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

elektor MAG
design › share › sell

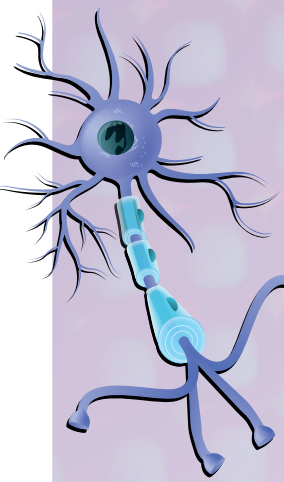
581
* SEIT 1961 *

SEPTEMBER/OKTOBER 2021
ELEKTORMAGAZINE.DE

FOTOGRAFIEREN UND VIDEO-STREAMING

mit dem Raspberry Pi 4

S. 50



Neue Serie:

Neuronen in neuronalen Netzwerken

S. 6

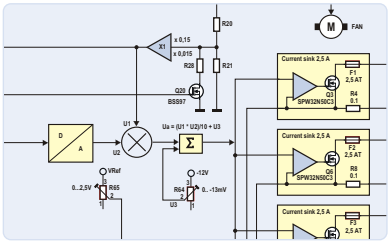
EILD-VERARBEITUNG mit dem Nvidia Jetson Nano

S. 30

60 years young

In dieser Ausgabe

und vieles mehr!



Elektronische Last für DC und AC
Bis zu 400 V und 10 A (Peak)
S. 20

Balkonkraftwerk
Selbst installiert = schnell amortisiert!
S. 44

ESP32-verbundenes Thermostat
Lagern Sie Ihren Wein bei der richtigen Temperatur!
S. 84

4 198630 314904 10010

UNSERE ANGEBOTSABTEILUNG IHR PERSÖNLICHER PREISVERGLEICHER

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

Holen Sie jetzt schnell & einfach mehr aus Ihrem Budget

Sie benötigen größere Mengen, wollen Ihr Einkaufsvolumen konzentrieren oder benötigen Produkte, die wir nicht im Programm führen? Wir vergleichen gern für Sie die besten Quellen am Markt und unterbreiten Ihnen gleich mit dem ersten Angebot den bestmöglichen reichelt Preis.

TRMS Digital-Multimeter, UT 161-Serie

Die preisgünstigen Allrounder

Einfache Bedienung, wichtige Grundmessfunktionen somit ausgelegt für Hobby, Schulung, Studium bis hin in den Bereich der Entwicklung. Dank USB lässt sich das Multimeter an den Computer anschließen und fernsteuern. Außerdem können Sie so einfach Messwerte archivieren und als XLS-Datei zur weiteren Verwendung exportieren.

- Spannung: 1000 V AC/DC
- Strom: 0-10 A AC/DC (D u. E 20 A)
- Frequenz: 10 Hz - 10 MHz (D u. E 220 MHz)
- automatische und manuelle Bereichswahl
- Diodentest und Durchgangsprüfung
- Kapazitäts- und Widerstandsmessung

UNI-T

UT 161D:
mit Temperatur-Messung

UT 161 E:
mit hFE-Messung

TRMS | EN 61010-1 CAT III 1000 V | EN 61010-1 CAT IV 600 V | USB | PC SOFTWARE

ab 69,99

Bestell-Nr.:		Digits / Bargraf	Grundgenauigkeit	Kapazität	Widerstand	Ausführung
UT 161B	**69,99**	6000 / Ja	0,10 %	bis 60 mF	0,1 Ohm - 60 MOhm	
UT 161D	**79,99**	6000 / Ja	0,10 %	bis 60 mF	0,1 Ohm - 60 MOhm	Temperaturmessung
UT 161E	**99,99**	20.000 / Ja	0,10 %	bis 220 mF	0,1 Ohm - 220 MOhm	hFE-Messung

Messgeräte für viele andere Anwendungen finden Sie online!

Gleich entdecken ▶ www.reichelt.de/messtechnik

NUTZEN SIE DIE ZEIT

EFFIZIENTE TECHNIK FÜR IHRE SOMMERWARTUNG

Jetzt informieren ▶

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

www.reichelt.de

Bestellhotline: +49 (0)4422 955-333

reichelt elektronik – The best part of your project

TAGESPREISE! Preisstand: 11. 8. 2021

IMPRESSUM

52. Jahrgang, Nr. 581
September/Oktober 2021
ISSN 0932-5468

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

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

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

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

Anzeigen
Raoul Morreau (Leitung)
Mobil: +31 6 440 399 07
E-Mail: raoul.morreau@elektor.com

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

Es gilt die Anzeigenpreisliste ab 01.01.2021.

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

Druck
Pijper Media, Groningen (NL)

EDITORIAL

Von Jens Nickel

Chefredakteur ElektorMag

Getestet und für gut befunden

Eigentlich wollte ich Ihnen mitteilen, dass ich nun auch vom Raspi-Virus befallen bin. Doch solche Vergleiche dürften spätestens ab diesem Sommer nicht nur bei mir Aversionen auslösen. Während ich diese Zeilen schreibe, stehen die vorsichtigen Lockerungen der verschiedensten Lockdown- und Schutzmaßnahmen leider noch auf sehr fragilen Beinen.
Für alle Fälle finden Sie in diesem Heft wieder eine Fülle von spannenden Projekten, die Sie ganz pandemie-kompatibel in Ihrem Heimlabor aufbauen, ausbauen und anpassen können. Eine lange Tradition haben unsere Leserschaltungen (zum Beispiel auf Seite 20). Oft wurden sie in der Freizeit entwickelt. Von der Konzeption bis zur Dokumentation spiegeln sie dennoch wider, dass die meisten unserer Leser über einen professionellen Hintergrund verfügen. Eine Reihe der Projekte, die uns von externen Autoren eingereicht wurde, wird von meinen Kollegen im Labor nachgebaut und getestet. Dies unterscheidet uns von vielen anderen Magazinen und Web-Plattformen! Mit unserem „Elektor Lab Tested" Stempel werden wir ab dieser Ausgabe auf solche Projekte hinweisen. Darüber hinaus veröffentlichen wir schon seit Jahrzehnten auch Designs, die von unserem Labor selbst entwickelt wurden. In den zugehörigen Artikeln werden wir von jetzt an einen „Elektor Lab Original" Stempel abdrucken.
Wenn alles gut geht, gibt's in einer der nächsten Ausgaben auch mal einen „Lab Original" Artikel von mir selbst. Und damit schließt sich der Kreis zum Raspberry Pi. Da ich des Öfteren auch mal Live-Videos streame (Sie wissen schon, die DJs), hat mich der Artikel meines Kollegen Mathias Claußen auf Seite 50 inspiriert. Er zeigt, wie man einen Raspberry Pi einsetzen kann, um Livebilder im Netz zu teilen. Ich versuche mich gerade an einer kleinen, mobilen Box, mit der man das HDMI-Signal einer Videokamera an eine Plattform wie Twitch streamen kann. Für den kurzen (Test-) Stream zwischendurch, etwa auf einer Wiese, sollte das Ganze natürlich kompakt, stromsparend und halbwegs preisgünstig sein. Mehr darüber in Kürze unter www.elektormagazine.de.

ELEKTOR WIRD 60: ES KOMMT NOCH MEHR!

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

Denise Bodrone, „Elektor 60" Festkomitee-Koordinatorin

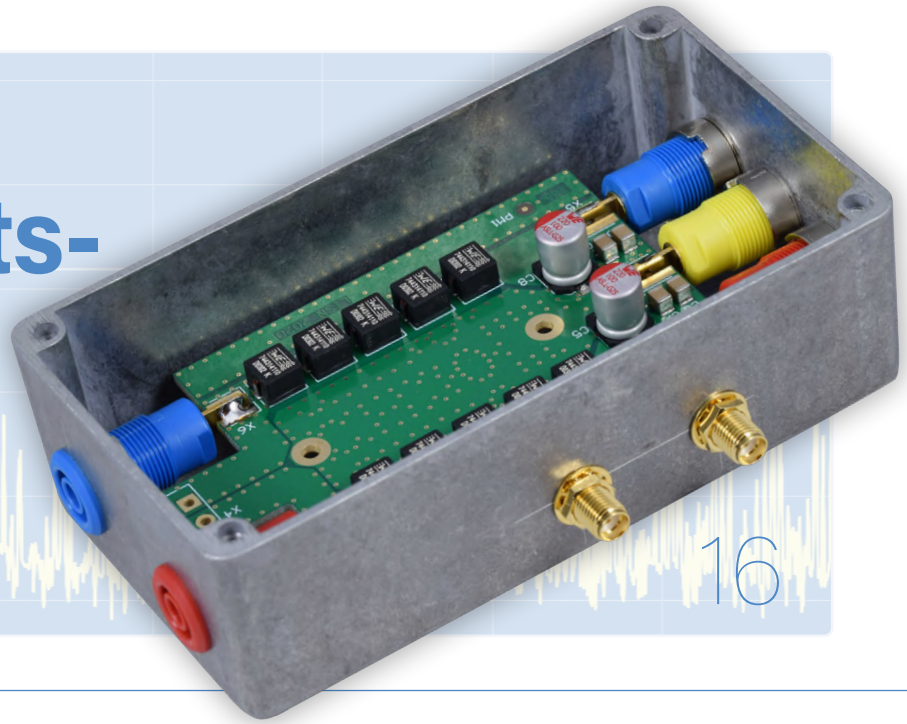
Unser Team

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

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

EMV-Vor-Konformitäts-tester

Was ist eine Netznachbildung?



16

Rubriken

Hintergrund

Kompassrose
mit dem GY-271

78

Projekte

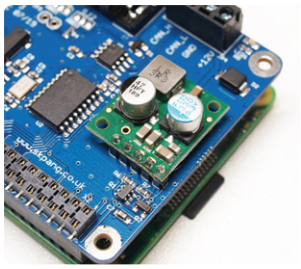
Vorschau

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

Aus dem Inhalt:
- Pi-KVM: Fernsteuerung anderer Computer
- GPS-Tracking mit Open-Source-Software
- Musiksynthesizer mit Rauschen
- Magnesium-Blitz als analoger Speicher
- Kit CO_2-Meter
- ML-Framework für Einsteiger: Edge Impulse
- Reparatur von Lithium-Ionen-Batterien
- Identifikation von Bauteilen

Und vieles mehr!

Elektor November/Dezember 2021 erscheint am 4. November 2021.
Änderungen vorbehalten.



CAN-Bus-HAT
für den
Raspberry Pi 4
41

60 elektor MAG
sixty > years > young

Die Neuronen in neuronalen Netzwerken verstehen

Teil 1. Künstliche Neuronen

Von Stuart Cording (Elektor)

Künstliche Intelligenz (KI) und maschinelles Lernen (ML) revolutionieren die Elektronikindustrie. In dieser Artikelserie untersuchen wir das neuronale Netzwerk. Der erste Teil befasst sich mit der Forschung zu künstlichen Neuronen, einer MLP-Softwareimplementierung (Multilayer Perceptron) und mehr.

Künstliche Intelligenz (KI) und maschinelles Lernen (ML) sind zwei der heißesten Themen in der Branche. Dank spektakulärer Erfolge, zum Beispiel, dass KI die weltbesten Go-Spieler besiegt hat [1], und Misserfolgen wie den Unfällen mit autonomen Fahrzeugen [2] ist der Begriff KI zu einem populären Ausdruck geworden. Während KI und ML dank Cloud-basierter Tools wie TensorFlow für die Allgemeinheit anwendbar wurden, können diese riesigen, leistungsstarken Plattformen undurchdringbar erscheinen, wenn man versucht zu verstehen, wie ML „unter der Motorhaube" funktioniert. In dieser Reihe über neuronale Netzwerke gehen wir zurück zu den Grundlagen und erforschen den Grundbaustein der meisten dieser Systeme, das neuronale Netzwerk. Dabei helfen uns viele Beispiele zum Ausprobieren, Beispiele für coole und obskure ML-Projekte, und am Ende werden wir sogar einen Arduino mit einem „Gehirn" ausstatten.

Computing-Herausforderungen

KI und ML sind *die* Computing-Herausforderungen unserer Zeit. KI, bei der es darum geht, mit Hilfe von Computern die menschliche Intelligenz nachzuahmen, und ML, das auf Mustererkennung von (halb-) strukturierten Daten abzielt, beanspruchen jährlich beträchtliche Investitionen - von Forschungsprojekten bis hin zur Entwicklung in der Halbleitertechnologie und in Computerplattformen. Und dank „der Cloud" ist die Technologie für diejenigen, die ihre Ideen erforschen und testen wollen, leicht zugänglich.
Aber wie funktionieren all diese cleveren Algorithmen? Wie lernen sie? Was sind ihre Grenzen? Und ist es möglich, mit den Grundlagen von ML herumzuspielen, ohne sich für (noch) einen weiteren Cloud-Dienst zu registrieren? Genau diese Fragen werden in dieser kurzen vierteiligen Artikelreihe behandelt:

- Teil 1 - Künstliche Neuronen: Wir beginnen in den 1950er Jahren und sehen uns die frühen Forschungen zur Entwicklung eines künstlichen Neurons an. Wir bewegen uns schnell zu einer Software-Implementierung eines Multilayer-Perzeptrons (MLP), das Backpropagation zum „Lernen" verwendet.
- Teil 2 - Logische Neuronen: Eine der Herausforderungen bei frühen Neuronen war ihre Unfähigkeit, die XOR-Funktion zu lösen. Wir untersuchen, ob unser MLP dieses Problem lösen kann und zeigen, wie das Neuron lernt.
- Teil 3 - Praktische Neuronen: Wir wenden unser MLP auf einen Teil des Problems des autonomen Fahrens an, der Erkennung des Zustands von Ampeln mit Hilfe eines PC-basierten Programms.
- Teil 4 - Eingebettete Neuronen: Wenn es auf einem PC funktioniert, könnte es dann auch auf einem Mikrocontroller funktionieren? Wir erkennen Ampelfarben mit einem Arduino und einem RGB-Sensor.

Kleine Geschichte der Neuronen

Frühe Versuche, ein digitales oder künstliches Neuron zu erstellen, orientierten sich an der Natur. Das biologische Neuron nimmt Eingaben über seine Dendriten auf und gibt die daraus resultierende Ausgabe über sein Axon an die Axon-Ausgänge weiter (**Bild 1**). Die Entscheidung, ob ein Reiz über den Ausgang abgegeben wird, das so genannte Feuern des Neurons, erfolgt durch einen Prozess, der als Aktivierung bezeichnet wird. Entsprechen die Eingaben einem gelernten Muster, feuert das Neuron, andernfalls nicht. Es ist schnell zu erkennen, dass Ketten miteinander verbundener biologischer Neuronen sehr komplexe Muster erkennen können. Die ersten künstlich entwickelten Neuronen waren McCulloch-Pitts-Netzwerke, auch bekannt als *Threshold Logic Units* (TLU). Diese einfachen „Entscheidungsmaschinen" konnten die Funktion von Logikgattern nachahmen, akzeptierten und gaben nur logische Werte von 0 und 1 aus. Um diese Fähigkeit zur Mustererkennung zu nutzen, mussten für jeden Eingang mathematisch oder heuristisch Gewichtungswerte bestimmt werden. Manchmal wäre auch ein zusätzlicher Eingang erforderlich (**Bild 2**, UND- und ODER-Verknüpfungen).
Die Eingänge des Netzes werden also einfach mit ihrer Gewichtung multipliziert und aufsummiert. Die Entscheidung, eine 1 auszugeben (den Ausgang zu aktivieren), wird über eine lineare

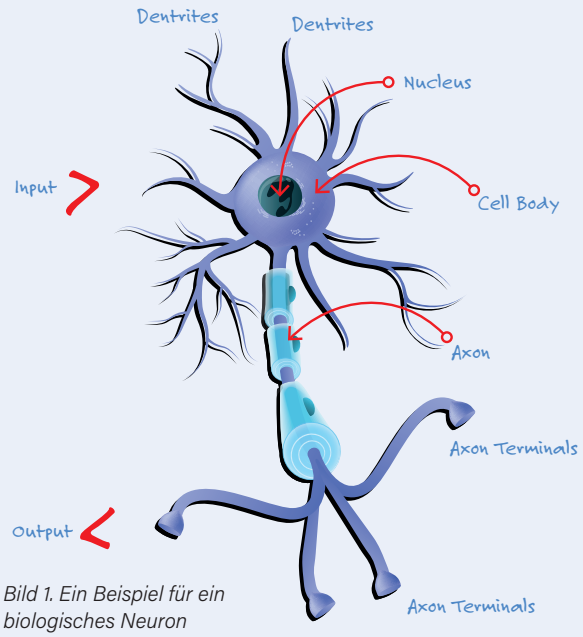


Bild 1. Ein Beispiel für ein biologisches Neuron

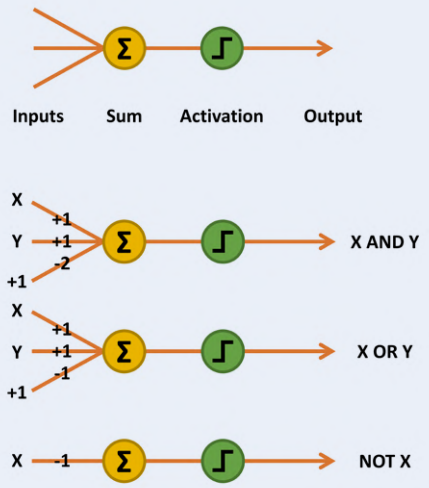


Bild 2. Das McCulloch-Pitts-Netzwerk summiert die Eingänge, multipliziert mit ihrer Gewichtung auf und aktiviert einen Ausgang, wenn das Ergebnis gleich oder größer als 0 ist.

Schwellwert-Einheit realisiert. Wenn das Ergebnis gleich oder größer als 0 ist, wird eine 1 ausgegeben, ansonsten eine 0 (**Tabelle 1**).

Eingänge			Gewichtung			Ausgänge für X UND Y	
X	Y	+1	X	Y	+1	Σ (Eingang × Gewichtung)	Ausgang
0	0	+1	+1	+1	-2	-2	0
0	1	+1				-1	0
1	0	+1				-1	0
1	1	+1				0	1

Tabelle 1. McCulloch-Pitt-Netzwerk zur Implementierung einer UND-Funktion.

Perzeptronen

Die nächste Stufe der Entwicklung wurde in den 1950er Jahren mit der Arbeit des Psychologen Frank Rosenblatt [3] beschritten. Sein Perzeptron behielt die binären Eingänge und die lineare Schwellwert-Entscheidungseinheit der McCulloch-Pitts-TLU bei. Die Ausgabe war also ebenfalls ein binärer Wert von 0 oder 1, unterschied sich jedoch in zweierlei Hinsicht: Der Schwellwert Theta (Θ) für die Entscheidung über den Ausgabewert war einstellbar, und es wurde eine begrenzte Form des Lernens unterstützt (**Bild 3**).

Der Lernprozess funktioniert folgendermaßen: Das Perzeptron gibt nur dann den Wert 1 aus, wenn die Summe des Produkts der gewichteten Eingänge größer ist als Theta. Wenn die Ausgabe in Bezug auf die Kombination der Eingänge korrekt ist, wird nichts verändert.

Wird jedoch ein Wert von 1 ausgegeben, obwohl eine 0 erforderlich wäre, so wird der Schwellwert Theta um eins erhöht. Im umgekehrten Fall, wenn also ein Wert von 0 ausgegeben wird, obwohl eine 1 erforderlich wäre, werden alle Gewichtungen, die mit den Eingängen von 1 verbunden sind, um 1 erhöht.

Der Gedanke hinter diesem Verfahren ist, dass nur die Eingänge mit einem Wert von 1 zu einer unerwünschten Ausgabe von 1 beitragen können, so dass es sinnvoll ist, deren Auswirkung zu reduzieren, indem die entsprechenden Gewichtungen verringert werden. Umgekehrt können nur Eingänge mit einem Wert von 1 zu einer gewünschten Ausgabe von 1 beitragen. Wenn die Ausgabe 0 und nicht wie gewünscht 1 ist, müssen die zugehörigen Gewichtungen erhöht werden.

Im Jahr 1958 wurde das *Mark I Perzeptron* als Hardware-Implementierung erstellt, nachdem es zunächst als Software auf einem IBM 704 realisiert worden war [4]. Angeschlossen an 400 Cadmiumsulfid-Fotozellen, die eine rudimentäre Kamera bildeten, und mit motorgesteuerten Potentiometern, die die Gewichtungen während des Lernens einstellten, konnte es nach dem Training die Form eines Dreiecks erkennen [5].

Das Problem der Perzeptronen

Obwohl dies bewies, dass ein elektronisches System potenziell lernen konnte, gab es ein entscheidendes Problem mit diesem Design: Es konnte nur linear trennbare Probleme lösen. Um auf die frühere McCulloch-Pitts-TLU zurückzukommen: Alle einfachen UND-, ODER-, NICHT-, NAND- und NOR-Funktionen sind linear trennbar. Das bedeutet, dass eine einzige Zeile die (in Bezug auf die Eingänge) gewünschten Ausgänge von den unerwünschten Ausgängen trennen kann (**Bild 4**). Anders verhält es sich bei XOR- (und den komplementären XNOR-) Funktionen. Wenn die Eingänge gleich sind (00 oder 11), ist der Ausgang 0, aber wenn die Eingänge unterschiedlich sind (01 oder 10), ist der Ausgang 1. Dies erfordert, dass der gewünschte Ausgang in Bezug auf die Eingänge in eine Gruppe eingeteilt wird. Vereinfacht ausgedrückt, kann das Perzeptron nicht trainiert werden, um zu lernen, wie XOR oder XNOR funktioniert, oder um seine Funktion zu replizieren.

Das andere Hauptproblem lag in der verwendeten Aktivierungsfunktion. Die lineare Schwelleinheit machte einen scharfen Sprung zwischen inaktiv und aktiv. Die Forschungen, die zum *Delta-Rule*-Netzwerk [6] führten, zeigten, dass das Lernen durch Gradientenabstieg ein entscheidendes Element im Lernprozess eines neuronalen Netzwerks ist. Das bedeutete auch, dass jede Aktivierungsfunktion differenzierbar sein muss. Der plötzliche Sprung von 0 auf 1 in der linearen Schwelleneinheit ist aber am Übergangspunkt nicht differenzierbar (die Steigung wird unendlich), und der Rest der Funktion liefert einfach 0 (die Ausgabe bleibt unverändert).

Es wurde vorgeschlagen, das XOR-Problem mit einem Multilayer-Netzwerk mit mehreren versteckten Knoten zwischen den Eingangs- und Ausgangsknoten zu lösen. Außerdem könnte eine differenzierbare Funktion, etwa die logistische Funktion in **Bild 5**, eine Sigmoidkurve, eine stetige Aktivierungsfunktion liefern, die das Lernen mit Gradientenabstieg unterstützt. Die große

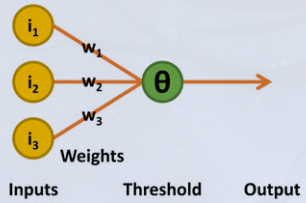


Bild 3. Das Perzeptron führte einen variablen Schwellwert Θ ein und verwendete einen iterativen Lernprozess.

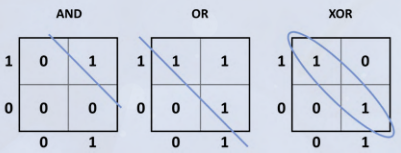


Bild 4. Von links nach rechts: AND-, OR- und XOR-Funktionen. Die 1-Ausgänge der ersten beiden können mit einer Geraden linear von den 0-Ausgangszuständen getrennt werden. Bei XOR ist das nicht möglich, so dass das Perzeptron es nicht lernen kann.

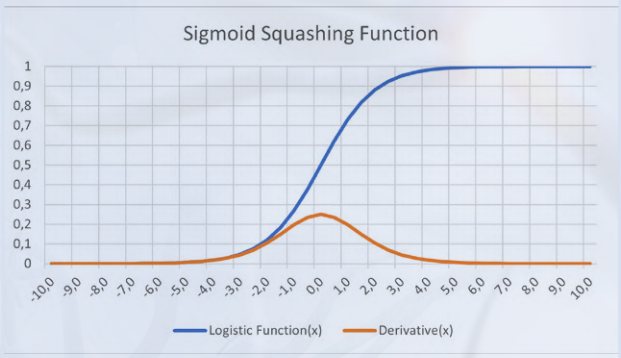


Bild 5. Die logistische Funktion schiebt die Eingabewerte schnell entweder in Richtung 0 oder 1. Dies hilft beim Lernen im Gradientenverfahren in neuronalen Netzen.

Herausforderung war das Lernen - wie würden alle Gewichtungen trainiert werden?
Der Delta-Rule-Ansatz hatte gezeigt, dass durch die Berechnung der quadratischen Abweichung des Netzes (gewünschte Ausgabe - tatsächliche Ausgabe) und die Implementierung einer Lernrate die Gewichtung im Netz sukzessive verbessert werden konnte, bis die optimalen Gewichtungsvektoren gefunden sind (**Bild 6**). Durch das Hinzufügen einer Schicht versteckter Knoten zwischen Ein- und Ausgang wurde die Berechnung zwar komplizierter, aber nicht unmöglich, wie wir sehen werden.

Multilayer-Perzeptron

Das Multilayer-Perzeptron (MLP) wurde durch das Hinzufügen der versteckten Schicht möglich. Die einfachste Form eines neuronalen Netzes mit MLP verwendet nur eine einzige versteckte Schicht. Alle Knoten sind miteinander verbunden (fully connected), wobei zwischen jedem Eingangknoten und verstecktem Knoten sowie zwischen jedem versteckten Knoten und Ausgangsknoten Gewichtungen zugewiesen sind. Die Linien zwischen den Knoten stellen die Gewichtungen dar. Die gewünschten Eingaben werden an die Eingangsknoten angelegt (Werte zwischen 0,0 und 1,0) und das Netzwerk berechnet die Antwort jedes versteckten Knotens und Ausgangsknotens - ein Schritt, der als Feedforward-Phase bezeichnet wird. Dies sollte ein Ergebnis liefern, das zeigt, dass die Eingangswerte mit einer Ausgangskategorie übereinstimmen, die das Netzwerk erlernt hat.
Die Eingänge könnten zum Beispiel mit einer 28 × 28 Pixel großen Kamera verbunden sein, die auf handgeschriebene Zahlen gerichtet ist. Die MNIST-Datenbank mit handgeschriebenen Ziffern, die genau solche Bildgrößen enthält, könnte die Trainingsmenge bilden [7]. Jeder der Ausgänge würde eine der Zahlen von 0 bis 9 repräsentieren. Wenn die Zahl 7 vor die Kamera gehalten wird, würde jeder Ausgang die Wahrscheinlichkeit angeben, dass die Eingabe dieser Zahl 7 entspricht. Der Ausgang 0 zeigt (hoffentlich) an, dass die handgeschriebene Zahl völlig unwahrscheinlich eine 0 ist, wie auch die acht anderen Ausgänge. Lediglich der Ausgang 7 sollte eine hohe Wahrscheinlichkeit anzeigen, dass die handgeschriebene Zahl eine 7 ist. Einmal trainiert, sollte dies die erwartete Funktionalität für eine 7 aus dem Trainingsdatensatz oder einer neu handgeschriebenen 7 sein, die von einem Menschen erkannt werden kann.

Bevor dies geschieht, muss das Netzwerk die vorliegende Aufgabe lernen. Dies wird erreicht, indem der Input (handgeschriebene Siebenen) angelegt und die Ergebnisse, die die Ausgänge liefern, analysiert werden. Da diese anfangs wahrscheinlich nicht korrekt sind, wird ein Lernzyklus durchgeführt, um die Gewichtungen so zu verändern, dass der Fehler reduziert wird. Dieser iterative Lernprozess, bekannt als *Backpropagation*, wird viele tausend Mal ausgeführt, bis die Genauigkeit des Netzwerks den Anforderungen der Anwendung entspricht (**Bild 7**). In der Welt des ML wird dies als *supervised learning*, überwachtes Lernen bezeichnet.
Es gibt zwei weitere wichtige Faktoren, die in der Feedforward- und Backpropagation-Phase berücksichtigt werden müssen. Der erste ist der Bias. Ein Bias-Wert von 1,0 multipliziert mit einer Gewichtung (zwischen 0,0 und 1,0) wird während der Feedforward-Phase auf die Knoten der versteckten und der Ausgabeschicht angewendet. Die Aufgabe des Bias ist es, die Problemlösungsfähigkeiten des Netzwerks zu verbessern und im Wesentlichen die logistische Aktivierungsfunktion (siehe wieder Bild 5) nach links oder rechts zu verschieben. Der andere Wert ist die Lernrate, wiederum ein Wert zwischen 0,0 und 1,0, die bestimmt, wie schnell das MLP lernt, das gegebene Problem zu lösen, auch bekannt als die Geschwindigkeit der Konvergenz. Wenn dieser Wert zu niedrig eingestellt ist, löst das Netzwerk das Problem mit einer angemessenen Genauigkeit möglicherweise nie, ist der Wert zu hoch, läuft das Netzwerk Gefahr, während des Lernens zu oszillieren und ebenfalls kein ausreichend genaues Ergebnis zu liefern (siehe auch *Grenzen des Gradientenlernens*).
Das hier beschriebene MLP könnte als „Vanilla-Entwurf" betrachtet werden. Neuronale Netze können jedoch auf eine Vielzahl von Arten implementiert werden. Dazu gehören mehr als eine versteckte Schicht, eine nicht vollständige Verbindung der Knoten, die Verknüpfung späterer mit früheren Knoten und die Verwendung unterschiedlicher Aktivierungsfunktionen [8].

MLP in Aktion

Das Prinzip ist hoffentlich klar, so dass wir ein konkretes Beispiel untersuchen können. Das Beispiel stammt aus einem hervorragenden Artikel von Matt Mazur, der sich viel Zeit genommen hat, um die Backpropagation in einem MLP mit nur einer einzigen versteckten Schicht zu erklären [9]. Das Beispiel besitzt ein hohen Niveau, aber

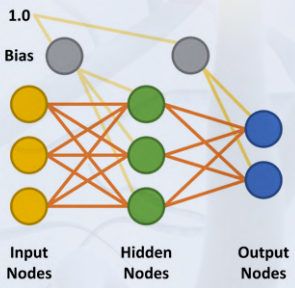


Bild 6. Ein beispielhaftes MLP mit drei Eingangsknoten, drei versteckten Knoten und zwei Ausgangsknoten. Die versteckten und Ausgangsknoten können eine Sigmoid-Kurve verwenden, um ihre Ausgabe in der Feedforward-Phase zu bestimmen.

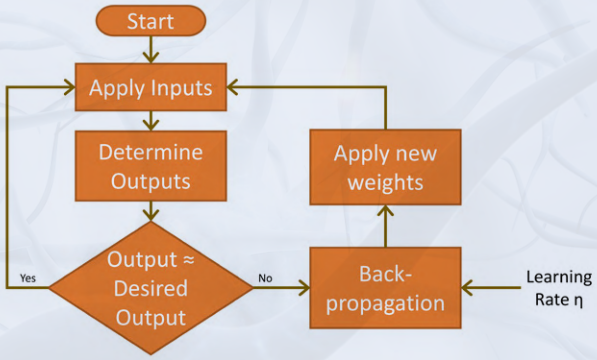


Bild 7. Das Flussdiagramm zeigt, wie das Lernen eines neuronalen Netzwerks implementiert wird.

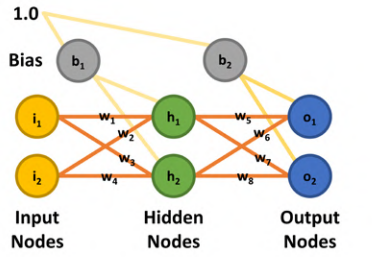


Bild 8. Für die Beispielrechnungen verwendete MLP-Konfiguration.

diejenigen, die an mehr Details interessiert sind (und keine Angst vor Mathematik haben), sollten Matts Artikel studieren. Sobald das mathematische Prinzip des Beispiels verstanden ist, können wir eine Software-MLP-Implementierung betrachten, die mit den gleichen Parametern wie in dieser theoretischen Analyse arbeitet. Wenn Sie mitmachen wollen; es gibt es eine Excel-Tabelle, die zu Matts Beispiel passt. Laden Sie einfach das Repository von GitHub [10] herunter oder klonen Sie es und werfen Sie einen Blick auf *workedexample/Matt Mazur Example.xlsx* im Ordner.

Der Einfachheit halber wird ein MLP mit zwei Eingängen, zwei Ausgängen und zwei versteckten Knoten verwendet. Die Eingangsknoten werden mit i_1 und i_2 bezeichnet, die versteckten Knoten h_1 und h_2 und die Ausgangsknoten o_1 und o_2. Ziel der Übung ist, das Netzwerk so zu trainieren, dass es am Knoten o_1 0,01 und am Knoten o_2 0,99 ausgibt, wenn der Knoten i_1 0,05 und i_2 0,10 ist. Es gibt acht Gewichtungen (w_1 bis w_8) und zwei Bias-Werte (b_1 und b_2). Damit die Mathematik durchgerechnet werden kann, sind allen Eingangsknoten, Bias' und Gewichtungen die in **Bild 8** gezeigten Werte zugewiesen und stimmen mit dem Artikel von Matt Mazur überein.

Feedforward

In der Feedforward-Phase zur Berechnung der Ausgänge o_1 und o_2 erhält jeder versteckte Knoten einen Eingang, der die Summe der Eingänge multipliziert mit den Gewichtungen ist, zuzüglich des Biaseingangs (b_1 = 0,35). Im Prozess und mit den Gleichungen von Matt Mazur ist der Eingang für h_1 die Summe von i_1 multipliziert mit w_1 (0,15), i_2 multipliziert mit w_2 (0,20) und b_1 (0,35).

$$(input\ to\ node)\ net_{h1} = w_1 \times i_1 + w_2 \times i_2 + b_1 \times 1$$

$$0.15 \times 0.05 + 0.20 \times 0.10 + 0.35 \times 1 = 0.3775$$

Der Ausgangswert jedes versteckten Knotens wird durch die logistische (Aktivierungs-) Funktion bestimmt, die den Ausgang des versteckten Knotens auf einen Wert zwischen 0,0 und 1,0 drückt. Dies wird wie folgt berechnet:

$$(output\ of\ node)\ out_{h1} = \frac{1}{1 + e^{-net_{h1}}}$$

$$= 0.593269992$$

Wiederholt man den Vorgang für h_2 (mit w_3 = 0,25, w_4 = 0,3 und b_1 = 0,35), erhält man:

$$(output\ of\ node)\ out_{h2} = 0.596884378$$

Die Netzeingänge zu den Ausgangsknoten werden auf die gleiche Weise berechnet, wobei die berechneten Ausgangswerte der versteckten Knoten verwendet und die Gewichte 5 bis 8 (w_5 = 0,4, w_6 = 0,45, w_7 = 0,5 und w_8 = 0,55) sowie der Bias-Wert b_2 = 0,60 eingesetzt werden.

$$(output\ of\ node)\ out_{o1} = 0.75136507$$

$$(output\ of\ node)\ out_{o2} = 0.772928465$$

Wie bereits erwähnt, ist es unser Ziel, dass o_1 einen Wert von 0,01 und o_2 einen Wert von 0,99 ausgibt, aber wir sehen, dass wir von diesem Ergebnis noch ein Stück entfernt sind. Der nächste Schritt ist deshalb die Berechnung des Fehlers in jedem Ausgang mit Hilfe der quadratischen Fehlerfunktion.

$$E_{total} = \sum \frac{1}{2}(target - ouput)^2$$

Dieser wird für jeden Ausgang wie folgt berechnet:

$$E_{o1} = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$$

$$E_{o2} = \frac{1}{2}(0.05 - 0.772928465)^2 = 0.023560026$$

Schließlich kann der Gesamtfehler für das Netzwerk ermittelt werden:

$$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$$

Der nächste Schritt ist, herauszufinden, wie man diesen Fehler verbessern kann.

Backpropagation

Bei der Backpropagation findet das eigentliche „Lernen" statt. Der Prozess umfasst die Bestimmung des Beitrags, den jede Gewichtung zum Gesamtfehler leistet. Es beginnt mit der Betrachtung der Gewichte zwischen den versteckten Knoten und den Ausgangsknoten. Was die Sache verkompliziert, ist, dass w_5 zum Gesamtfehler über die zwei Ausgangsknoten o_1 und o_2 beiträgt, die aber auch von den Gewichten w_6 bis w_8 beeinflusst werden. Es sollte auch beachtet werden, dass die Bias-Werte bei diesen Berechnungen keine Rolle spielen.

Die dazu erforderliche Mathematik ist recht komplex, aber sie reduziert sich endlich auf einige einfache Multiplikationen, Additionen und Subtraktionen. Die Berechnung des neuen Wertes für w_5 unter Berücksichtigung der gewählten Lernrate η (0,5) wird wie folgt durchgeführt:

$$w_5^+ = w_5 - \eta * \frac{\delta E_{total}}{\delta w_5}$$

Danach können wir die alten Gewichtungen überprüfen und mit den neuen Gewichten für w_5 bis w_8 vergleichen:

$$w_5 = 0.40 \quad w_5^+ = 0.35891648$$

$$w_6 = 0.45 \quad w_6^+ = 0.408666186$$

$$w_7 = 0.50 \quad w_7^+ = 0.511301270$$

$$w_8 = 0.55 \quad w_8^+ = 0.561370121$$

Schon bei einer kurzen Überprüfung sehen wir, dass dies sinnvoll ist. Wir wollen, dass o_1 durch w_5 und w_6 nach unten in Richtung 0,01 und o_2 durch w_7 und w_8 nach oben in Richtung 0,99 gedrückt wird. Der letzte Schritt besteht darin, den Einfluss zu bestimmen, den die Gewichte zwischen den Eingängen und den versteckten Knoten auf den Fehler am Ausgang ausüben. Wieder reduziert sich die Mathematik auf die Grundrechenarten, und tatsächlich sieht die

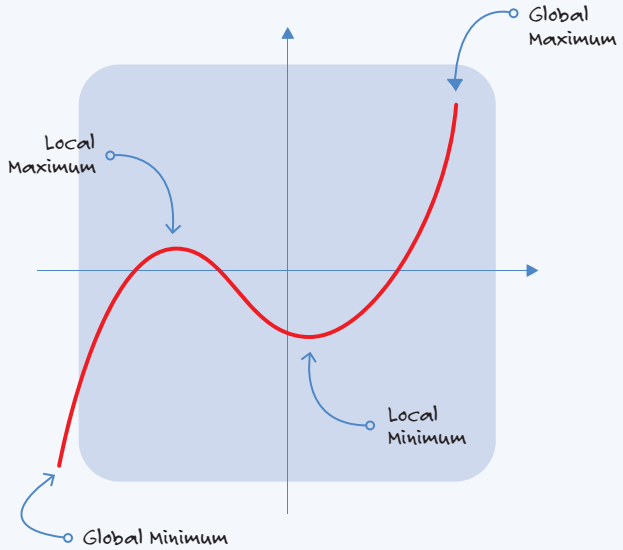
Lazy Learning

Lernen ist harte Arbeit für Menschen und scheinbar auch für KI. Untersuchungen der TU Berlin, des Fraunhofer HHI und der SUTD ließen die KI-Systeme ihre eigenen Entscheidungsstrategien erklären [13]. Die Ergebnisse waren verblüffend. Die KIs erfüllten nicht nur alle ihre Aufgaben bewundernswert, sondern zeigten dabei auch eine atemberaubende Cleverness oder besser Frechheit in ihrer Entscheidungsfindung. Ein Algorithmus erkannte korrekt die Anwesenheit eines Schiffes in einem Bild, begründete seine Entscheidung aber mit der Tatsache, dass sich Wasser auf dem Foto befand. Ein anderer erkannte korrekt, dass Bilder Pferde enthielten, stützte seine Entscheidung aber auf eine Copyright-Markierung in einigen Bildern, anstatt die visuellen Attribute eines Pferdes zu lernen.

Graphic: cjmacer / Shutterstock.com

Grenzen des Gradientenlernens

Es gibt Zeiten, in denen das neuronale Netz scheinbar nicht in der Lage ist, zu lernen, obwohl es zuvor die erforderliche Funktionalität mit der gleichen Knotenkonfiguration gelernt hat. Dies kann darauf zurückzuführen sein, dass das Netz in einem relativen Minimum stecken bleibt, anstatt ein absolutes Minimum der Fehlerfunktion zu finden.



Gleichung genauso aus wie die, die zur Berechnung der neuen Gewichtungen w_5 bis w_8 verwendet wurde. Der Unterschied besteht darin, wie der Gesamtfehler in Bezug auf die Gewichtung (ursprünglich w_1) berechnet wird, da die Ausgabe der versteckten Schicht von einer anderen Eingabe und einer anderen Gewichtung abhängig ist (für h_1, i_1 und w_1 sowie i_2 und w_2):

$$w_1^+ = w_1 - \eta * \frac{\delta E_{total}}{\delta w_1}$$

Auch bei dieser Berechnung spielt der Bias-Wert keine Rolle. Nun können wir die alten Gewichte w_1 bis w_4 und die neuen Gewichte vergleichen:

$$w_1 = 0.15 \quad w_1^+ = 0.149780716$$

$$w_2 = 0.20 \quad w_2^+ = 0.19956143$$

$$w_3 = 0.25 \quad w_3^+ = 0.24975114$$

$$w_4 = 0.30 \quad w_4^+ = 0.29950229$$

Die alten Gewichtungen werden von den ermittelten neuen Gewichtungen ersetzt und ein neuer Vorwärtsdurchlauf durchgeführt. Solange der Ausgangsfehler größer als gewünscht bleibt, können die beschriebenen Backpropagation-Durchläufe wie hier beschrieben wiederholt werden.

MLP-Implementierung in Processing

Um dieses einfache neuronale Netzwerk zu demonstrieren, wurde ein neuronales Netzwerk von Grund auf als Klasse kodiert, die in der Entwicklungsumgebung *Processing* [11] (die speziell für das Kodieren in der visuellen Kunst entwickelt wurde) verwendet werden kann. Diese visuellen Fähigkeiten erlauben der IDE die einfache Darstellung von 2D- und 3D-Grafiken. Durch die Ausgabe auf eine Textkonsole können zudem Ideen schnell und einfach getestet werden. Der folgende Code ist Teil des erwähnten Repositorys [10].

Der Code für die MLP-Implementierung befindet sich im Ordner *processing/neural/neural.pde*. Diese Datei wird einfach zu jedem Ihrer Processing-Projekte hinzugefügt. Die Klasse Neural kann instanziiert werden, um eine beliebige Anzahl von Eingabe-, versteckten und Ausgabeknoten zu unterstützen. Um das obige Beispiel nachzuvollziehen, sollte nun die Datei *processing/nn_test/nn_test.pde* in Processing geöffnet werden.

Das Erstellen eines neuronalen Netzwerks ist recht einfach. Zunächst wird der Klassenkonstruktor Neural (in *nn_test.pde*) aufgerufen, um ein Objekt mit (hier) dem Namen network zu erstellen, das die gewünschte Anzahl von Eingängen, versteckten Knoten und Ausgängen (2, 2 und 2) definiert. Nach der Erstellung werden weitere Member-Funktionen aufgerufen, um die Lernrate und die Biaswerte für den versteckten und den Ausgangsknoten gemäß der Beispielskizze einzustellen.

Der Konstruktor initialisiert auch die Gewichtungen mit Zufallswerten zwischen 0,25 und 0,75. Für das obige Beispiel ändern wir die Gewichtungen wie in **Listing 1** gezeigt, wo auch die Eingangswerte und die gewünschten Ausgangswerte definiert sind.

Der Beispielcode aktiviert auch einen Verbose-Modus, in dem die Arbeit für jeden Schritt der Berechnungen angezeigt wird. Diese sollten mit den Ergebnissen in der Excel-Tabelle übereinstimmen. Nach einem Klick auf *Run* in Processing sollte die Textkonsole folgendes ausgeben:

```
...forwardpass complete. Results:
For i1 =  0.05  and i2 =  0.1
o1 =  0.75136507  (but we want:  0.01 )
o2 =  0.7729285  (but we want:  0.99 )
Total network error is:  0.2983711
```

Danach wird *learning* aktiviert und wieder ein Vorwärtsdurchlauf ausgeführt, gefolgt von dem Backpropagation-Schritt. Dies führt zur Ausgabe der neuen und alten Gewichte und zur Berechnung eines neuen Vorwärtsdurchlaufs, um die neuen Ausgangsknotenwerte und Fehler zu bestimmen:

```
New Hidden-To-Output Weight [ 0 ][ 0 ] = 0.3589165,
   Old Weight = 0.4
New Hidden-To-Output Weight [ 1 ][ 0 ] = 0.40866616,
   Old Weight = 0.45
New Hidden-To-Output Weight [ 0 ][ 1 ] = 0.5113013,
   Old Weight = 0.5
New Hidden-To-Output Weight [ 1 ][ 1 ] = 0.56137013,
   Old Weight = 0.55
New Input-To-Hidden Weight[ 0 ][ 0 ] = 0.14978072,
   Old Weight = 0.15
New Input-To-Hidden Weight[ 1 ][ 0 ] = 0.19956143,
   Old Weight = 0.2
New Input-To-Hidden Weight[ 0 ][ 1 ] = 0.24975115,
   Old Weight = 0.25
New Input-To-Hidden Weight[ 1 ][ 1 ] = 0.2995023, Old
   Weight = 0.3
```

Wir können sehen, dass dies die Gewichtungen 5 bis 8 und dann 1 bis 4 darstellt. Sie stimmen auch mit den zuvor durchgeführten Überschlagsberechnungen überein (mit kleinen Abweichungen aufgrund von Rundungsfehlern).

Nächste Schritte mit MLP

Nachdem Sie ein gutes Verständnis für die Funktionsweise eines neuronalen MLP-Netzes und für die hier zur Verarbeitung bereitgestellte beispielhafte Neuronalklasse entwickelt haben, haben Sie eine gute Ausgangsposition, um selbständig weitere Experimente durchzuführen. Hier einige Ideen:

- Ausführen des Lernens in einer Schleife - wie viele Durchläufe sind erforderlich, um einen Netzwerkfehler von 0,001, 0,0005 oder 0,0001 zu erreichen?
- Spielen Sie mit unterschiedlichen Biaswerten und Gewichtungen - wie wirkt sich dies auf die Anzahl der zum Lernen erforderlichen Durchläufe aus? Scheitert das neuronale Netzwerk irgendwann beim Lernen?
- Ordnen Sie die Ausgaben den Eingaben zu – und versuchen Sie, eine 3D-Darstellung jeder Ausgabe gegen die Eingaben zu entwickeln, sobald das Netzwerk seine Aufgabe gelernt hat. Sieht es so aus, wie Sie es erwarten würden? Probieren Sie auch mal das Chart-Studio von Plotly [12] aus, anstatt eine Tabellenkalkulation zum Plotten der Ausgabedaten zu verwenden.

Im nächsten Teil dieser Artikelreihe werden wir unserem neuronalen Netzwerk beibringen, wie man Logikgatter implementiert und seinen Lernprozess visualisiert.

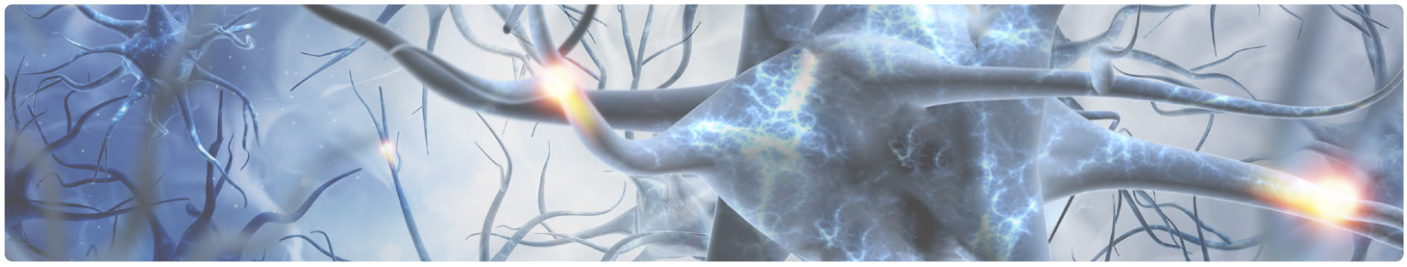
210160-02

Haben Sie Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Dann schreiben Sie dem Autor eine E-Mail an stuart.cording@elektor.com.

Ein Beitrag von

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



Listing 1. Verwendung der MLP-Neuralklasse in nn_test.pde

```
Neural network;

void setup() {
   // This neural network uses two inputs, two hidden nodes, and two output nodes.
   network = new Neural(2,2,2);

   // Set learning rate here
   network.setLearningRate(0.5);

   // Set Neural class to be 'verbose' so we can see what it is doing
   network.turnVerboseOn();

   println("Matt Mazur Neural Network Backpropagation Example");
   println("-------------------------------------------------");
   println();
   println("Structure is:");
   println("Input nodes: ", network.getNoOfInputNodes(), "; Hidden nodes: ", network.getNoOfHiddenNodes(),
"; Output nodes: ", network.getNoOfOutputNodes());

   // Set network biasing here
   // b1 = 0.35
   network.setBiasInputToHidden(0.35);
   // b2 = 0.60
   network.setBiasHiddenToOutput(0.6);

   // The Neural class constructor gives the weights random value.
   // Here we use the values from Matt Mazur's example.
   // We start with the input-to-hidden weights:
   // w1 = 0.15 (i1 to h1)
   network.setInputToHiddenWeight(0, 0, 0.15);
   // w3 = 0.25 (i1 to h2)
   network.setInputToHiddenWeight(0, 1, 0.25);
   // w2 = 0.20 (i2 to h1)
   network.setInputToHiddenWeight(1, 0, 0.2);
   // w4 = 0.30 (i2 to h2)
   network.setInputToHiddenWeight(1, 1, 0.30);

   // Next we configure the hidden-to-output weights:
   // w5 = 0.40 (h1 to o1)
   network.setHiddenToOutputWeight(0, 0, 0.4);
   // w7 = 0.50 (h1 to o2)
   network.setHiddenToOutputWeight(0, 1, 0.5);
   // w6 = 0.45 (h2 to o1)
   network.setHiddenToOutputWeight(1, 0, 0.45);
   // w8 = 0.55 (h2 to o2)
   network.setHiddenToOutputWeight(1, 1, 0.55);

   // Configure the inputs
   // i1 = 0.05
   network.setInputNode(0, 0.05);
   // i2 = 0.10
   network.setInputNode(1, 0.1);

   // Now declare the values we like to achieve at the outputs for this
   // input combination
   // o1 should be 0.01
   network.setOutputNodeDesired(0, 0.01);
   // o2 should be 0.99
   network.setOutputNodeDesired(1, 0.99);

   // We now perform a forwardpass using the configured inputs, weights
   // and bias value. Verbose is on, so you will see the working
```

```
  println();
  println("Calculating values for o1 and o2...");
  println();
  network.calculateOutput();
  println();

  // Let's summarise the current state
  println("...forwardpass complete. Results:");
  println("For i1 = ", network.getInputNode(0), " and i2 = ", network.getInputNode(1));
  println("o1 = ", network.getOutputNode(0), " (but we want: ", network.getOutputNodeDesired(0), ")");
  println("o2 = ", network.getOutputNode(1), " (but we want: ", network.getOutputNodeDesired(1), ")");
  println();
  println("Total network error is: ", network.getTotalNetworkError());
  println();

  // Now we'll perform a learning cycle using backpropagation
  // This enables a backpropagation cycle everytime the calculateOuput() method is called
  network.turnLearningOn();

  println("Learning is ON");
  println("Calculating outputs again and performing backpropagation...");
  println();
  network.calculateOutput();

  // We'll turn learning off again...
  network.turnLearningOff();
  // ...and run another forwardpass without verbose on:
  network.turnVerboseOff();
  network.calculateOutput();

  println("...backpropagation complete. Results:");
  println("For i1 = ", network.getInputNode(0), " and i2 = ", network.getInputNode(1));
  println("o1 = ", network.getOutputNode(0), " (but we want: ", network.getOutputNodeDesired(0), ")");
  println("o2 = ", network.getOutputNode(1), " (but we want: ", network.getOutputNodeDesired(1), ")");
  println();
  println("Total network error is: ", network.getTotalNetworkError());
  println();
}

void draw() {
}
```

WEBLINKS

[1] **M. Reynolds, „DeepMind's AI Beats World's Best Go Player in Latest Face-Off", NewScientist, Mai 2017**: http://bit.ly/3up6Xy3

[2] „Uber's self-driving operator charged over fatal crash", BBC News, September 2020: http://bbc.in/3pJLQ5R

[3] **Frank Rosenblatt**: https://de.wikipedia.org/wiki/Frank_Rosenblatt

[4] **Perzeptron**: https://de.wikipedia.org/wiki/Perzeptron

[5] **H. Mason, D. Stewart, and B. Gill, „Rival", The New Yorker, November 1958**: http://bit.ly/2NS4hrU

[6] **I. Russell, „The Delta Rule", University of Hartford**: http://bit.ly/3kgFt91

[7] **Y. LeCun and C. Cortes, „The MNIST Database", New York University**: http://bit.ly/3kkdN39

[8] **A. Tch, „The mostly complete chart of Neural Networks, explained", Towards Data Science, August 2017**: http://bit.ly/2ZJnQoY

[9] **M. Mazur, „A Step by Step Backpropagation Example", März 2015**: http://bit.ly/2NxLNx5

[10] **GitHub Repository - simple-neural-network**: http://bit.ly/2ZHLv9p

[11] **Processing-IDE**: https://processing.org/

[12] **Plotly Chart-Studio**: https://chart-studio.plotly.com

[13] **„Paper bei Nature Commmunications erschienen: Wissenschaftler stellen KI-Systeme auf den Prüfstand ", Fraunhofer HHI, März 2019**: https://bit.ly/3y0Ul0S

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!

