultimeter packen nicht einmal den Audiobereich.

Wer höhere Genauigkeit benötigt, kann zum teureren Geschwistermodell SDM3055 greifen, das eine zusätzliche Stelle bietet. Ein SDM3045X besitzt allerdings im Gegensatz zum SDM3055 keinen Lüfter - ein großer Vorteil, weil es mucksmäuschenstill ist. Die mitgelieferte Software ist übersichtlich und ideal für die Protokollierung der Messergebnisse. Ich habe seit diesem Test mein Handmultimeter kaum noch benutzt, denn das Siglent bietet so viel mehr!

Fazit

Wenn Sie bereit sind, etwas mehr für ein gutes Multimeter auszugeben, dann kann ich Ihnen auf jeden Fall empfehlen, sich die Features des SDM3045X bzw. sein Datenblatt anzusehen. Dieses Tischmultimeter ist ein hervorragender Begleiter sowohl für Profis als auch Amateure, denn es kann nicht nur viel mehr als ein Handmultimeter, sondern ist meines Erachtens auch einfacher zu bedienen und abzulesen.

200720-04

Bild 5. Das SDM3045X fühlt sich zwischen den anderen Laborgeräten wohl.

Sie haben Fragen oder Kommentare?
Gerne können Sie sich an die Elektor-Redaktion wenden unter der E-Mail-Adresse: redaktion@elektor.de.

Ein Beitrag von
Review und Text: **Harry Baggen**
Redaktion: **Jens Nickel**
Übersetzung: **Dr. Thomas Scherer**
Layout: **Giel Dols**

PASSENDE PRODUKTE

› **Siglent SDM3045X Multimeter**
www.elektor.de/siglent-sdm3045x-multimeter

DC-Stromzange selbst gebaut

Hall-Sensor + Ferrit-Kerne + Arduino

Von **Prof. Dr. Martin Oßmann**

Will man Ströme in einer Leitung messen, muss man diese üblicherweise auftrennen, um dann ein Amperemeter einzuschleifen. Mit einer Stromzange kann man das Auftrennen vermeiden. Einfache Stromzangen sind allerdings nur für Wechselstrom geeignet, denn sie arbeiten im Prinzip wie ein Trafo. Will man eine für Gleichstrom geeignete Stromzange bauen, kommen Hall-Sensoren zum Einsatz.

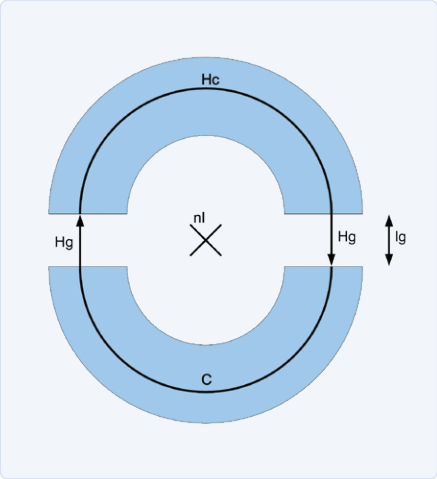


Bild 1. Geometrie des Magnetfelds.

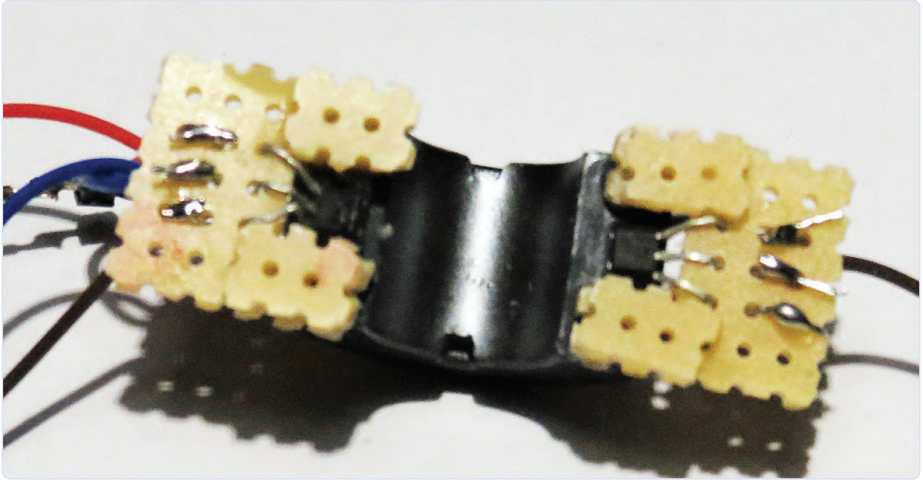


Bild 2. Anordnung der Hallelemente am Kern.

Bevor man ein solches Projekt angeht sollte man sich natürlich Gedanken machen, wie ein solches Konzept realisierbar ist. Hierzu kann man sich an den Aufbau einer „fertigen" Stromzange anlehnen. Eine Stromzange besteht im Prinzip immer aus zwei Ferritkernhälften. Der Kern ist dabei so zusammengesetzt, dass die beiden Häften einen Luftspalt aufweisen (**Bild 1**). Durch einen vom Kern umschlossenen Leiter fließt dann der Strom *nI*.

Mathematische Zusammenhänge

Dabei entspricht der Index *n* der Windungszahl (die angibt, wie oft der Leiter um den Kern gewickelt ist) und damit dem Multiplikationsfaktor, mit dem das Magnetfeld des Messstroms *I* den Kern magnetisiert. Die magnetische Flussdichte im Kern B_c (der Index c steht für Core = Kern) ist die gleiche wie im Luftspalt B_g (der Index $_g$ steht für Gap = Spalt) – es gilt daher $B_g = B_c$. Dies ist eine Folgerung aus den Maxwellschen Gleichungen. Vereinfachend geht man davon aus, dass die Magnetfelder in Kern und Spalt homogen sind. Im Luftspalt gilt die Materialgleichung $B_g = \mu_0 \times H_g$. Im Kern gilt entsprechend $B_c = \mu_r \times \mu_0 \times H_c$. Die Permeabilität μ_r typischer Ferrit-Materialien liegt bei etwa 2.000. Daraus ergibt sich $H_g = \mu_r \times H_c$, d.h. im Kern ist die magnetische Erregung um den Faktor μ_r kleiner als im Luftspalt. Praktisch kann man daher die Erregung im Kern oft vernachlässigen.

Das amperesches Gesetz sagt nun, dass das Wegintegral über *H* entlang einer Kurve um den Strom *nI* gleich *nI* ist. In **Bild 1** ist die Kurve mit *C* bezeichnet. Der Luftspalt hat hier die Länge $l_g = 1{,}6\ mm$ und die Länge in einer Kernhälfte ist $l_c = \pi \times d_c / 2$ mit dem Kerndurchmesser $d_c = 18\ mm$.
Das amperesches Gesetz liefert also:
$2 \times l_g \times H_g + 2 \times l_c \times H_c = nI$

Setzt man den Wert von H_c ein und löst nach H_g auf, erhält man:

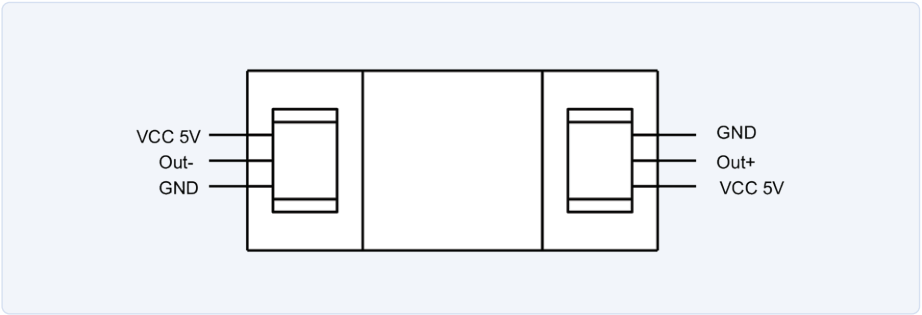


Bild 3. Anschlüsse der Hallelemente.

Bild 4. Mehrfaches Umwickeln des Kerns erhöht die Empfindlichkeit

$H_g = nI / (2 \times l_g + 2 \times l_c / \mu_r)$

Damit kann man das von den Sensoren erfasste Magnetfeld in Abhängigkeit vom Strom berechnen. In diesem Projekt werden zwei Hall-Sensoren des Typs A 1324 LUA-T mit einer Empfindlichkeit von $S = 5\ mV / G$ verwendet. Damit ergibt sich an den Sensoren eine Spannung von:

$U = 2 \times S \times \mu_0 \times nI / (2 \times l_g + 2 \times l_c / \mu_r)$

Wenn man das numerisch auswertet, ergibt sich:

$U = I \times 38{,}92\ mV / A$

Für übliche Ströme erhält man also Spannungen im Millivoltbereich, was nicht gerade viel ist. Vernachlässigt man das Feld im Ferrit, erhält man die vereinfachte Formel:

$U = S \times \mu_0 \times nI / l_g$

Danach ergibt sich mit $U = I \times 39{,}26\ mV / A$ fast der gleiche Skalierungsfaktor wie zuvor.

Der Ferrit-Kern

Der Kern der DC-Stromzange besteht aus zwei halbkreisförmigen Ferriten die einem EMV-Klappferrit entnommen wurden (**Bild 2**). Auf eine Hälfte wurden die zwei Hall-Sensoren aufgeklebt. Neben die Sensoren wurden 1,6 mm dicke Abstandshalter aus Pertinax-Platinenmaterial (ohne Kupfer) geklebt. Die Anordnung der Hall-Sensoren ist in der Skizze von **Bild 3** zu sehen.

Die zweite Kernhälfte kann man dann z.B. mit Klebeband oder einem Gummiring fixieren. Durch das Loch fädelt man den Leiter mit dem Messstrom. Wie man in **Bild 4** sieht, kann man eine Litze auch mehrfach um die Kernhälfte wickeln und so linear die Empfindlichkeit erhöhen.

Die Elektronik

Als Analog/Digital-Wandler kommt ein MCP3421 zum Einsatz. Dabei handelt es sich um einen sehr preiswerten 18-bit-Wandler. Er verfügt über eine interne Spannungsreferenz mit 2,048 V und hat einen differentiellen Eingang. Damit kann man die beiden Hall-Sensoren so anschließen, dass sich ihre Spannungen addieren. Die Auflösung des Wandlers ist mit $2{,}048\ V / 2^{17} = 0{,}015\ mV$ für diesen Zweck voll ausreichend.

Als CPU dient der Mikrocontroller eines Arduino-Nano. Dieses Board verfügt über genügend Anschlussmöglichkeiten für diese Anwendung. Als Anzeige dient ein OLED-Display mit 128 x 64 Pixeln. Die relativ einfache Gesamtschaltung ist in **Bild 5** zu sehen. Will man eine analoge Anzeige, kann man auch ein Drehspul-Voltmeter anschließen. Ein Volt entspricht dann einem Ampère.

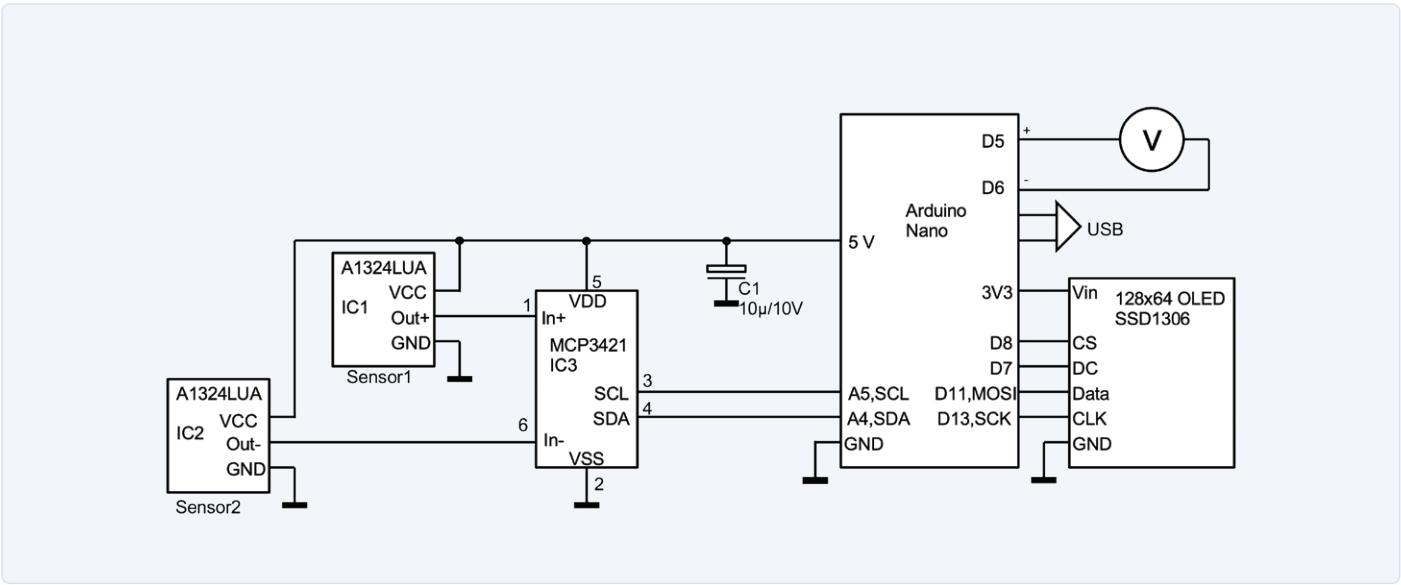


Bild 5. Vollständige Schaltung der Stromzange.

Befehle

Zur Bedienung und zum Setzen der Skalierungsparameter der Software des Autors kann man einige Kommandos über die serielle Schnittstelle eingeben. Über die serielle Schnittstelle werden auch die Messwerte angeboten.

Kommando „0“ = Nullabgleich

Dieser Befehl sollte nur angewendet werden, wenn gerade kein Messstrom durch die Stromzange fließt. Die Software führt dann den Nullabgleich durch und merkt sich den Offset im nichtflüchtigen EEPROM. Die Hall-Sensoren haben Offsets, die man für korrekte Messungen auf diese Weise kompensieren muss.

Kommando „1“
= Skalierungsfaktor für 1 A

Zur Einstellung des Skalierungsfaktors lässt man einen Messstrom von *nI = 1 A* durch die Zange fließen und führt dabei das Kommando „1“ aus. Dann wird der Skalierungsfaktor ermittelt und im EEPROM abgelegt.

Kommando „5“
= Skalierungsfaktor für 500 mA

Analog zum Kommando „1“ dienen hier 0,5 A als Referenzstrom.

Kommando „u“

Durch Eingabe eines Wertes nach dem „u“ wird dieser Wert als Spannung ausgegeben und angezeigt. Die Stromzange geht dann für eine gewisse Zeit in den „DVM-Kalibriermodus“. Mit „+“ und „-“ kann man die Ausgabe auf den Sollwert kalibrieren.

Kommando „d“

Es werden Default-Werte für die Parameter ins EEPROM geschrieben, um dieses für einen ersten Start zu initialisieren.

Aufbau

Um zu demonstrieren, dass man mit wenig Mitteln und begrenzter Hardware eine funktionierende Stromzange realisieren kann, habe ich diesen Hack „schnell mal“ auf einem Steckbrett aufgebaut (**Bild 6**). Man erkennt den Kern im Vordergrund, links das kleine Display und recht das Arduino-Nano-Board. Mit der in der Arduino-IDE für das Nano-Board zurechtgetüftelten Software – die übrigens kostenlos von der Elektor-Webseite zu diesem Artikel [1] heruntergeladen werden kann – lief diese selbstgebaute Stromzange auf Anhieb. Vielleicht lassen Sie sich durch diese Schaltung zu eigenen Experimenten oder Projekten rund um die magnetische Messung von Strömen inspirieren!

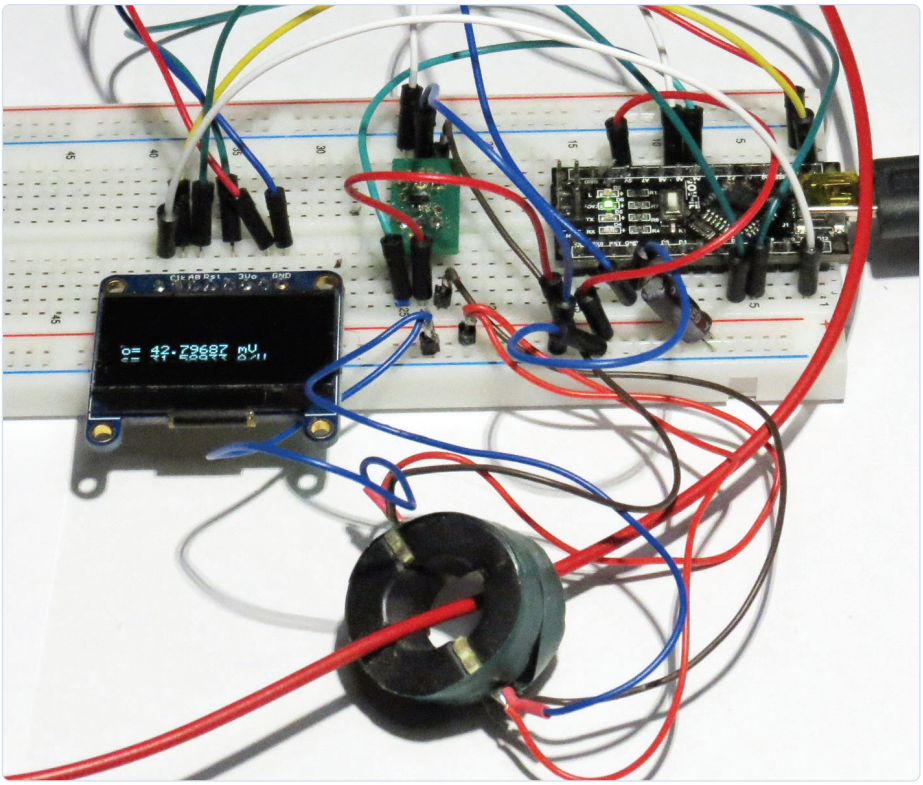


Bild 6. Prototyp auf Steckbrett.

200595-02

Sie haben Fragen oder Kommentare?
Gerne können Sie sich an die Elektor-Redaktion wenden unter der E-Mail-Adresse: redaktion@elektor.de.

Ein Beitrag von
Projekt und Text:
Prof. Dr. Martin Oßmann
Redaktion: **Dr. Thomas Scherer**
Layout: **Giel Dols**

WEBLINK

[1] **Artikel-Webseite:** www.elektormagazine.de/200595-02

PASSENDE PRODUKTE

- **0,96 Zoll 128x64 OLED-Display für Arduino**
  www.elektor.de/blue-0-96-oled-display-spi-6-pin
- **PeakTech Stromzange 4350**
  www.elektor.de/peaktech-4350-clamp-meter

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

Temperaturgesteuerte Lötstation 2021

Einfach selbstgebaut!

Von **Luc Lemmens** und **Mathias Claußen** (Elektor)

Obwohl fertige Lötstationen in Hülle und Fülle im Angebot sind, gibt es immer noch Leute, die diese Geräte am liebsten selbst bauen. Die DIY-Lötstation für Weller-RT-Lötkolben aus Elektor 10/2018 [1] hat sich bei unseren Lesern als sehr beliebt erwiesen. Aufgrund der Rückmeldungen zu diesem Projekt haben wir uns zu einem neuen Design für Weller RT entschlossen, das auch für die Lötkolben Hakko FX-8801 und JBC T245 geeignet ist.

INFOS ZUM PROJEKT

Tags
Werkzeuge, Labor, Löten, Weller, Hakko, JBC

Level
Einsteiger – **Fortgeschrittene** – Experte

Zeit
Etwa 4 Stunden

Werkzeuge
Lötwerkzeuge, mechanische Werkzeuge, 3D-Drucker (optional)

Kosten
ca. 70 €

Gleich das erste Kriterium des Entwurfs war uns sehr wichtig, nämlich die Gesamtkosten so niedrig wie möglich zu halten. Nichts Ausgefallenes, kein Zierrat, nur eine gut funktionierende, robuste Lötstation sollte es werden. Es mag im SMD-Zeitalter altmodisch wirken, aber wenn es ums Selberlöten geht, bevorzugen die meisten Maker immer noch bedrahtete Bauteile, und obwohl es immer schwieriger wird, solche Durchsteck-Bauteile aufzutreiben (insbesondere integrierte Schaltkreise), haben wir es geschafft, den gesamten Entwurf SMD-frei zu halten. Die Elektronik der Lötstation ist auf zwei Platinen verteilt, einer Haupt- und einer Panel-Platine, die hinter der Frontplatte montiert wird. Die Verbindung zwischen den Platinen beschränkt sich auf ein einziges Flachbandkabel. Hirn der Schaltung ist ein neuer AVR-Mikrocontroller von Microchip, dessen Firmware in der Arduino-IDE entworfen werden kann. Diese Entwicklungsplattform macht eine einfache Modifikation der Software für jedermann möglich.

Auf die kompakte Bauweise der „alten" Lötstation mit einer externen, handelsüblichen

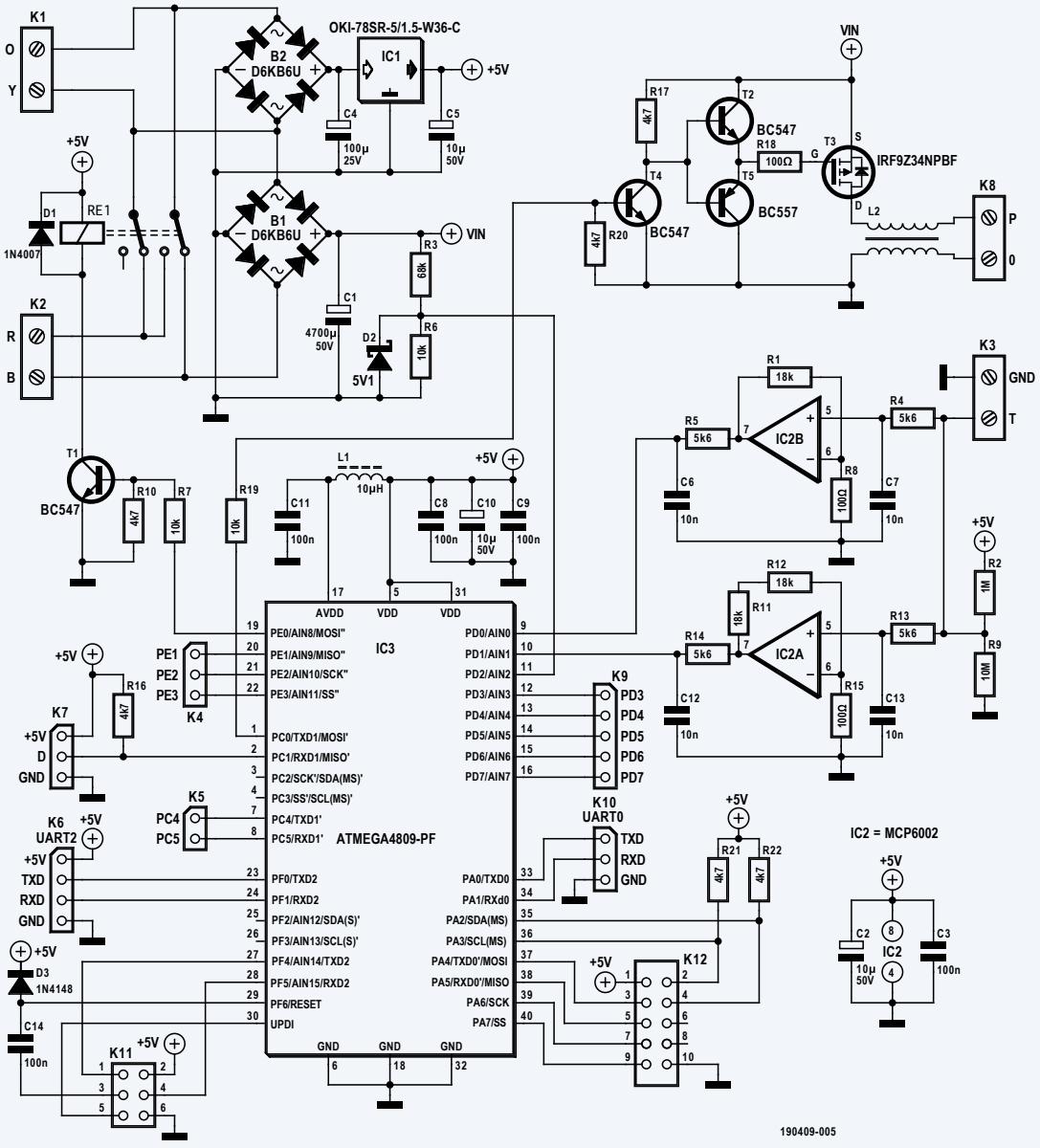


Bild 1. Die Schaltung der Hauptplatine.

12-VDC-Stromversorgung waren wir eigentlich ziemlich stolz, allerdings beschwerten sich viele unserer Leser, dass die Lötstation so viel zu klein und nicht schwer genug sei, um bei der Lötarbeit fest wie ein Fels auf dem Labortisch zu verharren. Wir müssen zugeben, dass ein kompaktes und leichtes Gerät für tragbare Anwendungen großartig ist, aber für eine Tischlötstation alles andere als ideal. Die Aufnahme des Netzteils in das Gehäuse der neuen Lötstation sollte eine einfache Möglichkeit sein, dieses Problem zu lösen. Ein 60-VA-Ringkerntransformator bringt ein dazu ausreichendes Gewicht in das Gehäuse. Weiterhin bemerkenswert: In einem größeren LED-Display (im Vergleich zum kleineren OLED-Display der älteren Station) lässt sich die Temperatureinstellung viel besser ablesen. Es zeigt zwar weniger Informationen an, aber der wichtigste Wert, die Temperatur, ist nun mit einem schnellen Blick zu erkennen.

Never change...

Die Hardware der alten Lötstation funktionierte allerdings hervorragend, so dass wir in dieser Hinsicht nicht viel geändert haben (**Bild 1**). Die Schaltung zur Temperaturmessung und -regelung ist recht ähnlich, allerdings hatten wir schon während der Prototyping-Phase einige Zweifel an der Notwendigkeit einer Strommessung mit einem Shunt-Widerstand und dem Sense-Verstärker INA138. Die Hard- und Software zur Regelung der Temperatur war unnötig kompliziert, so dass wir diesen Teil der Schaltung komplett weggelassen haben.

Stromversorgung

Die neue Lötstation wird von einem großen

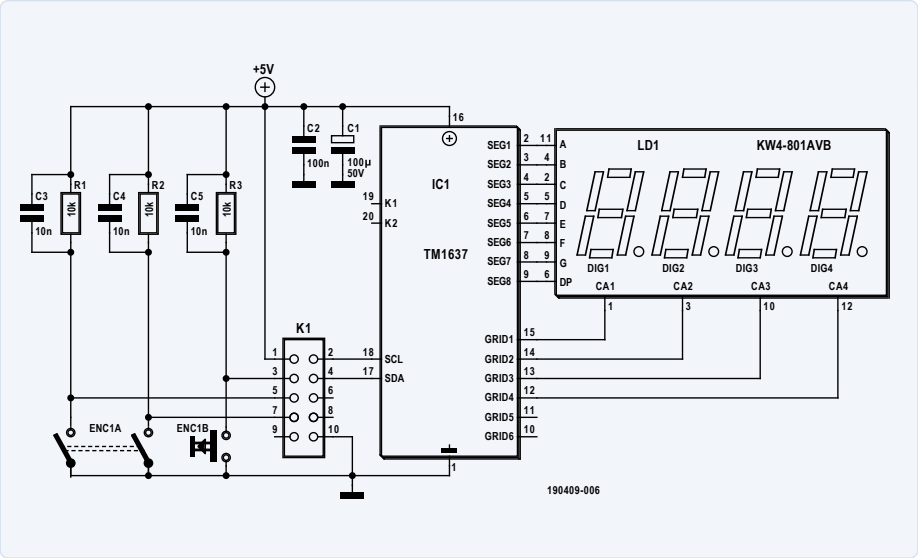


Bild 2. Schaltung der Frontpanel-Platine.

Ringkerntransformator (2x12 V, 60 VA) versorgt, wobei die beiden Sekundärwicklungen jeweils an K1 und K2 angeschlossen sind. Der in der Stückliste vorgeschriebene Typ hat ebenfalls zwei Primärwicklungen, so dass neben einer Netzwechselspannung von 230 V (Wicklungen in Reihe) auch der Betrieb mit 115 V Netzwechselspannung (Primärwicklungen parallel) möglich ist. Natürlich können Sie auch einen Transformator mit nur einer 230-V-Primärwicklung einsetzen, wenn Sie keine Expedition nach Übersee planen. Eine der beiden Sekundärwicklungen wird über den Brückengleichrichter B2 und den Gleichspannungswandler IC1 für die +5-V-Versorgung der Logik- und Steuerschaltung verwendet. Je nach angeschlossenem Lötkolbentyp können die Sekundärwicklungen über Relais RE1 parallel oder in Reihe geschaltet werden, wobei bei ausgeschaltetem Relais die Spannung geringer ist. Das Relais wird vom Mikrocontroller über Transistor T1 angesteuert. Die Wechselspannung wird gleichgerichtet und geglättet (B1 und C1), über dem Spannungsteiler R3 und R6 gemessen und auf den Eingangsbereich des internen ADC des Mikrocontrollers skaliert. Die Z-Diode D2 dient als Überspannungsschutz.

Die dem Lötkolben zugeführte Leistung, die die Temperatur der Lötspitze bestimmt, wird von einem PWM-Signal gesteuert, wobei wir die gleiche Schaltung (T2 bis T5) wie in der alten DIY-Lötstation (180348-1) verwenden. Die Gleichtaktdrossel L2 unterdrückt HF-Einstreuungen auf dem Kabel des Lötkolbens.

Messung der Temperatur

Die Opamp-Schaltungen um IC2A und IC2B verstärken die Spannung am Anschluss des Temperatursensors um das 361-fache (C-Typ bei Hakko) beziehungsweise 181-fache (K-Typ bei Weller und JBC). Statt eine umschaltbare Verstärkung haben wir zwei getrennte Verstärker eingesetzt, nur ein paar Bauteile mehr, aber viel weniger Aufwand in der Software. Der Typ des Lötkolbens wird in der Software eingestellt: der Mikrocontroller wählt die entsprechende Versorgungsspannung und den ADC-Eingang, um die Temperatur der Spitze zu messen. An K7 kann ein DS18B20 1-Wire-Temperatursensor angeschlossen werden, um die Umgebungstemperatur zur Kaltstellenkompensation des Thermoelements im Lötkolben messen zu können.

Programmierung und Fehlersuche

Glücklicherweise haben Mikrocontroller-Hersteller wie Microchip/Atmel entdeckt, dass es durchaus noch einen Markt für Through-Hole-Bauteile gibt. Sogar der aktuelle ATmega4809 mit Microchips „neuer" kombinierter Programmier- und Debug-Schnittstelle *Unified Program and Debug Interface* (UPDI) ist in einem 40-poligen DIP-Gehäuse erhältlich, ideal für Prototypen und Selbstbauprojekte. Das sechspolige UPD-Interface steht am Verbinder K11 zur Verfügung. Mehr Details zu diesem Nachfolger der Programmierschnittstelle AVRISP, die uns so viele Jahre treu gedient hat, finden Sie auf Elektor Labs [2]. Auf dieser Seite wird auch gezeigt, wie man ein Arduino-Uno-Board als Programmierschnittstelle implementiert, das für dieses Projekt verwendet werden kann. Bitte beachten Sie, dass die recht günstige Microchip-Snap-UPDI Programmier/Debugging-Schnittstelle in der Arduino-IDE (noch!) nicht unterstützt wird.

Die Verbindung zur Frontplattenplatine mit dem Display und dem Drehgeber verläuft über die Buchsenleiste K12, auf die wir später noch eingehen werden. Der auf der Platine mit „spare" beschriftete Pin 9 wird in der aktuellen Version der Hard- und Software nicht verwendet, kann aber in späteren Erweiterungen zum Beispiel für einen zusätzlichen Ein- oder Ausgang auf der Frontplatte genutzt werden.

Fast alle unbenutzten Pins des Mikrocontrollers sind auf SIL-Stiftleisten auf der Platine geführt. Zwei UARTS befinden sich auf K6 und K10 und weitere (analoge und digitale) I/Os auf K4, K5 und K9, aber keiner davon hat bislang eine Funktion oder wird in der aktuellen Version der Firmware unterstützt.

Auf der Frontpanel-Platine

Auf der Suche nach erschwinglichen Displays stießen wir auf eine Menge fertiger 4-stelliger I²C-gesteuerter Module, von denen viele von einem LED-Display-Treiber TM1637 der Firma Titan Microelectronics angesteuert werden. Diese Displays sind in der Maker-Community weit verbreitet und können auch in unserer Lötstation verwendet werden. Für eine Lötstation hätten drei Ziffern für die Temperaturanzeige gereicht, aber die vierstelligen Displays erwiesen sich als die günstigere Variante. Da wir auch einen Drehgeber mit Taster auf der Frontplatte benötigen und die Verdrahtung zwischen Haupt- und Frontplatine so einfach und ordentlich wie möglich gehalten werden soll, entschieden wir uns für eine eigene Platine mit all diesen Frontplatten-Komponenten darauf, die nur über ein Flachkabel miteinander verbunden sind. Aus diesem Grund haben wir kein komplettes LED-Modul, sondern einen einzelnen TM1637 und eine Vierfach-7-Segment-LED-Anzeige verwendet (**Bild 2**). Der TM1637 ist sowohl in DIP- als auch in SOIC-Gehäusen erhältlich. Obwohl wir die Lötstation vollständig nur in Durchstecktechnik ausführen wollten, haben wir hier eine Ausnahme gemacht und Footprints für beide Gehäusevarianten auf der Platine bereitgestellt. Der Hauptgrund dafür war, dass sich die Lieferung der von uns in Asien bestellten DIP-ICs wegen COVID-19 verzögerte und wir einige komplette Module im Labor herumliegen hatten. Ein SOIC von einer dieser Platinen zu plündern, war der schnellste Weg, unseren Prototyp weiter zu entwickeln.

Software

Der Code selbst besteht aus mehreren ineinandergreifenden Teilen, die als Klassen und daraus erzeugten Objekte implementiert sind. Wer mit dem Arduino-Framework arbeitet, verwendet Objekte und Klassen, oft ohne es zu wissen. Grob gesagt stellt eine Klasse einen Bauplan dar, mit dem im Code ein passen-

des Objekt erzeugt wird, also das, was beim Bauen nach dem Bauplan entsteht. Eine Einführung in die Objektorientierung von C++ würde hier viel zu weit gehen, daher werden wir nur sehr abstrakt auf die Objekte und Klassen eingehen.

Nehmen wir zum Beispiel das Display, in der Software *frontend* genannt. Es zeigt die Werte wie Temperatur, Menü und Fehlercodes an, die vom Kern der Station an das Frontend übertragen werden. Nun hätte es wenig Sinn, für jedes Display mit unterschiedlichen Anzeigeeigenschaften (OLED, alphanumerisches LCD oder 7-Segment-Display) die notwendigen Anpassungen im Code vorzunehmen.

Hier kommen die Klassen und Objekte ins Spiel: Für den Kern der Software ist das Frontend ein beliebiges Display, das immer die gleichen Funktionen bereitstellt. Mit den entsprechenden Klassen (Bauplänen) kann also die Software für eine Lötstation für die hier beschriebene Hardware erstellt werden. Wenn Sie statt des LED-Displays zum Beispiel ein 0,96"-OLED verwenden möchten, muss nur der Code für dieses Frontend integriert werden.

Der Code der alten Lötstation (180349) war und ist so konzipiert, dass dies möglich ist. Der Kern der Station bleibt grundsätzlich gleich, nur die Teile drum herum werden an die Hardware angepasst.

Auch für die Temperaturregelung wurden Anpassungen vorgenommen. Wie im Schaltplan zu sehen, sind zwei getrennte Operationsverstärker eingebaut, einer für Typ-K- und einer für Typ-C- Thermoelemente, so dass Spitzen und Handlötkolben verschiedener Hersteller verwendet werden können. Zusätzlich ist eine Kaltstellenkompensation auf Basis eines DS18B20 vorgesehen. Ein geeigneter Sensor wird automatisch erkannt und ausgewertet. Je nach Lötspitze, die in den Einstellungen der Lötstation konfiguriert ist, wird automatisch der Analogeingang A0 oder A1 ausgewählt und das Signal auf die entsprechende Temperatur umgerechnet. Anschließend wird, falls vorhanden, eine Kaltstellenkompensation durchgeführt.

Um den OneWire-Sensor DS18B20 auszulesen, mussten wir ein paar Tricks anwenden. Nach dem Start der Software wird nach diesem OneWire-Sensor gesucht und, wenn keiner gefunden wird, dies in der Software vermerkt, damit nicht weiter versucht wird, den Sensor erneut zu suchen. Ist ein Sensor vorhanden und wird er gefunden, kommt es zum Lesen des Temperaturwerts in zwei Schritten. Der erste Schritt wird alle zwei Sekunden durchgeführt und startet die Aufzeichnung der aktuellen Temperatur. Der Sensor benötigt mindestens 750 ms Wandlungszeit, bis der Wert zum Auslesen und Verarbeiten bereit ist. Da die MCU aber nicht 750 ms anhalten soll, wird der Beginn der Wandlung nur markiert. Liegt diese Startmarke der Wandlung mehr als eine Sekunde zurück, kann der Wert aus dem Sensor geholt und der Abschluss des Auslesevorgangs vermerkt werden. Nach einer Sekunde wird die nächste Lesesequenz gestartet. Auf diese Weise kann der Controller während der Temperaturwertwandlung andere Aufgaben ausführen.

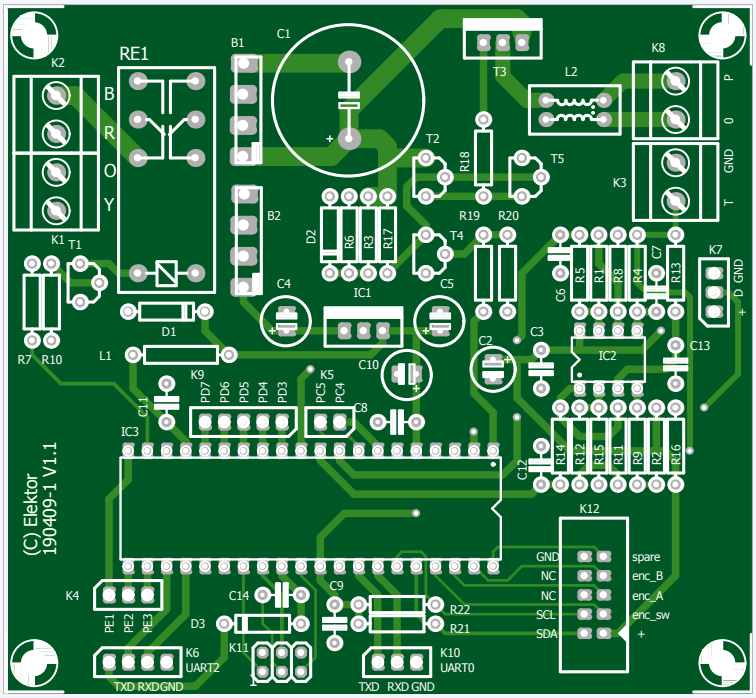


Bild 3a. Layout der Hauptplatine.

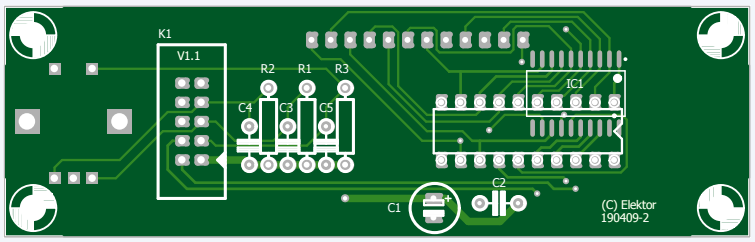


Bild 3b. Layout der Frontpanel-Platine.

PWM mit Timer A

Die Timer im ATmega4809 unterscheiden sich ein wenig von denen seiner Vorgänger. In diesem neuen Mikrocontroller kann Timer A sechs PWM-Ausgänge bereitstellen, was sehr praktisch für die Ansteuerung von Servos, LEDs und anderen Komponenten ist. Dies ist auch die Standardkonfiguration von MegaCoreX, dem Hardware-Paket, das in der Arduino-IDE für den ATmega4809 installiert werden muss. Der Timer A ist zwar ein 16-Bit-Timer mit drei PWM-Ausgängen, ist aber standardmäßig als 8-Bit-Timer mit sechs Ausgängen konfiguriert. Dies ist vielleicht für Servos ausreichend, aber für die Leistungssteuerung der Lötstation ist es nicht genug, die Leistung mit 10 kHz zu steuern. Deshalb muss der Timer A für die PWM-Erzeugung in unserer Lötstation auf den 16-Bit-Modus umgeschaltet werden.

Drehimpulsgeber mit Taster

Der Drehgeber wird in einer Timer-Interrupt-Routine verarbeitet. Alle 250 µs werden

STÜCKLISTE HAUPTPLATINE

Widerstände:
R1,R11,R12 = 18 k
R2 = 1 M
R3 = 68 k
R4,R5,R13,R14 = 5k6
R6,R7,R19 = 10 k
R8,R15,R18 = 100 Ω
R9 = 10 M
R10,R16,R17,R20,R21,R22 = 4k7 k

Induktivitäten:
L1 = Drossel 10 µ, 130 mA
L2 = Gleichtaktdrossel 10 A (Laird CM2545x171B-10)

Kondensatoren:
C1 = 4700 µ, 50 V, Raster 10 mm, 22x41 mm²
C2,C5,C10 = 10 µ, 50 V, Raster 2 mm, 5x11 mm²
C3,C8,C9,C11,C14 = 100 n, 50 V, X7R, Raster 5,08 mm
C4 = 100 µ, 50 V, Raster 3,5 mm, 8x11 mm²
C6,C7,C12,C13 = 10 n, 50 V, X7R, Raster 5 mm

Halbleiter:
D1 = 1N4007, 1000 V, 1 A
D2 = Z-Diode, 5,1 V, 500 mW
D3 = 1N4148, 100 V, 200 mA, 4 ns
B1,B2 = Brückengleichrichter D6KB6U, 600 V, 6 A
T1,T2,T4 = BC547C, 45 V, 100 mA, 500 mW, hfe=400
T3 = IRF9Z34NPBF, MOSFET-P, 55 V, 17 A, 100 mOhm
T5 = BC557C, -45 V, -100 mA, 500 mW, hfe=400
IC1 = DC/DC-Wandler OKI-78SR-5/1,5-W36-C, 5V, 1,5A
IC2 = Doppel-Opamp MCP6002-E/P
IC3 = 8-Bit-MCU ATmega4809-PF

Außerdem:
RE1 = Leistungsrelais, 5 VDC, DPDT, 8 A (Schrack RT424005)
K1,K2,K3,K8 = 2-polige Platinenklemme, Raster 5,08 mm, 630 V
K4,K7,K10 = 1x3-polige Stiftleiste, gerade
K5 = 1x2-polige Stiftleiste, gerade
K6 = 1x4-polige Stiftleiste, gerade
K9 = 1x5-polige Stiftleiste, gerade
K11 = 2x3-polige Stiftleiste, gerade
K12 = 2x5-polige Buchsenleiste, Raster 2,54 mm
Ringkerntransformator 60 VA, 2x115 V, 2x12 V (MCTA060/12)
Sicherung 630 mA, 20 mm
Kaltgerätekupplung mit Schalter und Si-Halter (K&B 59JR101-1FR-LR Serie 42R, bei Conrad 736709)
2x5-poliger IDC-Kabelverbinder (2 Stück)
10-adriges Flachbandkabel, ca. 20 cm
Platine 190409-1 V1.1

STÜCKLISTE FRONTPANEL-PLATINE

Widerstände:
R1,R2,R3 = 10 k

Kondensatoren:
C1 = 100 µ, 50 V, 3,5 mm Raster, 8x11 mm²
C2 = 100 n, 50 V, X7R, Raster 5,08 mm²
C3,C4,C5 = 10 nF, 50 V, X7R, 5 mm Raster

Halbleiter:
LD1 = Vierstellige 7-Segment-LED-Anzeige (Luckylight KW4-801AVB)
IC1 = TM1637 (I2C-LED-Ansteuerung)

Außerdem:
ENC1 = Encoder mit Drucktaster (Bourns PEC11R-4225F-N0024)
K1 = 2x5-polige Buchsenleiste, Raster 2,54mm
Platine 190409-2 V1.1

die Pins des Drehgebers ausgelesen und aus den letzten vier Zuständen der Pins die Drehrichtung ermittelt. Da der Drehgeber mechanische Kontakte hat, würde die Rotation keine sauberen Flanken an den Eingangspins des ATmega erzeugen. Ohne Filterung wäre immer ein gewisses Rauschen in den Signalen vorhanden, und um dies zu unterdrücken, sind drei RC-Tiefpassfilter und eine Software-Filterung eingesetzt.

Der Drucktaster des Drehgebers wird in der gleichen Funktion verarbeitet und somit ebenfalls alle 250 µs überprüft.

Fehler...

Alle 50 ms wird die Ist-Temperatur des Lötkolbens mit dem eingestellten Wert verglichen. Erkennt die Station eine höhere Temperatur als 650°C, wird dies als Sensorfehler interpretiert und die Heizung abgeschaltet, um eine Überhitzung zu vermeiden. Erkennt die Station nach sechs Sekunden Aufheizen keinen Temperaturanstieg an der Spitze, wird dies ebenfalls als Fehler gewertet und das Aufheizen abgebrochen. Bei einem Defekt des Temperatursensors zeigt die Station im Display E-01 an, bei einem Fehler beim Aufheizen E-03. Ein kurzer Druck auf den Taster quittiert die Fehlermeldung und die Station versucht, den normalen Betrieb fortzusetzen.

WEBLINKS

[1] **Temperaturgesteuerte Lötstation**: www.elektormagazine.de/magazine/elektor-69/42252
[2] **UPDI-Programmierer für ATmega4809 und ATtiny816/817**: www.elektormagazine.de/labs/arduino-for-updi-programming-for-atmega4809-and-tiny816817-with-jtag2updi
[3] **Downloads zu diesem Projekt**: www.elektormagazine.de/190409-03
[4] **STEP-Dateien für Gehäuse**: https://github.com/DK1CMB/Elektor-SolderironCase
[5] **STEP-Dateien für Knopf**: www.thingiverse.com/tracert/designs

Bild 4. Kaltgerätekupplung mit Sicherungshalter und Netzschalter.

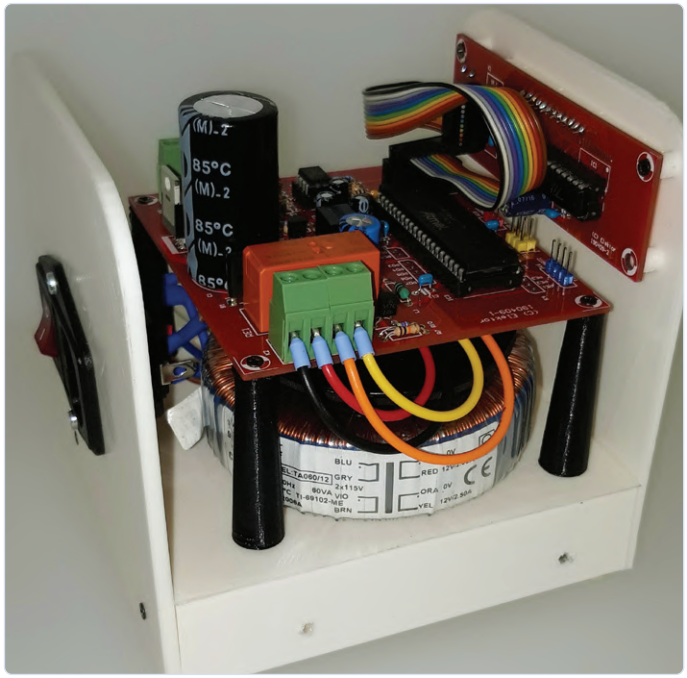


Bild 5. Innenseite des 3D-gedruckten Gehäuses.

Aufbau der Hardware

Die Gerberdateien für die beiden Platinen (**Bild 3**) können von [3] heruntergeladen werden. Mit diesen Dateien können Sie die Platinen bei einem Platinenservice bestellen. Das Löten von Platinen mit durchkontaktierten Bauteilen sollte Ihnen nicht schwer fallen. Beachten Sie, dass sich auf der Frontplattenplatine die meisten Bauteile auf der Oberseite befinden, mit zwei Footprints für den Displaytreiber IC1 (SOIC und DIL) – von denen natürlich nur einer verwendet werden soll. Die LED-Anzeige und der Drehgeber müssen auf die Unterseite der Platine verlötet werden.

Beim letzten Mal haben wir für unsere kleine Lötstation nur die Platine, aber kein Gehäuse zur Verfügung gestellt. Das hat einige Leser in Schwierigkeiten gebracht, die Elektronik in ein kommerzielles Gehäuse einzubauen. Deshalb wollen wir es diesmal beim Selbstbau nicht bei der Platine und der Verdrahtung belassen, sondern auch ein 3D-druckbares Gehäuse hinzufügen, das freundlicherweise von Caroline Claußen erstellt wurde. Alle STEP-Dateien können unter [4] heruntergeladen und auf den gängigen 3D-Druckern wie dem Anycubic I3 Mega-S ausgedruckt werden (siehe „Passende Produkte“). Eine Warnung: Einige Teile brauchen sehr lange beim Drucken, aber lassen Sie niemals einen laufenden 3D-Drucker unbeaufsichtigt, da diese Maschinen mit hohen Strömen arbeiten. Da ist bei einem Fehler schnell etwas Gefährliches passiert!

Falls Sie sich jemals gefragt haben, warum Platinen-CAD-Programme wie KiCad und Altium 3D-Modelle für die Bauteile/Platinen darstellen können: Es macht den Entwurf eines Gehäuses sehr viel einfacher. Aus diesen PCB-Design-Tools können Sie Ihre Kreation als 3D-STEP-Datei exportieren und sie mit jedem Programm weiterverwenden, das dieses Dateiformat verarbeiten kann. Fast jede 3D-Designsoftware kann dann verwendet werden, um ein schön aussehendes Gehäuse um das 3D-Modell der Platine(n) herum zu konstruieren. Hier ergaben sich neun Teile, die gedruckt werden müssen. Insgesamt dauert es bis zu 24 Stunden, je nach Drucker, bis die Teile fertig sind. Dann folgt als letzter Schritt der Zusammenbau des Gehäuses.

Es wird eine Kaltgerätekupplung mit integriertem Netzschalter und Sicherungshalter benötigt, mit Schrauben und Muttern zur Befestigung in der Rückwand, wie in **Bild 4** gezeigt. Einige Drähte und Flachstecker verbinden den Netzschalter und die Sicherung mit den Primärwicklungen des Trafos. Beachten Sie beim Anschluss des Transformators, dass die Klemmen an K1 und K2 mit den Anfangsbuchstaben der Farben der Isolation der Sekundärwicklungen gekennzeichnet sind (Schwarz, Rot, Orange, Gelb). Wenn Sie einen anderen Trafotyp verwenden, vergewissern Sie sich, dass die „Wicklungsanfangsmarkierungen“ der Drähte, die im Datenblatt normalerweise mit Punkten gekennzeichnet sind, mit dem von uns verwendeten Multicomp-Transformator übereinstimmen.

Zum Verschrauben von Rückwand und Deckel können Sie M2,5-Schrauben verwenden. Die Platine thront auf vier Säulen, die in den Gehäuseboden gesteckt werden. Die Platine wird aufgesetzt und mit vier Schrauben fixiert. Die Säulen gleichen auch die Ungenauigkeit Ihres 3D-Druckers aus. Mit ein wenig Geduld erhalten Sie am Ende eine Konstruktion, die wie in **Bild 5** aussieht.

Hinweis: Diese Lötstation wurde ausgiebig mit einem Weller-RT-Lötkolben getestet. Leider hatten wir keinen Hakko-FX-8801 oder JBC-T245 zum Testen zur Verfügung, aber die Lötstation sollte auch mit diesen Marken/Modellen gut funktionieren.

Der Anschluss für den Lötkolben (**Bild 6**) kann auf verschiedene Arten erfolgen. Eine vorgefertigte Bohrung im Gehäuse gibt es dafür nicht. Für unseren Prototyp mit dem Weller-RT-Lötkolben haben wir einen 3,5-mm-Audio-Klinkenstecker montiert, was zwar nicht die bestmögliche Lösung ist, aber funktioniert. Verwenden Sie kein flexibles Audio-Verlängerungskabel, der Querschnitt des Kupferdrahtes ist dabei zu klein für den Strom für das Heizelement. Wir haben das ausprobiert und hatten statt eines

heißen Lötkolbens innerhalb einer Minute ein heißes Kabel. Wir haben dies durch ein Kabel mit einem größeren Querschnitt gelöst. Last but not least: der Drehknopf. Da jeder seine eigenen Vorlieben für Knöpfe, Größe und Formen hat, können Sie einfach bei Thingiverse nach „rotary knob“ suchen und denjenigen auswählen, der Ihnen am besten gefällt. Wir haben hier als Beispiel das Design von Domain Mittu [5] verwendet. Für den Drehknopf haben wir TPU-Filament verwendet, das ist weicher und gibt einen guten Halt beim Drehen. Die rote Abdeckung haben wir mit PTEG-Filament gedruckt, das nicht so leicht handzuhaben, aber stärker als PLA ist. Es ist vollkommen in Ordnung, auch die Abdeckung mit PLA zu drucken.

Im Betrieb

Beim ersten Einschalten, bevor ein Lötkolben angeschlossen wird, halten Sie den Druckknopf des Drehgebers gedrückt, während Sie einschalten. Dadurch öffnet sich das Boot-Menü zur Auswahl des Lötkolbentyps, wobei Sie zwischen C0, C1 und C2 wählen können, was Hakko FX-8801, JBC T245 und Weller RT entspricht. Drücken Sie den Drehgeber zehn Sekunden lang, um diese Auswahl zu speichern. Dann wird die Station mit der richtigen Kolbenspannung (12 V/24 V) und dem richtigen Analogeingang für die Temperaturmessung neu gestartet. Natürlich bleibt diese Auswahl auch nach dem Ausschalten erhalten. Im Normalbetrieb wird mit dem Drehgeber die Temperatur des Lötkolbens eingestellt. Das Display leuchtet mit maximaler Helligkeit, wenn diese Einstellung geändert wird. Die neue Temperatureinstellung wird fünf Sekunden nach Loslassen des Drehgebers übernommen. Danach dimmt das Display auf normale Helligkeit und zeigt die aktuelle Temperatur der Spitze an. Wenn der Lötkolben aufheizt, blinkt der linke Dezimalpunkt der Anzeige.

Wir sind sicher, dass mit dieser neuen Lötstation die meisten Einwände gegen das frühere Design ausgeräumt sind, aber natürlich gibt es noch Raum für Verbesserungen. Um die Hardware einfach erweitern zu können, haben wir fast alle ungenutzten Mikrocontroller-Pins auf Erweiterungsstecker auf der Hauptplatine geführt. Sollten Sie selbst Anpassungen oder Erweiterungen vornehmen, die für andere Leser von Interesse sein könnten, teilen Sie diese bitte mit uns!

190409-03

Bild 6. Anschluss der Lötspitze Weller RT.

Haben Sie Fragen oder Kommentare?
Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Schicken Sie eine E-Mail an die Autoren unter mathias.claussen@elektor.com und luc.lemmens@elektor.com oder kontaktieren Sie Elektor unter editor@elektor.com.

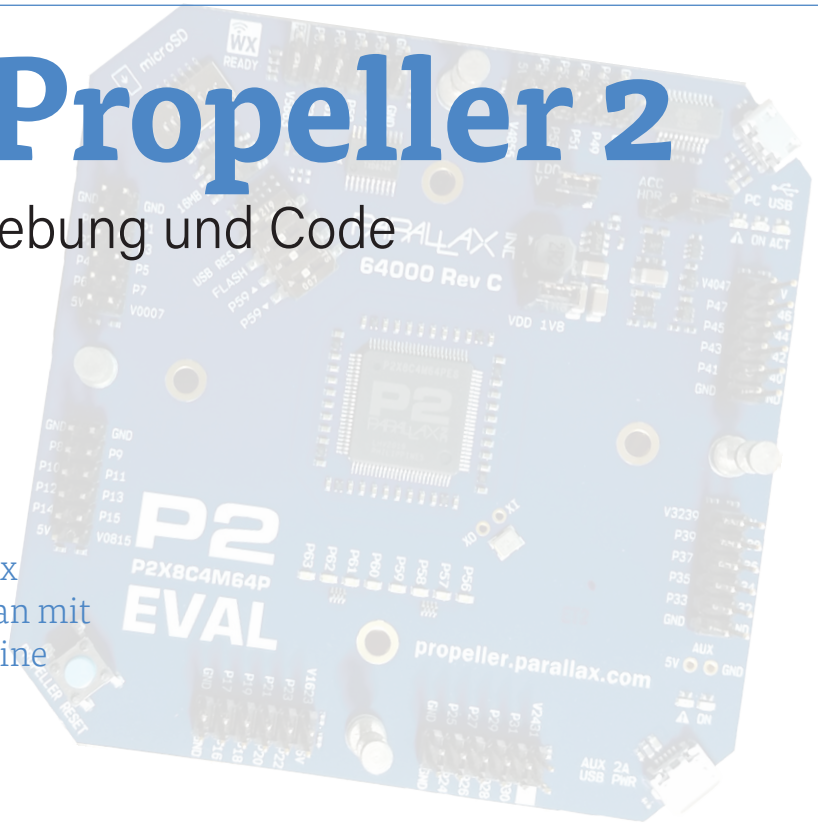
Ein Beitrag von
Idee, Gestaltung und Text:
Mathias Claussen, Luc Lemmens
Illustrationen: **Patrick Wielders**
Redaktion: **Luc Lemmens**
Übersetzung: **Rolf Gerstendorf**
Layout: **Giel Dols**

PASSENDE PRODUKTE

- **Weller WE 1010 Digitale Lötstation (Education Kit)**
  www.elektor.de/weller-we-1010-digital-soldering-station-education-kit
- **Fumetractor mit LED-Lampe**
  www.elektor.de/fumetractor-with-led-light
- **Anycubic i3 Mega-S 3D Printer (Kit)**
  www.elektor.de/anycubic-i3-mega-s-3d-printer-kit

Parallax Propeller 2

Teil 2: Entwicklungsumgebung und Code



Von **Mathias Claußen** (Elektor)

In Teil 1 dieser Artikelserie wurde mit Propeller 2 der neue fortschrittliche Multicore-Mikrocontroller von Parallax vorgestellt. Jetzt geht es darum, wie man mit Hilfe der Programmiersprache Spin2 eine LED zum Leuchten bringen kann.

