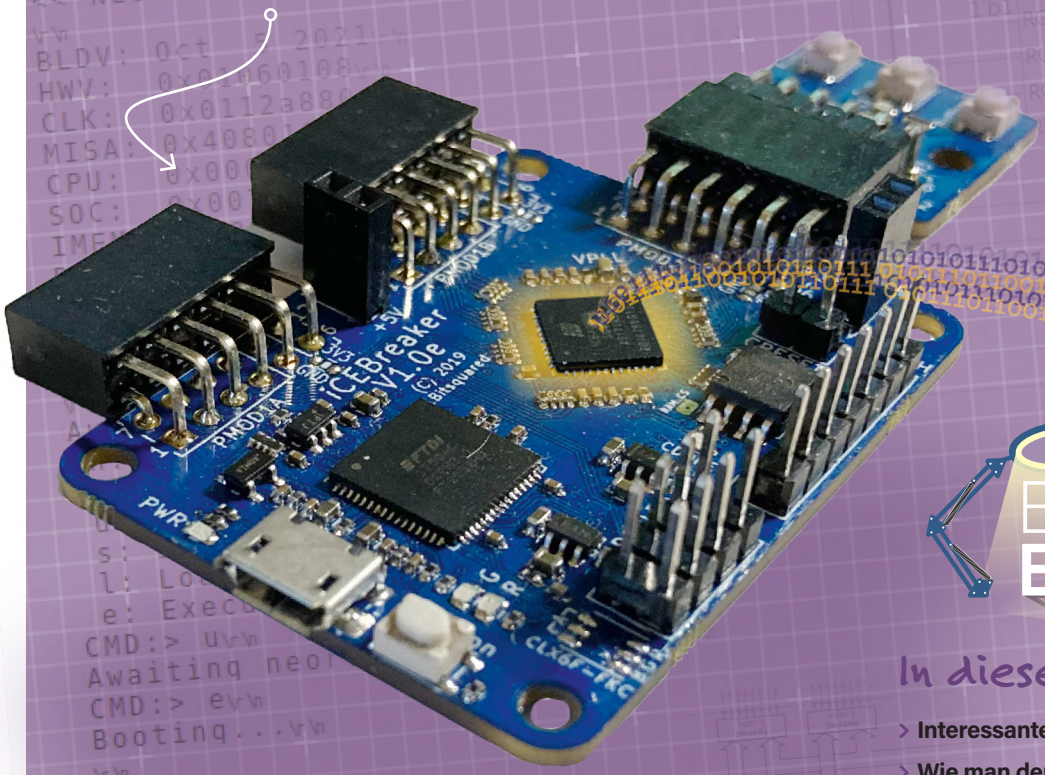
G3078 E (D) € 14,90 | CHF 27,00 | (A, B, L) € 16,00

elektorMAG
design > share > earn

584
MÄRZ/APRIL 2022
ELEKTORMAGAZINE.DE

Mein eigener RISC-V-Controller



IM FOKUS
Embedded-Entwicklung

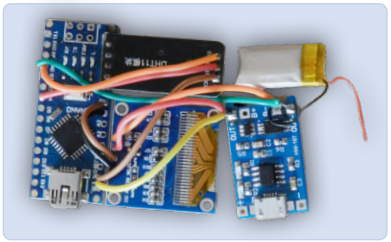
In dieser Ausgabe

- Interessante Raspberry Pi RP2040 Boards
- Wie man den Seriellen Plotter der Arduino-IDE nutzt
- Drahtloses Monitoring und Debugging
- Tipps und Tricks zur Identifizierung von Bauteilen
- Lithium-Akkus reparieren
- SPS-Programmierung mit dem Raspberry Pi und dem OpenPLC-Projekt
- Magnetische Levitation auf die sehr einfache Art
- Ethics: Die drei Fragezeichen
- Der Embedded-Markt: Anschauliche Fakten

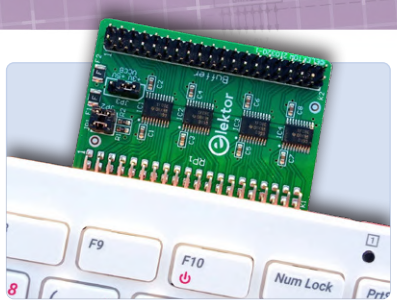
und vieles mehr!

S. 52 **Was gibt's Neues in der Embedded-Entwicklung?**
Rust und die Aktualisierung von IoT-Implementierungen

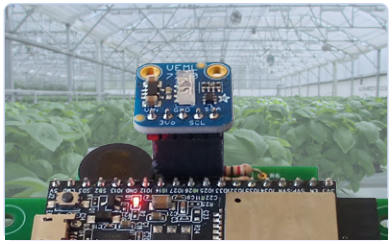
S. 86 **Unbekannte Controller**
über die Sie Bescheid wissen sollten



Temperatur- und Feuchtemessgerät
Wetterstation für die Hosentasche
S. 96



Puffer-Board für den Raspberry Pi 400
Schützen Sie die I/Os!
S. 24



Ein Touch-Schalter ohne Touch
mit Handgesten-Steuerung!
S. 42

4 198630 314904 20004

elektor e-zine

Your dose of electronics

Jede Woche, in der Sie den Elektor e-zine Newsletter nicht abonnieren, ist eine Woche mit großartigen Artikeln und Projekten zum Thema Elektronik, die Sie verpassen!

Also, worauf warten Sie noch? Melden Sie sich heute für unseren Elektor e-zine Newsletter unter www.elektor.de/ezine an und erhalten Sie zusätzlich ein kostenloses Raspberry Pi Projektbuch!

Was können Sie erwarten?

Redaktioneller Elektor-Newsletter
Jeden Freitag erhalten Sie die wichtigsten Artikel und Projekte der Woche. Wir zeigen MCU-basierte Projekte, IoT, Programmierung, KI und mehr!

Elektor-Newsletter mit exklusiven Angeboten
Verpassen Sie nicht unsere Shop-Angebote, jeden Dienstag und Donnerstag haben wir eine besondere Aktion für Sie.

Mailing von externen Partnern
Sie wollen über die laufenden Aktivitäten in der Branche informiert bleiben? Dann gibt Ihnen diese E-Mail die besten Einblicke. Unregelmäßig, aber immer mittwochs.

elektor
design > share > sell

IMPRESSUM

53. Jahrgang, Nr. 584
März/April 2022
ISSN 0932-5468

Erscheinungsweise: 8x jährlich

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 01.01.2022.

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

Druck
Senefelder Misset, Doetinchem (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.

© 2022 elektor international media b.v.

EDITORIAL

Von Jens Nickel

Chefredakteur ElektorMag

Aufgeschoben ist nicht aufgehoben

Das Frühjahr eines Elektronik-Profis wird traditionell von der Messe *embedded world* geprägt: Etwas familiärer als die Riesenmesse *electronica*, aber vielleicht gerade deshalb ein Pflichttermin für alle, die beruflich mit Mikrocontrollern und Software zu tun haben (und welcher Entwickler hat das heutzutage nicht mehr). Ich hätte Ihnen gern geschrieben, dass Sie uns auch in diesem Jahr am Elektor-Stand besuchen können, aber Corona hat uns schon wieder einen Strich durch die Rechnung gemacht. Doch dieses Jahr heißt es zum Glück: „Aufgeschoben ist nicht aufgehoben". Mit dem Ersatztermin am 21. bis 23. Juni wagt die Messe Nürnberg einen zweiten Anlauf - wir klopfen auf Holz!
Mit Blick auf die Messe hatten wir das Thema „Embedded Development" für diese zweite Ausgabe des Jahres ausgesucht; das konnten und wollten wir nicht mehr ändern. Doch werden wir für die nächste Ausgabe (Mai/Juni) das Thema „Internet of Things" vorziehen - dort finden Sie dann auch Artikel über neue Produkte und Trends, die in Nürnberg vorgestellt werden.
Dies nur vorab für Sie als Info - denn diese Embedded-Ausgabe haben wir (so oder so) vollgepackt mit Projekten und Hintergrundartikeln rund um die kleinen, rechnenden Chips. Auf Seite 52 berichtet mein Kollege Stuart Cording über zwei neue Trends in der Embedded-Welt, die sichere Programmiersprache Rust und das Management von Firmware à la Toit. Mathias Claußen aus dem Elektor-Labor zeigt ab Seite 6, wie sich ein eigener RISC-V-Prozessor „bauen" und betreiben lässt. Für Einsteiger gibt es eine Zusammenstellung von preiswerten Boards auf Basis des neuen RP2040 von Raspberry Pi, außerdem einen kleinen Workshop zum Seriellen Plotter der Arduino-IDE. Dazu kommen Projekte wie ein Buffer-Board für den Raspberry Pi, ein berührungsloser Lichtschalter und ein Drahtlos/Seriell-Interface.

Entwickeln Sie mit!

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: Erik Jansen

DEUTSCHE FACHPRESSE

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

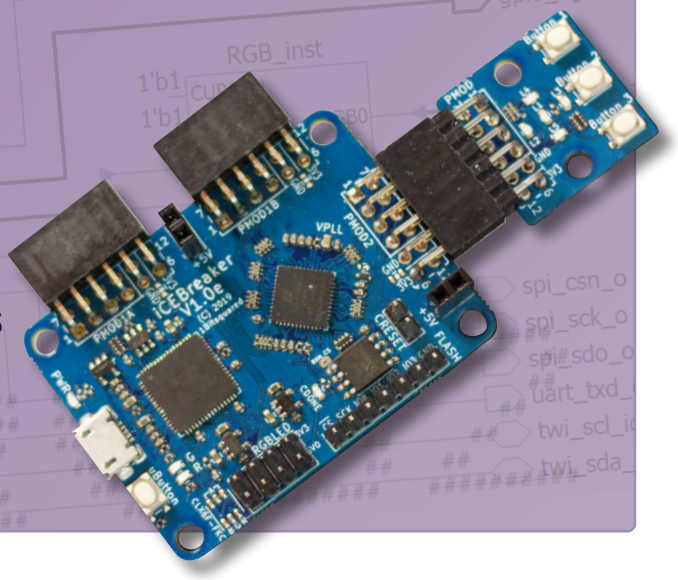
FIPP Connecting Global Media

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

Mein eigener RISC-V Controller

Erste Schritte mit dem NEORV32 RISC-V-Softcore für preiswerte FPGAs

6



Rubriken

3 **Impressum**

37 **Von Entwicklern für Entwickler**
Identifizierung von Bauteilen

49 **Aller Anfang …**
Anpassen und Transformieren

62 **Bemerkenswerte Bauteile**
Drehspul-Relais

66 **Zutritt für Unbefugte verboten**
Alles dreht sich um die Werkzeuge…

84 **Aus dem Leben gegriffen**
Einpacken und weg damit

112 **Ethics**
Die drei Fragezeichen

114 **Hexadoku**
Sudoku für Elektroniker

Hintergrund

FOCUS
15 **Wie man den Seriellen Plotter der Arduino-IDE nutzt**
Einfaches Plotten von Kurven mit Arduino

18 **CLUE - ein Clou von Adafruit?**
Angriff auf den micro:bit

FOCUS
28 **Frühjahrskollektion der Raspberry Pi RP2040 Boards**

32 **Ein DIY-Handbuch zu elektronischer Sicherheit und E-Spionage**
SRAM erhitzt oder tiefgefroren

68 **Neuronen in neuronalen Netzen verstehen**
Teil 4: Eingebettete Neuronen

FOCUS
86 **Unter dem Radar**
Mikrocontroller, über die Sie Bescheid wissen sollten

101 **Lithium-Akkus reparieren**
Geld sparen, Power gewinnen!

106 **GUIs mit Python**
Meme-Generator

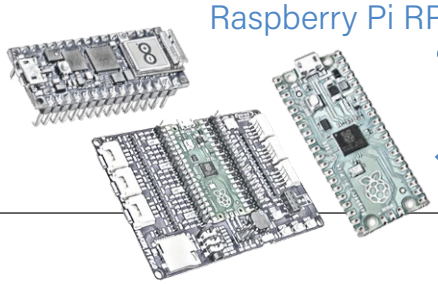
Industry

FOCUS
52 **Was gibt's Neues in der Embedded-Entwicklung?**
Rust und die Aktualisierung von IoT-Implementierungen

FOCUS
58 **Infografiken**
Anschauliche Fakten über den Embedded-Markt

60 **Neue Wege für Industrie und Automobilbranche mit 5G**

Frühjahrskollektion
Raspberry Pi RP2040 Boards



28

Projekte

FOCUS
6 Mein eigener RISC-V-Controller
Erste Schritte mit dem NEORV32 RISC-V Softcore für preiswerte FPGAs

FOCUS
24 Buffer Board für den Raspberry Pi 400
Schützen Sie die I/Os!

FOCUS
42 Ein Touch-Schalter ohne Touch

74 Magnetische Levitation auf die sehr einfache Art
Die dritte und kompakteste Version

FOCUS
79 SPS-Programmierung mit dem Raspberry Pi und dem OpenPLC-Projekt
Visualisierung von SPS-Programmen mit AdvancedHMI

FOCUS
91 Drahtloses Monitoring und Debuggen
Serielle Funk-Schnittstelle für Arduino, ESP32 & Co.

96 Tragbares Temperatur- und Feuchtemessgerät
Gebaut mit vorgefertigten Modulen

Vorschau

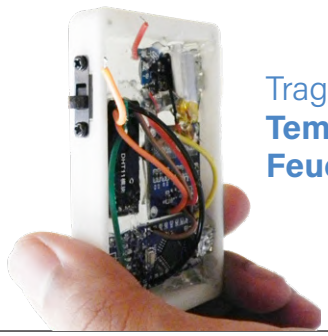
Elektor Mai/Juni 2022
Das nächste Heft ist wie immer randvoll gefüllt mit Schaltungsprojekten, Grundlagen sowie Tipps und Tricks für Elektroniker.

Aus dem Inhalt:

- Erste Schritte mit dem ESP32-C3 im IoT
- Dualer Geiger-Müller-Zähler für Arduino
- CO2-Meter im Netzwerk
- Hochpräziser Lichtschalter DeLux
- IoT-Cloud à la Arduino
- Design-Tools für analoge Filter
- WinUI 3: Neues Grafisches Framework für Windows-Apps
- NB-IoT in der Praxis

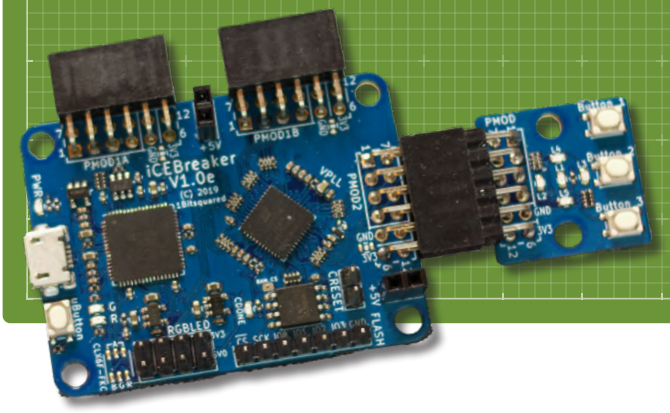
Und vieles mehr!

Elektor Mai/Juni 2022 erscheint am 5. Mai 2022.
Änderungen vorbehalten.



Mein eigener RISC-V-Controller

Erste Schritte mit dem NEORV32 RISC-V Softcore für preiswerte FPGAs



Von Mathias Claußen (Elektor)

Möchten Sie mit RISC-V experimentieren? Dazu bedarf es keines dedizierten Controllers, es ist auch mit einem preiswerten FPGA mit dem NEORV32-RISC-V-Softcore möglich!

Wer mit RISC-V-Mikrocontrollern anfangen möchte, hat inzwischen die Wahl zwischen einer Vielzahl von Prozessoren, wie zum Beispiel dem neuen ESP32-C3. Doch es muss nicht immer ein fest verdrahteter Chip sein: Das NEORV32-Projekt bietet einen RISC-V-Softcore für FPGAs. Das Ganze ist etwas weniger performant, aber deutlich flexibler, denn es lassen sich unterschiedliche Designs testen und selbst entwickelte Peripherie einbinden. Nebenbei lernt man eine Menge, zum Beispiel über das Innenleben einer CPU.

Auch wer sich schon mit FPGAs auskennt, wird beim Erschaffen eines eigenen kleinen Prozessor-Systems schnell auf Hürden stoßen. Das Ganze stellt auch Experten immer wieder vor Herausforderungen, die gemeistert werden müssen, wie unsere Serie über das SCCC-Projekt von Martin Oßmann [1] zeigt.

Doch man muss nicht bei Null anfangen, sondern kann auch fast schlüsselfertige Lösungen verwenden. Eine solche, noch dazu unter Open-Source-Lizenz stehende Lösung soll hier vorgestellt werden. Sicher eine gute Gelegenheit, das eine oder andere herumliegende FPGA-Board wieder einmal aus der Ecke zu holen. Dieser Artikel kann natürlich bei weitem kein kompletter RISC-V- oder FPGA-Kurs sein, sondern soll die Hürden beim Einrichten der nötigen Tools und dem Erstellen eines ersten Projektes nehmen.

FPGA, Synthese, Softcore, RISC-V und Compiler

Wer eine klassische Mikrocontrolleranwendung entwickelt, sucht für sein Projekt einen Chip eines Herstellers aus, der einen Satz der benötigten Peripherie enthält. Alle Funktionen sind fest im Chip verankert und können nicht verändert werden. Das ermöglicht kostengünstige Chips mit einer optimierten Performance. Bei einem FPGA ist dies anders.

Ein FPGA selbst besteht aus Logikzellen, den Lookup Tabels (LUT), die flexibel miteinander durch eine Matrix verschaltet werden können. Die Blöcke, die in einer solchen LUT vorhanden sind, werden in **Bild 1** gezeigt. Als Beispiel dient hier ein LUT-4-Element mit vier Eingangssignalen, einer Wahrheitstabelle, einem Flip-Flop und einen Multiplexer am Ausgang. Durch die Wahrheitstabelle kann man so ein beliebiges Logisches Element wie zum Beispiel UND, ODER, NICHT oder EXCLUSIV ODER formen. In Verbindung mit der Matrix innerhalb des FPGA lassen sich aus diesen Bausteinen komplexere Gebilde, wie Speicher, Addierer oder Multiplexer schaffen, die wiederum zu einem noch komplexeren System wie einem Prozessor oder einem kompletten System-on-Chip zusammengefügt werden können. Der FPGA lässt sich mit einer Bausteinkiste vergleichen, in der eine definierte Menge an Bausteinen vorhanden sind, die sich immer wieder zu neuen Formen zusammensetzen lassen.

Damit der FPGA eine bestimmte Funktion ausführen kann, muss er passend konfiguriert werden. Es ist jedoch nicht nötig, jede einzelne LUT von Hand anzulegen und in der Matrix zu verbinden, das ist Aufgabe der Synthesetools für den FPGA. Die gewünschte Funktionalität wird in Sprachen wie Verilog oder VHDL beschrieben. Die Synthesetools verstehen in der Regel beide Beschreibungssprachen. Das Synthesetool kennt die Eigenheiten des FPGA und erstellt aus

der Beschreibungssprache am Ende einen Bitstrom, der zur Konfiguration des FPGA verwendet wird. **Bild 2** zeigt den groben Ablauf einer Synthese für einen FPGA. Die meisten Hersteller von FPGAs bieten kostenfreie Tools an, die in der Regel unter Windows und Linux verwendet werden können. Für einige FPGAs gibt es auch Open-Source-Lösungen, die diese Synthese durchführen können und bevorzugt unter unixoiden Betriebssystemen wie Linux oder MacOS laufen.

Ein FPGA mit genug LUTs kann nicht nur einfache Logikfunktionen abbilden, sondern ganze Prozessoren oder Prozessorsysteme, diese werden ebenfalls mit Verilog oder VHDL beschrieben. Da dieser Prozessorkern jedoch nicht fest im Silizium des FPGA verdrahtet ist, sondern sich seine Funktion oder Verhalten durch Modifikation der Hardwarebeschreibung anpassen lässt, wird das Ganze als Softcore bezeichnet. Solche Softcores gibt es für unterschiedliche Prozessorarchitekturen; teilweise enthalten diese Softcores zusätzlich noch Peripherie wie zum Beispiel Bus-Interfaces und dergleichen.
Mehr und mehr kommt bei Softcores die freie Prozessorarchitektur RISC-V zum Einsatz. Da momentan die Auswahl an fest verdrahten RISC-V MCUs noch überschaubar ist, lassen sich auf diese Weise erste Erfahrungen mit der Architektur sammeln. Man erstellt sich so seine eigene RISC-V-MCU und kann diese testen und studieren.

Falls man RISC-V nutzt, fallen keine Lizenzgebühren an und es müssen auch keine NDA oder anderweitige Lizenzabkommen (wie bei anderen Architekturen) abgeschlossen werden. RISC-V bedeutet auch, dass schon Compiler für C-Quelltext vorhanden sind, unter anderem in Form des GNU C-Compilers (GCC). Damit stehen auch grundlegende Bibliotheken zur Verfügung.

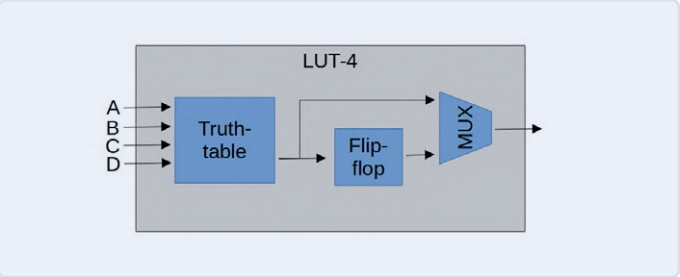


Bild 1. Aufbau eines LUT-4-Elements.

NEORV32

Dass der Weg zum eignen SoC (System on Chip) auf einem preiswerten FPGA nicht schwierig sein muss, zeigt das Projekt NEORV32 von Stephan Nolting [4]. Der NEORV32 ist eine RISC-V-kompatible CPU, die zusätzlich noch Peripherie mitbringt, um als Mikrocontroller-artiges SoC auf einem FPGA zu laufen. Das Projekt ist komplett in plattformneutralen VHDL realisiert und damit nicht an einzelne FPGA-Hersteller gebunden. Der NEORV32 ist nicht nur vollständig open-source, sondern bringt dazu noch eine umfangreiche Dokumentation, ein Software-Framework und Tools mit.

Die realisierten Peripheriebausteine sieht man in **Bild 3**. SPI, I²C und UARTs sind ebenso vorhanden wie GPIOs, PWM-Einheiten und ein WS2812-Interface. So erhalten Einsteiger und Fortgeschrittene ein komplettes System, das eine komplette Entwicklungsumgebung für

Bild 2. Grober Ablauf einer Synthese.

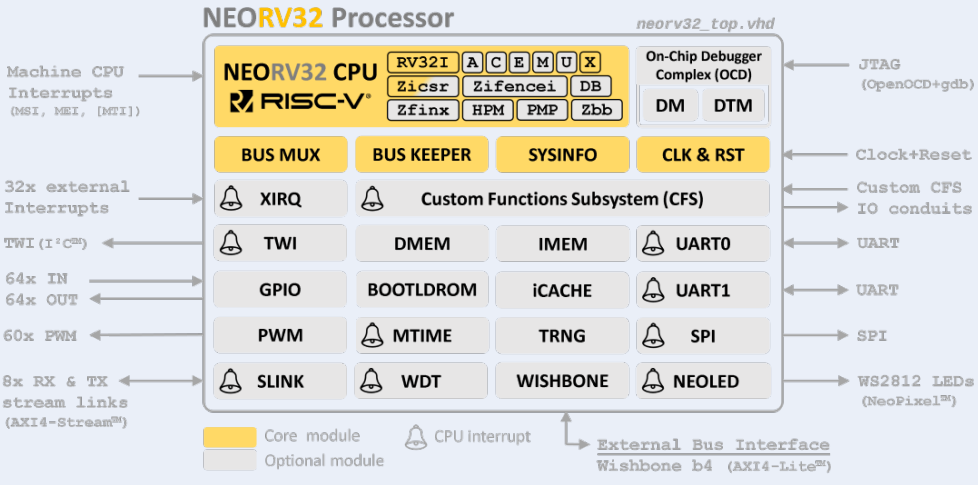


Bild 3. Übersicht der NEORV32-Funktionsblöcke (Quelle: Github / Nolting, S. [20]).

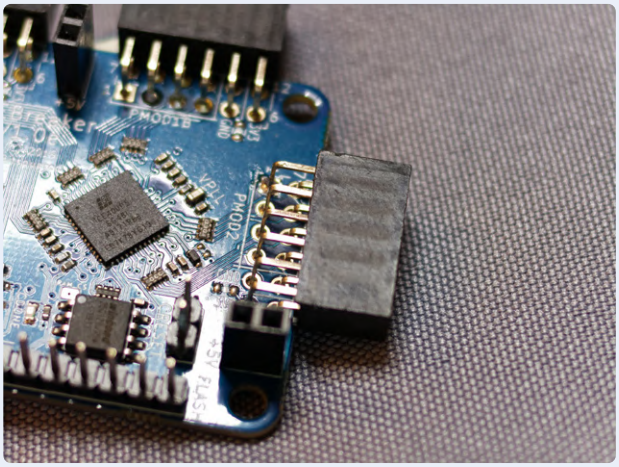


Bild 4. iCE40UP5K als QFN mit 7x7 mm Kantenlänge.

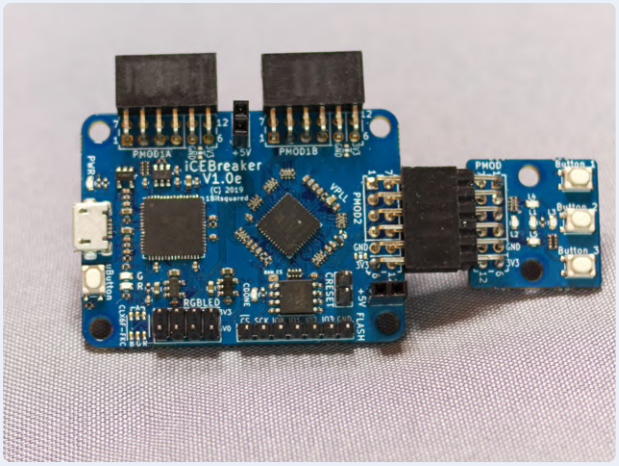


Bild 5. iCEBreaker-Board mit PMOD-Header.

den NEORV32 inklusive der nötigen Bibliotheken für die Hardware und Peripherie mitbringt. Zusätzlich sind für einige FPGA-Boards schon Beispielkonfigurationen vorhanden, so dass der Einstieg ohne größere Probleme starten kann. Doch welche Boards laufen „Out-of-the-Box" und wie schwierig ist es, den NEORV32 auf ein FPGA-Board zu bekommen, das nicht direkt unterstützt wird?

FPGA-Auswahl

Grundsätzlich kann jedes vorhandene Board mit FPGA genutzt werden, das genug Ressourcen bietet, da der NEORV32 keine herstellerspezifischen Erweiterungen nutzt. Ein FPGA, das auf mehreren preiswerten Boards zu finden ist, ist das Lattice iCE40UP5K [5].
Beim Lattice iCE40UP5K FPGA handelt es sich um die größte Ausbaustufe der iCE40 Ultra-Plus-Varianten. 5280 LUTs, 120 kBit (15 kByte) EBR RAM, 1024 kBit (128 kByte) SPRAM und fest verdrahtete Funktionseinheiten für SPI und I²C bilden eine solide Grundlage für die ersten eigenen Projekte. Nicht nur die Ausstattung des FPGAs macht dieses für eigene Projekte interessant, sondern auch das Package und der Preis. Ein QFN-48 mit 7x7 mm Kantenlänge (**Bild 4**) ist deutlich besser zu handhaben als ein Chip in BGA-Ausführung und mit etwa 5 € pro Chip liegt das Ganze auch noch in einem interessanten Preisbereich. Aktuell (Oktober 2021) ist das FPGA mit etwa 5 € bis 6 € pro Chip bei den meisten Distributoren leider ohne Bestand gelistet, mit Lieferzeiten von bis zu 46 Wochen.

Doch man muss nicht selbst ein passendes Board mit einem FPGA designen. Das iCEBreaker FPGA-Board (**Bild 5**) und das iCEBreaker Bitsy (**Bild 6**) von 1BitSquard [6] sind zwei Open-Hardware-Boards, auf dem das iCE40UP5K FPGA verbaut ist. Eine weitere Option und ebenfalls offene Hardware ist das UPduino V3.0 [7] von tinyVision.ai (**Bild 7**).
Das NEORV32-Projekt unterstützt das UPduino V3.0 direkt und bringt Beispielprojekte mit. Eine Anpassung für das iCEBreaker FPGA-Board ist recht schnell durchzuführen. Es wird hier erst einmal das UPduino V3.0 verwendet und anschließend gezeigt, was an einem Beispielprojekt angepasst werden muss, damit dieses auf dem iCEBreaker Board lauffähig ist.

In jedem Fall bekommt man in dieses FPGA ein SoC mit 64 kB Platz für Anwendungen, 64 kB RAM, SPI-Interface, I²C-Interface, 4 Input- und 4 Output-Pins, 3 PWM-Pins, ein UART; nicht zu vergessen natürlich der RV32IMAC-Kern mit 18 MHz Takt.

Toolchain

Bei der Toolchain für den Lattice iCE40up5k gibt es zwei Wege, die eingeschlagen werden können, die Nutzung der Tools von Lattice [8] und die Verwendung der OSS CAD Suite von YosysHQ [9], eine Toolchain, die komplett auf Open-Source-Tools basiert. Für dieses Projekt wird der Open-Source-Weg eingeschlagen. In diesem Fall bedeutet dies Ubuntu 20.04 LTS als Betriebssystem und die OSS CAD Suite, um den NEORV32 für den iCE40UP5K zu synthetisieren.

Neben den Tools für den FPGA soll später auch noch ein Demo-Programm für den synthetisierten RISC-V Prozessor übersetzt werden: Eine klassische Ausgabe von „Hello World" über den UART. Auch hier werden Open-Source-Tools eingesetzt und eine passende Toolchain (ähnlich wie schon für den Kendryte K210 [10]) generiert. Damit stehen dann ein GNU C-Compiler (GCC) und zusätzlich passende Bibliotheken für die Peripherie des NEORV32 bereit.

Mise en Place

Bei der Arbeit mit FPGAs ist es wie beim Kochen, wie mein Kollege Clemens Valens schon in seinem Video demonstriert hat [11]. Die Vorbereitung der Werkzeuge und Zutaten ist der erste Schritt, der zügig erledigt werden kann. Wer seine eigene Betriebssysteminstallation nicht beschädigen möchte, kann auch eine Virtuelle Maschine benutzen; hier wird von einem frisch installierten Ubuntu 20.04 auf einem AMD64-System ausgegangen. Die Verwendung eines Raspberry Pi sollte möglich sein, da sowohl Ubuntu 20.04 wie auch das Raspberry Pi OS auf Debian basieren, es kann aber zu kleinen Unterschieden durch die Architektur kommen. Ich selbst habe das Setup nur auf einem Ubuntu 20.04 und einer AMD64-Maschine getestet.

Um die „Hardware" für den FPGA zu synthetisieren, wird das aktuelle Release der OSS CAD Suite [12] in den Home-Ordner heruntergeladen, die Datei sollte *oss-cad-suite-linux-x64-xxxxxxxx.tgz* heißen. In einem Terminal wird diese nun mit `tar -xvzf oss-cad-suite-linux-x64-xxxxxxxx.tgz` entpackt und anschließend mit `sudo mv ~/oss-cad-suite /opt/` nach */opt* verschoben. Damit später auf den Ordner zugegriffen werden kann, werden mit einem `chmod 777 /opt/oss-cad-suite -R` die Rechte gesetzt. Es wird noch eine weitere Bibliothek benötigt, *libgnat-9*, die mit `sudo apt install libgnat-9` installiert wird. Damit wären die Tools für den FPGA vorbereitet.

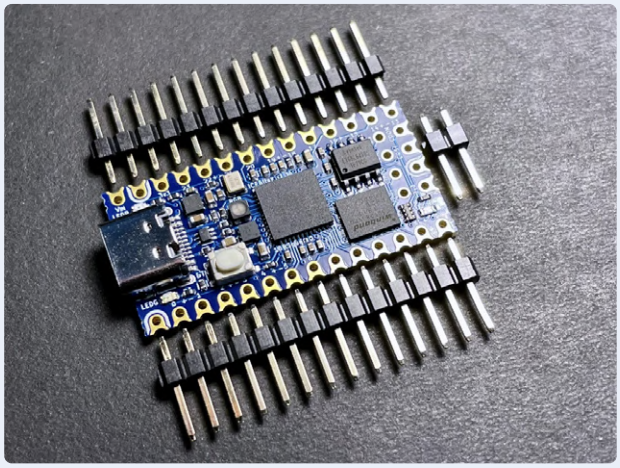


Bild 6. iCEBreaker Bitsy (Quelle: https://cdn.shopify.com/s/files/1/1069/4424/products/IMG_3859_large.jpg / 1BitSquare).

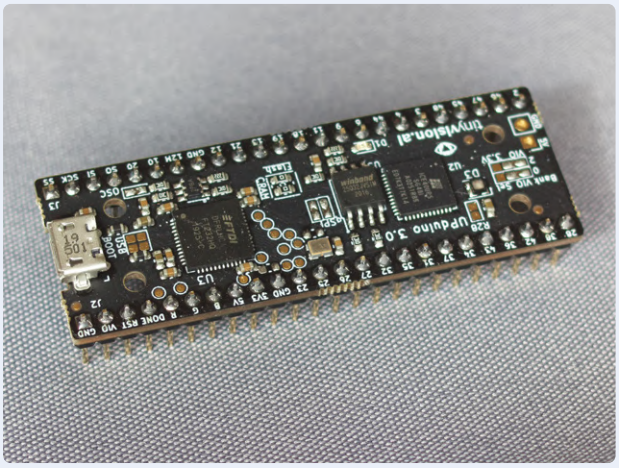


Bild 7. UPduino V3.0 mit Pinheadern.

Compiler

Als Nächstes werden die Daten aus dem GitHub-Repository des NEORV32 [4] benötigt. Dazu muss *git* mit `sudo apt install git` installiert werden. Anschließend wird mit `git clone https://github.com/stnolting/neorv32.git ~/neorv32` das Repository des NEORV32 geklont.
Um später mit dem NEORV32 zu kommunizieren, wird noch ein Terminalprogramm benötigt. Hier hat sich HTerm von Tobias Hammer bewährt, das Programm kann von seiner Homepage [13] mit `wget https://www.der-hammer.info/terminal/hterm-linux-64.tgz` heruntergeladen werden. Mit `mkdir ~/hterm && tar -xvf hterm-linux-64.tgz -C ~/hterm` wird der Inhalt der tgz-Datei im Ordner *~/hterm* entpackt. Da als normaler Benutzer später auf die seriellen Schnittstellen zugegriffen werden soll, muss der aktuelle Benutzer noch der Gruppe *dailout* hinzugefügt werden. Dazu wird in einem Terminal `sudo adduser $(whoami) dialout` eingegeben.

Nun soll auch einmal der C-Compiler selbst übersetzt werden. In den Beispielen für den iCE40up5k ist der NEORV32 als RV32IMAC konfiguriert, damit stehen zum Beispiel Befehle für Integer-Multiplikationen und -Divisionen zur Verfügung (siehe dazu auch den **Kasten RISC-V Namensschema**). Der Compiler muss für diese Architektur und die Befehls-Erweiterungen übersetzt werden; in der Dokumentation des NEORV32 [14] ist nachzulesen, warum diese Übereinstimmung wichtig ist.
In einem Terminal wird dazu zuerst mit cd ~ in den Home-Ordner gewechselt und anschließend mit `git clone https://github.com/riscv/riscv-gnu-toolchain --recursive` die RISC-V Toolchain geklont. Zusätzlich werden noch ein paar Pakete benötigt die mit

```
sudo apt-get install autoconf automake autotools-
   dev curl python3 libmpc-dev libmpfr-dev libgmp-
   dev gawk build-essential bison flex texinfo gperf
   libtool patchutils bc zlib1g-dev libexpat-dev
```

installiert werden müssen. Anschließend steht das Kompilieren des Compilers und der Bibliotheken an.
Bei einer RISC-V CPU gibt es einen kleinsten gemeinsamen Nenner, was die unterstützten Befehle angeht. Das bedeutet, dass Code, der für einen RV32I kompiliert wurde, auch auf einem RV32IMAC lauffähig ist, aber nicht umgekehrt. Daher ist die Empfehlung, erst einmal die Toolchain für den kleinsten gemeinsamen Nenner zu übersetzen. Mit `cd ~/riscv-gnu-toolchain` wird in einem Terminal in das gerade geklonte Repository gewechselt und anschließend mit `./configure --prefix=/opt/riscv --with-arch=rv32i --with-abi=ilp32` die Toolchain für das Kompilieren vorbereitet. Mit sudo make wird das Kompilieren gestartet. Die Toolchain steht danach in */opt/riscv* bereit. Damit jeder auf den Compiler zugreifen kann, müssen noch die Rechte von */opt/riscv* angepasst werden. In einem Terminal wird mit dem Befehl `chmod 777 /opt/riscv -R` jedem der Zugriff gewährt. Für die OSS CAD Suite und für die RISC-V GCC Toolchain muss noch eine Pfadvariable in */etc/enviroment* angepasst werden. Mit `sudo nano /etc/enviroment` wird die Datei zum Editieren geöffnet. Hinter `PATH="` wird `/opt/oss-cad-suite/bin:/opt/riscv/bin:` eingefügt und die Datei gespeichert.

Bei den meisten FPGA-Boards wird ein FT232H-Chip von FTDI zum Programmieren verwendet. Damit ein Benutzer ohne Root-Rechte darauf zugreifen kann, muss noch eine passende udev-Regel für den FTDI-Chip angelegt werden. In einem Terminal wird mit `sudo nano /etc/udev/rules.d/53-lattice-ftdi.rules` eine neue Datei zum Schreiben geöffnet. In dieser Datei muss dann

```
ACTION=="add", ATTR{idVendor}=="0403",
   ATTR{idProduct}=="6010", MODE:="666"
 ACTION=="add", ATTR{idVendor}=="0403",
   ATTR{idProduct}=="6014", MODE:="666"
```

eingefügt und die Datei gespeichert werden. Damit sind die Vorbereitungen abgeschlossen und es kann mit der Synthese und anschließendem Upload eines ersten Testprogramms losgehen. Damit alle Einstellungen wirksam werden, sollte ein Reboot durchgeführt werden. Eine Empfehlung ist noch, einen Editor mit Syntax-Highlighting für VHDL und Verilog zu installieren.

NEORV32 für den FPGA

Im stromlosen Zustand ist der FPGA nicht konfiguriert. Der iCE40UP5K liest, nachdem er mit Spannung versorgt wurde, die Verbindungsbeschreibung von einem externen SPI-Flash. Diese Beschreibung der internen Verbindungen muss dafür in das SPI-Flash auf dem UPDuino V3.0 oder dem iCEBreaker geschrieben werden.
In diesem Artikel erstellen wir ein Beispielprojekt für das UPduino

```
MEMORY
{
/* section base addresses and sizes have to be a multiple of 4 bytes */
/* ram section: first value of LENGTH => data memory used by bootloader (fixed!); second value of LENGTH => *physical* size of data memory */
/* adapt the right-most value to match the *total physical data memory size* of your setup */

  ram  (rwx) : ORIGIN = 0x80000000, LENGTH = DEFINED(make_bootloader) ? 512 : 8*1024

/* rom and iodev sections should NOT be modified by the user at all! */
/* rom section: first value of ORIGIN/LENGTH => bootloader ROM; second value of ORIGIN/LENGTH => maximum *logical* size of instruction memory */

  rom   (rx) : ORIGIN = DEFINED(make_bootloader) ? 0xFFFF0000 : 0x00000000, LENGTH = DEFINED(make_bootloader) ? 32K : 2048M
  iodev (rw) : ORIGIN = 0xFFFFFE00, LENGTH = 512

}
/* ************************************************************************** */
```

Bild 8. Anpassungen für die konfigurierte RAM-Größe.

```
user@user-VirtualBox:~/neorv32/sw/example/hello_world$ make exe
Memory utilization:
   text    data     bss     dec     hex filename
   5576       0     116    5692    163c main.elf
Executable (neorv32_exe.bin) size in bytes:
5588
user@user-VirtualBox:~/neorv32/sw/example/hello_world$
```

*Bild 9. **NEORV32_exe.bin** erstellt.*

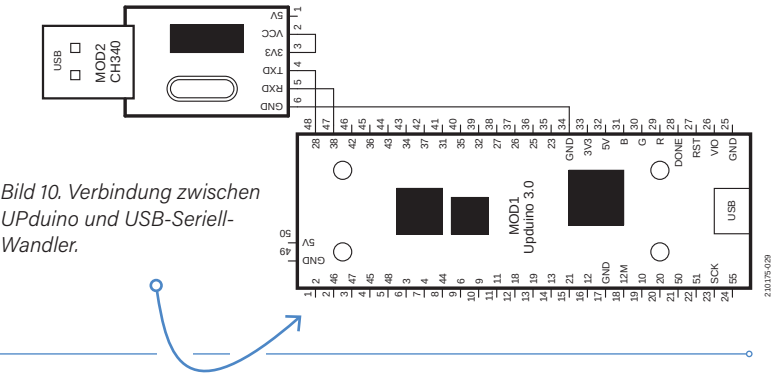


Bild 10. Verbindung zwischen UPduino und USB-Seriell-Wandler.

V3.0 Board, das aus dem Prozessor samt Peripherie besteht. Damit entsteht ein System, das auf dem FPGA direkt lauffähig sein sollte.

In einem Terminal muss nun in das Beispielverzeichnis für die Open Source Tools mit `cd ~/neorv32/setups/osflow/` gewechselt werden. Um nun die Synthese und damit die Erstellung des Bitstroms für den FPGA zu beginnen, reicht es aus, `make BOARD=UPduino UP5KDemo` als Befehl einzugeben, danach kann es ein wenig dauern, bis die Synthese abgeschlossen ist. Im Ordner *~/neorv32/setups/osflow/* ist nun unter anderem eine *neorv32_UPduino_v3_UP5KDemo.bit* Datei entstanden: Der Bitstrom beschreibt, wie im FPGA die logischen Grundbausteine verschaltet werden müssen. Dieser Bitstrom muss nun in das SPI-Flash auf dem FPGA-Board geschrieben werden.

Das UPduino-Board muss per USB mit dem PC verbunden werden; im Terminal müssen wir `iceprog ~/neorv32/setups/osflow/neorv32_UPduino_v3_UP5KDemo.bit` eingeben. Damit startet das Programmieren des externen SPI-Flashs und die Konfiguration wird anschließend in den FPGA geladen. Damit steht im UPduino V3.0 nun ein RISC-V System bereit, das nun mit Software versorgt werden kann.

Wir eingangs schon erwähnt, sind 64 kB für Anwendungen vorhanden und 64 kB stehen als RAM bereit. An Peripherie gibt es eine SPI-Schnittstelle, I²C, einen UART, vier Eingänge und vier Ausgänge, sowie drei PWM-Ausgänge. Die CPU, die hier synthetisiert wurde, ist ein RV32IMAC, der mit 18 MHz laufen wird. Das SoC in dem FPGA verfügt auch über einen kleinen Bootloader, der per UART bedient werden kann.

Hallo Welt

Da der FPGA nun mit dem NEORV32 ausgestattet ist, kann die erste *Hello World* Demo für den RISC-V kompiliert und hochgeladen werden. Da der NEORV32 konfigurierbar ist, muss im Linker-Skript die aktuelle Größe des RAM passend konfiguriert werden. Dazu wird in einem Terminal `nano ~/neorv32/sw/common/neorv32.ld` eingegeben und in der Zeile 62

```
ram (rwx) : ORIGIN = 0x80000000, LENGTH =
   DEFINED(make_bootloader) ? 512 : 8*1024
```

gegen

```
ram (rwx) : ORIGIN = 0x80000000, LENGTH =
   DEFINED(make_bootloader) ? 512 : 64*1024
```

ausgetauscht (**Bild 8**). Nun ist der RISC-V Compiler schon einsatzbereit. In einem geöffneten Terminal kommt man mit `cd ~/neorv32/sw/example/hello_world` in den Ordner des *Hello World* Programms. Um eine Executable (ausführbare Datei) zu erhalten, die in den NEORV32 geladen werden kann, reicht es aus, `make exe` einzugeben, damit *neorv32_exe.bin* generiert wird (**Bild 9**). Damit der NEORV32 das Programm ausführen kann, muss dieses hochgeladen werden. Dazu wird der integrierte Bootloader genutzt, der über den UART Daten entgegennimmt (19200 Baud, 8 Datenbits, 1 Stoppbit, keine Parität, kein Flusskontrolle). Wenn ein UPduino V3.0 verwendet wird, ist noch ein zusätzlicher USB-Seriell-Wandler nötig, so wie der CH340-basierende aus dem Elektor Store (siehe **Kasten Passende Produkte**). Dieser muss wie in **Bild 10** zu sehen verbunden werden.

Für den Upload selbst wird HTerm genutzt. Dieses kann aus einem Terminal mit `~/hterm/hterm` gestartet werden; es sollte ein Fenster wie in **Bild 11** erscheinen. Als Port muss nun der USB-Seriell-Wandler ausgewählt werden, der sich in der Regel als */dev/ttyUSB0* meldet; je nach Hardwarekonfiguration und gewähltem Adapter kann dies aber unterschiedlich sein.

Nach dem Verbinden mit dem USB-Seriell-Adapter kann der UPDuino mit Spannung versorgt werden und es sollte die Meldung des Bootloaders zu sehen sein (**Bild 12**). Wird nicht rechtzeitig ein Zeichen an den Bootloader gesendet, versucht dieser ein Autoboot vom SPI-Flash, das im Moment noch keine Software beinhaltet. Es reicht aus, nach dem Start des Bootloaders innerhalb von 8 Sekunden ein beliebiges Zeichen zu senden, um in den Kommandomodus zu gelangen. Anschließend muss ein u gesendet werden, um den Upload in Bootloader zu aktivieren, und in HTerm die Schaltfläche *Select File* angeklickt werden. Wie in

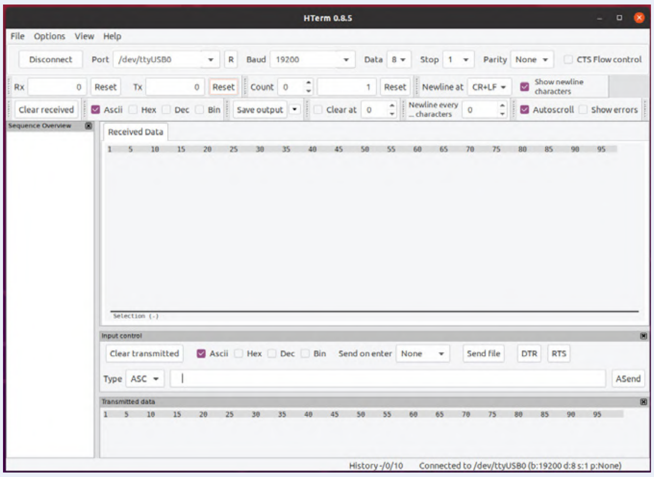


Bild 11. Geöffnetes HTerm-Fenster.

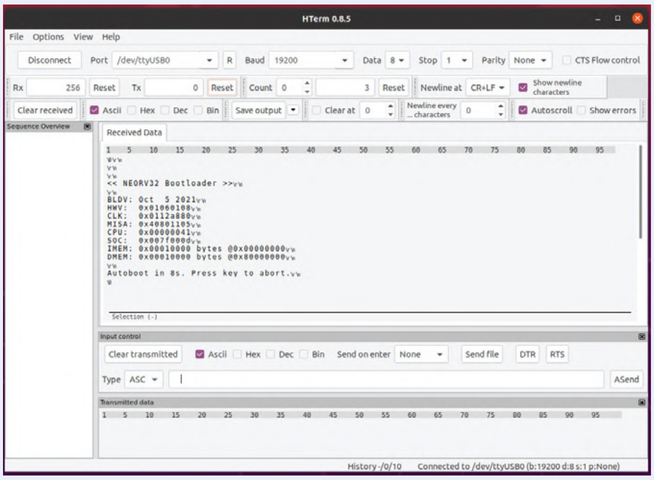


Bild 12. Bootloader des NEORV32.

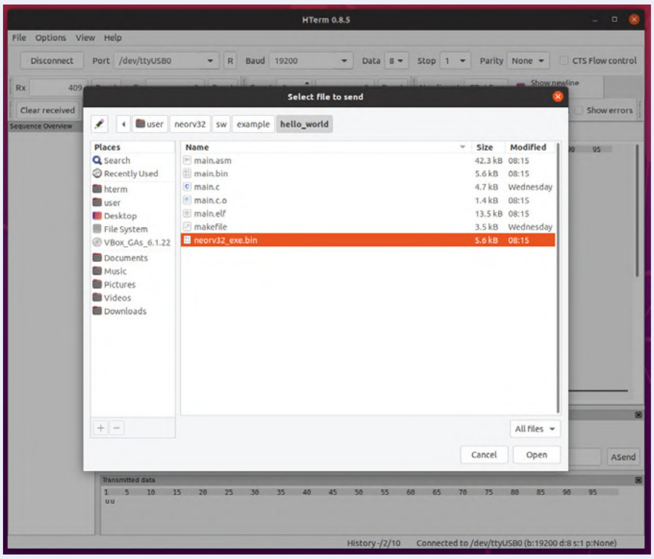


Bild 13. Dialog für den Upload der **neorv32_exe.bin**.

```
Available CMDs:\r\n
 h: Help\r\n
 r: Restart\r\n
 u: Upload\r\n
 s: Store to flash\r\n
 l: Load from flash\r\n
 e: Execute\r\n
CMD:> u\r\n
Awaiting neorv32 exe.bin... OK\r\n
CMD:>
```

Bild 14. Erfolgreicher Upload.

```
<< NEORV32 Bootloader >>\r\n
\r\n
BLDV: Oct  5 2021\r\n
HWV:  0x01060108\r\n
CLK:  0x0112a880\r\n
MISA: 0x40801105\r\n
CPU:  0x00000041\r\n
SOC:  0x007f000d\r\n
IMEM: 0x00010000 bytes @0x00000000\r\n
DMEM: 0x00010000 bytes @0x80000000\r\n
\r\n
Autoboot in 8s. Press key to abort.\r\n
Aborted.\r\n
\r\n
Available CMDs:\r\n
 h: Help\r\n
 r: Restart\r\n
 u: Upload\r\n
 s: Store to flash\r\n
 l: Load from flash\r\n
 e: Execute\r\n
CMD:> u\r\n
Awaiting neorv32 exe.bin... OK\r\n
CMD:> e\r\n
Booting...\r\n
\r\n
\r\n
                                                                             ##                          \r\n
                                                                             ##      ##   ##   ##        \r\n
 ##     ##   #########   ########    ########   ##      ##   ########    ########    ##    ###############  \r\n
####    ##  ##          ##      ##  ##      ##  ##      ##  ##      ##  ##      ##   ##  ####           ####\r\n
## ##   ##  ##          ##      ##  ##      ##  ##      ##          ##        ##     ##    ##   ######   ## \r\n
##  ##  ##  #########   ##      ##  #########   ##      ##      #####       ##       ##  ####   ######   ####\r\n
##   ## ##  ##          ##      ##  ##    ##     ##    ##           ##    ##         ##    ##   ######   ## \r\n
##    ####  ##          ##      ##  ##     ##     ##  ##    ##      ##  ##           ##  ####           ####\r\n
##     ##    #########   ########   ##      ##      ##       ########   ##########   ##    ###############  \r\n
                                                                             ##      ##   ##   ##        \r\n
                                                                             ##                          \r\n

Hello world! :)\r\n
```

Bild 15. „Hello World" vom NEORV32.

RISC-V Namensschema

RISC-V ist ein Oberbegriff für unterschiedliche Architektur-Varianten. Unser Kollege Stuart Cording hat einen schönen Artikel [2] verfasst, in dem die Details genauer erklärt werden. RISC-V beschreibt eine ISA, bei dem die Prozessoren als 32-, 64- oder 128-Bit-Ausführungen definiert sind. Ein 32-Bit-Prozessor hat eine Bezeichnung, die mit RV32 für 32 Bit anfängt, so wie ein RV64 auf einen 64-Bit-Prozessor hinweist. Hinzu kommen nach RV32 oder RV64 noch eine Reihe von Buchstaben, die angeben, welche Befehle und Erweiterungen der Prozessor beherrscht. Diese Buchstaben reichen von A bis Z; ihre Bedeutung kann in der aktuellen RISC-V-Spezifikation [3] nachgeschlagen werden. Je nach den unterstützten Befehlen und Erweiterungen unterscheidet sich die Leistungsfähigkeit der Prozessoren.

Bild 13 zu sehen, ist die Datei *neorv32_exe.bin* im Ordner *~/neorv32/sw/example/hello_world* auszuwählen und anschließend hochzuladen. Wenn alles beendet ist, meldet der Bootloader wie in **Bild 14** ein `OK` und das Programm kann mit e gestartet werden.
Das Ergebnis ist in **Bild 15** zu sehen. Das Programm wird nicht in das SPI-Flash geschrieben, sondern in das RAM-basierte „ROM" des NEORV32. Damit wird die Lebensdauer des SPI-Flashs durch Reduzierung der Schreibvorgänge verlängert. Der Nachteil ist so jedoch, dass bei jedem Neustart des NEORV32 auch wieder ein Upload des Programms nötig ist. Soll das Programm automatisch geladen werden, so muss durch den Bootloader ein Upload in das SPI-Flash erfolgen. Neben der „Hello World"-Demo gibt es noch weitere Beispiele zu entdecken, bis zu einem kompletten FreeRTOS, das auch schon Thema [15] in Elektor war. Dazu lohnt ein Blick in den Ordner *~/neorv32/sw/example*, in dem jedes Beispiel in einem eigenen Ordner zu finden ist.

Ein neues iCE40UP5K-Board hinzufügen

Das iCEBreaker Board soll hier demonstrieren, wie man den NEORV32 auch an andere iCE40up5k-Boards anpassen kann. Die Features des SoCs selbst werden hier nicht verändert, aber das Pinout am FPGA so angepasst, das ein iCEBreaker-Board verwendet und ein passender Bitstrom generiert werden kann.
Dazu muss das Makefile in *~/neorv32/setup/osflow* angepasst werden. Die Datei wird mit einem Texteditor der Wahl geöffnet und das neue Board als Target hinzugefügt. Für den iCEBreaker schreiben wir in Zeile 72 folgendes:

```
iCEBreaker:
$(MAKE) \
BITSTREAM=neorv32_$(BOARD)_$(DESIGN).bit \
NEORV32_MEM_SRC="devices/ice40/neorv32_imem.ice40up_
   spram.vhd devices/ice40/neorv32_dmem.ice40up_
   spram.vhd" \
run
```

Damit ist im primären Makefile schon einmal das Board bekannt, jedoch muss nun auch in *~/neorv32/setup/osflow/boards* eine *iCEBreaker.mk* mit folgendem Inhalt angelegt werden:

```
.PHONY: all

all: bit
echo "! Built $(IMPL) for $(BOARD)"
```

Damit wären die Makefiles vorbereitet, jedoch fehlen noch zwei VHDL-Dateien. Unter *~/neorv32/setup/osflow/board_tops/* muss eine *neorv32_iCEBreaker_BoardTop_UP5KDemo.vhd* und eine *neorv32_iCEBreaker_BoardTop_MinimalBoot.vhd* erzeugt werden. Da der Inhalt fast mit dem der *neorv32_UPduino_BoardTop_UP5KDemo.vhd* und *neorv32_UPduino_BoardTop_MinimalBoot.vhd* identisch ist, können diese Dateien kopiert und umbenannt werden. Dazu sollte in einem Terminal

```
cp ~/neorv32/setups/osflow/board_tops/neorv32_UPduino_
BoardTop_MinimalBoot.vhd ~/neorv32/setups/osflow/board_
tops/neorv32_iCEBreaker_BoardTop_MinimalBoot.vhd
```

und

```
cp ~/neorv32/setups/osflow/board_tops/neorv32_UPduino_
BoardTop_UP5KDemo.vhd ~/neorv32/setups/osflow/board_tops/
neorv32_iCEBreaker_BoardTop_UP5KDemo.vhd
```

eingegeben werden. In den beiden Dateien *neorv32_iCEBreaker_BoardTop_UP5KDemo.vhd* und *neorv32_iCEBreaker_BoardTop_MinimalBoot.vhd* müssen nun noch ein paar Anpassungen durchgeführt werden. In der Datei *neorv32_iCEBreaker_BoardTop_UP5KDemo.vhd* muss die Zeile 42 in `entity neorv32_iCEBreaker_BoardTop_UP5KDemo is` und Zeile 68 in `architecture neorv32_iCEBreaker_BoardTop_UP5KDemo_rtl of neorv32_iCEBreaker_BoardTop_UP5KDemo is` geändert werden. In der Datei *neorv32_iCEBreaker_BoardTop_MinimalBoot.vhd* ist Zeile 42 in `entity neorv32_iCEBreaker_BoardTop_MinimalBoot is` und Zeile 54 in `architecture neorv32_iCEBreaker_BoardTop_MinimalBoot_rtl of neorv32_iCEBreaker_BoardTop_MinimalBoot is` zu ändern.
Der letzte Schritt ist das Constraints-File *iCEBreaker.pcf,* das unter *~/neorv32/setups/osflow/constraints/* angelegt werden muss. Der Inhalt der Datei ist in **Listing 1** zu sehen. Mit *iCEBreaker.pcf* wird festgelegt, welche Funktion auf welchen Pin des FPGA geroutet werden soll. Damit wird auch der USB-Seriell-Wander, der sich auf dem iCEBreaker befindet, direkt mit eingebunden, sowie die Taster und LEDs auf die IO-Pins des NEORV32 gelegt. Etwas das noch stört, ist der fehlende Reset-Taster; auch wenn dieser schon in der Constraints-Datei definiert ist, hat er hier noch keine Funktion.

Mit den aktuellen Anpassungen kann nun genau wie für den UPduino ein Bitstrom erstellt werden. Dazu muss ein Terminal geöffnet werden und mit `cd ~/neorv32/setups/osflow` in den Ordner *osflow* gewechselt werden. Mit `make BOARD=iCEBeaker UP5KDemo` lässt sich das Erstellen starten. Das Programmieren des Bitsteams in den iCEBreaker erfolgt (wie beim UPduino) anschließend mit `iceprog ~/neorv32/setups/osflow/neorv32_iCEBreaker_UP5KDemo.bit`. Das Programmieren der Software für den NEORV32 geschieht auch hier über den integrierten Bootloader des NEORV32; nur dass hier kein externer USB-Seriell-Wandler verwendet werden muss, sondern der zweite Kanal des auf dem iCEBreaker integrierten Konverters genutzt wird.

Listing 1. iCEBreaker.pcf

```
#UART (uart0)
ldc_set_location -site {9} [get_ports uart_txd_o]
ldc_set_location -site {6} [get_ports uart_rxd_i]

#SPI - on-board flash
ldc_set_location -site {14} [get_ports flash_sdo_o]
ldc_set_location -site {15} [get_ports flash_sck_o]
ldc_set_location -site {16} [get_ports flash_csn_o]
ldc_set_location -site {17} [get_ports flash_sdi_i]

#SPI - user port
ldc_set_location -site {43} [get_ports spi_sdo_o]
ldc_set_location -site {38} [get_ports spi_sck_o]
ldc_set_location -site {34} [get_ports spi_csn_o]
ldc_set_location -site {31} [get_ports spi_sdi_i]

#TWI
ldc_set_location -site {2} [get_ports twi_sda_io]
ldc_set_location -site {4} [get_ports twi_scl_io]

#GPIO - input
ldc_set_location -site {18} [get_ports {gpio_i[0]}]
ldc_set_location -site {19} [get_ports {gpio_i[1]}]
ldc_set_location -site {20} [get_ports {gpio_i[2]}]
ldc_set_location -site {28} [get_ports {gpio_i[3]}]

#GPIO - output
ldc_set_location -site {25} [get_ports {gpio_o[0]}]
ldc_set_location -site {26} [get_ports {gpio_o[1]}]
ldc_set_location -site {27} [get_ports {gpio_o[2]}]
ldc_set_location -site {23} [get_ports {gpio_o[3]}]

#RGB power LED
ldc_set_location -site {39} [get_ports {pwm_o[0]}]
ldc_set_location -site {40} [get_ports {pwm_o[1]}]
ldc_set_location -site {41} [get_ports {pwm_o[2]}]

#Reset
ldc_set_location -site {10} [get_ports
   {user_reset_btn}]
```

Ein Reset-Taster für den NEORV32

Um den NEORV32 neu zu starten musste aktuell immer die Spannungsversorgung kurz getrennt und wieder verbunden werden. Das ist auf Dauer etwas lästig und da der iCEBreaker genug Tasten hat, kann eine davon als Reset benutzt werden. Dazu wird der uButton nahe der Micro-USB Buchse verwendet, welcher an Pin 10 des FPGA geführt ist.

Der NEORV32 hat intern einen Eingang für einen Reset, an dem aber die PLL zur Erzeugung des Prozessortaktes angeschlossen ist. Die sauberste Lösung wäre es, mit etwas Logik das externe Resetsignal und das Resetsignal der PLL korrekt logisch zu verknüpfen und dem NEORV32 bereitzustellen. Die einfachere Lösung ist es, den Reseteingang der PLL zu nutzen. **Bild 16** zeigt die Verbindung des uButton mit diesem Eingang. Wird die PLL resettet, so wird auch der NEORV32 mit resettet, da das *lock_o* Signal den Zustand wechselt, wenn die PLL im Reset ist.

Um das UP5K-Demo-Projekt entsprechend zu erweitern, müssen in der *neorv32_iCEBreaker_BoardTop_UP5KDemo.vhd* in dem Ordner ~/neorv32/setups/board_tops eine Zeile eingefügt und eine Zeile angepasst werden. Ab Zeile 44 wird `user_reset_btn : in std_ulogic;` als eigene Zeile eingefügt. Zeile 130 wird in `RESETB => user_reset_btn,` geändert, das Komma am Ende sollte nicht vergessen werden, da es sonst zu einem Syntaxfehler kommt. Damit wäre der uButton als Reset funktional. Es muss abschließend der Bitstrom einmal mit den aktuellen Anpassungen in der UP5K-Demo neu generiert und in den FPGA des iCEBreaker geladen werden.

Ausblick

Über das Thema RISC-V, FPGA und auch den NEORV32 könnte man mehr als nur ein Buch füllen. Der iCE40UP5K ist für Einsteiger eine preiswerte Wahl, wenn der Chip denn lieferbar wäre. Die beiden im

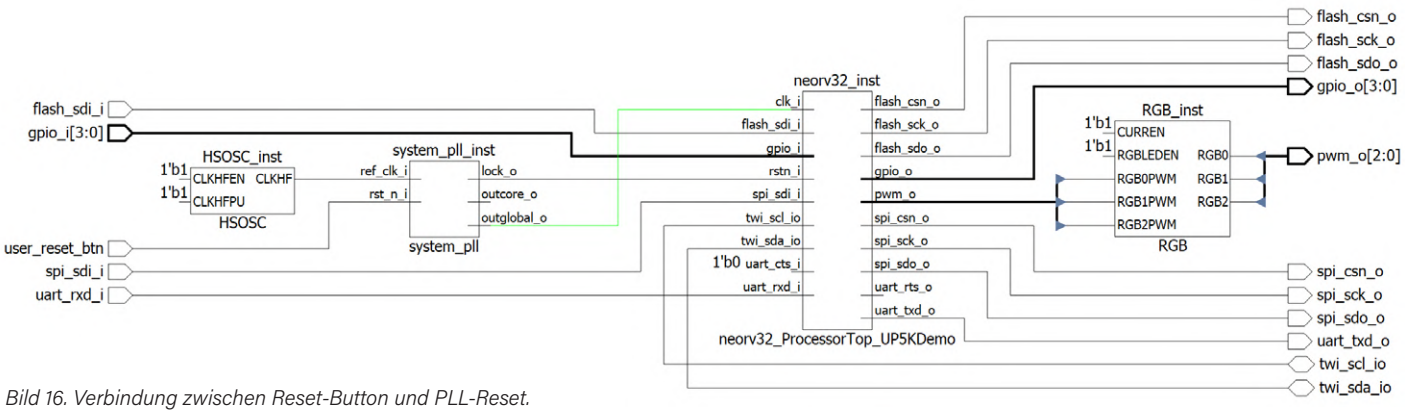


Bild 16. Verbindung zwischen Reset-Button und PLL-Reset.

Artikel vorgestellten Boards sind nicht die einzigen, die diesen FPGA nutzen, es gibt noch mindestens eine Handvoll weiterer Plattformen; vom Raspberry Pi Add-On bis hin zur kompletten Handheld-Konsole ist alles vertreten. Auch die Tools und Projekte, in denen dieser FPGA verwendet werden kann, sind einen Blick wert. Beispiele sind LiteX [16], mit dem das eigene SoC wie in einem Baukasten zusammengestellt werden kann und nicht zwangsweise auf RISC-V beschränkt ist, der RISCboy [17] von Luke Wren, der ein Gameboy-artiges System im FPGA bereitstellt sowie der OK-ice40-PRO [18], eine Gamepad-Konsole. Sollte die Beschaffungssituation besser werden, kommen sicherlich noch weitere Projekte und Ideen für den iCE40UP5K hinzu. Eine Mischung aus Raspberry Pi RP2040 und iCE40UP5K als kleine Spielekonsole scheint schon in der Konzeption zu sein [19].

210175-02

Ein Beitrag von

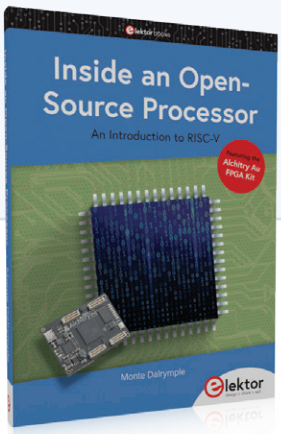
Text und Bilder: **Mathias Claußen**
Redaktion: **Jens Nickel**
Layout: **Harmen Heida**

Sie haben 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

- **CH340 USB/TTL-UART Konverter-Modul CH340G (3,3 V/5,5 V) (SKU 19151)**
  www.elektor.de/19151
- **Alchitry Cu FPGA Development Board (Lattice iCE40 HX) (SKU 19640)**
  www.elektor.de/19640
- **Buch: *Inside an Open-Source Processor* (SKU 19826)**
  www.elektor.de/19826



WEBLINKS

[1] Martin Oßmann, „Das SCCC-Projekt (1)", ElektorMag 3-4/2019: www.elektormagazine.de/magazine/elektor-87/42467
[2] Stuart Cording, „What Is RISC-V?", elektormagazine.com 04/2021: www.elektormagazine.com/articles/what-is-risc-v
[3] RISC-V Spezifikationen: https://riscv.org/technical/specifications/
[4] NEORV32 GitHub Repository: https://github.com/stnolting/neorv32/
[5] Lattice iCE40UltraPlus Produktseite: www.latticesemi.com/en/Products/FPGAandCPLD/iCE40UltraPlus
[6] iCEBreaker GitHub Repository: https://github.com/icebreaker-fpga/icebreaker
[7] UPduino GitHub Repository: https://github.com/tinyvision-ai-inc/UPduino-v3.0
[8] Lattice Radiant: www.latticesemi.com/LatticeRadiant
[9] YosysHQ oss-cad-suite-build : https://github.com/YosysHQ/oss-cad-suite-build
[10] Setup a toolchain for the Kendryte K210, Elektor Labs: www.elektormagazine.de/labs/setup-a-toolchain-for-the-kendryte-k210-1
[11] Linux-flavored Snickerdoodles with Zynq, ElektorTV: www.youtube.com/watch?v=EE4yYZ-FEoQ
[12] YosysHQ oss-cad-suite-build Releases: https://github.com/YosysHQ/oss-cad-suite-build/releases
[13] HTerm: www.der-hammer.info/
[14] NEORV32 - Build the Toolchain from scratch: https://stnolting.github.io/neorv32/ug/#_building_the_toolchain_from_scratch
[15] Warren Gay, „Praktisches ESP32-Multitasking", ElektorMag 1-2/2020: www.elektormagazine.de/magazine/elektor-138/56969
[16] LiteX: https://github.com/enjoy-digital/litex
[17] RISCBoy : https://github.com/Wren6991/RISCBoy
[18] OK-iCE40Pro Handheld: https://github.com/WiFiBoy/OK-iCE40Pro
[19] PicoStation3D: https://github.com/Wren6991/PicoStation3D
[20] Nolting S., The NEORV32 RISC-V-Processor, GitHub repository 2020: https://raw.githubusercontent.com/stnolting/neorv32/master/docs/figures/neorv32_processor.png

Wie man den Seriellen Plotter der Arduino-IDE nutzt

Einfaches Plotten von Kurven mit Arduino

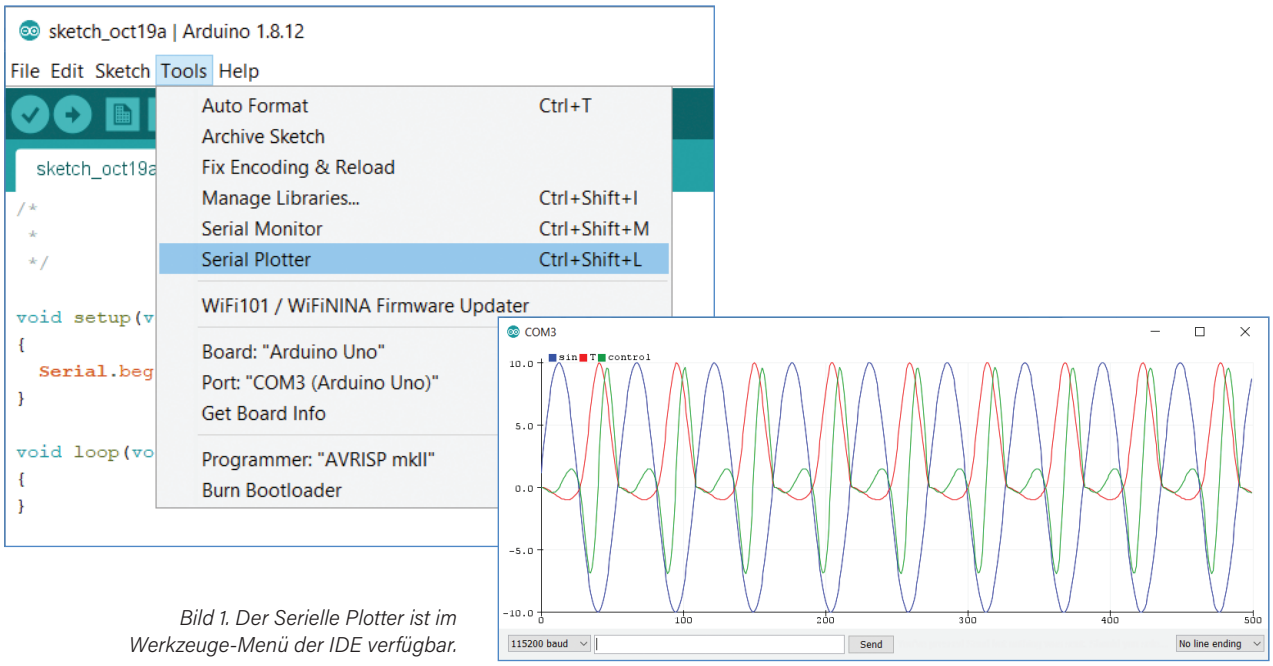


Bild 1. Der Serielle Plotter ist im Werkzeuge-Menü der IDE verfügbar.

Von **Clemens Valens** (Elektor)

Eine der Stärken der Arduino-IDE ist die leicht zu bedienende serielle Schnittstelle, die es einfach macht, Daten zu importieren, Befehle an Arduino zu senden oder bei der Fehlersuche zu helfen. Zur Visualisierung dieser Daten stehen ein Serieller Monitor und ein Serieller Plotter zur Verfügung. Zum Monitor muss man nicht viel sagen, aber wie benutzt man eigentlich den Seriellen Plotter?

Die meisten Arduino-Benutzer werden den in der IDE integrierten Seriellen Monitor kennen, der im Werkzeuge-Menü oder über die Tastenkombination <Strg+Shft+M> erreichbar ist. Er ist ein einfaches serielles Terminal, das alle über die serielle Schnittstelle empfangenen Daten anzeigt und auch das Senden von Zeichenfolgen (beispielsweise Befehle oder Daten) vom Computer zum Arduino-Board ermöglicht. Wenn der Sketch Textnachrichten darüber sendet, was er tut oder nicht tut, ist der Serielle Monitor für die Fehlersuche sehr nützlich.

Der Serielle Plotter

Der Serielle Monitor kann natürlich auch numerische Werte anzeigen. Die Entwicklung eines Parameters in einer langen Werteliste zu verfolgen, ist jedoch nicht immer einfach, weshalb die Arduino-IDE auch über einen Seriellen Plotter verfügt, der, wie **Bild 1** zeigt, direkt unter dem Seriellen Monitor im Werkzeugmenü zu finden ist und auch mit dem Tastaturkürzel <Strg+Shft+L> aufgerufen werden kann.