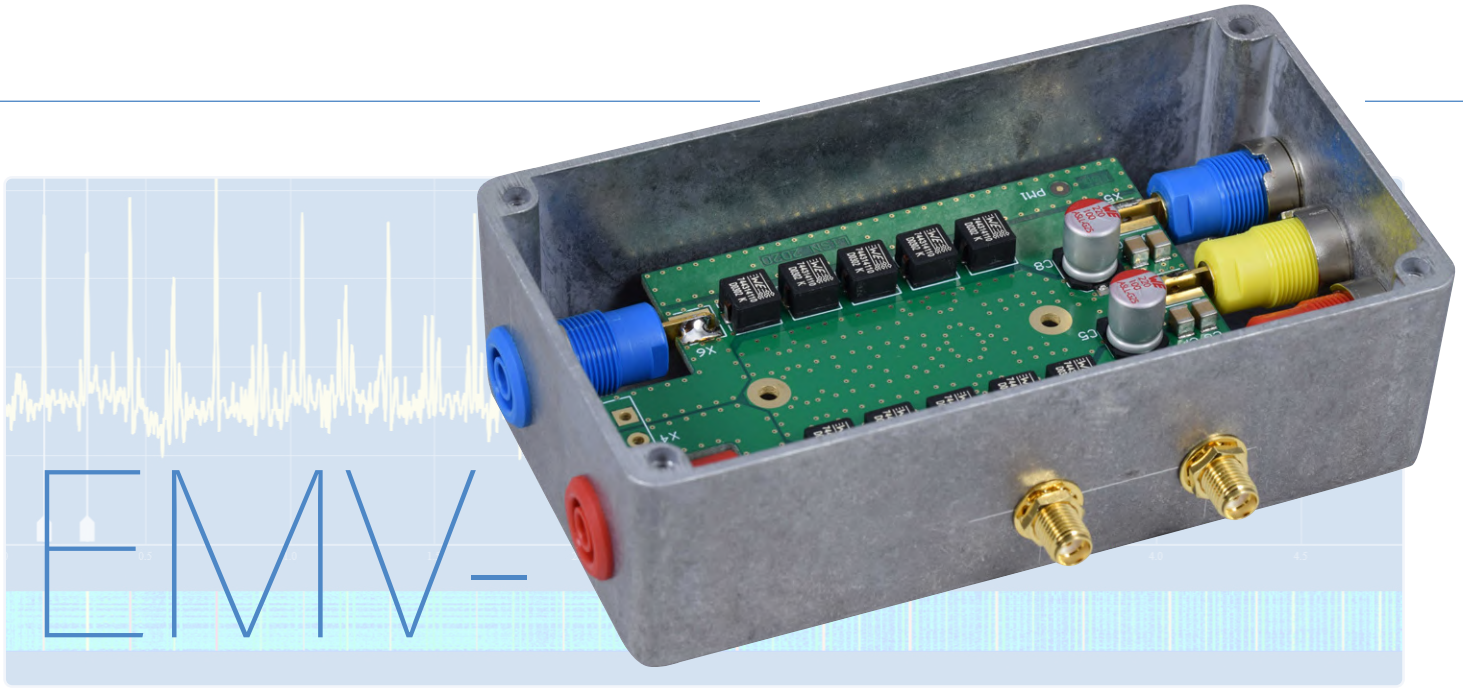
Was können Sie erwarten?

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



Vor-Konformitätstester für Ihr Projekt mit DC-Versorgung

Teil 1: Was ist eine Netznachbildung?

Von Ton Giesberts (Elektor) und **Robert Schillinger** (Würth Elektronik)

Die Messung der leitungsgebundenen Emissionen ist die einfachste und kostengünstigste Methode, um einen Hinweis darauf zu erhalten, ob ein Design die EMI/EMC-Anforderungen erfüllen kann. Eine Netznachbildung ist ein unverzichtbarer Bestandteil einer solchen Prüfung. Wir stellen in Zusammenarbeit mit **Würth Elektronik** ein entsprechendes Selbstbauprojekt vor. Im ersten Teil dieses Artikel werden wir besprechen, wozu eine Netznachbildung dient, im zweiten Teil werden wir sehen, wie die Schaltung aufgebaut ist und wie sie bei der EMV-Vorkonformitätsprüfung eingesetzt wird.

Eine Netznachbildung, die im Englischen die etwas sperrige Bezeichnung *Line Impedance Stabilization Network* (LISN) trägt, bietet eine relativ einfache und kostengünstige Möglichkeit, leitungsgebundene Emissionen an Netzleitungen von Geräten zu ermitteln, die für die EMV-Pre-Compliance geprüft werden müssen. Eine Netznachbildung wird von „normalen" Labors für grundlegende Pre-Compliance-Tests verwendet und macht Tests in einem zertifizierten Full-Compliance-Testlabor für eine spätere Zertifizierung in der Regel weniger kostspielig, da weniger oder keine Änderungen an einem Design mehr vorgenommen werden müssen, um es EMV-konform zu gestalten. Denn jedes Problem, das Sie vorab in Ihrem eigenen Labor lösen können, spart eine Menge Zeit, Geld und Frustration!

Was ist eine Netznachbildung?

Die Netznachbildung [1] wird zwischen die Stromquelle und das zu prüfende Gerät (EUT) geschaltet (siehe **Bild 1**) und bietet eine normierte 50-Ω-Netzleitungsimpedanz, wodurch die Tests unabhängig von der physikalischen Stromquellenimpedanz sind. Darüber hinaus blockiert sein Tiefpass-Eingangsfilter den Eintrag von HF-Störungen von der Stromquelle und/oder Rauschen auf den Stromkabeln in den Prüfling. Eine dritte Funktion einer Netznachbildung ist, dass sie einen oder mehrere 50-Ω-Ausgänge bietet, an die ein Spektrumanalysator und andere Messgeräte angeschlossen werden können, um

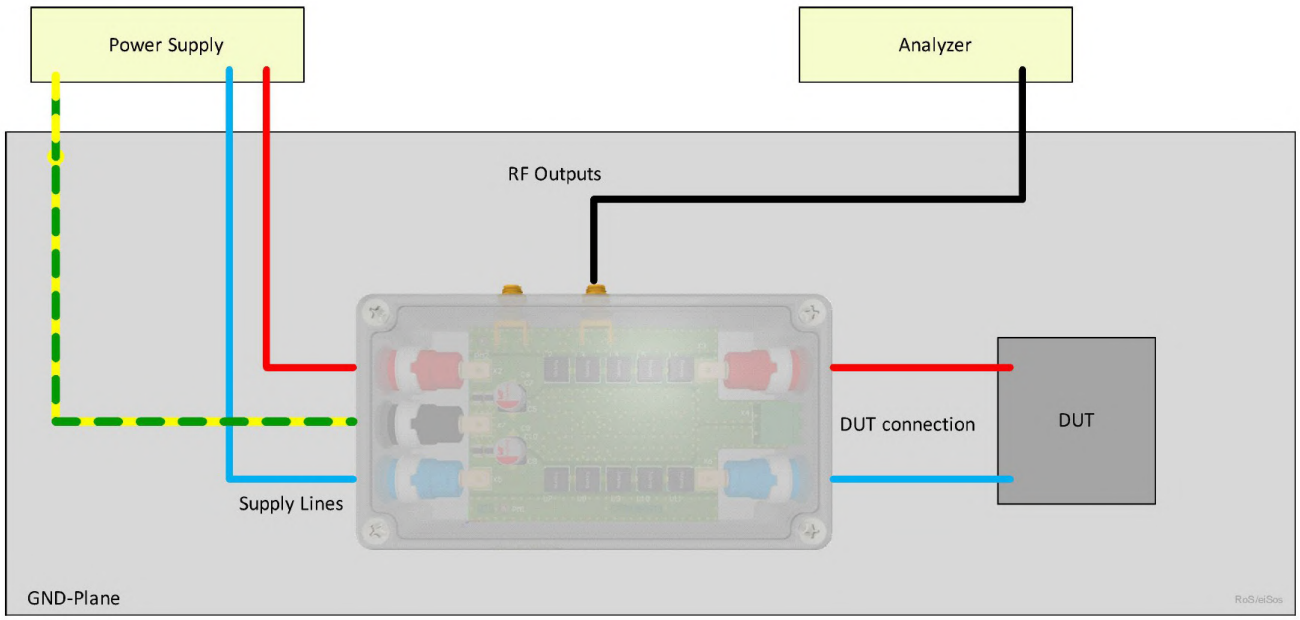


Bild 1. Typischer LISN-Testaufbau zur Messung der leitungsgeführten Emissionen.

die netzleitungsgebundene Emissionen zu bewerten. In den meisten Fällen schützen diese Messausgänge auch die Eingänge des Spektrumanalysators vor Transienten und Überspannungen. Das vom Prüfling erzeugte HF-Rauschen wird durch die Netznachbildung abgetrennt und dem Spektrumanalysator zur Messung, Aufzeichnung und Auswertung zugeführt.

Netznachbildungsschaltungen

Die grundlegende Schaltung für eine Netznachbildung, wie sie von FCC und CISPR empfohlen wird, ist in **Bild 2** dargestellt. Für militärische Standards wird eine 50-µH-Induktivität empfohlen, ein Wert, der einer Netzverkabelungslänge von ungefähr 50 m entspricht, wie sie auf Schiffen und größeren Flugzeugen vorkommt. Für kleinere Plattformen, etwa in Automobilen, kann die Induktivität jedoch viel kleiner sein: Es werden Netznachbildungen mit 5 µH empfohlen. Auf Leitungsnetze dieser Größenordnung zielt dann auch unsere Netznachbildung ab.

In der Praxis dürfte diese einfache Schaltung aufgrund von Streukapazitäten und Induktivitäten, die in jedem physikalischen Aufbau vorhanden sind, nicht wie vorgesehen funktionieren. Daher sind reale Netznachbildungen wie die in **Bild 3** etwas komplexer. Hinzu kommt, dass in der Grundschaltung unseres Projekts die erwähnten Schutzschaltungen für die Eingänge eines Spektrumanalysators entfallen. Der komplette Schaltplan,

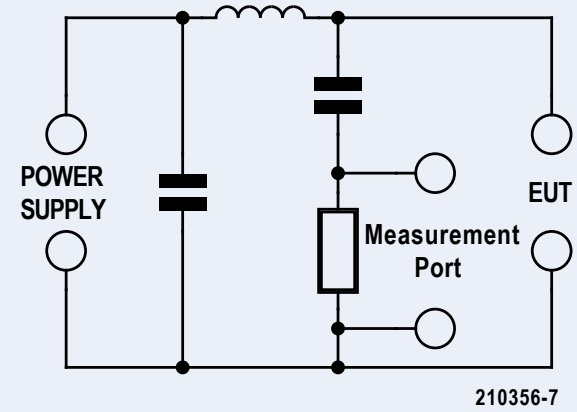


Bild 2. Grundlegendes, vereinfachtes Schaltbild einer Netznachbildung.

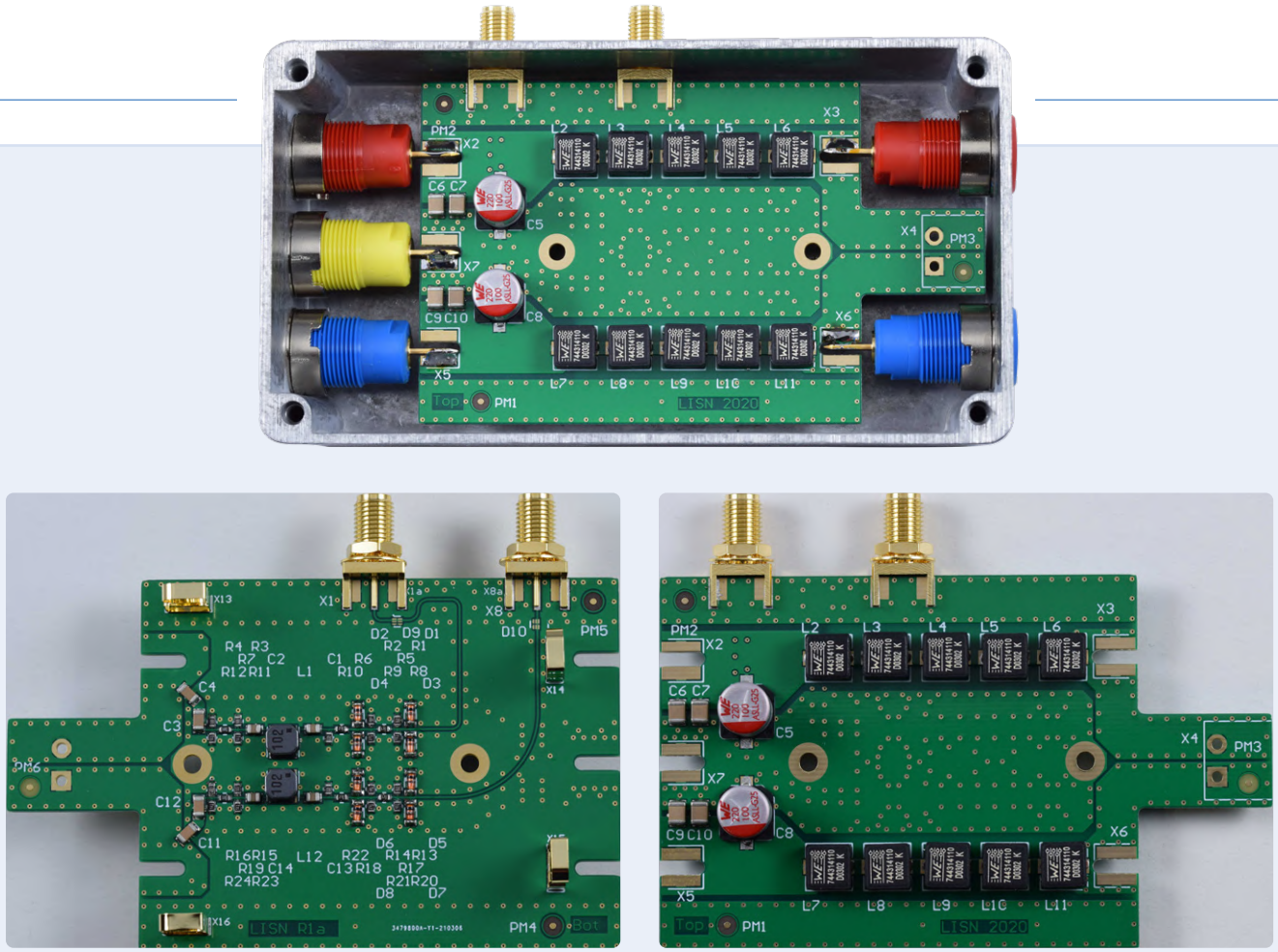


Bild 3. Bilder des Prototyps unseres DC-LISN.

die Platine und die Schaltungsbeschreibung unserer Netznachbildung werden im zweiten Teil dieses Projekts genauer beschrieben und erläutert.

Es gibt viele unterschiedliche Arten von Netznachbildungen, die speziell für verschiedene Prüfnormen, Frequenzbereiche, Betriebsspannungen und -ströme, die Art der Stromquelle (sowohl Wechsel- als auch Gleichstrom) und die zu prüfenden Geräte ausgelegt sind. Der Entwurf in diesem von Elektor/Würth Elektronik vorgestellten Projekt basiert auf einer Demonstrationsschaltung von Analog Devices (DC2130A - früher Linear Technology). Diese Netzwerksimulation „Dual DC LISN" dient der Messung leitungsgebundener Störaussendung eines Prüflings im Frequenzbereich bis 200 MHz, entsprechend der Normen CISPR 25 und ISO 7637 für Kfz-Bordnetze. Die Schaltung bietet zwei Ausgänge zur Messung der HF-Störungen an beiden Netzanschlüssen solcher Systeme und verwendet Sperrinduktivitäten von 5 µH. Unsere Netznachbildung ist sowohl für differentielle als auch für Gleichtakt-EMV-Messungen geeignet. Die

TECHNISCHE KENNDATEN

HF-Pfad

Kanäle:	2 (mit Klemmdioden)
Bandbreite:	200 MHz
Induktivität:	5 µH || 50 Ω
Interne Abschwächung:	10 dB

DC-Pfad

Max. Strom:	< 10 ADC
Max. Spannung:	< 60 VDC
DC-Widerstand:	< 2 x 70 mΩ
Platinengröße:	121 mm x 66 mm x 35 mm (ohne Steckverbinder)

KAUF DES LISN-BAUSATZES

Ein Selbstbaukit mit auf der Platine vormontierten SMDs wird in Kürze im Elektor Store erhältlich sein. Dieser Bausatz enthält auch alle Steckverbinder (die Sie selbst festlöten müssen) und ein vorgebohrtes Aluminium-Druckgussgehäuse. Der Preis steht noch nicht fest.

PASSENDE PRODUKTE

- **STEMlab 125-14 (Starter Kit) (SKU 17939)**
  www.elektor.de/17939

wichtigsten Merkmale und Spezifikationen sind in den technischen Daten (siehe Textbox) aufgeführt. Weitere Informationen zu diesem Projekt finden Sie bereits auf der Webseite von Elektor Labs [2].

Was brauchen Sie noch?

Genau betrachtet ist eine bloße Netznachbildung kein komplettes Messgerät, sondern eher eine spezielle Messsonde, die an ein Gerät wie einen Spektrumanalysator oder EMI-Empfänger angeschlossen wird, um die leitungsgebundenen Emissionen an den Netzanschlüssen des Prüflings zu messen. Dank der Hilfe und Mitarbeit von Würth Elektronik können wir das LISN-Kit (demnächst) in unserem Webshop zu einem enorm günstigen Preis anbieten.

Da die Kosten für einen professionellen Spektrumanalysator, selbst wenn nur eine relativ bescheidene Bandbreite für die Messsignale verlangt ist, deutlich höher sind als die der Netznachbildung, wollen wir im zweiten Teil des Artikels eine erschwinglichere Alternative besprechen, nämlich den Einsatz eines *Red Pitaya STEMlab 125-14*, auf dem eine Diskrete Fourier-Transformation (DFT) läuft. Zwar ist dies keine ideale Lösung, aber die Messungen können dennoch nützliche Erkenntnisse zur Bewertung der leitungsgebundenen Emissionen des Prüflings liefern.

EMV-Messgeräte sind recht teuer, aber die Gesamtinvestition in dieses LISN-Projekt kann sich relativ schnell amortisieren, wenn Sie selbst Pre-Compliance-Tests durchführen und so die Kosten für eine vollständige Zertifizierung reduzieren können.

210356-02

Haben Sie Fragen oder Kommentare?

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

Ein Beitrag von

Idee und Design: **Robert Schillinger (Würth Elektronik)**
Text: **Robert Schillinger, Ton Giesberts**
Illustrationen: **Patrick Wielders**
Redaktion: **Luc Lemmens**
Übersetzung: **Rolf Gerstendorf**
Layout: **Sylvia Sopamena**

WEBLINKS

[1] Netznachbildung: https://de.wikipedia.org/wiki/Netznachbildung

[2] Dieses Projekt auf Elektor-Labs: www.elektormagazine.de/labs/dual-dc-lisn-for-emc-pre-compliance-testing-210296

Elektronische Last für DC und AC

Bis zu 400 V und 10 A

Von **Rainer Schuster**

Beim Entwurf oder Test eines Netzteils wird eine Last benötigt, um zu überprüfen, ob es die Leistungsspezifikationen erfüllt. Die einfachste Last ist ein (Leistungs-) Widerstand, aber wenn das Verhalten des Netzteils bei verschiedenen Belastungen getestet werden soll, wird es zu einer zeitraubenden Aufgabe, wenn verschiedene Widerstände in Reihe oder parallel geschaltet werden müssen, um die Last zu ändern. Eine elektronische Last bietet die Lösung: Sie kann elektronisch gesteuert werden, ohne dass die Prüfschaltung immer wieder neu eingestellt werden muss.

Die hier beschriebene Elektronische Last kann eine Schaltung entweder mit einem konstanten Widerstand oder einem konstanten Strom belasten. Auch eine dynamische Belastung ist möglich, um das Verhalten einer Quelle bei schnell variierenden Ausgangsströmen zu untersuchen. Letztere Option erfordert eine (galvanisch getrennte) USB-Verbindung zu einem PC sowie eine Anwendung auf dem PC, die die Last steuert und berechnete wie gemessene echte Spannungs- und Strom-Effektivwerte anzeigt und protokolliert. Im Gegensatz zu vielen anderen Elektronischen Lasten ist diese Schaltung auch für den Test von AC-Stromquellen geeignet.

Die Hardware

Bild 1 zeigt eine Übersicht der gesamten Elektronischen Last und ihrer einzelnen Funktionsblöcke, die im weiteren Verlauf dieses Artikels beschrieben werden. Für eine maximale Last von 10 A wird eine klassische Stromsenke mit vier parallel geschalteten MOSFETs (Q3, Q6, Q9 und Q12) verwendet (**Bild 2**). Die Steuerspannung ist an den nichtinvertierenden Eingängen der Operationsverstärker IC1 und IC2 angeschlossen. Der resultierende Laststrom für jede Stufe ist gleich dieser Steuerspannung geteilt durch den Drain-Widerstand (R4, R8, R12, R16). Der maximale Strom durch jede Stufe beträgt 2,5 A.

Als Kühlkörper für die Lasttransistoren wird ein LK75 von ELV Elektronik eingesetzt, der mit einem 12-V-Lüfter betrieben wird. In dieser Anwendung besitzt der Kühlkörper einen Wärmewiderstand von 0,44 K/W. Der Wärmewiderstand der Isolierscheibe (1,25 K/W) und der Übergang zum Gehäuse des Transistors (ebenfalls 0,44 K/W) müssen addiert werden, so dass der gesamte Wärmewiderstand 2,13 K/W beträgt. Die sich daraus ergebende maximale Verlustleistung eines Transistors (bei 150°C maximaler Sperrschichttemperatur und 25°C Umgebungstemperatur) ist $P_v = (T_j - T_a)/R_{th} = 58$ W oder 232 W für alle vier Abteilungen. Aus Sicherheitsgründen und

WARNUNG! Das Arbeiten an hoher Spannung kann tödlich sein! Die hier beschriebene Schaltung ist nicht für Anfänger geeignet. Bauen Sie sie nicht auf und verwenden Sie sie nicht, wenn Sie keine Erfahrung im Umgang mit solch hohen Spannungen haben!

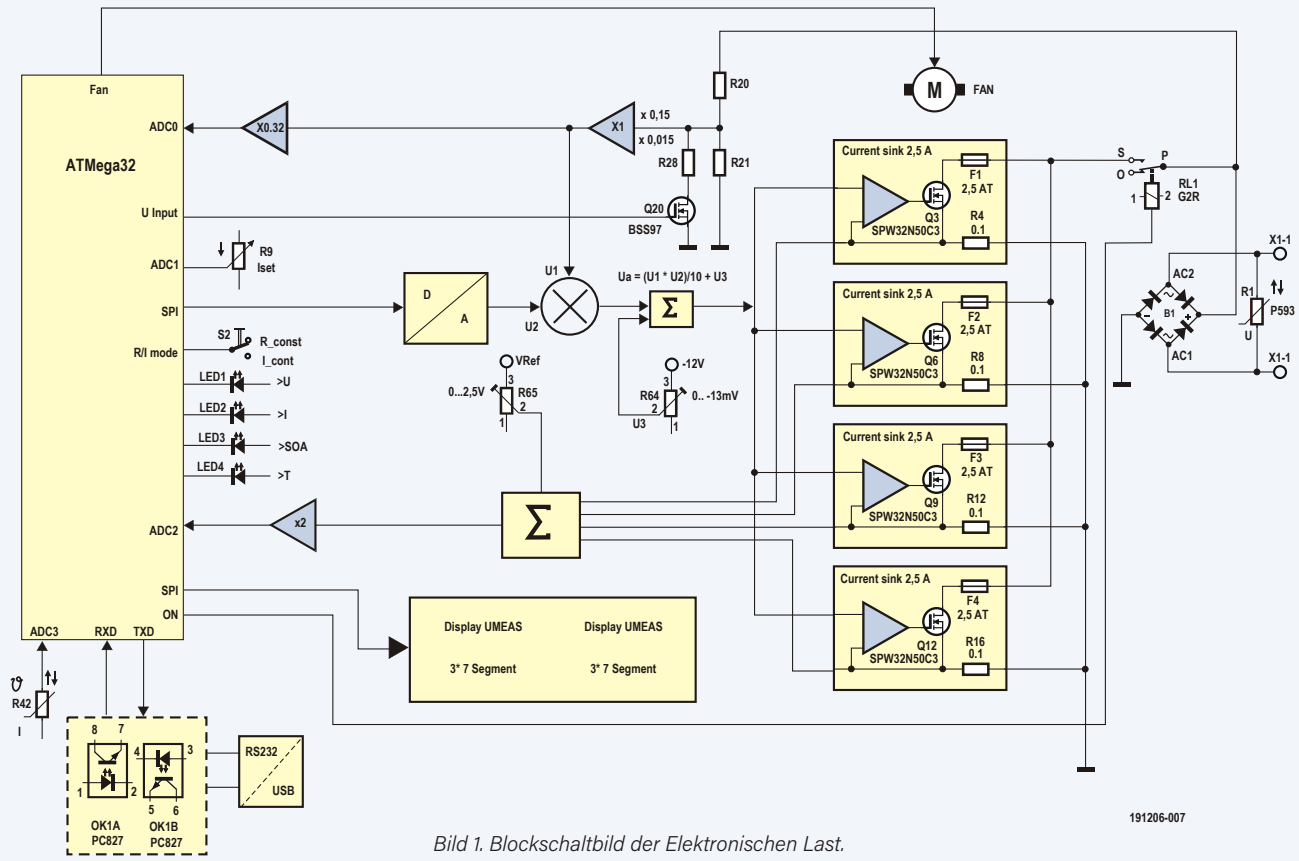


Bild 1. Blockschaltbild der Elektronischen Last.

wegen thermischer Einschränkungen ist die maximale Verlustleistung auf 200 W begrenzt.

Erzeugung der Steuerspannung

Eine weitere Anforderung an diese elektronische Last war, dass sie in der Lage sein sollte, Wechselstrom zu verarbeiten. In diesem Fall muss die Wellenform der Steuerspannung der Wellenform der Eingangsspannung folgen. Um diesen Strom steuerbar zu machen, wird die (skalierte) Eingangsspannung an den Eingang X1 des Analogmultiplizierers IC5, einen AD633 angeschlossen. Die Steuerspannung gelangt über Y1 zum Multiplizierer, dessen Ausgang W direkt mit dem Eingang aller vier Konstantstromsenken verbunden ist. Die Eingänge X2 und Y2 sind mit Masse und Z mit einer negativen Spannung von 0...-100 mV verbunden. Die Spannung am Ausgang W ist:

$$W = ((X1 * Y1)/10) + Z$$

Die Spannung an Z wird am Spannungsteiler R64/R67 eingestellt, um den Stromoffset zu kompensieren. Die Steuerspannung Y1 wird vom Mikrocontroller berechnet und an den 12-Bit-D/A-Wandler MCP4921 (IC6) übertragen. Dessen Referenzspannung ist durch D1 auf 2,5 V eingestellt und dies ist auch die maximale Ausgangsspannung des DACs.

Spannungsmessung

Die Eingangsspannung der Last wird durch den Mikrocontroller gemessen. Um den gesamten Bereich von 0...400 V mit ausreichender Auflösung abzudecken, werden zwei Spannungsbereiche verwendet, die ebenfalls vom Mikrocontroller gewählt werden. Bei eingeschaltetem Q20 wird die Eingangsspannung durch 66 geteilt (R20, R28||R21). Diese Spannung ist direkt mit dem Multiplizierer verbunden, so dass die Wellenform des Stroms der Eingangsspannung folgen kann. Dieser Wert wird wiederum durch R22 und R23 durch 10 geteilt und dann von IC3B mit 3,26 multipliziert. Wenn die Eingangsspannung unter 50 V liegt, wird Q20 ausgeschaltet und die Eingangsspannung wird durch 6,7 (R20/R21) geteilt. Bei einer Eingangsspannung von 50 V beträgt die Spannung am Ausgang von IC3B 2,77 V (Bitwert 567 bei 10-bit-A/D-Wandlung), die Auflösung beträgt 100 mV. Bei einer Eingangsspannung von 400 V beträgt die Spannung am Ausgang von IC3B 1,956 V (Bitwert 400) bei einer Auflösung von 1 V. Je nach Eingangsspannung (AC oder DC) wird entweder die Gleichspannung oder die gleichgerichtete Wechselspannung gemessen. Die Berechnung der echten Effektivwert-Spannung erfolgt in der Software.

Strommessung

Die Ströme durch die vier MOSFETs werden von IC7A und IC7B addiert, bei 2,5 A_{eff} beträgt die Ausgangsamplitude 1 V. Dieser Wert wird mit IC7D multipliziert, so dass sich eine Spannung von 0,43 V pro Ampere ergibt. Je nach Eingangsspannung (AC/DC) wird entweder der Gleichstrom oder der gleichgerichtete Wechselstrom gemessen. Die Berechnung des echten Effektivstroms erfolgt auch hier in der Software. Mit R65 muss der Offset der Stromanzeige eingestellt werden.
Hinweis: Bei 10 A_{DC} liegt am ADC-Eingang eine Spannung von 4,3 V. Bei Wechselströmen darf der Effektivwert nicht größer als 7 A werden!

Mikrocontroller

Zur Steuerung der Elektronischen Last wird ein ATmega32-Mikrocontroller verwendet. Der Controller kann über die ISP-Schnittstelle (SV1) zum Beispiel mit einem AVRISP MKII programmiert werden. Während der Programmierung muss der Controller mit der internen Spannungsversorgung verbunden sein.
Wichtiger Sicherheitshinweis: Wenn die Elektronische Last an den Programmer angeschlossen ist, besteht keine galvanische Trennung zu den Eingangsklemmen. Stellen Sie also sicher, dass beim Programmieren kein zu testendes Netzteil angeschlossen oder die negative Eingangsklemme sicher mit der Masse verbunden ist!

Display

Zur Anzeige von Spannung und Strom werden 7-Segment-LED-Anzeigen an SV2 verwendet, die von einem SPI-Display-Controller MAX7219 angesteuert werden. Der Schalt-

Bild 2. Hauptteil des Schaltplans.

plan wurde aus einem älteren Elektor-Projekt [2] mit einem leicht modifizierten Platinenlayout übernommen, das auf der Laborseite des Projekts [1] zu finden ist.

USB-Schnittstelle

Die Kommunikation mit einem PC wird über M2 abgewickelt. Das auf der Platine verwendete Modul wird von Conrad angeboten, es kann aber jeder USB-UART verwendet werden, der über USB mit Strom versorgt wird. Die Optokoppler OK3 und OK4 sorgen für eine galvanische Trennung zwischen PC und Mikrocontroller.

Spannungsversorgung

Diese elektronische Last benötigt +5 V für den Mikrocontroller und ±12 V für den analogen Teil der Schaltung. Um den Verdrahtungsaufwand so gering wie möglich zu halten, werden dazu zwei Platinentrafos verwendet. Die Regelung der Versorgungsspannungen erfolgt mit Standard-Spannungsreglern (78XX und 79XX) (**Bild 3**).

Firmware

Die Firmware für den ATmega32 wurde mit BASCOM AVR entwickelt. Wenn Sie sie verändern wollen, müssen Sie eine lizenzierte Version erwerben. Sowohl der Quellcode als auch die HEX-Datei stehen unter [1] zum Download bereit.

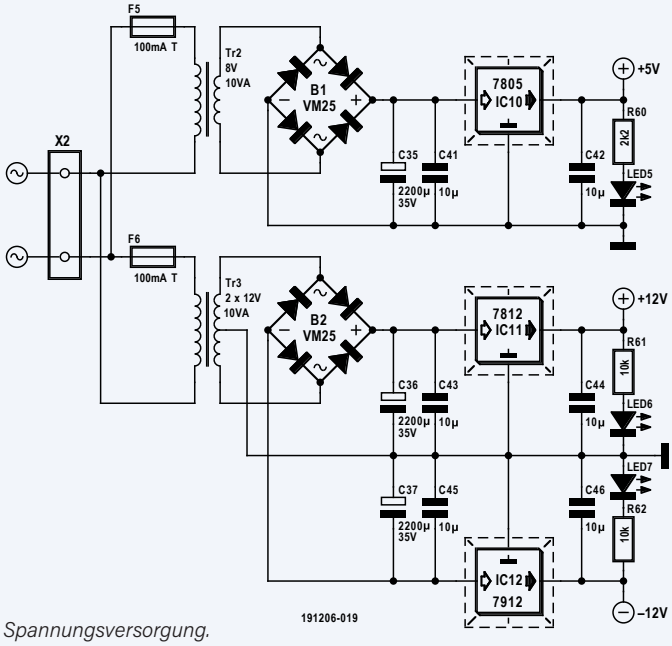
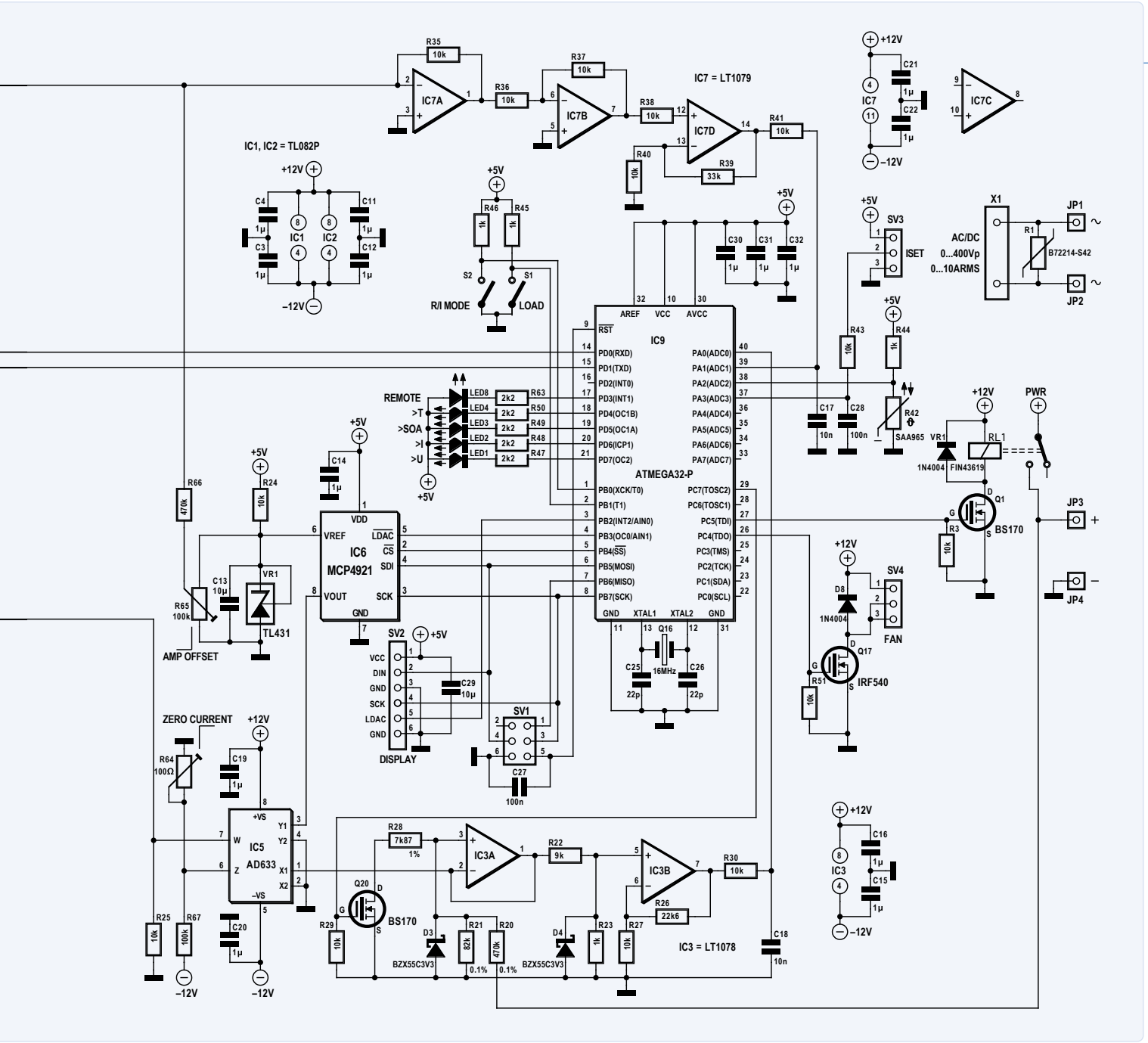


Bild 3. Die Spannungsversorgung.

Die echten Effektivwerte für Spannung und Strom werden alle 20 ms berechnet. Es können Wechselspannungen und Ströme bis zu 100 Hz gemessen werden. Zur „Glättung" der echten Effektivwerte wird ein Mittelwert aus zehn aufeinanderfolgenden Messungen gebildet, so dass eine komplette Messung 200 ms umfasst.

In der Firmware ist die Berechnung des Zielstroms abhängig von der Einstellung des Schalters R-const/I-const. Im Konstantstrommodus (I-const) muss der Strom konstant sein. Und im Konstantwiderstandsmodus muss der Widerstand durch Variation des Stroms auf $I = U_{input}/R\text{-const}$ konstant



gehalten werden. Um einen konstanten Strom zu erreichen, muss der DAC-Wert in Abhängigkeit vom eingestellten Strom und der aktuellen (Effektivwert-) Spannung berechnet werden.

Aufgrund von Bauteiltoleranzen können Soll- und Ist-Strom voneinander abweichen. Deshalb ist in der Funktion `Set_dest_current` eine Regelfunktion mit I-Kennlinie implementiert. Ist der Soll-Strom höher als der Ist-Strom, wird der Regelwert um 5 mA erhöht. Ist der Soll-Strom kleiner als der Ist-Strom, wird diese Größe um 5 mA verringert.

In der Hauptprogrammschleife werden zuerst die Fehlerbedingungen überprüft:

- **Spannung** > 400 V: LED **>U** leuchtet, Lastrelais aus, wird wieder eingeschaltet, wenn Spannung < 400 V.
- **Strom**: eine Sekunde nach Änderung des Soll-Stroms prüfen, ob Istwert < Soll-Strom + 100 mA. Wenn nicht: LED **>I** an, Lastrelais aus. Beim Wiedereinschalten der Elektronischen Last wird es wieder eingeschaltet.
- **Leistung**: Wenn Spannung * Soll-Strom > 200 W, LED **>SOA** an, Lastrelais wird ausgeschaltet. Das Lastrelais wird nach einem Wiedereinschalten der Elektronischen Last wieder eingeschaltet.
- **Temperatur**: Bei einer Kühlkörpertemperatur (gemessen mit einem an ADC2 angeschlossenen NTC) von 40°C wird der Lüfter eingeschaltet. Er wird auch eingeschaltet, wenn die Leistung 100 W überschreitet. Wenn die Temperatur unter 35°C fällt und die Leistung länger als 10 s unter 100 W liegt, wird der Lüfter ausgeschaltet. Wenn die Kühlkörpertemperatur 80°C überschreitet, leuchtet die LED **>T** auf und das Lastrelais wird ausgeschaltet. Wenn die Kühlkörpertemperatur unter 75°C fällt, wird das Lastrelais wieder eingeschaltet.

Diese Überprüfungen werden alle 500 ms durchgeführt. Auch die Spannungs- und Stromwerte auf dem Display werden alle 500 ms aktualisiert.

Wenn die Elektronische Last nicht ferngesteuert wird, erfolgt eine Ermittlung der Schalterstellung *Load On/Off* und *I-const/R-const*, ebenso des Potentiometerwerts für die Stromeinstellung.

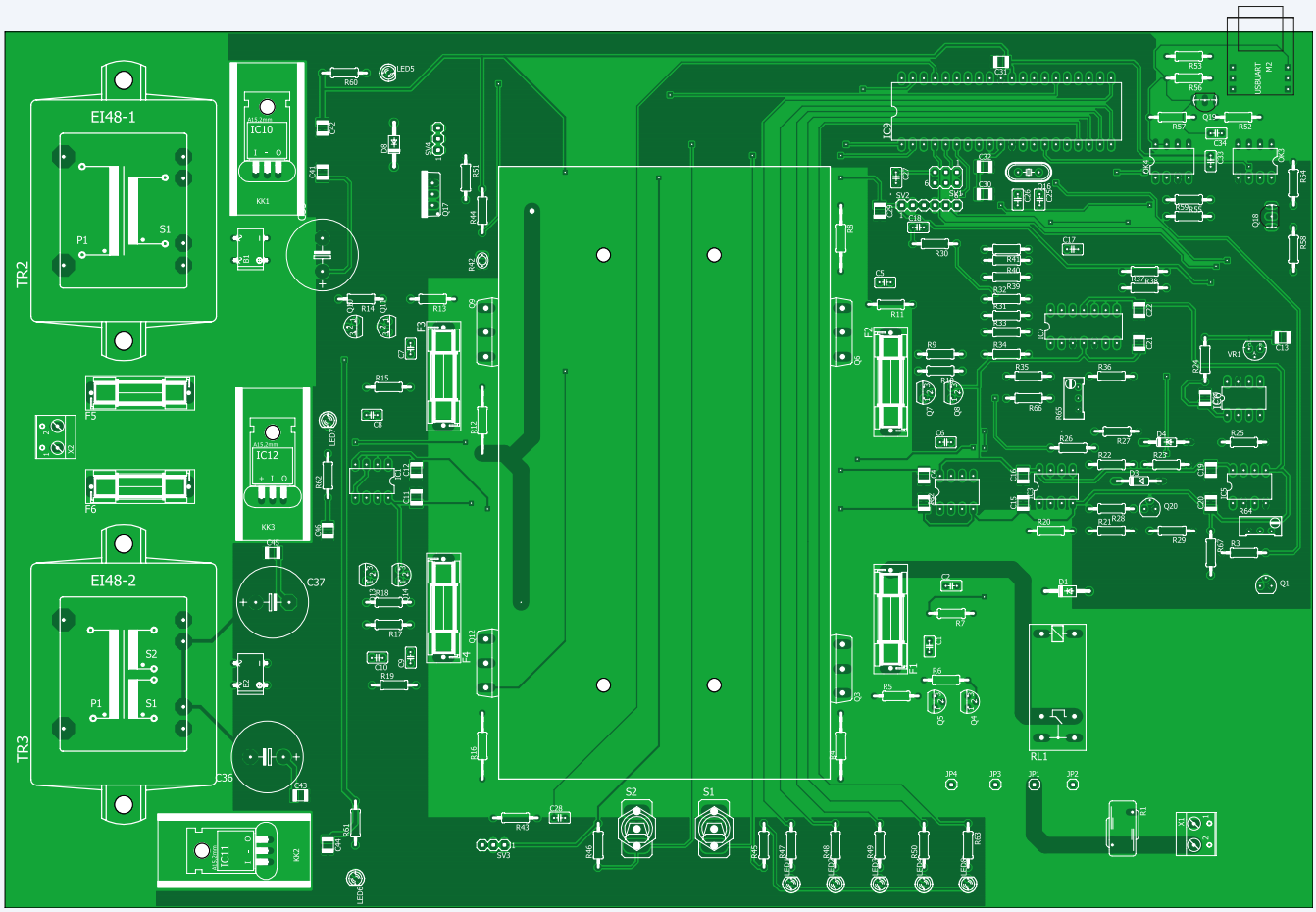


Bild 4. Platinenlayout der Hauptplatine (65% der tatsächlichen Größe).

Einstellen des Stroms/ Widerstandes

Nach dem Einschalten der elektronischen Last in der Betriebsart *I-const* beträgt der Strombereich 1 A. Nach dem Umschalten des Lastschalters auf *Off* wird der Strombereich entsprechend der Eingangsspannung eingestellt. Der Strombereich beträgt 10 A, wenn die Eingangsspannung kleiner als 20 V ist, andernfalls beträgt der Strombereich 200 W/aktuelle Spannung.
Im Modus *R-const* wird der minimale Widerstand auch dann berechnet, wenn der Lastschalter auf *Off* gestellt ist, und zwar nach Eingangsspannung/Soll-Strom-Bereich. Der maximal einstellbare Widerstandswert ist der Mindestwiderstand * 100.
Hinweis: Wenn der Modus *I-const* eingestellt ist, sorgt das Potentiometer im Linksanschlag für den minimalen, im Rechtsanschlag für den maximalen Strom. Um einen Überstrom beim Umschalten der Modi *I-const* zu *R-const* zu verhindern, entspricht der Linksanschlag des Potis dem maximalen Widerstandswert.

Aufbau der Hardware

Die Eagle- und die Gerberdateien zur Bestellung der Leiterplatten bei Ihrem bevorzugten Leiterplattenhersteller stehen unter [1] zum Download bereit. Das Layout der Hauptplatine ist in **Bild 4** dargestellt. Auf der Elektor-Labs-Seite für die Displayplatine finden Sie auch die Bauteillisten für beide Platinen. Wie die Platinen in ein Gehäuse eingebaut werden können, zeigt **Bild 5**. Die Leistungs-MOSFETs sind mit TOP3-Wärmeleitscheiben und Wärmeleitpaste auf dem Kühlkörper montiert. Die Diodenbrücke für die AC-Messungen ist ebenfalls auf diesem Kühlkörper angebracht und mit JP1..JP4 auf der Hauptplatine verdrahtet. Der Kühlkörper selbst wird mit 2-mm-Kunststoff-Distanzhaltern auf der Platine befestigt.

Steuerungssoftware

Zur Steuerung der elektronischen Last über USB wurde eine Anwendung in Visual Basic (für Visual Studio 2015) geschrieben. Die ausführbare Datei heißt *Electronic_ACDC_Load.exe*. **Bild 6** zeigt die Hauptbedienoberfläche. Diese Anwendung (sowohl Quellcode als auch ausführbare Datei) kann ebenfalls von [1] heruntergeladen werden. Auf dieser Webseite finden Sie auch die vollständige Dokumentation dieses Projekts (*Description.PDF*), die auch eine Tabelle mit allen verfügbaren Befehlen zur Kommunikation mit der Elektronischen Last über die serielle Schnittstelle enthält. Diese Informationen dürften sehr nützlich sein, wenn Sie die PC-Software ändern wollen.
Das Programm verbindet sich mit der Last, indem es alle verfügbaren COM-Ports scannt und den ID-String jede Sekunde (bis zu drei Mal) anfordert. Wenn das Programm den ID-String empfängt, wird er in der Statuszeile zusammen mit dem COM-Port angezeigt.
Danach werden alle Daten für Spannung, Strom, Leistung und Kühlkörpertemperatur im Sekundentakt abgefragt und angezeigt. Zum Einstellen des Soll-Stroms/Widerstandes muss die Schaltfläche *Remote* auf *On* gestellt werden. Dann kann das Lastrelais

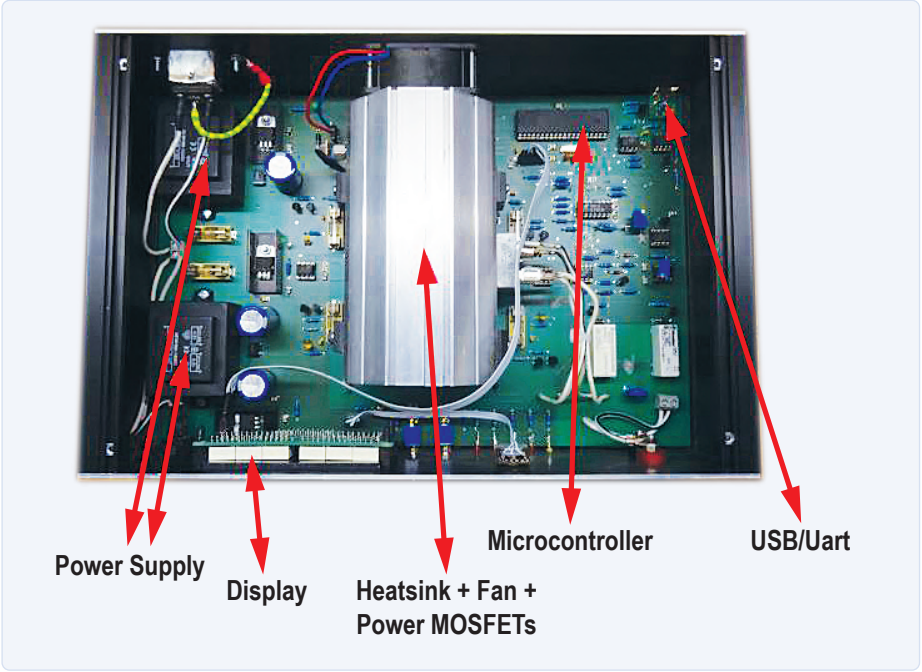


Bild 5. Übersicht der E-Load im Gehäuse.

(*Load On/Off*) und der R-*const*/*I-const*-Modus vom Programm eingestellt werden. Mit der Taste *mARMS* kann die Anzeige von *A* auf *mA* und umgekehrt umgeschaltet werden (nur im *I-const*-Modus). Mit den *Up/Down*-Tasten wird die Stromeinstellung erhöht beziehungsweise verringert. Die Werte verändern sich zunächst in 10-mA-Schritten, bald in 100-mA- und schließlich in 1-A-Schritten. Nach Loslassen der Tasten geht es wieder mit 10-mA-Änderungen weiter (nur im *I-const*-Modus).
Im *R-const*-Modus kann die Widerstandseinstellung erhöht oder verringert werden. Erhöhen ist unbegrenzt möglich, verkleinern bis zum Mindestwiderstand. Wie bei der Stromeinstellung kann der Widerstand in 1-Ω-, 10-Ω- und schließlich 100-Ω-Schritten geändert werden. Nach dem Loslassen der Tasten wird der Änderungsschritt wieder auf 1 Ω zurückgesetzt.

Dynamischer Lasttest

Hier können Sie zwei Ströme (*I-const*-Modus) oder zwei Widerstände (*R-const*-Modus) einstellen, beispielsweise um Netzteile zu testen. Obwohl die PC-Anwendung die Wellenformen der Stromquelle anzeigt, sollten Sie ein Oszilloskop an die Ausgangsklemmen des Netzteils anschließen, um dessen dynamisches Verhalten wirklich untersuchen zu können. Mit *Period* können Sie ein Intervall in Sekunden einstellen. Nach dem Drücken von *Start* werden die beiden Soll-Ströme/Widerstände vom ersten Wert auf den zweiten umgeschaltet und so weiter, bis Sie *Stop* drücken. Dann wird der Strom auf *0* im *I-const*-Modus oder auf den maximalen Widerstand im *R-const*-Modus gesetzt.

Anzeige von Spannungs-, Strom- und Leistungsmesskurven

Nach Drücken der Taste *Start Trace* können die zuvor aktivierten Messkurven *Actual Current*, *Actual Voltage* und *Actual Power* angezeigt werden. Die Aufzeichnungsgeschwindigkeit kann mit der Taste *Sampling time msec* eingestellt werden. Es werden maximal 60 Punkte angezeigt, danach werden die Messkurven gelöscht und eine neue Aufzeichnung beginnt. Das gleiche geschieht, wenn Sie einen beliebiges Zielwert ändern. Die Auflösung der vertikalen Achse kann mit *Set Y-Range* geändert werden.

Log-Datei

Durch Anklicken des Disketten-Symbols kann eine Log-Datei zur Langzeitaufzeichnung der Daten geöffnet werden. Die Daten werden im CSV-Format gespeichert, so dass Sie die Daten leicht in Excel oder OpenOffice Calc weiterverarbeiten können. Durch Klicken auf das Timer-Symbol kann die Abtastzeit für die Log-Datei geändert werden. Eine geöffnete Log-Datei kann durch erneutes Anklicken des Disketten-Symbols geschlossen werden.

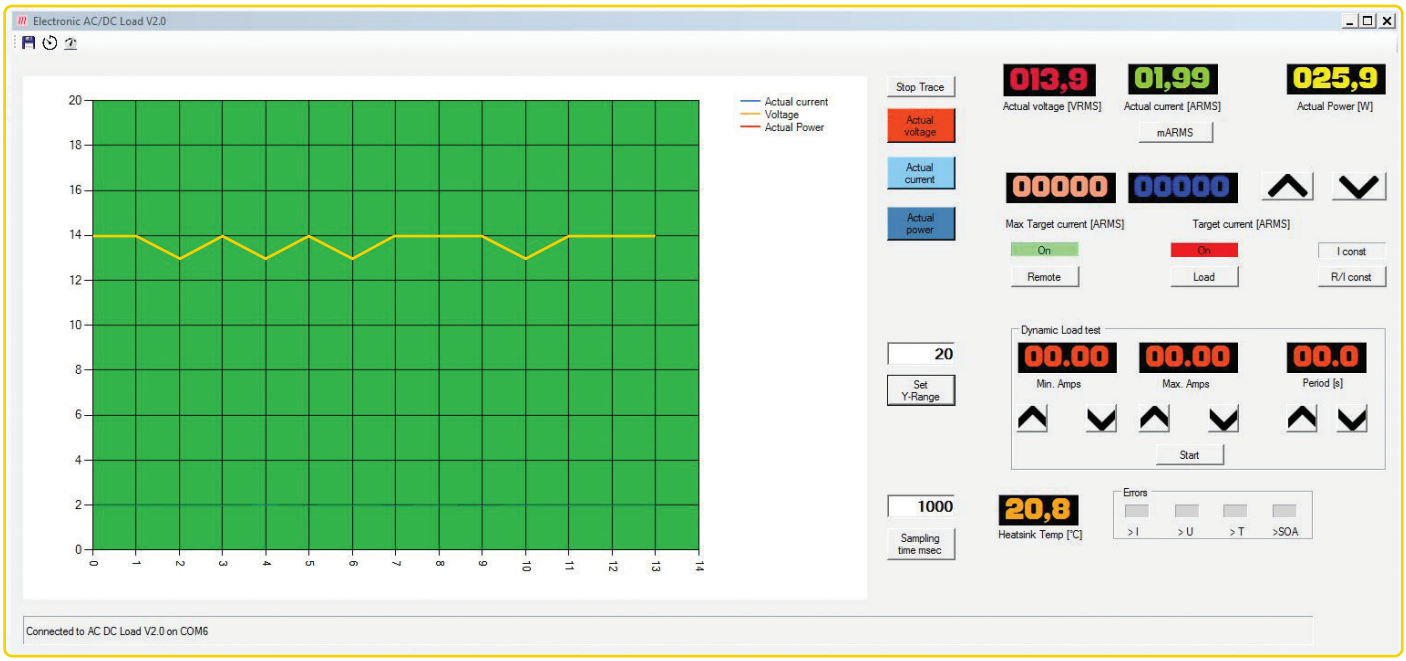


Bild 6. Hauptbildschirm des PC-Programms.

Kalibrierung

Durch Klicken auf das Gerätesymbol kann die Anzeige der elektronischen Last kalibriert werden. Für die Messung von Spannung und Strom benötigen Sie kalibrierte Prüfmittel. Durch Ändern der Werte für *Current display calibration value* und *Voltage display calibration value* können Sie die Anzeigewerte an die Messwerte der kalibrierten Messgeräte anpassen. Die Werte werden im EEPROM des Mikrocontrollers gespeichert.

Für viele Elektronik-Ingenieure dürfte eine Elektronische Last zwar nicht das am häufigsten verwendete Messgerät sein, aber es ist sehr nützlich, ein solches Instrument zur Hand zu haben, wenn Sie eine Stromversorgung bauen, testen oder reparieren. Und obwohl es sie auch fertig im Handel gibt, ist der Selbstbau eines solchen Geräts definitiv eine Überlegung wert!

Dieser Artikel basiert auf den Informationen, die auf der Elektor-Labs-Projektseite unter [1] zu finden sind. Auf dieser Seite können detailliertere Informationen zu dieser Elektronischen Last einschließlich aller Software, Platinenlayout-Dateien und Stücklisten gefunden und heruntergeladen werden. Ein Video, das diese Elektronische Last im Betrieb zeigt, gibt es unter [3].

191206-03

Haben Sie Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Schicken Sie eine E-Mail an den Autor unter rainerschuster@mnet-mail.de oder kontaktieren Sie Elektor unter editor@elektor.com!