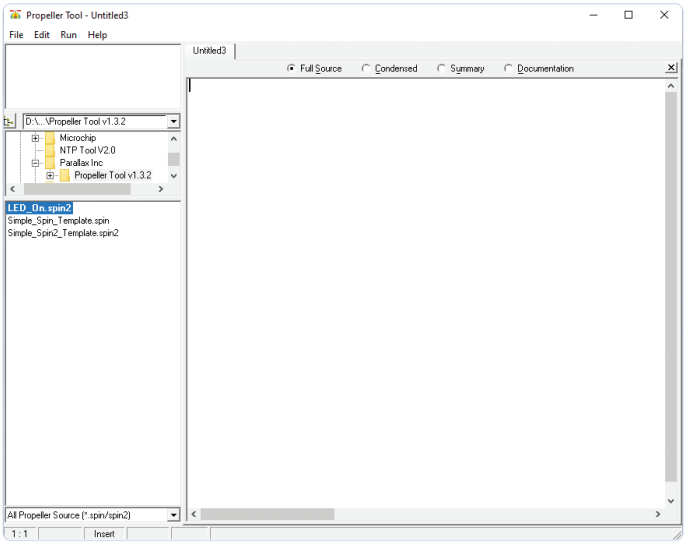


Bild 1. User-Interface des Propeller Tools.

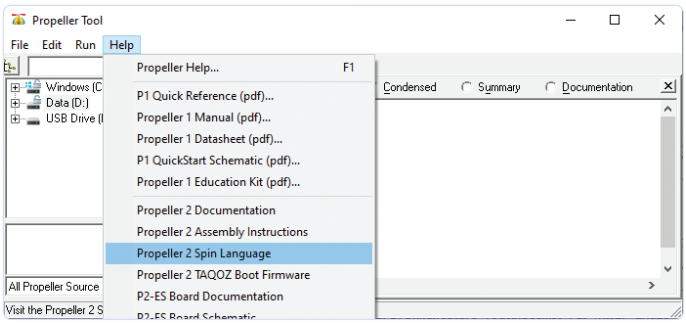


*Bild 2. Über das Menü **Help** erreicht man die Spin2-Dokumentation.*

Nach der grundlegenden Besprechung des Mikrocontrollers Propeller 2 von Parallax samt seinen Funktionen möchte ich nun auf die Entwicklungsumgebung und die Programmierung eingehen. Nachfolgend erfahren Sie, wie man damit z.B. eine LED ansteuern kann.

Zugriff auf Onboard-LEDs

Los geht es mit Software-Umgebung und der Ansteuerung einer der LEDs auf dem Board. Die Softwareentwicklung wird unter Windows 10 mit den vom Parallax angebotenen Tools durchgeführt. Am Ende steht ein grundlegendes Software-Setup zur Verfügung, mit dem Sie Code kompilieren und das Resultat auf den Propeller 2 schieben können.
Parallax stellt momentan sein Propeller-Tool in der Version 2.3 Alpha zur Verfügung. Es enthält die Propeller-2-Unterstützung für die Programmiersprache Spin/Spin2. Wer Assembler bevorzugt, kann auch PNut als Entwicklungsumgebung oder Inline-Assembler verwenden. Falls Sie wie ich gerne in C/C++ programmieren: Geeignete Compiler und Entwicklungsumgebungen sind derzeit leider noch nicht soweit. Erste Anstrengungen in dieser Hinsicht kann man einem Video [1] über C/C++ auf Propeller 2 beobachten.

Grundlegende Entwicklungsumgebung

Die URL zum Download des Propeller Tool 2.3 Alpha finden Sie unter [2]. Zuerst steht aber der Download des *Propeller Tool 1.3.2* an, und im zweiten Schritt wird die ausführbare Datei *Propeller Tool 2.3 Alpha* in das Programmverzeichnis kopiert. Anschließend können Sie den Editor starten und mit der Programmierung beginnen. **Bild 1** zeigt die Benutzeroberfläche.

Assembler oder Spin2?

Sollte man Spin2 oder Assembler für das erste Programm verwenden? Der Einfachheit halber geht es mit Spin2 weiter. Spin2 wird wie die meisten BASIC-Dialekte interpretiert - aber anders. Befehle und Sprachreferenzen zu Spin2 finden sich beim Propeller-Tool unter dem Menü *Help* (siehe **Bild 2**).

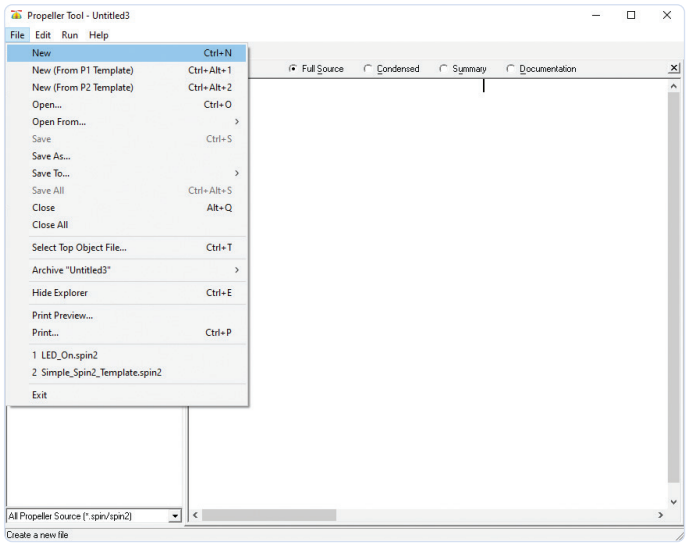


Bild 3. Ein neues Spin2-Projekt erstellen.

Bild 4. Die von den MCU-Pins 56 bis 63 angesteuerten LEDs.

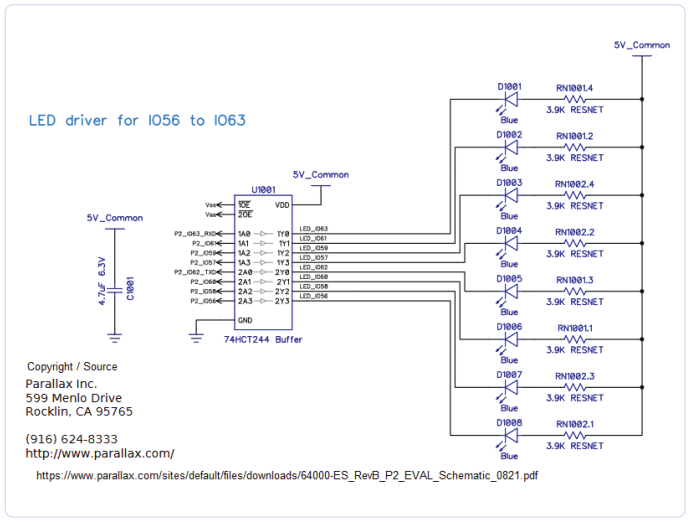


Bild 5. Der Buffer 74HCT244 als LED-Treiber.

Für den Einstieg gibt es ein noch vorläufiges Handbuch, dass leider noch nicht ganz fertig ist. Dieses führt zur aktuellen Spin2-Dokumentation für Propeller 2.
Aufgrund der Interpretierung benötigt ein Spin2-Befehl etwa sechs Instruktionszyklen = 12 Taktzyklen. Dies ergibt nicht gerade die allerschnellste Programmausführung, aber Spin2 bietet einen einfacheren Einstieg als Assembler. Da Spin2 eine High-Level-Sprache ist, stecken hinter den meisten Spin2-Befehlen viele auszuführende Assembler-Instruktionen, die dafür aber einfacher zu benutzen sind als purer Assembler-Code. **Bild 3** zeigt, wie ein neues Spin2-Projekt erstellt wird.

I/O-Pins

Bevor man mit der Programmierung beginnt, sollte man sich zunächst mit den I/O-Pins vertraut machen. Im Folgenden werden sie als gewöhnliche I/O-Pins eingesetzt, und erst später gibt es einen detaillierten Überblick über alle ihre Fähigkeiten. Um einen Low- oder High-Pegel an einem Pin zu erzeugen, muss man diesen zuvor als Ausgang konfigurieren. Auf dem Evaluations-Board befinden sich die mit P56 bis P63 bezeichneten LEDs (siehe **Bild 4**). Diese LEDs sind mit den Pins 56 bis 63 des Propeller 2 verbunden. Für das erste Beispiel soll die erste LED P56 angesteuert werden. Im Schaltplan wir klar, dass sie nicht direkt mit dem Chip, sondern mit dem Oktal-Buffer 74HCT244 verbunden ist. Die LEDs liegen mit ihrer Anode permanent an VCC. Um zu leuchten, muss der Buffer ihre Kathode auf Massepotential ziehen (siehe **Bild 5**).

Invertierte LED

Für den Code bedeutet dies, dass die LED bei einem High-Pegel am MCU-Pin dunkel bleibt und sie erst bei einem Low-Pegel aufleuchtet. Dank dieser invertierten Logik muss der Code die LED-Pins auf „high" setzen, damit die LEDs dunkel bleiben. Macht man nichts, fungieren diese Pins per default als Eingang, und die korrespondierenden Buffer verhalten sich so, als ob ihre Eingänge mit einem High-Pegel angesteuert würden. In der Folge bleiben daher die LEDs zunächst dunkel. Um per Spin2-Code die LED P56 aufleuchten zu lassen, muss folgendes geschehen:

- Initialisierung der MCU und mindestens eines Kerns (cog).
- Pin 56 als Ausgang setzen
- Pin 56 auf low setzen
- nichts tun

Die eigentlich notwendige Initialisierung im ersten Punkt ist schon erledigt und kann für dieses Beispiel ignoriert werden. Das Spin2-Programm benötigt nur ein paar Zeilen Code. Dieser beginnt mit dem Funktionsaufruf `pub main()`, gefolgt von der Methode `pinwrite()` (siehe **Bild 6**). Die Methode `pinwrite()` ist in Spin2 enthalten und erlaubt es, einen oder mehrere Pins mit einem bestimmten Wert zu belegen. Da die LED P56 von Pin 56 angesteuert wird, wird dieser Pin der Pegel *„0"* übergeben, um den Pin auf low zu bringen und damit die LED einzuschalten. Die letzte Zeile `repeat` führt in eine Endlosschleife, da der Code unter `repeat` immer wieder ausgeführt wird. In diesem Fall gibt es aber keinen weiteren Code, weshalb der entsprechende „cog" immer wieder nichts tut. Ohne dieses `repeat` würde das Code-Ende erreicht und die I/O-Pins nicht mehr angesteuert werden.

Code in Propeller 2 übertragen

Wenn der Code fertig ist, können Sie ihn auf das Propeller 2 Evaluation Board hochladen und ausführen lassen. Dazu wird das Board einfach

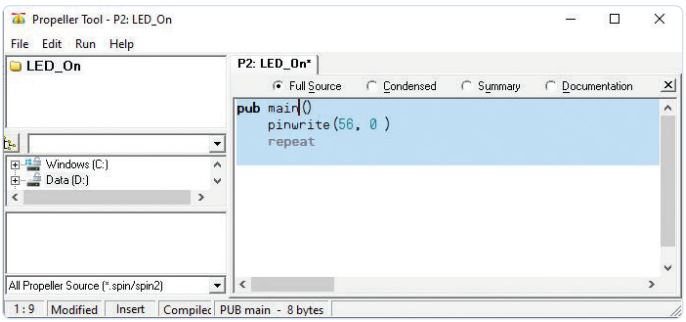


Bild 6. Dieser Code bringt die LED P56 zum Leuchten.

per USB mit dem PC verbunden und das Menü *Run->Compile Current->Load RAM* ausgewählt, damit der Code direkt in das RAM der MCU geladen und ausgeführt wird. Wenn nun die LED P56 leuchtet, war die MCU-Version von „Hello world" erfolgreich. Falls Sie sich nun fragen, wie man die LED zum Blinken bringen kann: In Spin2 gibt es den `WAITMS`-Befehl. Beispielsweise verzögert `WAITMS(500)` die Ausführung des nächsten Befehls um 500 ms. Da der Code nach `repeat` immer wieder ausgeführt wird, ist das die Stelle, an der man die LED zum Blinken bringen kann. Die nächste Übung besteht also darin, den Code entsprechend zu verändern. Mit den bisherigen Informationen sollte es Ihnen möglich sein, den Code so zu modifizieren, dass die LED P56 nicht nur leuchtet, sondern blinkt. Hierfür gibt es später noch eine exemplarische Lösung.

Wenn man einen I/O-Pin erfolgreich ansteuern kann, sollte man im nächsten Schritt herausfinden, wie serielle Daten gesendet werden. MCU-Programmierer sind an diese Art des Debuggens gewöhnt, gerade wenn man zum Beispiel Software für AVR-Controller geschrieben hat. Da dies mit den Smart-Pin-Funktionen gut funktioniert, wird der nächste Teil dieser Serie sich genau darum drehen.

200479-B-02

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.

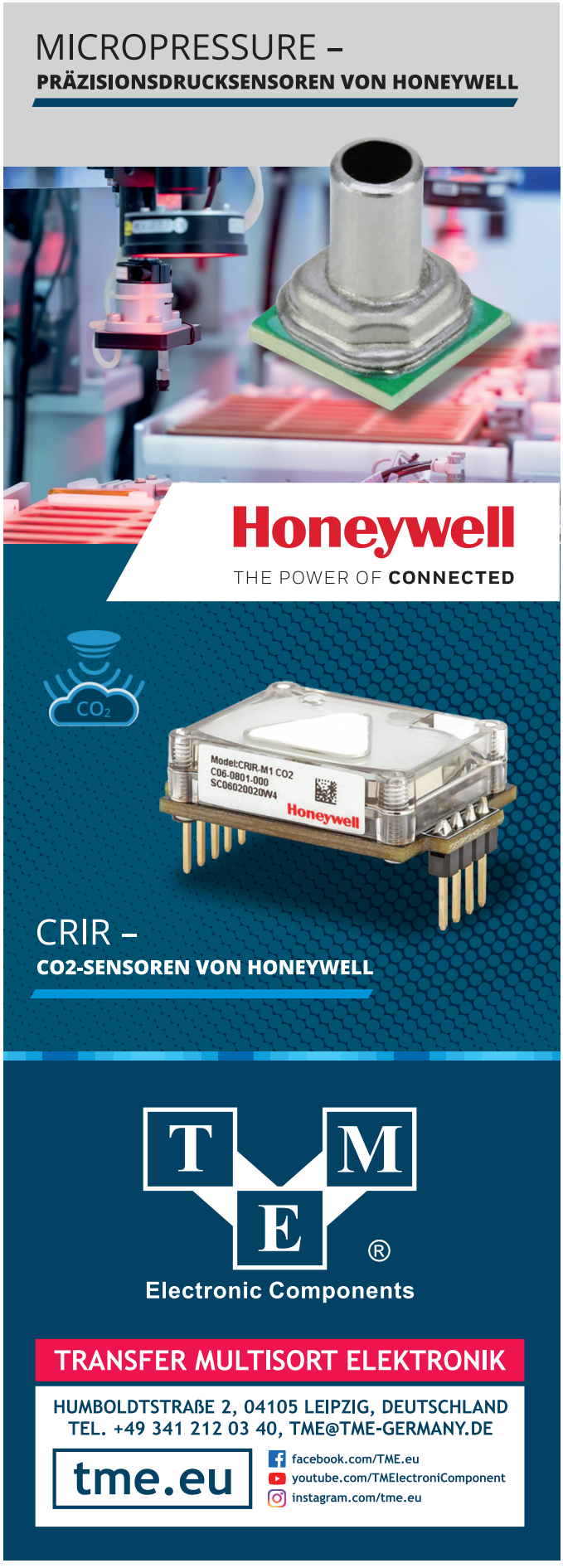
Ein Beitrag von
Autor: **Mathias Claußen**
Redaktion: **Jens Nickel** und **C. J. Abate**
Übersetzung: **Dr. Thomas Scherer**
Layout: **Giel Dols**

WEBLINKS

[1] **Parallax, „Propeller 2 Live Forum Early Adopter Series - C Programming with Eric Smith," 6. Juli 2020:** https://bit.ly/propeller2-c

[2] **Download Propeller Tool 2.3 Alpha:** https://propeller.parallax.com/p2.html#software

Anzeige

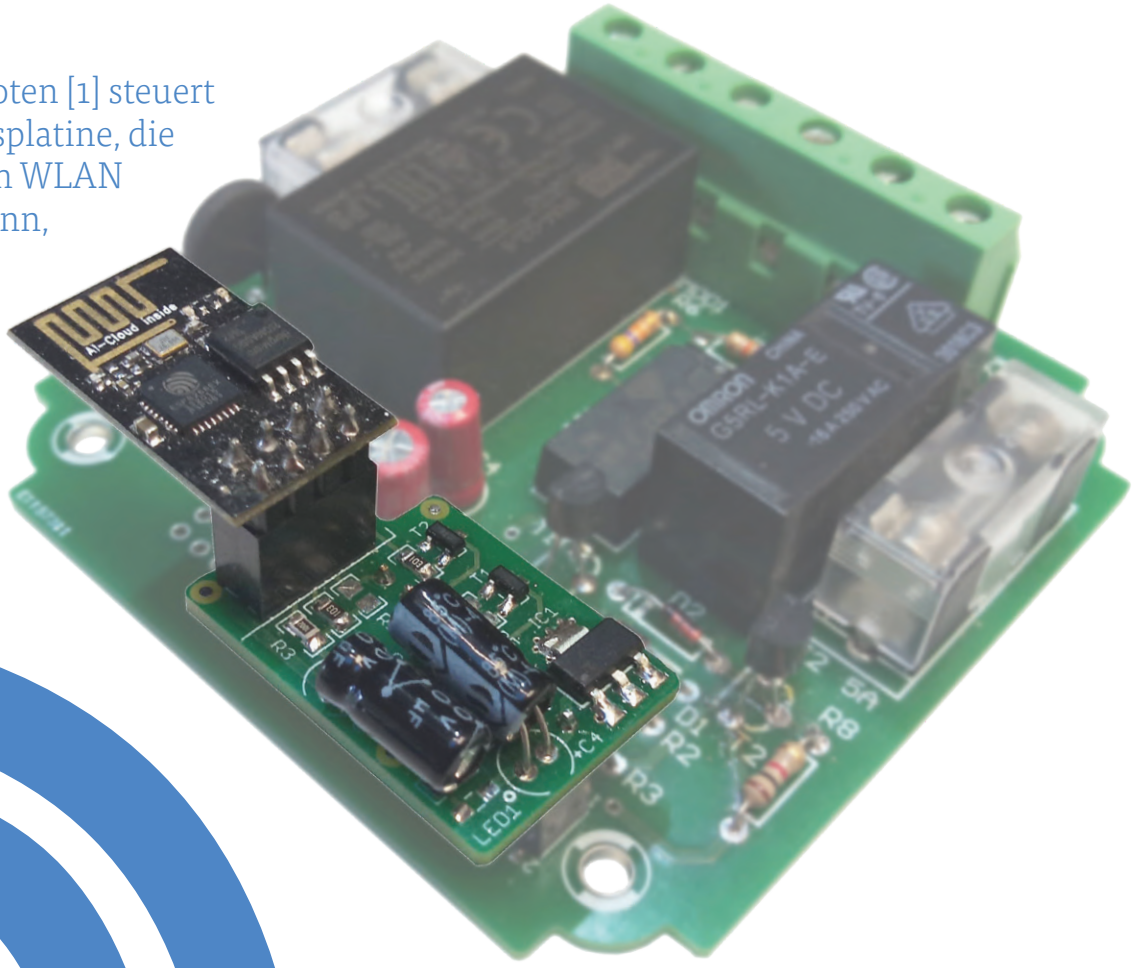


WLAN- statt LoRa-Schalter

Integriert in Home Assistant mit ESPHome

Von **Clemens Valens** (Elektor)

Der Elektor-LoRa-Knoten [1] steuert eine praktische Relaisplatine, die auch einfach in einem WLAN verwendet werden kann, wenn man das LoRa- durch ein WLAN-Modul ersetzt.



In Elektor März/April 2020 wurde ein ferngesteuerter Wechselspannungs-Leistungsschalter mit Zustandsrückmeldung vorgestellt [1], der über LoRa kommunizierte. Es war außerdem ein modulares Projekt mit dem Relais und der Leistungsschaltung auf einer Platine und dem LoRa-Kommunikationsteil auf einer anderen. Die Schaltung in einem wasserdichten IP66-Gehäuse war bestens für den Außeneinsatz geeignet.

Das interessierte mich, weil ich einen Schalter für den Außenbereich suchte, der sich leicht in mein Hausautomatisierungssystem integrieren ließ. Dieses System basiert auf Home Assistant und WLAN, aber nicht auf LoRa. Ich hätte mein System zwar um LoRa erweitern können, aber ich wollte mich nicht damit befassen. Stattdessen dachte ich, wenn ich das LoRa-Modul einfach durch ein WLAN-Modul ersetze, könnte ich es mit ESPHome betreiben, das hervorragend mit Home Assistant funktioniert (**Bild 1**).

Definieren der Schnittstelle

Wenn man eine Schaltung anpassen möchte, muss man zunächst etwas über die Verbindungsmöglichkeiten herausfinden. Der Schaltplan des Wechselstrom-Leistungsschalters ist nicht besonders kompliziert (**Bild 2**). Oben haben wir die Stromversorgung mit dem AC/DC-Wandler MOD1, der aus der Netzspannung eine Gleichspannung von +5 V mit einigen Schutz- und Filterfunktionen erzeugt. In der Mitte befindet sich das bistabile Relais

RE1, das die Last ein- und ausschaltet. Ein bistabiles Relais ist einem mechanischen Schalter sehr ähnlich, da es seinen Zustand auch nach der eigentlichen Betätigung beibehält. Zum Umschalten sind die beiden Steuersignale *Set* und *Reset* (active high) erforderlich, die von einem Controller stammen und von den MOSFETs T1 und T2 gepuffert werden.
Das Relais ist durch eine 5-A-Sicherung F2 geschützt, obwohl das Relais bis zu 16 A schalten könnte. Der Grund dafür ist, dass die Leiterbahnen auf der Platine nicht so viel Strom vertragen, so dass die Sicherung eher sie vor dem Durchbrennen schützt. Die Schaltplatine kann immerhin Lasten bis etwa 1 kW verarbeiten.
Unten links findet sich ein Optokoppler IC1, der parallel zur Last geschaltet ist und deshalb auch zusammen mit der Last eingeschaltet wird. Seine LED schaltet den Transistor durch, so dass der Ausgang auf Low geht. C4 und R4 sorgen dafür, dass aus der eigentlich pulsierenden Gleichspannung am Ausgang ein schöner, gleichmäßiger Pegel wird. Wenn die Last ausgeschaltet ist, ist der Ausgang des Optokopplers dagegen high.
Der Verbinder rechts unten stellt alle benötigten Signale zur Verfügung. An S1 kann man einen Taster im/am Gerät anschließen, damit der Leistungsschalter auch lokal gesteuert werden kann.
Ein beachtenswertes Detail: Der Optokoppler muss mit 3,3 V versorgt werden. Und da auf der Platine keine 3,3-V-Versorgung vorhanden ist, muss diese Spannung von der Steuerplatine bereitgestellt werden.

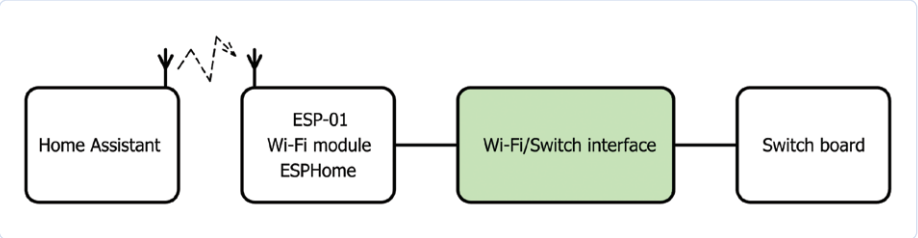


Bild 1. Übersicht des Systems. In diesem Artikel befassen wir uns hauptsächlich mit dem grünen Block.

Das WLAN-Modul ESP-01

Die Aufgaben einer WLAN-Steuerplatine für den Wechselspannungs-Leistungsschalter sind:

- Sie muss zwei digitale Eingänge zum Lesen des Optokoppler-Ausgangs und von S1 bereitstellen
- Sie benötigt zwei digitale Ausgänge zur Steuerung von T1 und T2
- Sie muss 3,3 V für den Optokoppler bereitstellen.

Wenn Sie „vier digitale I/O-Ports und WLAN und 3,3 V" sagen, denken Sie natürlich sofort an ein ESP-01-Modul, da dieses genau vier GPIO-Ports und WLAN hat und mit 3,3 V betrieben wird. Dieses beliebte Modul, das auf dem ESP8266EX basiert, ist günstig und leicht erhältlich (**Bild 3**).
Es gibt jedoch ein großes Aber. Die vier GPIO-Ports des ESP-01 müssen mit Vorsicht behandelt werden, da sie auch bestimmen, wie das Modul nach dem Einschalten arbei-

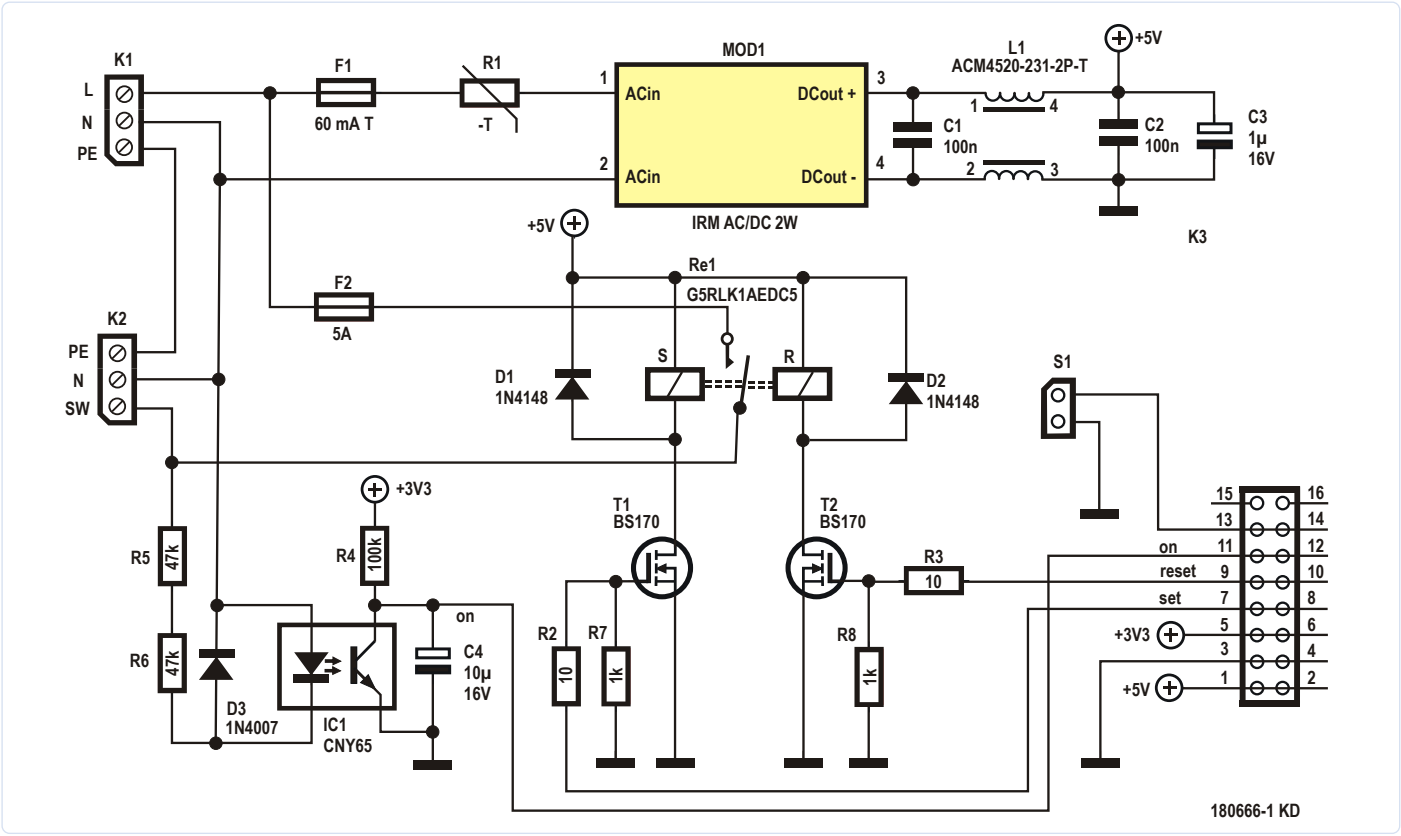


Bild 2. Die Schaltung des Wechselspannungs-Leistungsschalters des Elektor-LoRa-Knotens mit dem Herzstück, einem bistabilen Relais.

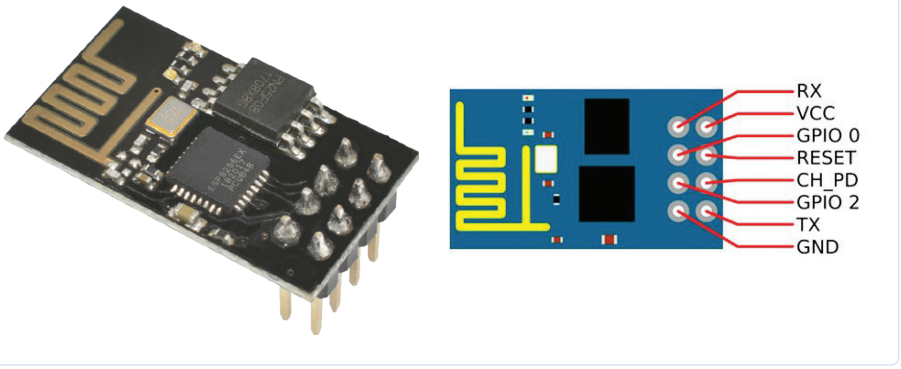


Bild 3. Das kleine und preiswerte ESP-01-Modul ist mit einem ESP8266EX-Mikrocontroller und WLAN ausgestattet. Es besitzt vier I/O-Ports (GPIO 0, GPIO 2, RX und TX) und benötigt eine 3,3-V-Spannungsversorgung (VCC).

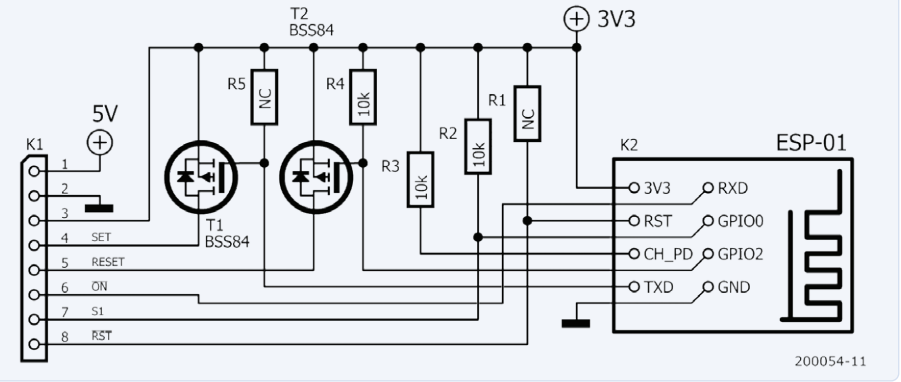


Bild 4. So kann man ein ESP-01-Modul an die Relaisplatine anschließen. Die MOSFETs entkoppeln Ein- und Ausgänge.

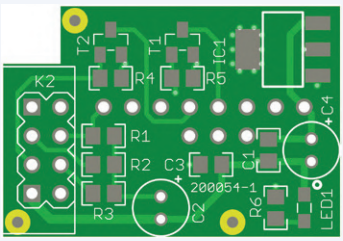
STÜCKLISTE

Widerstände:
Alle 5%, 50 V, 0,1 W, 0805
R1...R5 = 10 k
R6 = 470 Ω

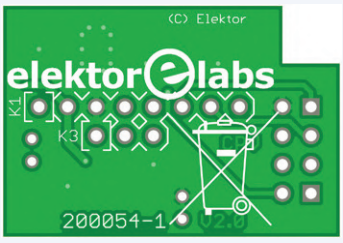
Kondensatoren:
C1,C3 = 100 n, 0805
C1,C4 = 10 µ, 16 V, Raster 2 mm

Halbleiter:
IC1 = LD1117AS33
LED1 = LED, grün
T1,T2 = BSS84

Außerdem:
K1 = 1x8-polige Stiftleiste, Raster 0,1"
K2 = 2x4-polige Buchsenleiste, Raster 0,1"
K3 = 1x3-polige Stiftleiste, Raster 0,1"
Platine 200054-1



shown at 150% of full size



tet. Für den normalen Betrieb müssen GPIO0, GPIO1 und GPIO2 beim Einschalten nämlich High sein, wie dies in den Hinweisen unterhalb der *Pin-Definitions*-Tabelle im Datenblatt des ESP8266EX erwähnt wird. Viele ESP8266-Anwender sind sich dieser Anforderungen an GPIO0 und GPIO2 bewusst, aber nicht jeder kennt die für GPIO1, besser bekannt als TXD oder serieller Ausgang.

Zuordnung von GPIOs zu Steuersignalen

Auf der Relaisplatine sind zwei der vier Steuersignale beim Einschalten immer low, nämlich „Set" und „Reset", bedingt durch die Widerstände R7 und R8. Wenn die Last beim Einschalten zufällig eingeschaltet ist, ist auch das „On"-Signal low, da das bistabile Relais seinen letzten Zustand beibehält, auch im stromlosen Zustand.
Der Eingang S1 ist potentialfrei. Sein Zustand beim Einschalten wird von der WLAN-Steuerplatine bestimmt, es sei denn, jemand drückt zufällig den Taster S1, wenn das System eingeschaltet wird. Alles in allem haben wir es also mit vier Signalen zu tun, die beim Einschalten alle low sein können. Sie werden jedoch an vier Ports angeschlossen, von denen drei beim Einschalten high sein müssen. Die Verwendung eines ESP-01-Moduls ist also in dieser Situation eine kleine Herausforderung.

Die Lösung: Transistoren hinzufügen

Die Lösung, die ich mir ausgedacht habe, sieht wie folgt aus (**Bild 4**):

- RXD (aka GPIO3) ist der einzige Pin, der ohne weiteres angeschlossen werden kann. Ich habe ihn daher mit dem Ausgang des Optokopplers verbunden, da sein Pegel beim Einschalten nicht vorhersehbar ist. RXD wird ein Eingang sein.
- Wenn GPIO0 beim Einschalten low ist, bootet das Modul im Flash-Programmiermodus. Dies kann zum Aktualisieren der Firmware nützlich sein, deshalb habe ich ihn mit S1 verbunden. Daher wird auch GPIO0 als Eingang fungieren.
- GPIO2 und TXD (auch bekannt als GPIO1) bleiben nun übrig und müssen daher als Ausgänge definiert werden. Um sie von den niedrigen Impedanzen zu entkoppeln, die durch R7 und R8 auf der Relaisplatine entstehen, habe ich P-MOSFETs mit je einem Pull-Up-Widerstand an ihren Gates eingesetzt. Dadurch wird sichergestellt, dass beim

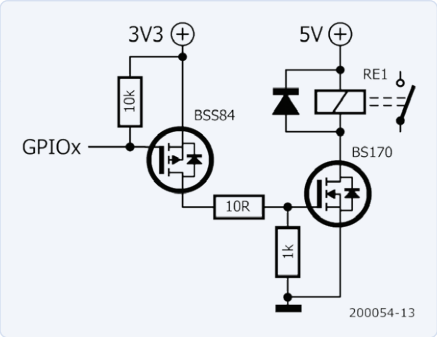


Bild 5. Der Ausgangstreiber: Ein P-MOSFET treibt einen N-MOSFET.

Booten GPIO2 und TXD einen hohen Pegel sehen. Wenn der ESP-01 mit dem Booten fertig ist, können sie als Aktiv-Low-Ausgänge konfiguriert werden.

Der Ausgangstreiber

Der Ausgangstreiber ist in **Bild 5** dargestellt. Wenn der GPIO auf low geht, wird der P-MOSFET BSS84 leitend. Dadurch wird das Gate des N-MOSFET BS170 auf high gezogen, so dass dieser ebenfalls leitet und die Spule des Relais mit Strom versorgt wird. Wenn das GPIO-Signal high ist, sperrt der P-MOSFET BSS84, und aufgrund des 1-kΩ-Pull-Down-Widerstands an seinem Gate auch der N-MOSFET BS170, so dass die Relaisspule nicht mit Strom versorgt wird. Der 10-kΩ-Widerstand am Gate des BSS84 sorgt dafür, dass der GPIO-Pin beim Einschalten auf high gezogen wird.

Auch das noch!

Für die 3,3-V-Stromversorgung habe ich einfach einen Low-Dropout-Regler an die 5-V-Versorgungsleitung angeschlossen (**Bild 6**). In der Praxis werden R1, R6 und LED1 nicht benötigt, da sie bereits auf dem ESP-01-Modul vorhanden sind, weshalb sie mit „NC“ gekennzeichnet sind, also nicht bestückt werden müssen. R5 wird ebenfalls nicht benötigt, aber ich habe ihn vorsichtshalber hinzugefügt. Geeignete Werte für R1 und R5 wären 10 kΩ, für R6 beispielsweise 470 Ω. Für diese Schnittstelle zwischen WLAN-Modul und Schalterplatine habe ich eine kleine Platine entworfen (**Bild 7**), deren Dateien unter [3] zu finden sind.

Die YAML-Datei für ESPHome

Nachdem die Ein- und Ausgänge definiert sind, können wir die ESPHome-Konfigurations-YAML-Datei schreiben (siehe Kasten, downloadbar unter [3]). Sie ist etwas komplizierter als üblich, da das bistabile Relais zwei Pins anstelle von einem verwendet. ESPHome und Home Assistant kennen zwar eine

YAML-Konfiguration für den WLAN-Switch

```
esphome:
  name: wifiswitch
  platform: ESP8266
  board: esp01_1m

wifi:
  ssid: “my_ssid”
  password: “my_passphrase”

# Logging impossible as UART0 pins are
# used for switches & sensors.
logger:
  baud_rate: 0 # Hardware UART off.

# Enable Home Assistant API
api:

# Enable Over-the-Air programming
ota:

output:
  - platform: gpio
    id: relay_set
    pin: GPIO1 # TXD, may not be pulled low at startup.
    inverted: true
  - platform: gpio
    id: relay_reset
    pin: GPIO2 # Do not pull low at startup.
    inverted: true

switch:
  - platform: template
    name: “Wi-Fi switch”
    id: wifiswitch
    turn_on_action:
      # Pulse the Set pin.
      - output.turn_on: relay_set
      - delay: 0.1s
      - output.turn_off: relay_set
    turn_off_action:
      # Pulse the Reset pin.
      - output.turn_on: relay_reset
      - delay: 0.1s
      - output.turn_off: relay_reset
    # Use optocoupler state as switch state.
    lambda: return id(optocoupler).state;

# Pushbutton & optocoupler.
binary_sensor:
  - platform: gpio
    name: “Wi-Fi switch pushbutton”
    id: pushbutton
    pin:
      number: GPIO0 # Do not pull low at startup.
      inverted: true
    on_press:
      then:
      - if:
          condition:
            binary_sensor.is_on: optocoupler
          then:
            switch.turn_off: wifiswitch
          else:
            switch.turn_on: wifiswitch
  - platform: gpio
    name: “Wi-Fi switch optocoupler”
    id: optocoupler
    pin:
      number: GPIO3 # RXD, may be pulled low at startup.
      inverted: true
```

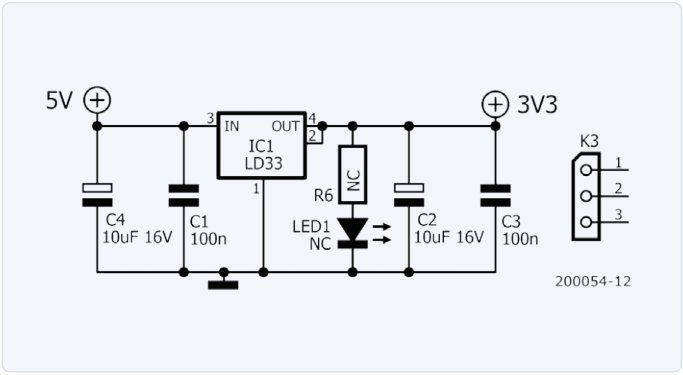


Bild 6. Ein einfacher 5-V-auf-3,3-V-Adapter. K3 sorgt für zusätzliche mechanische Stabilität, wenn das WLAN-Modul auf den Steckverbinder der Relaisplatine gesteckt wird.

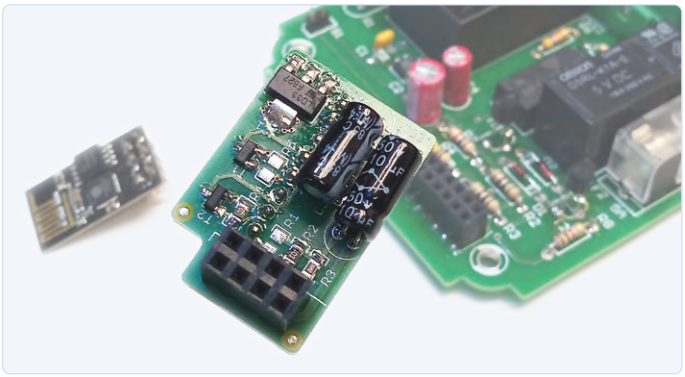


Bild 7. Die Schnittstelle zwischen dem EPS-01-WLAN-Modul und der Relaisplatine ist auf einer kleinen Adapterplatine montiert.

sogenannte Cover-Komponente, die solche Peripheriegeräte steuern kann, aber sie ist eher für Garagentore, Fensterjalousien und ähnliches mit einem Öffnungs- und einem Schließtaster gedacht. Sie haben auch ein spezielles Icon. Unser Schalter ist nur ein Schalter, und deshalb soll er sich wie ein Schalter verhalten und wie einer aussehen. Dies lässt sich mit einem Schaltertemplate erreichen, das es erlaubt, das Verhalten für das Ein- und Ausschalten separat festzulegen. Zuerst definieren wir GPIO1 (TXD) als einen invertierten Ausgang namens *relay_set.* GPIO2 wird als invertierter Ausgang mit der Bezeichnung *relay_reset* deklariert. Außerdem definieren wir GPIO3 als Binären Sensor (active low) für den Optokopplerausgang.

Dann spezifizieren wir einen kurzen 100-ms-Impuls auf dem *relay_set*-Ausgang als Einschaltaktion und dasselbe auf dem *relay_reset*-Ausgang als Ausschaltaktion. Als Schaltzustand geben wir den Zustand des Optokopplers zurück.

Damit der Taster S1 wie erwartet funktioniert, also ein Tastendruck die Last ein- und der nächste ihn ausschaltet, definieren wir GPIO0 ebenfalls als Binären Sensor (active low). Wenn er gedrückt wird, wird zuerst der Zustand des Optokopplers überprüft. Wenn die Last aktiv ist, dann schaltet das Relais sie aus, wenn die Last inaktiv war, dann schaltet das Relais sie ein. Ein *if-then-else* reicht dafür aus.

Fertig!

Nachdem Sie ESPHome mit dieser YAML-Datei kompiliert und das ESP-01-Modul geflasht haben, sollte der WLAN-Schalter im Home Assistant auftauchen und bereit sein für die Automatisierung. Jetzt kann er an seinem Bestimmungsort installiert werden und „etwas Last schalten".

200054-02

Sie haben Fragen oder Kommentare?
Wenn Sie Fragen oder Kommentare haben, wenden Sie bitte ein E-Mail an den Autor (clemens.valens@elektor.com) oder kontaktieren Sie die Elektor-Redaktion (editor@elektor.com).

Ein Beitrag von
Idee, Entwurf, Text und Illustrationen: **Clemens Valens**
Übersetzung: **Rolf Gerstendorf**
Layout: **Giel Dols**

PASSENDE PRODUKTE

- **ESP8266 ESP-01 WLAN-Modul**
  www.elektor.de/17326
- **IoT Home Hacks mit ESP8266**
  www.elektor.de/19158
- **LoRa-gesteuerter Schalter mit Zustandsrückmeldung - Slave-Einheit Leerplatine**
  www.elektor.de/lora-controlled-switch-with-state-feedback-slave-unit-bare-pcb-180666-1

WEBLINKS

[1] **L. Lemmens und M. Claussen, „Der Elektor-LoRa-Knoten", Elektor, März/April 2020**: www.elektormagazine.de/180666-02
[2] **C. Valens, „Hausautomation leicht gemacht", Elektor, September/Oktober 2020**: www.elektormagazine.de/200019-02
[3] **Projektdateien auf Elektor Labs**: www.elektormagazine.com/labs/wifi-switch

Keine Angst vor dem **Mobilfunkmodul**

Warum Sie Ihr eigenes Mobilfunkmodul entwerfen sollten

Von **Tam Hanna** (Slowakei)

Viele halten den Entwurf eines Kommunikationssystems, das auf einem Mobilfunkmodul basiert, für schwarze Magie. Tam Hanna hatte kürzlich das Vergnügen, verschiedene solcher Entwürfe zu begutachten. Hier ist eine Zusammenstellung seiner Notizen, die Ihnen beim Design eines eigenen Mobilfunk-Moduls behilflich sein könnten. Sie werden sehen, dass es weniger schwierig ist, als Sie glauben mögen!

Während fertige 3G- und 4G-Kommunikationsmodule und entsprechende USB-Dongles leicht im Internet bezogen werden können, liegt der Teufel, der diese für Ihre Anwendung untauglich macht und einen Neuentwurf des Systems erfordert, meist im Detail verborgen. Einige Netzbetreiber und Anwendungen verwenden beispielsweise „ungerade" Bänder, die vom Modul nicht unterstützt werden, während bei anderen Systeme das Hinzufügen eines COTS-USB-Dongles nicht möglich ist. Und dann gibt es noch Anwendungen, die ungewöhnliche Antennen (zum Beispiel SMA) oder auch GPS-Funktionalität benötigen.

Erste Schritte

Alles beginnt damit, ein Kommunikationsmodul zu finden, das für die von Ihnen gewählten Kommunikationsbänder geeignet ist und das den Größen- und Kostenkriterien entspricht. Die Qualität des Field-Application-Engineering-Teams des Modulherstellers, des Anbieters oder des Distributors ist ebenfalls wichtig, da Sie möglicherweise irgendwann Unterstützung benötigen. Der Autor empfand den Service von Quectel als besonders und hilfreich, der Gemalto-Distributor YAWiD in Deutschland und der polnische Distributor von Telit waren ebenfalls in Ordnung. Am anderen Ende der Skala ordnete der Autor jedoch den beliebten Schweizer Hersteller U-blox ein, da sein E-Mail-System ihm es nahezu unmöglich machte, auf Anfragen zu antworten. Aber das sind natürlich alles persönliche Erfahrungen. Es hängt viel von Ihrem Standort ab und vielleicht haben Sie ganz andere Erfahrungen gemacht.

Der nächste Schritt ist der Kauf eines Entwicklungs- oder Evaluierungskits für das Kommunikationsmodul. Sobald Sie es vor sich haben, lassen Sie Ihre Software-Ingenieure die Verwaltungssoftware des Moduls zu 100 % zum Laufen bringen, mit Betonung auf 100 %. Lassen Sie keine offenen Enden, die Ihren Entwurf später unbrauchbar machen könnten - und laut Murphy wahrscheinlich auch werden. Das Field Application Engineering Team des Herstellers sollte Ihr erster Ansprechpartner bei möglicherweise auftretenden Softwareproblemen sein. Der hier gezeigte Grad der Zusammenarbeit kann als Lackmustest für die Zusammarbeit bei Problemen mit der Hardware angesehen werden. Sind alle Softwarefragen geklärt, ist es an der Zeit, die Dokumentationen des Kommunikationsmoduls zu besorgen, die sich normalerweise hinter einem Anmeldeformular auf der Website des Herstellers verbergen. Zwei Dokumente sind dabei von besonderem Interesse: der Schaltplan des Evaluation Boards und das Hardware-Design-Dokument, das das eigentliche System beschreibt. Was Sie darin suchen, sind das Platinenlayout (Footprint oder Land Pattern) des Moduls und eine Tabelle, in der die Pins und ihre Funktionen aufgelistet sind.

Hallo Schnittstelle!

Die meisten Kommunikationsmodule kennen nur zwei Möglichkeiten, mit dem Host zu kommunizieren, über eine serielle Schnittstelle oder USB. Andere Schnittstellen wie I²C und SPI sind normalerweise für den Anschluss von Peripheriebausteinen reserviert und für uns hier nicht von Interesse. Beginnen wir mit USB, wegen der einfacheren Treiberintegration die bevorzugte Schnittstelle für Linux-basierte Host-Systeme. Das Modul erscheint normalerweise als ein oder mehrere TTY-Geräte. Obwohl USB 2.0 auch über ein Paar Jumper-Kabel funktioniert, sollten Sie versuchen Sie, die Grundregeln des Platinenlayouts zu beachten und den Stecker so nah wie möglich am Modul zu platzieren. Über die genaue Impedanz der USB-Leitungen müssen Sie sich aber keine Gedanken machen.

Die serielle Schnittstelle ist ein anderes Kaliber. Obwohl der Autor noch kein Linux-basiertes System gesehen hat, das die serielle Schnittstelle für die Kommunikation mit einem Kommunikationsmodul verwendet, lässt er diese Möglichkeit bei seinen Boards häufig zugänglich. Die

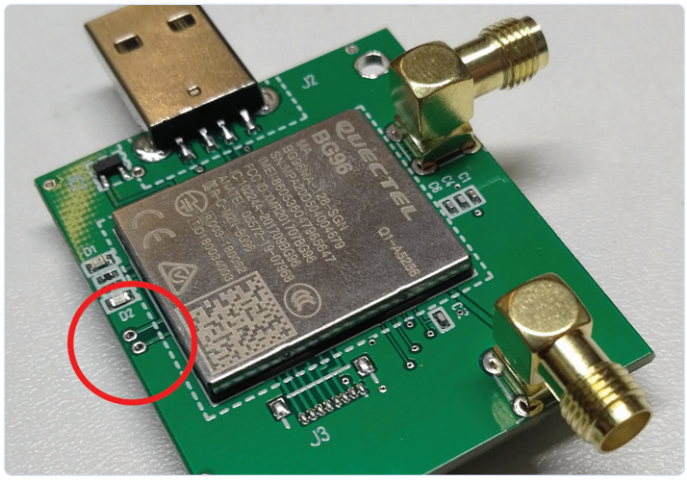
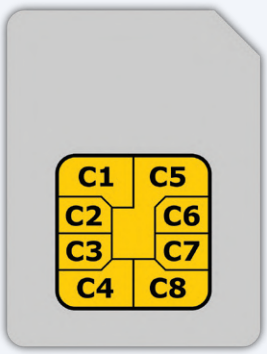


Bild 1. Die beiden freien Pads auf der linken Seite sind mit dem Debug-UART des Moduls verbunden. Ein Verbinder wird nur bei Bedarf bestückt.

Tabelle 1. Die Schnittstellenbelegung der SIM-Karte ist standardisiert.

Pin	Name	Beschreibung
C1	VCC	+5 VDC Spannungsversorgungseingang (optionale Verwendung durch die Karte)
C2	Reset	Reset-Signal, wird zum Zurücksetzen der Kommunikation der Karte verwendet. Wird entweder selbst verwendet (Reset-Signal wird vom Schnittstellengerät geliefert) oder in Kombination mit einer internen Reset-Steuerschaltung (optionale Verwendung durch die Karte). Wenn ein interner Reset implementiert ist, ist die Spannungsversorgung auf VCC obligatorisch.
C3	Clock	Versorgt die Karte mit einem Taktsignal, von dem das Timing der Datenkommunikation abgeleitet wird.
C4	Reserved	AUX1, wird optional für USB-Schnittstellen und andere Zwecke verwendet.
C5	GND	Masse (Referenzspannung)
C6	Vpp	Programmierungsspannungseingang (optional). Dieser Kontakt kann verwendet werden, um die zum Programmieren oder Löschen des internen nichtflüchtigen Speichers erforderliche Spannung zu liefern. ISO/IEC 7816-3:1997 bezeichnet dies als Programmierspannung: ein Eingang für eine höhere Spannung zur Programmierung von persistenten Speichern (z. B. EEPROM). ISO/IEC 7816-3:2006 bezeichnet es als SPU, entweder für Standard- oder proprietäre Verwendung, als Eingang und/oder Ausgang.
C7	I/O	Eingang oder Ausgang für serielle Daten (halb-duplex) zum integrierten Schaltkreis innerhalb der Karte.
C8	Reserved	AUX2, optional verwendet für USB-Schnittstellen und andere Verwendungen.



(Quelle: pinoutguide.com [1])

Hersteller von Modulen bezeichnen in der Regel eine der UARTs als Debug-Schnittstelle, und wenn sie nicht offen zugänglich ist, kann ein fehlerhafter Entwurf nicht analysiert werden, wenn er an das Labor des Herstellers geschickt wird. Daraus folgt, dass die serielle Schnittstelle zumindest in irgendeiner Weise frei und am besten mit einem simplen Verbinder ausgestattet sein sein sollte, der später dann in der Massenproduktion nicht bestückt wird (**Bild 1**).

Diverse Teile

Fast alle Kommunikationsmodule stammen aus einer großen Familie. Auch wenn man einen Entwurf so gestalten kann, dass er mehrere Modultypen unterstützt, ist dies meist nicht von besonderem Interesse. Insbesondere bei Geräten mit geringer Leistung wie LTE M1 kann man durch den Verzicht auf familienübergreifende Kompatibilität ein kleineres Netzteil verwenden und so Kosten sparen. Auch werden in manchen Fällen Reset-Schaltungen und ähnliches nicht für alle Familienmitglieder benötigt, was eine weitere, wenn auch kleine Kosteneinsparung ermöglicht.

Falls ein wohldefinierter Reset-Impuls benötigt wird, verwenden Sie eine Mikrocontroller-Reset-Supervisor-Schaltung wie den STM1001 von STMicroelectronics. Diese SOT23-Bauteile kosten nicht viel und sind zuverlässiger als eine Bastelei mit RC-Gliedern.

Zur Stromversorgung muss nicht viel gesagt werden. Die Wahl zwischen einem Schaltregler oder einem linearen Regler bleibt weitgehend Ihnen überlassen. Sparen Sie aber nie an Entkopplungskondensatoren. 4G-Module zum Beispiel sind notorisch „sprunghaft" in ihrer Leistungsaufnahme.

Eine weitere Sache, auf die Sie achten sollten, sind „magic pins". Lesen Sie sorgfältig die Hardware-Beschreibung für jeden Pin, um sicherzustellen, damit Sie verstehen, was zu tun ist. Einige von diesen Pins müssen auf eine bestimmte Spannung gelegt werden, während andere, zum Beispiel der „Boot-Mode-Selector"-Pin, durch ein Pad oder zumindest eine Durchkontaktierung zugänglich gehalten werden sollten. Wenn Sie sich nicht sicher sind, können Sie eine Struktur wie in **Bild 2** in das Layout aufnehmen, die Ihnen beim Prototyp alle Optionen bietet.

SIM-Karte

Der Anschluss der SIM-Karte an das Kommunikationsmodul ist immer eine interessante Herausforderung. SIM-Kartenhalter haben normalerweise nummerierte Kontakte, während das Modul Beschriftungen verwendet (Vcc, Rst, Data und Clk). Glücklicherweise sind die Kontakte auf der SIM-Karte genormt (**Tabelle 1**). Je nach Schaltung kann ein Pull-up-Widerstand zwischen Data und Vcc erforderlich sein, manchmal werden Entkopplungskondensatoren an Rst, Clk und Data verwendet. Der Autor lässt in der Regel die ESD-Schutzdioden weg, da der SIM-Kartenhalter bei seinen Schaltungen für den Endanwender nicht zugänglich ist.

Einige Kommunikationsmodule erkennen das Vorhandensein einer SIM-Karte durch einen Spannungspegel an einem bestimmten Pin. Wenn Sie diese Funktionalität nicht nutzen wollen, achten Sie darauf, diesen Pin auf eine Spannung zu legen, die „SIM-Karte vorhanden" entspricht. Wenn Sie diese Funktion nutzen wollen, achten Sie darauf, wie sie an Ihrem physikalischen SIM-Kartenhalter implementiert ist. Achten Sie auch auf die mechanischen Abstände. Manche SIM-Kartenhalter lassen sich aufklappen, bei anderen müssen Sie die (unflexible) SIM-Karte aus einer bestimmten Richtung einschieben.

Zum Schluss möchten wir Sie noch von dem Gedanken befreien, dass Kommunikationsmodule auf mindestens vierlagigen Platinen montiert werden müssen. Mit allen modernen PCB-Designprogrammen lassen sich automatisch Kupferebenen einrichten. Eine Massefläche auf beiden Seiten der Platine, die mit vielen Durchkontaktierungen verbunden ist, reicht in der Regel völlig aus.

Die Antenne ist wichtig

Bleibt noch die Antenne. Bisher hat es der Autor geschafft, die Verantwortung für das Design und die Evaluierung von PCB-Antennen zu vermeiden. Der Vektor-Netzwerkanalysator HP 8753C, den er einst auf Wunsch eines Kunden kaufte, hat sich seither als erstklassige „Hangar Queen" entpuppt, die sowohl stete Wartung als auch teure Kalibrierungs-Kits benötigt. Ohne Platinenantenne braucht man auf der Platine einen Verbinder für eine externe Antenne, sei es ein SMA- oder U.FL-Typ. Beachten Sie auf jeden Fall, welche Möglichkeiten dafür das Gehäuse bietet.

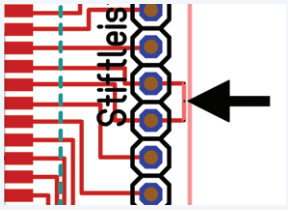


Bild 2. Die gekennzeichnete Leiterbahn kann bei Bedarf einfach aufgetrennt und neu verbunden werden.