Der Serielle Plotter erstellt Diagramme der empfangenen numerischen Daten und zeigt diese an. Er ist nicht besonders leistungsstark und besitzt auch nicht viele Optionen, aber er ist einfach zu bedienen. Leider gibt es dafür keine Dokumentation. Sie müssen im Java-Quellcode [1] nachschauen, um herauszufinden, wozu er in der Lage ist. Wenn Sie dies tun, werden Sie die folgenden Funktionen entdecken:

- Unbegrenzte Anzahl von Diagrammen
- Automatische Skalierung der vertikalen oder Y-Achse
- 500 Punkte auf der horizontalen oder X-Achse
- Anzeige der letzten 500 Datenpunkte

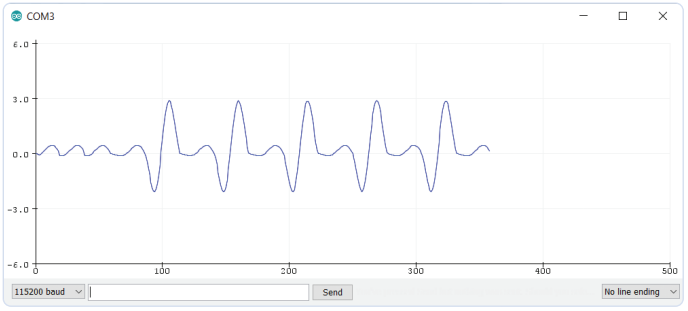


Bild 2. Die horizontale Achse bietet Platz für bis zu 500 Punkten. Wenn Sie mehr senden, beginnt sie zu scrollen.

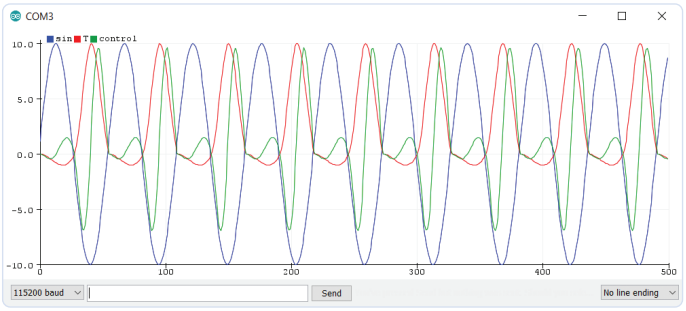


Bild 3. Die Anzahl der Diagramme ist (theoretisch) unbegrenzt. Verwenden Sie Labels, um den Überblick zu behalten, welches Diagramm welche Daten zeigt.

Aufzeichnen eines Diagramms

Wählen Sie nach dem Öffnen des Seriellen Plotters die gleiche Baudrate wie die bei `Serial.begin` im Sketch angegebene Geschwindigkeit. Das Kästchen neben dem Geschwindigkeitsauswahlfeld ist für das Zurücksenden von Daten an den Sketch vorgesehen, genau wie im Seriellen Monitor. Auf diese Weise können Sie den Sketch in gewisser Weise steuern.

Sie müssen keine neuen `Serial.port`-Befehle erlernen, um den Seriellen Plotter zu verwenden, sie sind identisch mit denen des Seriellen Monitors. Der Serielle Plotter ist nur eine weitere Möglichkeit, über die serielle Schnittstelle gesendete Daten zu visualisieren.

Das einzig Wichtige ist die Art und Weise, wie Sie die Daten formatieren. Um ein Diagramm eines einzelnen Parameters (**Bild 2**), etwa eines Sensorwertes zu zeichnen, genügt es, die Werte durch einen Zeilenumbruch zu trennen. Mit anderen Worten: Um die Werte zu senden, verwenden Sie

```
Serial.println(sensor_value);
```

Skalierung außer Kontrolle

Der serielle Plotter verwendet diese Werte für die vertikale Y-Achse und inkrementiert die horizontale X-Achse automatisch. Mit anderen Worten, der erste gesendete Wert wird als y-Wert für x = 0 betrachtet, der zweite Wert ist der y-Wert für x = 1 und so weiter und so fort. Wenn x den Wert 500 erreicht, beginnt das Diagramm horizontal zu scrollen: Neue Punkte werden rechts hinzugefügt, die alten Punkte fallen links aus dem Fenster.

Wenn der y-Wert außerhalb der Grenzen liegt, skaliert die vertikale Achse automatisch nach oben, um alles innerhalb der vertikalen Grenzen zu halten. Das Gleiche gilt umgekehrt: Wenn plötzlich alle Werte in einen kleineren Bereich passen (weil beim Scrollen des Diagramms ein hoher Wert aus dem Fenster fällt), geht die Skalierung nach unten.

Viele Kurven plotten

Um Diagramme für zwei oder mehr Parameter zu zeichnen, senden Sie einfach die Werte für die verschiedenen Diagramme als eine Liste von Werten, die entweder durch ein Komma (,), ein Leerzeichen ( ) oder ein Tabulatorzeichen (\t) getrennt sind, und schließen Sie jede Zeile der Liste mit einem Zeilenumbruch ab:

```
Serial.print(temperature);
Serial.print(',');
Serial.print(humidity);
Serial.print(',');
Serial.print(pressure);
Serial.println();
```

Wenn das Komma als Trennzeichen verwendet wird, wird das Format als CSV bezeichnet, was für durch „comma-separated values“ steht. Die meisten Tabellenkalkulationsprogramme können solche Dateien importieren (manche Programme erlauben sogar die Angabe des Trennzeichens). Wenn also im Seriellen Plotter alles gut aussieht, können Sie mit dem Seriellen Monitor viele dieser Zeilen exportieren und per Copy-Paste in eine Tabellenkalkulation einfügen, um noch hübschere Diagramme zu erstellen und andere Nachbearbeitungen vorzunehmen.

Die Datenwerte der Liste werden als y-Werte für denselben x-Wert interpretiert, so dass alle Diagramme zur gleichen Zeit und mit der gleichen Geschwindigkeit aktualisiert werden. Der Plotter wählt für jede Kurve eine andere Farbe, der Benutzer hat hierüber keine Kontrolle. Die einzige Möglichkeit, dies ein wenig zu beeinflussen, besteht darin, die Reihenfolge der Daten zu ändern.

Labels

Der Serielle Plotter verfügt über eine recht ausgefallene Option: Sie können eine Beschriftung für jedes Diagramm festlegen (**Bild 3**). Es gibt zwei Möglichkeiten, dies zu tun:

1. Bevor Sie Datenwerte senden, senden Sie zunächst eine Liste von Labels, wobei diese Labels entweder durch Kommatas, Leerzeichen oder Tabulatorzeichen getrennt sind und mit einem Zeilenumbruchzeichen enden. Daher dürfen die Labels keines dieser drei Zeichen enthalten. Die Daten müssen dann natürlich in der gleichen Reihenfolge wie die Labels gesendet werden. Beispiel:

   ```
   Serial.println("temperature,humidity,pressure")
   ```

2. Senden Sie Labels und Werte als Paare und trennen Sie beides durch Doppelpunkte (:), also:

   ```
   Serial.println("temperature:21,humidity:76,press
   ure:1007")
   ```

   Achten Sie darauf, diese Paare immer in der gleichen Reihenfolge zu senden, um zu verhindern, dass der Plotter die Daten für

verschiedene Kurven durcheinanderbringt. Eine Liste von Label-Wert-Paaren muss wiederum entweder durch ein Komma, ein Leerzeichen oder ein Tabulator-Zeichen getrennt werden und mit einem Zeilenumbruch-Zeichen enden.

Kleine Warnung: Sobald Sie einen Textstring senden, wird dieser als Label betrachtet. Das ist eine recht bequeme und gerne genutzte Möglichkeit, für heillose Verwirrung zu sorgen! Ebenso vielversprechend für ein Chaos ist es, Trennzeichen zu vergessen, so dass zum Beispiel Daten in Labels landen und umgekehrt.

Macken und Mängel

1. Die horizontale Achse läuft ewig weiter, es gibt keine Möglichkeit, das Diagramm programmgesteuert neu zu starten. Die einzige Möglichkeit dazu besteht darin, den Seriellen Plotter zu schließen und ihn dann wieder zu öffnen. Manchmal muss man das sogar zweimal tun, um veraltete Labels und/oder Daten restlos loszuwerden.
2. Es ist nicht möglich, einen Punkt an einer (x,y)-Position Ihrer Wahl zu plotten; neue Punkte werden immer am Ende hinzugefügt.
3. Die vertikale Achse skaliert automatisch. Das mag nett erscheinen, kann aber sehr ärgerlich werden, wenn es Ausreißer oder Störungen gibt, die den Maßstab nach oben zwingen und es dann schwierig oder sogar unmöglich machen, die viel kleineren, aber interessanten Daten zu sehen. Dies kann auch passieren, wenn eine Datenreihe im Bereich von beispielsweise [-1,+1] und eine andere im Bereich von [-100,+100] liegt. In diesem Fall klebt die Skala fest zwischen -100 und +100.
Einige Leute versuchen, den Plotter zu überlisten, indem sie der Liste der Datenwerte etwas wie `min:-100,max:100` hinzufügen, um die Autoskalierung zu stoppen, aber das funktioniert nur, wenn jeder Wert auch innerhalb dieser Grenzen bleibt (**Bild 4**).
4. Aus Gründen, die nur dem/den Entwickler(n) des Seriellen Plotters bekannt sind, liegt die Mindestskalierung normalerweise bei -6 bis +6, aber auch -5 bis +5 kann vorkommen.

Kleine-Hilfe-Funktion

Hier finden Sie eine kleine Hilfsfunktion, die Sie in einen Sketch einfügen können, um den Seriellen Plotter etwas benutzerfreundlicher zu gestalten und gleichzeitig den Sketch lesbar zu halten:

```
void plot(String label, float value, bool last)
{
  Serial.print(label); // May be an empty string.
  if (label!="") Serial.print(":");
```

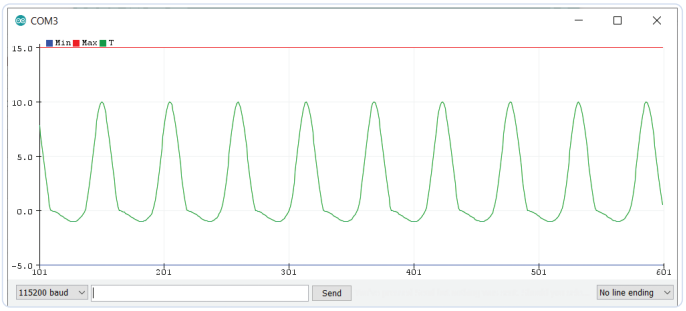


Bild 4. Das Senden konstanter Minimal- und Maximalwerte verhindert die automatische Skalierung der vertikalen Achse, aber nur solange alle anderen Werte innerhalb dieser Grenzen bleiben.

```
  Serial.print(value);
  if (last==false) Serial.print(",");
  else Serial.println();
}
```

Wenn die Zeichenfolge *label* angegeben wird, druckt diese Funktion ein Paar `label:value`. Wenn `label` eine leere Zeichenkette ist, wird nur der Wert ausgegeben. Dies ermöglicht die Ausgabe reiner CSV-formatierter Daten im Seriellen Monitor für die Verwendung in Tabellenkalkulationen.
Beachten Sie, dass Sie den Seriellen Monitor und den Seriellen Plotter nicht gleichzeitig öffnen können.
Nachdem Sie den seriellen Port geöffnet haben, rufen Sie ihn wie folgt auf:

```
// Graph1, more graphs follow.
plot("Temperature",temperature_value,false);
// Graph2, more graphs follow.
plot("Humidity",humidity_value,false);
// Graph3, last graph.
plot("Pressure",pressure_value,true);
```

Denken Sie daran, den Seriellen Plotter jedes Mal neu zu starten (schließen und wieder öffnen), wenn Sie den Sketch ändern, damit alle veralteten Daten und Labels entsorgt werden. Wenn Sie Zweifel haben, wiederholen Sie diesen Vorgang.

200540-02

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

Ein Beitrag von
Idee und Text: **Clemens Valens**
Redaktion: **C. J. Abate**
Layout: **Giel Dols**

WEBLINK

[1] **Quellcode des seriellen Plotters der Arduino-IDE:**
https://github.com/arduino/Arduino/blob/master/app/src/processing/app/SerialPlotter.java

CLUE - ein Clou von Adafruit?

Angriff auf den micro:bit

Von **Tam Hanna**

Der BBC micro:bit war ein großer Erfolg - weit über den edukativen Markt hinaus, für den er gedacht war. Mit einem vollwertigen Display und weit mehr Speicher tritt nun der CLUE von Adafruit an. Dank Bluetooth LE und einer Vielzahl von integrierten Sensoren empfiehlt er sich vor allem für kleinere IoT-Projekte

Spätestens seit dem Erfolg von Arduino und Raspberry Pi ist offensichtlich, dass man mit „Schulungscomputern" aller Herren Couleur Geld verdienen kann. Die BBC schickte im Jahr 2016 mit dem micro:bit einen Einplatinencomputer ins Rennen, der statt eines vollwertigen Linux-fähigen Prozessors „nur" ein Bluetooth-SoC aus dem Hause Nordic Semiconductor mitbrachte.

Der Rest ist eine für den Autor dieser Zeilen nicht wirklich verständliche Geschichte: Mittlerweile gibt es zum Beispiel in der Slowakei Unternehmen, die sich ausschließlich über den Vertrieb des micro:bit-Ökosystems ernähren [1].

Der alte Kalauer, dass man an einem „gut gefüllten" Topf nicht allzu lang alleine frisst, gilt auch im Bereich des Prozessrechnerwesens. Die Weiterentwicklung im Bereich der Bluetooth-SoCs hat dazu geführt, dass der mit 16 MHz arbeitende Prozessor des micro:bit (insbesondere auch mit seinen nur 16 KB SRAM) heute wie ein Methusalem aussieht. Zudem ist das aus 5×5 Leuchtdioden bestehende Display für die Ausgabe komplizierterer Grafiken nicht geeignet.

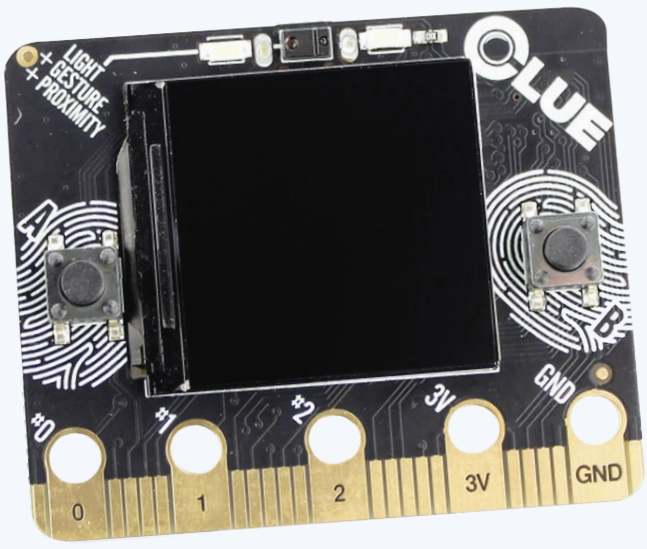


Bild 1. Der Angreifer von vorne ...

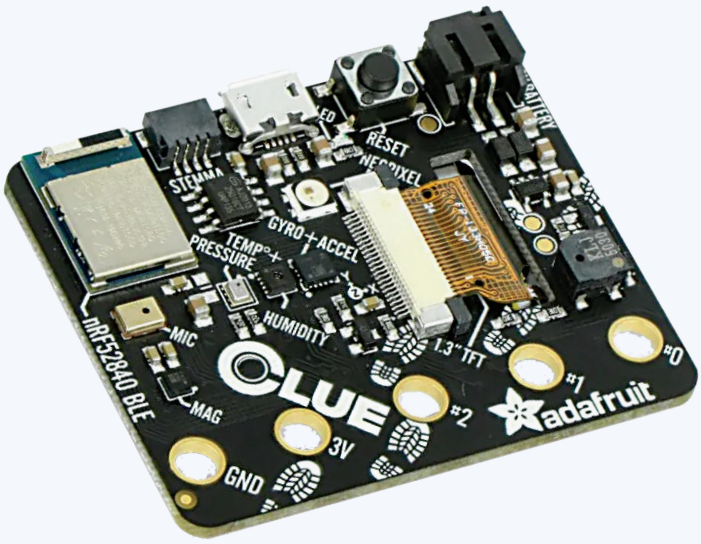


Bild 2. ... und von hinten.

Eigenschaften des CLUE

- Nordic nRF52840 Bluetooth LE Prozessor: 1 MB Flash, 256 KB RAM, 64 MHz Cortex M4 Prozessor
- 1,3" 240×240 Color IPS-TFT-Display für Texte und Grafiken
- Stromversorgung über eine beliebige 3...6-V-Quelle (interner Regler und Schutzdioden)
- Zwei Benutzertasten und eine Reset-Taste
- Bewegungssensor, Accelerometer/Gyrometer, Magnetometer
- Näherungs-, Licht-, Farb- und Gestensensor
- Mikrofon/Schallsensor
- SHT-Sensor für Luftfeuchtigkeit
- BMP280-Sensor für Temperatur und barometrischer Druck/Höhe
- RGB-NeoPixel-Anzeige-LED
- 2 MB interner Flash-Speicher für Daten, Bilder, Schriftarten oder CircuitPython-Code
- Summer/Lautsprecher
- Zwei helle weiße LEDs an der Vorderseite zur Beleuchtung / Farberkennung
- Qwiic / STEMMA QT-Anschluss für das Hinzufügen weiterer Sensoren, Motorsteuerungen oder Displays über I²C. GROVE-I²C-Sensoren lassen sich über ein Adapterkabel anschließen.
- Programmierbar mit Arduino IDE oder CircuitPython

Attacke von Adafruit

Spätestens als Nordic Semiconductor den nRF52840 vorstellte - ebenfalls ein einkerniges Bluetooth-SoC, dessen ARM-Prozessor allerdings bis zu 64 MHz erreicht und 256 KB RAM hat - war der „Angriffsvektor" klar. Die **Bilder 1** und **2** zeigen das Resultat - ein System, das dem BBC micro:bit durchaus ähnlich sieht.
Neben dem auf diesen Fotos nicht direkt erkennbaren SoC fällt auf der Vorderseite der wesentlich größere Bildschirm auf - statt mit Leuchtdioden bekommen wir es mit einem 240 × 240 Pixel großen Farbdisplay zu tun, das allerdings nicht auf organischer, sondern auf klassischer IPS-LCD-Technologie basiert.
Ein „netter Touch" des Moduls ist der auf der Rückseite befindliche und in **Bild 3** gezeigte Steckverbinder. Er exponiert im hauseigenen Format einen I²C-Bus, über den sich unbürokratisch weitere Sensoren anschließen lassen. Zudem gibt es einen Adapter auf das von Seeed betriebene Grove-Format, für das verschiedene fair bepreiste Sensoren im Handel angeboten werden.
Beachten Sie, dass der CLUE nur teilweise mit dem großen Vorbild kompatibel ist. Der Verbinder auf der Unterseite ist physikalisch identisch - ob der Nutzung eines anderen Displays gilt allerdings, dass der „Gutteil" der Gehäuse für den BBC micro:bit nicht auf den Adafruit CLUE passt.
Der Autor testete diese Hypothese mit dem unter [2] bereitstehenden ThingiVerse-Gehäuse von domw. Die Vorderseite passte offensichtlich nicht, da das Display des CLUE wesentlich größer war als die LED-Matrix des Originals der BBC. Besonders überraschend empfand der Autor in diesem Zusammenhang, dass die Rückwand des Gehäuses trotz der zusätzlichen Stecker problemlos passte - dies lag allerdings daran, dass das Gehäusedesign vergleichsweise großzügig gestaltet war. Hat Ihr Gehäusedesigner „knapper" konzipiert, so wäre wahrscheinlich schon an dieser Stelle komplettes Ende.

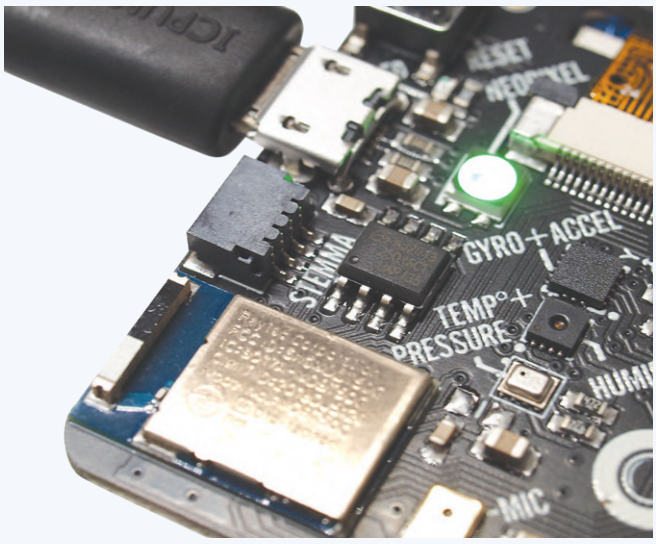


Bild 3. Der STEMMA-Port erlaubt dem Adafruit CLUE die Kontaktaufnahme zu Erweiterungen.

Eine Frage der Programmierung

Als „Ausbildungssystem" ist der micro:bit fernab von klassischen Embedded-Entwicklungsumgebungen wie ARM Keil angesiedelt - dies mag für Embedded-Puristen ärgerlich sein, ist in der Praxis aber notwendig, weil an vielen Hochschulen nicht ausreichend kompetentes Personal zum Debuggen von C++-Programmfehlern zur Verfügung steht (glauben Sie dem Autor: Kadetten finden magisch die debilsten Wege).
Stattdessen wird im Allgemeinen auf ein Quadriumvirat aus Arduino-IDE, CircuitPython, MakeCode und Scratch gesetzt. Im Fall des

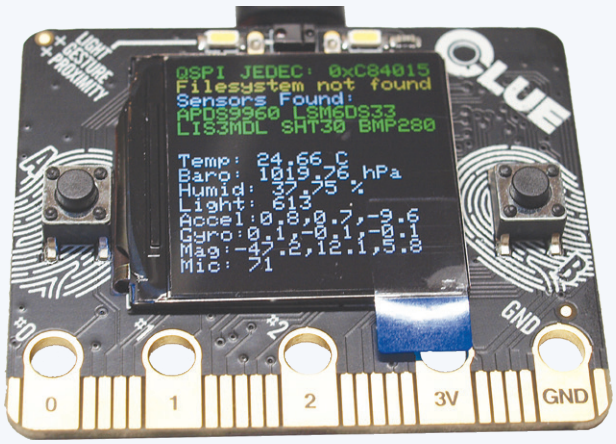


Bild 4. Diese App zeigt die von den Sensoren zurückgelieferten Informationen.

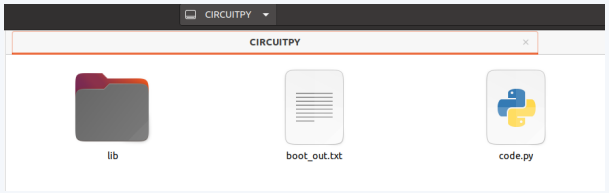


Bild 5. Die Python-Runtime exponiert ebenfalls ein virtuelles Laufwerk.

CLUE gilt, dass nur zwei der Systeme zur Verfügung stehen: An MakeCode wird ohne Liefertermin gearbeitet, zu Scratch gibt es keine Informationen.
Wichtig ist, dass Adafruit den CLUE mit einem seriellen Bootloader ausstattet, um das „Deployment“ von Code - ganz analog zum Raspberry Pi Pico - zu ermöglichen.
Für einen ersten kleinen Versuch starten wir zunächst einmal CircuitPython.
Verbindet man ein jungfräuliches Board über die auf der Rückseite befindlichen MicroUSB-Buchse mit dem Rechner, so zeigt das Display eine Statusseite (**Bild 4**), die Informationen über den Betriebszustand anbietet.
Doppeltes Drücken des auf der Rückseite befindlichen Reset-Tasters sorgt im ersten Schritt dafür, dass die Bildschirmanzeige ob des im Displaycontroller enthaltenen Framebuffers einfriert. Die angeschlossene Workstation (beim Autor läuft sie unter Linux) sieht das Aufscheinen eines neuen USB-Laufwerks, auf dem sich Kompilate ablegen lassen.
Interessanterweise ist der Adafruit CLUE für den Rechner „immer“ sichtbar - ist er nicht im Bootloader-Modus, so erkennt ihn *dmesg* folgendermaßen:

```
tamhan@TAMHAN18:~$ dmesg
. . .
[28292.202193] usb 1-2.7: Manufacturer: Adafruit LLC
[28292.202195] usb 1-2.7: SerialNumber: 7687A137B6FDB874
[28292.204040] cdc_acm 1-2.7:1.0: ttyACM0: USB ACM device
```

Nach dem Doppel-Druck auf den Button sehen wir stattdessen nach folgendem Schema ein USB-Laufwerk:

```
tamhan@TAMHAN18:~$ dmesg
. . . .
[28371.624193] sd 10:0:0:0: Attached scsi generic sg6 type 0
```

Wichtig ist, dass dieses Laufwerk nicht „ewig“ aktiviert bleibt - die Firmware setzt sich nach etwa 30 Sekunden Totzeit wieder in den normalen Betrieb zurück.

Dateien suchen

Wir besuchen im ersten Schritt die URL *https://circuitpython.org/board/clue_nrf52840_express/*, um die Datei *adafruit-circuitpython-clue_nrf52840_express-en_US-6.1.0.uf2* herunterzuladen. Sie enthält die Runtime, die Sie auf dem USB-Laufwerk ablegen.
Interessant ist in diesem Zusammenhang, dass Sie im Laufwerk zudem eine Datei namens *CURRENT.UF2* finden - sie erlaubt das einfache Abernten des Kompilats, dass gerade im Speicher des Prozessrechners liegt.
Lustigerweise wird die Runtime nicht mit einem vollständigen Bibliotheks-Set ausgeliefert, das alle enthaltenen Sensoren abdeckt. Wir müssen stattdessen die URL *https://circuitpython.org/libraries* aufrufen, wo wir das Archiv *adafruit-circuitpython-bundle-6.x-mpy-20210329.zip* herunterladen und es in einen bequemen Ordner im Dateisystem extrahieren.
Spätestens an dieser Stelle sollten Sie einen weiteren Blick auf den Bildschirm des CLUE werfen - die Runtime gibt die Inhalte der Konsole permanent am Bildschirm aus. Ein netter Touch ist, dass das Gerät auf dem PC - wie in **Bild 5** gezeigt - den internen Speicher der Python-Arbeitsumgebung exponiert.
Wichtig ist, dass Sie im *Libs*-Ordner einerseits die folgenden Ordner aus dem Archiv unterbringen:

adafruit_apds9960
adafruit_bus_device
adafruit_display_shapes
adafruit_display_text
adafruit_lsm6ds
adafruit_register

Als wäre dies noch nicht genug der Arbeit, mutet Adafruit den Käufern auch noch das Zusammensuchen der folgenden

Einzeldateien zu - warum man nicht ein „schlüsselfertiges“ Archiv anbietet, ist dem Rezensenten nicht klar:

adafruit_bmp280.mpy
adafruit_clue.mpy
adafruit_lis3mdl.mpy
adafruit_sht31d.mpy
adafruit_slideshow.mpy
neopixel.mpy

Code-Beispiel

Für einen ersten kleinen Gehversuch mit der Python-Umgebung bietet sich das unter *https://learn.adafruit.com/adafruit-clue/clue-spirit-level* bereitstehende Beispiel an - es realisiert einen Lagemesser, nutzt dabei aber eine Reihe CLUE-spezifischer Idiome.
Die erste Amtshandlung des Codes besteht darin, eine Gruppe von Bibliotheken einzubinden:

```
import board
import displayio
from adafruit_display_shapes.circle import Circle
from adafruit_clue import clue
```

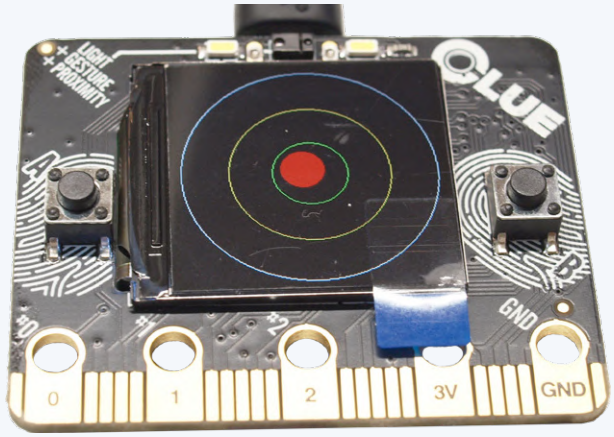


Bild 6. Eine Wasserwaage auf Basis eines Eval-Boards.

Anzeige

TAILORED TO YOUR NEEDS.
Custom & Standard Terminal Blocks

WÜRTH ELEKTRONIK
MORE THAN YOU EXPECT

Würth Elektronik Terminal Blocks
Neben einem Portfolio von mehr als 2000 Standardbauteilen, bietet Würth Elektronik verschiedene Möglichkeiten, die Produkte auf Ihre spezifischen Anforderungen zuzuschneiden. Personalisierte Modifikationen von Standard Terminal Blocks sind bei kleinen bis mittleren Stückzahlen als besonderer Service innerhalb weniger Tage verfügbar. Komplett kundenspezifische Lösungen sind in hohen Stückzahlen innerhalb weniger Wochen möglich. Eigene Konstruktion, Werkzeugbau und Prototypenbau stellen sicher, dass alle kundenspezifischen Anforderungen erfüllt werden.
Weitere Informationen finden sie unter: **www.we-online.de/TBL**

- Kundenspezifische Produkte nach Ihren Anforderungen
- Über 2000 Standardbauteile
- Ab Lager verfügbar ohne MOQ
- Schnelle Lieferung
- Personalisierte Modifikationen von Standardteilen für kleine Mengen
- Farb- & Druckmöglichkeiten mit MOQ für Massenproduktion

Neben dem `clue`-Objekt, das diverse boardbezogene Funktionen zur Verfügung stellt, ist hier auch der Import der `Circle`-Klasse interessant. Der GUI-Stack erlaubt nicht nur das „direkte" Zeichnen in einen Framebuffer, sondern unterstützt auch die Arbeit mit Objekten, die von der Firmware in am Bildschirm erscheinende Elemente umgesetzt werden.
Im nächsten Akt kümmert sich die Firmware darum, einen Verweis auf das Display zu verschaffen und ein Display-Group-Objekt zusammenzustellen:

```
display = board.DISPLAY
clue_group = displayio.Group(max_size=4)
```

Das `clue_group`-Objekt ist insofern interessant, als es ein an einen DOM-Baum erinnerndes Eltern-Element generiert, in das unser Code danach mehr oder weniger beliebige Objekte für die Anzeige schreibt.
Aus der Logik folgt, dass die nächste Amtshandlung des Programms darin besteht, drei für die Darstellung der Stärke der Auslenkung zuständigen Kreise zu erzeugen und für die Ausgabe anzumelden:

```
outer_circle = Circle(120, 120, 119, outline=clue.WHITE)
middle_circle = Circle(120, 120, 75, outline=clue.YELLOW)
inner_circle = Circle(120, 120, 35, outline=clue.GREEN)
clue_group.append(outer_circle)
clue_group.append(middle_circle)
clue_group.append(inner_circle)
```

Als Nächstes folgen noch einige weitere Housekeeping-Tasks, deren Sinn sich am einfachsten beim Blick auf das im Artikel weiter unten gezeigte Beispiel-Rendering erschließt:

```
x, y, _ = clue.acceleration
bubble_group = displayio.Group(max_size=1)
level_bubble = Circle(int(x + 120), int(y + 120), 20,
   fill=clue.RED, outline=clue.RED)
bubble_group.append(level_bubble)

clue_group.append(bubble_group)
display.show(clue_group)
```

Zu guter Letzt benötigen wir eine Arbeitsschleife, die die von der Adafruit-Bibliothek über das Attribut `acceleration` ausgegebenen Lagewerte analysiert und in die Koordinaten-Eigenschaften des „Ziel-Objekts" weiterschreibt:

```
while True:
    x, y, _ = clue.acceleration
    bubble_group.x = int(x * -10)
    bubble_group.y = int(y * -10)
```

Der bequemste Weg zur „schnellen" Ausführung vom Code am CLUE ist die in Bild 5 gezeigte Datei *code.py*. Sie wird von der CircuitPython-Firmware automatisch im Rahmen jedes Starts ausgeführt. **Bild 6** zeigt, was Sie sich erwarten dürfen.
Ob des Vorhandenseins eines Bluetooth-Transmitters ist das System auch zur Kommunikation mit dem Rechner befähigt. Unter [3] stellt Adafruit ein sehr lustiges Beispiel zur Verfügung, das die Verwendung einer in Google Chrome implementierten Web-Bluetooth-API illustriert.

Und jetzt mit C

Python mag ein schneller Weg sein, um „unbürokratisch" Ergebnisse aus einem Prozessrechner zu bekommen. Die maximale Performance erreicht man aber durch C.
Insbesondere auf einem Einkern-Funksystem ist das Realisieren von kommunikativem Code schwierig, wenn es nicht zu Timing-Problemen kommen soll. Damit zwingt Adafruit Entwicklern die Nutzung der Arduino-IDE mehr oder weniger auf. Im Hintergrund arbeitet dann ein Echtzeit-Betriebssystem, das sich um die Zuweisung von Rechenleistung an die verschiedenen Aufgaben kümmert.
Unter Linux ist im ersten Schritt ein Erweiterungspaket erforderlich, das der Arduino-IDE (ab Version 1.8.6) die Kommunikation mit dem nicht-standardisierten Bootloader des CLUE ermöglicht:

```
tamhan@TAMHAN18:~$ pip3 install --user adafruit-nrfutil
Collecting adafruit-nrfutil
  . . .
Successfully installed adafruit-nrfutil-0.5.3.post13
```

Als Nächstes müssen wir im Board Manager die URL *https://www.adafruit.com/package_adafruit_index.json* einpflegen, um das Board-Package *Adafruit nRF52* zum Download verfügbar zu machen. Nach getaner Arbeit steht das Board unter *Tools* > *Board* > *Adafruit CLUE* zur Verfügung.
Wie im Fall von CircuitPython gilt leider auch hier, dass die

WEBLINKS

[1] StreamIT, s. r. o.: https://edutronik.sk/v2/
[2] Gehäuse: https://www.thingiverse.com/thing:1767446
[3] Demo-Applikation: https://learn.adafruit.com/bluefruit-dashboard-web-bluetooth-chrome
[4] Infos: https://learn.adafruit.com/adafruit-clue?view=all

Einrichtung von Bibliothek & Co. ein arbeitsintensiver Prozess ist. Weitere Informationen dazu finden sich unter [4].

Lohnt es sich?

Wer den CLUE in die Hand nimmt, bekommt eine durchaus attraktive Evaluationsplattform, die - insbesondere dank des Bildschirms - im Bereich Interfacing angenehm zu bedienen ist. Andererseits steht der im Vergleich zum BBC micro:bit vergleichsweise hohe Preis - das „Original" kostet wesentlich weniger. Als wäre dies noch nicht genug, ist der CLUE eben doch nicht zu 100% mit dem micro:bit kompatibel - die traurige Lebenserfahrung lehrt, dass ein „insignifikantes Detail", das von 99% der Anwendungen nicht tangiert wird, genau im vorliegenden Projekt für Probleme sorgt.
Wer eine „saubere" Nordic-basierte Plattform realisieren möchte, ist mit einem klassischen Evaluationsboard besser bedient. Unterm Strich ist der CLUE also ein liebenswertes Produkt für jene, die einer kompatiblen Kleinstserie von BBC micro:bit-Projekten zu mehr Leistungsfähigkeit verhelfen wollen.

210395-02

Haben Sie Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Schicken Sie eine E-Mail an den Autor unter tamhan@tamoggemon.com oder kontaktieren Sie Elektor unter redaktion@elektor.de

Ein Beitrag von

Text und Bilder: **Tam Hanna**
Redaktion: **Jens Nickel**
Layout: **Giel Dols**

PASSENDE PRODUKTE

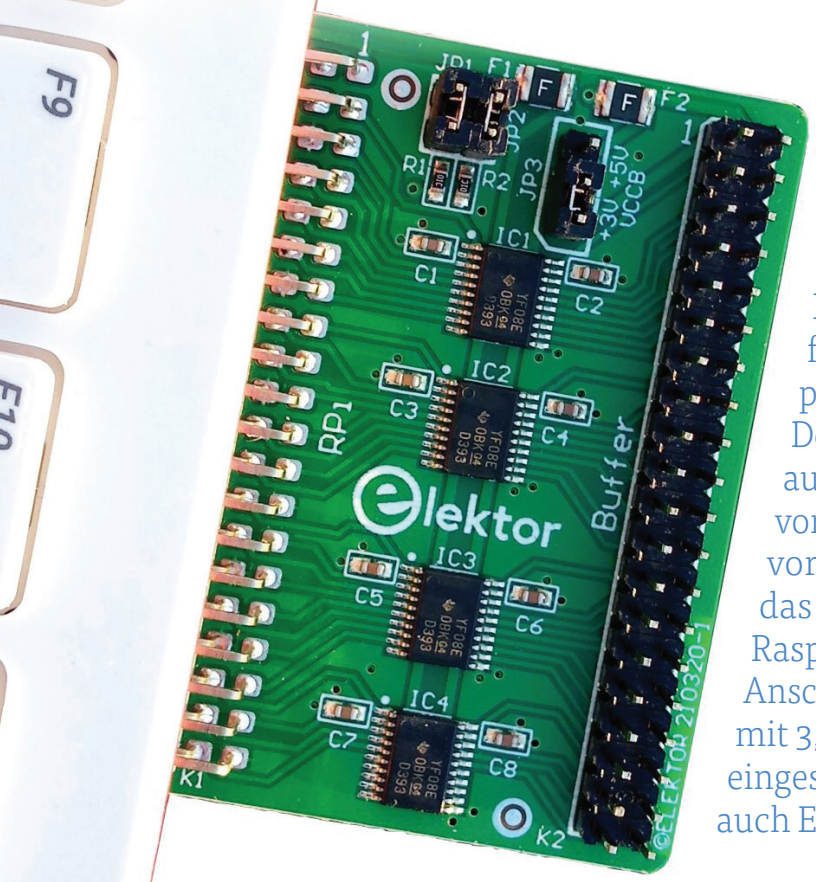
- Adafruit CLUE
  www.elektor.de/19512

Anzeige

Puffer-Board

für den Raspberry Pi 400

Schützen Sie die I/Os!



Von **Ton Giesberts** (Elektor)

Der im November 2020 in den Markt eingeführte Raspberry Pi 400 ist eigentlich ein Raspberry Pi 4, der in einer Tastatur eingebaut ist. Der GPIO-Erweiterungsanschluss befindet sich auf der Rückseite des Gehäuses. Der Anschluss von zusätzlicher Hardware birgt immer Risiken, vor allem beim Prototyping. Das Pufferboard, das wir hier vorstellen, wurde speziell für den Raspberry Pi 400 entwickelt. Es ermöglicht den Anschluss externer Logik für alle 26 GPIOs, sowohl mit 3,3-V- als auch mit 5-V-Versorgung. Die dafür eingesetzten Puffer/Pegelwandler bieten zusätzlich auch ESD-Schutz.

Raspberry-Pi-Boards müssen hier kaum vorgestellt werden. Seit der Einführung der ersten Version im Jahr 2012 sind sie Gegenstand (oder Teil) vieler Projekte in Elektor gewesen. Wir haben eine ganze Reihe unterschiedlichster Hardware-Projekte und Erweiterungsplatinen für diese Prozessorboards vorgestellt. Die jetzt hier gezeigte Schaltung ist nicht neu; die erste Version wurde bereits 2015 auf Elektor Labs [1] besprochen. Im Jahr 2018 wurde dieser Entwurf an den erweiterten 40-poligen I/O-Anschluss angepasst, der seit der Einführung des Raspberry Pi B+ [2] Standard für den Anschluss zusätzlicher Hardware an den Raspberry Pi ist. Und nun wird die letztere Version an eines der neueren Mitglieder der Raspberry-Pi-Produktfamilie angepasst - den Raspberry Pi 400. Ganz einfach gesagt, ist der 400er ein Raspberry 4, der in eine Tastatur eingebaut ist. Er ähnelt damit den einst berühmten Computern der 1980er Jahre wie dem Commodore 64, Sinclair Spectrum, Acorn BBC und ähnlichen. Von den Spezifikationen her ist der Raspberry Pi natürlich um ein Vielfaches leistungsfähiger als seine mittlerweile antiken Vorgänger, aber es gibt auch eine auffällige Ähnlichkeit, nämlich einen GPIO-Erweiterungsanschluss auf der Rückseite des Gehäuses, an den der Benutzer externe (zugekaufte oder selbst entworfene) Hardware anschließen kann. Der Anschluss von zusätzlicher Hardware birgt aber immer Risiken, vor allem in der Entwicklungsphase. Die hier vorgestellte Pufferplatine vermindert das Risiko, dass der Raspberry Pi dabei beschädigt wird. An die Pufferplatine kann externe Logik mit einer Versorgungsspannung von 3,3 V oder 5 V angeschlossen werden, und die dafür verwendeten Puffer/Pegelwandler bieten einen eingebauten ESD-Schutz.

Die Hardware

Der Schaltplan des Projekts (**Bild 1**) ist exakt derselbe wie im Artikel aus dem Jahr 2018. Die verwendeten 8-Bit-Puffer/Spannungswandler TXS0108E sind bidirektional, und jeder A-Port- und B-Port-Pin verfügt über einen Pull-up-Widerstand. Der Pull-Down-Widerstand eines GPIOs des Raspberry Pi liegt typischerweise in der Größenordnung von 40...60 kΩ, was zu hoch ist, um den I/O richtig „herunterzuziehen“, wenn die Pufferplatine eingesteckt ist. Bei dieser Eingangskonfiguration ist also der logische Pegel nicht richtig low; der Pull-Down funktioniert nicht wie vorgesehen. Die I/Os haben separate Versorgungspins zur Raspberry-Pi-Seite (VCCA) und zur Außenwelt (VCCB). Jeder A-Port-I/O des TXS0108E verfügt über einen Pull-up-Widerstand zu VCCA, der mit der +3,3-V-Spannungsversorgung des Raspberry Pi 400 verbunden ist, und jeder B-Port-I/O besitzt einen Pull-up-Widerstand zu VCCB. VCCB für den Pegel der I/Os auf K2 kann am Jumper JP3 entweder auf +3,3-V- oder auf +5-V-Logik eingestellt werden. Die Pull-ups der Puffer haben einen Wert von 40 kΩ, wenn

der Ausgang auf Low steht, und einen Wert von 4 kΩ, wenn der Ausgang auf High steht. Es handelt sich also faktisch um Open-Drain-Ausgänge. Wenn zum Beispiel eine LED vom Ausgang des Puffers nach Masse angeschlossen wird, entsteht ein Spannungsteiler, wenn ein zusätzlicher Vorwiderstand verwendet wird. Eine ohmsche Last am Ausgang führt zu einer Verringerung des logischen High-Pegels, etwas, das Sie beachten sollten!

Es gibt zwei rücksetzbare 0,5-A-PPTC-Sicherungen (F1 und F2) in den Versorgungsleitungen zwischen K1 und K2 auf der Pufferplatine, um die +5-V- und +3,3-V-Stromversorgung des Raspberry Pi 400 zu schützen. Für die Implementierung eines I²C-Busses zur Kommunikation mit externer Hardware (GPIO2 ist SDA und GPIO3 ist SCL) können die zusätzlichen Pull-up-Widerstände R1 und R2 durch die Jumper JP1 und JP2 aktiviert werden.

Während des Hochfahrens des Raspberry Pi werden GPIO0 (ID_SD) und GPIO1 (ID_SC) verwendet, um das EEPROM eines I²C-HAT (Hardware Attached on Top) zu lesen. Nach dem Booten können diese GPIOs wie die 26 anderen verwendet werden, aber es muss darauf geachtet werden, dass das System nicht beim Booten beeinträchtigt wird, wenn ein I²C-HAT montiert ist. Um das Lesen von GPIO0 und GPIO1 während des Bootens zu verhindern, fügen Sie den folgenden Eintrag in die Konfigurationsdatei */boot/config.txt* ein:

```
force_eeprom_read=0
```

Weitere Informationen über die Datei config.txt finden Sie in der Raspberry-Pi-Dokumentation [3].

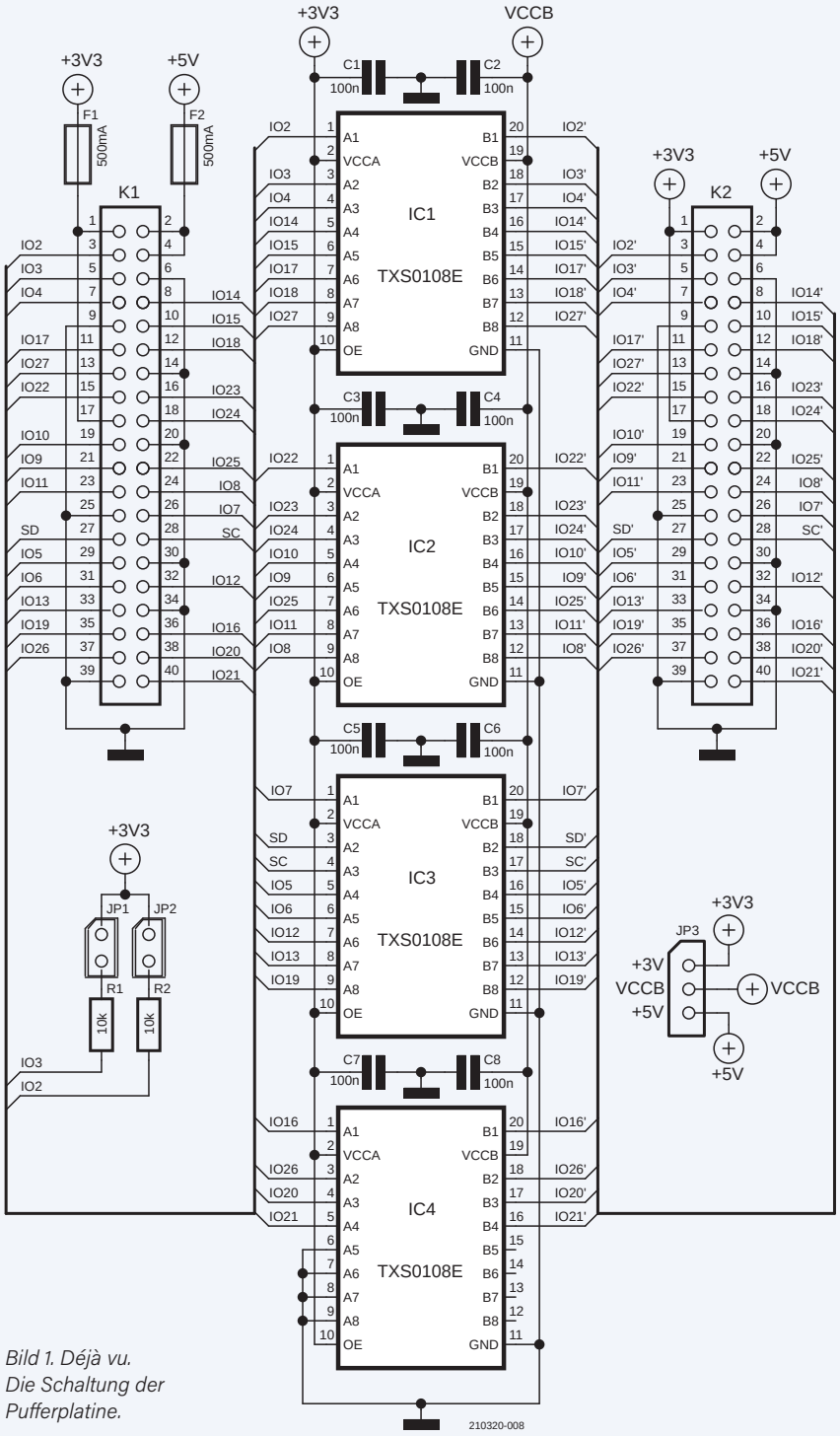


Bild 1. Déjà vu. Die Schaltung der Pufferplatine.

Das Platinenlayout

Der Schaltplan mag alt sein, aber die Platine (**Bild 2**) wurde speziell für den Raspberry Pi 400 angepasst. Die Gerber-Dateien für diese neue Platine stehen zum Download bereit, so dass Sie sie beim Platinenhersteller Ihrer Wahl bestellen können. Viel bequemer ist es natürlich, die komplett bestückte Pufferplatine im Elektor Store [4] zu kaufen.

Für den Stecker auf der Raspberry-Pi-400-Seite (K1) wird eine rechtwinklige Buchse verwendet, so dass die Pufferplatine in den GPIO-Header auf der Rückseite des Gehäuses eingesteckt werden kann (**Bild 3**). Beim Anschluss K2, der die gepufferten I/Os bereitstellt, handelt es sich um einen vertikalen 40-poligen Standardheader (K2). Das Board ist mit 55 x 44 mm einschließlich der Buchse K1, die über die Kante der Platine herausragt, etwas kleiner als die ursprüngliche Pufferplatine aus dem Jahr 2018. Im Vergleich zur Originalplatine 150719-1 sind die beiden Stiftreihen von K1 vertauscht, da hier eine Buchse verwendet wird. Das Anbringen einer vertikalen 40-poligen Stiftleiste für K1, um diese Pufferplatine wie bei der älteren Version des Puffers über ein Flachkabel mit einem „normalen" Raspberry Pi zu verbinden, funktioniert hier zwar nicht, man kann, wie **Bild 4** zeigt, dieses Modul aber einfach sekrecht direkt auf den GPIO-Header eines Raspberry Pi 2...4 aufstecken.

STÜCKLISTE

Widerstände:

R1,R2 = 10 k, 100 mW, 1 %, SMD 0603

Kondensatoren:

C1…C8 = 100 n, 50 V, 10 %, X7R, SMD 0603

Halbleiter:

IC1…IC4 = TXS0108EPWR, SMD TSSOP-20

Außerdem:

K1 = 2x20-polige Stiftleiste, rechtwinklig, Raster 2,54 mm
K2 = 2x20-polige Stiftleiste, gerade, Raster 2,54 mm
JP1,JP2 = 1x2-polige Stiftleiste, gerade, Raster 2,54 mm
JP3 = 1x3-polige Stiftleiste, gerade, Raster 2,54 mm
JP1,JP2,JP3 = Jumper, Raster 2,54 mm
F1,F2 = Zurücksetzbare PPTC-Sicherung, SMD
(Polyfuse, 1210L050YR Littelfuse)
Platine 210320-1 v1.0

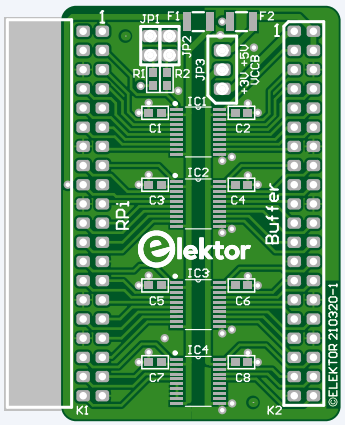


Bild 2. Das Layout der neuen Pufferplatine.

Der Ausgangsanschluss K2 kann über ein kurzes 40-poliges Flachbandkabel mit zwei 2x20-Buchsen mit externen Schaltungen verbunden werden, oder nur eine einzelne Buchse verwenden, die mit angelöteten kurzen Käbelchen angeschlossen wird, oder gar nur über einzelne Buchsen mit Drähten. Seien Sie jedoch vorsichtig, wenn Sie eine 40-polige Buchse auf K2 drücken oder sie von der Platine abziehen. Tun Sie dies nicht, solange die Pufferplatine noch im Raspberry Pi 400 steckt, da dies einen gewissen Kraftaufwand erfordert und beim Herumwerkeln der GPIO-Header des Raspberry Pi 400 beschädigt werden könnte.

Testen der Pufferplatine

Zwei sehr einfache Python-Programme zum Testen der Pufferplatine - entlehnt aus dem älteren Projekt - stehen auf der Elektor-Labs-Seite dieses Projekts als Download *210320-11.zip* bereit [5]. Eines testet alle GPIOs als Output (*Check_all_GPIOs_as_output.py*), das andere testet alle GPIOs als Input (*Check_all_GPIOs_as_input.py*). Im Raspberry Pi OS (aka Raspbian) genügt ein Doppelklick auf eine der Dateien, um die Standard-IDE für Python zu öffnen. Dann wählen Sie RUN, um den Test zu starten.

Wenn man die GPIOs als Ausgang testet, braucht man nur eine einzelne Low-Current-LED zwischen einem Pin und GND, um zu sehen, ob ein Ausgang funktioniert oder nicht. Als Vorwiderstand für die LED kann ein 1,8-kΩ-Widerstand verwendet werden, wobei der Wert nicht wirklich kritisch ist. Er begrenzt den Strom durch die LED, wenn diese direkt an die positive Versorgungsspannung angeschlossen ist. Die Ausgänge werden nacheinander in vier Gruppen IOA bis IOD von jeweils maximal acht Pins getestet. Aufgrund des Open-Drain-Ausgangs beträgt die Spannung über einer (roten) LED plus Widerstand etwa 2,6 V, wenn 5 V als Versorgungsspannung für die Ausgänge gewählt ist (JP3). Schließen Sie den Widerstand plus LED an einen der gewählten Ausgänge an, und er wird für 0,2 s eingeschaltet. Die Wiederholrate dieses Impulses hängt von der Größe der Gruppe ab: 1,6 s für die Gruppen A…C mit jeweils acht Ausgängen und nur 0,4 s für Gruppe D, die nur zwei Ausgänge besitzt. In der Zeile

```
for i in IOA:            # leds blink 0.2 s in IOx group
```

ist „IOA“ für den Test der anderen Gruppen entsprechend zu ändern. Natürlich können auch GPIO0 und GPIO1 (ID_SD und ID_SC) zu einer der Gruppen hinzugefügt werden.

Das Programm zum Testen der GPIOs als Eingang verwendet als Anzeige des Testprogramms einen I/O als Ausgang, und zwar standardmäßig GPIO3. Schließen Sie dazu einen 1,8-kΩ-Widerstand und eine LED zwischen diesem Pin 5 von K2 und GND an. Vom Quellcode wird jeweils nur ein Eingang getestet, um sicherzustellen, dass auch nur dieser als Eingang funktioniert. Ändern Sie die Zahl in der folgenden Zeile, um einen anderen GPIO als Eingang zu testen:

```
IN1 = 2  #selected GPIO to test as input
```

Das Programm gibt den ausgewählten GPIO und seinen Eingangspegel aus. Die Pull-ups der Eingänge sind aktiviert, sodass der aktuelle Eingangspin mit Masse verbunden werden muss, damit die angeschlossene LED leuchten kann. Wählen Sie am Ende der Prüfungen einen anderen als GPIO3 für den Ausgang, damit Sie auch diesen IO3 als Eingang testen können. Natürlich gibt es auch zahlreiche andere Möglichkeiten, die GPIOs zu testen. Wenn jemand also einen effizienteren und/oder schnelleren Weg entdeckt: bitte mitteilen!

Mit diesem Puffer können Sie neue Hardware unbesorgter an den Raspberry Pi 400 anschließen, da die Gefahr, dass der Pi bei Experimenten beschädigt wird, deutlich geringer ist. Dass das Risiko geringer ist, heißt nicht, dass die Pufferplatine garantiert, dass gar nichts mehr schief gehen kann. In vielen Fällen könnte es nicht schaden, der Elektronik auch etwas logisches Denken hinzuzufügen!

210320-02

WEBLINKS

[1] Raspi Buffer Board bei Elektor Labs: www.elektormagazine.de/labs/raspi-buffer-board
[2] Guy Weiler: „Puffer-Board für Raspberry Pi“, Elektor 11-12/2018 : www.elektormagazine.de/magazine/elektor-51/42024
[3] Mehr zur config.txt: www.raspberrypi.com/documentation/computers/config_txt.html
[4] Buffer-Board für den Raspberry Pi 400 im Elektor-Store: www.elektor.de/raspberry-pi-400-buffer-board
[5] Projektseite bei Elektor Labs: https://bit.ly/3GdRJ4G

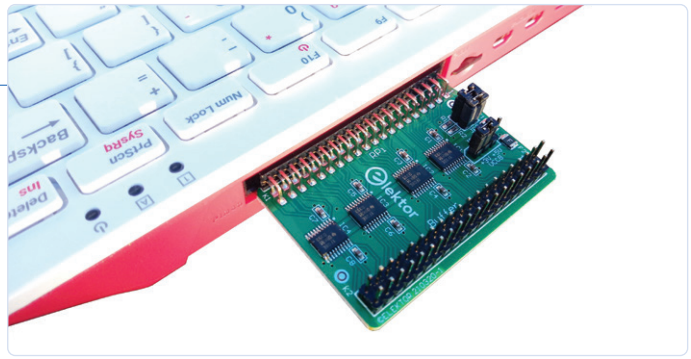


Bild 3. Die Pufferplatine, eingesteckt in den Raspberry Pi 400.

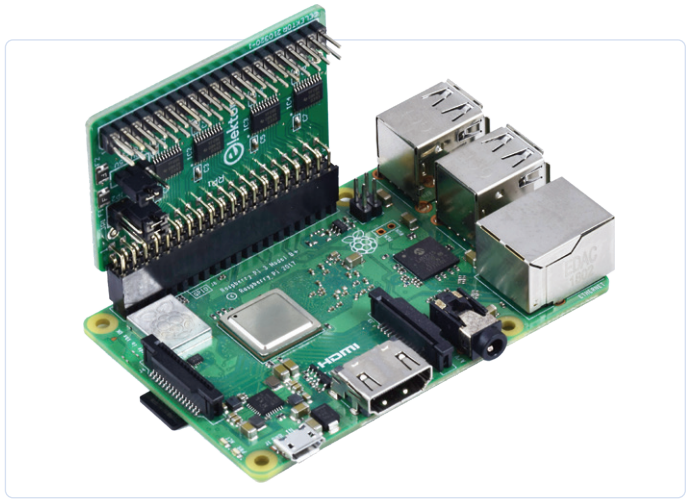


Bild 4. Die Platine kann auch mit „klassischen" Raspberry-Pi-Boards verwendet werden.

Ein Beitrag von

Entwurf, Text und Illustrationen: **Ton Giesberts**
Redaktion: **Luc Lemmens**
Übersetzung: **Rolf Gerstendorf**
Layout: **Giel Dols**

Haben Sie Fragen oder Kommentare?

Wenn Sie technische Fragen oder Kommentare zu diesem Artikel haben, können Sie sich per E-Mail an die Elektor-Redaktion wenden: redaktion@elektor.de.

PASSENDE PRODUKTE

- **Raspberry Pi 400 Puffer-Board (SKU 19884)**
  www.elektor.de/19884
- **Raspberry Pi 400 Kit – Raspberry Pi 4-basiertes PC-Kit (DE) + GRATIS GPIO-Header (QUERZ-Tastatur, SKU 19432)**
  www.elektor.de/19432
- **Raspberry Pi 400 Kit – Raspberry Pi 4-based PC Kit (EU) + FREE GPIO Header (QUERTY-Tastatur, SKU 19431)**
  www.elektor.com/19431

Anzeige

Weller®

WSW LÖTDRAHT
DIE PERFEKTE
LÖTSTELLE

LEISTUNG & PRODUKTIVITÄT
Optimierte Leistung durch garantierten und konsistenten 100 % durchgängigen Flussmittelkern.

HOCHWERTIGE LÖTVERBINDUNGEN
Langlebige, hochfeste Lötverbindungen ohne Rissbildungen, auch auf schwierigen Oberflächen.

NIEDRIGE BETRIEBSKOSTEN
Bis zu 70 % weniger Spitzenverbrauch und niedrige Gesamtbetriebskosten durch Einsparung von Arbeitszeit und Ressourcen bei höherer Produktivität.

REDUZIERTES SPRITZVERHALTEN
Erhöht die direkte Anwendersicherheit sowie die Sauberkeit am Arbeitsplatz.

WSW Lötdraht
Weitere Informationen
www.weller-tools.com/wsw

Frühjahrskollektion

Interessante Raspberry Pi RP2040 Boards

Von **Mathias Claußen** (Elektor)

Der RP2040 ist der erste Mikrocontroller aus dem Hause Raspberry Pi. Er ist auf dem maker-freundlichen Raspberry Pi Pico verbaut; inzwischen hat er seinen Weg aber auch in Boards und Kits von Drittanbietern gefunden. Es wird Zeit, diese einmal vorzustellen!

Der RP2040 ist sicherlich eine interessante Alternative zu den etablierten Mikrocontrollern. Dabei überzeugen nicht nur das Preis-Leistungs-Verhältnis, sondern auch die aktuelle Verfügbarkeit des Chips. Dokumentation und Support durch die Raspberry Pi Foundation haben ihren Anteil daran, dass der Controller als sehr einsteigerfreundlich angesehen werden kann. All dies kann in Artikeln [1][2][3], einem Webinar [4] oder Videos [5] von Elektor nachgelesen und angeschaut werden.

Der Raspberry Pi Pico - das herstellereigene Board, auf dem der RP2040 verbaut ist - wurde nur sehr übersichtlich mit Zusatzhardware ausgestattet, um den Preis von 5 € pro Board halten zu können. Ein Jahr nach seinem Release hat der RP2040 daher seinen Weg auf etliche Boards von Drittanbietern gefunden, die unterschiedlichste Peripherie ergänzt haben. Daher sollen diese nachfolgend einmal vorgestellt werden, so dass Interessierte das passende Board für ihre Anwendung entdecken können.

Raspberry Pi Pico

Der Raspberry Pi Pico (**Bild 1**) weist das Minimum an Hardware auf, mit dem ein Start mit dem RP2040 möglich ist. Zu seiner „Sonderausstattung" gehören eine grüne LED, die durch den Nutzer gesteuert werden kann, sowie ein DC/DC-Wandler, der den Betrieb des RP2040 auch mit Akkupacks und Batterien erlaubt. Der Raspberry Pi Pico ist vom Preis-Leistungs-Verhältnis her immer noch eines der besten Boards, wenn man mit dem RP2040 anfangen möchte und schon auf eine eigene Sammlung an Modulen und externer Elektronik zurückgreifen kann. Als Starterkit für die Lehre oder das Selbststudium ist das Board allein nur bedingt geeignet. Dagegen lässt es sich aber ohne große Mühen als Debugger für einen anderen RP2040 einsetzen. Bei 5 Euro pro Raspberry Pi Pico ist das Board sehr preiswert. Wer jedoch schon vorinstallierte Stiftleisten und ein passendes Micro-USB-Kabel haben möchte, sollte sich den Raspberry Pi Pico mit vorinstallierten Headern (**Bild 2**) im Elektor Store ansehen, wo man auch passende Erweiterungen für den Raspberry Pi Pico findet.

Adafruit Feather RP2040

Beim Adafruit Feather RP2040 (**Bild 3**) handelt es sich um ein RP2040-Board im Feather-Formfaktor. Der Unterschied zum Raspberry Pi Pico ist vor allem die Peripherie und die Menge an Flash-Speicher, die auf

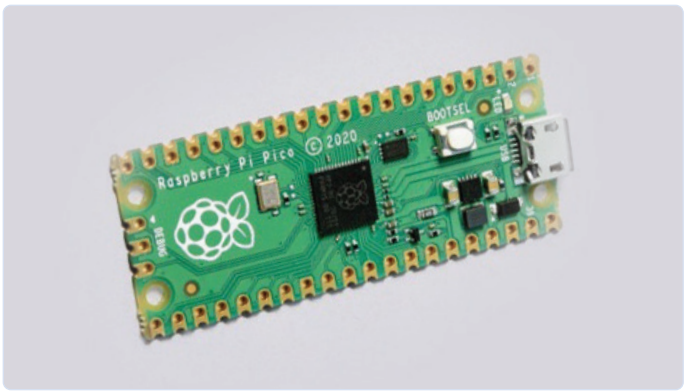


Bild 1. Raspberry Pi Pico (Quelle: Raspberry Pi).

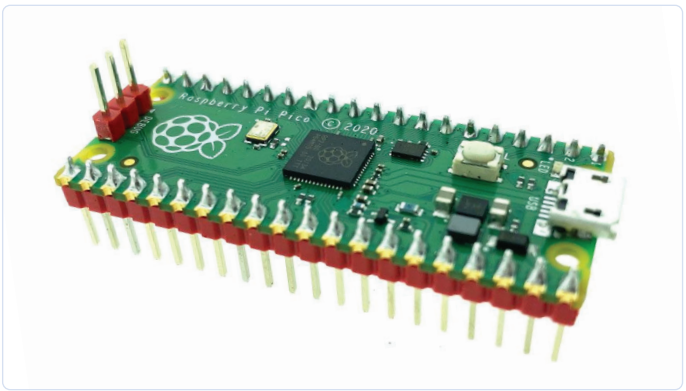


Bild 2. Raspberry Pi Pico mit Headern (Quelle: Elektor).

dem Board integriert wurde. Besitzt der Raspberry Pi Pico nur 2 MB Flash, so sind es auf dem Feather-Board 8 MB, also deutlich mehr Platz für eigene Software. Auch ist eine RGB-LED auf dem Board verbaut sowie ein Qwiic-Connector für Peripherie und eine USB-C-Buchse. Auffallend ist die Anschlussmöglichkeit für einen Lithium-Akku an der Seite. Es befindet sich auch gleich die passende Ladeschaltung auf dem Board, sodass man das Feather RP2040 leicht mit einem Akku versorgen kann.

SparkFun Thing Plus – RP2040

Das SparkFun Thing Plus – RP2040 (**Bild 4**) ist dem Adafruit Feather RP2040 sehr ähnlich, auch was das Pinout angeht. Jedoch sind hier die vom RP2040 maximal ansprechbaren 16 MB Flash verbaut, dazu kommen eine RGB-LED und noch drei Status-LEDs. Wie beim Adafruit Feather RP2040 ist auch hier eine Ladeschaltung für einen Lithum-Akku und ein Qwicc-Connector integriert worden. Dreht man das Board um, so kommt ein SD-Karten-Slot zum Vorschein. So lassen sich mit den PIO-Statemachines des RP2040 SD-Karten ansprechen und als Massenspeicher benutzen. Warum das Team von Raspberry Pi mit der Wahl der GPIOs für die SD-Karte etwas überrascht war, zeigt ein Webinar [4]. Wer schon Peripherie mit Qwiic-Connector besitzt oder Add-Ons mit Feather-Pinout, der sollte sich einmal dieses Board ansehen!

Arduino Nano RP2040 Connect

Der Arduino Nano RP2040 Connect (**Bild 5**) bringt eine RP2040-MCU mit Wi-Fi und Bluetooth zusammen. Mit 27 € ist es sicherlich nicht das günstigste Board, jedoch kommen zum Wi-Fi- und Bluetooth-Modul Nina W102 von u-blox noch eine 6-Achsen-IMU (STM LSM6DSOXTR), ein Mikrofon (MP34DT05) und ein Microchip ATECC608A Crypto-Chip hinzu. Das Ganze weist ein kompaktes Format auf und wird natürlich bestens von der Arduino-IDE unterstützt.

Für die ersten Schritte mit einem RP2040 ist mehr als genug Hardware auf dem Board. Die Möglichkeit, Daten per Bluetooth und WLAN austauschen zu können, sowie das integrierte Mikrofon lassen auch an einen Einstieg in die ersten KI-Applikationen möglich erscheinen. Doch auch wenn ein Arduino Nano RP2040 Connect in Reichweite meines Schreibtisches liegt, so ist das Board nicht meine Empfehlung für jemanden, der mit einem Raspberry Pi RP2040 anfangen möchte. Wer mit Wi-Fi und IoT starten will, sollte erst einmal einen Blick auf einen ESP32 oder den neuen ESP32-C3 werfen und damit seine ersten Schritte unternehmen (ein Blick in das Datenblatt des Moduls Nina W102 zeigt sowieso, dass darin ein ESP32 seinen Dienst verrichtet). Doch diejenigen Entwickler(innen), die schon erste Erfahrungen mit Wi-Fi und Bluetooth gemacht haben und einen RP2040 mit Cloudanbindung betreiben möchten, sollten sich den Arduino Nano RP2040 Connect näher ansehen. Noch ein Tipp: Mein Kollege Clemens Valens hat ein kleines Video [6] über das Board gemacht, das auf dem Youtube-Kanal von Elektor zu finden ist.

Cytron Maker Pi Pico (mit fest verlötetem Raspberry Pi Pico)

Wer mit dem Raspberry Pi Pico und dem RP2040 starten möchte, wird mit dem Cytron Maker Pi Pico (**Bild 6**) sicherlich eine mehr als gute Basis finden. Der Raspberry Pi Pico ist auf dem Board fest verlötet, so dass dieser nicht einfach getauscht werden kann. Doch nur so ist ein Preis von nur 18 € möglich.

Doch was ist anders an dem Board als bei den bisher genannten? Zuerst einmal das Format. Das Cytron Maker Pi Pico versteht sich als Experimentierplattform und nicht als Mikrocontrollerboard für das

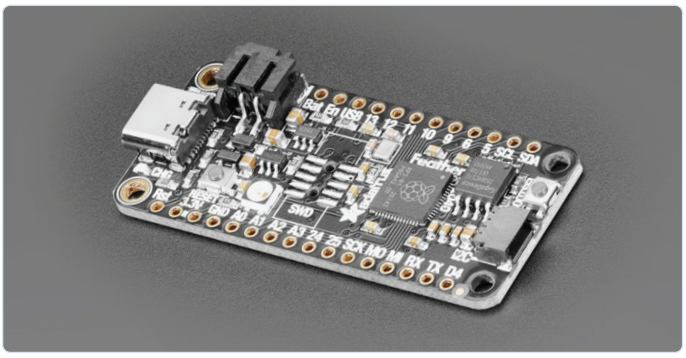


Bild 3. Adafruit Feather RP2040 (Quelle: Adafruit).

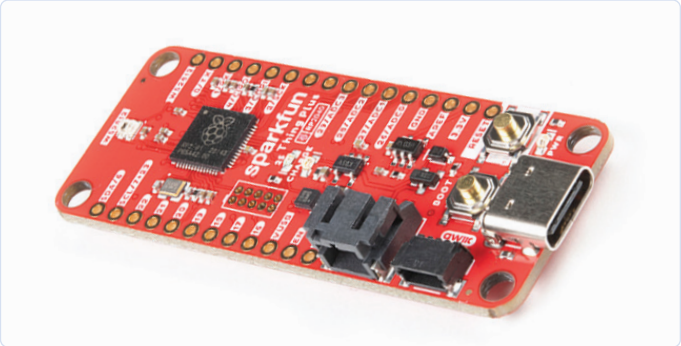


Bild 4. SparkFun Thing Plus RP2040 (Quelle SparkFun).

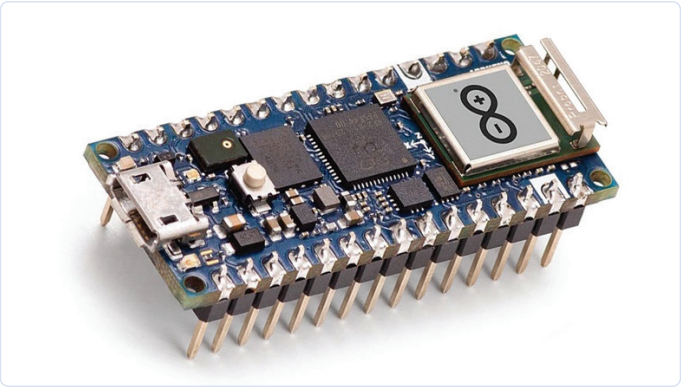


Bild 5. Arduino Nano RP2040 Connect (Quelle: Arduino.cc).

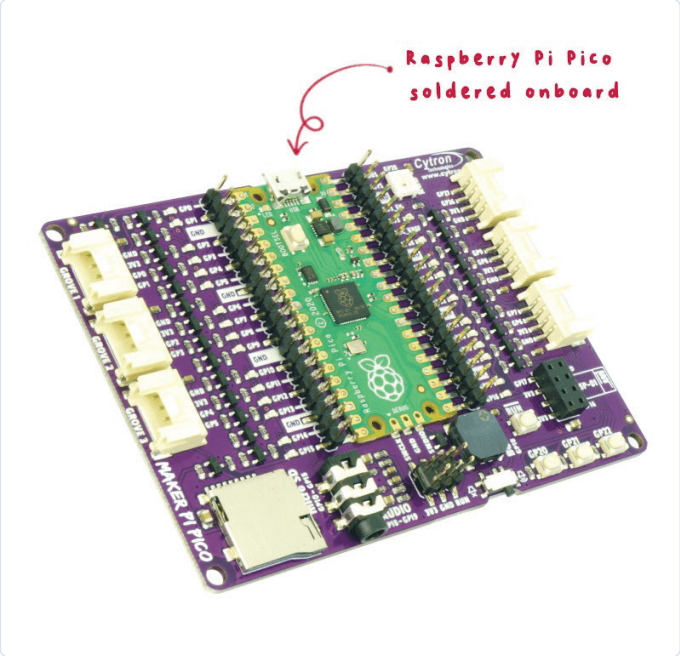


Bild 6. Cytron Maker Pi Pico (Quelle: cytron.io).