Ein Beitrag von

Idee, Entwurf, Text und Illustrationen: **Rainer Schuster**

Redaktion: **Luc Lemmens**

Schaltbilder: **Patrick Wielders, Kurt Diedrich**

Übersetzung: **Rolf Gerstendorf**

Layout: **Giel Dols**

WEBLINKS

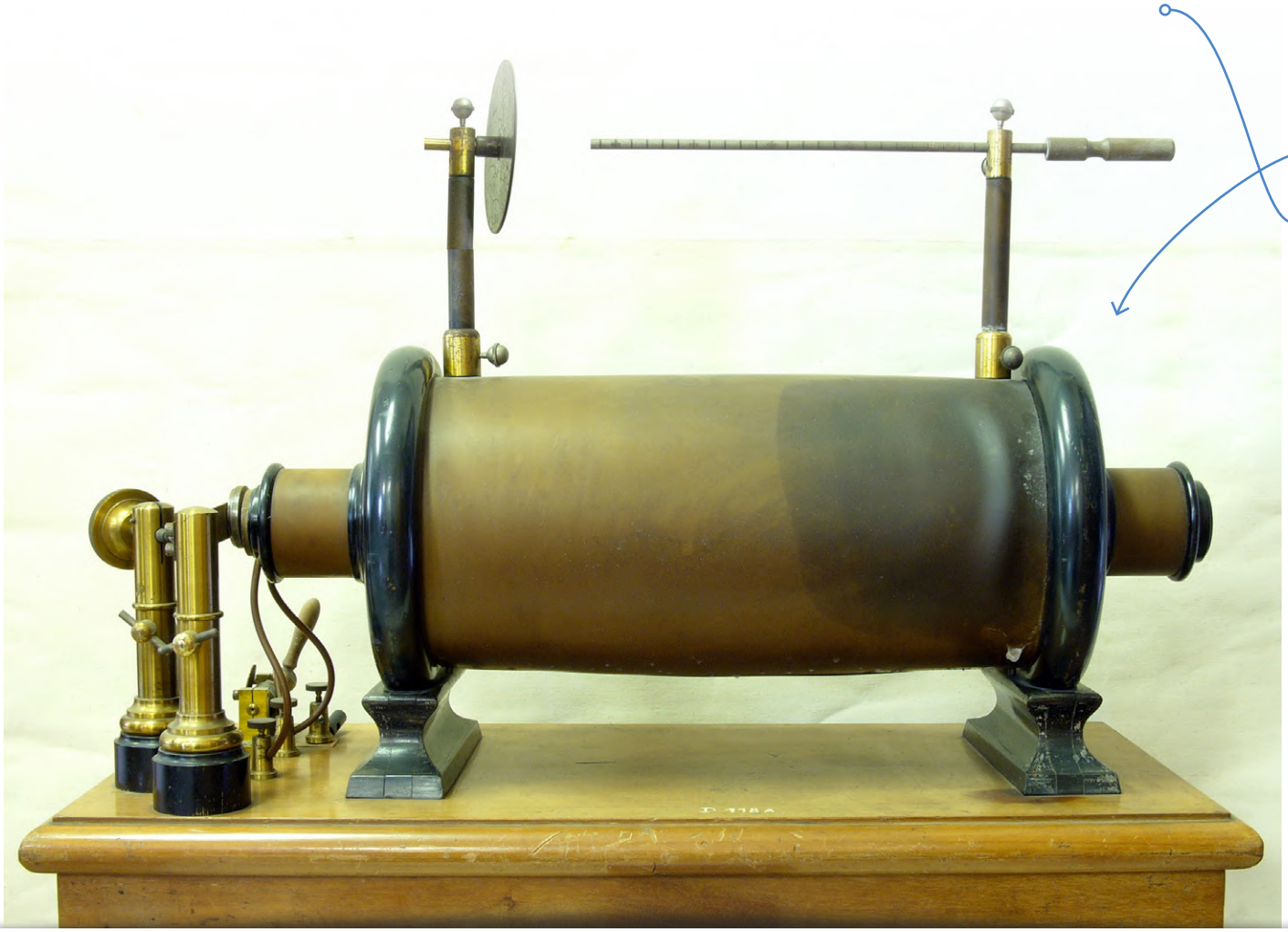
[1] **Elektor Labs-Seite zu diesem Projekt:** www.elektormagazine.de/labs/electronic-load-for-dc-and-ac

[2] **6-Digit-Display mit SPI:** www.elektormagazine.de/magazine/elektor-200907/19106

[3] **Video:** www.electronics-lab.com/ac-dc-200w-electronic-load/

Aller Anfang ...

ist gar nicht schwer (auch wenn es sich um Spulen dreht)!



Funkeninduktor (Quelle: Hannes Grobe, https://commons.wikimedia.org/wiki/File:Induktionsapparat_hg.jpg)

Von **Eric Bogers** (Elektor)

Seien wir mal ehrlich: Die meisten (Hobby-) Elektroniker hassen Spulen. Fertige Drosseln zur Rauschunterdrückung sind ja gerade noch erträglich, aber wenn man sie selbst wickeln muss, wird es schnell zu viel. Meist bedarf es unzähliger Versuche, bis man auch nur in die Nähe der gewünschten Eigenschaften kommt.

Spulen, Drosseln, Induktivitäten, wie man sie auch nennen mag, sind in jeder Hinsicht das Gegenteil von Kondensatoren. Bei einem Kondensator nimmt die Impedanz (Wechselstromwiderstand) mit zunehmender Frequenz ab, bei einer Spule dagegen nimmt die Impedanz mit zunehmender Frequenz zu. Und während bei einem Kondensator der Strom der Spannung vorauseilt (siehe die vorherige Folge von „Aller Anfang ...“), hinkt bei Spulen der Strom der Spannung nach.

In **Bild 1** sind ein paar „Lehrbuchbeispiele“ für Spulen dargestellt. Ganz links ist eine (axiale) Filterdrossel dargestellt, die aufgrund ihres Aufbaus und der farbigen Ringe leicht mit einem Widerstand verwechselt werden kann, um so mehr, als dass der Farbcode für diese Drosseln vergleichbar ist mit dem von Widerständen.

In der Mitte oben befindet sich eine Ringkerndrossel, wie sie besonders häufig in Entstörnetzwerken zu finden ist. Und oben rechts eine Luftspule in voller Pracht - diese finden wir oft in Frequenzweichen in Lautsprechern.
Unten schließlich sehen wir einen Spulenkörper mit Ferritkern für Hochfrequenzanwendungen. Hier müssen wir den Draht selbst auf den Spulenkörper wickeln und können so in gewissen Grenzen die Induktivität der Spule frei bestimmen. Nebenbei: Ich fand es immer sehr verwirrend, dass sowohl das konkrete Bauelement als auch dessen wichtigste Eigenschaft als Induktivität benannt wird.

Die Induktivität als Energiereservoir

Wenn ein Strom durch eine Spule fließt, entsteht um die Spule ein Magnetfeld, in dem Energie gespeichert wird. Wird der Strom durch die Spule dann unterbrochen, wird diese Energie wieder freigesetzt: Es entsteht eine Spannung, die versucht, den ursprünglichen Strom aufrecht zu erhalten.
Das Maß für die Fähigkeit einer Spule, Energie zu speichern, nennt man die Induktivität L (ausgedrückt in Henry und abgekürzt H). Für die Energiemenge, die in einer Spule gespeichert wird, gilt:

$$W = \frac{1}{2} \cdot L \cdot I^2$$

Sie werden wahrscheinlich die Ähnlichkeit mit der entsprechenden Formel für einen Kondensator bemerken - aber bei einer Spule gibt es einen Strom I und bei einem Kondensator eine Spannung U.

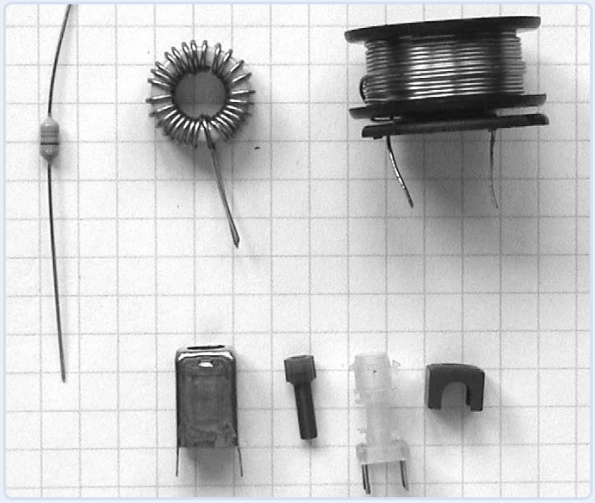


Bild 1. Ein paar Induktivitäten.

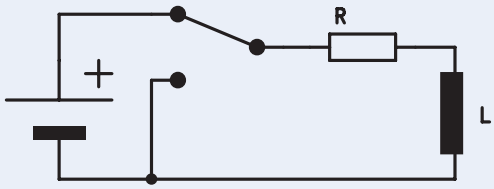


Bild 2. Umkehr der Stromrichtung in/aus einer Spule.

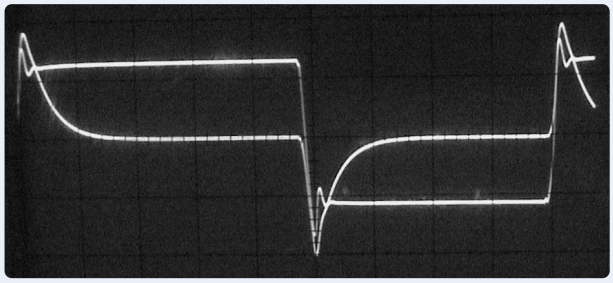


Bild 3. Shoot-Through beim Ein- und Ausschalten einer H-Brücke.

Einschalt- und Ausschaltverhalten

Auch das Ein- und Ausschaltverhalten einer Spule ist dem Lade-/Entladeverhalten eines Kondensators entgegengesetzt. Beim Anlegen einer Spannung an eine Spule fließt zunächst kaum Strom, die Spule ist also hochohmig und die Spannung über der Spule am höchsten. Der Strom nimmt jedoch langsam zu, während die Spannung über der Spule entsprechend abnimmt. Der Strom durch die Spule nähert sich asymptotisch einem Maximalwert (der durch den Widerstand der Wicklung bestimmt wird), und die Spannung über der Spule sinkt auf ein Minimum, das durch ihren Drahtwiderstand bestimmt wird.
Besonders interessant ist das Verhalten der Spule im Schaltzeitpunkt: Wenn die Polarität der angelegten Spannung umgekehrt wird (**Bild 2**), bricht das Magnetfeld der Spule zusammen und es wird eine Spannung mit einer Polarität induziert, die versucht, den Strom in der ursprünglichen Richtung aufrecht zu halten (Selbstinduktion). Diese Polarität ist also der der ursprünglich angelegten Spannung entgegengesetzt. Die Folge ist, dass im Schaltzeitpunkt eine erhebliche Rücklaufspannung (englisch: flyback voltage) auftritt (**Bild 3**).
Induktivitäten können beim Abschalten also Spannungen erzeugen, die wesentlich höher sind als die ursprünglich angelegte Spannung. Bei Kondensatoren sehen wir etwas Vergleichbares, da diese wesentlich höhere Ströme liefern können als den, der zum Aufladen verwendet wurde.
Diese Eigenschaft von Spulen kann man sich zunutze machen, um aus einer niedrigen eine hohe Spannung zu erzeugen. Ein Beispiel dafür ist die Induktionsspule [1]. Haben Sie zufällig noch ein altes Auto mit einem Benzinmotor? Die Hochspannung für die Zündkerzen wird von der Zündspule geliefert, die genau nach diesem Prinzip arbeitet.
Die Kehrseite der Medaille und ein unerwünschter Effekt dieses Phänomens ist, dass die induzierte Spannung andere Bauteile in einem Stromkreis zerstören kann. Ein bekanntes Beispiel ist ein Relais (ein elektromagnetischer Schalter, bei dem das Magnetfeld einer Spule zur Betätigung der Schaltkontakte genutzt wird), das von einem Transistor gesteuert wird. Beim Ausschalten des Relais wird eine Gegenspannung induziert, die diesen Transistor nicht nur zerstören kann, sondern dies in vielen Fällen auch wird. Eine Lösung für dieses Problem ist die sogenannte Freilaufdiode, die die induzierte Spannung kurzschließt und damit unschädlich macht (**Bild 4**).

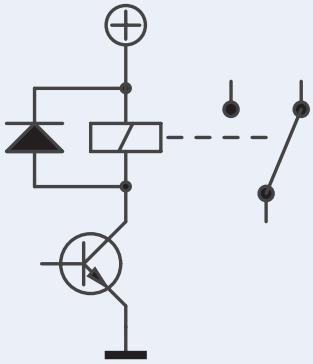


Bild 4. Die Diode schützt den Transistor, wenn der Strom durch das Relais abgeschaltet wird.

Reihen- und Parallelschaltungen

Ebenso wie Widerstände und Kondensatoren können auch Induktivitäten parallel und in Reihe geschaltet werden. In dieser Hinsicht verhalten sie sich genauso wie Widerstände (was einleuchtend ist, wenn man darüber nachdenkt: bei Reihenschaltung erhöht sich die Windungszahl). Für die Reihenschaltung gilt also Folgendes:

$$L_{tot} = L_1 + L_2 + L_3 + \ldots + L_n$$

Und für die Parallelschaltung ist:

$$L_{tot} = \frac{1}{\frac{1}{L_1} + \frac{1}{L_2} + \frac{1}{L_3} + \ldots + \frac{1}{L_n}}$$

Die Spule als Wechselstromwiderstand

Auch in einem Einsatz als Wechselstromwiderstand verhält sich eine Spule genau umgekehrt wie ein Kondensator. Bei einem Kondensator sinkt die Impedanz mit steigender Frequenz, bei einer Spule steigt die Impedanz mit der Frequenz:

$$X_L = 2 \cdot \pi \cdot f \cdot L$$

Eine Spule führt auch zu einer Phasenverschiebung zwischen Spannung und Strom: Der Strom eilt der Spannung um 90° nach.

Induktion

Wenn sich eine Spule in einem variierenden Magnetfeld befindet, wird eine Spannung in dieser Spule induziert. Wir können die Höhe dieser induzierten Spannung mit dieser Formel

$$U_{ind} = n \cdot A \cdot B' \cdot \cos\alpha$$

berechnen. Dabei ist n die Anzahl der Windungen in der Spule, A die von der Spule eingenommene Querschnittsfläche, B' die zeitliche Ableitung der magnetischen Flussdichte und α der Winkel zwischen der Ebene der Windungen und dem Magnetfeld.
Wenn wir es mit einem sich sinusförmig ändernden Magnetfeld zu tun haben, können wir für die Ableitung der Flussdichte schreiben:

$$B' = 2 \cdot \pi \cdot f \cdot B \cdot \cos\phi$$

Ohne Induktion sähe die Welt ganz anders aus. Transformatoren, in denen die Wechselspannung in der Primärwicklung ein veränderliches Magnetfeld erzeugt, das wiederum eine Spannung in der Sekundärwicklung induziert, würden nicht funktionieren, ebenso gäbe es keine Generatoren (man denke an die Lichtmaschine unter der Motorhaube).
Bei Audioanwendungen kann die Induktion jedoch auch störend sein: Eine unbeabsichtigte Erdschleife (die vielleicht durch nicht korrekt zusammengeschaltete Audiogeräte entstanden ist) „sammelt" Störfelder im Raum ein und lässt sie durch ein irritierendes Brummen im Lautsprecher ertönen.
Über Spulen kann ja noch so viel mehr gesagt werden, und genau das wollen wir in der nächsten Folge von „Aller Anfang …" auch tun!

210342-03

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

Haben Sie Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Senden Sie eine E-Mail an den Autor oder 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: **Harmen Heida**

WEBLINK

[1] Funkeninduktor:
https://de.wikipedia.org/wiki/Funkeninduktor

Nvidia Jetson Nano - BILDVERARBEITUNG für Einsteiger

Teil 1: Hard- und Software im Überblick

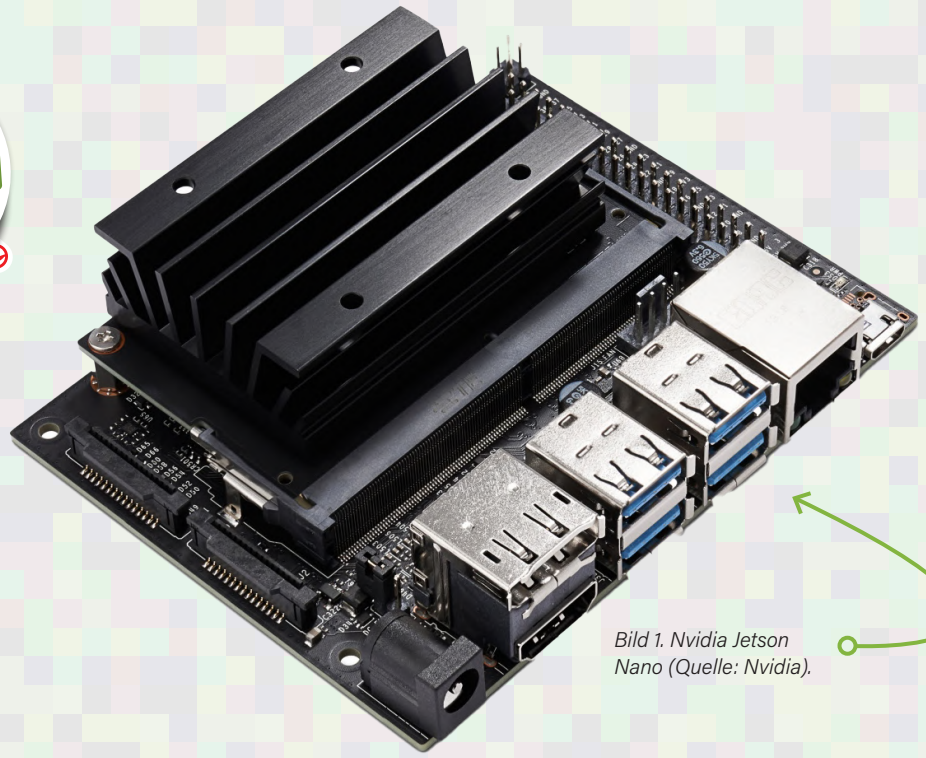


Bild 1. Nvidia Jetson Nano (Quelle: Nvidia).

Von **Mathias Claußen** (Elektor Lab)

Um die Beschäftigung mit KI kommt wohl kein Entwickler mehr herum. Für erste Schritte eignen sich zum Beispiel der Sipeed Maixduino oder der Maxim MAX78000, die wir bereits in Elektor vorgestellt haben. Der Nvidia Jetson Nano ist eine deutlich leistungsfähigere Hardware aus der Klasse der Single Bord Computer, ausgestattet mit einem Vierkern-ARM-Prozessor, 4 GB RAM und einer GPU mit 128 CUDA Kernen. Mit einer kleinen Artikelserie wollen wir zeigen, welche Anwendungen möglich sind, zum Beispiel im Bereich der Bildverarbeitung oder Robotik.

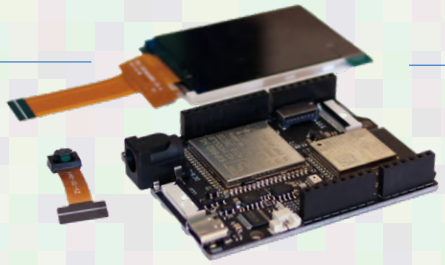


Bild 2. Sipeed MAix Bit mit K210 MCU.

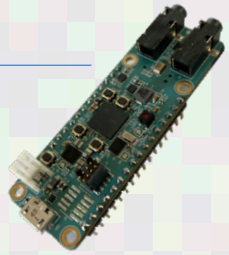


Bild 3. MAX78000 FTHR-Board mit AI-Beschleuniger.

Bild 4. Raspberry Pi 4 Board (Quelle: Raspberry Pi Foundation).

KI hält mehr und mehr Einzug in unseren Alltag. Neuronale Netze ermöglichen es, Objekte zu erkennen, Bilder zu verbessern oder die Worte einer Unterhaltung in Text umzuwandeln. Dieses spannende Feld neuer Möglichkeiten wird unsere Zukunft mitbestimmen. Damit unsere Leser (und wir) erste eigene Schritte in Sachen KI unternehmen können, haben wir uns schon den Sipeed Maixduino (**Bild 1**) [1] und das Maxim MAX78000 FTHR-Board (**Bild 2**) [2] angeschaut. Beiden Boards ist gemein, dass Mikrocontroller und Beschleuniger für Neuronale Netzwerke verbaut sind, jedoch mit beschränkten Ressourcen.

Das Nvidia Jetson Nano (**Bild 3**) [3] ist eine deutlich leistungsfähigere Hardware aus der Klasse der Single Board Computer (SBC). Ausgestattet mit einem Vierkern-ARM-Prozessor, 4 GB RAM und einer GPU mit 128 CUDA Kernen stellt er eine gute Plattform dar, um mit dem Thema KI und Neuronale Netze zu beginnen. Dabei reichen die Anwendungen von der Spracherkennung über die Klassifizierung von Objekten in aufgenommenen Videobildern bis hin zur ausgewachsenen Robotersteuerung. Für alle Bereiche steht reichlich „Zubehör" zur Verfügung - zum Beispiel die Roboterplattform JetBot von Sparkfun.

Die GPU macht den Unterschied

Beim Nvidia Jetson Nano liegt ein Vergleich mit dem beliebten Raspberry Pi 4B (**Bild 4**) nahe. Beides sind SBCs, beide haben vier Prozessorkerne, und beide sind mit 4 GB RAM erhältlich. Auf beiden Boards lässt sich auch ein Ubuntu 20.04 installieren und auch eine Stiftleiste für Erweiterungen ist beiden gemeinsam. Beim Raspberry Pi 4 ist ein Broadcom VideoCore6 als GPU (Graphics Processing Unit) verbaut. Im Jetson Nano steckt eine Maxwell-basierte GPU mit 128 CUDA Kernen (GM20B). Diese Kerne geben dem Jetson Nano seine Leistungsfähigkeit. Die CUDA-Kerne stellen programmierbare Rechenwerke innerhalb der GPU dar, die nicht nur zur Verarbeitung von Grafikdaten gedacht sind, sondern generische Rechenoperation parallel ausführen können - also auch jene, die zum Trainieren und Anwenden eines Neuronalen Netzes benötigt werden.

Training mit dem Jetson

Hier erkennt man den Unterschied zu einem MAX78000: Das Training des Neuronalen Netzes sollte dabei auf einem Linux-PC erfolgen, der am besten noch mit einer CUDA-fähige Nvidia-Karte ausgestattet ist. Dafür kommen PyTorch und weitere Software-Tools zum Einsatz.

Der Jetson dagegen hält schon alle Zutaten bereit, um ein Neuronales Netz selbst trainieren zu können. Sicherlich geht das nicht so schnell wie auf einem aktuellen PC mit einer Geforce 1660 oder Geforce 1050Ti. Dafür erhält man den Jetson Nano aber schon für etwa 100 Euro. Eine Grafikkarte, wenn momentan überhaupt verfügbar, verschlingt allein das Zweifache.

Nvidia stellt eine große Zahl von Tutorials [4] [5] und selbstverständlich Software [6] zur Verfügung. Sie zeigen unter anderem, wie sich mit Hilfe einer Webcam Neuronale Netze für die Bilderkennung/-verarbeitung auf dem Nvidia Jetson Nano anlernen lassen.

Jetson Nano und der fahrende JetBot

Wenn man sich mit der Umgebung vertraut gemacht hat, mag der Wunsch nach einer handfesten Anwendung aufkommen. Durch den Fokus auf Bildverarbeitung bietet sich ein selbst fahrender Roboter an. Der JetBot (**Bild 5**) [7] von Nvidia ist ein solcher; maker-freundlich sind eine Bauteileliste sowie 3D-Modelle für den Selbstdruck der Teile verfügbar. Ein fahrender Roboter mit Objekterkennung und Interaktion erlaubt es, die Intelligenz der selbst trainierten Neuronalen Netze direkt in Aktion zu sehen. Das Webinar „Ai for

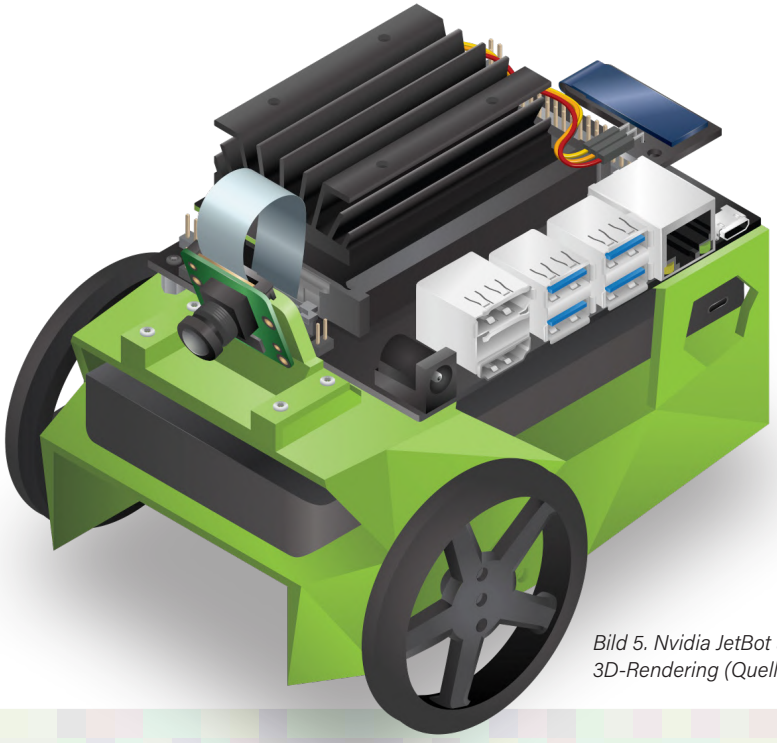


Bild 5. Nvidia JetBot als 3D-Rendering (Quelle: Nvidia).

Makers - Learn with JetBot" demonstriert, wie man einem solchen Roboter eine Interaktion mit der Umgebung beibringen kann [8].

Entwicklung per Webbrowser

Für die Entwicklung der Software des JetBot ist ein Webbrowser ausreichend. Die Entwicklungsumgebung, die im Browser läuft, ist das JupyterLab (**Bild 6**) [9], sie erlaubt das Anlegen, Editieren und Ausführen von Python-Skripten. Dies erlaubt auch Zugriff auf etliche Bibliotheken von Drittanbietern. Motortreiber oder Sensoren, z.B. von Sparkfun oder Adafruit, lassen sich direkt aus Python ansprechen.

Mit dem JupyterLab kann dann auch das Training von Neuronalen Netzwerken gestartet werden. Es lassen sich auch neue Trainingsdaten für die Neuronalen Netzwerke generieren, um das Verhalten eines Roboters mit Daten aus dem laufenden Betrieb zu verbessern und neue Funktionen hinzuzufügen. Die IDE hat man ja immer in einem Browser dabei.

Die Steuerung des Roboters und andere Dienste immer von Grund auf neu zu entwickeln, ist zeitraubend und nicht notwendig. Als flexible Basis für Roboter aller Art kann das ROS™ (Robot Operation System) [10] eingesetzt werden. Es erlaubt, auf einem definierten Kern und einer gut dokumentierten Softwarebasis aufzubauen. ROS als Betriebssystem auf Linux-Basis ermöglicht aber auch Funktionen wie das Erstellen von Karten der Umgebung und andere Aktionen. Für den Nvidia Jetson und JetBot steht ein passendes OS-Image mit vorinstalliertem ROS zur Verfügung.

All in One Kit - SparkFun JetBot AI Kit v2.1

Images downloaden, den Jetson Nano vorbereiten, Bauteile für den JetBot von unterschiedlichen Quellen beschaffen und eventuell noch 3D-drucken? Die einzelnen Softwareteile passend konfigurieren und dann erst mit dem Thema KI anfangen? Einfacher ist es, ein fertiges Kit zu nutzen, um gleich eine definierte und funktionstüchtige Hardware zu haben, so dass man sich ganz in das Thema KI vertiefen kann. Das SparkFun JetBot AI Kit v2.1 (**Bild 7**) beinhaltet die gesamte Hardware, die nötig ist, um mit dem Thema KI und Robotik zu starten. Basierend auf dem JetBot von Nvidia erhält man so Zugriff auf ein großes Ökosystem an Tutorials und Software, mit der gewohnt guten Hardwaredokumentation von SparkFun [11]. Auch das Zusammenstellen von Softwarekomponenten entfällt. Das Kit enthält eine fertige SD-Karte, die schon alles Nötige beinhaltet. Die Anschaffung eines solchen Kits mag mit etwa 270 € hochpreisig erscheinen. Jedoch beinhaltet es den Jetson Nano, die mechanischen Teile des Roboters, Motoren, Motortreiber, ein OLED, WiFi-Adapter, Kamera, SD-Karte und alle anderen Teile, die für den Start benötigt werden (**Bild 8**).

Was kommt als Nächstes?

Wie in diesem kurzen Artikel zu sehen ist, gibt es viel zu lernen und viel zu entdecken. KI zeigt viele neue spannende Wege, ein Fahrzeug mit einer autonomen Steuerung auszustatten. Für die meisten Leser müssen Begriffe wie PyTorch und ROS jedoch sicherlich noch in den passenden Kontext gebracht werden. PyTorch übernimmt das Modifizieren und Anlernen der KI. Wie jede Software will auch PyTorch passend bedient werden, und wie bei vielen Programmen aus der Linux-Welt üblich, bevorzugt mit Konfigurationsdateien. Die Steuerung des Fahrzeuges/Roboters erfolgt dann durch das ROS.

Allein der Umgang mit diesen zwei Tools bietet genug Stoff, um ganze Bücher zu füllen. Daher werden wir uns langsam an die einzelnen Bestandteile herantasten und immer weiter in die Details eintauchen!

210318-02

Ein Beitrag von

Entwicklung und Text: **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.

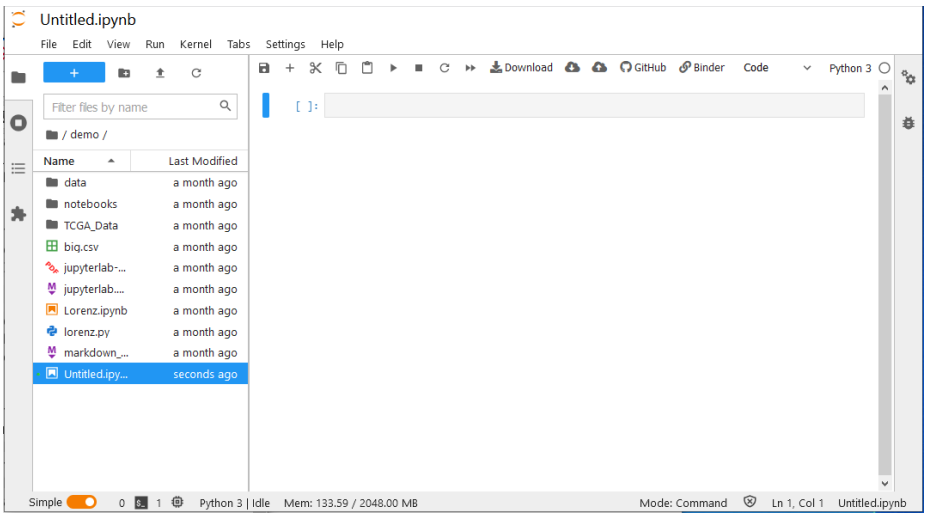


Bild 6. Jupyter Lab in einem Browser.

Bild 7. SparkFun JetBot AI Kit v2.1 (Quelle: SparkFun).

Bild 8. Teile im SparkFun JetBot AI Kit v2.1 (Quelle: SparkFun).

PASSENDE PRODUKTE

- **NVIDIA Jetson Nano Developer Kit (SKU 19001)**
  www.elektor.de/nvidia-jetson-nano-developer-kit
- **SparkFun JetBot AI Kit v2.1 (inkl. NVIDIA Jetson Nano Developer Kit) (SKU 19576)**
  www.elektor.de/sparkfun-jetbot-ai-kit-v2-1-incl-nvidia-jetson-nano-developer-kit
- **SparkFun JetBot AI Kit v2.1 (ohne NVIDIA Jetson Nano Developer Kit) (SKU 19685)**
  www.elektor.de/sparkfun-jetbot-ai-kit-v2-1-without-nvidia-jetson-nano-developer-kit

WEBLINKS

[1] **Walter Trojan, KI für Anfänger, ElektorMagazine.de**: www.elektormagazine.de/articles/ki-fr-einsteiger-1
[2] **Luc Lemmens, KI mit dem MAX78000 Feather Board: Hardware-Essentials, ElektorMagazine.de**: www.elektormagazine.de/news/ai-max78000-feather-board-de
[3] **Nvidia Jetson Nano**: https://developer.nvidia.com/embedded/jetson-nano-developer-kit
[4] **Jetson AI Fundamentals Course Outline**: https://developer.nvidia.com/embedded/learn/jetson-ai-certification-programs#course_outline
[5] **Jetson AI Fundamentals auf Youtube**: www.youtube.com/watch?v=uvU8AXY1170&t=322s
[6] **Jetson Download Center**: https://developer.nvidia.com/embedded/downloads#?search=Jetson%20Nano
[7] **Nvidia JetBot auf GitHub**: https://github.com/NVIDIA-AI-IOT/jetbot
[8] **AI for Makers - Learn with JetBot**: www.youtube.com/watch?v=zOCSRzDUI-Y&t=2248s
[9] **JupyterLab**: https://jupyter.org/
[10] **ROS.org**: https://www.ros.org/
[11] **SparkFun JetBot AI Kit v2.1 Powered by Jetson Nano**: www.sparkfun.com/products/16417

Netztransformatoren aus der Nähe betrachtet

Wie verhalten sie sich beim Ein- und Ausschalten?

Von Dipl.-Ing. (FH) Andreas R. Fecht

Beim Einschalten eines Netztransformators fließt ein Einschaltstrom, dessen Stärke vom Winkel der sinusförmigen Eingangswechselspannung zum Zeitpunkt des Einschaltens abhängt, aber auch vom Winkel zum Zeitpunkt des letztmaligen Ausschaltens.

Ziel dieses Artikels ist es nicht, noch eine Schaltung zur Vermeidung oder Verringerung des Einschaltstroms zu entwickeln, sondern nur den Effekt im Detail zu untersuchen. Da in der Literatur nur sehr wenige genaue Informationen dazu zu finden sind, soll diese Untersuchung ein für allemal Fragen zu diesem Thema umfassend beantworten.
Es gibt einige theoretische Überlegungen dazu, wie sich Transformatoren aufgrund der Remanenz des Eisenkerns verhalten. Inwieweit Theorie und Praxis wirklich übereinstimmen, kann aber nur durch Ausprobieren herausgefunden werden. Dabei wird auch der Frage nachgegangen, ob sich Ringkerntransformatoren anders verhalten als Transformatoren mit Luftspalt.

Zuerst: eine Testanordnung bauen

Um die Trafoströme im Detail zu untersuchen, wurde eine spezielle Testschaltung entwickelt, deren Blockschaltbild in **Bild 1** dargestellt ist. Die zentralen Elemente sind der Mikrocontroller und der von ihm gesteuerte MOSFET-Schalter. Die 3,3-V-Versorgung ist für die Messelektronik und den Mikrocontroller zuständig, während der Gate-Treiber für den MOSFET, ein Optokoppler, von einer galvanisch getrennten 12-V-Versorgung betrieben wird.
Die Messelektronik besteht aus zwei Rail-to-Rail-Opamps mit ausreichender Bandbreite. Die Netzspannung wird von einem Spannungsteiler in den für den Opamp erlaubten Bereich reduziert. Der Strom wird mit einem Shunt-Widerstand R1 gemessen, dessen Wert so gewählt wurde, dass der Messbereich sich bis zu ±75 A erstreckt.
Der Mikrocontroller erfasst die beiden analogen Signale der Opamps mit seinen beiden 12-Bit-AD-Wandlern gleichzeitig mit einer ausreichend hohen Abtastrate von etwa 20…50 kHz und berechnet die Ausgangsinformationen, die über vier Optokoppler galvanisch sicher getrennt zum PC gelangen.

Die Testanordnung im Detail

Das dataillierte Schaltbild der Testanordnung ist in **Bild 2** dargestellt.

Spannungsversorgungen

Zur Versorgung der Schaltung sind drei galvanisch getrennte Spannungen notwendig. Die Versorgung der Messschaltung erfolgt über den 6-V/1-W-Transformator TR1, gefolgt von einem Gleichrichter, einem Elektrolytkondensator und einem 3,3-V-Spannungsregler. Der Gate-Treiber (T4 mit IC10) wird von einem Netzteil, dem 12-V/1-W-Transformator TR2 mit Gleichrichter und Filterung, mit

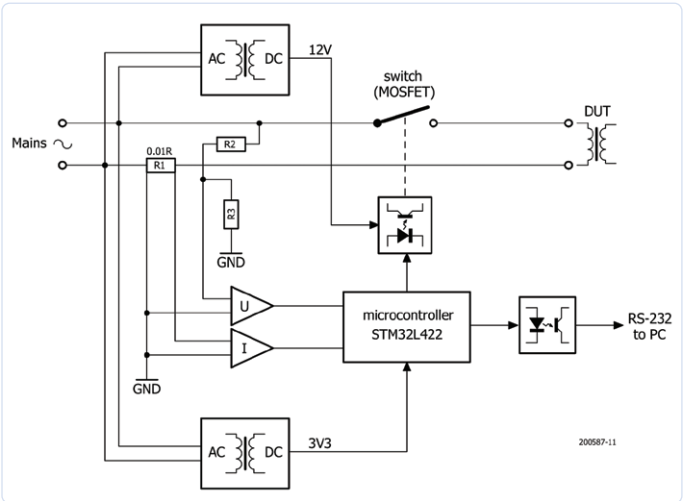


Bild 1. Blockschaltbild der Testanordnung.

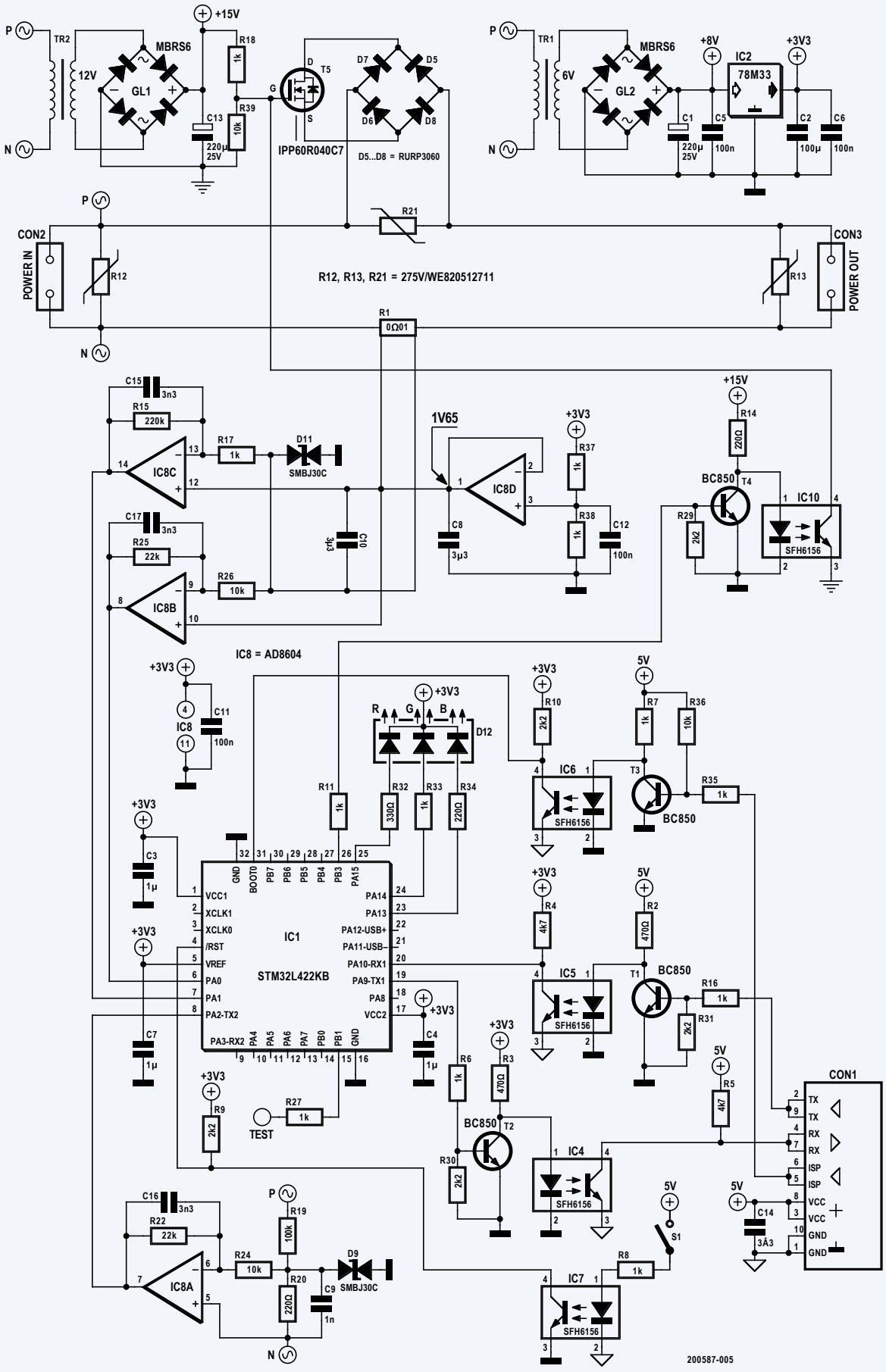


Bild 2. Das Schaltbild der Testanordnung sieht aufgrund der vielen Optokoppler, die für eine galvanische Trennung sorgen, komplizierter aus, als sie in Wahrheit ist.

Tabelle 1. Übergangswiderstand der Klemmen (gemessen) für verschiedene Ströme

Strom [A]	Widerstand bei geschlossenem Schalter [Ω]
1	1.5
2	0.85
5	0,4 (100 mΩ differentiell)
10	0,24 (80 mΩ differentiell)

Tabelle 2. Extrapolierte Kontaktwiderstände

Strom [A]	Widerstand bei geschlossenem Schalter [Ω]
20	140 (40 mΩ differentiell)
50	68 (20 mΩ differentiell)
100	40 (12 mΩ differentiell)

einer ungeregelten Gleichspannung von ungefähr 15 V versorgt. Die PC-Seite der RS232-Schnittstelle wird mittelbar (in der Verbindung sitzt ein 5-V-USB-zu-RS-232-Konverter) über die USB-Buchse des PCs versorgt.

Ein/Aus-Schalter

Als Ein/Aus-Schalter für den zu untersuchenden Trafo wurde ein MOSFET (T5) mit einem Brückengleichrichter (D5…D8) eingesetzt. Im Gegensatz zu einem Thyristor oder Triac kann ein MOSFET zu jedem Zeitpunkt einer Periode ein- oder ausgeschaltet werden. Der gewählte MOSFET kann einen Dauerstrom von 73 A und einen Spitzenstrom von 211 A verarbeiten, die verwendeten Dioden des Brückengleichrichters unterstützen allerdings nur Dauerströme von 30 A und Spitzenströme bis zu 70 A. Drei Varistoren (R12, R13, R21) mit einer Durchbruchspannung von jeweils 430 V sorgen für den Schutz gegen induzierte Spannungen.
Die Ansteuerung des MOSFETs erfolgt über den Optokoppler IC10. Die gemessene Flankenanstiegszeit beträgt 20 µs, was für diese Art der Messung mehr als ausreichend ist. Die Übergangswiderstände an den Klemmen der Schaltung für verschiedene Stromwerte sind in **Tabelle 1** (gemessen) und **Tabelle 2** (extrapoliert) aufgeführt.

Analog messen, digital auswerten

Der gewählte Mikrocontroller STM32L422KB verfügt über zwei weitgehend unabhängige 12-Bit-Analog-Digital-Wandler zur gleichzeitigen Erfassung von Spannung und Strom.
Für die Spannungsmessung mit Opamp IC8A wird zunächst die Netzeingangsspannung mit R19/R20 um den Faktor 455 geteilt. Der Opamp besitzt eine Verstärkung von 2,2 und hebt die Spannung zusätzlich um 1,65 V an. Der Eingangsspannungsbereich beträgt somit ±341,6 V. Der Ausgang des Opamps wird dem ADC-Eingang PA2 des Controllers zugeführt.
Für die Strommessung sind die Opamps IC8B…D zuständig. Um sowohl positive als auch negative Werte messen zu können, wird mit IC8D eine virtuelle Masse mit der halben Versorgungsspannung (1,65 V) erzeugt, die als Referenzpunkt für die beiden anderen Opamps dient. Der Stromverstärker IC8B verstärkt die Spannung über den Shunt-Widerstand um den Faktor 2,2 und verschiebt die verstärkte Spannung um 1,65 V nach oben. Mit IC8C lassen sich kleinere Ströme etwas genauer messen, da er die Spannung über dem Shunt-Widerstand um den Faktor 220 verstärkt. Die Ausgangsspannungen der beiden Strommess-Opamps werden dem zweiten ADC des Controllers (PA0/PA1) zugeführt. Damit sind Strommessungen bis zu ±0,75 A mit einer Auflösung von etwa 0,37 mA möglich. Die kleinste erfassbare Spitzenleistung bei 230 VAC beträgt somit ungefähr 0,1 W.

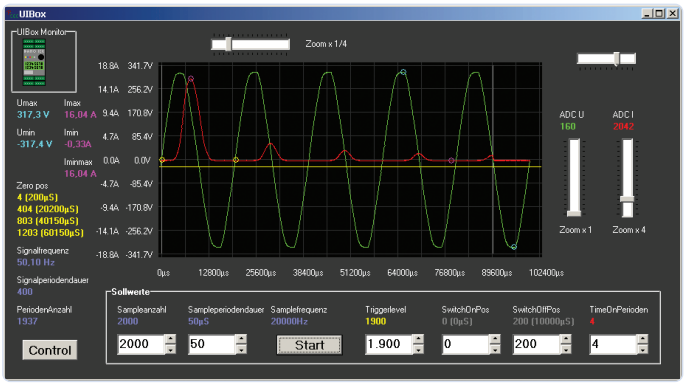


Bild 3. Die PC-Software zeigt die Netzspannung (grün) und den Strom (rot) an. Hier wurde die Spannung während eines Nulldurchgangs an den Prüfling angelegt (linke senkrechte graue Linie) und neun Nulldurchgänge später wieder entfernt (rechte senkrechte graue Linie).

PC-Schnittstelle

Das PC-Interface wurde mit zwei Optokopplern für die Leitungen TX (IC4) und RX (IC5) als RS-232-Schnittstelle (CON1) realisiert. Ein externer 5-V-USB-zu-RS-232-Konverter stellt eine Verbindung zu einem USB-Anschluss des PCs her. Die beiden zusätzlichen Optokoppler steuern den In-System-Programming-Pin (ISP- der MCU (IC5) beziehungsweise gewährleisten den sicheren Betrieb der Reset-Funktion (IC7). Die maximale Baudrate beträgt mit den verwendeten Optokopplern 19.200 Baud.

Auf der Platine

Die komplette Schaltung wurde auf einer doppelseitigen Platine aufgebaut. Der Hochstrompfad wurde auf der Unterseite mit 1,5 mm² dicken Kupferdrähten verstärkt. Der Widerstand der Schaltung beträgt circa 11 mΩ für die Strommessstrecke und circa 240 mΩ bei 10 A zwischen den Anschlüssen des Schalters. Der Differenzwiderstand zwischen den Anschlüssen des Schalters liegt bei 80 mΩ bei 10 A.

Die Software

Die Software für die Trafo-Experimente besteht aus der Firmware für den Controller zur Aufzeichnung der Rohdaten und der PC-Software zur Datenverarbeitung.
Die Firmware für die MCU wurde mit STM32Cube von STMicroelectronics und der Programmiersprache C99 erstellt. Als C-Compiler kam der ARM-GNU-Compiler zum Einsatz, der als Eclipse-Derivat im STM32Cube verfügbar ist. Die PC-Software (**Bild 3**) wurde in

Delphi erstellt und ist somit in Pascal geschrieben.
Die Daten des Controllers werden am RS-232-Verbinder CON1 ausgegeben und über den erwähnten USB-zu-RS-232-Pegelwandler zum USB-Anschluss des PCs geführt. Für jede einzelne Messung wird zur Kontrolle zusätzlich der Verlauf von Spannung und Strom angezeigt. Der Spannungstrigger-Pegel, der die Messungen startet (gelbe Linie), ist etwas unter Null eingestellt, um die Aufzeichnung des ersten Nulldurchgangs der Spannung sicherzustellen.

Nun wird gemessen!

Bei allen Messungen wurde der Transformator zu verschiedenen Phasenwinkeln der Netzwechselspannung ein- und ausgeschaltet. Es wurden die folgenden Parameter verwendet:

- Phasenwinkelauflösung: 5°
- Transformator für mindestens 25 Perioden (0,5 s) eingeschaltet
- Pause zwischen zwei Messungen: 0,5 s
- Abtastrate: 20 kHz

Der Strom wird über zwei komplette Spannungsperioden aufgezeichnet. Nach jedem Messdurchlauf werden die maximalen positiven und negativen Werte extrahiert und der größere Wert von beiden als Ergebnis festgehalten. Die ersten drei Messungen eines jeden Durchlaufs werden verworfen, um eine vorherige Restmagnetisierung des Kerns auszuschließen.
Jeder Durchlauf für den gleichen Endphasenwinkel wird von der PC-Software in einer Tabelle mit 72 Zeilen im HTML-Format gespeichert, die man dann einfach in eine Tabellenkalkulation importieren kann. Bei einer Winkelauflösung von 5° ergeben sich so 72 x 72 = 5184 Punkte, die als 3D-Plot dargestellt werden.

Beispiel: Einzelmessung

Das Beispiel in Bild 3 zeigt eine Einzelmessung ohne Last beim stärksten Stromimpuls. Für dieses Worst-Case-Szenario wurde die Spannung am Anfang der Kurve in einem Nulldurchgang eingeschaltet (senkrechte graue Linie links) und viereinhalb Perioden später wieder ausgeschaltet, wiederum in einem Nulldurchgang (senkrechte graue Linie rechts).
Beachten Sie, dass der stärkste Stromimpuls beim ersten Nulldurchgang auftritt. Dieser Impuls reicht jedoch nicht aus, um die bereits vorhandene Magnetisierung des Trafokerns zu kompensieren. Erst nach mehreren weiteren Nulldurchgängen verschwinden die Stromimpulse allmählich. Dieser Effekt ist deutlich hörbar, denn beim Einschalten ist für etwa eine halbe Sekunde ein kurzes Brummen zu hören.

Ringkerntransformator mit ohmscher Last

Für diesen Test wurde ein 100-VA-Ringkerntransformator von ELNA mit primärer 230-V-Wicklung und 2×24 V sekundär verwendet (**Bild 4**). An jede Sekundärwicklung wurde ein 50-W-Widerstand von 10 Ω für eine besseren Wärmeableitung auf einer Aluminiumplatte befestigt und als ohmsche Last angeschlossen. Die Gesamtlast beträgt etwa 115 W und der Transformator ist damit leicht überlastet.
Die Ergebnisse sind als 3D-Diagramme in **Bild 5** (ohne Last) und **Bild 6** (mit Last) dargestellt.

Bild 4. Der für die Experimente verwendete 100-W-Ringkerntransformator.

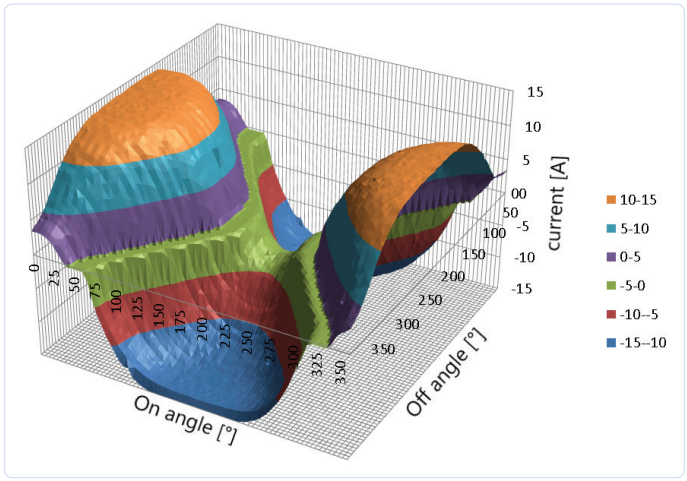


Bild 5. Ein 3D-Phasen-Strom-Diagramm für den Ringkerntransformator ohne Last.

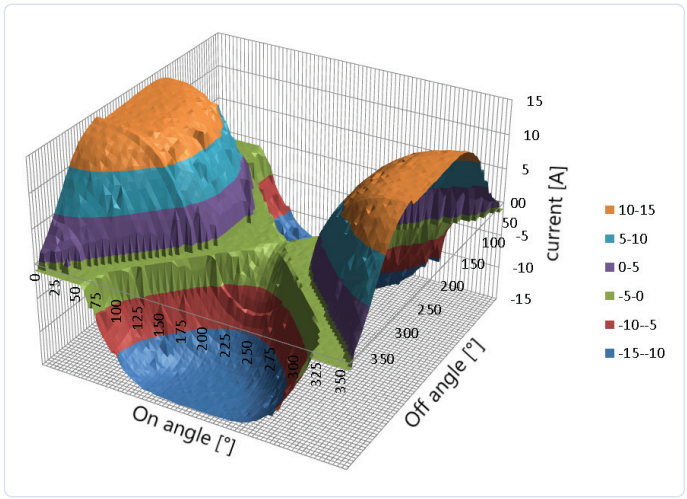


Bild 6. Ringkerntrafo mit ohmscher Last. Beachten Sie die Diskontinuität beim Abschaltwinkel von etwa 300°. Aus einem unbekanntem Grund ist die Sicherung durchgebrannt. Der Transformator kühlte etwas ab, bevor der Versuch fortgesetzt wurde, was zu einem etwas höheren Strom nach dem Neustart führte. Der Grund für den erhöhten Strom könnte der temperaturabhängige Widerstand des Kupferdrahtes der Primärwicklung sein.

Bild 7. Verhält sich dieser 120-W-EI-Kern-Transformator anders als der Ringkerntransformator?

EI-Kern-Transformator mit ohmscher Last

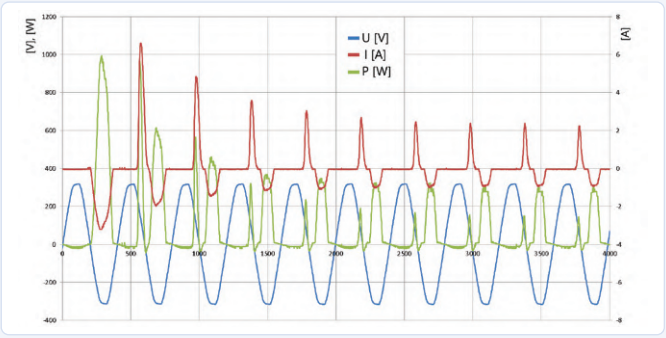
Der gleiche Test wurde mit einem 120-VA-Transformator mit EI-Kern des Herstellers Block wiederholt (**Bild 7**). Er besitzt eine 230-V-Primärseite und zwei in mehrere Abschnitte unterteilte 15-V-Sekundärwicklungen. Als ohmsche Last wurden zu jeder 15-V-Sekundärwicklung zwei 50-W-Widerstände á 10 Ω parallel geschaltet (ergibt je 5 Ω), wie zuvor wieder auf einem Aluminium-Kühlkörper montiert. Die Gesamtlast beträgt etwa 90 W, was den Trafo nicht ganz bis an seine Grenzen belastet. In **Bild 8** (ohne Last) und **Bild 9** (mit Last) sind die Ergebnisse zu sehen.