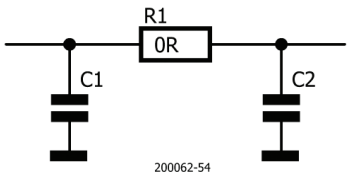


Bild 3. Wenn eine Antennenanpassung geplant ist, wird ein pi-Netzwerk benötigt. Es kann in der Produktion umgangen werden, indem man R1 auf 0 Ω setzt und C1 und C2 nicht einsetzt.

Versuchen Sie, die physikalische Verbindung zum Antennenstecker so kurz wie möglich zu halten. Verwenden Sie ein Online-Tool zur Impedanzberechnung (transmission line calculator) und versuchen Sie, die empfohlene Geometrie einigermaßen einzuhalten, das heißt, gute Masse auf beiden Seiten und keine Durchkontaktierungen im Signalpfad. In der Regel gibt es keine ernsthaften Probleme, solange die Leiterbahnen kurz gehalten werden.

Die meisten Hersteller empfehlen in ihren Referenzdesigns, für den Fall der Fälle eine Pi-Schaltung ähnlich der in **Bild 3** aufzunehmen. In der Produktion wird sie meist mit einem 0-Ω-Widerstand und nicht bestückten Kondensatoren außer Funktion gesetzt. Wenn Sie keine erweiterte Antennenanpassung geplant haben, können Sie in vielen Fällen bessere Ergebnisse erzielen, indem Sie den Abstand zwischen dem Modul und dem Antennenanschluss verringern und das pi-Netzwerk weglassen.

Beachten Sie, dass es zwei Arten von GPS-Antennen gibt: aktive und passive. Aktive Antennen benötigen eine Phantomversorgung, die aus einigen wenigen passiven Bauteilen besteht und die normalerweise im Referenzschaltplan des Modulherstellers zu finden ist.

Bestellung eines Prototypen

Nun ist es an der Zeit, die eigentliche Hardware zu produzieren. Abhängig von Ihrem (möglichst gutem) Verhältnis zu den Applikationsingenieuren des Modulherstellers können Sie das Platinenlayout und den Schaltplan zur Überprüfung schicken. Der Autor war besonders beeindruckt von den Review-Services von Quectel.

Das Hauptproblem bei der Bestückung eines Prototyps per Hand ist das Löten der Bauteile. Zwar konnte der Autor erfolgreich einige Module in seinem Reflow-Ofen löten, aber wegen der erforderlichen Präzision der Stencil- und Bauteilplatzierung und des Fehlens eines Röntgeninspektionsgeräts ist die Bestückung eine Aufgabe, die man besser an eine auf den Bau von Prototypen spezialisierte Firma auslagert (zusammen mit einer freundlichen Erinnerung zur Röntgeninspektion). Wenn es sich bei dem montierten Prototyp um einen USB-Dongle handelt, verwenden Sie für die ersten Tests einen USB-Hub mit eigener Stromversorgung - das schützt Ihre teure Workstation vor Schäden am Mainboard!

Frisch gewagt ...

Wie bei so vielen anderen Aufgaben ist der Anfang die größte Hürde, die es zu überwinden gilt. Fast alle vom Autor gestalteten Entwürfe funktionierten schon beim ersten oder zweiten Versuch. Den Entwurf zum Laufen zu bringen, ist jedoch nur ein kleiner Teil der Gesamtlösung, zumal Software-Probleme die Systemintegration erschweren können. Doch keine Angst, bewaffnet mit den Erfahrungen, die der Autor auf die harte Tour gesammelt hat und hier präsentiert, können auch Sie den Job erledigen.

200062-02

STAATLICHE ZERTIFIZIERUNG!

Wenn Sie ein braver Bürger sein wollen: Die Fertigstellung des Moduls kann eine Zertifizierung erfordern. Dies ist jedoch ein Thema für eine ganz andere Geschichte.

Haben Sie Fragen oder Kommentare?
Haben Sie Fragen oder Kommentare zu diesem Artikel? Kontaktieren Sie Elektor unter editor@elektor.com.

Ein Beitrag von:
Idee und Text: **Tam Hanna**
Übersetzung: **Rolf Gerstendorf**
Redaktion: **Clemens Valens**
Layout: **Giel Dol**

WEBLINK

[1] **Pinout der SIM-Karte**: https://pinoutguide.com/Memory/SmartCardIso_pinout.shtml

PASSENDE PRODUKTE

› **SeeedStudio RF Explorer 3G Combo – Handheld Spectrum Analyzer (SKU 19099)**
www.elektor.de/seeedstudio-rf-explorer-3g-combo-handheld-spectrum-analyzer

Zeiterfassung
mit ESP32 und Toggl
Heimarbeiten mit dem M5Stack

Von **Mathias Claußen** (Elektor)

Möchte man die Stunden nachverfolgen, die man für Elektronikprojekte aufwendet, kann man verschiedene Dienste einsetzen. Wir verwenden hier Toggl.com, um Ihnen zu zeigen, wie man Web-Anfragen über HTTPS stellt und wie man grundlegende GUI-Funktionen implementiert. Das alles wird mit dem M5Stack gemacht, der eine schnelle Prototyping- und Entwicklungsplattform für ESP32-basierte Projekte bietet.

Bild 1. M5Stack Core.

In der heutigen Zeit, in der Sie viel von zu Hause aus arbeiten, kann es nützlich sein, Ihre Tätigkeiten und die Zeit, die Sie dafür aufwenden, zu protokollieren. Das macht es auch einfacher, Ihre Arbeit zu besprechen, sei es mit Ihren Kollegen oder Ihren Kunden, und hilft bei der Erstellung von Rechnungen. Es gibt viele Lösungen wie openTimetool, Toggl und Kimai auf dem Markt, die ein Zeit- und Projektmanagement ermöglichen. Für diesen Artikel habe ich den Web Service *Toggl Track* verwendet, eine webbasierte Lösung, die auch ein Plugin für Ihren Webbrowser oder eine App für Ihr Telefon zur Zeiterfassung anbietet. Toggl Track bietet eine offene API für die Schnittstelle zu einem Zeiterfassungssystem, die es Ihnen ermöglicht, Ihre eigene App oder Ihr eigenes Hardwaregerät für die Zeiterfassung mit den Diensten des Unternehmens zu verbinden.

KISS

Zeiterfassung ist für einige von uns eher eine Pflicht, die wir erfüllen müssen, als etwas, das wir gerne tun. Deshalb ist für ein Zeiterfassungssystem „keep it simple stupid“ (KISS) das Beste, was Sie einem Benutzer bieten können. Er braucht nur einen Knopf zum Drücken, um „auf Arbeit“ zu sein und einen für „Feierabend“. Da Toggl ein Online-Dienst ist, brauchen wir etwas, das sich mit dem Internet verbinden kann (drahtlos oder über Kabel) und unsere Anfragen senden kann. Für diese Aufgabe kommt uns natürlich der ESP32 in den Sinn. Um ein Durcheinander von Drähten und Bauteilen auf einem Breadboard zu vermeiden, folgen wir der KISS-Route und greifen zu einem fertigen *M5Stack Core Basic* (**Bild 1**), dessen Dokumentation online zugänglich ist [1]. Zusätzlich zum EPS32 erhalten wir dadurch drei Taster, ein Display, eine Batterie und einige andere Peripheriefunktionen in einem schönen und kompakten Gehäuse. Alles, was übrig zu bleiben scheint, sind ein paar Zeilen Code.

Grundlegendes Konzept

Wir brauchen ein Gerät, das auf den Toggl-Webdienst zugreifen und auf mindestens einen Taster reagieren kann, der signalisiert, ob man im (Home-) Office sitzt oder unterwegs ist. Beim M5Stack bedeutet dies, dass wir für die Internetverbindung ein WLAN nutzen, wenn ein Access Point

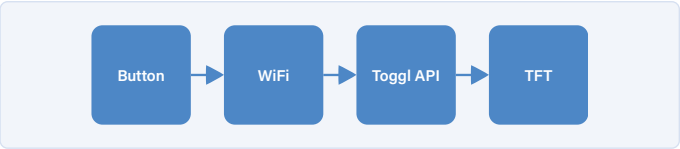


Bild 2. Datenfluss zum Toggl-Dienst.

Bild 3. Datenfluss zum Abrufen des aktuellen Status.

in Reichweite ist und wir davon ausgehen können, dass wir permanent online sind. Wie Sie im Foto sehen können, haben wir drei Tasten zur Auswahl. Und da es ein LCD gibt, warum nicht ein paar Grafiken anzeigen und den Arbeitsstatus (Arbeit oder Abwesenheit) visualisieren?

Wie kompliziert ist der Zugriff auf Toggl? Theoretisch nicht so schwer, da die API gut dokumentiert ist. Das heißt, man drückt einen Knopf und sendet damit einen entsprechenden Befehl an den Toggl-Webservice. Daraufhin wird eine Statusänderung durchgeführt und diese wird dem Benutzer zurückgemeldet. Das hört sich zunächst einmal einfach an. **Bild 2** zeigt den Datenfluss von einem Tastendruck zur Toggl-API. Für eine LCD-Aktualisierung zeigt **Bild 3** einen Fluss zum Toggle-Server irgendwo im Netz und zurück zum ESP32. Das klingt einfach und sieht auch einfach aus.

Für den WLAN-Teil wurde dies schon oft in Elektor praktiziert, so dass wir auf den Code vieler Elektor-Projekte mit einem ESP32 zurückgreifen können. Damit haben wir schon einen modularen Webserver, einen Mechanismus, um Zeit- und OTA-Updates handzuhaben, was mehr Luxus ist als wir benötigen. Für die Kommunikation mit dem Toggl-Dienst wird ein Stack von Protokollen verwendet. **Bild 4** zeigt den Kommunikationsstack und die beteiligten Bibliotheken. Das ESP32-Arduino-Framework enthält eine *WIFIClient*-Bibliothek zur Netzwerkkommunikation, die eine Verbindung über das WLAN, mit dem der ESP32 verbunden ist, zu Servern ermöglicht. Darüber hinaus gibt es im ESP32 Arduino-Framework die *HTTPClient*-Bibliothek, die das HTTP-Protokoll und HTTP-Anfragen verwaltet. Da Toggl den Austausch von Daten über HTTP im JSON-Format erfordert, benötigen wir eine zusätzliche Bibliothek, um JSON zu verarbeiten, in diesem Fall die *ArduinoJSON*-Bibliothek.

Neben der Kommunikation benötigen wir auch eine Bibliothek zur Ansteuerung des Displays im M5Stack. Eine Bibliothek, die mit dem Display des M5Stacks zusammenarbeitet und bereits bekannt ist, ist *TFT_eSPI*, die einen großen Satz an fix und fertigen Funktionen zum Zeichnen von Grafiken liefert. Da diese Bibliothek kompatibel ist, alle benötigten Funktionen zum Zeichnen von Text und Grafiken bereitstellt und bereits in früheren Projekten eingesetzt wurde, wäre es eine Sünde, sie nicht zu verwenden. Einen kurzen Überblick über den Grafik-Stack, den TFT_eSPI bietet, zeigt **Bild 5**. Für den Einstieg ist eine kurze Einführung in verschiedene Themen hilfreich. Ich beginne mit HTTP, das für den Zugriff auf den Toggl-Dienst erforderlich ist.

HTTP

Das Hypertext Transfer Protocol (HTTP), das 1989 am CERN entwickelt wurde, wird verwendet, um eine Ressource durch einen Client (zum Beispiel Ihrem Webbrowser) anzufordern und eine Antwort von einem Webserver zu erhalten. Wenn Sie beispielsweise *www.elektor.com* öffnen wollen, sendet Ihr Client eine Anfrage an den Webserver. Dazu wird eine TCP/IP-Verbindung aufgebaut und ein Anforderungsstring wie in **Listing 1** gesendet.

Die erste Zeile, ein HTTP-Header mit `GET`, teilt dem Server mit, dass wir etwas anfordern. Der `/` sagt dem Server, dass wir die Standardwebseite geliefert bekommen wollen. Mit `HTTP/1.1` teilen wir dem Server mit, dass wir HTTP in der Version 1.1 benutzen. Die Arbeit an der dritten Protokollgeneration HTTP/3 hat schon begonnen, aber für die meisten unserer Anwendungsfälle ist HTTP/1.1 ausreichend.

Bei der zweiten Zeile, die den Host angibt, ist es gut zu wissen, dass die Namen, die

Listing 1. HTTP-Request

```
GET / HTTP/1.1
Host: www.elektor.com
User-Agent: curl/7.55.1
Accept: */*
```

Sie in Ihrem Browser eingeben, von einem DNS-Dienst in eine IP-Adresse übersetzt werden. Das bedeutet, dass elektor.com in 83.96.255.227 übersetzt wird, die IP-Adresse, die für die TCP/IP-Verbindung verwendet wird. Hinter dieser IP-Adresse kann ein Webserver stehen, der nicht nur elektor.com, sondern auch elektor.de oder elektor.nl bedient. Damit der Webserver weiß, welche Seite Sie wünschen, ist die zweite Zeile mit `Host` erforderlich.

Mit `User-Agent` in Zeile 3 teilen wir dem Webserver mit, welche Art von HTTP-Client wir sind, in diesem Fall curl. Dies ermöglicht es dem Webserver, Seiten so zu präsentieren, dass sie für Ihren Webbrowser optimiert sind, weil sich beispielsweise Safari anders verhält als Firefox. Der letzte HTTP-Header in Zeile 4 weist unseren Client an, alle Arten von Inhalten zu akzeptieren. Jede HTTP-Anfrage wird durch eine Leerzeile beendet.

An dieser Stelle stellt der Webserver eine Antwort auf unsere Anfrage zusammen, die ebenfalls mit einem Satz von HTTP-Headern beginnt, wie in **Listing 2** zu sehen. In der ersten Zeile stehen die Protokollversion und ein Antwortcode. Da wir nach HTTP/1.1 gefragt haben, wird der Server auch in

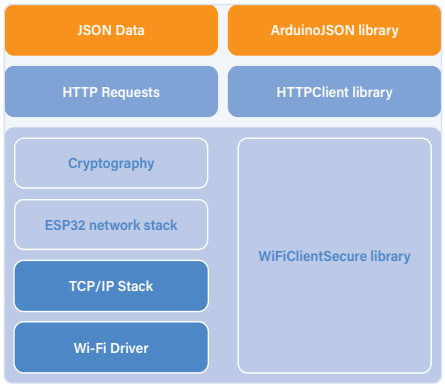


Bild 4. Protokollstack und Bibliotheken.

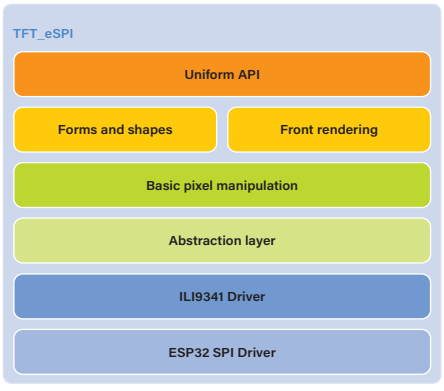


Bild 5. Zeichenfunktionen für das LCD.

Listing 2. HTTP-Response.

```
HTTP/1.1 200 OK
Server: unknown
Date: Tue, 17 Nov 2020 13:34:04 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Strict-Transport-Security: max-age=63072000
Pragma: no-cache
Expires: -1
Cache-Control: no-store, no-cache, must-revalidate, max-age=0
Accept-Ranges: bytes
Strict-Transport-Security: max-age=63072000
```

Index	Binär	Kodiert
0	000000	A
1	000001	B
2	000010	C
3	000011	D
4	000100	E
5	000101	F
6	000110	G
7	000111	H
8	001000	I
9	001001	J
10	001010	K
11	001011	L
12	001100	M
13	001101	N
14	001110	O
15	001111	P
16	010000	Q
17	010001	R
18	010010	S
19	010011	T
20	010100	U
21	010101	V
22	010110	W
23	010111	X
24	011000	Y
25	011001	Z
26	011010	a
27	011011	b
28	011100	c
29	011101	d
30	011110	e
31	011111	f
32	100000	g
33	100001	h
34	100010	i
35	100011	j
36	100100	k
37	100101	l
38	100110	m
39	100111	n
40	101000	o
41	101001	p
42	101010	q
43	101011	r
44	101100	s
45	101101	t
46	101110	u
47	101111	v
48	110000	w
49	110001	x
50	110010	y
51	110011	z
52	110100	0
53	110101	1
54	110110	2
55	110111	3
56	111000	4
57	111001	5
58	111010	6
59	111011	7
60	111100	8
61	111101	9
62	111110	+
63	111111	/
PADDING		=

Tabelle 1. BASE64-Kodierungstabelle

dieser Protokollversion antworten. Der Antwortcode, den wir erhalten, ist `200` und signalisiert, dass unsere Anfrage bearbeitet werden konnte. Es folgen weitere Header, die zusätzliche Informationen liefern.
Wenn wir nur an dem angeforderten Inhalt interessiert sind, können wir den Header überspringen. Der Header wird mit einer Leerzeile vom in Listing 2 nicht gezeigten angeforderten Inhalt getrennt. Allein im Header können 8 KB und noch mehr Daten stehen. Der Body und seine Länge können durch `Transfer-Encoding: chunked` oder `Content-Length:<length>` bestimmt werden, wobei ersteres für Inhalte mit unbekannter Länge (zum Beispiel dynamisch generierte Webseiten), letzteres für Inhalte mit im Voraus bekannter Größe wie Bilder verwendet wird.
Es gibt noch weitere http-Methoden, zum Beispiel POST, bei der Daten vom Client an den Server übertragen werden, was etwa beim Absenden von Informationen in einem Webformular verwendet wird. Auch die PUT-Methode kann verwendet werden, um Daten an einen Webserver zu übertragen. Weitere Informationen über weitere definierte Methoden finden Sie im Mozilla Developer Network MDN [2].

Von HTTP zu HTTPS

Eine Sache, die bei der Entwicklung von HTTP nicht berücksichtigt wurde, war die integrierte Sicherheit für das Kommunikationsprotokoll. Die Informationen werden als Klartext übertragen und jeder mit ein wenig Netzwerkwissen kann sie lesen und/oder verändern. HTTPS, wobei das „S“ für „secure“ steht, fügte dem Kommunikationsstack eine zusätzliche Schicht hinzu. Der Server und der Client kommunizieren immer noch über das HTTP-Protokoll, aber statt einer Klartextkommunikation über eine TCP/IP-Verbindung wird der gesamte Verkehr zwischen beiden Parteien zunächst über Transport Layer Security (TLS) abgewickelt, das den Austausch von kryptografischen Schlüsseln und die Verschlüsselung verwaltet.
Das bedeutet auch, dass wir mehr Rechenleistung, Flash und RAM benötigen, wenn wir HTTPS einsetzen, da die Ver- und Entschlüsselung durchgeführt und die zusätzlichen Routinen für TLS in unsere Software aufgenommen werden müssen. Auf einem modernen Computer ist genügend Rechenleistung vorhanden, aber bei kleineren Embedded-Systemen kommt es eventuell zu einer Verlangsa-

mung der Kommunikation, wenn die Ver- und Entschlüsselung sowie die Schlüsselverwaltung und der Schlüsselaustausch rein softwaremäßig erfolgt. Daher finden Sie in modernen MCUs kryptografische Beschleuniger, die die Verschlüsselungsarbeit von der MCU auslagern. Beim ESP32 steht zu diesem Zweck eine Hardware-Beschleunigung zur Verfügung.

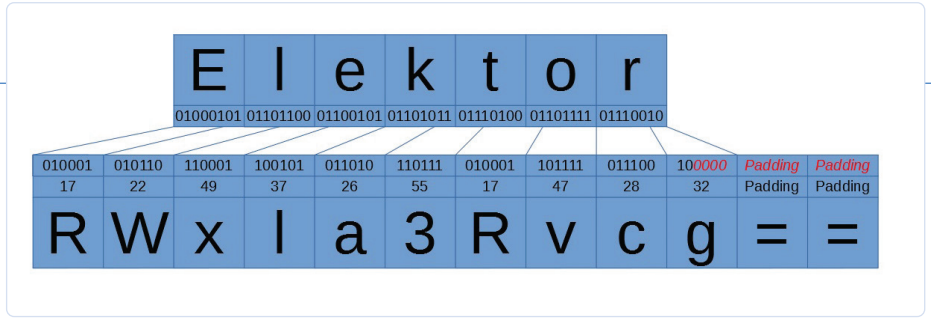


Bild 6. Elektor kodiert in BASE64.

Request und Response

Der Toggl-Dienst bietet eine offene und dokumentierte API [3] zum Austausch von Daten. Die API funktioniert, indem POST-, PUT- und GET-Anfragen an eine URL mit definiertem Header und einer Payload gesendet werden. Alle Aktionen erfordern einen Schlüssel zur Authentifizierung, der von der Toggle-Webseite bezogen werden kann. Mit den bereitgestellten Informationen kann das Erstellen einer Anfrage beginnen. Wir verwenden die Klasse `WiFiClientSecure` in Verbindung mit dem http-Client-Objekt. Die Methode, mit einem Server zu kommunizieren, ist bewährt, wir müssen das Rad also nicht neu erfinden, wenn es um die Verarbeitung von http-Daten geht.

Was wir „von Hand" machen, ist die Generierung des Autorisierungs-Tokens, einer Base64-kodierten Kombination aus Benutzernamen und Passwort. Als Eingabe für die Base64-Kodierung haben wir einen API-Schlüssel von Toggl und den `:api_token` als String. Da der API-Schlüssel nicht im Klartext gesendet wird, kann er Zeichen enthalten, die mit den reservierten Zeichen für den Header selbst kollidieren. Um dennoch jede Art von Daten in diesen Anfragen einschließen zu können, können wir sie BASE64-kodieren. Normalerweise haben unsere Daten innerhalb eines Bytes einen Wert zwischen 0 und 255. BASE64 verwendet nur 64 Werte (Übersetzung siehe **Tabelle 1**), die auf das Äquivalent der ASCII-Zeichen `A-Z, a-z, 0-9, +,/` und `=` zum Auffüllen abgebildet werden. Wenn wir zum Beispiel den String *Elektor* in BASE64 kodieren, erhalten wir *RWxla3Rvcg==* (**Bild 6**). Wenn E-Mail-Anhänge aus ähnlichen Gründen ebenfalls mit BASE64 kodiert werden, kann sich dadurch die Größe des Anhangs um etwa 37 % gegenüber der ursprünglichen Größe erhöhen.

Nach den Anmeldeinformationen, die dem HTTP-Client zur Autorisierung übergeben wurden, kann die Anfrage gestellt werden. Das folgende Beispiel zeigt, wie eine GET-Anforderung implementiert und die zurückgegebenen Informationen verarbeitet werden. Das Code-Schnipsel in **Listing 3** gibt einige Einblicke. Die Funktion `GET()` gibt den Antwortcode des Servers für unsere Anfrage zurück, oder, falls ein Kommunikationsproblem vorliegt, einen Wert kleiner als Null. Der Antwortcode kann verschiedene Werte einnehmen [4]. Der am häufigsten bekannte Code ist *404 Resource not found*. Hier wird der Code *200 Response okay* erwartet und daraufhin geprüft. Wenn Daten vom Server empfangen werden, können diese mit `getString()` zur weiteren Verarbeitung abgefragt werden.

Es gibt ähnliche Codes für PUT und POST, aber mit einigen kleinen, wichtigen Unterschieden, wie in **Listing 4** gezeigt. Die Funktion `addHeader`, die vor `PUT()` verwendet wird, fügt der Anfrage zwei zusätzliche Header hinzu. Der erste ist Content-Type und teilt dem Webserver mit, welche Art von Daten wir an ihn senden werden, der zweite ist die Content-length. Hier wird HTTP plötzlich „ein wenig anders". Normalerweise senden wir nämlich nach dem Header Daten, zum Beispiel Text, getrennt durch zwei neue Zeilen und endend mit zwei neuen Zeilen. Das funktioniert für Text, aber wenn wir nun ein JPEG-Bild an den Server senden, kann es aus einer beliebigen Kombination von beliebigen Bytes

Listing 3. Snippet für HTTP-Request

```
httpclient.begin(client,link);
   httpclient.setReuse(true);
   httpclient.setAuthorization((const char*)b64.c_str());
   int httpCode = httpclient.GET();
    if (httpCode > 0) { //Check for the returning code
        if(httpCode!=200){
            Serial.println("Response Error");
            String payload = httpclient.getString();
            Serial.println(httpCode);
            Serial.println(payload);
            *error = 1;
        } else{
            *Result = httpclient.getString();
        }
    } else {
        Serial.print("Request Error");
        Serial.println(httpclient.errorToString(httpCode));
    }
    httpclient.end();
```

Listing 4. Hinzugefügter HTTP-Header

```
httpclient.setAuthorization((const char*)b64.c_str());
   httpclient.addHeader("Content-Type","application/json");
   httpclient.addHeader("Content-length","0");
   int httpCode = httpclient.PUT((uint8_t*)(NULL),0);
      if (httpCode > 0) { //Check for the returning code

      . . .
```

Listing 5. Verbindung wiederverwenden

```
httpclient.begin(client,link);
httpclient.setReuse(true);
httpclient.setAuthorization((const char*)b64.c_str());
httpclient.addHeader("Content-Type","application/json");
int httpCode = httpclient.POST(payload);
   if (httpCode > 0) {

. . .
```

Listing 6. Beispiel für einen JSON-String

```
{ "name": "Test", "value": 1351824120,"array": [ 123.45, 321.89 ] }
```

Listing 7. JSON-Erzeugung

```
DynamicJsonDocument doc(capacity);
JsonObject time_entry = doc.createNestedObject("time_entry");
   if(description.length()>0){
       time_entry["description"] = description.c_str();
   }
time_entry["created_with"] = "ESP32";
serializeJson(doc, payload);
```

Listing 8. DeserializerJSON

```
DeserializationError er = deserializeJson(doc, Result.c_str());
 if (er) {
      //something went wrong
         } else {
         if(false == doc["data"].isNull() ){
             JsonObject data = doc["data"];
             uint32_t data_id = data["id"];
             const char* data_start = data["start"];
            if(false == data["description"].isNull()){
                 const char* data_description = data["description"];
                  Element->description = String(data_description);
             } else {
                 Element->description = "";
             }
```

Listing 9. JSON-String in Toggl

```
{
"data":
   {
          "id":436694100,
          "pid":123,
          "wid":777,
          "billable":false,
          "start":"2013-03-05T07:58:58.000Z",
          "duration":1200,
          "description":"Brew some coffee",
          "tags":["billed"]
   }
}
```

bestehen. An dieser Stelle kommt die *Content-length* ins Spiel. Header werden durch zwei neue Zeilen von den Daten getrennt. Mit der eingestellten *Content-length* können Sie dann jede Art von Zeichen- oder Binärdaten senden. Dem Webserver wird mit der angegebenen *Content-length* mitgeteilt, wie viele Bytes er zu erwarten hat.

Für POST müssen wir nur den Inhaltstyp angeben, der übertragen werden soll. Es wird keine eigentliche Nutzlast übertragen. Würden wir eine Nutzlast oder einen Inhalt größer als 0 Byte anliefern, würde der http-Client von sich aus einen Content-Length-Header hinzufügen. In diesem speziellen Fall, da wir keinen Inhalt senden und die Länge auf Null gesetzt ist, fügt er den *Content-Length*-Header überhaupt nicht hinzu. Da sich aber der Server beschwert, wenn die Angabe zur Inhaltslänge fehlt, müssen wir sie selbst mit der Funktion `addHeader` hinzufügen, wie in Listing 4 gezeigt.

Wie Sie in **Listing 5** sehen können, ist `setReuse()` auf true gesetzt. Nach einer Anfrage, sei es POST, PUT oder GET, wird die Verbindung mit dem Server normalerweise vom Client geschlossen. Wenn `setReuse()` auf true gesetzt ist, wird diese Verbindung offen gehalten und für die nächste Anfrage wiederverwendet. Dies spart Zeit. Außerdem gibt es zurzeit ein Fehler irgendwo im Netzwerkstack, bei dem der Speicher nicht richtig freigegeben wird. Dies könnte früher oder später zu Verbindungsabbrüchen führen, da dem ESP32 der freie Speicher ausgeht.

Datenaustausch mit JSON

JSON steht für JavaScript Object Notation und beschreibt einen Weg, wie leichtgewichtige Daten zwischen zwei Systemen ausgetauscht werden können. Alle Daten werden in einen String gepackt, der wie in **Listing 6** aussehen kann. Dieser String muss generiert und geparst werden. Die meisten Umgebungen, die Daten für Webanwendungen verarbeiten und austauschen, verwenden JSON und verfügen über eingebaute Funktionen oder zusätzliche fertige Bibliotheken für die JSON-Generierung und das Parsen. Dies gilt auch für kleine Geräte wie einen Arduino oder ESP32.

Eine Bibliothek, die mit dem Arduino Framework und dem ESP32 nutzbar ist, ist *ArduinoJSON* und kann JSON-Strings parsen und generieren. Da Toggl JSON

zum Datenaustausch verwendet, bietet sich diese Bibliothek für unsere Zwecke an. Aus der Toggl-API erhalten wir die Informationen, die in JSON-Strings erwartet werden. Diese Dokumentation beschreibt auch, wie Daten generiert werden sollen, wenn Informationen gesendet werden müssen. Für einen neuen Zeiteintrag müssen wir einen JSON-String generieren, der ein *time_entry*-Objekt enthält. Innerhalb dieses *time_entry* ist ein *description*-String erforderlich und ein *created_with*-String optional für den neuen Eintrag.

Der Code in **Listing 7** erzeugt ein neues `DynamicJsonDocument doc` mit einer definierten Kapazität auf dem Heap-Speicher des ESP32. Das bedeutet, dass der Speicher mit `malloc()` allokiert und später mit `free()` aus dem nicht statisch verwendeten Speicher des ESP32 freigegeben wird. Das Gegenteil wäre ein `StaticJsonDocument`, das statisch im RAM alloziert und lokal in einer Funktion auf dem Stack abgelegt wird. StaticJsonDocument ist zwar schneller, aber da es lokal an den Stack gebunden ist, kann die Größe geringer sein als bei DynamicJsonDocument. Die Funktion `serializeJson()` erzeugt den gewünschten String, der gesendet werden kann.

Die Dekodierung von JSON kann mit der Funktion `deserializeJson()` erfolgen. Sie gibt alle Fehler zurück, auf die sie beim Parsen stößt. Der Codeteil ist in **Listing 8** dargestellt. Der Zugriff auf die Elemente innerhalb des JSON ist oben notiert. Der von Toggl zurückgegebene String sieht wie in **Listing 9** aus. Der Code prüft nach der Deserialisierung, ob die Objektdaten vorhanden sind, also ob wir ein gültiges Ergebnis haben.

Danach greift der Code auf die Elemente innerhalb *data* zu und holt sich nur die Werte, die später für die Anzeige und das Datenhandling benötigt werden, nämlich *id*, *start* und *description*. Letzteres ist eine Besonderheit, denn dieses Feld ist nur dann in *data* integriert, wenn es einen Wert in der Toggl-Datenbank hat. Deshalb wird mit `data["description"].isNull()` überprüft, ob der Eintrag tatsächlich existiert, bevor versucht wird, darauf zuzugreifen. Andernfalls generiert die Bibliothek eine Ausnahme (exeption).

Für den gesamten Datenaustausch ist die *toggleClient*-Bibliothek zuständig. Hier wird alles Nötige unterstützt, nämlich das Anzeigen des aktuellen Eintrags, das Starten eines neuen Eintrags und das Stoppen des aktuell laufenden Eintrags.

Listing 10. LCD-Initialisierung

```
tft.init();
tft.invertDisplay( true );
tft.setRotation(1);
tft.fillScreen(TFT_WHITE);
```

Da dieser Code mehr als Ausgangspunkt denn als fertige Bibliothek gedacht ist, darf jeder Leser versuchen, Verbesserungen und Fehlerkorrekturen beizusteuern.

Zeichnen von Pixeln

Die Daten müssen nicht nur gesammelt und übermittelt, sondern auch irgendwie angezeigt werden. Zum Zeichnen wird die *TFT_eSPI*-Bibliothek verwendet, die Sie vielleicht vom Elektor-Projekt *GUI with Touch* [5] her kennen und die mit verschiedenen Displays und ESP32-Versionen verwendet werden kann. Da diese Bibliothek das Display des M5Stack unterstützt, kann auch etwas Code von anderen Elektor-Projekten wiederverwendet werden. Im Library-Ordner muss die Datei *User_Setup_Select.h* modifiziert werden, um die darin enthaltene Konfiguration für den M5Stack zu verwenden. In der Datei *TogglButtonM5.ino* wird mit der Zeile `TFT_eSPI tft = TFT_eSPI();` ein neues TFT_eSPI-Objekt erzeugt, das später zum Zeichnen verwendet werden kann.

Für die Initialisierung des Displays benötigen wir in der `setup()`-Funktion nur vier Codezeilen (**Listing 10**). Das `inverted display` muss auf *true* gesetzt werden, um die richtige Farbreihenfolge für das M5Stacks-Display einzuhalten. Mit `setRotation(1)` wird der Bildschirm so gesetzt, dass sich sein unterer Rand genau über den M5Stack-Tasten befindet. Nach dieser Codezeile können alle Möglichkeiten der Pixelmanipulation, die diese Bibliothek anbietet, genutzt werden.

Ein kleiner Hinweis zur Zeichenleistung des ILI9341-basierten Displays. Diese Displays sind daraufhin optimiert, ganze Display-Bilder oder größere „Brocken“ von Display-Daten zu verarbeiten. Ein pixelweiser Zugriff verlangsamt das Zeichnen mindestens um das Siebenfache, eher noch mehr. Dies liegt an der Art und Weise, wie die Befehle und Zugriffe auf einzelne Pixel erfolgen. Dennoch: Die TFT_eSPI Bibliothek zeichnet in optimierter Weise und mit optimalen Zugriffszeiten auf das Display, etwas Besseres können wir uns kaum wünschen. Neben dem Setzen von Pixeln oder dem Füllen des Bildschirms mit einzelnen Farben bietet die Bibliothek auch andere Zeichenfunktionen wie Linien, Rechtecke und die Verwendung von Druckfunktionen, um Strings auf dem Bildschirm auszugeben.

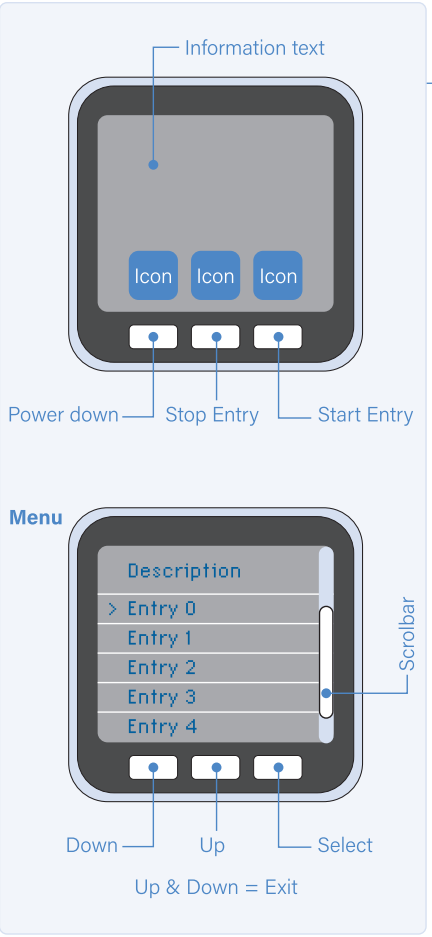


Bild 7. Konzeptzeichnung der GUI.

Das GUI und ein Menü

Der M5Stack verfügt über ein Display und drei Tasten. Andere Lösungen zur Zeiterfassung bieten in der Regel nur eine einzige Taste zum Starten und Stoppen einer Eingabe. Da der M5Stack auch mit einer Batterie betrieben werden kann, kann aber eine Ausschalttaste sehr praktisch sein. **Bild 7** zeigt das GUI-Konzept und die groben Positionen für die wenigen anzuzeigenden Elemente.

Das Zeichnen von Text ist recht einfach. Die wichtigsten Elemente, die angezeigt werden sollen, sind die Beschreibung, wenn es denn eine gibt, und die Startzeit. Wir verwenden die Symbole aus der offenen Symbolbibliothek [6]. Im Main-Screen (**Bild 9**) kann ein neuer Eintrag gestartet, gestoppt und das Gerät ausgeschaltet werden. Keep it stupid simple!

Wenn wir einen neuen Toggl-Eintrag starten, enthält dieser keine Beschreibung

Bild 8. DAS Hauptmenü des GUI.

Bild 9. Auswahl der GUI-Beschreibung.

und im Grunde ist dies auch nicht nötig. Doch ein bisschen Luxus wäre schon toll, wenn man zumindest aus einigen vordefinierten Beschreibungen wählen könnte. Die Eingabe von Beschreibungen mit nur drei Tasten ist zwar prinzipiell möglich, aber sicher nicht benutzerfreundlich. In Anlehnung an das Projekt *Wecker mit Dreifach-Anzeige* [7] wurde ein einfaches Menüsystem entwickelt, das mit der *TFT_eSPI*-Bibliothek verwendet werden kann. Bildschirmauflösung und Ausrichtung des Menüsystems entspricht auch der Konfiguration beim Toggl-Button. Der Code kann in dieses Projekt verpflanzt werden, um ein Menü wie das in Bild 9 zu präsentieren.

Listing 11. ShowMenuSettingsList

```
void Menus::ShowMenuSettingsList(uint8_t selected_idx, bool refresh){

  /* Draw Headline */
  if(true == refresh){
    _lcd->fillRect(0,0,320,40,_lcd->color565( 45,47,50 ) );
    _lcd->setTextColor(_lcd->color565( 112,116,122 ),TFT_BLACK);
    _lcd->setFreeFont(FSB18);
    _lcd->setCursor(160- ( _lcd->textWidth("Description") / 2) ,30);
    _lcd->print("Description");
  }
  /* Headline done */

  /* Draw scrollbar */
  _lcd->fillRect(303,43,15,197,TFT_WHITE);
  _lcd->drawRect(300,40,20,200,_lcd->color565( 112,116,122 ));
  _lcd->drawRect(301,41,18,198,_lcd->color565( 112,116,122 ));
  _lcd->drawRect(302,42,16,196,_lcd->color565( 112,116,122 ));
  _lcd->fillRect(305,45 +(196/GetUsedEntryCount())*selected_idx
   ,10,(196/GetUsedEntryCount())-2 , TFT_DARKGREY);
  /* Scrollbar done */

  /* Menu entries */
  uint8_t startidx=0;
  uint8_t endindex=0;
  if(selected_idx>4){ //We can display 5 Elements
    startidx = selected_idx -4 ;
  }
  if((startidx+5)>GetUsedEntryCount()){ //Limit last drawn item
      endindex=GetUsedEntryCount();
  }else{
      endindex=startidx+5;
  }
  /*Draw up to five elements*/
  for(uint8_t i=startidx;i<endindex;i++){
     DrawSettingsMenuEntry( (40*(i-startidx)),
       DescriptionArray[i].c_str() ,( i==selected_idx ) );
  }
  /* Menu entries drawn */
}
```

Die Logik für das Menü steckt in der `RenderMenu()`-Funktion. Je nach Zustand zeichnet die Funktion das Menü und gibt true zurück. Wenn kein Menü gezeichnet wird, gibt die Funktion false zurück, was bedeutet, dass der Inhalt der Anzeige nicht verändert wurde. Wenn true zurückgegeben wird, sollte es vermieden werden, anschließend einen weiteren Zeichenvorgang auszuführen. Ein vereinfachtes Flussdiagramm ist in **Bild 10** zu sehen.

Die Funktion `RenderMenu()` enthält die Menülogik, aber das Zeichnen selbst wird von `ShowMenuSettingsList()` erledigt. Diese Funktion zeichnet den Menükopf und bis zu fünf Menüeinträge. Die Menülogik sorgt dafür, dass der ausgewählte Eintrag in einem gültigen (sichtbaren) Bereich bleibt. Die Zeichenroutine errechnet die Startposition und zeichnet fünf Einträge als Liste auf den Bildschirm. Auch ein Scrollbalken, wie er in vielen Windows-basierten Systemen zu sehen ist, wird berechnet und mit einem kleinen Indikator für die Position innerhalb der Liste gezeichnet.

Der Code von `ShowMenuSettingsList()` in **Listing 11** kann in drei Teile unterteilt werden. Der erste davon ist die Überschrift. Um Verarbeitungszeit zu sparen, wird diese nur bei Bedarf neu gezeichnet. Die Zentrierung des Überschriftentextes erfolgt, indem die halbe Textlänge in Pixeln berechnet und der Wert von der halben Anzeigebreite subtrahiert wird. Dies funktioniert, wenn der Text kleiner als 160 Pixel ist und kann zu erstaunlichen Ergebnissen führen, wenn der Text länger ist. Der zweite Teil ist die Bildlaufleiste an der rechten Seite des Menüs, die sich aus mehreren Rechtecken zusammensetzt. Die Länge des Scrollbal-

kens wird aus der Anzahl der Einträge innerhalb des Menüs errechnet. Dies funktioniert nur, wenn die Liste weniger als 196 Einträge enthält, ansonsten können unvorhersehbare Dinge mit dem Menü passieren.
Für den dritten Teil werden der Statusindex und der Endindex für die zu zeichnenden Listeneinträge berechnet. Da wir nur fünf Elemente zeichnen, müssen wir diesen Bereich entsprechend des gewählten Index verschieben. Die drei Teile sind für die angezeigte Liste verantwortlich. Da jeder Eintrag innerhalb der Liste gleich aussieht, wird dieser Eintrag mit einer eigenen Funktion gezeichnet, die nur einen Offset zum Zeichnen in der richtigen Höhe und ein Flag benötigt, wenn dieser Eintrag gewählt ist. Damit haben wir ein schlankes Menüsystem geschaffen, das für unsere Zwecke funktioniert und einige grundlegende Menükonzepte zeigt.

Web-basierte Konfiguration

Die Software wurde nicht von Grund auf neu geschrieben. Ich habe Teile aus früheren Projekten wiederverwertet, auch den für die webbasierte Konfiguration. Wenn die beiden linken Tasten während des Bootens gedrückt werden, startet die Software einen Access-Point, den man mit einem Computer oder einem portablen WLAN-fähigen Gerät erreichen kann. Andernfalls versucht die Software, sich mit dem konfigurierten WLAN zu verbinden. Das Starten eines Webservers auf dem ESP32 ist sehr einfach und wird von der Software erledigt, nachdem das WLAN gestartet ist und uns eine Weboberfläche zum Einstellen des WLANs (**Bild 11**) und eine zur Benutzerauthentifizierung mit dem Toggl-API-Schlüssel (**Bild 12**) präsentiert.

Erfahrungswerte und Weiterentwicklungen

Obwohl der Toggl-Button funktioniert und die effektive Zeiterfassung ermöglicht, gibt es noch Raum für Verbesserungen. In diesem Artikel habe ich nur einige der vielen Aspekte moderner IoT-Geräte angesprochen, insbesondere wenn sie mit irgendeiner Art von Webserver oder Webbrowser interagieren. Begriffe wie WLAN, HTTP-Server, HTTP-Client, TLS, JSON, JavaScript, HTML, GUI und C/C++ mögen wie Modewörter klingen, aber sie sind tatsächlich die wesentlichen Bestandteile für dieses einfache Projekt. Sich mit HTTP und HTTPS, JSON auf der einen, GUI-Design auf der anderen Seite zu

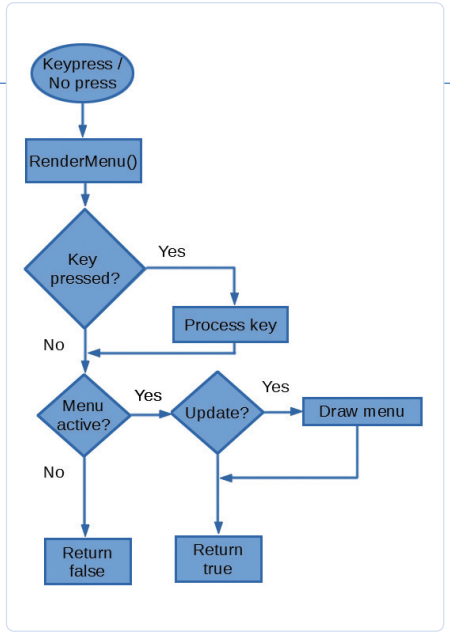


Bild 10. Flussdiagramm zum Menüsystem.

Anzeige

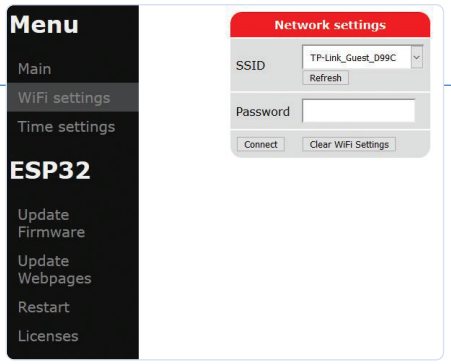


Bild 11. Die WLANKonfigurationsseite.

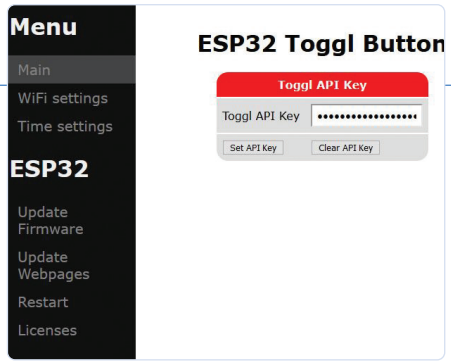


Bild 12. Eingabe des Toggl-API-Schlüssels.

beschäftigen und gleichzeitig eine webbasierte Konfigurationsoberfläche bereitzustellen, war in den frühen Tagen des IoT ungewöhnlich. Aber heutzutage ist dies das absolute Minimum für ein vernetztes Gerät. Diese Mischung aus verschiedenen Bereichen - darunter grundlegende Webentwicklung, JSON, HTML, C/C++ und JavaScript - könnte ein ganzes Buch füllen. Und wenn einmal ein solches Buch veröffentlicht wird, wird Elektor sicherlich darüber berichten.

Während ich diesen Artikel schrieb, kamen mir einige Ideen, wie man den Button noch benutzerfreundlicher gestalten und Funktionen hinzufügen könnte, die die Zeitabrechnung verbessern. Die erste Verbesserung wäre die Umstellung der GUI auf LVGL. Was die GUI betifft, bringt uns dies zum Thema „Erfahrungswerte“. Wenn Sie eine GUI zu Ihrem Projekt hinzufügen wollen, suchen Sie nach etwas, das sich bewährt hat und verfügbar ist. Es gibt sogar Lösungen, die eine freundliche Lizenz für Ihr Projekt haben. Ein eigenes Menüsystem aus Spaß an der Freude zu implementieren mag toll sein, aber man kann seine Zeit auch sinnvoller verbringen.

Wenn Sie für den ESP32 das Arduino-Framework und die Bibliotheken verwenden, sollten Sie sich das GitHub-Repository und die „open issues“ ansehen. Dies kann eine gute Ressource sein, um Probleme zu umschiffen oder bekannte Einschränkungen zu umgehen. Der Code für diesen Toggl-Button ist auf der Elektor-GitHub-Seite [9] veröffentlicht und bietet eine bequeme Möglichkeit, den Code zu klonen und zu erforschen, da seine Historie erhalten bleibt. Wenn Sie Fehler finden oder Vorschläge für den Code haben, können Sie die GitHub-Issues-Funktion nutzen. Da dieses Projekt viele Themen berührt, geben Sie mir ein Feedback, wenn bestimmte Themen in einem zukünftigen Artikel ausführlicher erklärt werden sollen.

200631-02

Haben Sie Fragen oder Kommentare?
Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Schicken Sie eine E-Mail an den Autor unter mathias.claussen@elektor.com oder kontaktieren Sie Elektor unter editor@elektor.com.

PASSENDE PRODUKTE

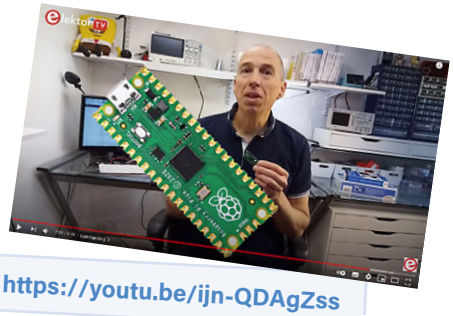
- **M5Stack – ESP32 Basic Core Development Kit**
  www.elektor.de/m5stack-esp32-basic-core-development-kit
- **M5Stack ESP32 Arduino MPU9250 Development kit**
  www.elektor.de/m5stack-esp32-arduino-mpu9250-development-kit
- **W. Gay, FreeRTOS for ESP32-Arduino (Elektor 2020)**
  www.elektor.de/freertos-for-esp32-arduino

Ein Beitrag von:
Entwicklung, Text und Diagramme: **Mathias Claußen**
Illustrationen: **Patrick Wielders**
Übersetzung: **Rolf Gerstendorf**
Layout: **Giel Dols**

WEBLINKS

[1] **M5Core Basic**: https://docs.m5stack.com/#/en/core/basic
[2] **HTTP | MDN**: https://developer.mozilla.org/en-US/docs/Web/HTTP
[3] **Toggl.com API-Dokumente**: https://github.com/toggl/toggl_api_docs
[4] **HTTP Response Status Codes | MDN**: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status
[5] **GUI mit Touch**: www.elektormagazine.com/articles/guiwithtouchforesp32rpico
[6] **Open Icon Library**: https://sourceforge.net/projects/openiconlibrary/
[7] **3-Display-Wecker mit TFT-Bildschirm**: www.elektormagazine.de/magazine/elektor-201805/41484
[8] **Leichte und vielseitige Grafikbibliothek**: https://lvgl.io/
[9] **Projektseite bei GitHub**: https://github.com/ElektorLabs/200631_ESP32_Toggl_Button

Raspberry Pi Pico und der Mikrocontroller RP2040



https://youtu.be/ijn-QDAgZss

Sehen Sie das „Raspberry Pi Pico Review" im Elektor-TV!

Von **Mathias Claußen** (Elektor) und **Luc Lemmens** (Elektor)

Der Raspberry Pi ist ein SBC (Single Board Computer), auf dem Linux läuft – aber der neue Raspberry Pi Pico ist anders. Auch wenn es sich immer noch um ein Produkt der Raspberry Pi Foundation handelt, tritt dieses Board an, die Welt von Arduino, STM32 BluePill und ESP32 zu erobern. Der Pico schlägt eine Brücke von leistungsstarken, Linux-basierten SBCs zur Welt der Embedded-Mikrocontroller.

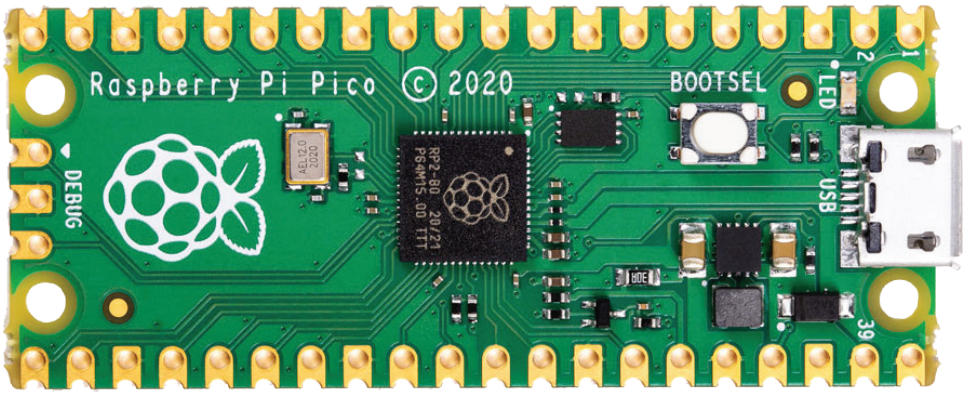


Bild 1. Ein Blick auf den Pico.

Schon wieder ein neuer Raspberry Pi? Allerdings! In **Bild 1** können Sie den niegelnagelneuen Raspberry Pi Pico bewundern. Vorweg: Der Pico ersetzt keine ältere RPi-Variante, sondern ist eine Erweiterung der Raspberry-Pi-Familie. Obwohl es sich ebenfalls um einen Raspberry Pi handelt, ist dieses Modell völlig anders als alle bisherigen RPi-Boards. Ein Pico ist ein mit einem Dual-Core-ARM-Controller bestücktes Board, das nur die nackte, zum Betrieb notwendige Elektronik mitbringt. Auch wenn der Name Assoziationen zu einer Art neuem RPi Zero nahelegt, handelt es sich doch um etwas ziemlich anderes. Ein wichtiger Unterschied: Auf dem Board eines Raspberry Pi Pico befindet sich - anders als bei den bisherigen Raspberry-Pi-Boards - kein SoC von Broadcom. Stattdessen ist der hier aufgelötete Chip RP2040 eine Eigenentwicklung der Raspberry Pi Foundation.

Beim RP2040 handelt es sich nämlich um einen Dual-Core-Mikrocontroller auf der Basis von ARM Cortex-M0+. Es ist also keiner der üblichen Linux-fähigen SoCs. Ein Pico gehört eher zur Welt der Mikrocontroller-Boards, die heute von Arduino, STM32-BluePill und ESP32-basierten Boards dominiert wird, und in der RISC-V-basierte SBCs (wie das Modell GD32VF103) immer beliebter werden. Ein Pico-Board ist mit nur ca. 5 € ungewöhnlich preiswert, weshalb ein Pico durchaus mit billigen Clones von Arduino Nano und STM32 BluePill konkurrieren kann. Der Pico ist auch günstiger als ein ESP32 Pico Kit.

Nachfolgend geht es vor allem um die Spezifikationen und Funktionen des Pico und seiner MCU RP2040. In diesem Zusammenhang ist das Unboxing-Video von Clemens Valens auf Elektors YouTube-Kanal [1] sowie die Übersicht unter [2] sehr informativ.

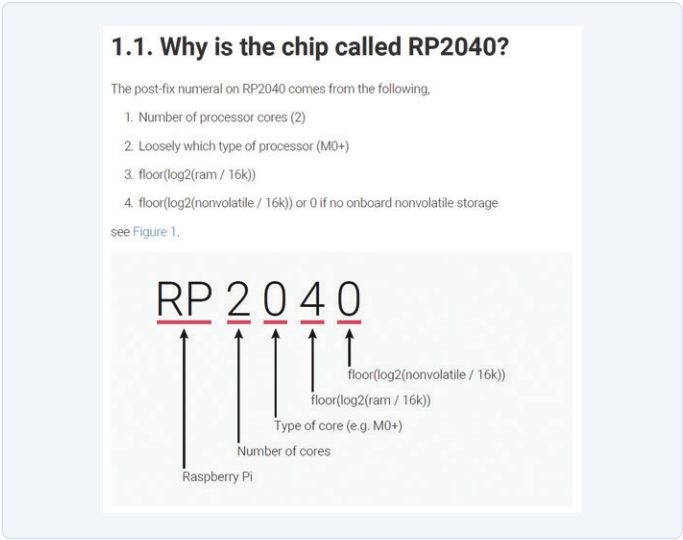


Bild 2. Nomenklatur [9].

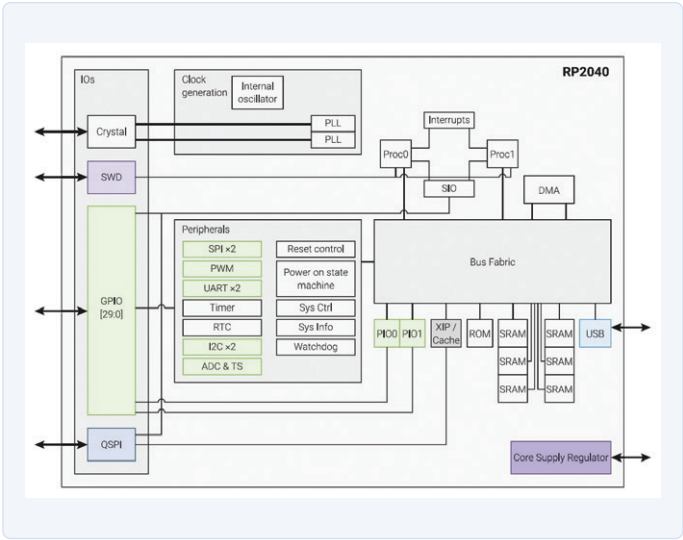


Bild 3. Innenleben des RP2040 [9].

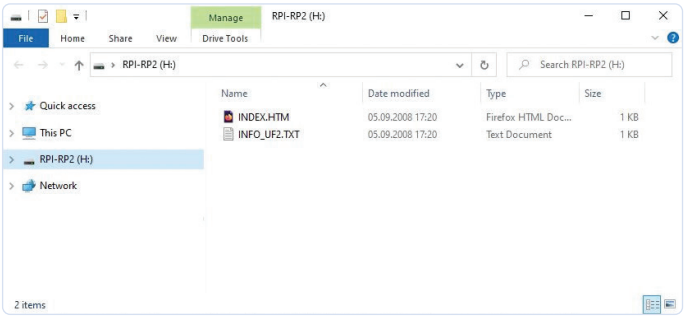


Bild 4. Der Raspberry Pi Pico als Massenspeicher.

Tabelle 1. Technische Daten des Raspberry Pi Pico

CPU-Typ	Cortex M0+
Anzahl der Kerne	2
Maximaler Takt	133 MHz
SRAM	264 KB in 6 Bänken
Flash intern	0 kB
Flash extern	2 MB QSPI (bis zu 16 MB unterstützt)
GPIOs	26 (inkl. 4 ADC)
USB	1.1, Host/Slave
ADC	12 bit @ 500 kS/s
ADC-Kanäle	5 (inkl. Temperatursensor)
SPI	2
UART	2
I^2C	2
PWM	16 Kanäle
Timer	1x 64 bit
RTC	ja
Besonderheiten	Programmierbare IO-Zustandsmaschine, Boot-ROM mit USB-Massenspeicher-Bootloader

Erste Eindrücke von Rasberry Pi Pico

Der Dual-Core-Mikrocontroller RP2040 kann mit bis zu 133 MHz Takt arbeiten. Dual-Core-Cortex-M-MCUs sind heute nichts Ungewöhnliches. Sie enthalten aber normalerweise nicht zwei Cores des Typs Cortex-M0+, sondern bieten leistungsfähigere Cortex-M4-, Cortex-M7- oder gar die neueren Cortex-M33-Cores. Die Bezeichnung der MCU RP2040 folgt der Nomenklatur von **Bild 2**, die ein Indiz für weitere Varianten darstellt, die noch kommen könnten. Unsere Nachfrage bei der Raspberry Pi Foundation brachte allerdings keine weiteren Hinweise. Die technischen Spezifikationen des RP2040 sind in **Tabelle 1** aufgelistet.