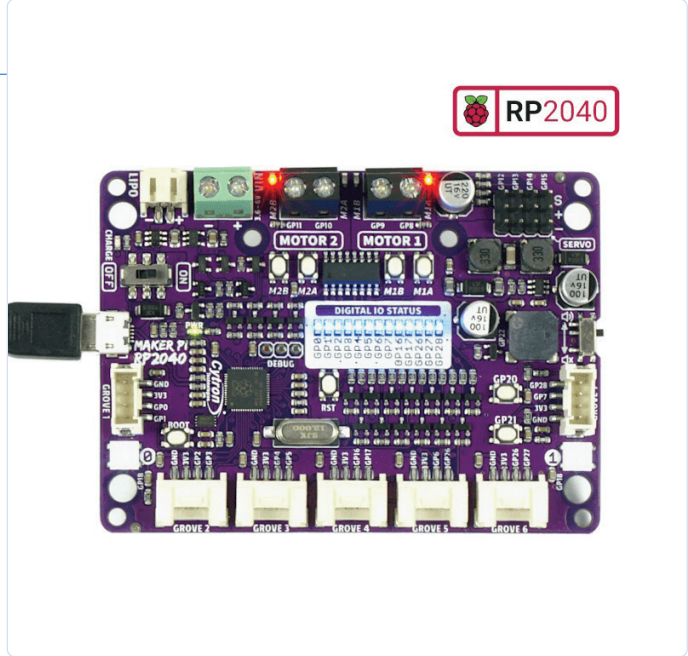


Bild 7. Cytron Maker Pi RP2040 (Quelle: cytron.io).

Steckbrett. Alle Pins des Raspberry Pi Pico sind noch einmal auf zwei Stiftleisten herausgeführt, so dass man mit einem Messgerät leicht an diese herankommt. Auch bietet das Board etliche Grove-Anschlüsse für Peripherie. Jeder Pin auf dem Board ist mit einer LED verbunden, so dass schnell ersichtlich ist, welche Spannung an den IO-Pins des Raspberry Pi Pico anliegt. Taster und Buzzer runden das Board ab.
Wer schon ein Sortiment an Grove-kompatibler Hardware besitzt, kann diese einfach verwenden. Durch die Stiftleisten lassen sich aber auch andere Module schnell und ohne großen Aufwand anschließen. Mit Buzzer (samt Ein/Aus-Ausschalter), WS2812 RGB-LED, Tastern und SD-Karten-Slot kann man sich Stück für Stück in immer umfangreichere Pico-Projekte vortasten. Hinzu kommt noch ein ESP-01-Pinheader, mit dem das Board um WLAN-Funktionalität erweitert werden kann. Natürlich ist auch die Debug-Schnittstelle des Raspberry Pi Pico auf dem Board nach außen geführt.

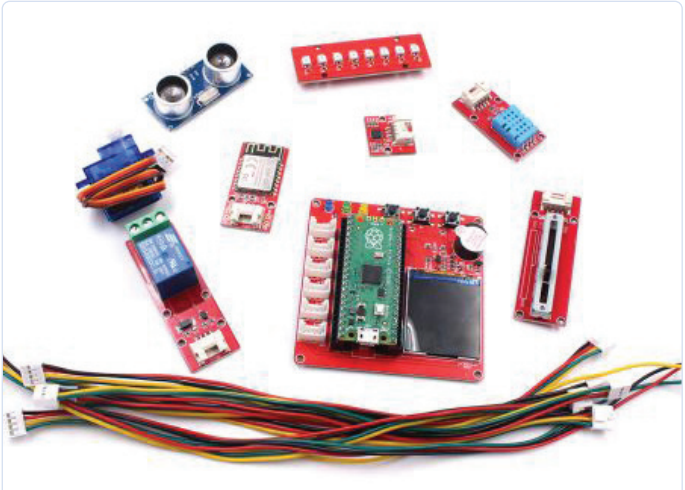


Bild 8. Elektor Raspberry Pi Pico Experimenting Kit (Quelle: Makerfabs).

Cytron Maker Pi RP2040

Motorsteuerung und Robotik? Mit dem Raspberry Pi PR2040? Manchmal wäre es praktisch, mit dem Raspberry Pi Pico kleine Motoren oder einen Schrittmotor steuern zu können. Mit dem Cytron Maker Pi RP2040 (**Bild 7**) ist genau das möglich. Ein integrierter Motortreiber (MX1508/TC1508) mit zwei H-Brücken ist bereits auf dem Board integriert. Falls nun die Frage aufkommt, was eine H-Brücke ist, und wie die Motoransteuerung im Inneren arbeitet, empfiehlt sich ein Blick in meinen Grundlagen-Artikel [7].
Kleine Motoren mit bis zu 6 V und 1 A pro Kanal können direkt an dem Board betrieben werden. Auch bietet das Board die Möglichkeit, bis zu drei Servos direkt anzuschließen. Da das Board für Robotik-Anwendungen ausgelegt ist, gehört auch ein Batterieanschluss für eine LiPo-Zelle inklusive Ladeelektronik dazu. Auf dem Board kann das Laden des Akkus deaktiviert werden, so dass die LiPo-Zelle nicht ungewollt geladen wird. Auch ist für die LiPo-Zelle eine Schutzbeschaltung vorhanden, die Unter- und Überspannung verhindern soll. Der RP2040 ist hier nicht in Form eines Raspberry Pi Pico verbaut, bietet aber die exakt gleichen Spezifikationen wie der Raspberry Pi Pico, was den Flash-Speicher angeht, nämlich 2 MB. Auf dem Board sind 13 Status-LEDs für die GPIOs des RP2040 integriert, zwei WS2812-LEDs, ein Buzzer, zwei Taster und sieben Grove-Ports für Erweiterungen.

Elektor Raspberry Pi Pico Experimenting Kit

Das Elektor Raspberry Pi Pico Experimenting Kit (**Bild 8**) ist mit seinen 45 € das teuerste hier vorgestellte Board. Basis bildet der Raspberry Pi Pico, der auf diesem Board nur gesteckt ist, so dass er bei Bedarf einfach ausgetauscht werden kann. Das Board des Kits selbst stellt Taster, LEDs, Buzzer, ein TFT-Display und Grove-Verbinder bereit. Mit dem Zubehör aus WS2812-LEDs, DHT11-Temperatur- und Luftfeuchte-Sensor, Relais, Potentiometer, Ultraschall-Abstandssensor, Servo, Gyro- und Beschleunigungssensor MPU6050 sowie einem ESP8266 kommt man so zu einem Kit, das für Einsteiger und experimentierfreudige Fortgeschrittene gleichermaßen geeignet ist. Da die Komponenten als

Module (über die Grove-Verbinder) mit dem Board verbunden werden, lässt sich der Raspberry Pi Pico flexibel nutzen und ausprobieren. Dank des 1,44-Zoll-Displayy (mit ST7735-Controller) können so leicht aus Python oder C/C++ heraus erste Grafikanwendungen entwickelt werden. Wer noch keine Module oder andere Komponenten haben sollte, erhält hier einen sehr runden Einstieg in die Welt des RP2040.

Welches Board sollte man nehmen?

Das ist eine Frage, die jeder für sich selbst beantworten muss und stark davon abhängt, was man machen möchte. Jedes Board oder Kit hat seine Vor- und Nachteile, und je nach Budget muss man auch hier abwägen, was wirklich benötigt wird. Brauche ich ein Board, auf dem alle Peripherie schon verbaut ist, oder ist nicht doch ein Kit mit Modulen besser? Ist das Board für mich selbst, oder möchte ich mit dem Board einen Einsteiger für die Welt der Mikrocontroller begeistern? Brauche ich eine Ladeschaltung für einen Akku?
An dieser Stelle kann deshalb keine pauschale Empfehlung für ein Board gegeben werden. Klar ist aber, dass mit dem Raspberry Pi RP2040 ein Mikrocontroller zum Einsatz kommt, der sowohl für Anfänger als auch für erfahrene Entwickler jede Menge Potential bietet - und vor allem aktuell verfügbar ist.

210629-01

Sie haben 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.

Ein Beitrag von

Entwurf und Text: **Mathias Claußen**
Redaktion: **Jens Nickel**
Layout: **Giel Dols**

PASSENDE PRODUKTE

- **Elektor Raspberry Pi Pico Experimenting Kit (SKU 19834)**
  www.elektor.de/19834
- **Cytron Maker Pi RP2040 (SKU 19926)**
  www.elektor.de/19926
- **Cytron Maker Pi Pico (SKU 19706)**
  www.elektor.de/19706
- **Arduino Nano RP2040 Connect (SKU 19754)**
  www.elektor.de/19754
- **SparkFun Thing Plus – RP2040 (SKU 19628)**
  www.elektor.de/19628
- **Adafruit Feather RP2040 (SKU 19689)**
  www.elektor.de/19689
- **Raspberry Pi Pico RP2040 (SKU 19562)**
  www.elektor.de/19562
- **Raspberry Pi Pico RP2040 (mit bestückten Stiftleisten) (SKU 19568)**
  www.elektor.de/19568

WEBLINKS

[1] „Raspberry Pi Pico und der Mikrocontroller RP2040", Elektormagazine.de: https://www.elektormagazine.de/articles/raspberry-pi-pico-und-der-mikrocontroller-rp2040
[2] „NEU: Raspberry Pi Pico und der Mikrocontroller RP2040", Elektormagazine.de: https://www.elektormagazine.de/news/neu-raspberry-pi-pico-und-der-mikrocontroller-rp2040
[3] Raspberry Pi Pico - mit vorinstallierten Stiftleisten: https://www.elektormagazine.de/news/raspberry-pi-pico-mcu-with-preinstalled-pin-headers-de
[4] Eben Upton und Nathan Seidle über Raspberry Pi Pico und RP2040: https://www.elektormagazine.de/news/eben-upton-and-nathan-seidle-about-raspberry-pi-pico-and-rp2040-de
[5] Elektor TV auf YouTube: https://www.youtube.com/user/ElektorIM
[6] Clemens Valens, „Board Review: Arduino Nano RP2040 Connect", ElektorTV: https://www.youtube.com/watch?v=2EnCf64zZSA
[7] Mathias Claußen, „Motorsteuerung mit H-Brücken", Elektor Januar/Februar 2022: https://www.elektormagazine.de/210491-02

Ein DIY-Handbuch zu **elektronischer Sicherheit und E-Spionage**

SRAM erhitzt oder tiefgefroren

Von **Luka Matic** (Kroatien)

Hier wird gezeigt, wie man sich in die Daten des statischen RAM (SRAM) einhackt und sie mit „inoffiziellen“ Methoden verschlüsselt, wodurch die Fähigkeit des Bausteins, Daten zu speichern, im Grunde genommen überstrapaziert wird. Nachdem Sie die Theorie verdaut haben, können Sie sich Ihre eigenen Varianten ausdenken oder Wege finden, um das hier Gezeigte zu verbessern und zu automatisieren. Probieren Sie dies zu Hause aus!

Anmerkung der Redaktion: *Dieser Artikel ist ein Auszug aus dem 224-seitigen Buch* A Handbook on DIY Electronic Security and Espionage, *der formatiert und leicht bearbeitet wurde, um den redaktionellen Standards und dem Seitenlayout von Elektor zu entsprechen. Da es sich um einen Auszug aus einer größeren Publikation handelt, können sich einige Begriffe in diesem Artikel auf Diskussionen an anderer Stelle im Buch beziehen. Der Autor und die Redaktion haben ihr Bestes getan, um solche Fälle auszuschließen, und sind gerne bereit, bei Fragen zu helfen. Kontaktinformationen finden Sie im Kasten* Fragen oder Kommentare?

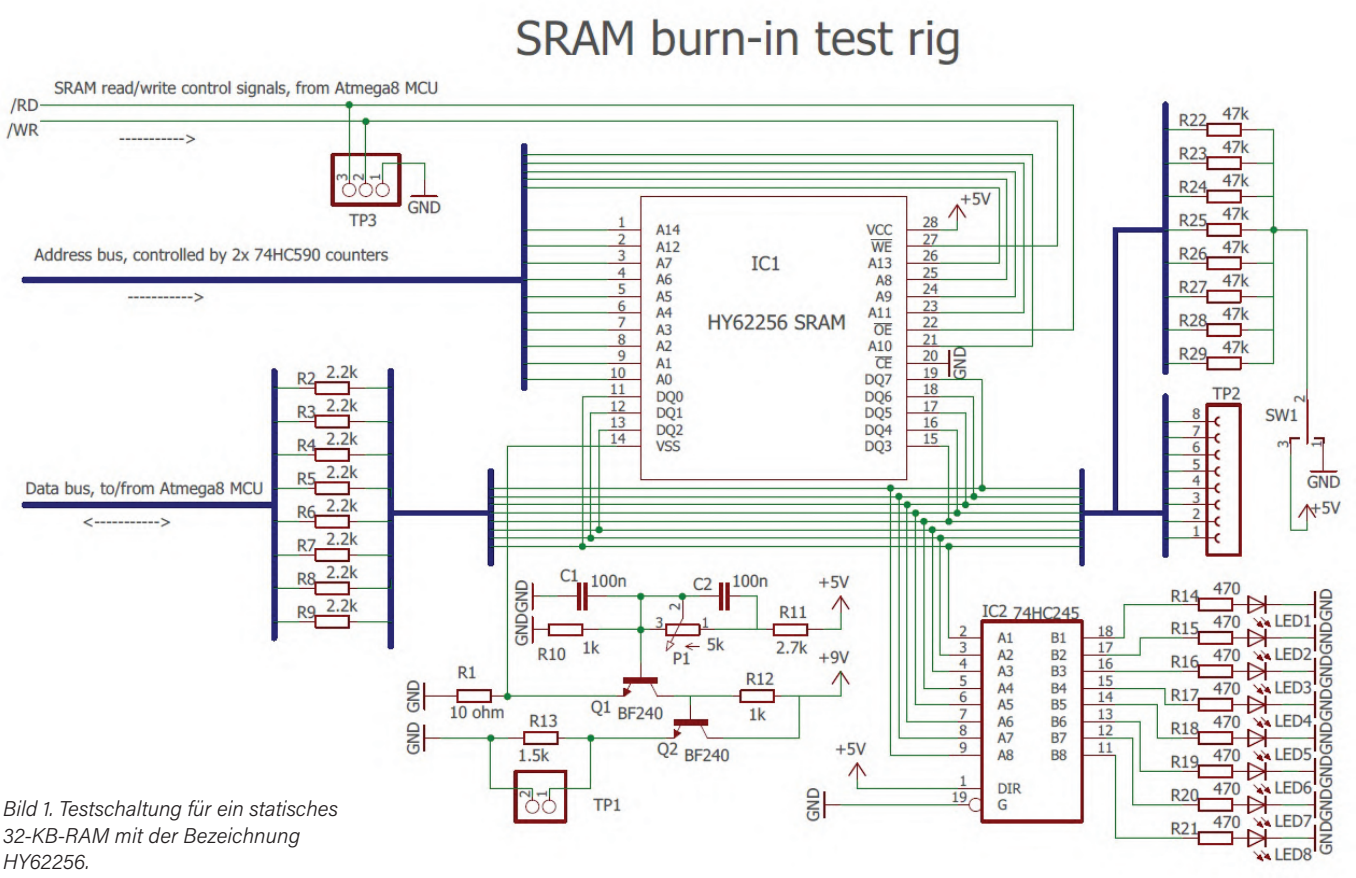


Bild 1. Testschaltung für ein statisches 32-KB-RAM mit der Bezeichnung HY62256.

Wiederherstellung „eingebrannter“ SRAM-Daten

Die Schaltung des Testaufbaus für das Experiment ist in **Bild 1** dargestellt. Der zweistufige Analogverstärker aus einer ersten Stufe Q1 mit gemeinsamer Basis, dann Q2 mit gemeinsamem Kollektor (ein Aufbau für die größtmögliche Bandbreite für schnelle Signale) verstärkt den Spannungsabfall an R1 zur Messung des SRAM-Versorgungsstroms. Das aufzeichnende Oszilloskop ist an Testpunkt TP1 angeschlossen. Der gesamte Aufbau wird von einer ATmega8-MCU gesteuert (im Schaltplan nicht dargestellt). Der Adressbus wird auf eine zu prüfende Speicheradresse eingestellt. Dies läuft zwar relativ langsam ab, da zwei 8-Bit-Zähler 74HC590 verwendet werden, aber da sich die Adresse nicht schnell ändern muss, ist das in Ordnung. Der Datenbus wird von der MCU gelesen/geschrieben, die auch die /RD- und /WR-Befehle an das SRAM steuert. Diese beiden Steuerleitungen werden als Trigger für Oszilloskopmessungen am Testpunkt TP3 verwendet. Die Widerstände R2…R9 entkoppeln den Datenbus, falls dieser gleichzeitig von der MCU und dem SRAM aktiviert wird.

Die Widerstände R22…R29 ziehen den Adressbus nach oben oder unten (das wird mit SW1 ausgewählt), um die Anstiegs- und Abfallzeiten des Lesezugriffs und des Datenbusses über den Testpunkt TP2 zu messen. Der achtfache Puffer IC2 steuert ein Arrays von acht LEDs an, mit dem sich der Datenbus leichter überwachen lässt.

Die wichtigsten zu messenden Größen sind der Strom der Stromversorgung während des Lesens/Schreibens vom/zum SRAM und die Spannungen (eigentlich Delay-, Anstiegs- und Abfallzeiten) auf dem Datenbus. Die Unterschiede der gemessenen Signale zeigen die Einbrenneffekte an und ermöglichen, das zuletzt vor dem Abschalten des SRAMs geschriebene Byte zu erkennen.

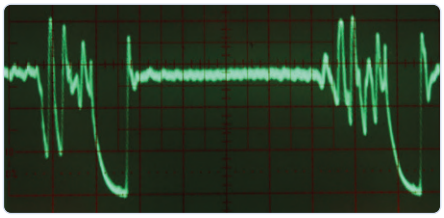


Bild 2. Der I_{dd}-Strom beim Schreiben der Bytes 0xDE (links) und 0x01 (rechts) in das SRAM auf einem alten Tek-466-Oszilloskop (Auflösung 200 ns/Div).

Die wichtigste Voraussetzung ist, dass die SRAM-Zellen über einen relativ langen Zeitraum konstante Werte enthalten haben. Die für unseren Hack notwendigen Zeiten können je nach SRAM-Typen variieren.

Ein 100-Ω-Widerstand (5 W) ist neben dem SRAM-IC als Chipheizung angeschlossen, mit einem Temperatursensor dazwischen, so dass der Speicher bei erhöhten Temperaturen von bis zu 80…90 °C getestet werden kann. Dies ist auf dem Foto der

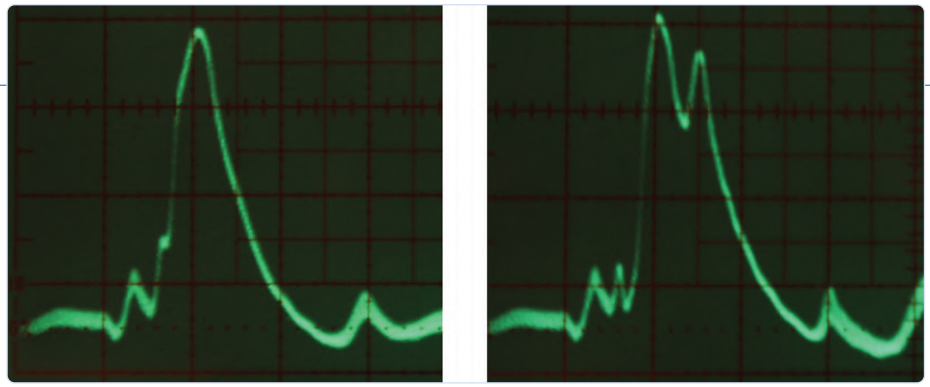


Bild 3. I_{dd}-Strom beim Lesen von 0xDE (links) und 0x01 (rechts) (bei 200 ns/div).

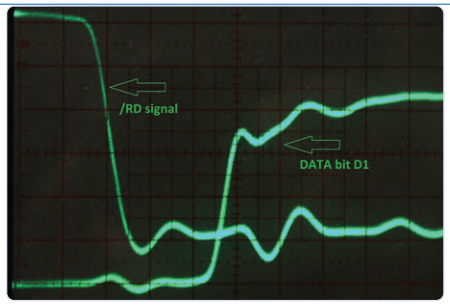


Bild 4. Signale auf der /RD-Leitung und eines Datenbits beim Lesevorgang (bei 10 ns/div).

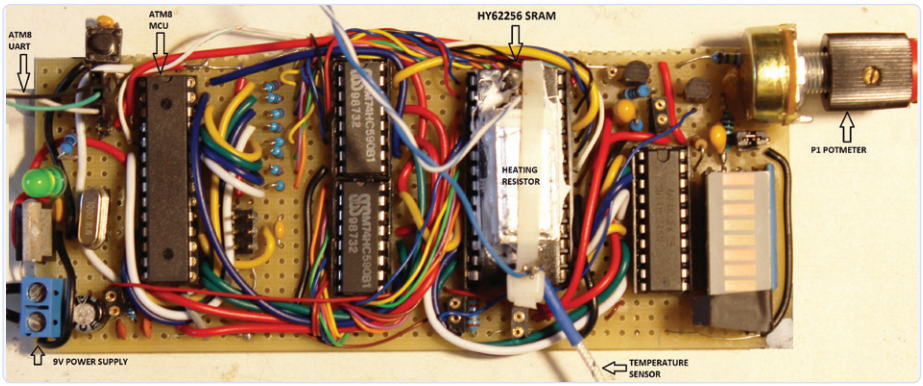


Bild 5. Der SRAM-Burn-in-Prüfstand des Autors.

Testanordnung in Bild 5 zu sehen.
Variationen der Anstiegs- und Abfallzeiten in der Größenordnung von 1...2 ns müssen zuverlässig gemessen werden können, weshalb ein 100-MHz-Oszilloskop erforderlich ist. Ich habe ein altes Oszilloskop (Tektronix 466) mit einer analogen Bildspeicherung (mit einstellbarer Persistenz und „flood guns“) für schnelle, nicht repetitive Signale verwendet. In Elektor gab es einst einen schönen Artikel über dieses Oszilloskop in der brillanten Rubrik Retronik [1]. Eine Speicherung ist jedoch nicht unbedingt erforderlich, da die Lese-/Schreib-Testsequenzen in sich wiederholenden Schleifen ablaufen können, die von der MCU gesteuert werden. Ich habe mein Tek 466 in Ljubljana (Slowenien) für nur 150 € auf einem Elektronikflohmarkt gekauft.
Wir beginnen mit einigen grundlegenden Oszilloskop-Messungen des RAM-Strombedarfs und der /RD-Spannung, siehe **Bild 2, Bild 3** und **Bild 4**. Die Messung des Stroms beim Schreiben und/oder Lesen von SRAM-Zellen mit eingebrannten Daten (das heißt Zellen, die lange Zeit dasselbe Byte enthielten) beziehungsweise ohne Daten (Zellen, die ebenso lange alle Bits in beiden Zuständen enthielten) und der Vergleich der Ergebnisse kann zur Wiederherstellung der Daten verwendet werden. Der Vergleich von Lesezugriffszeiten und Signalformen zwischen den Bits auf dem Datenbus kann, wie in Bild 4 zu sehen, ebenfalls zur Wiederherstellung von Daten beitragen.
Ein Bild meines Testaufbaus für das Experiment zur Wiederherstellung von SRAM-Daten ist in **Bild 5** zu sehen.
Ich habe gleichbleibende Daten zum SRAM geschrieben und es so belassen, dass alle Einbrenneffekte erhalten bleiben. Ich programmierte andere SRAM-Zellen über einen Zähler, so dass sie alle 10 ms inkrementiert wurden. Dann erhitzte ich das SRAM auf 80 °C, um die Einbrenneffekte zu verstärken und ließ den Chip zwölf Stunden lang eingeschaltet, während die MCU weiterhin die Zähler inkrementierte, um Einbrenneffekte auf diesen Zellen zu vermeiden. Zwölf Stunden später schaltete ich den Heizwiderstand aus und ließ das SRAM auf Raumtemperatur abkühlen. Nachfolgend die Ergebnisse der Tests, die ich an zwei SRAM-Zellen durchgeführt habe:

- An der Adresse 0x204F befand sich ein konstanter Wert 0x66, von dem man annehmen kann, dass er durch Einbrennen verursacht wurde.
- Bei 0x7FF1 handelte es sich um einen dauerhaft inkrementierenden Zähler, der keine Spuren von Einbrennen aufwies.

Die Messungen der Lesezugriffszeiten (wie in Bild 4) ergaben sehr geringe Unterschiede zwischen 0- und 1-Bits von weniger als 1 ns, so dass ich sie nicht als relevant ansah. Nur die „bistabilen“ MOSFETs, die also ein 0- oder ein 1-Bit halten, sind von Burn-in-Effekten betroffen, nicht aber die anderen MOSFETs, die für den Zugriff auf die SRAM-Zelle und die Übertragung auf den Datenbus beim Lesen verwendet werden. Messungen des I_{dd}-Versorgungsstroms beim Schreiben verschiedener Bytes in eingebrannte Zellen ergaben jedoch bessere Ergebnisse. Diese Methode ist für andere Konfigurationen besser geeignet, zum Beispiel für das Lesen von SRAM aus einer MCU, da sie keinen physischen Zugriff auf die acht Bits des SRAM-Datenbusses erfordert.
Auch hier enthielt die Zelle mit der Adresse 0x204F das Byte 0x66 und erfuhr ein Burn-in bei 80 °C. Bei der anderen Zelle mit der Adresse 0x7FF1 kam es zu keinem Einbrennen (alle Bits wurden regelmäßig und rechtzeitig gekippt). Die Ergebnisse der Messungen in **Bild 6** und **Bild 7** zeigen, dass das Schreiben eines Bytemusters mit mehr Nullen in eine eingebrannte Speicherzelle mehr Strom verbraucht als das Schreiben desselben Musters in eine Zelle ohne Einbrennen. Der Unterschied des benötigten Stroms beim Schreiben von 0x66 (ein eingebrannter Wert) und 0x99 (1er-Komplement eines eingebrannten Wertes) in eine eingebrannte Zelle ist im Vergleich zu der anderen Zelle ohne Einbrennen viel höher. Diese Messungen allein reichen aber nicht aus, um das genaue Bytemuster zu ermitteln, das in die Zelle eingebrannt wurde. Sie erfordern eine mathematische

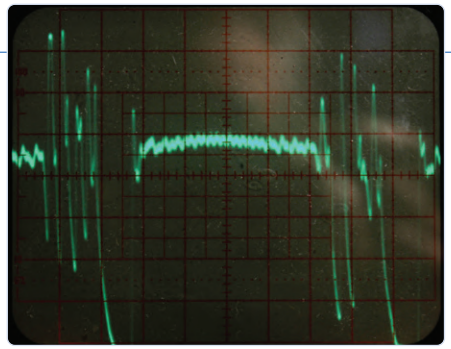
Nachbearbeitung (Filtern, Korrelieren mit bekannten Mustern und so weiter). Der 62256 ist ein altes und zuverlässiges SRAM ohne große Neigung zu Einbrenneffekten, die bei weitem nicht so stark sind wie bei neuen hochintegrierten Bausteinen.
Es sind noch viele weitere Experimente mit anderen SRAM-Typen und unter anderen Betriebsbedingungen erforderlich. Einige Autoren haben über widersprüchliche Ergebnisse berichtet, was darauf hindeutet, dass die verbleibenden Einbrenneffekte nicht gut erforscht und verstanden wurden. Dies eröffnet ein neues und weites Feld für weitere Experimente!

Senken der Versorgungsspannung

Neben den oben erwähnten Methoden gibt es noch einen weiteren guten Ansatz zur Extraktion der eingebrannten Daten, den ich noch nicht ausprobiert habe. Dabei wird die Versorgungsspannung schrittweise bis zu dem Punkt gesenkt, an dem eine bestimmte Speicherzelle einen falschen Wert anzeigt oder sie beim Schreiben eines Bytes versagt. Eingebrannte und nicht eingebrannte Zellen werden dabei ständig miteinander verglichen. Normalerweise können nur bestimmte Bits in einer eingebrannten Speicherzelle nicht korrekt gelesen/geschrieben werden (zum Beispiel beim Schreiben von 0x00 oder 0xFF), je nach ihrem eingebrannten Zustand (0 oder 1). Dies liegt daran, dass die MOSFET-Gate-Schwellspannungen von eingebrannten Zellen im Vergleich zu Zellen ohne Einbrennen immer leicht höher oder niedriger sind.

Die Anwendung

Die „Burn-in-Methode“ lässt sich in Verbindung mit fortgeschrittener Steganografie zu einem Kanal geheimer Kommunikation ausbauen, vor allem, wenn er neuere Typen hochintegrierter SRAMs verwendet, die anfälliger für Burn-in-Effekte sind als der hier getestete gute alte 62256. Ein SRAM-Chip, der für diesen Ansatz geeignet ist, sollte nach einigen Stunden Erwärmung auf 80 °C (im eingeschalteten Zustand, mit einer geheimen, verschlüsselten Nachricht) eine signifikante Burn-In-Retention erreichen, wenn er die Burn-In-Effekte 10…15 Tage lang beibehalten kann, nachdem er bei Raumtemperatur ausgeschaltet und aus dem Schaltkreis entfernt wurde. Eine solche Kommunikation würde wie folgt funktionieren:



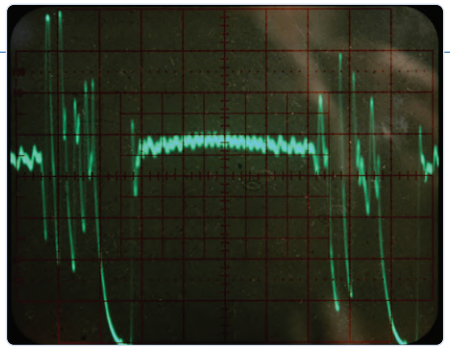
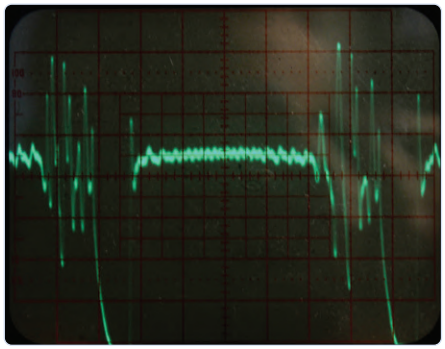


Bild 6. I_{dd} beim Schreiben der Bytes 0x00, dann 0xFF bis 0x7FF1 (links) und 0x204F (rechts).



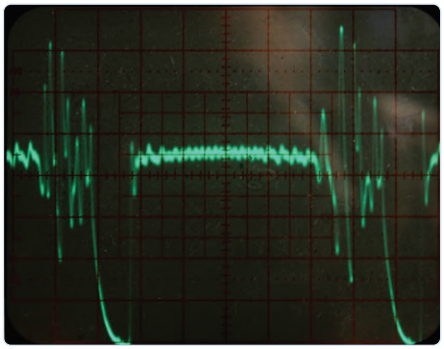


Bild 7. I_{dd} beim Schreiben der Bytes 0x66, dann 0x99 bis 0x7FF1 (links) und 0x204F (rechts).

1. Die Absenderin Alice verschlüsselt die geheime Nachricht für den Adressaten Bob und speichert sie dann im SRAM.
2. Alice erhitzt das SRAM (im eingeschalteten Zustand, mit der geheimen verschlüsselten Nachricht) 6…12 Stunden lang auf 80 °C.
3. Alice lässt das SRAM auf Raumtemperatur abkühlen, während es eingeschaltet ist.
4. Alice entfernt das abgekühlte SRAM aus der Schaltung.
5. Alice steckt den SRAM-Chip in einen Briefumschlag und schickt ihn per Post an Bob.
6. Wenn Spionin Eve die Post abfängt, sieht sie nur eine Tüte mit elektronischen Bauteilen. Selbst wenn sie versucht, die eingebrannten Daten wiederherzustellen, sieht das SRAM nicht verdächtig aus, weil die Nachricht verschlüsselt ist. Das bedeutet nicht, dass Alice die Daten nur zu diesem Zweck eingebrannt hat. Vielleicht wurde der Chip einfach aus einem Server entnommen, auf dem über einen sehr langen Zeitraum von beispielsweise einigen Monate bei 30 °C konstante Daten gespeichert wurden.
7. Bob erhält nach ein paar Tagen die Post und führt das gerade beschriebene Verfahren zur Datenwiederherstellung durch.
8. Bob entschlüsselt die Nachricht.

Fallen Ihnen noch andere Möglichkeiten ein, wie Absender Alice, Adressat Bob und Spionin Eve (und andere Beteiligte) die Effekte des SRAM-Einbrennens ausnutzen könnten? Es gibt viele!

Demonstration eines Cold-Boot-Angriffs

Die zweite Demonstration ist viel einfacher als die vorherige. Die nachfolgende Angriffsart ist wahrscheinlich für viele alte und neue RAM-Typen geeignet.
Das Abkühlen eines Chips auf bis zu -50 °C verringert die Leitfähigkeit von schwach dotierten Silizium-Mikroschaltungen nicht wesentlich, so dass der Chip weiterhin normal funktioniert, aber die Daten nach dem Abschalten der Stromversorgung erhalten bleiben, weil die Entladung der MOSFET-Gate-Kapazitäten viel langsamer erfolgt. Für das Experiment wird derselbe Aufbau verwendet, nur ohne den Heizwiderstand. Die acht Bar-Graph-LEDs zeigen verschiedene Lauflichtmuster direkt aus dem SRAM an. Wenn sie in regelmäßigen Mustern weiterlaufen (nach einem Stromausfall), bedeutet das, dass der SRAM-Inhalt erhalten geblieben ist. Wenn unregelmäßige, sprunghafte Muster erscheinen, bedeutet dies, dass der SRAM-Inhalt verloren

Bild 8. Kältespray sorgt für die nötigen Minusgrade für einen Cold-Boot-Angriff auf die Demoschaltung.

waren zu erwarten, da die Stromversorgungspins im ausgeschalteten Zustand über einen Widerstand im Kiloohm-Bereich „kurzgeschlossen" bleiben, so dass der V_{dd}-Pin als „geerdet" betrachtet werden kann.

Überraschend einfach

Wie Sie sehen, können einige Angriffe in der Praxis mit sehr billiger Ausrüstung und auch in Ihrem Heimlabor durchgeführt werden, zum Beispiel Cold-Boot-Angriffe, während andere Angriffe wie die Wiederherstellung von eingebrannten SRAM-Daten anspruchsvollere Hardware und mathematische Nachbearbeitung erfordern (auch wenn dies für einen Low-Budget-Spion durchaus möglich ist!)

Neue, hochintegrierte elektronische Systeme sind anfälliger für die hier beschriebenen Angriffe (außer vielleicht moderne Laser- und Tintenstrahldrucker im Vergleich zu alten Nadeldruckern oder Typenraddruckern). Die Verbesserung der hier gezeigten Ideen kann Sie dazu bringen, Ihre eigenen Angriffsgeräte und -verfahren und den Schutz dagegen zu entwickeln. Es gibt viel zu tun und keine Ausreden mehr für die typische Apathie der Entwicklungsingenieure des 21. Jahrhunderts! Ich bin mir sicher, dass eine gute Verschlüsselung heutzutage auch mit einem geringen Budget möglich ist. Je weiter die Technologie fortschreitet und je billiger und zugänglicher sie wird, desto besser funktioniert das Verfahren für Alice und Bob!

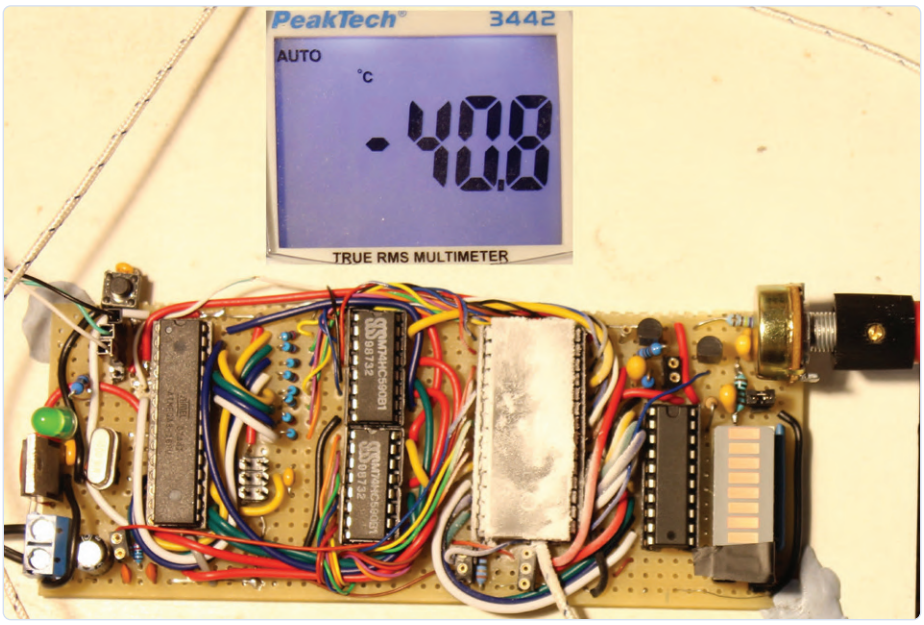


Bild 9. Das SRAM-IC wurde „tiefgefroren", um eine Datenerhaltungszeit von zehn Sekunden zu erreichen.

gegangen ist. Ich habe eine maximale Datenerhaltungszeit bei etwa -30 °C gemessen, was mit dem bekannten Gefrierspray „KÄLTE 75" leicht zu erreichen ist (**Bild 8**). Bei Raumtemperatur beträgt die Zeit der Datenerhaltungszeit nur 0,1 s, bei -30 °C sind jedoch bis zu 10 s zu erwarten.

Das SRAM-IC wird zunächst auf etwa -30 °C abgekühlt und dann die Einschaltdauer schrittweise erhöht, während die Temperatur mit dem Spray aufrechterhalten wird. Der 10-Sekunden-Datenerhalt wurde bei rund -40 °C erreicht (**Bild 9**). Diese Ergebnisse

WEBLINK

[1] Retronik: Speicher-Oszilloskop Tektronix 564 (1963), Elektor 5/2011: www.elektormagazine.de/magazine/elektor-201105/3804

210628-02

Haben Sie Fragen oder Kommentare?

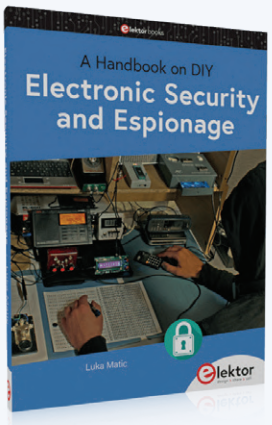
Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Senden Sie eine E-Mail an den Autor unter luka.matic@gmail.com oder an Elektor unter redaktion@elektor.de.

Ein Beitrag von

Text: **Luka Matic**
Herausgeber: **Jan Buiting**
Übersetzung: **Rolf Gerstendorf**
Layout: **Giel Dols**

PASSENDE PRODUKTE

- **Buch: L. Matic, *A Handbook on DIY Electronic Security and Espionage* (SKU 19903)**
  www.elektor.de/19903
- **E-Buch: L. Matic, *A Handbook on DIY Electronic Security and Espionage* (SKU 19904)**
  www.elektor.de/19904



- **E-Buch: J. Buiting, *Retronics, 80 tales of electronics bygones* (SKU 16885)**
  www.elektor.de/16885

Identifizierung von Bauteilen

Tipps & Tricks, Best Practices und andere nützliche Informationen

Von **David Ashton** (Australien)

Die Fähigkeit, die Teile auf einer Platine identifizieren zu können, ist eine Grundvoraussetzung für die Reparatur defekter Schaltungen. Mit dieser Fähigkeit ist es Ihnen auch möglich, einen Vorrat an Ersatzteilen anzulegen, der eines Tages nützlich sein könnte. Es ist jedoch nicht immer so einfach, die einzelnen Baueile zu identifizieren. Hier sind einige Tipps, die Ihnen dabei helfen.

Viele Leser sind wie ich Hobbyelektroniker, die auch schon einmal Teile aus alten elektronischen Geräten ausschlachten. Ich begann mich schon als Teenager - vor etwa 50 Jahren - für Elektronik zu interessieren und mein Vater, der im Vertrieb einer Firma für Buchhaltungsmaschinen tätig war, besorgte mir von dort Platinen mit Transistoren, Widerständen, Dioden und ein paar Kondensatoren. Die Leitungen auf der Unterseite der Platine waren verbogen, und selbst mit der großen Lötpistole, die mein Vater ebenfalls für mich besorgt hatte, hatte ich bei einigen Bauteilen Probleme, sie auszulöten. Aber das war in Simbabwe, das nicht gerade das Zentrum der Elektronikwelt war. Bauteile waren teuer und nicht immer leicht zu bekommen, also waren diese Platinen für mich Gold wert.

Die Transistoren steckten in sehr seltsamen Gehäusen, aber mit einem einfachen Transistortester, den ich nach einem Artikel in einer Zeitschrift gebaut hatte, konnte ich sie als NPN-Typen identifizieren. Allerdings waren sie mit *B686* gekennzeichnet und hatten einen Farbpunkt - braun, rot, orange, gelb oder grün - in einer Delle oben auf dem Gehäuse. Damals gab es noch kein Internet, und Datenbücher waren selten und teuer. Ich wusste zwar, dass mit *Bxxx* gekennzeichneten Transistoren oft aus der japanischen 2SB-Serie stammten, aber als ich in meinem Exemplar des ehrwürdigen *Towers International Transistor Selector Book* nachschlug, war der 2SB686 ein PNP-Leistungstransistor und meine waren eindeutig NPN-Typen. Es war also eine Herausforderung, die tatsächliche Typennummer dieser Transistoren herauszufinden. Und keine leichte!

Als ich dann zufällig eine Tabelle mit Transistordaten in der Zeitschrift *Practical Electronics* sah, fand ich den 2N2926 mit dem gleichen Gehäuse und den Hinweis, dass der Farbpunkt auf dem Gehäuse den Verstärkungsfaktor anzeigt. Der 2N2926 **(Bild 1**) ist ein seltsam aussehender Transistor, und es musste einfach derselbe sein, auch wenn meiner nicht als solcher gekennzeichnet war. Ich vergewisserte mich, dass die Verstärkungen meiner Transistoren mit denen in der Tabelle übereinstimmten, und damit war der Deal besiegelt. Der 2N2926 mit seinem Farbfleck ist ein so ungewöhnlicher Transistor, dass er es einfach sein musste, auch wenn er eine andere Kennzeichnung hatte.

Ich habe die meiste Zeit meines Lebens in der Elektronik- und Telekommunikationsbranche gearbeitet und schlachte immer noch alte Platinen aus - professionelle Geräte liefern oft qualitativ hochwertige Bauteile. Seit meiner Jugend habe ich viele Versuche unternommen, noch seltsamere „erbeutete" Bauteile zu identifizieren, und war nicht immer erfolgreich. Aber ich will Ihnen ein paar der Techniken und Methoden zeigen, auf die ich mich im Laufe der Jahre verlassen habe. Und ich habe ein Quiz zur Identifizierung von Bauteilen angefügt, damit Sie Ihre eigenen Fähigkeiten testen können. Das Quiz behandelt eine breite Mischung von Bauteiltypen, von sehr alten bis hin zu neueren SMD-Typen, und Beispiele für die Probleme und Techniken, die ich hier beschreibe.

Sammeln von Bauteildaten

Als ich mit der Elektronik anfing, waren Datenbücher schwer zu finden und ihr Gewicht in Gold wert. Heutzutage, wo das Internet zur Verfügung steht, ist das nicht mehr so wichtig, aber wenn Sie nützliche Informationen finden - Farbcodes, Herstellerinformationen, Datenblätter und so weiter - bewahren Sie sie in Ihren Favoriten oder besser als Download oder gedruckte Kopie auf. Ich habe immer noch die Tabelle mit den Transistordaten von vor so vielen Jahren.

Heute sollten Sie sich im Internet zurechtfinden. Es gibt viele gute Websites mit Datenblättern. Die Suchmaschine findet sie, aber Sie

müssen ihrer Suchmaschine auch unter die Arme greifen! Wenn Sie ein Datenblatt suchen, geben Sie „datasheet" in das Suchfeld ein. Und verwenden Sie die kürzestmögliche Teilenummer. Kürzlich hatte ich einige ICs mit der Aufschrift *2026-1SM*, und da ich mehrere davon besaß, hatte ich festgestellt, dass dies wahrscheinlich die Teilenummer war. Aber ich gab „2026 datasheet" in die Suchmaschine ein und erhielt sofort das Datenblatt für den MIC2026 von Micrel (jetzt Microchip) - einen Zweikanal-High-Side-Power-Schalter für USB.
Einige Datenblattseiten nehmen Ihre Teilenummer und geben Datenblätter an, die entweder genau übereinstimmen, mit den von Ihnen eingegebenen Daten beginnen oder nur diese enthalten - das kann nützlich sein, um Ihre Suche einzugrenzen. Ich speichere die meisten Datenblätter auf meiner Festplatte - man weiß ja nie, wann man sie wieder suchen muss - aber das ist eine persönliche Vorliebe. Ich habe am Ende einen Link zu meiner meistgenutzten Website für Datenblätter angegeben (und der Übersetzer hat seine liebste Halbleiteridentifizierungssuchmaschine hinzugefügt).

Wie man Datenblätter liest

Die meisten Datenblätter beginnen mit einer kurzen Beschreibung des Bauteils und den Absolutwerten. Es folgt die genaue Beschreibung der Verwendung, Pinbelegung und so weiter. Die Gehäuseinformationen befinden sich in der Regel am Ende, und Sie müssen sich oft vergewissern, dass das Bauteil, das Sie haben, auch wirklich dem Datenblatt entspricht - also die gleiche Anzahl von Pins und das gleiche Gehäuse besitzt.
Nicht alle Hersteller stellen die gleichen Bauteile in allen Gehäusebauformen her, so dass Sie sich eventuell umsehen müssen. In der Regel, aber nicht immer, hat ein Bauteil eines Herstellers die gleichen Spezifikationen wie das gleiche Bauteil eines anderen Herstellers.

Bauteile richtig erkennen

Manche Bauteile sehen einem anderen Bauteil sehr ähnlich. Heutzutage kann ich ein Bauteil, das wie ein Widerstand aussieht, anhand der Form und der Farbe des Bauteils mit ziemlicher Sicherheit als Kondensator oder als Induktivität identifizieren.
Die meisten Leute wissen, dass etwas mit der Aufschrift *2Nxxxx* ein Transistor ist, und ein IC-ähnliches Ding mit vier oder sechs Anschlüssen wahrscheinlich ein Opto-Isolator. Und ich weiß, dass ein transistorähnliches Bauteil mit der Bezeichnung *xxNyy* (etwa 35N60) oder eines mit der Bezeichnung *IRFxxx* (beispielsweise IRF540) ein FET ist. Das lernt man mit der Zeit …
Gerade bei der Recherche zu diesem Artikel habe ich festgestellt, dass ein *V* auf einem IC oft bedeutet, dass es von Vishay hergestellt wurde, was ich mir merken werde, da es mir in Zukunft vielleicht etwas Suchzeit erspart.

Wie man Bauteile testet

Etwa zu der Zeit, als ich auf diese 2N2926 stieß, baute ich aus einem alten Messgerät, einem Schalter, einem 500-kΩ-Poti und einer 1,5-V-Batterie einen einfachen Transistortester. Ich habe ihn heute noch, aber heutzutage besitzen die meisten Digitalmultimeter einen eingebauten Transistortester, und einige können auch Kondensatoren testen. FETs sind etwas schwieriger, aber man kann sie auch in einem durchschnittlichen Heimlabor testen. Ich besitze auch ein einfaches LCR-Meter, das ich oft benutze, obwohl ich gerne ein besseres hätte. Wenn Sie ein Bauteil als Transistor, FET, Kondensator oder was auch immer identifizieren und messen können, haben Sie schon den halben Weg hinter sich und können es vielleicht auch ohne weitere Nachforschungen verwenden. Einige SMDs, insbesondere Kondensatoren, sind überhaupt nicht gekennzeichnet, so dass Sie sie vor der Verwendung einfach nur messen müssen. Eine Pinzettensonde für Ihr Messgerät kann das Testen von SMD-Bauteilen erheblich erleichtern, und im Elektor Store gibt es ein sehr nützliches digitales Pinzettenmessgerät, das so ziemlich alles messen kann. Ein Link dazu befindet sich in der Textbox **Passende Produkte**.
ICs benötigen natürlich spezielle Tester, aber Sie können sich leicht selbst einen Opamp-Tester bauen und einen einfachen Tester für Logik-ICs kaufen, wenn Sie oft Logik-ICs verwenden. Bei ICs und Transistoren sind die Teilenummer und das Datenblatt das A und O, aber viele Bauteile, vor allem passive, sind schon brauchbar, wenn man nur ihren Wert kennt. Das Testen von Bauteilen ist ein Thema für sich.

Der viel sagende Aufdruck

Auf einem Transistor oder IC sind in der Regel mehrere Nummern aufgedruckt oder eingeprägt, und es lohnt sich, die Typennummer zu ermitteln. Bei kleinen SMDs werden die Präfixe oft weggelassen. Lernen Sie die Herstellerlogos kennen. Wenn Sie direkt zur Website

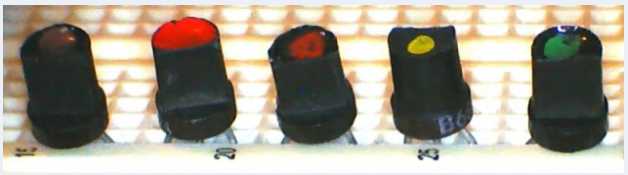


Bild 1. 2N2926-Transistoren. Die Farbe gibt die Transistorverstärkung h_{FE} an: braun 35…70, rot 55…110, orange 90…180, gelb 150…300 und grün 235…470.

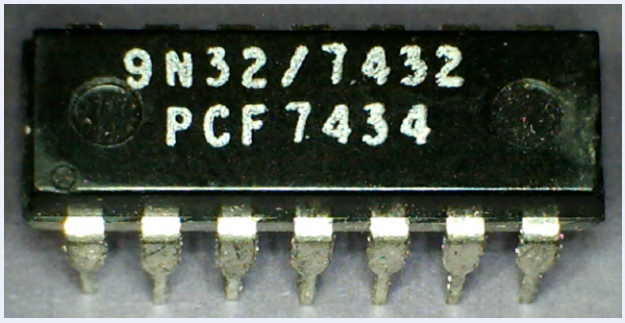


Bild 2. Ist dies ein 7432 oder ein 7434? Man könnte meinen, dass PCF ein Herstellerpräfix ist, ist es aber nicht (damals jedenfalls nicht!). Es gibt keinen TTL-Typ 7434, sondern es ist ein 7432, hergestellt in der 34. Woche des Jahres 1974. 9N32 ist ein weiterer Hinweis, denn viele 74xx-ICs waren auch mit 9Nxx gekennzeichnet, obwohl es sehr schwierig ist, Informationen darüber zu erhalten.

des Herstellers gehen, können Sie viel Zeit sparen. Die meisten Bauteile haben einen Datumscode, der früher aus Jahr und Woche in amerikanischer Notation bestand (etwa 8634 für 34. Woche 1986), aber heutzutage können es auch kryptische Chargencodes sein. Ein altes TTL-Logik-IC der Serie 74 aus dem Jahr 1974 mit dem Datumscode 74xx **(Bild 2**) kann ein echtes Rätsel sein! Wenn Sie mehrere Bauteile eines Typs haben, suchen Sie nach einem Code, der auf allen gleich ist. Das ist dann die Teilenummer, die anderen Codes sind Datums- oder Chargencodes, die nicht von Interesse sind.

Werte auf passiven Bauteilen werden entweder direkt angegeben (zum Beispiel 47 pF) oder als Zahlen- oder Widerstandsfarbcode in der Form *Digit1*, *Digit2*, *Multiplikator* (die Anzahl der Nullen) auf dem Bauteil. Bei SMD-Drosseln wird der Wert oft in Mikrohenry angegeben, also *3R3* für 3,3 µH oder *333* für 33 mH (33.000 µH). Kondensatoren können in Picofarad angegeben werden. Ein Tantalkondensator mit der Bezeichnung 227 hat 22×10^7 pF = 220 µF. Einige Bauteile können fünf oder sechs Ringe mit Farbcodes haben, verwirrend, aber das Internet ist eine große Hilfe bei der Entschlüsselung.

Die meisten kleinen SMD-Kondensatoren sind überhaupt nicht beschriftet, also nutzen Sie Ihre Fähigkeiten und Geräte zur Bauteilprüfung, um sie zu identifizieren. Und besorgen Sie sich eine Lupe oder ein USB-Mikroskop (damit habe ich die meisten Fotos für diesen Artikel gemacht), denn damit ist es viel einfacher, die winzigen Beschriftungen auf kleinen Bauteilen zu erkennen.

Im Internet gibt es viele Quellen, die Ihnen helfen können - suchen Sie nach „IC-Hersteller-Logos“ oder „SMD-Codes“, wenn Sie weitere Informationen benötigen. Und schlagen Sie „EIA-96“ nach, um SMD-Widerstände mit einem seltsamen Code aus zwei Zahlen und einem Buchstaben zu entschlüsseln.

Berücksichtigen Sie den Kontext

Wenn Sie Teile von Platinen ablöten oder anderweitig wissen, woher das Bauteil stammt, kann Ihnen schon das einen Hinweis darauf geben, um was es sich handelt. Eine Stromversorgung enthält wahrscheinlich ein Schaltwandler-PWM-IC, während auf einer Audioplatine eher Opamps zu finden sein dürften.

Nicht alles ist bestimmbar!

Ich habe einen Beutel mit Transistoren, die mit 0V8F beschriftet sind und sich hartnäckig weigern, identifiziert zu werden. SMD-Bauteile können schwer oder gar nicht zu identifizieren sein, da sie oft mit verkürzten Teilenummern versehen sind. Selbst mit den umfangreichen Ressourcen im Internet ist es nicht einfach.

Seien Sie wählerisch

Ich habe die Platinen, die ich als Kind bekam, die mit den umgebogenen Bauteilanschlüssen, mit Haut und Haar ausgeschlachtet. Heutzutage fasse ich solche Bauteile nur noch an, wenn sie wirklich etwas Besonderes sind, sonst ist es der Mühen nicht wert.

Elektrolytkondensatoren sollten immer getestet werden, vor allem große Typen für die Stromversorgung, und achten Sie auf „Kuppeln“ an den Oberseiten - ein untrügliches Zeichen dafür, dass sie trocken oder undicht geworden sind. Ältere Bauteile wie Kohlewiderstände sind es wirklich nicht wert, aufbewahrt zu werden, und ältere Elektrolytkondensatoren sind im Verhältnis zu ihren Nennwerten oft viel größer als moderne Typen. Viele moderne SMD-ICs haben sehr feine Anschlussdrähte oder sind BGA-Typen (Ball-Grid-Array), für deren Einbau und Ausbau auf der Platine spezielle Geräte erforderlich sind. Wenn Sie also keine besondere Verwendung dafür sehen, werfen Sie sie weg. Wenn Sie glauben, dass Sie es gebrauchen können, identifizieren Sie es sicher, bevor Sie sich die Mühe machen, es auszulöten. Wenn Sie in der Lage sind, Bauteile aus alten Platinen zu identifizieren und zu verwenden, kann sich die Zeit, die Sie dafür aufwenden, durchaus lohnen. Sie können qualitativ hochwertige Bauteile retten, und wenn Sie sie systematisch aufbewahren, brauchen Sie oft keine Bauteile für ein Projekt kaufen. Wenn Sie fit sind bei der Identifizierung von Bauteilen, können Sie diese Fähigkeiten gut bei Reparaturversuchen an Platinen oder Geräten gebrauchen.

210024-02

Ein Beitrag von

Idee, Text und Illustrationen: **David Ashton**
Redaktion: **Clemens Valens**
Übersetzung: **Rolf Gerstendorf**
Layout: **Harmen Heida**

Haben Sie Fragen oder Kommentare?

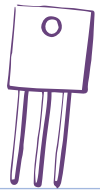
Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Schicken Sie dem Autor eine E-Mail an stn564@yahoo.com.au oder kontaktieren Sie Elektor unter redaktion@elektor.de.

PASSENDE PRODUKTE

- **Andonstar AD407 HDMI Digital Microscope (SKU 19079)**
  www.elektor.de/19079
- **Miniware DT71 Mini Digital Tweezers (SKU 19422)**
  www.elektor.de/19422
- **OWON OW16B Digital Multimeter with Bluetooth (SKU 18780)**
  www.elektor.de/18780

WEBLINKS

[1] Gute Datenblatt-Seite mit vielen Optionen:
https://www.alldatasheet.com/
[2] Suchmaschine für Halbleiteridentifizierung:
http://www.aufzu.de/semi/

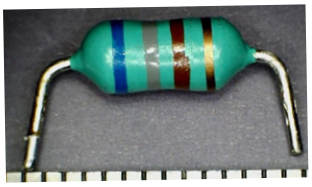


Das Quiz zur Bauteilindentifikation

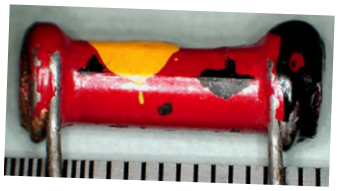
DAS QUIZ ZUR BAUTEILINDENTIFIKATION

Machen Sie mit und finden Sie heraus, wie Sie abschneiden! Die Skalen geben stets Millimeter an.

A – Was ist das? Können Sie es auf den ersten Blick erkennen?

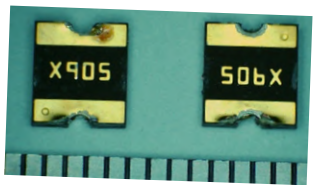


B - Und das hier?

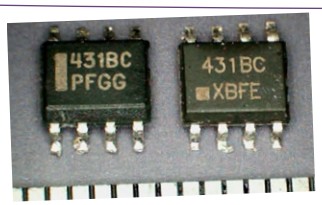


C - Was ist das und welchen Wert hat das Bauteil?

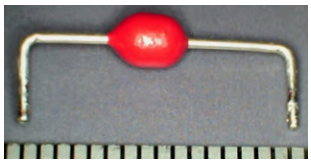
D - Was ist das für ein seltsam aussehendes IC? Seltsames Gehäuse, aber schöne Bauteilnummer LH1540. Oder ist 431 die Teilenummer?



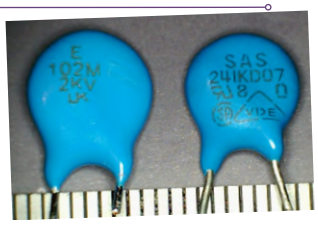
E - 506X oder X905? Diese SMD-Bauteile weisen einen sehr geringen Widerstand auf (einige Ohm). Um welche Teile handelt es sich?



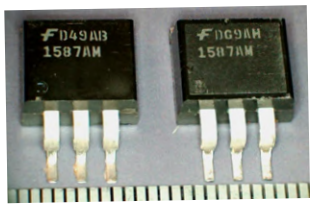
F - Was könnte dieses IC sein?



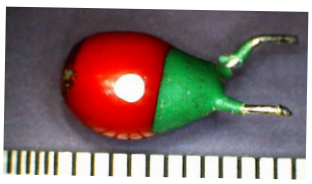
G - Das ist ein seltsames Bauteil. Es stehen die Ziffern 431 darauf, die kaum sichtbar sind.



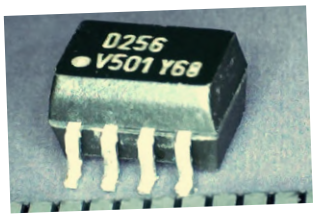
H - Handelt es sich um denselben Bauteiltyp oder um einen anderen?



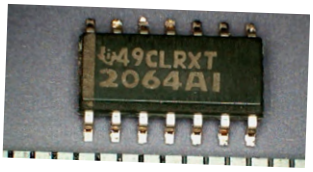
I - Was ist das? Können Sie ein Datenblatt finden?



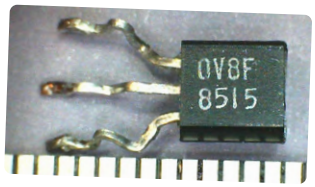
J - Ist das etwa ein Tantal-Kondensator? Oder etwas anderes?



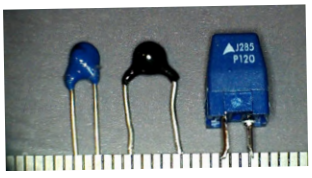
K – Gekennzeichnet mit D256. Beachten Sie die Dicke des ICs!



L - Das hier ist ziemlich einfach. Aber wer ist der Hersteller?

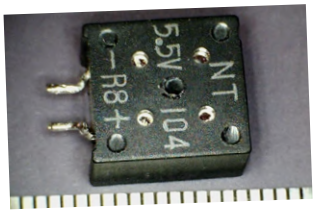


M - Erinnern Sie sich? Davon habe ich Ihnen erzählt! Was ist das?



N - Die Bauteile sind sich ähnlich. Aber was sind sie? Tantal-Kondensatoren?

O - Keine Beschriftung. Aber wozu ist das Bauteil gut? Was sind das für „Griffe“? Kühlkörper?



P – Dieses Gehäuse hat vier Löcher (zwischen den Schriftzeilen). Ist das vielleicht ein Summer oder gar ein Temperatur- oder Feuchtigkeitssensor?

DAS QUIZ - DIE AUFLÖSUNG

A - Dies ist ein Röhrenkeramikkondensator – 22 nF. Er hat nicht ganz die Form eines Widerstands und die falsche Farbe. Mit abgeblättert ist - im Uhrzeigersinn – 47 pF, 1 nF, und ... ein 1%iger 486-kΩ-Widerstand, der ob seiner ungewöhnlichen Länge am wenigsten nach einem normalen Widerstand aussieht! Als ich ihn das erste Mal gesehen habe, war ich verblüfft. Vertrauen in die eigene Erfahrung ist gut, aber: Kontrolle ist besser!

B – Während der Kondensator in A zu lang für einen Widerstand war, ist dieser hier zu kurz und zu dick, und die grün-blaue Farbe wäre ungewöhnlich für einen Widerstand. Es ist eine Induktivität von 680 nH. Hier ist sie (oben links) zusammen mit Induktivitäten von (im Uhrzeigersinn) 470 nH, 47 nH und 820 nH. Der 47-nH-Typ besitzt die üblische cremefarbene Widerstandsfarbe, ist aber ein bisschen zu kurz für einen Widerstand.

C - Dies ist ein sehr alter Widerstand mit einem Body-Tip-Spots-Farbcode. Rot-schwarz-gelb bedeutet 200 kΩ und keine Toleranz, also wahrscheinlich 20%. Der Widerstand weist gemessene 225 kΩ auf, also eine Toleranz von nur 12,5%. Wahrscheinlich lohnt es sich nicht, ihn zu behalten, außer aus sentimentalen Gründen. Und weitere Eigenschaften? Der Widerstand ist wahrscheinlich 80+ Jahre alt!

D - Diese Bauteile verursachen eine Menge Ärbeit. Man kann den LH1540 - einen Opto-Isolator - leicht in verschiedenen Gehäusen finden, aber nicht in einem solchen. Es steht auch 431 drauf – ist es vielleicht eine seltsame Version des TL431? Schließlich sah ich ganz hinten in einem Vishay-Datenblatt ein Bild dieses Gehäuses (genannt 8 PowerSOIC) mit keinen weiteren Informationen außer einem Link zu einem Datenblatt über den Footprint. Das wiederum führte mich zum LH1540ACD-Datenblatt, das sich auf dieses spezielle Gehäuse bezieht. Manchmal muss man eben suchen.

E - Niedriger Widerstand? Was hat das zu bedeuten? Aus einer Laune heraus habe ich ihn an mein Netzteil angeschlossen und den Strom langsam erhöht. Bei etwa 700 mA wird das Bauteil warm und ziemlich hochohmig. Es handelt sich um eine Polyfuse - ein sehr praktisches Bauteil als selbstrückstellende Schmelzsicherung. Die Codes auf dem Gehäuse waren allerdings keine Hilfe.

F - Erfahrungsgemäß handelt es sich bei allem, auf dem 431 steht, um die TL431-Spannungsreferenz von Texas Instruments oder eines Second-Source-Anbieters. Es handelt sich um eine weit verbreitete, vielseitige Spannungsreferenz, die auch für die Spannungsmessungen in Stromversorgungen verwendet werden kann. Dies ist die SMD-SOIC-8 Version - normalerweise sind sie in TO-92-Transistorgehäusen untergebracht.

G - Dieser Baustein hat mich lange Zeit vor ein Rätsel gestellt. Man kann gerade noch die Zahl 431 auf dem roten Klecks erkennen, aber besagte Spannungsreferenz ist es natürlich nicht. Es handelt sich um einen 430 VDR (Voltage Dependent Resistor oder Varistor), eine Art Überspannungsableiter. Das Bauteil ist physikalisch sehr klein, so dass ich nicht erwarten würde, dass es einen hohen Stromstoß absorbieren kann.

H – Die Bauteile sind in Größe und Farbe fast identisch. Auf der linken Seite ist ein Kondensator, auf der rechten Seite ein VDR (Voltage Dependent Resistor) zu sehen. Auf dem Kondensator ist der Wert (102 = 1 nF) aufgedruckt und er ist für 2 kV ausgelegt. Auf dem VDR steht 241 (24 und eine Null) – es ist also ein 240 VDR. Es ist unwahrscheinlich, dass Sie einen VDR mit 1000 V sehen, und er hat auch keine andere Spannung aufgedruckt! Auch hier gilt: Erfahrung hilft, und man sollte nie etwas als selbstverständlich annehmen!

*I - 1587AM ist für beide Teile gleich. Wenn Sie nach „1587 datasheet“ suchen, werden Sie zu Alldatasheet.com weitergeleitet. Wählen Sie Bauteile aus, die auf 1587 enden (*1587), wird Ihnen der LDO-Spannungsregler LM1587 angezeigt. Der Regler ist von Texas, aber wenn Sie das Fairchild-Logo erkennen und nach „Fairchild 1587“ suchen, werden Sie zu Fairchild RC1587 weitergeleitet, bei dem es sich um das gleiche Bauteil handelt.*

J - Dies ist ein Tantal-Kondensator von 20 µF aus den alten Zeiten, als sie noch farblich gekennzeichnet waren. Hier sind einige seiner Verwandten zu sehen, 2,2 µF, 3,3 µF und 5 µF. Nur zwei Farbcodes! Seien Sie vorsichtig mit alten „Tantys“, sie neigen zu Kurzschlüssen. Aber sie sind hübscher als die neueren!

K - Die Dicke des ICs ist ein Anhaltspunkt. ICs wie dieses sind normalerweise Optoisolatoren. Ein weiterer Anhaltspunkt ist das „V“. Bei der Recherche zu diesem Artikel habe ich herausgefunden, dass dies normalerweise auf einen IC von Vishay hinweist - siehe Antwort D oben. Das kann Ihnen manchmal eine Menge Zeit ersparen. Die vollständige Bezeichnung lautet ILD2561, und es handelt sich um einen zweikanaligen Optoisolator mit AC-Eingängen und mit gut abgestimmten Stromübertragungswerten (Current Transfer Ratio, CTR). Schön.

L - Sehen Sie das Logo von Texas Instruments kurz vor der 49 in der ersten Zeile? Gehen Sie auf die TI-Website, geben Sie 2064 ein, und Sie erhalten drei Möglichkeiten: eine Stromversorgungsplatine, einen Stromverteilungsschalter mit acht Pins und diesen TLE2064 - eine verbesserte Version des TL064 Vierfach-JFET-Eingangsoperationsverstärkers. Wie üblich gibt es keine Präfixe auf SMD-Bauteilen.

*M - Hier ist mein nicht identifizierbarer 0V8F-Transistor im Standard TO-92-Gehäuse, auf den im Text Bezug genommen wird. Zehn Punkte für den ersten Leser, der ihn mit Hersteller und Datenblatt identifizieren kann! Ich habe ein paar davon und die Typennummer ist **0**V8F, die anderen Nummern sind ein Datumscode und variieren. Und ja, ich habe es auch mit **O**V8F versucht!*

N - Die kleinen blauen und schwarzen Bauteile sind NTC-Thermistoren aus Klimaanlagen-Sensoren. Wenn die Temperatur steigt, sinkt der Widerstand. Der große ist ein PTC-Thermistor, der eher zum Schutz dient. Wenn zu viel Strom fließt, erwärmen sie sich und werden hochohmig, um den Strom zu begrenzen - wie die Polyfuses oben in E.

O - Ich konnte nicht widerstehen, das hier mit aufzunehmen. Dachten Sie, es sei ein Stromsensor? Sie haben recht! Das Bild stammt aus dem Datenblatt des Hall-Effekt-Stromsensors ACS756 von Allegro. Die „Griffe“ sind der Stromfpfad, und die Stifte sind der Hall-Sensor. Bis zu 3 kV Isolierung, gut geeignet für Stromversorgungen.

P - Dies ist ein Superkondensator: 104 = 100.000 µF = 0,1 F. Beachten Sie, dass der Wert in µF angegeben ist, nicht in pF wie bei anderen Kondensatoren. 5,5 V ist eine übliche Spannung für diese Kondensatoren. Sie werden in der Regel für die Sicherung von Speichern oder Echtzeituhren verwendet. Hier ist er mit einem üblichen 0,1-F-Supercap dargestellt. Und warum diese Lücher im Gehäuse? Das weiß ich auch nicht!

Ein Touch-Schalter ohne Touch

Von **Mathias Claußen** (Elektor)

Eine Idee, die in der Pandemie geboren wurde: ein Lichtschalter, der eine Handgeste auswertet und so berührungslos das Licht ein- und ausschaltet. So können weniger Viren und Bakterien übertragen werden, was vor allem in stark frequentierten Räumen wie Großraumbüros sinnvoll wäre. Der Artikel zeigt das Konzept und die praktische Umsetzung mit einem ESP32 und einem Shield, das Gesten auswerten kann.

Wie das Projekt begann

Während die erste COVID-19-Welle Anfang 2020 um die Erde lief, wurden in den Büros Maßnahmen ergriffen, um die Ausbreitung einzudämmen. Die Desinfektion von Oberflächen und gemeinschaftlich genutzten Gegenständen war eine der Maßnahmen, die täglich durchgeführt wurde. Und während man sich bei der Nutzung von Kaffeemaschine und Teekocher gut arrangieren konnte, blieb ein Problem: die Lichtschalter für die Großraumbüros und Flure.

So wurde im Elektor-Labor die Idee geboren, sich mit einem Lichtschalter zu beschäftigen, bei dem keine Berührung notwendig ist, um zwischen An und Aus zu unterscheiden.

Bei einem Blick in den Elektor-Shop [1] fand der Autor ein *3D Gesture & Tracking Shield* für den Raspberry Pi (**Bild 1**). Eventuell ließ sich mit diesem eine Lösung bauen?

Der MGC3130

Ein Lichtschalter, der berührungslos arbeitet, war schon Anfang 2019 auf der hackster.io-Plattform zu finden: Der *Open Smart Switch* [2] von James Rowley und Mark Omo. In dem Projekt wurde ein MGC3130 von Microchip verwendet. Das *3D Gesture & Tracking Shield* für den Raspberry Pi verwendet denselben Chip, auch einen

Warnung: Alle in diesem Artikel vorgestellten Schaltungen arbeiten mit Netzspannung. Weder die Schaltungen noch die hier vorgestellten Platinen sind hinsichtlich ihrer Funktion und hinsichtlich ihrer Sicherheit getestet. Es wird davon abgeraten, die Platinen ungeprüft nachzubauen oder zu betreiben. Ein Nachbau dieses Projekts geschieht, wie immer, auf eigene Gefahr und sollte nur von Personen mit Fachkenntnis durchgeführt werden!

MGC3130. Auf dem YouTube-Kanal von ElektorTV [3] wurden der Chip und seine Möglichkeiten schon ausführlich besprochen.

Der MGC3130 ist ein E-Feld-Sensor, ein Chip, der ein elektrisches Feld über eine Sensoroberfläche aufspannt und Veränderungen dieses Feldes misst. Aus diesen Veränderungen lässt sich berechnen, wo sich Objekte wie eine Hand befinden. **Bild 2** zeigt die prinzipielle Funktion des Sensors und des erzeugten Feldes. Neben der Erkennung und Berechnung der Position eines Fingers können zusätzlich 3D-Gesten ausgewertet werden: das „Air-Wheel" und „Flick" in allen vier Richtungen. Ein Flick ist das Bewegen eines Fingers in der Luft über den Sensor hinweg.
Der Chip selbst kann mit den von Microchip bereitgestellten Tools kalibriert und konfiguriert werden. Um den Chip in ein eigenes Design zu überführen, sind alle nötigen Information auf der Produktseite [4] verfügbar.
Als Interface zur Außenwelt stellt der Chip seine Daten per I^2C bereit. Zusätzlich zu den Daten und der Taktleitung benötigt der Chip noch ein Reset und eine bidirektionale Interrupt-Leitung. Die Details des Interfaces sind im Dokument DS40001718E [5] von Microchip bereitgestellt. Wer den Chip auf einer eigenen Platine einsetzen möchte, sei darauf hingewiesen, dass diese mindestens vier Lagen benötigt. Auch muss dann der Chip passend zur Platine parametriert und konfiguriert werden.