Zwei Halbwellen-Gleichrichter gleichzeitig

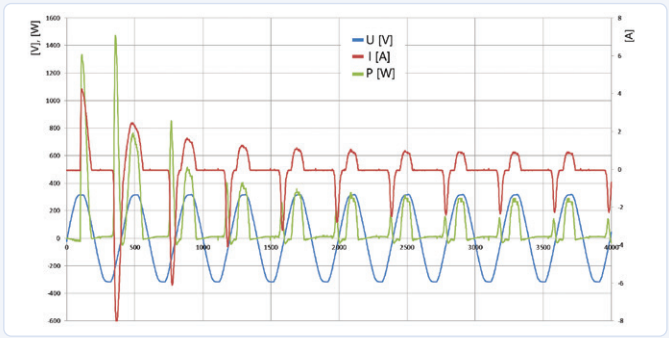
Alle anderen hier beschriebenen Versuche wurden nur mit dem Ringkerntransformator durchgeführt. An jede Sekundärwicklung

ASYMMETRISCHE LAST

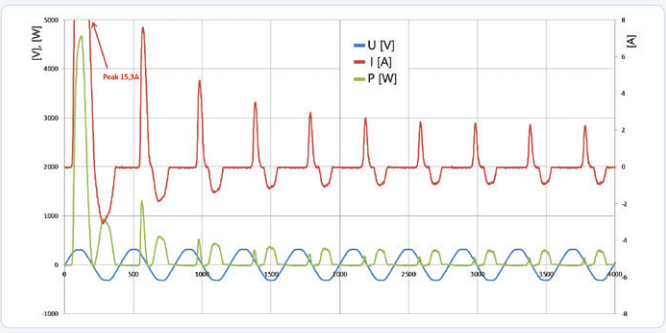
In den folgenden Experimenten wird die Schaltung aus Bild 10 verwendet, wobei jedoch jeweils nur eine Sekundärseite belastet wird. Jeder Versuch wurde zweimal durchgeführt, einmal mit der Last an S1 (A) und dann mit der umgekehrten Last an S2 (B). Die Ausgangsbedingungen waren die gleichen wie bei den entsprechenden Experimenten mit symmetrischer Belastung.



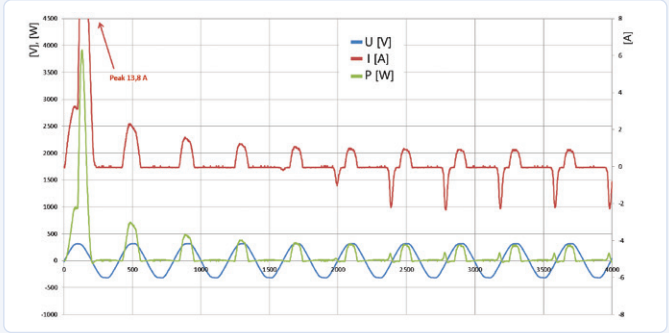
Bester Fall A. *Nur die Energie der negativen Stromimpulse fließt in die Last. Die positiven Stromimpulse fallen auch nach einiger Zeit nicht auf Null und pendeln sich bei einem Wert von etwa 2 A ein. Die Remanenz des Trafokerns wird durch die unsymmetrische Belastung der Sekundärseite etwas gestört. Die von den positiven Stromimpulsen transportierte Energie ist nicht negativ, wird also als Wirkleistung im Transformator und nicht in der Last verheizt.*



Bester Fall B. *Der größte Stromimpuls tritt während der zweiten Halbwelle auf. Die Energie der ersten Spitze fließt in die Last, die Energie der zweiten Spitze in den Transformator. Auch hier fallen die Stromimpulse nicht nach einiger Zeit auf Null ab, sondern pendeln sich bei ungefähr 2 A ein.*



Schlechtester Fall A. *Auch hier ist die Einschaltspitze ähnlich dem Leerlauf zu sehen. Nach einiger Zeit stellt sich der gleiche Zustand ein wie im Besten Fall.*



Schlechtester Fall B. *Dieses Szenario erzeugt nicht den größten Stromimpuls. Dies liegt wahrscheinlich daran, dass die unsymmetrische Belastung auf der Sekundärseite die im Kern gespeicherte Remanenz für einige Perioden etwas kompensieren kann. Nach etwa fünf Perioden verhält sich die Anordnung jedoch genau wie vorher.*

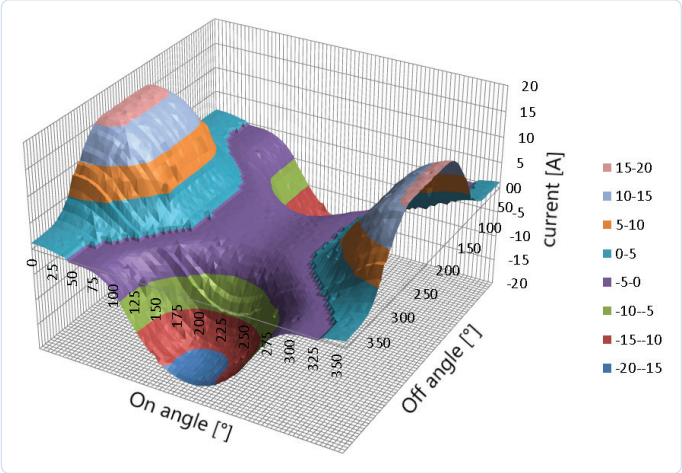


Bild 8. Das 3D-Diagramm für den unbelasteten EI-Kern-Transformator.

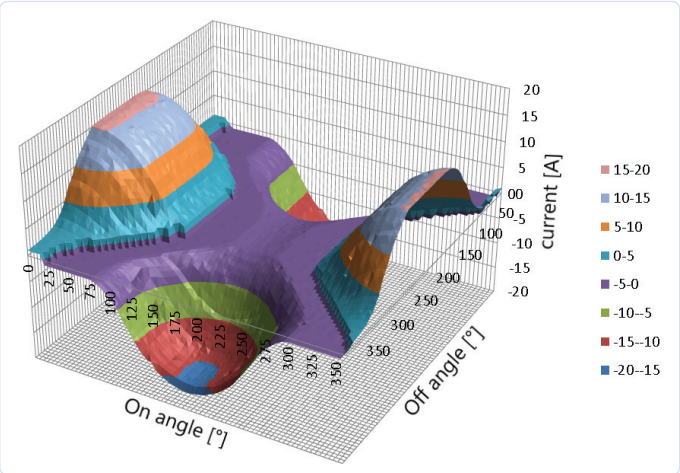


Bild 9. Der EI-Kern-Transformator mit einer ohmschen Last von 90 W.

des Transformators wurde ein Einweggleichrichter angeschlossen. Im Vergleich zur ersten Wicklung sind die Anschlüsse der zweiten Wicklung vertauscht, was zu einer gleichmäßigen magnetischen Belastung des Transformators führt, wie es bei einem Vollwellengleichrichter der Fall wäre. Jede Diode ist mit einem 100-W-Widerstand von 20 Ω und einem parallelen 10.000-µF-Elektrolytkondensator verbunden (**Bild 10**).

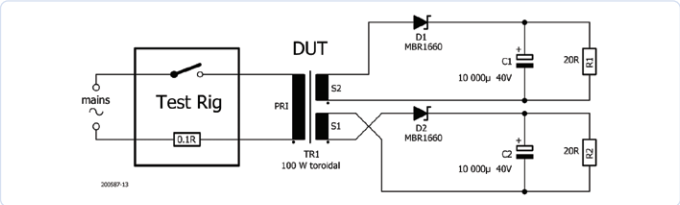


Bild 10. Die für die symmetrischen Lastversuche verwendete Schaltung. Bei den asymmetrischen Lastversuchen wurde jeweils nur eine Sekundärseite belastet.

Bester Fall: Schaltwinkel 90°/90°

Der Transformator wurde beim ersten Spannungsmaximum eingeschaltet, nachdem er zuvor bei genau einem solchen Maximum ausgeschaltet wurde. Dies entspricht dem besten Fall, bei dem keine zusätzlichen magnetischen Effekte wie in den vorherigen Experimenten auftreten sollten.

Die Ergebnisse **(Bild 11**) sind wie erwartet. Der größte Strom fließt in den Spannungsmaxima, um den Kondensator wieder aufzuladen. Der erste Impuls ist der größte, weil der Kondensator zu diesem Zeitpunkt noch leer ist. Die Stromimpulse pendeln sich langsam auf einen Wert von etwa 0,9 A ein (etwa das Doppelte des Nenn-Volllaststroms). Die Leistungskurve (grün) ist nie negativ. Es wird fast keine Blindleistung umgesetzt.

Schlechtester Fall: Schaltwinkel 0°/180°

Der Transformator wurde beim ersten Nulldurchgang der Spannung eingeschaltet, nachdem er zuvor beim gegenüberliegenden Nulldurchgang ausgeschaltet wurde. Dies entspricht dem

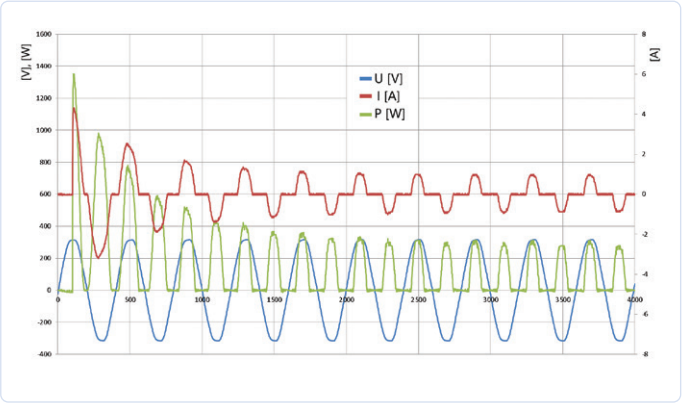


Bild 11. Im besten Fall mit symmetrischer Last ist die Leistungskurve (grün) nie negativ. Es wird fast keine Blindleistung umgesetzt.

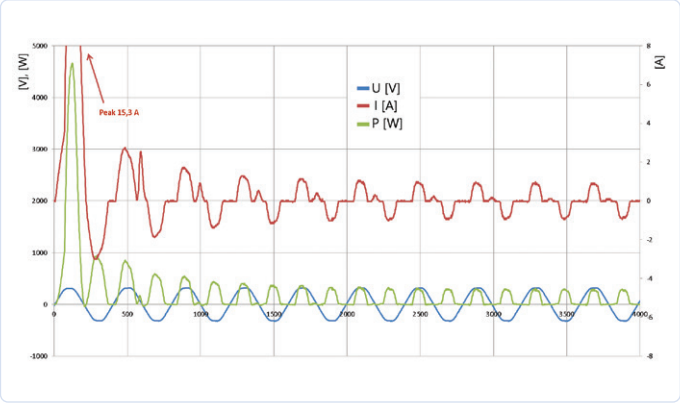


Bild 12. Im schlechtesten Fall mit symmetrischer Last erhalten wir für die umgesetzte Leistung in der ersten Spitze bemerkenswerte 4,65 kW (!), geleistet von einem kleinen 100-W-Transformator. Der Wert ist positiv, das heißt, es handelt sich um reine Wirkleistung.

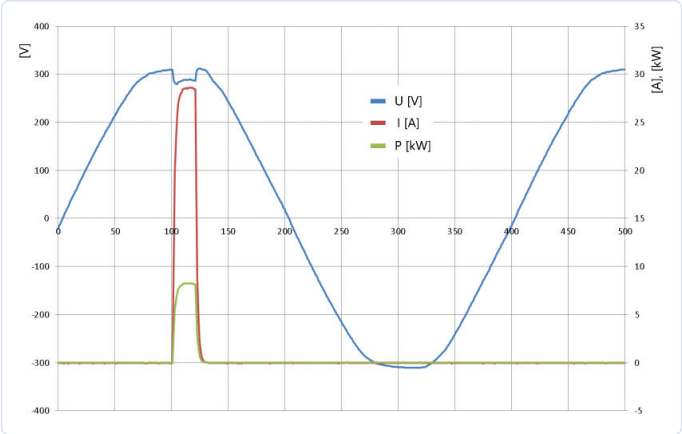


Bild 13. Für die Netzimpedanz wurde ein Wert von etwa 640 mΩ ermittelt.

ungünstigsten Fall, bei dem wie in den vorherigen Abschnitten die stärksten magnetischen Remanenzeffekte zu erwarten sind. Die Stromspitze von 15,3 A beim Einschalten (**Bild 12**) liegt nahe am Leerlauffall. Auch die folgenden Spitzen sind ähnlich und verschwinden nach 10...20 Perioden. Es wird fast keine Blindleistung umgesetzt.

Netzimpedanz

Zur Messung der Netzimpedanz wurde für ungefähr 1 ms während der Spannungsspitze der Netzspannung ein 50-W-Widerstand von 10 Ω zugeschaltet. Der Einbruch der Netzspannung kann zur Messung der Netzimpedanz verwendet werden, wie in **Bild 13** zu sehen.
Die Netzspannungsspitze wurde kurz vor dem Einschalten (310 V) und etwa 0,5 ms nach der Einschaltflanke (289 V) gemessen. Der hier fließende Strom beträgt 28,5 A (Leistungsspitze von 8,25 kW). Die Impedanz beträgt dann (310 - 289) / 28,5 = 740 mΩ. Die Schaltung besitzt bei diesem Strom einen Übergangswiderstand von etwa 100 mΩ, so dass die Netzimpedanz etwa 640 mΩ beträgt.

Schlussfolgerungen

Nach der Überprüfung der Ergebnisse können die folgenden Schlussfolgerungen gezogen werden:

- Der Einschaltstrom hängt stark vom Phasenwinkel ab, bei dem die Spannung zuvor abgeschaltet wurde.
- Das Einschalten bei einem Spannungsmaximum (oder -minimum) ist immer richtig, unabhängig vom Trafotyp. Es sollte aber das gleiche Maximum (beziehungsweise Minimum) sein wie beim vorherigen Ausschalten. Das Gegenteil ist gerade noch tolerierbar (etwa fünffacher Volllast-Normalstrom), allerdings verhalten sich Ringkerntransformatoren dabei etwas heftiger.
- Ein Einschalten während eines Nulldurchgangs ist nicht zu empfehlen. Das Ausschalten in der entgegengesetzten Nulldurchgangsrichtung erzeugt den höchsten Stromimpuls (etwa den 30-fachen Volllast-Normalstrom). Das Ausschalten im gleichen Nulldurchgang ist weniger kritisch (etwa zehnfacher Volllast-Normalstrom).
- Unbelastete Transformatoren reagieren mit wesentlich stärkeren Stromimpulsen als im belasteten Zustand. Der Toleranzspielraum hinsichtlich des richtigen Phasenwinkels unter Last ist deutlich größer.
- Transformatoren verhalten sich bei rein ohmscher Last und Gleichrichter/Kondensatorlast gleich.

Für diejenigen, die einen tieferen Einblick in die hier vorgestellten Versuche gewinnen wollen, stehen alle Versuchsdaten zusammen mit den Konstruktionsdateien des Prüfstands und der Software zum Download bereit [1].

200587-02

Haben Sie Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Senden Sie eine E-Mail an den Autor unter elektor@aftec.de oder an den Redakteur unter clemens.valens@elektor.com.

Ein Beitrag von

Idee, Design, Text und Illustrationen: **Andreas R. Fecht**
Redaktion und Illustrationen: **Clemens Valens**
Schaltpläne: **Patrick Wielders**
Übersetzung: **Rolf Gerstendorf**
Layout: **Giel Dols**

WEBLINK

[1] Testdaten, Konstruktionsdateien und Software: www.elektormagazine.de/200587-02

PiCAN 3 - yes we CAN!

CAN-Zusatzboard für Raspberry Pi 4

Von **Tam Hanna**

Seitdem Bosch 1986 den CAN-Bus spezifizierte und Mercedes-Benz ihn in der S-Klasse von 1991 erstmals einsetzte, ist viel passiert. Mittlerweile ist CAN, das Akronym für Controller Area Network, aus dem Automotive-Bereich nicht mehr wegzudenken, wird aber auch für viele andere MSR-Anwendungen eingesetzt. Zu Diagnose- und Testzwecken ist mobiles Equipment hilfreich - so wie das hier vorgestellte Zusatzboard für den Raspberry Pi.

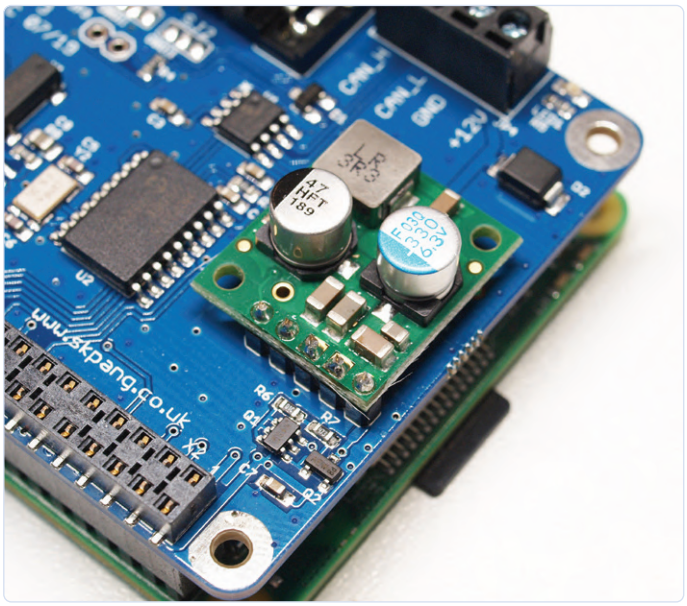


Bild 1. Fünf Pins im RM 1/10" sorgen für die Verbindung von Netzteil und Board.

Automobile sind einfach testequipmentfeindlich. Ich erinnere mich mit Schrecken daran, wie ich vor einigen Jahren einmal ein 40 m langes Kabel verlegen musste, um ein Oszilloskop in einem Auto in Betrieb zu nehmen. Sofern sich ihr Vorhaben aber auf die Auswertung von CAN-Bus-Daten beschränken lässt, steht Ihnen jetzt mit PiCAN 3 eine attraktive Messalternative zur Verfügung.

Wieso Raspberry Pi?

Wie bei der Arbeit mit I²C, SPI etc. gilt auch im Fall von CAN, dass einem Gutteil der Probleme nicht die mangelhafte Qualität der Signale bzw. der physikalischen Datenübertragung zugrunde liegt, sondern dass es an den übertragenen Daten selbst liegt. Sendet man falsch geformte Pakete oder nutzt z.B. die falsche Endianness, so muss man sich nicht wundern, wenn das Motor- oder sonstige Steuergerät verschnupft reagiert.

Tools wie PiCAN 3 erlauben im Zusammenspiel mit einem portablen Raspberry Pi mobiles Messen. Im Idealfall reicht ein Raspberry Pi 4 mit einem kleinen Mobil-Bildschirm. Integriert man die Entwicklungsumgebung des CAN-Systems in die ARMbian-Distribution des Systems, steht produktiven Experimenten nichts mehr im Wege.

Hardware-Aufbau

Die Umgebungen in Autos sind für digitale Elektronik durchaus schwierig. Der Hersteller SK Pang trägt dieser Situation insofern Rechnung, als dass er seine PiCAN-3-Platine huckepack mit einem Schaltnetzteil (**Bild 1**) bestückt, das über einen fünfpoligen Header Kontakt mit dem Board bekommt. Das Board soll angeblich auch ohne das Schaltnetzteil zu haben sein – praktisch erhältlich ist derzeit aber nur die Version mit.

Das hier vorliegende CAN-Shield hört auf den langen Namen *„PiCAN 3 – CAN-Bus Board for Raspberry Pi 4 with 3 A SMPS & RTC"* und unterscheidet sich von den Vorgängern insofern, als dass das inkludierte Schaltnetzteil immerhin 3 A liefert und so auch für die Versorgung des energiehungrigen Raspberry Pi 4 ausreichende Reserven mitbringt.

Dank intelligentem Schaltungsdesign kann das Shield auch vom Raspberry Pi versorgt werden, wenn dieser über seinen USBC-Port gespeist wird.

Die unter [1] abrufbare Schaltung des Boards ist nicht nur theoretisch interessant: SK Pang nutzt für die Realisierung des Interfaces eine Kombination absoluter Klassiker, die Microchip seit vielen Jahren fast unverändert verkauft, und die sich auch in einem eigenen Design ohne große Probleme einpassen lassen.

Für die Bereitstellung der CAN-Signalformen setzt SK Pang auf das IC MCP2515, das mit dem Raspberry Pi per SPI und über einen GPIO-Interrupt-Pin Kontakt aufnimmt. Die von ihm ausgegebenen Signal-Wellenformen wandern dann zum IC MCP2562 - eine Ein-Chip-Realisierung eines recht kompletten CAN-PHYs. Interessant ist außerdem, dass die von manchen Bauteilen benötigte 3,3-V-Spannungsversorgung aus dem Raspberry Pi entnommen wird, was den sonst nötigen Linearregler einspart.

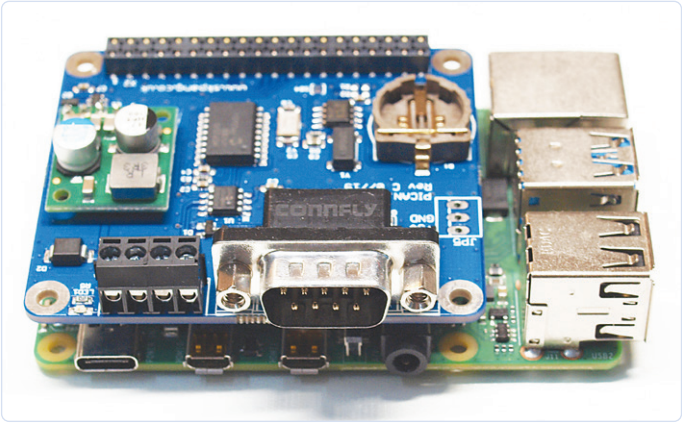


Bild 2. PiCAN 3 bietet zwei Anschlussmöglichkeiten an den CAN-Bus.

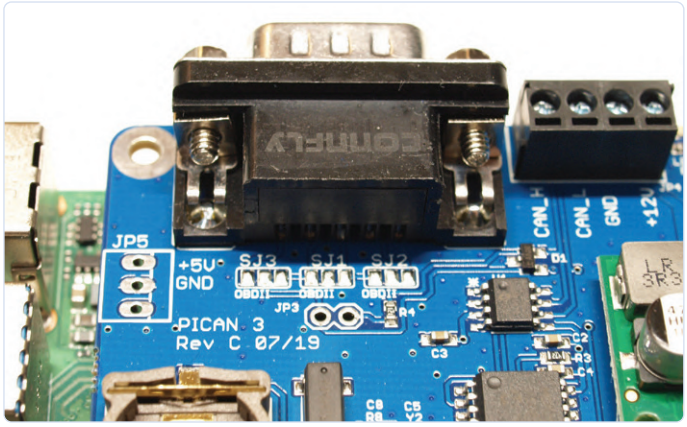
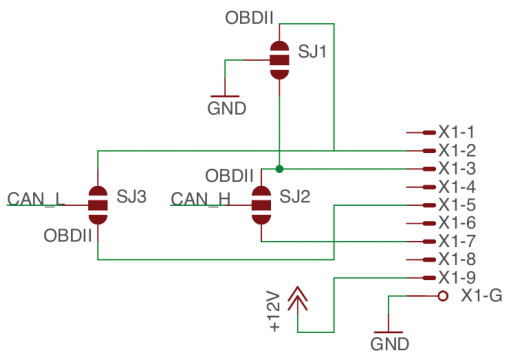


Bild 3. Dank großzügiger Dimensionierung lassen sich diese Pins auch ohne viel Können mit einem normalen Lötkolben überbrücken.



	CAN ANALYSER	OBDII	SOLDER-JOINT
CAN H	7	3	SJ2
CAN L	2	5	SJ3
GND	3	2	SJ1

Bild 4. Diese Schaltung mit Tabelle informiert über die Belegung der Löt-Pads.

Die nächste interessante Frage betrifft die Art des Verbindungsaufbaus zwischen Board und CAN-Bus. Hierzu dienen einerseits ein bei RS232 üblicher D-Sub-DE9-Stecker und andererseits eine vierpolige Schraubklemme (**Bild 2**).
Sinn des DE9-Steckers ist, dass vergleichsweise preiswert erhältliche ODB-II-Kabel direkt mit dem Board verbunden werden können. Leider gibt es keinen wirklichen Standard für die Zuordnung zwischen Pins und CAN-Signalen. Aus diesem Grund existieren auf dem Board einige Löt-Pads (**Bild 3**), über die man mit Hilfe eines Belegungsplans (**Bild 4**) die korrekte Zuordnung realisieren kann. Aber Achtung! Die Löt-Pads sind im Auslieferungszustand komplett offen. Ohne vorheriges Löten werden am DE9-Stecker keine CAN-Signale anliegen.
Automotive-Applikationen benötigen häufig genaue Zeitangaben. Die Echtzeituhr vom Typ PCF8523 des PiCAN 3 ist via I²C mit dem Raspberry Pi verbunden. Für die Stromversorgung der Echtzeituhr ist noch eine Knopfbatterie vom Typ CR1220 erforderlich, die in den dafür vorgesehenen Batteriehalter platziert wird.

CAN-Ökosystem

Wer ein komplettes System aus Decoder und mehr anbieten kann, hat einen Wert geschaffen. TeleDyne hat in der Vergangenheit einige Unternehmen mit solchen Produkten aufgekauft. Da Linux im Automotive-Bereich eine gewisse Relevanz aufweist, gibt es in der Linux-Welt ein ganzes Support-Ökosystem für den CAN-Bus. Die Bandbreite reicht dabei vom Kernel-Treiber bis hin zu Kommandozeilen-Utilities und anderen Helferlein.
Für die Kommunikation mit dem PiCAN 3 sind in einem frischen Raspbian-System einige Anpassungen der Datei */boot/config.txt* erforderlich. Der SPI-Bus muss durch folgenden Block freigegeben werden (man beachte den zusätzlichen Overlay-Aufruf):

```
dtparam=spi=on
dtoverlay=mcp2515-can0,oscillator=16000000,interrupt=25
dtoverlay=spi-bcm2835-overlay
```

Für die Echtzeituhr braucht es noch folgende Statements:

```
dtparam=i2c_arm=on
dtoverlay=i2c-rtc,pcf8523
```

Im nächsten Schritt ist das Herunterladen von Kernelmodulen und sonstigen Utilities fällig. Erfreulicherweise steht hierfür ein fertiges Paket in den Repositories bereit:

```
sudo apt-get install can-utils
```

Nach dem Deployment der CAN-Utilities muss das Interface beim Betriebssystem angemeldet werden. Hierzu reicht es aus, nach folgendem Schema ein neues Interface zu generieren. Der hier übergebene Wert von 500.000 steht für die maximale Bandbreite bzw. Geschwindigkeit, die von der Hardware unterstützt wird:

```
sudo /sbin/ip link set can0 up type can bitrate 500000
```

Ist das Interface bereit, können Sie es analog zu Produkten teurerer Hersteller verwenden. Ein klassischer Einsatzzweck wäre die Nutzung von *candump*. Dieses Tool lässt sich in der Kommandozeile durch Eingabe des folgenden Befehls aktivieren:

```
candump
```

Nach der Aktivierung zeigt es automatisch und permanent alle für das Shield sichtbaren CAN-Messages an. Dies ist insbesondere für

das Reverse Engineering unbekannter Motorsteuerungen hilfreich. Weiter gibt es noch ein Python-API, das sich nach folgendem Schema bereitstellen lässt:

```
git clone https://github.com/hardbyte/python-can
cd python-can
sudo python3 setup.py install
```

Echtzeituhr-Erweiterung

Linux unterstützt Echtzeituhren von Laptops und PCs schon ewig. Systeme, die (aus Kostengründen) ohne Echtzeituhr auskommen müssen, sind ein vergleichsweise neuer Trend, dem durch das Emulator-Modul *fake-hwclock* begegnet wird - so auch beim Raspberry Pi.

Um die Echtzeituhr auf dem CAN-Board nutzen zu können, muss man im ersten Schritt für die Deaktivierung des Emulator-Moduls sorgen:

```
sudo apt-get -y remove fake-hwclock
sudo update-rc.d -f fake-hwclock remove
sudo systemctl disable fake-hwclock
```

Im nächsten Schritt wird die Datei */lib/udev/hwclock-set* geöffnet. Hierzu sind normalerweise Superuser-Rechte erforderlich. Im nächsten Schritt werden die folgenden beiden Blöcke auskommentiert:

```
#if [ -e /run/systemd/system ] ; then
# exit 0
#fi

#/sbin/hwclock --rtc=$dev --systz --badyear
#/sbin/hwclock --rtc=$dev --systz
```

Für die Kommunikation mit der Echtzeituhr kommt dann das Tool *hwclock* zum Einsatz. Es verhält sich analog zu einem PC und weist die Möglichkeiten von **Bild 5** auf.

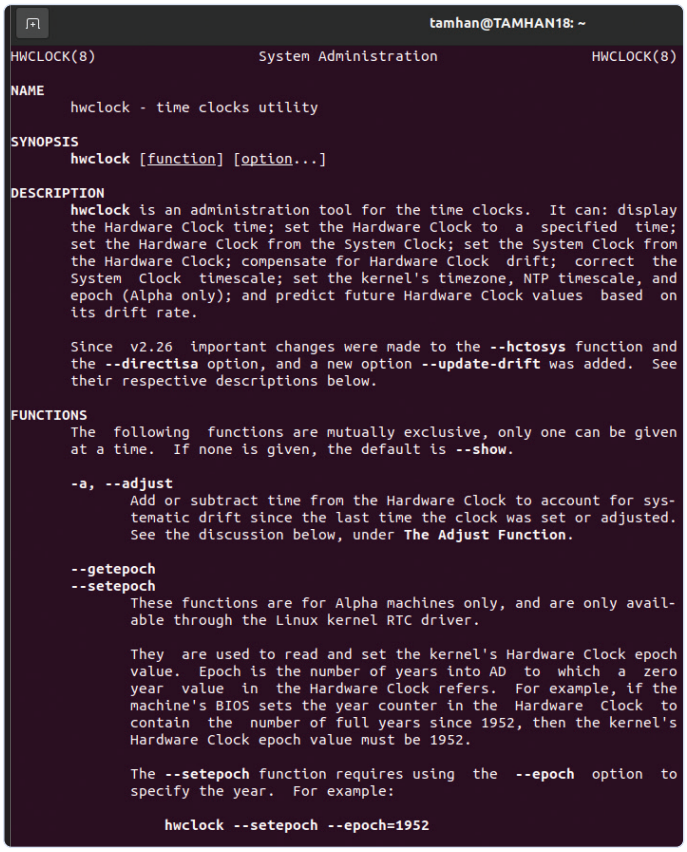


Bild 5. Wer mehr über eine Kommandozeilen-Bedienung wissen möchte, wird mit einer durch `man <kommando>` *aktivierbaren Manpage informiert. Hier geht es um* `hwclock`*, womit sich die Echtzeituhr auf dem CAN-Board steuern lässt.*

Fazit

Mit PiCAN 3 bietet SK Pang eine preiswerte und kompakte CAN-Lösung an, die durch die Kombination mit einem Raspberry Pi 4 einfach und mobil einsetzbar ist. Wenn Sie schon immer einmal mit dem CAN-Bus experimentieren wollten, finden Sie hier eine nicht nur preiswerte, sondern auch sehr leistungsfähige Experimentierplattform. Dank der Open-Source-Hardware lässt sich das System auch in eigene Designs einbinden, wenn die nötige Evaluation abgeschlossen ist.

210303-02

Ein Beitrag von
Text und Fotos: **Tam Hanna** Layout: **Giel Dols**
Redaktion: **Dr. Thomas Scherer**

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

WEBLINK

[1] **Schaltung des PiCAN-3-Boards:** https://cdn.shopify.com/s/files/1/0563/2029/5107/files/pican3_rev_C.pdf?v=1619981690

Balkon-kraftwerk

Selbst installiert = schnell amortisiert!

Von **Dr. Thomas Scherer**

Eine eigene Solaranlage muss nicht groß sein, um ökologisch und ökonomisch sinnvoll zu sein. Sogenannte Balkonkraftwerke gelten als neuer Trend, kosten nicht viel und werden von manchen Städten in Europa sogar finanziell gefördert. Falls sich bei Ihnen wie bei mir ein „Habenwollen-Effekt“ einstellt, dann lesen Sie, was es damit auf sich hat, welche Regeln gelten, was man kaufen sollte und welche Erfahrung ich mit diesem Projekt gemacht habe.

Schon länger war ich leicht neidisch auf meine Nachbarn, von denen viele das „schwarze Gold“ in Form einer Solaranlage auf dem Dach haben. Aber erstens habe ich noch nicht lange ein Haus, und zweitens schien mir die Gewinnung von Sonnenenergie doch recht teuer. Drittens widerstrebt es mir, „was mit Strom“ an irgendwelche Handwerker zu vergeben (bei denen zur Zeit eh die Nachfrage das Angebot übersteigt). Außerdem: In Deutschland ist so ein Vorhaben mit viel Bürokratie verbunden und die aktuellen Einspeisevergütungen sind – leider für mich, aber gut für die Allgemeinheit – sehr niedrig. Und wussten Sie, dass der deutsche Fiskus schon bei „normalen“ Anlagen auf dem Dach eines Einfamilienhauses Steuern auf selbst erzeugten und verbrauchten Strom erhebt?

Aus all diesen Gründen habe ich den Gedanken an Solarstrom in den letzten Jahren so schnell wieder verdrängt, wie er mir in den Sinn kam. Doch ab 2020, als ich meinen Rasenmäher-Roboter mit einer Solarstromversorgung ausgestattet habe [1], war ich angefixt. Bei der Recherche nach Solarmodulen stolperte ich nämlich mehrfach über den Begriff „Balkonkraftwerk“ (siehe Kasten **Was ist ein Balkonkraftwerk?**).

Und dieses Jahr war es dann soweit. Ich las mich ein, suchte mir die nötigen Infos zusammen, berechnete und überlegte. Dann bestellte und baute ich…

Verzinsung

Nach einer Ausleuchtung der rechtlichen Grauzonen wollte ich wissen, ob sich so eine Mini-Solaranlage auch finanziell rentiert. Da sich quasi zwei Klassen von Balkonkraftwerken etabliert haben, nämlich die Sparversion mit nur einem Panel und etwa 300 Wp (das kleine „p“

steht für *peak*, also Spitzenleistung) und der Vollausbau mit zwei Panels mit der maximalen Leistung von 600 Wp, war die Berechnung einfach.

Vom Deutschen Wetterdienst gibt es Strahlungskarten [2], aus der sich die mittlere Solarenergie pro Jahr in Europa und auch spezifischer für Deutschland entnehmen lässt (**Bild 1**). Im Internet findet man sogar umfangreiche Rechner für den solaren Ertrag. Für andere Länder finden sich im Netz ebenfalls Karten und Berechnungsgrundlagen. Da ich im sonnenverwöhnten Südbaden lebe, kann ich unter optimalen Bedingungen mit fast 1.200 kWh pro m² und Jahr an solarer Globalstrahlung rechnen. Setzt man einen Wirkungsgrad von 20 % für monokristalline Solarzellen an, könnte ich also idealerweise bis zu 240 kWh/m²/y produzieren (das kleine „y“ steht für *year* bzw. Jahr). Da für mich eher die Variante mit zwei Panels in Frage kommt, und ein modernes Panel im Leistungsbereich von 320...370 Wp eine nutzbare Fläche von etwa 1,6 m² hat, wäre mit zwei Panels immerhin eine Ernte von fast 770 kWh/y drin. Kalkuliert man den aktuellen deutschen Strompreis von rund 30 ¢/kWh, würde mein Balkonkraftwerk demnach bis zu 230 €/y einsparen. Vorab sei verraten, dass die Gesamtkosten der Anlage bei mir ziemlich genau bei 600 € lagen. Die Anlage hätte sich – so gerechnet – also schon in zwei Jahren und sieben Monaten amortisiert!

Schön wär's! Denn erstens müssten dazu die Solarmodule im optimalen Winkel am optimalen Ort aufgestellt sein, zweitens dürften keine Verluste (Wechselrichter und Kabel) auftreten und drittens müsste man den gesamten, selbst erzeugten Strom auch selbst verbrauchen. Klar ist, dass bei einer Einzelperson mit Zweizimmerwohnung, die tagsüber nicht zuhause ist, wohl schon bei 300 Wp ein großer Teil nicht genutzt würde. In meinem Haus aber stehen zwei Kühlschränke und eine Tiefkühltruhe; meine Mutter sieht als Rentnerin viel fern und ich arbeite im „Home Office“, habe also tagsüber einen PC samt großem Monitor laufen. Zusammen mit anderen Kleinigkeiten werden 300 W Grundlast tagsüber deutlich überschritten. Eine 600-Wp-Anlage ist also nicht überdimensioniert.

Rechne ich mit konservativen 75 % Auslastung, dann komme ich unter optimalen Bedingungen auf etwa 170 €/y gesparte Stromkosten. Meine Bedingungen sind aber nicht optimal: Zwar kann ich die Panels ohne Abschattung montieren, aber die Neigung beträgt nur 5° und die Ausrichtung ist mit 135° exakt Südsüdost. Ich profitiere also weniger von der Abendsonne, aber mehr von diffuser Strahlung.

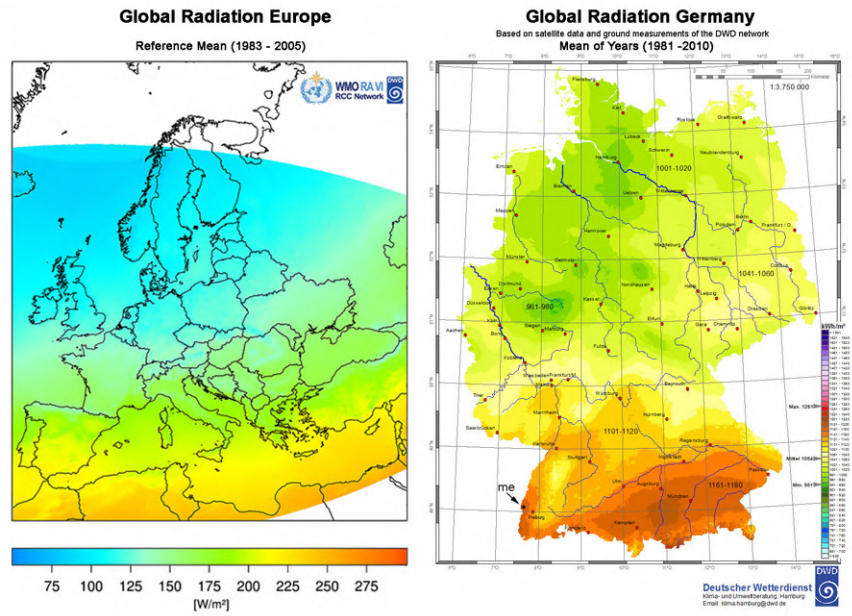


Bild 1. Karten der gemittelten Sonneneinstrahlung in Europa und Deutschland. (Bild: DWD [2]).

Was ist ein Balkonkraftwerk?

Die Antwort auf diese Frage ist zunächst ganz einfach: Ein Balkonkraftwerk ist eine kleine Solaranlage. Wie der Name schon sagt, ist sie so klein, dass man sie tatsächlich an Balkonen anbringen kann, wenn man will. Doch diese Definition ist noch nicht erschöpfend.

Zentral ist, dass es sich um eine Solaranlage kleiner Leistung handelt. In der EU fallen Anlagen mit einer Spitzenleistung von maximal 800 Wp unter eine Bagatellgrenze. Folglich darf man sich einfach so eine Anlage installieren und niemand kann einem dreinreden (solange man elektrisch sauber arbeitet) oder mit Bürokratie erdrücken (außer in Deutschland, wo man nur mit maximal 600 Wp auf der sicheren Seite steht).

Weiter wichtig ist der Zweck: Mit einem Balkonkraftwerk erzeugt man gerade so viel Strom, dass quasi der „Ruhestrom“ eines Hauses gedeckt wird. Also der Verbrauch eines Kühlschranks, eines Fernsehers, des Internet-Routers, der Umwälzpumpe der Heizung sowie zig Steckernetzteile und sonstige Standby-Verbräuche. Alles was die Anlage mehr produziert, schenkt man dem Netzbetreiber, um die Anlage simpel und preiswert zu halten. Folglich sollte sie nicht überdimensioniert sein.

Ziehe ich dafür noch einmal konservativ 25 % ab, lande ich bei etwas über 125 €/y. Bei Kosten von 600 € wird sich mein Balkonkraftwerk vermutlich schon in vier Jahren bezahlt gemacht haben und ab da klingelt nur noch die Kasse!

Wenn man einen Zuschuss bekommt (die benachbarte Stadt Freiburg schießt zum Beispiel glatt 200 € zu), ist das schon viel früher der Fall. Voraussetzung ist natürlich, dass die Kosten für die Installation wegfallen, weil man das selbst übernimmt. Außerdem sind hier keine Kosten für einen Zählerwechsel berücksichtigt, die manche Netzbetreiber erzwingen wollen. Beispielsweise über die Notwendigkeit eines sogenannten Zweirichtungszählers wird gestritten (eine Rücklaufsperre allerdings ist die Mindestanforderung). Weiter sollte die gekaufte Hardware zuverlässig sein, denn wenn Teile früh ausfallen, wird die Anlage weit weniger rentabel.

Wenn Sie sich fragen, warum ich die Panels nicht steiler gestellt habe: Das war nicht nur einfacher, sondern ich erhalte so im Sommer, wo manchmal tagsüber eine Klimaanlage bei mir läuft, eine bessere Ausbeute (auf Kosten des Winterbetriebs).

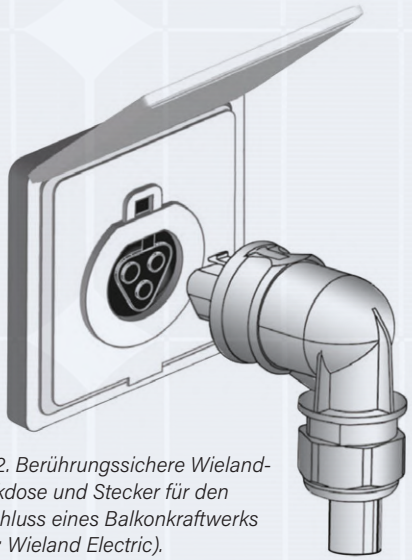


Bild 2. Berührungssichere Wieland-Steckdose und Stecker für den Anschluss eines Balkonkraftwerks (Bild: Wieland Electric).

Bild 4. Netz-Kupplung und Abdeckkappe für den Wechselrichter.

Bild 3. Der Wechselrichter HM-600 im Größenvergleich mit einem Elektor-Heft.

Bild 5. Der Wechselrichter HM-600 von der anderen Seite.

Bild 6. Ein Panel des Typs JKM330M-60 von Jinko Solar lehnt noch unausgepackt an der Hauswand.

Bild 7. Der Sicherungskasten. Eine zusätzliche 10-A-Sicherung (mittig links) und ein kleiner Stromzähler (unten links) wurden eingebaut. Der moderne Zähler rechts hat eine Rücklaufsperre.

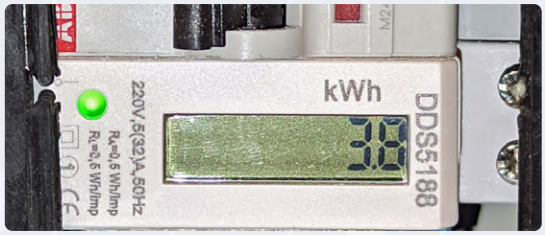


Bild 8. Der preiswerte Extra-Stromzähler erfasst die gesamte, vom Balkonkraftwerk erzeugte Energie. Auf der Anzeige sieht man die Ausbeute des ersten Betriebstages.

Passende Produkte

- OWON OW16B DVM mit Bluetooth (SKU 18780)
  www.elektor.de/owon-ow16b-digital-multimeter-with-bluetooth
- PeakTech Stromzange 4350 (SKU 18161)
  www.elektor.de/peaktech-4350-clamp-meter
- Umwelt-Monitor-Kit (SKU 19148)
  www.elektor.de/enviro-environmental-monitoring-station-for-rpi

Komponenten

Ein Balkonkraftwerk besteht aus Solarzellen, einem Wechselrichter und der Anschlusseinrichtung. Bei letzterem handelt es sich vielfach um eine spezielle Wieland-Steckverbindung (**Bild 2**), denn im Prinzip ist es erlaubt, unter Beachtung der Sicherheit und bestimmter Einschränkungen die Anlage einfach einzustöpseln. Eine zentrale Rolle kommt dem Wechselrichter zu. Er wandelt die niedrige Gleichspannung eines Solarpanels in eine normgerechte und einspeisefähige Wechselspannung von 230 V um. So ein Wechselrichter checkt, ob am Netzanschluss auch eine Wechselspannung geeigneter Höhe und Frequenz anliegt – erst dann liefert er selbst Strom. Zieht man den Stecker, schaltet der Wechselrichter seinen Ausgang zur Sicherheit in wenigen Millisekunden ab.

Zu beachten ist bei der Auswahl die Abgabeleistung, die maximale Einspeiseleistung, der Arbeitsspannungsbereich für die Solarmodule und die Qualität. Aus diesen Parametern ergibt sich auch die Wahl der passenden Solarpanels. Ich habe mich für den Wechselrichter HM-600 des Herstellers Hoymiles entschieden. Er ist für 220 € zu haben und soll laut Berichten anderer Anwender sehr zuverlässig sein. Über etwas preiswertere Wechselrichter habe ich teilweise Bedenkliches gelesen, und daher die Finger davon gelassen. Es gibt aber auch noch diverse andere, gut für Balkonkraftwerke geeignete Wechselrichter – Google hilft.

Mein Modell bietet eine Spitzenausgangsleistung von 600 W und ist für den Anschluss von zwei Panels im Leistungsbereich von 240 bis 380 Wp gedacht. Sein Wirkungsgrad soll immerhin bei 96,5 % liegen und nachts benötigt er weniger als 50 mW. **Bild 3** zeigt am Vergleich mit einem Elektor-Heft, dass er gar nicht so klein ausfällt. Das flache Ding wiegt immerhin 3 kg und wird einfach unter ein Panel geschraubt. Das lange Kabel unten ist der Netzanschluss (für den es die passende Kupplung braucht), und das kurze Kabel mit dickem Stecker oben ist der Anschluss für weitere Wechselrichter (für den es eine Abdeckkappe braucht). Beide Teile (**Bild 4**) sollte man mitbestellen. **Bild 5** zeigt den Wechselrichter von unten.

Aus den Daten des Wechselrichters und Effizienzüberlegungen ergibt sich die Auswahl der Panels. Für 600 W Output sollte etwas mehr am Input anliegen, sonst hat man keine Reserve. Da Standardpanels mit steigender Effizienz teurer werden, habe ich mich gegen 375-W-Modelle entschieden, die ja durchaus anschließbar wären. Stattdessen habe ich zwei monokristalline 330-Wp-Panels von Jinko Solar geordert (**Bild 6**). Das gibt immerhin 10 % Reserve. In der Klasse oberhalb 300 Wp haben fast alle Panels mit minimalen Abweichungen das gleiche Maß von 166,5 x 100,2 cm und ein Gewicht von knapp 20 kg. Jinko garantiert, dass die Panels auch nach 25 Jahren noch mindestens 80 % der Anfangsleistung haben. 330-Wp-Panels gibt es ab 150 €. Bei der Bestellung muss man ein Auge auf das Porto haben, denn bei so großen Teilen wird das schnell teuer. Besser man zahlt vor Ort ein bisschen mehr und kann es dann direkt abholen. Mit umgeklappter Rücklehne passen zwei Panels in etliche nicht zu kleine Autos oder Kombis.

Anschluss

In anderen Ländern mag es gestattet sein, ein Balkonkraftwerk einfach über eine normale Netzsteckdose anzuschließen. In Deutschland ist man da trotz uneindeutiger Rechtslage pingeliger und auch in anderen Ländern ist es sinnvoll, eine berührungssichere Alternative wie die Steckverbindung von Wieland zu wählen. Dies gilt auch, wenn der Wechselrichter beim Steckerziehen sofort abschaltet. Für Steckdose und Stecker von Wieland muss man 35 € einkalkulieren.

Die Steckdose kann an geeigneter Stelle parallel zu einer anderen Steckdose geschaltet oder aber direkt mit dem Sicherungskasten über eine eigene Sicherung verbunden werden. In beiden Fällen bleibt die Anlage dann halbwegs portabel. Dabei gibt es allerdings Folgendes zu beachten: Wird für die Wieland-Dose kein eigener Netzanschluss gelegt, dann muss der Netzzweig, an den sie angeschlossen wird, niedriger abgesichert werden!

Die Überlegung dahinter ist folgende: Haushaltssteckdosen sind für Ströme bis 16 A gemacht und deshalb in der Regel auch mit diesem Strom abgesichert. Bei einem 600-W-Balkonkraftwerk kommen jetzt aber in diesem Zweig bis zu 2,6 A hinzu, die nicht über die Sicherung laufen. Damit könnte mehr Strom als erlaubt über eine parallele Steckdose fließen. Da dies gefährlich werden kann, muss die Sicherung hier von 16 A auf den nächstkleineren Wert von 10 A reduziert werden.

Diese Punkte führten bei mir zum Entschluss, das Kraftwerk direkt mit dem Sicherungskasten zu verbinden und ihm eine eigene 10-A-Sicherung zu spendieren. Nachteil ist, dass hierfür 15 m Kabel gelegt und etliche Löcher gebohrt werden mussten. Vorteil ist, dass ich die 35 € für den Stecker spare und die Möglichkeit habe, sehr einfach die eingespeiste Leistung zu erfassen. Es gibt hierzu Fancy-Lösungen, zum Beispiel mit einer aktuellen Anzeige der geernteten Leistung auf dem Handy, doch mir schien das zu teuer und übertrieben. Im Netz gibt es nämlich für unter 10 € simple und passende Stromzähler im Format einer Sicherung. **Bild 7** zeigt den erweiterten Sicherungskasten. Unten links befindet sich der kleine Zähler. Die Großaufnahme von **Bild 8** beweist, dass sich die Sache lohnt. Am 9.5.2021 – dem ersten und glücklicherweise komplett wolkenfreien Tag der Inbetriebnahme – konnte ich schon 3,8 kWh ernten. Brutto ergibt das immerhin 1,14 €. Ein Reinertrag von 125 €/y wie Anfangs geschätzt ist also durchaus realistisch.

Recht und Bürokratie

Wie Bild 7 beweist, habe ich schon einen modernen Zähler mit sogenannter Rücklaufsperre. Wer noch einen alten,

Bild 9. Die beiden 330-Wp-Panels bei der Arbeit auf dem Dach der Veranda.

elektromechanischen Ferrariszähler hat, bei dem läuft bei nicht selbst verbrauchtem Solarstrom der Zähler rückwärts. Auf diese Weise könnte man also bequem das Netz als Pseudo-Akku nutzen. Auch wenn sogar manche Politiker schon meinten, dass man Energie im Netz speichern könne, ist das weder logisch möglich noch erlaubt! Man würde einfach illegal die saldierende Messung der bezogenen Leistung reduzieren und damit sogar Steuerhinterziehung begehen.

Um nicht in die Bredouille zu kommen, sollten man die in seinem Land geltenden Spielregeln einhalten. Für Deutschland bedeutet dies, dass man am besten am Tag der Inbetriebnahme sein Balkonkraftwerk sowohl beim Netzbetreiber als auch bei der Bundesnetzagentur anmeldet. Das ist kein Kann, sondern ein Muss – auch für sehr kleine Anlagen wie meine. Man kann das online erledigen [3] und braucht jeweils keine zehn Minuten. Ist die eigene Anlage im sogenannten „Marktstammdatenregister" registriert, wird dies auch an den jeweiligen Netzbetreiber gemeldet, was aber nicht von der Anmeldung dort entbindet. Der kann dann einen eventuell vorhandenen alten Zähler gegen einen neuen mit Rücklaufsperre oder einen Zweirichtungszähler austauschen, und man ist seinen Verpflichtungen nachgekommen.

Beim Zähler (vor allem bei einem umstrittenen Zweirichtungszähler) können je nach Netzbetreiber Austauschkosten von etwas über 100 € zustande kommen, was den Zeitraum der Amortisierung verlängert. Wer überlegt, sich die eingespeiste Energie vergüten zu lassen: Keine gute Idee! Aktuell liegt die deutsche Einspeisevergütung bei 7,47 ¢/kWh. Werden z.B. 25 % der von mir geschätzten Real-Ernte von 480 kWh/y eingespeist, würde mir das 120 kWh x 7,47 ¢/y = 8,96 €/y einbringen. Das würde nicht einmal erhöhte Gebühren abdecken und schon gar nicht den bürokratischen Mehraufwand rechtfertigen. Besser man schenkt diese Energie dem EVU und hat ein gutes Gefühl, denn auch bei anderen Verbrauchern reduziert dieses Geschenk den Verbrauch fossiler Energie und wirkt sich CO_2-reduzierend aus.

Aufbau und Installation

Wie man an **Bild 9** sehen kann, habe ich meine beiden Module mit Hilfe einiger kräftiger Edelstahlwinkel einfach auf das direkt ans Haus anschließende Dach über einer Veranda montiert. Auf Ziegeldächern wird es mit den dafür nötigen Haken etwas komplizierter. Für Flachdächer gibt es fertige Stahl-Halter im für Mitteleuropa richtigen Aufstellwinkel. Vom unter einem Panel befestigten Wechselrichter aus geht es bei mir über eine wetterfeste Schlauchleitung in Kabelkanälen ab in den Keller und von dort nach Wechsel auf NYM-Verkabelung in den Sicherungskasten. Dass man dabei sorgfältig arbeiten sollte, versteht sich von selbst. Für Arbeiten an der Hausinstallation und am Sicherungskasten sollte man den nötigen Sachverstand aufweisen **und** das auch dürfen. Hier sollte man im Zweifel einen vom Netzbetreiber zugelassenen Installateur beauftragen.

Als einfachere Alternative bietet es sich an, die gesteckte Variante mit einer Wieland-Steckverbindung zu realisieren, denn dahinter zählt nur saubere und fachgerechte Arbeit, zu der man als Elektroniker ja durchaus in der Lage sein sollte. Den Sicherungswechsel im Sicherungskasten muss man dann aber immer noch durch einen zugelassenen Elektriker erledigen lassen.

Insgesamt brauchte das Schlitzeklopfen, Kabelkanäleanbringen, Kabelverlegen, genaues Planen der Löcher in den Edelstahlwinkeln, Modifikation des Sicherungskastens und der Montage der Panels nicht ganz drei Arbeitstage. Ein geübter „Solarteur" wäre bestimmt doppelt so schnell gewesen. Aber so gebührt (fast) die ganze Ehre nach Fertigstellung mir allein!

Bild 10 zeigt die Anlage von der Seite direkt nach Fertigstellung. Die Leiter stand noch.