Bild 3 zeigt die Blockschaltung von Peripherie und Funktionsblöcken des RP2040. Eine leicht zu übersehende Besonderheit ist das Fehlen von integriertem Flash-Speicher. Auf dem Pico-Board befindet sich deshalb mit dem W25Q16JUXIQ ein kleines IC neben der MCU, das 2 MB NOR-Flash bietet. Ein RP2040 unterstützt bis zu 16 MB an externem Speicher, so dass eine Speichererweiterung eine Frage des Austauschs dieses Flash-Chips ist. Ein Pico verfügt auch über einen USB-1.1-Port, der als Host oder Slave konfiguriert werden und USB-Peripherie mit bis zu 12 Mbit/s anbinden oder den Pico als Slave selbst in ein USB-Device verwandeln kann.

In der MCU RP2040 steckt ein Boot-ROM, das es ermöglicht, neuen Code von einem PC zu laden. Um die Notwendigkeit spezieller Programmer-Tools zu vermeiden, wird der Flash-Speicher des angeschlossenen Boards von einem PC als Massenspeicher angezeigt. Ein Pico kann durch einfaches Kopieren neuer Firmware-Dateien auf diesen Speicher programmiert werden (**Bild 4**) - eine besondere einfache und bequeme Art, ein Mikrocontroller-Board mit neuer Firmware zu versorgen.

Auch die Spannungsversorgung ist unkritisch. Das Board funktioniert nicht nur mit 5 V via Micro-USB-Buchse, sondern akzeptiert auch Spannungen von 1,8 bis 5,5 V über einen DC/DC-Buck-Boost-Converter an seinem Pin VSYS. Auf diese Weise kann man einen Pico z.B. sehr einfach durch einen Lithium-Akku oder einen Satz von zwei oder drei NiMH-Akkus versorgen.

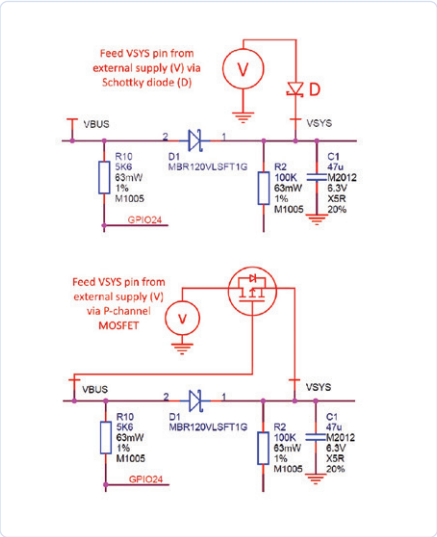


Bild 5. Wie das Board mit Strom versorgt werden kann [3].

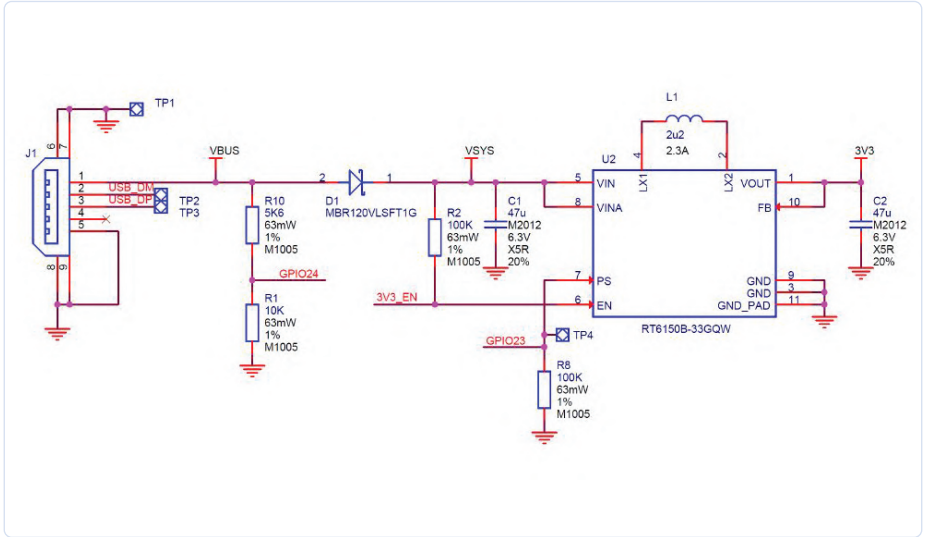


Bild 6. Der On-Board-Buck/Boost-Wandler [3].

Bild 5 des Datenblatts [3] illustriert den Anschluss verschiedener Spannungsquellen. Das verwendete DC/DC-Buck-Boost-IC ist ein RT6150, dessen Datenblatt unter [4] zu finden ist. Dieses IC benötigt laut Schaltung von **Bild 6** nur wenige externe Bauteile.
40 Pins des Pico sind an die Seiten der Platine geführt, wodurch 26 GPIOs (darunter drei ADC-Eingänge des RP2040 zugänglich sind. Die anderen Pins führen größtenteils die Versorgungsspannungen und Masse. Maximal sind Spannungen bis 3,3 V an allen GPIO-Pins erlaubt. Auf der Unterseite der Platine (**Bild 7**) ist jeder Pin beschriftet. Auf dem Pico-Board befinden sich drei zusätzliche Pins (SWCLK, SWIO und GND) für den Serial-Wire-Debug-Port, der die Programmierung des RP2040 und auch das Debugging ermöglicht.

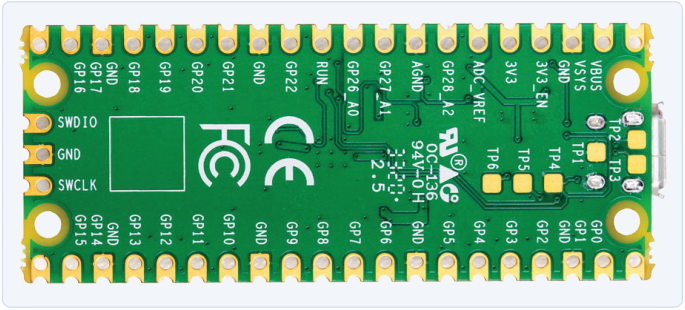


Bild 7. Ansicht der Platine von unten (mit Pin-Bezeichnungen).

Peripherie und programmierbarer Input/Output-Block

Ein Pico bietet mit zwei UARTs, zwei I²C-Controllern, zwei SPI-Controllern, 16 PWM-Kanälen, einem 12-Bit-ADC mit 500 kS/s, einem integrierten Temperatursensor, einer Echtzeituhr und Timern die grundlegenden GPIO-Funktionen und die übliche Peripherie. Ebenfalls enthalten aber weniger üblich für Mikrocontroller sind eine als Host oder Slave konfigurierbare USB-Schnittstelle sowie State Machines für acht PIOs (Programmable IOs).
Mit den State-Machines lassen sich verschiedene (zusätzliche) Schnittstellen wie UART, I²C, I²S und SPI, sowie ein SD-Karten-Interface, VGA, DPI und vieles mehr realisieren. Das Datenblatt des RP2040 enthält einen eigenen Abschnitt, der die Verwendung und Beispielprogramme erläutert. **Bild 8** zeigt einen Block mit vier State Machines. Diese Zustandsautomaten können einen Satz von neun Befehlen ausführen: JMP, WAIT, IN, OUT, PUSH, PULL, MOV, IRQ und SET. Da diese Zustandsmaschinen unabhängig von den beiden CPU-Kernen laufen können, lassen sich damit Schnittstellen realisieren, die nicht schon als MCU-Peripherie vorhanden sind. Außerdem entlasten diese autonomen State Machines die CPU-Kerne.

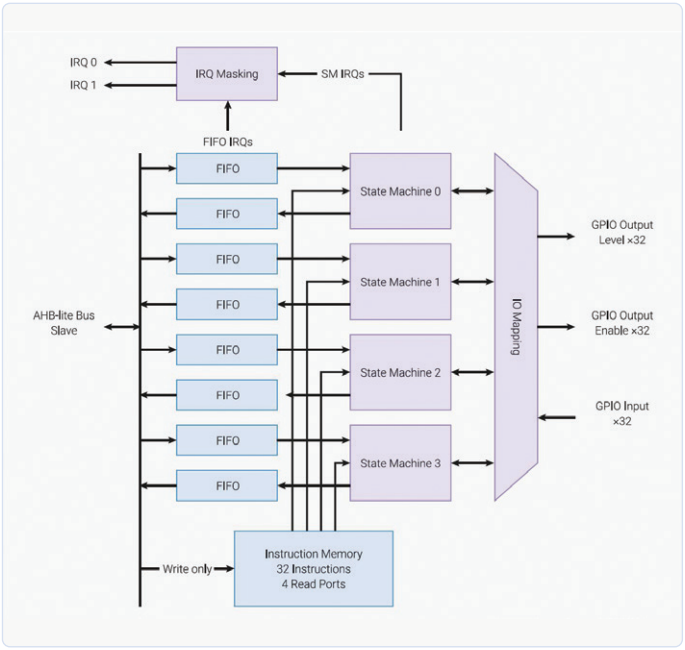


Bild 8. I/O mit programmierbarer State-Machine [9].

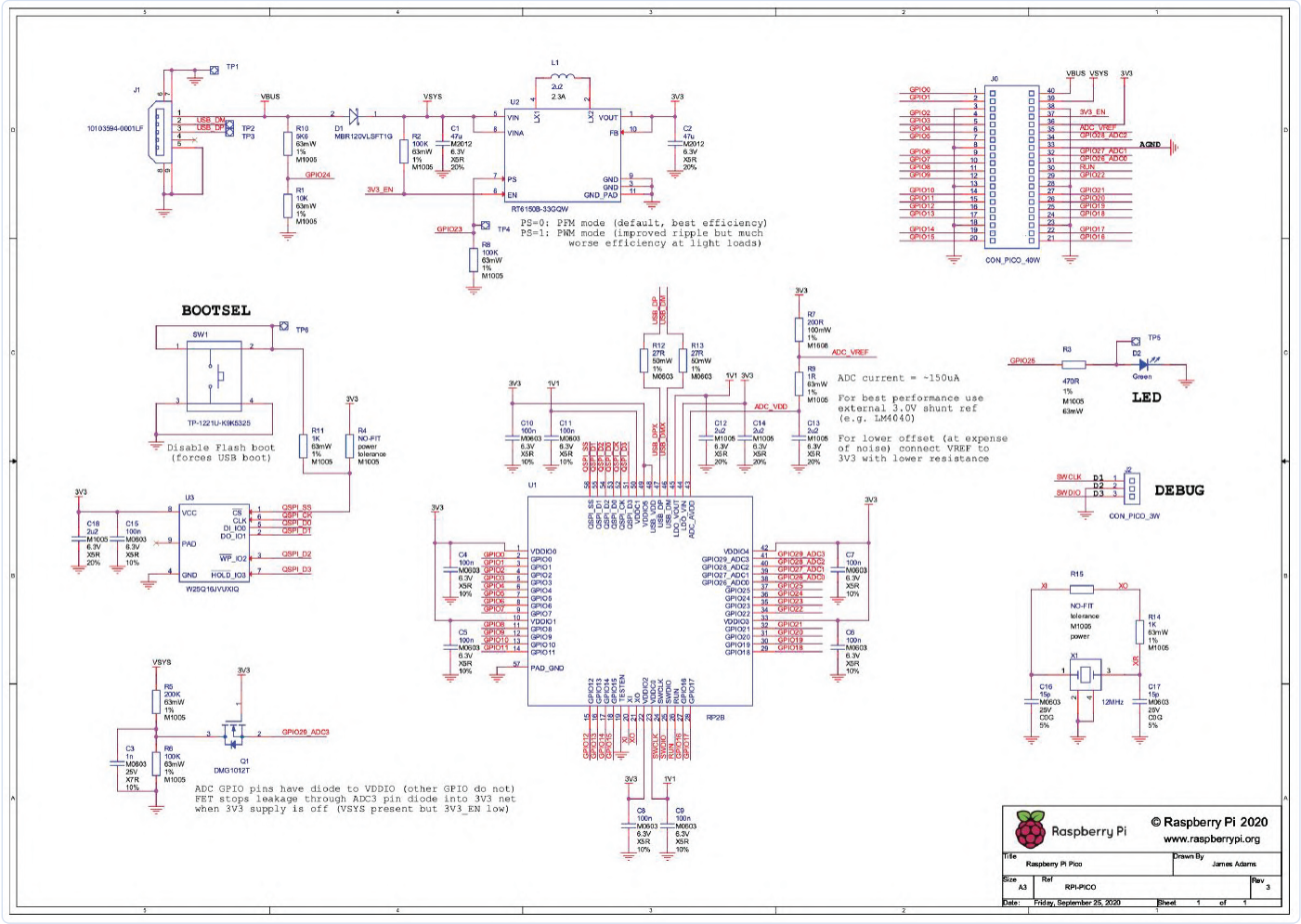


Bild 9. Schaltung des Pico-Boards [3].

Die nativen UARTs des RP2040 basieren auf der ARM Primecell UART (PL011), die auch von den SoCs anderer Raspberry-Pi-Boards verwendet wird. Mit ihnen ist eine maximale Geschwindigkeit von 961,6 kBd möglich. Die beiden SPI-Schnittstellen basieren auf der ARM Primecell SPI (PL022) - ganz analog zu anderen RPis. Sie können als Master Daten mit einem SPI-Takt von bis zu F_CPU/2 übertragen. Der integrierte I²C-Controller unterstützt die Taktraten Standard (100 kHz), Fast (400 kHz) und Fast-plus (1 MHz) mit 7- und 10-Bit-Adressierung als Master oder Slave. Diese drei Schnittstellen ermöglichen den Anschluss der meisten externen Hardware und bieten eine gute Balance zwischen Funktionen und Komplexität. Der ADC des RP2040 ist ein Basis-SAR mit 500 kS/s bei einer Auflösung von 12 bit. Er hat vier Eingänge und einen Temperatursensor. Wenn man die ADCs eines AVRs gewohnt ist, erwartet man eine Bandgap-Referenz, die 2,56 oder 1,1 V als Referenzspannung zur Verfügung stellt. Bei einem Pico muss allerdings bei Bedarf eine externe Referenzspannung an den Pin ADC_AVDD gelegt werden. In der Schaltung des Pico in **Bild 9** kann man sehen, wie dieser Pin gefiltert über das RC-Glied aus R7 + R9 und C13 an 3,3 V angeschlossen ist.

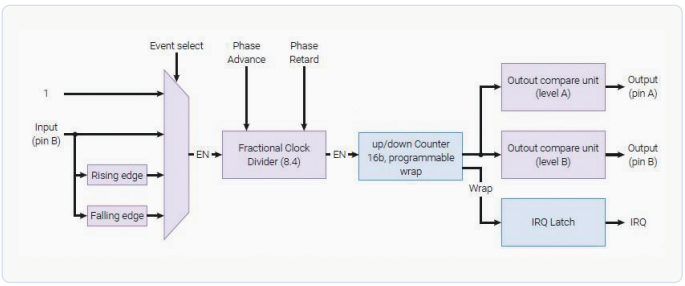


Bild 10. Ein RP2040-PWM-Block [9].

Ein weiteres interessantes Feature ist die USB-Schnittstelle des RP2040. Sie kann entweder als Host oder als Slave konfiguriert werden und bietet eine Übertragungsgeschwindigkeit von bis zu 12 Mbit/s (USB Full Speed). Daher kann man nicht nur USB-Geräte wie Mäuse und Tastaturen nutzen, sondern das Board auch an einen PC oder einen anderen Raspberry Pi anschließen. Als USB-Host unterstützt der Controller die Verwendung von USB-Hubs, so dass Sie nicht auf den Anschluss nur eines Geräts begrenzt ist.

Der RP2040 hat einen 64-bit-Timer mit einer Zeitbasis von 1 µs integriert. Er erlaubt bis zu vier Alarm-Timer und kann zur Erzeugung von Verzögerungen im µs-Bereich verwendet werden. Für sich wiederholende zeitgesteuerte Ereignisse oder Interrupts eignet sich eine der acht 16-bit-PWM-Einheiten mit jeweils zwei Kanälen.

Bild 10 zeigt den Block einer PWM-Einheit. Die GPIO-Pins der PWMs können auch für die Messung von Frequenz oder Tastverhältnis verwendet werden, um damit Interrupts und DMA-Anfragen zu generieren. Sie verfügen über einen „fraktionalen" Takt-Teiler für eine genauere Frequenzeinstellung.
Für weitergehende Programmierungen bietet der RP2040 eine DMA-Einheit, die den Datentransfer im System ohne Belastung der CPU ermöglicht. So ist es z.B. möglich, A/D-Wandlungen durchzuführen und Ergebnisse in einen vordefinierten Puffer im Speicher zu schreiben, ohne dass die CPU eingreifen muss. Man kann auch Daten via DMA aus dem Speicher an einen UART senden. Selbst das Kopieren einer größeren Anzahl von Puffern innerhalb des RP2040 von einem Ort zum einem anderen kann ohne Beteiligung der CPUs erfolgen - sogar schneller als es die CPUs selbst tun könnte. Dies ist praktisch zum Transport von Daten auf ein externes Display. Da die DMA-Einheit auch mit den PIO-Zustandsmaschinen verbunden werden kann, erlaubt dies den hochperformanten Datenaustausch mit externen Geräten.

Schaltpläne, Handbücher und Design-Dateien

Das Pico-Board ist das erste Board mit einem RP2040. Bild 9 zeigt dessen Schaltplan. Die MCU ist umgeben vom DC/DC-Wandler, dem QSPI-NOR-Flash-IC, der USB-Schnittstelle und Elektronik zur Messung der Versorgungsspannung sowie zum Versetzen des DC/DC-Wandlers in den Low-Power-Modus.
Die Raspberry Pi Foundation stellt auch KiCad- und Fritzing-Dateien zur Entwicklung eigener Designs zur Verfügung. Interessanterweise handelt es sich dabei nicht etwa um Pico als Referenzdesign, sondern um eine Demonstration für den Selbstbau eines eigenen RP2040-Boards. Wer Hardware entwickelt, weiß, dass bei den meisten anderen Mikrocontrollern kein besonderes Gewicht auf die Pin-Anordnung gelegt wurde. Beim RP2040 jedoch kann man sich bei der Raspberry Pi Foundation dafür bedanken, dass die Pins in sinnvolle Gruppen zusammengefasst wurden, die leicht an externe Hardware geroutet werden können. Die Pin-Belegung von **Bild 11** beweist, dass intensiv über das Hardware-Design nachgedacht wurde.
Als dieser Artikel geschrieben wurde, waren sowohl die Handbücher als auch die Software noch nicht auf dem endgültigen Stand. In der Zwischenzeit kann es daher noch zu kleineren Änderungen oder Ergänzungen gekommen sein. Nichtsdestotrotz ist die Dokumentation für diese Raspberry Pi-Variante von Anfang an fertig und vollständig, und es handelt sich um „offene" Hardware. Außerdem steht von Anfang an ein offenes SDK zur Verfügung, das die Entwicklung auf einem Raspberry Pi oder einem PC unter Linux und Windows sowie sogar unter MacOS (auf einem Intel-basierten Mac) ermöglicht. Für die Entwicklung eigener Anwendungen gibt es derzeit die Wahl zwischen MicroPython und C/C++.

Entwickeln mit MicroPython

Viele RP2040-Programmierer werden C/C++ bevorzugen (oder bald die Arduino-IDE, denn die Unterstützung für RP2040 ist bereits angekündigt). Allerdings ist MicroPython auch eine erwägenswerte Option. Hierfür ist ein Raspberry Pi 4 mit Raspberry Pi OS oder laut Handbuch eine äquivalente Debian-basierte Linux-Distribution auf einer anderen Plattform erforderlich. Die Befolgung der Anweisungen im Handbuch [5] führt zu einer eigenen Pico-MicroPython-Portierung. Man kann aber auch schlicht eine fertige Binärdatei von der Webseite *Pico getting started* [6] herunterladen.

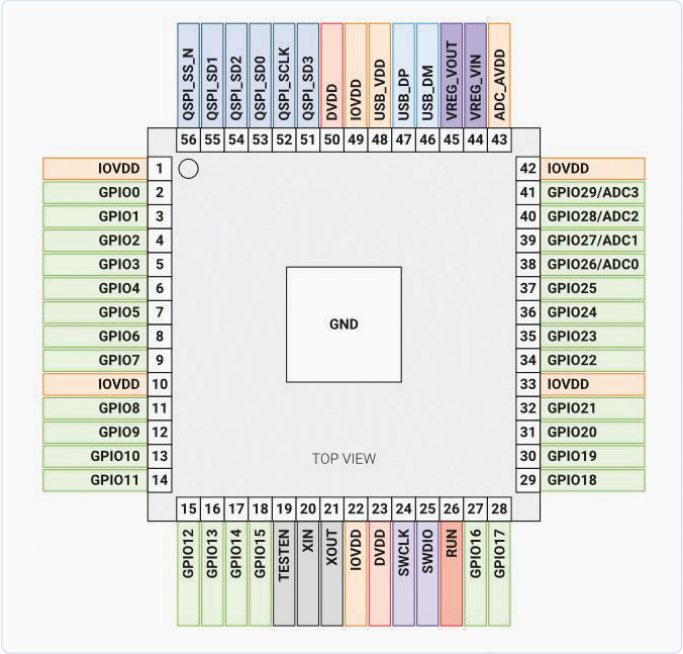


Bild 11. Pin-Belegung des RP2040 [9].

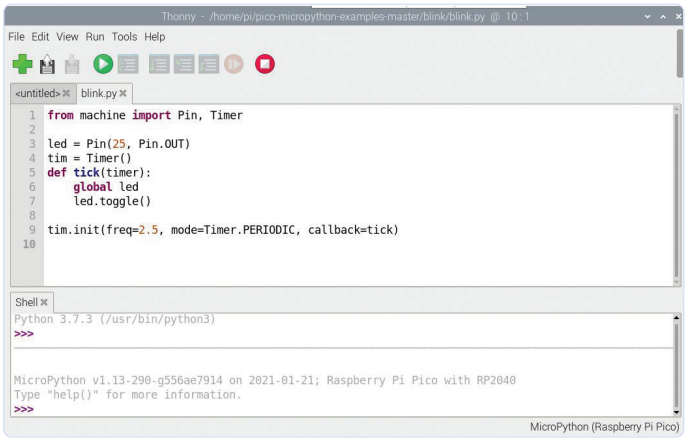


Bild 12. Die Thonny-IDE läuft auf einem RPi 4.

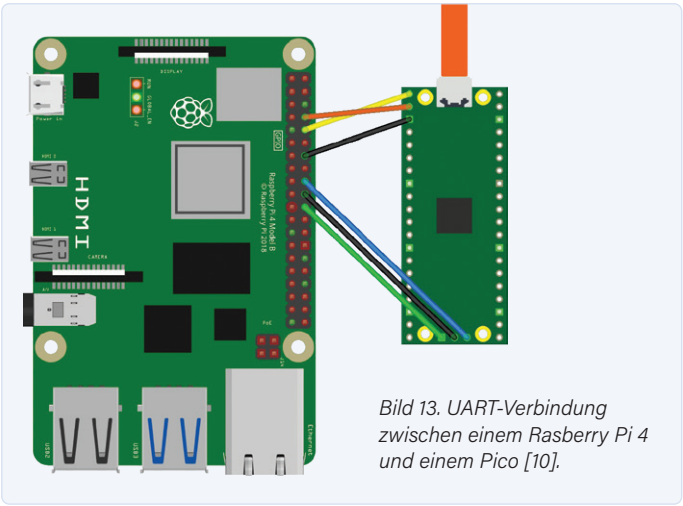


Bild 13. UART-Verbindung zwischen einem Rasberry Pi 4 und einem Pico [10].

HABEN SIE DEN ABSCHNITT ÜBER DEN GERINGEN STROMVERBRAUCH IN DIESEM ARTIKEL VERGESSEN?

Diese Frage löste einige Diskussionen unter den Redakteuren aus, die bereits mit dem Pico zu tun hatten. Normalerweise ist eine der relevanteren Aspekte einer neuen MCU ihr Stromverbrauch. In der Regel zieht eine MCU im Sleep-Modus nur ein paar hundert nA oder ein paar µA. Das Datenblatt des RP2040 spricht aber von einer durchschnittlichen Stromaufnahme von 0,18 mA im Deep-Sleep-Mode. Das klingt vielleicht nicht so schlimm und 180 µA kann man wohl vielfach tolerieren, aber beim Pico ist es damit noch nicht getan: Auf dem Board befinden sich nicht nur der RP2040, sondern auch der externe Flash-Chip und der DC/DC-Converter. Während das Flash-IC im Standby-Modus mit 50 µA oder gar nur 15 µA auskommt, muss der DC/DC-Wandler auch bei sehr geringen Lasten laufen. Daraus ergibt sich laut Pico-Datenblatt ein Standby-Strom von 0,8 mA bei 25 °C. Das ist gerade dann wichtig, wenn das Board über längere Zeit mit Batterien betrieben werden soll. Der Low-Power-Betrieb wurde also nicht vergessen, aber irgendwie musste ein Kompromiss zwischen Funktionen, Strombedarf und Verwendungszweck gemacht werden. Den perfekten Mikrocontroller gibt es nicht, aber die erste MCU der Raspberry Pi Foundation bietet Anlass für Erwartungen an zukünftige Chips. Bezüglich Stromverbrauch wird sich zeigen, ob und wie das Design getweakt werden kann, um noch ein paar µA weniger für eine längere Laufzeit herauszuholen. Das gilt auch für die Software-Entwicklung: Beim Code ist schließlich nicht nur wichtig, dass er schnell ist, sondern dass er auch energieeffizient auf batteriebetriebenen Geräten läuft.

Wenn Sie das Board mit einem Micro-USB-Kabel und gedrückter BOOTSEL-Taste an einen Rasberry Pi 4 anschließen, wird das Pico-Board als USB-Massenspeicher gemountet. Die Binärdatei kann dann schlicht auf das Board kopiert werden, um seinen Flash-Speicher zu programmieren. Nach dieser Installation können Sie sich mit dem Pico im „MicroPython Interactive Interpreter Mode“ (auch REPL genannt) über USB oder den UART des Pico verbinden.

Alternativ funktioniert die MicroPython-Schnittstelle zum Pico und anderen RP2040-basierten Boards mit gängigen integrierten IDEs wie etwa Thonny (**Bild 12**). Das Handbuch enthält eine Anleitung zur Installation und Konfiguration dieser IDE für die MicroPython-Programmierung mit dem Pico.

Interessieren Sie sich für MicroPython? Im Buch „Get Started with MicroPython on Rasberry Pi Pico“ von G. Halfacree und B. Everard (Raspberry Pi Foundation 2021) lernen Sie, wie Sie mit MicroPython Programme schreiben und Hardware anschließen können, um Ihren Pico mit seiner Umwelt interagieren zu lassen. (Siehe „**Passende Produkte**“).

Raspberry Pi-Entwicklung mit C/C++

Die Entwicklung mit C/C++ ist laut Beschreibung und den mitgelieferten Tools auf einem Rasberry Pi 4 vorgesehen. Die Einrichtung einer Entwicklungsumgebung auf anderen Systemen ist im Handbuch [7] beschrieben. Für den Rasberry Pi kann ein praktisches Script heruntergeladen werden [8], das alle für die Installa-

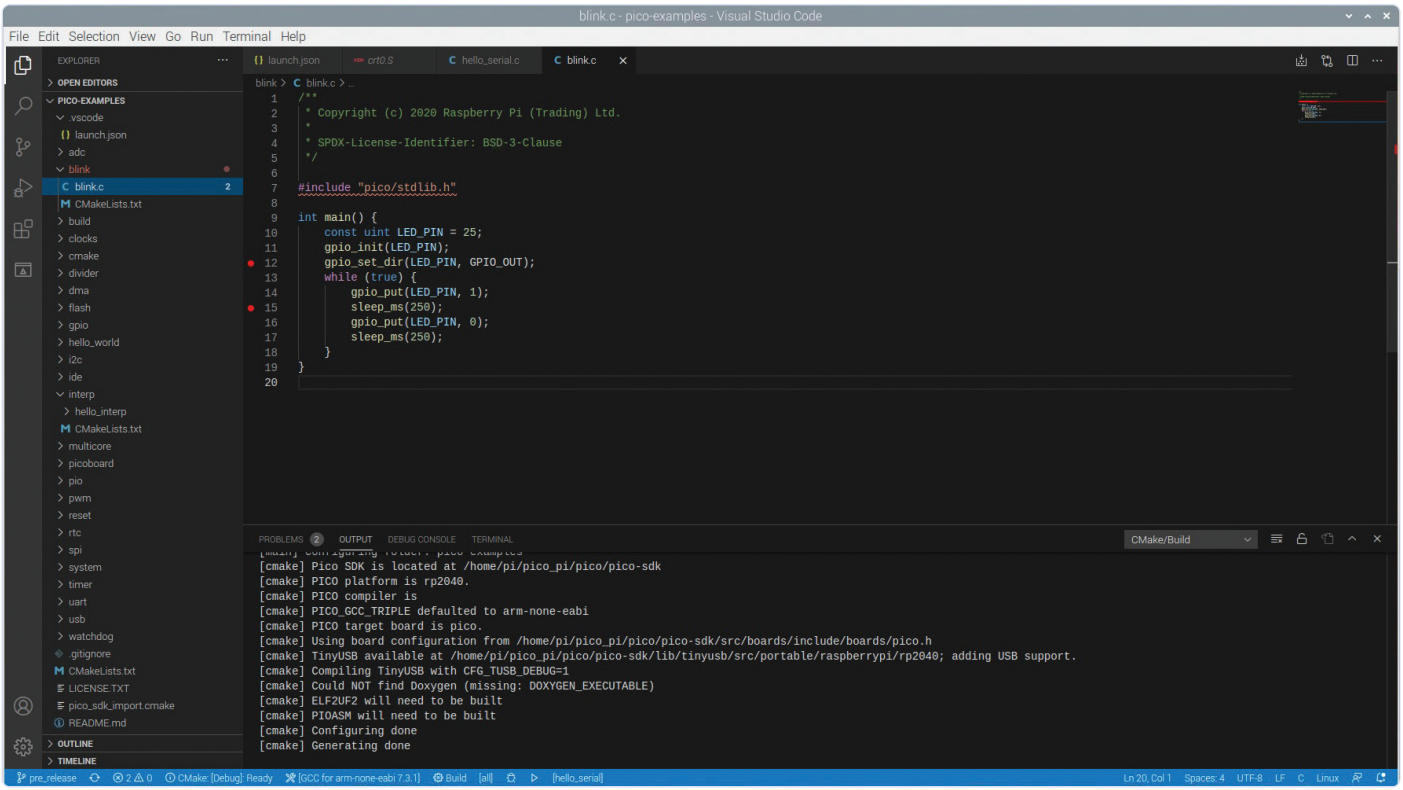


Bild 14. IDE Visual Studio Code auf einem Rasberry Pi 4.

tion erforderlichen Schritte übernimmt. Die Verbindung zwischen dem Rasberry Pi 4 und Pico wird gemäß **Bild 13** vorgenommen. In diesem Setup fungiert der Rasberry Pi 4 auch als Debug-Interface unter Verwendung einer gepatchten Version von OpenOCD. Wer einen PC oder Mac verwenden möchte, kann wie im Handbuch beschrieben einen Debugger mit einem zweiten Pico realisieren. Wem eine IDE mehr zusagt als das Entwickeln in einer Shell, dem bietet das mitgelieferte Script Visual Studio Code (**Bild 14**) eine IDE für den Einstieg. Bei unseren Tests haben die Beispiele ganz gut funktioniert, aber der Projektgenerator zum einfachen Start neuer Projekte (ohne alles Mögliche manuell auf der Kommandozeile erledigen zu müssen) ist noch nicht ganz ausgereift. Auch wenn es zum Start noch ein paar kleine Ärgerlichkeiten gibt: Wie schon beim Raspberry Pi 4 wird es mit der Zeit sicherlich Verbesserungen geben, da mehr Benutzer die mitgelieferte Software nutzen und Fehler melden werden.

Für den RP2040 gibt es einen reichhaltigen Satz an Bibliotheken und Beispielen, so dass Sie nicht für jeden Teil des Chips selbst Treiber schreiben müssen. Für USB wurde die TinyUSB-Bibliothek um die Unterstützung des RP2040 erweitert, was einen schnellen Einstieg ermöglicht. Da der gesamte Code offen ist, können Sie in den Dateien herumwühlen und studieren, wie die Dinge gemacht wurden. Wenn Sie eher an ein zu Arduino Wiring kompatibles Framework gewöhnt sind: Dies war zur Zeit der Arbeit an diesem Beitrag noch nicht fertig. Allerdings ist es wohl wie bei anderen interessanten Nicht-Arduino-Plattformen eine Frage der Zeit, bis es zur Verfügung steht.

Wenn Sie auf einem PC oder Mac entwickeln, haben Sie vermutlich schon Erfahrung mit virtuellen Maschinen gemacht, auf denen die Entwicklungsumgebung eingerichtet wurde. Gibt es da Probleme, kann man auf den Stand eines früheren Snapshots zurückschalten, um innerhalb kurzer Zeit wieder ein gut funktionierendes System vor sich zu haben.

Schlussbemerkungen

Erschienen Ihnen Mikrocontroller bisher als zu teuer oder nicht ausreichend dokumentiert? Jetzt zieht diese „Ausrede“ nicht mehr. Schnappen Sie sich ein paar Picos, solange sie noch verfügbar sind, und fangen Sie an zu programmieren. Bei einem Preis von 5 € pro Stück und mit dem von der Raspberry Pi Foundation zur Verfügung gestellten Support samt Dokumentation ist es keine Frage, **ob** Sie sie kaufen sollten, sondern eher, **wie viele** Pico-Boards Sie kaufen sollen. Durch die verfügbaren Datenblätter, Entwicklungswerkzeuge, Beispiele und die einsteigerfreundlichen Handbücher macht die Arbeit mit einem Pico sehr viel Spaß. Der Pico ist nicht auf erfahrene Entwickler beschränkt, sondern eignet sich auch prima für Studenten und sogar Kinder. Wenn Sie neue Projekte mit dem Board starten, lassen Sie es uns wissen und teilen Sie Ihre Erfahrungen. Wir sind gespannt, was Sie sich einfallen lassen.

210045-02

Sie haben Fragen oder Kommentare?
Haben Sie Fragen oder Anmerkungen zu diesem Artikel? Schicken Sie eine E-Mail an die Autoren unter mathias.claussen@elektor.com oder kontaktieren Sie Elektor unter editor@elektor.com.

Ein Beitrag von
Autoren: **Mathias Claußen** und **Luc Lemens**
Redaktion: **Jens Nickel** und **C. J. Abate**
Übersetzung: **Dr. Thomas Scherer**
Layout: **Giel Dols**

PASSENDE PRODUKTE

- **Raspberry Pi Pico Mikrocontroller-Board**
  www.elektor.de/rpi-pico-board
- **Buch: „Get Started with MicroPython on Raspberry Pi Pico”**
  www.elektor.de/micropython-on-rpi-pico
- **Raspberry Pi 4 B (8 GB RAM)**
  www.elektor.de/rpi4b8
- **Offizielles EU-Netzteil für Raspberry Pi 4 (schwarz)**
  www.elektor.de/eu-power-rpi4-blck
- **Offizielles HDMI-Kabel für Raspberry Pi 4 (schwarz, 1 m)**
  www.elektor.de/hdmi-rpi4-b-1m

WEBLINKS

[1] **C. Valens, „Raspberry Pi Pico Review“, Elektor.TV, 21.01.2021**: https://youtu.be/ijn-QDAgZss
[2] **C. Valens, „Video: RP2040 in Raspberry Pi Pico“, ElektorMagazine.de, 22.01.2021**: www.elektormagazine.de/news/now-playing-rp2040-raspberry-pi-pico
[3] **Das Raspberry Pi Pico Datenblatt**: https://datasheets.raspberrypi.org/pico/pico_datasheet.pdf
[4] **RT6150-Datenblatt**: www.richtek.com/assets/product_file/RT6150A=RT6150B/DS6150AB-04.pdf
[5] **Handbuch zu Pico MicroPython**: https://datasheets.raspberrypi.org/pico/sdk/pico_python_sdk.pdf
[6] **Pico Getting Started**: https://www.raspberrypi.org/documentation/pico/getting-started/
[7] **C/C++ Setup**: https://datasheets.raspberrypi.org/pico/sdk/pico_c_sdk.pdf
[8] **IDE-Installations-Script**: https://github.com/raspberrypi/pico-setup/blob/master/pico_setup.sh
[9] **RP2040-Datenblatt**: https://datasheets.raspberrypi.org/rp2040/rp2040_datasheet.pdf
[10] **Datenblatt „Pico Getting Started“**: https://datasheets.raspberrypi.org/pico/getting_started_with_pico.pdf

MicroPython
für Mikrocontroller

Bildschirme im Kleinformat: Displaytechnik

Von **Dr. Günter Spanner** (Germany)

Dies ist ein Auszug aus dem neuen Elektor-Buch *MicroPython für Microcontroller* von Günter Spanner. Er behandelt die erfolgreiche Verbindung des beliebten Mikrocontroller-Boards ESP32 mit dem OLED-Display SSD1306 in der Programmiersprache MicroPython. Programmieren Sie Prototypen einer EKG-Anzeige und einer Digitaluhr und entdecken Sie dabei die Leistungsfähigkeit und Benutzerfreundlichkeit von MicroPython!

Die Anzeige von Texten und Werten in der Konsole ist für Testzwecke und einfache Anwendungen vollkommen ausreichend. Allerdings hat diese Variante den großen Nachteil, dass immer ein PC oder zumindest ein Laptop dafür erforderlich ist. Bei permanentem Betrieb über Tage oder Wochen hinweg wird durch den Betrieb dieser Geräte unnötige viel Strom verbraucht. Zudem ist es nicht immer erwünscht oder praktikabel, für jede Mikrocontroller-Anwendung auch einen eigenen Rechner einzusetzen.
Geht es nur darum, eine Uhrzeit oder die Messdaten von Klimasensoren anzuzeigen, dann ist es wesentlich besser, ein eigenes kleines Display an den Controller anzuschließen. Ein weit verbreiteter Typ ist das SSD1306. Diese Anzeigeeinheit verfügt bei einer Größe von nur 0,96 Zoll (ca. 2,5 cm) über eine Auflösung von 128 x 64 Pixel.
MicroPython bringt bereits standardmäßig einen Treiber für diese Displayversion mit. Dieser wird bereits beim Upload des Dateisystems auf den ESP32 mit geladen. Der Treiber ermöglicht sowohl die Darstellung von Texten und numerischen Daten als auch die Anzeige von einfachen Grafiken. Das SSD1306-Display ist mit internem RAM und einem eigenen Oszillator ausgestattet. Dadurch kann es ohne weitere externe Komponenten betrieben werden. Darüber hinaus verfügt es über eine Helligkeitsregelung mit 256 einstellbaren Stufen.
Die wichtigsten Eigenschaften des SSD1306-Displays sind:

- Auflösung: 128 x 64 Punktmatrix
- Stromversorgung: 1,65 V bis 3,3 V
- Betriebstemperaturbereich: - 40 °C bis + 85 °C
- Integrierter 128 x 64-Bit-SRAM-Anzeigepuffer
- Kontinuierliche Bildlauffunktion in horizontaler und vertikaler Richtung
- On-Chip-Oszillator

OLED-Anzeigen benötigen keine Hintergrundbeleuchtung, da jedes

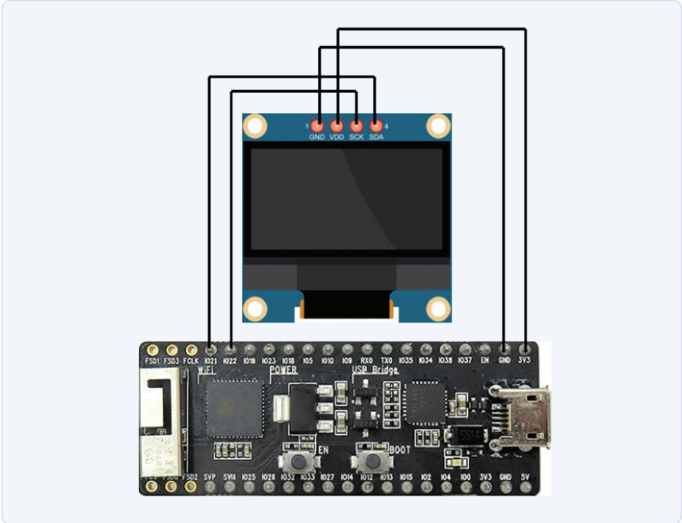


Bild 1. SD1306-Display am ESP32.

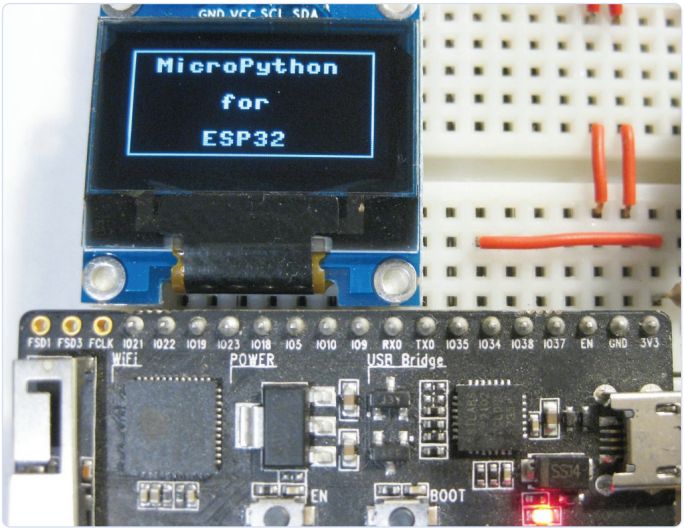


Bild 2. SSD1306-Display mit Text- und Grafikausgabe.

einzelne Pixel in der Lage ist, Licht auszusenden. Aus diesem Grund ist diese Variante auch bei ungünstigen Lichtverhältnissen noch gut ablesbar. Zudem ist der Kontrast im Vergleich zu Flüssigkristallanzeigen (LCDs) deutlich höher. Da ein Pixel nur dann Energie verbraucht, wenn es tatsächlich leuchtet, sind OLED-Displays sehr energieeffizient.
Die einfachsten Versionen von SSD1306-Boards verfügen über lediglich vier Pins. Dies ist ausreichend, um das Display über den I²C-Bus anzusteuern. Andere Varianten weisen Reset-Pins oder eine zusätzliche SPI-Schnittstelle auf. Für die meisten Anwendungen ist die einfache Bauform jedoch ausreichend. Die folgende Tabelle zeigt alle erforderlichen Verbindungen.

OLED Pin	ESP32
VDD	3V3
GND	GND
SCK	GPIO 22
SDA	GPIO 21

Der Schaltplan dazu ist in **Bild 1** dargestellt. Das folgende Skript gibt eine Textnachricht und ein einfaches Grafikelement in Form eines Rahmens auf das Display aus:

```
# SSD1306_DEMO.py

from machine import Pin, I2C
from ssd1306 import SSD1306_I2C

i2c=I2C(-1,scl=Pin(22),sda=Pin(21))
# I2C Pin assignment
oled_width=128
oled_height=64
oled = SSD1306_I2C(oled_width, oled_height, i2c)
lin_hight = 9
col_width = 8
def text_write(text,lin, col):
oled.text(text,col*col_width,lin*lin_hight)

oled.fill(0)
text_write("MicroPython", 1, 2)
text_write("for", 3, 6)
text_write("ESP32", 5, 5)

oled.rect(5, 5, 116, 52, 1)
oled.show()
```

und liefert das Ergebnis in **Bild 2**. Für die Ansteuerung des Displays müssen die erforderlichen Module importiert werden. Wie bereits erwähnt, sind die Libraries „machine“ und „SSD1306“ im Normalfall bereits als Standardbibliotheken vorhanden. Falls erforderlich, kann jedoch auch eine ssd1306.py-Datei separat auf das Board hochgeladen werden. Die Pin-Deklaration für den I²C-Bus erfolgt über:

```
i2c = I2C (-1, scl = Pin (22), sda = Pin (21))
```

Die Anzahl der Pixel des angeschlossenen Moduls wird mit folgenden Variablen

```
oled_width = 128
oled_height = 64
```

erfasst. Der Parameter „-1“ gibt an, dass das verwendete Modul weder über Reset- noch über Interrupt-Pins verfügt. Mit diesen Angaben kann ein `SSD1306_I2C`-Object mit dem Namen `oled` erstellt werden.
Hier werden die zuvor definierten Daten übernommen:

```
oled = ssd1306.SSD1306_I2C (oled_width, oled_height, i2c)
```

Damit ist das Display betriebsbereit. Über die Funktion `text()` werden Informationen auf die Anzeige ausgegeben. Mit der Methode `show()` wird die Anzeige aktualisiert. Die `text()`-Funktion akzeptiert die folgenden Argumente:

- Nachricht (Typ String)
- X-Position und Y-Position des Textfeldes in Pixeleinheiten
- Optionale Textfarbe: 0 = schwarz (dunkel) und 1 = weiß (hell)

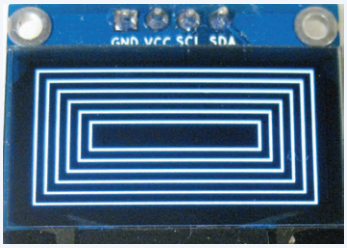


Bild 3. Konzentrische Rahmen auf dem OLED-Display.

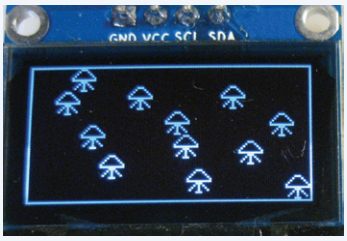


Bild 4. Grafikelemente auf dem OLED-Display.

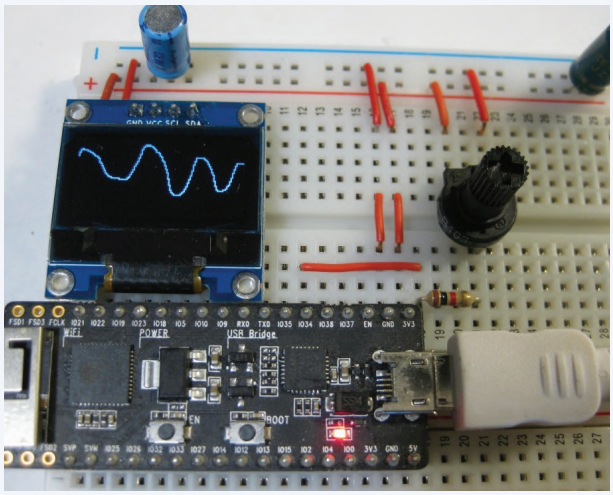


Bild 5. Datenausgabe auf dem OLED-Display.

Listing 1. SSD1306-Bitmap-Demo

```
# SSD1306_bitmap_DEMO.py

from machine import Pin, I2C
import ssd1306
import urandom

i2c = I2C(-1, scl=Pin(22), sda=Pin(21))
oled_width = 128
oled_height = 64
oled = ssd1306.SSD1306_I2C(oled_width, oled_height,
   i2c)

# frame
oled.hline(0, 0, oled_width-1, 1)
oled.hline(0, oled_height-1, oled_width-1, 1)
oled.vline(0, 0, oled_height-1, 1)
oled.vline(oled_width-1, 0, oled_height, 1)
oled.show()
```

```
ICON = [
    [0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0],
    [0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0],
    [0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0],
    [0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
    [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
    [0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0],
    [0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0],
    [0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0],
    [0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0],
    [0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0],
]
for n in range(12):
  xofs = urandom.randint(1, oled_width-12)
  yofs = urandom.randint(1, oled_height-12)
  for y, row in enumerate(ICON):
    for x, c in enumerate(row):
      oled.pixel(x+xofs, y+yofs, c)
oled.show()
```

Listing 2. Rollende EKG-Anzeige

```
# Rolling_ECG_display.py
from machine import Pin, ADC, I2C from time import
   sleep
import ssd1306

i2c = I2C(-1, scl=Pin(22), sda=Pin(21))
  oled_width = 128
  oled_height = 64
  oled = ssd1306.SSD1306_I2C(oled_width, oled_
   height, i2c)

  pot = ADC(Pin(34))
  pot.atten(ADC.ATTN_11DB) #Full range: 3.3v vMax =
   3.4
  dotPos_old = int(oled_height/2)
while True:
  pot_value = pot.read()
  voltage = 0.000816*pot_value + 0.037822 #
   print(voltage)
dotPos_new = int(voltage/vMax*oled_height)

oled.line(0, dotPos_new, 0, dotPos_old, 1)
oled.scroll(1, 0)
oled.line(0, dotPos_new, 0, dotPos_old, 0)

dotPos_old = dotPos_new oled.pixel(0, int(oled_
   height/2), 1)
oled.show()
```

Die folgende Anweisung gibt eine Meldung in weißer bzw. blauer Farbe auf dunklen Hintergrund aus. Der Text beginnt an der Position x = 0 und y = 0:

```
oled.text ('MicroPython!', 0, 0)
```

Die Methode `show()` macht die Änderungen auf dem Display sichtbar. Die Bibliothek enthält noch weitere nützliche Methoden. Die Funktion `fill(1)` erzeugt einen komplett weißen Bildschirm, d. h. alle Pixel sind beleuchtet. Mit `oled.fill(0)` werden alle Pixel auf schwarz oder dunkel gesetzt.

Die Methode `pixel()` ermöglicht grafische Darstellungen. Sie akzeptiert die folgenden Argumente:

- X-Koordinate: Pixelposition horizontal
- Y-Koordinate: Pixelposition vertikal
- Pixelfarbe: 0 = schwarz, 1 = weiß

Ein einzelnes weißes Pixel in der oberen linken Ecke kann damit so erzeugt werden:

```
oled.pixel (0, 0, 1)
```

Mit

```
oled.invert (True)
```

werden die OLED-Farben invertiert werden. Dabei wird weiß zu schwarz und umgekehrt.

Um zu den ursprünglichen Farben zurückzukehren, kann `oled.invert (False)` verwendet werden.

Grafische Darstellungen

Neben den Pixel-Anweisungen sind auch noch weitere Grafik-Befehle verfügbar. Horizontale und vertikale Linien können mit `.hline ()` bzw. `.vline ()` gezeichnet werden. Dabei werden die XY-Startposition sowie die Linienlänge und -farbe angegeben.
Das folgende Programm zeichnet beispielsweise konzentrische rechteckige Rahmen auf das Display (**Bild 3**):

```
# SSD1306_frames.py
from machine import Pin, I2C
import ssd1306

i2c = I2C(-1,scl=Pin(22),sda=Pin(21))
oled_width = 128
oled_height = 64
oled = ssd1306.SSD1306_I2C(oled_width, oled_height, i2c)

for n in [0,5,10,15,20,25]:
oled.hline(n, n, oled_width-1-2*n, 1-2*n)
oled.hline(n, oled_height-1-n, oled_width-1-2*n, 1-2*n)
oled.vline(n, n, oled_height-1-2*n, 1-2*n)
oled.vline(oled_width-1-n, n, oled_height-2*n, 1-2*n)
oled.show()
```

Diagonalen werden über `.line(x1,y1, x2, y2, c)` zwischen zwei festgelegten Punkten (x1, y1) und (x2, y2) gezogen. Der Parameter c steuert die Farbe der gezeichneten Linie. Für einfache Grafiken können Bitmaps Pixel für Pixel in den Displaypuffer geschrieben werden. Das Programm in **Listing 1** erzeugt eine Anzeige wie in **Bild 4.**

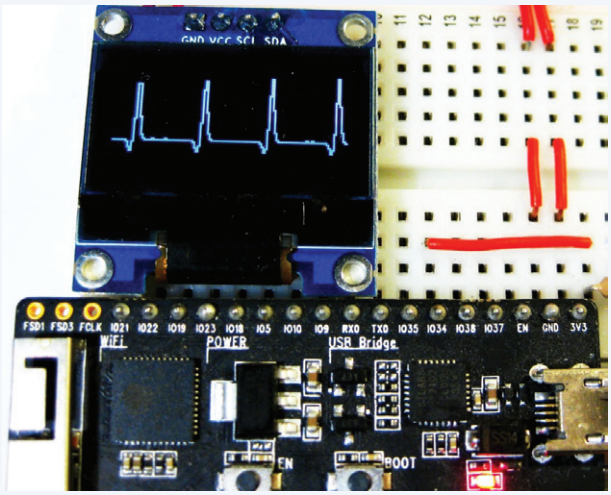


Bild 6. Ein EKG-Signal auf dem OLED-Display.

OLED-Display als Datenplotter

Neben Bitmaps lassen sich auf dem OLED-Display auch Messdaten graphisch darstellen. Damit ist man nicht mehr auf die Plotter-Funktion in Thonny angewiesen, sondern kann auch Stand-alone-Geräte mit Grafikausgabe aufbauen.
Das Programm in **Listing 2** liefert ein rollendes Display, wie es etwa von professionellen EKG-Geräten in Krankenhäusern her bekannt ist.
Für die Messwert-Aufnahme kann wieder ein Potentiometer an den ADC-Eingang 34 angeschlossen werden. Nach dem Start des Programms werden die Spannungswerte kontinuierlich auf dem Display dargestellt. Ermöglicht wird dies durch die integrierte Scroll-Funktion. Über die Anweisung

```
oled.scroll(1, 0)
```

wird der gesamte Bildschirminhalt um ein Pixel verschoben.

Wenn man nicht einzelne Pixel aufzeichnen will, sondern eine durchlaufende Kurve, dann muss man die Punkte über Linien verbinden. Hierzu sind zwei Variablen erforderlich:

```
dotPos_old and dotPos_new
```

Damit wird eine Linie zwischen dem aktuellen und dem letzten Messwert gezogen. Dann wird das Display um ein Pixel verschoben. Schließlich wird die neue Position an den Zwischenspeicher übergeben:

```
dotPos_old = dotPos_new
```

und das Spiel beginnt von vorne. **Bild 5** zeigt ein Beispiel für kontinuierlich veränderte Potentiometer-Werte.

Wenn man über einen EKG-Verstärker verfügt, kann man auf diese Weise auch die elektrischen Signale des menschlichen Herzens aufzeichnen. Es entsteht das typische Elektrokardiogramm, wie es aus den Intensivstationen der Krankenhäuser bekannt ist (**Bild 6**).

Digitaluhr mit OLED-Display

Eine weitere nützliche Anwendung für das Display ist die Ausgabe der Uhrzeit. Damit verwandelt sich der ESP32 zusammen mit dem

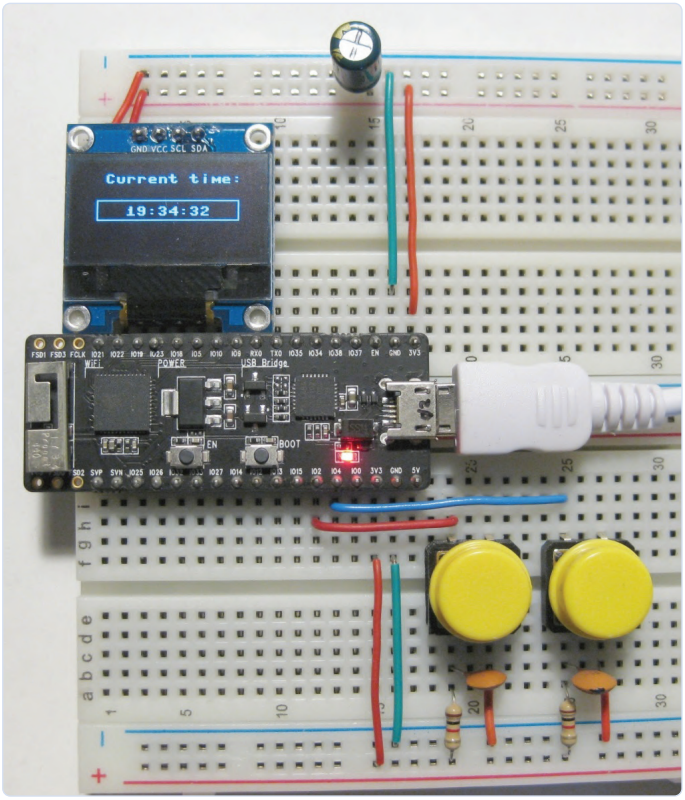


Bild 7. Digitaluhr mit OLED-Display.

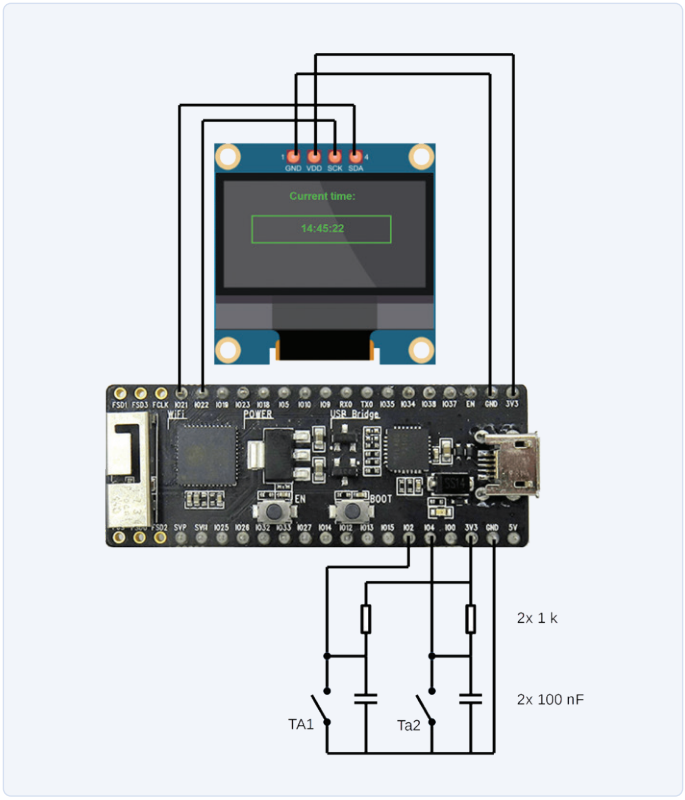


Bild 8. Schaltplan der in MicroPython programmierten Digitaluhr.