MCU der Wahl

Bei der MCU standen mehrere Optionen zur Auswahl. Darunter der Raspberry Pi Pico, ein STM32 Blue Pill, ein bewährter ATmega328, der ESP8266 oder der ESP32. Für das Verarbeiten der Daten ist keine große Rechenleistung nötig, ebenso wenig wie für das Betätigen des Schalter-Relais.
Bei der Auswahl stellt sich an dieser Stelle die Frage, was an Funktionen implementiert werden soll. Eine Integration in ein Hausautomatisierungssystem wie ESPHome wäre erstrebenswert, damit der Schalter von einer zentralen Stelle geschaltet werden kann. Auch kommt die Frage auf, wie eventuelle Firmwareupdates in das System eingespielt werden sollen. Bei dem ESP8266 oder dem ESP32 wäre das über eine WLAN-Verbindung möglich.
Für dieses Projekt wird deshalb ein ESP32 gewählt. Er hat WLAN, Bluetooth und eine ausreichende Menge Flash an Bord, um Updates per WLAN durchzuführen. Als Hardware wurde ein *ESP32 Pico Kit* (**Bild 3**) gewählt.

Das Grundkonzept

Wie bei jeder Entwicklung steht am Anfang ein Grundkonzept mit den wichtigsten Bestandteilen, hier ein Sensor in Form des *3D Gesture & Tracking Shields* für den Raspberry Pi und ein ESP32 in Form eines *ESP32 Pico Kits*. Um später einen Schalter zu haben, der das Licht ein- oder ausschalten kann, wird noch ein Relais

Bild 1. 3D Gesture & Tracking Shield.

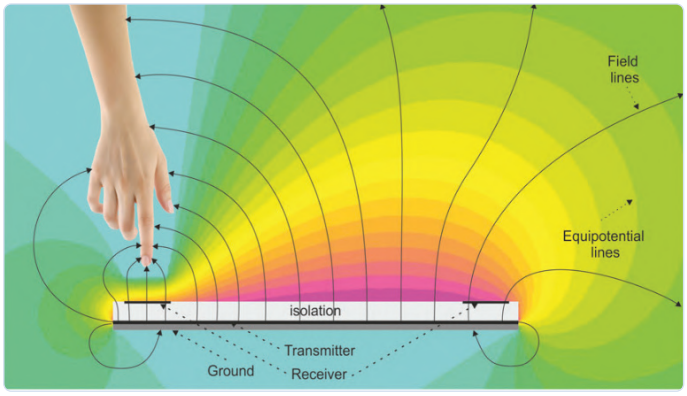


Bild 2. MGC3130 Sensorfeld (Quelle: Microchip).

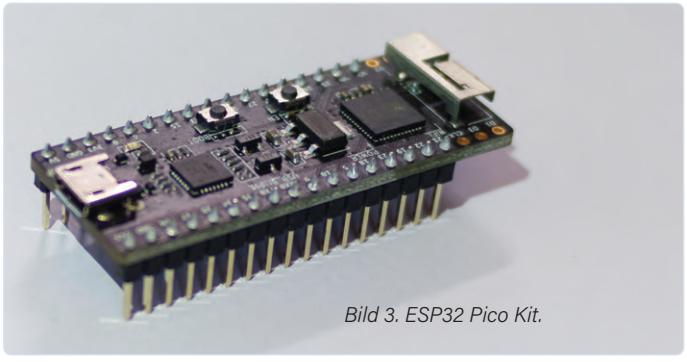


Bild 3. ESP32 Pico Kit.

benötigt. Zu guter Letzt muss noch eine Stromversorgung hinzugefügt werden. Wenn man nun noch etwas Software hinzufügt, hat man dann ein fertiges Produkt?
Soweit die Theorie, und wie viele von uns wissen, ist dies nicht das bevorstehende Finale eines Projekts, sondern der teilweise verschlungene Weg, mit dem die Entwicklung anfängt. Das Ganze teilt sich hier in drei Pfade auf, die miteinander verflochten sind. Einmal die Software, denn etwas muss am Ende des Tages die Sensordaten auswerten und entsprechende Aktionen durchführen. Der zweite Pfad ist die Elektronik und damit eine Platine. Alle Teile müssen miteinander verbunden werden. Dazu ist es nötig,

Bild 4. Unterputzdose (Quelle: Würth [9]).

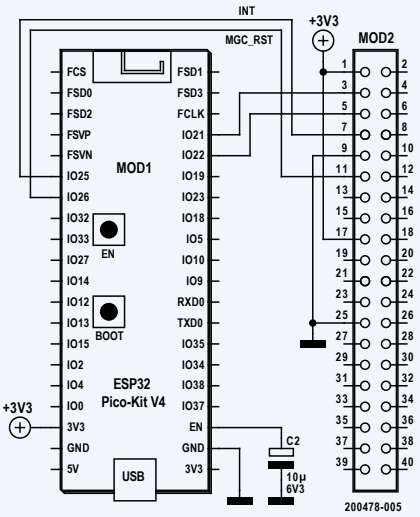


Bild 5. Schaltplan für den Testaufbau.

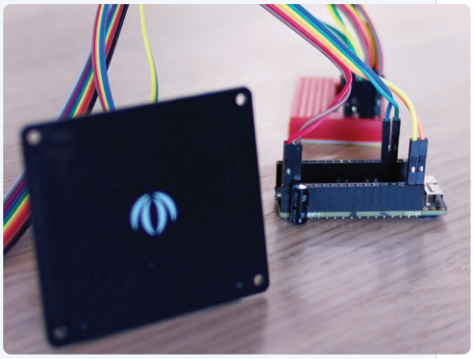


Bild 6. Testaufbau mit Steckbrett.

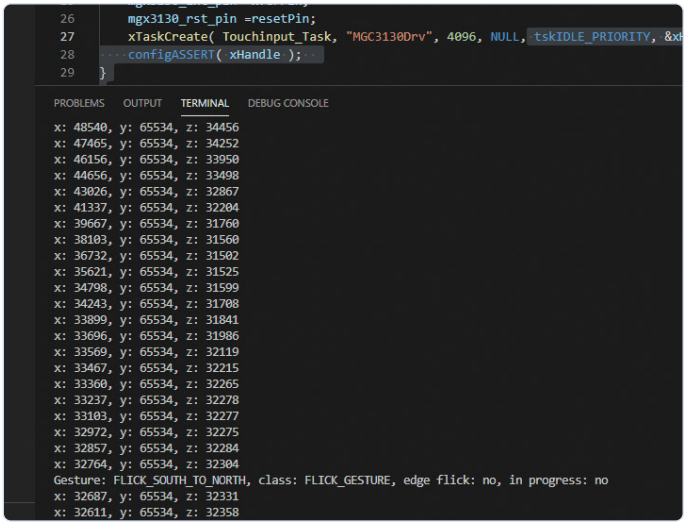


Bild 7. Erste Daten vom Sensor.

einen Schaltplan zu erstellen und die elektrischen Parameter für die Schaltung zu bestimmen. Idealerweise wird bei diesem Prozess schon die Rückmeldung der Softwareentwickler eingeholt, was das Pinout betrifft oder ob Sonderwünsche bestehen. Auch wenn Software sich schneller anpassen lässt als Hardware, so ist es immer ärgerlich, wenn Unschärfen im Hardwaredesign dann mühevoll durch die Softwareentwickler ausgeglichen werden müssen.

Der dritte Pfad ist das mechanische Design eines Gehäuses. Der Schalter soll ja nicht nur eine Platine werden, die freifliegend an ein paar Drähte angeschlossen wird. Entweder entwickelt man seine Platine und baut ein Gehäuse drumherum. Oder man hat gegebene Abmessungen für den Raum, den man nutzen kann. Bei diesem Projekt wird der Raum vorgegeben und es soll versucht werden, die Elektronik in ein Volumen von 65 mm x 55 mm x 25 mm unterzubringen. Diese ist nicht ganz passend für die meisten europäischen Unterputzdosen (**Bild 4**). Die Abmessungen passen aber schon einmal für die meisten Brüstungskanäle, die in Büros verbaut sind. Zusätzlich kann abgeschätzt werden, ob eine weitere

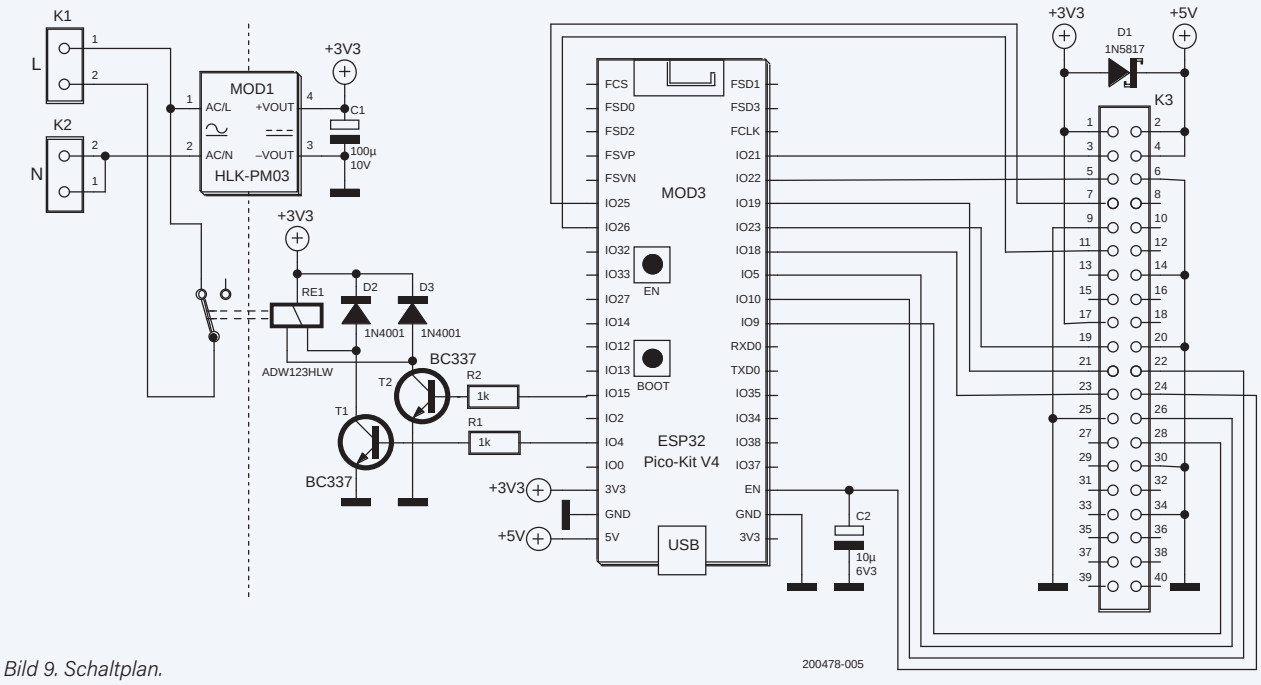


Bild 9. Schaltplan.

Bild 8. Hi-Link AC/DC-Konverter.

Miniaturisierung mit den gewählten Komponenten für den Einbau in bestehende Unterputzdosen möglich wäre.

Erste Tests auf dem Steckbrett

Für die ersten Versuche, das *3D Gesture & Tracking Shield* für den Raspberry Pi mit dem ESP32 zu verbinden, wird der Schaltplan aus **Bild 5** verwendet. Der Aufbau erfolgt auf einem Steckbrett (**Bild 6**), um die Funktionalität testen zu können. Für die Ansteuerung des MGC3130 gibt es mehrere Bibliotheken. Die Bibliothek, die in diesem Projekt verwendet wird, stammt von Seeed Studio [6] und ist für den Einsatz auf einem Raspberry Pi gedacht.

Für den ESP32 mussten ein paar Anpassungen bei der I²C-Schnittstelle und den verwendeten I/O-Pins vorgenommen werden. Nach diesen Modifikationen konnte die Bibliothek dann für erste Tests verwendet werden. In **Bild 7** sind die Ausgaben von Positionen und erkannten Gesten zu sehen.

Stromversorgung und Relais

Etwas, das auf dem Steckbrett noch nicht berücksichtigt war, ist die Stromversorgung und das Relais. Unser finaler Schalter soll mit Netzspannung betrieben werden können. Dazu brauchen wir ein Netzteil, das uns 5 V oder 3,3 V bereitstellt und genug Leistung bietet, um den ESP32, das *3D Gesture & Tracking Shield* für den Raspberry Pi sowie das Relais zu versorgen.

Eines der kompakteren Netzteilmodule ist das HLK-PM03 oder HLK-PM01 von Hi-Link (**Bild 8**). Diese liefern 3 Watt Ausgangsleistung, also 600 mA bei 5 V und etwa 900 mA bei 3,3 V. Für den Betrieb des ESP32 benötigen wir etwa 500 mA (2,5 W bei 5 V, 1,65 W bei 3,3 V). Als Relais soll ein ADW12 von Panasonic zum Einsatz kommen, das nur einen kurzen Impuls bei 3,3 V mit 67 mA (220 mW) benötigt. Das *3D Gesture & Tracking Shield* benötigt selbst noch einmal etwa 20 mA bei 3,3 V (66 mW). Wenn wir nun alles zusammenzählen, so müssen wir mindestens etwa 2,8 W bereitstellen können, falls wir das ESP32 Pico Kit mit 5 V versorgen. Damit würde das Netzteilmodul sehr weit an sein Leistungsmaximum bei 5 V Ausgangsspannung gebracht.

Warum das ESP32 Pico Kit bei 3,3 V weniger Leistung benötigt als bei 5 V, liegt an dem verwendeten LDO. Dieser erzeugt die nötigen 3,3 V für den ESP32 aus einer 5-V-Eingangsspannung. Die Spannungsdifferenz zwischen Eingangs- und Ausgangsspannung wird dann in Abhängigkeit des Stromes durch den LDO in Wärme umgesetzt. Sparsamer wäre es deswegen, direkt die 3,3 V bereitzustellen. Daher wird das HLK-PM03 in dem Projekt verwendet und wir erhalten so etwa 900 mA bei 3,3 V.

Für das Relais wird ein Panasonic ADW1203HLW gewählt, ein einpoliges Relais, das bei maximal 277 VAC für einen Nenn-Kontaktstrom von 16 A ausgelegt ist. Dies ist ein kompaktes selbsthaltendes Relais, das mit 24 mm x 10 mm x16 mm eine überschaubare Menge Platz einnimmt. Zum Setzen oder Rücksetzen des Relais ist nur ein kurzer Impuls nötig und kein permanenter Strom durch die Spule des Relais.

Anzeige

UNSER SORTIMENT
VON TECHNIKERN FÜR TECHNIKER
The best part of your project: www.reichelt.de

Nur das Beste für Sie – von über 900 Markenherstellern
Unsere Produktmanager sind seit vielen Jahren bei reichelt tätig und kennen die Anforderungen unserer Kunden. Sie stellen ein breites Spektrum an Qualitätsprodukten zusammen, optimal auf den Bedarf in Forschung & Entwicklung, Instandhaltung, IT-Infrastruktur und Kleinserienproduktion sowie auf Maker zugeschnitten.

Arduino bei reichelt
Der TinkerKit Braccio ist ein voll funktionsfähiger Roboterarm, der über Arduino gesteuert wird. Er kann auf verschiedene Weise für Aufgaben, wie das Bewegen von Objekten zusammengebaut werden.
- SpringRC SR431 - Dual Output Servo
- maximale Reichweite: 80 cm
- maximale Höhe: 52 cm
- Traglast: maximales Gewicht bei 32 cm Reichweite: 150 g

ARDUINO

Bestell-Nr.: ARD TINKER BOT
234,99

Gleich entdecken ▶ www.reichelt.de/arduino

TAGESPREISE! Preisstand: 15. 2. 2022

- Top Preis-Leistungs-Verhältnis
- über 120.000 ausgesuchte Produkte
- zuverlässige Lieferung – aus Deutschland in alle Welt

reichelt elektronik – The best part of your project

www.reichelt.de
Bestellhotline: +49 (0)4422 955-333

Es gelten die gesetzlichen Widerrufsregelungen. Alle angegebenen Preise in € inklusive der gesetzlichen MwSt., zzgl. Versandspesen für den gesamten Warenkorb. Es gelten ausschließlich unsere AGB (unter www.reichelt.de/agb, im Katalog oder auf Anforderung). Abbildungen ähnlich. Druckfehler, Irrtümer und Preisänderungen vorbehalten. reichelt elektronik GmbH & Co. KG, Elektronikring 1, 26452 Sande, Tel.:+49 (0)4422 955-333

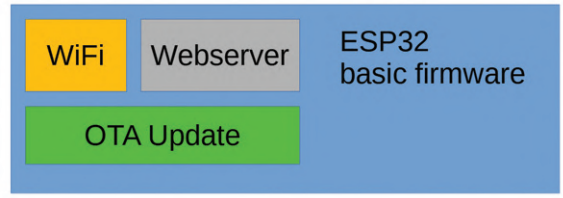


Bild 10. Komponenten der Basisfirmware.

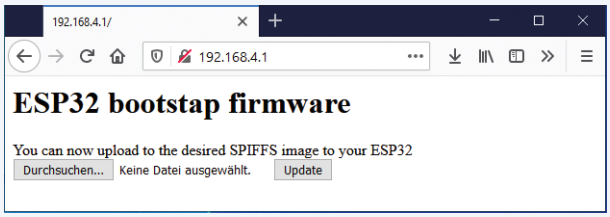


Bild 11. SPIFF-Upload per Webbrowser.

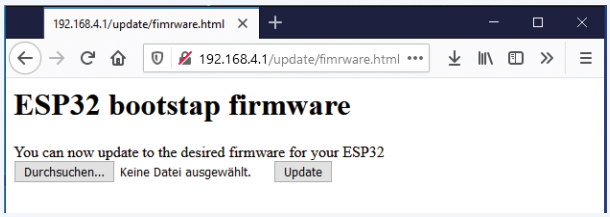


Bild 12. OTA-Update per Webbrowser.

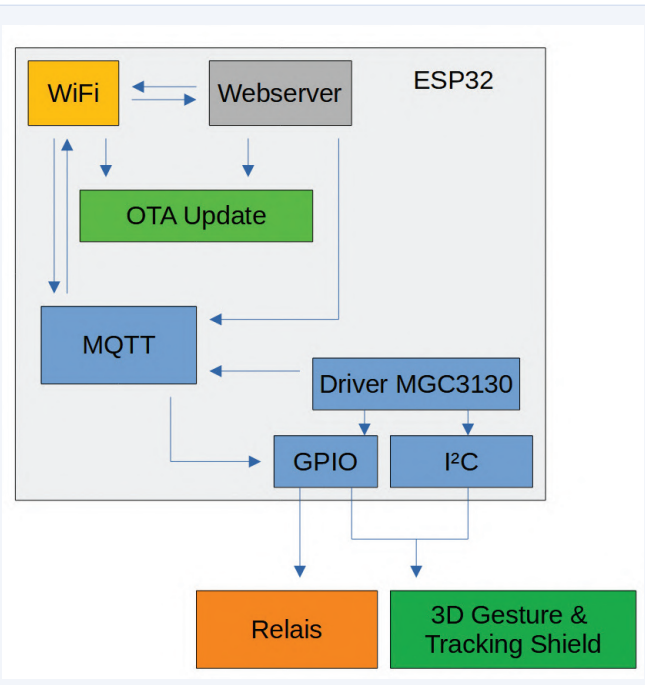


Bild 13. Zusammenspiel der Firmwaremodule.

Der aktuelle Schaltplan

Den ganzen Schaltplan sieht man in **Bild 9**. Etwas, das aktuell noch komplett außer Acht gelassen ist, betrifft den Schutz für das HLK-PM03. Es sind weder eine Sicherung, noch ein PTC noch ein Varistor verbaut. Auch besitzt der 3,3-V-Zweig keinerlei Sicherungen oder eine Filterung. Diese Filterung muss noch eingesetzt werden, da so weder eine EMI noch eine gefilterte Ausgangsspannung bereitstehen. Die fehlende Schutzbeschaltung am Eingang kann im schlimmsten Fall zu einem Defekt des AC/DC-Wandlers und einen damit einhergehenden Brand führen. Die Ausgangsspannung kann die WLAN-Reichweite ohne Filterung deutlich reduzieren oder aber zu ungewollten Abstürzen der Software führen. Die Spannungsversorgung ist alles andere als ideal.

An dem ESP32 ist zwischen GND und Reset ein 10-µF-Kondensator eingefügt. Dieser soll verhindern, dass es innerhalb der Reset-Sequenz zu Timing-Problemen kommt und der Chip sich nicht programmieren lässt. Das Ganze ist auch hilfreich, falls sich der ESP32 an einem USB-3.0-Port spontan nicht mehr programmieren lassen will.

Auf dem ESP32 Pico Kit ist - wie schon erwähnt - ein LDO verbaut, der aus 5 V dann 3,3 V für den ESP32 erzeugt. Bei diesem LDO handelt es sich um einen AM1117-3.3. Da hier die 3,3 V aus dem AC/DC-Wandler bereitgestellt werden, wird dieser LDO rückwärts gespeist. Sollte dieser Betriebsfall vorkommen, ist ein Blick in das Datenblatt des LDO angebracht, ob diese Betriebsart ohne Beschädigung des LDO möglich ist. Im Zweifelsfall sollte eine Schottky-Diode (D1 im Schaltplan) zwischen 3,3 V und 5 V gesetzt werden, um den LDO vor Beschädigungen zu schützen.

Damit das Relais umgeschaltet werden kann, werden mindestens 67 mA bei 3,3 V benötigt. Dies ist deutlich mehr, als die Pins des ESP32 bereitstellen. Daher muss ein Treiber eingesetzt werden, in diesem Fall zwei Transistoren (T1 und T2) vom Typ BC337. Zusätzlich dürfen beim Einsatz eines Relais die Freilaufdioden D2 und D3 nicht vergessen werden.

Durch die zweireihige Stiftleiste K3 wird das *3D Gesture & Tracking Shield* mit dem ESP32 verbunden. Für das Shield selbst werden nur 3,3 V, I²C und zwei I/O-Pins benötigt. Das Shield bringt schon die nötigen Pull-Up-Widerstände für den I²C-Bus mit. Um die Platine eventuell in einem zweiten Projekt einsetzen zu können, sind zusätzlich noch SPI sowie die nötigen I/O-Pins zur Ansteuerung eines 3.5“-TFT für den Raspberry Pi vorgesehen.

Die I/O-Matrix im ESP32 macht das Anbinden von Peripherie sehr flexibel. Für das *3D Gesture & Tracking Shield* werden Pin 21 und Pin 22 für den I²C Bus verwendet sowie Pin 25 als Interrupt und Pin 26 für den Reset des MGC3130. Wenn die Option für das TFT nicht benötigt wird, würde der Schalter mit einem deutlich kleineren ESP32 oder ESP32-C3 Modul auskommen können. Dies ergibt einen sehr übersichtlichen Aufbau.

Firmware

Nachdem die Pins des ESP32 festgelegt sind, gilt es, die Firmware für den Einsatz zusammenzustellen. Es kommt wieder eine Basisfirmware zum Einsatz, die schon OTA-Updates und einen Webserver bereitstellt. Die Bausteine der Basisfirmware sind in **Bild 10** zu sehen. Mit diesem Grundkonstrukt soll verhindert werden, dass für jedes neue Projekt Basisfunktionen neu entwickelt werden müssen. Die Basisfirmware beinhaltet zwei minimale Webseiten. Eine, um ein neues SPIFFS-Image auf den ESP32 zu schreiben (**Bild 11**) und eine, um ein OTA-Update der Firmware selbst durchzuführen (**Bild 12**).

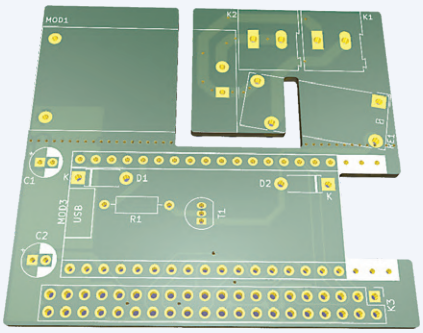


Bild 14. Fertige Platine.

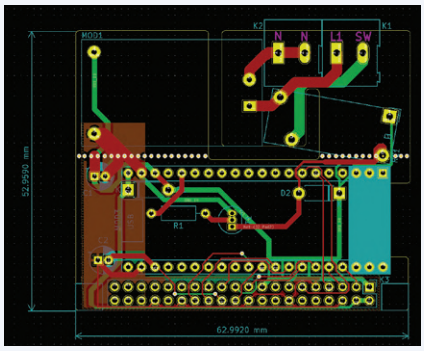


Bild 15. Fertiges Layout.

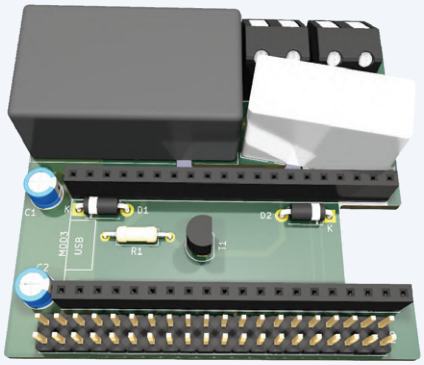


Bild 16. 3D-Ansicht der Platine.

Diese Firmware ist die Ausgangsbasis für den Schalter. Hinzu kommt noch einmal eine Komponente, um die Daten des MGC3130 auswerten zu können und eine minimale Webseite zur Konfiguration einer MQTT-Anbindung sowie zum Ein- oder Ausschalten per Webbrowser. Das Zusammenspiel der Module kann **Bild 13** entnommen werden. Das Steuern des Lichtschalters kann so automatisiert oder manuell erfolgen.

Platinen-3D-Puzzle

Da Firmware und Schaltplan fertig sind, muss nun nur noch die Elektronik in das gegebenen Bauvolumen. Bei den *PCB Design Tools* hat jede(r) seine Vorliebe und sollte das Tool verwenden, mit dem er/sie am besten umgehen kann. In meinem Fall ist das Tool der Wahl KiCad, da so Schaltpläne und Layouts entstehen, die ohne große finanzielle Hürden von jedem weiterbearbeitet werden können. Zu KiCad hat Elektor übrigens das Buch *KiCad wie ein Profi* herausgebracht [7].

Die fertige Platine kann in **Bild 14** betrachtet werden und ist sicherlich keine Schönheit. Die Anordnung der Bauteile ist hier dem Platz und der Funktion untergeordnet. Um die Isolation zwischen den Komponenten mit Netzspannung und dem isolierten Teil der 3,3-V-Versorgung zu trennen, sind Stege in der Platine vorgesehen; es werden nur dort Kupferflächen verwendet, wo es nötig ist. Das Resultat des Routens sieht man in **Bild 15**. Bei Stegen oder Konturen sollten keine 90°-Winkel verwendet werden, denn die Stege werden durch Fräsen erzeugt, was 90°-Winkel unmöglich macht. Wenn die Platine gefertigt werden soll, empfiehlt es sich vorher, auch die Fertigungsspezifikationen des gewünschten Leiterplattenherstellers anzusehen. Dort ist der Kurvenradius und die Stärke des Fräsers vermerkt, der ohne Aufpreis eingesetzt wird.

Für fast jedes Bauteil ist inzwischen ein 3D-Modell verfügbar. Und je nach Anordnung, Lage und Platzierung kann so ermittelt werden, wie der Platzbedarf der Platine aussehen wird. Damit lässt sich auch die Frage beantworten, ob Bauteile auf weitere Platinen ausgelagert werden müssen, die zum Beispiel im Winkel von 90 Grad zur Hauptplatine eingefügt werden. Eine 3D-Ansicht der Platine ist in **Bild 16** zu sehen.

Wie wird es passen?

Das Schöne ist, das wir aus KiCad heraus eine STEP-Datei exportieren können, um diese in einem 3D-Konstruktionsprogramm weiterzuverwenden. Hierfür werden das Modell unserer Platine und ein Gehäuse (3D-Druck in **Bild 17**) in FreeCAD geladen und passend platziert. Wie in **Bild 18** zu erkennen ist, passt die Platine

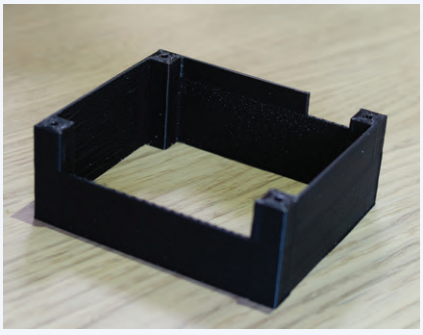


Bild 17. Gedrucktes 3D-Gehäuse.

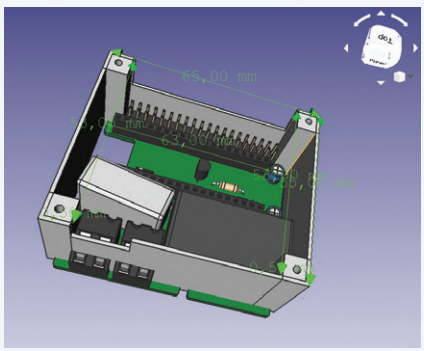


Bild 18. Die Platine passt nicht in das Gehäuse.

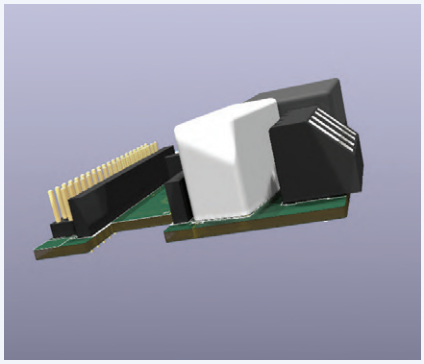


Bild 19. Bauteile stehen über!

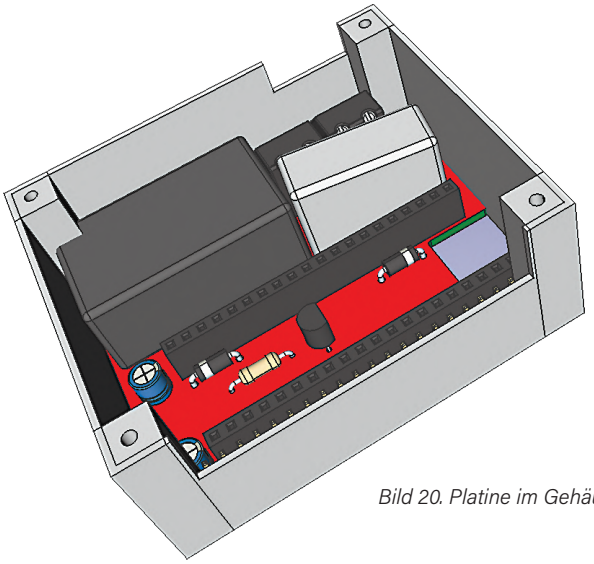


Bild 20. Platine im Gehäuse.

anscheinend nicht. Sind die 3D-Modelle korrekt importiert? Hat sich etwas an der Skalierung geändert? Oder liegt schlicht ein Messfehler vor? Die Antwort ist recht einfach, es war ein schlichter Messfehler bei den Platinenabmessungen.
Nach den Anpassungen der Abmessungen wird es mehr als nur eng auf der Platine. **Bild 19** zeigt, dass nicht mehr alle Bauteile auf der Platine Platz haben. Näher zusammenrücken ist keine Option, da sonst die Abstände zwischen Netzspannung führenden Bauteilen und dem isoliertem Teil der Platine noch kleiner werden. Wird nun alles in FreeCAD zusammengesetzt, ist **Bild 20** das Resultat. An dieser Stelle heißt es dann: Zurück zur Teileauswahl und noch einmal mit dem 3D-Puzzle beginnen!

Was nun?

Grundsätzlich würde der berührungslose Schalter funktionieren. Es gilt, das 3D-Puzzle zu Lösen und alle nötigen Bauteile in den vorhandenen Bauraum zu bekommen. Dazu kommt, dass aus dem eckigen Gehäuse ein rundes werden muss. Bild 4 zeigt den in Deutschland verwendeten Einsatz für Unterputzinstallationen. Für die Spannungsversorgung gibt es kompaktere Netzteile und auch der ESP32 sollte als kompaktes Modul und nicht als ESP32 Pico Kit verwendet werden.
Während dieser Artikel geschrieben wird, machen sich hoffentlich schon ein paar ESP32-C3 auf den Weg Richtung Labortisch. Die Module fallen kompakter aus, sind preiswert, haben WiFi und Bluetooth sowie einen geringeren Stromverbrauch. Letzteres könnte wieder zu einem kleineren Netzteil führen und Platz sparen. An dieser Stelle wird das Projekt erst einmal geparkt. Über Anregungen und Vorschläge für kompaktere Bauteile oder ein optimaleres Routing würden wir uns freuen. Auch Tipps und Tricks, um KiCad und FreeCAD optimal miteinander verzahnen zu können, sind immer willkommen. Wer einen Blick in Quelltext, Platine, Schaltplan oder Gehäusedaten werfen möchte, kann die Projektdaten auf der Elektor-GitHub-Seite [8] herunterladen.

200478-02

Ein Beitrag von

Text und Bilder: **Mathias Claußen**
Redaktion: **Jens Nickel**
Layout: **Giel Dols**

Sie haben 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 redaktion@elektor.de.

PASSENDE PRODUKTE

- **ESP32-PICO-Kit V4 (SKU 18423)**
  www.elektor.de/18423
- **PeakTech 3445 True RMS Digitales Multimeter mit Bluetooth (6000 Counts) (SKU 18774)**
  www.elektor.de/18774
- **Weller WT 1013 Digitale Lötstation (95 W) (SKU 19338)**
  www.elektor.de/19338

WEBLINKS

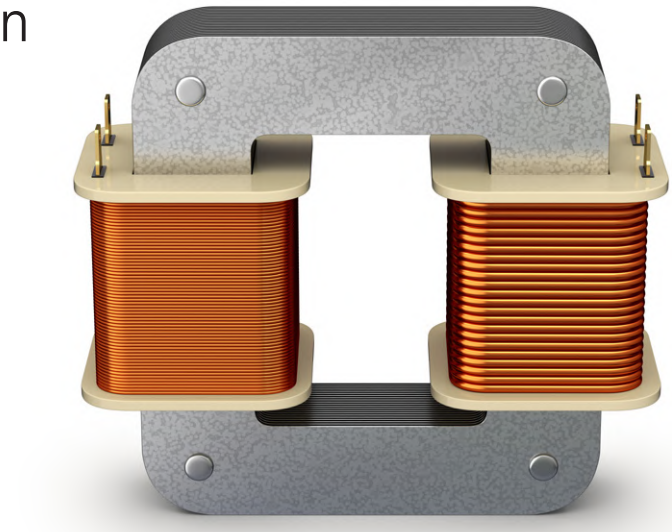
[1] Elektor Store: www.elektor.de
[2] Open Smart Switch: www.hackster.io/133854/open-smart-switch-44fa54#toc-future-developments-8
[3] YouTube-Video zum Tracking- und Gestensensor: www.youtube.com/watch?v=WVSVhEeMi_4
[4] MGC3130 Produktseite: www.microchip.com/wwwproducts/en/MGC3130
[5] MGC3130 Datenblatt: https://ww1.microchip.com/downloads/en/DeviceDoc/40001718E.pdf
[6] Seeed Studio MGC3130 Bibliothek: https://github.com/Seeed-Studio/Seeed_Linux_mgc3x30
[7] Peter Dalmaris, Kicad wie ein Profi, Elektor Verlag: www.elektor.de/kicad-wie-ein-profi-pdf
[8] GitHub Repository: https://github.com/ElektorLabs/200478-Touchless-Lightswitch
[9] Hohlwand-Geräteverbindungsdose von Würth:
https://eshop.wuerth.de/Produktkategorien/-/14015502030501.cyid/1401.cgid/de/DE/EUR/

Aller Anfang …

muss nicht schwer sein: Anpassen und Transformieren

Von **Eric Bogers** (Elektor)

Mit dieser Folge schließen wir das Kapitel „Spulen“ ab. Wir werfen einen letzten Blick auf Filter in Frequenzweichen und besprechen die wichtigsten Aspekte von Transformatoren.



Impedanzanpassung

Wir kehren kurz zum Bandpassfilter aus der letzten Folge [1] zurück (siehe **Bild 1**). Sie werden feststellen, dass wir keinen Lautsprecher als Last gezeichnet haben, sondern einen Widerstand, denn nur an einem Abschlusswiderstand mit einem konstanten Wert verhält sich das Filter „nach Vorschrift“ - und der Widerstand eines Lautsprechers ist alles andere als konstant. Zwei Faktoren stören: Erstens erreicht die Impedanz ihren Höchstwert bei der Resonanzfrequenz, und zweitens steigt die Impedanz mit zunehmender Frequenz (als Folge der Induktivität der Schwingspule).

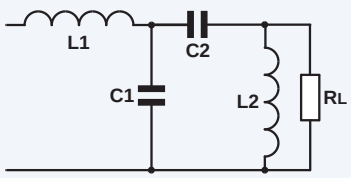


Bild 1. Bandpassfilter.

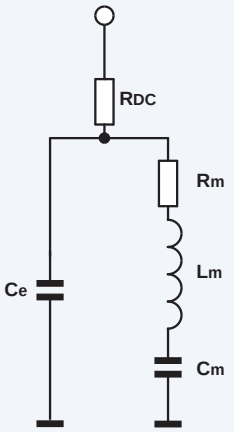


Bild 2. Impedanz-Kompensation.

Wenn solche Störungen mindestens zwei Oktaven von der Trennfrequenz des Filters entfernt auftreten (was einer Verdoppelung oder Halbierung der Frequenz entspricht), müssen Sie sich keine großen Sorgen machen. Wenn dies jedoch nicht der Fall ist, sind die ursprünglichen Annahmen, die bei der Berechnung des Filters gemacht wurden, nicht gültig und das Filter funktioniert nicht wie vorgesehen. Glücklicherweise ist es möglich, dies zu kompensieren. **Bild 2** zeigt ein praktikables Kompensationsnetz.

Dieses Netzwerk besteht aus zwei Teilen: links der Kondensator C_e, der die Induktivität der Schwingspule kompensieren muss, und rechts ein Serienschwingkreis, der die Resonanzspitze kompensiert. Wenn ein Störfaktor weit genug von der Übergangsfrequenz entfernt ist, ist eine Kompensation nicht notwendig und der entsprechende Teil des Netzwerks kann weggelassen werden. Die Werte der Bauteile werden wie folgt bestimmt:

R_{DC} = Gleichstromwiderstand des Lautsprechers

$$C_e = \frac{1000 \cdot L_e}{R_{DC}^2} \qquad [\mu\text{F, mH}]$$

$$R_m = \frac{R_{DC} \cdot Q_e}{Q_m} - R_{loss}$$

$$L_m = \frac{159 \cdot R_{DC} \cdot Q_e}{f_s} \qquad [\text{mH}]$$

$$C_m = \frac{159000}{f_s \cdot R_{DC} \cdot Q_e} \qquad [\mu\text{F}]$$

Die Werte der Parameter R_{DC}, Q_m, Q_e und f_s können mit speziellen Messprogrammen (wie Kirchner ATB) ermittelt werden. R_{DC} ist der Gleichstromwiderstand des Lautsprechers und f_s die Mittenfrequenz der Resonanzspitze. R_{loss} ist der Ersatzwiderstand der Spule und L_e die Induktivität.

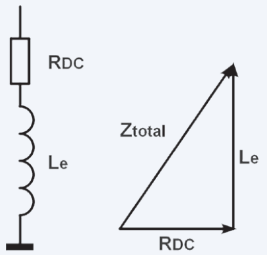


Bild 3. Induktivität und Gleichstromwiderstand.

Wie aber bestimmt man L_e? Bei höheren Frequenzen kann der Lautsprecher als Reihenschaltung aus einer Induktivität und einem Gleichstromwiderstand betrachtet werden, wie in **Bild 3** skizziert. Bei einer ausreichend hohen Frequenz kann man den Wert der Gesamtimpedanz Z_{total} aus dem Impedanzdiagramm ablesen. Für die Induktivität L_e gilt Folgendes:

$$L_e = \frac{\sqrt{Z_{tot}^2 - R_{DC}^2}}{2 \cdot \pi \cdot f}$$

Im Beispiel von Bild 3 beträgt bei 10 kHz die Gesamtimpedanz 42 Ω, während R_{DC} gleich 7 Ω ist. Wir berechnen, dass eine Induktivität von 0,66 mH mit einem Kondensator von etwa 10 µF kompensiert werden muss.

Transformatoren und Übertrager

Transformatoren können verwendet werden, um Wechselspannungen „nach oben“ oder „nach unten“ zu transformieren. Wenn Netzspannungen abwärts transformiert werden, spricht man normalerweise von einem Netztransformator, wenn Signalspannungen transformiert werden müssen, spricht man von Signaltransformatoren oder Übertragern. Die Funktionsweise dieser beiden Transformatorarten ist jedoch genau dieselbe.

Signaltransformatoren werden in der Regel nicht zum Hoch- oder Heruntertransformieren von Signalspannungen verwendet, sondern eher, um (Teil-) Stromkreise galvanisch voneinander zu trennen - zum Beispiel, um zu verhindern, dass eine Erdschleife entsteht. In solchen Fällen spricht man von einem „symmetrischen Übertrager“.

Bild 4 zeigt einige Vertreter der Gattung „Transformator“. Rechts ist ein Teil eines Ringkerntransformators zu sehen, bei dem die Wicklungen um einen ringförmigen Eisenkern gewickelt sind. Diese Art von Transformator zeichnet sich durch ein sehr geringes Streufeld aus, weshalb sie gerne in Audio-Leistungsverstärkern eingesetzt werden.

Oben links sehen Sie einen Platinentransformator, der direkt in eine Platine gelötet werden kann. Zur Isolierung und zum Schutz vor mechanischer Beschädigung ist dieser Transformator in einem Kunststoffgehäuse vergossen.

Unten links schließlich sehen wir zwei Audio-Signaltransformatoren, die ebenfalls für die Montage auf einer Platine vorgesehen sind. Audio-Übertrager sind oft mit einer Abschirmung aus Blech versehen, um zu verhindern, dass sie störende magnetische Streufelder auffangen. Dies ist jedoch bei den Beispielen in Bild 4 nicht der Fall.

In **Bild 5** sehen Sie das Schaltsymbol für einen Transformator. Wenn Sie dabei an zwei Spulen denken, dann haben Sie recht: Ein Transformator besteht aus zwei (oder mehr) Spulen, die auf einen gemeinsamen Kern gewickelt sind. Im Prinzip könnte dieser Kern ein großes Eisenstück sein, aber das würde zu unannehmbar hohen Wechselstromverlusten führen. Deshalb besteht bei 50-Hz-Transformatoren (also Netztransformatoren) der Kern aus dünnen Blechen aus Transformatorstahl, die gegeneinander isoliert sind.

Für höhere Frequenzen als 50 Hz wird Eisenpulver mit einem isolierenden Füllstoff vermischt und in die gewünschte Form gepresst. Auf diese Weise erhält man Ferritkerne. Bei sehr hohen Frequenzen ist ein Kern nicht mehr erforderlich und man kann zwei Luftspulen verwenden.

Ein Transformator funktioniert folgendermaßen: An die Primärwicklung wird eine Wechselspannung gelegt, so dass ein Wechselstrom durch die Wicklung fließt, der wiederum ein veränderliches Magnetfeld erzeugt. Dieses veränderliche Magnetfeld induziert dann eine Wechselspannung in der Sekundärwicklung. Zwischen der Primär- und der Sekundärwicklung gibt es keinen elektrisch leitenden Pfad: Sie sind galvanisch getrennt.

Die Bezeichnungen „primär“ und „sekundär“ könnten den Eindruck erwecken, dass ein Transformator nur in eine Richtung funktioniert. Es ist aber durchaus möglich, eine Spannung an die Sekundärwicklung zu legen und anschließend die transformierte Spannung an der Primärwicklung zu nutzen. Dabei muss man allerdings darauf achten, dass der Eisenkern nicht durch eine zu hohe Spannung in die (magnetische) Sättigung getrieben wird. Wird zum Beispiel ein 12-V-Transformator (also ein Transformator, der normalerweise die 230-V-Netzspannung auf 12 V heruntertransformiert) verkehrt herum angeschlossen, so ist eine durchgebrannte Sicherung oder eine durchgebrannte Trafowicklung die garantierte Folge. Infolge der magnetischen Sättigung des Kerns würde ein viel zu hoher Strom fließen. Es ist jedoch durchaus möglich, an die Sekundärwicklung eine Wechselspannung von 12 V anzulegen, so dass an der Primärwicklung 230 V anliegen. Nach diesem Prinzip arbeiten die so genannten Inverter.

Das Verhältnis zwischen der Anzahl der Primärwindungen und der Anzahl der Sekundärwindungen wird als Transformator-Windungsverhältnis bezeichnet. Es gilt folgendes:

$$T = \frac{W_1}{W_2} = \frac{U_1}{U_2} = \frac{I_2}{I_1}$$

Das Verhältnis zwischen Eingangs- und Ausgangsspannungen ist gleich dem Verhältnis zwischen der Anzahl der Primär- und Sekundärwindungen. Das Verhältnis zwischen Eingangs- und Ausgangsströmen ist genau umgekehrt. Das ist natürlich offensichtlich, denn es muss Folgendes gelten:

$P1 = P2$

Die Leistung in der Sekundärwicklung ist aber nicht genau gleich groß wie die in der Primärwicklung. Ein Transformator ist nämlich kein ideales Bauteil, so dass es immer einen gewissen Verlust gibt. Diese Verluste sind jedoch in der Regel so gering, dass wir sie bei den meisten Berechnungen vernachlässigen können.
Bei Netztransformatoren wird in der Regel kein Wort über das Windungsverhältnis des Transformators verloren: Hier wird nur die Spannung angegeben, die an der Sekundärwicklung auftritt, wenn die Netzspannung von 230 V an die Primärwicklung angelegt wird. Viele Netztransformatoren haben mehrere Sekundärwicklungen, manchmal mit unterschiedlichen Ausgangsspannungen. Es ist möglich, Sekundärwicklungen in Reihe zu schalten, aber man muss dabei die Phase und die maximale Last (maximaler Strom) berücksichtigen. Wenn man mehrere Wicklungen parallel schalten will, müssen diese die gleiche Spannung liefern und eine (fast) identische Leistung haben. Außerdem müssen die Phasen der parallel geschalteten Wicklungen gleich sein.
Darüber hinaus haben Netztransformatoren oft mehrere Primärwicklungen, damit die Geräte, in denen sie eingebaut sind, an die unterschiedlichen Netzspannungen in verschiedenen Ländern angepasst werden können.
Es ist jedoch niemals möglich, einen 230-V-Netztransformator an eine Spannung von 400 V anzuschließen: Der Transformator gerät in magnetische Sättigung, was zu einer durchgebrannten Sicherung oder Primärwicklung führen würde. Das ist der Grund, warum falsch installierte Drehstromsteckdosen so gefürchtet sind. Das Gegenteil ist jedoch völlig in Ordnung. In elektronischen Geräten kann ein 400-V-Transformator eingebaut werden, der problemlos mit 230 V betrieben werden kann, aber natürlich dann eine entsprechend niedrigere Ausgangsspannung liefert.
Damit ist das Thema „Spulen“ abgeschlossen. Das nächste Mal werden wir uns (endlich!) mit Halbleitern beschäftigen.

210626-01

Die Artikelreihe „Aller Anfang …“ gründet auf dem Buch „Basiskurs Elektronik“ von Michael Ebner, erschienen im Elektor Verlag.

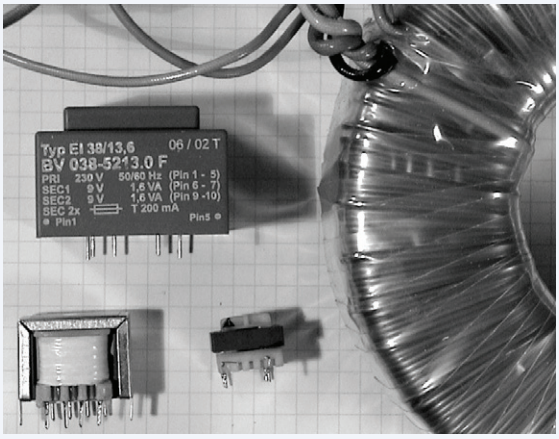


Bild 4. Verschiedene Netztransformatoren und Signal-Übertrager.

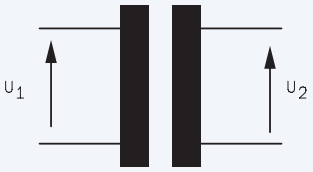


Bild 5. Schaltzeichen eines Transformators.

Haben Sie Fragen oder Kommentare?

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

Ein Beitrag von

Idee und Illustrationen: **Michael Ebner**
Text und Redaktion: **Eric Bogers**
Übersetzung: **Rolf Gerstendorf**
Layout: **Giel Dols**

PASSENDE PRODUKTE

- **Ebner, Michael, „Basiskurs Elektronik“**
  www.elektor.de/basiskurs-elektronik-pdf
- **B. Kainka, Elektronik-Grundlagen und Einsteiger-Projekte (E-Book), Elektor 2020**
  www.elektor.de/elektronik-grundlagen-und-einsteiger-projekte-pdf

WEBLINK

[1] „Aller Anfang …“, Elektor Magazin Januar/Februar 2022: https://www.elektormagazine.de/210564-03

Was gibt‘s Neues in der Embedded-Entwicklung?

Rust und die Aktualisierung von IoT-Implementierungen

Embedded Development

Von **Stuart Cording** (Elektor)

Obwohl die Technologie für eingebettete Systeme scheinbar schnell voranschreitet, ist die Branche selbst im Vergleich dazu eher langsam. Deshalb war es fast ein kleiner Schock, als ein bekannter Hersteller von Einplatinencomputern, die Raspberry-Pi-Foundation, den RP2040-Mikrocontroller mit zwei Cortex-M0+-Kernen und ohne Onboard-Flash herausbrachte. Ein Dual-Core-Prozessor in dieser Geräteklasse ist ein absolutes Novum, aber der RP2400 ist auch schon das Spannendste auf dem Markt, dessen Fortschritt ansonsten messbar, sinnvoll und überlegt ist. Es deuten sich aber Fortschritte an, die die Entwicklung eingebetteter Systeme im kommenden Jahrzehnt verändern könnten.

Die Entwicklung eingebetteter Software ohne C ist kaum vorstellbar. Als Assembler für die Entwicklung vollständiger Anwendungen zu umständlich wurde, verdrängte C diese Sprache (außer wenn ein hochoptimierter, von Hand geschriebener Assembler die einzige Option war). Die von Dennis Ritchie [1] in den Bell Labs entwickelte Sprache ist ausreichend flexibel für komplexe Anwendungen und bietet gleichzeitig einen einfachen Zugriff auf Register. Dies ist entscheidend für einen kompakten Mikrocontroller-Code, der Registerzugriffe in Interrupt-Routinen verarbeitet. Auch Aufgaben wie die Bitmanipulation von Registern lassen sich damit leicht umsetzen. Und im Gegensatz zu Assembler-Code ist C-Code leichter zu lesen. Außerdem rangiert C in Umfragen und Marktanalysen stets unter den drei am häufigsten verwendeten Programmiersprachen (**Bild 1**) [2][3].

C ist alt

Allerdings ist C, das im Jahr 1972 entwickelt wurde, inzwischen 50 Jahre alt. Die Programmiersprache hat eine Reihe von bekannten Einschränkungen, von denen viele mit der Verwendung von Pointern zusammenhängen. Pointer erleichtern Entwicklern von eingebetteten Systemen zwar den Zugriff auf Register, sie können aber auch zu unerwünschten Speicherzugriffen außerhalb des zulässigen Bereichs führen. Außerdem führen C-Compiler im Vergleich zu moderneren Programmiersprachen vergleichsweise wenige Codeprüfungen durch. So werden ungenutzte Variablen einfach ignoriert, was ein Zeichen für einen Fehler im Code sein kann.

Um unsicheren C-Code in eingebetteten Systeme zu vermeiden, verwenden Entwickler Codierungsstandards wie MISRA C [4]. Dieser Standard entstand, als C in der Automobilindustrie als Programmiersprache für eingebettete Systeme an Bedeutung gewann. C++ löste

einige der Probleme mit Pointern durch *references* [5], die nicht geändert werden können, um auf ein anderes Objekt zu verweisen, nicht NULL sein können und bei der Erstellung initialisiert werden müssen. Trotzdem hat AUTOSAR, eine Partnerschaft von Entwicklern von Automobilsystemen, in einem mehrere hundert Seiten umfassenden Dokument Richtlinien für die Verwendung von C++ für sicherheitsrelevante Anwendungen herausgegeben [6]. Obwohl also die Beherrschung dieser etablierten Sprachen für Entwickler von eingebetteten Systemen unabdingbar ist, weist jede dieser Sprachen so viel Schwächen auf, dass Richtlinien erforderlich sind, um häufige Programmierfehler zu vermeiden.

Einführung von Rust

Rust hat sich als potenzieller Herausforderer herauskristallisiert und wirbt mit der Eignung zur Entwicklung sicherer Systeme. Rust begann 2006 als privates Projekt von Graydon Hoare und wurde 2010 zu einem von seinem Arbeitgeber Mozilla Research geförderten Projekt. Als 2021 Umstrukturierungen des Unternehmens Auswirkungen auf das Rust-Entwicklungsteam hatten, wurde die Rust Foundation [7] gegründet.
Das Besondere an Rust ist, dass viele Probleme schon bei der Kompilierung erkannt und markiert werden, oft auch solche, die C/C++-Compiler ignorieren würden. Es wurde auch ein *Ownership*-System für Variablendeklarationen implementiert, das einen *Borrow-Checker* verwendet, um eine missbräuchliche Anwendung von Variablen bereits während der Kompilierung auszuschließen. Außerdem muss der Lese- und Schreibzugriff auf Variablen, die per Referenz übergeben werden, explizit deklariert werden. Die Syntax von Rust ähnelt weitgehend C und C++, wobei geschweifte Klammern um Funktionen und die bekannten Steuer-Keywords verwendet werden.
Da der Schwerpunkt auf sicherem Code liegt, wurde Rust besonders für die Bare-Metal-Community zur Entwicklung eingebetteter Software interessant. Aufgrund der Natur der eingebetteten Programmierung kann die vom Compiler eingesetzte statische Prüfung jedoch bei der Implementierung einiger Codetypen zu Problemen führen. Einige Codeabschnitte können als *unsafe* markiert werden, um zum Beispiel die De-Referenzierung von Pointern zu ermöglichen. Durch die explizite Definition von Codeabschnitten als *unsafe* kann die Umgehung der Rust-Regeln verdeutlicht werden.

Rust auf den Prüfstand gestellt

Um ein Gefühl für Rust zu bekommen, installiert man Rust am besten auf einem Raspberry Pi. Die Installation ist einfach, wenn man den Hinweisen auf der Website *rustup.rs* [8] folgt. Geben Sie auf der Kommandozeile einfach den folgenden Befehl ein und folgen Sie den Anweisungen:

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.
   rs | sh
```

Im Gegensatz zu C/C++, das Binärdateien erzeugt, wird die Ausgabe von Rust als *Crate* bezeichnet. Der Paketmanager Cargo sorgt für eine einfache Befehlszeilenkompilierung. Er enthält die Daten, die für die Erzeugung des *Crate* benötigt werden und ermöglicht es dem Entwickler, die für die Erstellung des *Crate* erforderlichen Pakete zu definieren. Durch den Aufruf von Cargo auf der Kommandozeile wird ein neues Rust-Projekt wie folgt erzeugt:

```
cargo new rust_test_project
```

Wenn Sie das Verzeichnis des neuen Projekts öffnen, sehen Sie eine Datei namens *Cargo.toml*. Diese Datei muss unter Umständen geändert werden. Für eine einfache Raspberry-Pi-Anwendung, die eine an einen

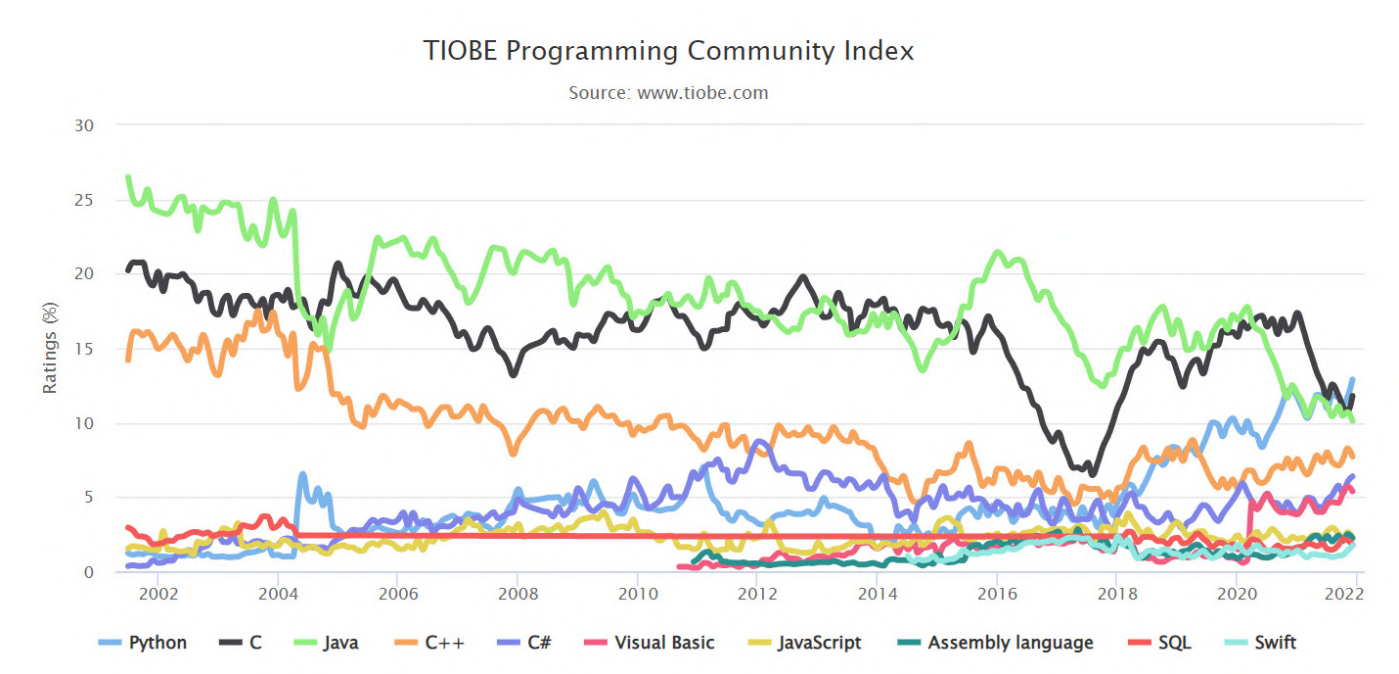


Bild 1. C ist als Programmiersprache nach wie vor sehr beliebt und belegt in Umfragen und Marktanalysen regelmäßig einen der ersten drei Plätze. (Quelle: www.tiobe.com)

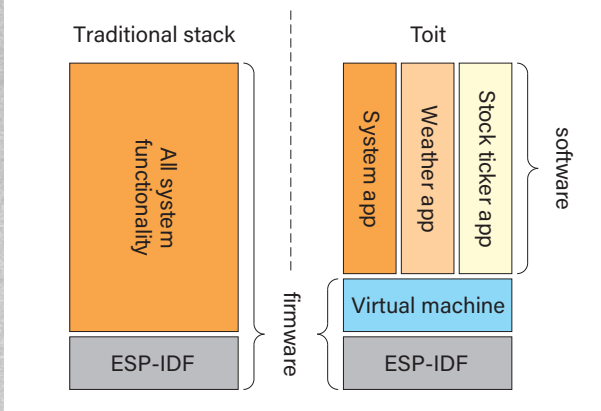


Bild 2. Toit führt IoT-Code als Apps auf einer virtuellen Maschine aus, die auf einem ESP32 läuft. (Quelle: Toit)

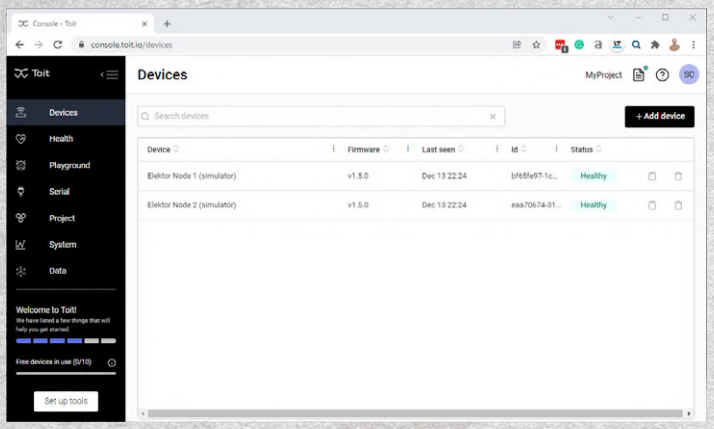


Bild 3. Die Toit-Konsole in einem Browser, hier mit zwei simulierten ESP32-Knoten.

GPIO-Pin angeschlossene LED blinken lässt, muss eine geeignete Crate-Abhängigkeit für den Zugriff auf die GPIO-Pins definiert werden. Crates werden auf der Crate-Plattform der Rust-Community crates.io [9] zur Verfügung gestellt. Ein geeigneter Crate ist *rppal*, der über die Suchfunktion gefunden werden kann. Die Plattform gibt die Versionsnummer an und stellt eine Dokumentation und Beispielcode zur Verfügung. Der Name des Crate und die Versionsnummer werden dann als *dependencies* in *Cargo.toml* hinzugefügt (**Listing 1**).

Listing 1. Hinzufügen der rppal-Abhängigkeit zu Cargo.toml in einem Rust-Projekt.

```
[package]
name = "rust_test_project"
version = "0.1.0"
edition = "2021"
[dependencies]
rppal = "0.13.1"
```

Im *src*-Verzeichnis findet der Entwickler ein einfaches *Hello-World*-Beispielprojekt in der Rust-Quellcodedatei *main.rs*. Wenn man diesen Code durch **Listing 2** ersetzt, kann man eine LED an GPIO 23 (Pin 16 des Raspberry-Pi-Headers) zehnmal blinken lassen. Die Kompilierung erfolgt über die Kommandozeile mit `cargo build`, das Crate wird mit `cargo run` ausgeführt.

Um Rust auf Mikrocontrollern verwenden zu können, ist eine LLVM-Toolchain erforderlich. Wenn diese vorhanden ist, können Sie mit Hilfe eines der verschiedenen Online-Tutorials loslegen. Ein Beispiel für den BBC micro:bit [10] zeigt, dass das Schreiben von Code, hat man erst einmal die Rust-Syntax im Griff, dem von C/C++ sehr ähnlich ist (**Listing 3** [11]). Ein weiteres Beispiel für den STM32 [12] ist ebenfalls enthalten.

Gehört Rust die Zukunft?

Was steht also der Einführung von Rust für Embedded-Anwendungen im Wege? Wenn Sie nach einer robusten Programmiersprache für den

Listing 2. Blinken einer LED in Rust.

Das Blinken einer LED in Rust ähnelt dem in C geschriebenen Code.
Dieses Beispiel basiert auf dem Code, der im rppal-Crate enthalten ist.

```
use std::error::Error;
use std::thread;
use std::time::Duration;
use rppal::gpio::Gpio;
use rppal::system::DeviceInfo;
// Gpio uses BCM pin numbering. BCM GPIO 23 is tied to physical pin 16.
const GPIO_LED: u8 = 23;
fn main() -> Result<(), Box<dyn Error>> {
    let mut n = 1;
    println!("Blinking an LED on a {}.", DeviceInfo::new()?.model());
    let mut pin = Gpio::new()?.get(GPIO_LED)?.into_output();
    while n < 11 {
        // Blink the LED by setting the pin's logic level high for 500 ms.
        pin.set_high();
        thread::sleep(Duration::from_millis(500));
        pin.set_low();
        thread::sleep(Duration::from_millis(500));
        n += 1;
    }
    Ok(())
}
```

Listing 3. Vereinfachter Codeschnipsel, der den Zustand eines Eingangspins des BBC micro:bit mit Rust überprüft.

```
#![no_std]
#![no_main]

extern crate panic_abort;
extern crate cortex_m_rt as rt;
extern crate microbit;

use rt::entry;
use microbit::hal::prelude::*;

#[entry]
fn main() -> ! {
     if let Some(p) = microbit::Peripherals::take() {
        // Split GPIO
        let mut gpio = p.GPIO.split();

        // Configure button GPIO as input
        let button_a = gpio.pin17.into_floating_input();

        // loop variablew
        let mut state_a_low = false;

        loop {
             // Get button state
             let button_a_low = button_a.is_low();

            if button_a_low && !state_a_low {
                 // Output message
             }

            if !button_a_low && state_a_low {
                 // Output message
             }

            // Store button states
             state_a_low = button_a_low;
         }
     }
     panic!("End");
 }
```

Einsatz in sicherheitskritischen Systemen suchen, ist doch Ada [13] eigentlich eine gute Wahl. Doch obwohl Ada bereits 40 Jahre alt ist, hat sie C nicht verdrängen können, obwohl sie speziell für den Einsatz in eingebetteten Echtzeitsystemen entwickelt wurde. Ein weiterer Aspekt ist der jahrelange Erfolg von C, mit unzähligen Entwicklern, Werkzeugen und Code-Bibliotheken.

Der Crate-Paketmanager könnte jedoch dazu beitragen, die Verbreitung von Rust zu beschleunigen. Den Überblick über die in C/C++ geschriebenen Peripherie-Bibliotheken der Halbleiterhersteller zu behalten, kann schwierig und undurchsichtig sein, so dass die explizite Definition der Version der in *Cargo.toml* verwendeten Crates als bedeutende Verbesserung angesehen werden könnte. Es könnte auch das Leben der MCU-Hersteller erleichtern, die ihre riesigen Produktportfolios unterstützen wollen. Ada hat im Jahr 2020 bereits mit der Veröffentlichung des Paketmanagers *Alire* reagiert [14] und unterstützt genau wie Rust den BBC micro:bit. Damit können Sie beide Sprachen auf der gleichen MCU vergleichen [15].

IoT-Geräte auf dem neuesten Stand halten

Da immer mehr eingebettete Geräte mit Netzwerken verbunden sind, besteht die größte Herausforderung darin, sie mit der neuesten Version der Firmware auf dem neuesten Stand zu halten. Traditionell müssen Firmware-Updates über Funk heruntergeladen werden, wobei die Binärdaten über einen On-Chip-Bootloader im geschützten Bereich des Geräts in den Flash-Speicher programmiert werden. Dabei besteht das Risiko, dass die neue Codeversion nicht korrekt installiert wird, so dass das Produkt nicht mehr funktioniert und unbrauchbar wird. Alternativ kann der Code zwar funktionieren, aber ein Softwarefehler im neuen Code könnte verhindern, dass künftige Updates eingespielt werden, so dass die Geräte in der Praxis anfällig bleiben. Auf diese Weise wird das Gerät vielleicht nur aufgrund eines Fehlers in einer einzigen Codezeile unbrauchbar, obwohl der Großteil der Anwendung korrekt funktioniert.

Virtuelle Maschinen sind seit Jahren eine Standardmethode, um Anwendungen oder Betriebssysteme auf Servern bereitzustellen. Die virtuelle Maschine gaukelt einem Betriebssystem vor, es würde auf dedizierter Hardware ausgeführt. Sollte das Betriebssystem schwerwiegend versagen, ist nur die betroffene virtuelle Maschine davon betroffen, nicht aber die übrigen auf dem Server laufenden Systeme. Desktop-Benutzer kennen Virtualisierungssoftware wie VirtualBox, VMware und Parallels, mit der sie Software oder alternative Betriebssysteme testen können, ohne ihren primären Rechner einem Risiko auszusetzen.

Aktualisierung der IoT-Knoten-Firmware: Der Weg zu Toit

Das Team von Toit, einem dänischen Unternehmen, das von ehemaligen Google-Ingenieuren gegründet wurde, fragte sich, warum die Virtualisierung nicht auch auf Mikrocontrollern eingesetzt wird, auf denen IoT-Anwendungen laufen. Schließlich müssen diese regelmäßig aktualisiert werden, um Fehler zu beheben und möglicherweise um Verbesserungen in der Cloud zu unterstützen.

Für die ersten Schritte haben sie sich für den ESP32 von Espressif entschieden, genauer gesagt, den ESP32-WROOM-32E mit einem 32-bit-Dual-Core Xtensa LX6 Mikroprozessor, 520 KB SRAM und 4 MB Flash. Für diese Plattform gibt es bereits das ESP-IDF (Espressif IoT Development Framework), so dass sie für den Einsatz als IoT-Knoten gut vorbereitet ist. Toit hat dann eine virtuelle Maschine (**Bild 2**) entwickelt, auf der Anwendungen sicher ausgeführt werden können. Die Anwendungen sind in Toit geschrieben, einer objektorientierten und sicheren Hochsprache.

Testen von Toit mit simulierten ESP32-Knoten

Die Verwaltung und Bereitstellung von Anwendungen auf einer Ansammlung von IoT-Knoten erfolgt über die Toit-Konsole in Verbindung mit Visual Studio Code von Microsoft. Für diejenigen, die nur daran interessiert sind, die Plattform zu testen, ermöglicht die Toit Console die Verwendung von simulierten ESP32-Geräten. Nach der Installation der Toit-Erweiterung für Visual Studio Code können Anwendungen geschrieben und lokal simuliert oder an Geräte verteilt werden, die mit der Konsole verbunden sind.

Toit-Apps werden von einer YAML-Datei begleitet. In dieser Markup-Language-Datei werden die Namen der Toit-App und der Quellcodedatei deklariert und Trigger definiert, die die App nach dem

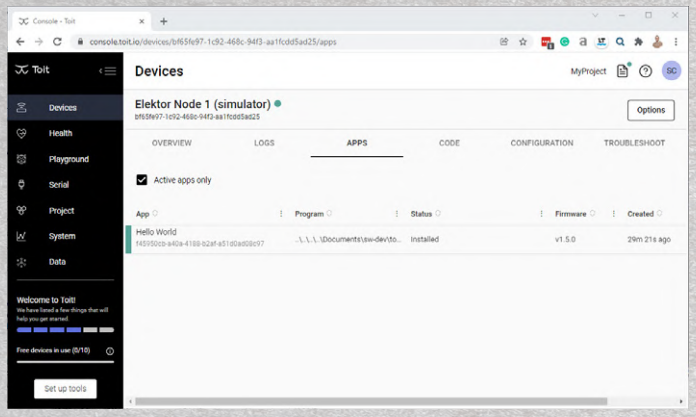


Bild 4. Die Hello-World-App wurde erfolgreich auf einem simulierten Knoten installiert

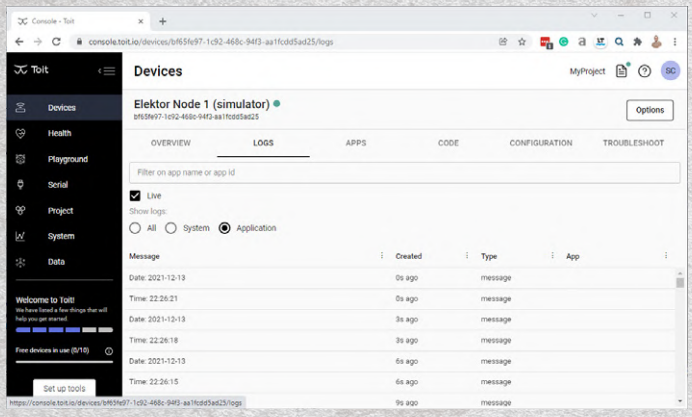


Bild 5. Die LOGS-Ausgabe zeigt die Uhrzeit und das Datum, die von der Toit-App alle drei Sekunden ausgegeben werden, wie in der YAML-Datei definiert.

Booten oder nach bestimmten Zeitintervallen ausführen.
Die Bereitstellung von Apps oder Updates erfordert nicht, dass das ESP32-Zielgerät deaktiviert oder die Stromversorgung unterbrochen werden muss. Stattdessen aktualisiert die virtuelle Maschine die App an Ort und Stelle und triggert sie gemäß den in der YAML-Datei angegebenen Einstellungen. Wenn die App in irgendeiner Weise versagt, zum Beispiel durch einen Speicherüberlauf, werden die übrigen Apps ohne Probleme weiterhin ausgeführt [16]. Solche Fehler können über die Konsole diagnostiziert werden, indem man sich das Protokoll ansieht.
Listing 4 zeigt eine einfache *Hello-World*-App, die Uhrzeit und Datum ausgibt, **Listing 5** die zugehörige YAML-Datei. In **Bild 3** sind zwei simulierte Knoten in der Konsole zu sehen, während **Bild 4** die erfolgreich installierte *Hello-World*-App zeigt. **Bild 5** zeigt die Ausgabe von Datum und Uhrzeit in Drei-Sekunden-Intervallen, wie sie mit dem Trigger `on_interval: "3s"` in der YAML-Datei festgelegt wurden.
Der Projektcode wurde in Visual Studio Code erstellt und mit der Toit-Erweiterung auf dem ausgewählten Knoten bereitgestellt, der mit dem Konsolen-Konto des Users verbunden ist (**Bild 6**).
Auf den ersten Blick mag der Toit-Ansatz ein wenig restriktiv erscheinen. Die Virtualisierungsumgebung erlaubt nur die Steuerung der GPIOs, der seriellen Schnittstelle (UART), SPI oder I²C. Da jedoch viele IoT-Knoten Sensordaten sammeln und diese an die Cloud melden, dürfte dies für die meisten Anwendungen ausreichend Flexibilität bieten. Toit betreibt auch einen eigenen Paketmanager (Registry) [17], der Treiber für eine Reihe von Sensoren, Eingabegeräten, LCDs und anderen Dienstprogrammen zur Verfügung stellt. Die Umgebung unterstützt auch den Stromsparmodus des ESP32, der es angeblich ermöglicht, das Gerät jahrelang mit zwei AA-Batterien zu betreiben [18]. Wenn eine App-Aktualisierung erfolgt, während sich der IoT-Knoten im Tiefschlafmodus befindet, stellt die Konsole sie dann bereit, wenn der Knoten das nächste Mal aufwacht.

Rust und Toit - die Zukunft von Embedded?

Zwei Dinge stechen bei Rust und Toit hervor. Beide sind einfach in Betrieb zu nehmen und beide verwenden Paketmanager, um die Low-Level-Treiber zu verwalten. Dadurch können sich die Entwickler auf ihre eigentliche Aufgabe konzentrieren, nämlich das Erstellen von Anwendungen. Da die Anwendungen immer komplexer werden, ist eine solche Wiederverwendung von Code unerlässlich, um die Entwicklungszeit zu verkürzen und Produkte schneller auf den Markt zu bringen.
Rust steht vor einer gewaltigen Aufgabe: Da C und C++ in der Welt der Embedded-Entwicklung so fest verankert sind, dürfte es schwierig sein, eine Lücke zu finden, in der Rust einen so gewichtigen Vorteil bietet, dass er Entwickler anlockt. Und da sich Ada bereits als die

Listing 4. Eine einfache Toit-Anwendung, die formatierte Zeit- und Datumsstrings in der Log-Konsole ausgibt.