Außerdem

Natürlich kann man es sich auch einfacher machen und ein fertig komponiertes Set an Komponenten für ein Balkonkraftwerk kaufen. Anbieter hierfür gibt es massenhaft im Internet. Man muss dabei aber darauf achten, ob das Set für die eigene Anwendung vollständig ist und ob der Gesamtpreis wirklich stimmt. Ich habe es vorgezogen, die nötigen Teile nach meinen Ansprüchen zu kombinieren.

Würde ich das nochmals machen, jetzt wo das Balkonkraftwerk fertig ist? Auf jeden Fall! Zwar erwische ich mich die letzten Tage bei überflüssigen Kontrollgängen zum Sicherungskasten, aber das häufige Ablesen des Einspeisezählers stützt meine ursprünglichen Kalkulationen. Es zeigte sich zudem, dass auch an bewölkten Tagen mit Regen noch einige hundert Wattstunden geerntet werden können. Ich bin schon ganz gespannt auf diesen Sommer und den folgenden Winter. Nächstes Jahr um diese Zeit weiß ich dann, wie es tatsächlich lief mit der Stromernte 2021.

Und: Nein, man darf es nicht, was Sie jetzt vielleicht denken! Es ist Ihnen doch bestimmt schon die ganze Zeit durch den Kopf gewandert, ob man nicht einfach zwei oder drei solcher Balkonkraftwerke installieren könnte, nicht wahr? Schließlich sind die Wechselrichter schon mit Steckverbindungen zum Verketten ausgestattet. Kurz gesagt: Können kann man schon, aber dürfen darf man nicht, denn alle Balkonkraftwerke zählen als eine Anlage.

Breaking News: Bundesregierung beschließt Steuerbefreiung für Solaranlagen ≤10 kWp

2 x 300 Wp würden gehen, aber gegenüber 1 x 600 Wp teurer ausfallen. Doch 2 x oder 3 x 600 Wp liegen eindeutig über der Bagatellgrenze. Da darf dann das EVU mitreden und die Bürokratie wird heftiger. Und überhaupt: Wozu denn mehr als 600 Wp? Ohne Zähler mit Einspeiseerfassung würden Sie dann nur mehr elektrische Energie verschenken.

210326-02

Ein Beitrag von

Idee, Durchführung und Text:
Dr. Thomas Scherer
Redaktion: **Jens Nickel**
Layout: **Harmen Heida**

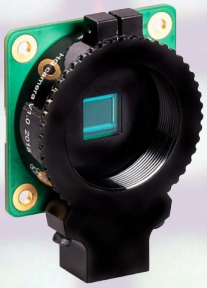
Sie haben Fragen oder Kommentare?

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

Bild 10. Geschafft: Die beiden Solarpanels samt Wechselrichter sind fest auf dem Vordach montiert.

WEBLINKS

[1] **Solaranlage für Mähroboter, ElektorMag 7-8/2021**: www.elektormagazine.de/200553-02
[2] **Strahlungskarten des DWD**: www.dwd.de/DE/leistungen/solarenergie/lstrahlungskarten_mi.html?nn=16102
[3] **Anmeldung im Marktstammdatenregister der Bundesnetzagentur**: www.marktstammdatenregister.de
[4] **Gute Sammlung von FAQs zu Balkonkraftwerken von Alpha Solar**: www.alpha-solar.info/info/faq.html
[5] **Weitere Elektor-Artikel zum Thema „Solar“**: https://bit.ly/2Ri4YwZ



Die High-Quality-Kamera des Raspberry Pi in der Praxis

FOTOGRAFIEREN UND VIDEO-STREAMING mit dem Raspberry Pi 4

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

Der Raspberry Pi 4 als netzwerkbasierte Kamera wurde schon öfters behandelt. Diesmal wollen wir zeigen, wie man eine Kamera für Streaming und Fotos einrichtet. Natürlich ist etwas billiger, das von Haus aus für diesen Zweck eingerichtet ist, aber wo bleibt da der Spaß und die Bildung?

Sie können direkt aufhören mit dem Lesen, wenn Sie nur auf der Suche nach einer preiswerten Webcam auf Raspberry-Pi-Basis sind, die „out of the box" funktioniert. Doch wenn Sie daran interessiert sind, Dingen auf den Grund zu gehen und sich etwas Wissen anzueignen, ist dieser Artikel das Richtige für Sie! Das Ergebnis wird eine Kamera sein, die auf einem Raspberry Pi 4B basiert, der mit einer *Raspberry Pi High Quality Camera*, einem Objektiv, einem TFT-Bildschirm und einem Stativ (aus einem anderen Projekt entlehnt) ausgestattet ist. Als Gehäuse habe ich eines von Adafruit gewählt, das am ehesten einem guten alten Kameragehäuse ähnelt. Mit dem Raspberry Pi 4B verwenden wir das neueste verfügbare Pi-Modell. **Bild 1** zeigt das Setup bei der Arbeit.

Physikalisches Licht wird zum digitalen Bild

Bevor ich den Aufbau der Kamera beschreibe, lassen Sie uns ein wenig Theorie der Bildverarbeitung und Datenausgabe betrachten. Um zu verstehen, was die Kamera sieht und was man am Ende erhält, sind einige Schritte an Signalverarbeitung und -aufbereitung erforderlich. Die erste Stufe ist die Optik. Abhängig vom verwendeten Objektiv ergibt sich daraus die Fähigkeit, ein Bild in einer bestimmten Entfernung zu fokussieren. Dies hat bereits Auswirkungen auf die Bilder, die Sie später aufnehmen werden, da die Optiken in diesen Objektiven das einfallende Licht zu Ihrem Bildsensor leiten. Wenn diese schlecht konstruiert oder von schlechter Qualität sind, kann dies zu Bildverzerrungen führen. Hier verwenden wir ein 16-mm-Objektiv, das mit 10 MP gekennzeichnet ist. Wenn Sie sich fragen, warum ein Objektiv eine Auflösung in Megapixeln (MP) haben kann, schauen Sie mal unter [1] nach. Das bedeutet auch, dass ein 10-MP-Objektiv niemals einen Bildsensor dazu bringen wird, mehr Pixel auszugeben, als für ihn spezifiziert sind. Außerdem werden wir später ein 6-mm-3-MP-Objektiv zum Vergleich verwenden. Welches Objektiv Sie benötigen, hängt dann von Ihrer Anwendung ab.

Neben der Optik spielt auch der Bildsensor eine große Rolle. Je höher die Auflösung, die er aufnehmen kann, desto besser ist er – oder nicht? Ist ein 18-MP-Sensor immer besser als ein 12-MP-Sensor, da er doch 6 MP mehr Auflösung bietet? Nicht unbedingt, denn es kommt mindestens noch ein zweites Charakteristikum ins Spiel, nämlich die Größe. Je kleiner ein Bildsensor bei einer bestimmten Auflösung ist, desto weniger Licht steht pro Pixel zur Verfügung. Je kleiner diese Sensoren werden, desto lichtschwächer sind sie also, und desto wahrscheinlicher ist es, dass sich benachbarte Pixel nicht nur durch Licht, sondern auch durch thermische Effekte innerhalb des Chips gegenseitig beeinflussen. Das bedeutet, dass die Fläche, die ein Sensor nutzt, die Bildqualität ebenso beeinflussen kann wie die angegebene Anzahl der Pixel.

Schwarz, Weiß und viele Grautöne

Wenn Sie sich über Schwarz, Weiß und Grau wundern, obwohl am Ende ein Farbbild entsteht, müssen Sie berücksichtigen, was der Sensor sehen kann und was er an unser System liefert. Vielleicht haben Sie schon einmal mit Fotodioden gespielt, die auf einfallendes Licht mit einer Widerstandsänderung reagieren. Auf einem Kamerachip passiert im Prinzip dasselbe, aber es gibt Millionen von Fotodioden dicht nebeneinander gepackt. Jede dieser Dioden ist nur in der Lage, die Menge an Licht zu unterscheiden, die auf sie trifft. Die jeweilige Farbe wird dabei nicht berücksichtigt, der Chip kann also nur Schwarz und Weiß erkennen, mit vielen Grautönen dazwischen. Das bedeutet, dass der Chip im Grunde genommen farbenblind ist.

Um die drei wichtigsten Farbkomponenten unseres Bildes, Rot, Blau und Grün erkennen zu können, sind optische Filter erforderlich. Solche Filter werden jeder einzelnen Diode oder, um die Formulierung zu vereinfachen, jedem „rohen" Pixel vorgesetzt. Das heißt, wir haben Millionen von Rohpixeln, die für jede Farbe, auf die sie abgestimmt sind, einen individuellen Pegel melden können. Die Pixel sind nicht in einem gleichmäßigen Rot-Blau-Grün-Muster über den Chip verteilt, sondern in einer Bayer-Matrix mit 50% grünen und und je 25% roten und blauen Rohpixeln angeordnet, was in der menschlichen optischen Wahrnehmung begründet liegt. Eine genauere Erklärung liefert der Wikipedia-Artikel unter [2]. Der Sensor muss jedes dieser Rohpixel abtasten und aus diesen Millionen von Informationen ein entsprechendes RGB-Pixelbild erzeugen. Ein Bildsensor ist heutzutage aber nicht mehr nur ein einfacher Analog-Digital-Wandler, er behandelt auch schon Teile der Bildverarbeitung.

Bildübertragung

Nachdem der Bildsensor die eingehenden Informationen aufbereitet hat, müssen sie an einen Host zur Weiterverarbeitung übergeben werden. Die grundlegende Übertragung, die durchgeführt werden kann, ist die Auslagerung der Daten Takt für Takt auf einen parallelen Bus. Dies ist einfach und wird auf kleineren eingebetteten MCUs auch so durchgeführt, aber mit Einschränkungen verbunden. Die parallele Übertragung vieler Bits bei hohen Geschwindigkeiten kann nämlich eine echte Herausforderung darstellen, insbesondere in Bezug auf das Timing und die Datenintegrität über längere Strecken, und kann für jedes Kameramodul proprietär sein.

Die *Mobile Industry Processor Interface Alliance* (MIPI) hat das *Camera Serial Interface* (CSI) definiert, um die Kompatibilität verschiedener Kameramodule zu ermöglichen. Das CSI-2 definiert eine serielle Datenverbindung zwischen Kamera und dem Host, die aus mehreren Leitungspaaren (data lanes) besteht. Beim Raspberry Pi 4B werden nur zwei Leitungspaare zum Kameraanschluss geführt, was zu einem Durchsatz von 3 Gbit/s führt, da jede Lane 1,5 Gbit/s übertragen kann. Da die CSI-2 Dokumentation der MIPI-Allianz leider nicht frei verfügbar ist, bedeutet dies, dass der Treiber für die Kamera-Schnittstelle

Spezifikationen der Raspberry Pi High Quality Camera

- Sensor: gestackter, rückseitig beleuchteter Sony IMX477R mit 12,3 Megapixel, 7,9 mm Sensordiagonale, 1,55 µm × 1,55 µm Pixelgröße
- Ausgabe: RAW12/10/8 oder COMP8
- Backfokus: einstellbar (12,5 mm bis 22,4 mm)
- Objektivanschluss: C-Mount, CS-Mount (C-CS-Adapter im Lieferumfang enthalten)
- IR-Sperrfilter: Integriert
- Flachbandkabel-Länge: 200 mm
- Stativanschluss: 1/4"-20

Bild 1. Aufbau der High Quality Kamera mit dem Raspberry Pi.

auf dem Raspberry PI 4B in einer proprietären Firmware versteckt ist, die auf der GPU läuft.
Wenn wir etwas rechnen, ergibt die Übertragung von Bildern mit mehr als 1920 x 1080 Pixel und 24 Bit Farbtiefe (acht Bit pro Kanal) 49,77 MBit pro Bild. Bei 3 Gbit/s Bandbreite können also rund 60 Bilder pro Sekunde übertragen werden. Theoretisch könnten bei 2560 x 1440 Pixel und 24 Bit Farbtiefe knapp über 30 Bilder pro Sekunde übertragen werden, allerdings kann diese Auflösung nicht für Videostreaming verwendet werden. Bei Standbildern ist die Bandbreite weniger das Problem, selbst bei einer maximalen Auflösung von 12 MP, die das Raspberry Pi High Quality Modul bietet.

Ein Kamera-Baukasten

Betrachten wir die Zutaten, die wir zum Bau unserer Kamera benötigen. Im Zentrum steht die *Raspberry Pi High Quality Camera* mit einer Auflösung von 12 MP (**Bild 2** und **Bild 3**). Das Kameramodul besitzt einen CS-Mount-Objektivanschluss. Die Spezifikation finden Sie im entsprechenden Textkasten. Wir verwenden das 16-mm- Objektiv mit C-Mount (**Bild 4**) und das 6-mm-Objektiv mit CS-Mount (**Bild 5**). Beide Objektive sind im Elektor Store [3][4] erhältlich.
Zur Verarbeitung der Bilder kommt der leistungsstarke Raspberry Pi 4B zum Einsatz (**Bild 6**). Alle Teile werden in einem 3D-druckbaren Gehäuse von Adafruit untergebracht, dem nur noch ein paar Schrauben hinzugefügt werden müssen. Weitere Details zum Gehäuse finden Sie unter [5]. Die gedruckten Teile sind in **Bild 7** zu sehen.
Der optionale 3,5-Zoll-TFT-Screen (**Bild 8**) und das Stativ (**Bild 9**) sind Überreste aus früheren Projekten. **Bild 10** zeigt schließlich die „eigenhändig" zusammengebaute Kamera, die man nicht vor Publikum verstecken muss. Im Gegenteil!.
Die Objektive haben unterschiedliche Gewindeanschlüsse, C-Mount und CS-Mount. Beide Anschlüsse passen zwar mechanisch auf den CS-Mount-Anschluss der Kamera, haben aber einen unterschiedlichen Abstand zur Bildebene des Sensors. Denken Sie also daran, stets den Adapterring in **Bild 11** hinzuzufügen oder zu entfernen, je nachdem, welches Objektiv Sie verwenden. Wenn Sie scheinbar nicht fokussieren können und unscharfe Bilder erhalten, haben Sie wahrscheinlich vergessen, den Ring beim Einsatz des CS-Mount-Objektivs zu entfernen oder beim C-Mount-Objektiv einzusetzen.

Aufnehmen des ersten Bildes

Wenn alles zusammengebaut ist, wird ein frisches Raspberry Pi OS auf einer SD-Karte installiert. Da das Display über HDMI angeschlossen ist, sind keine weiteren Schritte nötig, um ein Bild darauf zu bekommen, nur ein passendes HDMI-Kabel. Wenn Sie eines der billigeren 3,5-Zoll-Displays mit reiner SPI-Anbindung verwenden, müssen Sie eventuell einige Dateien im Raspberry Pi OS anpassen. Wenn Sie ein neues Display kaufen, erspart Ihnen ein HDMI-Eingang diese Modifikation und Sie erhalten in der Regel direkt beim Einschalten eine Bildausgabe.
Nachdem das Raspberry Pi OS die Ersteinrichtung vorgenommen hat, muss das Kameramodul aktiviert werden. Hierfür können Sie die *Raspberry Pi Configuration* verwenden. Aktivieren Sie auf dem Tab *Interfaces* die Option *Camera*. Wenn Sie das System über ein Netzwerk

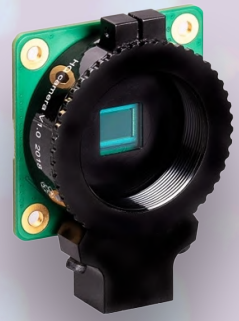


Bild 2. Das Raspberry Pi High Quality Camera Module (Quelle: www.raspberrypi.org).

Bild 3. Befestigungsmöglichkeit am Kameramodul (Quelle: www.raspberrypi.org).

Bild 4. 16-mm-Objektiv für das Raspberry Pi High Quality Camera Module.

Bild 5. 6-mm-Objektiv für das Raspberry Pi High Quality Camera Module.

steuern möchten, können Sie auch *SSH* und *VNC* aktivieren. Nach einem Neustart sollte die Kamera einsatzbereit sein.
Ein erster Test kann am Terminal durchgeführt werden. Geben Sie `raspistill -fp -s` ein, um eine Vorschau auf das aktuelle Kamerabild zu erhalten. Bei VNC für den Fernzugriff auf Ihren Desktop sehen Sie zunächst im VNC-Viewer keine Kameraausgabe. Dazu müssen Sie erst die VNC-Server-Einstellungen wie in **Bild 12** angegeben ändern.
Um ein Foto zu machen und es als Bilddatei zu speichern, geben Sie `raspistill -q 100 -o img.jpg` ein: Es wird eine Datei *img.jpg* erstellt und die Qualität wird für eine Komprimierung von 100 gesetzt.

Erste Testaufnahmen

Wenn die Kamera diese erste Aufnahme erfolgreich absolviert hat, können wir einige Testbilder schießen, zunächst mit dem 6-mm-Objektiv aus Bild 5. **Bild 13** zeigt eine A5-Notizkarte mit geraden Linien. Die Nahaufnahme in **Bild 14** beweist, dass das Objektiv als eine Art Fischauge arbeitet. Das gesamte einfallende Licht wird zwar in ein „glattes" Bild umgewandelt, das aber eine leichte Kugeloptik besitzt, wenn die Karte 13 cm von der Linse entfernt ist. Die technischen Daten für dieses Objektiv finden Sie unter [4].
Machen wir einen Vergleich mit dem 16-mm-Objektiv. Dieselbe A5-Notizkarte aus derselben Entfernung betrachtet wird, wie **Bild 15** beweist, anders als beim 6-mm-Objektiv nicht verzerrt. Die Spezifikationen für dieses Objektiv finden Sie unter [3].
Doch das Objektiv ist, was die Bildqualität angeht, nur die halbe Wahrheit. Für die andere Hälfte ist das Raspberry Pi High Quality Camera Module aus Bild 3 mit dem Sony-Sensor IMX477 [6] verantwortlich.

Ein RPi ist keine Fotokamera

Mit dem Raspberry Pi 4B haben wir die grundlegende Hardware, die man auch in einer Fotokamera finden würde. Eine bei modernen Kameras übliche, ja fast unverzichtbare Funktion fehlt allerdings bei unserer Konstruktion: der Autofokus. Das Scharfstellen des Objektivs muss deshalb manuell vorgenommen werden.
Wenn Sie moderne Fotokameras gewohnt sind, werden Sie auch schnell die fehlende Software bemängeln. Der Raspberry Pi ist softwaretechnisch halt kein echter Kameraersatz. Bildkorrektur, Filter und andere Einstellungen, die Sie bei einer Kamerasoftware finden, sind nicht vorhanden. Es liegt also an Ihnen, eine Kamerasoftware zu entwickeln, mit der Sie arbeiten können. Derzeit konzentriere ich mich nicht auf die Softwareentwicklung für die Kamera, aber ich könnte dies in naher Zukunft tun.

Video-Streaming

Obwohl dies also keine typische Fotokamera ist, ähnelt sie einer solchen. Aber was ist mit dem Streamen von Videos? Der Raspberry Pi kann Videos streamen, aber wir sind auf 1080p mit 30 Frames pro Sekunde (FPS) beschränkt. Trotzdem klingt es gut, eine Kamera zu haben, die über WLAN verbunden werden kann, und bei der wir Einfluss haben auf die Software, die auf der Kamera läuft

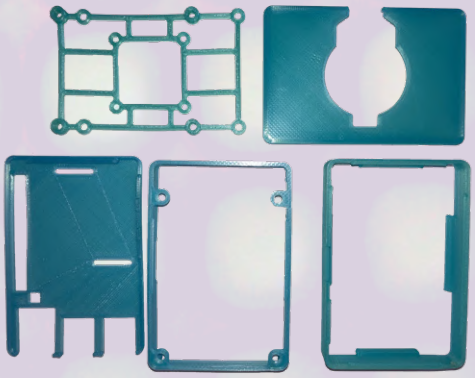


Bild 6. Alle Einzelteile des Kameragehäuses.

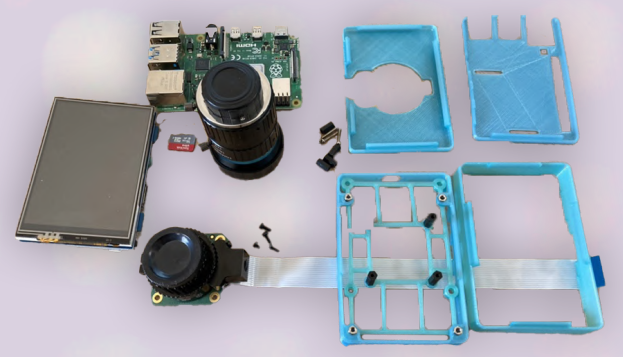


Bild 7. Alle Teile des Kamera-Baukastens.

Bild 8. Das optionale TFT-Modul.

Bild 9. Standfest: das optionale Stativ.

Bild 10. Die zusammengebaute Kamera.

Für die Streaming-Software gibt es Sie verschiedene Optionen. Ich habe mich für *µStreamer* aus dem pi-kvm-Projekt entschieden. Wenn Sie dasselbe tun wollen, müssen Sie der Anleitung [7] folgen, um die Software mit Unterstützung für OpenMAX zu kompilieren. Mit OpenMAX können Sie die Hardwarebeschleunigung für Multimedia-Aufgaben wie Bildkomprimierung nutzen, so dass ein MJPEG-Stream erzeugt wird, ohne dass die CPU die ganze Arbeit machen muss.

Klonen Sie das Repository mit `git clone --depth=1 https://github.com/pikvm/ustreamer` und installieren Sie die benötigten Bibliotheken mit `sudo apt install libevent-dev libjpeg8-dev libbsd-dev libraspberrypi-dev` auf dem Raspberry Pi. Dann müssen Sie im Terminal das µstreamer-Verzeichnis aufrufen und `make OMX=1` eingeben, um die Kompilierung zu starten. Es dauert nur einen kurzen Moment, bis eine ausführbare Datei erstellt ist.

Um einen Stream zu starten, geben Sie beispielsweise `./ustreamer --format=YUYV --encoder=OMX --workers=3 --host=0.0.0.0 --port=8080 -r 1920x1088 -q 90 -f 30` ein. Mit diesen Optionen wird ein MJPEG-Stream mit 1920 x 1080 Pixeln bei 30 FPS eingerichtet. Wenn Sie nun mit einem Browser auf den Raspberry Pi an Port 8080 zugreifen, sehen Sie eine Webseite wie in **Bild 16**. Die Framerate des Streams dürfte sehr niedrig sein, etwa 3 FPS bis 4 FPS, allerdings mit einer guten Videoqualität.

Treiber und Treibermodi

Der Raspberry Pi bietet für alle Kameramodule einen V4L2-Treiber, mit dem man bequem Anwendungen für Video- oder Audioeingaben erstellen kann. Der Treiber macht die Verwendung der Kamera mit dem Raspberry Pi sehr einfach und eröffnet eine breite Palette von Software-Lösungen, hat aber einen Haken: Solange wir den Treiber auffordern, Bilder mit einer Auflösung von 1270 x 720 Pixeln oder niedriger zu erzeugen, wird er das angeschlossene Kameramodul im Videomodus verwenden. Das bedeutet, dass wir bis zu 60 FPS (bei 1270 x 720 Pixel) erhalten, was völlig in Ordnung ist, wenn Sie einen kleinen und billigen Sensor für eine Art Überwachungskamera verwenden. Wenn Sie aber eine höhere Auflösung, zum Beispiel 1920 x 1080 Pixel wünschen, fällt der Treiber in den Fotomodus zurück und behandelt den angeschlossenen Sensor wie eine Fotokamera. Er führt alle Bildbearbeitungen, Helligkeitskorrekturen und Nachbearbeitungen durch, als ob er Fotos knipsen würde. Dies führt zwar zu exzellent aussehenden Bildern, aber auch zu einer sehr niedrigen Framerate.

Um dies zu umgehen, müssen wir den V4L2-Treiber mit `sudo rmmod bcm2835-v4l2` „entladen" und ihn mit `sudo modprobe bcm2835-v4l2 max_video_width=4056 max_video_height=3040` wieder neu laden. Die Parameter erzwingen den Videomodus für alle Auflösungen, die der Raspberry Pi unterstützt. Wenn wir nun den µStreamer mit `./ustreamer --format=YUYV --encoder=OMX --workers=3 --host=0.0.0.0 --port=8080 -r 1920x1088 -q 90 -f 30` starten, ist die resultierende Framerate deutlich höher. Wenn Sie den Stream erneut öffnen, können Sie auch auf die Statistik zugreifen und sehen, dass die resultierende Framerate nun bei etwa 20 FPS liegt, je nach Browser etwas mehr oder weniger.

Bild 11. Der CS-Mount-Adapterring.

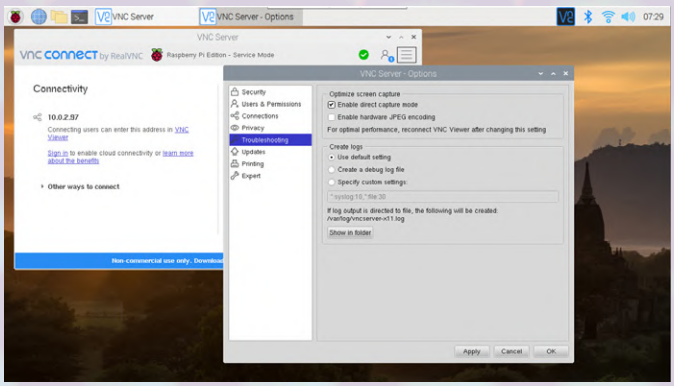


Bild 12. VNC-Einstellungen zum Betrachten der Kameraausgabe.

Bild 13. A5-Notizkarte mit geraden Linie.

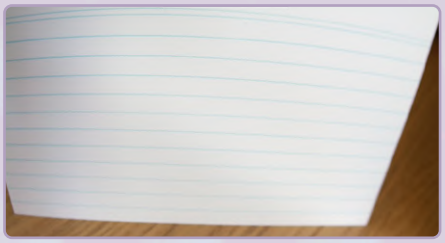


Bild 14. Nahaufnahme mit dem 6-mm-Objektiv.

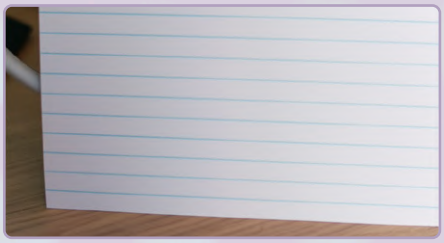


Bild 15. A5-Notizkarte, aufgenommen mit 16-mm-Objektiv.

Wenig Delay und noch mehr Frames

Ein Nachteil der obigen Methode ist die Verzögerung, die in den Stream eingebracht wird. Der gesamte Prozess vom Start am Raspberry Pi bis zur Darstellung in Ihrem Browser dauert etwa eine Sekunde. Diese Verzögerung kann beim Streaming über ein Netzwerk sehr unangenehm sein. Eine mögliche Verringerung der Verzögerung hängt vom Raspberry Pi selbst ab, von der Zeit, die für die Aufnahme und das Senden über das Netzwerk benötigt wird, und auch davon, wie lange die Dekodierung und die Anzeige des Bildes auf dem Wiedergabegerät dauert. Eine Möglichkeit, die Zahl der Frames pro Sekunde zu erhöhen, ist das Kommandozeilentool *raspivid*. Mit `raspivid -t 0 -w 1920 -h 1080 -fps 30 -l -o tcp://0.0.0.0:5000` ermöglicht man einem Videoplayer wie VLC, sich mit Ihrem Raspberry Pi zu verbinden und den Videostream wiederzugeben, immer noch mit einer Latenz von etwa einer Sekunde.

RTSP-Streaming mit H.264

Neben dem MJPEG-Streaming oder der Verwendung von raspivid mit TCP können wir das *Real Time Streaming Protocol* (RTSP) verwenden, das zur Steuerung des Datenstreaming über das Netzwerk entwickelt wurde und den Datenaustausch zwischen Clients und Streaming-Server regelt. Da RTSP den Datenaustausch aber nur verwaltet, wird auch etwas benötigt, das einen komprimierten Datenstrom erzeugen kann, den wir transportieren können. An dieser Stelle kommt das Webserver-Tool *UV4L* (User space Video4Linux) ins Spiel. UV4L verbindet den Kameraeingang des Raspberry Pi 4B mit dem im Raspberry Pi enthaltenen H.264-Encoder und gibt einen H.264-komprimierten Datenstrom aus.

Dieses Setup erzeugt einen Video-Stream (kein Audio) mit einer Verzögerung von weniger als 1 s bei 6 Mbit/s und nur etwa 10 % CPU-Auslastung. Um die Verzögerung zu messen, wurde die Kamera auf einen Monitor gerichtet, der die lokale Zeit anzeigt, und darüber ein VLC-Player, der das gestreamte Video abspielt (**Bild 17** und **Bild 18**). Wir können eine Verzögerung von 670 ms für die komplette Signalverarbeitung ausmachen, was zwar immer noch nicht perfekt ist, aber viel praktikabler. **Bild 19** zeigt die Details für den von diesem Setup erzeugten Stream.

Und wie richtet man das alles ein? Zuerst müssen Sie Basis-UV4L auf Ihrem Raspberry Pi 4B gemäß der Anleitung unter [8] und dann die Extras mit `sudo apt-get install uv4l-raspicam-extras` installieren. Sie haben damit ein Videogerät, das H264-Streams ausgeben kann. Die Konfiguration wird in einem späteren Schritt vorgenommen. Der nächste Schritt betrifft den v4l2rtspserver [9]. Klonen Sie das Repository mit `git clone https://github.com/mpromonet/v4l2rtspserver.git` auf Ihren Raspberry Pi 4B, installieren Sie, wenn noch nicht vorhanden, *CMake* mit `sudo apt install cmake`, um den v4l2rtspserver kompilieren zu können. Geben Sie im Terminal `v4l2rtspserver` ein, der nun erstellt wird. Geben Sie `cmake . && make` ein, um den Code zu kompilieren. Wenn alles wie erwartet funktioniert, geben Sie `sudo make install` ein.

Angesichts des Forenbeitrags [10] können wir die Anleitung etwas abkürzen. Geben Sie im Terminal `sudo nano /etc/uv4l/uv4l-raspicam.conf` ein und suchen Sie nach der Zeile `encoding`.

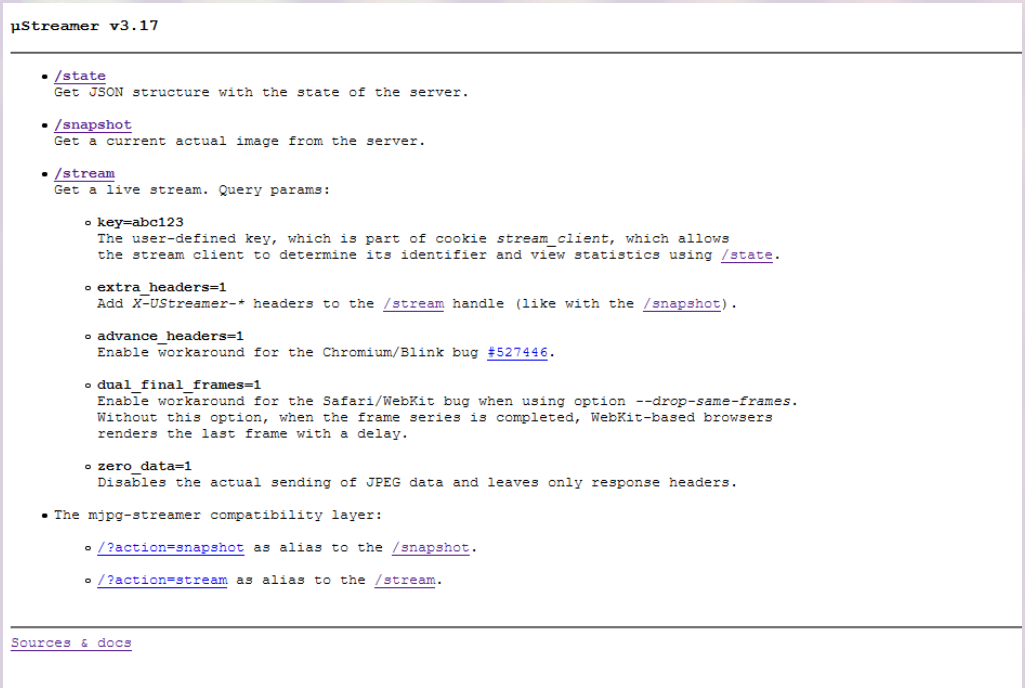


Bild 16. Das µStreamer-Webinterface.

Ändern Sie diese auf `encoding = h264`. Suchen Sie auch nach `width` und `height` und ändern Sie diese Werte auf `width = 1296` und `height = 972`. Als Nächstes suchen Sie nach `framerate` und setzen den Wert auf `framerate = 30`. Speichern Sie diese Änderungen.
Sie werden sich an dieser Stelle vielleicht fragen, warum wir uns nicht für Full HD (1920x1080 Pixel) entscheiden. Diese Auflösung hätte jedoch ein Seitenverhältnis von 16:9. Da der Sensor aber ein 4:3-Bild liefert, würden wir einige Teile des Bildes verlieren. Wenn wir wirklich nur 16:9 benötigen, können wir dies später einstellen.
Als nächstes müssen wir ein Skript bearbeiten, um beim Start uv4l für die Raspberry High Quality Camera zu laden. Geben Sie in einem Terminal `sudo nano /etc/systemd/system/uv4l_raspicam.service` ein und ersetzen Sie die Zeile `ExecStart` durch `ExecStart=/usr/bin/uv4l -f -k --sched-fifo --mem-lock --config-file=/etc/uv4l/uv4l-raspicam.conf --driver raspicam --driver-config-file=/etc/uv4l/uv4l-raspicam.conf --enable-server off`. Speichern Sie die Datei und starten Sie den Raspberry Pi neu.
Um nun von Ihrer Kamera zu streamen, öffnen Sie ein Terminal und geben Sie `export LD_PRELOAD=/usr/lib/uv4l/uv4lext/armv6l/libuv4lext.so` ein, sonst lädt v4lrtspserver gar nichts und verabschiedet sich mit einer Fehlermeldung. Um einen Teststream zu starten, geben Sie in einem Terminal `v4l2rtspserver -F 30 -H 960 -W 1280 -P 8555 /dev/video0` ein. Der v4lrtspserver sollte starten und ein Vorschaufenster auf dem Desktop anzeigen.
Um den Videostream anzuzeigen, können Sie VLC [11] verwenden und dort zu *Medien/Netzwerkstream öffnen* gehen. Als URL verwenden Sie *rtsp://<deine.raspberry.pi.ip>:8555/unicast*. Unter *Mehr Optionen anzeigen* sollte *Zwischenspeicherung* auf maximal *100 ms* gesetzt sein, um die beste Echtzeit-Performance zu erreichen. Nach einem Klick auf *Wiedergabe* sollten Ihren Kamera-Stream sehen.

USB-OTG

Da der Transport von Videos über eine Netzwerkverbindung zu Verzögerungen führen kann, wie wäre es, stattdessen USB zu verwenden? Der Raspberry Pi 4B+ bietet USB-OTG-Unterstützung, wodurch er als USB-Gerät fungieren kann, wenn er an einen PC angeschlossen ist. Derzeit hat der USB-Teil, der benötigt wird, um den Raspberry Pi 4 in eine Webcam zu verwandeln, noch einige Bugs [12], daher werden wir ihn für diesen Artikel nicht verwenden. Außerdem ist für dieses Setup ein Y-Kabel nötigt, um Stromversorgung und Daten vom USB-Anschluss zu trennen, das derzeit noch von Hand gefertigt werden muss. Also ist keine stabile Videoausgabe mit niedriger Latenz möglich, zumindest nicht mit dem Raspberry Pi 4. Beim Raspberry Pi Zero W könnte das anders sein, wie die Show-me-Webcam [13] demonstriert.

Uuuuund: action!

Die montierte Kamera steht, wie in Bild 1 zu sehen, auf meinem Schreibtisch und wird für die Aufnahme von Bildern und die Aufzeichnung und Übertragung von Videos verwendet. Der Bildsensor in Kombination mit den C- und CS-Mount-Objektiven erzeugt ein tolles Bild im Fotomodus und bei der Videoaufnahme. Die Software und der Treiber aber lassen mich in Verbindung mit der Hardware mit gemischten Gefühlen zurück. Der Raspberry Pi 4B hat Einschränkungen in Bezug auf die Videokodierung und die Implementierung der Kamera-Schnittstelle. Während ich diesen Artikel schrieb, kamen mir einige Ideen, um die vorhandene Software zu verbessern oder zu nutzen. Die erste wäre, ein übersichtliches UI für einfache Fotografie mit Live-Vorschau und Bildaufzeichnung zu implementieren. Im besten Fall würde dieses auch über Netzwerke mit weniger als 1 s Latenz funktionieren. Und, da die Hardware des Raspberry PI 4B dazu in der Lage ist, sollte auch der Einsatz von USB-OTG genauer unter die Lupe genommen werden.

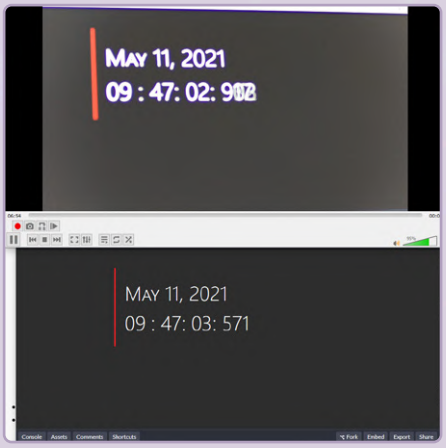


Bild 17. Erste Messung für die Stream-Verzögerung.

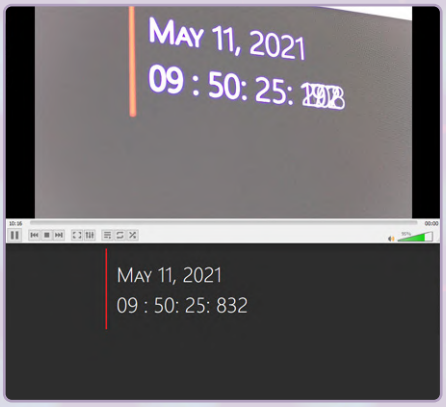


Bild 18. Zweite Messung für die Stream-Verzögerung.

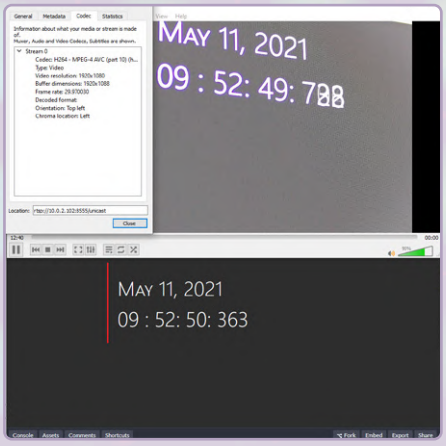


Bild 19. Die Stream-Parameter auf einem Blick.

Es bleibt Ihnen also noch allerhand Gestaltungsmöglichkeiten – denn ehrlich, wenn es von Anfang an perfekt funktioniert hätte, wo wäre dann der Spaß geblieben? Die besten Lektionen, die man lernen kann, kommen von den Herausforderungen, die ein Projekt bietet.

200582-02

Ein Beitrag von

Text und Bilder: **Mathias Claußen**
Redaktion: **Jens Nickel, 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 mathias.claussen@elektor.com oder kontaktieren Sie Elektor unter editor@elektor.com.

PASSENDE PRODUKTE

- **16 mm C-Mount-Objektiv (10 MP) für Raspberry Pi HQ Kameramodul**
  www.elektor.de/16-mm-c-mount-lens-10-mp-for-raspberry-pi-hq-camera-module
- **6 mm CS-Mount-Objektiv (3 MP) für Raspberry Pi HQ Kameramodul**
  www.elektor.de/6-mm-cs-mount-lens-3-mp-for-raspberry-pi-hq-camera-module
- **Raspberry Pi High Quality Kameramodul**
  www.elektor.de/raspberry-pi-high-quality-camera-module
- **Raspberry Pi 4 B (2 GB RAM)**
  www.elektor.de/raspberry-pi-4-b-2-gb-ram
- **Raspberry Pi 4 B (4 GB RAM)**
  www.elektor.de/raspberry-pi-4-b-4-gb-ram
- **Anycubic i3 Mega-S 3D Printer (Kit)**
  www.elektor.de/anycubic-i3-mega-s-3d-printer-kit

WEBLINKS

[1] Why MP for camera lense?: www.1stvision.com/machine-vision-solutions/2018/02/megapixel-machine-vision-lenses-2.html
[2] Bayer-Sensor: https://de.wikipedia.org/wiki/Bayer-Sensor
[3] 16-mm-Objektiv im Elektor-Store: www.elektor.de/16-mm-c-mount-lens-10-mp-for-raspberry-pi-hq-camera-module
[4] 6-mm-Objektiv im Elektor-Store: www.elektor.de/6-mm-cs-mount-lens-3-mp-for-raspberry-pi-hq-camera-module
[5] 3D-druckbares Gehäuse: https://learn.adafruit.com/raspberry-pi-hq-camera-case
[6] Informationen zum IMX477-Sensor: www.sony-semicon.co.jp/products/common/pdf/IMX477-AACK_Flyer.pdf
[7] µStreamer auf GitHub: https://github.com/pikvm/ustreamer
[8] UV4L- Installation: www.linux-projects.org/uv4l/installation/
[9] v4l2rtspserver auf GitHub: https://github.com/mpromonet/v4l2rtspserver
[10] Forenbeitrag zu h264-Streaming mit niedriger Latenz:
www.bensoftware.com/forum/discussion/3254/raspberry-pi-h264-rtsp-low-latency-camera-instructions/p1
[11] VLC-Player: www.videolan.org/
[12] Forenbeitrag: UVC gadget limitations for Raspberry Pi 4B: www.raspberrypi.org/forums/viewtopic.php?t=294140
[13] Show-me webcam auf GitHub: https://github.com/showmewebcam/showmewebcam

Verwendung von Displays in Raspberry-Pi-Projekten

Beispiel-Kapitel: Organische Leuchtdioden-Displays (OLED)

Von **Dogan Ibrahim** (Großbritannien)

In dieser Ausgabe von *Elektor Bücher* stellen wir Ihnen einen Teil eines Kapitels aus Dogan Ibrahims Buch *Using Displays in Raspberry Pi Projects* vor, das im März 2021 bei Elektor erschien und inzwischen ein Bestseller ist. OLED-Displays sind seit ihrer Einführung bei Entwicklern und Ingenieuren von Embedded-Systemen sehr beliebt. In diesem Artikel gehen wir der Frage nach, wie man OLED-Displays mit minimalem Programmieraufwand in Projekte einbinden kann - Anforderungen, die eindeutig auf den Raspberry Pi als perfekten Kandidaten hindeuten, wie wir meinen. Finden wir es heraus!

Bild 1. Einige OLED-Displays.

Anmerkung der Redaktion: Dieser Artikel ist ein Auszug aus dem Buch *Using Displays in Raspberry Pi Projects*, neu formatiert und leicht bearbeitet, um den Standards und dem Seitenlayout der Zeitschrift *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 des Buchs beziehen. Der Autor und die Redaktion haben jedoch ihr Bestes getan, um solche Fälle zu vermeiden, und helfen bei Rückfragen gerne weiter. Kontaktinformationen finden Sie im Kasten „Fragen oder Kommentare?".

Eine organische Leuchtdiode ist eine Anzeige auf LED-Basis, bei der die emittierende Elektrolumineszenzschicht aus einem Film einer organischen Verbindung besteht, der Licht emittiert, wenn eine elektrische Spannung anliegt. Die organische Schicht befindet sich zwischen zwei Elektroden, von denen mindestens eine durchsichtig ist. OLED-Displays werden in vielen kommerziellen Anwendungen eingesetzt, etwa in Fernseh- und Computerbildschirmen, Smartphones, Spielekonsolen und anderen digitalen Anzeigen. OLEDs können entweder mit Passiv-Matrix- (PMOLED) oder Aktiv-Matrix-Controllern (AMOLED) angesteuert werden. Typischerweise wird bei PMOLED-Displays jede Zeile und jede Spalte sequentiell, also einzeln nacheinander angesteuert. Bei AMOLED-Displays wird eine Dünnschichttransistor-Backplane verwendet, um direkt auf jedes einzelne Pixel zuzugreifen und es an- oder auszuschalten.

Im Allgemeinen haben die OLED-Displays die folgenden Vorteile gegenüber den LCDs:

- Der Strombedarf von OLEDs ist sehr gering, was sie für mobile Display-Anwendungen attraktiv macht.
- OLED-Displays haben eine bessere Bildqualität, einen besseren Kontrast, eine höhere Helligkeit, einen größeren Betrachtungswinkel und eine schnellere Bildwiederholrate.
- OLED-Displays sind robuster und für den Einsatz in einem breiteren Temperaturbereich geeignet.
- OLED-Displays können so hergestellt werden, dass sie flexibel sind. Sie können zum Beispiel faltbar sein, am Handgelenk getragen werden und so weiter.

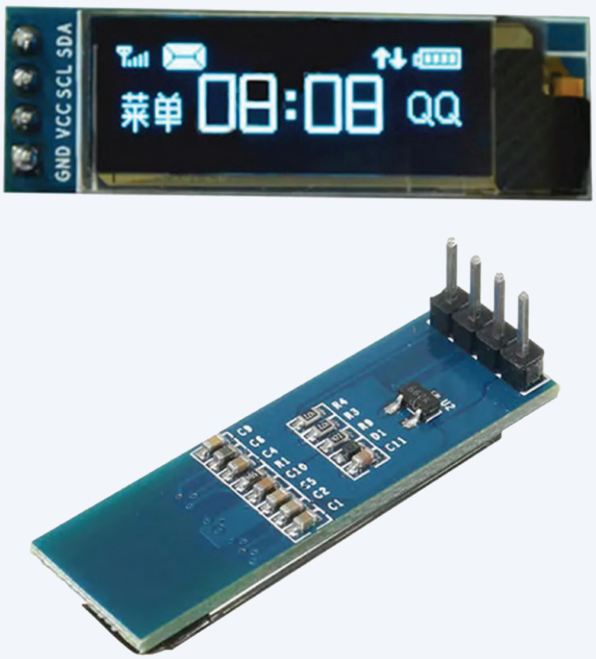


Bild 2. 128 x 32-Pixel-OLED. (Quelle: https://uk.banggood.com)

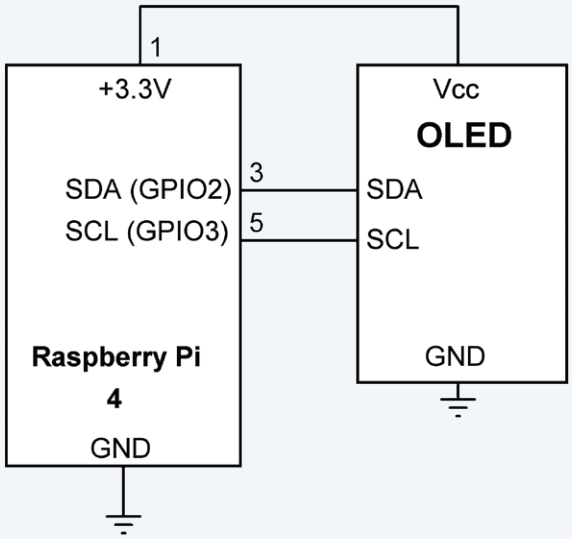


Bild 3. Der Schaltplan.

OLED-Displays sind jedoch nicht perfekt und sie haben auch einige Nachteile gegenüber LCDs:

- Die Kosten für OLED-Displays sind deutlich höher.
- OLED-Displays haben eine begrenzte Lebensdauer, obwohl sich dies in letzter Zeit verbessert.
- OLED-Displays können bei direkter Sonneneinstrahlung problematisch sein, da sie einen hohen Emissionsgrad haben.

Bild 1 zeigt Beispiele von einigen OLED-Displays.

Verwendung von OLED-Displays

Es gibt so viele Größen von OLED-Displays mit unterschiedlichen Pixelmatrizen, dass es nicht möglich ist, alle Arten in diesem Buch zu behandeln. In diesem Kapitel werden wir das weit verbreitete 128 × 32 Pixel große OLED-Display (**Bild 2**) verwenden, das mit dem On-Board-Chip SSD1306 angesteuert wird. Bevor wir es in unseren Projekten verwenden, sollte man sich mit den Eigenschaften des Displays beschäftigen.

Die Spezifikationen dieses OLED-Displays sind:

- Auflösung: 128 × 32 Pixel
- Pixelfarbe: weiß oder blau
- Displaydiagonale: 0,91 Zoll
- Schnittstelle: I²C
- Treiber: SSD1306
- 4 Pins: GND, VCC, SCL (I²C), SDA (I²C)
- Betrachtungswinkel: Größer als 160 Grad
- Betriebsspannung: +3,3 V / +5 V

Das auf dem SSD1306-Treiber basierende Display wird über die I²C-Schnittstelle mit dem Host-Rechner verbunden, was bedeutet, dass neben den Power- und GND-Pins nur zwei Pins für die Schnittstelle benötigt werden. Die I²C-Adresse des Treibers ist standardmäßig 0x3C. **Bild 3** zeigt die einfache Verbindung zwischen einem Raspberry Pi und dem OLED-Display. Der Chip selbst arbeitet normalerweise mit +3,3 V, es befindet sich jedoch ein Spannungsregler auf dem Board, der den Betrieb auch mit +5 V ermöglicht.

RPi-Anschluss	RPi-Pin	OLED-Pin
SDA	(GPIO2)	2
SCL	(GPIO3)	3
GND	39	GND
+3,3V	1	V_{cc}

Installieren der OLED-Bibliothek

Die I²C-Schnittstelle muss auf dem Raspberry Pi im Konfigurationstool *sudoraspi-config* aktiviert werden. Danach sollten Sie die OLED-Bibliothek installieren. In diesem Kapitel wollen wir die Bibliothek *Adafruit_Python_SSD1306* verwenden.
Die Schritte sind wie folgt (einige dieser Bibliotheken sind möglicherweise bereits auf Ihrem Raspberry Pi installiert):

```
pi@raspberrypi:~ $ sudo apt install -y python3-dev
python3- pil python3- pip python3-setuptools python3-
rpi.gpio
pi@raspberrypi:~ $ sudo pip3 install
adafruit-circuitpython-ssd1306
```

```
pi@raspberrypi:~ $ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- 3c -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
pi@raspberrypi:~ $
```

Bild 4. Erkennung des OLED-Displays.

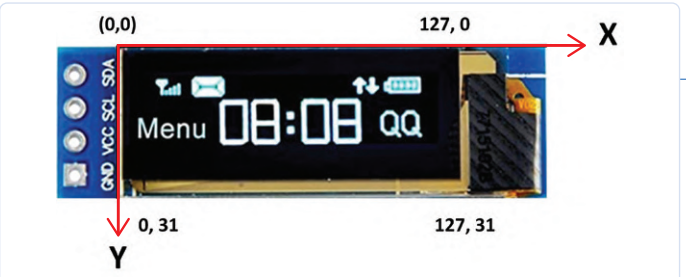


Bild 5. Koordinaten des Displays.

Listing 1. OLEDCorners.py

```
#-----------------------------------------------------------------
   --
#        SHOW THE PIXELS AT CORNERS OF THE DISPLAY
#        =========================================
#
# This program shows the pixels at the four corners of the display
#
# Author: Dogan Ibrahim
# File   : OLEDCorners.py
# Date   : November 2020
#-----------------------------------------------------------------
   --
from board import SCL, SDA
import busio
import adafruit_ssd1306

i2c = busio.I2C(SCL, SDA)
display = adafruit_ssd1306.SSD1306_I2C(128, 32, i2c)

display.fill(0)                 # Clear pixels
display.show()                  # Display data

display.pixel(0,0,1)            # Pixel at (0,0)
display.pixel(127,0,1)          # Pixel at (127,0)
display.pixel(0,31,1)           # Pixel at (0,31)
display.pixel(127, 31, 1)       # Pixel at (127,31)
display.show()                  # Display the data
```

Listing 2. OLEDText.py

```
#-----------------------------------------------------------
#            DISPLAY TEXT
#            ============
#
# This program displays the text E L E K T O R at (15,5)
#
# Author: Dogan Ibrahim
# File   : OLEDText.py
# Date   : November 2020
#-----------------------------------------------------------
from PIL import Image,ImageDraw,ImageFont
from board import SCL, SDA
import busio
import adafruit_ssd1306

i2c = busio.I2C(SCL, SDA)
display = adafruit_ssd1306.SSD1306_I2C(128, 32, i2c)
font = ImageFont.load_default()         # Default font

display.fill(0)                         # Clear display
display.show()

width = display.width                   # Width
height = display.height                 # Height
image = Image.new('1',(width,height))
draw = ImageDraw.Draw(image)
draw.text((15, 5), "E L E K T O R", font = font, fill = 255)
display.image(image)
display.show()
```

Bevor ein an die I²C-Schnittstelle angeschlossenes Gerät verwendet werden kann, muss zunächst dessen I²C-Adresse ermittelt werden. Um die I²C-Adresse des Displays zu ermitteln, sollte der Befehl

```
pi@raspberrypi:~ $ i2cdetect -y 1
```

im Terminal ausgeführt werden und ein Ergebnis wie in **Bild 4** erscheinen.

Projekt 1: Anzeige von Pixeln an den vier Ecken des Displays

Die Koordinaten des OLED-Displays sind in **Bild 5** dargestellt. Die linke obere Ecke ist (0, 0). Die X-Koordinate verläuft waagrecht nach rechts über den Bildschirm von 0 bis 127, die Y-Koordinate von 0 bis 31 nach unten.

Programmlisting: Das Programm *OLEDCorners.py* ist in **Listing 1** dargestellt. Dieses Programm ist in der Software-Archivdatei zum Buch enthalten, die Sie unter [1] unter *Downloads* finden. Zu Beginn des Programms wird die OLED-Bibliothek von Adafruit in das Programm importiert. Anschließend werden die Pixel durch den Aufruf der Funktion `display.fill(0)` gelöscht, wie es immer am Anfang eines Programms geschehen sollte. Anschließend werden die Pixel an den vier Ecken des Displays eingeschaltet. Beachten Sie, dass abschließend die Funktion `display.show()` aufgerufen werden muss, um die Daten auf dem Display anzuzeigen. **Bild 6** zeigt das Display mit den vier eingeschalteten Eckpixeln.

Projekt 2: Anzeige von Text

In diesem Projekt wird der Text „ELEKTOR“ ab der Koordinate (15, 5) des Displays angezeigt.

Programmlisting: **Listing 2** zeigt das Programm *OLEDText.py*. Es verwendet die Funktionen der PIL-Bibliothek [2]. Zu Beginn des Programms werden die Bild- und Schriftmodule der PIL-Bibliothek in das Programm importiert und danach die Standard-Schriftbibliothek geladen. Anschließend löscht das Programm die Pixel. In `width` und `height` sind die Breite (128) und die Höhe (32) des Displays gespeichert.
Um ein Bild zu erstellen, muss zunächst ein leeres Bild mit den Abmessungen des Bildschirms erzeugt werden. Dies geschieht mit der Anweisung:

```
image = Image.new('1', (width, height))
```

Mit diesem Image-Objekt wird das Draw-Objekt zum Zeichnen von Formen und Text erzeugt:

```
draw = ImageDraw.Draw(image)
```

In diesem Programm wollen wir einen Text an der Koordinate (15, 5) anzeigen und verwenden deshalb die Funktion `draw.text`:

```
draw.text((15, 5), „E L E K T O R“, font = font, fill
   = 255)
```

Beachten Sie, dass `fill = 255` das Bild in Weiß und `fill = 0` in Schwarz zeichnet. Danach müssen wir das Bild durch den Aufruf der Funktionen `display.image(image)` und `display.show()` anzeigen. In **Bild 7** ist der angezeigte Text zu sehen.