Listing 3. ESP32/SSD1306-Uhr in MicroPython

```
# setable_clock.py

from machine import Pin,I2C
import time
from ssd1306 import SSD1306_I2C

i2c = I2C(-1,scl=Pin(22),sda=Pin(21))
oled_width=128
oled_height=64
oled = SSD1306_I2C(oled_width,oled_height,i2c)

time_offset=20*3600+00*60+0 # hh+mm+ss
lin_hight=5
col_width=8

def handle_interrupt_min(pin):
  global time_offset
  time_offset+=60
  time.sleep(.2)

def handle_interrupt_hr(pin):
  global time_offset
  time_offset+=3600
  time.sleep(.2)

button_min = Pin(4, Pin.IN)
button_min.irq(trigger=Pin.IRQ_RISING,
   handler=handle_interrupt_min)

button_hr = Pin(2, Pin.IN)
button_hr.irq(trigger=Pin.IRQ_RISING,
   handler=handle_interrupt_hr)

def text_write(text, lin, col=0):
  oled.text(text, col*col_width, lin*lin_hight)

def time_text(time):
  secs=time%60
  mins=(time//60)%60
  hours=(time//3600)%24
  return "{:02d}:{:02d}:{:02d}".
   format(hours,mins,secs)

def show():
  oled.fill(0)
  text_write("Current time:",1,2)
  current_text = time_text(current_time)
  text_write(current_text,6,4)
  oled.rect(10,25,108,16,1)
  oled.show()

while True:
  current_time=time_offset+time.time()
  show()
```

SSD1307 in eine praxistaugliche Digitaluhr. Die Funktion `time()` aus dem time-Modul gibt die Anzahl der Sekunden seit dem Einschalten oder einem Reset des Boards zurück. Über eine Variable

```
time_offset=20*3600+00*60+0 # hh+mm+ss
```

kann eine Startzeit vorgegeben werden.
In diesem Fall startet die Uhr mit der Zeit 20:00:00. Die Routine `time_text()`:

```
def time_text(time):
secs=time%60
mins=(time//60)%60
hours=(time//3600)%24
return "{:02d}:{:02d}:{:02d}".format(hours,mins,secs)
```

wandelt die fortlaufenden Sekunden in Stunden, Minuten und Sekunden um, also in die übliche hh:mm:ss-Darstellung (**Bild 7**). Die Uhr kann über zwei Taster (Ta1 und Ta2) an den Ports 02 und 04 eingestellt werden, wie in **Bild 8** gezeigt. Die Taster sind wieder mit 100-nF-Kondensatoren zu Entprellung versehen. Die 1-kΩ -Widerstände dienen als Pull-Ups.

Die zugehörigen Interrupt-Routinen

```
def handle_interrupt_min(pin):
global time_offset
time_offset+=60
time.sleep(.2)
```

und

```
def handle_interrupt_hr(pin):
global time_offset
time_offset+=3600
time.sleep(.2)
```

sorgen dafür, dass sich bei jedem Tastendruck die Minuten (60 Sekunden) bzw. Stundenzahl (3600 Sekunden) um eins erhöht. Das komplette Programm zur Digitaluhr finden Sie in **Listing 3**.

200581-02

Haben Sie Fragen oder Kommentare?
Haben Sie Fragen oder Kommentare zu diesem Artikel? Kontaktieren Sie Elektor unter editor@elektor.com!

Ein Beitrag von:
Text und Abbildungen: **Dr. Günter Spanner**
Lektorat: **Rolf Gerstendorf**
Redaktion: **Jan Buiting**
Layout: **Giel Dols**

WEBLINK

[1] **Software-Archiv für MicroPython für Mikrocontroller:** www.elektormagazine.de/200581-02

PASSENDE PRODUKTE

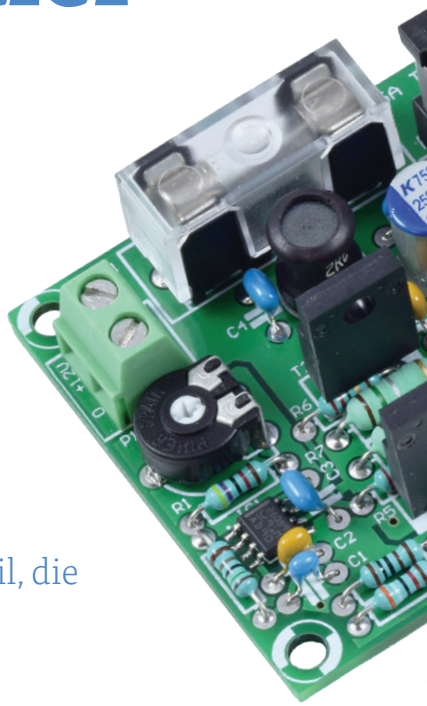
MicroPython für Mikrocontroller
Projekte mit uPyCraft-IDE und ESP32

Die Programmiersprache „Python“ hat in den letzten Jahren einen enormen Aufschwung erlebt. Nicht zuletzt haben verschiedene Einplatinensysteme wie der Raspberry Pi zu deren Bekanntheitsgrad beigetragen. Aber auch in anderen Gebieten, wie der Künstlichen Intelligenz oder dem Machine Learning, hat Python weite Verbreitung gefunden. Es ist daher naheliegend, Python bzw. die Variante „MicroPython“ auch für den Einsatz in SoCs (Systems on Chip) zu verwenden.
Leistungsfähige Controller wie der ESP32 der Firma Espressif Systems bieten eine hervorragende Performance sowie Wi-Fi- und Bluetooth-Funktionalität zu einem günstigen Preis. Mit diesen Eigenschaften wurde die Maker-Szene im Sturm erobert. Im Vergleich zu anderen Controllern weist der ESP32 einen deutlich größeren Flash und SRAM-Speicher, sowie eine wesentlich höhere CPU-Geschwindigkeit auf. Aufgrund dieser Leistungsmerkmale eignet sich der Chip nicht nur für klassische C-Anwendungen, sondern insbesondere auch für die Programmierung mit MicroPython.
Das vorliegende Buch führt in die Anwendung der modernen Ein-Chip-Systeme ein. Neben den technischen Hintergründen steht vor allem MicroPython selbst im Vordergrund. Nach der Einführung in die Sprache werden die erlernten Programmierkenntnisse umgehend in die Praxis umgesetzt. Die einzelnen Projekte sind sowohl für den Einsatz im Labor als auch für Alltagsanwendungen geeignet. Neben dem eigentlichen Lerneffekt steht also auch die Freude am Aufbau kompletter und nützlicher Geräte im Vordergrund. Durch die Verwendung von Laborsteckboards können Schaltungen aller Art mit geringem Aufwand realisiert werden, sodass das Austesten der selbstgebauten Geräte zum lehrreichen Vergnügen wird.

Das Buch **MicroPython für Mikrocontroller** ist unter www.elektor.de/micropython-fur-mikrocontroller in Papierform und als E-Book erhältlich. Das zum Buch gehörende Software-Paket finden Sie kostenlos auch auf der Supportseite des Artikels [1].

Gleichspannungswandler 12 V auf 200 V

Sichere Hochspannung für Röhrenverstärker



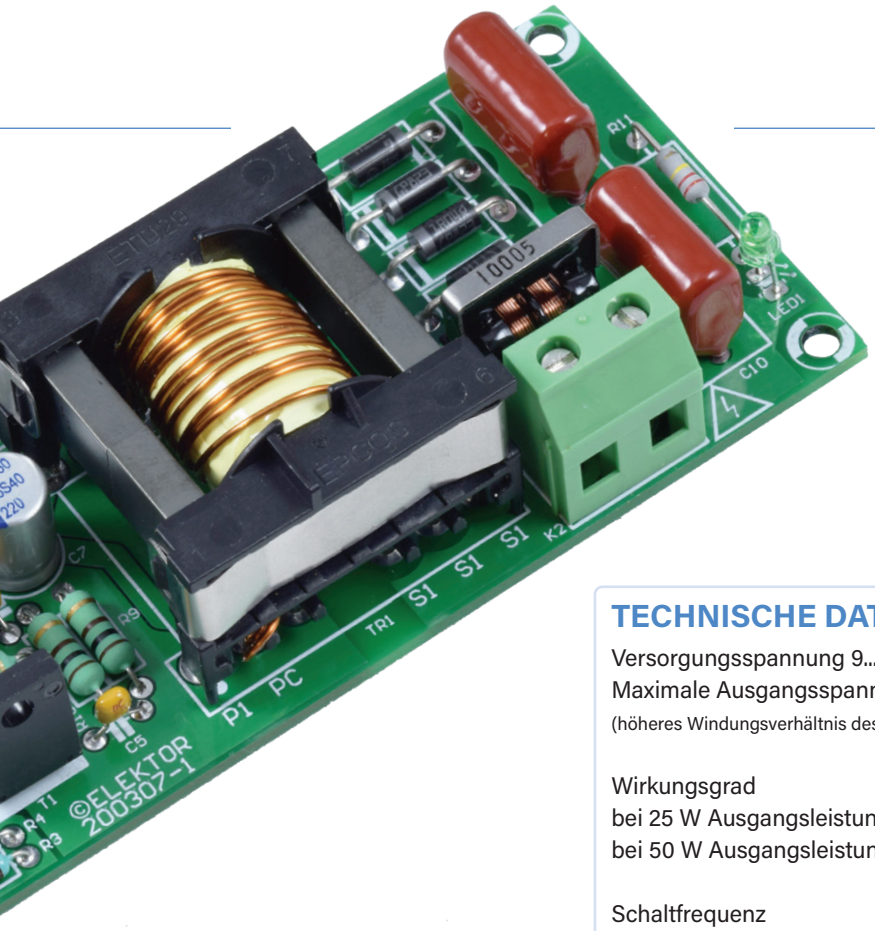
Von **Ton Giesberts** (Elektor)

Hier kommt ein (relativ) einfach zu bauender DC/DC-Wandler für Röhrenverstärker in Durchstecktechnik mit nur einem SO-8-IC mit galvanisch getrenntem Hochspannungsausgang. Der Transformator ist Marke Eigenbau und sein Windungsverhältnis bestimmt die Ausgangsspannung. Die Energie am Eingang liefert ein 12-V/5-A-Netzteil, die maximale Ausgangsleistung beträgt 50 W.

Die Hochspannungsversorgung eines Röhrenverstärkers ist ein wichtiger, aber auch gefährlicher Teil der Schaltung. Am Eingang haben wir die potentiell tödliche Netzspannung und am Ausgang lauert eine hohe Gleichspannung, die auch ziemlich „schockierend" sein kann.
Die einfachste Stromversorgung, die wir gefahrlos an der Netzspannung betreiben können, besteht aus einem gut isolierten Transformator, einem Gleichrichter und einem Glättungskondensator. Seine Ausgangsspannung hängt von der Netzspannung, dem Wicklungsverhältnis des Transformator, dem Spannungsabfall über der/den Diode(n), der Belastung und der resultierenden Brummspannung ab. Die Hochspannung in Röhrenverstärkern wird oft auf diese Weise erzeugt, doch durch das feste Windungsverhältnis handelsüblicher Transformatoren ist die Hochspannung oft entweder zu niedrig oder zu hoch (obwohl dies bei vielen Verstärkern nicht wirklich ein Problem darstellt).
In diesem Projekt verwenden wir ein 12-V-Netzteil und einen speziell angefertigten Gleichspannungswandler, um die niedrigere Gleichspannung auf 200 V anzuheben. So vermeidet man eine direkte Verbindung zur Netzspannung und hat zusätzlich eine niedrigere Gleichpannung zur Verfügung, die für die Röhrenheizungen verwendet werden kann.
Hier präsentieren wir einen Auszug aus den viel umfangreicheren theoretischen und praktischen Informationen, die der Autor auf der Elektor-Labs-Seite für dieses Projekt zusammengestellt hat. Wenn Sie nur an einigen grundlegenden Hintergründen und dem Bau dieses Gleichspannungswandlers interessiert sind, bietet dieser Artikel alles Material, das Sie dazu brauchen. Wenn Sie aber mehr über die konstruktiven Überlegungen beim Bau eines Gegentakt-DC/DC-Wandlers und die damit verbundenen (Trafo-) Berechnungen wissen wollen, folgen Sie bitte dem Link unter [1].

Wie der Gleichspannungswandler funktioniert

Das Schaltbild des DC/DC-Wandlers ist in **Bild 1** dargestellt. Um den Aufbau der Schaltung zu vereinfachen, haben wir mit einer Ausnahme, einem ICs in SO-8-SMD-Gehäuse, ausschließlich Durchsteck-Bauteile verwendet. Bei besagtem IC1, einem UCC28089, handelt es sich um einen primärseitigen Gegentaktoszillator mit Totzeitsteuerung. Die Ausgangstreiber können 1 A aufnehmen und 0,5 A liefern und sind ideal zur Ansteuerung von MOSFETs. Das Fehlen einer Regelschleife bedeutet, dass es keine Instabilität oder Rauschen durch die PWM-Steuerung gibt. Das Tastverhältnis ist maximal, oder anders ausgedrückt: Die Totzeit zwischen dem Einschalten eines der MOSFETs (T1 oder T2) ist auf ein absolutes Minimum eingestellt (siehe unten). Dadurch wird die gleichgerichtete sekundäre Ausgangs- beinahe eine perfekte Gleichspannung, so dass nur ein kleiner Glättungskondensator benötigt wird.
Der UCC28089 verfügt über eine Überstromerkennung, aber die Abschaltschwelle liegt typisch bei 0,725 V über einem Shunt, der den Strom durch die MOSFETs misst. Bei mehreren Ampere Strom ist die Verlustleistung im Shunt zu hoch und würde den Gesamtwirkungsgrad erheblich beeinflussen. Bei 5 A, wo wir den Grenzwert für diese Schaltung ansetzen, würde dies 5 A x 0,725 V = 3,625 W betragen! Der Shunt-Widerstand müsste groß und für eine Verlustleistung von mindestens 5 W dimensioniert sein. Bei 12 V und 50 W Ausgangsleistung und würde dies bedeuten, den Wirkungsgrad von 86 % um 7 % zu reduzieren. Eine Möglichkeit, diesen Verlust im Shunt zu verkleinern, ist ein Spannungsteiler (R3 und R4) in der Versorgungsspannung, um dem Strommesseingang einen kleinen Offset hinzuzufügen.
Die Oszillatorfrequenz des UCC28089 kann mit dem Trimmpoti P1 von etwa 200 kHz bis 560 kHz eingestellt werden. Die Schaltfrequenz beträgt die Hälfte der Oszillatorfrequenz und ist damit von 100 kHz bis 280 kHz einstellbar. Laut Datenblatt beträgt der ON-Widerstand der TK30A06N1 MOSFETs maximal 15 mΩ bei V_{GS}=10 V und ist damit nahezu vernachlässigbar. Dieser MOSFET glänzt mit einer sehr geringen Sperrschicht-



INFOS ZUM PROJEKT

Tags
Stromversorgung, DC/DC-Wandler, Röhrenverstärker, Hochspannung, Durchsteckmontage, handgefertigter Transformator

Level
Einsteiger – **Fortgeschrittene** – Experte

Zeit
circa 8 Stunden

Werkzeuge
Lötwerkzeug, kleine Spitze für SO-8 SMD-Löten

Kosten
circa 45 €

TECHNISCHE DATEN

Versorgungsspannung 9...14 V_{DC}
Maximale Ausgangsspannung 350 V
(höheres Windungsverhältnis des Transformators)

Wirkungsgrad
bei 25 W Ausgangsleistung: 88%
bei 50 W Ausgangsleistung: 86%

Schaltfrequenz
100/150/280 kHz (P1 min/mittel/max)

Leerlaufstrom 12 VDC
300/190/120 mA (P1 min/mittel/max)

Eine sehr hohe Last verhindert das Anlaufen des Wandlers. Dieser Wandler wurde für Röhrenverstärker entwickelt, bei denen die Belastung der Hochspannung beim Einschalten sehr gering ist!

kapazität von 33 pF und schnellen Schaltzeiten (t_{on}=21 ns, t_{off}=28 ns). Die Eingangskapazität ist mit 1050 pF geringer als bei den meisten anderen MOSFETs mit einem solch niedrigen ON-Widerstand. Bei maximaler Leistung ist keine zusätzliche Kühlung der Transistoren erforderlich, obwohl ein Dauerbetrieb mit maximaler Leistung nicht empfohlen wird.

In einem Gegentaktwandler wird die Energie direkt auf die Sekundärseite übertragen. Wenn ein Schalter leitend ist, wird der andere ausgeschaltet und die Energie wird direkt übertragen. Beim Umschalten ist eine kurze Totzeit notwendig, um einen kurzzeitigen Kurzschluss der Primärwicklung zu verhindern. Wären nämlich beide Schalter gleichzeitig eingeschaltet, würden sich die Magnetfelder beider Primärwicklungen gegenseitig aufheben, was zu einem extrem hohen Strom führt, der hauptsächlich durch den Serienwiderstand aus Primärwicklungen, MOSFETs, Shunt, Leiterbahnen und der 12-V-Gleichstromversorgung begrenzt wird. Die Shunt-Widerstände würden wahrscheinlich durchbrennen und bei anhaltendem Kurzschluss auch die MOSFETs (selbst bei geringer Zeitüberlappung!). Zum Schutz der 12-V-Versorgung wird eine 5-A-Sicherung verwendet. Ein Vorteil eines Gegentaktwandlers ist, dass der Spitzenstrom durch die Schalter nur geringfügig höher ist als der durchschnittliche Laststrom auf der Primärseite, also der Strom, den das Netzteil liefern muss. Geringere Spitzenströme in den Schaltern bedeuten weniger Leitungsverluste und führen zu einem höheren Wirkungsgrad des Wandlers.

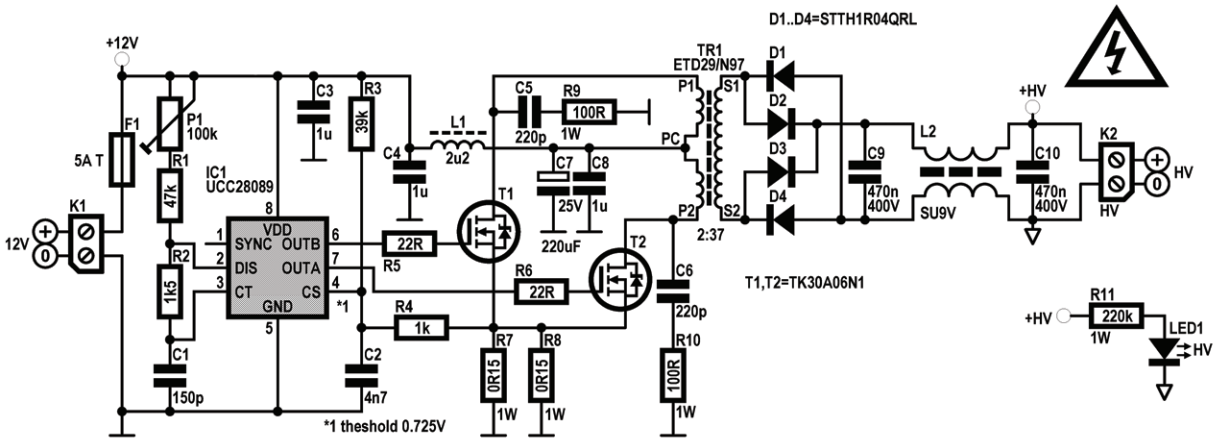


Bild 1. Schaltung des Gleichspannungswandlers.

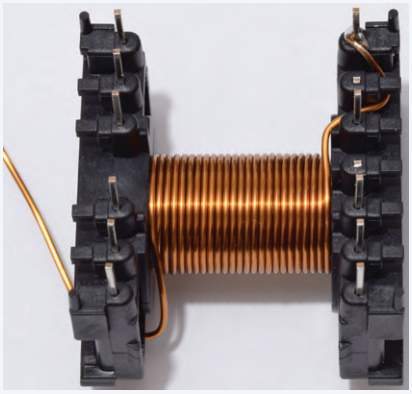


Bild 2a. Erste Lage der Sekundärwicklung.

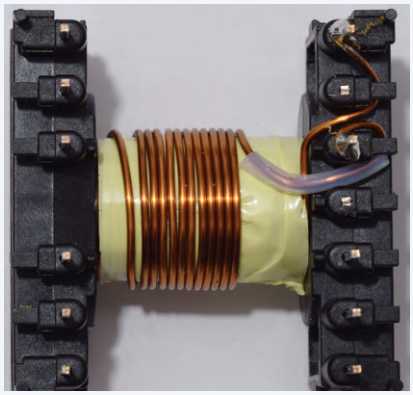


Bild 2b. Zweite Lage der Sekundärwicklung.

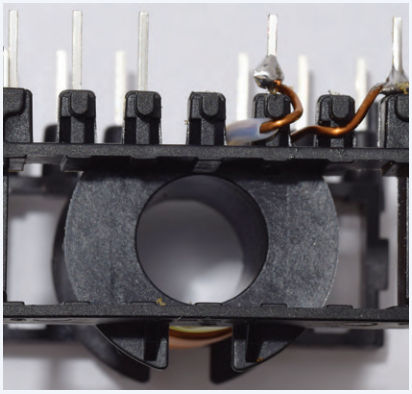


Bild 2c. Die festgelötete Sekundärwicklung.

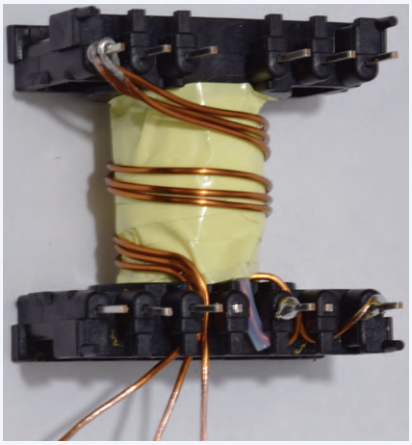


Bild 2d. Erste Lage der Primärwicklung. Beachten Sie die drei parallel geschalteten Drähte!

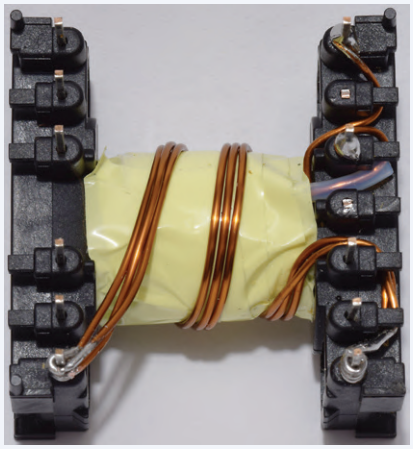


Bild 2e. Erste Primärwicklung, vor dem Löten.

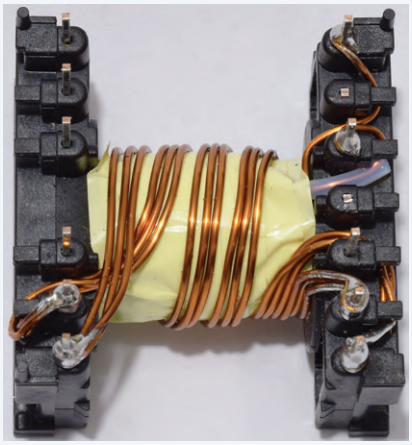


Bild 2f. Ist die zweite Lage der Primärwicklung hinzugefügt, ist der Transformator fertig.

Bild 2. Aufbau des Transformators.

Der Transformator

Der kniffligste Teil dieses Gleichspannungswandlers ist der Übertrager, der in Handarbeit gefertigt werden muss. Bei 150 kHz kann die Ausgangsleistung theoretisch mehr als 170 W betragen, wenn ein ETD29-Spulenkörper [2] und N97-Material verwendet wird, aber in der Praxis spielen auch Kupferverluste eine Rolle. Im Übertrager wird kein Luftspalt verwendet, was die Anzahl der Windungen für Primär- und Sekundärwicklung reduziert und den Aufbau des Übertragers noch weniger aufwändig macht. Kein Luftspalt bedeutet höhere Induktivität und bessere Kopplung zwischen den Wicklungen. Sie können für alle Wicklungen den gleichen Draht verwenden, am besten 0,7 mm²-CuL-Draht, den Sie noch bequem um den Spulenkörper wickeln und dabei die Anzahl der Windungen im Auge behalten können.

Die Idee der Schaltung ist, dass das Windungsverhältnis des Transformators die Ausgangsspannung bestimmt, im Falle eines Gegentaktwandlers das Verhältnis von einer Primärwicklung zur Sekundärwicklung. In diesem Projekt werden zwei Primärwicklungen mit je zwei Windungen und eine einzige Sekundärwicklung mit 37 Windungen verwendet. Bei einer Primärspannung von 12 V würde ein Windungsverhältnis von 2:37 theoretisch zu einer Sekundärspannung von 222 V führen. Da kein Transformator ideal ist, wird dieses Verhältnis bei einer bestimmten Last aber ohnehin nicht genau die gewünschte Ausgangsspannung erzeugen. Im Vergleich zur hohen Ausgangsspannung ist der Spannungsabfall über dem Gleichrichter und

KLEINE TRAFOTHEORIE

Die in einer Spule induzierte Spannung ist nach dem Faradayschen Gesetz (vereinfacht):

$E = N * B * A / t$

mit:
E: Spannung [V}
N: Anzahl der Windungen
B: Flußdichte [T]
A: die Fläche der Spule [m²]
$t = T/2 = 1 / 2f$

In einem Gegentaktwandler sind die an den Primärwicklungen anliegenden Spannungen nahezu Rechteckwellen. In der Hälfte der Periode kann der Kern pro Wicklung von B_{max} auf $-B_{max}$ magnetisiert werden und umgekehrt. Die andere Wicklung macht das Gegenteil, für die Zeit der Magnetisierungsänderung von $2 * B_{max}$ gilt für die induzierte Spannung:
$E = N * 2B_{max} * A * 2f$
Für eine der Primärwicklungen erhalten wir dann die folgende Gleichung
$N_P = E_P * 10^4 / (4 * f * B_{max} * A)$
mit der Fläche A in cm² (daher die 10^4).

ÜBER KERNGRÖSSEN UND DIE MAXIMALE AUSGANGSLEISTUNG

Dieses Thema scheint der schwierigste Teil bei der Berechnung eines Gleichspannungswandlers mit einem Transformator zu sein. Wenn man im Internet sucht, findet man eine Menge Informationen zur Berechnung von Transformatoren, sogar komplette Rechner, die den Entwurf sehr einfach machen. Die meisten Webseiten verzichten aber auf die Berechnung der Kerngröße und verweisen auf so ziemlich dieselbe Art von Tabellen für alle Arten von verschiedenen Kernen und Spulenkörpern wie Exx, EExx, EFxx, EFDxx, EIxx, ETDxx und EERxx.

Diese Tabellen sind meist nach der maximal erreichbaren Leistung sortiert, aber die Schaltfrequenz fehlt oft. Um einen Anhaltspunkt zu haben, wie groß der Kern sein muss, kann eine Formel mit dem Kernflächenprodukt W_aA_c (Multiplikation von Kernquerschnittsfläche und der für Wicklungen zur Verfügung stehende Fläche, dem Wicklungsfenster) verwendet werden. Allerdings müssen Faktoren wie Topologiekonstante (abhängig vom Typ des Wandlers), Stromdichte (abhängig von der maximal zulässigen Erwärmung) und Fensterausnutzungsfaktor in die Formel(n) eingesetzt werden. Eine exakte Berechnung ist das aber nicht, es gibt zu viele Abhängigkeiten.

Vor vielen Jahren gab es von der Firma Block Transformator-Bausätze. Ein Bausatz enthielt eine kleine Rolle Isolierfolie, Isolierhülsen, zwei kleine Platinenstücke für einen definierten Luftspalt, Spulenkörper, zwei Halbkerne und zwei Klemmen. Der Kupferdraht musste anderweitig beschafft werden.

Im Bausatz befand sich auch ein Faltblatt namens *Berechnungsbogen für Schaltnetzteil-Übertrager EB* mit Formeln zur Berechnung der minimalen Kerngröße, der Primär- und Sekundärwindungen, des Drahtdurchmessers für Primär und Sekundärwicklung(en). Das damalige Kernmaterial war N27. Der größte Unterschied zum heutigen N97 ist, dass die relativen Kernverluste bei N97 erheblich geringer sind, 300 kW/m³ statt 920 kW/m³ (bei 100 kHz/200 mT/100 °C). Die Formel zur Berechnung der Mindestkerngröße lautet dann

Minimale Kerngröße [mm] = 4,7 * 10^6 * Maximale Gesamtausgangsleistung / Frequenz [W/Hz]

Mit dieser Formel zur Berechnung der maximalen Leistung bei 150 kHz und einer Kerngröße von 5350 mm³ ergeben sich 170 W. Das meinte ich, als ich sagte, der ETD29-Kern wäre größer als unbedingt nötig.

Wenn jemand weiß, wie man diese Formel oder eine ähnliche (bessere) ableitet und/oder einen Einblick gibt, wie man die Kerngröße genauer berechnen kann, bitte teilen Sie Ihr Wissen.

dem Filter vernachlässigbar. Wenn die Abweichung der Ausgangsspannung größer als erwartet ist, kann man am Transformator mit außen liegender Sekundärwicklung verhältnismäßig leicht Änderungen vornehmen. Die innen liegenden Primärwicklungen werden mit Spezialband und Isolierschläuchen isoliert. In diesem Verfahren wurde der erste Prototyp-Transformator hergestellt, für einen zweiten Prototyp erwies es sich jedoch als praktischer, die Sekundärwicklung auf der Innen- und den beiden Primärwicklungen auf der Außenseite anzubringen [1], so dass hier nur der zweite Transformator beschrieben wird.

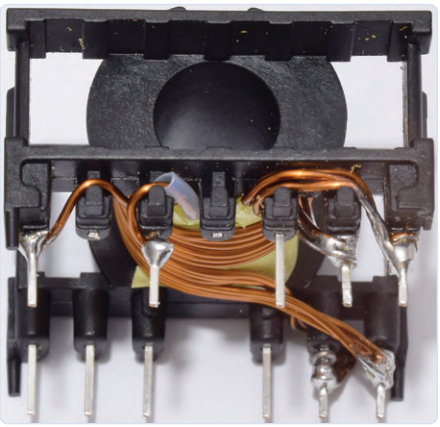


Bild 3. Blick auf die Anschlüsse des fertigen Transformators.

Aufbau des Transformators

Um den Übertrager leichter reproduzierbar zu machen, wurde als Spulenkörper die kleinste ETD-Version gewählt, die einen größeren als für die Ausgangsleistung von etwa 60 W (bei einem theoretischen Wirkungsgrad von 100 %) benötigten Kern besitzt.

Zwei Pins des Spulenkörpers werden abgeschnitten (Pin 8 und Pin 10 gemäß der Nummerierung auf dem Spulenkörper von TDK/Epcos), um die korrekte Ausrichtung des Übertragers auf der Platine zu ermöglichen. Sehen Sie sich das Platinenlayout vorher genau an, damit Sie auch wirklich die richtigen Pins abschneiden. Die Lötaugen für Primär- und Sekundärwicklungen sind auf der

Anzeige

LPN liefert Leiterplatten aus Deutschland, vom Weltmarkt, aus NATO-Partnerländern oder mit anderen Restriktionen.

LPN ist nach ISO 9001:2015 Zertifiziert und das Personal beim FraunhoferInstitut geschult.

LPN liefert jedes Basismaterial und jede in Deutschland oder am Weltmarkt verfügbare Technik.
- Multilayer bis 56 Lagen.
- Starrflex, Flex, Semiflex.
- Aluminium, auch Bergquist, Kupferkern, Messingkern, Stahlkern.
- Teflon, auch Rogers.
- Montagehilfen Kaptonband, Abziehlack und Weiteres.

LPN Qualitätsprüfungen
- 100% Kontrolle
- Kupferstärkenmessung mit Magnetfeld Messgeräten.
- Nachmessen gedruckter Induktivitäten.
- Schliffbildauswertung.
- Lot-Benetzungs-Test.
- Delaminations-Test.
- alle Fertigungsstätten halten ISO 14001 ein.

LPN Dienstleistungen
- Datenaufbereitung incl. Nutzenaufbau,
- Machbarkeitsprüfung,
- EMPB.
- geklebte Vorlagen digitalisieren.
- Filme digitalisieren.
- Leiterplatten clonen.
- Leiterplatten nachlayouten.
- Terminaufträge.
- Abruflager für Jahreslose.

Profitieren Sie von den LPN Qualitätsstandards und den weltweiten Kontakten.

LPN Leiterplatten Nord GmbH
Hermann-Bössow-Straße 13-15
23843 Bad Oldesloe
leiterplatten-nord.de

Anfragen/Bestellungen:
lpn@lp-nord.de
Telefon 04531 1708 0

STÜCKLISTE

Widerstände:

R1 = 47k
R2 = 1k5
R3 = 39 k
R4 = 1k
R5,R6 = 22 Ω
R7,R8 = 0Ω15, 1 W
(Multicomp Pro MCKNP01SJ015KA10)
R9,R10 = 100 Ω, 1 W
(Multicomp Pro MCKNP01SJ0101A10)
R11 = 220 k, 1 W, 350 V
P1 = 100 k Trimmpoti, von oben einstellbar

Kondensatoren:

C1 = 150 p, 50 V, Raster 5 mm
C2 = 4n7 n, 50 V, Raster 5 mm
C3,C4,C8 = 1 µ, 50 V, Raster 5 mm
C5,C6 = 220 p, 100 V, Raster 5 mm
C7 = 220 µ, 25 V, 20 %, Polymer-Aluminium, Raster 3,5 mm, ∅ 8 mm, ESR 15 mΩ (A750KS227M1EAAE015 Kemet)
C9,C10 = 470 n, 400 V, 5 %, Polypropylen, Raster 15 mm (19x9 mm² max.)

Induktivitäten/Transformatoren:

TR1 = 2 x Kern ETD29, N97, ohne Luftspalt, B66358G0000X197 TDK/Epcos
TR1 = Spulenkörper, ETD29, B66359W1013T001, TDK/Epcos
TR1 = 2 x Schelle, B66359S2000X000, TDK/Epcos
L1 = 2,2 µH, 10%, 5,6 ARMS, 21 mΩ, Abstand 5 mm, ∅ 8,5 mm max, RLB0913-2R2K Bourns
L2 = Gleichtaktdrossel 500 µH, 1 A, 0,3 Ω, SU9V-10005 Kemet

Halbleiter:

D1,D2,D3,D4 = STTH1R04QRL
LED = 3mm LED grün
T1,T2 = TK30A06N1, TO-220SIS
IC1 = UCC28089D, SOIC-8

Außerdem:

K1 = 2-Wege-Klemme, Raster 5,08 mm
K2 = 2-Wege-Klemme, Raster 7,68 mm
F1 = Patronensicherung, 5x20 mm, 5 A träge mit Sicherungshalter für Patronensicherung 5x20mm, 500V/10A und Abdeckung
TR1 = Elektrisches Isolierband, Polyesterfolie, 3M 1350 12 MM
TR1 = Kupferlackdraht, 0,71 mm für alle Wicklungen, insgesamt 49 Windungen, 3 Meter
TR1 = PTFE-Schutzschläuche, Innendurchm. 1,02 mm min. , Pro Power STFE 18 CLR, 10 cm
PCB 200307-1 v1.1

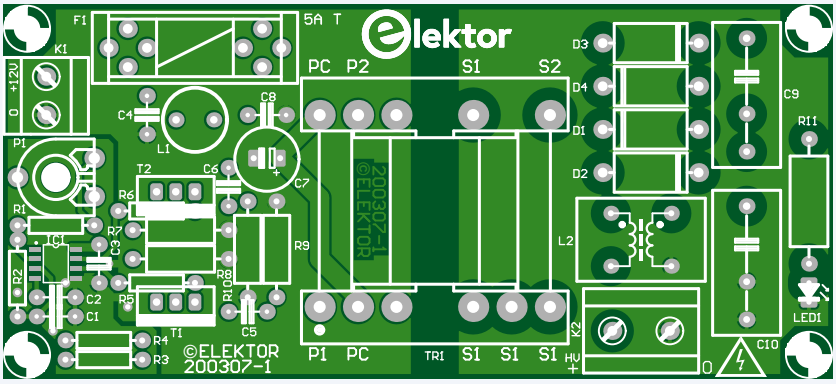


Bild 4. Das Platinenlayout.

Platine gut voneinander getrennt, ebenso wie die Ausgänge der Sekundärspannung. Ursprünglich besitzt ein Spulenkörper vom Typ ETD29 13 Pins mit einer Lücke zwischen Pin 3 und Pin 4. Ein Ende der Sekundärwicklung kann an zwei möglichen Anschlüssen gelötet werden. Je nach Windungszahl und Dicke des verwendeten Kupferdrahtes wird nämlich mehr als eine Lage benötigt und das Ende der Wicklung liegt entweder näher an der Gegenseite oder an der gleichen Seite wie der Anfang. Aus diesem Grund sind die Pins 4...6 und Pin 9 auf der Platine verbunden. Pin 7 ist der andere Anschluss der Sekundärwicklung. Beginnen Sie die Sekundärwicklung immer an Pin 7.

Pin 1 und Pin 13 sind die Anschlüsse für die erste Primärwicklung, Pin 2 und Pin 12 für die die zweite. Auf der Platine finden Sie die Bezeichnungen neben dem Transformator, P1/PC für die erste und P2/PC für die zweite Primärwicklung. PC steht für den gemeinsamen Anschluss der beiden Primärwicklungen (common). S1 und S2 sind die Anschlüsse für die Sekundärwicklung. Sie beginnen an Pin 7 mit der Sekundärwicklung, drehen 24 Windungen um den Spulenkörper (**Bild 2a**) und isolieren die Wicklung mit geeignetem Klebeband. Anschließend wird die zweite Lage mit 13 Windungen aufgebracht, so dass die Sekundärwicklung aus insgesamt 37 Windungen besteht (**Bild 2b**). Der zweite Anschluss wurde mit einem Stück PTFE-Schlauch isoliert, wie in **Bild 2c** zu sehen. Es wäre besser gewesen, auch den ersten Anschluss der Sekundärwicklung zu isolieren (fehlt auf den Fotos von TR2). Dann wird die zweite Sekundärlage mit Klebeband isoliert. **Erhitzen Sie die Drähte zu den Stiften des Spulenkörpers nicht zu lange, denn dadurch würde der Kunststoff des Spulenkörpers schmelzen. Deshalb sollte man die Drahtenden ordentlich verzinnen, schon bevor sie um die Stifte gewickelt werden!**

Als nächstes beginnt die erste Windung der ersten Primärwicklung an Pin 1. Entfernen Sie mit einem scharfen Messer die Lackisolierung jedes Drahtes und verlöten Sie dieses Ende, bevor Sie es um einen Pin wickeln. Verwenden Sie eine Zange, um das Drahtende fest um den Stift zu quetschen, bevor Sie zwei Windungen um den Spulenkörper legen. Bringen Sie den Draht an, machen Sie ihn (etwas mehr als) lang genug, um den gegenüberliegenden Stift zu erreichen, aber schließen Sie ihn noch nicht an. Tun Sie dies für die nächsten zwei Windungen dieser Primärwicklung (drei Drähte sind parallel geschaltet, siehe **Bild 2d**). Entfernen Sie dann die Isolierung der drei Enden. Löten Sie die Enden zuerst an und quetschen Sie alle drei um den Stift, so dass die Wicklung optimal platziert ist (möglichst flach und gleichmäßig verteilt, **Bild 2e**). Wiederholen Sie dies für die zweite

WEBLINKS

[1] **Elektor-Labs-Seite zum Gleichspannungswandler**: www.elektormagazine.de/ labs/12v-200v-dc-dc-converter-for-valve-amplifiers
[2] **Datenblatt TDK-Spulenkörper**: https://product.tdk.com/info/en/documents/data_sheet/80/db/fer/etd_29_16_10.pdf
[3] **Downloads zu diesem Projekt**: www.elektormagazine.de/200583-02

Primärwicklung, beginnend an Pin 2, siehe **Bild 2f**. **Bild 3** zeigt den fertigen Transformator mit allen Wicklungen. Bevor Sie aber die fertige Konstruktion im Schwung der Begeisterung auf die Platine löten, vergessen Sie nicht, den Kern anzubringen!

Bau des DC/DC-Wandlers

Die Gerber- und Bohrdateien zur Herstellung der Platine (**Bild 4**) stehen Ihnen unter [3] zur Verfügung. Bestellen Sie die Platine mit diesen Dateien bei Ihrem bevorzugten Lieferanten. Die Platine, die auf den Fotos des Prototyps zu sehen ist, ist die Version 1.0, in der Version 1.1 wurde eine kleine Korrektur vorgenommen, indem der Text „+HV" und „0" neben die Schraubklemme K2 gesetzt wurde. Zuerst wird IC1 auf die Platine gelötet, das kleine SO-8-SMD. Für die 1-W-Leistungswiderstände (R7...R10) auf der Primärseite haben wir aus Platzgründen kleine Ausführungen (Gehäuseabmessungen 10 mm x 3,5 mm verwendet. Achten Sie bei der Bestellung der Bauteile auf die in der Stückliste genannte Herstellerreferenz! Außerdem sollte C7 ein Polymer-Aluminium-Typ sein, verwenden Sie keinen gewöhnlichen Elko, denn dieser würde höchstwahrscheinlich durchbrennen! Der gelistete Typ (A750KS227M1EAAE015 von Kemet) hat einen sehr geringen Serienwiderstand von 15 mΩ (100 kHz/20 °C) und kann deshalb einen Ripplestrom von 4,42 A bei 100 kHz verkraften. Die Montage der restlichen Bauteile ist ziemlich einfach. Der Vorwiderstand R11 der LED, die die Hochspannung (HV) anzeigt, muss mindestens ein 350-V-Typ und kann ein größerer 1-W-Widerstand sein. Beachten Sie, dass die Schaltung nicht den Anforderungen der Schutzklasse II entspricht!

Der Wandler und auch die Kupferfläche am Ausgang weist einen erhöhten mechanischen und elektrischen Abstand auf und auch der Transformator sorgt bei richtigem Aufbau für eine hervorragend galvanisch getrennte hohe Ausgangsspannung. Stellen Sie das Trimmpoti P1 in die Mittelposition, um eine Schaltfrequenz von 150 kHz zu erreichen.

Achten Sie beim Einsatz dieses DC/DC-Wandlers darauf, dass sich in der Nähe von Bauteilen, die an die Hochspannung angeschlossen sind, keine elektrisch leitenden Gegenstände befinden. Je größer der Abstand, desto besser!

Die Ausgangsspannung kann durch Änderung der Windungszahl der Sekundär-, aber auch durch Verändern der Windungszahl der Primärwicklungen verändert werden, wenn die richtige Schaltfrequenz beachtet wird. Das Ändern von C1 (und R2) für eine andere Oszillatorfrequenz führt auch zu einer Änderung der Totzeit! Bevor Sie die Werte ändern, sollten Sie sich die Formeln im Datenblatt des UCC28089 genau ansehen.
Der Entwurf von Gegentakt-Gleichspannungswandlern und vielleicht noch mehr die Berechnung und Konstruktion des benötigten „maßgeschneiderten" Transformators ist ziemlich kompliziert. Wie bereits erwähnt, finden Sie weitere Hintergrundinformationen zu diesem Thema auf [1], wo Sie auch Fragen, Kommentare, Anregungen und Erfahrungen mit diesen Wandlern und dem Trafobau mit anderen Lesern teilen können.

200583-02

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

Haben Sie Fragen oder Kommentare?
Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Schreiben Sie dem Autor eine E-Mail an ton.giesberts@elektor.com.

PASSENDE PRODUKTE

- **Menno van der Veen, Trans Röhrenverstärker (Elektor-E-Book)**
www.elektor.de/trans-roehren-pdf
- **Menno van der Veen, Der Entwurf von Röhrenverstärkern (Elektor-E-Book)**
www.elektor.de/der-entwurf-von-rohrenverstaerkern-pdf

Anzeige

SIGLENT®
Das Beste Preis-Leistungs-Verhältnis in der Messtechnik
Schneller zur Marktreife dank innovativer Messtechnik

- Oszilloskope 50MHz –> 1GHz
- Signal Generatoren 10MHz –> 500MHz
- Labornetzteile –> 200W
- Elektronische Lasten 200 –> 300W
- Multimeter 4½ –> 6½ Stellen

Siglent HF-Messgeräte: Helfer bei der Entwicklung fehlerfreier und zuverlässiger Kommunikation

Spektrum- und Vektornetzwerkanalysator
- EMI-Pre-Compliance
- Modulationsanalyse
- Distance-to-Fault Analyse

HF-Signalquellen
- mit/ohne IQ-Modulation

Siglent Technologies Germany GmbH
www.siglenteu.com
Info-eu@siglent.com

Der Hitze auf der Spur

Thermo-Kamera Seek Shot Pro

Von **Alfred Rosenkränzer**

Die Thermo-Kamera Seek Shot Pro ist klein. Zu klein? Und taugt sie etwas? Wir haben die Kamera für Sie anhand von einigen Praxisbeispielen ausprobiert.

Bild 1. Vorderansicht mit IR- und Normalobjektiv sowie Beleuchtungs LED.

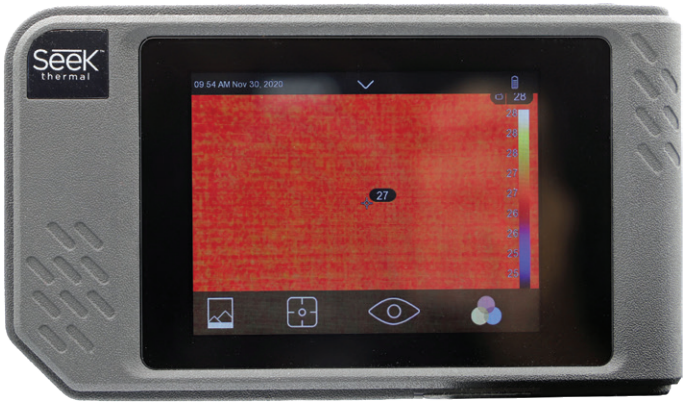


Bild 2. Rückansicht mit aktivem Display.

Die Thermo-Kamera wird in einem stabilen Karton mit den Maßen 173 x 115 x 44 mm³ geliefert. Er enthält neben der Kamera ein USB-Kabel, eine Trageschlaufe und eine gedruckte Anleitung in mehreren Sprachen. Die deutsche Version ist gut übersetzt. Die Kamera selbst ist etwa 140 x 80 x 25 mm³ klein und hat mehr Ähnlichkeit mit einem Smartphone als mit einer Kleinbildkamera (**Bild 1**). Das Gehäuse ist gummiert, liegt gut in der Hand und bietet etwas mechanischen Schutz. Es gibt lediglich zwei mechanische Bedienelemente: einen Ein/Ausschalter und den Auslöser für Fotos und Videos an der Oberseite. Alle anderen Einstellungen werden über den Touchscreen vorgenommen.

Drin & dran

Auf der Unterseite ist der USB-Anschluss unter einer Abdeckung verborgen. Über ihn werden der fest verbaute Akku der Kamera geladen und die Bilder zum PC übertragen. Der interne Speicher ist nicht erweiterbar, denn es gibt keinen SD-Karten-Slot. Der größte Unterschied zu einem Smartphone ist das ¼-Zoll-Gewinde zur Montage auf ein Stativ. Auf der Vorderseite befindet sich rechts oben das größere Objektiv der Thermo-Kamera. Direkt darunter sitzt die eher unscheinbare Normalbild-Kamera plus eine „Blitz-LED" links neben dem Thermo-Objektiv zur besseren Ausleuchtung der Normalbilder bei wenig Umgebungslicht, die man auch manuell als Taschenlampe schalten kann.
Es gibt zwei Versionen, die sich durch die Auflösung der Thermokamera unterscheiden. Die normale Ausführung bietet 208 x 156 Pixel und die Pro-Version immerhin 320 x 240 Pixel. Wem das bescheiden vorkommt: Thermokameras haben durchweg geringe Auflösungen, wenn man sie mit normalen Kameras vergleicht. Zur Erstellung von Thermobildern ist die gebotene Auflösung aber durchaus genug.
Die Pro-Version bietet einen Bildwinkel 57° - dies entspricht etwa 40 mm Brennweite bezogen auf das Kleinbild-Format. Sie sieht also ein kleines bisschen weitwinkliger als das menschliche Auge. Die normale Variante geht mit einem Bildwinkel von 36° (≈ 66 mm bezogen auf KB)

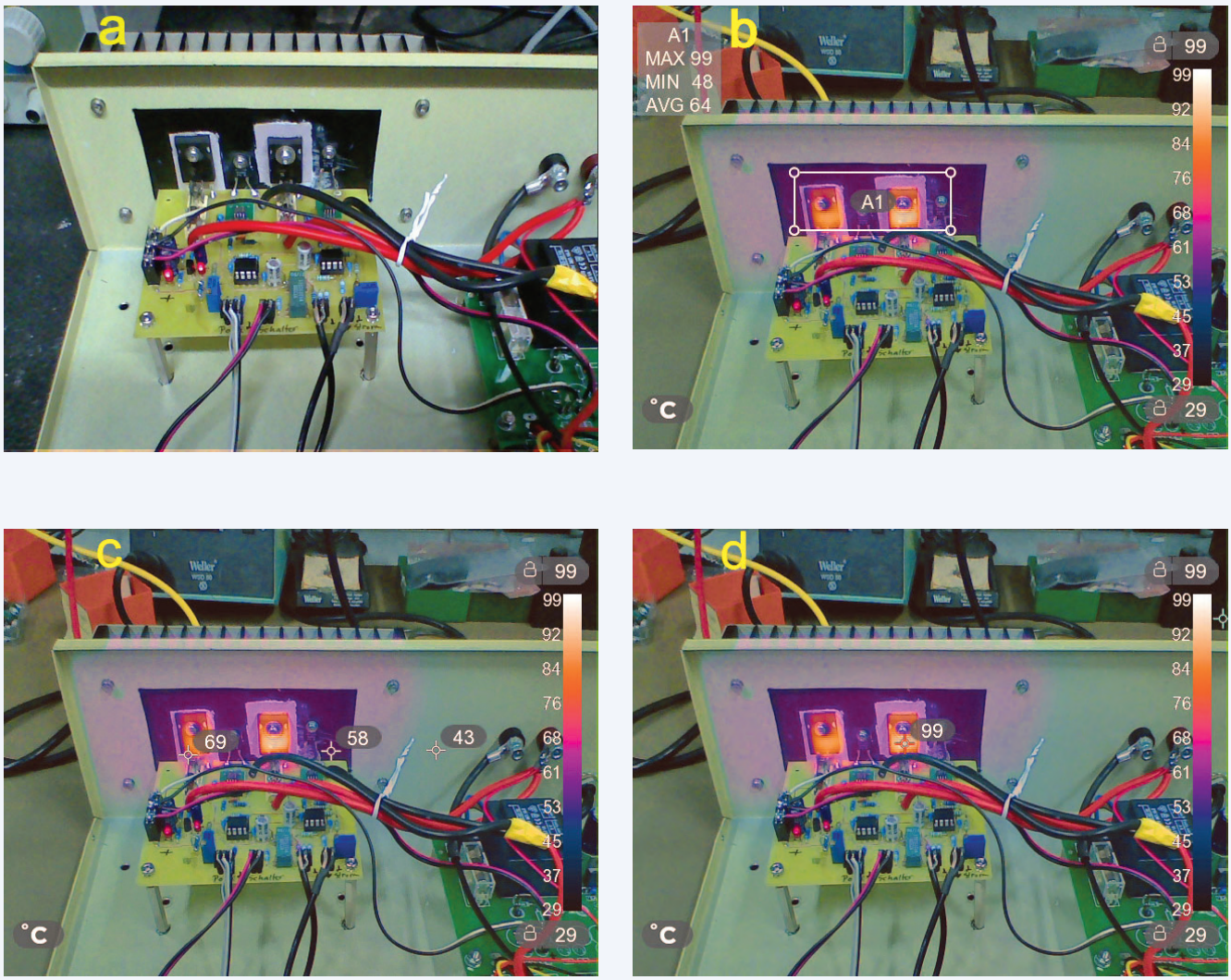


Bild 3. Der Prototyp einer elektronischen Last im Normalbild (3a), als Mischbild (3b), Mischbild mit drei Messpunkten (3c) und Mischbild mit maximaler Temperatur (3d).

etwas mehr in den Telebereich. Rein äußerlich unterscheiden sich die beiden Versionen durch die Farbe der Abdeckung rund um die Objektive: Bei der normalen Kamera ist sie grau und bei der Pro-Version rot. Wie man an Bild 1 sehen kann, habe ich die Seek Shot Pro getestet.

Generelles

Während die Kamera geladen wird, kann man sich in Ruhe die Bedienungsanleitung durchlesen und sich die entsprechenden Videos auf der Webseite [1] anschauen – zumindest ich bin so vorgegangen. Es lohnt sich, die verschiedenen Möglichkeiten der Kamera im Einzelnen auszuprobieren, bevor man die ersten ernsthaften Bilder machen möchte. Diese Kamera arbeitet weitgehend automatisch und es gibt daher keine Einstellmöglichkeiten für Blende, Belichtungszeit etc. Außerdem sind beide Objektive vom Typ „fixed focus“ - damit entfällt auch die Scharfeinstellung. Die Naheinstellungsgrenze liegt nach Aussagen des Supports bei ca. 10...15 cm. Bis in diesen Bereich sollte auch das Überlagern von Normal- und Thermobildern gelingen.

Vor weiteren Details möchte ich zunächst das in meinem Augen Wichtigste erwähnen: Alle möglichen Analysen können auch **nach** der Aufnahme vorgenommen werden und als Kopie abgespeichert werden. Man kann also von einer Aufnahme viele verschiedene Bilder erstellen und speichern. Gerade bei nicht wiederholbaren Situationen ist das praktisch; man macht einfach mehrere Aufnahmen und wertet sie später aus.

Bildschirm

Nach dem Einschalten dauert es einige Sekunden, bis im oberen Balken des Displays (siehe **Bild 2**) links die Uhrzeit, das Datum und rechts der Ladezustand des Akkus angezeigt wird. In der Mitte des Balkens zeigt ein Pfeil nach unten. Mit ihm wird ein Menü geöffnet, in dem man Dinge wie etwa die Helligkeit des Bildschirms einstellen kann. Bezüglich Temperatur kann aus den Einheiten *Celsius*, *Fahren-*

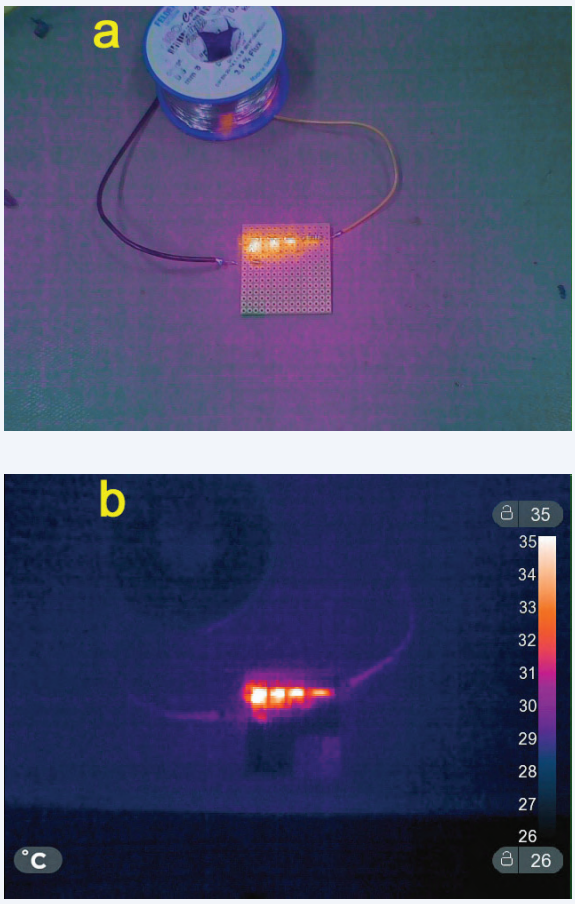


Bild 4. (4a) ist die Lochraster Platine im Overlay und (4b) ein reines Thermobild.

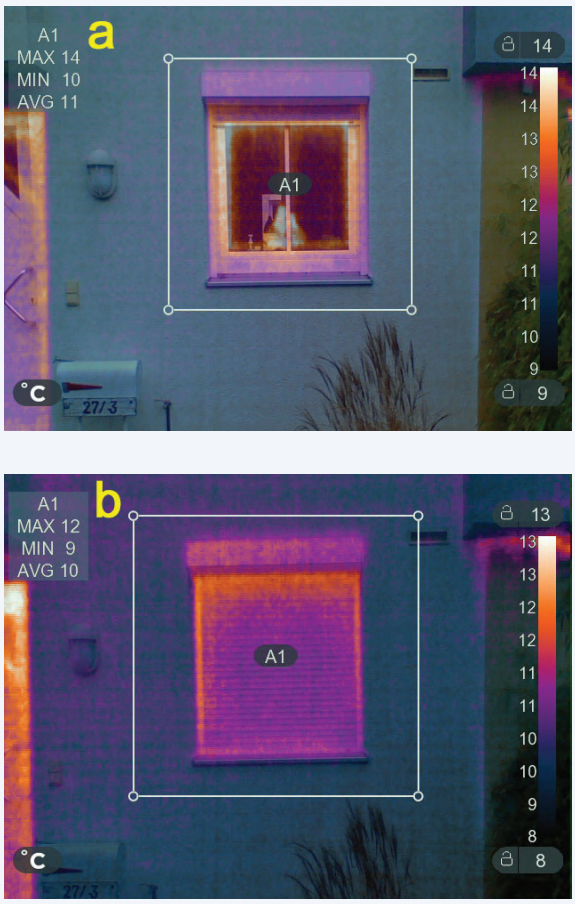


Bild 5. Ein Fenster meines Hauses mit offenem (5a) und geschlossenem (5b) Rollladen.

heit und *Kelvin* ausgewählt werden. Außerdem kann man hier den Emissionsgrad der zu fotografierenden Oberflächen einstellen. Dies ist wichtig, damit die Bilder die korrekten Temperaturen zeigen.
Mit dem Taschenlampen-Icon rechts kann man die LED auf der Front einschalten und ein Blitz-Icon dient zur Aktivierung der Blitzfunktion der LED. Weiter lassen sich in diesem Dialog das WLAN aktivieren und einstellen oder eine Beschreibung der Kamera über die integrierte Hilfefunktion abrufen. Man gelangt über das übliche Zahnrad-Icon zu einem Menü zur Einstellung von Zeit, Datum, Menüsprache und mehr. Nach der Rückkehr aus dem Menü zeigt sich das Display etwa so wie in Bild 2.
Am unteren Rand des Display befinden sich vier Icons. Mit dem linken Landschafts-Icon gelangt man zu einer Galerie bereits aufgenommener Bilder, die sich dann bearbeiten lassen. Dazu später mehr. Mit dem zweiten Icon von links öffnet sich ein weiteres Menü, in dem die verschiedenen Temperaturmessungen ausgewählt werden. Ein Klick auf das linke Einpunkt-Icon zeigt die Temperatur in der Mitte des Bildes an. Durch Verschieben der Kamera lassen sich so verschiedene Punkte messen. Allerdings braucht es dazu eine ruhige Hand, da die Messung mit merklicher Verzögerung erfolgt. Das Dreipunkt-Icon aktiviert bis zu drei Messpunkte, die sich manuell verschieben und bei Bedarf wieder löschen lassen. Durch das Icon aus einem Quadrat mit Punkten an den Ecken aktivieren sich drei Rechtecke, deren Lage und Größe sich ändern lässt. Die Kamera zeigt dann die maximale, minimale und die Durchschnittstemperatur für jedes dieser Felder an. Diese Felder lassen sich natürlich wieder löschen. Das Icon mit Plus- und Minuszeichen ganz rechts aktiviert die Messung des wärmsten und des kältesten Punktes im Bild. Diese Funktionen schaut man sich am besten nochmal genau in den Schulungsvideos an.
Das Augen-Icon öffnet ein Menü zum Ändern der Ansicht. Wählt man hier das „normale" Auge, wird das normale Bild der optischen Kamera angezeigt. Das Augen-Icon rechts mit den unterbrochenen Linien führt zur reinen Ansicht der Thermobildes. Das mittlere Bild indiziert beides, die Überlagerung von optischem und Thermobild als *Fusion View*. Ein Klick bringt zwei Regler hervor. Mit dem linken stellt man die Bildmischung zwischen optischem und Thermobild ein. Mit dem rechten lassen sich beide Bilder vertikal verschieben und sie so bei geringer Entfernung trotzdem gut zur Deckung bringen, da die beiden Objektive ja eine vertikale Lagendifferenz aufweisen, die sich bei kürzerer Entfernung stärker auswirkt.
Über die drei Farbpunkte rechts unten kann man die Farbpalette auswählen. Das Bild der Thermokamera weist ja eine waschechte Falschfarbendarstellung auf. Je nach den Temperaturen im Bild ist eine spezifische Palette unter Umständen besser zur Diskriminierung der Temperaturunterschiede geeignet als eine andere.