```
main:
  time := Time.now.local
  print "Time: $(%02d time.h):$(%02d time.m):$(%02d time.s)"
  print "Date: $(%04d time.year)-$(%02d time.month)-$(%02d time.day)"
```

Listing 5. Die zugehörige YAML-Datei, die der Toit-Konsole mitteilt, wie die Anwendung bereitgestellt werden soll.

```
name: Hello World
entrypoint: hello_world.toit
triggers:
  on_boot: true
  on_install: true
  on_interval: "3s"
```

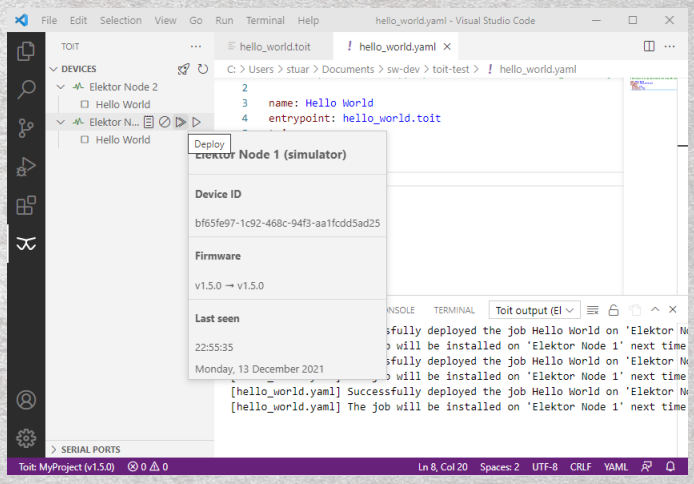


Bild 6. Mit der Toit-Erweiterung in Visual Studio Code können Änderungen an Anwendungen auf IoT-Knoten bereitgestellt werden, ohne dass die Geräte vor Ort aus dem Netz entnommen werden müssen.

Sprache für sicherheitskritische Systeme etabliert hat, verschwinden die Vorteile von Rust.

Toit hingegen löst ein großes Problem: Es hält entfernte IoT-Knoten auf dem neuesten Stand, stellt neue Funktionen zur Verfügung und das alles, ohne die Geräte aus dem System zu entfernen. Einige Entwickler dürften über nur 4 MB Flash und die Tatsache, dass derzeit nur der ESP32 unterstützt wird, die Stirn runzeln, sollte jedoch eine Nachfrage nach anderen MCUs entstehen, dürfte es keinen technischen Grund zu geben, warum andere Plattformen in Zukunft nicht unterstützt werden sollten.

210652-02

Haben Sie Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Dann wenden Sie sich bitte an den Autor unter stuart.cording@elektor.com oder an die Elektor-Redaktion unter redaktion@elektor.de.

Ein Beitrag von

Text und Illustrationen: **Stuart Cording**
Redaktion: **Jens Nickel, CJ Abate**
Übersetzung: **Rolf Gerstendorf**
Layout: **Harmen Heida**

PASSENDE PRODUKTE

- Buch: D. Ibrahim, BBC micro:bit (Elektor, 2016, SKU 17972) www.elektor.de/bbc-micro-bit-book
- Joy-IT BBC micro:bit Go Set (SKU 18930) www.elektor.de/18930

WEBLINKS

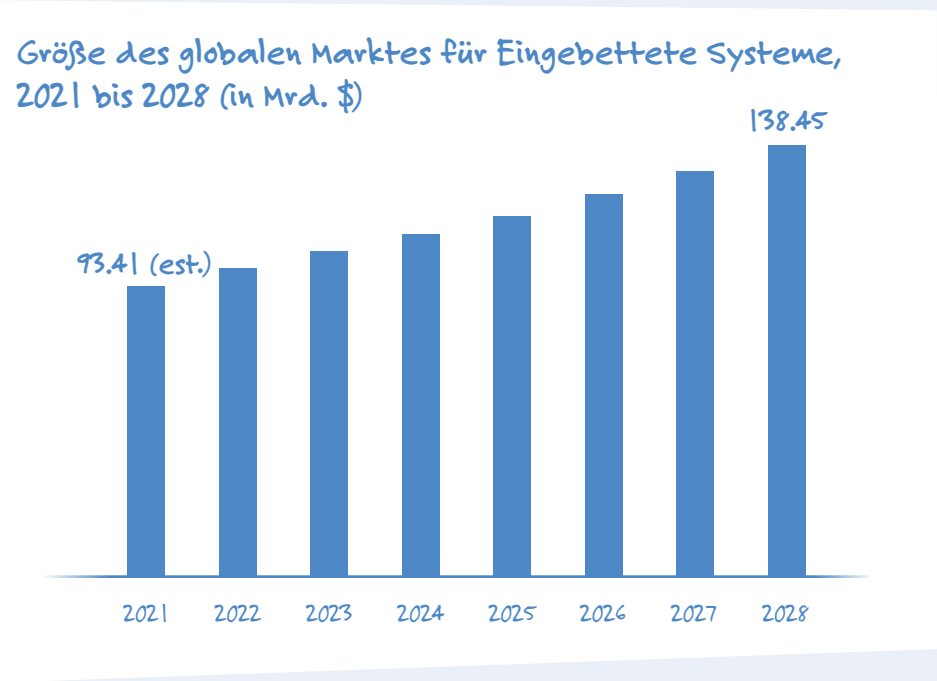
[1] Dennis Ritchie, Computer History Museum: https://bit.ly/3oOXG1A
[2] P. Jansen, „TIOBE Index for Dezember 2021", TIOBE Software BV, Dezember 2021: https://bit.ly/3EXx8Ri
[3] S. Cass, „Top Programming Languages 2021", IEEE Spectrum, 2021: https://bit.ly/3oPJq8z
[4] Webseite MISRA: https://bit.ly/3s1Mv7D
[5] „C++ References", Tutorials Point: https://bit.ly/31Y6k53
[6] „Guidelines for the use of the C++14 language in critical and safety-related systems" AUTOSAR, Oktober 2018: https://bit.ly/3dO3zWl
[7] Webseite Rust Foundation: https://bit.ly/3DNgO45
[8] Webseite rustup: https://bit.ly/3EUTiDR
[9] Webseite Rust Crate Registry: https://bit.ly/3dLKmor
[10] droogmic, „MicroRust", April 2020: https://bit.ly/3EUWFdM
[11] droogmic, „MicroRust: Buttons", April 2020: https://bit.ly/3DWes30
[12] bors and NitinSaxenait, „Discovery", Dezember 2021: http://bit.ly/3GERDT7
[13] Webseite Get Ada Now: https://bit.ly/3GHyikw
[14] F. Chouteau, „First beta release of Alire, the package manager for Ada/SPARK", AdaCore, Oktober 2020: https://bit.ly/3EUXOSC
[15] F. Chouteau, „Microbit_examples", Dezember 2021: https://bit.ly/3mnH7bx
[16] „Watch us turn an ESP32 into a full computer!", Toit, März 2021: https://bit.ly/3IM0WT8
[17] Webseite Toit Package Registry: https://bit.ly/3yokpo7
[18] Webseite Toit-Docs, Voraussetzungen: https://bit.ly/3oNSECq

Von Robert van der Zwan (Elektor)

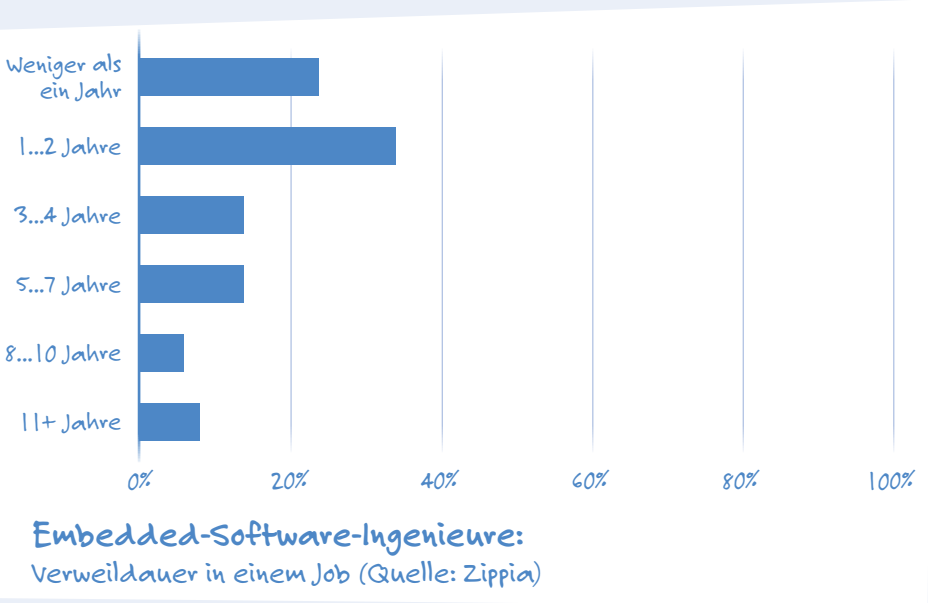
Embedded: Das Wachstum ist tatsächlich solide eingebettet

Laut dem Marktforschungsunternehmen The Brainy Insights sieht der globale Markt für eingebettete Systeme immer noch so vielversprechend aus, als ob es Covid-19 nie gegeben hätte. Die Untersuchungen führen zu einer durchschnittlichen jährlichen Wachstumsrate (CAGR) von 5,73 % für die Jahre 2021 bis 2028. Obwohl diese Schätzung im Vergleich zu einigen anderen Marktforschungsunternehmen eher vorsichtig ist, hat der Bericht des indischen Forschungsunternehmens wirklich einen sehr optimistischen Unterton. Automotive (Marktanteil 5 %), Gesundheit (10 %) und Kommunikation (45 %) sind vielversprechende Bereiche innerhalb des Embedded-Sektors und machen bereits 60 % des Umsatzes aus.

(Quellen: The Brainy Insights, The Insight Partners)

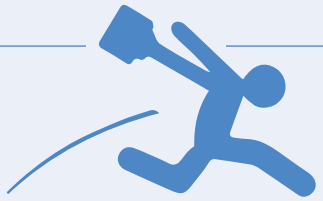


Embedded-Software-Ingenieure haben die (Qual der) Wahl!



Gibt es Hindernisse, die das Potenzial des Embedded-Marktes einschränken? Auf jeden Fall. Einer davon ist der Mangel an qualifizierten Software-Ingenieuren. Während (in den USA) im Jahr 2010 etwa 4,5 % der Embedded-Software-Ingenieure arbeitslos waren, sind es heute weniger als 2 %. Laut dem Personalvermittler *Built In* wird die Nachfrage nach Softwareingenieuren im Jahr 2028 um 21 % höher sein als 2021/2022. Das durchschnittliche Wachstum für alle Berufe wird im gleichen Zeitraum bei etwa 5 % liegen. Ingenieure für Embedded Software können sich aussuchen, wo sie arbeiten, was bei 60 % von ihnen zu einem Job-Hopping führt.

(Quellen: Built In, Career research site Zippia)



Wechseln von einem Job zum anderen

Szenario	Kosten	Dauer
Platinenlayout-Service mit optionaler Analyse und Optimierung der Signalintegrität	~5.000...35.000 $	~1...8 Wochen
Entwurf einer mikrocontroller-basierten Schaltung mit geringer bis mittlerer Komplexität, **Platinenlayout** und **Firmware-Entwicklung**	~25.000...50.000 $	~6...12 Wochen
FPGA-VHDL-Entwicklung	~15.000...125.000 $	~1 Monat bis mehr als 6 Monate
Firmware/Software-Entwicklung	~10.000...125.000 $	~2 Wochen bis mehr als 6 Monate

Warum können Ingenieure für eingebettete Software von einem Job zum anderen wechseln, ohne sich dabei unwohl zu fühlen? Es liegt daran, dass die Entwicklung eines typischen Embedded-Systems ungefähr sechs Monate dauert. Werfen Sie einen Blick auf die Tabelle, die der Embedded-Systems-Entwickler *AppliedLogix* erstellt hat. Dieses amerikanische Unternehmen hat Kunden in einer Vielzahl von Branchen und ist daher in der Lage, einige Durchschnittswerte anzugeben. Diese Durchschnittswerte zeigen, wie schnell ein Ingenieur für eingebettete Software einen Auftrag abschließen, sich gut dabei fühlen und sich dem nächsten Auftrag innerhalb oder außerhalb desselben Unternehmens zuwenden kann.

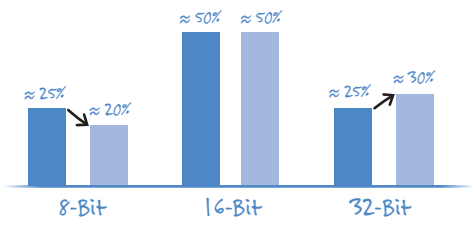
(Quellen: AppliedLogix, Zippia)

8-Bit-MCUs geraten etwas ins Hintertreffen

Obwohl es am Ende dieses Jahrzehnts immer noch 8-Bit-Mikrocontroller geben wird, sind die 32-Bit-MCUs stärker auf dem Vormarsch als ursprünglich erwartet. Diese Tatsache hat höchstwahrscheinlich mit dem schnellen Wachstum von eingebetteten Echtzeitsystemen zu tun. Während eingebettete Systeme bis 2028 insgesamt jährlich um 5,7 % wachsen werden, dürften eingebettete Echtzeitsysteme um etwa einen Prozentpunkt stärker wachsen (6,9 %). Auch der Stückpreis von 32-Bit-MCUs ist stetig gesunken. Die einfache Formel 25 % 8-Bit, 25 % 32-Bit und 50 % 16-Bit gehört der Vergangenheit an.

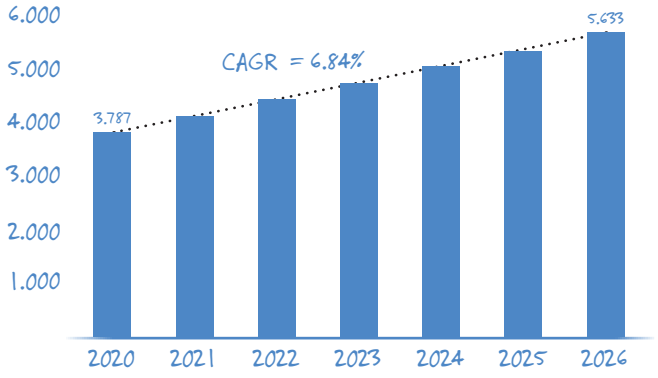
(Quellen: Allied Market Research, The Brainy Insights, Grand View Research)

Marktanteile Mikrocontroller, 2021 vs. 2027



Globaler Markt für Embedded Security, 2020...2026 (Mrd. $)

(Quelle: Knowledge Sourcing Intelligence)



Wird ausreichend für **Sicherheit** gesorgt?

Als wir vor zwei Jahren das letzte Mal eine Infografik über Security-Lösungen für eingebettete Systeme erstellt haben, wurde deutlich, dass die Ingenieure bei der Implementierung dieser Lösungen im Rückstand sind. Dies ist jetzt nicht mehr der Fall! Der Markt für Security-Lösungen für eingebettete Systeme wächst schneller als der Markt für eingebettete Systeme selbst. Der Unterschied in der Wachstumsrate ist zwar nicht spektakulär (5,73 % gegenüber 6,84 % in den kommenden Jahren), reicht aber aus, um nicht länger beunruhigt zu sein. Nicht nur Softwareschutz wird implementiert, sondern auch Hardware-Identifikation.

(Quellen: The Brainy Insights, Knowledge Sourcing Intelligence)

Neue Wege für **Industrie** und **Automobilbranche** mit **5G**

Von Mark Patrick (Mouser Electronics)

5G ist mehr als nur ein Fortschritt für den Mobilfunk, wo dank höherer Download-Geschwindigkeiten das Surfen auf einem Smartphone spürbar verbessert wird. Die eigentlichen Auswirkungen von 5G werden aber eher von Applikationen ausgehen, die erst noch entwickelt werden müssen. Werfen wir dazu einen Blick auf die Industrie- und Automobilbranche und darauf, wie Sie hier mit 5G neue Wege gehen können.

Warum 5G im Bereich der industriellen Fertigung?

Die kabelgebundene Architektur vieler intelligenter Fabriken stellt ein Hemmnis für die Weiterentwicklung dar. Die verschiedenen Sensoren, Aktoren und Steuerungen in automatisierten Anlagen sind über durchaus bewährte Netzwerktechnologien wie Industrial Ethernet, Profinet und CANbus miteinander verbunden, allerdings sind selbst kleinste Änderungen an den Produktionsanlagen wegen der fest definierten Konnektivität zeitaufwändig und kostspielig. Die bisherigen Generationen drahtloser Netzwerke, selbst wenn sie auf der 4G/LTE-Technologie basieren, sind nicht in der Lage, die für die Autonomie erforderliche Echtzeit-Reaktionsfähigkeit und niedrige Latenzen zu bieten. Außerdem sind Fertigungsbereiche eine schwierige Betriebsumgebung, in der die Kommunikation mit den bisherigen Funktechnologien durch ein hohes Maß an elektrischem Rauschen und Störungen in ihrer Leistung beeinträchtigt ist. Die erweiterten Netzwerkfähigkeiten von 5G können einige dieser Probleme lösen und die Systemeffizienz und -flexibilität erhöhen.

Eine der Schlüsselfunktionen jeder automatisierten Fabrik ist das Monitoring. 5G ermöglicht mMTC (Massive Machine-Type Communications) und wird damit den Anforderungen umfangreicher drahtloser Sensornetzwerke (WSN) gerecht. 5G ist auch energieeffizienter als seine Vorgänger, was entscheidend für längere Akkulaufzeiten der angeschlossenen Geräte ist und somit zur Minimierung des Wartungsaufwands beiträgt.

Die für Bewegungssteuerung und Industrierobotik erforderliche Präzision und Echtzeit-Empfindlichkeit wurde bisher über Time-Sensitive Networking (TSN) mit kabelgebundenem Industrial Ethernet als bevorzugter Netzwerktechnologie umgesetzt. Mit seiner ultrazuverlässigen Kommunikation mit geringer Latenz (URLLC) ist 5G eine effiziente drahtlose Alternative und ermöglicht zudem den Einsatz von Cloud-Robotik.

Drei verwandte Technologien, die zunehmend in Fertigungsumgebungen eingesetzt werden, sind Virtual Reality (VR), Augmented Reality (AR) und Künstliche Intelligenz (KI). Dank hoher Geschwindigkeit und URLLC ermöglicht 5G die Edge-Verarbeitung, sodass energieintensive Berechnungen an die Cloud übergeben werden können, was den Einsatz von weniger komplexen und somit kostengünstigeren Geräten in der Fertigung ermöglicht.

Chancen und Herausforderungen bei der Implementierung von 5G

Um bisherige Investitionen in kabelgebundene und drahtlose Netzwerktechnologien zu schützen, müssen sich 5G-Projekte nahtlos in die bestehende Infrastruktur integrieren. Eine der größten Herausforderungen besteht in der Komplettabdeckung in Gebäuden, die für Mobilfunknetzbetreiber (MNOs) bisher keine Priorität hatte. Neue Entwicklungen bei Open-RAN-Technologien senken die Betriebskosten von 5G-Funkzugangsnetzen (5G RAN), sodass private 5G-Netze eine realistische Option darstellen. Für Unternehmen, die diese Non-Public Networks (NPN) bevorzugen, stellen Regulierungsbehörden weltweit ein dediziertes, kosteneffizientes Frequenzspektrum zur Verfügung. Darüber hinaus können private 5G-Netze je nach den betrieblichen Anforderungen des Unternehmens entweder vollständig vom öffentlichen Netz isoliert oder mit diesem gemeinsam genutzt werden.

5G und die Ära der vernetzten Fahrzeuge

Der Automobilsektor wird den Prognosen zufolge ebenfalls eine Vorreiterrolle bei der Einführung von 5G spielen, auch wenn es noch einige Jahre dauern könnte, bis der

Automatisierungsgrad Level 5 (Vollautomatisierung) kommerzielle Relevanz erreicht. Es ist jedoch wahrscheinlich, dass das nächste Auto, das Sie kaufen, internetfähig ist, um Telematik, C-V2X (Cellular Vehicle-to-Everything) und Infotainment zu verwalten.

Das vernetzte Auto von heute kann bis zu 4 Terabyte an Daten pro Tag generieren, was etwa 500 Filmen entspricht. Neueste Entwicklungen bei C-V2X-Kommunikationstechnologien nutzen diese Daten bereits auf vielfältige Weise. So werden Daten aus den Motormanagementsystemen zur vorausschauenden Wartung an entfernte Servicezentren gesendet. Lokale Informationen zur Verkehrssituation und zum Wetter können zudem in öffentliche Sicherheitssysteme einfließen. Sogar das Fahrerverhalten und der Kilometerstand des Fahrzeugs lassen sich in Datenbanken für nutzungsbasierte Versicherungstarife nutzen.

In den letzten 5 Jahren hat das 3GPP (3rd Generation Partnership Project), ein weltweites Standardisierungsgremium für Mobilfunktechnologien wie Funkzugang, Kernnetz und Dienstfunktionen zur Bereitstellung vollständiger Systemspezifikationen für die mobile Telekommunikation, die Funktionalität von C-V2X auf Basis neuester Entwicklungen im Mobilfunk erweitert. Die in Release 16 vereinbarten Standards ebnen den Weg für fortgeschrittene Fahrerassistenzsysteme (FAS).

Auch wenn die breite Verfügbarkeit von selbstfahrenden Kraftfahrzeugen noch Zukunftsmusik ist, gab es bereits einige viel beachtete Versuche. Tesla, Google und BMW sorgen für Schlagzeilen, schüren die Erwartungen der Öffentlichkeit und bringen Schwung in die Sache. Viele Fahrzeuge der Oberklasse verfügen bereits über ein gewisses Maß an Autonomie, einige bis Level 3 (Automatisierter Modus), die ebenfalls auf C-V2X-Technologien beruhen.

Obwohl 4G/LTE-Netze viele der oben genannten Applikationen unterstützen, wird die verfügbare Bandbreite durch das steigende Volumen der gemeinsam genutzten Daten immer stärker beansprucht. Da die kritischen Sicherheits- und Energiemanagementsysteme in den Fahrzeugen immer komplexer werden, werden niedrige Latenzen dringend notwendig. Um ein höheres Maß an Autonomie zu erreichen, müssen die Latenzen der Netzwerke und der Cloud-Edge-Verarbeitungsfunktionen das Niveau menschlicher Reflexe erreichen. Auch für anspruchsvollere FAS muss das vernetzte Fahrzeug in Echtzeit auf Ereignisse in der Umgebung reagieren. Die derzeitigen drahtlosen Netzwerke stoßen hier an ihre Grenzen und werden immer mehr zu einem Hindernis – ohne 5G wird es keine selbstfahrenden Autos geben.

Fazit

Hauptaugenmerk beim Ausbau der 5G-Netze wurde zunächst auf die Verbesserung von 4G/LTE durch Einführung der 3GPP-Spezifikationen für New Radio Non-Standalone (5G NR NSA), Release 15, gelegt, was die Einführung einer begrenzten Anzahl von 5G-Diensten ermöglichte. Das wahre Potenzial von 5G wird jedoch erst bei Einführung von 3GPP, Release 16 und dem angekündigten Release 17, zum Tragen kommen. Applikationen wie autonome Fahrzeuge und autonome Fabriken lassen sich nur dann umsetzen, wenn sie einfachen Zugang zu Netzen dieses Leistungsniveaus haben. Die Einführung von 5G verlief anfangs nur zögerlich, was vor allem an den Auswirkungen der weltweiten Pandemie lag. Die zweite Phase der Einführung des 5G-Netzes wird sicherlich die Nachfrage nach einem breiten Spektrum von noch zu entdeckenden Applikationen beschleunigen.

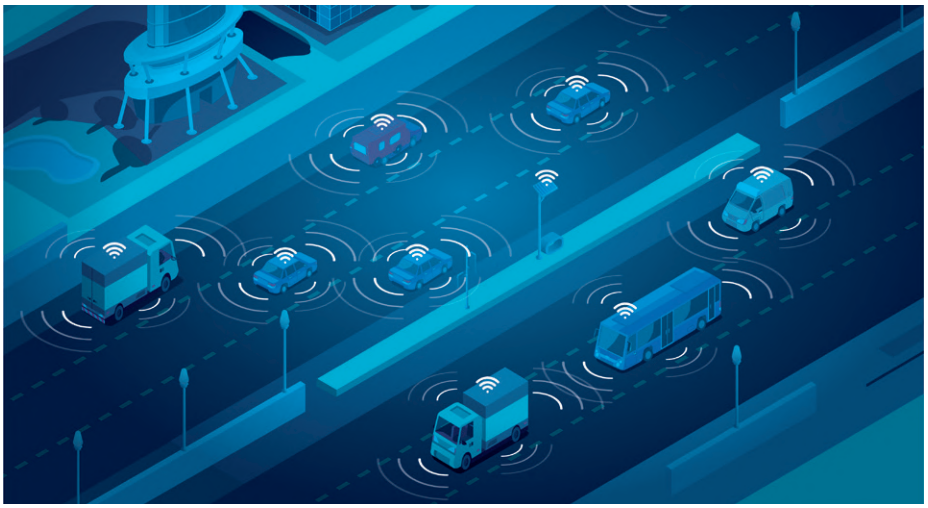
220061-02

Über den Autor

Als Technical Marketing Manager für EMEA bei Mouser Electronics ist Mark Patrick für die Erstellung und Verbreitung von technischen Inhalten in der Region verantwortlich - Inhalte, die für Mousers Strategie zur Unterstützung, Information und Inspiration der Elektronik-Branche von zentraler Bedeutung sind.

Bevor er das Technische Marketing Team leitete, war Patrick Teil des EMEA-Lieferanten-Marketing-Teams und spielte eine wichtige Rolle beim Aufbau und der Entwicklung von Beziehungen zu wichtigen Produktionspartnern. Zusätzlich zu einer Vielzahl von technischen und Marketing-Positionen war Patrick acht Jahre lang bei Texas Instruments in den Bereichen Anwendungsunterstützung und technischer Vertrieb tätig.

Weitere Informationen zu 5G finden Sie auf der Mouser-Website zu „Empowering Innovation Together“ unter **www.mouser.com/empowering-innovation/5G**



Drehspul**relais**

Bemerkenswerte Bauteile

Von **David Ashton** (Australien)

In dieser Ausgabe befassen wir uns mit einem wirklich merkwürdigen Bauteil. So merkwürdig, dass außer seinen physikalischen Eigenschaften nicht viel darüber herauszufinden war. Aber trotz des Aufbaus mit einer Drehspule sind wir ziemlich sicher, dass es sich nicht um ein Messgerät handelt!

Wenn man Drehspule sagt, denken die meisten von uns sicher sofort an ein Messgerät. Auch wenn sie inzwischen etwas aus der Mode gekommen sind, haben die meisten Leute schon mal eines gesehen oder benutzt. Ich spreche jedoch von Drehspulrelais. Stellen Sie sich vor, Sie schließen einen Draht an die Nadel eines solchen Messgeräts an und verbinden einen weiteren mit dem Endanschlag. Wenn das Messgerät den vollen Skalenausschlag erreicht, wird der Kontakt hergestellt. So funktionieren im Grunde diese Relais.

Ich habe ein solches Drehspulrelais vor einigen Jahren in einer Schalttafel gefunden, und es hat einige Zeit gedauert, bis ich herausgefunden habe, was es überhaupt ist. Sie sind nicht nur wunderschön, sondern gehören auch zu den empfindlichsten elektrischen Bauteilen, die ich je gesehen habe.

Sie wurden von BBC Goerz Electro hergestellt. Obwohl BBC Goerz viele professionell aussehende Test- und Messgeräte herstellte, ist es nicht einfach, Informationen über dieses Unternehmen zu finden. Wir wissen jedoch, dass Goerz ein österreichisches Optikunternehmen war, dass BBC Brown Boveri Corporation bedeutet und ein schweizerisches Elektrounternehmen war. Die Firmen haben im Wandel der Jahrzehnte viele Veränderungen, Umfirmierungen und Fusionen durchgemacht. Um darüber zu berichten, fehlt hier der Platz, aber Sie können gerne die interessante Historie einer mehr als 120-jähigen paneuropäischen Industriegeschicht im Internet erforschen.

Trotz dieses Hintergrundwissens über den Hersteller konnte ich keine Informationen über diese Relais finden, abgesehen von jemandem, der einige auf eBay für etwa 100 $ pro Stück verkauft. Sie sind auf einer 8-poligen Röhrenfassung aufgebaut. Als ich sie zum ersten Mal zu Gesicht bekam, fragte ich mich, ob es sich um eine Art Röhre handeln könnte. Sie besitzen jedoch ein durchsichtiges Kunststoffgehäuse, das sich abschrauben lässt, so dass man den komplizierten Mechanismus im Inneren genau betrachten kann. Die Typennummer 91041-2 ist deutlich auf dem Gehäuse aufgedruckt, zusammen mit einer Seriennummer (**Bild 1**). Außerdem befindet sich an der Seite ein Anschlussplan, der das Testen der Geräte erleichterte (**Bild 2**).

Ich habe eine Prüfvorrichtung für diese Relais gebaut - praktischerweise habe ich drei passende Fassungen auf einer Halterung

Bild 1. Die Relais, hier von hinten gezeigt, mit ihrer Typennummer auf dem Kunststoffgehäuse.

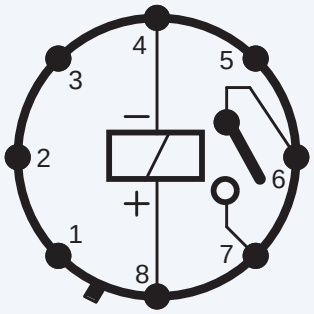


Bild 2. Dieser Anschlussplan ist auf der Seite des Relais aufgedruckt.

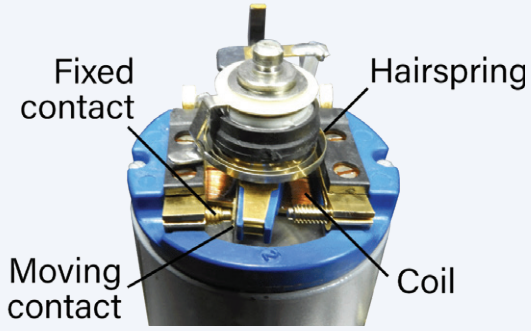


Bild 3. Auf dem Foto ist die Lage der Spiralfeder, der Spule und der Kontakte eingezeichnet.

mit vielen Löchern für die Montage anderer Bauteile. Ich habe ein 470-kΩ-Potentiometer in Reihe mit der Relaisspule und einer 12-V-Spannungsversorgung verwendet und den Kontakt zur Ansteuerung einer LED genutzt, damit ich sehen konnte, wann er sich schließen würde. Die Relais arbeiten mit einem Strom von nur 260 µA bei einer Spannung von 113 mV! Zeigen Sie mir ein anderes Relais, das so empfindlich ist!

Die Kontakte sind klein (**Bild 3**), und man könnte sie zum Betätigen eines anderen Relais verwenden, um eine sinnvolle Leistung zu schalten. Heutzutage würde man zu diesem Zweck natürlich einen Optoisolator verwenden, was mich zu der Annahme bringt, dass diese Relais ziemlich alt sind, aber man bräuchte immer noch etwas mehr Elektronik, um solche Spezifikationen zu erzielen. Wahrlich, diese Relais sind wirklich sehr eigenartig!

210557-02

Haben Sie Fragen oder Kommentare?

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

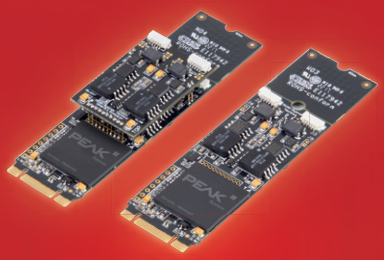
Anzeige

You CAN get it...

Hardware und Software
für CAN-Bus-Anwendungen...

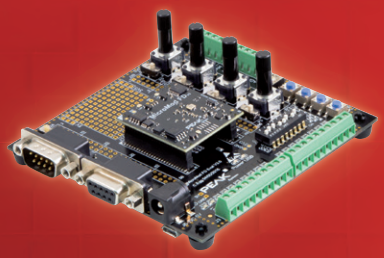
PCAN-MiniDiag FD

Handheld zur grundlegenden Diagnose von CAN- und CAN-FD-Bussen. Messung der Bitrate, Terminierung, Buslast und Pegel am D-Sub-Anschluss.



PCAN-M.2

CAN-FD-Interface für M.2-Steckplätze. Erhältlich als Ein-, Zwei- und Vierkanalkarte inkl. Software, APIs und Treiber für Windows und Linux.



PCAN-MicroMod FD

Einsteckmodul mit I/O und CAN-FD-Interface. Erhältlich mit Evaluation-Board für die Entwicklung eigener Anwendungen.

Irrtümer und technische Änderungen vorbehalten.

www.peak-system.com

PEAK System

Otto-Röhm-Str. 69
64293 Darmstadt
Germany
Tel.: +49 6151 8173-20
Fax: +49 6151 8173-29
info@peak-system.com

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!ng!

Elektor Archiv 1970-2021 (USB-Stick)

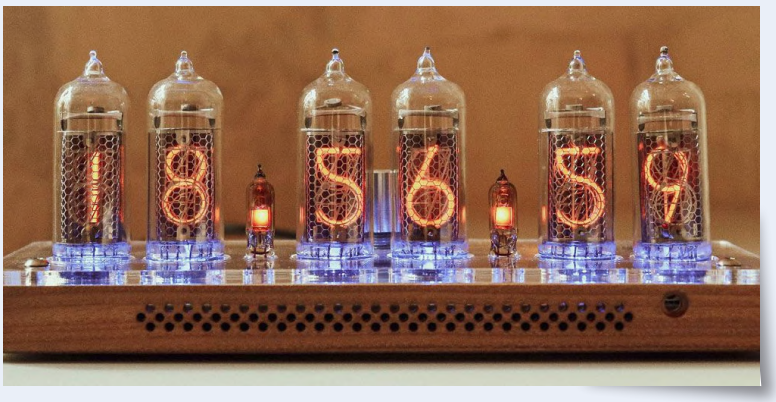
Preis: 199,95 €
Mitgliederpreis: 99,95 €

www.elektor.de/20072

6-stellige Nixie-Uhr mit IN-14 Röhren

Preis: ~~299,00 €~~
Sonderpreis: 269,00 €

www.elektor.de/20044



Great Scott Gadgets Opera Cake – Antennenschalter für HackRF One

Preis: 169,95 €

Mitgliederpreis: 152,96 €

www.elektor.de/20083

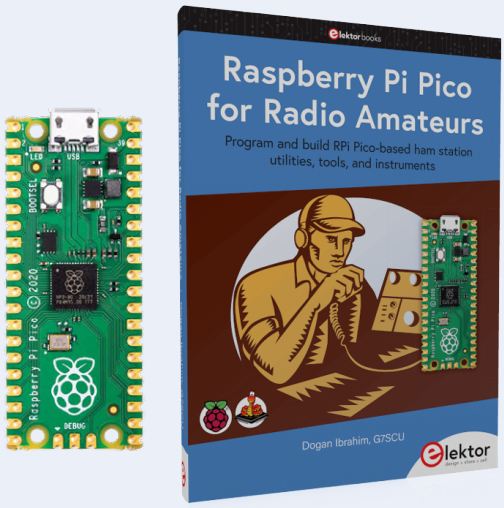
Velleman VTSS210 Vielseitige Reparaturstation für SMD-Komponenten

Preis: 84,95 €

Mitgliederpreis: 76,46 €

www.elektor.de/19948

Raspberry Pi Pico for Radio Amateurs + GRATIS Raspberry Pi Pico RP2040

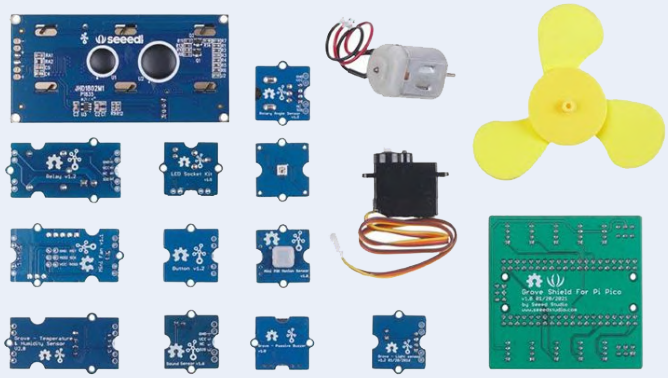


Preis: 34,95 €

Mitgliederpreis: 31,46 €

www.elektor.de/20041

Seeed Studio Grove Starterkit für Raspberry Pi Pico



Preis: 54,95 €

Mitgliederpreis: 49,46 €

www.elektor.de/20016

Zutritt für Unbefugte verboten

Alles dreht sich um die Werkzeuge...

Von **Joachim Schröder** und **Eric Bogers** (Elektor)

Wer sich auch nur ansatzweise ernsthaft mit Elektronik beschäftigt, kommt früher oder später nicht um ein wenig Handwerk - sprich: Holz- und Metallbearbeitung - herum, und sei es nur, um der mit viel Einfallsreichtum und Mühe gebauten Schaltung ein ansehnliches oder zumindest stabiles Gehäuse zu spendieren. Und dazu braucht man eine gewisse Sammlung von Werkzeugen.

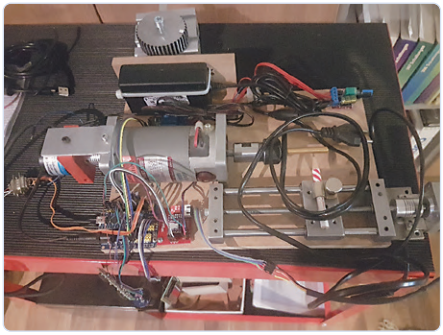


Bild 1. Ein früher Prototyp mit einem Gleichstrommotor für die „Hauptspindel", einem Nema-17-Schrittmotor für die Führungsspindel und einer Motorsteuerung mit einem einfachen PWM-Modul.

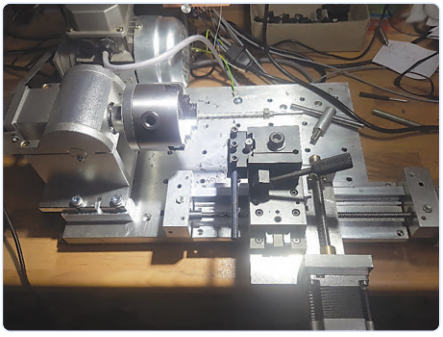


Bild 2. Ein besser ausgestatteter Prototyp mit AC-Motor, variablem Drehzahlregler und X- und Z-Schrittmotoren.

Wenn es nur um ein paar Löcher für die Spindeln von Potentiometern oder dergleichen in einem gekauften, vorgefertigten Gehäuse geht, dann könnte man (zumindest für den Anfang) mit einer Stichsäge (und einer ruhigen Hand) sowie einer Hobbybohrmaschine mit einem nicht allzu wackeligen Bohrständer auskommen. Aber bei aufwändigeren Mechatronik-Projekten mit beispielsweise komplizierten Spindelsystemen oder ähnlichem macht sich eine (kleine) Drehmaschine sehr schnell bezahlt, es sei denn, man ist bereit, alle mechanischen Arbeiten für relativ viel Geld außer Haus zu geben.

Herr Schröder ist seit langem im Besitz einer kleinen, manuell gesteuerten Drehbank, die er (unter anderem) beruflich für Spezialaufträge und Reparaturen einsetzt. Vor einiger Zeit überlegte er, diese Drehbank mit einer Computersteuerung auszustatten. Das Stichwort heißt CNC (Computer Numerical Control) und meint eine elektronisch synchronisierte Führungsspindel.

Bei einer solchen synchronisierten Führungsspindel wird der Schlitten mit dem Schneidwerkzeug von einem Servo- oder Schrittmotor synchron zur Drehung der Hauptspindel bewegt (zum Beispiel 0,1 mm pro Umdrehung). Die Hauptspindel wird zu diesem Zweck mit einem Drehgeber ausgestattet. Der Vorteil einer solchen CNC-Konstruktion besteht darin, dass keine Zahnräder getauscht werden muss, wenn ein größerer oder kleinerer Verfahrweg pro Umdrehung erforderlich ist. Schließlich entschied sich Herr Schröder für den Bau einer völlig neuen CNC-Drehmaschine - und sei es nur, weil er es sich nicht leisten konnte, auf die vorhandene manuelle Drehmaschine für längere Zeit zu verzichten. Zunächst wurden einige Prototypen gebaut (**Bild 1** und **Bild 2**). Dabei zeigte sich, dass eines der größten Probleme eigentlich softwaretechnischer Natur ist: die Berechnung, wann der Schrittmotor einen Schritt vorwärts gehen sollte. Wegen der dafür erforderlichen Rechenzeit war es nicht möglich, dies mit Fließkommaberechnungen zu tun (was der naheliegende Ansatz gewesen wäre). Denn wenn sich die Hauptspindel mit (zum Beispiel) 1200 U/min dreht und ein Encoder verwendet wird, der 1024 Impulse pro Umdrehung erzeugt, wären 20480 Fließkommaberechnungen pro Sekunde erforderlich! Die Umwandlung dieser Berechnungen in reine Ganzzahloperationen erwies sich als eine der größeren Herausforderungen.

Aus China...

Also wurde eine kleine (manuelle) Drehbank mit stabilem Gussrahmen chinesischer Provenienz bestellt (**Bild 3** und **Bild 4**). Das Gerät war allerdings nicht mehr als eine gute Ausgangsbasis, denn die Kunststoffzahnräder waren offensichtlich völlig unzureichend (und wurden ohnehin entfernt). Immerhin: Der Drehgeber für die Synchronisierung der Führungsspindel wurde mit einem Plastik-Zahnrad aus dem Originalgetriebe ausgestattet und mit Hilfe eines Adapters montiert (**Bild 5** und **Bild 6**).

Nach nur fünf Tagen gab der ursprüngliche (chinesische) Motor-Drehzahlregler für den 230-V-Gleichstrommotor mit einem lauten Knall den Geist auf und wurde durch einen Drehstrommotor und einen Qualitäts-Frequenzumrichter ersetzt. **Bild 7** zeigt den neuen Wechselstrommotor mit einer Leistung von 0,37 kW mit seinem Drehzahlregler.

Bild 8 schließlich zeigt das (vorläufige?) Endergebnis all dieser Bemühungen. Leider ist es im Rahmen dieses Artikels nicht möglich, auf weitere Einzelheiten dieses Projekts einzugehen, aber Interessierte sind eingeladen, sich direkt an Herrn Schröder zu wenden (siehe Textkasten).

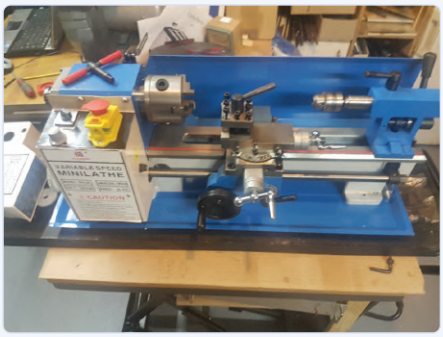


Bild 3. Nach langem Warten kam die Drehmaschine aus China an, …

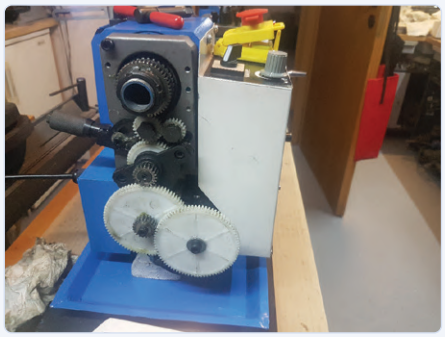


Bild 4. … die zwar ein gutes Chassis, aber unbrauchbare Kunststoff-Zahnräder besaß. Da wurde eindeutig am falschen Ende gespart!

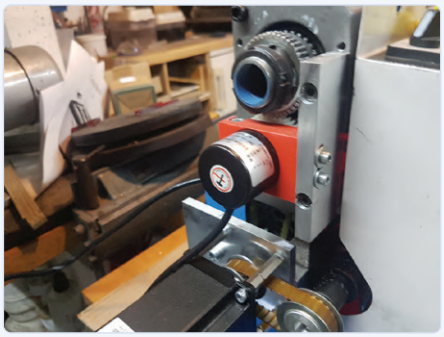


Bild 5. Der Geber für die Synchronisation wird mit einem Adapter montiert…

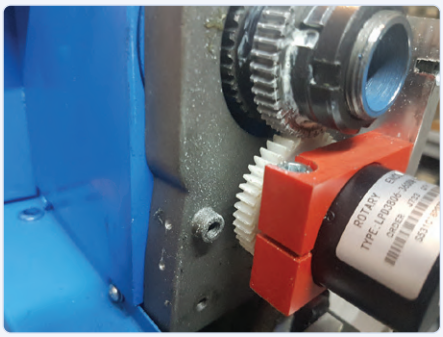


Bild 6. …und über ein aus dem demontierten Getriebe „gerettetes" Zahnrad angetrieben.

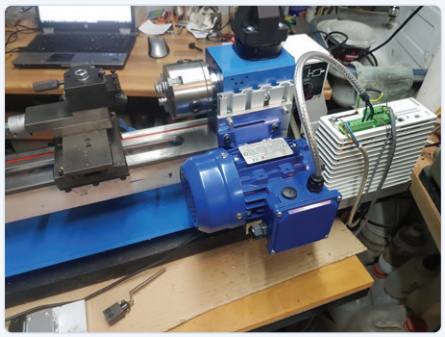


Bild 7. Hier ist schon der 0,37-kW-Motor mit einer Adapterplatte montiert.

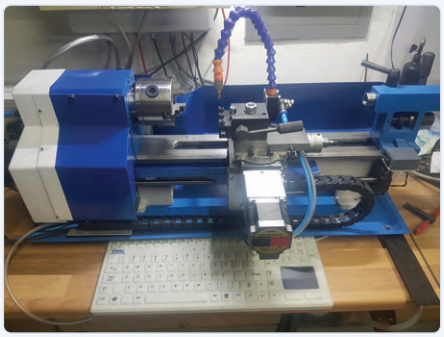


Bild 8. Das Ergebnis all dieser Bemühungen.

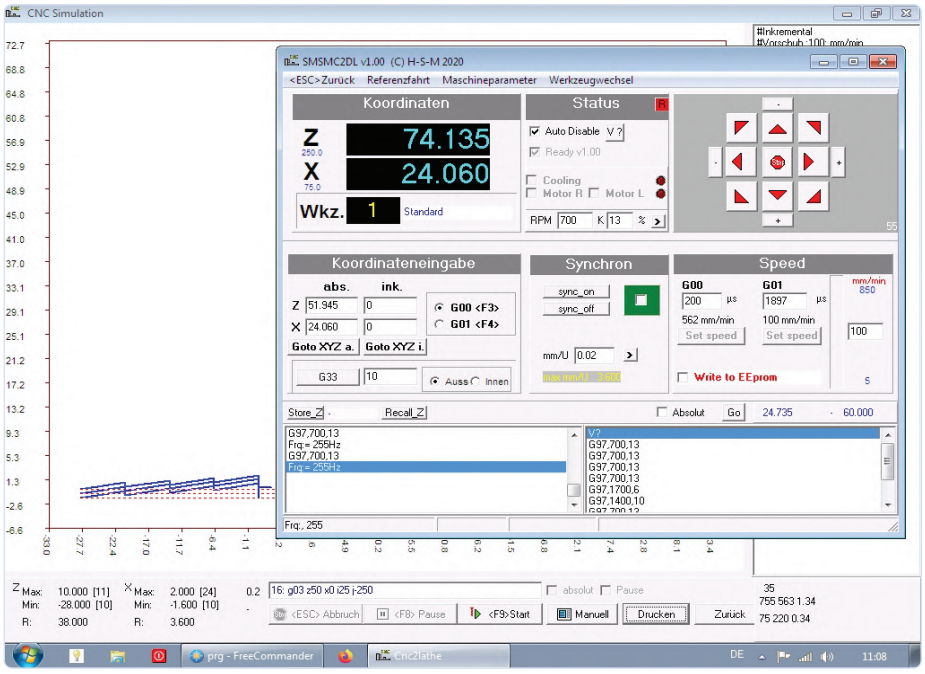


Bild 9. Ein Eindruck von der Software.

Ein Wort zur Software

Zum Abschluss dieses Artikels lassen wir Herrn Schröder kurz zu Wort kommen: „Vor mehr als 30 Jahren habe ich meine erste CNC-Fräsmaschine gebaut und die Software dafür selbst geschrieben. Das war damals noch unter DOS, wo die Hardware direkt über den LPT-Port angesteuert wurde. In den folgenden Jahren baute ich modernere und immer bessere Maschinen, für die die Software auf Windows portiert wurde. Da Windows aber keine Echtzeitfähigkeiten bietet, brauchte ich einen Mikrocontroller für die Steuerung der Achsen. Dieses Problem löste ich anfangs (etwa zu Beginn dieses Jahrhunderts) mit Hilfe von CCBasic-Controllern von Conrad. Das Fehlen von Fließkommafunktionen machte diese Aufgabe jedoch zu einer echten Herausforderung. Jetzt verwende ich Arduino-Boards, die hervorragend funktionieren und 3D-Bewegungen mit Geschwindigkeiten von bis zu 1200 mm/min direkt steuern. Es war naheliegend, diese Software für die 3D-CNC-Fräse als Ausgangspunkt für die 2D-Drehmaschine zu verwenden. Leicht gesagt, leicht getan: **Bild 9** vermittelt einen Eindruck der Benutzerschnittstelle."

210605-03

Ein Beitrag von

Text und Fotos: **Joachim Schröder**
Herausgeber: **Eric Bogers**
Übersetzung: **Rolf Gerstendorf**
Layout: **Giel Dols**

Haben Sie Fragen oder Kommentare?

Haben Sie technische Fragen oder Anmerkungen zu diesem Artikel? Senden Sie eine E-Mail an den Autor über js@hsm-ebs.de oder an die Redaktion von Elektor über editor@elektor.com.

Die Neuronen in neuronalen Netzen verstehen

Teil 4: Eingebettete Neuronen

Von **Stuart Cording** (Elektor)

Da unser neuronales Netz auf PCs und Laptops voll funktionsfähig ist, können wir auf die Implementierung des mehrschichtigen Perzeptrons vertrauen. Viele Anwendungen erfordern eine geringe Leistungsaufnahme und eine minimale Latenzzeit, die ein Mikrocontroller bietet. Andere möchten ihre privaten Daten einfach nicht mit fremden Cloud-KI-Diensten teilen. Deshalb machen wir unser neuronales Netz „Arduino-fähig“ und übertragen den Ampelklassifizierungscode in die Welt der eingebetteten Systeme.

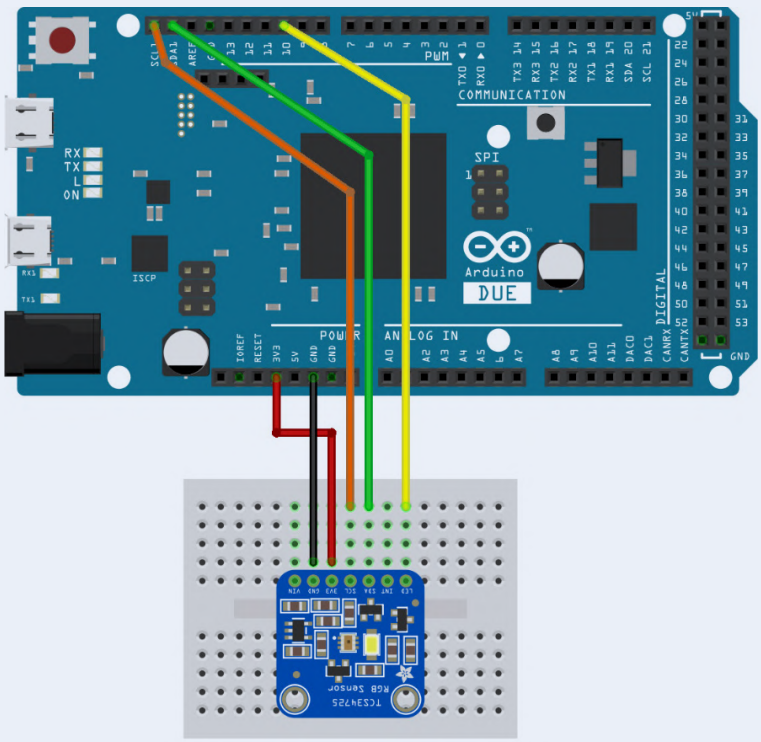


Bild 1. Schaltplan für den RGB-Sensor TCS34725 in Verbindung mit einem Arduino Due.

Tools und Plattformen für maschinelles Lernen (ML) scheinen wie Pilze aus dem Boden zu schießen. Ist die Aufgabe noch so komplex und der Datensatz noch so groß, es gibt immer einen Cloud-Dienst, der diese Aufgabe bewältigen kann. Dennoch gibt es viele Fälle, in denen eine Cloud-basierte ML-Implementierung ungeeignet ist. Wenn Sie sensible oder persönliche Daten verarbeiten, möchten Sie diese vielleicht nicht über das Internet übertragen, um sie von einem ML-Tool analysieren zu lassen. Ein weiteres Beispiel sind eingebettete Anwendungen für die Automobilindustrie oder andere Echtzeitanwendungen. Solche Systeme erfordern eine sofortige Entscheidung, so dass die Latenzzeit einer Internetverbindung oder die Gefahr, überhaupt keine Verbindung zu haben, eine lokale ML-Lösung erfordert. Dies wird auch als „On-Edge Machine Learning“ [1] bezeichnet.

Der Betrieb eines neuronalen Netzes On-Edge erfordert, dass selbst einfache Mikrocontroller ML-Algorithmen ausführen können. Aufgrund der für das Training erforderlichen Prozessorleistung wird das Netz jedoch häufig in der Cloud oder auf einem leistungsstarken PC trainiert. Die resultierenden Gewichte werden anschließend auf den Mikrocontroller geladen, um einen internetfreien Betrieb mit geringer Latenz zu ermöglichen.

Dieser letzte Artikel der Serie überträgt unser Netzwerk mit mehrschichtigen neuronalen Perzeptronen (MLP) auf einen Arduino. In Verbindung mit einem RGB-Sensor bringen wir ihm bei, die Ampelfarben zu klassifizieren, wie wir es mit Processing auf dem PC oder Laptop getan haben. Mit einem 32-Bit-Arduino (Due oder Mo Pro) könnten wir die Lernphase zwar auf dem Mikrocontroller durchführen, werden dabei jedoch feststellen, dass dieser Vorgang ziemlich langwierig ist. Daher teilen wir die Aufgabe auch so auf, dass das Netz mit der Leistung eines PCs trainiert und dann in der latenzarmen, stromsparenden Umgebung einer eingebetteten Mikrocontroller-Anwendung ausgeführt wird.

Auswahl des Sensors

Zur Erkennung der Ampelfarben benötigt der Arduino einen geeigneten Sensor am Eingang. Der TCS34725 ist eine geeignete RGB-Sensorlösung, die von Adafruit auf einem Platinchen von 2 × 2 cm einsatzbereit erhältlich ist [2]. Damit können wir RGB-Daten genauso erfassen, wie wir es zuvor mit der Kamera getan haben, und sie auf die gleiche MLP-Konfiguration anwenden, die in den vorherigen PC-basierten Beispielen verwendet wurde. Das Board kann sowohl mit 5-V- als auch mit 3,3-V-Arduinos verwendet werden. Die

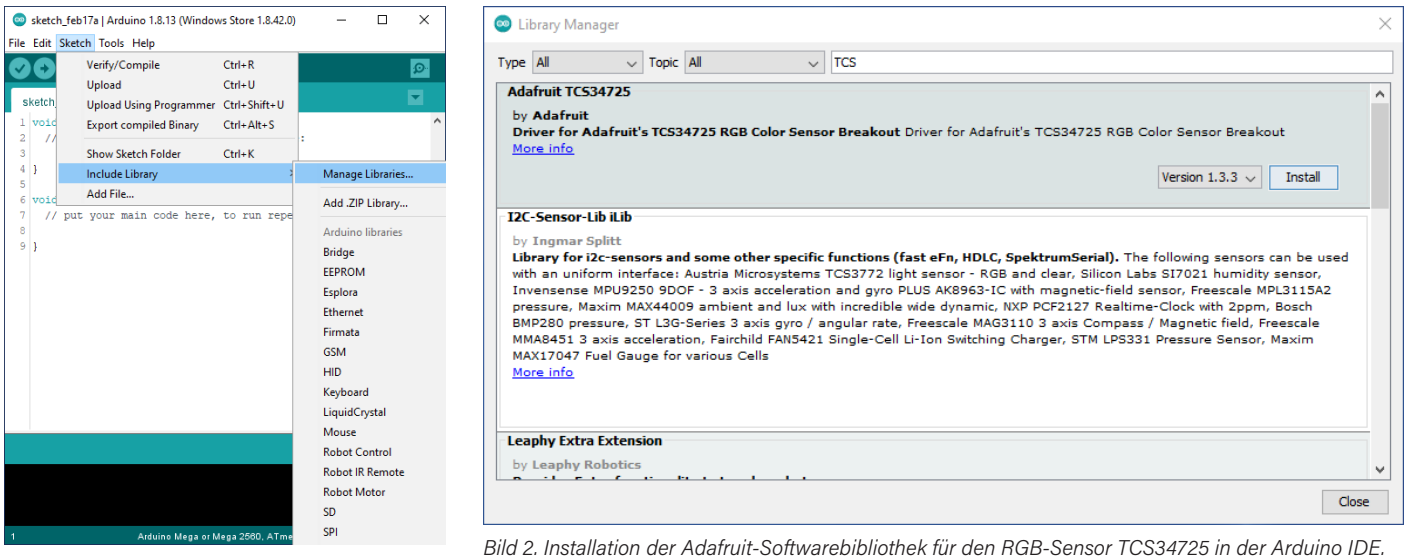


Bild 2. Installation der Adafruit-Softwarebibliothek für den RGB-Sensor TCS34725 in der Arduino IDE.

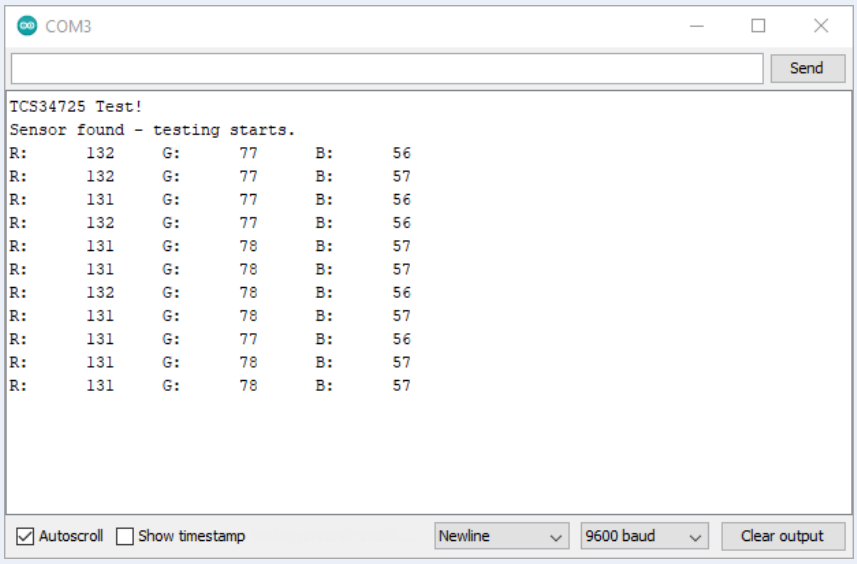


Bild 3. Ermittelte RGB-Werte, die von tcsrgbsensor.ino *ausgegeben werden.*

Sensordaten werden über I²C erfasst, wofür die Wire-Bibliothek erforderlich ist.
Um die „Ampel" gleichmäßig auszuleuchten, enthält das Board auch eine weiße LED, die über einen digitalen Arduino-Ausgang (Pin 10) gesteuert wird (**Bild 1**). Glücklicherweise wird das Board von einer gut aufgebauten Bibliothek unterstützt, sodass der Einstieg einfach und unkompliziert ist.
Wie zuvor werden wir den Code aus dem GitHub-Repository verwenden, der für diese Artikelserie vorbereitet wurde [3].
Vor der Ausführung des Codes müssen wir die Bibliothek für den RGB-Sensor installieren. Dies dürfte zum Glück auch einfach sein, da sie über den Bibliotheksmanager in der Arduino-IDE zugänglich ist. Wählen Sie in der Menüleiste einfach *Sketch -> Bibliothek einbinden -> Bibliotheken verwalten* und geben Sie dann „*tcs*" in das Suchfeld des Bibliotheksmanagers ein. Es sollte die Bibliothek *Adafruit TCS34725* erscheinen. Klicken Sie zur Installation einfach auf *Installieren* (**Bild 2**). Sollten Schwierigkeiten auftreten, können der Quellcode und die Header-Datei vom Adafruit GitHub-Repository [4] heruntergeladen und manuell zu den Projekten hinzugefügt werden.
Damit der Sensor und die Erfassung der RGB-Werte für das ML-Training richtig funktioniert, ist der Sketch *arduino/tcsrgbsensor/tcsrgbsensor.ino* erforderlich. Nach der Initialisierung des RGB-Sensors schaltet der Code die LED ein und beginnt, die RGB-Werte über den seriellen Ausgang auszugeben (**Bild 3**). Öffnen Sie *Werkzeuge -> Serieller Monitor*, um sie anzuzeigen. Die Baudrate ist auf 9600 eingestellt. Zur Verbesserung der Qualität der Messwerte und zur Verringerung des Einflusses anderer Lichtquellen lohnt es sich, eine mechanische Abschirmung um das Sensorboard herum zu bauen. Ein etwa 3 cm hoher und 10 cm langer Streifen aus schwarzem Karton ist dafür ideal (**Bild 4**). Die Abschirmung sorgt auch dafür, dass das Bild der Ampel in einem gleichmäßigen Abstand zum RGB-Sensor gehalten werden kann. Da der RGB-Sensor zuverlässige Ergebnisse liefert, können wir nun seine Bewertung der roten, gelben und grünen Farben anhand unseres ausgedruckten Ampelbildes (aus *trafficlight/resources*) erfassen. Die vom Autor erzielten Ergebnisse sind in **Tabelle 1** dargestellt.

Tabelle 1. Mit TCS34725 erfasste RGB-Werte für die drei Ampelfarben.

Ampelfarbe	R	G	B
Rot	149	56	61
Gelb	123	77	61
Grun	67	100	90

Eine MLP-Bibliothek für den Arduino

Nachdem wir die RGB-Werte bestimmt haben, implementieren wir das neuronale Netz für die Arduino-Plattform. Da Processing Java verwendet, können wir den Code der Neural-Klasse nicht einfach in ein Arduino-Projekt einfügen. Daher wurde die Java-Neural-Klasse leicht modifiziert, um sie in eine C++-Klasse zu verwandeln. In der Arduino-Welt können solche wiederverwendbaren Code-Objekte in „Bibliotheken" umgewandelt werden. Diese bestehen aus einer C++-Klasse und einer zusätzlichen Datei zur Hervorhebung der Schlüsselwörter der Klasse. Wenn Sie Ihre eigene Bibliothek schreiben oder besser verstehen möchten, wie C++-Klassen auf Arduino funktionieren, finden Sie auf der Arduino-Website ein hervorragendes Tutorial [5].
Der Code für unsere `Neural`-Bibliothek ist in *arduino/neural* zu finden. Die folgenden Arduino-Projekte enthalten der Einfachheit halber nur den Quellcode der Neural-Klasse in Sketch. Die Header-Datei *neural.h* muss ebenfalls dem Ordner hinzugefügt werden, in dem das Projekt gespeichert ist.
Die Verwendung der `Neural`-Klasse auf einem Arduino ist größtenteils identisch mit der Verwendung in Processing Das Erstellen unseres `network`-Objekts als globale Variable ist etwas anders und wird wie folgt durchgeführt:

```
Neural network;
```

Zur Erstellung des Objekts mit der gewünschten Anzahl von Eingangs-, verborgenen und Ausgangsknoten geben wir Folgendes ein:

```
network = new Neural(3,6,4);
```

Die Grundkonfiguration der Bias-Werte und der Lernrate wird dann wie zuvor in Processing kodiert:

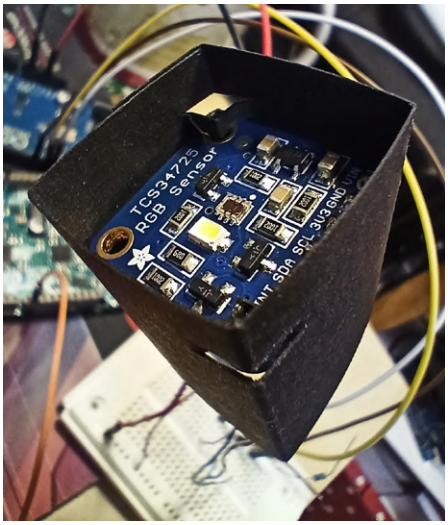


Bild 4. RGB-Sensor TCS34725 mit montierter schwarzer Kartonblende.