Projekt 3: Anzeige von Formen

Wir können verschiedene Formen auf dem Display zeichnen. Es gibt viel mehr Formen und Funktionen als die hier kurz beschriebenen. Eine vollständige Liste der Funktionen finden Sie in der Dokumentation der PIL-Bibliothek.

Rechteck (rectangle):
(x,y) ist die linke obere Ecke des Rechtecks. `Outline` ist die Farbe des Umrisses der Form (`255` für Weiß, `0` für Schwarz), `fill` ist das Innere der Form (`255` für Weiß, `0` für Schwarz).

```
draw.rectangle((x, y, width, height), outline = nn,
   fill = mm)
```

Ellipse (ellipse):

```
draw.ellipse((x, y, width, height), outline = nn,
   fill = mm)
```

Linie (line):
(x1,y1) ist die erste Koordinate, (x2,y2) ist die zweite Koordinate.

```
draw.line((x1, y1, x2, y2), fill = mm)
```

Kreisbogen (arc):

```
draw.arc((x1,x2,y1,y2), s1, s2, fill = mm)
```

Zeichnet innerhalb der Box (x1,x2,y1,y2) einen Kreisbogen zwischen Start- (s1) und Endwinkel (s2).

Kreissegment (chord):

```
draw.chord((x1,x2,y1,y2), s1, s2, fill = mm)
```

Zeichnet innerhalb der Box (x1,x2,y1,y2) eine Linie zwischen dem Start- (s1) und Endwinkel (s2).

Kreissektor (pieslice):

```
draw.pieslice((x1,y1,x2,y2), s1, s2, fill = mm)
```

„Tortenstück“, wie der Kreisbogen, zeichnet aber zusätzlich gerade Linien zwischen den Endpunkten und dem Mittelpunkt der Box.

Punkt (point):

```
draw.point((x1,x2), fill = mm)
```

Zeichnet Pixel an den angegebenen Koordinaten. Es kann mehr als ein Pixel gezeichnet werden, wie unten gezeigt, wo drei einzelne Pixel an den angegebenen Koordinaten gezeichnet werden:

```
draw.point([(x1,y1), (x2,y2), (x3,y3)], fill = mm)
```

Vieleck (polygon):

```
draw.polygon([(x1,y1), (x2,y2), (x3,y3), ……], outline
   = nn, fill = mm)
```

Bild 6. Wir haben Pixel an den vier Ecken des OLED-Displays gezeichnet!

Bild 7. Der angezeigte Text. „E L E K T O R“ - was sonst?

Listing 3. LEDRect.py

```
#-----------------------------------------------------------------
#
#        DISPLAY A RECTANGLE WITH TEXT INSIDE
#        ====================================
#
# In this progra a rectangle is displayed at the corners of the
# display and text R E C T A N G L E is displayed inside
#
# Author: Dogan Ibrahim
# File  : OLEDRect.py
# Date  : November.py
#-----------------------------------------------------------------
from PIL import Image,ImageDraw,ImageFont
from board import SCL, SDA
import busio
import adafruit_ssd1306

i2c = busio.I2C(SCL, SDA)
display = adafruit_ssd1306.SSD1306_I2C(128, 32, i2c)
font = ImageFont.load_default()

display.fill(0)
display.show()

width=(display.width)
height=(display.height)
image=Image.new('1',(width,height))
draw=ImageDraw.Draw(image)
draw.rectangle((0,0, 127, 31),outline = 255, fill = 0)
draw.text((15, 12),"R E C T A N G L E", font = font, fill = 255)
display.image(image)
display.show()
```

Listing 4. OLEDShape1.py

```
#---------------------------------------------------------------
#
#    DISPLAY 4 RECTANGLES WITH NUMBERS 1,2,3,4
#    ========================================
#
# In this progra 4 rectangles are displayed. Number 1 is displayed
# inside rectangle 1, number 2 inside rectangle 2, and so on
#
# Author: Dogan Ibrahim
# File  : OLEDShape1.py
# Date  : November 2020
#---------------------------------------------------------------
from PIL import Image,ImageDraw,ImageFont
from board import SCL, SDA
import busio
import adafruit_ssd1306

i2c = busio.I2C(SCL, SDA)
display = adafruit_ssd1306.SSD1306_I2C(128, 32, i2c)
font = ImageFont.load_default()

display.fill(0)
display.show()

width=(display.width)
height=(display.height)
image=Image.new('1',(width,height))
draw=ImageDraw.Draw(image)
draw.rectangle((0,0, 127, 31),outline = 255, fill = 0)
draw.line((64, 0, 64, 31), fill = 255)
draw.line((0, 16, 127, 16), fill = 255)
draw.text((32, 4), "1", font = font, fill = 255)
draw.text((94, 4), "2", font = font, fill = 255)
draw.text((32, 17),"3", font = font, fill = 255)
draw.text((94, 17),"4", font = font, fill = 255)
display.image(image)
display.show()
```

Listing 5. LEDShape2.py

```
#---------------------------------------------------------------
#
#        DISPLAY A CHORD AND A POLYGON
#        =============================
#
# In this program a chord and a polygon are drawn. The polygon is
# filled with white
#
# Author: Dogan Ibrahim
# File  : OLEDShape2.py
# Date  : November 2020
#---------------------------------------------------------------
from PIL import Image,ImageDraw,ImageFont
from board import SCL, SDA
import busio
import adafruit_ssd1306

i2c = busio.I2C(SCL, SDA)
display = adafruit_ssd1306.SSD1306_I2C(128, 32, i2c)
font = ImageFont.load_default()

display.fill(0)
display.show()

width=(display.width)
height=(display.height)
image=Image.new('1',(width,height))
draw=ImageDraw.Draw(image)
draw.chord((0, 0, 90, 30), 10, 180, fill = 255)
draw.polygon([(100,5), (120,9), (120,25),
   (100,30)],outline=255,fill=255)
display.image(image)
display.show()
```

Bild 8. Ein Rechteck mit dem Text RECTANGLE.

Bild 9. OLEDShape1.py erzeugt Rechtecke auf dem Bildschirm.

Bild 10. Die Anzeige nach dem Ausführen von OLEDShape2.py (Kreissegment und Polygon).

Haben Sie Fragen oder Kommentare?
Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Dann schreiben Sie eine E-Mail an den Autor unter d.ibrahim@btinternet.com oder Elektor unter editor@elektor.com.

Ein Beitrag von
Text: Dogan Ibrahim
Redaktion: Jan Buiting
Übersetzung: Rolf Gerstendorf
Layout: Giel Dols

Zeichnet ein Vieleck, dessen Umriss aus geraden Linien zwischen den angegebenen Koordinaten sowie einer geraden Linie zwischen der letzten und der ersten Koordinate besteht. Die Koordinatenliste kann ein beliebiges Sequenzobjekt sein, das entweder Zweiertupel [(x, y), ...] oder numerische Werte [x, y, ...] enthält. Ein Vieleck sollte mindestens drei Koordinaten enthalten.

Das in **Listing 3** gezeigte Programm *OLEDRect.py* zeigt, wie ein Rechteck an den Ecken des Displays und der Text „RECTANGLE" innerhalb dieses Rechtecks gezeichnet wird (**Bild 8**).
Das in **Listing 4** gezeigte Programm *OLEDShape1.py* zeichnet vier Rechtecke. In Rechteck 1 wird die Zahl „1" angezeigt, in Rechteck 2 die Zahl „2" und so weiter (**Bild 9**)
Das in **Listing 5** gezeigte Programm *OLEDShape2.py* stellt ein Kreissegment und ein Polygon mit vier Ecken dar. Der Füllparameter des Polygons ist auf 255 gesetzt, so dass es mit einem weißen Hintergrund gefüllt wird. **Bild 10** zeigt die resultierende Anzeige.

Projekt 4: Erstellen und Darstellen einer Bitmap

In diesem Abschnitt erstellen wir ein Bitmap-Bild mit den Abmessungen 128 × 32 Pixel und zeigen es anschließend auf dem Display. Das Bild, das wir erstellen werden, ist die Handschrift des Buchstabens „A".

Listing 6. OLEDBitmap.py

```
------------------------------------------------------
#
#        DISPLAY BITMAP
#        ==============
#
# This progra displays a bitmap image
#
# Author: Dogan Ibrahim
# File  : OLEDBitmap.py
# Date  : November 2020
#------------------------------------------------------
from PIL import Image,ImageDraw,ImageFont
from board import SCL, SDA
import busio
import adafruit_ssd1306

i2c = busio.I2C(SCL, SDA)
display = adafruit_ssd1306.SSD1306_I2C(128, 32, i2c)

display.fill(0)
display.show()

width=(display.width)
height=(display.height)
image=Image.new('1',(width,height))
image=Image.open('LetterA.png').
   resize((width,height),\
Image.ANTIALIAS).convert('1')
display.image(image)
display.show()
```

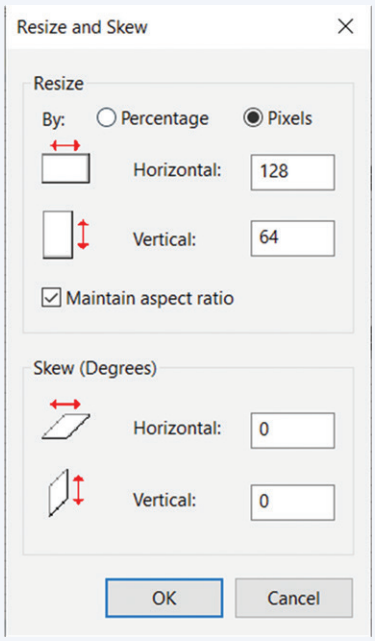


Bild 11. Setzen Sie in MS Paint die Pixelanzahl auf 128 × 64.

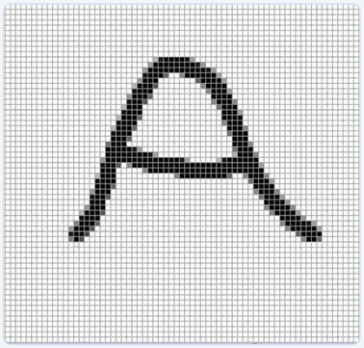


Bild 12. Zeichnen Sie die gewünschte Form in MS Paint.

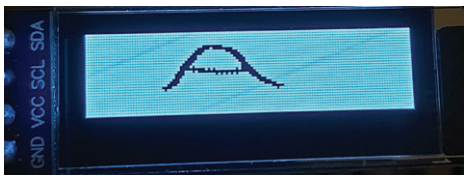


Bild 13. Erfolg! Ein quasi handgeschriebener Buchstabe auf dem OLED-Display.

Alle in diesem Artikel besprochenen Programme sind auf der Elektor-Webseite zum Buch [1] erhältlich. Scrollen Sie auf dieser Seite nach unten zu *Downloads*, um die Datei mit den gesammelten Programmen zu finden. Laden Sie die .zip-Datei herunter, entpacken Sie sie auf Ihrem Computer und suchen und öffnen Sie dann die sechs hier besprochenen Python-Programme.

210197-02

Erstellen des Bildes: Wir verwenden das Microsoft-Programm Paint, um unser Bild zu zeichnen. Starten Sie dazu das Programm MS Paint.

- Klicken Sie auf *Start/Größe ändern* und dann auf *Pixel*. Stellen Sie *Horizontal* auf 128 Pixel und die *Vertikal* auf 64 Pixel ein, wie in **Bild 11** gezeigt. Klicken Sie auf OK.
- Klicken Sie auf *Ansicht*, auf *Gitternetzlinien* und dann auf *Vergrößern*, um das Gitter zu vergrößern.
- Zeichnen Sie die gewünschte Form mit der Maus. In diesem Projekt wird der Buchstabe A mit der Hand gezeichnet, wie in **Bild 12** dargestellt.
- Speichern Sie die Datei, zum Beispiel unter dem Namen *LetterA.png*.
- Kopieren Sie die Datei in das Home-Verzeichnis auf dem Raspberry Pi (*/home/pi*). Zum Kopieren der Datei können Sie das freie Programm *WinSCP* verwenden.

Das fertige Programm *OLEDBitmap.py* ist in **Listing 6** zu sehen. Die Funktion `Image.open` wird verwendet, um die Bilddatei zu öffnen, die Datei wird in der Größe angepasst und in 1-Bit-Farbe konvertiert, wie es die Bibliothek verlangt. **Bild 13** zeigt schießlich die Darstellung auf dem Display.

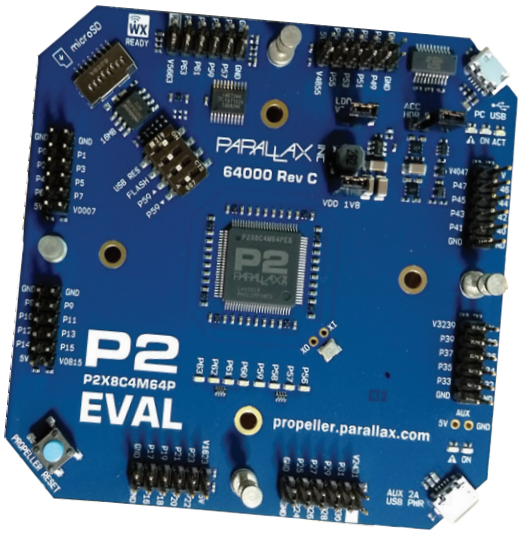
WEBLINKS

[1] **Download aller Programme**: www.elektor.de/using-displays-in-raspberry-pi-projects

[2] **PIL-Bibliothek**: http://effbot.org/imagingbook/imagedraw.htm

Parallax Propeller 2

Teil 4: Senden von Strings



Von **Mathias Claußen** (Elektor)

Unsere Serie über den Parallax Propeller 2 geht weiter! Diesmal betrachten wir einige Probleme, die beim Senden von Strings mit der SPIN2-Sprache auftreten können.

In diesem Artikel werden wir unsere Möglichkeiten erweitern und Strings so zu versenden, wie Sie es mit `print` auf Arduino-Systemen tun. Mit dieser ersten Aufgabe werden wir in der Lage sein, auf dem klassischen Weg Debug-Informationen zu senden. Unser Ausgangspunkt ist: Der Code ist in der Lage, ein einzelnes Zeichen

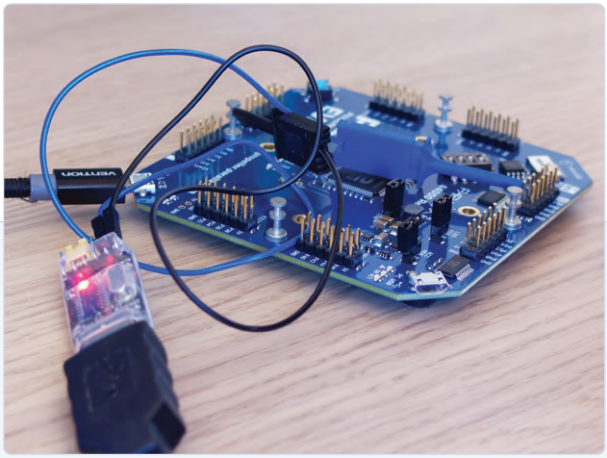


Bild 1. Der Propeller 2 ist über einen seriellen USB-Konverter am Rechner angeschlossen.

Listing 1. Der vollständige Code

```
'We run in the internal RC giving us 25 MHz clockspeed

PUB main()
   pinwrite(56, 1 )
   serial_start()
  repeat
     waitms( 250 )
     pinwrite(56, 1 )
     tx("0")
     waitms( 250 )
     pinwrite(56, 0 )
     tx("1")

 PUB serial_start()
     WRPIN( 57, %01_11110_0  )             'set async tx mode for txpin
     WXPIN( 57, ((217<<16) + (8-1 )) )     'set baud rate to sysclock/115200 and word size to 8 bits
   org
     dirh    #57
   end

PUB tx(val)
    WYPIN(57,val) 'load output word
   org
    WAITX   #1             'wait 2+1 clocks before polling busy
     wait
     RDPIN   val,#57  WC    'get busy flag into C
     IF_C   JMP   #wait     'loop until C = 0
  end
```

Listing 2. Der Text, Zeichen für Zeichen

```
'We run in the internal RC giving us 25MHz clockspeed
PUB main()
  pinwrite(56, 1 )
  serial_start()
  repeat
     waitms( 250 )
     pinwrite(56, 1 )
     tx("C")
     tx("o")
     tx("d")
     tx("e")
     tx(" ")
     tx("a")
     tx("l")
     tx("i")
     tx("v")
     tx("e")
     waitms( 250 )
     pinwrite(56, 0 )
     tx("1")

 PUB serial_start()
   WRPIN( 57, %01_11110_0  )               'set async tx mode for txpin
   WXPIN( 57, ((217<<16) + (8-1 )) )       'set baud rate to sysclock/115200 and word size to 8 bits

  org
     dirh    #57
   end

PUB tx(val)
       WYPIN(57,val) 'load output word
   org
     WAITX   #1             'wait 2+1 clocks before polling busy
      wait
      RDPIN   val,#57  WC    'get busy flag into C
      IF_C   JMP   #wait     'loop until C = 0
  end
```

mit 115.200 Baud, 8 Datenbits, keiner Parität und einem Stoppbit zu senden. Der komplette Code in **Listing 1** kann von [1] heruntergeladen werden. Der Propeller 2, angeschlossen an einen USB-Seriell-Konverter, ist in **Bild 1** dargestellt.

Wenn wir, wie in **Listing 2** gezeigt, ein „Code alive" senden wollen, führt dies zu einer Menge von Aufrufen unserer `tx()`-Funktion, einen für jedes Zeichen. Immer, wenn Codezeilen sich wiederholen, sollte man anfangen, über eine Generalisierung der Aufgabe nachzudenken. Vielfach kopierte Abschnitte von Codefragmenten im Projekt sind nie eine gute Idee, vor allem, wenn man später etwas hinzufügen oder den Code korrigieren muss. Wir werden deshalb eine Funktion erstellen, die eine Zeichenkette als Argument nimmt, sie in einzelne Zeichen zerlegt und diese nacheinander an unsere `tx()`-Funktion sendet. Das klingt einfach, und wenn Sie etwas Programmiererfahrung haben, sollte es das auch sein. Wenn Sie SPIN2 oder SPIN aber zum ersten Mal benutzen, müssen Sie sicher etwas lesen und herumtüfteln.

Übergabe von Zeichenketten

Da es sich bei Strings nur um eine fortlaufende Speicherstelle handelt, die alle Zeichen hintereinander enthält und mit einer binären Null (\0) abgeschlossen wird, können wir versuchen, einfach die Startadresse dieses Speichers an eine Funktion zu übergeben. Wenn wir C/C++ verwenden, erfüllt der `&`-Operator in Verbindung mit dem ersten Element den Zweck. In SPIN2 wird dies in der Syntax etwas anders gehandhabt, aber es funktioniert weitgehend gleich. Anders ist nur die Art und Weise, wie wir die Funktion und das Argument selbst deklarieren. Der Typ der Variablen, die an unsere Funktion übergeben wird, ist nicht angegeben, so dass SPIN2 annimmt, dass es sich um einen Long-Typ (32 Bit) handelt. Diese automatische Annahme für Variablen, die an eine Funktion übergeben werden, kann in SPIN2 manchmal praktisch sein, aber auch einige unerwünschte Nebeneffekte mit sich bringen. In unserem Fall ist es die Speicheradresse für

unseren String und dürfte für unseren Zweck gut funktionieren. Die Art und Weise der Deklaration lokaler Variablen ist allerdings verschieden. Man kann deutlich sehen, dass alle Variablen (hier `c`) im Funktionskopf deklariert werden müssen. Dies geschieht mit einem `|` nach der schließenden Klammer in der Zeile. Wenn wir mehr als eine Variable benötigen, setzen wir sie in diese Zeile und trennen sie mit `,`. **Listing 3** zeigt, wie unser Funktionskopf aussieht.

Innerhalb der Funktion (siehe **Listing 4**) müssen wir die Daten byteweise einlesen. Wir haben die Anweisung `c := byte[s++]`, die ein paar Worte der Erklärung bedarf. Für unsere Variable `c` ist momentan keinen Typ definiert, oder besser gesagt, es ist der Standardtyp. Ohne `byte[]` würden wir 32 Bits (Variable `uint32_t`) auf einmal lesen, mit `byte[s++]` dagegen lesen wir ein Byte an der Speicheradresse `s`, an der der String beginnt. `s++` in Verbindung mit `byte[]` stellt sicher, dass die Adresse jeweils nur um ein Byte inkrementiert wird. Wenn wir nicht so vorgehen, würden alle Operationen 32 Bit breit sein.

Da nun sichergestellt ist, dass wir byteweise lesen, schauen wir uns die `repeat while`-Anweisung an. Solange `c` ungleich (`<>`) zu binär Null ist, wird alles innerhalb der `repeat while`-Anweisung in einer Schleife durchlaufen. Innerhalb der Schleife, die in **Listing 5** zu sehen ist, haben wir unsere `tx()`-Funktion, die ein Byte sendet, und eine weitere Zeile, die das nächste Byte abgreift und in unsere Variable `c` schreibt. Danach geht die Schleife weiter und prüft, ob `c` immer noch ungleich binär Null ist.

Damit haben wir nun unser `prints()` realisiert und sind in der Lage, Strings über UART an einen angeschlossenen PC oder Logikanalysator zu senden. In der nächsten Folge lesen wir die Zustände von I/O-Pins und geben sie über den UART aus. Wie schwer kann es sein, den Status eines I/O-Pins auslesen? Nun, in der Theorie ist es sehr einfach, aber manchmal sind es die netten Extras, die den Unterschied ausmachen und zeigen, wie viele Möglichkeiten in den Smartpins stecken.

200479-D-02

Lisiting 3. Der Funktionskopf

```
PUB prints( s ) | c
```

Lisintg 4. Die prints-Funktion

```
PUB prints( s ) | c
   c := byte[s++]
   REPEAT WHILE (c <> 0)
     tx(c)
     c := byte[s++]
```

Listing 5. REPEAT WHILE

```
  REPEAT WHILE (c <> 0)
     tx(c)
     c := byte[s++]
```

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

Ein Beitrag von
Entwurf und Text: **Mathias Claußen**
Redaktion: **Jens Nickel** und **C. J. Abate**
Übersetzung: **Rolf Gerstendorf**
Layout: **Giel Dols**

WEBLINK

[1] **Artikel-Webseite**: www.elektormagazine.de/200479-D-02

50 Jahre Elektor: September-Renaissance

Der Plan, der Plan, ein leerer Wahn …

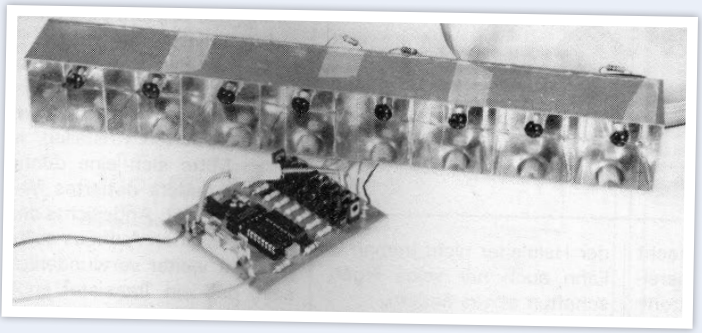
Von **Rolf Gerstendorf** (Elektor)

Nach der Sommer-Doppelausgabe (alias Halbleiterheft) und dem anschließenden wohlverdienten Sommerurlaub war das Septemberheft von Elektor stets ein Neuanfang - für größere Bauprojekte und praxisnahe Hintergrundartikel, aber immer aufgelockert durch diese oder jene kleine Schaltung. Der Autor hat hier einige bemerkenswerte September-Artikel herausgesucht, die ihm in 35 Jahren als Elektorianer besonders in Erinnerung geblieben sind. Eine höchst subjektive Auswahl - aber lesen Sie selbst!

Der Plan, der Plan, ein leerer Wahn - so bedichtete einst der DDR-Volksmund die Planvorgaben der SED-Parteiführung. Elektor ist sicher weit von realsozialistischer Planwirtschaft entfernt, aber eine Planung für die nächsten Elektor-Ausgaben gab es schon immer. Heutzutage ist das Räderwerk in unserem multinationalen Verlag so weit verzweigt und ein Rädchen greift so eng ins andere, dass eine Planung (von kleinen Anpassungen abgesehen) auch eingehalten werden muss.
Früher war das anders. Die Vorschau auf das nächste Heft glich eher einer Wunschliste, und was dann tatsächlich eine Ausgabe später veröffentlicht werden konnte, hing von vielen Unwägbarkeiten ab. Wird der Autor mit der Schaltung fertig? Tauchen bei der Überprüfung im Labor irgendwelche unüberwindbaren Hindernisse auf? Ist irgendwelches unverzichtbares Personal krank oder mit anderen Dingen beschäftigt oder im Urlaub?
Solche Unwägbarkeiten erreichten in der sommerlichen Urlaubszeit stets ihren Höhepunkt. Als ich nach dem Halbleiterheft und meinem Sommerurlaub in den Bergen, auf Hochtouren braungebrannt und von Kletterpartien gestählt, wieder in der Redaktion erschien, lag die Septemberausgabe meist frisch gedruckt auf meinem Schreibtisch. Mit für mich manchmal ziemlich überraschenden Inhalten!

Knight-Rider (1989)

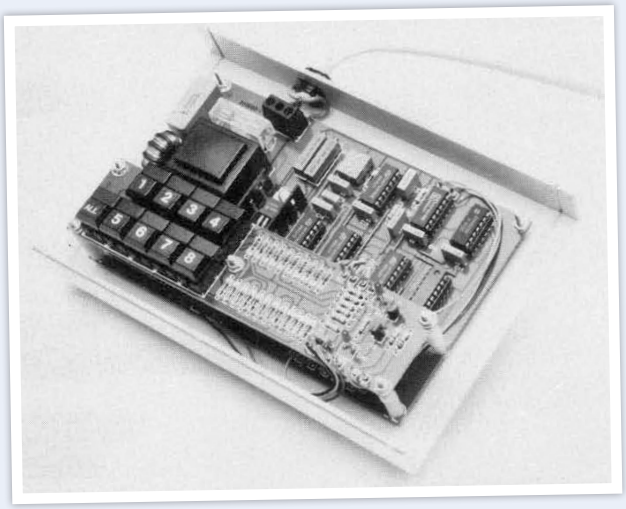
Zu dieser Zeit war K.I.T.T., das in doppelter Hinsicht smarte Wundermobil aus der US-Fernsehserie Knight-Rider, in Deutschland noch völlig unbekannt. Das Fahrzeug lernt seine Umgebung durch einen 8-bit-Scanner am Kühlergrill kennen, dessen Betriebsweise durch den Elektor-Artikel vollständig der Weltöffentlichkeit enthüllt wurde: Ein Hin-und-her-Lauflicht mit einem Oszillator und einem Schieberegister im CMOS-Stil! Schaut man sich die heutigen Kreationen der Lichtdesigner in der Autoindustrie an, so könnte man an eine Renaissance des *Knight Industries Two Thousand* glauben. Allerdings werden solche Lauflichter nicht mehr von CMOS-Zählern, sondern von Mikrocontrollern über einen CAN-Bus gesteuert, was die Sache nicht besser macht. Der Weg zu einem K.I.T.T. scheint aber noch genauso weit wie damals.



www.elektormagazine.de/magazine/elektor-198909/48941

Lichtnetz-Fernsteuerung (1991)

Im Umfeld des Elektor-Verlags gab es damals eine innovative, aber leider schlecht geführte Firma namens ELSA, die hauptsächlich höchstwertige Modems entwickelte, sich aber nebenbei mit Lichtnetz-Kommunikation beschäftigte. Das ließ das Elektor-Labor natürlich nicht auf sich sitzen und entwickelte eine nachbaufähige Schaltung zum Schalten und Dimmen von Lampen über das Lichtnetz, natürlich nur mit Durchsteckbauteilen und ohne Mikrocontroller.
Unglaublich! Ich habe den Artikel nach 30 Jahren zum ersten Mal wieder aufgeblättert und mein erster Gedanke war: Wie gut, dass es heutzutage Mikrocontroller gibt! Sender und Empfänger sind ja reine CMOS-Gräber! Im Sender gab es eine 9-Taster-Matrix, aus der mit Standard-CMOS-ICs ein 8-bit-Code erzeugt wird, der aus einem Startbit, drei Adressbits, drei Helligkeitsbits und einem Steuerbit besteht. Dieser serielle Code tastet mit 40 Hz einen 200-kHz-Träger, der dann über eine selbstgewickelte Entstörspule in das Lichtnetz eingekopppelt wird. Die Sache mit der Spule ist gnädigerweise verjährt!
Der Empfänger dekodiert diese (analog verstärkte) Information und vergleicht die empfangene Adresse mit der im Empfänger voreingestellten Bitkombination. Bei Übereinstimmung übernimmt der Empfänger die Leistungssteuerbits in ein Register und steuert durch Phasenanschnitt die an die angeschlossene Lampe abgegebene Leistung. Alles mit Standard-CMOS-Bausteinen!
Beeindruckend finde ich, dass die Empfängerschaltungen, die in kleinen Steckergehäusen untergebracht sind, auf jeweils drei (!) Platinen verteilt werden mussten, was natürlich für eine drangvolle Enge im Gehäuse sorgte. Ein Wunder, dass dazwischen noch ein Schukostecker passte!



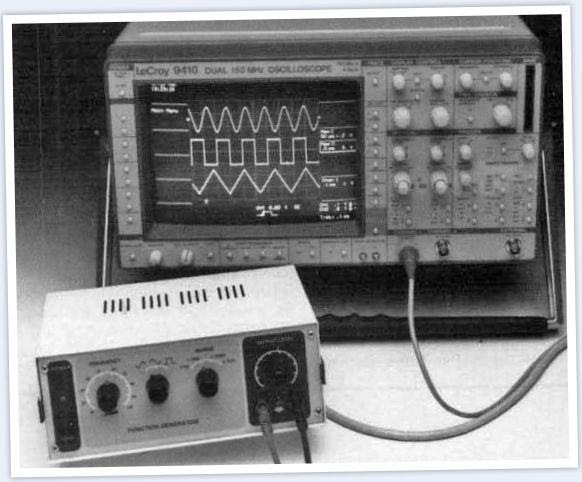
Kleine Anmerkung: Aus der zehn Jahre später bankrotten ELSA AG ist die bekannte Firma devolo hervorgegangen, die heute noch sehr erfolgreich hochwertige PLC-Adapter anbietet (aber ohne CMOS-ICs).

www.elektormagazine.de/magazine/elektor-199109/29689

Einfacher Funktionsgenerator (1992)

Der kleine Funktionsgenerator hat den GHz-Zähler und das Compuboard im gleichen Heft in meiner Gunst ausgestochen, aus persönlichen und nostalgischen Gründen. Das zentrale IC der Schaltung, ein integrierter Funktionsgenerator XR2206 der Firma Exar, war genauso

fragwürdig in seinen Leistungen wie wunderbar bequem einzusetzen. Bei einem Studentenjob habe ich eines dieser ICs einfach aus dem Plastikbeutel auf die Platte des Labortischs geschüttelt, mit dem Resultat, dass der Funktionsgenerator partout nicht funktionieren wollte (vergleiche Elektor 7-8/2021: „ESD – Der unsichtbare Blitz“). Knapp 50 Mark hat das IC damals gekostet (glaube ich), was für einen Studenten eine Menge Geld war. Und wer nicht hören will, muss fühlen!



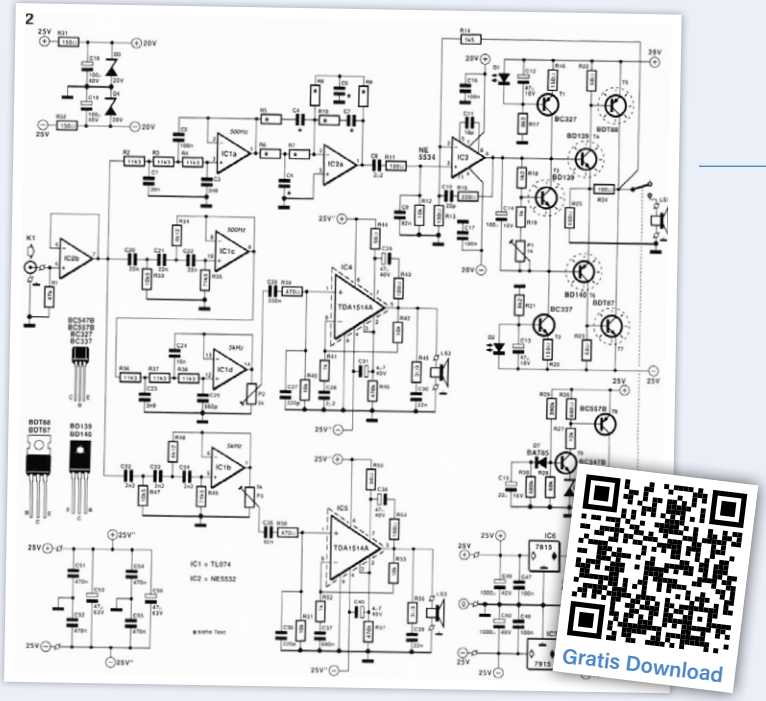
www.elektormagazine.de/magazine/elektor-199209/30004

3-Wege-Aktiveinschub (1993)

Meine erste selbstgebaute Schaltung war in den frühen 1970ern ein Audioverstärker mit einem 2N2955/2N3055-Pärchen (Kosten: ein Monat Taschengeld, zerstört: in einer Millisekunde). Seither habe ich eine Menge passive wie aktive Audioverstärker nachgebaut und selbst entworfen. Fast alle hatten eines gemeinsam: Ich musste sie ob ihrer Größe in klobigen 19“-Gehäusen unterbringen.

Der 3-Wege-Aktiveinschub jedoch versprach die platzsparende Integration der Verstärker samt elektronischer Weiche in eine Lautsprecherbox. Die Linkwitz-Riley-Frequenzweiche (!) konnte nach eigenen Wünschen und den Anforderungen von Gehäuse und Chassis konfiguriert werden. Platzsparend war der Einsatz integrierter Endstufen TDA1514A für Hoch- und Mitteltöner, außerdem gab es eine klassische AB-Gegentaktendstufe für 70 W Basspower.

Ich habe diesen Aktiveinschub ein paar Jahre später nachgebaut und in den Boxen meiner pubertierenden Tochter versenkt, damit sie in allerbester Elektor-Qualität ihre Kelly-Family hören konnte. Ich bin heute der festen Überzeugung, dass dies der Vaterliebe entschieden zu viel war!

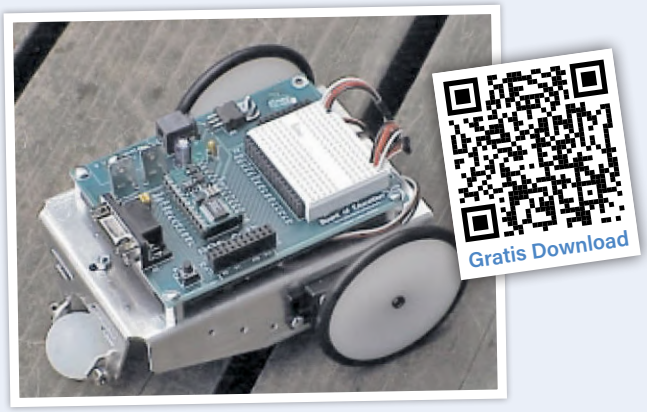


www.elektormagazine.de/magazine/elektor-199309/30299

Programmieren mit der BASIC Stamp 2 (1999)

Ein kleiner und bezahlbarer Einplatinencomputer im DIL-Format, mit einem BASIC-Interpreter im ROM und jeder Menge (kostenloser!) Unterstützung zum Programmieren Lernen; was heute kein außergewöhnliches Konzept mehr ist, mischte in den 1990ern den Hobbymarkt auf. Die BASIC-Briefmarke der Firma Parallax wurde auf eine Trägerplatine (mit Experimentierfeld) aus dem Hause Elektor gesteckt, auf einem einfachen fahrbaren Chassis montiert und fertig war der Roboter, der in den sechs Folgen des Kurses mit Sensoren ausgestattet und programmiert wurde.

Mikrocomputer-Kurse haben also Tradition in Elektor. Aktuell können Sie die modernste Ausführung der BASIC-Stamp in einem Kurs kennenlernen, den Parallax-Propeller.



www.elektormagazine.de/magazine/elektor-199909/31971

Lithium-Ionen-Akkus (2001)

Als gerade die umweltschädlichen NiCd-Akkus von den schwermetallfreien NiMH-Akkus auf dem Massenmarkt abgelöst wurden, tauchte eine gänzlich neue Akkutechnologie in mobilen Geräten auf, nämlich die Lithium-Ionen-Akkus. Zu Beginn der Nullerjahre waren Lithium-Ionen-Akkus noch sehr teuer und der Ladevorgang kritisch. Der Grundlagenartikel streicht nicht nur die Vorteile der Lithium-Ionen-Akkus heraus - hohe Kapazität, kleines Volumen und geringes Gewicht,

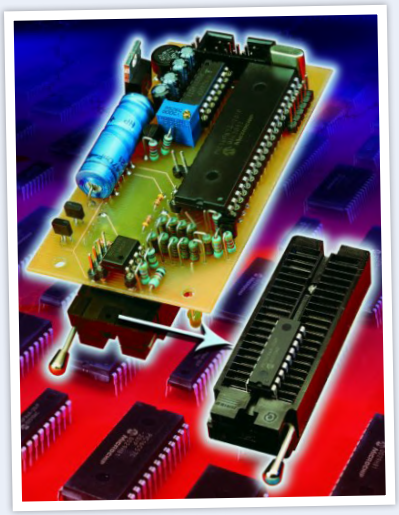
hohe Klemmenspannung – sondern stellt ganz Elektor-like auch einige einsetzbare Ladeschaltungen mit speziellen Lade-ICs vor.

www.elektormagazine.de/magazine/elektor-200109/1269

Universeller PIC-Programmer (2003)

Das größte Hindernis für Otto Normalverbaucher, Mikrocontroller und andere programmierbare ICs mit Software auszustatten, war früher der hohe Preis von Entwicklungsumgebungen, sowohl was Soft- als auch Hardware betraf. Ich weiß nicht, ob die „Open-" Community, zu der sich ja auch Elektor zählt, wirklich dafür verantwortlich ist, aber meiner Meinung haben erst zahlreiche Hacks, freie Dokumentationen und Tutorials im aufstrebenden Internet die Industrie dazu gebracht, Entwicklungswerkzeuge als kostenlose oder niederpreisige Hilfsmittel zur Verfügung zu stellen, um die industrielle Verwendung und den Verkauf ihrer Mikrocontroller anzukurbeln.

Ein tolles Beispiel dafür ist der PIC-Programmierer, der mit seinen vielfältigen Möglichkeiten eine bezahlbare Alternative zu professionellen Programmern darstellte. Der PIC-Prog bestand eigentlich nur aus einem programmierten PIC, einer 40-poligen ZIF-Fassung, einer Spannungsversorgung und einem RS232-Wandler für die Kommunikation mit dem PC. Es war die Software, die die luxuriöse Programmierung von mehr als 85 verschiedenen PIC-Modellen ermöglichte!



www.elektormagazine.de/magazine/elektor-200309/1735

Schwerpunkt RFID (2006)

Neuen Technologien auf den Grund zu gehen und mit praktischen Anwendungen gleich auszuprobieren, das war schon immer eine Spezialität von Elektor. In dieser Schwerpunkt-Ausgabe wurden die Grundbegriffe und die Funktion diverser Arten von RFID-Chips vorgestellt, eine eigene (kostenlose!) Elektor-RFID-Karte aktiviert und mit einem Selbstbau-RFID-Writer/Reader beschrieben und ausgelesen. Für die echten Maker wurde auch eine experimentelle Schaltung vorgestellt, die ohne spezielles RFID-IC, sondern nur mit einem Standard-Mikrocontroller funktionieren sollte.

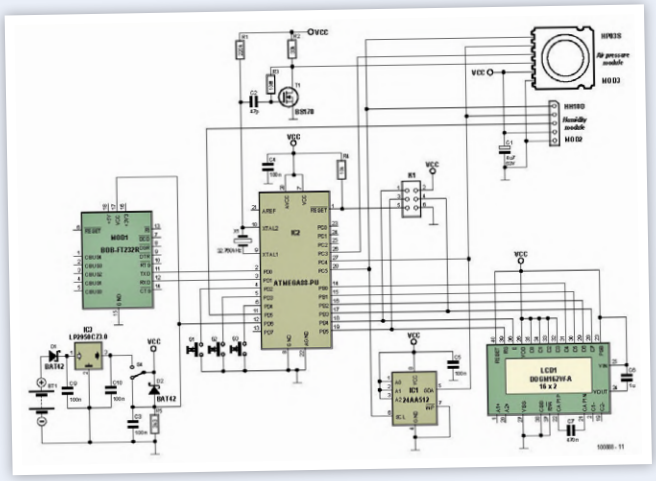
www.elektormagazine.de/magazine/elektor-200609

USB-Wetterlogger und I²C-Sensoren (2011)

Dank stromsparender Mikrocontroller und ausgefeilter Low-Energy-Programmierung wurde der Selbstbau batterieversorgter Langzeit-Anwendungen möglich. Die Hardware schrumpfte dabei (im Vergleich zum Analogzeitalter) auf ein lächerliches Minimum.

Ein schönes Beispiel für diese Entwicklung war der Wetterlogger, der nur aus einem AVR-Controller mit Spannungsversorgung, einem EEPROM, einem Display und zwei I²C-Sensoren für Feuchte und Luftdruck/Temperatur bestand. Die Kommunikation mit der PC-Software zur Darstellung der Messwerte erfolgte bei Bedarf über eine RS232/USB-Brücke.

Apropos I²C-Sensoren: Solche digitalen Umweltsensoren waren zu dieser Zeit nicht mehr gerade brandneu auf dem Markt, wurden aber im Hobby- und Makerbereich eher selten eingesetzt. Das war einen weiteren Artikel in der gleichen Ausgabe wert, der die Funktionsweise und die inneren Werte der Sensoren genau beleuchtete.



www.elektormagazine.de/magazine/elektor-201109/3922
www.elektormagazine.de/magazine/elektor-201109/3923

Swiss Pi (2016)

Als in den frühen 2010er Jahren der Raspberry Pi in der Maker-Szene einschlug wie eine Bombe, schien die gute alte Zeit der Hard- und Softwareexperimente am Atari oder C64 wiedergekommen. Bald jedoch wurde klar, dass der Pi in dieser Hinsicht die eine oder andere Schwäche aufwies (was ihm angesichts seines spottgünstigen Preises verziehen sei): die ungeschützten GPIO-Ports, die nur rudimentäre PWM-Unterstützung, der fehlende Analogeingang und einiges mehr. Glücklicherweise kann man den Pi um eine Hardware-on-Top (HAT) erweitern. Die HAT „Swiss Pi" behebt die Mängel und ergänzt darüber hinaus den Raspberry Pi um einige Extras, zum Beispiel eine batteriegepufferte Echtzeituhr, eine RS485- und eine I²C-Schnittstelle. Die Software (Server und Bibliothek) des Swiss Pi läuft unter Windows und Linux.

Das anerkannte HAT-Modul, das auf alle Raspberry Pi A+/B+/2B/3B/Zero/4B passt, wurde zum Bestseller und ist heute noch im Elektor-Shop erhältlich. Fünf Jahre in unseren schnelllebigen Mikrocomputerzeiten – das ist ein stolzes Alter!

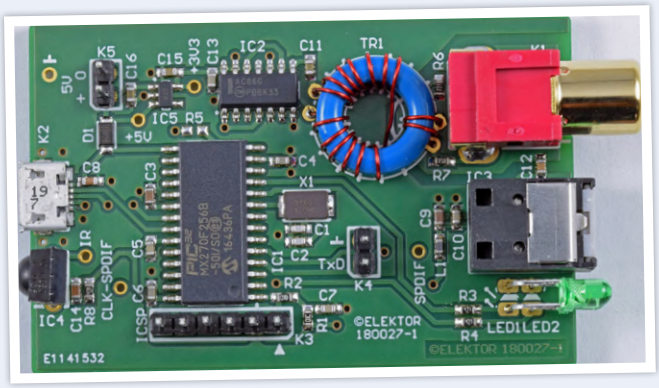
www.elektormagazine.de/magazine/elektor-201609/39757

USB-S/PDIF-Schnittstelle (2020)

Im Internet gibt es viele Lösungen dafür, einen Computer mit einer S/PDIF-Schnittstelle auszustatten, aber ob billig oder teuer, aus China oder in einer europäischen High-End-Schmiede zusammengelötet, alle Lösungen haben eines gemeinsam: Man weiß nicht, was drin steckt und wofür man da eigentlich sein Geld ausgibt. Elektor und seine Leser entwickeln dagegen gern Schaltungen, die nicht nur „state of the art" und niedrige Kosten miteinander verbinden, sondern auch detailliert offenlegen, wie diese Schaltungen funktionieren.

Diese USB-S/PDIF-Schnittstelle ist ein geradezu klassisches Beispiel dafür: Der zentrale Controller bietet ausreichende Peripherie, um USB-Audiosignale zu dekodieren, als S/PDIF-Frames zu speichern und synchron über SPI auszugeben. Das Signal steht dann an einem Toslink-Verbinder oder isoliert an einer Cinch-Buchse zur weiteren Verfügung bereit. Zu allem Luxus kommt hinzu, dass sich das S/PDIF-Interface über eine IR-Fernbedienung steuern lässt.

Die Schaltung habe ich zu Coronazeiten wohl ein halbes Dutzend Mal aufgebaut, um die Mediabox-Raspberry-Pis in meinem Wohnzimmer, im Wohnmobil und in meinem Bekanntenkreis mit einem „highendigen" Audio-Anschluss auszustatten!



www.elektormagazine.de/magazine/elektor-154/58919

210343-02

Ein Beitrag von
Autor und Redaktion: **Rolf Gerstendorf**
Layout: **Harmen Heida**

Zutritt für Unbefugte ~~verboten~~ erwünscht

In Friesland, wo die Röhren blühen …

Von **Menno van der Veen** (Niederlande) und **Eric Bogers** (Elektor)

Menno van der Veen ist in Audio-Enthusiasten-Kreisen eine bekannte Größe. Er entwickelt nicht nur High-End-Röhrenverstärker, sondern auch spezielle Ringkerntransformatoren. Außerdem ist er ein begnadeter Lehrer, der sein Wissen über alles, was mit Röhren zu tun hat, bereitwillig mit den Alumni seiner TubeSociety teilt.

Bild 1. Nach bescheidenem Start…

Bild 2. …wurde daraus ein wirklich geräumiges Labor.

Wenn es um High-End-Audio geht, ist die Elektronik-Gemeinde in zwei schier unversöhnliche Lager gespalten: Auf der einen Seite stehen die Anhänger von Röhrenverstärkern, auf der anderen die Fans des Transistors. Und nicht selten werden zwischen den Anhängern beider Lager hitzige Diskussion – manche sprechen auch von Glaubenskriegen – geführt. Und das ist eigentlich sehr schade, denn beide Technologien haben ihre spezifischen Eigenschaften mit Vor- und Nachteilen. Denn am Ende geht es doch nur darum, was das größte Hörvergnügen bereitet. Menno van der Veen gehört seit seinen Teenagerjahren zwar definitiv zum Röhrenlager, aber ganz sicher nicht zu den „unversöhnlichen Glaubenskriegern".

„Ich habe mit dem Bau von Röhrenverstärkern begonnen, als ich etwa vierzehn Jahre alt war, und um ehrlich zu sein, mit dem Kopieren fremder Entwürfe. Mein Wissen über die Materie reichte damals bei weitem nicht aus, um eigene Verstärker zu entwerfen. Erst nach meinem Studium der angewandten Physik an der Universität Groningen begann ich zu verstehen, worum es bei Röhren geht, und ich konnte mit eigenen Entwürfen und Entwicklungen beginnen.
1983 veröffentlichte ich meinen ersten Entwurf mit vier EL84 in einer Push-Pull-Konfiguration und mit einem Ringkerntransformator am Ausgang. Vor allem der Ringkerntrafo sorgte für viel Gesprächsstoff. 1987 baute und veröffentlichte ich einen 100-W-Verstärker, wieder mit Ringkerntransformatoren, mit dem Ergebnis, dass so ziemlich alle Audio-Enthusiasten in den Niederlanden mich mit Kritik überschütteten (heute würde man Shitstorm sagen): ‚Das kann nicht gut sein, weil jedermann EI-Kerne für Ausgangsübertrager verwendet'. Grund genug für mich, den eingeschlagenen Weg weiter zu verfolgen: Ich habe mich eingehend mit Übertragern beschäftigt (und auch darüber publiziert), außerdem habe ich angefangen, Bücher über Röhren und Übertrager zu schreiben.
Wie die meisten Elektronikingenieure begann ich meine Karriere in einem bescheidenen Labor im Keller (**Bild 1**). Später war mein Labor in einem monumentalen Gebäude in der Innenstadt von Zwolle, doch seit 2014 lebe und arbeite ich auf dem friesischen Land, wo ich ein neues Labor gebaut habe (**Bild 2**). Dieses Labor ist so groß, weil es auch als Unterrichtsraum für meine TubeSociety dient. Wenn ich hier alleine arbeite, benutze ich hauptsächlich die Werkbank, die im Hintergrund zu sehen ist.
Anders als Sie vielleicht erwarten, ist meine Sammlung von Laborge-

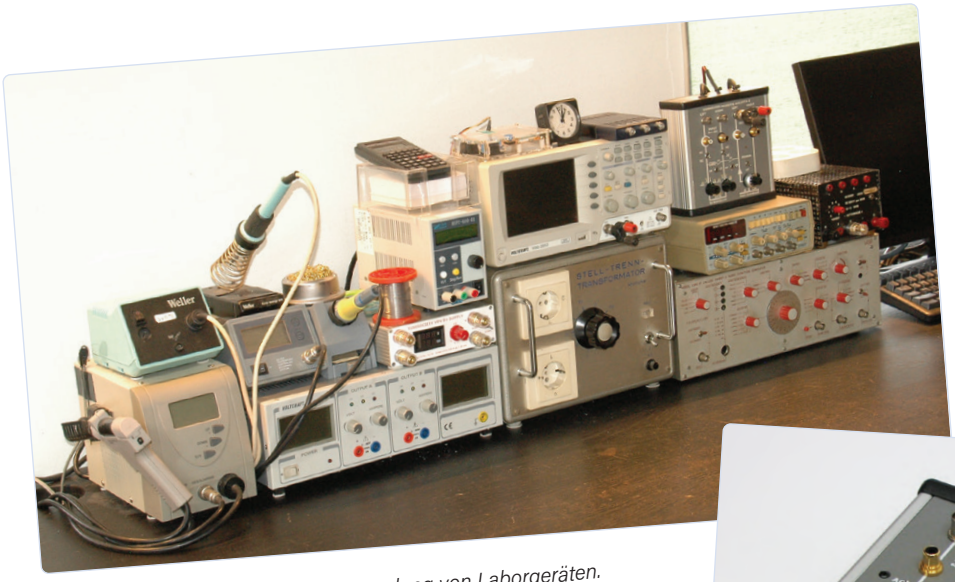


Bild 3. Eine wenig umfangreiche Sammlung von Laborgeräten.

Bild 4. Die ARTA-2 Messeinheit.

Bild 5. Ein Vanderveen-Ringkerntransformator.

räten recht bescheiden (**Bild 3**): Lötstation, Netzgeräte, Oszilloskope, Oszillatoren und ein ARTA-2-Messgerät (**Bild 4**). Letzteres ist meine eigene Konstruktion, die mit Hilfe von Tentlabs vermarktet wurde. Sie ermöglicht es, kalibrierte Messungen mit einer (guten) Soundkarte durchzuführen. Mit der Messeinheit kann ich ‚tief' bis zu -140 dBV (0,1 μV_{rms}) sehen und kleinste Unregelmäßigkeiten in einem Signal sichtbar machen. Schaltplan und Platinenlayout des ARTA-2 wurden jetzt frei veröffentlicht, so dass jeder Interessierte ein eigenes Gerät bauen kann [1].

Im Moment bin ich eigentlich an vier Fronten beschäftigt. Ich entwerfe Ringkerntransformatoren für Röhrenverstärker, die ursprünglich von Plitron und Amplimo, jetzt aber von Trafco hergestellt werden (**Bild 5**). Außerdem entwerfe ich (natürlich) Röhrenverstärker, darunter auch den neuen Trans-SE10. Ich schreibe Artikel und Bücher [2][3][4] und schließlich habe ich die TubeSociety [5] gegründet, um mein Wissen weiterzugeben.

Ich werde oft gefragt, warum ich meine Arbeit veröffentliche und mit meinen TubeSociety-Studenten teile. Darauf gebe ich immer ungefähr die gleiche Antwort. Natürlich könnte ich in meinem Labor neue Dinge entwickeln, diese mit Patenten schützen und sie dann auf den Markt bringen. Dann bekomme ich im besten Fall Rückmeldungen von ein paar Kunden und vielleicht erscheinen ein paar Artikel in (Fach-) Zeitschriften. So aber teile mein Wissen mit begeisterten Studenten, denen ich in ihrer Entwicklung zuschauen kann und von denen ich viel Feedback bekomme, während ich mich auch an ihren Entwürfen und Konstruktionen erfreue - und nebenbei entwickle ich meine eigenen Produkte. Ich denke, dass ich eher Lehrer als Geschäftsmann bin und gebe jedem die Möglichkeit, seinen ‚Audio-Traum' auszuleben."

Trans-SE10

„Vor kurzem habe ich (zusammen mit Tentlabs) einen bemerkenswerten Röhrenverstärker auf den Markt gebracht: den Vander-

veen-Trans-SE10 (**Bild 6**). Dabei handelt es sich um einen 2x10-W-Single-Ended-Röhrenverstärker, bei dem einige ungewöhnliche Techniken eingesetzt wurden. Der Grundgedanke dahinter war, dass ich eine lokale Rückkopplung einsetzen wollte, um die Verzerrungen im Ausgangsübertrager und in der Ausgangsröhre so weit wie möglich zu minimieren. Das Thema Rückkopplung wird in der High-End-Audio-Welt nach wie vor heiß diskutiert. Auch hier lassen sich zwei ‚feindliche' Lager unterscheiden: Die Befürworter, die Rückkopplung als Segen für Audioverstärker betrachten, während die Gegner die Rückkopplung als Katastrophe begreifen, die die kleinen Details im Klangbild praktisch unhörbar macht. Beim SE10 versuche ich, einen Mittelweg zu finden. Ich verwende keine Gesamtrückkopplung vom Ausgang zum Eingang, sondern korrigiere Verstärkerfehler lokal, wobei ich den Ausgangsübertrager explizit aus der Rückkopplungsschleife ausschließe. Auf diese Weise kann ich die Vorteile beider Ansätze vereinen.