Testbilder

Die Temperaturskala ist normalerweise „autoranging“, lässt sich aber fixieren, damit man Bilder besser vergleichen kann. **Bild 3a** zeigt das normale optische Bild einer elektronischen Last – **Bild 3b** hingegen die Überlagerung von optischem und Thermobild. Die maximale Temperatur des ausgewählten Rechtecks beträgt 99 °C. Rechts in den Bildern sieht man jeweils die Farbskala der Temperaturen. In **Bild 3c** wurden drei Messpunkte für die Temperatur des Kühlkörpers und des Gehäuses positioniert. In **Bild 3d** wird die maximale erfasste Temperatur des Bildes angezeigt.

Ein anderes Beispiel: **Bild 4a** zeigt das Überlagerungsbild einer Lochrasterplatine. Hier habe ich fünf unterschiedliche 100-Ω-Widerstände (1 x bedrahtet plus die SMD-Formate 1206, 0805, 0603 und 0402) in Reihe geschaltet und einen Strom von 10 mA durch die Serienschaltung fließen lassen. Obwohl die Temperaturunterschiede recht klein sind, sind kann man die Unterschiede gut erkennen. Das enge Temperaturspektrum ist gut im Thermobild von **Bild 4b** zu sehen.

Selbstverständlich ist eine Thermo-Kamera nicht nur ein gutes Hilfsmittel für Elektroniker, sondern man kann damit sehr gut Isolationsmängel an Häusern erkennen. Aus diesem Grund habe ich auch zwei Bilder von einer Hauswand gemacht. **Bild 5a** zeigt ein Fenster meines Hauses mit geöffnetem Rollladen. In **Bild 5b** ist der Rollladen komplett geschlossen. Man kann immerhin eine Differenz von 2° bei der Maximaltemperatur erkennen.

Zu guter Letzt noch eine Art Thermo-Selfie. Auch wenn Infrarot-Aufnahmen von Menschen so schön eindrucksvoll sind: Die Kamera ist nicht für die Fiebermessung bei Menschen geeignet. Laut **Bild 6** hätte ich mich besser sofort ins Bett legen sollen. Doch erstens ist die Genauigkeit mit ±2 ° angegeben, und zweitens hängen die gemessenen Temperaturen stark von der Einstellung des Emmisionsgrads ab – die ich anscheinend nicht wirklich korrekt vorgenommen habe.

Alle diese Bilder habe ich nur einmal aufgenommen, anschließend in der Galerie bearbeitet und dann als Kopie abgespeichert. Dazu öffnet man die Galerie durch Antippen des Landschafts-Icons, wählt mit den Pfeiltasten das richtige Bild und aktiviert die Bearbeitung. Ist man fertig, speichert man das Bild als Kopie ab. So bleibt das Original für andere Bearbeitungen erhalten.

Per WLAN und der entsprechenden App auf einem Handy oder Tablet soll all dies auch remote möglich sein. Das habe ich allerdings nicht überprüft. Auch eine App für den PC, wie sie für andere Thermokameras verfügbar ist, wäre höchst wünschenswert. Die Bearbeitung auf dem kleinen Bildschirm mit Fingern von Männerhänden verlangt nämlich viel Konzentration und Geduld. Bei mir kommt diese Infrarot-Kamera hauptsächlich zur Überprüfung elektronischer Schaltungen zum Einsatz. Und das kann sie sichtbar gut.

Fazit

Eine Thermokamera ist ein hervorragendes Hilfsmittel zum „Debugging“ von Problemen, die sich durch ihre thermischen Effekte verraten. Eine IR-Kamera ist um Klassen effektiver, flexibler und schneller als das manuelle Anbringen von Temperaturfühlern an elektronischen Schaltungen. Außerdem ist es äußerst hilfreich, zunächst lediglich Bilder aufzunehmen und sie erst nachträglich auszuwerten und zu speichern. Die Überlagerung von Normal- und Thermobild bringt mehr optische Anhaltspunkte und vereinfacht so die Orientierung und damit die Suche nach dem (zu) heißen Bauteil. Da die Auflösung des eigentlichen Thermobildes nicht sonderlich hoch sein muss, gehe ich davon aus, dass auch die niedrigere Auflösung der normalen Kamera-Ausführung akzeptabel wäre. Ich kann dies aber nicht aus eigener Erfahrung sagen, da ich mich für die bessere und auch teurere Pro-Version entschieden habe. Was ich aber sagen kann: Die Seek Shot Pro braucht für ihre etwas über 800 € keinen Vergleich mit einer zehnfach teureren (aber inzwischen auch mehr als zehn Jahre alten) Profikamera von Flir zu scheuen. Und das ist doch schon was!

Bild 6. Selbstportrait in Infrarot.

200654-02

Sie haben Fragen oder Kommentare?
Gerne können Sie sich an die Elektor-Redaktion wenden unter der E-Mail-Adresse: redaktion@elektor.de.

Ein Beitrag von
Review und Text:
Alfred Rosenkränzer

Redaktion:
Dr. Thomas Scherer
Layout: **Giel Dols**

PASSENDE PRODUKTE

Seek Shot Pro
www.elektor.de/seek-shot-pro-thermal-imaging-camera-320x240

WEBLINK

[1] **Webseite zur Seek Shot:**
www.thermal.com/seekshot-series.html

Objektorientiertes Programmieren

Eine kurze Einführung unter C++

Von **Roland Stiglmayr**

Die Methode der objektorientierten Programmierens (OOP) wird bereits seit über 40 Jahren genutzt, wegen der besseren Verfahren zur Qualitätssicherung, der einfacheren Pflege der Software und der hervorragenden Möglichkeit, Software im Team zu erstellen. In den letzten Jahren hielt OOP auch im Bereich der Embedded-Programmierung Einzug. Da uns selbst Arduino mit OOP konfrontiert, scheint es eine gute Idee zu sein, sich etwas näher mit dem Thema zu beschäftigen!

Dieser Artikel kann nur ganz grob die Grundzüge der OOP aufzeigen, schafft aber die Grundlagen für das Verständnis weitergehender Ausführungen und macht vielleicht auch Lust, eigene objektorientierte Programme zu schreiben.
Bei der Softwareentwicklung kennen wir zwei unterschiedliche Denkmodelle. Das eine basiert auf dem Konzept der prozeduralen Programmierung, das andere auf der objektorientierten Programmierung (OOP). Die prozedurale Programmierung ist die, die wir schon immer anwenden, zum Beispiel bei der Programmierung in Assembler oder auch in Standard-C. Kennzeichnend dabei ist, dass das Programm in kleine Module, also Prozeduren oder Funktionen zerlegt und sequentiell abgearbeitet wird. Die Module

Listing 1. Die Klassendeklaration von Virt_M (Lektion_1).

```
class Virt_M
{
  private:                              //zugriff auf diese mitglieder nur
                                        //innerhalb der klasse
    float voltage;                      //fiktive spannung/ V
    float current;                      //fiktiver strom/ A
    float p_result;                     //berechnete leistung/ W
    float r_result;                     //berechneter widerstand/ Ohm.

  public:                               //zugriff auf diese mitglieder auch von
                                        //außerhalb der klasse
    String s= ("public: Demoattribut s, typ String"); //zur demo

//  der konstruktor, seine methode wird beim anlegen eines objektes
//  ausgeführt

    Virt_M (float v, float c);          //deklaration des konstruktors

//  deklaration der methoden von Virt_M

    float get_P ();                     //methode zum lesen der leistung
    float get_R ();                     //methode zum lesen des widerstands
    float get_Voltage ();               //methode zum lesen der spannung
    float get_Current ();               //methode zum lesen des stroms
    void prep_Meas (float v, float c);  //methode simuliert messung

  private:                              //zugriff nur innerhalb der klasse

    void set_Voltage (float v);         //schnittstellenmethode, schreibt spannung
    void set_Current (float c);         //schnittstellenmethode, schreibt strom

};                                      //klassen deklaration ende
```

kommunizieren über gemeinsame, oft global deklarierte Daten und tauschen darüber ihre Verarbeitungsergebnisse aus. Die OOP hingegen kapselt die Daten und die darauf anzuwendenden Funktionen in Objekten. Der Zugriff auf Daten und Funktionen geschieht über hierfür vorgesehene Funktionen, die auch Schnittstellen genannt werden. Die Kapselung schützt so die Daten vor fehlerhaftem Zugriff.
Was zeichnet also die OOP aus? Bei der OOP werden Funktionseinheiten mit logisch zusammengehörenden Mitgliedern wie Daten und Funktionen gebildet. Die Funktionen nennt man hier Methoden. Denkt man etwa an einen Datenlogger, dann könnte man die Messwerterfassung mit ihren vielen gleichartigen Kanälen, die Verarbeitungseinheit und die Benutzerschnittstelle als jeweils eine Funktionseinheit verstehen. Eine Funktionseinheit kann Eigenschaften anderer Funktionseinheiten erben, so dass sich eine modulare Struktur ergibt, die auf Basis-Einheiten aufbaut. Dabei bleibt der Schutz der Daten und der Methoden durch das Prinzip der Kapselung erhalten.
Wir können uns so eine Funktionseinheit als eigenen Datentyp namens Klasse vorstellen. Die Klasse stellt eine Bauvorschrift für die daraus zu erstellenden Variablen dar. Die Variablen heißen Objekt oder Instanz dieser Klasse. Um Missverständnisse zu vermeiden, sollte immer auf den Unterschied zwischen Klasse und Objekt geachtet werden! Aus einer Klasse können beliebig viele Objekte erzeugt werden. So könnte jede Person einer Firma ein Objekt darstellen, wobei jedes dieser Objekte aus ein und derselben Klasse erzeugt wurde.
Genug der Theorie, kommen wir jetzt zur Praxis! Wie immer geht das am einfachsten mit fertigen Beispielen, die man nach eigenen Vorstellungen verändert und erweitert. Wenn dies in einer vertrauten Programmierumgebung stattfindet, spart man sich viel Einarbeitungszeit. Deshalb wird die gut verfügbare und kostenlose Arduino-IDE verwendet. Der integrierte Compiler ist ein vollwertiger C++-Compiler und damit bestens zur objektorientierten Programmierung geeignet.Ein beliebiges Arduino-Board, auf das die Sketches hochgeladen werden, hängt am USB-Port des Computers. Unsere Beispielprogramme, in der Arduino-Welt Sketches genannt, sind in sieben „Lektionen“ geordnet und können/sollten von der Projekt-Seite [1] zu diesem Artikel heruntergeladen, entpackt und in den Sketch-Ordner der Arduino-IDE kopiert werden. Damit sind die Vorbereitungen auch schon abgeschlossen. Den Beispielen liegt die Idee eines virtuellen Messgerätes zugrunde. Die Beispielsketche sind zwar für ein „echtes“ Messgerät nicht tauglich, zeigen aber an einer uns gut verständlichen Aufgabe sehr schön die Grundzüge der OOP.

Die erste Klasse

Zu Beginn wollen wir das Erstellen einer Klasse, die Bedeutung des Zugriffsspezifizierers, die Grundzüge der Kapselung und das Erzeugen von Instanzen kennenlernen. Dazu rufen wir den Sketch aus *Lektion_1* auf, kompilieren ihn und laden ihn auf das Board hoch. Bevor wir uns mit dem seriellen Monitor das vom Programm erzeugte Ergebnis ansehen, widmen wir uns zuerst dem Quelltext in **Listing 1**.
Dieser beginnt mit dem Bezeichner `class` und dem Namen `Virt_M`. Damit wird dem Compiler gesagt, dass eine Klasse mit dem Namen `Virt_M` formuliert wird. Es folgt die Deklaration der klasseninternen Daten, auch Attribute genannt, und der Methoden unserer Klasse. Die Zugriffskontrolle auf die Klassenmitglieder (Englisch *member*), also die Attribute und Methoden der Klasse, wird durch die Zugriffsspezifizierer `private` und `public` festgelegt. Auf als `private` spezifizierte Mitglieder kann nur innerhalb der Klasse zugegriffen werden. Auf als `public` spezifizierte Mitglieder ist auch der Zugriff von außerhalb möglich. Was das bedeutet, werden wir später sehen.
Die erste Methode, die so heißt wie die Klasse und keinen Rückgabewert besitzt, ist der Konstruktor. Der Konstruktor wird immer beim Anlegen eines Objekts vom Typ dieser Klasse aufgerufen. Zur Initialisierung der Objekt-Attribute werden dem Konstruktor üblicherweise Werte übergeben.
Kurz zu den weiteren Methoden: Die mit `get_` beginnenden Methoden erlauben das Lesen der privaten Attribute der Klasse, die mit `prep_` beginnende Methode das Setzen dieser Attribute. Diese Methoden sind öffentlich und stellen so die Schnittstellen auf private Attribute her. Da die `set_`-Methoden als `private` spezifiziert sind, können sie nur innerhalb der Klasse genutzt werden. Der `String s` dient der Demonstration des Zugriffsspezifizierers `public`. Die schließende Klammer mit dem angehängten Semikolon beendet die Klassendeklaration.
Es folgt die Definition der eben deklarierten Methoden. Mit Ausnahme der Syntax ist dies identisch mit der Definition von Funktionen in Standard-C. Die Syntax ist:

```
Klassenname::Methodenname(Parameter-Liste)
```

Mit dem Bezugsoperator, dem zweifachen Doppelpunkt, wird der Bezug zur Klasse hergestellt.
Unsere erste Klasse ist jetzt fertig und wir können daraus Objekte erstellen. Wie bei einer Funktionsdeklaration nennen wir den Namen der Klasse als Typ, gefolgt vom Namen unseres Objekts und, falls vereinbart, eine Liste mit Übergabewerten (**Listing 2**). Beim Erzeugen eines Objekts, auch Instanziieren genannt, werden die Konstruktormethode der Klasse aufgerufen und die Attribute dieses Objekts angelegt. Ganz wichtig ist, dass man sich bewusst macht, dass jedes Objekt seinen eigenen, individuellen Datenteil, eben die Attribute, besitzt.
Noch gar nicht erwähnt wurde, was ein Objekt dieser Klasse tut. Es wird hier eine Messwerterfassung simuliert, die fiktive Werte für Spannung und Strom erzeugt und diese zusammen mit den resultierenden Werten für Widerstand und Leistung über entsprechende Methoden verfügbar macht. Der Aufruf einer öffentlichen Methode geschieht mit:

```
Objektname.Methodenname ();
```

Auf die durch `private` geschützten Attribute kann nur über die hierfür vorgesehenen Methoden zugegriffen werden. Versucht man den direkten Zugriff, dann liefert der Compiler eine Fehlermeldung. Durch Entfernen der Kommentar-Anweisung innerhalb der Setup-Routine lässt sich das ausprobieren (**Listing 3**).
Als weiteres Beispiel zur Zugriffskontrolle wurde der `String s` bewusst als `public` deklariert. Auf so deklarierte Attribute kann mit `Objektname.Attributname` direkt zugegriffen werden. Aber Vorsicht! Das Objekt schützt `public`-Mitglieder nicht gegen fehlerhafte Benutzung im Anwenderprogramm.
Das Hauptprogramm bringt Leben in die Messwerte. Hierzu wird die Methode `prep_Meas` benutzt, die die aktuellen Werte über die `get`-Methoden ermittelt und manipuliert.
Die ausgiebig kommentierten Programmbeispiele enthalten viele Hinweise und tragen so zum weiteren Verständnis bei. Nach dem

Listing 2. Erzeugen der Instanzen (Lektion_1).

```
//  erzeugen der instanzen/ objekte vom
//  typ Virt_M. übergabe der startwerte

  Virt_M My_M1 (220.0,0.5);      //aufruf des konstruktors von Virt_M
                                 //erzeugt das objekt My_M1

  Virt_M My_M2 (10.0,1.0);       //aufruf des konstruktors von Virt_M
                                 //erzeugt das objekt My_M2
```

Listing 3. Demo der Zugriffskontrolle (Lektion_1).

```
//  zum test der zugriffskontrolle Kommentar-anweisung "//" löschen
//  -------------------------------------------------------

//  My_M1.voltage= 0;             //error > attribut is private
//  My_M1.set_Voltage(0);         //error > methode ist private
  String str= (My_M1.s);          //zulässig > attribut s ist public
  Serial.println (My_M1.s);
  delay (2000);
```

Start der Anwendung werden durch die Setup-Routine die Attribute der Objekte ausgegeben, die wir uns im seriellen Monitor anschauen können (**Bild 1**).

Klasse als Programmbibliothek

Bibliotheken sind nichts Neues, die gibt es auch in Standard-C, aber bei der OOP werden sie sehr intensiv eingesetzt. Der Grund dafür ist, dass man den Quelltext einer Klasse nicht im Detail kennen muss, um damit zu arbeiten. Das Benutzen von Bibliotheken unterstützt die Wiederverwendbarkeit von Programmteilen und damit die Modularisierung eines Programms.

Ein Beispiel zeigt *Lektion_2*, die die Klasse aus dem ersten Sketch als Programmbibliothek darstellt. Eine Bibliothek besteht aus zwei Dateien, eine Header-Datei (.h), die den Deklarationsteil der Klasse enthält, und die Quelltextdatei (.cpp) mit der Methodendefinition der Klasse. Obwohl nicht erforderlich, ist es sinnvoll, beiden Dateien den gleichen Namen zu geben. Um unsere Klasse `Virt_M` in Form einer Bibliothek darzustellen, müssen lediglich der Deklarationsteil in die Header-Datei und der Definitionsteil in die Quelltextdatei übertragen werden. Während der Entwicklung ist es empfehlenswert, die Bibliotheksdateien im selben Verzeichnis wie das Anwenderprogramm zu speichern. Dabei hilft uns die Tab-Funktion von Arduino, doch dazu später.

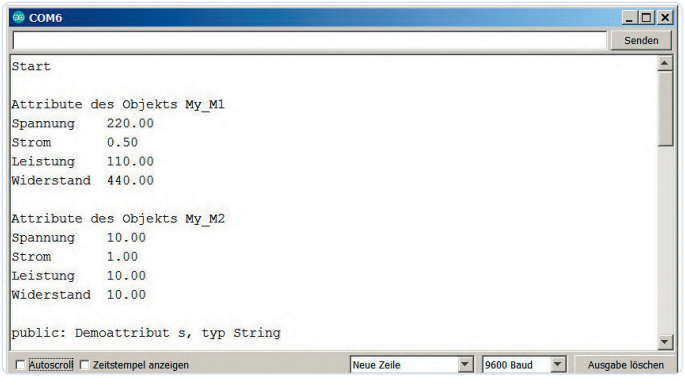


Bild 1. Ausgabe der Setup-Routine in Lektion_1.

Üblich ist es, am Anfang der Headerdatei abzufragen, ob der Compiler diese bereits eingebunden hat, zum Beispiel, wenn eine andere Headerdatei unseren Header schon inkludiert hat. Dann wären nämlich gleichnamige Mitglieder mehrfach deklariert. Die an den Präprozessor gerichtete Syntax hierzu lautet:

```
#ifndef  Virt_M_h  //teste ob Virt_M_h, ein
      //beliebiges Hilfs-Symbol, bereits besteht
#define Virt_M_h   //wenn Hilfssymbol nicht
      //vorhanden ist, dann erzeuge es
  class ....       //und deklariere die Klasse
    {   ....  };   //Deklarationsende
#endif             //beende bedingte Kompilierung
```

In der Quelltextdatei wird mit der Anweisung `#include "Virt_M.h"` die Headerdatei eingefügt. Bei unserem Beispiel wird auch noch `Arduino.h` eingebunden. Dies ist erforderlich, da zu diesem Zeitpunkt der Compiler die Standardbibliotheken noch nicht kennt. Damit ist unsere Bibliothek fertig. Um jetzt die Klasse `Virt_M` im Anwenderprogramm zu verwenden, wird einfach mit `#include "Virt_M.h"` die Header-Datei geladen.

Arduino-Interna: Oft liefern Arduino-Bibliotheken bereits eine Instanz ihrer Klassenbibliothek. So ist zum Beispiel bei der Wire-Bibliothek *Wire* ein Objekt der Klasse *TwoWire*. Das Objekt wird über `#include <Wire.h>` geladen.

Nach Abschluss der Entwicklung können wir unsere Bibliotheksdateien in einem eigenen Verzeichnis speichern. Arduino legt dafür im Sketchbook-Verzeichnis das Verzeichnis *libraries* an. Wir legen dort ein weiteres Verzeichnis an, in dem wir unsere Dateien abspeichern. Das funktioniert auch über die IDE mit *Sketch -> Bibliothek einbinden -> .ZIP Bibliothek hinzufügen*.

Wie bereits erwähnt, ist die Tab-Funktion der Arduino-IDE sehr praktisch beim Schreiben mehrerer zu einem Projekt gehörenden Dateien. Aktiviert wird sie über das kleine Dreieck am rechten oberen Rand des Fensters (**Bild 2**).

Vererbung – die Königsdisziplin

Beschäftigt man sich mit der OOP, dann stößt man sehr schnell auf das Thema Vererbung. Schauen wir uns die Vorteile dieses Verfah-

Bild 2. Arduinos Tab-Funktion (Lektion_2).

rens an! Unter Vererbung versteht man das Verfügbarmachen der Attribute und Methoden einer Basisklasse für eine erbende Klasse. Die erbende Klasse wird auch Kindklasse, abgeleitete Klasse oder Unterklasse genannt. Die Basisklasse bezeichnet man entsprechend auch als Elternklasse, Oberklasse oder Superklasse. Die Kindklasse fügt weitere Eigenschaften zu den geerbten hinzu und kann auch weiter vererbt werden. Da aus einer vererbenden Klasse viele unterschiedliche abgeleitete Klassen entstehen können, ist die entstehende Struktur nicht zwangsläufig geradlinig, sondern beliebig verzweigt. Eine sich weit verzweigende Verwandtschaft! *Lektion_3* zeigt, wie die Vererbung in unserer Aufgabe angewandt werden kann. Damit das Beispiel etwas „sinnhafter" wird, verwenden wir jetzt Multiplexadressen, um unterschiedliche Messkanäle auswählen und diese an die erzeugten Objekte binden zu können. `Virt_M` ist die Basisklasse. Hier fällt auf, dass drei Konstruktoren mit einer unterschiedlichen Anzahl von Parametern vorhanden sind. C++ lässt dieses so genannte Überladen von Funktionsdefinitionen generell zu: gleichnamige Funktionen mit unterschiedlicher Anzahl und/oder unterschiedlichen Typen von Parametern. In der OOP wird das als Polymorphie bezeichnet. Wichtig ist der neue Zugriffsspezifizierer `protected`. Damit wird der erbenden Klasse der Zugriff auf die so deklarierten Mitglieder gewährt. In unserem Beispiel heißt dies, dass alle abgeleiteten Klassen auf die `set_Mux`-Methode zugreifen können.
Zur Demonstration, dass dies zulässig ist, wird diese Methode direkt im Deklarationsteil definiert (was allerdings nur bei sehr kurzen Funktionen sinnvoll ist). Die erste abgeleitete Klasse ist `Volt_M`. Sie erbt die Klasse `Virt_M`. Die Syntax hierzu ist:

```
class Volt_M : public Virt_M
     //klasse Volt_M erbt klasse Virt_M
```

Das Schlüsselwort `public` bewirkt, dass die Zugriffskontrolle auf die Mitglieder von `Virt_M` erhalten bleibt. Zu den geerbten Mitgliedern fügt `Volt_M` das Attribut `sV` , den Konstruktor und die Methode `get_Unit` hinzu. Die Deklaration des Konstruktors bringt nichts Neues, die Definition aber schon. Es wird hier der Konstruktor 2 von `Virt_M` aufgerufen. Bei der Instanziierung werden die an `Volt_M` übergebenen Parameter an die `Virt_M`-Klasse weitergereicht, der Startwert `ini_m` direkt über den Konstruktor von `Virt_M`, die Multiplexadresse über die `set_Mux`-Methode von `Volt_M`. Die Definition des Konstruktors lautet:

```
Volt_M::Volt_M (byte v_mux, float ini_m):Virt_M (ini_m)
                //ini_m über Konstruktor von Virt_M
   { set_Mux (v_mux); }    //v_mux über set-Methode
```

Die Klasse `Amp_M` ist ebenfalls ein Erbe von `Virt_M`. Sie ist ähnlich wie `Volt_M` aufgebaut, mit dem Unterschied, dass der Konstruktor den Konstruktor 3 von `Virt_M` aufruft. Ohne eine zusätzliche Methode wird so die gesamte Werteübergabe an `Virt_M` realisiert. Bemerkenswert ist, dass ein Erbe eine vererbte Methode überschreiben kann.
Die Klassen sind jetzt vollständig formuliert, so dass wir Instanzen davon anlegen können. Die Objekte `M1`, `M2` und `M3` vom Klassentyp `Virt_M` zeigen den Aufruf des überladenen Konstruktors. `My_Volt_M` und `My_Amp_M` sind Objekte der vererbten Klassen `Volt_M` und `Amp_M`. Nach Starten des Programms lässt sich das Verhalten unserer Objekte mit dem seriellen Monitor verfolgen und verstehen. Im Sketch sind bewusst ungültige Zugriffe eingebaut, die im Beispiel anfänglich auskommentiert sind. Um die Fehlermeldungen des Compilers studieren zu können, müssen wir die Kommentaranweisungen `//` entfernen.
Es muss uns bewusst sein, dass die erbenden Objekte die geerbten Objekte benutzen, als wären sie ein Teil von ihnen selbst. Allerdings hat jedes Objekt seinen eigenen individuellen Datensatz. Das ist so ähnlich, als wenn der Code von `Virt_M` einfach in `Volt_M` und `Amp_M` einfügt wird. Neben dem Mehraufwand hätte dies allerdings den großen Nachteil, dass bei Änderungen an `Virt_M` alle Klassen, die `Virt_M` enthalten, ebenfalls geändert werden müssten.

Eine Klasse zum Freund haben

Im letzten Beispiel haben wir gesehen, dass es aufgrund der Kapselung nicht möglich ist, die Multiplexadresse eines Objekts zu ändern. Gerade bei der hardwarenahen Programmierung ist es aber oft wünschenswert, erweiterte Zugriffsmöglichkeiten zu haben. Wollen wir beispielsweise beim Debuggen alle Multiplexadressen sequentiell ansprechen, dann müssen wir für jede Adresse ein Objekt erstellen. Eine weitere Möglichkeit wäre es, nach dem Aufruf das Objekt zu zerstören und dann wieder neu zu erstellen. Natürlich könnten wir die Basisklasse von vornherein so formulieren, dass der Zugriff öffentlich ist. Damit würden wir jedoch auf einen Vorteil der OOP, den Schutz der Daten, verzichten. *Lektion_4* zeigt uns eine schöne und saubere Lösung. Wir vererben die Basisklasse `Virt_M` an eine neue

Listing 4. Klassendeklaration von Virt_M mit Debug_M als „friend“ Klasse (Lektion_4).

```
class Virt_M
{
  friend class Debug_M;                //klasse Debug_M erhält zugriff
                                        //auf private mitglieder

  private:                              //klasseninterne mitglieder
    float value=0;                      //fiktiver messwert, default
    byte mux_adr= MUX_INVALID;          //mux-adresse zur simulation

  public:                               //mitglieder auch von außen erreichbar
    Virt_M (byte mux, float ini_m);     //konstruktor mit startwerten
    void set_Value (float val);         //methode setzt fiktiven messwert
    float get_Value ();                 //methode liest fiktiven messwert
    byte get_Mux ();                    //methode liest mux adresse
    String s= "Virt_M Demostring";      //zur demo

};                                      //klassen deklaration ende
```

Listing 5. Deklaration des Strukturdatentyps t_result (Lektion_5).

```
struct t_result    //typ zur rückgabe von daten
 {
   byte mux;       //multiplexadresse
   String s;       //ein string
   float val;      //messwert vom typ float
 };
```

Listing 6. Deklaration der Attribute der Klasse Dev_M (Lektion_5).

```
class Dev_M
{
  private:
    Volt_M *ptr_v;                      //zeiger auf ein objekt vom typ Volt_M
    Amp_M *ptr_a;                       //zeiger auf ein objekt vom typ Amp_M
    t_result result;                    //zur rückgabe aus funktion
    const String sP=( "Power/ W   : ");
    const String sR= ("Resist/ Ohm: ");
```

Klasse namens `Debug_M` und erklären diese in der Basisklasse zum Freund. Damit ist die Funktion von `Debug_M` identisch mit der von `Volt_M` mit der Ausnahme, dass sie auch auf die privaten Attribute von `Virt_M` zugreifen kann. Um die Multiplexadresse ändern zu können, stellt `Debug_M` die öffentlich deklarierte `set`-Methode zur Verfügung. **Listing 4** zeigt uns die Deklaration der `friend`-Klasse.

Die Klasse `Virt_M` enthält keine Methode zum Setzen der Multiplexadresse. Stattdessen erhält der Konstruktor zwei Übergabewerte zum Initialisieren der Klassenattribute. Eines davon ist die Multiplexadresse. Beim Instanziieren der Basisklasse wie auch der Unterklassen werden die Parameter übergeben.

Wieder zeigt uns die Ausgabe der `Setup`-Routine das Ergebnis unserer Arbeit.

`Friend`-Klassen werden selten eingesetzt, obwohl sie doch den Vorteil bieten, dass sie die vorhanden Methoden nutzen, die gewünschte Funktionalität bieten und trotzdem bei richtiger Anwendung das Sicherheitskonzept der Kapselung nicht schwächen.

Eine Klasse greift auf die Methoden anderer Klassen zu

Bei den letzten Beispielen haben wir die Berechnung der Leistung und des Widerstands im Hauptprogramm durchgeführt. Die dazu erforderlichen Attribute von `Volt_M` und `Amp_M` haben wir über deren Methoden gelesen. Jetzt werden wir dies in einer eigenen Klasse realisieren. *Lektion_5* definiert dafür die neue Klasse `Dev_M`, die Zugriff auf die Methoden der Objekte von `Volt_M` und `Amp_M` erhält. Dazu werden beim Erstellen einer Instanz von `Dev_M` über den Konstruktor Zeiger auf die aufzurufenden Objekte übergeben. Zunächst erstellen wir jedoch mit `struct` einen Strukturdatentyp, um das aus mehreren Elementen bestehende Ergebnis darzustellen. Der Datentyp trägt den Namen `t_result` (**Listing 5**).

Listing 7. Die Konstruktormethode von Dev_M speichert die Zeiger (Lektion_5).

```
//  der konstruktor übergibt die zeiger
//  auf die objekte von Volt_M, Amp_M
//  ---------------------------------
  Dev_M::Dev_M (Volt_M *ptrv, Amp_M *ptra)
    {
       ptr_v= ptrv;       //zeiger auf ein objekt von Volt_M
       ptr_a= ptra;       //zeiger auf ein objekt von Amp_M
    }
```

Listing 8. Definition der get_Powr-Methode von Dev_M (Lektion_5).

```
t_result Dev_M::get_Powr ()      //methode ermittelt akt. leistung
{
  result.s= sP;                  //größe, einheit
  result.val= ptr_v -> get_Value () * ptr_a -> get_Value ();
  return result;                 //rückgabe der werte
}
```

Listing 9. Instanziieren der Objekte, Übergabe der Zeiger auf die Objekte (Lektion_5).

```
Volt_M My_Volt_1 (V_MUX0, 1.00);          //objekt vom typ Volt_M
Volt_M My_Volt_2 (V_MUX1, 2.00);          //objekt vom typ Volt_M
Amp_M My_Amp_1 (A_MUX0, 1.10);            //objekt vom typ Amp_M
Amp_M My_Amp_2 (A_MUX1, 2.10);            //objekt vom typ Amp_M
Dev_M My_Dev_1 (&My_Volt_1, &My_Amp_1);   //objekt von Dev_M, zeigerübergabe
Dev_M My_Dev_2 (&My_Volt_2, &My_Amp_2);   //objekt von Dev_M, zeigerübergabe
```

Nun folgt die Deklaration der Klasse `Dev_M`. Bei den Attributen werden die Zeiger `ptr_v` und `ptr_a`, die auf die Klassen `Volt_M` und `Amp_M` zeigen, vereinbart. Der `*`-Operator ist der Hinweis darauf, dass der folgende Name als Zeiger interpretiert wird (**Listing 6**). Die Konstruktormethode speichert die übergebenen Zeiger unter den Attributen ab (**Listing 7**).

Die Methoden von `Dev_M` können jetzt mit Hilfe der Zeiger auf die externen Objekte zugreifen. Der Zugriff erfolgt mit der Anweisung `zeigername -> methodenname`. Der `->`-Operator wird zum Methodenaufruf beim Einsatz eines Zeigers statt dem Punktoperator benutzt (**Listing 8**).

Um eine Instanz von `Dev_M` anlegen zu können, ist es erforderlich, zuvor die Instanzen der indizierten Klassen zu erzeugen. Diese müssen vom Typ `Volt_M` und `Amp_M` sein. Beim Instanziieren von `Dev_M` tragen wir die Zeiger auf diese Objekte in die Parameterliste des Konstruktors ein. Dazu benutzen wir den `&`-Operator (**Listing 9**). Wie die Ausgabe der `Setup`-Routine in **Bild 3** unserer Anwendung zeigt, liefern die Methodenaufrufe der Objekte `My_Dev_1` und `My_Dev_2` alle relevanten Ergebnisse.

Die Mehrfachvererbung

Bei dem vorhergehenden Beispiel mussten wir erst zwei eigenständige Instanzen anlegen, bevor wir die Methoden von `Dev_M` nutzen konnten. Es gibt Anwendungen, bei denen dies sicher sinnvoll ist, bei unserer aber sollte es primär die Verwendung von Zeigern auf Objekte vorstellen. Mancher wird sich gefragt haben, ob man nicht einfach die Basis- und Kind- Klassen an eine weitere Klasse vererben kann. Das ist genau das, was in *Lektion_6* umgesetzt wird, die Mehrfachvererbung. Dabei werden die zwei abgeleiteten Klassen `Volt_M` und `Amp_M` an die Klasse `Dev_M` vererbt. Da die abgeleiteten Klassen `Virt_M` enthalten, erhält `Dev_M` Zugriff auf die Mitglieder aller Klassen.

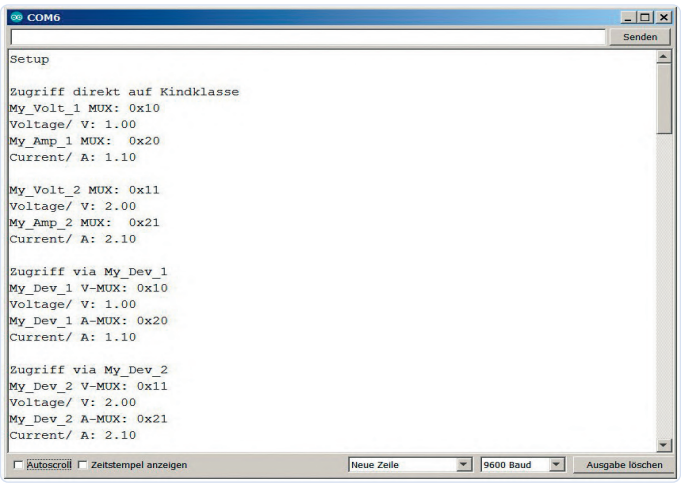


Bild 3. Ausgabe der Setup-Routine in Lektion_5.

Listing 10. Explizite Bezugnahme auf ein Mitglied bei Mehrdeutigkeit (Lektion_6).

```
t_result Dev_M::get_Volt ()           //methode ermittelt akt. spannung
{
  result.s= Volt_M::get_Unit ();      //aufruf von Volt_M-methode
  result.val= Volt_M::get_Value ();   //aufruf von Virt_M-methode
  return result;
}
```

Listing 11. Mehrdeutigkeit der Methode get_Mux (Lektion_6).

```
Serial.print ("My_Dev_1, Volt MUX: 0x");
Serial.println (My_Dev_1.Volt_M::get_Mux(),HEX);  // Methode der Basisklasse
Serial.print (My_Dev_1.get_Volt().s);
Serial.println (My_Dev_1.get_Volt().val);         // Startwert von Instanziierung von My_Dev_1
Serial.println ();
```

Listing 12. Auflösung der Mehrdeutigkeit durch eigene Methoden (Lektion_6).

```
Serial.print ("My_Dev_2, Volt MUX: 0x");
Serial.println (My_Dev_2.get_VMux(),HEX);   //methode der abgeleiteten klasse
Serial.print (My_Dev_2.get_Volt().s);
Serial.println (My_Dev_2.get_Volt().val);   //startwert von instanziierung von My_Dev_M
Serial.println ();
```

Um die komplexen Resultate der `Dev_M`-Methoden einfach zurückzugeben, deklarieren wir wieder den Strukturdatentyp `t_result`. Beim Anlegen der Klasse werden die zu erbenden Klassen - durch Kommata getrennt - nach dem `:`-Operator gelistet:

```
class Dev_M: public Volt_M, public Amp_M
```

Bei der folgenden Deklaration des Konstruktors werden die Parameter, die an die vererbenden Klassen übergeben werden sollen, in die Parameterliste eingetragen:

```
Dev_M (byte v_x, float i_v, byte a_x, float i_a);
       // Konstruktor von Dev_M
```

Die Methoden-Definition des Konstruktors ordnet dann die Parameter den Übergabewerten der vererbenden Klassen zu. Beim Instanziieren von `Dev_M` werden die Konstruktor-Methoden von `Volt_M` und `Amp_M` aufgerufen und dabei die Parameter den Attributen dieser Klassen zugewiesen.

```
Dev_M::Dev_M (byte v_x, float i_v, byte a_x,
   float i_a): Volt_M (v_x, i_v), Amp_M (a_x, i_a)
```

Wie bereits gesagt, können wir über `Dev_M` auf alle Attribute und Methoden der vererbenden Klassen zugreifen, sofern die Zugriffsspezifikation dies erlaubt. `Dev_M` stellt uns dazu entsprechende Methoden zur Verfügung. Dabei entsteht aber ein Problem: Da die zwei vererbten Klassen selbst von der Klasse `Virt_M` abgeleitet sind, haben sie Methoden und Attribute mit gleichem Namen. Das führt zu Mehrdeutigkeiten. Daher müssen wir beim Aufruf über den Bezugsoperator `::` explizit angegeben, welches Mitglied welcher Klasse wir meinen (**Listing 10**).

Unsere Klassen sind jetzt definiert und wir können Objekte daraus instanziieren. Im Prinzip reichen die Methoden von `Dev_M` völlig aus, um eine Anwendung zu schreiben. Zum besseren Verständnis erstellen wir aber auch noch Instanzen der Basisklasse und der abgeleiteten Klassen. Da bei der Instanziierung für jedes Objekt eine andere Multiplexadresse vergeben wird, zeigt sie uns, welches Objekt gerade angesprochen wird.

Nachdem wir die Anwendung gestartet haben, stellt der serielle Monitor die Resultate der Methodenaufrufe dar. Interessant ist die Abarbeitung der `Setup`-Routine. Die Attribute der Basisklasse werden über unterschiedliche Wege gelesen. Beim Zugriff auf die Multiplexadresse des Objekts `My_Dev_1` wird die Methode der Basisklasse genutzt. Wie bereits bei den Methodendefinitionen von `Dev_M` ergibt sich auch hier wieder das Problem der Mehrdeutigkeit. Beim Lesen der Multiplexadresse müssen wir explizit sagen, ob wir die Adresse der Spannung oder des Stroms meinen. Die Methode heißt beide Male `get_Mux` (**Listing 11**). Durch Angabe des Klassenbezugs lösen wir dieses Problem.

Wesentlich übersichtlicher und letztendlich einfacher ist es natürlich, in den Unterklassen `Volt_M` und `Amp_M` eigene Methoden vorzusehen (**Listing 12**). Im Beispiel sind dies die Methoden `get_VMux` beziehungsweise `get_AMux`. Diese werden beim Auslesen der Attribute von `My_Dev_2` benutzt.

Mit der Funktion `sizeof` lässt sich die Anzahl der Bytes der Datenfelder unserer Objekte ermitteln. Die Basisklasse reserviert 11 Bytes, die daraus abgeleiteten Klassen 17 Bytes. Da die Basisklasse in jeder

abgeleiteten Klasse enthalten ist, beanspruchen die abgeleiteten Klassen selbst jeweils 6 Bytes. `Dev_M` erbt zweimal die 17 Bytes und benötigt selbst 22 Bytes, summa summarum also 56 Bytes (**Bild 4**).

Ein Schmankerl für Arduino-Fans

Der eine oder andere Arduino-Anhänger unter uns wird sich schon öfters gewünscht haben, eine der Arduino-Methoden in seinem Programm integrieren zu können. Hat man zum Beispiel eine eigene Displaybaugruppe entwickelt, wäre es natürlich toll, wenn man die wirklich leistungsfähige `print`-Methode nutzen könnte. Wie sicher bereits vermutet, geht das mit Vererbung der `Print`-Klasse. *Lektion_7* liefert eine Gebrauchsanweisung dazu. Unsere Klasse `My_Print_C` erbt alle Methoden von `Print` und überschreibt die originale `write`-Methode, die zur Datenübergabe an die spezifische Hardware gedacht ist. Die `write`-Methode erhält von der `print(ln)`-Methode ein einzelnes Zeichen als Argument, das sie dann über die Hardwareschnittstelle, etwa den I²C-Bus, an das Display übergibt. Da unser Beispiel ohne Hardware auskommt, schreiben wir die übergebenen Zeichen in einen String, dessen Inhalt wir im seriellen Monitor sehen können.

Weil die `print(ln)`-Methoden überladen sind, spielt es beinahe keine Rolle, von welchem Typ die Werte sind, die wir „printen“ können. Das war es schon. Ab jetzt kann unser Eigenbaudisplay mit `print` angesprochen werden.

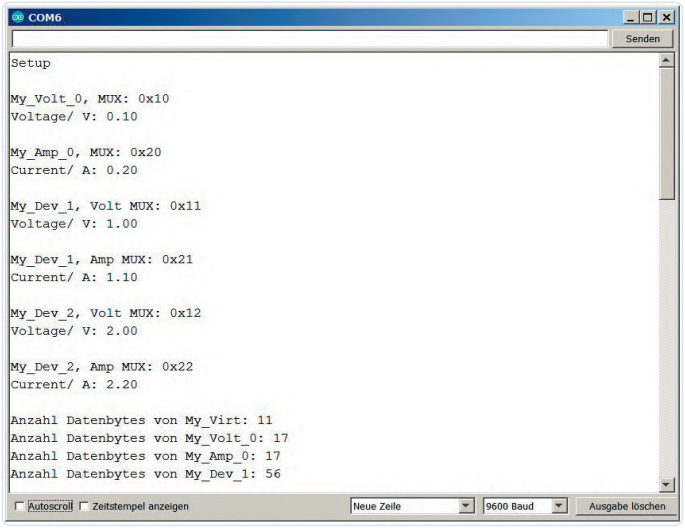


Bild 4. Ausgabe der Setup-Routine in Lektion_6.

Fazit

Sicherlich vermittelt dieser Artikel eine gute Grundlage, um sich mit der objektorientierten Programmierung weiter zu beschäftigen. Wir können fremde Klassenbibliotheken verstehen und in unsere Anwendungen einbinden. Wie das letzte Beispiel gezeigt hat, stellt auch das Erben und Modifizieren von fremden Klassen kein Problem mehr dar. Auch wenn es noch viel zu lernen gibt, die Basis, um selber leistungsfähige objektorientierte Programme zu schreiben, ist damit vorhanden.

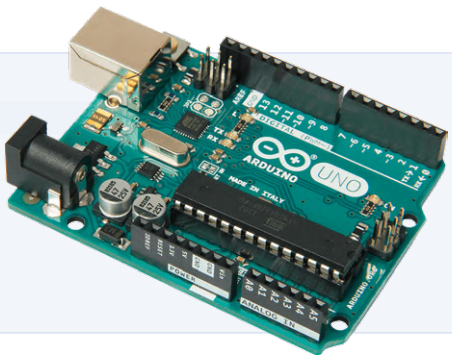
Ich hoffe, dass die Vorteile der OOP gegenüber der prozeduralen Programmierung etwas klarer geworden sind. Es ist die Denkweise, die so anders ist und den Programmierern, die schon lange die prozedurale Programmierung einsetzen, Probleme macht. Aber nur Mut, wie immer gilt auch hier: Übung macht den Meister!

200563-02

PASSENDE PRODUKTE

› **Arduino Uno R3**
www.elektor.de/arduino-uno-r3

› **Elektor Arduino Elektronik Bundle**
www.elektor.de/elektor-arduino-elektronik-bundle



Ein Beitrag von
Idee und Text: **Roland Stiglmayr**
Redaktion: **Rolf Gerstendorf**
Layout: **Giel Dols**

Sie haben Fragen oder Kommentare?
Haben Sie Fragen oder Kommentare zu diesem Artikel? Schicken Sie eine E-Mail an den Autor unter 1134-715@online.de oder kontaktieren Sie Elektor unter editor@elektor.com.

WEBLINKS

[1] **Elektor-Projektseite**: www.elektormagazine.de/200563-02
[2] **C++-Programmierung**: https://de.wikibooks.org/wiki/C%2B%2B-Programmierung
[3] **CPP-Vorlesungsskript (Urheber: Prof. Diepold)**: www.ei.tum.de/fileadmin/tueifei/ldv/Vorlesungen/cpp/CPP-Skript.pdf

Zutritt für Unbefugte verboten!

Ein analoger Synthesizer aus dem Home Lab von Kurt Diedrich

Von **Kurt Diedrich** und **Eric Bogers** (Elektor)

Um 1964 erschien der erste Moog-Synthesizer [1] - ein völlig analoges, modulares elektronisches Instrument, das mit seinen Klangmöglichkeiten die Musikwelt revolutionierte. Das Modell wurde in verschiedenen Variationen bis 1980 hergestellt. Dieser Synthesizer war eine Inspiration für viele Musikprojekte - unter anderem für den Elektor Formant von 1977, der sich bis heute großer Beliebtheit erfreut [2]. Das Heimlabor von Kurt Diedrich, das wir in dieser Folge besuchen, ist auch die Geburtsstätte eines Moog-ähnlichen Synthesizers; ein Projekt, das ständig weiterentwickelt wird und sich hervorragend zum Selbermachen eignet, nicht zuletzt, weil der Entwickler nur gut erhältliche und erschwingliche Komponenten verwendet.

Bild 1. Kurt Diedrichs Elektronik-Labor ...

Bild 2. ...in welchem er grade an einer neuen Version seines Synthesizers arbeitet.

Der Name Kurt Diedrich mag den etwas älteren Lesern ein Begriff sein – nicht ohne Grund. Zunächst lassen wir Kurt selbst zu Wort kommen:

„Schon während meines Geologie-Studiums in den Siebziger-Jahren stellte ich fest, dass ich Elektronik noch interessanter fand als Geologie. So baute ich 1973 meinen ersten, noch relativ primitiven Synthesizer (und viele weitere sollten folgen). Meine ersten Kontakte mit Elektor stammen aus der gleichen Zeit. Mein erstes Buch (über Synthesizer) erschien 1981, herausgegeben vom Frech Verlag [3].
Nach meinem Studium arbeitete ich zunächst einige Zeit als Assistent an der Universität des Saarlandes, bis mir 1981 eine Stelle bei Elektor im niederländischen Beek als Redakteur und Entwickler angeboten wurde. Dies war der Beginn einer wunderbaren Zeit mit vielen tollen Kollegen. Dort entwickelte ich unter anderem einen Kompakt-Synthesizer mit Curtis ICs (CEM), sowie eine große Anzahl von Schaltungen im Bereich Messtechnik, elektronische Spiele und Musikelektronik.
Ab 1985 arbeitete ich auch als Entwickler und Autor für Elex - eine Zeitschrift für angehende Elektronik-Enthusiasten, bis ich 1987 leider zu den zahlreichen Mitarbeitern gehörte, die aufgrund vorübergehender wirtschaftlicher Schwierigkeiten des Verlags ihren Dienst quittieren mussten. Danach arbeitete ich als technischer Handbuch-Autor für verschiedene Firmen, die letzten 17 Jahre bis zum Erreichen des Rentenalters bei *HEAD acoustics* in Herzogenrath bei Aachen.
Im Anschluss daran fragte ich mich, wie ich nun all die freie Zeit sinnvoll und angenehm verbringen könnte. Und dann erinnerte ich mich daran, dass ich früher Synthesizer gebaut habe, und fing 2014 wieder damit an. Mein Ziel ist es, gut funktionierende (sprich: gut klingende) Instrumente mit gewöhnlichen, leicht verfügbaren und preisgünstigen Teilen zu entwickeln. Ich denke, dass mir das recht gut gelungen ist.“

Kurt hat das Glück, dass er über mehrere Arbeitsplätze verfügt. ***Bild 1*** *zeigt sein gut organisiertes und geräumiges Elektroniklabor. In* **Bild 2** *arbeitet Kurt an seinem neuesten Modell.* ***Bild 3*** *zeigt das*

Büro mit Computer und Bibliothek, und ***Bild 4*** *die Werkstatt für die eher groben, mechanischen Arbeiten. Auf so viel Platz könnte man direkt neidisch sein.*
*Es würde den Rahmen dieser Artikelserie überschreiten, zu sehr ins Detail zu gehen. Wir werden uns daher mit einer kurzen (und daher leider unvollständigen) Beschreibung des Blockdiagramms der aktuellen Version (****Bild 5****) begnügen, wohlwissend, dass sich dieses Projekt in ständiger Entwicklung befindet.*

„Das Diagramm zeigt das Blockschaltbild des Synthesizers. Jedes grüne Rechteck stellt eine Platine dar. Jede Platine enthält wiederum ein einzelnes Modul, mit Ausnahme der Dreifach-VCA-Platine, die auch die Schaltungen für den Rauschgenerator und den Mischer enthält (weil diese sehr wenig Platz beanspruchen).
Ich habe gerade ein Laptop-Schaltnetzteil mit einer Ausgangsspannung von 19 VDC mit nachgeschaltetem DC/DC-Wandler (19 V > 2 x 12 V) getestet. Es hat sich wegen der störenden HF-Komponenten jedoch leider als ungeeignet herausgestellt, so dass ich weiterhin ein konventionelles Trafo-Netzteil von 2 x 12 V und 2 x 500 mA verwende.
Der mit einem Keyboard verbundene MIDI-Konverter wird von einem *Arduino Nano* zum Leben erweckt. Diese Schaltung stammt nicht von mir, aber ich habe aber die Platine dafür entworfen.
Der Sound wird durch zwei kontinuierlich weiterentwickelte Rechteck-Dreieck-Sägezahn-VCOs erzeugt, denen, wie allgemein üblich, ein spannungsgesteuerter Verstärker und ein spannungsgesteuertes Filter nachgeschaltet wurde, die beide über individuell einstellbare ADSR-Hüllkurvengeneratoren (ebenfalls aus eigener Entwicklung) verfügen. Durch allerlei Kniffe ist es mir gelungen, den Signal-Rausch- bzw. Störabstand auf hohem Niveau zu halten.
Natürlich darf auch ein LFO, der für allerlei zusätzliche Modulationseffekte sorgt, nicht fehlen. Eine spezielle Retrigger-Unit sorgt für ein reibungsloses Auslösen der Hüllkurvengeneratoren, wenn eine neue Taste vor dem Loslassen der vorherigen Taste gedrückt wurde, was sehr häufig vorkommt.
Apropos Taste: Zum virtuosen Spiel wird natürlich eine Tastatur benötigt, der einzige Teil des Projektes, den man fertig kaufen muss. Der Synthesizer ist kompatibel zu einem Standard-MIDI-Keyboard. Der Markt bietet eine große Auswahl an günstigen Exemplaren. Ich verwende das Modell ‚Easy Key 49' der Firma SwissSonic.
Wer an einer vollständigen Beschreibung inklusive Schaltplänen und dergleichen interessiert ist, kann sich direkt an mich wenden (siehe Textbox)."

200681-03

Bild 3. Der Computerraum

Bild 4. Die Werkstatt für gröbere, mechanische Arbeiten

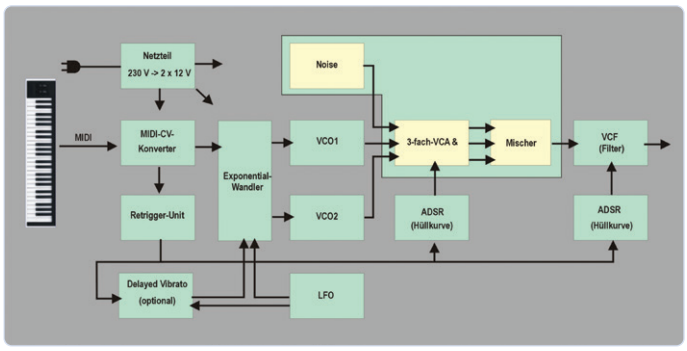


Bild 5. Aktuelles Blockschaltbild des Synthesizers

Haben Sie Fragen oder Anmerkungen?
Haben Sie Fragen oder Anmerkungen zu diesem Artikel? Dann schicken Sie ein E-Mail an den Autor über subroutine-sy@gmx.de oder Elektor über editor@elektor.com.

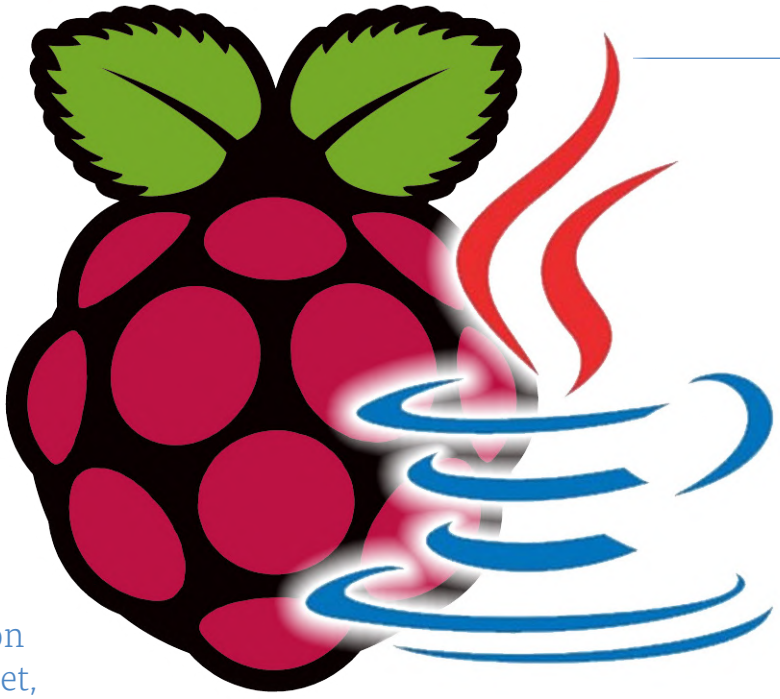
Ein Beitrag von:
Text, Fotos und Blockschaltbild: **Kurt Diedrich**
Redaktion: **Eric Bogers**
Layout: **Giel Dols**

WEBLINKS

1] **Moog-Synthesizer**: https://de.wikipedia.org/wiki/Moog-Synthesizer
[2] **Jan Buiting, „Formant Synthesizer", Elektor 4/2008**: www.elektormagazine.de/magazine/elektor-200804/18762
[3] **Bücher von Kurt Diedrich**: www.subroutine.info/bücher/

Java auf dem Raspberry Pi

Teil 1: GPIOs



Von **Frank Delporte** (Belgien)

Obwohl es die Programmiersprache Java schon lange gibt, ist sie immer noch sehr gut geeignet, um Code für moderne Computerplattformen wie den Raspberry Pi zu entwickeln. Im ersten Teil dieser kleinen Artikelreihe lernen wir nach einigen Hintergrundinformationen zu dieser Programmiersprache schon die Fähigkeiten von Java kennen, die GPIO-Pins des Raspberry Pi zu steuern und zu lesen.