```
network.setLearningRate(0.5);
network.setBiasInputToHidden(0.35);
network.setBiasHiddenToOutput(0.60);
```

Von hier aus können wir die Methoden weiter verwenden, wie wir sie zuvor in Processing verwendet haben.

Arduino erkennt Ampelfarben

Jetzt können wir mit der Erforschung des MLP auf dem Arduino beginnen. Der Sketch */arduino/tlight_detect/tlight_detect.ino* folgt der gleichen Struktur wie das Processing-Projekt *tlight_detect.pde*. Das neuronale Netz (3/6/4) wird ab Zeile 40 konfiguriert und in einer Schleife ab Zeile 51 mit den RGB-Daten trainiert. Bevor der Code ausgeführt wird, müssen die zuvor erfassten RGB-Werte für „Rot“, „Gelb“ und „Grün“ ab Zeile 56 eingegeben werden:

```
teachRed(220, 56, 8);
teachAmber(216, 130, 11);
teachGreen(123, 150, 128);
```

Laden Sie den Code hoch und öffnen Sie den *Serial Monitor*, um die ausgegebenen Werte zu sehen (**Bild 5**). Wie zuvor sollte die Baud-Einstellung 9600 betragen. Das Projekt prüft, ob der RGB-Sensor funktionsfähig ist, bevor es die weiße LED während des Lernprozesses ausschaltet. Sobald MLP konfiguriert ist, durchläuft sie den Lernzyklus 30.000 Mal und verbessert dabei jedes Mal ihre Fähigkeit, alle drei Farben zu klassifizieren.

Zur Kontrolle des Lernprozesses wird alle 1.000 Zyklen (3.000 Lernepochen) ein Punkt ausgegeben. Im Vergleich zum PC ist der Lernprozess wirklich langsam: Jeder Aufruf einer Lernfunktion (`learnRed()`, etc.) benötigt etwa 5,55 ms bis zum Abschluss. Das Erlernen aller drei Farben benötigt auf einem Arduino Mo Pro mit einem SAMD21-Mikrocontroller etwa 8,5 Minuten. Wenn Sie die Ausführungszeit Ihrer Plattform überprüfen möchten, aktivieren Sie das Kontrollkästchen *„Zeitstempel anzeigen“* im Seriellen Monitor und ersetzen Sie Zeile 66 durch:

```
Serial.println(".");
```

Dadurch wird das Zeilenumbruchzeichen hinzugefügt und für jedes ausgegebene „.“-Zeichen ein Zeitstempel erzeugt.

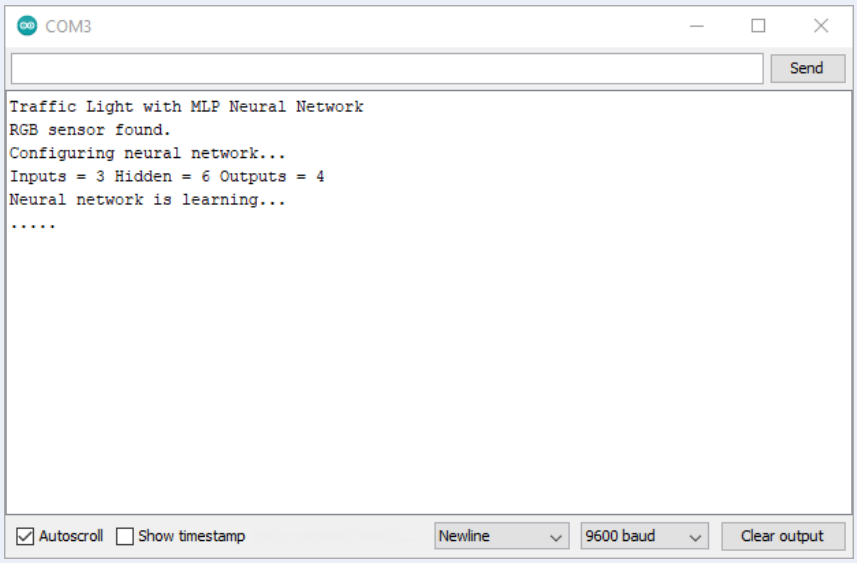


Bild 5. Ausgabe von tlight_detect.ino *während des Lernens.*

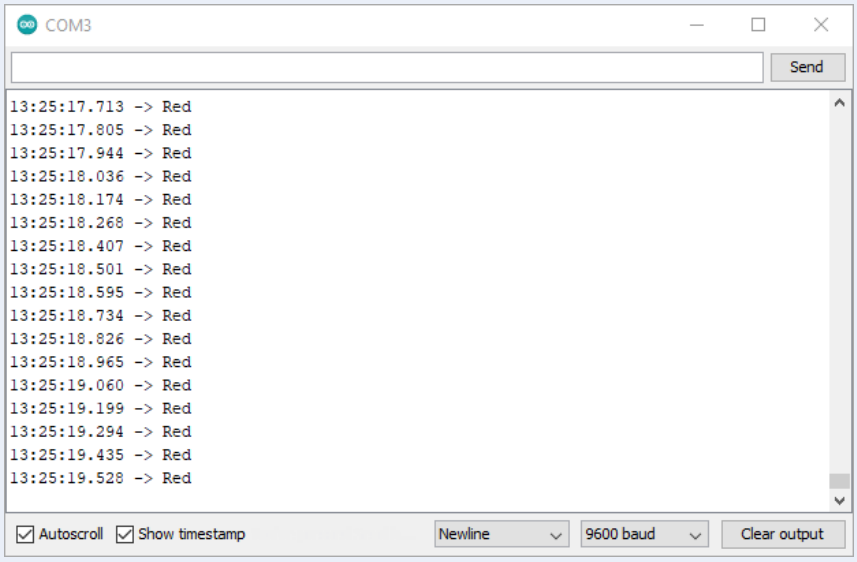


Bild 6. Ausgabe von tlight_detect.ino *nach dem Lernen und Erkennen von Farben.*

Sobald der Lernprozess abgeschlossen ist, macht sich der Arduino sofort daran, seine neuen Fähigkeiten unter Beweis zu stellen Immer, wenn eine Ampelfarbe erkannt wird, wird sie an den seriellen Monitor ausgegeben (**Bild 6**). Bei den Experimenten des Autors erkannte das neuronale Netz auch dann die Farbe „Gelb“, wenn nichts vor den Sensor gehalten wurde. Obwohl dies mit der Umgebungsbeleuchtung zusammenzuhängen schien, zeigt es eine Schwachstelle in der Implementierung auf.

Zur Verbesserung des Codes können wir dem MLP „andere“ Farben beibringen, wie wir es zuvor in Processing getan haben. Dies kann auch verwendet werden, um die Klassifizierung „Gelb“ zu unterdrücken, wenn kein Bild vorhanden ist. Der Sketch *tcsrgbsensor.ino* kann verwendet werden, um die Messwerte des Sensors für die Ampelumgebung, den Rahmen und den Bildhintergrund zu erfassen. Diese können dann ab Zeile 60 im Sketch *tlight_detect.ino* in den `teachOther()`-Funktionsaufrufen eingegeben werden.

Mit dem Sketch *tlight_detect.ino* wurden die in **Tabelle 2** gezeigten Werte erreicht und getestet. Die Klassifizierung hat sich verbessert, aber die falsche Klassifizierung keines Bildes als „Gelb“ wurde nicht vollständig behoben. Wie immer gibt es Raum für Verbesserungen!

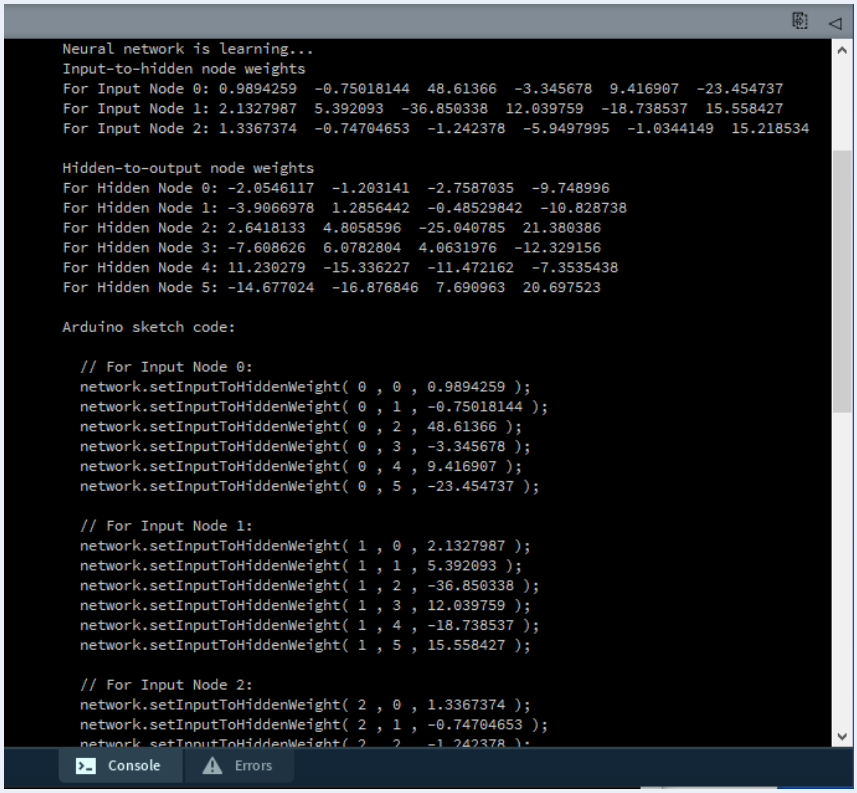


Bild 7. Mit learnfast.pde *in Processing können Gewichte schnell auf einem PC berechnet werden. Die Ausgabe der Textkonsole kann dann in* tlight_weights.ino *eingefügt werden.*

Tabelle 2. Mit TCS34725 erfasste RGB-Werte für die „unerwünschten" Farben.

Farbe	R	G	B
Dunkle Blende der Ampel	92	90	82
Weißer Rand	92	90	75
Blauer Hintergrund	73	93	89

Optimiertes Lernen

Beim Erlernen der „anderen" Farben im Arduino-Sketch dauert es etwa 18 Minuten, bis ein Mo Pro die gewünschten Klassifizierungen gelernt hat. Das lässt definitiv Raum für eine Optimierung des Prozesses. Da unser leistungsfähiger PC die Gewichte in weniger als einer Sekunde berechnen kann, wäre es sinnvoll, das Lernen dort durchzuführen und die Ergebnisse erst danach auf den Arduino zu übertragen. Mit diesen Gewichten haben wir auch die Möglichkeit, mehrere Mikrocontroller mit demselben „Wissen" zu programmieren. In der Annahme, dass die RGB-Sensoren in Bezug auf unsere Eingabe alle ähnlich funktionieren, sollte jeder Mikrocontroller das Ampelbild korrekt klassifizieren. Dies werden wir also als Nächstes in Angriff nehmen.

Wir kehren aber kurz zu Processing zurück, um das Projekt */arduino/learnfast/learnfast.pde* zu öffnen. Die gesamte Anwendung wird in der Funktion `setup()` ausgeführt. Das neuronale Netz wird mit denselben Eingangs-, versteckten und Ausgangsknoten konfiguriert, die auf dem Arduino-Board verwendet werden (3/6/4). In der Lernschleife (Zeile 36) werden die Werte verwendet, die vom Arduino mit dem RGB-Sensor und dem Sketch *tcsrgb-sensor.ino* erfasst wurden. Wenn der Code ausgeführt wird, gibt er Text auf seiner Konsole aus. Der letzte Abschnitt enthält den Code, der alle Gewichte (Eingang zu versteckten Knoten und versteckte Knoten zu Ausgang) konfiguriert (**Bild 7**). Kopieren Sie einfach den generierten Code, beginnend bei `// For Input Node =0` bis zum Ende der Textausgabe.

Zurück in der Arduino-IDE können wir nun den Sketch */arduino/tlight_weights/tlight_weights.ino* öffnen. Dies ist derselbe wie der Sketch *tlight_detect.ino*, aber anstatt das neuronale Netz zu lehren, werden die Gewichte vorprogrammiert. Dies geschieht in Zeile 51 mit der Funktion `importWeights()`. Fügen Sie einfach den Code aus der *learnfast.pde*-Ausgabe in die `importWeights()`-Funktion in Zeile 86 in *tlight_weights.ino* ein. Nach der Programmierung des Arduino-Boards sollte es die Ampelfarben wie zuvor genau erkennen. Tatsächlich können wir jetzt, da wir diesen optimierten Lernprozess haben, sogar denselben *tlight_weights.ino-Sketch* in einen Arduino Uno programmieren. Schließen Sie einfach den RGB-Sensor an das Board an, öffnen Sie den seriellen Monitor und beobachten Sie, wie dies genau so gut funktioniert wie auf einem Arduino Mo Pro oder Due. Zum Vergleich können Sie den digitalen Pin 9 überwachen, um zu sehen, wie lange es dauert, bis die Methode `calculateOutput()` die Berechnungen durchführt

Was kommt als Nächstes?

Und wie geht es jetzt weiter? Mehr eingebettete Systeme? Nun, wir haben ein funktionierendes neuronales MLP-Netzwerk, das sowohl auf einem PC als auch auf einem Mikrocontroller funktioniert. Außerdem verfügen wir über einen optimierten Lernprozess, um die Gewichte zu erzeugen, die in Mikrocontroller-Anwendungen benötigt werden. Sie könnten versuchen, diese MLP auf andere Aufgaben anzuwenden, die zu schwierig sind, um sie mit `if-else`-Anweisungen und festen Grenzwerten zu lösen. Vielleicht ist es sogar möglich, ein einfaches Spracherkennungsprojekt zu implementieren, um ein paar Wörter zu erkennen. Sie könnten auch das Folgende erforschen:

- Die Neural-Klasse verwendet den Datentyp `Double`. Kann sie mit `Float` beschleunigt werden und trotzdem die Genauigkeit beibehalten? Wieviel schneller kann sie laufen?
- Die Sigmoid-Funktion verwendet die mathematische Funktion `exp()` und berechnet den Exponentialwert, der auf das übertragene Argument bezogen wird. Kann die Aktivierungsfunktion durch eine Näherung vereinfacht werden und trotzdem eine genaue Klassifizierung liefern?
- Wenn Sie einen Versuch zur Spracherkennung durchführen: Wie würden Sie eine Sprachprobe vorbereiten, um sie dieser MLP-Implementierung zu

präsentieren?

- Wie wäre es mit einigen bedeutenden Veränderungen? Wie würden Sie eine zweite Schicht mit versteckten Knoten implementieren? Können Sie eine andere Aktivierungsfunktion implementieren?

In dieser Reihe von Artikeln haben wir eine Menge Themen behandelt. Wir haben die Oberfläche der frühen Forschung zu künstlichen Neuronen angekratzt. Danach haben wir untersucht, wie MLPs mit Hilfe von Backpropagation lernen, und wir haben ihre Funktionsweise unter Verwendung der leistungsstarken Verarbeitungsmöglichkeiten eines PCs untersucht. Anschließend haben wir den MLP als Beispiel für Edge ML auf einen Mikrocontroller transferiert. Wenn Sie diese Beispiele weiterentwickeln möchten, können Sie Ihre Ergebnisse gerne mit uns hier bei Elektor teilen.

210160-D-02

Haben Sie Fragen zu eingebetteten Neuronen?

Haben Sie Fragen oder Kommentare zu eingebetteten Neuronen? Dann schreiben Sie bitte dem Autor eine E-Mail an: stuart.cording@elektor.com.

Ein Beitrag von

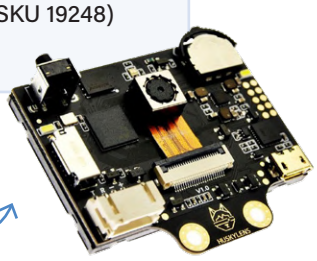
Idee, Text und Screendumps: **Stuart Cording**
Redaktion: **Jens Nickel, C. J. Abate**
Illustrationen: **Patrick Wielders**
Übersetzung: **Textmaster**
Layout: **Harmen Heida**

Bessere Landwirdschaft

Die Landwirtschaft hat sich schon immer auf die Natur und die Jahreszeiten verlassen, um den besten Zeitpunkt zum Ernten und Säen zu bestimmen. Die Tradition diktierte schon immer den optimalen Zeitpunkt für die Aussaat, um zum Beispiel von der Monsunzeit zu profitieren. Doch mit den sich ändernden Klimabedingungen haben die Ernteerträge gelitten. All dies ändert sich dank der KI. Indische Landwirte, die an einem Pilotprojekt teilnahmen, planten ihre Erdnussernte Ende Juni, drei Wochen später als üblich. Die Microsoft Cortana Intelligence Suite lieferte diese Anleitung. Dank historischer Klimadaten führten die Empfehlungen, die die Landwirte erhielten, zu einem durchschnittlich 30% höheren Ertrag [6].

PASSENDE PRODUKTE

- B. van Dam, Artificial Intelligence (E-Buch) (SKU 18090)
  www.elektor.de/18090
- Google AIY Vision Kit für Raspberry Pi (SKU 19365)
  www.elektor.de/19365
- HuskyLens AI Camera mit Silikongehäuse (SKU 19248)
  www.elektor.de/19248



WEBLINKS

[1] M. Patrick, „ML at the Edge: a Practical Example“, codemotion, September 2020: https://bit.ly/2ZQYipU
[2] RGB-Farbensor mit IR-Filter und weißer LED - TCS34725: http://bit.ly/2NKFS7T
[3] GitHub-Repository - simple-neural-network: https://bit.ly/2ZHLv9p
[4] GitHub-Repository - Adafruit_TCS34725: http://bit.ly/37RJg81
[5] „Writing a Library for Arduino“, Arduino: http://bit.ly/3aWeNaP
[6] „Digital Agriculture: Farmers in India are using AI to increase crop yields“, Microsoft News Center, November 2017: http://bit.ly/2PacsQZ

Magnetische Levitation

auf die sehr einfache Art

Die dritte und kompakteste Version

Von **Peter Neufeld**

Werfen wir einen Blick auf eine einfache analoge Schaltung zum Schwebenlassen kleiner Objekte, die Gegenstand vieler interessanter Diskussionen und Experimente ist. Die Hardware ähnelt dem Entwurf, den wir bereits in Elektor veröffentlicht haben, aber mit noch weniger Bauteilen. Die Elektronik passt jetzt sogar in das Gehäuse eines modifizierten Industrierelais!

Ich habe auf Elektor Labs einiges über meine Experimente mit Magnetschwebe-Schaltungen geschrieben. Die ersten beiden Versionen wurden online und in Elektor veröffentlicht: Die eine war eine vollständig analoge Lösung mit einem LM311-Komparator [1], die zweite war mit einem ESP32-basierten Mikrocontroller-Modul digitalisiert [2]. Im Folgenden werde ich diese Artikel als Teil 1 und Teil 2 bezeichnen. Während einiger hilfreicher Diskussionen über Magnetschwebetechniken mit dem Elektor-Ingenieur Luc Lemmens kamen uns ein paar Gedanken, die ich weiterverfolgte, um die schon entwickelten Projekte zu verbessern und zu optimieren. Einige dieser Änderungen wurden bereits in den beiden veröffentlichten Schaltungen umgesetzt, aber die Diskussionen gaben auch Anlass zu weiteren Optimierungen. Das Ergebnis war eine weitere starke Vereinfachung der „alten" analogen Schaltung, die es mir ermöglicht, kleine (magnetische) Objekte schweben zu lassen, und die wir in diesem Artikel besprechen wollen. Dieses ist also der dritte und - zumindest vorläufig - letzte Teil über meine Magnetschwebeprojekte. Auch wenn die Schaltung im Vergleich zu den anderen beiden Projekten noch einfacher ist, funktioniert sie sehr gut.

Das Prinzip, kurz und bündig

Die Idee hinter allen drei Magnetschwebeprojekten ist, dass nur ein kleiner Teil der Kraft, die benötigt wird, um ein Objekt schweben zu

lassen, von einem elektromagnetischen Feld einer modifizierten Relaisspule aufgebracht werden muss. Die eigentliche Schwerarbeit wird durch die dauerhafte Kraft zwischen einem Permanentmagneten im schwebenden Objekt und dem Eisenkern der Magnetspule geleistet. Ein Hallsensor misst die Stärke des gesamten Magnetfelds (das heißt, die Summe aus statischem und elektromagnetischem Feld). Der Spannungspegel an seinem Ausgang ist auch ein Maß für den Abstand zwischen der Spule und dem schwebenden Objekt und wird als Rückkopplungssignal zur Steuerung des Stroms durch den Elektromagneten verwendet, der wiederum den Abstand zwischen dem Kern und dem schwebenden Objekt steuert.

Ein voreingestelltes Potentiometer dient der Feinabstimmung des Steuerkreises und der Anpassung an das magnetische Objekt, das unter der Spule schweben soll. Wenn Sie mit diesen Schaltungen experimentieren, kann sich die korrekte Einstellung als echte Herausforderung erweisen. Im Notfall, sollte es partout nicht klappen, liefert der erste Artikel [1] nützliche Tipps und Tricks. Levitation ist möglich, aber natürlich gibt es Grenzen für die Größe und das Gewicht der Gegenstände, die mit der begrenzten Leistung dieser Schaltung angehoben werden können.

Vereinfachte Schaltung

Der Schaltplan dieser sehr einfachen Version in **Bild 1** ist deutlich eine abgespeckte Version des ersten Teils. Auch hier sorgt ein LM311-Komparator [3] für den relativ kleinen Steuerstrom durch die Spule, ohne dass ein zusätzlicher externer Treibertransistor nötig wäre. Wie Sie vielleicht wissen, besitzt der LM311 einen Open-Collector-Ausgang, der eine kleine Last treiben kann. Dies vereinfacht die Schaltung beträchtlich im Vergleich zu der ersten vollständig analogen Lösung in Teil 1, wo ein BC550 NPN-Transistor am Komparatorausgang den Elektromagneten versorgt.

Um die Anzahl der Bauteile noch weiter zu reduzieren und noch mehr Platz zu sparen, kann der aus zwei Festwiderständen und einem Mehrgang-Trimmpoti bestehende Spannungsteiler für den Referenzpegel am invertierenden Eingang des Komparators ebenfalls geändert werden. Es reichen auch ein Festwiderstand und ein Eingang-Trimmpoti (P1 und R4), die sogar in das Gehäuse des Relais passen, wie in **Bild 2** zu sehen ist. Der Wert ist nicht kritisch, alles im Bereich von 1 kΩ bis 1 MΩ funktioniert.

Die hier verwendeten Bauteile sind weit verbreitet, leicht erhältlich und erschwinglich. In Teil 1 wurde gesagt, dass billigere und leicht erhältliche Hall-Sensoren wie der SS49E nicht verwendet werden können und dass nur die Sensoren A1302 oder A1308 von Allegro Microsystems den Zweck erfüllen würden. In Teil 2 konnten wir dann erfreulicherweise berichten, dass diese Annahme falsch war. Weitere Tests haben nämlich gezeigt, dass der Typ und die Marke von IC2 gar nicht so entscheidend sind wie zunächst angenommen, was die Suche nach den benötigten Bauteilen für diese Projekte erheblich erleichtert. Die Allegro-Sensoren sind nämlich teurer und schwer zu beschaffen.

Das richtige Relais: Beschaffung und Modifizierung

Der Elektromagnet (L1) besteht aus der Magnetspule eines handelsüblichen 5-V-Relais, genauer gesagt, eines Finder-Relais mit der Typennummer 40.52.7.005. Dieser Typ ist leicht erhältlich und wesentlich einfacher zu modifizieren als die zuvor verwendeten 5-V-Relais. Die Fotos in **Bild 3** zeigen den gesamten Vorgang der Modifizierung.

Das durchsichtige Gehäuse eines Finder-Relais ist oft mit seinem Sockel verklebt. Es lässt sich jedoch leicht öffnen, indem man die Klebeverbindung mit einem Fön oder einer Heißluft-Lötstation bei mäßiger Temperatur erhitzt. Ein Cuttermesser, ebenfalls leicht erwärmt, kann nützlich sein, um das Gehäuse beschädigungsfrei zu öffnen. Bleibt die Kappe intakt, kann man sie als Gehäuse für die gesamte Elektronik dieses Schwebegeräts wiederverwenden.

Genau wie bei den in den früheren Veröffentlichungen verwendeten Relais muss der Kern der Magnetspule von seiner U-Form in eine I-Form geändert werden, um einen magnetischen Kurzschluss zu vermeiden. Dies ist recht einfach, da der runde Kern und die rechteckige flache Verlängerung nur zusammengefügt sind. Diese Fügeverbindung kann mit einer kleinen Trennscheibe, einem Fräskopf oder einer Bohrmaschine entfernt werden.

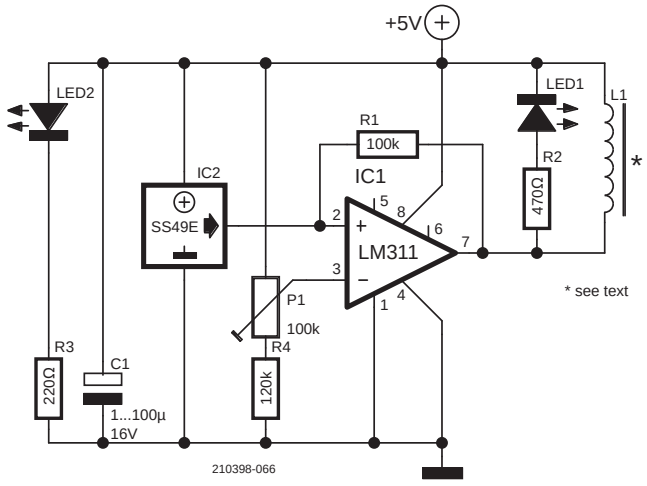


Bild 1. Das Schaltbild dieser Levitationsversion.

Bild 2. Ein normales Trimmpoti plus Vorwiderstand passt ebenfalls in das Gehäuse.

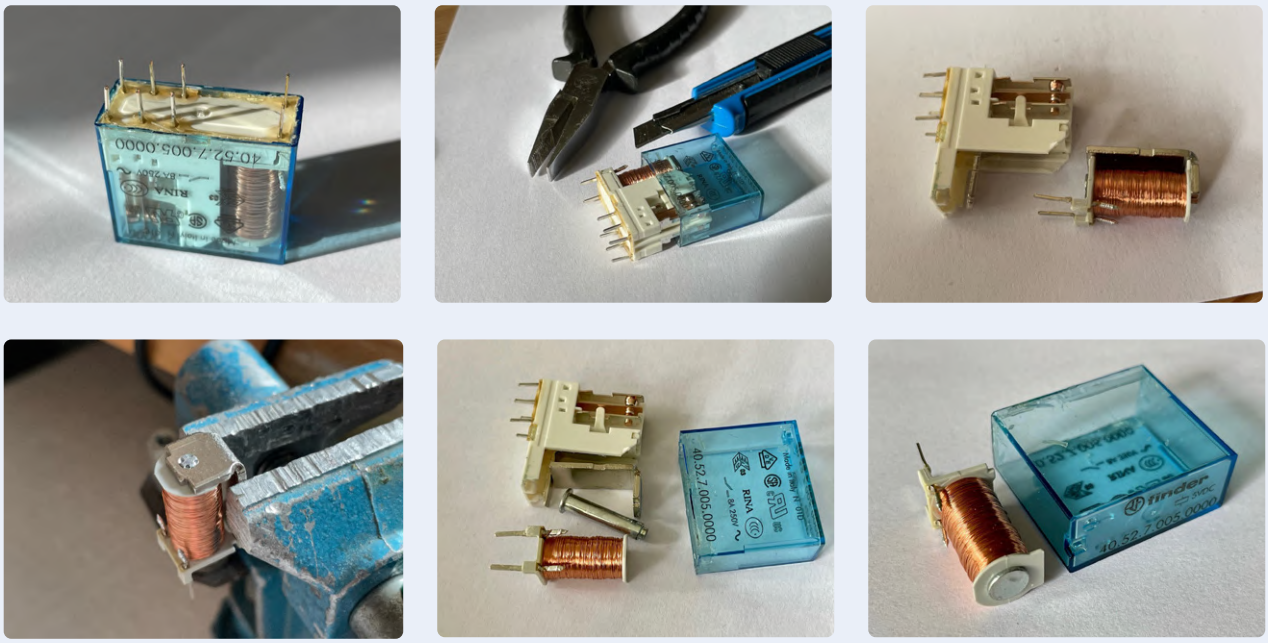


Bild 3. Demontage des Relais und Vorbereitung der Magnetspule.

Schließlich kann der Eisenkern herausgenommen - wenn er nicht von selbst herausfällt - und umgedreht werden, so dass man den Hall-Sensor auf die etwas größere und flache Polfläche kleben kann (**Bild 4**). Zumindest der Elektromagnet und der Hallsensor sollten im Gehäuse des Relais montiert werden, damit sie gut gegen den Magneten geschützt sind.

Die Stromaufnahme bei einem unter der Spule schwebenden magnetischen Objekt beträgt nur 50 mA bei einer Betriebsspannung von 5 V und bis zu maximal 90 mA, wenn die Spule permanent mit Strom versorgt wird, zum Beispiel, wenn sich kein Magnet in der Nähe des Kerns befindet.

Durch die Modifikation des Finder-Relais ist ein sehr guter Elektromagnet entstanden, der mit der Kappe sogar einen guten mechanischen Schutz der gesamten Schaltung gegen die Permanentmagnete bietet, die auf den Kern und/oder den Hall-Sensor treffen. Wenn man wie ich wirklich alles an Elektronik in das ehemalige Relaisgehäuse einbauen will, ist das mit vertretbarem mechanischem Aufwand machbar, auch wenn es zwar einige Zeit dauern dürfte, bis man herausgefunden hat, wie man die Bauteile in diesem relativ kleinen Volumen anordnen muss. Aber natürlich lässt sich die Schaltung auch sehr einfach auf einem Breadboard oder einer Lochrasterplatine aufbauen (siehe **Bild 5**). Dabei werden nur vier Drähte benötigt, um die Kombination aus Spule, LED, Widerstand und Hallsensor mit dem Rest der Schaltung zu verbinden. Ein 20 cm langes Stück Flachbandkabel war mir für diesen Zweck beim Prototyp sehr hilfreich.

Guter Ingenieur oder mutiger, aber leichtsinniger Tüftler?

Luc vom Elektor-Team hat kritisch, aber völlig richtig analysiert, dass der LM311 am Ausgang 50 mA aufnehmen kann. Aber was ist mit 90 mA? Gut, das Argument ist nicht von der Hand zu weisen, aber mein Ziel war es ja, die Schaltung so einfach wie möglich zu halten und die Anzahl der Bauteile auf ein absolutes Minimum zu reduzieren. Und so wollte der Tüftler in mir nicht einfach einen zusätzlichen Treibertransistor verwenden, „nur" um zu verhindern, dass meine Schaltung den Hitzetod stirbt.

Ein genauerer Blick auf die Innenschaltung des LM311 (**Bild 6**) und die Diagramme im Datenblatt ergab, dass der Chip über eine interne Schutzschaltung verfügt, die den Ausgangsstrom auch im Falle eines Ausgangskurzschlusses auf ein unbedenkliches Maß begrenzt und eine maximale Verlustleistung von 350 mW bei 5 V Versorgungsspannung gewährleistet. Das macht den Ausgang fast unsterblich - oder zumindest unempfindlich gegenüber einer niederohmigen Last. Zum Test habe ich meine Schaltung zwei Tage lang eingeschaltet gelassen, und sie hat, wie ich erwartet und gehofft hatte, überlebt!

Ok, gehen wir dennoch einen Schritt zurück. Ich halte mich an Geschwindigkeitsbegrenzungen, ich zahle meine Steuern und ich lese und befolge Datenblätter - und ich kenne viele gute Gründe, die sicheren Betriebsspezifikationen von Halbleitern einzuhalten. Auch wenn integrierte Schaltkreise über alle möglichen Schutzschaltungen verfügen, ist es eine schlechte Praxis, sie bis an ihre Grenzen zu belasten, denn das dürfte die Lebensdauer der Chips mit Sicherheit verkürzen. Man kann zwar argumentieren, dass der Chip auch in diesem Fall noch lange genug hält und dass der Ersatz eines LM311 kein Vermögen kostet, aber bei dieser kompakten Bauweise ist es doch recht mühsam, die Schaltung zu reparieren.

Lieber auf Nummer sicher gehen! Es war eine Herausforderung für mich, auszuprobieren, ob die Schaltung auch mit einem geringeren Ausgangsstrom funktionieren kann. Eine einfache und offensichtliche Möglichkeit, dies zu erreichen, besteht darin, das 5-V-Relais Finder 40.52.7.005.0000 mit 50 Ω Spulenwiderstand durch ein 6-V-Relais mit 75 Ω (Finder 40.52.7.006.0000) zu ersetzen, aber ich hatte kein solches Relais zur Hand.

Also wandte ich mich der nächstbesten Lösung zu: Ich behielt die

5-V-Versorgung bei, reduzierte aber die Spannung an der Spule, indem ich zwei 1N4148-Dioden in Reihe zwischen die Stromversorgung und die Magnetspule schaltete (**Bild 7**). Ein einzelner 22-Ω-Widerstand statt der Dioden hätte übrigens die gleiche Wirkung. Dadurch wurde der Spulenstrom ohne Nutzlast auf 50 mA und mit Nutzlast auf durchschnittlich 39 mA reduziert. Und auch das funktionierte gut und der Komparator bleibt innerhalb seiner elektrischen Spezifikationen. Leider verringert sich durch diese Maßnahme auch der Abstand zwischen dem Elektromagneten und dem schwebenden Objekt. Noch schlimmer ist, dass das Steuersystem weniger Spielraum hat, um Störungen zu kompensieren. Mit anderen Worten: Bei dieser geringeren Leistung dürfte das schwebende Objekt herunterfallen oder gegen den Elektromagneten stoßen, wenn es auch nur ein klein wenig vom eingestellten Abstand zum Kern abweicht. Die Schaltung ist weniger in der Lage, einen solchen Fehler zu korrigieren.

Irgendwelche Ideen?

Alles in allem gefällt mir diese kleinere Version besser als die erste analoge Version aus Teil 1, die ich auf Lochraster aufgebaut hatte. Sie sieht schöner aus, und die Elektronik ist durch das Gehäuse des Relais besser geschützt. Die zweite Version - mit einem ESP32-basierten M5Stack Atom - zeigte, dass der analoge Komparator durch einen Mikrocontroller und einen einfachen Algorithmus ersetzt werden kann. Meiner Meinung nach haben die verschiedenen Varianten dieses

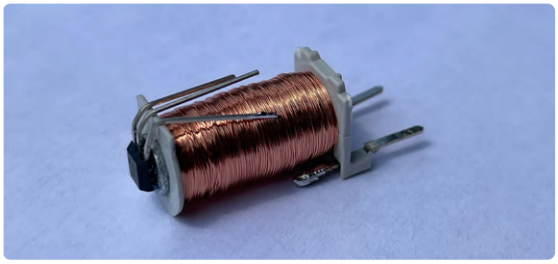


Bild 4. Der Hallsensor, der auf den Eisenkern der Spule geklebt ist.

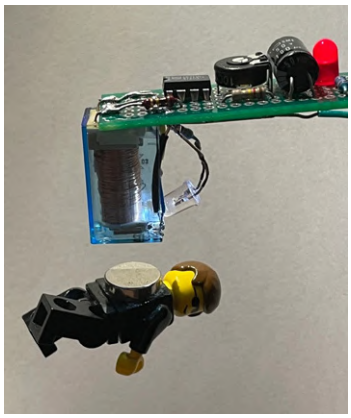


Bild 5. Diese Schaltung kann auch auf einem Stück Lochraster aufgebaut werden.

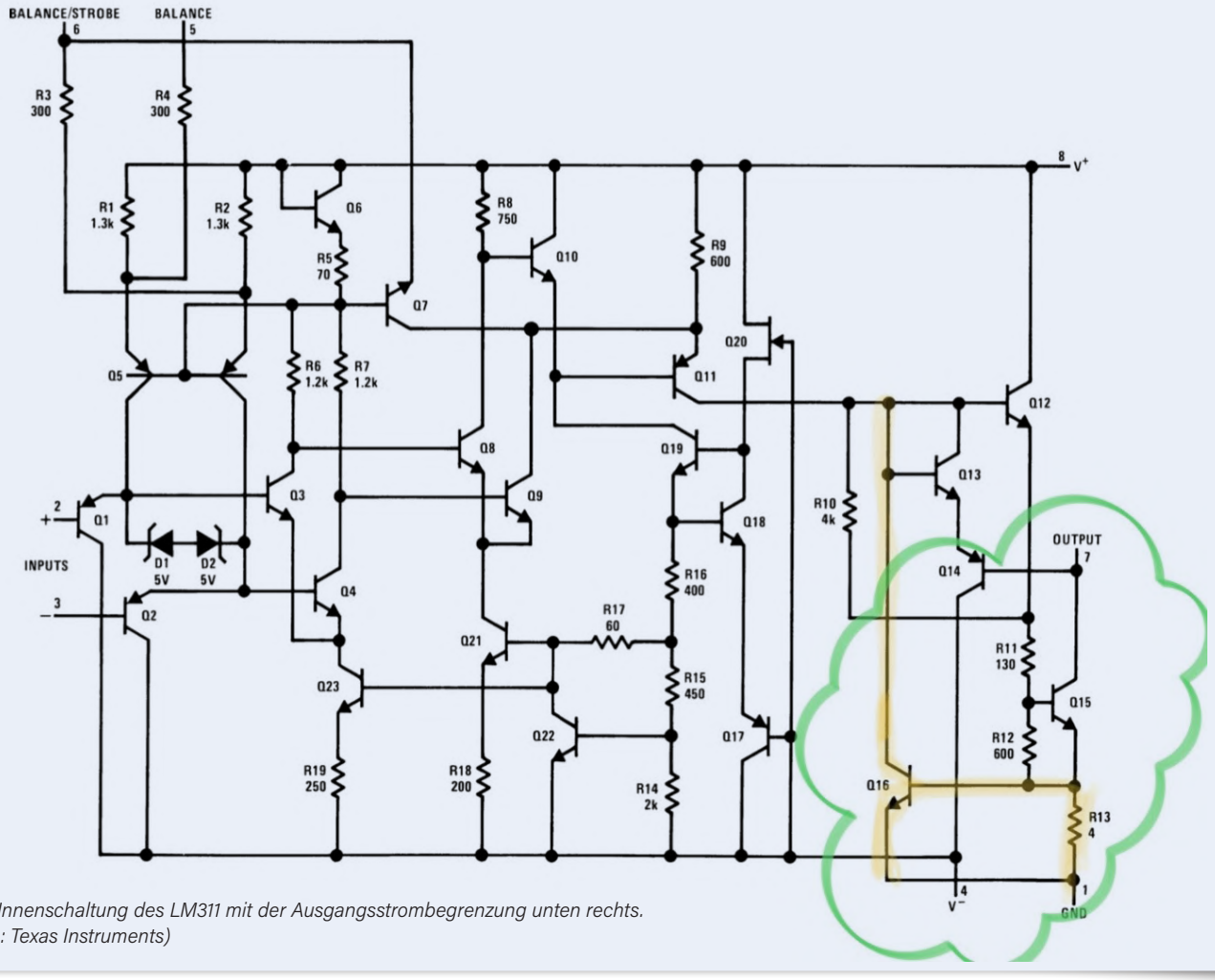


Bild 6. Innenschaltung des LM311 mit der Ausgangsstrombegrenzung unten rechts. (Quelle: Texas Instruments)

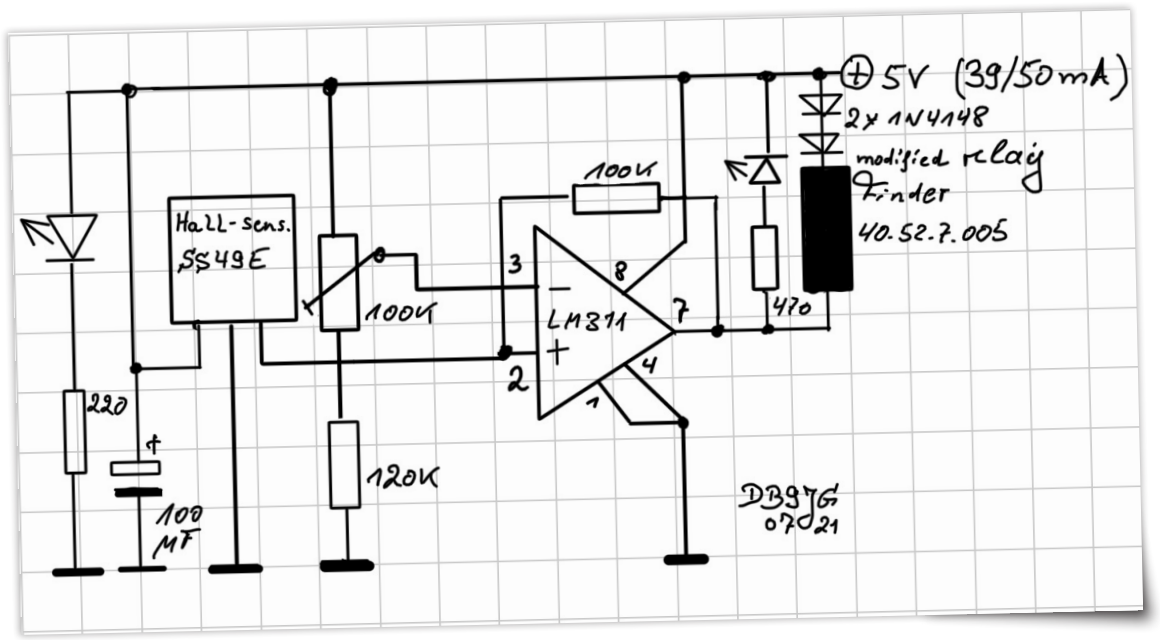


Bild 7. Die Magnetschwebeschaltung mit zwei 1N4148-Dioden zur Reduzierung des Spulenstroms.

kleinen Projekts mit seinem eigentlich sehr allgemeinen regelungstechnischen Problem gezeigt, dass ein analoger Aufbau auch in der heutigen, von Mikrocontrollern dominierten Zeit noch seine Vorteile hat! Wie ich zu Beginn dieses Artikels angekündigt habe, ist dies die letzte und endgültige Version meiner Magnetschwebeprojekte, es sei denn... Vielleicht haben Sie Ideen oder Vorschläge, wie man das Design noch weiter verbessern kann?

210398-02

Haben Sie Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Wenden Sie sich bitte an die Redaktion unter luc.lemmens@elektor.com.

Ein Beitrag von

Gestaltung und Text: **Peter Neufeld**
Redaktion: **Luc Lemmens**
Illustrationen: **Peter Neufeld, Patrick Wielders, Luc Lemmens**
Übersetzung: **Rolf Gerstendorf**
Layout: **Harmen Heida**

STÜCKLISTE

Widerstände:
R1 = 100 k
R2 = 470 Ω
R3 = 220 Ω
R4 = 120 k
P1 = 100 k Trimmpoti

Kondensatoren:
C1 = 1...100 µ, 16 V, radial

Halbleiter:
LED1,LED2 = LED grün
IC1 = Hall-Sensor SS49E
IC2 = Komparator LM311

Außerdem:
L1 = 5V-Relais (Finder 40.52.7.005.0000, siehe Text)

PASSENDE PRODUKTE

- **Velleman Helfende Hand mit Lupe, LED-Lampe und Lötkolbenhelter (SKU 19864)**
  www.elektor.de/19864
- **Velleman VTSS220 Lötstation mit Temperaturregelung (SKU 19865)**
  www.elektor.de/19865

WEBLINKS

[1] „Magnetische Levitation auf die einfache Art", ElektorMag 7-8/2021: https://www.elektormagazine.de/200311-02
[2] „Magnetische Levitation die digitale Art", ElektorMag 9-10/2021: https://www.elektormagazine.de/200278-02
[3] Datenblatt zum Komparator LM311: https://www.ti.com/lit/ds/symlink/lm311-n.pdf

SPS-Programmierung mit dem Raspberry Pi und dem OpenPLC-Projekt

Visualisierung von SPS-Programmen mit AdvancedHMI

Von **Josef Bernhardt**

In industriellen Steuerungssystemen, aber auch in der Heimautomatisierung ist es üblich, den aktuellen Prozess über sogenannte Monitore zu steuern und zu überwachen. In diesem dem neusten Buch von Josef Bernardt entnommenen Kapitel über SPS-Programmierung lernen Sie die Software AdvancedHMI kennen. Sie greifen auf die Variablen der SPS zu und versuchen, diese zu visualisieren. Es ist auch möglich, auf den Prozess zuzugreifen und damit Schalter zu simulieren.

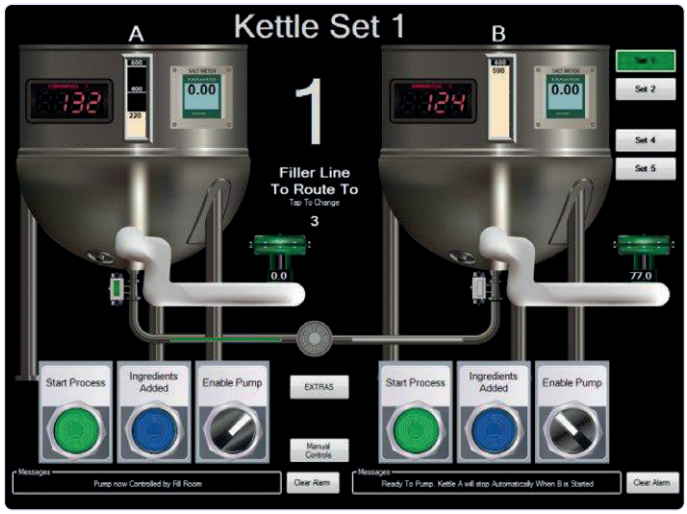


Bild 1. AdvancedHMI-Beispiel.

AdvancedHMI ist ein .NET-Programm, das mit der Visual Studio Community Software von Microsoft in Visual Basic geschrieben wurde. Wenn Sie das Framework Mono auf Ihrem Raspberry Pi installieren, können Sie die Software auch auf dieser Plattform ausführen und erhalten Zugriff auf das laufende SPS-Programm. Sowohl die AdvancedHMI- als auch die Software Visual Studio Community können kostenlos heruntergeladen werden. Ein Beispiel für AdvancedHMI in Aktion ist in **Bild 1** dargestellt.
Laden Sie zunächst die Software von der Website des Herstellers herunter und schreiben Sie ein Testprogramm für Ihre Raspberry-Pi-SPS. Gehen Sie dazu auf die Website des Herstellers von AdvanceHMI und legen Sie das „AdvancedHMI Base Package“ in den Warenkorb, wie in **Bild 2** dargestellt. Danach klicken Sie auf das Feld *Checkout*.

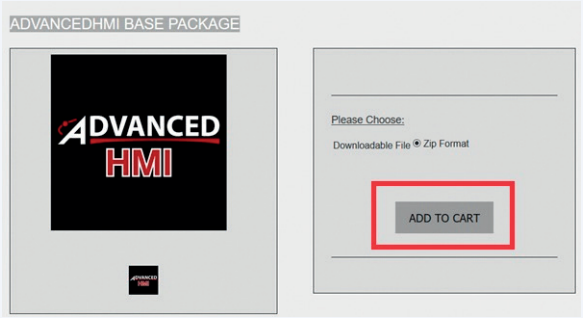


Bild 2. Download des AdvancedHMI-Pakets.

Anmerkung der Redaktion: Dieser Artikel ist ein Auszug aus dem 194-seitigen Buch *SPS-Programmierung mit dem Raspberry Pi und dem OpenPLC-Projekt - ModbusRTU- und ModbusTCP-Beispiele mit dem Arduino Uno und dem ESP8266*. Das Beispielkapitel wurde formatiert und leicht bearbeitet, um den redaktionellen Standards und dem Seitenlayout von Elektor zu entsprechen. Da es sich um einen Auszug aus einer größeren Publikation handelt, können sich einige Begriffe in diesem Artikel auf Diskussionen an anderer Stelle im Buch beziehen. Der Autor und die Redaktion haben ihr Bestes getan, um solche Fälle auszuschließen, und sind gerne bereit, bei Fragen zu helfen. Kontaktinformationen finden Sie im Kasten Fragen oder Kommentare?

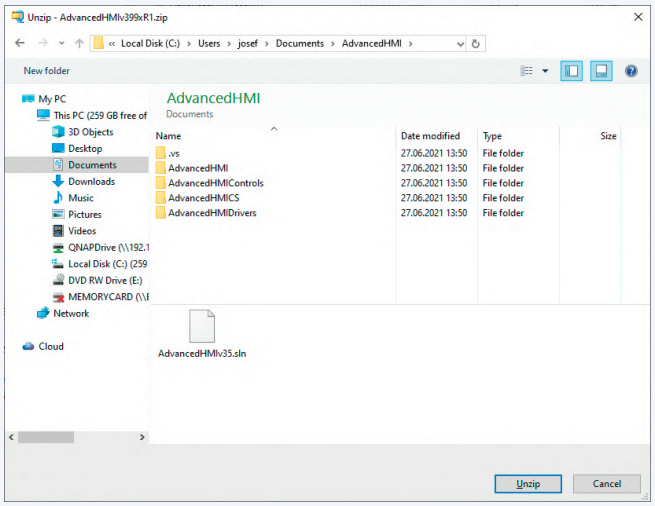


Bild 3. AdvancedHMI entpackt.

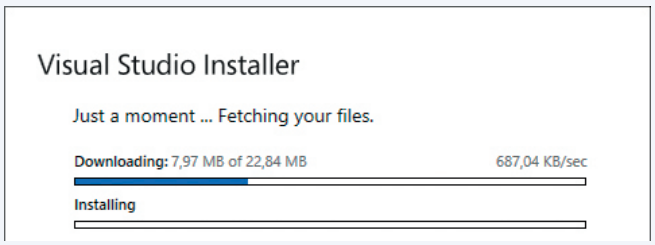


Bild 5. Der Visual Studio-Installer bei der Arbeit.

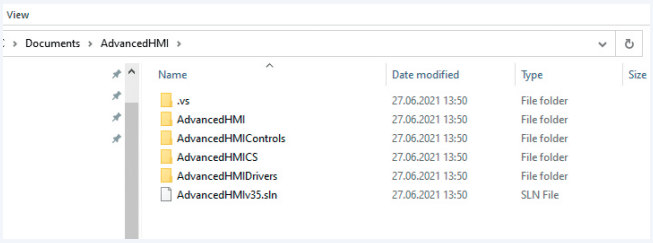


Bild 4. AdvancedHMI-Projektverzeichnis.

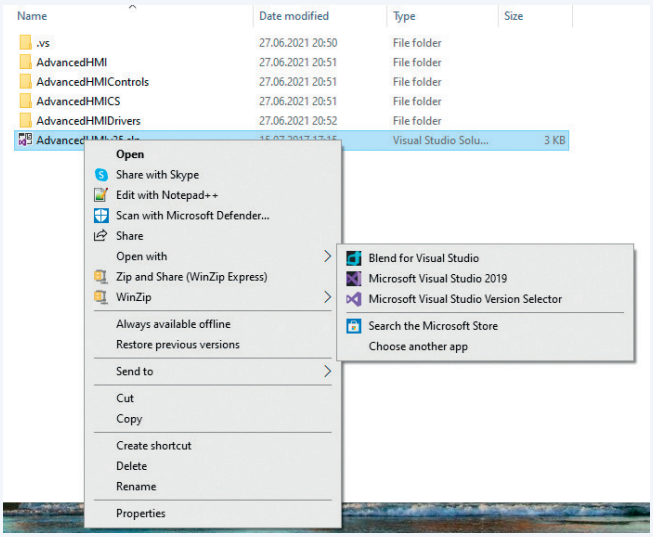


Bild 6. Öffnen Sie Ihr AdvancedHMI-Projekt.

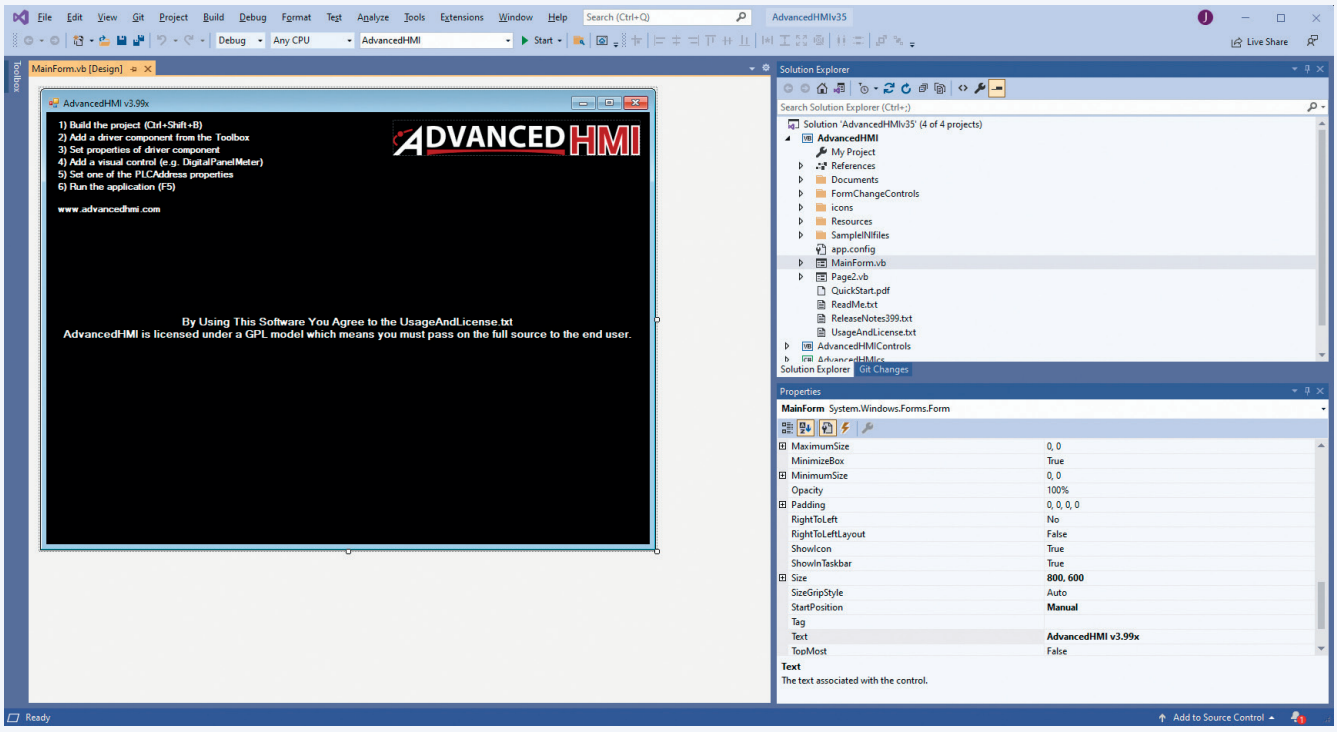


Bild 7. Benutzeroberfläche von Visual Studio.

Vorwort zum Buch

Dieses Buch soll dem Leser eine praktische Einführung bieten, um den Raspberry Pi als SPS für eigene Projekte einsetzen zu können. Das Projekt wurde durch die Programmierer Edouard Tisserant und Mario de Sousa ermöglicht. Diese haben nach der Einführung der IEC-Norm 61131-3 im Jahr 2003 das „Matiec-Projekt" begonnen. Damit war es möglich, die in der Norm vorgestellten Programmiersprachen in C-Programme zu übersetzen. Später, als der Raspberry Pi immer populärer wurde, begann Thiago Alves mit dem „openplcproject". Er erweiterte den Editor aus dem „Beremiz-Projekt" und schrieb eine Laufzeitbibliothek sowie eine Weboberfläche für den Raspberry Pi und den PC. Damit wurde es möglich, Programme auf dem PC zu schreiben und sie auf dem Raspberry Pi zu installieren.
Viele Benutzer des Raspberry Pi haben jetzt die Möglichkeit, eigene Steuerungen und Regelungen auf ihrer Hardware zu verwirklichen. Auch für Schulungszwecke ist die Hard- und Software wegen der Anlehnung an die IEC-Norm hervorragend geeignet.
Einsteiger und Profis erfahren hier auch alles über die Installation und die Programmierung in den fünf Programmiersprachen, um eigene Steuerungen aufzubauen. In einem späteren Kapitel wird die Visualisierung mit AdvancedHMI behandelt, um Prozesse auf dem Bildschirm darzustellen.
Schaltungen mit Arduino und ESP8266, die für Modbus notwendig sind, werden ebenfalls erläutert.

Nach dem Registrierungsvorgang kann das Programm kostenlos heruntergeladen und in ein Verzeichnis entpackt werden (**Bild 3**). Speichern Sie auch die ZIP-Datei zur Wiederverwertung.
Erstellen Sie nun einen Ordner *AdvancedHMI-SPS-Test* und kopieren Sie den Inhalt der Datei *AdvancedHMIv399xR1.ZIP* in dieses Verzeichnis (**Bild 4**). Um das Projekt zu öffnen, benötigen Sie eine Entwicklungsumgebung. Installieren Sie daher die Software *Visual Studio Community* von der Microsoft-Website, wie in **Bild 5** dargestellt.
Nachdem der Installation können wir das erste Demoprojekt starten. Mit einem rechten Mausklick öffnen wir die *AdvancedHMI* mit Microsoft Visual Studio 2019 (**Bild 6** und **Bild 7**). Mit einem Doppelklick auf *MainForm.vb* erhalten Sie die noch leere HMI-Ansicht des Demoprojekts (WinForm). Nun können Sie das Programm mit einem Klick auf *Starten* starten (**Bild 8**). Zu diesem Zeitpunkt ist das Projekt kompiliert und in dem in **Bild 9** gezeigten Verzeichnis gespeichert.
Diese Projektdateien werden benötigt, wenn Sie das Programm auf einem anderen Computer verwenden möchten. Sie können diese Dateien auch auf dem Raspberry Pi mit dem Mono-Framework ausführen.
Ziehen Sie nun die Komponente *ModbusTCPCom* aus der linken Toolbox (**Bild 10**) auf die Arbeitsfläche. Im Fenster *Eigenschaften* können Sie nun die IP-Adresse Ihres Raspberry Pi eingeben (**Bild 11**).
Um die Kommunikation zu testen, erstellen Sie ein neues SPS-Programm mit dem *Open PLC Editor* (**Bild 12**), laden es auf den Raspberry Pi und starten es. Es handelt sich um eine einfache Ein/Aus-Schaltung mit den Tasten *Ein* und *Aus* (**Bild 13**).
Interessant wird es bei der Taste *BTN_HMI* auf der Adresse *%QX2.0*. Diese Adresse liegt außerhalb des Adressbereichs der GPIOs des Raspberry Pi und ist für die HMI-Schnittstelle notwendig, um einen Tastendruck auszulösen. Mit den Timern *TON0* und *TOF0* können Sie eine blinkende LED-Schaltung für das *LEDBlink*-Programm realisieren.

Wechseln Sie nun zurück zu Ihrem HMI-Projekt, ziehen Sie einen Schalter (*MomentaryButton*) und drei Kontrollleuchten (*PilotLight*) aus der Toolbox und beschriften Sie diese wie in **Bild 14** zu sehen. Wählen Sie dazu das Objekt aus und ändern Sie den Text im Fenster *Eigenschaften* so wie in Bild vorgegeben.
Setzen Sie die Felder *PLCAdressClick* wie folgt:

- **BTN_HMI** ist Adresse 17, weil Sie *%QX2.0* für *BTN_HMI (8+8+1)* verwendet haben,
- **LED** kommt auf 1,
- **LEDModbus** kommt auf 2 und
- **LEDBlink** kommt auf 3

Der *MomentaryButton* kann unter *Eigenschaften OutputType* von einem Taster in einen Schalter umgewandelt werden, wie in **Bild 15** gezeigt. Mit diesen Eigenschaften haben Sie die richtigen Adressen für den Zugriff auf die Variablen Ihres SPS-Programms. Nun können Sie Ihr Programm starten und einen Test durchführen.
Die drei LEDs zeigen immer den gleichen Zustand wie die LEDs auf der Testplatine. Mit dem Schalter *BTN_HMI* können Sie LED 2 auf der Testplatine umschalten.
Tabelle 1 zeigt die Modbus-Adresszuordnung für den Raspberry Pi mit OpenPLCproject und AdvancedHMI. OpenPLC bietet auch einen separaten Adressraum für Speichervariablen mit Unterstützung für 16-, 32- und 64-Bit-Variablen. Dies ist in **Tabelle 2** zusammengefasst.
Für die Installation des Mono-Frameworks auf dem Raspberry Pi werden folgende Befehle benötigt:

```
sudo su
apt-get update
apt-get install mono-complete
apt-get install mono-vbnc
```

Function Code	Usage	PLC Address	Modbus Register	Register Size	Value Range	Access	Advanced HMI	Example
FC01 Read Coil	Digital Outputs	%QX0.0 - %QX99.7	0-799 (1-800)	1 Bit	0 or 1	Read / Write	"00001" – "00800"	LED
FC02 Discrete Input	Digital Inputs	%IX0.0 – %IX99.7	0-799 (1-800)	1 Bit	0 or 1	Read Only	"100001" – "100800"	BTN
FC04 Input Register	Analog Inputs	%IW0 – %IW99	0-1023 (1-1024)	16 Bit	0-65535	Read Only	"30001" – "31024"	ADC
FC03 Holding Register	Analog Outputs	%QW0 – %QW99	0-1023 (1-1024)	16 Bit	0-65535	Read / Write	"40001" – "41024"	DAC

Tabelle 1

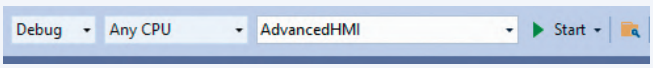


Bild 8. Menüleiste im Visual Studio.

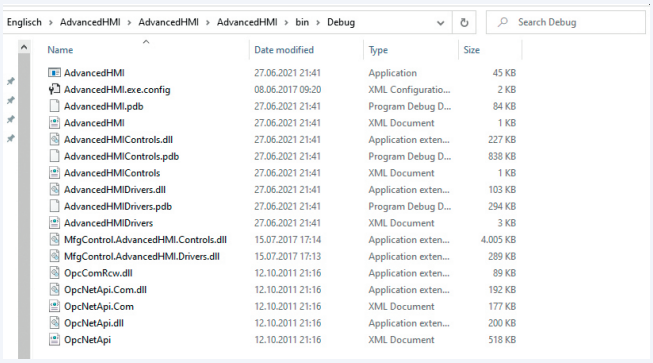


Bild 9. Debug-Verzeichnis von AdvancedHMI.

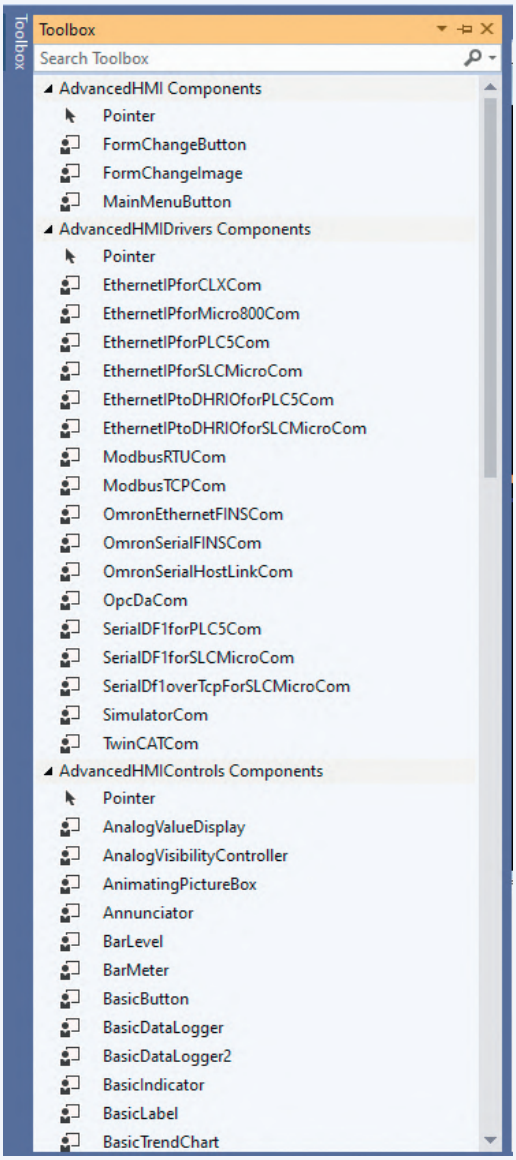


Bild 10. Die umfangreiche Toolbox des Visual Studios.

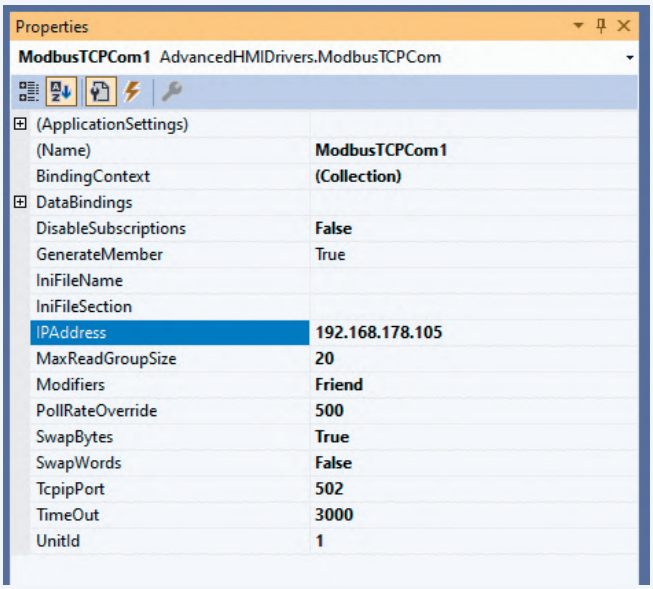


Bild 11. Modbus-Einstellungen (IP und Port).

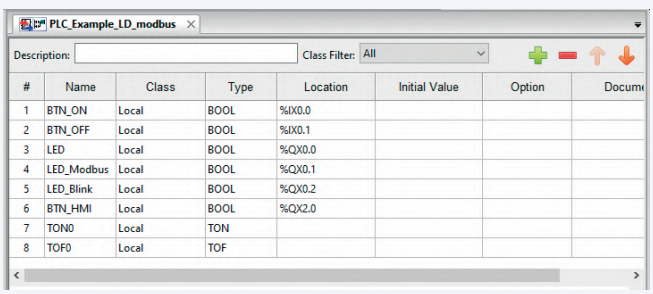


Bild 12. Variablenliste des PLC-Beispiels.

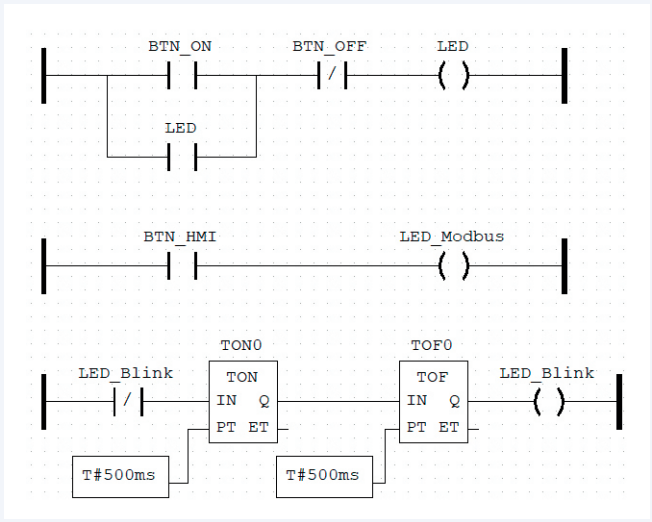


Bild 13. Beispielprogramm mit LEDs.

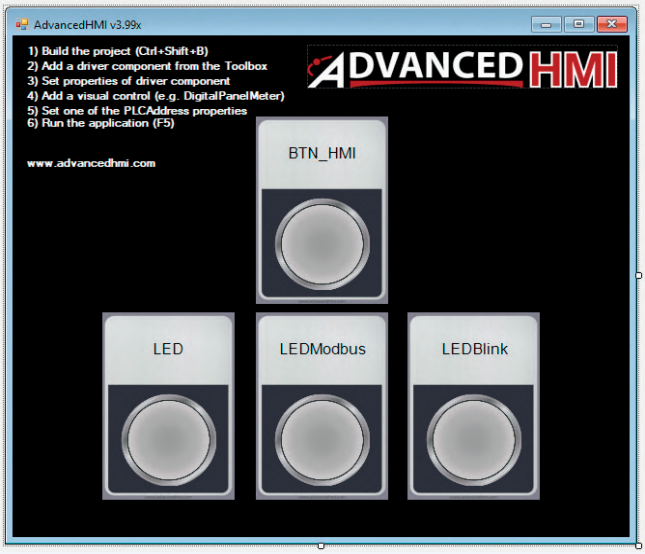


Bild 14. HMI-Programm.

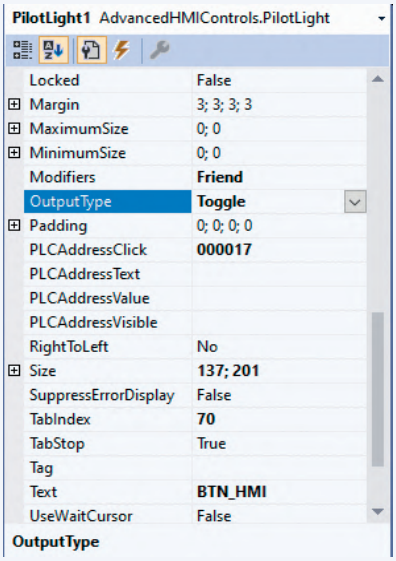


Bild 15. Einstellung der HMI-Adresse.

Nun können Sie ein Verzeichnis auf dem Raspberry Pi erstellen und *WinSCP* verwenden, um Ihr Projekt vom PC auf den Raspberry Pi zu kopieren und es dort zu testen.
Die .EXE-Beispielprogramme sind ist in dem Software-Bundle enthalten, das der Autor zur Unterstützung seines Buches [1] zusammengestellt hat. Die Software steht zum kostenlosen Download zur Verfügung. Sie finden Sie im *OpenPLC-Projekt.zip* im Unterverzeichnis

Software_SPS-Programmierung mit dem RPi und dem OpenPLC-Projekt\AdvancedHMI\AdvancedHMI_openplcproject\AdvancedHMI\bin\Debug

Speichern Sie die ZIP-Archivdatei lokal (128,94 MB) und entpacken Sie sie dann.

210642-02

Haben Sie Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Senden Sie eine E-Mail an den Autor unter josef@bernhardt.de oder an Elektor unter redaktion@elektor.de.

Ein Beitrag von

Text: **Josef Bernhardt**
Redaktion: **Jan Buiting**
Übersetzung: **Rolf Gerstendorf**
Layout: **Giel Dols**

PASSENDE PRODUKTE

- **Buch: J. Bernhard, SPS-Programmierung mit dem Raspberry Pi und dem OpenPLC-Projekt (SKU 19712)**
  www.elektor.de/19712
- **E-Buch: J. Bernhard, SPS-Programmierung mit dem Raspberry Pi und dem OpenPLC-Projekt (SKU 19713)**
  www.elektor.de/19713

Register Type	Usage	PLC Address Range	Modbus Address	Register Size	Value Range	Access	Advanced HMI
Holding Register	General 16Bit Register	%MW0 - %MW1023	1024..2047 (1025-2048)	16 Bit	0-65535	Read / Write	"41025" – "42048"
Holding Register	General 32Bit Register	%MD0 - %MD1023	2048..4095 (2049-4096)	32 Bit	0-4.294.967.295	Read / Write	"42049" – "44096"
Holding Register	General 64Bit Register	%ML0 - %ML1023	4096..8191 (4097-8192)	64 Bit	0-huge	Read / Write	"44097" – "48192"

Tabelle 2

WEBLINK

[1] Buch-Infoseite: www.elektor.de/sps-programmierung-mit-dem-raspberry-pi-und-dem-openplc-projekt

Aus dem Leben gegriffen

Einpacken und weg damit

Von Ilse Joostens (Belgien)

Wir leben in einer großen, bösen Welt, und normalerweise vergeht kein Tag, an dem wir uns nicht über die unbedeutendsten Dinge ärgern. Auf der langen Liste der kleinen und großen Ärgernisse, die uns auf die Palme bringen können, stehen Verpackungen ganz weit oben. Nehmen wir zum Beispiel die Blister-Verpackung. Diese verflixten Dinger muss man mit Scheren, Messern und anderen scharfen Werkzeugen traktieren, um sie zu öffnen. Mit etwas Pech schneidet man sich dabei nicht nur in die Hand am scharfen Plastik, sondern beschädigt bei all der Gewalt, ob nun von Kraftausdrücken begleitet oder nicht, auch das Produkt. Oder was halten Sie von Verpackungen, die sich zwar leicht auspacken lassen, bei denen es aber absolut unmöglich ist, alles wieder zurück in den Karton zu packen, falls das nötig sein sollte? Rücksendungen sind schließlich keine Seltenheit. Willkommen in der wunderbaren Welt der Verpackungen!

Inverse Tetrismatik

Im Artikel über die Sanduhr [1] von Januar 2017 habe ich das Prinzip der „inversen Kinematik" beschrieben, deren Details ich Ihnen in dieser Kolumne ersparen möchte. Wenn Sie wollen, können Sie – als Form der Selbstbestrafung – gerne den entsprechenden Artikel lesen. Etwa zu dieser Zeit stand ich mit meiner besseren Hälfte in einer Warteschlange an der Kasse im örtlichen Supermarkt. Hier machten wir uns einen Spaß daraus, die Artikel aus dem Einkaufswagen so auf dem Kassenband zu arrangieren, dass wir sie nach dem Scannen schnell und mit den schwersten Artikeln ganz unten in unsere Taschen packen konnten. Da wir ab und zu gerne ein kleines Wortspiel machen, wurde der Begriff „inverse Tetrismatik" geboren.

Als Elektronik-Enthusiast richtet man seine Aufmerksamkeit natürlich auf die elektronische Seite der Dinge wie Schaltpläne, Bauteile, Schaltungen, Leiterplatten, Module, Messgeräte und vieles mehr. Die mechanischen Aspekte, etwa Gehäuse und Frontplatten, sind im Allgemeinen weniger beliebt und vielen bricht schon beim bloßen Gedanken daran der kalte Schweiß aus. Stundenlanges Stehen und Feilen an einem Stück Stahlblech ist nicht gerade die interessanteste oder intellektuell anspruchsvollste Tätigkeit. Aber gut, während die Mechanik vielleicht noch eine - wenn auch etwas weniger positive - Emotion auslöst, ist das bei einer Verpackung im Allgemeinen nicht der Fall. Wer, um Himmels willen, hat deswegen schlaflose Nächte? Im besten Fall sind es YouTuber wie Marco Reps [2] oder Dave Jones vom EEVBlog [3], die nach eigenen Angaben ihren Lebensunterhalt damit verdienen, dass sie auf YouTube mit einem absurd großen Messer Verpackungen öffnen. Ich muss in aller Bescheidenheit zugeben, dass vor allem das Video von Marco Reps meine Gefühle tief berührt hat. Ihr solltet wissen, mit wie viel Sorgfalt und Liebe ich diese Kits verpackt habe.

Das Verpacken von Produkten und Kits sieht lächerlich einfach aus, aber nichts ist weiter von der Wahrheit entfernt; es erfordert ein gerüttelt Maß an Hirnschmalz und Knobelaufwand. Muss die Verpackung attraktiv aussehen oder ist das weniger wichtig? Welcher Karton und welches Material und welche genaue Größe? Wie viel darf es kosten? Natürlich darf der Karton - vor allem bei zerbrechlichen Gegenständen - nicht zu klein, aber auch nicht zu groß sein, weil man vielleicht mehr Füllmaterial oder größere Verpackungseinsätze braucht. Der Inhalt darf im Karton nicht hin und her schlackern, wenn er transportiert wird, denn das könnte ihn oder die Teile darin beschädigen, und es ist auch nicht Sinn der Sache, dass der Kunde den Inhalt durch Schütteln des Pakets erraten kann. Letzteres ist nicht nur bei der etwas jüngeren Bevölkerung in Mode, sondern liegt auch in der Weihnachtszeit im Trend [4]. Bei

Elektronik gibt es noch weitere Anforderungen wie antistatische Verpackungsmaterialien. Wenn ein Produkt aus mehreren Teilen besteht, lohnt es sich, es so zu verpacken, dass der Kunde beim Auspacken des Kartons die Teile sofort in der richtigen Reihenfolge zur Montage vorfindet, inverse Tetrismatik vom Feinsten also.

Manchmal spielen auch weniger offensichtliche Faktoren eine Rolle bei der Wahl eines bestimmten Verpackungsmaterials: Als ich die Anweisung gab, den Sand für die Sanduhr in einem antistatischen Beutel zu verpacken, führte dies zunächst zu einigen hochgezogenen Augenbrauen. Keinem ist nämlich der Gedanke gekommen, dass trockener feiner Sand aufgrund elektrostatischer Anziehung hartnäckig an gewöhnlicher Plastikfolie haftet, was das Verschließen des Beutels erschwert!

Gewichtige Angelegenheiten

Das Verpacken von unzähligen Kleinteilen, wie es bei Elektronikbaukästen der Fall ist, birgt seine eigenen Herausforderungen. Es lohnt sich, eine Packliste zu erstellen und die Teile nach Art und Funktion in einzelne kleine Tüten zu sortieren, die dann in eine größere Tüte kommen. So hat man einen besseren Überblick und das Risiko, Fehler zu begehen, ist deutlich geringer, als wenn man einfach alles auf einen Haufen in den Karton wirft.

Außerdem ist es eine gute Idee, die Tüten einzeln auf einer Präzisionswaage zu wiegen, denn wenn ein Teil fehlt oder eines zu viel ist, merkt man das sofort.

Das Sortiment an Waagen und Beuteln, das wir zu diesem Zweck besitzen, würde so manchen Händler von bewusstseinserweiternden Kräutern und Pulvern vor Neid erblassen lassen, aber auch der Packtisch, die leeren Behälter, die Klebebandabroller, die Versiegelungsgeräte, die Schlauchfolie, das Beschriftungsgerät und das Industrieregal sind verdammt praktisch.

Ein weiteres Ärgernis ist unser treues belgisches Postwesen. Aus dem einen oder anderen Grund verdoppeln sich fast die Kosten für den Versand eines Pakets ab einem bestimmten Gewicht plötzlich. Die Kunst besteht also darin, unter dieser magischen Grenze zu bleiben, und ich habe mich schon mehrmals stundenlang mit einer Waage und Verpackungsmaterial abgemüht, um endlich zur Post zu gehen, immer noch ohne viel Zuversicht. Umgekehrte Tetrismatik auf Steroiden also, etwas, das nach einer Weile ziemlich ermüdend werden kann.

Schließlich gibt es zum Glück noch den Fernen Osten mit Onkel Ali und Konsorten. Dank ihrer demokratisch niedrigen Preise, ihres unerbittlichen Wettbewerbs und anderer zweifelhafter wirtschaftlicher Aktivitäten müssen wir schon seit einiger Zeit keine Bausätze mehr einpacken. Das erspart mir einen ganzen Haufen Arbeit, würde ich sagen: Herzlichen Dank dafür!

210625-03

Ein Beitrag von

Text: **Ilse Joostens**
Redaktion: **Eric Bogers**
Übersetzung: **Sophia Gerstendorf**
Layout: **Harmen Heida**

Haben Sie Fragen oder Kommentare?

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

WEBLINKS

[1] Ilse Joostens, „Sanduhr“, Elektor 1/2017: www.elektormagazine.de/magazine/elektor-201701/40035
[2] Marco Reps: Elektor 6 Digit Nixie Clock: https://youtu.be/ge_9CNiZZ_A?t=25
[3] Dave Jones EEVBlog: Elektor IV-22 VFD clock: https://youtu.be/SyXiWNZs7l4?t=1733
[4] DIY Hacks and How Tos: A Prank for People That Shake Their Christmas Presents: www.youtube.com/watch?v=ZeCjiEiPqAM

Unter dem Radar

Unbekannte Mikrocontroller, über die Sie Bescheid wissen sollten

Von Clemens Valens (Elektor)

Der globale Markt für Mikrocontroller ist vielfältiger, als viele Menschen denken. Werfen wir einen Blick auf einige der Mikrocontroller und Hersteller, die nicht so oft in Elektor zu finden sind. Vielleicht finden Sie den einen oder anderen davon bei einem zukünftigen Projekt nützlich.

Die Wahl des Mikrocontrollers für eine Elektor-Schaltung basiert meist auf der Verfügbarkeit von kostengünstigen Software-Entwicklungswerkzeugen und Programmiergeräten sowie der Möglichkeit, diese zu kaufen oder zu bauen. Daher haben viele Elektronik-Enthusiasten einen eher begrenzten Blick auf den weltweiten Mikrocontroller-Markt. Es gibt viel mehr als PIC, AVR, ARM oder ESP, Controller, die wir so oft in DIY-Projekten sehen. Werfen wir einen Blick auf einige der MCUs, die in Ihrem blinden Fleck leben.

Alles begann mit vier Bits

Der 1971 vorgestellte Intel 4004 gilt als der erste kommerziell hergestellte Mikroprozessor (mehr dazu in unserem Elektor-Industry-Spezial über 60 Jahre Elektronik [1]). Der 4-Bit-Prozessor war Bestandteil der MCS-4-Familie. Ihm folgte die MCS-40-Familie mit der 4040-CPU. Der erste erfolgreiche Mikrocontroller (kein Mikroprozessor) war der TMS1000 von Texas Instruments aus dem Jahr 1974, ebenfalls ein 4-Bit-Baustein, der wie der 4040 seinen Weg in viele Taschenrechner fand.

In einer Welt, in der die Hersteller von Mikrocontrollern anscheinend nach möglichst hohen Datenwortbreiten streben - 64 Bit sind keine Seltenheit -, dürfte es Sie überraschen, wie viele 4-Bit-Mikrocontroller noch heute im Einsatz sind. Aber warum? Die Antwort ist wahrscheinlich eine Mischung aus Legacy-, Strombedarfs- und Kostengründen.

Eine 4-Bit-MCU kann mit weniger Transistoren gebaut werden als Bausteine mit einer größeren Wortbreite. Daher benötigen sie unter sonst gleichen Randbedingungen weniger Strom, was zu einer längeren Batterielebensdauer beiträgt. Weniger Transistoren bedeuten auch weniger Platz, und so kann ein 4-Bit-Kern in einer Ecke des Chips untergebracht werden oder der Chip kann kleiner sein, was wiederum Kosten spart (auch wenn man sich fragen mag, wie viel). Großvolumige Anwendungen wie Taschenrechner, Timer, Uhren, Fahrradcomputer, Spielzeug und Fernbedienungen verwenden 4-Bit-MCUs, und das schon seit vielen Jahrzehnten. Da die Hersteller es in der Regel vermeiden, Produkte zu ändern, die sich in der Praxis bewährt haben - wenn es läuft, lass es laufen - erklärt dies gut, warum es immer noch einen Markt für solche kleinen MCUs gibt.

Falls Sie einen 4-Bit-Mikrocontroller ausprobieren möchten, werfen Sie einen Blick auf die NY-Familien des taiwanesischen Unternehmens Nyquest. Die modernen Entwicklungswerkzeuge können kostenlos heruntergeladen werden (**Bild 1**). Weitere Hersteller sind EM Microelectronic aus der Schweiz, CR Micro aus China und Tenx Technology aus Taiwan.

8051

Bevor ARM zum wichtigsten MCU-Core-Lieferanten für fast alle Halbleiterhersteller der Welt wurde, gab es den 8-Bitter 8051. Der 1980 von Intel als MCS-51 entwickelte Kern (**Bild 2**) wurde an mehrere Wettbewerber lizenziert und fand seinen Weg in eine Vielzahl von Produkten. Viele dieser Produkte oder ihre Derivate, Varianten und Geschwister werden auch heute noch hergestellt, und auch 40 Jahre nach der Markteinführung werden 8051-Derivate aktiv in neue Produkte eingesetzt.

Ein weiterer Grund dafür ist, dass Legionen

von 8051-Anwendern in dieser Zeit viel Software und Know-how entwickelt haben, und eine solche Ressource wirft man nicht leichtfertig weg, nur weil ein besserer Mikrocontroller auf den Markt kommt.
Schließlich ist der 8051-Kern sehr billig, wenn nicht sogar kostenlos geworden. Das macht ihn zu einer interessanten Option für Halbleiterhersteller, die extrem preiswerte Geräte entwickeln. Nicht immer wird dies im Datenblatt erwähnt, aber wenn dort so etwas wie „1T instruction cycle“ steht, können Sie mit Sicherheit davon ausgehen, dass es sich um ein 8051-Derivat handelt. Der ursprüngliche 8051 benötigte für die meisten Befehle zwölf Taktzyklen (genannt „12T“), während die modernsten Versionen nur einen benötigen (daher „1T“). Abgesehen davon, dass weniger Taktzyklen pro Befehl benötigt werden, ist die Programmausführung auch (viel) schneller, zumal einige dieser modernen Derivate mit Frequenzen von bis zu 450 MHz anstelle von 12 MHz des Originals laufen. Daher sind moderne 8051-MCUs preiswerte und dennoch leistungsfähige 8-Bit-Mikrocontroller.
Die wohl größte Unannehmlichkeit des 8051 aus heutiger Sicht ist wahrscheinlich seine Inkompatibilität mit modernen Programmiersprachen wie C und C++, was in seiner seltsamen Speicherstruktur begründet liegt. Um das Beste aus dem 8051 herauszuholen, bedarf es einer kommerziellen Toolchain von Keil oder IAR oder ähnlichen Compiler-Profis. Die populäre GCC-Toolchain, die auf alle möglichen Mikrocontroller portiert wurde, bietet leider keine Unterstützung für den 8051. Die freie Toolchain SDCC macht einen einigermaßen vernünftigen Job, ist aber bei weitem nicht perfekt. Die Programmierung des 8051 in Assembler ist natürlich auch eine Option. Oder bevorzugen Sie Pascal? Dann werfen Sie einen Blick auf Turbo51 [2]!
Sie finden 8051-basierte MCUs in preisgünstigen Produkten mit hohen Stückzahlen wie Spielzeug, PC-Tastaturen und -Mäusen, Zahnbürsten, Haushaltsgeräten, Fernbedienungen und so weiter, hergestellt hauptsächlich in Asien, wo der 8051 besonders beliebt zu sein scheint. Silan, SiGma Micro, SinoWealth, Silicon Laboratories, Sonix, STC und SyncMOS (um nur einmal die mit „S“ beginnenden Firmen zu nennen) stellen

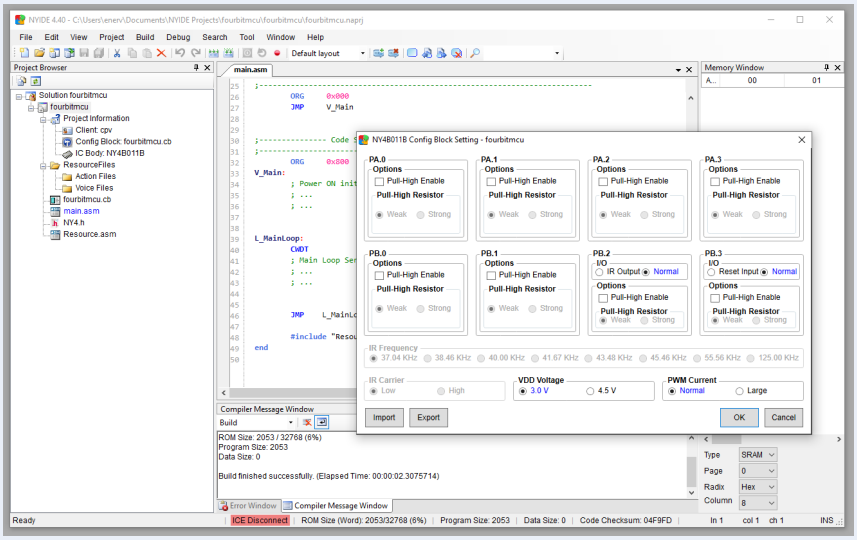


Bild 1. NYIDE 4.40 von Nyquest zeigt, dass auch 4-Bit-Mikrocontroller moderne IDEs haben können.

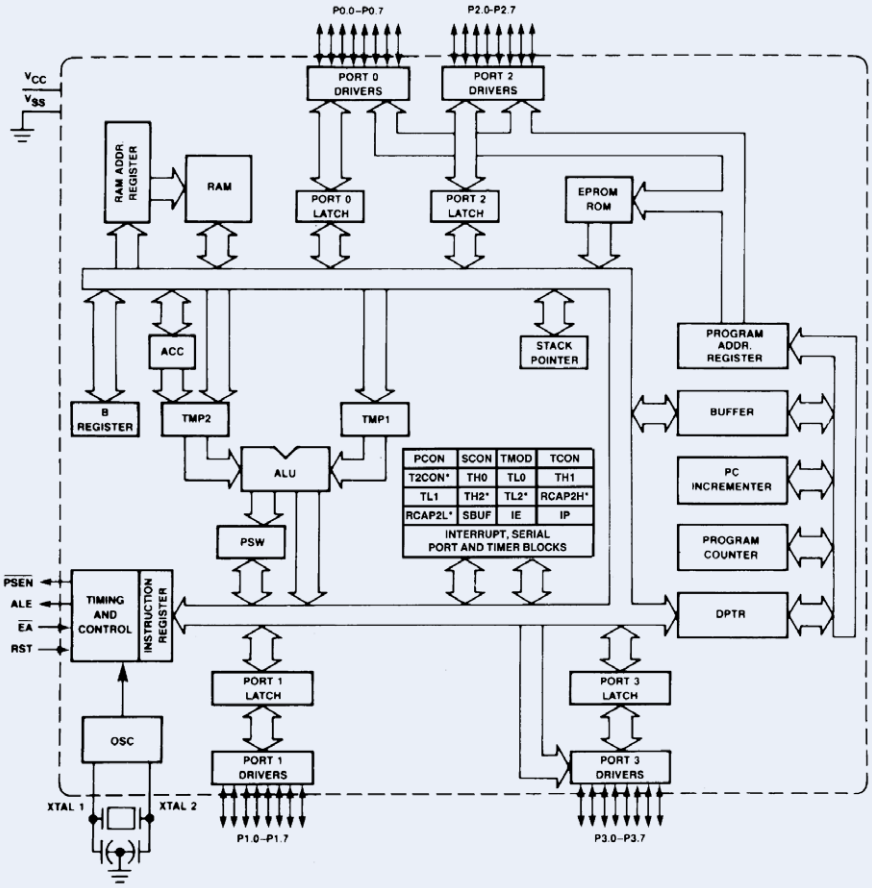


Bild 2. Die grundlegende Architektur des weltweit vielleicht am häufigsten verwendeten Mikrocontroller-Kerns 8051. (Quelle: Intel)

Bild 3. Im Commodore 64, einem der berühmten Heimcomputer aus den 1980er Jahren, schlägt ein Herz aus 6502. (Quelle: ralfsfotoseite bei Pixabay)

alle Produkte mit 8051ern her.
Wenn auch Sie mit einem modernen 8051-Derivat experimentieren möchten, sollten Sie sich die CH55x-Familie von WCH ansehen. Sie sind zwar nur auf Chinesisch dokumentiert, aber billig und verfügen über eine USB-Schnittstelle, die eine einfache Programmierung ermöglicht. Und sie werden von Open-Source-Projekten unterstützt. Einen 8051 in einen Arduino verwandeln? Kein Problem, wie zum Beispiel in [3] zu sehen.

Evergreens und Ewiggestrige

Neben dem 8051 aus dem Jahr 1980 gibt es noch einige andere Prozessorkerne aus dieser Zeit, vor allem den 6502 und den Z80, aber auch den etwas jüngeren 68000 von Motorola (jetzt NXP). Der Z80 und die CMOS-Version des 6502, der W65C02, werden immer noch von ihren Entwicklern Zilog (6502) beziehungsweise WDC(W65C02) hergestellt und aktiv unterstützt. Der 68000 hingegen scheint hauptsächlich zur Unterstützung älterer Anwendungen am Leben erhalten zu werden (aber wer weiß, wie viele andere Hersteller eine Lizenz erworben haben?).

6502

Der 6502 wurde in mehreren berühmten frühen Computern wie dem Commodore 64, dem Apple II und dem BBC Micro eingesetzt und war tatsächlich sehr erfolgreich (**Bild 3**). Seine verbesserte und energiesparende CMOS-Variante ist immer noch aktuell, auch wenn sie recht teuer ist. Der Grund dafür ist, dass die meisten Anwender den Kern nur für die Verwendung in FPGAs, ASICs und ähnlichen kundenspezifischen Chips lizenzieren. Da dies alles „confidential" ist, fällt es schwer herauszufinden, um welche Beuteile es sich handelt. WDC stellt jedoch echte 65er-ICs her, die Sie ausprobieren können, zum Beispiel den Mikrocontroller W65C265S mit einer 16-Bit-CPU W65C816S, die vollständig mit dem 8-Bit-Chip W65C02S kompatibel ist und bereits ab 1,8 V läuft. Es gibt auch Controller-Module und sogar ein „Companion's Board" mit Seeed-Studio-Grove-, Sparkfun-QWIIC- und MikroE-Click-Anschlüssen.

Z80

Der Z80 ist ein weiterer sehr erfolgreicher Prozessor aus den späten siebziger und frühen achtziger Jahren des vorigen Jahrhunderts. Der Kern wurde 1975 von Zilog entwickelt und wird immer noch hergestellt. Er wurde an viele andere Hersteller weltweit lizenziert und von diesen kopiert und geklont, was zu einer riesigen Anwenderbasis geführt hat. Es gibt mehrere Familien auf dem Markt, zum Beispiel den eZ80 mit Taktfrequenzen bis zu 50 MHz oder den Z8 und den eZ8 Encore! Letzterer ist beispielsweise in der ZMOTION-Produktlinie zu finden, einer MCU-Familie, die für PIR-Bewegungserkennung optimiert ist.
Wenn Sie experimentierfreudig sind, sollten Sie den Z8FS040BSB mit seinen 4KB-Flash-Speicher und fünf GPIO-Pins in einem praktischen 8-poligen SOIC-Gehäuse ausprobieren. Der kostenlose Compiler SDCC unterstützt mehrere Z80-basierte MCUs wie die von Rabbit (jetzt Digi) und den Nintendo Gameboy.

Ultra-Low-Cost

Viele elektronische Produkte mit Mikrocontrollern werden in großen Mengen hergestellt. Denken Sie an viele Haushaltsgeräte, Uhren, elektrische Zahnbürsten, E-Zigaretten, Corona-Virentester, Chipkarten, Rauchmelder, Spielzeug und so weiter und so fort (**Bild 4**). Um eine Vorstellung von den Zahlen zu bekommen: Spielkonsolen wie der Nintendo-Gameboy, -Wii und -Switch haben jeweils die 100-Millionen-Marke verkaufter Einheiten überschritten. Stellen Sie sich vor, wie hoch die Zahlen beispielsweise bei Smartcards sind. Winzige Einsparungen bei einem Exemplar senken die Kosten für das Massenprodukt enorm! Und so gibt es halt einen großen Markt für extrem billige Mikrocontroller. Einige

Bild 4. Beispiele für Anwendungen, die durch extrem preisgünstige Mikrocontroller ermöglicht werden. (Quelle: Holtek.com)

dieser Mikrocontroller-Hersteller, auf die Elektronik-Bastler und -Maker aufmerksam geworden sind, sind Padauk, MDT und Holtek.

Padauk

Padauk stellt 3-Cent-MCUs her, die einmalig programmierbar (OTP) und flashbasiert sind. Die Besonderheit dieser Bausteine ist die auf FPPs (Field-Programmable Processing Units) basierende Architektur. Dabei handelt es sich um Registerbänke mit einem Programmzähler, einem Stapelzeiger, einem Akkumulator und einem Flagregister, die eine schnelle Kontextumschaltung ermöglichen. Dies ist zum Beispiel für die Interrupt-Verarbeitung und für Multitasking nützlich. Sie erinnern ein wenig an die vier Registerbänke des 8051. Da die meisten ihrer Produkte jedoch nur ein FPP haben (einige haben zwei, der PFC460 hat vier und der MCS11 acht), sind sie sehr einfache MCUs.

Der PMS150 ist ein guter Startpunkt. Eine exzellente Beschreibung dieser Bausteine finden Sie unter [4]. SDCC unterstützt den PDK14 und den PDK15, während an der Unterstützung für den PDK13 gearbeitet wird.

MDT

Wie bereits erwähnt, stellt MDT a.k.a. Micon Design Technology Klone oder Derivate von PIC-Controllern von Microchip her. Da PIC-MCUs bei vielen Herstellern sehr beliebt sind, haben auch MDT-ICs einige Aufmerksamkeit auf sich gezogen. Während der Recherchen für diesen Artikel ging die MDT-Website jedoch plötzlich offline. Bei der Internet-Suche nach MDT-Bausteinen fand ich jedoch mehrere Ergebnisse von MCU-Cracking- und Reverse-Engineering-Diensten unter anderem für MDT-MCUs, was darauf schließen lässt, dass sie weit verbreitet sind.

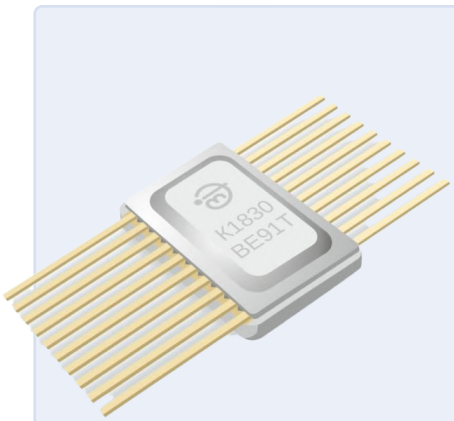


Bild 6. Der K1830BE91T von NIIET besitzt einen 8051-Kern und entspricht funktionell dem AT89C2051 von Microchip. (Quelle: https://niiet.ru)

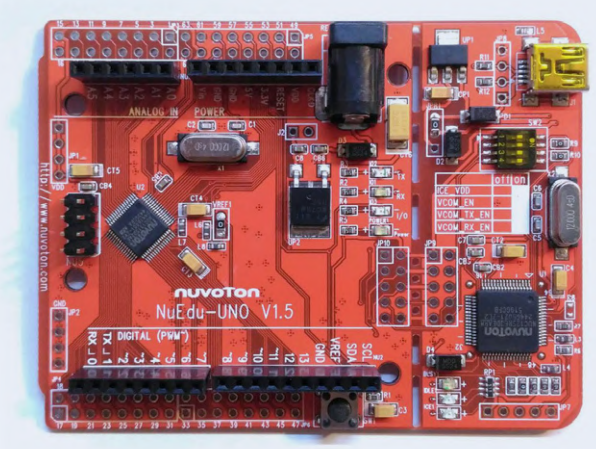


Bild 5. Das Uno-Board NuMaker von Nuvoton besitzt ein abnehmbares NuLink Programmier/Debug-Modul. (Quelle: https://danchouzhou.blogspot.com)

Holtek

Einige Holtek-Produkte wurden in der Vergangenheit in Elektor-Projekten eingesetzt. Dabei handelte es sich aber nicht um Mikrocontroller, sondern um Tastatur- und RC-Kanaldecoder. Holtek stellt aber auch MCUs her, und zwar viele, von 8051-Typen über ARM-Cortex-M0 und -M3 bis hin zu Bausteinen, die auf einem eigenen proprietären Kern basieren. Wie viele der Low-Cost-MCU-Anbieter ist auch die Produktlinie von Holtek in anwendungsspezifische und Allzweck-MCUs („I/O-Typ") unterteilt. Die Dokumentation ist gut und die HT-IDE3000-IDE (Assembler und C) ist kostenlos (wenn auch etwas schwer zu finden), aber ein Holtek-Programmiergerät ist erforderlich.

Eine MCU, den ich interessant fand, ist der analoge HT66F4550 mit zwei Operationsverstärkern und einem Audioausgang „on chip".

Und die Großen aus Asien?

Elektor hat hunderte von mikrocontrollerbasierten Projekten veröffentlicht, und in den meisten Fällen enthielten sie einen PIC- oder AVR-Baustein von Microchip (und früher Atmel), einen ESP-Baustein von Espressif oder eine MCU mit einem ARM-Kern von NXP oder ST. Auch der MSP430 von Texas Instruments tauchte einige Male in Projekten auf. Mit Ausnahme von Espressif sind all diese Hersteller in Europa und den USA beheimatet. Das zeigt, wie voreingenommen wir sind, denn es gibt natürlich viele große asiatische Unternehmen, die MCUs herstellen.

Renesas

Einer der größten, wenn nicht sogar der größte asiatische Halbleiterhersteller ist Renesas, der sich aus Abteilungen von NEC, Hitachi und Mitsubishi zusammensetzt. Manche behaupten sogar, dass Renesas weltweit die Nummer eins unter den MCU-Anbietern ist. Langjährige Elektor-Leser erinnern sich vielleicht noch an die R8C- und R32C/111-Artikelserie von vor etwa 15 Jahren [5]. Die aktuelle RX671-Familie von 32-Bit-Mikrocontrollern ist auf schnelle Echtzeitsteuerung und berührungslose Mensch-Maschine-Schnittstellen (HMI) in Verbindung mit Näherungsschaltern und Spracherkennung spezialisiert und eignet sich perfekt für moderne, hygienische HMI-Designs. Renesas bietet auch eine große Anzahl anderer Entwicklungs- und Evaluierungsboards an, und ich möchte Sie ermutigen, diese auszuprobieren und über Ihre Ergebnisse zu berichten.

Nuvoton

Nuvoton wurde 2008 aus Winbond ausgegliedert und übernahm 2020 die Chip-Sparte von Panasonic. Nuvoton hat ein großes Angebot an 8051- und ARM-Core-MCUs und auch einige proprietäre Core-Bausteine. Im Gegensatz zu vielen Mitbewerbern verfügt Nuvoton nicht über eine eigene Toolchain. Für die 8051-Bausteine setzen sie auf Keil

und IAR, für die ARM-Bausteine auf Eclipse. Auf GitHub findet man Unterstützung für die Verwendung von SDCC bei einigen Nuvoton-Bausteinen.
Der NuMaker Uno (**Bild 5**) ist eine gute Möglichkeit, mit Nuvoton-Controllern loszulegen. Es handelt sich offensichtlich um ein Arduino-kompatibles Board mit einem NUC131-ARM-Cortex-M0 Controller. Enthalten ist ein abnehmbares Debugger/Programmiermodul namens Nu-Link, das auch mit anderen Boards verwendet werden kann. Software-Unterstützung findet man auf GitHub (OpenNuvoton). Im NuMaker-Repository finden Sie Unterstützung für mbed, Arduino, MicroPython und mehr.

Russische MCUs?

Um diesen Artikel zu vervollständigen, wollte ich einige Informationen über russische Mikrocontroller hinzufügen. Leider kann ich kein Russisch, und die meisten Websites sind (nur) auf Russisch, was die Suche nach nützlichen Informationen erschwert. Ich bin auf Milandr, Mikron und die „fabless" Firma Syntacore gestoßen, die alle RISC-V-basierte Controller entwickeln. Laut [6] hat Milandr auch eine Lizenz für ARM-Cores, aber auf ihrer Website werden solche Bauteile nicht erwähnt.
NIIET stellt den K1921VK01T her, der auf Motorsteuerungs- und Smart-Metering-Anwendungen ausgerichtet ist und auf einem ARM-Cortex-M4F-Kern basiert. OpenOCD bietet Unterstützung für diese MCU. Im Oktober 2021 kündigte NIIET einen RISC-V-basierten Controller an, der die STM32- und MSP430-Bauteile ersetzen soll, die derzeit in „zivilen Geräten" (wie sie es nennen) in Russland verwendet werden. Außerdem gibt es 8- und 16-Bit-RISC-MCUs sowie einige MCS-51- (Intel 8051) und MCS-96- (Intel 80196) Bausteine (**Bild 6**).

Eine Welt voller Möglichkeiten

In diesem Artikel wurden einige Mikrocontroller und Hersteller vorgestellt, die nicht oft in Elektor-Projekten zu sehen sind, die aber einen wichtigen Teil des weltweiten MCU-Marktes darstellen. Natürlich ist dieser Artikel bei weitem nicht vollständig, und es kann sein, dass ich einige interessante MCUs oder Hersteller übersehen habe. Während der Recherche zu diesem Artikel habe ich eine Liste von mehr als 50 aktiven Mikrocontroller-Herstellern zusammengestellt, und ich bin sicher, dass es noch viele mehr gibt. Wenn Sie weitere unbekannte, aber interessante Mikrocontroller (-Familien) kennen, die Sie den anderen Lesern vorstellen möchten, lassen Sie es mich bitte wissen.

210630-02

QUIZ

Sowohl der 4004 als auch der 4040 wurden von Federico Faggin entworfen. Welche anderen berühmten Prozessoren stammen ebenfalls von ihm?

Intel 8080, Zilog Z8 & Z80

Ein Beitrag von

Idee und Text: **Clemens Valens**
Redaktion: **Jens Nickel** und **C. J. Abate**
Übersetzung: **Rolf Gerstendorf**
Layout: **Harmen Heida**

Haben Sie Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Schicken Sie eine E-Mail an den Autor unter clemens.valens@elektor.com oder kontaktieren Sie Elektor unter redaktion@elektor.de.

PASSENDE PRODUKTE

- Buch: *Mastering Microcontrollers Helped by Arduino* (SKU 17967)
  www.elektor.de/17967
- Raspberry Pi RP2040 Microcontroller (SKU 19742)
  www.elektor.de/19742
- Buch: *ARM Microcontroller Projects* (SKU 17620)
  www.elektor.de/17620

WEBLINKS

[1] Stuart Cording, „Die Geburt des Mikroprozessors", Elektor Industry 11/2021: https://www.elektormagazine.de/magazine/elektor-242/60078
[2] Pascal für 8051: https://turbo51.com/
[3] CH55xduino: https://github.com/DeqingSun/ch55xduino
[4] Padauk PMS150: https://jaycarlson.net/2019/09/06/whats-up-with-these-3-cent-microcontrollers/
[5] Gunther Ewald, „Der R8C und seine Familie", Elektor 11/2005: https://www.elektormagazine.de/magazine/elektor-200511/2270
[6] Russische Mikrocontroller: https://geek-info.imtqy.com/articles/M4836/index.html

Drahtloses Monitoring und Debuggen

Serielle Funk-Schnittstelle für Arduino, ESP32 und Co.

Von **Peter Tschulik** (Österreich)

Moderne Mikroprozessor-Boards, die beispielsweise mit dem ESP32 oder dem ESP8266 bestückt sind, bieten viel Leistung für wenig Geld und einen hohen Programmierkomfort, etwa in der Arduino-IDE. Besonders interessant ist die Over-the-Air-Funktion (OTA). Sie ermöglicht es, über WLAN sowohl Programmupdates als auch Daten bequem über einen Bootloader aufzuspielen, ohne das Board an den USB eines PCs oder Laptops anzuschließen. Um aber serielle Daten (etwa Debugging-Infos) drahtlos zu übertragen, braucht man eine andere Lösung.

Das Wireless UART-RS232 long distance Set.

Besonders IoT-Geräte, die „im Feld" arbeiten und nicht so leicht zugänglich sind, profitieren von drahtlosen Firmware-Updates. Ein gutes Tutorial zu Over-the-Air-Updates über die Arduino-IDE findet man unter [1]. Ein Manko hat diese Methode jedoch: Der Serielle Monitor wird nicht unterstützt, so dass Ausgaben und Debugging-Informationen nicht übertragen werden können.

Bei meinem letzten Projekt, einer auf dem ESP32 basierenden Gießanlage, gab es ein Problem mit einem nicht richtig funktionierenden Sensor. Statt nun ein USB-Kabel quer über die Terrasse in mein Wohnzimmer zu meinem Computer zu verlegen, machte ich mich auf die Suche nach einer drahtlosen Lösung. Doch trotz intensiver Suche fand ich keinen fertigen Baustein, der meine Ansprüche befriedigen konnte, so dass ich mich entschloss, eine eigene Lösung zu entwickeln. Dazu fiel mir ein, dass es früher Systeme gab, die eine drahtlose Übertragung der seriellen Schnittstelle ermöglichten; und tatsächlich wurde ich in China rasch fündig. Das *Wireless UART-RS232 long distance Set* [2] bietet Übertragungsraten bis zu 115.200 Baud mit Konfigurationsmöglichkeit und eine Reichweite bis zu 1 km. Bei einem Preis von 40 € für das Sende- und Empfangspaar rechnete ich mir ein gutes Preis/Leistungsverhältnis für mein Projekt aus.

USB-Host-Adapter

Weitaus schwieriger gestaltete sich die Auswahl eines USB/Seriell-Wandlers. Zwar bestand die Möglichkeit, die serielle Schnittstelle aus dem Gießgerät herauszuführen und direkt an das Übertragungssystem anzuschließen, aber ich wollte eine universelle Lösung finden, die an USB-Anschlüssen unterschiedlicher Boards funktioniert. Hierzu benötigt man einen sogenannten „USB-Host-Adapter", der wie ein PC die jeweilige USB-Schnittstelle unterstützt. Zuerst stieß ich bei meiner Internetrecherche auf „USB-Host Shields" für den Arduino, siehe [3] und [4]. Ich entschied mich wegen der raschen Lieferung für das Modul aus Deutschland, das wenige Tage später

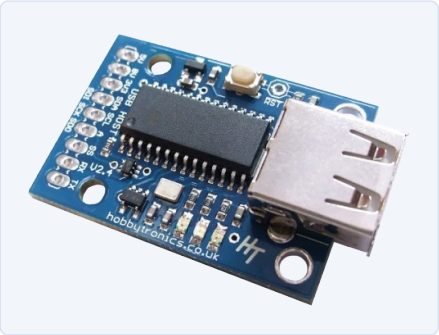


Bild 1. Ein kleines aber feines USB-Host-Board (Quelle: www.hobbytronics.co.uk).

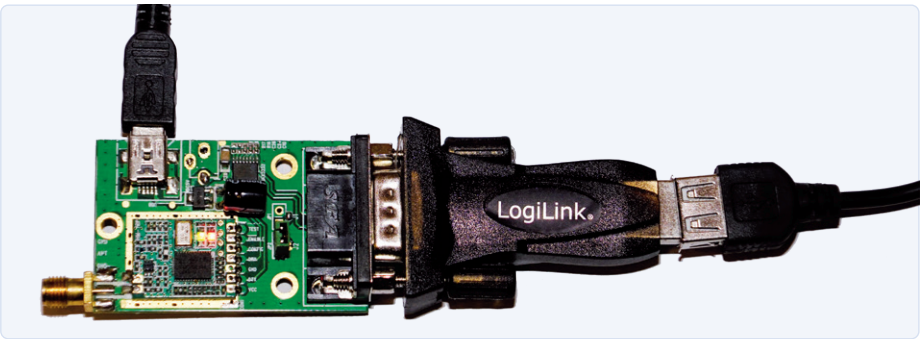


*Bild 2. Die **Wireless UART-RS232 long distance Module** werden konfiguriert. Im Bild kann man den gesteckten Jumper sowie die leuchtende grüne und rote LED erkennen.*

auf meinem Schreibtisch lag. Doch ich stieß gleich auf das nächste Problem: Unterschiedliche Boards verwenden unterschiedliche USB-Chips; die ESP32-Boards zum Beispiel den CP210x oder Chips von FTDI, Arduino-Boards den eigenen Atmel-Controller oder ebenfalls FTDI-Chips. Leider unterstützen die Bibliotheken für das Arduino-USB-Host-Shield nur jeweils einen Chiptyp. Das heißt, man müsste die Software konfigurierbar machen und immer genau wissen, welcher USB-Chip auf dem verwendeten Board sitzt – weit entfernt von einer Plug-and-Play Lösung! Schließlich wurde ich dann doch in einem kleinen Shop in England fündig (**Bild 1**) [5].

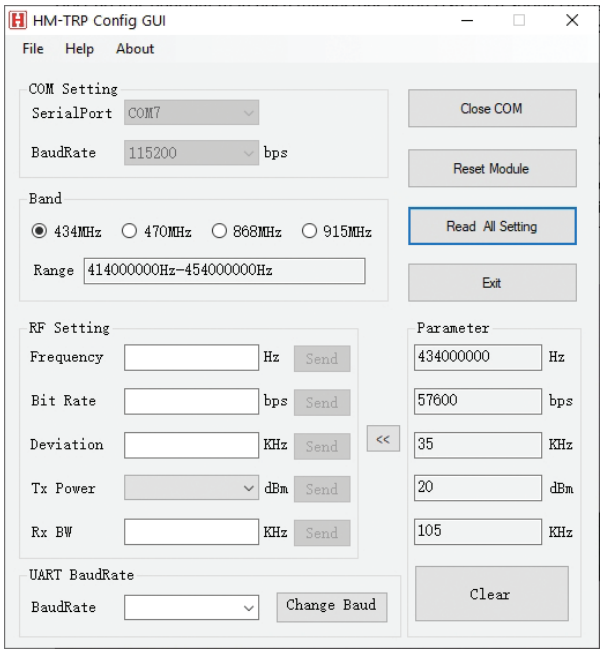


*Bild 3. Empfohlene Konfiguration für das **Wireless UART-RS232 long distance Modul.***

Interessant an diesem Board ist vor allem, dass es für die verschiedensten Anwendungen wie Flash-Memory-Stick, USB-Keyboard, USB-Joystick, USB-Maus, PS3- und PS4-Dual-Shock-Controller, Serial-Driver für FTDI, CP210X, PL2303, CH340/1 und CDC, MIDI-Gerät und USB-Modem verschiedene kostenlose Firmware-Anwendungen gibt. Bei der Bestellung kann man die gewünschte Firmware gleich mit angeben. Für meine Anwendung ist die Firmware für *USB Host - Serial Driver for FTDI, CP210X, PL2303, CH340/1 and CDC* genau richtig [6], die alle relevanten USB-Chips unterstützt.

Für die Programmierung und Konfiguration des Wireless-UART-RS232-Moduls beziehungsweise des USB-Host-Boards benötigt man noch mindestens einen USB-nach-Seriell-Umsetzer, den man entweder vom gleichen Hersteller des Wireless-UART-RS232-Moduls [8] bezieht oder beispielsweise von Logilink in vielen einschlägigen Geschäften findet [9]. Für mein Projekt benötigte ich dann nur noch zwei RS232-Pegelwandler. Hier fiel die leichte Wahl auf den guten alten MAX232 [7].

Einstellungssache

Bei der Bestellung des *Wireless UART-RS232 long distance Sets* war mir noch nicht klar, wie diese Module eigentlich zu konfigurieren seien. Als die Module aus China endlich auf meinem Labortisch lagen, stellte sich zunächst erst einmal Frust ein: Keinerlei Bedienungsanleitung oder sonstige Hinweise waren der Hardware beigefügt. So schrieb ich den Verkäufer an und schon nächsten Tag bekam ich per E-Mail die Software sowie ein Handbuch und ein Konfigurationsvideo [10]. Zuerst befreite ich die Module aus den Plastikgehäusen, lötete an J2 (zwischen den Bohrungen hinter dem RS232-Verbinder) eine Stiftleiste und steckte nach der beigefügten Anleitung einen Jumper auf die Stifte. Dann verband ich das Modul über den USB/Seriell-Adapter mit dem PC. Über ein Mini-USB-Kabel wird das Modul von einer 5-V-Stromversorgung (zum Beispiel einem USB-Steckernetzteil) mit Energie versorgt (**Bild 2**).

Leuchten dann die beiden LEDs, kann man das Programm *HY-TRP Setting GUI.exe* starten und dort den korrekten COM-Port sowie die Default-Baudrate auf 9.600 einstellen. Danach drückt man auf den Button *Open COM* und liest mit *Read All Setting* alle Werte aus. Danach werden die Werte wie in **Bild 3** gezeigt eingestellt.

Weitere Details können dem Handbuch [10] entnommen werden. Noch ein Hinweis: Ich habe zunächst eine Übertragungsrate von 115.200 Baud eingestellt, aber erfahren, dass die Übertragung mit der halben Bitrate von 57.600 Bd stabiler und zuverlässiger ist, da das *Wireless UART-RS232 long distance Modul* dann nicht ständig am Limit arbeiten muss. Das USB-Host-Board verfügt über einen Buffer, der die unterschiedlichen Übertragungsraten ausgleicht. Beide Module werden gleich eingestellt! Danach darf man nicht vergessen, die Jumper zu entfernen.

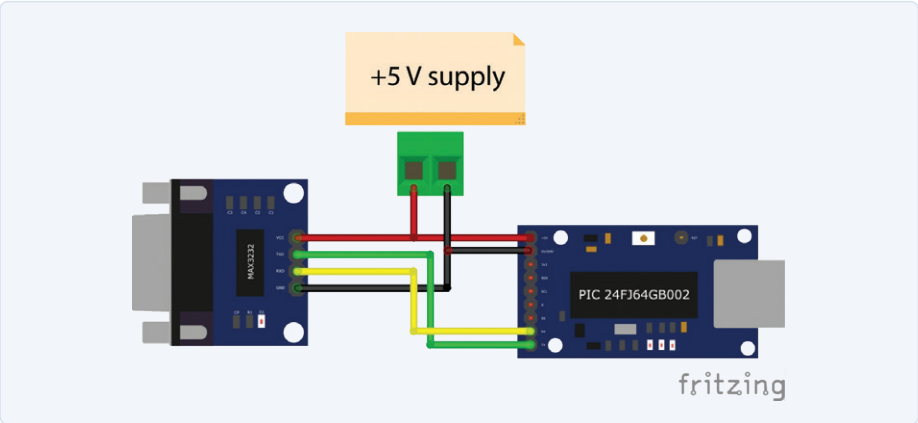


Bild 4. Verschaltung des USB-Host-Boards zur Konfiguration.

Konfigurierung des USB-Host-Boards

Für die Programmierung des USB-Host-Boards mit der korrekten Firmware sei auf die Beschreibung [5] verwiesen. Dort steht ein eigenes Programm zum Download bereit, mit dem das Board geflasht werden kann. Unter [6] kann die korrekte Firmware auf das Board geladen werden, und dort findet man auch eine Beschreibung der Parameter der Kommandoschnittstelle. Für die Konfiguration muss das USB-Host-Board noch – wie in **Bild 4** gezeigt – mit dem Pegelwandler-Platinchen verbunden werden. An den RS232-TTL-Adapter schließt man den RS232-zu-USB-Konverter an, der wiederum über ein USB-Kabel mit dem PC verbunden wird. Die Schaltung muss dann nur noch an eine stabile 5-V-Versorgung angeschlossen werden.

Danach startet man ein Terminalprogram (hier beispielsweise Hterm), wählt dort den korrekten USB-Port aus und stellt die Baudrate auf den Defaultwert von 9.600 Baud ein. Mit *Connect* links oben verbindet man sich mit dem USB-Host-Board. Rechts oben werden *Newline at* und weiter unten *Send on Enter* auf *CR+LF* umgestellt. Gibt man jetzt *HELP* im Eingabefeld von Input Control ein, sollten wie in **Bild 5** gezeigt die aktuellen Parameter aufgelistet werden.

Im Eingabefeld von *Input Control* werden nun die Parameter für unsere Anwendung modifiziert. Zunächst stellt man mit dem Befehl

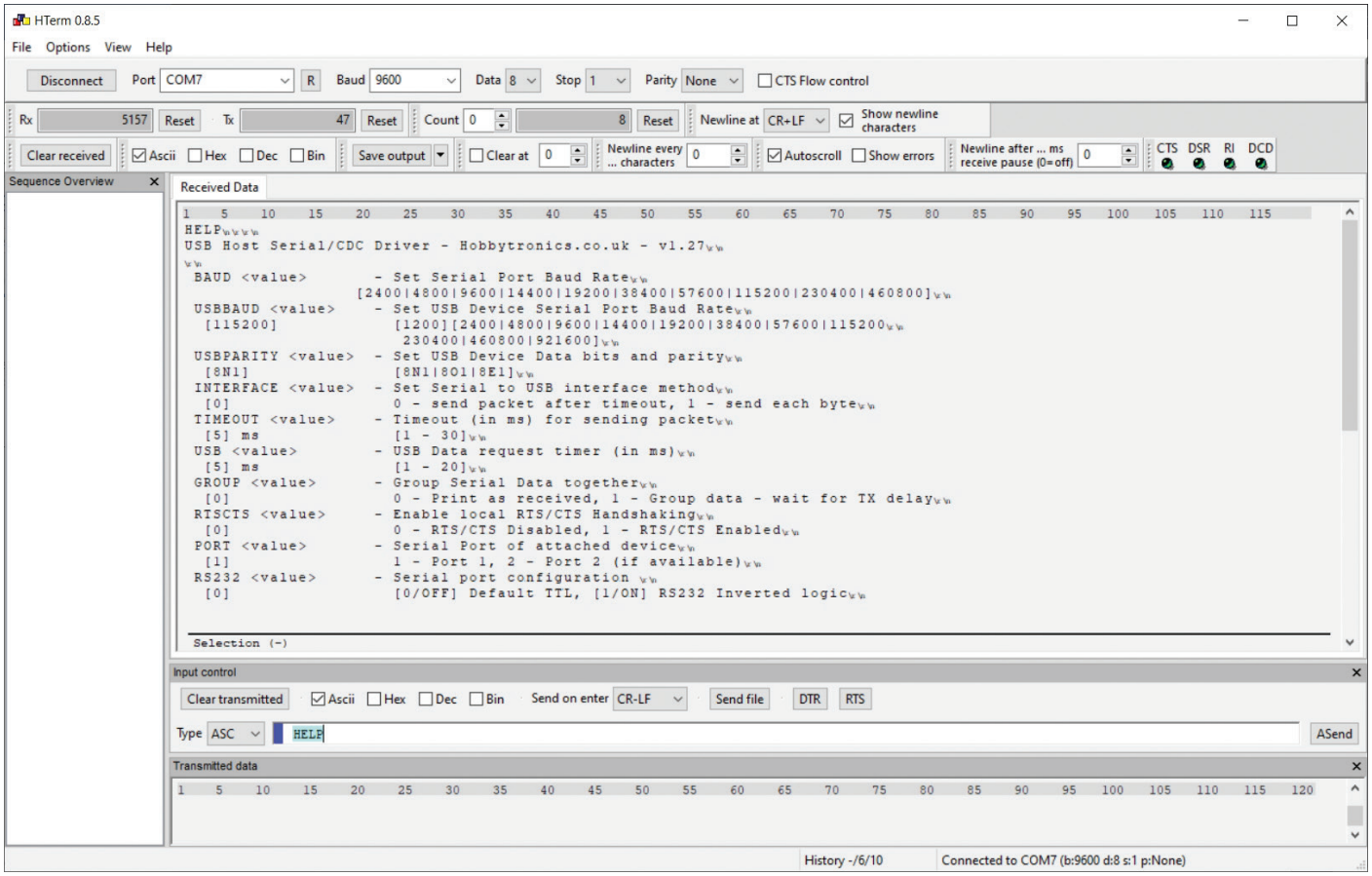


Bild 5. Konfiguration des USB-Host-Boards über Hterm.

Tabelle 1. Parameter-einstellungen in HTerm.

BAUD	57600
USBBAUD	115200
USBPARITY	8N1
INTERFACE	0
TIMEOUT	5ms
USB	5ms
GROUP	0
RTSCTS	0
PORT	1
RS232	0

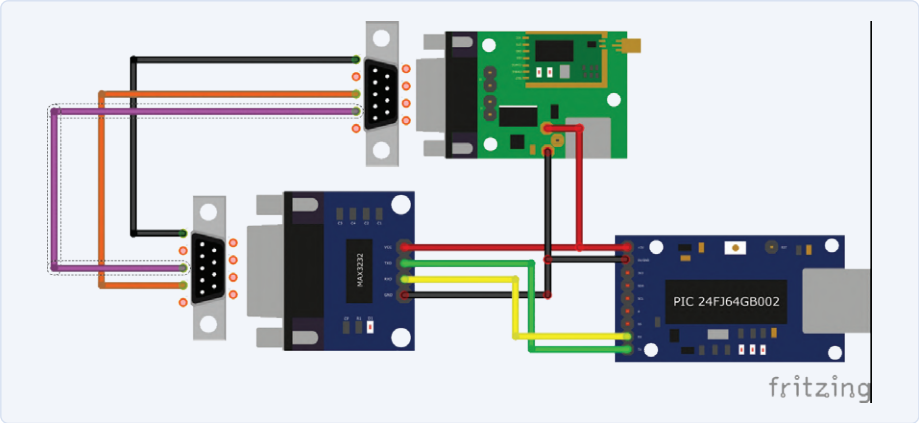


Bild 6. Verdrahtung des Senders (Seite des IoT-Boards). Das grüne Board ist das Wireless-Modul. Unten rechts wird das zu untersuchende IoT-Board über USB angeschlossen.

BAUD 57600 die Baudrate um, die mit der eingestellten Baudrate des *Wireless UART Systems* übereinstimmen muss. Unterbricht und verbindet man das Terminal wieder mit dem System (*Disconnect-Connect*), sollte man mit *HELP* die Kommunikation mit dem USB-Host-Board überprüfen können.

Mit dem Befehl *USBBAUD 115200* wird die USB-Baudrate auf 115.200 eingestellt. Dies bedeutet, dass die serielle Schnittstelle des angeschlossenen Arduino/ESP32/ESP8266-Boards in der Arduino-IDE immer mit dem Befehl *Serial.begin(115200);* konfiguriert werden muss. Ich habe diese hohe Baudrate gewählt, damit die seriellen Daten die Abarbeitung des eigentlichen Programms nicht zu lange verzögern.

Einstellungen für *USBPARITY*, *INTERFACE*, *TIMEOUT*, USB, *RTSCTS*, *PORT* und *RS232* bleiben unangetastet. Mit dem Befehl *GROUP 0* wird der Wert neu gesetzt. **Tabelle 1** zeigt zusammengefasst die korrekten Werte. Damit ist die Konfiguration abgeschlossen.

Verdrahtung

Bevor Sender (Seite des IoT-Boards im Feld) und Empfänger (Seite des PCs) fertig für den Betrieb verdrahtet werden können, werden noch auf den *Wireless UART-RS232 long distance Modulen* die Buchse für die Stromversorgung (neben der USB-Buchse) entfernt, damit sind die Pads zugänglich.

Auf der Sender-Seite wird die Mini-USB-Buchse des USB-Host-Boards als Stromversorgung genutzt. **Bild 6** zeigt die korrekte Verdrahtung des Senders. Für die Verbindung

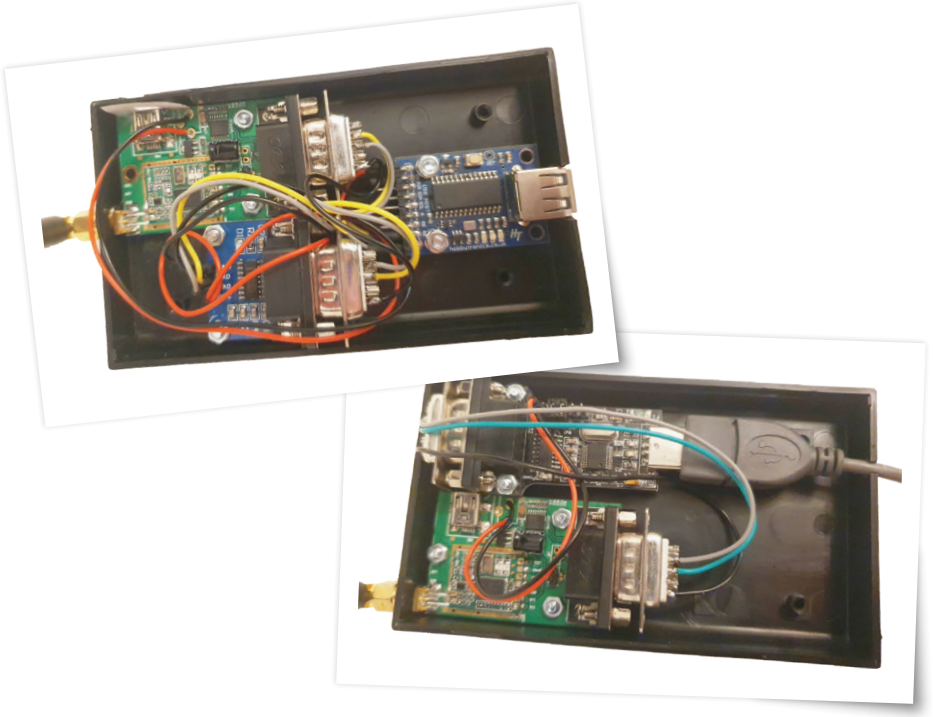


Bild 7. Verdrahtung des Empfängers. Unten rechts wird der PC über ein USB-Verlängerungskabel angeschlossen.

des *Wireless UART-RS232 long distance Moduls* mit dem RS232/TTL-Adapter bastelt man sich ein kurzes Kabel mit zwei männlichen SUBD-9-Verbindern. Pin 2 und Pin 3 sind wie im Bild sichtbar gekreuzt.
Für den Empfänger wird als RS232/USB-Konverter das kompakte Modul aus [8] verwendet und wie in **Bild 7** verschaltet. An den RS232/USB-Konverter des Empfängers wird ein USB-Verlängerungskabel angeschlossen, damit der Empfänger vom PC abgesetzt anschlossen werden kann.
Schließlich werden Sender und Empfänger in zwei passende kleine Gehäuse eingebaut. Die Tests meines OTA-Systems ergaben eine problemlose Übertragung über zwei Stockwerke mit Stahlbetondecken. Viel Spaß beim Nachbau!

200549-02

Gebrauchsanleitung für den praktischen Einsatz

- Konfiguration der seriellen Schnittstelle des eingesetzten (IoT-)Boards auf 115.200 Baud mit der Zeile `Serial.begin(115200);` in der Arduino-IDE.
- Nach Übertragung des Programms auf das eingesetzte Board Versorgung des Senders mit 5 V über die Mini-USB-Buchse des Wireless UART-RS232 long distance Moduls und Anstecken des eingesetzten Boards über ein passendes USB-Kabel.
- Anstecken des Empfängers an den PC an eine freie USB-Buchse. Der Empfänger wird vom PC aus mit Strom versorgt.
- Start der Arduino-IDE, Auswahl eines beliebigen Boards, Auswahl der USB-Schnittstelle des Empfängers und Start des Seriellen Monitors.
- Und schon sollten dort die vom Board gesendeten Daten sichtbar werden.

Sie haben Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare? Bitte wenden Sie sich per E-Mail an den Autor unter peter.tschulik@chello.at oder an die Elektor-Redaktion unter redaktion@elektor.de.

Ein Beitrag von

Idee, Ausführung und Text: **Peter Tschulik**
Redaktion: **Rolf Gerstendorf**
Layout: **Giel Dols**

PASSENDE PRODUKTE

- **Bundle: The Complete ESP32 Projects Guide + ESP32-DevKitC-32D (SKU 19897)**
  www.elektor.de/19897

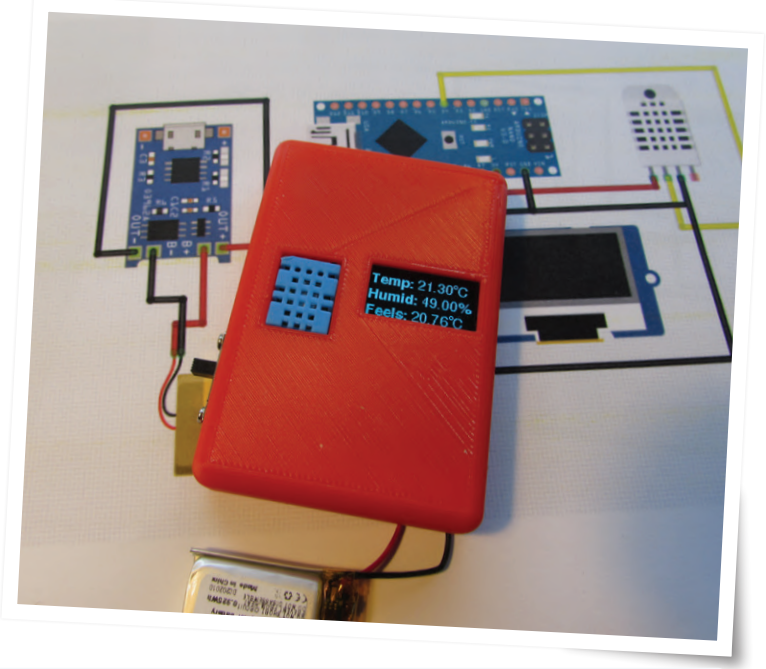
- **ESP32 & ESP8266 Kompilation (PDF) (SKU 18568)**
  www.elektor.de/18568

WEBLINKS

[1] ESP32 Basic Over The Air (OTA) Programming In Arduino IDE: https://lastminuteengineers.com/esp32-ota-updates-arduino-ide/
[2] Wireless UART-RS232 long distance Module: https://bit.ly/3c9iY34
[3] SparkFun USB Host Shield: http://www.sparkfun.com/products/9947
[4] USB-Host-Shield für Arduino: https://bit.ly/3fJQrTO
[5] USB Host Controller Board V2.4: https://www.hobbytronics.co.uk/usb-host-board-v24
[6] USB Host - Serial Driver for FTDI, CP210X, PL2303, CH340/1 and CDC : https://www.hobbytronics.co.uk/usb-host-serial
[7] RS232-TTL-Adapter: https://bit.ly/3peJH3r
[8] 2 Functions USB COM Port DB9 TTL232 RS232 TTL 232 CH340T USB2.0 Cable Adapter : https://bit.ly/3fHUGQ3
[9] LogiLink USB-2.0-zu-Seriell-Adapter: https://bit.ly/3EERA9K
[10] Konfigurationspaket zu „Wireless UART-RS232 long distance module HY035_HM-TRP-RS232": https://1drv.ms/u/s!AjrAGEbCBLlsugxiqmd1FRRCJjic

Tragbares **Temperatur- und Feuchtemessgerät**

Gebaut mit vorgefertigten Modulen



Von **Aarav Garg** (Indien)

Eine der einfacheren Herangehensweisen an moderne Elektronik ist es, Projekte durch die Kombination von vorgefertigten Modulen zu verwirklichen. Es ist nur etwas Verdrahtung und in den meisten Fällen Software erforderlich, um ein neues Gerät zu erstellen, wie das, das wir in diesem Artikel vorstellen.

Luc Lemmens, Elektor-Labor: Der Entwickler ist der 15-jährige Tüftler Aarav Garg aus Indien, der sich freut, seine „Pocket Weather Station" in unserer Zeitschrift vorstellen zu können. Sie besteht aus einem Arduino Nano, einem 0,96-Zoll-OLED-Display und einem DHT11-Feuchtigkeits- und Temperatursensor. Das Gerät wird von einem LiPo-Akku versorgt, der über ein USB-Lademodul aufgeladen werden kann. Ein maßgeschneidertes 3D-gedrucktes Gehäuse gibt dem handlichen, tragbaren Gerät ein angemessenes Zuhause. Ich habe es im Elektor-Labor nachgebaut und erfolgreich getestet, einige zusätzliche Anleitungen und Anmerkungen in den Textkästen hinzugefügt. Doch lassen wir Aarav selbst zu Wort kommen und erzählen, wie er zu seinem Entwurf kam!

In diesem Artikel erfahren Sie, wie Sie mit einem Arduino Nano-Board eine kleine Wetterstation für die Hosentasche bauen können. Es handelt sich um ein kompaktes Gerät, das Sie überallhin mitnehmen können, und das in der Lage ist, die aktuelle Temperatur und Luftfeuchtigkeit auf seinem OLED-Display anzuzeigen. Dies ist ein großartiges Gerät zum Selbstschutz, da Sie immer wissen, wann Sie einen Schirm mitnehmen müssen, sowohl bei Regen als auch bei sengender Hitze! Das Gerät verfügt über einen eingebauten wiederaufladbaren 160-mAh-Li-Po-Akku. Es ist ein wirklich großartiges Lernprojekt und macht auch noch Spaß.

Schritt 1: Sammeln der Komponenten

Das erste, was man zu Beginn eines Projekts tun sollte, ist die Zusammenstellung der benötigten Komponenten, wie in **Bild 1** dargestellt. Die benötigten Bauteile und Module für dieses Projekt sind:

- Arduino Nano mit Kabel
- Temperatursensor-Modul DHT11
- 0,96"-OLED-Anzeige
- Akku-Lademodul TP4056
- kleiner Akku (ich habe einen 160-mAh-Li-Po-Akku verwendet)
- Schiebeschalter

Und dann stellt man die benötigten Werkzeuge zusammen:

- Lötkolben
- Jumper-Drähte
- Heißklebepistole
- 3D-Drucker für Gehäuse (optional)

Sammeln und/oder kaufen Sie alle benötigten Dinge und gehen Sie über zu

Schritt 2: Anordnen der Komponenten

Jetzt müssen wir die Anordnung aller Komponenten in einem Gehäuse planen. Ich wollte das Gerät so dünn wie möglich halten, so dass man es bequem in die Tasche stecken kann. Daher habe ich alle Komponenten nebeneinander angeordnet, statt sie zu stapeln. Dadurch vergrößern sich natürlich die Breite und die Länge, nicht aber die Höhe des Geräts.

In **Bild 2** ist zu sehen, wie ich die Komponenten in meiner Taschenwetterstation angeordnet habe. Sie können aber auch Ihre eigenen Ideen verwirklichen.

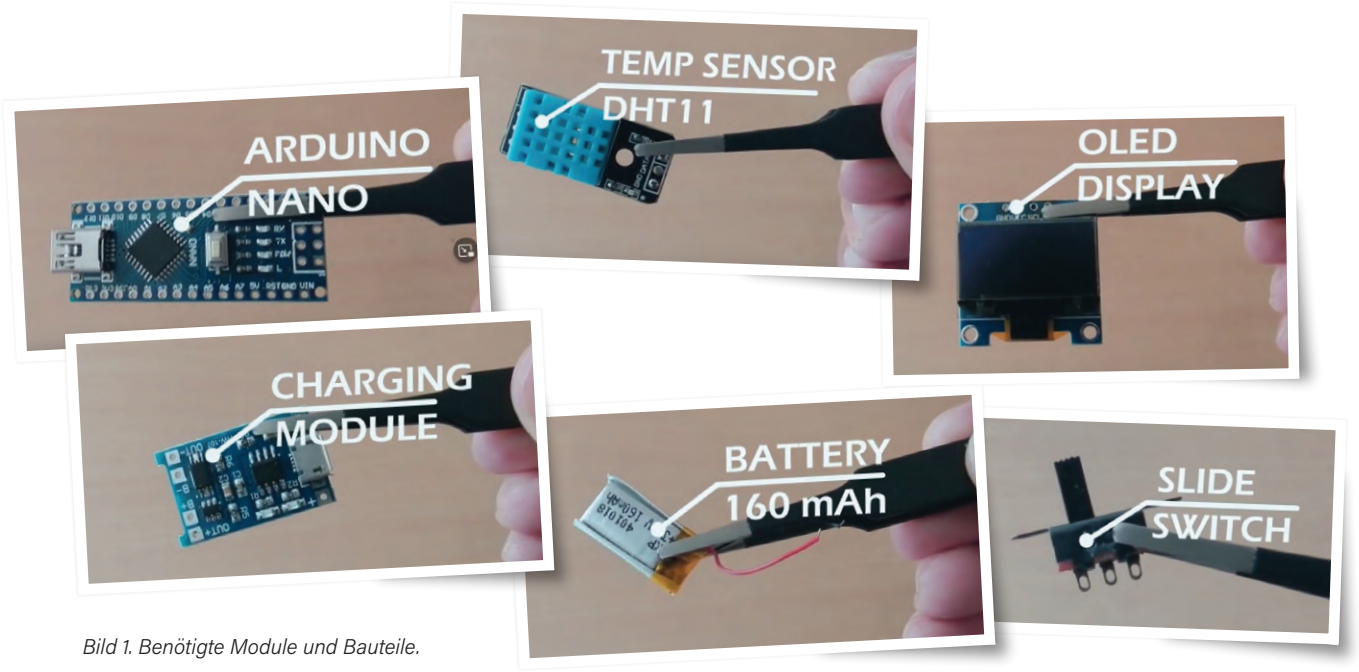


Bild 1. Benötigte Module und Bauteile.

Schritt 3: Schaltplan

Jetzt müssen wir einen Schaltplan für unsere Taschen-Wetterstation entwerfen, was sehr einfach ist, weil nur wenige Module miteinander verbunden werden müssen, ohne dass irgendwelche Modifikationen erforderlich wären. Mein Fritzing-Schaltplan ist in **Bild 3** zu sehen, wenn Sie es genauso machen, können Sie Ihrer Dokumentation eine Kopie hinzufügen. Die Batterie wird mit dem Batterielademodul und der Ausgang des Batterielademoduls mit dem Arduino-Nano-Board verbunden, das sich aufgrund seiner geringen Abmessungen perfekt für dieses Projekt eignet! Als nächstes schließen Sie das Temperatursensormodul und das OLED-Display an das Arduino-Board an.

Schritt 4: Löten/Verbindungen herstellen

Jetzt müssen Sie nur noch in der Realität alle Komponenten genauso verdrahten, wie es der Schaltplan vorgibt. Die Kabel sollten nicht zu kurz und nicht zu lang sein, damit alles passt und Sie ein Wirrwarr von Drähten vermeiden. Löten Sie alles so genau wie möglich, um jede Art von Kurzschluss zu vermeiden. Das mag mühsam sein, aber glauben Sie mir, später werden Sie es der Mühe wert finden. Vergessen Sie nicht die beiden Kabel für den Schiebeschalter, aber löten Sie den Schalter noch nicht an. Nach den Lötarbeiten sollte die Konstruktion in etwa so aussehen wie in **Bild 4**. Fahren Sie einfach mit dem nächsten Schritt fort.

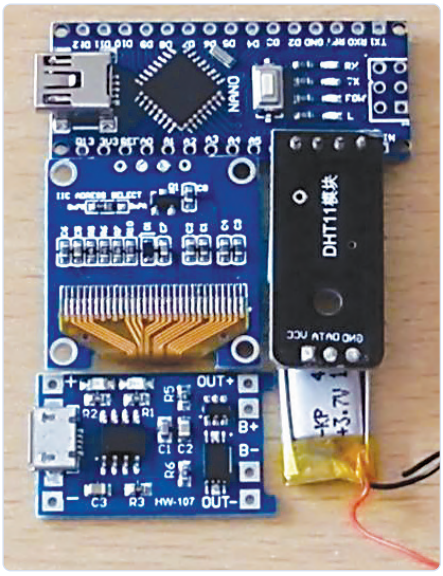


Bild 2. Anordnung der Module.

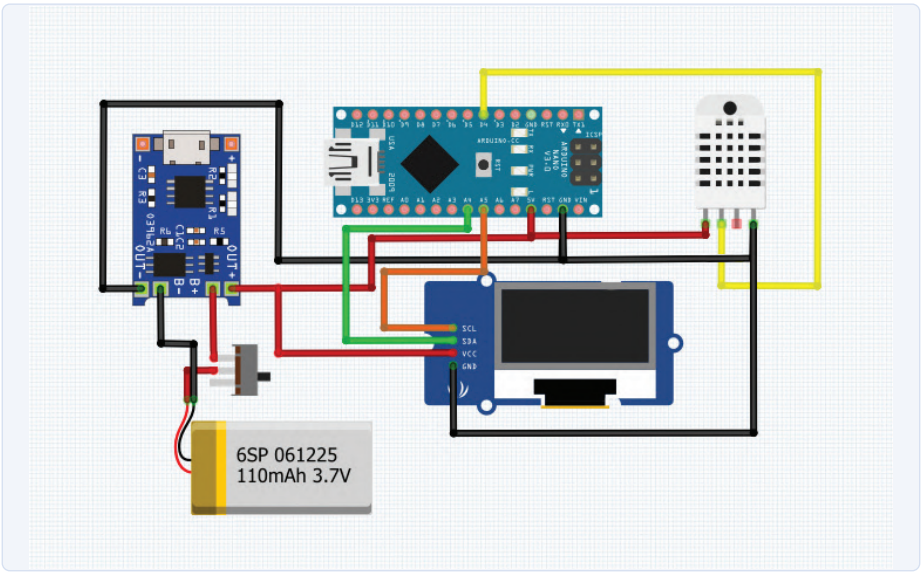


Bild 3. Ein Fritzing-Diagramm mit den Verbindungen.

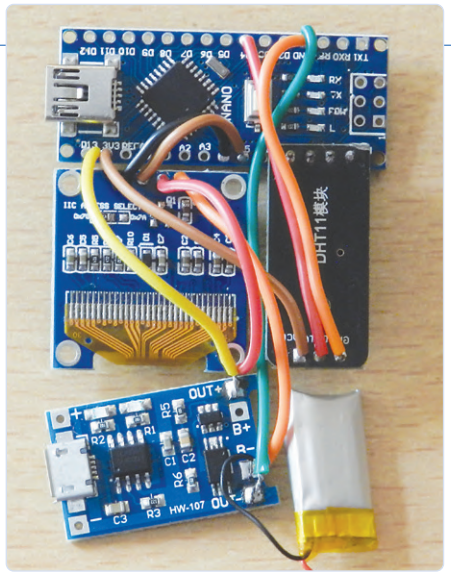


Bild 4. Die meisten Kabelverbindungen sind hier schon angebracht.

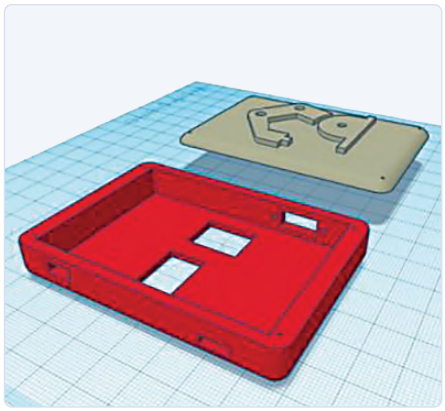


Bild 5. Entwurf des Gehäuses, erstellt in Tinkercad.

Schritt 5: Nach dem Löten

Ich bin mir sicher, Sie wollen dieses verdrahtete Durcheinander nicht so mit sich herumtragen. Es ist ziemlich klar, dass Sie ein Gehäuse für die Taschenwetterstation brauchen, um ihr das richtige professionelle Aussehen zu verleihen. Und die beste Option, ein solches Gehäuse zu konstruieren, ist der 3D-Druck. Gehen Sie also zum nächsten Schritt über, um das Gehäuse zu entwerfen und zu drucken!

Schritt 6: Herstellung des Gehäuses

Ich habe das Gehäuse für die Taschenwetterstation (**Bild 5**) in der erstaunlichen CAD-Software Tinkercad entworfen. Sie unterstützt alle Schwierigkeitsgrade, so dass der Entwurf auch für CAD-Anfänger kein Problem sein sollte. Da ich keinen 3D-Drucker besitze, habe ich meine STL-Dateien bei *IAmRapid* hochgeladen, um ein Angebot zu erhalten, und die Teile sofort bestellt. Das Gehäuse zeigte eine hervorragende Verarbeitungsqualität.

Glücklicherweise erwiesen sich alle Aussparungen, die ich im Entwurf für die verschiedenen Module und Anschlüsse vorgesehen hatte, als 100%-ig perfekt. Die 3D-Dateien für den Druck des Gehäuses stehen auf der Elektor-Labs-Seite für dieses Projekt zum Download bereit [1]. Nun geht es darum, die gesamte Schaltung im Gehäuse zu platzieren und zu befestigen.

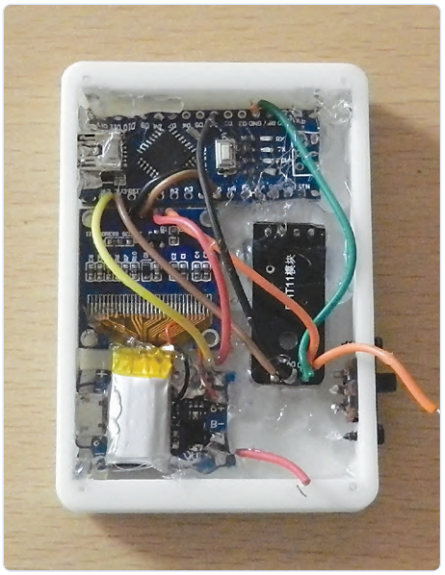


Bild 6. Die Elektronik wird in das Gehäuse „heißgeklebt".

Bild 7. Der Schalter ist an der Außenseite des Gehäuses befestigt.

Schritt 7: Platzierung der Schaltung im Gehäuse

Nun müssen wir die gesamte Schaltung in das Gehäuse einsetzen. Es ist sehr wichtig, dass alle Anschlüsse in die entsprechenden Aussparungen passen, um dem Gerät das dringend gewünschte professionelle Aussehen zu verleihen. Überprüfen Sie auch, ob der noch nicht verdrahtete Schiebeschalter von außen in die Aussparung passt und bohren Sie eventuell zwei kleine Löcher für seine Schraubbefestigung vor. Außerdem ist es wichtig, dass alle Komponenten fest an ihrem Platz sitzen und nicht im Gehäuse umherwackeln können, damit das Gerät einwandfrei und reibungslos funktioniert! Ich habe, wie **Bild 6** zeigt, alles mit etwas Heißkleber fixiert. Wenn Sie damit fertig sind, fahren Sie mit dem nächsten Schritt fort.

Schritt 8: Hinzufügen des Schalters

Nachdem die Schaltung im Gehäuse platziert wurde, ist es an der Zeit, sich um den Schiebeschalter zu kümmern (**Bild 7**). Nachdem Sie den Schalter in seinen Durchbruch gesteckt und mit zwei Schräubchen befestigt haben, bringen Sie die beiden Drähte an, einen von V_{CC} des Arduino-Boards und einen vom positiven Ausgang des Batterielademoduls. Sollten Sie sich nicht trauen, mit dem heißen Lötkolben im Gehäuse zu arbeiten, können Sie den Schalter auch vor der Befestigung „von außen" verdrahten. Wie auch immer, mit dem Schalter können wir die gesamten Spannungsversorgung ein- und ausschalten.

Schritt 9: Schließen des Gehäuses

Jetzt können wir das Gehäuse schließen. Da ich bereits Schraublöcher in das Gehäusedesign eingearbeitet habe, gibt es keine Probleme, mit vier Schrauben den Deckel des Gehäuses (das heißt, die Unterseite) zu befestigen.

Achten Sie darauf, dass der Deckel ordentlich fest sitzt, damit das Gerät auch professionell aussieht und bequem zu tragen ist! Ich habe den Deckel mit einem Logo versehen, um dem Gerät ein ästhetisches und individuelles Aussehen zu verleihen.

Schritt 10: Programmieren!

Jetzt müssen wir etwas sehr Wichtiges tun, nämlich unsere Taschen-Wetterstation programmieren, denn ohne den Code im Arduino, der

Sensordaten

Der DHT11 ist ein in der Maker-Szene wohlbekannter Feuchtigkeits- und Temperatursensor. Er ist nicht der genaueste Sensor, aber erschwinglich und leicht erhältlich. Die Temperatur- und Luftfeuchtigkeitsmesswerte des Sensors werden angezeigt und zur Berechnung des so genannten Hitzeindex [2] verwendet, dem dritten Wert, der unten am Label „Feels“ auf dem LC-Display angezeigt wird. Es handelt sich dabei um die von einer Person gefühlte Temperatur bei einer bestimmten Kombination von Temperatur und relativer Luftfeuchtigkeit. Der Wert wird in der Arduino-DHT11-Bibliothek berechnet. Er ist nicht zu verwechseln mit der bekannteren Windkühle (Chill-Faktor), bei der auch der Einfluss der Windgeschwindigkeit in die Berechnung einfließt.

Der Arduino-Sketch

Bei Projekten wie diesem geht nichts ohne Software, in diesem Fall einen Sketch für den Arduino Nano. Wie für viele gängige Module, integrierte Schaltungen und Sensoren gibt es in der Arduino-Welt fertige Bibliotheken, die dem Programmierer das Leben erleichtern. In Aaravs Sketch werden unter anderem Bibliotheken für die Grafik und die Steuerung des OLED-Displays sowie das Lesen und Verarbeiten der DHT-Sensordaten verwendet. Das folgende Listing enthält die `setup()`- und `loop()`-Funktionen und zeigt, dass nur eine minimale Menge an Code benötigt wird, um die Wetterstation zum Laufen zu bringen. Der größte Teil der Verarbeitung erfolgt nämlich innerhalb der Funktionen, die in den Bibliotheken bereitgestellt werden. Zwei Anweisungen werden benötigt, um die Temperatur und Luftfeuchtigkeit aus dem DHT11 zu lesen, und eine dritte, um den Hitzeindex [2] aus den Sensordaten zu berechnen. Der Rest von `loop()` wird verwendet, um die resultierenden Werte auf dem Display anzuzeigen. Der größere Teil des Sketches, der hier nicht gezeigt ist, wird benötigt, um das Logo des Autors auf dem Display anzuzeigen, wenn das Gerät eingeschaltet oder zurückgesetzt wird. Die Sensordaten werden zu Debugging-Zwecken auch an den seriellen Monitor der Arduino-IDE gesendet.