Bild 6. Der Vanderveen-Trans-SE10.

Die Ausgangsröhre wird von einer spannungsgesteuerten Stromquelle (VCCS) angesteuert, so dass diese eine 100%ige interne Rückkopplung besitzt. Oder anders ausgedrückt: Die Ausgangsröhre nutzt ihre gesamte Verstärkung, um sich selbst zu korrigieren. Dadurch wird die harmonische Verzerrung der Röhre stark reduziert und der Innenwiderstand wird sehr klein. Dies ist wichtig, weil dann die Verzerrungen im Ausgangsübertrager minimiert werden. Wenn der VCCS richtig eingestellt ist, hat der Verstärker als Ganzes sehr geringe Verzerrungen. Aus diesem Grund wird der Betrieb des VCCS mit einer zusätzlichen lokalen Rückkopplungsschaltung optimiert. Das Endergebnis ist ein Verstärker mit etwa 70 dB lokaler Rückkopplung (was mit einer Gesamtrückkopplung aus Stabilitätsgründen niemals erreicht werden kann). Und das wiederum ergibt ein Klangbild, bei dem die Mikrodetails nicht verschwunden sind und von den Lautsprechern ‚befreit' werden können.

Ist damit das ultimative Endziel all meiner jahrelangen Forschung erreicht? Ich glaube nicht, denn Experimente mit trans-symmetrischen Verstärkern zeigen, dass diese Technik auch bei hohen Leistungen eingesetzt werden kann. Diese Forschung ist noch nicht abgeschlossen - haben Sie also Geduld!"

210184-03

FORTSETZUNG GEFÄLLIG?

Möchten Sie mehr über den Trans-SE-10 oder andere Röhrenverstärker erfahren? Lassen Sie es uns über redaktion@elektor.de wissen. Wenn genügend Interesse besteht, werden wir dem Thema in Zukunft vielleicht einen ausführlichen Artikel widmen.

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

Ein Beitrag von
Text und Fotos: **Menno van der Veen**
Übersetzung: **Rolf Gerstendorf**
Redaktion: **Eric Bogers**
Layout: **Giel Dols**

WEBLINKS

[1] **ARTA-2 (Niederländisch)**: https://bit.ly/38zzIPb
[2] **Menno van der Veen, Moderne High-End-Röhrenverstärker (E-Book), Elektor**: www.elektor.de/moderne-high-end-rohrenverstarker-pdf
[3] **Menno van der Veen, High-End-Röhrenverstärker (E-Book), Elektor**: www.elektor.de/high-end-rohrenverstarker-pdf
[4] **Menno van der Veen, Der Entwurf von Röhrenverstärkern (E-Book), Elektor**: www.elektor.de/der-entwurf-von-rohrenverstaerkern-pdf
[5] **TubeSociety**: https://bit.ly/3l9fotf

Hybride Schaltungen

Bemerkenswerte Bauteile

Von David Ashton (Australien)

Manchmal gibt es auf dem Markt einfach kein IC, das eine gewünschte Aufgabe erfüllt. Für solche Fälle werden kleine Schaltungen konstruiert, die wie ein Standardbauteil in einer breite Palette von Produkten eingesetzt werden können. Die mit ICs und regulären diskreten Bauteilen ausgestatteten Hybride sind sowohl faszinierend zu erforschen als auch attraktiv anzuschauen.

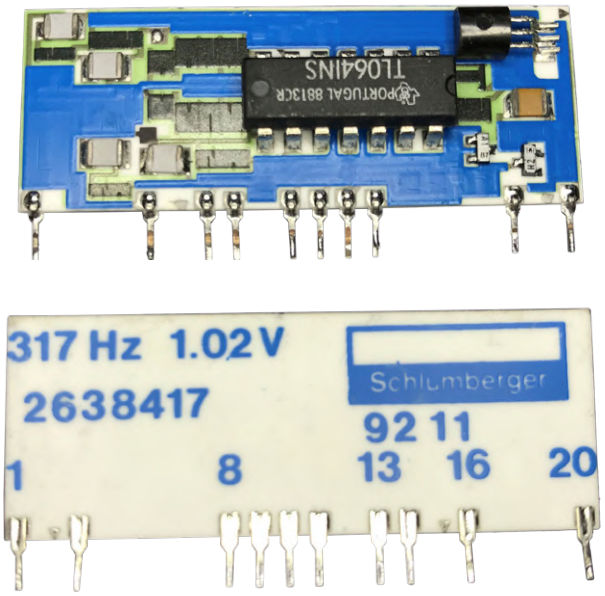


Bild 1. Dieser Hybrid dekodiert Steuerbefehle, die der Netzversorgung überlagert sind.

Was ist eine Hybride Schaltung? Im Grunde genommen ist eine Hybride Schaltung eine kleine Platine, die auf einer größeren (Mutter-) Platine montiert wird, um eine bestimmte Funktion zu erfüllen, die für die Anwendung benötigt wird. Hybride werden in der Regel auf einem Keramiksubstrat mit gedruckten Leiterbahnen und in der Regel mit Widerständen, oberflächenmontierten Kondensatoren und aktiven Bauteilen wie Transistoren, Dioden, integrierten Schaltungen und so weiter aufgebaut. Tatsächlich werden sie sogar manchmal als Hybride Integrierte Schaltungen (HIC) bezeichnet, aber die meisten Leute nennen sie kurz und prägnant einfach Hybride. Sie können aufgrund ihrer hohen Bauteildichte viel Platz in einer Anwendung sparen. Und weil sie konventionelle diskrete Bauteile verwenden können, sind sie vielseitiger als echte integrierte Schaltungen. Eine kleine Platine dagegen, die so konstruiert ist, dass sie auf einer Mutterplatine angebracht wird, würde man wohl nur als Tochterboard bezeichnen und nicht als Hybrid. Aber das ist eine Grauzone.

Ein Hybrid beruht nicht auf einer Platine, sondern auf einem leeren rechteckigen Keramiksubstrat, das bis zu einigen Quadratzentimetern groß sein kann. Darauf werden Leiterbahnen und Widerstände in Dickschichttechnik aufgedruckt. Bei Bedarf werden die Widerstände in automatischen Prozessen per Laser abgeglichen, was zu sehr präzisen Werten führt. Kondensatoren und aktive Bauelemente, meist in SMD-Bauweise, werden dann zusammen mit den für den Anschluss an die Hauptplatine erforderlichen Pins aufgelötet. Abschließend kann die Assemblage zum Schutz vergossen oder beschichtet werden.

Der in **Bild 1** gezeigte Hybrid ist ungewöhnlich, da herkömmliche bedrahtete Bauteile für die Verwendung als SMDs angepasst wurden. Es handelt sich dabei um die Filterplatine eines Frequenzinjektionsrelais. Es nimmt Befehle auf, die als Impulse mit einer bestimmten Frequenz kodiert sind und der Netzspannung überlagert werden. Diese werden zum Ein- und Ausschalten von Straßenbeleuchtungen oder Warmwasseranlagen verwendet. Das IC ist ein Vierfach-Operationsverstärker TL064 im DIP-Format mit teilweise abgeschnittenen Beinchen, das TO-92-Bauteil ist ein Spannungsregler 79L05. Die gedruckten Widerstände sind lasergeschnitten: Die Laserschnitte sind auf den Bauteilen gleich links neben dem TL064 deutlich sichtbar. Durch das Ändern der Bauteilwerte während der Fertigung kann das Filter auf unterschiedliche Frequenzen

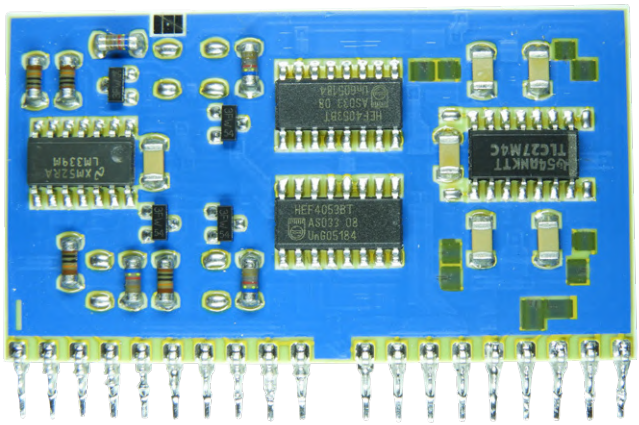


Bild 2. Durch die Kombination von SMD-ICs mit diskreten und gedruckten Dickschichtwiderständen fungiert dieser Baustein als analoger Multiplexer.

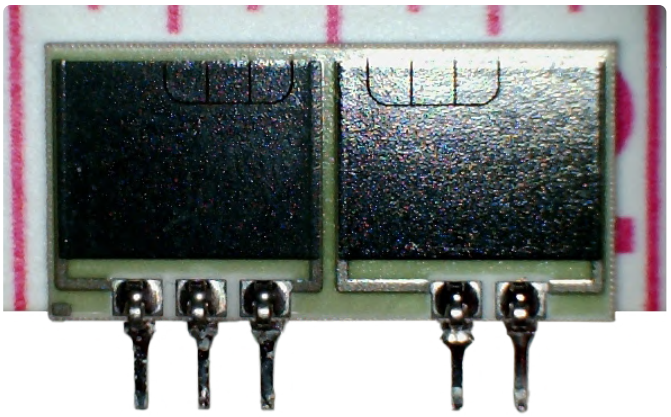


Bild 3. Kaum als Hybrid zu bezeichnen, besteht dieses Telekommunikationsgerät nur aus zwei Präzisionswiderständen.

abgestimmt werden, in diesem Fall, wie auf der Rückseite zu sehen, auf 317 Hz.

Bild 2 zeigt einen Hybrid, der in einem analogen Eingangsmodul einer SPS verwendet wird. Die ICs sind ein analoger Multiplexer 4053, ein Vierfach-Operationsverstärker TLC27M4 und ein Vierfach-Komparator LM339. Der Baustein leitet ein vom System definiertes Signal an einen ADC auf der Hauptplatine weiter. In diesem Beispiel werden SMD-ICs und Transistoren verwendet, wobei diskrete Norm-Widerstände in Rundbauweise neben gedruckten, lasergeschnittenen Widerständen eingesetzt werden.

Unser drittes Beispiel in **Bild 3** kann gerade noch als Hybrid bezeichnet werden. Diese einfache Schaltung besteht nur aus zwei 600-Ω-Präzisionswiderständen aus einem Telekommunikationsgerät. Beim Fotografieren dieses Bauteils musste die Beleuchtung sorgfältig arrangiert werden, damit die Lasertrimmung gut zu erkennen ist. Im Hintergrund ist ein 1-Zoll-Abschnitt eines Maßbandes zu sehen, damit Sie die Größenverhältnisse besser einschätzen können.

Hybride finden sich auch in Gehäusen, wie man in **Bild 4** sehen kann. Diese hybride Schaltung von Hitachi ist ein Treiber, der in einer Klimaanlage verwendet wird. Schon lange stellen Firmen wie Sanken hybride Audioverstärker und Stromversorgungsschaltungen in einem ähnlichen Format her.

Um physikalischen und chemischen Belastungen zu widerstehen, werden Hybride oft beschichtet, wie die in **Bild 5** gezeigte Schaltung zeigt. Das macht sie zwar sehr robust, gibt aber dem Neugierigen (wie mir!) leider keine Informationen darüber, welche Bauteile sich auf dem Hybrid befinden oder welchen Zweck er erfüllt.

Leider ist es nicht einfach, Informationen über Hybride zu erhalten, da sie für jeden Hersteller, jedes Gerät und jede Systemfunktion einzigartig sind. Manchmal kann man die Schaltung „rückwärts" entwickeln, was eine informative Lernerfahrung sein kann. Es ist jedoch unwahrscheinlich, dass Sie als Otto-Normalbastler in der Lage sind, sie zu verwenden - sie sind zu spezifisch in ihrem Design, um von allgemeinem Nutzen zu sein. Trotzdem sind sie oft sehr dekorativ und auf ihre Art angenehm anzuschauen!

210358-02

Haben Sie Fragen oder Kommentare?

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

Bild 4. Durch den Vollverguss ist die Funktionsweise dieses Treibers für Klimageräte nicht einfach zu ermitteln.

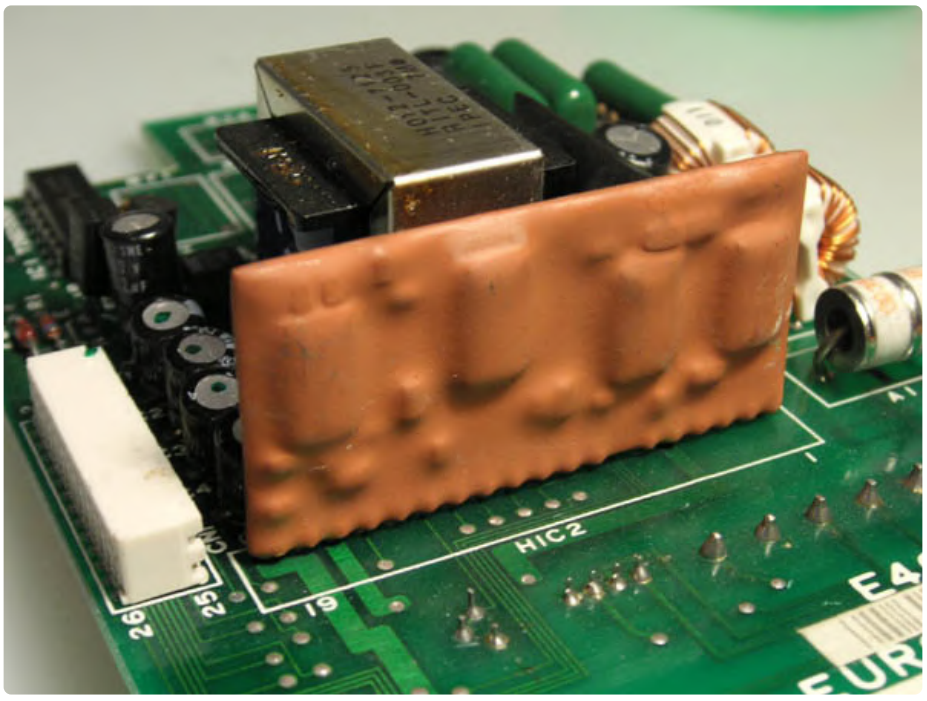
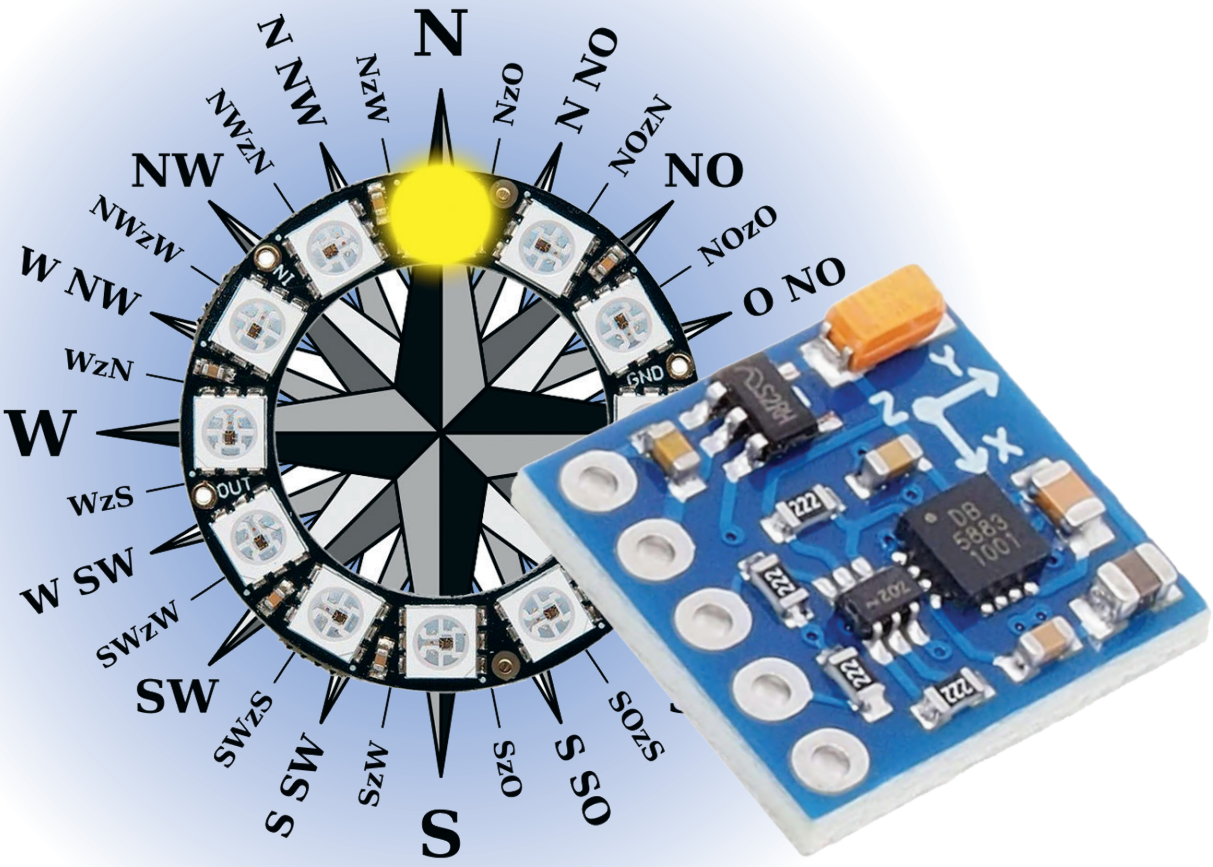


Bild 5. Beschichtungen schützen vor physikalischen und chemischen Schäden, aber auch vor den neugierigen Blicken der Bastler.
(Quelle: Janke, https://de.wikipedia.org/wiki/Dickschicht-Hybridtechnik)

Kompassrose
mit dem GY-271

Oder, warum man mit dem Handy Achten gehen muss...



Die schöne Kompassrose ist ein Public-Domain-Werk von Soeren Gasch (https://commons.wikimedia.org/wiki/File:Compass_rose_german.svg)

Von **Rolf Hase**

Ein Magnetfeldsensor auf einem Breakout-Board, ein Neopixel-LED-Ring als Windrose und ein Arduino, dessen vorgefertigtes Programm die Angelegenheit steuert: Was kann einfacher sein, als auf diese Art einen elektronischen Kompass zu bauen? Doch wenn der Kompass nicht so will, wie man es sich vorstellt, dann hilft nur „Grundlagenforschung“, um einen verblüffenden Fehler aufzudecken und eine nicht minder verblüffend einfache Lösung zu finden!

Früher war, wenn man mit dem Fahrrad durch eine fremde Gegend radelte, die Kartentasche unumgänglich. Heute hat man ein Handy in der Halterung am Lenker oder, wenn man noch mehr Überblick haben will, ein Tablet im Fahrradkorb. Ich verwende auf meinem Mobilgerät dazu die OsmAnd-App für OpenStreetmap-Karten. Leider hat mein Handy keinen Kompass, so dass man auf den winzigen dynamischen Kompass auf dem OsmAnd-Screen angewiesen ist, der zudem wegfällt, wenn man stehen bleibt,

um sich neu zu orientieren. Dazu dreht sich gefühlt dreimal die Karte auf dem Bildschirm, so dass man erst einmal einige 20 m (natürlich in die falsche Richtung) fahren muss, um die Orientierung wiederzuerlangen.

So kam mir die Idee, einen separaten übersichtlichen Kompass mitzuführen, bei dem der Zeiger aka Kompassnadel nicht bei jeder Erschütterung zu tanzen beginnt. Bei der Sichtung älterer Elektor-Hefte verdichtete sich die Idee zu einer elektronischen Lösung, die nun, mindestens eine Elektronik-Generation später, eigentlich noch zuverlässiger und empfindlicher sein müsste.

Bei [1] wurde eine Lösung mit dem Arduino Nano, einem Neopixel-LED-Ring und dem GY-271 vorgestellt, mit Schaltung und Programmcode, was offenbar alles funktionieren sollte und mich zum Kaufen animierte. Da ich gern immer mehrere ICs einer Sorte zur Hand habe, wurden für wenig Geld gleich drei Exemplare des GY-271 im Internet erstanden.

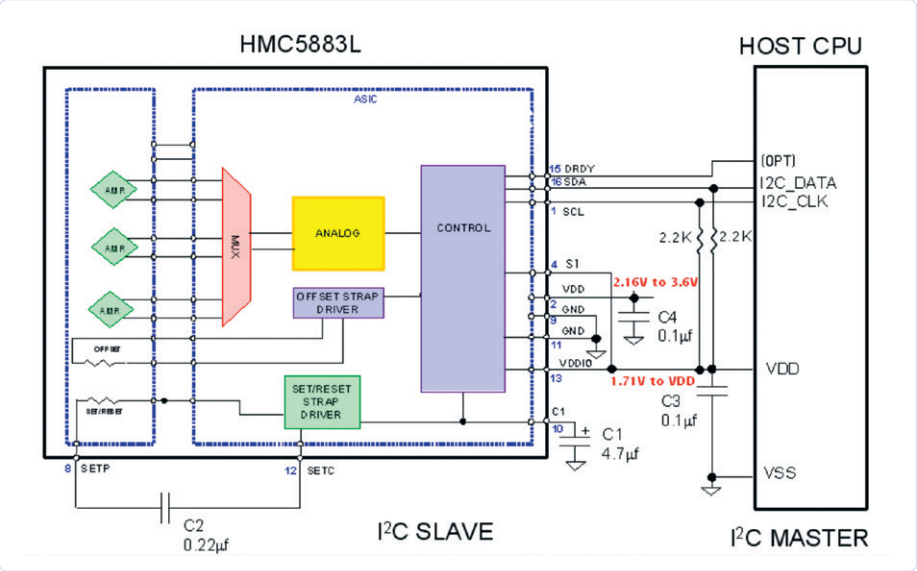


Bild 1. Innenschaltung des Kompasssensors HMC5883L (Quelle: Datenblatt Honeywell).

Das Kompassmodul GY-271

Der GY-271 ist ein für den Arduino geeignetes Breakout-Board, dessen Steckerleiste wunderbar auf ein Steckbrett passt. Das eigentliche IC auf der Platine ist ein 3-Achsen-Kompass-IC mit der Bezeichnung HMC5883L/QMC5883, das offenbar auch in Handys verbaut wird, und daher ein preisgünstiges Massenprodukt ist. Die Innenschaltung des ICs ist in **Bild 1** zu sehen. Außer diesem IC sind lediglich ein 3,3-V-LDO-Spannungsregler, einige Puffer- und Bypasskondensatoren sowie zwei Pegelwandler für den I²C-Bus auf der Platine. Daraus können wir folgern, dass das Modul mit 3...5 V versorgt wird und mit einem angeschlossenen Controller (sei es ein Arduino oder ein Raspberry Pi oder sonst ein „intelligenter" Baustein) über den I²C-Bus kommuniziert. Wie diese Kommunikation vonstatten geht, wird in der beiliegenden detaillierten Dokumentation und in einem kostenlosen, in mehreren Sprachen verfügbaren Anwendungs-E-Book [2] genauestens beschrieben.

Die Verdrahtung des Moduls beschränkt sich auf die – da wir es nirgendwo mit Analogsignalen zu tun haben - unproblematische Stromversorgung. Der LED-Ring wird über die beiden Lötpads für Plus und Masse versorgt. Es gibt nur ein Logik-Signal, das den Takt für das Neopixel bereitstellt. Ansonsten wurde, um Platz zu sparen, ein Arduino Mini Pro verwendet, der nur in der Arduino-IDE entsprechend eingestellt werden muss. Zum Programmieren ist ein FTDI-USB-Seriell-Wandler erforderlich.

So weit, so schlecht

Eigentlich sprach dies alles für einen problemlosen Aufbau und Einsatz des Kompasses an meinem Bike. Doch schon der erste Einsatz war ernüchternd! Es funktionierte eigentlich nur ein Quadrant, dann gab es es einen Sprung der „Nadel" um 180 Grad. Eine Durchforstung des (mittlerweile zurückgezogenen) Programmlistings auf mögliche Ursachen verlief ergebnislos, ein Fehler war nicht zu finden. Eine Internet-Recherche förderte weitere unzufriedene Stimmen zu Tage, aber der Fehler wurde auf die nicht vorhandene Qualität von Noname-Produkten bei ebay geschoben.

Kurz und gut, als Praktiker musste ich empirisch vorgehen, um dem Fehler auf den Grund zu gehen. Schon in ersten Versuchen konnte ich feststellen, dass eine starke Abhängigkeit von Fremdfeldern gegeben ist, die zum Beispiel in der Nähe meines Laptops auftreten. Von einem allzu kippeligen Versuchsaufbau ist

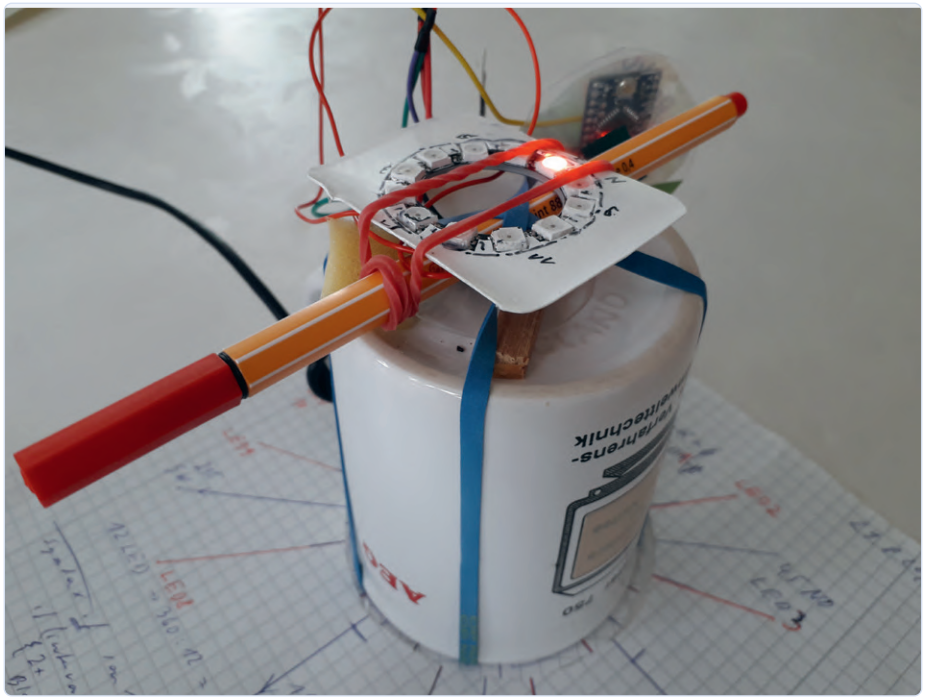
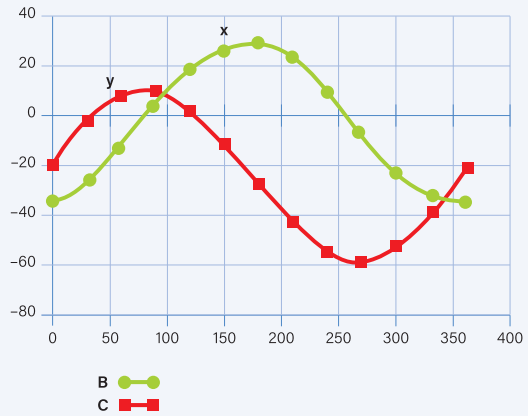


Bild 2. Ein ausgefuchster Versuchsaufbau zur Fehlersuche.



Winkel	x-Wert	y-Wert
0	-34	-20
30	-26	-2
60	-14	7
90	3	10
120	19	2
150	26	-11
180	28	-28
210	22	-43
240	9	-55
270	-7	-58
300	-23	-51
330	-32	-38
360	-34	-22

Bild 3. Die ermittelten Werte deuten auf einen Offset hin, der zum fehlerhaften Verhalten führt.

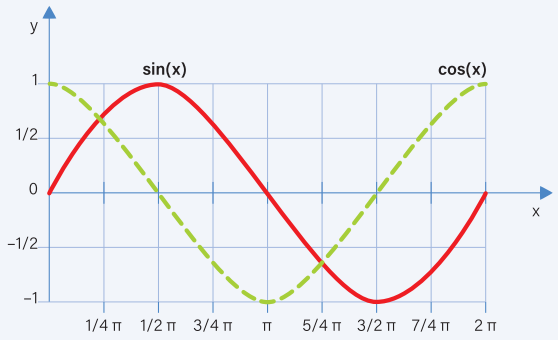


Bild 4. So sehen die beiden Kurven im Idealfall aus.

aufgrund der sperrigen Drähte abzuraten, so dass ich einen soliden Versuchsaufbau für die Messungen schuf, wie er in **Bild 2** zu sehen ist. Damit Sie erkennen, was Sie dort sehen: Als stabile Basis für den Kompass wurde eine umgedrehte, magnetisch garantiert unverdächtige Kaffeetasse gewählt. Auf dem Tassenboden ist mit Gummis die Platine mit dem (hier unsichtbaren) Sensor und dem Ring der LEDs sowie darunter ein Schreibstift als „Nordweiser" befestigt, wobei ein Hölzchen und ein Schwämmchen als Abstandshalter und Widerlager gleichzeitig dienen. Auf der Bestückungsseite des GY-271 ist die x- und die y- Richtung angegeben. Die rote Spitze des Stiftes entspricht der Richtung der LED 0 des Rings und der Minus-(!)-x-Richtung des Sensors, der mit der Bestückungsseite nach unten fixiert wurde. Die z-Werte sollen hier nicht betrachtet werden, aber sie sollten möglichst auf Null gehalten werden, um „Unruhe" im System zu vermeiden. Eventuell lassen sich die z-Werte später mit ein wenig Mathematik nutzen, um Hors-Catégorie-Steigungen im Gebirge zu messen. Die Tasse steht auf einem Blatt Papier, auf dem eine rudimentäre Windrose gezeichnet und dessen Mittelkreis ausgeschnitten ist, damit er sich mitdrehen kann. Und auch wenn er vielleicht nicht so aussieht, hat dieser Aufbau nicht nur zu reproduzierbaren Messergebnissen, sondern auch zu einem bemerkenswerten Aha-Effekt geführt.

Ich habe den Kaffeetassen-Kompass also gedreht, dabei die x- und y-Werte aufgezeichnet und in dem Excel-Sheet in **Bild 3** eingetragen (x-Werte blau, y-Werte rot). Vergleicht man dies mit den idealen Kurven **Bild 4**, erkennt man sofort das Malheur: Man sieht eine Sinus- und eine Cosinus-Kurve (ein um 90° verschobener Sinus).

In der Schule haben wir gelernt, dass in einem rechtwinkligen Dreieck der Sinus das Verhältnis von Gegenkathete zur Hypotenuse, beim Cosinus von Ankathete zur Hypothenuse und beim Tangens von Gegenkathete zur Ankathete ist. Was einem dazu spontan einfällt: Der Quotient von Sinus zu Cosinus ist der Tangens, weil die Hypotenuse sich ja wegkürzt. Um den Winkel aus den Seiten zu bestimmen, benötigen wir die Umkehrfunktion, den Arkustangens oder arctan.

Arduino kennt den Befehl arctan2, bei dem beide Werte x und y verwendet werden. Eine Normierung auf 1 ist nicht erforderlich, aber die Amplituden sollten gleich sein. Die Nullpunkt-Verschiebung ist als Ursache des Fehlers anzusehen. Nun wird auch klar, warum man mit dem Handy mysteriöse 8-förmige Bewegungen ausführen soll, um eine hohe Genauigkeit der Kompassmessung zu erreichen. Offenbar sind dort HMC-Kompasssensoren eingebaut! Bei einigen Bekannten funktioniert diese Vorgehens- (oder sollte man sagen: Achtgehens-)weise trotzdem nicht, weil die Verschiebung offenbar zu groß war. Ähnlich könnte ein Verfahren zum „automatischen" Abgleich beim Arduino erfolgen, bei dem die Maximalwerte erfasst und Korrekturwerte zugefügt werden. Hierzu wäre aber ein Umschalter erforderlich, was den schaltungstechnischen Aufwand von ein auf zwei Pins glatt verdoppeln würde. Da ich aber die komplette Kurve aufgenommen hatte, kannte ich den Grad der Verschiebung und brauchte diesen Wert nur im Programm zu korrigieren.

Ein neues Programm

Großes Plus der ursprünglichen Software war, dass auf die Initialisierung und Signalauswertung des HMC aufgebaut werden konnte. Ich habe einige Änderungen im Monitorprogramm durchgeführt, die riesigen Werte durch 100 dividiert und übersichtlicher angeordnet, denn schließlich kennt der Neopixel-Ring nur zwölf Stellungen, so dass eine Genauigkeit von nur etwa +/-10% ausreichend ist. Wenn man einen größeren Neopixel-Ring oder gar ein graphisches Display verwenden möchte, muss man die großen Werte entsprechend anpassen beziehungsweise verfeinern. Die Ansteuerung des Neopixels habe ich auf der Basis der sehr übersichtlichen Programme von [3] ebenfalls abgespeckt, was sich bei zwölf LEDs als

Listing 1. Hardcode-Korrektur der Abweichungen.

```
Serial.print("X Value: ");
x = x/100;
x = x+2;
Serial.print(x);
Serial.print("  Y Value: ");
y = y/100;
y = y+20;
Serial.print(y);
Serial.print("  Z Value: ");
z = z/100;
Serial.print(z);
```

praktikabel herausgestellt hat. Die eigentliche Änderung zur Korrektur des HMC-Fehlers ist in dem Ausschnitt des Source-Codes in **Listing 1** zu sehen. Das vollständige und vollständig modifizierte Arduino-Programm kann von [4] heruntergeladen werden. Besonders anschaulich in diesem Sketch ist die Monitor-Funktion, die zu eigenen Experimenten einlädt.

Bewährung in der Praxis

Der Kompass sollte in ein durchsichtiges Kunststoff-Gehäuse eingebaut und am Fahrrad-Lenker befestigt werden. Dabei wurde der freie Aufbau beibehalten, die einzelnen Ebenen durch Plastik-Scheiben getrennt und unten durch eine Platte mit Klebestreifen abgeschlossen. Mit einer weiteren, innen kreisförmig ausgesparten und „gezackten" Platte wurde der Neopixel-Ring fixiert.

Die Werte wurden mit LED 0 in Richtung Norden aufgenommen, bei den ersten Tests mit der Software stellte sich heraus, dass der LED-Ring aber die Südrichtung anzeigte. Jede Ambition zur Software-Änderung wurde abgewürgt und das Problem pragmatisch gelöst, indem der Kompass mit der LED 6 Richtung Norden fixiert und das Ganze dann in Fahrrad-Längsrichtung montiert wurde. Wenn das Fahrrad in Richtung Norden gedreht wird, leuchtet die vorderste LED. Dreht man das Fahrrad dann nach rechts, leuchtet eine LED weiter links, also weiter Richtung Norden. Genau so ist es auch beabsichtigt!

Allerdings hat mir das Eisenmaterial des Lenkers mit seinem Restmagnetismus einen Strich durch die Rechnung gemacht. Indirekt habe ich aber den Nachweis erbracht, dass der Rahmen tatsächlich aus Aluminium besteht, was durch seine dielektrischen Eigenschaften beim Kompass nicht stört. Da die Schaltung klein ist, wurde sie im Blickfeld am Rahmen fixiert (**Bild 5**). Beim ersten Praxistest „im Feld" konnte eine einwandfreie Funktion nachgewiesen werden. Eine Abweichung gab es nur bei einer Eisenbahnbrücke, wahrscheinlich verursacht durch das Feld des Fahrdrahts. Beim nächsten Mal an selber Stelle war keine Abweichung zu erkennen. Insgesamt kann ich mich durch den Kompass im unbekannten Gelände nun viel besser orientieren!

200597-02

DER KOMPASS AM E-BIKE

Die Stromversorgung der Schaltung habe ich an meinem E-Bike folgendermaßen realisiert: Unter dem Lenker versteckt wurde eine normale Autosteckdose montiert, die über zwei Mini-Stecker (rot und blau aus der Puppenstube) mit der Ladebuchse des E-Bike-Akkus verbunden wird. Auf genügende Isolation der Drähte muss geachtet werden, denn auch bei der geringen Spannung des Akkus von üblicherweise 36 V ist genügend Energie vorhanden, um Unheil anzurichten. Im Kurzschlussfall brennt ein ungeeignetes Kabel durch wie eine Sicherung! Die Betriebsspannung von 5 V wird mit einem USB-Wandler mit 24 V-Eingang aus dem LKW-Handel erzeugt. Da die Ladespannung bei LKWs deutlich höher als der Nennwert ausfallen kann, ist in der Regel ein Sperrwandler des Typs MC34063A verbaut, der Eingangsspannungen bis zu 40 V verkraftet. Der USB-Wandler wurde aufgeschraubt und eine Doppel-Litze mit Steckverbinder direkt an 5 V gelötet, da der Verbrauch des Kompasses recht gering ist.

Bild 5. Der aufgebaute Kompass am Fahrradrahmen.

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

Sie haben Fragen oder Kommentare?

Haben Sie Fragen oder Anmerkungen zu diesem Artikel? Senden Sie eine E-Mail an den Autor unter Kolde322@magenta.de und/oder die Elektor-Redaktion unter redaktion@elektor.de.

WEBLINKS

[1] **GY-271-Modul**: www.az-delivery.de/products/gy-271-kompassmodul-kompass-magnet-sensor-fuer-arduino-und-raspberry-pi

[2] **Kostenloses E-Book zum GY-271**: www.az-delivery.de/products/gy-271-kompassmodul-kompass-magnet-sensor-fur-arduino-und-raspberry-pi

[3] **NeoPixel-Bibliothek**: https://github.com/adafruit/Adafruit_NeoPixel

[4] **Projektseite**: www.elektormagazine.de/200597-02

Berechnen Sie die CO_2-Bilanz Ihrer Elektronik

Kennen Sie Ihren Fußabdruck?

Über die Autorin

Priscilla Haring-Kuipers schreibt aus einer sozialwissenschaftlichen Perspektive über Technologiethemen. Sie ist besonders daran interessiert, dass Technologie „das Gute im Menschen" unterstützt und ist Anhängerin der Wirkungsforschung. Sie hat einen MSc in Medienpsychologie und ist Gründerin von *This Is Not Rocket Science* [6].

Von **Priscilla Haring-Kuipers** (Niederlande)

Das Thema Ethik in Entwicklung, Produktion und Nutzung der Elektronik sollte für professionelle Ingenieure und Maker gleichermaßen wichtig sein. Möchten Sie die CO2-Bilanz Ihres technischen Equipments berechnen? Mit den richtigen Tools und ein wenig Anleitung können Sie den CO2-Fußabdruck Ihrer Elektronikprojekte berechnen.

Wenn man der Welt etwas hinzufügt, was es vorher nicht gab, hat das Konsequenzen. Ein Maßstab für negative Konsequenzen ist unser CO_2-Fußabdruck. Wir können die CO_2-Bilanz aus all unseren Lebensentscheidungen (zum Beispiel das Autos, das wir fahren) berechnen. Dies trifft auch auf die Elektronik zu, die wir herstellen, kaufen und benutzen. Sollten wir nicht wissen, wie groß dieser negative Fußabdruck ist und vielleicht versuchen, ihn auszugleichen? Mit *Idemat* und ein paar intelligenten Vermutungen berechnete ich die Ökobilanz unserer kleinen Synthesizer-Produktion.

CO_2-Fußabdruck

Der Kohlenstoff-Fußabdruck ist zu einem gängigen Konzept geworden, um etwas über den Druck auszusagen, den wir auf die Welt ausüben. „Die Ökobilanz umfasst den gesamten Ausstoß von Treibhausgasen, der durch eine Person, eine Veranstaltung, eine Organisation, eine Dienstleistung, einen Ort oder ein Produkt verursacht wird, ausgedrückt als Kohlendioxid-Äquivalent" [1]. Es gibt viele Online-Rechner, mit denen Sie den Fußabdruck Ihres Energiebedarfs zu Hause oder im Büro ermitteln können. Und für den Urlaub, den Sie im vergangenen Jahr nicht gemacht haben. Sobald Sie eine Zahl haben, können Sie diese nutzen, um als Verbraucher Kaufentscheidungen zu treffen oder den Fußabdruck zu kompensieren, indem Sie in eines der vielen kohlenstoff-negativen Projekte investieren. Oft bieten diese Organisationen einen weiteren Online-Rechner an, der Ihnen zeigt, wie viel es kosten würde, Ihren Kohlenstoff-Fußabdruck auszugleichen, und eine Möglichkeit nennen, dies zu tun. Berechnen und bezahlen! Ich möchte betonen, dass der Kohlenstoffausgleich eine Endverbraucherlösung ist und keineswegs ein Ersatz für die sehr notwendigen politischen und industriellen Veränderungen, aber es ist etwas, das wir selbst tun können und etwas, das wir jetzt tun können.

Ich betreibe ein kleines Produktionsunternehmen und würde gerne zumindest den Kohlenstoff-Fußabdruck der Elektronik, die wir in die Welt setzen, kompensieren. Es ist nicht leicht herauszufinden, wie groß der Kohlenstoff-Fußabdruck der Dinge ist, die wir herstellen. Wir wissen, dass die Chipindustrie einen riesigen Kohlenstoff-Fußabdruck hat, aber das sagt uns nicht viel über einen bestimmten Chip [2]. Wir wissen zum Beispiel, dass die Herstellung eines iPhones 65...95 kg CO_2 kostet und ein Fairphone3 in der Herstellung nicht einmal 40 kg. Aber das, was wir herstellen, hat wenig Ähnlichkeit mit einem Smartphone [3]. Es gibt eine enorme Vielzahl von Ökobilanz-Tools und Datenbanken, aber diese erfordern eine zeitaufwändige Einarbeitung in die Herstellungskette. So etwas scheint für große Konzerne gemacht zu sein und nicht für unsere Kleinserie oder Ihr nächstes Hardware-Experiment zu Hause [4].

Fußabdruck eines Fenix

Nach einigem Stöbern im Internet fand ich *Idemat* als das brauchbarste und offenste Tool, mit dem Sie die Daten Ihres Elektronikprojekts zusammenstellen und dessen CO_2-Fußabdruck berechnen können [5]. *Idemat* bietet drei verschiedene Recycling-Möglichkeiten an, die sich auf die Kohlenstoff-Bilanz auswirken. Wir wählen das umweltfreundlichste „Closed-Loop-Recycling" für unsere Synthesizer, da wir selbstverständlich jederzeit ein Gerät zurücknehmen und versuchen, es zu reparieren. Er wurde schießlich dafür entworfen und gebaut, repariert zu werden. Musikinstrumente sind eine Klasse von

Elektronikgeräten, die für einen jahrzehntelangen Einsatz gedacht ist. Sie müssen gereinigt und repariert werden, wenn sie kaputt sind, und fast jedes Synthesizer-Geschäft bietet auch einen Reparaturservice an. Die meisten Hersteller unterstützen dies, indem sie Reparatur- und Wartungsanleitungen mit Schaltplänen zur Verfügung stellen. Wir tun das auch.

Das Produkt, das wir untersuchen, ist unser Fenix IV, ein modularer Synthesizer mit einer Produktionsauflage von 100 Stück. Mit *Idemat* können wir beginnen, den Fußabdruck unserer Materialien zu berechnen: Aluminium für das Gehäuse und die Frontplatte, ABS-Kunststoff für die Drehknöpfe, Karton und Polyethylen-Luftpolsterfolie für die Verpackung und vieles mehr. Es gibt eine eher enttäuschende Pauschalkategorie für „PCB", die an mit ICs bestückte Platinen in einem Laptop angelehnt ist. Wir verwenden zwar diese Kategorie für einen Teil unserer Elektronik, haben aber auch unsere eigene fundierte Schätzung für eine unbestückte Leiterplatte erstellt. Wir gehen von einem gewichteten Mix aus Glasfaser, Epoxid und Kupfer aus (1 kg FR4 kostet 2 kg CO_2) und fügen unsere Schätzungen für Buchsen und Potentiometer an. Wir wählen Prozesse wie das Spritzgießen unserer Knöpfe sowie die Bearbeitung des Aluminiums und die Pulverbeschichtung des Gehäuses. Wir addieren die Kohlenstoffkosten der Logistik durch interkontinentale Luft- und Eisenbahnfracht. Es ist eine lange Liste von Dingen [6] und dennoch nicht perfekt und nicht so detailliert wie eine richtige Ökobilanz, aber ich denke, wir nähern uns der Realität an. Mit einem anderen Kohlenstoff-Rechner fügen wir unsere persönliche Logistik hinzu. Eine Einkaufsfahrt nach Shenzhen (5450 kg CO_2) und eine Fahrt von 2680 km zum Besuch der Fabrikationsstätten in den Niederlanden (290 kg CO_2). Unser persönliches Umherziehen hat den mit Abstand größten Fußabdruck.

Ich betreibe eine kleine Produktion und möchte zumindest die Kohlenstoff-Last der Elektronik, die wir in die Welt setzen, kompensieren.

Am Ende kommen wir auf eine Belastung von 316 kg CO_2 für einen Fenix IV. Aluminium (29 kg CO_2) trägt erstaunlich wenig dazu bei, obwohl wir viel davon für das Gehäuse und die Frontplatte verwenden. Nickel, das nur ein kleiner Bestandteil unserer Buchsen und Patchkabel ist, trägt überraschend viel bei (3 kg CO_2). Der Beitrag der Platinen mit bestückten Halbleitern ist sehr hoch (208 kg CO_2) und unser größter Beitrag in absoluten Zahlen. Bambus hat einen superniedrigen Beitrag von nur 150 g CO_2. Wir kompensieren die gesamte Belastung von 316 kg CO_2 mit unserer Lieblings-Kohlenstoff-Kompensationsorganisation Trees For All [7].

Ihr Fußabdruck

Idealerweise sollten in den Bauteilbibliotheken der Entwicklungssoftware Angaben zur CO_2-Bilanz und ähnlichen Kategorien enthalten sein. Dann hätte man diese Informationen zu einem Zeitpunkt, an dem man noch Entscheidungen darüber treffen kann, was man verwenden will und was nicht. So ist aber alles, was wir tun können, es nachträglich zu korrigieren.

Und was ist mit Ihrem Projekt? Ist es ordnungsgemäß recyclebar und kann bei Ihrem örtlichen Wertstoffcenter getrennt abgegeben werden? Oder kann Ihre Hardware sogar repariert und upgecycelt werden? Ich hoffe, dass Sie versuchen, den Kohlenstoff-Fußabdruck der Dinge, die Sie bauen, zu berechnen. Wenn Sie diese Berechnungen machen, dann teilen Sie sie bitte hier bei Elektor, denn wir könnten wirklich mehr Informationen über die Fußabdrücke all unserer elektronischen Kreationen gebrauchen. Ich bin sehr daran interessiert, das herauszufinden!

210340-02

WEBLINKS

[1] Wikipedia, „CO2-Bilanz": https://de.wikipedia.org/wiki/CO2-Bilanz

[2] A. Crawford, I. King, und D. Wu, „The Chip Industry Has a Problem With Its Giant Carbon Footprint", Bloomberg.com, 8. April 2021: http://bit.ly/bloom-carbon-foot

[3] M. Proske, et al, „Life Cycle Assessment of the Fairphone 3", Fraunhofer IZM, Juli 2020: http://bit.ly/fairphone3-lca

[4] Europäische Plattform für Ökobilanzen: https://eplca.jrc.ec.europa.eu/LCDN/

[5] Idemat: http://idematapp.com/

[6] TiNRS Fenix IV LCA: https://bit.ly/3h0W8y0

[7] TreesForAll: https://treesforall.nl/en

WORLD ETHICAL ELECTRONICS FORUM

Haben professionelle Ingenieure und Maker eine ethische Verantwortung, ihre Fähigkeiten auf positive Weise einzusetzen? Sollte Ethik eine Rolle dabei spielen, wie Elektronikfirmen ihre Produkte herstellen und verkaufen? Elektor und seine Partner gehen diesen Fragen nach, und wir laden Sie zum Mitdenken ein. Im November 2021 werden wir das erste jährliche *World Ethical Electronics Forum* (WEEF) veranstalten, bei dem eine Vielzahl von ethischen Themen diskutiert werden soll. Besuchen Sie *ElektorMagazine.com* für aktuelle Informationen.

ESP32-verbundenes Thermostat

Lagern Sie Ihren Wein bei der richtigen Temperatur!

Von **Yves Bourdon** (Frankreich)

Wein muss in einem dunklen und kühlen Raum gelagert werden, um eine gute Langzeitreifung zu ermöglichen. Fachleute sind sich zwar nicht einig, welche Temperatur dafür die beste ist, aber sie alle empfehlen, die Temperatur möglichst konstant zu halten. Das hier vorgestellte Thermostat versucht dies nicht nur, sondern sendet auch E-Mail-Warnungen, wenn die Temperatur zu hoch wird.

Die Idee zu diesem Projekt kam mir beim Bau des exzellenten WLAN-Thermostats von Elektor [1], das auf Basis des ESP8266 von Espressif entwickelt wurde. Ich habe versucht, die Software dieses Projekts auf einen ESP32 zu portieren, aber wegen des damit verbundenen Aufwands aufgegeben. Statt dessen fing ich „von vorne" an und ließ mich dabei von verschiedenen Elektor-Projekten inspirieren (Webinterface, NTP-Uhr, Temperaturregelung mit programmierbarer Hysterese, und so weiter).

Die kleine Unzulänglichkeit des Webinterfaces, dass die Benutzeroberfläche nicht in Echtzeit aufgefrischt wird, wurde beseitigt. Auch habe ich ein kleines OLED-Display hinzugefügt, das sehr praktisch ist und es ermöglicht, dieses Thermostat auch dann zu verwenden, wenn die WLAN-Verbindung unterbrochen ist.
Auch wenn Sie nicht passionierter Weinkenner und -trinker sind, können Sie diesem Projekt, eventuell angepasst, bestimmt etwas abgewinnen. Ich habe zum Beispiel die gleiche Hardware mit einem Feuchtigkeitsdetektor DHT11 (oder BME280) ausgestattet, um das mechanische Lüftungssystem in meinem Badezimmer zu steuern. Der optisch isolierte Eingang erkennt, ob das Licht eingeschaltet ist (12-V-Spotlights), so dass er den lauten nervigen Lüfter vorübergehend abschaltet, wenn man das Badezimmer besucht.

Die Schaltung

Der Schaltplan des ESP32-Thermostats ist in **Bild 1** zu sehen. Um Kosten, Zuverlässigkeit und Größe zu optimieren, habe ich mich entschieden, das Modul WiFi-Kit 32 von Heltec (**Bild 2**) als Herzstück des Thermostats zu verwenden. Dieses ESP32-basierte Modul, das leicht im Internet zu finden und zu bestellen ist, besitzt ein schönes blaues 0,96"-OLED-Display mit 128 × 64 Pixeln Auflösung und verfügt zudem über eine benutzerprogrammierbare weiße LED. Das Modul wird auf K5 und K6 gesteckt.

Technische Daten

- Temperatursensor DS18B20, Fehler ±0,5°C, Anzeigeauflösung 0,1°C
- Relais mit geschützten Kontakten ermöglicht das Schalten von AC-Lasten bis zu 2200 W
- 100...240 VAC Spannungsversorgung
- I²C-Erweiterungsbus kompatibel mit Grove und Qwiic
- Platine unterstützt WiFi-Kit 32 von Heltec mit integriertem OLED-Display und ESP32.DevKitC mit externem OLED-Display
- Summer für akustischen Alarm
- Eingang für externen Schalter
- Optisch isolierter Eingang
- Drei programmierbare Schwellenwerte: Minimum, Maximum und Alarm
- Alarm kann E-Mail senden
- NTP-Zeitbasis
- Wochentag-Timer
- Kann einen Kühler oder eine Heizung steuern

An Stelle des Heltec-Moduls kann auch ein ESP32-DevKitC-Modul verwendet werden, das auf die Verbinder K7 und K8 gesteckt wird. Dieses Modul muss aber mit einem separaten I²C-OLED-Displaymodul (ebenfalls 0,96" 128 × 64) ausgestattet werden. Auch diese beiden Module sind weit verbreitet (**Bild 3**). Aufgepasst! Das OLED-Display gibt es in zwei Versionen (mit vertauschten Pins für VCC und GND). Auf der Platine befinden sich daher zwei Anschlussmöglichkeiten (K9 und K10). Die Software ist für beide Module identisch.

Spannungsversorgung

Zur Stromversorgung wird ein kleiner 3-W-AC/DC-Wandler eingesetzt. Er akzeptiert eine Eingangsspannung von 100...240 VAC und gibt eine Gleichspannung von 5 V bei einem Strom von bis zu 600 mA aus. LED3 zeigt an, ob die Stromversorgung eingeschaltet ist. Eine Sicherung auf der Netzspannungsseite schützt das Gerät; Varistor VAR1 bietet einen von den Compliance-Regeln sehr geschätzten Überspannungsschutz. Die Netzspannung wird an der Platinenanschlussklemme K1 angeschlossen.

Das ESP32-Modul kann mit 5 V versorgt werden, da es über einen eigenen internen 3,3-V-Regler verfügt. Das Modul kann auch direkt über den Micro-USB-Bus versorgt werden.

Relais

Die Last, die das Thermostat steuert, wird an Relais RE1 angeschlossen, das von Port GPIO18 über einen kleinen NPN-Transistor (T1) gesteuert wird. LED2 zeigt den Zustand des Relais an. Die Diode D1 schützt die Elektronik vor vom Relais induzierten Strömen, während das Snubber-Netzwerk C4/R9 beziehungsweise C4/R10 (nur einen Widerstand montieren!) die Relaiskontakte schützt, indem es Schaltfunken bei hoher Last verhindert oder zumindest dämpft.

Das verwendete Relais kann 250-VAC-Lasten bis zu 10 A verarbeiten. An K2 stehen sowohl der Schließer- (NO) als auch der Öffnerkontakt (NC) zur Verfügung. K1 und K2 können zu einer 5-poligen Platinenanschlussklemme kombiniert werden.