Kurze Einführung in Java

Java ist eine der Programmiersprachen, die es schon sehr lange gibt. Die erste Version wurde 1995 veröffentlicht [1], im selben Jahr, in dem auch JavaScript, PHP und Qt geboren wurden. Python ist ein wenig älter, die erste Version wurde 1991 veröffentlicht. Während die Marke Java von Sun Microsystems zu Oracle übergegangen ist, wird die Entwicklung der Programmiersprache als Open-Source-Projekt betrieben, das 2020 zu GitHub umgezogen ist [2]. Seit 2018 werden jährlich zwei neue Java-Releases zur Verfügung gestellt, was dazu führt, dass Fehlerkorrekturen und neu entwickelte Funktionen regelmäßig auf die Rechner gelangen. Zu den neuen Funktionen gehören auch Verbesserungen, die auf eingebettete Plattformen wie den Raspberry Pi abzielen.

Es gibt die Ansicht, dass „Java tot ist“, aber die zahlreichen Konferenzen - auch wenn sie in Zeiten von Corona alle virtuell geworden sind - und viele Java-bezogenen Projekte sind ein Beweis dafür, dass die Sprache immer noch quicklebendig ist. Für die Weiterentwicklung von Java ist eine Vielzahl namhafter Unternehmen verantwortlich, darunter Oracle, Microsoft, SAP und Google [3]. Es wird auch gemunkelt, dass die Hälfte der Microsoft Azure Cloud auf Java läuft! Heute gibt es Java-Distributionen von verschiedenen Open-Source- (jdk.java.net, adoptopenjdk.net) und kommerziellen Anbietern (Azul, BellSoft, Oracle und andere).

Java auf dem Raspberry Pi?!

Wurde der Raspberry Pi nicht dafür entwickelt, um mit Python programmiert zu werden? Das mag ja so sein, aber das bedeutet nicht, dass man nicht auch andere Sprachen verwenden kann. Und, obwohl dies ein Java-Artikel ist, wollen wir nicht behaupten, dass diese Sprache gegenüber Python, C oder einer anderen Sprache vorzuziehen ist! Jedes Projekt hat seine eigenen Anforderungen, für die eine bestimmte Sprache perfekt geeignet sein könnte, oder

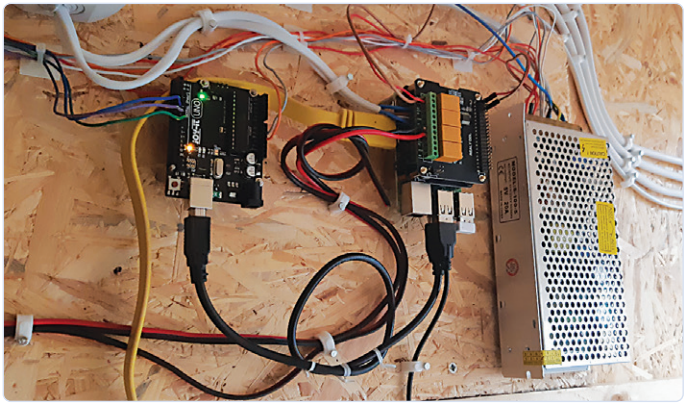


Bild 1. Der Kern des Touchscreen-Drum-Booth-Controllers.

Bild 2. Touchscreen-Oberfläche zur Steuerung von Relais und LED-Streifen.

in der das Entwicklerteam die besten Kenntnisse hat.
Als mein erster Versuch, ein Python-basiertes Pong-Spiel zu bauen, erfolglos endete und sich die Benutzeroberfläche als enttäuschend erwies, wollte ich als Java-Entwickler erkunden, ob ich mein Fachwissen nicht vielleicht auf den Raspberry PI anwenden könnte. Glücklicherweise lassen sich Benutzeroberflächen mit JavaFX erzeugen, einem unabhängigen Projekt [4, 5], das Java um ein Framework zur Erstellung von GUIs (Graphical User Interface) erweitert. Es bietet alle typischen Basiskomponenten (Schaltflächen, Labels, Tabellen, Diagramme) und es gibt eine Vielzahl von freien und Open-Source-Bibliotheken, die diese Liste noch erweitert. Hier fand ich auch die perfekte Kombination von Java mit dem Raspberry Pi, da ich eine Touchscreen-Oberfläche für das Drum-Booth meines Sohnes bauen wollte (**Bild 1** und **Bild 2**). Kombiniert mit einer Relaisplatine, einem Arduino und einigen LED-Streifen sollte dies die Basis für meine Experimente in der neuen Welt der Embedded-Java-Programmierung sein.

Vorbereitungen

Für die Experimente, die in diesem Artikel beschrieben werden, benötigen Sie einen aktuellen Raspberry Pi mit einem ARMv7- oder ARMv8-Prozessor. Die älteren Boards mit einem ARMv6 haben ein anderes internes Layout, für das Standard-Java nicht funktioniert. Wenn Sie Java auf einem solchen ARMv6-Board verwenden wollen, gibt es eine Lösung, die das Azul Zulu JDK [6] verwendet. Sie müssen keine in der Artikelreihe verwendeten Code-Beispiele abtippen, sie sind in einem GitHub-Repository [7] verfügbar. Beginnen Sie mit der Vorbereitung der SD-Karte für den Raspberry Pi. Seit einiger Zeit gibt es ein praktisches „Imager"-Tool [8], in dem Sie „Raspberry Pi OS Full (32-bit)" wählen (**Bild 3**). Sobald die SD-Karte fertig ist und Sie Ihren Raspberry Pi gestartet haben, geben Sie am Terminal `java -version` ein, um zu überprüfen, ob es die Version 11 ist. Wie Sie sehen werden, ist OpenJDK in dieser Vollversion des Betriebssystems vorinstalliert:

```
$ java -version
openjdk version "11.0.9" 2020-10-20
OpenJDK Runtime Environment (build
   11.0.9+11-post-Raspbian-1deb10u1)
OpenJDK Server VM (build 11.0.9+11-post-Raspbian-
   1deb10u1, mixed mode)
```

Visual-Studio-Code

Sie können Ihren Java-Code auf einem PC entwickeln, testen und ausführen, bevor Sie ihn nach Fertigstellung auf den Raspberry Pi übertragen. Aber es gibt noch einen anderen Ansatz: Visual Studio Code [9]. Diese kostenlose IDE (Integrated Development Environment) von Microsoft hat eine lange Liste von Erweiterungen, die sie zum perfekten Begleiter für jedes Programmierprojekt machen. Es gibt eine Version für 32-Bit-ARM-Systeme (zum Beispiel das Raspberry-Pi-OS) und 64-Bit-ARM-Systeme (das kommende Raspberry-Pi-OS, das sich noch in der Entwicklung befindet) oder auch Ubuntu Desktop [10].
Es gibt auch ein „Java Extension Pack", das der IDE mehrere Java-Erweiterungen hinzufügt, um sie zu einer vollständigen IDE für Java-Entwickler zu machen (**Bild 4**)!

Experimentieren mit Hello World!

Lassen Sie uns unser erstes Java-Programm auf dem Raspberry Pi ausprobieren. Schreiben Sie in Visual Studio Code, in einem Texteditor oder im Terminal eine neue Textdatei namens „HelloWorld.java" mit folgendem Inhalt:

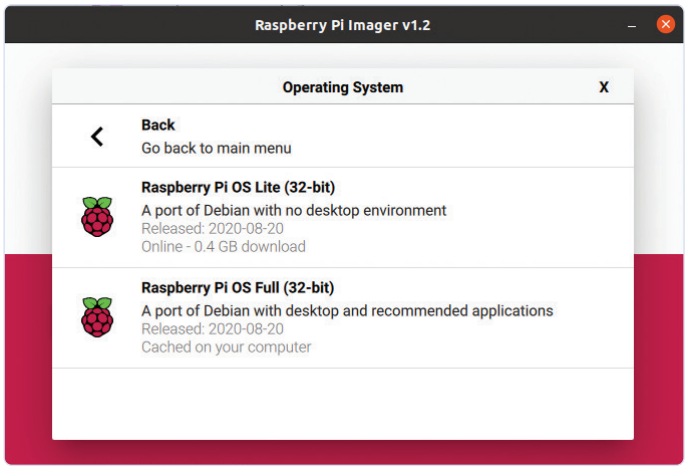


Bild 3. Das neue Raspberry Pi Imager Tool - der optimale Weg zur SD-Karte.

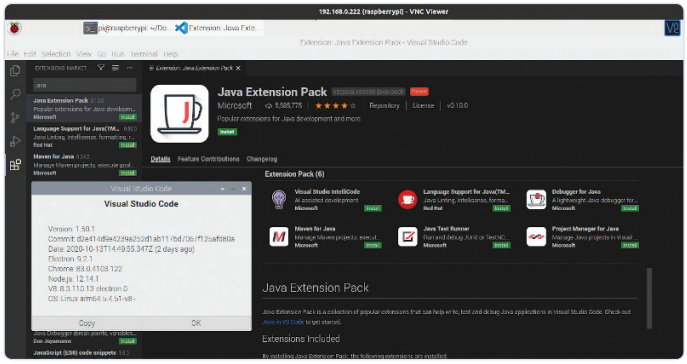


Bild 4. Visual Studio Code mit dem Java-Extension-Pack kann auf dem Raspberry Pi verwendet werden.

```
public class HelloWorld {
  public static void main(String args[]) {
    System.out.println("Hello World!");
  }
}
```

Unser gesamter Code ist Teil der Klasse `HelloWorld`, die per Konvention gleich dem Dateinamen ist. Ein Java-Programm beginnt mit der Methode `public static void main(String args[])`. Das einzige, was wir hier tun müssen, ist „Hello World!" als Ausgabe des Programms anzugeben.
Mit Java 11 können wir dieses einfache Programm ohne Kompilierung des Codes ausführen. Geben Sie im Terminal im Verzeichniss der Java-Datei den Befehl `java HelloWorld.java` ein, um das folgende Ergebnis zu erhalten

```
pi@raspberrypi:~/elektor $ java HelloWorld.java
Hello World!
```

Listing 1. Der Code CPUtemp.java zum Auslesen der Prozessortemperatur des Raspberry Pi [7]

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.util.List;
import java.util.ArrayList;

public class CPUTemp {
    private static final String FILE = “/sys/class/thermal/thermal_zone0/temp”;
    private static final List<Integer> values = new ArrayList<>();

    public static void main(String[] args) throws InterruptedException {
        while(true) {
            checkTemp();
            Thread.sleep(1000);
        }
    }

    private static void checkTemp() {
        try (BufferedReader br = new BufferedReader(new FileReader(FILE))) {
            int value = Integer.valueOf(br.readLine());
            values.add(value);
            int total = values.stream().mapToInt(Integer::valueOf).sum();
            System.out.println(“Now: “ + value
                + “ - Average: “ + (total / values.size())
                + “ - Number of measurements: “ + values.size());
        } catch (Exception ex) {
            System.err.println(“Error during temperature check: “
                + ex.getMessage());
        }
    }
}
```

Sicherlich ist `print()` in Python kürzer im Vergleich zu `System.out.println()` in Java, aber wir verzeihen Java dies als Jugendsünde.

Auslesen der CPU-Temperatur

Auf viele der Systemwerte, Ein- und Ausgaben kann über das Verzeichnis `/sys/` des Linux-Dateisystems zugegriffen werden. Die dort gespeicherten Werte können einfach als Textdatei ausgelesen werden. Versuchen wir einmal, die Temperatur des Raspberry-Pi-Prozessors aus einer der Dateien in diesem `sys`-Verzeichnis zu ermitteln. Die Namensgebung in der Verzeichnisstruktur ist nicht immer sehr eindeutig und auch die Interpretation der Werte ist nicht immer einfach, aber es ist ein guter Ausgangspunkt, um auch ein wenig das Linux-System kennenzulernen. Führen Sie den folgenden Befehl aus:

```
$ ls -l /sys/
total 0
drwxr-xr-x   2 root root 0 Dec  2 15:44 block
drwxr-xr-x  29 root root 0 Feb 14  2019 bus
drwxr-xr-x  64 root root 0 Feb 14  2019 class
drwxr-xr-x   4 root root 0 Feb 14  2019 dev
drwxr-xr-x  10 root root 0 Feb 14  2019 devices
drwxr-xr-x   3 root root 0 Feb 14  2019 firmware
```

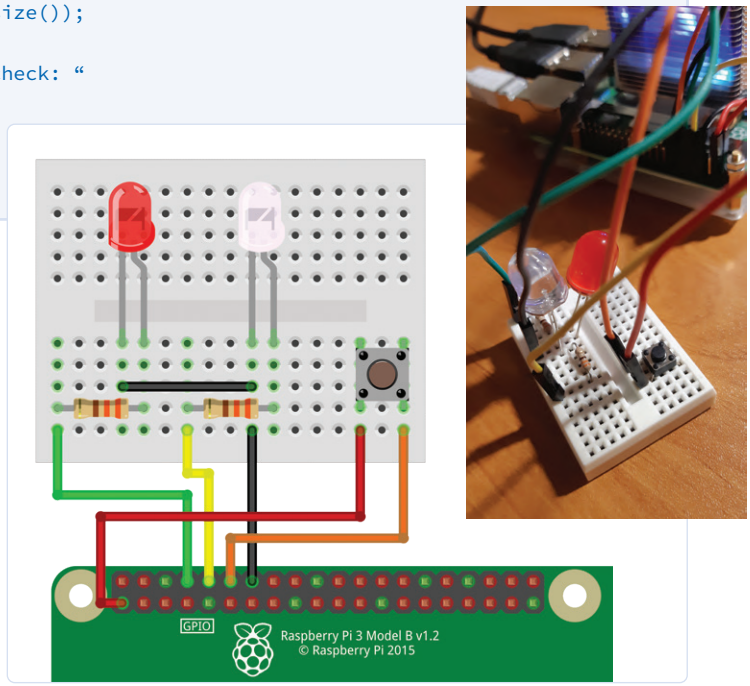


Bild 5. Layout für LEDs und Taster in Fritzing und in natura.

Haben Sie Fragen oder Kommentare?
Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Dann kontaktieren Sie den Autor direkt über javaonraspberrypi@webtechie.be oder die Redaktion über editor@elektor.com.

```
drwxr-xr-x   8 root root 0 Jan  1  1970 fs
drwxr-xr-x  12 root root 0 Jan  1  1970 kernel
drwxr-xr-x 122 root root 0 Feb 14  2019 module
drwxr-xr-x   2 root root 0 Dec 15 11:39 power
```

Es scheint, dass wir hier eine Menge Informationen finden können! Betrachten einige davon etwas genauer:

```
$ cat /sys/firmware/devicetree/base/cpus/cpu@0/
   compatible
arm,cortex-a72
$ cat /sys/class/net/wlan0/address
dc:a6:32:c5:b7:9d
$ cat /sys/class/bluetooth/hci0/uevent
DEVTYPE=host
$ cat /sys/class/thermal/thermal_zone0/temp
30667
```

Die letzte Angabe sieht tatsächlich wie eine Temperatur in Grad Celsius aus, wenn wir den Wert durch 1000 teilen würden! Lassen Sie uns ein einfaches Programm schreiben, das diesen Wert jede Sekunde ausliest und die Durchschnittstemperatur berechnet.
Der Code in **Listing 1** verwendet einige zusätzliche Java-Methoden, die zu Beginn importiert werden. Die `io`-Methoden lesen eine `sys`-Datei als Textdatei. `List` und `ArrayList` erzeugen eine Liste aller Messwerte. In der `main()`-Methode rufen wir eine separate Methode `checkTemp()` auf, die die Temperatur erfasst und in der Liste speichert. Es ist gute Praxis, jede Funktionalität in einer eigenen Methode zu isolieren, um den Code sauber zu halten.
`Thread.sleep(1000)` sorgt dafür, dass zwischen jeder Temperaturermittlung eine Sekunde verstreicht. Innerhalb der Methode wird der Durchschnittswert berechnet, indem die Gesamtsumme der Werteliste mit Hilfe eines Streams berechnet wird. Streams wurden in Java 8 eingeführt und sind eine sehr leistungsfähige Möglichkeit, mit Listen von Elementen zu arbeiten. Auch hier ist der Dateiname auch der Name der Klasse „CPUTemp.java“.
Wie zuvor ist dies eine einfache Java-Klasse ohne zusätzliche Abhängigkeiten, so dass wir sie ohne Kompilierung ausführen können. Die Ausgabe kann mit „Strg+c“ angehalten werden:

```
pi@raspberrypi:~/elektor $ java CPUTemp.java
Now: 36998 - Average: 36998 - Number of measurements: 1
Now: 34563 - Average: 35780 - Number of measurements: 2
Now: 35537 - Average: 35699 - Number of measurements: 3
Now: 36024 - Average: 35780 - Number of measurements: 4
Now: 35537 - Average: 35731 - Number of measurements: 5
```

Ansteuern einer LED und Auslesen eines Tasters

Lassen Sie uns eine weitere Single-File-Java-Anwendung erstellen, die zwei LEDs umschaltet und den Zustand eines Tasters auslesen kann. Der Schaltung, um zwei LEDs und einen Taster an separate GPIOs anzuschließen, ist recht einfach. Der Taster benötigt 3,3 V, nicht 5,0 V, verwenden Sie also den richtigen Power-Pin des Raspberry Pi! In diesem Aufbau sind die LEDs an die Pins BCM 14 und BCM 15 angeschlossen, der Taster an BCM 18 (**Bild 5**).

Testen am Terminal

Mit den im Raspberry-Pi-OS integrierten Befehlen können wir vom Terminal aus auf die GPIOs zugreifen. Dies nutzen wir, um unsere Verdrahtung zu testen. Führen Sie die folgenden Befehle aus, um Pin 14 als Ausgangspin (op) zu konfigurieren und ihn auf High (dh) oder Low (dl) zu schalten. Versuchen Sie das Gleiche für die andere LED an BCM 15:

```
$ raspi-gpio set 14 op
$ raspi-gpio set 14 dh
$ raspi-gpio set 14 dl
```

Wir können einen ähnlichen Ansatz verwenden, um die Taste zu testen, indem wir den Pin als Eingangspin konfigurieren und den Zustand des Eingangs abfragen:

```
$ raspi-gpio set 18 ip
$ raspi-gpio get 18
GPIO 18: level=0 fsel=0 func=INPUT pull=DOWN
$ raspi-gpio get 18
GPIO 18: level=1 fsel=0 func=INPUT pull=DOWN
```

Wenn die Taste gedrückt wird, ändert sich der Wert „level“ auf 1.

Mach es in Java!

Wir verwenden den gleichen grundlegenden Ansatz wie in „CPUTemp.java“ und nutzen die Terminalbefehle in einem Java-Programm, indem wir `Runtime.getRuntime().exec(cmd)` angeben. Wenn wir die Benutzereingabeklasse `Scanner` hinzufügen, können wir auch das Ergebnis des Befehls erfassen und es zur Überprüfung des Zustands der Schaltfläche verwenden. Dies wird in **Listing 2** gezeigt.
Der Code nutzt ein `Action`-Enum, um die Verarbeitung des Befehls zu vereinfachen. In seiner einfachsten Form ist ein Enum nur eine Liste von vordefinierten Namen, aber in diesem Beispiel fügen wir jedem Namen einen Wert hinzu. Dies ist ein starker Pluspunkt von Enums, da sie es ermöglichen, den Code viel lesbarer zu machen und ihn meist gleichzeitig auch zu vereinfachen. Wie in der `doAction`-Methode zu sehen ist, wird das `Action`-Enum als Parameter genutzt und sein Wert verwendet, um den auszuführenden Befehl zu generieren.
Die Kommentare im Code sollten alles klar machen, da dies nur einfaches Java ist, um die Verwendung von Eingangs- und Ausgangs-GPIOs zu illustrieren. Der komplexeste Teil ist die `runCommand`-Methode, die einige kombinierte Methoden verwendet, um den Befehl auszuführen, das Ergebnis zu lesen und mögliche Fehler zu behandeln. Falls Ihnen dies im Moment noch nicht klar ist, keine Sorge - das wird es, sobald der Code läuft!
Da dieses Beispiel wiederum keine Abhängigkeiten aufweist, können wir es ohne Kompilierung ausführen. Die Ausgabe zeigt, was vor sich geht, Sie sollten also sehen, wie Ihre LEDs in verschiedenen Intervallen ein- und ausschalten. Wenn die LEDs aufhören zu

PASSENDE PRODUKTE

- **F. Delporte, *Getting Started with Java on the Raspberry Pi***
  www.elektor.de/19292
- **Raspberry Pi 4 Starter Kit**
  www.elektor.de/19427

Listing 2. Code für RaspiGpio.java zur Steuerung von LEDs und zur Erfassung des Tasterzustands [7].

```
import java.io.IOException;
import java.util.Scanner;

public class RaspiGpio {

    private static final String LED_1 = “14”;
    private static final String LED_2 = “15”;
    private static final String BUTTON = “18”;

    private enum Action {
        OUTPUT_PIN(“op”),
        INPUT_PIN(“ip”),
        DIGITAL_HIGH(“dh”),
        DIGITAL_LOW(“dl”);
        private final String action;
        Action(String action) {
            this.action = action;
        }
        String getAction() {
            return action;
        }
    }

    public static void main(String[] args) throws InterruptedException {
        // Configure the two LEDs as output pins
        doAction(LED_1, Action.OUTPUT_PIN);
        doAction(LED_2, Action.OUTPUT_PIN);

        // Configure the button as input pin
        doAction(BUTTON, Action.INPUT_PIN);

        // Blink the LEDs with different interval
        for (int i = 0; i < 20; i++) {
            System.out.println(“Blink loop: “ + i);
            if (i % 2 == 0) {
                doAction(LED_1, Action.DIGITAL_HIGH);
            } else {
                doAction(LED_1, Action.DIGITAL_LOW);
            }
```

blinken, können Sie die Taste drücken, die zehnmal im Abstand von einer Sekunde überprüft wird. Die erwartete Ausgabe ist wie folgt:

```
$ java RaspiGpio.java
Executing: raspi-gpio set 14 op
Executing: raspi-gpio set 15 op
Executing: raspi-gpio set 18 ip
Blink loop: 0
Executing: raspi-gpio set 14 dh
Executing: raspi-gpio set 15 dh
Blink loop: 1
Executing: raspi-gpio set 14 dl
Executing: raspi-gpio set 15 dl
...
Executing: raspi-gpio get 18
Button check 0: GPIO 18: level=0 fsel=0 func=INPUT pull=DOWN
 - PUSHED: false
...
Button check 3: GPIO 18: level=1 fsel=0 func=INPUT pull=DOWN
 - PUSHED: true
...
```

Hausaufgaben für’s nächste Mal!

Mit einigen Grundlagen haben wir eine kleine Einführung in Java auf dem Raspberry Pi gegeben. Sie können gerne mit dem Beispielcode herumspielen und in den angegebenen Links die Hintergründe nachlesen, um Ihr Verständnis weiter zu vertiefen. Dies wird hoffentlich die nächsten Schritte im nächsten Artikel einfacher machen, wenn wir eine vollständige Java-Anwendung entwickeln und unsere GPIOs mit einer Webseite verknüpfen.

200617-02

Ein Beitrag von

Idee, Text und Bilder: **Frank Delporte**
Redaktion: **Stuart Cording**
Übersetzung: **Rolf Gerstendorf**
Layout: **Giel Dols**

```
            if (i % 3 == 0) {
                doAction(LED_2, Action.DIGITAL_HIGH);
            } else {
                doAction(LED_2, Action.DIGITAL_LOW);
            }
            Thread.sleep(500);
        }
        doAction(LED_1, Action.DIGITAL_LOW);
        doAction(LED_2, Action.DIGITAL_LOW);

        // Read the button state 10 times
        for (int j = 0; j < 10; j++) {
            String result = runCommand("raspi-gpio get " + BUTTON);
            System.out.println("Button check " + j
                + ": " + result
                + " - PUSHED: " + (result.contains("level=1")));
            Thread.sleep(1000);
        }
    }

    private static void doAction(String pin, Action action) {
        runCommand("raspi-gpio set " + pin + " " +
              action.getAction().toLowerCase());
    }