```
void setup() {
   Serial.begin(9600);
   dht.begin();
  // SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V internally
   if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3C for 128x64
     Serial.println(F(“SSD1306 allocation failed”));
     for(;;); // don’t proceed, loop forever
   }
  testdrawbitmap();    // draw the required image
   delay(1000);
 }
void loop() {
   float h = dht.readHumidity();                    // read humidity
   float t = dht.readTemperature();                 // read temperature
   float hic = dht.computeHeatIndex(t, h, false);   // compute heat index for temp in °C
  // Printing the results on the serial monitor
   Serial.print(“Temperature = “);
  .............
}
```

Wenn etwas nicht funktioniert

Wenn Sie hier sind, dann gehe ich davon aus, dass Sie das Projekt gebaut haben. Entweder funktioniert es, was großartig ist, oder funktioniert nicht, was ebenfalls großartig ist, weil Sie nicht nur wissen, wie Sie etwas bauen, sondern auch, wie Sie es nicht bauen sollen, und diese Art des Lernens ist sehr nützlich.

LED-Display dunkel: Entweder ist das OLED-Display durchgebrannt oder es liegt ein Fehler im Code vor, zum Beispiel wurde die Initialisierung des Displays vergessen oder es wird unter der falschen I²C-Adresse angesprochen. Überprüfen Sie, ob das Display grundsätzlich funktioniert, und wenn dies der Fall ist, verbessern/korrigieren Sie den Code.

Alle Messwerte sind „NA“: Dies kann passieren, wenn der Temperatursensor ein Problem hat. Es könnte sein, dass Sie einen Fehler beim Anschluss des Temperatursensors an das Arduino-Board gemacht haben. Überprüfen Sie einfach die Anschlüsse, und wenn diese in Ordnung sind, liegt wahrscheinlich ein Problem mit dem Sensor selbst vor. Versuchen Sie, den Sensor auszutauschen!

Funktioniert mit USB-Kabel, aber nicht mit Batterie: Wenn das passiert, dann gibt es ein Problem mit der Batterie oder vielleicht mit den Anschlüssen der Batterie!

Hardware, Notizen aus dem Elektor-Labor

Für die kompakte Bauweise der Taschenwetterstation, die Aarav gebaut hat, benötigt man einen Arduino Nano ohne Stiftleisten, aber leider: Die beiden Nano-Boards in meiner Schublade waren schon mit Stiften ausgestattet. Dadurch sind die Module nicht nur zu hoch, um sie in ein flaches Gehäuse einzubauen, es ist auch nicht so einfach, Anschlussdrähte an die Stifte zu löten. Natürlich kann man sich nach einem Modul ohne Stifte umsehen, aber mit ein wenig Vorsicht lassen sich die Stiftleisten entfernen, ohne die Platine zu beschädigen. Schneiden Sie die Stifte mit einem kleinen Seitenschneider bis zum Plastiksockel ab, dann schneiden oder hebeln Sie das Plastik vorsichtig weg. Anschließend löten Sie die Reste der Stifte aus der Platine und entfernen das Lot mit einer Entlötpumpe und/oder Entlötlitze. Die 6-polige ISP-Stiftleiste kann nach der gleichen Methode entfernt werden, auch wenn es etwas schwieriger sein dürfte. Das 0,96"-Display, das ich bestellt habe, besaß auch eine Stiftleiste. Ich habe sie ebenfalls entfernt, um Platz im Gehäuse zu sparen. Bei diesen Displays gibt es vielleicht noch ein anderes Problem: Die meisten Displays sind entweder für eine SPI- oder eine I²C-Schnittstelle konfiguriert, wobei letztere für dieses Projekt benötigt wird. Ich habe den Fehler gemacht, die SPI-Version zu bestellen, und auch wenn es möglich ist, die Schnittstelle dieses Moduls umzukonfigurieren, würde ich das für dieses Projekt nicht empfehlen. Nicht nur, weil es SMD-Lötarbeiten erfordert, sondern auch, weil einige zusätzliche Bauteile und Kabel notwendig sind. Anders als das Fritzing-Diagramm (Bild 3) vermuten lässt, hat Aarav ein DHT11-Modul verwendet, eine Platine, die nicht nur den DHT11 selbst, sondern unter anderem auch einen Pull-up-Widerstand am Datenausgangspin enthält. Der interne Pull-up-Widerstand eines Arduino-Nano-Eingangs kann aber zu hoch sein, um den ordnungsgemäßen Betrieb des bloßen DHT11-Sensors zu gewährleisten, so dass eventuell ein zusätzlicher 10k-Pull-up-Widerstand erforderlich sein kann. Bei meinem eigenen Prototyp war das aber nicht nötig, das Datensignal sah auf einem Oszilloskop schön sauber aus. Auch wenn der Aufbau der Hardware relativ einfach ist, würde ich empfehlen, den Arduino Nano zu programmieren und die Wetterstation zuerst zu testen, bevor die Module in das Gehäuse geklebt werden. Es ist dann einfacher, eventuelle Lötfehler zu korrigieren.

alles steuert, ist unser Gerät nur eine Plastikbox ohne Funktionalität. Programmieren wir also unsere Taschen-Wetterstation so, dass sie funktioniert und möglichst effizient ist. Der Arduino-Sketch kann von der Elektor-Labs-Seite dieses Projekts heruntergeladen werden [1]. Wenn Sie möchten, können Sie auch selbst ans Werk gehen und den Code modifizieren.

Schritt 11: Und los geht's!

Und nun haben wir unsere voll funktionsfähige Taschenwetterstation fertiggestellt, mit einem OLED-Display, damit Sie blau auf schwarz lesen können, wie das Wetter ist. Sie verfügt auch über eine wiederaufladbare Batterie mit einem USB-Ladeanschluss. Die Kapazität der Batterie reicht für eine lange Zeit, so dass sie nur selten wieder geladen werden muss. Außerdem verfügt das Gerät über einen Arduino-Nano-Anschluss zum Hochladen oder Optimieren des Codes in der Zukunft. Der Schalter an der Außenseite ist ebenfalls sehr bequem zu bedienen. Das Gerät passt zudem dank seines kompakten Designs auch in die kleinste Tasche!

210394-02

Ein Beitrag von

Idee, Entwurf und Text: **Aarav Garg**
Illustrationen: **Aarav Garg, Patrick Wielders, Luc Lemmens**
Redaktion: **Luc Lemmens**
Übersetzung: **Rolf Gerstendorf**
Layout: **Giel Dols**

Haben Sie Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Senden Sie eine E-Mail an den Autor unter gargaarav79@gmail.com oder kontaktieren Sie Elektor unter redaktion@elektor.de.

PASSENDE PRODUKTE

- **JOY-iT Nano V3 (SKU 18615)**
  www.elektor.de/18615
- **Blue 0.96" OLED Display I²C 4-pin (SKU 18747)**
  www.elektor.de/18747
- **Buch: *The Ultimate Compendium of Sensor Projects* (SKU 19103)**
  www.elektor.de/19103

WEBLINKS

[1] Elektor-Labs-Seite zu diesem Projekt: https://www.elektormagazine.de/labs/pocket-weather-station
[2] Wikipedia zum Thema Hitzeindex: https://de.wikipedia.org/wiki/Hitzeindex

Lithium-Akkus reparieren

Geld sparen + mehr Power!

Von **Dr. Thomas Scherer**

Wer kennt das nicht? Nach ein paar Jahren macht der Akkuschrauber schlapp, der Akkusauger saugt nicht mehr oder der Rasenmäher-Roboter bleibt mitten auf dem Rasen stehen und gibt auf. Letzteres ist mir gerade passiert. Da könnte man einfach einen Ersatz-Akku ordern – oder aber die Zellen austauschen und so Geld sparen und die Umwelt schonen. Außerdem kann man so gleich mehr Kapazität einbauen, als ursprünglich vorgesehen war.

Gegenüber den früheren NiMH-Akkus haben die mit Lithium-Akkus betriebenen Elektrogeräte eindeutig eine längere Lebensdauer, bis die Energiequelle das Zeitliche segnet. Aber irgendwann ist auch der bravste Lithium-Akku erschöpft und hat sich seinen Platz im Recycling-Nirwana verdient. Das habe ich schon ein paar Mal erlebt, auch und gerade bei Bekannten und Verwandten, die den clan-eigenen Elektroniker um Rat baten, wenn ihr akkubetriebenes Gerät nicht mehr oder immer weniger wollte.

Natürlich könnte man dann kurz nach einem passenden Ersatz-Akku googeln und nach kurzer Wartezeit klingelt der Paketbote. Problem gelöst! Was aber, wenn der Drang zum Selbermachen übermächtig wird? Oder wenn weder Hersteller noch Nachbauer nach ein paar Jahren noch einen mechanisch passenden Ersatz-Akku liefern können? Oder aber wenn man die Gelegenheit nutzen will, einen gepimpten Akku mit erhöhter Kapazität zu bauen? Wie Sie sehen, gibt es mehr als einen Grund für die manuelle Reparatur eines Lithium-Akkus. Hat man sich dafür entschieden (oder keine andere Wahl), muss der Lötkolben angeheizt werden.

Auslöser

In meinem Fall fiel mir Anfang der Woche auf, dass mein Rasenrobbi [1] nur noch eine halbe Stunde mähen wollte, bevor er sich zum erneuten Laden für 1,5 h in seine Ladestation zurückzog. Normalerweise dauerte das Mähen und Laden jeweils eine Stunde. „Zufall?“ fragte ich mich, denn mein Misstrauen war geweckt. Der Robbi war nun knapp über vier Jahre im Dauereinsatz – da kann ein Lithium-Akku schon mal schlapp machen.

Am Nachmittag desselben Tages blickte ich neugierig aus dem Fenster, weil es längere Zeit verdächtig still war. Und siehe da: Mein Roboter stand wirklich still, mitten auf dem Rasen. Ich ging hin: Auf Tastendrücke gab es kein Lebenszeichen. Also schob ich ihn in die Ladestation. Hier erwachte er und nach Eingabe des Passworts klickte ich mich durchs Menü zum Punkt Betriebszeit. Da waren 2.938 Stunden abzulesen. Da dies fast 1.500 Ladezyklen entspricht, musste ich mich wohl demnächst um einen neuen Akku kümmern.

Das „demnächst“ trat schneller ein, als ich angenommen hatte. Nachdem der Robbi am gleichen Tag zunächst mehrere kurze Mäh- und längere Ladezyklen problemlos absolvierte, stand er am nächsten Morgen vor einem kleinen Hügel und tat nichts mehr. Auch das Verbringen in die Ladestation half nicht. Da ich den Rasen vor kurzem gedüngt hatte und wir 2021 ein schön feuchtes Frühjahr hatten, musste schnell Abhilfe geschaffen werden, denn das Gras wuchs mit enormem Tempo.

Akku wechseln?

Kurzes Googeln machte klar: Ein neuer Original-Akku kostete rund 100 €. „Geht doch noch!“ dachte ich mir. Ein Ersatz-Akku eines Drittherstellers war sogar zwischen 50 € und 65 € zu haben. Auch kein schlechter Deal, denn der gesuchte Akku hatte 18 V und eine Kapazität von

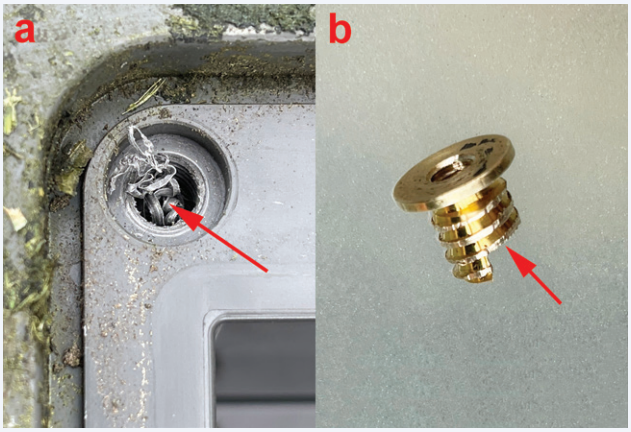


Bild 1. Bei der Fertigung wurde eine Schraube verkantet. Folglich drehte sich die Messing-Einschraubmutter ungewollt aus ihrem Platz. Sie musste mit Epoxidkleber wieder fixiert werden.

2,1 Ah. Also stecken vermutlich fünf Zellen des Formats 18650 drin. Davon wollte ich mich mit eigenen Augen überzeugen, und daher legte ich den Roboter rücklings auf einen Gartentisch und begann, den Deckel des Akkufachs zu lösen.

Doch erstens ist es nicht so einfach, und zweitens als man denkt: Nachdem sich drei von vier Schrauben problemlos lösen ließen, machte eine richtig Stress. Der Akkuschrauber brauchte mehr Drehmoment und bekam sie schließlich raus – allerdings mitsamt Einschraubmutter dran. Bei der Herstellung hatte ein Arbeiter offensichtlich die Edelstahlschraube etwas schräg und mit viel Gewalt in die Gewindegänge der Messingmutter gequält. Das Resultat zeigt **Bild 1a**: In der Bohrung des Kunststoffgehäuses fanden sich viele Späne durch das Rausdrehen der Einschraubmutter. Diese Mutter bekam ich dann nur mit viel Kraft von der Schraube runter. Dabei ist unten rechts etwas Messing abgebrochen (**Bild 1b**).

Da das Gehäuse für den Akku aber wasserdicht sein sollte, konnte ich diese Mutter nicht einfach wieder reindrehen, denn das hätte nicht gehalten. Also habe ich Zweikomponentenkleber angemischt und die Mutter in ihr Loch geklebt. Das Resultat kann man in **Bild 2b** bewundern. Die Messingmutter links oben sitzt so, als wenn es nie anders gewesen wäre. Nach Überwindung dieser kleinen Hürde nun zum Eigentlichen: Bild 2b zeigt ebenfalls den Akku in seiner natürlichen Behausung. Man erkennt die Silhouette von fünf Zellen. Ein Lineal bestätigte meine Vermutung: 5 x 18650. Daneben ist, wie man in **Bild 2c** sehen kann, viel Luft. Sofort kam mir die Idee, die Luft mit mehr Kapazität auszufüllen. Damit war meine Entscheidung gegen den Kauf eines „fertigen" Akkus und für den Austausch der Zellen gefallen.

Neue Zellen

Bild 2a ist übrigens nicht redundant, denn es zeigt, dass die beiden Griffmulden im Deckel so tief sind, dass leider keine zweite Reihe Zellen mehr reinpasst. Also nichts mit zehn Zellen in 5s2p-Anordnung (fünf Zellen in Serie, jeweils zwei parallel) zur schlichten Verdoppelung der Kapazität. Aber es gibt ja (teure) Zellen im Format 18650 mit erhöhter Kapazität. Einschlägige Versender haben Zellen bis 3.500 mAh im Programm. Bei eBay, AliExpress und Co. gibt es sogar Zellen mit noch viel größerer Kapazität, doch von diesen Fantasieangaben sollte man sich nicht blenden lassen, denn HTML ist geduldig. Hochkapazitive Markenzellen hingegen kosten gut 10 € pro Stück, und da hätte ich gleich einen Fertig-Akku kaufen können.

Glücklicherweise entdeckte ich Zellen im etwas ungewöhnlicheren Format 21700. Diese waren nur wenig größer und boten doch deutlich mehr Kapazität fürs Geld. Sie müssten auch irgendwie reinpassen (mir schwebte eine Anordnung vor, die von oben wie ein W ausgesehen hätte). Außerdem kosteten sie mit „Mengenrabatt" und Porto zusammen nur 26 € und das für Exemplare mit 4.000 mAh. Also ab in den Warenkorb damit und auf „Kaufen" geklickt. Zwei Tage später waren sie da – der Rasen hatte mittlerweile bedrohliche Halmlängen erreicht. Es war also höchste Zeit!

Überlegungen

Wer einzelne Zellen zu einem Akkupaket kombinieren will, der braucht neben etwas Fingerfertigkeit und elektrischer Expertise eigentlich etwas zum Punktschweißen. Wie man an der Auswickelzeremonie von **Bild 3** sehen kann, sind die fünf grünen Zellen des alten Akkupacks schön oben und unten mit dünnem Nickelband verbunden, das punktgeschweißt ist. Diese Art der elektrischen Verbindung schont den Akku, da der Wärmeübertrag bei dieser Methode recht gering ist. Löten wird bei Akkuzellen allgemein nicht für gut befunden, da hier mehr Wärme appliziert wird. Doch für die paar Mal Gebrauch im Elektronikerleben ein Punktschweißgerät kaufen? Das wäre für meine Zwecke (und auch die vieler anderer Elektroniker) übertrieben, selbst wenn man Tools bekanntlich nie genug haben kann…

Doch zuerst zum Aufbau des Akkupacks: Das Exemplar von **Bild 3** ist typisch auch für viele andere Geräte. Es besteht aus in Serie

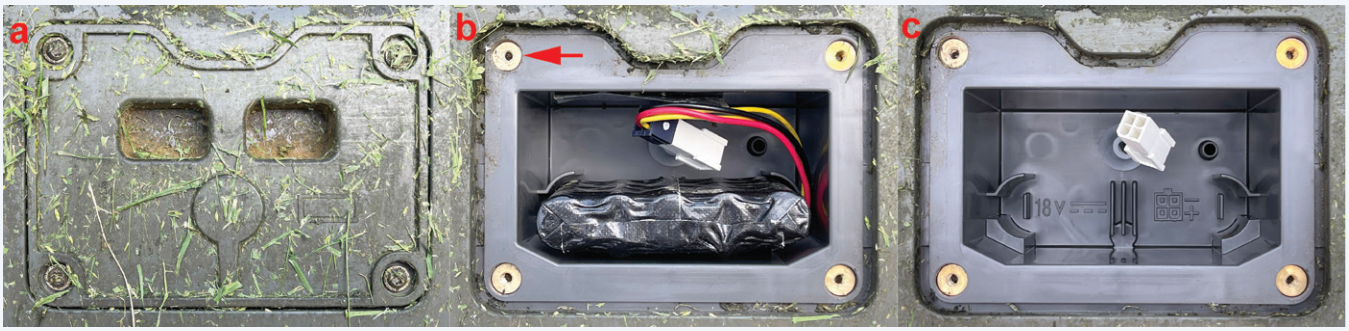


Bild 2. Der Deckel des Akkufachs (2a), der alte Akku im Akku-Fach (2b) und das leere Akku-Fach (2c).

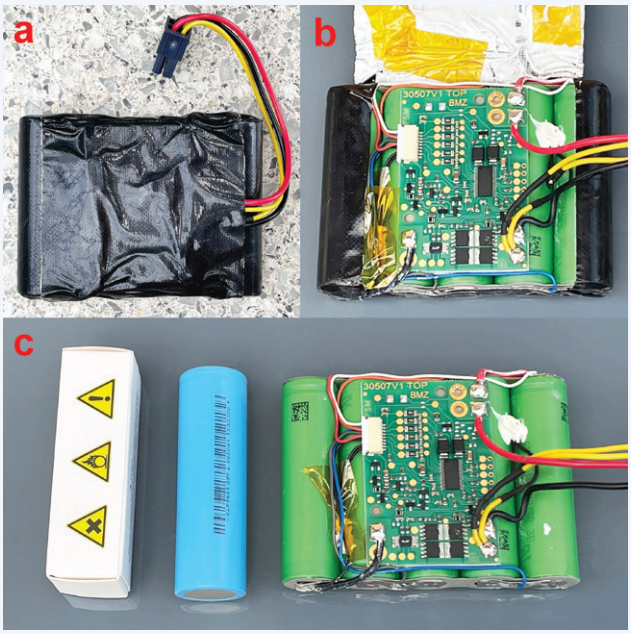


Bild 3. Der alte Akku verhüllt mit schwarzem Gaffer-Tape (3a). Nach dem Lösen der oberen Schicht Klebeband kam die Platine des BMS zum Vorschein (3b). Eine verpackte und eine ausgepackte neue 21700er-Akkuzelle neben dem voll ausgepackten alten Akku aus fünf 18650er Zellen (3c).

Ein BMS als Flipflop

Als ich den ausgepackten Akku wie in Bild 3c vor mir liegen hatte, staunte ich nicht schlecht: An den Zellen direkt gemessen lagen 19,2 V an. Das hatte ich nicht erwartet: Ein voller Akku? War doch etwas anderes defekt und ich hatte voreilig neue Akkuzellen bestellt? Am Stecker des Akkus und an den entsprechenden Anschlüssen rechts der BMS-Platine waren hingegen nur etwa 18,5 V zu messen – allerdings sehr hochohmig. Zwei Finger reichten schon, um die Spannung am Ausgang auf wenige Volt zusammenbrechen zu lassen. Also war das BMS kaputt?
Mit Stirnrunzeln versuchte ich zunächst, den Akku direkt an seinen Polen zu belasten. An 24 Ω waren bei rund 0,75 A immer noch gut 18,2 V zu messen. Ein Ladeversuch über den Stecker klärte das merkwürdige Verhalten auf. Nach nur wenigen Sekunden des Ladens mit 0,5 A blieb auch am Stecker des Akkus die Spannung niederohmig erhalten. Ergo war die Spannung bei einem weichen Akku am Ende seines Lebens wohl zu tief gesunken - das BMS hatte dann abgeschaltet und sich den Zustand einfach gemerkt. Ich konnte das reproduzieren, indem ich den Akku mit 12 Ω belastete. Nach fünf Minuten schaltete das BMS bei knapp über 13 V ab. Einmal Laden und es schaltete den Akku wieder ein.
Das BMS war also nicht defekt (seufz!) und ich konnte den Umbau des Akkus bedenkenlos starten…

geschalteten Akkuzellen plus einem „Battery Management System", kurz BMS. Das ist die Platine in den **Bildern 3b** und **3c**. In letzterem Bild sieht man auch einen Größenvergleich von alten und neuen Akkuzellen. Die Aufgaben eines BMS sind dreifach:

1. Es muss die Zellen „balancieren", das heißt, auf gleicher Spannung beziehungsweise gleichem Ladezustand halten.
2. Es soll verhindern, dass die Zellen mit zu viel Spannung überladen werden können.
3. Es soll die Last bei Unterspannung abschalten, um eine Tiefentladung zu vermeiden.

Hierfür dient der Chip mit den vielen Beinchen; ein spezialisierter Mikrocontroller, der die Zellenspannungen überwacht (links) und die Zellen bei Überspannung und Unterspannung mit Hilfe von 2 x 2 MOSFETs abschaltet (unten). Mehr zum Thema Balancing von Lithium-Akkus findet sich unter [2] und [3].
Kauft man einen neuen Akku, kommt die noch prima funktionierende BMS-Platine in den Müll. Tauscht man hingegen lediglich die Zellen, muss die Platine transplantiert werden, um die neuen Akkuzellen adäquat zu schützen. Im **Kasten Ein BMS als Flipflop?** ist eine zweite Hürde beschrieben, die es zuvor noch zu überwinden galt.

Löten?

In Ermangelung eines passenden kleinen Punktschweißgeräts entschloss ich mich, die Verbindungen in Form von Litze direkt an die Akkupole zu löten. Wenn man weiß, was man tut, klappt das, ohne die Akkus zu schädigen. Da ein potentieller Schaden durch unerwünschte chemische Vorgänge in den Zellen in etwa proportional zum Integral aus Zeit und Temperatur ist, empfiehlt es sich, schnell und zügig zu löten, damit der Lötvorgang schon beendet ist, bevor sich die innere Chemie nennenswert aufzuheizen beginnt. Dabei sind drei Dinge wichtig: Der Lötkolben muss genug Leistung haben, die Temperatur der Lötspitze muss hoch genug sein (um die darin gespeicherte Wärmeenergie in kurzer Zeit zu übertragen) und das Lötzinn muss gut und bei nicht zu hoher Temperatur schmelzen. Mein Lötkolben wird mit 90 W geheizt und lässt sich auf über 400 °C einstellen – das passt. Außerdem sollte man so ein Manöver nicht mit modernem, bleifreiem Lot versuchen, denn das braucht eine höhere Temperatur und benetzt Oberflächen nicht so schön wie das von mir präferierte, gute alte SnPb 60/40. Die Metalloberflächen der Pole von Lithiumzellen lassen sich meiner Erfahrung nach gut löten. Ich brauchte bei einer Lötspitzentemperatur von 385 °C und 1 mm starkem, bleihaltigem Lötzinn mit Flussmittelseele etwa 1 s pro Lötstelle. Das ist schnell genug. Wenn man diese Regeln beherzigt, ist das Löten an Lithium-Zellen ungefährlich. Wenn man aber zu lange (wegen zu schwachem Lötkolben, zu niedriger Temperatur oder bleifreiem Lot) auf dem Metallgehäuse einer Lithium-Zelle herumfuhrwerkt, wird deren Kapazität oder mögliche Zyklenzahl leiden. Im Sekundenbereich wird einem die Zelle nicht gleich um die Ohren fliegen. Alternativ könnte man auch leicht teurere Zellen mit angeschweißten Fahnen ordern und nur an den Enden dieser Metallstreifen löten. Das ist im Prinzip sicherer, aber man

WEBLINKS

[1] Solaranlage für Mähroboter, ElektorMag 7-8/2021: https://www.elektormagazine.de/200553-02
[2] Auto-Balancer, Elektor 5/2010: https://www.elektormagazine.de/magazine/elektor-201005/3506
[3] Intersil: Entmystifizierung von Batteriemanagementsystemen, Elektor Business Magazin 2/2017: https://www.elektormagazine.de/magazine/elektor-11/40430

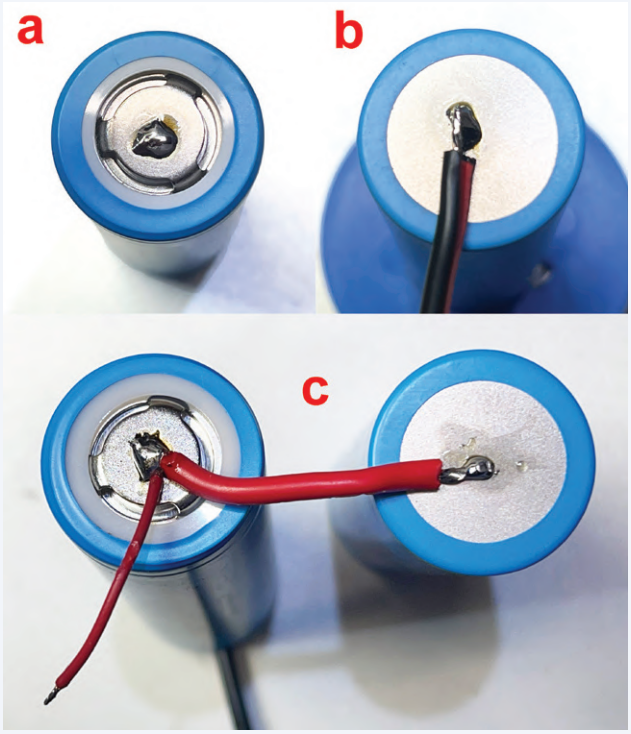


Bild 4. Der Pluspol ist die am wenigsten kritische Stelle zum Löten (4a). An die Minuspole werden zügig kleine Litzenstücke gelötet (4b). Zum Schluss werden die Zellen in Serie geschaltet und dünne Litzen für das Messen der Zellenspannungen durch das BMS angelötet (4c).

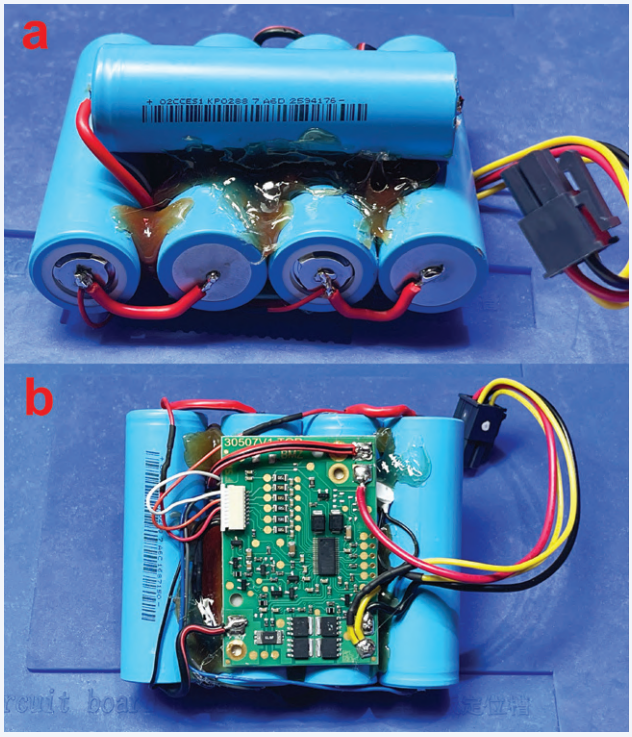


Bild 5. Rückseite des fertigen nackten Akkus mit einer Zelle quer (5a) und Vorderseite mit BMS-Platine (5b).

muss die überstehenden Fahnen dann sehr gut isolieren. Kurzschlüsse sind deutlich gefährlicher als kurzes direktes Löten.

Bild 4 zeigt meine Löttechnik. Der Pluspol hat eine erhabene, schlecht wärmeleitende Kappe, die nur an drei Stellen mit dem eigentlichen Pol verbunden ist. Hier ist die Wärmeempfindlichkeit nicht so hoch. Man bringt also zunächst an allen Pluspolen kleine Lötzinntropfen auf (**Bild 4a**), die man dann später nach Abkühlung zum Blitzlöten benutzt. Auf die Minuspole lötet man direkt kleine Litzenstücke passender Länge (**Bild 4b**). Letzteres kann man in zwei Durchgängen erledigen: Erst alle Minuspole mit einem Lötzinntropfen versehen und nach Abkühlung im zweiten Durchgang die vorher verzinnten Litzenstücke anlöten. Damit man etwas Spielraum hat, können diese etwas länger sein als nötig. Ich habe Litzenstücke von etwa 3 cm Länge mit einem Querschnitt von 1,5 mm² verlötet. Hier sollte man zügig arbeiten. In **Bild 4c** ist zu sehen, dass jeweils ein Pluspol eines anderen Akkus ans andere Litzenende gelötet wurde, um die Zellen in Serie zu schalten. Das dünnere rote Litzenstück führt zu den Messstrippen für die einzelnen Zellenspannungen des BMS.

Zusammenbau und Test

Wie schon angedeutet, hatte ein Zellenpaket in W-Anordnung einfach keinen Platz. Aber man muss nur einmal um die Ecke denken, dann klappt das schon: Wie **Bild 5** zeigt, habe ich einfach vier Zellen nebeneinander gelegt sowie vorsichtig und Stück für Stück mit Heißkleber fixiert. Die fünfte Zelle kam einfach quer auf den Rücken. Dank Heißkleber wird diese Struktur sehr stabil. Wer schonender vorgehen will, nimmt Silikon statt Heißkleber.

Die BMS-Platine hatte auf der Rückseite doppelseitiges Klebeband, was auch auf dem neuen Akkupack gut hielt. Nun mussten nur noch die sechs Leitungen des weißen Steckers (**Bild 5b**) an die jeweiligen Zellenpole, sowie Plus- und Minuspol des Gesamtpacks an die BMS-Platine - dann konnte der neue Akku getestet werden. Beim Zusammenbau sollte man sorgfältig arbeiten und keine Fehler machen – ein paar Fotos vom alten Akku helfen bei Unsicherheiten. Bei mir funktionierte alles direkt und der Akku ließ sich problemlos laden und entladen. Etwas anderes hätte ich auch nicht erwartet ;-).

Zur besseren mechanischen Stabilität, Isolierung und Feuchteresistenz wurde das fertige Akkupaket dann noch mit kräftigem Gaffer-Tape beklebt (**Bild 6**) und das Resultat in den Roboter gesteckt – alles passte. Nach Zuschrauben des Batteriefachs und Einschalten wollte der Rasenroboter sich zuerst mit den Signalen des Leitkabels kalibrieren und dann direkt loslegen. Ich habe Letzteres aber abgebrochen und ihn zunächst in die Ladestation gesteckt. Dort lud er volle drei Stunden – ein Indiz dafür, dass der Akku jetzt fast die doppelte Kapazität des Originals hat.

Der Roboter mäht mittlerweile genauso schön wie die Jahre zuvor: Eine Stunde Mähen gefolgt von einer Stunde Laden. Der Akku wird also nur teilentladen. Dadurch kann ich begründet davon ausgehen, dass die größere Kapazität zu deutlich mehr Ladezyklen führen wird, bis der Akku das nächste Mal schlapp macht. Durch die relativ zur Kapazität kleineren Zyklen sollte er sogar überproportional lange halten. Ich rechne mit gut der doppelten Lebensdauer des Originals. Trifft das zu, hat sich der Aufwand von etwa drei Stunden Basteln gelohnt. Der Stundenlohn ist zwar nicht besonders gut, aber dafür habe ich einen Akku, den man so nicht kaufen kann!

Diese Art des Akkureparierens ist übrigens nicht auf Rasenmäher beschränkt. Sie eignet sich auf ähnliche Weise auch zur Verlängerung der Laufzeit von Akku-Staubsaugern und anderen Geräten, selbst wenn da die Platzverhältnisse nicht ganz so großzügig sind. Der letzte Staubsauger, den ich mit der Methode (aber hochkapazitiven 18650er Zellen) repariert habe, läuft schon drei Jahre ohne Beanstandung.

210368-02

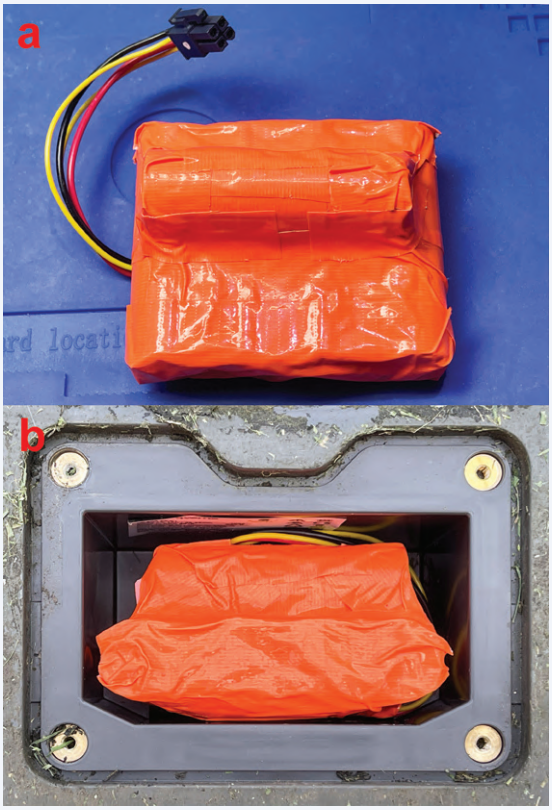


Bild 6. Der mit orangem Gaffer-Tape gesicherte neue Akku (6a); fertig eingebaut ins Akku-Fach (6b).

Sie haben Fragen oder Kommentare?

Bei technischen Fragen können Sie sich gern an die Elektor-Redaktion wenden unter der E-Mail-Adresse redaktion@elektor.de.

Ein Beitrag von

Idee, Durchführung und Text: **Dr. Thomas Scherer**
Redaktion: **Jens Nickel**
Layout: **Giel Dols**

PASSENDE PRODUKTE

- **OWON OW16B Digital-Multimeter mit Bluetooth (SKU 18780)**
  www.elektor.de/18780
- **PeakTech Stromzange 4350 (SKU 18161)**
  www.elektor.de/18161
- **JOY-iT RD6006 Labornetzteil (360 W) (SKU 19564)**
  www.elektor.de/19564

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

Teil 03

GUIs mit Python: Meme-Generator

Erstellen Sie Memes mit selbst-programmierter GUI-App.

MAKER

Laura Sach

Laura leitet das A-Level-Team bei der Raspberry Pi Foundation und erstellt Ressourcen für Schüler, um ihnen die Informatik näher zu bringen.

@CodeBoom

MAKER

Martin O'Hanlon

Martin arbeitet im Lernteam der Raspberry Pi Foundation, wo er Online-Kurse, Projekte und Lernressourcen erstellt.

@martinohanlon

Memes sind Fotos, die sich in den sozialen Medien verbreiten und sich, oft als Satire gedacht, zum Teil auf bekannte Personen beziehen und durch lustige Kommentare einen völlig neuen Sinn erhalten. Schreiben Sie ein Programm mit grafischer Benutzeroberfläche, das Memes erzeugt und auf den vorhergehenden Lektionen basiert.

Geben Sie einen Text und einen Bildnamen ein, und Ihre GUI wird beides zu einem Meme zusammensetzen. Beginnen Sie mit der Erstellung einer einfachen GUI mit einem oberen und einem unteren Textfeld. Dort geben Sie den Text ein, der in Ihrem Bild eingefügt wird. Mit der folgenden Zeile importieren Sie die benötigten Widgets.

```
from guizero import App, TextBox, Drawing
```

Fügen Sie dann diesen Code für die App hinzu:

```
app = App("meme")

top_text = TextBox(app, "top text")
bottom_text = TextBox(app, "bottom text")

app.display()
```

Das Meme wird in einem Drawing-Widget erstellt, das das Bild und den Text enthält.

Ein Meme erstellen

Fügen Sie folgenden Code direkt vor der Zeile `app.display()` ein. Die Höhe und Breite des Bildes sollte so eingestellt werden, dass es die verfügbare Fläche komplett ausfüllt.

```
meme = Drawing(app, width="fill",
height="fill")
```

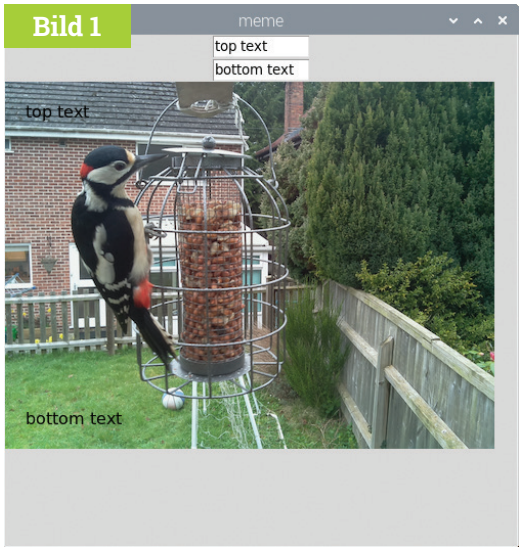
Das Meme wird erzeugt, wenn sich der Text im oberen und unteren Textfeld ändert. Dazu müssen wir eine Funktion erstellen, die das Meme zeichnet.

Die Funktion soll ein Bild einfügen (im Beispiel einen Specht) und den Text oben und unten in das Bild einsetzen.

Erinnern Sie sich, dass Sie **name.value** verwendet haben, um den Wert des Text-Widgets mit dem Nickname in Teil 2 dieser Serie zu bestimmen? Sie können die Eigenschaft `value` aber auch verwenden, um den Wert eines Text-Widgets abzurufen, so dass `top_text.value` in diesem Fall bedeutet: "Bitte frage den Wert ab, der in das Feld `top_text` eingegeben wurde".

```
def draw_meme():
    meme.clear()
    meme.image(0, 0, "woodpecker.png")
    meme.text(20, 20, top_text.value)
    meme.text(20, 320, bottom_text.value)
```

Die ersten beiden Zahlen in `meme.image(0, 0)` und `meme.text(20, 20)` sind die x- und y-Koordinaten, an denen das Bild und der Text gezeichnet werden sollen. Das Bild wird an der linken oberen Ecke angesetzt (Koordinaten 0, 0).
Nun muss noch die Funktion `draw_meme` aufgerufen



▲ **Bild 1** Meme mit vorgegebenem Standardtext.

werden. Fügen Sie folgenden Code direkt vor der Zeile `app.display` ein:

```
draw_meme()
```

Ihr Code sollte nun wie **meme1.py** aussehen. Wenn Sie Ihre App (**Bild 1**) ausführen, werden Sie feststellen, dass eine Textänderung im Meme nicht aktualisiert wird. Dazu müssen Sie Ihr Programm erst so ändern, dass es die Funktion `draw_meme` aufruft, wenn sich der Text ändert. Fügen Sie dazu einen Befehl zu den beiden TextBox-Widgets der App hinzu.

> Erzeugen Sie Ihr individuelles Meme, indem Sie den oberen und unteren Text ändern

```
top_text = TextBox(app, "top text",
command=draw_meme)
bottom_text = TextBox(app, "bottom text",
command=draw_meme)
```

Ihr Code sollte nun wie derjenige in **meme2.py** aussehen. Führen Sie ihn aus und aktualisieren Sie Ihr Meme, indem Sie den oberen und unteren Text ändern.

meme1.py

LADEN SIE DEN KOMPLETTEN CODE HERUNTER:
magpi.cc/guizerocode

▸ Sprache: **Python 3**

```
001.  # Imports ---------------
002.
003.  from guizero import App, TextBox, Drawing
004.
005.
006.  # Functions -------------
007.
008.  def draw_meme():
009.      meme.clear()
010.      meme.image(0, 0, "woodpecker.png")
011.      meme.text(20, 20, top_text.value)
012.      meme.text(20, 320, bottom_text.value)
013.
014.
015.  # App -------------------
016.
017.  app = App("meme")
018.
019.  top_text = TextBox(app, "top text")
020.  bottom_text = TextBox(app, "bottom text")
021.
022.  meme = Drawing(app, width="fill", height="fill")
023.
024.  draw_meme()
025.
026.  app.display()
```

meme2.py

▸ Sprache: **Python 3**

```
001.  # Imports ---------------
002.
003.  from guizero import App, TextBox, Drawing
004.
005.
006.  # Functions -------------
007.
008.  def draw_meme():
009.      meme.clear()
010.      meme.image(0, 0, "woodpecker.png")
011.      meme.text(20, 20, top_text.value)
012.      meme.text(20, 320, bottom_text.value)
013.
014.
015.  # App -------------------
016.
017.  app = App("meme")
018.
019.  top_text = TextBox(app, "top text", command=draw_meme)
020.  bottom_text = TextBox(app, "bottom text", command=draw_meme)
021.
022.  meme = Drawing(app, width="fill", height="fill")
023.
024.  draw_meme()
025.
026.  app.display()
```

Top Tipp

Da die Codezeilen immer länger wurden, haben wir sie auf mehrere Zeilen aufgeteilt, um sie leichter lesbar zu machen. Dies hat keinen Einfluss auf die Funktion des Programms, sondern nur auf sein Aussehen.

Sie können das Aussehen Ihres Memes ändern, indem Sie die Parameter für `color`, `size`, und `font` des Textes ändern. Zum Beispiel:

```
meme.text(
    20, 20, top_text.value,
    color="orange",
    size=40,
    font="courier")
meme.text(
    20, 320, bottom_text.value,
    color="blue",
    size=28,
    font="times new roman",
    )
```

meme3.py

Sprache: **Python 3**

```
001.  # Imports ---------------
002.
003.  from guizero import App, TextBox, Drawing
004.
005.
006.  # Functions -------------
007.
008.  def draw_meme():
009.      meme.clear()
010.      meme.image(0, 0, "woodpecker.png")
011.      meme.text(
012.          20, 20, top_text.value,
013.          color="orange",
014.          size=40,
015.          font="courier")
016.      meme.text(
017.          20, 320, bottom_text.value,
018.          color="blue",
019.          size=28,
020.          font="times new roman",
021.          )
022.
023.
024.  # App -------------------
025.
026.  app = App("meme")
027.
028.  top_text = TextBox(app, "top text", command=draw_meme)
029.  bottom_text = TextBox(app, "bottom text", command=draw_meme)
030.
031.  meme = Drawing(app, width="fill", height="fill")
032.
033.  draw_meme()
034.
035.  app.display()
```

Ihr Code sollte nun wie **meme3.py** aussehen. Probieren Sie verschiedene Stile aus, bis Sie etwas finden, das Ihnen gefällt (**Bild 2**).

Individuelle Anpassung

Für einen wirklich interaktiven Meme-Generator sollte der Benutzer in der Lage sein, Schriftart, Größe und Farbe selbst einzustellen. Dazu können Sie zusätzliche Widgets auf der Benutzeroberfläche bereitstellen. Für die Farbe und die Schriftart könnten Sie zum Beispiel eine Dropdown-Liste, auch Combo-Box genannt, verwenden. Die Schriftgröße wiederum könnte zum Beispiel mit einem Slider-Widget eingestellt werden. Ändern Sie zunächst Ihre Import-Anweisung, um die Combo- und Slider-Widgets einzubinden.

```
from guizero import App, TextBox, Drawing,
Combo, Slider
```

So erstellen Sie ein neues Combo-Widget zur Auswahl der Farben.

```
bottom_text = TextBox(app, "bottom text",
command=draw_meme)
color = Combo(app,
    options=["black", "white", "red",
"green", "blue", "orange"],
    command=draw_meme)
```

> Die Optionen werden in der Reihenfolge angezeigt, in der sie in die Liste eingetragen wurden

Der Parameter `options` legt fest, welche Farben der Benutzer aus der Combo-Box auswählen kann. Jede Farbe ist ein Element in einer Liste. Sie können auch beliebige andere Farben in die Liste aufnehmen.
Die Optionen werden in der Reihenfolge angezeigt, in der Sie sie in die Liste setzen. Die erste Option ist die Standardoption, die zuerst angezeigt wird. Farben, die von der Standardeinstellung abweichen, können über den ausgewählten Parameter (`selected`) abgerufen werden, z. B. `"blue"`.

▲ **Bild 2** Ändern der Schriftarten und Schriftfarben

```
color = Combo(app,
    options=["black", "white", "red",
"green", "blue", "orange"],
    command=draw_meme,
    selected="blue")
```

Jetzt kann eine Farbe ausgewählt werden. Als nächstes müssen Sie die Funktion `draw_meme` so ändern, dass der Wert von Combo verwendet wird, wenn Sie den Text in Ihrem Meme erstellen. Zum Beispiel:

```
meme.text(
    20, 20, top_text.value,
    color=color.value,
    size=40,
    font="courier")
```

Führen Sie dasselbe für den unteren Textblock aus. Ihr Programm sollte nun **meme4.py** ähneln.

Fügen Sie nach den obigen Schritten eine zweite Combo-Box zu Ihrer Anwendung hinzu, damit der Benutzer eine Schriftart aus dieser Liste von Optionen auswählen kann: [`"times new roman"`, `"verdana"`, `"courier"`, `"impact"`]. Denken Sie daran, die Funktion `draw_meme` so zu ändern, dass beim Hinzufügen des Textes der Wert der Schriftart (font) verwendet wird.

Erstellen Sie nun ein Slider-Widget, um die Größe des gewünschten Textes einzustellen.

meme4.py

▸ Sprache: **Python 3**

```
001.  # Imports ---------------
002.
003.  from guizero import App, TextBox, Drawing, Combo, Slider
004.
005.
006.  # Functions -------------
007.
008.  def draw_meme():
009.      meme.clear()
010.      meme.image(0, 0, "woodpecker.png")
011.      meme.text(
012.          20, 20, top_text.value,
013.          color=color.value,
014.          size=40,
015.          font="courier")
016.      meme.text(
017.          20, 320, bottom_text.value,
018.          color=color.value,
019.          size=28,
020.          font="times new roman",
021.          )
022.
023.
024.  # App -------------------
025.
026.  app = App("meme")
027.
028.  top_text = TextBox(app, "top text", command=draw_meme)
029.  bottom_text = TextBox(app, "bottom text", command=draw_meme)
030.
031.  color = Combo(app,
032.                options=["black", "white", "red", "green",
      "blue", "orange"],
033.                command=draw_meme, selected="blue")
034.
035.  meme = Drawing(app, width="fill", height="fill")
036.
037.  draw_meme()
038.
039.  app.display()
```

Drawing-Widget

Das Drawing-Widget (Zeichnen) ist sehr vielseitig und kann zur Anzeige vieler verschiedener Formen, Muster und Bilder verwendet werden. Lesen Sie dazu auch Anhang C des Buches **magpi.cc/pythongui** oder werfen Sie einen Blick in die Online-Dokumentation: **lawsie.github.io/guizero/drawing**.

Python 3 Programmierung und GUIs

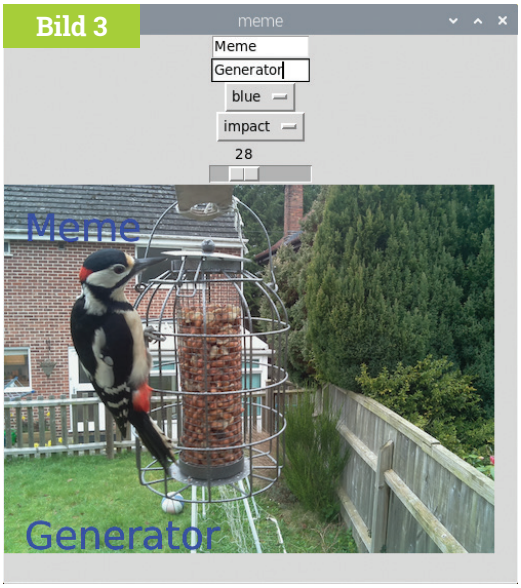
Dies ist die zweite überarbeitete und aktualisierte Auflage eines Buches, das sich an Ingenieure, Wissenschaftler und Maker wendet, die PCs mittels grafischer Benutzeroberflächen mit Hardware-Projekten verbinden wollen, wobei sowohl Desktop- als auch webbasierte Anwendungen nicht zu kurz kommen. Als Programmiersprache wird Python 3 verwendet, eine schnelle – und eine der populärsten Sprachen überhaupt.
Mit diesem Buch lassen sich praktische Entwürfe leicht realisieren. Ein Texteditor ist alles, was zur Erstellung von Python-Programmen erforderlich ist. **www.elektor.de/python-3-programming-and-guis**

04-meme-generator.py

Sprache: **Python 3**

```
001.  # Imports ---------------
002.
003.  from guizero import App, TextBox, Drawing, Combo, Slider
004.
005.
006.  # Functions -------------
007.
008.  def draw_meme():
009.      meme.clear()
010.      meme.image(0, 0, "woodpecker.png")
011.      meme.text(
012.          20, 20, top_text.value,
013.          color=color.value,
014.          size=size.value,
015.          font=font.value)
016.      meme.text(
017.          20, 320, bottom_text.value,
018.          color=color.value,
019.          size=size.value,
020.          font=font.value,
021.          )
022.
023.
024.  # App -------------------
025.
026.  app = App("meme")
027.
028.  top_text = TextBox(app, "top text", command=draw_meme)
029.  bottom_text = TextBox(app, "bottom text", command=draw_meme)
030.
031.  color = Combo(app,
032.                options=["black", "white", "red", "green",
      "blue", "orange"],
033.                command=draw_meme, selected="blue")
034.
035.  font = Combo(app,
036.               options=["times new roman", "verdana", "courier",
      "impact"],
037.               command=draw_meme)
038.
039.  size = Slider(app, start=20, end=50, command=draw_meme)
040.
041.  meme = Drawing(app, width="fill", height="fill")
042.
043.  draw_meme()
044.
045.  app.display()
```

Bild 3



▲ **Bild 3** Der fertige Meme-Generator.

```
  size = Slider(app, start=20, end=40,
command=draw_meme)
```

Der Bereich des Schiebereglers wird mit den Parametern `start` und `end` festgelegt. In diesem Beispiel hat also der kleinste verfügbare Text die Größe 20 und der größte Text die Größe 40.

Ändern Sie die Funktion `draw_meme` so, dass der Wert aus dem Slider für die Textgröße verwendet wird.

```
  meme.text(
      20, 20, top_text.value,
      color=color.value,
      size=size.value,
      font=font.value)
```

Ihr Code sollte nun demjenigen in **04-meme-generator.py** ähneln. Wenn Sie ihn auszuführen, werden Sie ein Ergebnis erhalten, das ähnlich aussieht wie in **Bild 3** dargestellt.

Können Sie die grafische Oberfläche so ändern, dass der Name der Bilddatei in eine TextBox eingegeben oder vielleicht aus einer Liste in einer Combo-Box ausgewählt werden kann? Das würde Ihre Anwendung in die Lage versetzen, auch Memes mit verschiedenen, auswählbaren Bildern zu erzeugen.

Anmerkung: Dieser Artikel erschien zuerst im MagPi Magazin. Elektor gibt MagPi in niederländischer, französischer und deutscher Sprache heraus.

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:

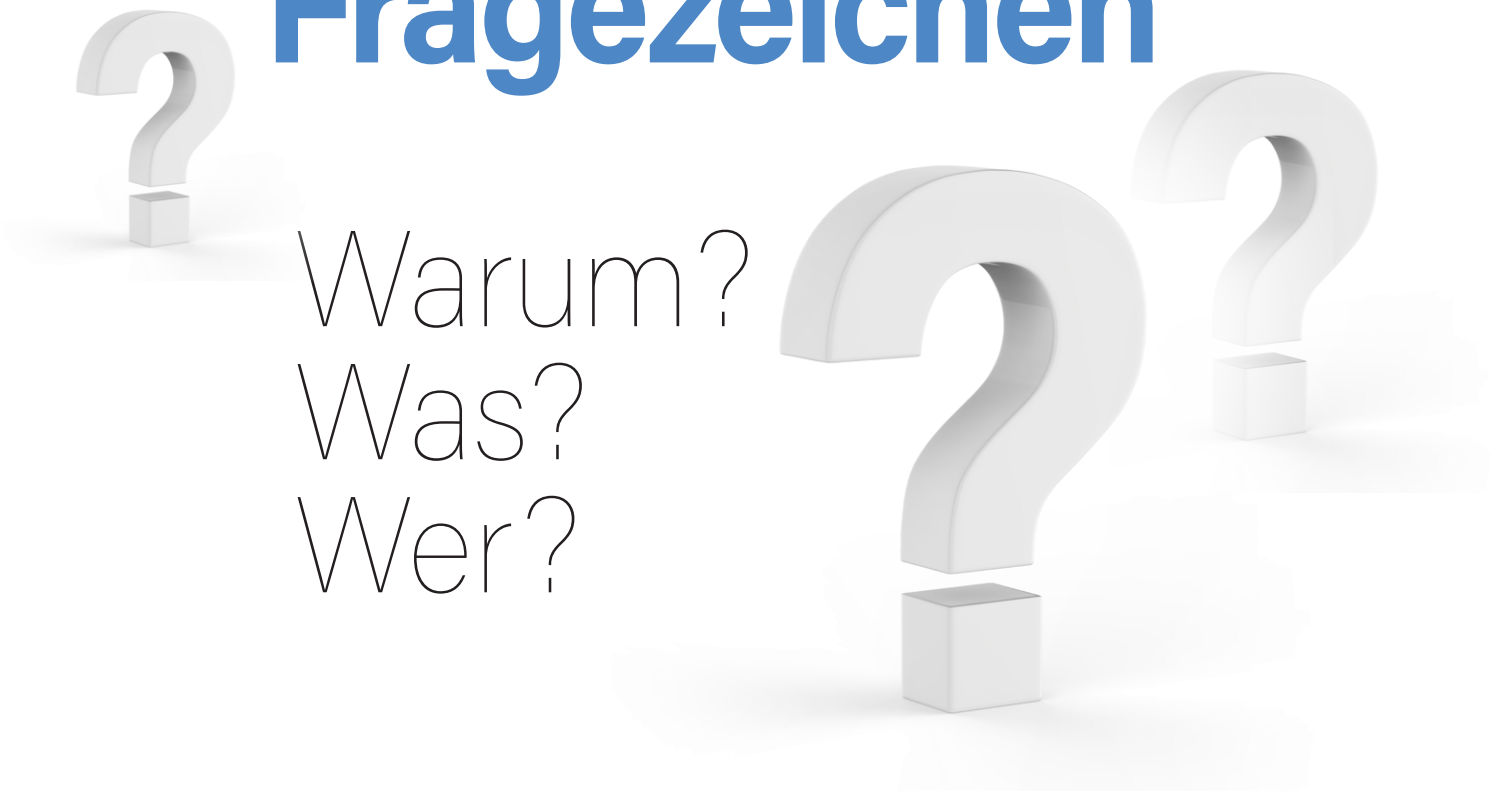
- 6 gedruckte Ausgaben der MagPi
- ein Willkommensgeschenk
- Zugang zu dem Online Archiv mit allen MagPi Ausgaben seit 2016

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

Die drei Fragezeichen



Warum?
Was?
Wer?

Von **Priscilla Haring-Kuipers** (Niederlande)

Die Begleitumstände können sehr unterschiedlich sein, ob man nun ein medizinisches Gerät, ein Musikinstrument oder ein fabelhaftes IoT-Objekt baut. Aber ich denke, die Fragen sind dieselben.

Warum bauen Sie es? Hat es einen ausreichenden Zweck, um seine Existenz zu rechtfertigen? Was würde passieren, wenn dieser Gegenstand nicht (von Ihnen) hergestellt würde?
Was, bezüglich der Materialien und Verfahren, die bei seiner Herstellung zum Einsatz kommen, wird benötigt? Was geschieht mit dem Material während und am Ende der Lebensdauer des Objekts?
Wer sind die Menschen, die an der Herstellung beteiligt sind, und wie werden sie behandelt? Was bedeutet Ihr Produkt für die Menschen, die es später einmal haben werden? Was bedeutet es für Sie, es herzustellen?

Lassen Sie mich als Herstellerin von Boutique-Synthesizern in diese Fragen eintauchen.

Warum

Ich glaube, dass jeder, der etwas baut, ein wenig (oder viel) von einem künstlerischen Bedürfnis hat, etwas zu schaffen. Oftmals entstehen die Gegenstände, die wir produzieren, aus dem Bedürfnis heraus, diesen einen bestimmten Sequenzer zu bauen, der geradezu danach schreit, gefertigt zu werden, oder einen anderen Ansatz für einen Low-Frequency-Oszillator, der in der Welt eindeutig fehlt. Die Befriedigung dieses Bedürfnisses ist an sich schon wertvoll, *wenn* man glaubt, dass Kunst und Kreativität einen Wert haben.
Für mich wäre dieses Bedürfnis allein Grund genug, ein Gerät herzustellen, aber nicht Grund genug, 100 Geräte zu produzieren. Sobald sich etwas aus diesem *Bedürfnis* heraus entwickelt hat, fragen wir uns, ob es der Welt etwas hinzufügt. Wir fällen unsere eigenen, sehr subjektiven Urteile, indem wir uns ansehen, was auf dem Markt der modularen Synthesizer erhältlich ist, und stellen Hypothesen auf, ob unser Gegenstand „genug anders“ ist, um von Bedeutung zu sein. Wir fragen uns auch, ob wir über die richtigen Fähigkeiten verfügen, damit unser Gegenstand den beabsichtigten Zweck erfüllt. Da wir die Einzigen sind, die es überhaupt interessiert, ob dieser Gegenstand hergestellt werden soll, lautet die Antwort in der Regel „ja“, wenn wir ihn technisch so bauen können, dass dieser Zweck erfüllt wird.

Was

Dann beginnen wir zu überlegen, was es bedeuten würde, dieses Ding zu bauen. Wir denken über die damit einhergehenden Materialien und Verfahren nach und erwägen unsere Möglichkeiten. Hier legen wir auch - fast unmerklich - fest, welche Auswahl wir moralisch zu akzeptieren bereit sind. Wir

sind uns bewusst, dass wir nicht alles tun, was wir könnten. Wir haben keine ethischen Entscheidungen bezüglich der Rohstoffe oder Komponenten getroffen. Wir würden gerne besser informierte Entscheidungen treffen, aber wir akzeptieren und arbeiten mit dem, was uns leicht zugänglich ist.

Unsere erste Antwort auf die Frage der Nachhaltigkeit besteht darin, etwas Langlebiges zu bauen. Das ist die Richtschnur für die Wahl der Materialien. Wir erwarten, dass unsere Produkte eine Lebenserwartung von zwanzig bis einhundert Jahren haben, also muss alles an diesem Gegenstand so lange halten oder zumindest ersetzt werden können. Dies gilt auch für die elektronischen Produkte, die wir herstellen. Musikinstrumente werden geliebt, gewartet und repariert. Wir erleichtern dies, indem wir unsere Entwürfe öffentlich machen und Kunden und Musikgeschäfte dabei unterstützen, Gegenstände selbst zu reparieren, wenn sie dazu in der Lage sind, oder sie zur Reparatur an uns zurückzuschicken.

Wir haben uns entschieden, sowohl Aluminium als auch Bambus zu verwenden, wo immer wir können. Aluminium wird bereits sehr häufig recycelt. Unsere Frontplattenfabrik verwendet eine Mischung aus etwa 50 % recyceltem Aluminium, was in der Branche ziemlich üblich ist, und wieder recycelt werden kann. Bambus hat die Eigenschaften eines harten Holzes, ohne dass dafür Wälder abgeholzt werden müssen. Beide Materialien sind sehr haltbar und ästhetisch ansprechend.

Die Wiederverwertung unserer Produkte wurde noch nicht richtig bedacht, da wir noch nicht lange genug dabei sind. Die Abfälle, die bei der Herstellung unserer Prototypen anfallen, werden bei den entsprechenden örtlichen Recyclingstellen entsorgt.

Wer

An der Herstellung unserer Produkte sind überall Menschen beteiligt - auch wir selbst. Diejenigen, von denen wir am wenigsten wissen, sind in der Kette am weitesten von uns entfernt. Wir wissen nicht, wer unsere Rohstoffe abbaut oder wer sie zu den Komponenten verarbeitet, die wir verwenden. Einige Bauteile kaufen wir direkt von Fabriken in Taiwan, und wir nehmen an, dass sie dort hergestellt werden, aber wir haben eigentlich keine Ahnung. Unsere Platinen kommen aus Shenzhen, wo wir mehrere Platinenfabriken und Montagefirmen besucht haben (von denen eine unserer Vermutung nach eine Vorzeigefabrik ist). Die Arbeitsbedingungen haben sich zwar verbessert, sind aber immer noch weit von dem entfernt, was wir in der Europäischen Union für akzeptabel halten würden. Wir haben uns entschlossen, den größten Teil unserer Montage in den Niederlanden durchführen zu lassen, auch weil wir die Arbeitsbedingungen hier als ethisch betrachten. Wir haben auch diese Fabriken besucht und mit den Frauen, die unsere Elektronik herstellen, Tee getrunken. Wir mutmaßen, wie andere Menschen unseren Gegenstand verwenden würden. Wir glauben, dass das Erzeugen von Klängen, das Musizieren, das Aufführen von Musik, das Zuhören und das Tanzen zu Musik alles würdige Beschäftigungen sind, die die Welt zu einem besseren Ort machen. Wir freuen uns, wenn wir das Gefühl haben, dass wir mit unseren Produkten einen Beitrag dazu leisten können, und wir wollen die Kreativität beim Musikmachen fördern, wo immer wir können. Dafür entwerfen wir.

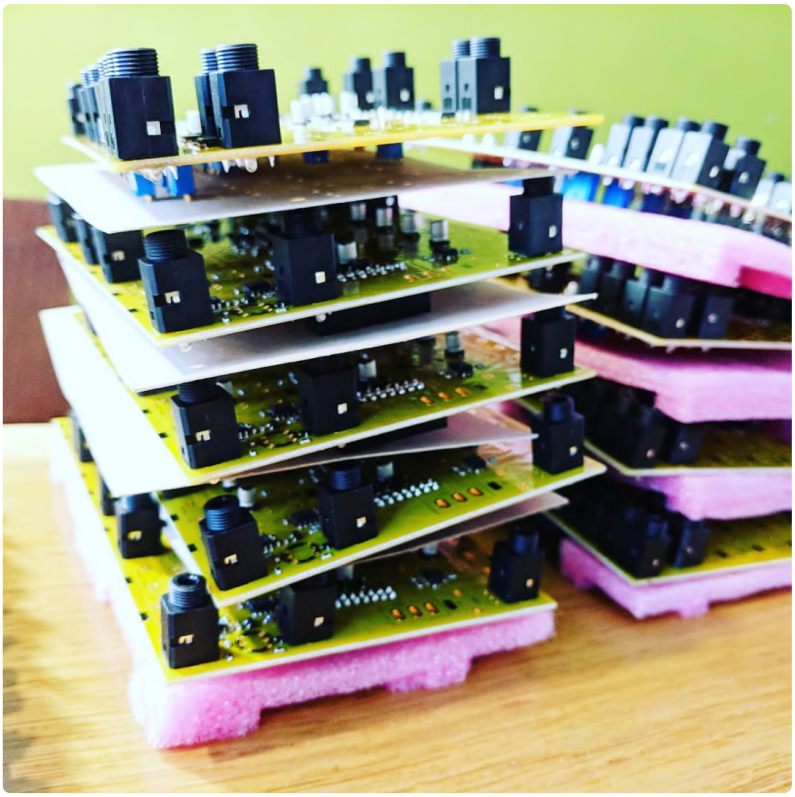


Bild 1. Platinen für den RectangularThing-Synthesizer auf Halde.

Wir wollen mit den Nutzern unserer Gegenstände auf eine Weise in Verbindung sein, die auch unser Leben bereichert.

Dieses ganze Puzzle muss am Ende wirtschaftlich machbar sein. Wenn wir diesen Gegenstand herstellen, von dem wir sicher sind, dass wir ihn herstellen wollen, mit den Materialien, die wir verwenden wollen, und unter Einbeziehung der Leute, mit denen wir ihn herstellen wollen, ist es dann wahrscheinlich, dass die Leute unseren Gegenstand zu dem Preis kaufen wollen, auf den wir kommen? Nur wenn die Antwort auf all diese Fragen „Ja“ lautet, beginnen wir einen Produktionslauf, in der Regel von 100 Stück.

Stellen Sie sich die Fragen nach dem Warum, Was und Wer, wenn Sie etwas bauen wollen? Bitte teilen Sie Ihre Überlegungen mit: redaktion@ elektor.de

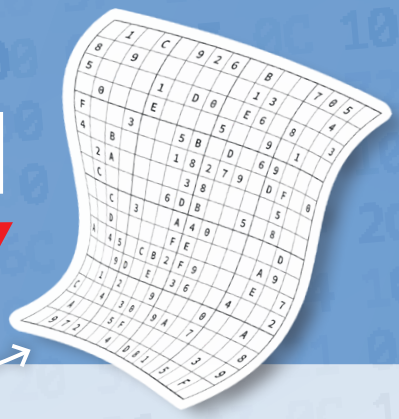
210663-02

Das World Ethical Electronics Forum

Ziel des World Ethical Electronics Forum (WEEF) ist es, globale Innovatoren der Elektronikbranche durch eine offene Diskussion über Ethik und nachhaltige Entwicklungsziele zu inspirieren. Die Gründungsveranstaltung des WEEF fand am 18. November 2021 in München statt. Besuchen Sie die WEEF-Webseite – **www.elektormagazine.com/weef** – für Details zur Veranstaltung und um Neues über das WEEF 2022 zu erfahren.

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 15. April 2022.

DIE GEWINNER DES HEXADOKUS AUS DER AUSGABE JANUAR/FEBRUAR STEHEN FEST!

Die richtige Lösung ist: **FA028**

Aus allen Einsendungen mit der richtigen Lösung haben wir die fünf Gewinner eines Elektor-Wertgutscheins über je 50 € gezogen.

Die Namen der Gewinner werden unter www.elektormagazine.de/hexadoku bekannt gegeben.

Herzlichen Glückwunsch!

E	2	6			3	C			5	7			4		D
4						5		E			6	A			
				E	F			C		8	A	3	6		9
		1	A	2		7			D				C		E
A				5			8	4	6		9		7		
	D	B							7		5	C			
	5					A	B		3				0	D	
					7		0	1		B		2			
					A		F	B		1		7			
	6					8	E		4				D	0	
	4	5							A		E	6			
B				7			4	9	0		D		1		
		3	2	8		6			E				9		B
				F	E			7		5	8	D	2		3
C						2		A			0	8			
D	9	8			1	B			F	2			E		4

2	4	7	E	3	D	B	F	A	5	C	0	9	6	8	1
8	1	D	F	C	9	0	2	B	7	E	6	3	A	4	5
B	0	9	5	A	4	1	6	D	F	3	8	7	C	E	2
C	A	3	6	5	7	8	E	9	1	2	4	F	B	D	0
3	F	0	7	1	8	2	C	E	A	5	9	B	4	6	D
9	2	1	A	4	3	7	5	6	C	B	D	8	E	0	F
4	5	8	B	6	E	9	D	F	0	1	7	A	2	3	C
6	D	E	C	B	F	A	0	2	8	4	3	5	7	1	9
E	6	B	3	7	A	F	9	C	4	0	1	D	5	2	8
A	7	5	0	D	C	E	1	3	2	8	B	6	F	9	4
D	9	F	2	8	0	3	4	5	E	6	A	C	1	B	7
1	8	C	4	2	5	6	B	7	9	D	F	E	0	A	3
5	3	2	D	9	B	4	A	0	6	7	C	1	8	F	E
F	B	A	1	0	2	5	7	8	3	9	E	4	D	C	6
0	C	4	9	E	6	D	8	1	B	F	5	2	3	7	A
7	E	6	8	F	1	C	3	4	D	A	2	0	9	5	B

Der Rechtsweg ist ausgeschlossen. Mitarbeiter der in der Unternehmensgruppe Elektor International Media B.V. zusammengeschlossenen Verlage und deren Angehörige sind von der Teilnahme ausgeschlossen.

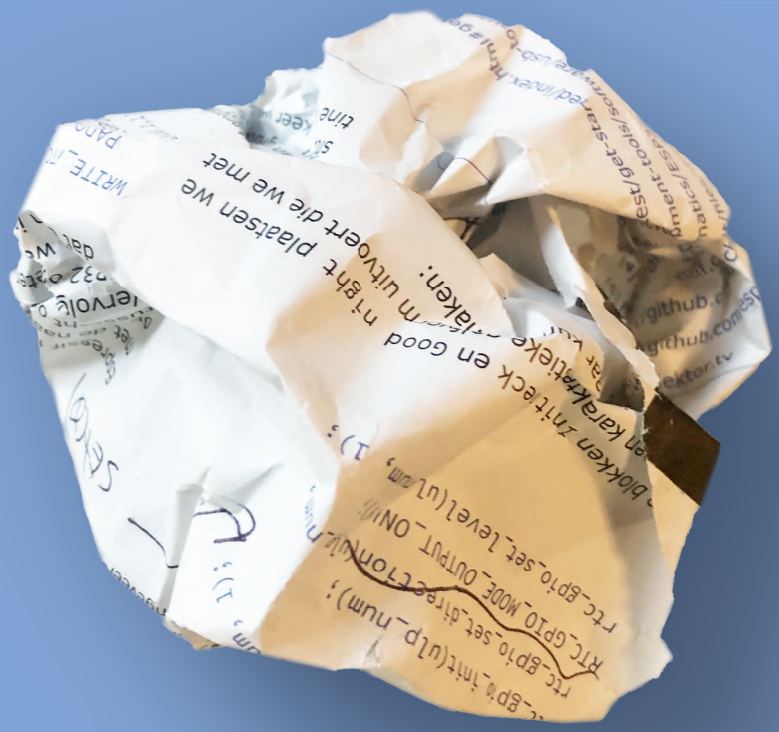
Development Tools alle an einem Ort

Tausende Tools von hunderten zuverlässigen Herstellern

Wählen Sie Ihr Produkt aus unserer breiten Palette auf **mouser.de/dev-tools**

MOUSER ELECTRONICS

Grrr. Stecken geblieben?



Sie finden unsere Elektor Projekte richtig klasse, kommen aber aus irgendeinem Grund nicht weiter und benötigen Hilfe? Oder haben Sie gar eine Idee oder einen Kommentar zu einem bestimmten Artikel?

Kein Problem. Unsere Elektor-Ingenieure, -Redakteure und Community-Mitglieder sind auch in den **sozialen Medien aktiv**.

Sie können uns hier finden:

www.elektor.de/FB

www.elektor.de/TW

www.elektor.de/YT

www.elektor.de/insta

www.elektor.de/LI

elektor
design > share > sell