    private static String runCommand(String cmd) {
        System.out.println("Executing: " + cmd);
        Scanner s = null;
        try {
            s = new Scanner(Runtime.getRuntime().exec(cmd).getInputStream())
                .useDelimiter("\\A");
            return s.hasNext() ? s.next() : "";
        } catch (Exception ex) {
            System.err.println("Error during command: " + ex.getMessage());
            return "";
        } finally {
            if (s != null) {
                s.close();
            }
        }
    }
}
```

WEBLINKS

[1] **Andrew Binstock, „Java's 20 Years Of Innovation", Forbes**: http://bit.ly/3qeIpVu

[2] **OpenJDK-Project**: https://github.com/openjdk

[3] **Liam Tung, Oracle: „Programming language Java 14 is out with these 16 major feature improvements", ZDNet**: http://zd.net/35wVW2O

[4] **OpenJFX-Project**: https://github.com/openjfx

[5] **OpenJFX**: https://openjfx.io/

[6] **Frank Delporte, „How to install and use Java 11 and JavaFX 11 on Raspberry Pi boards with ARMv6 processor"**: http://bit.ly/35yRrFc

[7] **Begleitende Code-Beispiele auf GitHub**: http://bit.ly/3i9bP4v

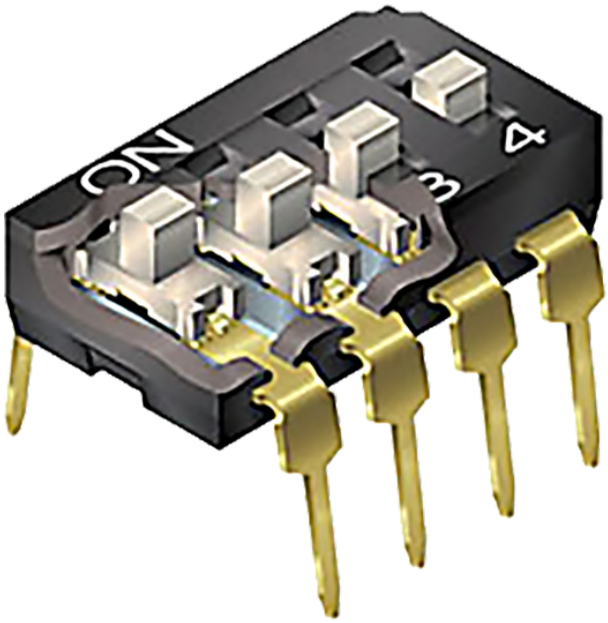
[8] **Raspberry Pi Imager**: http://bit.ly/3i58seU

[9] **Frank Delporte, „Visual Studio Code on the Raspberry Pi (with 32- and 64-Bit-OS)"**: http://bit.ly/3qeh9GN

[10] **Visual Studio Code Downloads für Arm**: http://bit.ly/2LqUng3

DIP-Schalter

Bemerkenswerte Bauteile



Von **David Ashton** (Australien)

Moment mal! DIP-Schalter - wie können die bemerkenswert sein? Schließlich sind es doch nur simple Schalter! Nun, es gibt tatsächlich einige sehr bemerkenswerte DIP-Schalter. Hier stelle ich einige davon aus meiner Sammlung vor, in aufsteigender Reihenfolge ihrer Merkwürdigkeit.

DIP, für die Uneingeweihten, bedeutet Dual Inline Package, was darauf hinweist, dass die Pins in zwei parallelen Reihen angeordnet sind, genau wie bei ICs. Die meisten haben den üblichen Reihenabstand von 0,3 Zoll (wie ICs), aber einige nur 0,2 Zoll. DIP-Schalter werden zum Einstellen von Optionen, Funktionen, Adressen oder anderen Dingen verwendet, die eine robuste, selten genutzte Konfigurationsmöglichkeit erfordern. Die meisten DIP-Schalter sind achtpolig, aber es gibt viele verschiedene Typen, wie wir noch sehen werden.

Als erstes wollen wir uns die Standard-DIP-Schalter ansehen, wie wir sie alle kennen. Ein Gruppenbild einiger DIP-Schalter meiner Sammlung ist in **Bild 1** zu sehen. Es gibt DIP-Schalter in vielen Farben und sie beleben dadurch ansonsten langweilige Platinen. Einige haben versenkte Schaltwippen, was bedeutet, dass man

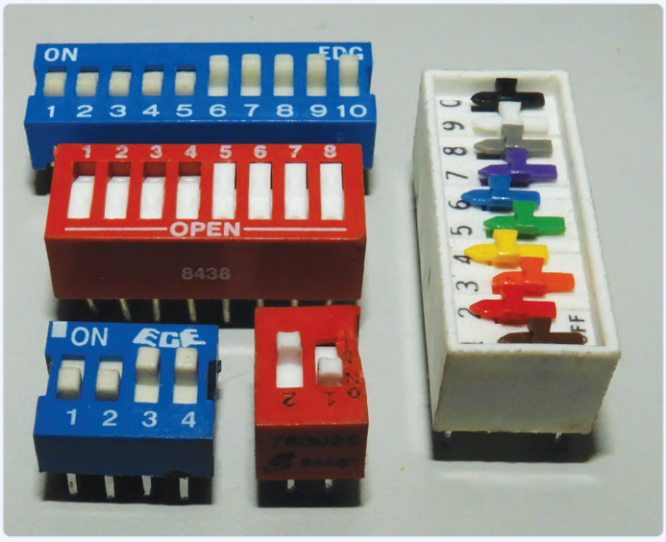


Bild 1. Eine Sammlung typischer DIP-Schalter.

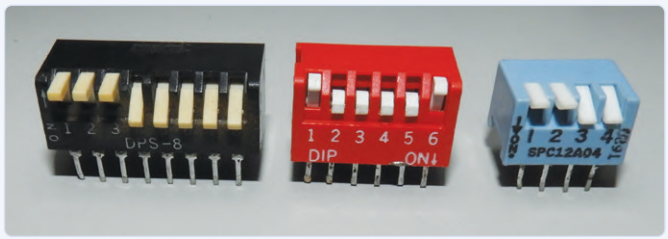


Bild 2. DIP-Schalter im Klavierstil lassen sich leicht einstellen.

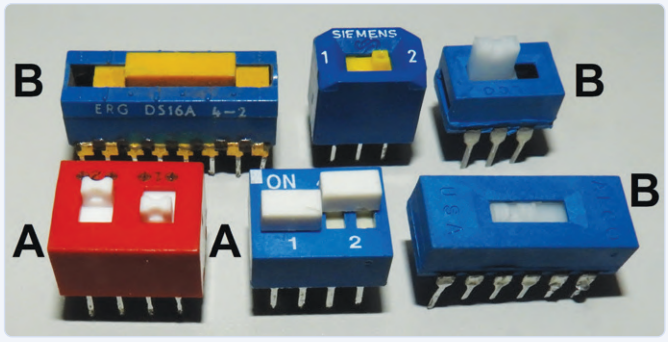


Bild 3. Duale und Umschalter.

nach einer Büroklammer oder (schlechter) einem Kuli oder (ein absolutes No-No) einem Bleistift suchen muss. Bei anderen steht der Aktuator aus dem Gehäuse heraus, so dass man ihn im Notfall mit dem Fingernagel umlegen kann. Einige, wie der rechts abgebildete, haben farbcodierte Aktuatoren. Wenn Sie den Farbcode von Widerständen im Kopf haben, sind solche Schalter leicht zu lesen und einzustellen.

Bei einigen Modellen befinden sich die Schalter an der Seite (**Bild 2**). Dies ist nützlich, um sie an Rückwänden von Geräten zugänglich zu machen. Schaut man sich solche Schalter an, braucht man sich nicht über den Spitznamen „Mäuseklavier" wundern. Die Klaviertasten machen es jedenfalls einfach, die Einstellungen mit dem Fingernagel zu ändern.

Jetzt wird es ein wenig undurchsichtiger. Es gibt duale Typen (**Bild 3a**), bei denen zwei unabhängige Schalter auf einmal betätigt werden. Es gibt auch DIP-Umschalter (**Bild 3b**). Die kleineren besitzen zwei Umschalter (das heißt, DPDT) und die größeren haben vier oder mehr Umschalter. Das macht sie zum Beispiel für das Ein- und Ausschalten von Dämpfungsgliedern interessant.

Ich habe auch einige Wahlschalter (**Bild 4**), die eines von acht oder zehn Kontaktpaaren auswählen. Die blauen Schalter links haben eine Abdeckung (die beim unteren Schalter entfernt wurde) und benötigen einen kleinen Schraubendreher, um den Betätiger in eine von zehn Positionen zu schieben. Mit einem solchen Schalter könnten Sie zum Beispiel einen Ausgang eines CD4017-Zähler-ICs für einen Synthesizer oder eine Alarmanlage anwählen.

Es gibt auch dezimale und hexadezimale DIP-Schalter (**Bild 5**), die verwendet werden, um einen Binär- oder BCD-Code voreinzustellen, oft für eine Adresse auf einer Platine. Das ist viel einfacher, als eine Zahl im Kopf in BCD oder Hex umzuwandeln und dann auf einem Schalter wie im Bild 1 einzustellen. Diese Schalter sind sehr kompakt und haben normalerweise einen Common- sowie vier binär- oder BCD-codierte Kontakte. Die Schalter gibt es in horizontaler, vertikaler oder Pultform.

In **Bild 6a** ist ein einpoliger, einseitiger Schalter zu sehen. Ich habe gezögert, ihn mit in den Artikel aufzunehmen, da er ziemlich groß ist und keinen Standardabmessungen entspricht, aber solche Schalter sind besser als Jumper, die nie da sind, wenn man sie braucht!

Dann gibt es noch die in **Bild 6b** gezeigten Typen. Sie besitzen federnde Drahtverbindungen, anstatt echte Schalter zu sein, aber sie erfüllen die gleiche Aufgabe und sind einfach zu bedienen. Außerdem können sie wahrscheinlich etwas mehr Strom übertragen als die winzigen „normalen" DIP-Schalter.

Und schließlich ist da noch das Monster in **Bild 7**. 16 Schalter mit jeweils drei Positionen (verbunden mit Kontakt 1, aus, verbunden mit Kontakt 2). Einen solchen Schalter können Sie verwenden, um die Bit eines HEX-Bytes als 1, 0 oder „don't care" zu definieren. Vielleicht könnte er in einem Logik-Analysator von Nutzen sein? Fernbedienungen verwenden oft diese Art von Schaltern. Mit drei Positionen für jeden Schalter erhöht sich die Anzahl der möglichen Codes beträchtlich - 38 (6561) gegenüber 28 (256) für einen 8-Wege-Schalter.

200716-02

Haben Sie Fragen oder Kommentare?
Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Senden Sie eine E-Mail an Elektor unter editor@elektor.com.

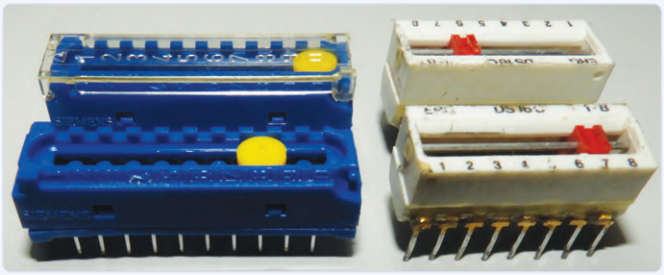


Bild 4. DIP-Schalter im „1-aus-10"-Drehschalter-Stil.

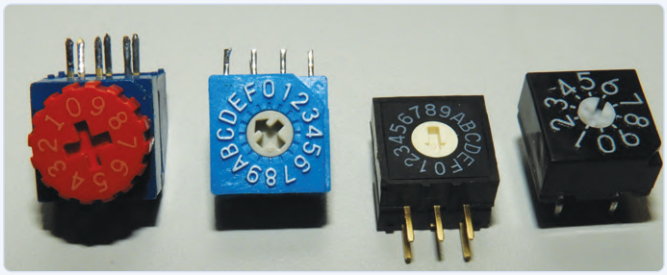


Bild 5. Hex- und BCD-Schalter.

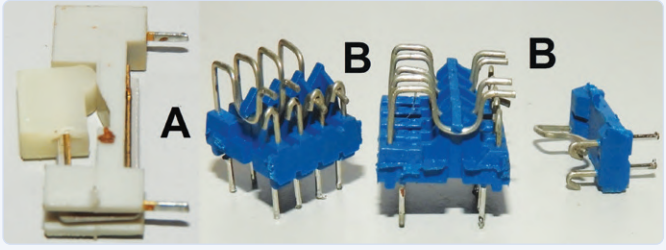


Bild 6. Robuste einfache und federnde Drahtverbindungen.

Bild 7. Ein dreifacher „0/1/Don't Care"-DIP-Schalter.

Projekt 2.0

Korrekturen, Updates und Leserbriefe

Redaktion: **Jens Nickel**

Einfacher Funktionsgenerator

Heft 07-08/2020, S. 20 (160548)

Im Schaltbild ist uns ein Fehler unterlaufen: P3 muss zwischen Pin 2 und Pin 6 von IC4 geschaltet sein, nicht zwischen Pin 2 und Pin 7. IC4 ist als Puffer für den Open-Kollektor-Ausgang des LM311 gedacht und P3 dient der Einstellung der Verstärkung (Amplitude).

Elektor Uno R4

Heft 07-08/2016, S. 98 (150790)

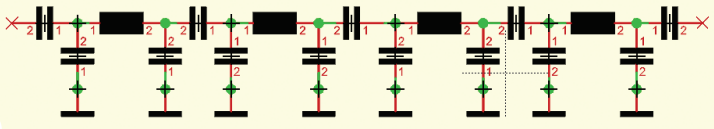
Der ATmega328PB wird nun vollständig von der Standard-AVR-Toolchain der Arduino-IDE unterstützt. So ist die Installation des Elektor-Uno-R4 viel einfacher und schneller, da nicht mehr eine spezielle Toolchain heruntergeladen werden muss. Es macht das Board auch zukunftssicher, da neue Toolchain-Funktionen automatisch verwendet werden.
Um den Elektor-Uno-R4 zu installieren, fügen Sie in der IDE den Link
https://github.com/ElektorLabs/Arduino/releases/download/v1.0.1/package_elektor_uno_r4_1_8_x_index.json unter *Datei →Voreinstellungen* in das Feld *Zusätzliche Boardverwalter-URLs* ein. Im *Boardverwalter* (Arduino-IDE-Menü *Werkzeuge → Board → Boardverwalter*) filtern Sie auf „R4“ und stellen Sie sicher, dass die Version auf „2.0.0“ gesetzt ist. Dies wurde mit der Arduino-IDE 1.8.13 getestet.

Design analoger Filter (3)

Heft 01-02/2021, S. 72 (200522)

Mit großem Interesse habe ich mir den informativen Artikel von Alfred Rosenkränzer angeschaut. Dabei ist mir aufgefallen, dass ein Filtertyp keine Erwähnung findet. Er weist hervorragende Eigenschaften auf, insbesondere in Bezug auf die erreichbare Weitabselektion.

Zum Hintergrund: Im Rahmen unseres Projektes *Charly 25*, bei dem wir einen STEMlab 16 als SDR-Kern in einem TRX einsetzen, haben wir lange mit mehr oder weniger brauchbaren Bandpassfiltern als Preselector experimentiert. Übliche Bandpassfilter-Strukturen haben sich dabei als relativ schlecht hinsichtlich der erreichbaren Weitabselektion erwiesen.
Unser Ziel war es, >80 dB bis 100 MHz zu erreichen; das ist notwendig, da A/D-Wandler stark rauschen und ein sensitiver Empfänger daher typischerweise über leistungsfähige schaltbare aber breitbandige Vorverstärker verfügt – in unserem Falle bis zu +36 dB.


1

Auf der Suche nach besseren Filterstrukturen sind wir auf einen Typ gestoßen, der im englischen Sprachgebrauch auch als *Tubular Filter* bezeichnet und hauptsächlich für Mikrowelle eingesetzt wird. Dieser Filtertyp zeichnet sich durch C-gekoppelte Pi-Strukturen aus (**Bild 1**).
Leider wird diese Filterstruktur in den üblichen kostenlosen Simulations-Programmen ignoriert – die teuren kommerziellen Programme für die Simulation von Mikrowellenschaltungen standen uns naturgemäß nicht zur Verfügung. Die ersten Filterstrukturen entstanden daher durch Simulation und Schritt-für-Schritt-Anpassung mit dem Simulationsprogramm Elsie. Später erbarmte sich eines unserer Team-Mitglieder und schrieb uns ein kleines Programm zur Berechnung dieses Filtertyps:
https://charly25-sdr.github.io/hardware/rx-filters-v2/calculator

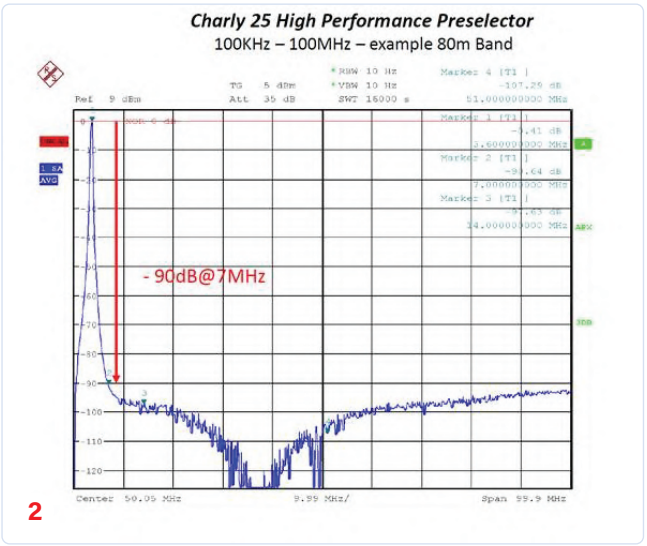
Zusätzlich bin ich dann noch auf ein kostenloses Programm gestoßen, dass diesen Filtertyp ebenfalls beherrscht (unter *Tubular Filter*): *https://rf-tools.com/lc-filter/*
Ergebnis der Bemühungen ist ein Preselector, der 11 Amateurfunkbänder erfasst (plus ein Durchgang für Rundfunkbänder) und unsere Erwartungen in Bezug auf Weitabselektion bravourös erfüllt (**Bild 2**).

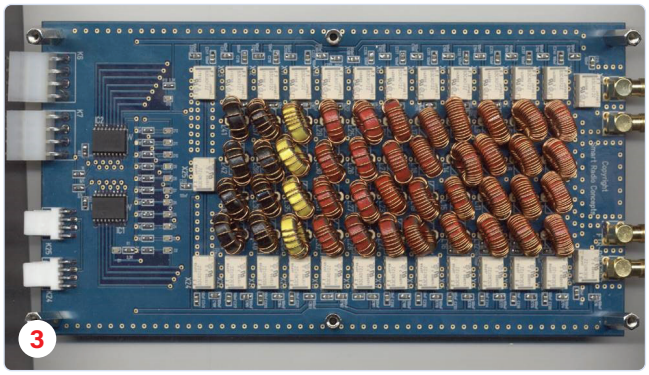
Vorsichtshalber muss ich allerdings darauf hinweisen, dass sich eine derartige Performance nur mit einem ausgeklügelten Layout auf einer 4-Layer-Platine erreichen lässt, der Masseverlauf ist extrem kritisch. Aber selbst mit einfacheren Aufbauten auf 2-Lagen-Material sollten 70 dB leicht erreichbar sein.

Bild 3 zeigt eine Ansicht des kompletten Preselectors. Dieser Preselector wird per I²C gesteuert und kann daher in jedem beliebigen Empfänger-Projekt Verwendung finden. Info findet sich unter *www.smartradioconcepts.com*.

Noch ein Kommentar zum Thema Kernmaterial: Für diese Anwendung kommen nur Eisenpulver-Kerne in Frage. Nur sie garantieren eine ausreichend gute Intermodulationsfreiheit.
Unser Filter verwendet übrigens keine PIN-Dioden zum Umschalten, weil uns die Gefahr von Intermodulation als zu groß erschien; daher die große Menge an Relais.

Edwin Richter (DC9OE)





SAT-Störungen nach Sonnenuntergang

Ich befinde mich in der angenehmen Situation, meinen wohlverdienten Ruhestand im europäischen Winter in der Wärme des brasilianischen Bahia verbringen zu können. Allerdings hat man auch hier mit den bekannten Tücken der Technik zu kämpfen.
Wir hatten hier plötzlich stark gestörten SAT-Empfang (noch analog), der immer ziemlich exakt bei Sonnenuntergang einsetzte und dann auf allen Kanälen keinen Empfang mehr zuließ. Morgens war immer wieder alles ok.
Nun ist mir vom terrestrischen Rundfunk bekannt, dass dort atmosphärische Störungen auftreten können, aber im Dezimeterbereich und quasi direkter Sicht schien mir das doch eher unwahrscheinlich. Also wandte ich mich schließlich an den örtlichen Installateur für Satellitenanlagen, der uns die Schüssel vor ein paar Jahren installiert hatte.
Dieser konnte das Problem dann auch kompetent lösen, nachdem er aufs Dach gestiegen war und den gut festkorrodierten LNB getauscht hatte!
Der Plastikdeckel, der den Eintritt verschließt, war wohl durch Hitze und UV-Strahlung spröde und löchrig geworden und es hatten sich dort einige Wespen (deutlich größer als die europäischen Kollegen) eingenistet. Zumindest waren ihre aus Lehm gebauten Eiablagen noch darin zu finden. Der SAT-Bauer meinte, da die letzten Nächte recht frisch (20°C) waren und sich die LNBs tagsüber gut aufheizen, kommen die Tierchen bei Sonnenuntergang heim, suchen Unterschlupf im LNB und stören damit den Signalweg. Morgens fliegen sie wieder aus.
So wurde das Problem für umgerechnet rund 15 € gelöst!

Wolfgang Meyer

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

200715-02

Wem **gehört das Produkt?**

Das Recht auf Reparatur gewinnt an Schwung

Von **Tessel Renzenbrink** (Niederlande)

Die Recht-auf-Reparatur-Bewegung steht für das Recht der Menschen ein, ihre eigenen Produkte reparieren und modifizieren zu können. Es gibt viele Vorteile, Gegenstände zu reparieren anstatt sie zu ersetzen. Es spart Energie, Ressourcen und Geld und reduziert obendrein den Abfall. Außerdem gibt es nur wenige Aufgaben, die befriedigender sind, als kaputte Dinge wieder in ihren ursprünglichen Zustand zu versetzen. Gesetzgebende Gremien in ganz Europa arbeiten an Regelungen zur Verbesserung der Reparierbarkeit. Die wirkungsvollste davon ist eine vom Europäischen Parlament verabschiedete Resolution für einen umfassenden Ansatz zur Gewährleistung des Rechts auf Reparatur. Sie schlägt eine lange Liste von Maßnahmen vor, die von der Verlängerung der Garantie bis hin zur Verpflichtung zur Verfügbarkeit von Ersatzteilen reicht.

Geplante Obsoleszenz

Die meisten EU-Bürger würden es vorziehen, Produkte zu reparieren, anstatt sie zu ersetzen, so eine 2018 veröffentlichte Studie der Europäischen Kommission [1]. Basierend auf der Verhaltensforschung fanden Wissenschaftler heraus, dass je nach Produkttyp 62 % bis 83 % der Menschen die Option einer Reparatur dem Neukauf vorziehen. Die Bürger sind durchaus bereit, die Wegwerfprodukte gegen langlebigere und reparierbare Produkte einzutauschen. Aber zahlreiche Hindernisse erschweren es, dieser Entscheidung zu folgen. Die vielleicht dreisteste davon ist die geplante Obsoleszenz, die Praxis, absichtlich Schwachstellen in einen Entwurf einzubauen, um ein Produkt zu einem bestimmten Zeitpunkt veralten zu lassen. Bei digitalen Geräten kann die geplante Obsoleszenz die Form annehmen, dass Software- oder Sicherheitsupdates nach nur einem oder zwei Jahren eingestellt werden. Das Gerät wird dann unbrauchbar, obwohl die Hardware noch voll funktionsfähig ist.

Ein weiteres Hindernis ist die Verhinderung von Reparaturen. Dazu gehören Konstruktionen, die es nahezu unmöglich machen, Teile auszutauschen. Der berühmten Smartphone-Reparierbarkeits-Index von iFixit listet solch unterschiedliche Übel wie verklebte Abdeckungen und schwer zugängliche Akkus auf [2]. Das Fehlen von verfügbaren Ersatzteilen oder deren Verteuerung fällt ebenfalls in diese Kategorie. Eine weitere Vermeidungsstrategie besteht darin, das Handbuch als geistiges Eigentum zu deklarieren und damit Verbrauchern und unabhängigen Reparaturwerkstätten den Erwerb zu verbieten.

Reparierbarkeitsindex

Fehlende Informationen sind eine weitere Hürde, die es den Menschen erschwert, sich für reparierbare Produkte zu entscheiden. Die Studie aus dem Jahr 2018 ergab, dass, wenn schon im Geschäft Informationen über die Reparierbarkeit bereitgestellt werden, die Verbraucher „mehr als doppelt so häufig Produkte mit den höchsten Reparierbarkeitsbewertungen wählen“. Das französische Ministerium für Ökologische Umstellung behebt diesen Informationsmangel mit der

Bild von Jose Conejo Saenz (Pixabay.com, Pixabay-Lizenz)

Einführung eines Reparierbarkeitsindex [3]. Ähnlich wie beim iFixit-Score werden dort Produkte mit einem Reparierbarkeitsindex von 1 bis 10 gekennzeichnet. Der Index basiert auf fünf Kriterien, darunter die einfache Demontierbarkeit und die Verfügbarkeit von Ersatzteilen. Die Kennzeichnung ist seit Januar dieses Jahres für Fernseher, Waschmaschinen, Laptops, Telefone und Rasenmäher verpflichtend und mit der Zeit sollen weitere Produktkategorien hinzukommen. Ziel des Ministeriums ist es, in fünf Jahren eine Reparaturquote von 60 % bei elektronischen Produkten zu erreichen.

Recht auf Reparatur für europäische Bürger

Ein Reparierbarkeitsindex könnte es in absehbarer Zukunft in allen EU-Länder geben. Im November 2020 hat das Europäische Parlament für eine Resolution für einen nachhaltigeren Binnenmarkt [4] gestimmt. Sie fordert ausdrücklich ein Recht auf Reparatur für europäische Bürger und schlägt eine lange Liste von Maßnahmen vor. Dazu gehören die kostenlose Verfügbarkeit von Handbüchern, die Förderung der Standardisierung von Ersatzteilen für die Interoperabilität, die Festlegung einer verbindlichen Mindestfrist für die Bereitstellung

von Ersatzteilen zu einem angemessenen Preis. Um die Informationsasymmetrie zu bekämpfen, sollten die Verkäufer zum Zeitpunkt des Kaufs Informationen über die Reparierbarkeit und Haltbarkeit der Produkte bereitstellen, auch was die Verfügbarkeit von Ersatzteilen und Software-Updates betrifft. Der Reparierbarkeitsindex wird als ein Mittel vorgeschlagen, um dieses Ziel zu erreichen.
Um die geplante Obsoleszenz auszumerzen, schlägt das Parlament eine umfassende Strategie vor. Die Praxis des absichtlichen Hinzufügens von Schwachstellen in einen Entwurf soll verboten werden, und während der geschätzten Lebensdauer eines Produkts müssen Software-Updates zur Verfügung gestellt werden. Um das Vertrauen der Verbraucher in gebrauchte und instandgesetzte Waren zu stärken, sollten Garantien übertragbar werden. Anstatt eine Garantie an eine Person sollte sie an das Produkt selbst gebunden sein. Um gleiche Wettbewerbsbedingungen für europäische Unternehmen zu schaffen, die diese strengen Anforderungen erfüllen müssen, fordert das Parlament die Kommission und die Mitgliedstaaten auf, mehr Anstrengungen zu unternehmen, um importierte, nicht konforme Waren von den Verkaufsregalen fernzuhalten.

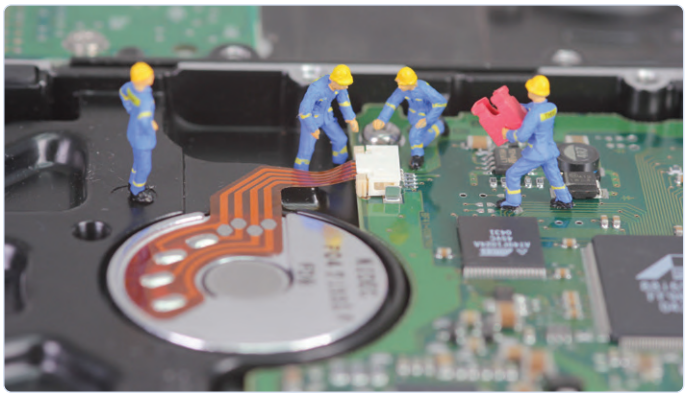


Bild von Wilfried Pohnke (Pixabay.com, Pixabay-Lizenz)

> Die Bürger sind durchaus bereit, Wegwerfprodukte gegen langlebigere und reparaturfähige Produkte einzutauschen.

Transformation der Wegwerfwirtschaft

Das Europäische Parlament ist nicht das einzige EU-Gremium, das ein Recht auf Reparatur fordert. Im März 2020 verabschiedete die Europäische Kommission den Aktionsplan für eine Kreislaufwirtschaft, der ebenfalls explizit das Recht auf Reparatur enthält [5]. Sowohl das Parlament als auch die Kommission fordern, dass das Recht auf Reparatur Teil der europäischen Green-Deal-Initiative wird, dem Aktionsplan, der Europa bis zum Jahr 2050 klimaneutral machen soll. Das Ziel ist es, die derzeitige „Take-Make-Use-Discard"-Wirtschaft in eine nachhaltige Kreislaufwirtschaft zu verwandeln. Die Erleichterung von Reparaturen zur Verlängerung des Lebenszyklus von Produkten passt zu diesem Ziel. Eine nachhaltige Wirtschaft ist nicht nur ökologisch ausgeglichener, sondern auch widerstandsfähiger. In seiner Entschließung weist das Parlament darauf hin, dass die Covid-19-Krise deutlich gemacht hat, dass das derzeitige Wirtschaftssystem mit seinen globalen Lieferketten ein fragiles System ist. Es fordert daher neue Geschäftsmodelle und Unterstützung für kleine und mittlere europäische Unternehmen. Ein Fokus auf Reparaturen vor Ort anstatt neue Waren aus Übersee einfliegen zu lassen würde zur Entwicklung von Wissen, Fähigkeiten und lokaler Wirtschaft beitragen.
Die Initiativen müssen noch das EU-Gesetzgebungsverfahren durchlaufen, bevor sie Gesetz werden, und sie könnten auf dem Weg dorthin noch verwässert werden. Aber die Tatsache, dass das Recht auf Reparatur in den Green Deal der Europäischen Union eingebettet ist, eine der obersten Prioritäten auf der Agenda der EU, ist sehr vielversprechend.

200713-02

Haben Sie Fragen oder Kommentare?
Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Senden Sie eine E-Mail an den Autor über die Elektor-Redaktion (editor@elektor.com).

WEBLINKS

[1] **Dr. Annette Cerulli-Harms et al. „Behavioural Study on Consumers' Engagement in the Circular Economy", Seite 11, Oktober 2018**: https://bit.ly/39PvWkA

[2] **Smartphone-Reparierbarkeits-Index, Ifixit**: http://bit.ly/2M8yZfJ

[3] **Ministerium für Ökologische Transformation, „Das Anti-Abfallgesetz für eine Kreislaufwirtschaft: Schlüsselmaßnahmen", Januar 2021**: http://bit.ly/3sHioAi

[4] **Entschließung des Europäischen Parlaments vom 25. November 2020 zu dem Thema „Auf dem Weg zu einem nachhaltigeren Binnenmarkt für Unternehmen und Verbraucher"**: https://www.europarl.europa.eu/doceo/document/TA-9-2020-0318_DE.html

[5] **Europäische Kommission, „Änderung unserer Produktions- und Verbrauchsmuster: neuer Aktionsplan für Kreislaufwirtschaft ebnet Weg zu klimaneutraler und wettbewerbsfähiger Wirtschaft mit mündigen Verbrauchern", März 2020**: https://ec.europa.eu/commission/presscorner/detail/de/ip_20_420

Aus dem Leben gegriffen

Das große Buch der Patzer

Von **Ilse Joostens** (Belgien)

Das Leben eines Elektronik-Enthusiasten ist nicht immer einfach, ganz gleich, ob Sie Anfänger oder eingefleischter alter Hase sind. Ob man will oder nicht, die Technik entwickelt sich in einem rasanten Tempo weiter, was bedeutet, dass man ständig lernen muss, wenn man mit den neuesten Entwicklungen Schritt halten will. Lernen ist ein Prozess des Hinfallens und Wiederaufstehens, und die wichtigsten Dinge lernt man in der Regel auf die harte Tour, auf der man unweigerlich Gefühle der Enttäuschung, des Versagens und vor allem der großen Scham zu bewältigen hat. Wenn Sie nicht zur „Millennials-Generation" gehören, werden Sie aus diesen Erfahrungen gestärkt hervorgehen, denn aus den eigenen Fehlern kann man am meisten lernen. Und wenn es wirklich aus dem Ruder gelaufen ist, dann gibt es immer noch die Möglichkeit, Ihre besten Managementfähigkeiten aufzubringen und zu versuchen, jemand anderem die Schuld zu geben.

Fallende Messer und Pyrotechnik

Als ich noch jung war, irgendwann in den achtziger Jahren des vorigen Jahrhunderts, gab es, verglichen mit heute, relativ wenige Informationen über Elektronik. Computer waren noch nicht sehr verbreitet und das Internet musste erst noch geboren werden. Ich musste mich daher meist mit einem alten Buch über die Grundlagen der Elektronik und einigen obskuren kleinen Paperbacks begnügen, die ich hier und da fand. Auch die Anzahl der Bücher über Elektronik in der örtlichen Leihbibliothek war relativ begrenzt und Datenbücher waren damals nur für Firmen und ihre streng dreinblickenden, graubärtigen Elektronikingenieure in ihren Laborkitteln erhältlich.

Irgendwann experimentierte ich fröhlich mit ein paar Dioden, Lämpchen und einem Modelleisenbahntrafo. Durch Vertauschen der Polarität oder durch Wechselspannung konnte ich mit nur zwei Drähten zwei Sätze von Birnchen unabhängig voneinander ansteuern, bis ich (natürlich) auf die unselige Idee kam, dies auch mit Netzspannung zu versuchen. Für meinen ersten Versuch hatte ich einen kleinen Brückengleichrichter gebaut, an den ich eine kleine Glühbirne angeschlossen hatte. Ich kann mich noch sehr gut daran erinnern, dass ich die Lampe für den Bruchteil einer Sekunde hell aufleuchten sah, bevor mir der Brückengleichrichter mit einem lauten Knall und einem hellen Blitz um die Ohren flog, mit einem Funkenregen wie ein Feuerwerkskörper zu Silvester. Was wusste ich schon, dass Dioden eine bestimmte maximale Sperrspannung haben? Und dass ich diese gerade bei weitem überschritten hatte? Die Hauptsicherung sprang heraus und die Panik in den Augen meiner Mitbewohner will ich gar nicht erst erwähnen.

Ein Jahr oder so später kaufte ich mir meinen ersten Lötkolben, einen dieser Bakelit-Kocher aus dem örtlichen Baumarkt. Meine ersten Lötverbindungen sahen ziemlich „klumpig" aus, aber ich war froh, dass ich nicht mehr improvisieren musste, um Verbindungen herzustellen. Damals waren Plattenspieler mit eingebautem Verstärker beliebt, und irgendwann machte ich mich daran, den Plattenspieler einer Freundin zu reparieren. Irgendwo im Gehäuse war wohl ein loses Kabel. Ich hatte nicht viel Platz auf dem Tisch, und als ich fast fertig war, rutschte mir der Lötkolben hinunter, den ich dann in einem Reflex zu fangen versuchte, natürlich indem ich das heiße Ende anfasste… Analog zur bekannten Weisheit „Versuche nie, ein fallendes Messer zu fangen", musste ich zu meinem Leidwesen erfahren, dass dies auch für heiße Lötkolben gilt. Das Ergebnis war der Geruch von verbranntem Fleisch und eine erhebliche Verbrennung an meiner rechten Hand. Ich kämpfte gegen den Schmerz an und vollendete die Arbeit, während ich nach allem griff, was sich auch nur ein wenig kalt anfühlte, um den Schmerz zu lindern, einschließlich eines Satzes von Alkali-Batterien der Größe D, die zufällig herumlagen und die ich einzeln in der Hand hielt. Ein paar Wochen später gab es wieder Probleme mit dem Plattenspieler und jemand anderes hatte ihn sich angesehen, woraufhin mir die Frage gestellt wurde, warum ich eine Batterie in den Plattenspieler gelegt hatte…

Wir hatten eine altmodische Bohrmaschine herumliegen, die keine Drehzahlregelung hatte, und der „Bohrmaschinen-Drehzahlregler" aus dem Elektor-Buch *302 Schaltungen* sah wie eine tolle Idee aus. Damals ätzte ich schon selbst Platinen mit einem Stift mit ätzmittelresistenter Tinte und das Ganze war recht schnell gebaut und erfolgreich getestet. Leider habe ich mangels finanzieller Mittel häufig „recycelte" Bauteile zweifelhafter Herkunft verwendet. Als ich den Regler einige Zeit später benutzte, fiel einer der Kondensatoren aus und füllte sofort den gesamten Arbeitsraum mit einem übel riechenden Rauch. Nachdem sich der Rauch etwas gelegt hatte und ich den bösen Kondensator mit Rauchbomben-Ambitionen ersetzt hatte, versah das Ding noch viele Jahre lang treue Dienste.

Trügerische Einfachheit

Auch als Profi arbeitete ich fleißig am Großen Buch der Patzer, aber der Platz für diesen Artikel ist glücklicherweise zu begrenzt, um

sie alle zu behandeln. Ich werde daher nicht zu sehr auf das eine Mal eingehen, als ich einen Telefonanruf nach dem anderen von Kunden hatte und ich, ermattet vom vielen Reden, das Netzkabel zu einem Gerät durchtrennte, ohne vorher den Stecker aus der Steckdose gezogen zu haben, woraufhin ein großer Teil der Firma in Dunkelheit getaucht wurde. Oder das eine Mal, dass ich mir mit einer Bohrmaschine in die Hand gebohrt habe... Oder das eine Mal, als ich - wieder einmal schläfrig vom vielen Reden - sowohl den Außen- als auch den Neutralleiter der Netzspannung berührte, woraufhin mich der heftige Stromschlag zwei Meter weit samt Schreibtischstuhl und allem Drum und Dran gegen einen Aktenschrank fliegen ließ.

Und auch jetzt sind es die scheinbar einfachsten Dinge, die mich aus der Fassung bringen. Ein IC mit mehr als hundert Pins beim Entwurf der Platine zu platzieren, ist ein Kinderspiel, aber diesen Stecker mit nur wenigen Pins beim ersten Mal richtig herum zu einzusetzen, nein, vergiss es! Nun, die Polarität der Dinge wird für immer meine Nemesis sein...

Bevor ich zum Schluss komme, möchte ich Sie dazu aufrufen, Ihre persönlichen Patzer und peinlichen Momente mit der Elektor-Community zu teilen, indem Sie einen Kommentar zu diesem Artikel hinterlassen. Es gibt nichts, wofür Sie sich schämen müssten... Nun ja, vielleicht doch ein bisschen, aber auch Schadenfreude ist eine Freude.

200682-03

Haben Sie Fragen oder Kommentare?
Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Senden Sie eine E-Mail an den Autor über die Elektor-Redaktion (editor@elektor.com).

Ein Beitrag von
Text und Auswahl der Illustration: **Ilse Joostens**
Redaktion: **Eric Bogers**
Übersetzung: **Rolf Gerstendorf**
Layout: **Giel Dols**

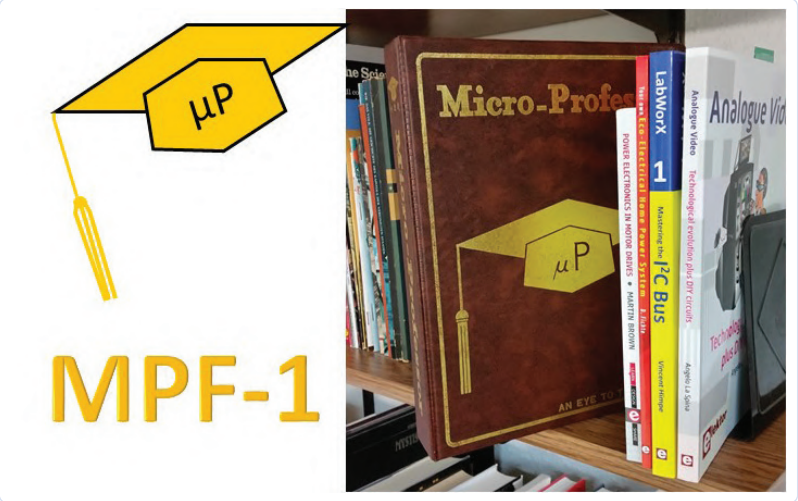
WEBLINKS

[1] **Die Millennials**: https://de.wikipedia.org/wiki/Generation_Y
[2] **ElectroBOOM Funny Compilation**: www.youtube.com/watch?v=gAnBc4iFCvk
[3] **YouTube Channel ElectroBOOM**: www.youtube.com/c/Electroboom/featured

Der Micro-Professor:
Assembler lernen auf Z80

Weiterbildung anno 1981

Von **Dipl.-Inf. Karl-Ludwig Butte**



Bücher über den 1976 erschienen Mikroprozessor Z80 von Zilog gab es 1981 zuhauf. Aber *dieses* Buch war anders! Mit rund 4 cm Rückenbreite ließ es auf rund 500 Seiten geballten Wissens schließen, und damit auf viele Stunden mühseligen Lernens zu einem komplizierten und völlig abstrakten Thema. Doch die Überraschung war perfekt, wenn man erst einmal das Buch geöffnet hatte: Statt der erwarteten schwer verdaulichen Beschreibung eines Mikroprozessors kam dieser gleich selbst zum Vorschein - einsatzbereit auf einer Platine verbaut. Neugierig geworden? Dann erkunden Sie mit mir, wie „Weiterbildung anno 1981“ mit dem Multitech Micro-Professor MPF-1 funktionierte.

Wenn sich eine Industriesparte so rasant weiterentwickelt wie die Elektronik, dann ist eine ebenso rasante Weiterbildung das einzige Rezept, um nicht innerhalb kürzester Zeit abgeschlagen zurückzubleiben. Dies erkannten auch Stan Shih, seine Frau Carolyn Yeh und fünf weitere Geschäftspartner, die im fernen Taipeh (Taiwan) 1976 die Firma Multitech gründeten – heute besser bekannt unter dem Namen „Acer“. Neben der Distribution von Ersatzteilen für elektronische Geräte und der Beratung im Mikroprozessor-Bereich stellte Multitech auch Mikroprozessor-Lernsysteme in einer eigenen Produktionsstätte her. Man hatte also schon Erfahrung mit solchen Lernsystemen, als 1981 der „Micro-Professor MPF-1“ der Weltöffentlichkeit vorgestellt wurde (**Bild 1**).

Die Hardware

Der MPF-1 war ein Einplatinen-Computer mit sechsstelliger Sieben-Segment-Anzeige und einer 36 Tasten umfassenden Tastatur. Das Herz des „Micro-Professors“ war ein Z80-Prozessor der Firma Zilog, welcher mit 1,79 MHz getaktet wurde. **Bild 2** zeigt das Blockschaltbild. Um den Z80-Prozessor scharen sich gerade mal ein Z80-PIO, ein CTC, ein 8255 sowie 2 K RAM und 4 K EPROM. Der 8255 ist ein programmierbarer, paralleler Ein-/Ausgabe-Baustein, der die Tastatur und die Sieben-Segment-Anzeige an die CPU anbindet. Außerdem

Bild 1. Der Micro-Professor MPF-1B in der aufgeschlagenen Buchhülle und die Dokumentation.

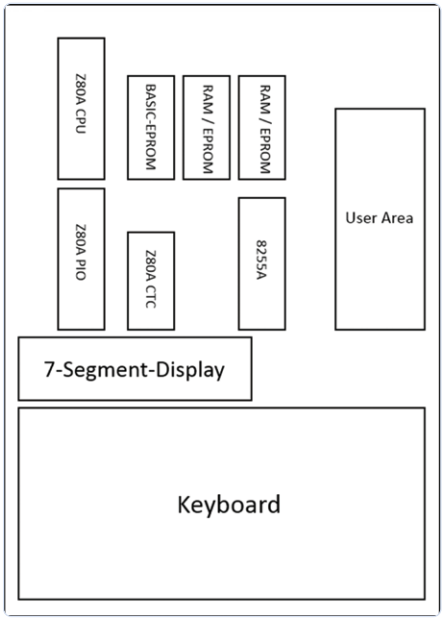


Bild 2. Das Blockschaltbild des Micro-Professors.

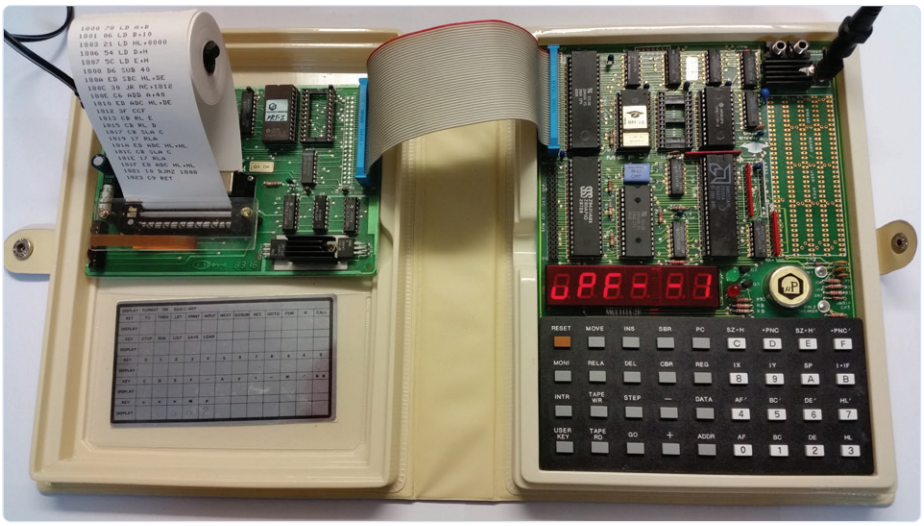


Bild 3. Der MPF-1B mit angeschlossenem Thermodrucker.

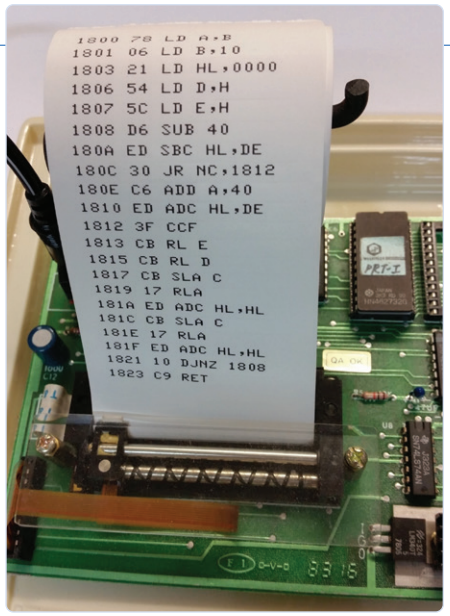


Bild 4. Das Assembler-Listing des Wurzel-Berechnungsprogramms.

ist er für den eingebauten Lautsprecher und den Kassettenrekorder-Anschluss zuständig. Dieser wiederum wurde für die Speicherung von Daten und Programmen auf Tonband-Kassetten benutzt, weil dies damals die preiswerteste Methode zur Datenspeicherung war. Der Z80-CTC ist ein Zähler-Zeitgeber-Baustein (**C**ounter/**T**imer **C**ircuit), der vier unabhängige programmierbare Zähler/Timer enthält. Und wozu brauchte man den Z80-PIO? Nun, dabei handelt es sich um einen parallelen Ein-/Ausgabe-Baustein, der zum Anschluss von Peripheriegeräten verwendet wird. Multitech bot unter anderem einen Thermodrucker, den Sie in **Bild 3** bewundern können, und ein EPROM-Programmiergerät an. Selbst ein Sprachausgabe-Board hatte Multitech im Programm. Es gab aber auch andere Firmen, die Erweiterungen für den Micro-Professor herstellten beziehungsweise herstellen ließen. So wertete das Fernlehrinstitut Christiani in Konstanz seinen Micro-Professor-Lehrgang mit einer eigenen I/O-Karte auf, die speziell auf den Fernlehrkurs zugeschnitten war.

Dokumentation

Genauso spartanisch wie die Hardware-Ausstattung war auch die Dokumentation (Bild 1) gehalten – kein Vergleich mit dem liebevoll gestalteten Begleitbuch des CP1-Lerncomputers vom Stuttgarter Kosmos-Verlag, den ich Ihnen in Elektor 7-8/2018 vorgestellt habe [1]. Nicht umsonst nahm sich ein namhaftes Fernlehrinstitut wie Christiani des Micro-Professors an und brachte einen eigenen, didaktisch aufbereiteten Kurs heraus. Multitech spendierte dem MPF-1 ein Handbuch mit gerademal 95 Seiten. Immerhin lag dieses Handbuch meinem Gerät in deutscher Übersetzung bei. Das *BASIC-MPF Operation Manual* hatte gar nur 39 Seiten und war, genauso wie das ebenfalls mitgelieferte *MPF-1 Experiment Manual*, nur im englischen Original verfügbar. Der oder die Autoren setzten beim Leser wesentliche Grundkenntnisse der Mikroprozessor-Technik voraus. So beginnt das MPF-1 Handbuch, nach einer Aufzählung der technischen Daten und einer Kurzbeschreibung der verschiedenen Tasten, mit folgenden Sätzen: „*Im Monitor Programm sind bereits alle Service-Routinen enthalten, um den Benutzern ein einfaches Arbeiten zu ermöglichen: (1) Einspeichern von Programmen in das RAM, anschließendes Überprüfen und/oder Ändern*“. Im weiteren Verlauf finden sich zwar alle wesentlichen Informationen wie der Z80-Befehlssatz, das Programmiermodell des Z80-Prozessors, die Adress- und I/O-Belegung und so weiter, aber alles meist in tabellarischer Form ohne jegliche Erläuterungen. Man tat also gut daran, sich mit passender Literatur wie „*How to Program the Z80*“ von Rodnay Zaks einzudecken. Er wusste sehr viel mehr über den Z80 zu erzählen – und sein Buch umfasste über 600 Seiten.

Programmieren in Assembler

Um hautnah zu erfahren, wie es sich mit dem Micro-Professor arbeiten lässt, nahm ich mir das Beispiel-Programm „Quadratwurzel“ aus dem Handbuch vor. Zum Glück war es nicht nur in den mnemotechnischen Abkürzungen der Z80-Assemblersprache, sondern auch in Hex abgedruckt, denn nur die Hex-Codes kann man über die Tastatur eingeben. Über die „ADDR“-Taste muss zunächst die Start-Adresse eingegeben werden, dann kann mit der „DATA“-Taste in den Daten-Eingabe-Modus geschaltet und das erste Byte eingegeben werden. Mit der „+“-Taste schaltet man ein Byte weiter und kann direkt das nächste Byte des Programms eingeben. So ließ sich das Programm recht flott eintasten. Zur Kontrolle druckte ich es über die *Disassembler-Listing-Utility* aus, die ab Hex-Adresse 6000 durch den angeschlossenen Thermo-Drucker zur Verfügung stand (**Bild 4**).
Nachdem der Ausdruck mit dem Handbuch übereinstimmte, wagte ich einen Probelauf. Der Micro-Professor sollte die Wurzel aus 81 berechnen. Das Programm erwartete diesen Wert im Register BC des Prozessors, natürlich in hexadezimaler Form! Mit den Tasten „REG“ und „BC“ konnte ich mir den momentanen Wert dieses Registers ansehen und mit „DATA“ war ich in der Lage, das erste Byte einzugeben, denn BC ist ein 16-Bit-Register. Kam nun 51H (= 81 dez.) zuerst und dann 00H oder umgekehrt? Frei nach dem Motto „Probieren geht über Studieren“ versuchte ich zuerst 00H und dann 51H, was aber ein völlig unsinniges Resultat ergab. Also, zurück auf „Start“ und mit den Tasten „REG“, „BC“, „DATA“, „5“, „1“, „+“, „0“, „0“ (**Bild 5A**) alles noch mal von vorn. Um den Probelauf zu starten, musste jetzt auf die Adresse 1800H positioniert und die „GO“-Taste betätigt werden (**Bild 5B**). Nun kam der spannende Moment, ich inspizierte das Register DE, welches das

INTERVIEW MIT MAX D. SOFFE,
MANAGING DIRECTOR VON FLITE ELECTRONICS INTERNATIONAL LIMITED

Max D. Soffe ist Mitglied des Industriebeirats für das Department of Electronic Engineering an der Royal Holloway, University of London, und das Institute of Engineering Technology (www.theiet.org). Von 2000 bis 2004 war er Vorsitzender der Educational Technology Engineering Manufacturers Association (ETEMA) in Großbritannien. Außerdem war er Direktor der British Educational Suppliers Association (BESA).

Karl-Ludwig Butte: Flite Electronics International hat die geistigen Eigentumsrechte des Micro-Professors MPF-1B von Acer gekauft. Was war die Motivation?
Max D. Soffe: Acer hieß früher Multitech. Ich traf die Gründer von Multitech auf meiner ersten Reise nach Japan im Jahr 1981, als sie den MPF-1B Z80 Mikroprozessor-Trainer in Tokio ausstellten, und konnte mich mit ihnen anfreunden. Damals gab es nur 20 Mitarbeiter in der Firma, die ihren Sitz im Hsinchu Science Park in Taipeh hatte. Ich schaltete eine doppelseitige Anzeige in einer Zeitschrift namens Wireless World in Großbritannien und verkaufte hundert MPF-1B in drei Wochen. Daraufhin gab mir Multitech die Vertriebsrechte für Großbritannien, obwohl wir bald MPF-1Bs in die ganze Welt verkauften. Später musste Acer seinen Umsatz steigern und wollte sich auf den boomenden PC-Markt konzentrieren, der, wie wir wissen, sehr erfolgreich war. Die kleinen Z80-Trainer waren effektiv verlustbringend. In elf Jahren hatte Flite Electronics Tausende von MPF1-Bs verkauft. Da das Produkt für einen Privatanwender immer noch teuer war, waren unsere Kunden vor allem der tertiäre Bildungsbereich, Universitäten und Colleges. Wir verkauften immer noch eine Menge und so verhandelte ich mit Acer über den Kauf der geistigen Eigentumsrechte. Das wurde im Februar 1993 unter Dach und Fach gebracht.
Die Unternehmung war nicht so erfolgreich, wie ich es mir gewünscht hatte, denn Acer wusch einfach seine Hände in Unschuld und überließ es uns, die „Piraterie" des Produkts zu stoppen. Wir konnten es uns nicht leisten, es mit drei oder vier Firmen aufzunehmen, die einfach das gesamte MPF-1B, das Handbuch, die Platine und die Firmware kopierten. Dennoch machten wir weiter und gestalteten die Platine leicht um, behielten aber die gesamte Firmware bei. Die Bauteile wurden immer schwieriger zu beschaffen, und es gab Probleme mit den Geschwindigkeiten von Z80-Prozessor, EEPROM und RAM.

Karl-Ludwig Butte: Es gibt Webseiten im Internet (z.B. https://en.wikipedia.org/wiki/Micro-Professor_MPF-I), die behaupten, dass der Micro-Professor auch heute noch von Flite Electronics International hergestellt und vertrieben wird. Ist das richtig? Und wenn ja, wo können die Leser ihr Gerät bestellen und zu welchem Preis?
Max D. Soffe: Ja, natürlich stellen wir den MPF-1B immer noch her, wenn auch in sehr kleinen Stückzahlen. Die letzte Charge, die wir vor etwa sechs Monaten gemacht haben, waren 25 Stück für einen Auftrag einer Universität im Nahen Osten. Die Dozenten, die in Großbritannien ausgebildet wurden, haben das MPF-1B schon vor 20 Jahren benutzt und lehren immer noch Maschinencode/Hexadezimalcodierung in ihrem Universitätslehrplan.
Wenn Elektor-Leser eine E-Mail an *sales@flite.co.uk* schicken und etwas Geduld haben, werden wir die E-Mail beantworten und ein Angebot machen. Wir haben zurzeit etwa acht Stück auf

Bild 5. A: Eingabewert $81_{dez.}$ im Register BC. B: Startadresse. C: Ergebnis 9 in Register DE.

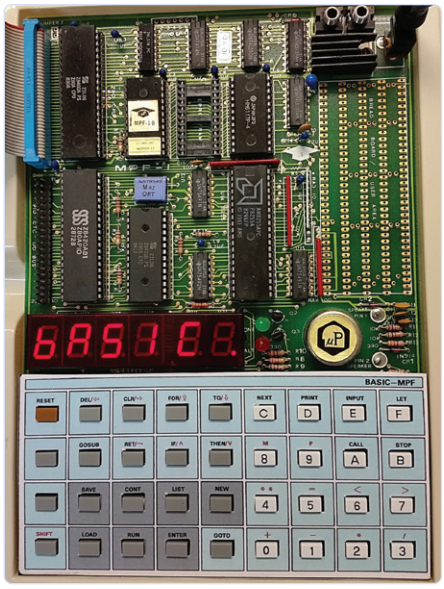


Bild 6. Der Micro-Professor im BASIC-Modus mit Tastatur-Overlay-Schablone.

Ergebnis enthielt – und siehe da, es leuchtete mir eine wunderschöne „9" entgegen (**Bild 5C**).

Der BASIC-Interpreter

Kurz nach dem Verkaufsstart des MPF-1 kam der MPF-1B auf den Markt und fortan hieß der MPF-1 nun MPF-1A, um ihn von der B-Version unterscheiden zu können. Was konnte der 1B nun mehr als sein Vorgänger? Multitech hatte ein neues EPROM spendiert, welches zusätzlich einen kleinen BASIC-Interpreter enthielt. Den wollte ich natürlich auch ausprobieren, denn BASIC mit Sieben-Segment-Anzeige und Hex-Tastatur – so etwas hatte auch ich noch nie gesehen. Dem MPF-1B lag zu diesem Zweck eine Tastatur-Overlay-Schablone bei, die bei meinem Gerät jedoch abhandengekommen war. Aber wozu gibt es heutzutage Internet? Die

Lager. Wenn diese Geräte verkauft sind, kann es allerdings zwei bis drei Monate dauern, bis wir eine weitere Charge produzieren. Ich habe auch festgestellt, dass einige gebrauchte Geräte bei eBay verkauft werden.
Wie ich bereits erklärt habe, sind die Bauteile inzwischen recht teuer für das, was sie leisten, und Sie können am Produktionsdatum auf dem IC erkennen (z.B. 9243), dass die Bauteile in der 43. Woche des Jahres 1992 hergestellt wurden. Wir haben also eine Menge Zeit, um die Komponenten und das fertige Produkt auf Geschwindigkeit und Wettlaufsituation zu testen, wobei die Komponenten nicht kompatibel sind. Aus diesem Grund haben wir alle ICs in Fassungen gesetzt, damit die ICs leicht austauschbar sind. Wenn wir ein fertiges Neugerät liefern, ist es natürlich voll funktionsfähig und hat eine Garantie von einem Jahr.
Aufgrund all dessen ist uns klar, dass wir aus einem solchen Produkt niemals ein erfolgreiches Geschäftsmodell machen können. Es war damals das Sprungbrett für Acer, um eine riesige internationale Marke zu werden, aber genau wie Acer wird auch uns der Micro-Professor niemals nennenswerte Gewinne einbringen. Leider können wir es uns nicht leisten, das Produkt für weniger als 250 Pfund plus Mehrwertsteuer und Porto zu verkaufen.

Karl-Ludwig Butte: Welche Art von Dokumentation liefern Sie mit dem Micro-Professor aus?
Max D. Soffe: Die Dokumentation ist sehr wichtig. Wir liefern jeden Micro-Professor mit einem Student-Workbook aus. Dieses Handbuch soll den Studenten vom Auspacken und Einschalten des Micro-Systems bis zum Verständnis von Mikroprozessorsystemen, Architektur, Hardware, Software und Schnittstellen führen. Ursprünglich war es als Selbstlernbuch gedacht, um Studenten in der hexadezimalen Programmierung zu schulen. Wie wir wissen, arbeiten Computer binär, die nächste Stufe davon ist hexadezimal. Die Schönheit und der Nutzen der hexadezimalen Codierung lagen in der Zeit, als Speicher teuer und sehr begrenzt war. Der Micro-Professor lehrte den Studenten, sehr „straffen" Code zu schreiben, und nicht ausufernden, wie es viele Programmierer heutzutage tun. In der Tat ist es nicht wirklich Programmieren, wenn man einfach nur „Sprechblasen" herumschiebt.

Karl-Ludwig Butte: Was ist heute das Hauptgeschäft von Flite Electronics International?
Max D. Soffe: Aufgrund der Raubkopien des Micro-Professors in den 1990er Jahren schloss sich Flite Electronics mit einer Firma in Taiwan namens K&H zusammen, die Geräte für den Bildungssektor herstellt. Das lag daran, dass unser Kundenstamm hauptsächlich aus Bildungseinrichtungen bestand, nicht aus Privatpersonen. Wir sind Distributor von K&H für Produkte für den Technik-Bildungsbereich in Großbritannien. Vor kurzem haben wir ein 240.000 Pfund teures Labor für elektrische Maschinen in der Universität Coventry installiert. Wir haben das System nicht entworfen, sondern aus bestehenden K&H-Modulen zusammengesetzt. Letzte Woche haben wir dieses Labor für weitere 50.000 Pfund erweitert.
Außerdem liefern wir nach wie vor allgemeine Mikroprozessor-Trainer für den 68000 von Motorola, den 8086 von Intel und den 8032-Mikrocontroller. Auch hier verkaufen wir jeweils zehn Stück pro Jahr, wobei wir hauptsächlich bestehende Kunden und Vertretungen unterstützen, aber wir erstellen auch gerne Angebote für diese Produkte. Ein weiteres Zusatzprodukt, das wir in den 1990er Jahren als Ergänzung zu all unseren Mikroprozessor-Schulungssystemen entwickelt haben, ist ein All-in-One-Labor namens APB (Application Board). Auch dieses wird mit einem Schulungshandbuch geliefert.

Karl-Ludwig Butte: Vielen Dank für dieses Gespräch.

Schablone war schnell gefunden und auf etwas dickeres Papier ausgedruckt. Nur die Löcher mit dem Skalpell auszuschneiden dauerte eine Weile. Das Ergebnis können Sie in **Bild 6** bewundern.
Als Beispielprogramm wählte ich eine Routine, die mehrfach einen Signalton erzeugen kann, aus dem *BASIC-MPF Operation Manual* aus. Der BASIC-Interpreter liegt im Speicher ab Adresse 0800H und wird mit einem „GO"-Befehl gestartet. Er meldet sich mit „bASIC" in der Anzeige, so wie in Bild 6 zu sehen. Jetzt kann das Programm eingegeben werden: Zuerst die Zeilennummer, dann ein BASIC-Statement und gegebenenfalls Variablen, Zahlen oder Parameter. Die BASIC-Statements sind jeweils mit einem Tastendruck abrufbar. Zum Schluss wird „ENTER" gedrückt, ganz so, wie früher auf einem Apple II, PET 2001 oder TRS-80. Soweit klappte alles ganz gut, nur das `FOR A=1 TO C` wollte sich nicht eingeben lassen. Statt des „="-Zeichens erschien hartnäckig eine „5" im Display, obwohl ich die SHIFT-Taste zuvor gedrückt hatte. Es war wohl eine alte Gewohnheit: Weil die Tastatur des Micro-Professors so sehr an eine Taschenrechner-Tastatur erinnerte, betätigte ich die „SHIFT"-Taste und ließ sie wieder los, statt sie gedrückt zu halten, wie bei einem PC. Es dauerte eine Weile, bis es mir dämmerte, dass der Micro-Professors den Anspruch hatte, ein PC zu sein. Ich musste die SHIFT-Taste gedrückt halten, während ich die „="-Taste betätigte.
Nachdem diese kleine Hürde genommen war, ließ sich der Rest des Programms leicht eingeben. In **Bild 7** sehen Sie zwei Kostproben, wie die BASIC-Befehle in der Sieben-Segment-Anzeige dargestellt werden. **Bild 7A** heißt im Klartext

Bild 7. A: 10 Input C. B: 50 Next B. C: Eingabe-Aufforderung.

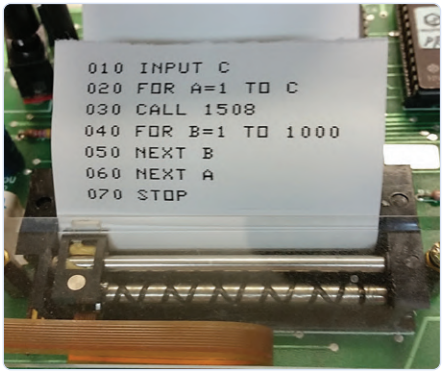


Bild 8. Das Beispiel-BASIC-Programm, ausgedruckt auf dem Thermodrucker.

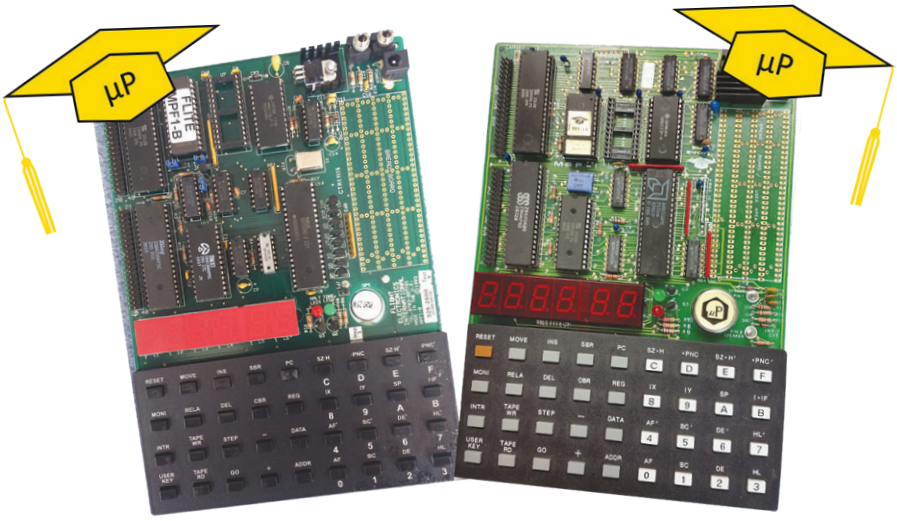


Bild 9. Der Micro-Professor von Flite Electronics International (links) und sein Kollege von Multitech (rechts).

`10 Input C` und **Bild 7B** stellt die Zeile `50 Next B` dar. Im Handbuch ist eine Tabelle der BASIC-Befehle mit ihrer jeweiligen Darstellung auf der Anzeige abgedruckt. Das ist auch nötig, denn teilweise braucht man sehr viel Fantasie. Es gibt auch Editier-Möglichkeiten, aber dagegen ist selbst der Editor „vi“ aus der Unix-Welt höchst komfortabel.

Zeit, das erste, selbst eingegebene BASIC-Programm auszuprobieren. Nach dem Start mit der „RUN“-Taste wurde die Anzahl der zu erzeugenden Signaltöne abgefragt (**Bild 7C**). Ich gab „3“ und „ENTER“ ein und hörte, ganz wie erwartet, drei recht hohe Signaltöne. Bei etwas längeren Programmen ist die Wahrscheinlichkeit recht hoch, dass man sich doch irgendwo vertippt hat. Einen solchen Tipp-Fehler nur mit der kryptischen Sieben-Segment-Anzeige zu finden, erscheint mir doch sehr mühselig. Wie gut, wenn man auch den Drucker am System angeschlossen hat, denn die BASIC-Programme lassen sich im Klartext ausdrucken. **Bild 8** zeigt das Signalton-Programm.

Fazit

Eins ist sicher: Wenn Sie sich ausgiebig mit dem Micro-Professor auseinandergesetzt haben, sind Sie mit jedem Bit des Z80-Prozessors und seinen Peripherie-Bausteinen per Du. Der Weg dorthin ist zwar nicht immer einfach und Sie werden Ausdauer brauchen, aber in meinen Augen lohnt es sich. Sie werden auch verblüfft sein, wie sehr sich die Konzepte gleichen, wenn Sie zum Beispiel den „Assembler-Crashkurs“ von Miroslav Cina lesen, der in Elektor ab Heft 7-8/2015 [2] erschienen ist.

Im Laufe der Recherchen zu diesem Artikel fand ich Hinweise im Internet, dass der Micro-Professor MPF-1B noch heute hergestellt und vertrieben wird. Ich ging den Hinweisen nach und nahm Kontakt mit der Firma Flite Electronics International Ltd. in Waltham Chase, Hampshire [3] in Großbritannien auf. Sie bestätigte mir, dass sie die Rechte am Micro-Professor übernommen hat und das Gerät auch heute noch herstellt. **Bild 9** zeigt den Micro-Professor von Flite Electronics International - Seite an Seite mit seinem taiwanesischen Kollegen von Multitech. Ich konnte mich von der hervorragenden Qualität des britischen Geräts überzeugen, welches Flite Electronics großzügig zur Verfügung stellte. Jetzt wollte ich es natürlich genau wissen. Freundlicherweise erklärte sich Max D. Soffe, Managing Director von Flite Electronics, zu einem Interview bereit (siehe Kasten).

200679-02

Sie haben Fragen oder Kommentare?
Gerne können Sie sich mit dem Autor unter LKL_Butte@web.de oder der Redaktion unter redaktion@elektor.de in Verbindung setzen.

PASSENDE PRODUKTE

› **E-Book: Assembly Language Essentials**
www.elektor.de/assembly-language-essentials-e-book

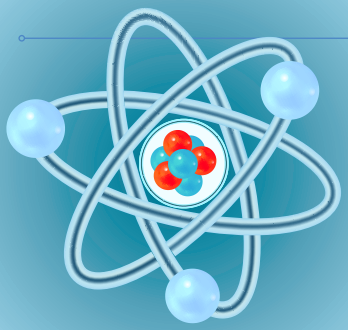
Ein Beitrag von
Text und Bilder: **Karl-Ludwig Butte**
Redaktion: **Jens Nickel**
Layout: **Giel Dols**

WEBLINKS

[1] **Butte, Karl-Ludwig: „Einstieg in die Mikroprozessortechnik mit dem Kosmos CP1 (1983)“, Elektor 7-8/2018:** www.elektormagazine.de/magazine/elektor-201807/41697

[2] **Cina, Miroslav: „Assembler Crashkurs“, Elektor 7-8/2015:** www.elektormagazine.de/magazine/elektor-201507/27858

[3] **Flite Electronics International Limited:** http://www.flite.co.uk



Aller Anfang …

muss nicht schwer sein!

Von **Eric Bogers** (Elektor)

Nach den Widerständen aus der letzten Folge [1] sind nun die Kondensatoren an der Reihe. Kondensatoren können auch als Widerstände betrachtet werden, allerdings für Wechselstrom. Das liegt daran, dass Kondensatoren Gleichstrom nicht durchlassen, Wechselstromsignale aber schon. Kondensatoren werden aber nicht nur als Wechselstromwiderstand verwendet, sondern können auch zur Speicherung von elektrischer Energie genutzt werden.

Ein Kondensator besteht aus zwei Leitern (Metallplatten zum Beispiel oder allgemeiner Elektroden), die in einem sehr geringen Abstand zueinander angeordnet sind, aber eigentlich keinen elektrischen Kontakt zueinander haben. Dieses Konzept wird auch in dem Schaltsymbol (**Bild 1**) ausgedrückt: zwei parallele Leitungen, die nicht miteinander verbunden sind.

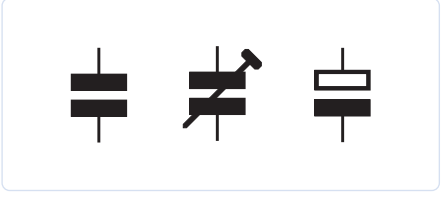


Bild 1. Symbole für Kondensatoren. Von links nach rechts: „normaler" Kondensator, einstellbarer Kondensator („Trimmer"), Elektrolytkondensator.

Aber Moment mal! Wie ist es möglich, dass ein Bauteil Gleichstromsignale nicht durchlässt, Wechselstromsignale aber schon? Wir können uns das klar machen, wenn wir uns überlegen, was „Strom" eigentlich ist: der Transport von Ladung. Wenn Ladung transportiert wird, fließt ein Strom. Wenn wir einen Kondensator an eine Gleichspannung anschließen, wird es einen anfänglichen Strom geben, denn Elektronen bewegen sich von einer Platte des Kondensators zum positiven Anschluss der Gleichspannungsquelle und vom negativen Anschluss zur anderen Platte. Aber nachdem der Kondensator auf diese Weise aufgeladen wurde, können die Elektronen nirgendwo mehr hin. Sie können nicht auf magische Weise von einer Platte zur anderen wandern, oder anders ausgedrückt, es fließt kein Strom mehr. Der Kondensator verhält sich wie ein offener Stromkreis (oder Isolator oder Widerstand mit unendlich hohem Wert).

Anders verhält es sich, wenn man eine Wechselspannung an den Kondensator anlegt: Dann wird der Kondensator ständig aufgeladen, entladen, anderes herum geladen und so weiter. Es fließt also ein Strom hin und her (also ein Wechselstrom), auch wenn sich – um das klar zu sagen - keine Elektronen durch den Kondensator von einer Platte oder Elektrode zur anderen bewegen!

Kapazität

Wir haben es schon angesprochen: Ein Kondensator kann Ladung (und damit elektrische Energie) speichern, und zwar umso mehr, je größer die *Kapazität* des Kondensators ist.
Für die Kapazität C eines Kondensators gilt:

$$C = \frac{\varepsilon \cdot A}{d}$$

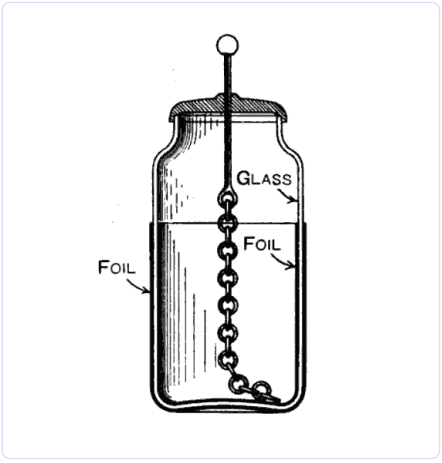


Bild 2. Aufbau einer Leidener Flasche (Quelle: Wikimedia Commons, http://bit.ly/wiki-leyden-jar).

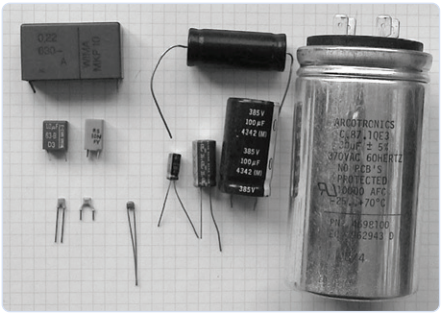


Bild 3. Einige gängige Kondensatortypen.

Dabei ist ε die ***Dielektrizitätskonstante***, A ist die Oberfläche der Elektroden („Platten") und d ist der Abstand zwischen den Platten. Diese Formel können Sie eigentlich gleich wieder vergessen. Wir zeigen sie nur, damit Sie eine Vorstellung von den Einheiten haben, die die Kapazität eines Kondensators ausmachen.
Es ist offensichtlich, dass die Kapazität mit der Fläche der Elektroden zunimmt („je größer sie sind, desto mehr Ladung passt hinein"). Und wenn die Elektroden näher beieinander liegen, erhöht sich auch die Kapazität. Die Dielektrizitätskonstante aber, die einen großen Einfluss auf die Kapazität hat, erfordert ein wenig mehr Erklärung.
Zwischen den Elektroden eines Kondensators befindet sich ein nicht leitendes Medium („Material"). Für die Dielektrizitätskonstante ε haben wir die folgende Beziehung:

$$\varepsilon = \varepsilon_0 \cdot \varepsilon_r$$

Dabei ist ε_0 die Dielektrizitätskonstante (elektrische Feldkonstante, Permittivität) des Vakuums (eine Naturkonstante mit dem extrem niedrigen Wert von $8{,}85 \cdot 10^{-12}$ $C^2/N\,m^2$) und ε_r ist die relative Dielektrizitätskonstante des Mediums, die eine Eigenschaft des Materials zwischen den Platten ist. Für Papier hat ε_r den Wert 2, in Vakuum und Luft den Wert 1 und bei Keramik und Glimmer beträgt der Wert 7. Damit sollte klar sein, dass durch eine geeignete Wahl des Dielektrikums die Kapazität eines Kondensators erheblich gesteigert werden kann.
Die Kapazität eines Kondensators wird mit der Einheit *Farad* (abgekürzt F) angegeben. Diese Einheit erweist sich in der Praxis als unpraktisch groß. Wir haben es in der Praxis normalerweise mit Werten in der Größenordnung von Picofarad (10^{-12} F), Nanofarad (10^{-9} F) und Mikrofarad (10^{-6} F) zu tun. Es gibt aber auch Kondensatoren mit Werten von 1...10 F oder noch mehr, die sogenannten Supercaps, die von verschiedenen Herstellern angeboten werden und als eine Art Pufferbatterie eingesetzt werden, zum Beispiel um eine Echtzeituhrschaltung bei Netzausfall in Gang zu halten.

Geschichte

Der Kondensator tauchte erstmals im Jahr 1746 im Labor von Pieter van Musschenbroeck an der Universität Leiden auf. Dieser Kondensator bestand aus einem (Glas-) Gefäß, das innen und außen mit Zinnfolie überzogen war (**Bild 2**). Kombiniert mit einem elektrostatischen Generator, der die Ladung lieferte, war diese Leidener Flasche zu spektakulären (und sicher nicht ganz ungefährlichen) Experimenten fähig.
Heutzutage können Kondensatoren glücklicherweise viel kleiner gebaut werden. Die gängigsten Modelle sind in **Bild 3** zu sehen: ganz rechts einen „dicken" Metall-/Papierkondensator, wie sie oft zur Entstörung von Motoren verwendet werden. Ein Vorteil ist, dass diese Kondensatoren sehr robust, ein Nachteil, dass sie nicht besonders klein sind. Die Elektrolytkondensatoren in der Mitte sind wesentlich kleiner. Bei Elektrolytkondensatoren (vulgo: Elko) ist das Dielektrikum ein fester oder flüssiger Elektrolyt. Ein wesentlicher Nachteil von Elkos ist, dass sie gepolt sind, also einen Plus- und einen Minuspol besitzen. Diese Kondensatoren müssen unbedingt richtig herum eingebaut werden, Fehler können (insbesondere bei den sogenannten Tantal-Kondensatoren) spektakuläre und extrem übelriechende Folgen haben. Es gibt auch bipolare Elektrolytkondensatoren, bei denen es keinen Unterschied macht, wie herum sie eingesetzt werden.
Oben links in Bild 3 sind ein paar Polypropylen-Kondensatoren und darunter ein paar Keramik-Kondensatoren zu sehen. Letztere sind angenehm klein und billig, aber nur mit relativ kleinen Kapazitätswerten erhältlich.
Es gibt so viel zu sagen über all die verschiedenen Arten von Kondensatoren und ihre spezifischen Eigenschaften, dass wir leicht eine ganze Elektor-Ausgabe mit dem Thema füllen könnten. Aber wenn Sie Ihre ersten Schritte auf dem Weg zur Elektronik machen, werden die oben genannten Typen in der Regel Ihre Bedürfnisse abdecken. Die „anderen" Kondensatoren lernen Sie im Laufe der Zeit kennen - garantiert!

Der Kondensator als Energiespeicher

Ein Kondensator kann elektrische Ladung speichern, und zwar umso mehr, je höher seine Kapazität ist und je höher die Spannung ist:

$$Q = C \cdot U$$

Daraus folgt die offizielle Definition der Einheit der Kapazität, das Farad: Die Kapazität eines Kondensators ist die Ladungsmenge C, die bei einer Spannung von einem Volt gespeichert werden kann. Die in einem Kondensator gespeicherte Energie wird wie folgt berechnet:

$$W = \frac{1}{2} \cdot C \cdot U^2$$

Kondensatoren als Energiespeicher begegnen uns vor allem in Netzteilen: Die (heruntertransformierte) Wechselspannung wird gleichgerichtet, so dass eine pulsierende Gleichspannung entsteht. Diese wird in einem (recht großen) Kondensator gespeichert, der als Speicher oder Puffer fungiert. Dies nennt man Pufferung oder Glättung der Spannung und dieser Kondensator wird Puffer- oder Glättungskondensator genannt. Sobald wir zu den Themen Dioden und Gleichrichterschaltungen kommen, werden wir uns dieses Thema genauer ansehen.

Lade- und Entladeeigenschaften

Wenn ein Kondensator geladen oder entladen wird, hängt die Spannung über dem Kondensator von der vorhandenen Ladung ab:

$$U = \frac{Q}{C}$$

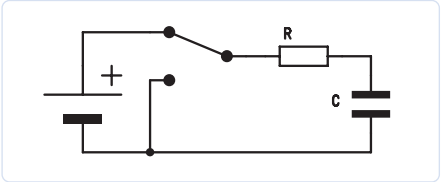


Bild 4. Periodisches Laden und Entladen eines Kondensators über einen Widerstand.

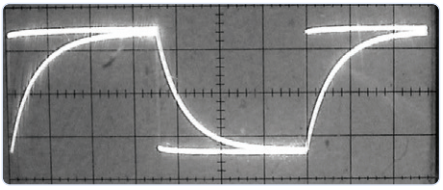


Bild 5. So sehen die Lade- und Entladezyklen auf einem Oszilloskop aus.

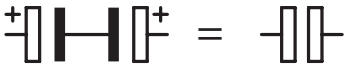


Bild 6. Zwei gegeneinander gepolte Elektrolytkondensatoren bilden einen bipolaren Elektrolytkondensator.

Wenn wir einen Kondensator mit einem konstanten Strom aufladen, dann steigt die Spannung über dem Kondensator linear an - und beim Entladen mit einem konstanten Strom sinkt die Spannung linear. In der Praxis werden Kondensatoren selten mit einem konstanten Strom ge- oder entladen, viel häufiger wird mit einer konstanten Spannung gearbeitet (**Bild 4**).

Da die Spannung am Kondensator während des Ladevorgangs kontinuierlich ansteigt, nimmt die Spannung am Widerstand kontinuierlich ab (schließlich ist die Spannung am Widerstand gleich der konstanten Ladespannung minus der Spannung am Kondensator) und das bedeutet natürlich, dass der Ladestrom kontinuierlich abnimmt. Und das bedeutet auch, dass sich die Aufladung des Kondensators immer langsamer vollzieht: Die Spannung über dem Kondensator nähert sich asymptotisch der Ladespannung, wie das Oszillogramm in **Bild 5** beweist.

Beim Entladen eines Kondensators über einen Widerstand passiert genau das Gegenteil: Durch die hohe Kondensatorspannung gibt es zunächst einen großen Entladestrom, aber da die Spannung über dem Kondensator kontinuierlich abnimmt, wird auch der Entladestrom immer kleiner.

Beim Aufladen eines Kondensators über einen Widerstand gelten die unten stehenden Beziehungen für Spannung und Strom (wir werden Sie nicht mit der Herleitung belästigen).

$$U_C = U_0 \cdot \left(1 - e^{-t/RC}\right)$$

$$I_C = \frac{U_0}{R} \cdot \left(e^{-t/RC}\right)$$

Bei Entladen gelten diese Formeln:

$$U_C = U_0 \cdot \left(e^{-t/RC}\right)$$

$$I_C = \frac{-U_0}{R} \cdot \left(e^{-t/RC}\right)$$

Reihen- und Parallelschaltungen von Kondensatoren

Zum Abschluss dieser Folge wollen wir noch einen kurzen Blick auf die Reihen- und Parallelschaltung von Kondensatoren werfen. Die Parallelschaltung ist eigentlich ganz einfach. Wenn wir Kondensatoren parallel schalten, können wir uns vorstellen, dass die „Platten" des resultierenden Kondensators größer geworden sind und somit auch die resultierende Kapazität. Es gilt:

$$C_{total} = C_1 + C_2 + \ldots + C_n$$

Aber das ist doch super! Für parallel geschaltete Kondensatoren gilt der gleiche Zusammenhang wie für die Reihenschaltung von Widerständen! Würde etwas Ähnliches auch für die Reihenschaltung von Kondensatoren gelten? Und tatsächlich, bei der Reihenschaltung von Kondensatoren gilt der gleiche Zusammenhang wie bei der Parallelschaltung von Widerständen:

$$C_{total} = \frac{1}{\frac{1}{C_1} + \frac{1}{C_2} + \ldots + \frac{1}{C_n}}$$

Zum Schluss noch eine Bemerkung zu Elektrolytkondensatoren: Wenn zwei Elektrolytkondensatoren in in „Anti-Reihe" geschaltet werden, entsteht ein bipolarer Elektrolyt (**Bild 6**).

Wir belassen es für diese Folge dabei. In der nächsten Folge werden wir uns die Kondensatoren als Wechselstromwiderstände genauer ansehen. Dazu gibt es einiges zu sagen, denn bei Kondensatoren gibt es eine Phasenverschiebung zwischen Spannung und Strom - im Gegensatz zu ohmschen Widerständen.

200680-03

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

Haben Sie Fragen oder Kommentare?
Haben Sie technische Fragen oder Anregungen zu diesem Artikel? Senden Sie eine E-Mail an den Autor oder an die Elektor-Redaktion über editor@elektor.com.

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

PASSENDE PRODUKTE

- **B. Kainka, Elektronik-Grundlagen und Einsteiger-Projekte, Elektor 2020**
www.elektor.de/elektronik-grundlagen-und-einsteiger-projekte
- **B. Kainka, Elektronik-Grundlagen und Einsteiger-Projekte (E-Book), Elektor 2020**
www.elektor.de/elektronik-grundlagen-und-einsteiger-projekte-pdf

WEBLINK

[1] **E. Bogers, „Aller Anfang...muss nicht schwer sein (6)", Elektor Januar/Februar 2021**: www.elektormagazine.de/200551-03

elektorstore

www.elektor.de

Der Elektor Store

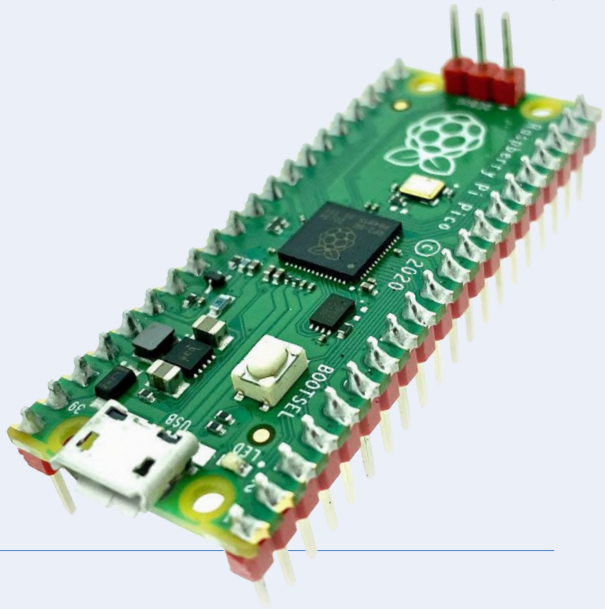
Nie teuer, immer überraschend!

Der Elektor Store hat sich vom Community-Store für Elektor-eigene Produkte wie Bücher, Zeitschriften, Bausätze und Module zu einem umfassenden Webshop entwickelt, der einen großen Wert auf überraschende Elektronik legt. Wir bieten die Produkte an, von denen wir selbst begeistert sind oder die wir einfach ausprobieren wollen. Wenn Sie einen Produktvorschlag haben, sind wir hier erreichbar (sale@elektor.com). Unsere Bedingungen:
Nie teuer, immer überraschend!

Raspberry Pi Pico
(mit bestückten Headern)

Preis: 9,95 €

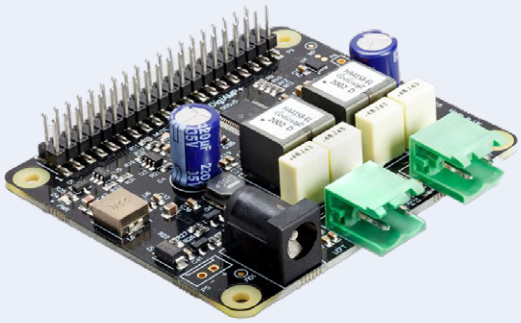
www.elektor.de/19568



JOY-iT JT-RD6006 Labornetzteil (360 W)

Preis: 89,95 €
Mitgliederpreis: 80,96 €

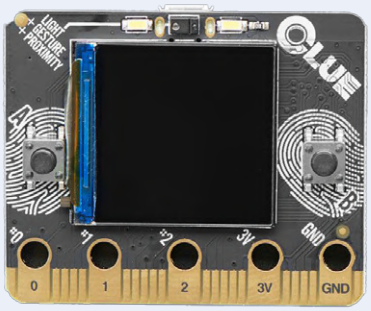
www.elektor.de/19564



IQaudIO DigiAMP+ DAC & Class-D-Verstärker für Raspberry Pi

Preis: 29,95 €
Mitgliederpreis: 26,96 €

www.elektor.de/19538



Adafruit CLUE – nRF52840 Express mit Bluetooth LE

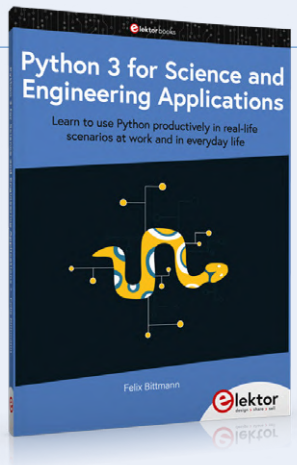
Preis: 49,95 €
Mitgliederpreis: 44,96 €

www.elektor.de/19512

IoTize TapNLink Primer NFC-BLE-WiFi Evaluation Kit

Preis: 59,95 €
Mitgliederpreis: 53,96 €

www.elektor.de/19494



Python 3 for Science and Engineering Applications

Preis: 29,95 €
Mitgliederpreis: 26,96 €

www.elektor.de/19441

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 14. Juni 2021.

Die Gewinner des Hexadokus aus der Ausgabe März/April stehen fest!

Die richtige Lösung ist: **29BF4**

Aus allen Einsendungen mit der richtigen Lösung haben wir die fünf Gewinner eines SparkFun-MicroMod-Bundles gezogen.

Die Namen der Gewinner werden unter www.elektormagazine.de/hexadoku bekannt gegeben.

Herzlichen Glückwunsch!

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

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

Der Rechtsweg ist ausgeschlossen. Mitarbeiter der in der Unternehmensgruppe Elektor International Media B.V. zusammengeschlossenen Verlage und deren Angehörige sind von der Teilnahme ausgeschlossen.

JETZT ABONNIEREN UND SIE ERHALTEN

WILLKOMMENSGESCHENK

ALLE 2 MONATE, EINE NEUE AUSGABE MIT DEN BESTEN ÜBER RASPBERRY PI

NUR 54,95 € PRO JAHR (6 AUSGABEN)

Mit einem MagPi Abo erhalten Sie:

- 6 gedruckte Ausgaben der MagPi
- ein Willkommensgeschenk
- Zugang zu dem Online Archiv mit allen MagPi Ausgaben seit 2016

Ihre Vorteile:

- 6 gedruckte Ausgaben der MagPi
- ein Willkommensgeschenk
- Zugang zu dem Online Archiv mit allen MagPi Ausgaben seit 2016
- Günstiger als Einzelkauf
- Jede Ausgabe direkt zu Ihnen nach Hause
- Alle Ausgaben auch als PDF
- Willkommensgeschenk mit tollen Inhalten

JETZT ABONNIEREN: WWW.MAGPI.DE

elektor MAG
sixty > years > young

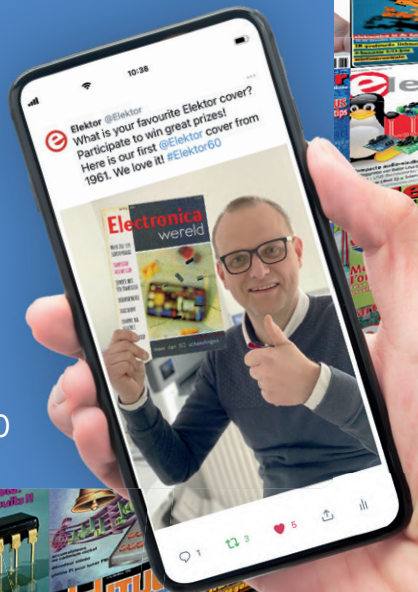
CONTEST

Poste ein Bild von deinem Lieblingstitelblatt von Elektor auf Social Media.

1. **Verlinke @ElektorDE**
2. **#Elektor60 hinzufügen**

Nimm teil und erhalte die Chance tolle Preise zu gewinnen!

Weitere Informationen:
www.elektor.de/contest-Elektor60



1961 1971 1981 1991 2001 2011 2021