Summer

Für die akustische Rückmeldung steht der Summer Buz1 zur Verfügung, der von Port GPIO19 über T2 angesteuert wird. Da der Summer einen eingebauten Oszillator besitzt, müssen Sie nur eine Spannung anlegen, um ihn zum Piepsen zu bringen. Es kann auch ein sogenannter AC-Summer ohne Oszilla-

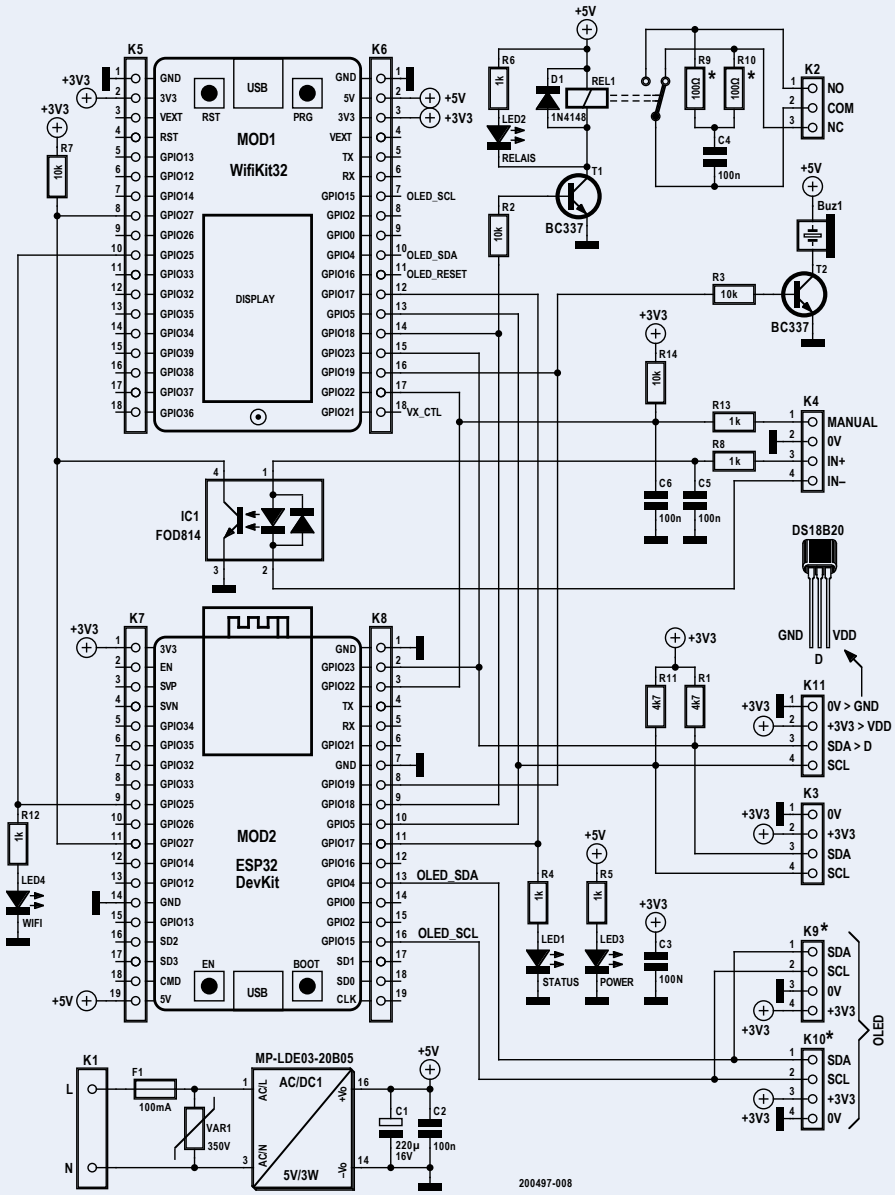


Bild 1. Der Schaltplan des ESP32-Thermostats zeigt viele Anschlüsse für mannigfaltige Konfigurationsmöglichkeiten.

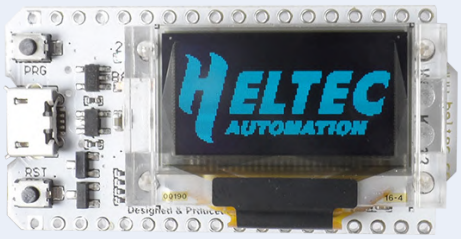


Bild 2. Das WiFi-Kit 32 von Heltec Automation.

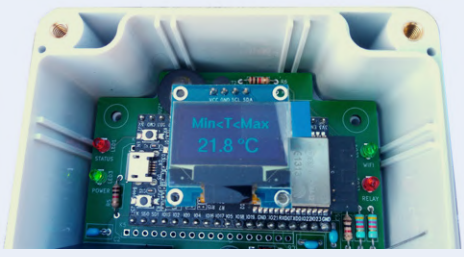


Bild 3. Der Prototyp aus dem Elektor-Labor verwendet ein ESP32-DevKitC-Modul mit einem separaten OLED-Display-Modul.

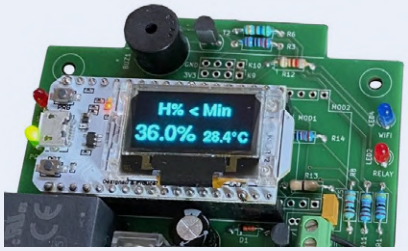


Bild 4. Das zum Feuchteregler umprogrammierte Thermostat ist mit einem WiFi-Kit 32 von Heltec versehen, das ein eigenes OLED-Display mitbringt.

Bild 5. Die Hauptseite des GUI zeigt die aktuelle Temperatur und die eingestellten Schalt- und Alarmschwellen.

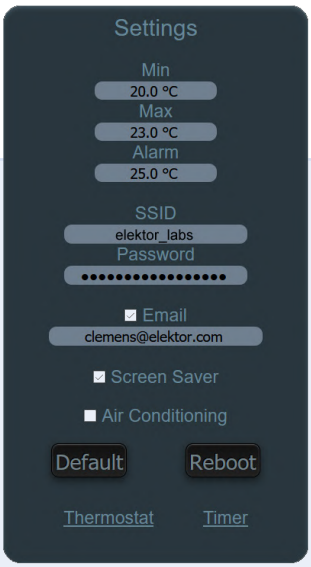


Bild 6. Unter „Settings" werden nicht nur die minimale und maximale Temperatur sowie die Alarmschwelle eingestellt, sondern auch die SSID und das Passwort des WLANs angegeben, mit dem das Thermostat verbunden wird.

tor verwendet werden, aber dann müsste der ESP32 den (PWM-) Ton selbst erzeugen.

Status-LED

LED1 an Port GPIO17 ist die Statusanzeige. Schnelles Blinken signalisiert einen Fehler, der entweder auf schlechte oder ungültige EEPROM-Daten oder auf Kommunikationsprobleme des DS18B20-Sensors zurückzuführen ist. Solche Fehler blockieren das Programm: Es wird angehalten, bis das Problem behoben ist. Kurzes Blinken im normalen Betrieb zeigt an, dass auf den integrierten Webserver zugegriffen wird.

Externe Bedienelemente

Die Klemmen an K4 ermöglicht den Anschluss eines externen Schalters (*Manual*) zwischen Port GPIO22 und GND. In der Standardsoftware setzt dieser Schalter das Thermostat außer Kraft und aktiviert das Relais. An diesen Eingang ist ein RC-Netzwerk (1,5 kHz mit den angegebenen Bauteilwerten) angeschlossen. Wenn ein bistabiler Schalter verwendet wird, kann C6 weggelassen und R13 durch einen Draht ersetzt werden.

Ein vom Optokoppler IC1 optisch isolierter digitaler Eingang IN, der an Port GPIO2 angeschlossen ist, steht für zukünftige Erweiterungen zur Verfügung und wird von der aktuellen Software nicht benutzt. Ich habe ihn in der bereits erwähnten Badezimmerlüftersteuerung verwendet, um die eingeschaltete Raumbeleuchtung zu erkennen (**Bild 4**).

Sensoranschluss und I²C-Port

Der Temperatursensor, ein One-Wire-DS18B20, ist für den Anschluss an die Klemmleiste K3 vorgesehen. Pin 1 ist an GND, sein DQ-Pin 2 wird mit Pin 3 von K3 verbunden und der VDD-Pin mit 3V3. Das bedeutet zwar, dass sich die Leitungen 2 und 3 kreuzen. Das stellt aber kein Problem dar, da bei typischen Thermostatanwendungen der Temperatursensor ohnehin nicht im Inneren des Gehäuses montiert werden sollte, da die Luft dort durch die Stromversorgung und das ESP32-Modul erwärmt wird, was eine Temperaturregelung unmöglich macht.

R1 ist für den ordnungsgemäßen Betrieb des DS18B20 erforderlich. Wenn man sich statt für diesen Temperatursensor für einen anderen oder etwa für einen Feuchtigkeitssensor entscheidet (wie einen BME280, SHT11 oder DHT22), wird R1 möglicherweise nicht benötigt.

Eigentlich ist K3 schon so belegt, dass er kompatibel zu den De-facto-I²C-Busstandards wie Grove (Seeed Studio) und Qwiic (SparkFun) ist. Da aber K3 ein Rastermaß von 3,5 mm und Grove ein Rastermaß von 2 mm besitzt, haben wir mit K11 einen „echten" Grove-Verbinder hinzugefügt, um die vielen erhältlichen Erweiterungsplatinen an die Thermostatschaltung anfügen zu können. So wird aus dem Thermostat eine hervorragende Basis für eine Unzahl anderer Anwendungen!

Die Platine

Die von Elektor entwickelte Platine ist so bemessen, dass sie in ein kostengünstiges Gehäuse mit einer transparenten Abdeckung passt. Die Leiterbahnen des Relais, die mit der Last verbunden sind, können und sollten mit Lötzinn verstärkt werden.

Software und Programmierung

Das Programm für das Thermostat ist in der Arduino-IDE (Version 1.8.12), also in C/C++ geschrieben. Das Programm ist recht umfangreich - mehr als 1.500 Zeilen Code - aber es ist ausführlich in einem Mix aus Englisch und Französisch kommentiert. Mit ein wenig Programmier- und auch ohne Sprachkenntnissen sollte Sie damit klarkommen und die Firmware an Ihre eigenen Bedürfnisse anpassen können. Ich habe ausschließlich frei verfügbare und bewährte Open-Source-Bibliotheken verwendet.

Bevor Sie den ESP32 in der Arduino-IDE verwenden können, müssen Sie zunächst das entsprechende Boards-Package installieren. Es gibt viele Webseiten, die diese

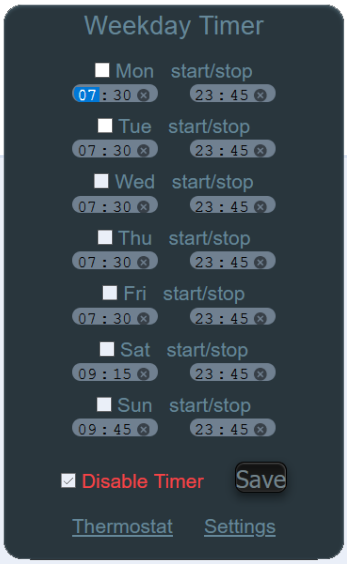


Bild 7. Auf der Seite „Weekday Timer" können Sie für jeden Wochentag eine Zeitspanne festlegen, in der das Thermostat die Temperatur kontrollieren soll.

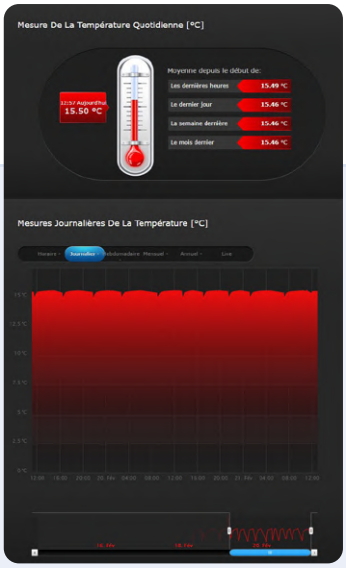


Bild 8. Die Regelung funktioniert recht gut, wie dieser Screenshot meines Hausautomationssystems zeigt. Eine kurzzeitige Abweichung von 1°C im Keller entspricht einer Abweichung von nur 0,1°C des Weins in der Flasche.

Prozedur erklären, siehe [2] für Details (mit Video). Das offizielle ESP32 Boards Package unterstützt sowohl das WiFi-Kit 32 von Heltec als auch das DevKitC (ESP32 Dev Module). Unabhängig davon, welches Modul Sie verwenden, können Sie alle anderen Einstellungen (CPU-Geschwindigkeit, Flash-Speichergröße, et cetera) auf ihren Standardwerten belassen. Nur der richtige COM-Port ist noch auszuwählen.

Neben der Installation des ESP32-Boards-Package ist es auch notwendig, das SPIFFS-Tool *ESP32 Sketch Data Upload* zu installieren. Auch dazu finden Sie unter [2] eine detaillierte Anleitung.

Wenn die Arduino-IDE vorbereitet ist, laden Sie den Thermostat-Sketch von [3] herunter und entpacken Sie ihn in den *Datei -> Sketchbook*-Ordner der Arduino-IDE. Die Datei *Thermostat.ino* und die dazugehörigen Dateien und der Datenordner müssen sich in einem Ordner namens *Thermostat* befinden. Der Unterordner *data* enthält die HTML- und CSS-Dateien sowie die vom Webserver benötigten JavaScripts. Die Datei *D7MR.woff2* enthält die von mir verwendete 7-Segment-Schriftart. Ein großes Dankeschön an den Autor!

Damit Sie es so bequem wie möglich haben, habe ich alle für das Projekt benötigten Bibliotheken in den Download [3] aufgenommen. Sie müssen korrekt installiert, das heißt, in den Unterordner *libraries* des *Sketchbook*-Ordners der IDE kopiert werden. Wo sich dieser auf Ihrem Computer befindet, hängt davon ab, wie Sie die Arduino-IDE eingerichtet haben. Siehe [4] für Details zur Arduino-IDE.

Sobald die gesamte Software heruntergeladen und installiert ist, müssen Sie einige Standardparameter (zu Beginn) des Programmlistings an Ihre Heim- oder Anwendungsumgebung anpassen. Da das Thermostat eine feste IP-Adresse verwendet, müssen Sie diese zusammen mit dem Gateway und anderen typischen Netzwerkdingen im Programm „hartkodieren".

E-Mail-Konfiguration

Da der Thermostat E-Mails senden kann, ist es notwendig, die Details des E-Mail-Servers einzugeben. Ich empfehle, dafür ein Gmail-Konto anzulegen. Wenn Sie das tun, lautet der E-Mail-Server *smtp.gmail.com* und der zu verwendende Port 465. Aus Sicherheitsgründen muss das Gmail-Konto etwas konfiguriert werden. Wenn Ihr Konto eine Ein-Faktor-Authentifizierung verwendet (es ist nur ein Passwort nötig), müssen Sie das Konto über den Link [5] konfigurieren. Im Falle einer Zwei-Faktor-Authentifizierung (Passwort plus Code per SMS) lesen Sie bitte [6].

Nun ist es an der Zeit, den Sketch wie jeden anderen Arduino-Sketch zu kompilieren und hochzuladen. Eigentlich sollten dabei keine oder nur geringfügige Probleme auftreten, und wenn doch, so überprüfen Sie bitte Ihre Installation.

Nachdem Sie den Sketch programmiert haben, laden Sie die Webseiten und andere Data-Dateien mit dem Befehl *ESP32 Sketch Data Upload* aus dem Menü *Werkzeuge* der IDE hoch. Dadurch wird alles aus dem Datenordner in den Flash-Speicher (SPIFFS) des ESP32 übertragen.

Erstes Booten

Wenn Sie das Gerät zum ersten Mal booten, verbinden Sie sich mit einem eigenen WLAN-Access-Point (AP) mit der SSID „Thermostat ESP32". Ein Passwort gibt es nicht. Öffnen Sie einen Browser und gehen Sie zur IP-Adresse 192.168.4.1. Es sollte nun die Weboberfläche erscheinen. Klicken Sie auf den Link *Settings* und dann auf die Schaltfläche *Default*. Dadurch werden alle Parameter einschließlich der Timer- und Temperatureinstellungen mit ihren Standardwerten initialisiert und die angegebene SSID aktiviert, so dass sich das Thermostat mit Ihrem WLAN verbinden kann. Starten Sie das Gerät über die Schaltfläche *Reboot* neu, so dass ein Soft-Reset durchgeführt und die Parameter aktiv werden.

Beachten Sie, dass das System automatisch jedesmal neu startet, wenn die SSID und/oder das Passwort geändert wird.

Wenn Sie die Einstellungen des Timers oder der Temperaturen ändern, müssen Sie dies nicht extra bestätigen, es reicht, wenn Sie den Mauszeiger aus dem entsprechenden Feld bewegen. Außerdem wird zur Sicherheit die Sinnhaftigkeit der eingestellten minimalen und maximalen Temperaturschwellenwerte überprüft: Wenn der Maximalwert niedriger als der Minimalwert ist, werden die beiden vertauscht.

Farbenfrohe Benutzeroberfläche

Dem Design der Benutzeroberfläche habe ich viel Aufmerksamkeit geschenkt. Das Ziel war es, eine gut aussehende Weboberfläche zu schaffen, die auf den meisten gängigen und aktuellen Browsern sowie auf iPhones und Android-Smartphones korrekt angezeigt

PASSENDE PRODUKTE

- **ESP32 DevKitC (SKU 18701)**
  www.elektor.de/esp32-devkitc-32d
- **0.96" 128×64 I²C OLED Display (SKU 18747)**
  www.elektor.de/blue-0-96-oled-display-i2c-4-pin
- **Elektor ESP32 Smart Kit Bundle (SKU 19033)**
  www.elektor.de/19033

wird. Die Oberfläche ist sehr reaktionsschnell ohne spürbare Latenz und alle Werte werden jede Sekunde aktualisiert.
Die webbasierte Benutzeroberfläche besteht aus drei Seiten: Hauptseite *Main* (**Bild 5**), Einstellungen *Settings* (**Bild 6**) und *Timer* (**Bild 7**).
Auf der Hauptseite werden Farbcodes verwendet, um den Status der Parameter intuitiv anzuzeigen. So wird beispielsweise die WLAN-Signalstärke (RSSI) je nach Signalqualität in Grün, Orange oder Rot angezeigt. Die Uhrzeit, die von einem NTP-Server aus dem Internet bezogen wird, wird in rot angezeigt, wenn die Zeitschaltuhr das Relais steuert; andernfalls ist sie grün. In ähnlicher Weise wird die Temperatur in hellblau angezeigt, wenn sie unter dem Mindestwert liegt, in grün, wenn sie im erlaubten Bereich liegt und in rot, wenn sie über dem Maximalwert liegt. Blinkt der Temperaturwert orange, ist irgendwo ein Problem im System aufgetreten.
Eine kleine LED auf der Hauptseite zeigt den Echtzeit-Status des Relais an (grün für aus, rot für ein). Glauben Sie mir, das CSS-Stylesheet erforderte viel Arbeit, um diese farbige Anzeige zu gestalten!
Das OLED-Display zeigt ebenfalls den Status des Systems an. Da diese Displays recht schnell abbauen, wenn sie dauerhaft eingeschaltet bleiben, wurde eine Bildschirmschoner-Option hinzugefügt. Durch Drücken eines Tasters, der zwischen GPIO05 (K11, Pin 4) und GND (K11, Pin 1) angeschlossen ist, wird das Display nur kurz für einige Minuten eingeschaltet. Diese Funktion kann auf der Seite *Settings* aktiviert oder deaktiviert werden.
Die Ein- und Ausschaltzeitpunkte für jeden Tag sowie das entsprechende Aktivierungshäkchen müssen auf der Timer-Seite mit der Schaltfläche *Save* gespeichert werden. Wenn Sie darauf klicken, wird die gesamte Seite gespeichert. Die Checkbox *Disable Timer* (de-) aktiviert den Timer und muss nicht mit der *Save* bestätigt werden. Wenn der Timer deaktiviert ist, steuert das Thermostat die Temperatur kontinuierlich.

Debug-Modus

Durch Abhören der seriellen Schnittstelle (115.200 Baud) über die eingebaute USB-Schnittstelle kann man den Initialisierungsprozess des Thermostats nach dem Einschalten der Stromversorgung beobachten. Dies ist sehr nützlich, um zum Beispiel Problemen mit der WLAN-Verbindung auf die Spur zu kommen.

Bild 9. Der DS18B20-Sensor wurde in eine separate kleine Kunststoffbox integriert, die strategisch geschickt im Keller platziert wurde. Ein zweiter, identischer Sensor meldet die Temperatur über ein eigenes Modul an mein Hausautomatisierungssystem. Dies ist eine zusätzliche Sicherheit für den Fall, dass das Thermostat ausfallen sollte, denn in diesem Fall würde mich mein Hausautomatisierungssystem warnen.

Ein zuverlässiges System

Das in diesem Artikel beschriebene Thermostat regelt die Temperatur in meinem Keller nun schon seit über einem Jahr und es wurden keine größeren Probleme festgestellt. Das System ist zuverlässig und die Regelung ist sehr gut (**Bild 8**). Ich konnte die Weboberfläche auf den meisten Internetbrowsern (Chrome, Edge, Safari, ...) sowie auf dem iPhone und einigen Android-Telefonen testen und habe keine Kompatibilitätsprobleme festgestellt.
Ich möchte mich an dieser Stelle bei Stephane Calderoni (PhD, Computer Science) bedanken, der auf Réunion (einer Insel vor Ostafrika) lebt und mir bei der Entwicklung der HTML-Schnittstelle und des dazugehörigen JavaScripts sehr geholfen hat. So habe ich den Corona-Lockdown im März 2020 in Frankreich dazu genutzt, um meine mangelhaften Kenntnisse in der „Internet-Programmiererei" zu verbessern.

200497-02

Ein Beitrag von

Idee, Design und Text: **Yves Bourdon**
Platinendesign und 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 eine E-Mail an die Redaktion unter clemens.valens@elektor.com.

STÜCKLISTE

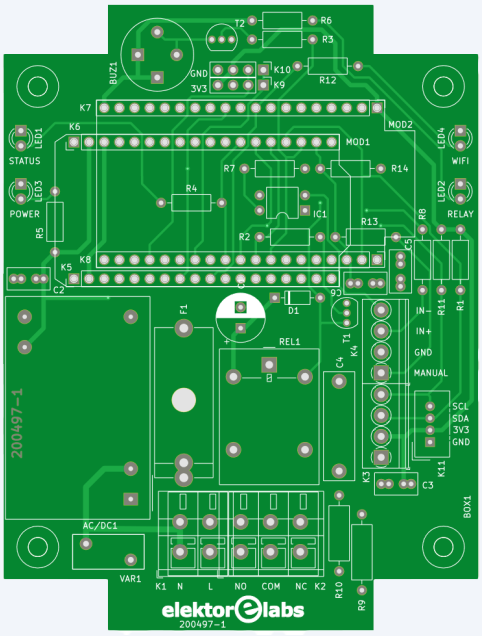
Widerstände:
(alle 5%, 0,25W)
R9,R10 = 100 Ω, 1 W (siehe Text)
R4,R5,R6,R8,R12,R13 = 1 k
R1,R11 = 4k7 k
R2,R3,R7,R14 = 10 k

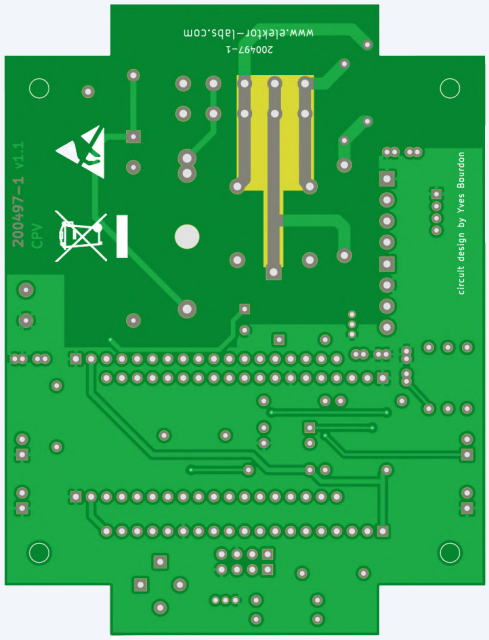
Kondensatoren:
C2,C3,C5,C6 = 100 n, Raster 2,5 mm oder 5 mm
C4 = 100 n, X2, Raster 15 mm
C1 = 220 µ, 16 V, Raster 3,5 mm

Halbleiter:
D1 = 1N4148
LED1,LED2 = LED, rot, 3 mm
LED3,LED4 = LED, grün, 3 mm
IC1 = FOD814, DIP4
T1,T2 = BC337

Sonstiges:
BUZ1 = Summer mit Oszillator, Raster 6,5 oder 7,6 mm
F1 = Sicherungshalter, 5x20 mm mit Schmelzsicherung 250 V, 100 mA, träge
K1 = 2-polige Platinenanschlussklemme, Raster 5 mm
K2 = 3- polige Platinenanschlussklemme, Raster 5 mm
K3,K4 = 4-polige Platinenanschlussklemme, Raster 3,5 mm
K5,K6 = 1x18-polige Stiftleiste, Raster 0,1" (für MOD1)
K7,K8 = 1x19-polige Buchsenleiste, Raster 0,1" (für MOD2)
K9,K10 = 1x4-fach Buchsenleiste, 0,1" Raster, 20 mm Höhe (bei MOD2)
K11 = 1x4-polige Stiftleiste, Raster 2 mm (Grove)
REL1 = Relais, 10 A, SPDT, 5VDC
VAR1 = Varistor 350 V

AC/DC1 = AC-DC-Wandler 5 V, 3 W, z. B. Multicomp MP-LDE03-20B05
BOX1 = Gehäuse, IP65, transparenter Deckel (Multicomp MC001106)
MOD1 = WiFi Kit 32 (Heltec)
oder
MOD2 = ESP32 DevKitC plus OLED-Display, I²C, 128x64 Pixel





WEBLINKS

[1] R. Aarts und C. Valens, „WLAN-Thermostat", Elektor, Januar 2018: www.elektormagazine.de/magazine/elektor-201801/41213
[2] ESP32 – Getting Started: www.elektormagazine.de/labs/1874
[3] ESP32-Thermostat bei Elektor Labs: www.elektormagazine.de/labs/4216
[4] Arduino-FAQ: www.elektormagazine.de/labs/1876
[5] Gmail: Ein-Faktor-Authentifizierung: https://myaccount.google.com/lesssecureapps
[6] Gmail: Zwei-Faktor-Authentifizierung: https://security.google.com/settings/security/apppasswords

Magnetische Levitation die digitale Art

Ein ESP32 Pico ersetzt den analogen Komparator

Von **Peter Neufeld**
und **Luc Lemmens** (Elektor)

Im Artikel „Magnetische Levitation - die einfache Art“ haben wir gesehen, dass magnetisches Schweben mit einem Permanentmagneten, der Spule und dem Kern eines Standard-5-V-Relais, einem Hall-Sensor und einem guten alten Analogkomparator nicht allzu schwierig zu erreichen ist. In diesem zweiten Projekt übernimmt das System-on-Chip ESP32 Pico in dem handlichen M5Stack ATOM die Steuerung.

Schon der Titel des ersten Artikels, „Magnetische Levitation - die einfache Art“ [1][2], ließ vermuten, dass es noch eine komplexere Herangehensweise an das Wunder der Levitation geben würde. Und wirklich - in diesem zweiten Teil des Levitationsprojekts [3] wird der analoge Komparator des ersten Entwurfs entweder durch einen M5Stack ATOM Lite oder ATOM Matrix ersetzt. Diesem kleinen ESP32-basierten Modul (siehe Textbox **M5Stack...**) wird die Aufgabe anvertraut, die Ausgangsspannung des Hall-Sensors zu messen und das Messergebnis zur Steuerung der Spule heranzuziehen. Tatsächlich braucht man, wenn man die Komponenten im M5Stack-Modul nicht einzeln mitzählt, viel weniger Teile als für die analoge Schaltung, um diese digitale Version zu bauen. Das Lustige daran ist, dass für einige unserer Leser der Mikrocontroller der einfache Weg ist, während andere sich beschweren und fragen, warum wir den Overkill eines ESP32-Moduls verwenden, wenn im Grunde auch ein einfacher LM311 die Aufgabe erfüllen kann. Ja, es ist schwer, es allen recht zu machen, und deshalb haben wir uns entschieden, Ihnen beide Möglichkeiten zu präsentieren.

Jede der Varianten des ESP32-Moduls [4] würde für dieses Projekt ausreichen. Da wir in einem der ersten Prototypen einen Taster zur Einstellung des Schwebe-Regelkreises benötigten, haben wir uns für ein ATOM-Modul entscheiden, das einen solchen Taster besitzt. Diese M5Stack-Module sind zudem preiswert, extrem klein und in ihren Kunststoffgehäusen gut gegen Kurzschlüsse und neugierige Finger geschützt. Alle für dieses Projekt benötigten I/Os sind an den SIL-Buchsen an der Unterseite der ATOMs leicht zugänglich. Die 5x5-Neopixel-LEDs im Matrix-Modell sind natürlich nur schmückendes Beiwerk und tragen nicht zum Schwebevorgang bei. Deshalb kann man auch das Lite-Modell mit nur einer Neopixel-LED für das Projekt verwenden.

Als ich anfing, diesen Artikel zu schreiben, kam Peter in einem Gespräch auf die Idee, anstelle des Druckknopfes ein Trimmpoti für die Einstellung des Regelkreises zu verwenden. Er hat diese Lösung schließlich auch aufgebaut und getestet und stellt diese „Extraversion“ auf seinen Webseiten [5] und auch später in diesem Artikel vor.

Die Hardware, Originalversion

Wie der Schaltplan in **Bild 1** zeigt, entspricht die verwendete Hardware in etwa der der analogen Version. Die Modifikation der Relaisplatine ist im Wesentlichen identisch mit Teil 1, der wichtigste Unterschied ist, dass der Hallsensor nun mit 3,3 V vom internen Spannungsregler des ATOM-Moduls versorgt wird. **Bild 2** zeigt den bearbeiteten Elektromagneten mit dem auf seinem Kern montierten Hall-Sensor. Hier ist die übliche Freilaufdiode für das Relais durch zwei weiße LEDs ersetzt, die gleichzeitig als „Scheinwerfer“ zur Beleuchtung des schwebenden Objekts fungieren. Eine einzelne LED wie bei der analogen Lösung würde zwar auch reichen, aber offensichtlich trägt der Aufbau mit zwei LEDs zur Stabilität des schwebenden Objekts bei. Es ist aber wichtig, den Freilaufstrom durch die LEDs gering zu halten, da ein höherer Strom das elektromagnetische Feld nach dem Abschalten des Transistors zu lange aufrecht erhalten würde [6]. Dieses zusätzliche Feld könnte den Regelkreis in seiner Reaktion beeinträchtigen und den Schwebezustand der Last destabilisieren. Die LEDs und der Widerstand sind aber unbedingt notwendig und müssen die ursprüngliche Diode ersetzen, da sie die Induktionsspannung am Relaistransistor

Apropos Hall-Sensoren...

Zunächst wurde auf Peters Webseiten für beide Versionen des Levitationsprojekts erwähnt, dass nur die Sensoren A1302 oder A1308 von Allegro Microsystems verwendbar seien, und dass der günstigere SS49E von Honeywell nicht funktioniere. Der Grund dafür blieb ihm zunächst unklar, bis er feststellte, dass er während der Entwicklung des allerersten Prototyps ein paar Bauteile gemeinsam austauschte, darunter den SS49E zugunsten eines A1302, woraufhin es schließlich funktionierte. Er vermutete fälschlicherweise, dass der Hall-Sensor der Übeltäter war, aber jüngste Tests bewiesen, dass es an den anderen getauschten Teilen lag und der SS49E in beiden Versionen des Projekts doch gut funktioniert. Eine gute Nachricht, nicht nur wegen des Preises, sondern auch, weil die Honeywell-Sensoren viel besser erhältlich sind!

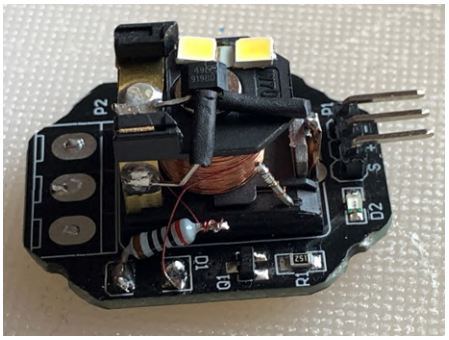


Bild 2. Die modifizierte Relaisplatine mit dem Hallsensor auf dem Kern des Elektromagneten.

auf ein unschädliches Maß von unter 50 V begrenzen und für schön kurze Schaltimpulse sorgen. Zwei weiße LEDs mit einem Vorwiderstand von 220...330 Ω erfüllten die Aufgabe sehr gut.

Natürlich hält die 3,3-V-Versorgungsspannung auch den Ausgangspegel des Sensors zwischen 0 V und 3,3 V und überschreitet damit nicht die maximale Eingangsspannung des ADCs des M5Stack ATOM. Diese Versorgungsspannung entspricht eigentlich nicht den Spezifikationen der meisten Hall-Sensoren (siehe Textbox „**Apropos...**"), die laut Datenblatt des A1302/A1308 wenigstens 4,5 V betragen sollte, aber anscheinend funktionieren sie in diesem (!) Projekt auch bei niedrigeren Spannungen. Dadurch ersparen wir uns einen zusätzlichen Spannungsteiler, der die Sensor-Ausgangsspannung an die Eingangsgrenzen des ADC anpasst. Allerdings gibt es auch einen Nachteil: Offenbar wird die Ausgangsspannung des Sensors bei dieser Versorgungsspannung bereits bei einer geringeren Magnetfeldstärke begrenzt. Auch für den SS49E ist das ein limitierender Faktor, obwohl dieser Hall-Sensor laut Datenblatt für eine minimale Versorgungsspannung von 2,6V spezifiziert ist

Die Schaltung wird mit 5 V versorgt, entweder über den USB-C-Anschluss, den Grove-Port-Anschluss oder die SIL-Buchse auf der Unterseite des M5Stack ATOM.

Die Software

Der Algorithmus im Arduino-Sketch zur Steuerung der Levitation ist im Grunde sehr einfach: Der ADC des ESP32 misst die Ausgangsspannung des Hall-Sensors, vergleicht sie mit einem Triggerpegel (mit einer gewissen Hysterese) und schaltet die Spule ein oder aus, genau wie der LM311-Komparator im ersten Entwurf. Im ersten Beispiel auf Peters Webseite (MagLev_1.INO) sind der Triggerpegel und die Hysterese als konstante Werte im Sketch definiert und müssen auf die verwendete Last abgestimmt werden. Es kann einige Zeit dauern, den richtigen Wert für den Triggerpegel zu finden, da er von der Last und natürlich von der Größe und Stärke des verwendeten Permanentmagneten abhängt. Der Quellcode dieses ersten Sketches ist eher eine einfache Demonstration des Steuerungsalgorithmus, da Sie ihn jedes Mal neu kompilieren und hochladen müssten, wenn Sie eine dieser Konstanten ändern, was nicht sehr praktisch ist. Der zweite Sketch verwendet den integrierten Taster des M5Stack ATOM, um während der Laufzeit den Schwellwert für die Ausgangsspannung des Hall-Sensors ändern zu können. Beim Start dieses Programms ist der Pegel auf einen vordefinierten Wert

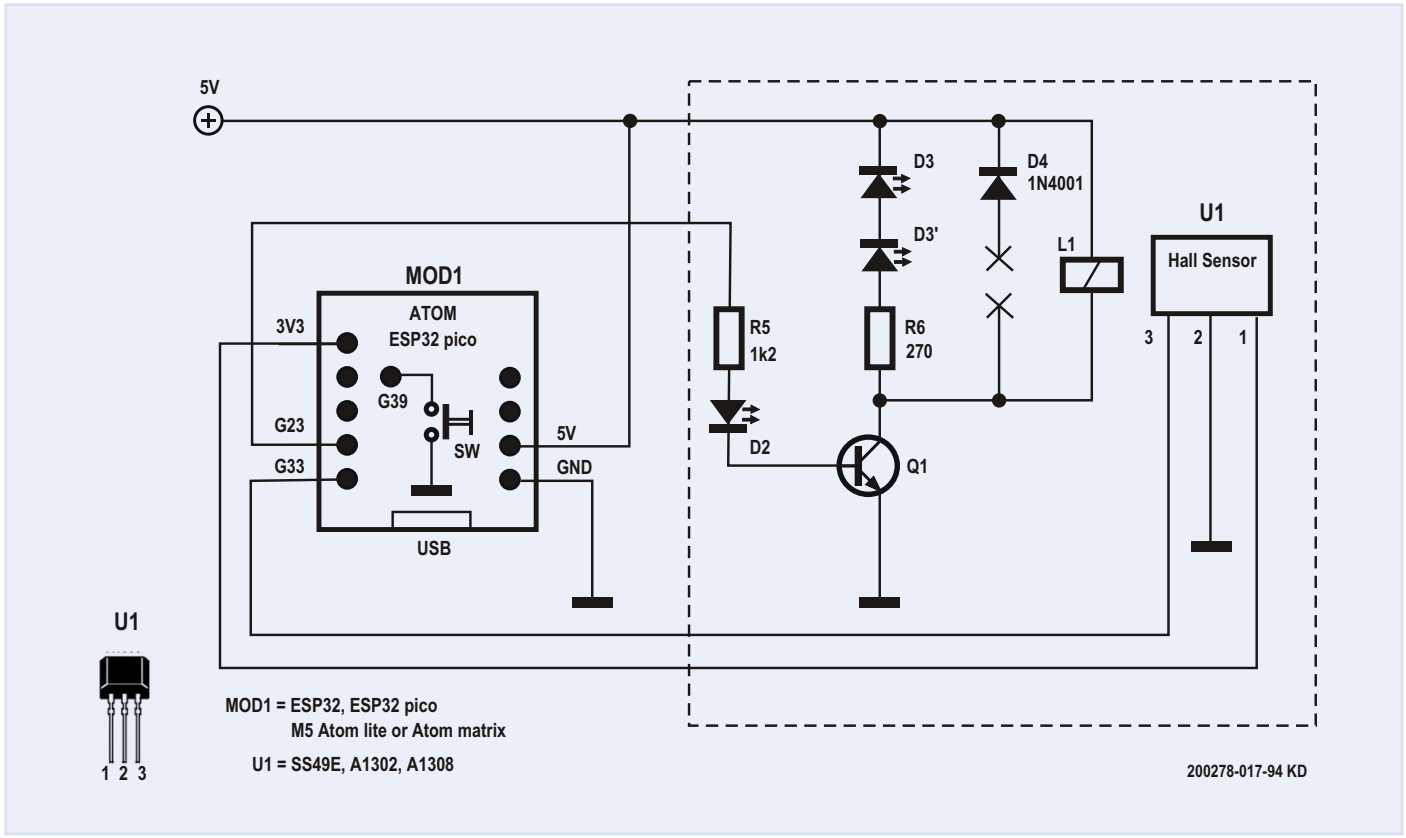


Bild 1. Der Schaltplan sieht aus wie bei der analogen Version, weist aber weniger Bauteile auf

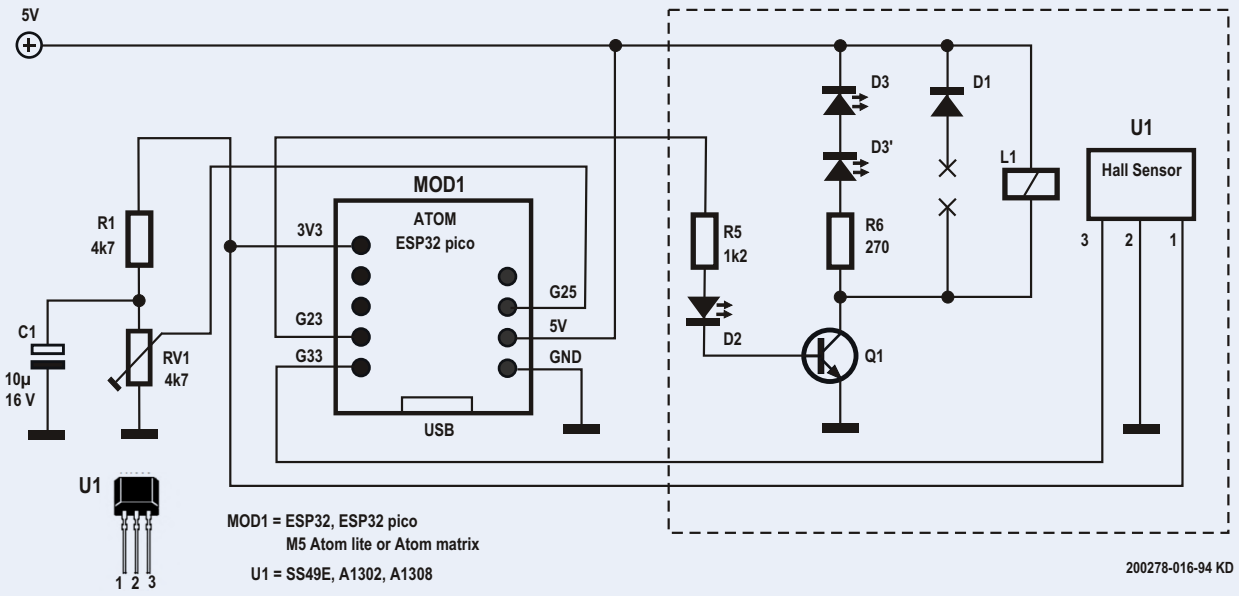


Bild 3. Drei zusätzliche Bauteile vereinfachen den Abgleich

eingestellt, der durch Drücken des Drucktasters des M5Stack ATOMs gesenkt werden kann, bis die Last unter dem Elektromagneten schwebt. In der aktuellen Version dieses Sketches (MagLev_2.INO) und mit nur einem einzigen benutzerdefinierten Taster zur Steuerung der Kalibrierung kann der Schwellwert allerdings nur verringert werden. Ist er einmal zu niedrig eingestellt, muss der M5Stack zurückgesetzt und der Kalibrierungsvorgang von vorne durchgeführt werden. Man kann die Einstellung im Seriellen Monitor der Arduino-IDE verfolgen, und wenn der richtige Schwellwert ermittelt ist, den Wert in den Quellcode des Sketches eintragen. Der Quellcode wird neu kompiliert, auf das M5Stack-ATOM-Modul hochgeladen und es sind, solange die Last nicht verändert wird, keine weiteren Anpassungen nötig, wenn man die Hardware aus- und wieder neu einschaltet. Wenn Sie sich den Quellcode dieses Sketches ansehen, werden Sie feststellen, dass mehrere Zeilen mit unterschiedlichen Voreinstellungen für die Schwelle auskommentiert sind. Natürlich hat jede Last und jeder Magnet einen anderen Schwellwert und verschiedene Typen von Hallsensoren können unterschiedliche optimale Schwellwerteinstellungen haben. Und es gibt sogar Unterschiede, wenn man den M5Stack ATOM Matrix gegen einen ATOM Lite austauscht. Offenbar unterscheiden sich die ADC-Charakteristika (oder deren Referenzspannungen?) von Modul zu Modul. Die Hysterese scheint etwas weniger kritisch zu sein, aber es kann nicht schaden, mit verschiedenen Werten zu experimentieren, um das schwebende Objekt zu stabilisieren. In allen Software-Versionen ist der Wert dieses Parameters hartkodiert und kann nur im Quellcode des Sketches geändert werden, was jedes Mal eine Neukompilierung und ein erneutes Hochladen der Firmware erfordert.

M5Stack: ATOM Lite oder ATOM Matrix?

Das Gehäuse des (etwas billigeren) Lite ist einige Millimeter niedriger als das der Matrix-Version, was wohl an der Neopixel-Matrix liegt. Zusätzlich enthält Matrix eine IR-LED und die *Inertial Measurement Unit* (IMU) MPU8668. Der Preisunterschied zwischen den beiden Versionen ist nur gering, und wenn Sie das Modul auch für zukünftige Projekte verwenden wollen, sollten Sie wegen der Extras den Kauf der etwas teureren Matrix-Version in Betracht ziehen.

Wo Digital und Analog sich treffen…

Sie sehen schon, trotz aller Digitalisierung ist es nicht der bequemste Weg, dieses Schwebeprojekt zu tunen. Auch nach dem ersten Programmieren der Firmware ist eine Verbindung zu einem Computer und eine (wiederholte) Neuprogrammierung des ESP32 erforderlich, um die Softwaresteuerung an die Last anzupassen (wenn man nicht bei jedem Einschalten die beiden Parameter mit dem Taster korrekt einstellen will). Und wenn die Last zu einem späteren Zeitpunkt geändert werden soll, ist erneut eine Verbindung zu einem Computer erforderlich, um den Schwellwert auf das neue Objekt abzustimmen. In dieser Hinsicht war die analoge Lösung in Teil 1 wirklich „der einfache Weg“,

Einrichtung der Arduino-IDE

Um den M5Stack (ATOM Lite oder ATOM Matrix) in der Arduino-IDE zu verwenden, sind einige Vorbereitungen erforderlich. Vorausgesetzt, die IDE selbst ist bereits installiert, wird im ersten Schritt der ESP32-Arduino-Kern hinzugefügt: Dazu gibt man unter *Datei -> Voreinstellungen* im Feld *Zusätzliche Boardsverwalter-URLs* die Adresse *https://m5stack.oss-cn-shenzhen.aliyuncs.com/resource/arduino/package_m5stack_index.json* an. Schließen Sie *Voreinstellungen* mit *OK*, öffnen Sie dann *Werkzeuge -> Board -> Boardverwalter...*, geben Sie *M5stack* in das Suchfeld oben im Fenster ein und drücken Sie den Installationsknopf für *M5Stack by M5Stack official*. Als dieser Artikel entstand, war die Version 1.0.7 aktuell. Nach der Installation können Sie das Modul M5Stack-ATOM unter *Werkzeuge -> Board -> M5Stack-ATOM* (unter M5Stack Arduino) auswählen, und vergessen Sie nicht, unter *Werkzeuge -> Port* die serielle Schnittstelle anzugeben, an der der M5Stack angeschlossen ist. Wenn Sie die Neopixel-LEDs verwenden möchten, müssen Sie außerdem die Bibliothek *Adafruit Neopixel* über *Sketch -> Bibliothek einbinden -> Bibliotheken verwalten* installieren.

STÜCKLISTE

Widerstände:
R5* = 1k2
R6 = 270 Ω

Halbleiter:
D2* = rote LED
D3, D3' = weiße LED (siehe Text)
D4* = nicht bestückt (oder von der Relaisplatine entfernen)
Q1* = BC550
U1 = Hallsensor A1302 oder A1308 (Allegro) oder SS49E (Honeywell)

Außerdem:
5V-Relaisplatine*
MOD1 = M5Stack ATOM Lite oder M5Stack ATOM Matrix
Neodym-Scheibenmagnete
*= siehe [1] für die Details zum Elektromagneten und der Relaisplatine

Zusätzlich benötigte Bauteile für den Analogabgleich

Widerstände:
R1 = 4k7
RV1 = Trimmpoti 5 k

Kondensator:
C1 = 10 µF, 10 V radial

indem er ein Potentiometer verwendete, um die richtige Position für die neue Schwebelast einzustellen.

Aber wie bereits erwähnt, kam Peter auf die Idee, ein Trimmpoti hinzuzufügen, um den Schwellwert während der Laufzeit einzustellen. Im Schaltbild in **Bild 3** sind links die drei zusätzlichen Bauteile zu sehen. Der Schleifer dieses Trimmpotentiometers ist mit einem zweiten Analogkanal des ESP32 verbunden. Dies kann eine vorübergehende, aber deutlich bequemere Lösung sein, mit der dritten Version der Software (MagLev_3.INO) den richtigen Schwellwertpegel zu ermitteln, der dann in MagLev_2.INO hartkodiert wird. Der Quellcode ist in der Textbox dargestellt. Der Einfachheit halber ist im Sketch die Unterstützung für die Neopixel-LED(s) des M5Stack ATOM weggelassen. Wenn Sie die Hardware auf ein absolutes Minimum beschränken wollen, könnten Sie das zusätzliche Trimmpoti RV1 mit dem Widerstand R1 und dem Elektrolytkondensator C1) später, wenn der richtige Schwellwert in der zweiten Version des Sketches eingetragen ist, wieder entfernen. Das Foto in **Bild 4** zeigt aber, dass diese Bauteile nicht viel Platz beanspruchen und gut versteckt werden können. Meiner Meinung nach ist diese analoge Abstimmung des digitalen Projekts eine wertvolle Ergänzung des Schwebeprojekts. Ich würde sie nicht entfernen! Der Download der Quellcodes für die drei Sketches ist unter [7] zu finden.

Aufbau

Die Fotos in **Bild 5** zeigen, wie das digitale Levitationsprojekt mechanisch aufgebaut werden kann. Der Elektromagnet (die modifizierte Relaisplatine) mit dem Hallsensor wird oben an einem Holzrahmen mit 10 cm Durchmesser montiert, das M5Stack-Modul unten. Wenn die Trimmpoti-Option verwendet wird, ist der beste Platz für die drei zusätzlichen Bauteile nahe am Boden der Konstruktion. Das ist während der Justierung mechanisch stabil und es ist einfacher, die Bauteile dort zu verstecken.

Bild 4. Trimmer an der Basis der mechanischen Konstruktion

Bild 5. Übersicht des mechanischen Aufbaus

PASSENDE PRODUKTE

› **Magnetsphere Kit (SKU 19133)**
www.elektor.de/magnetsphere-kit

Analog oder digital?

Einfach analog oder fortschrittlich digital, das ist hier die Frage. Wir überlassen es Ihnen, zu entscheiden, ob Sie die Old-School-Version mit dem analogen Komparator oder die modernere Mikrocontroller-Lösung aus diesem zweiten Teil bevorzugen. Wie erwähnt, begrenzt die 3,3-V-Versorgungsspannung für den Hall-Sensor das Gewicht der Last, die man mit der digitalen Lösung schweben lassen kann. Die ESP32-basierten Module wurden wegen ihres Preises und ihrer geringen Größe verwendet, nicht, weil irgendeine Art von drahtloser Verbindung benötigt wurde. Schließlich ist in diesem Entwurf das schwebende Objekt das einzige, das keinen Draht benötigt!

200278-02

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

Der Sketch: Der digitale Weg

```
int HALL_PIN  = 33;//analog input from HALL-sensor
int HALL_VAL  = 0; //holds output of the HALL-sensor
int ADJ_PIN   = 25;//analog input from trimmer (0 to 2V5)
int ADJ_VAL   = 0; //holds trigger level for actual payload
int HYSTERESIS= 50;//absolute hysteresis for trigger level
int HYST      = 0; //adds or subtracts hysteresis
int RELAIS_PIN= 23;//Output to 5V-relay/electromagnet
int i         = 0;

//-------------------------------------------------------------
//output to electromagnet
void setup()

// ADCs default is 12 bit with 11 dB attenuation
//-------------------------------------------------------------
void loop(){
  HALL_VAL  = 0;
  ADJ_VAL   = 0;
  for (i=1; i <= 2; i++){    //Read 2 times to avoid noise
   HALL_VAL = (HALL_VAL + analogRead(HALL_PIN));
   ADJ_VAL  = ADJ_VAL   + analogRead (ADJ_PIN);
  }
  HALL_VAL  = HALL_VAL / (i-1);     //compute the average
  ADJ_VAL   = ADJ_VAL  / (i-1);     //compute the average
  ADJ_VAL   = (ADJ_VAL/3)+2450;     //shift to a good range
  if (HALL_VAL < (ADJ_VAL + HYST) ){//Payload too low!
    digitalWrite(RELAIS_PIN, HIGH); //Lift the payload
    HYST = HYSTERESIS;         //Set upper hysteresis level
  }
  else {                           //Payload too high:
    digitalWrite(RELAIS_PIN, LOW);  //Drop the payload
    HYST = 0 - HYSTERESIS;     //Set lower hysteresis level
  }
}
//-------------------------------------------------------------
```

WEBLINKS

[1] **„Magnetische Levitation – die einfache Art“, Elektor 7-8/2021**: www.elektormagazine.de/200311-02
[2] **P. Neufeld, „Magnetic Levitation – The Easy Way“**: https://peterneufeld.wordpress.com/2020/06/20/magnetic-levitation-the-easy-way/
[3] **P. Neufeld, „Magnetic Levitation – The Digital Way“**: https://peterneufeld.wordpress.com/2020/07/04/magnetic-levitation-the-digital-way/
[4] **M5Stack-Varianten**: https://docs.m5stack.com/en/products
[5] **P. Neufeld, „Magnetic Levitation – Reloaded“**: https://peterneufeld.wordpress.com/2021/05/01/magnetic-levitation-reloaded/
[6] **Freilaufdiode**: https://de.wikipedia.org/wiki/Schutzdiode
[7] **Software-Download**: www.elektormagazine.de/200278-02

ULTIMATE Arduino Uno Hardware Manual

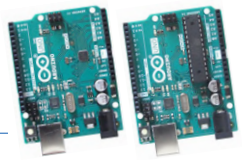
Ein beispielhaftes Kapitel: Bootloader für den Haupt-Mikrocontroller

Von **Warwick A. Smith** (Republik Südafrika)

In diesem Monat stellen wir in der Rubrik *Elektor Bücher* einen Auszug aus dem *Ultimate Arduino Uno Hardware Manual* von Warwick A. Smith vor, das im Juni 2021 im Elektor-Verlag erschienen ist. Auch wenn der Arduino Uno weit verbreitet und dazu auch noch preiswert ist, kann man leicht wichtige Hardware-Spezifikationen übersehen und dann in seiner Zielanwendung auf Probleme stoßen. Probleme mit dem Uno-Bootloader und den Fuse-Einstellungen verwirren immer noch viele Anwender und rangieren weit oben in den Top-10 der Suchanfragen bei Elektor Labs. Es ist also Zeit für ein paar „ultimative“ Antworten, die dieses neue Elektor-Buch gibt.

Bootloader für den Haupt-Mikrocontroller

Wenn der Haupt-Mikrocontroller ATmega328P in einem Arduino Uno ausgetauscht wird und kein Bootloader im neuen Controller vorprogrammiert ist, muss zunächst ein Bootloader auf diesen Controller geladen werden. Dies kann mit dem *Microchip Studio* und einem externen USB-Programmiergerät erfolgen, das an den ICSP-Verbinder angeschlossen wird. Die Programmierung erfolgt auf die gleiche Weise wie für den ATmega16U2 beschrieben, allerdings muss der ICSP-Header am Rand der Platine, auf der dem USB-Anschluss gegenüberliegenden Seite, verwendet werden. Auf die gleiche Weise kann auch ein Backup eines vorhandenen Bootloaders eines ATmega328P erstellt werden.



Diese Aktionen werden in den folgenden Abschnitten näher erläutert, wobei wir es auch mit den Fuse-Einstellungen für den ATmega328P in einem Arduino Uno zu tun bekommen. Der Bootloader kann auch schnell aus der Arduino-IDE heraus wiederhergestellt werden, aber bei schon installiertem Microchip Studio funktioniert dies aufgrund eines Treiberkonflikts nicht.

Sichern der ATmega328P-Firmware mit Microchip Studio

Die Firmware des ATmega328P kann auch als Vorsichtsmaßnahme von diesem Controller auf das Arduino-Uno-Board kopiert und mit dem *Device Programming Utility* von Microchip Studio in HEX- und BIN-Dateien gesichert werden. Dies ist vor allem dann sinnvoll, wenn Sie ein Klon-Board verwenden, das möglicherweise einen anderen Bootloader enthält als der Controller auf einem echten Arduino Uno.
Schließen Sie ein geeignetes USB-Programmiergerät an den Computer an und stecken Sie dann das Kabel des Programmiergeräts mit dem 6-poligen Buchsenverbinder auf die ICSP-Stiftleiste am Rand des Arduino-Uno-Boards. Beachten Sie die Pinbelegung der ICSP-Stiftleiste für den ATmega328P, die zeigt, wo sich Pin 1 befindet. Schalten Sie den Arduino Uno entweder über USB- oder eine externe Stromversorgung ein.
Starten Sie das Microchip Studio und öffnen Sie das Device Programming Utility, indem Sie entweder auf das Icon *Device Programming* in der oberen Symbolleiste klicken oder *Tools Device Programming* im oberen Menü des Microchip Studios wählen. Wählen Sie im Dialogfeld *Device Programming* unter *Tools* das richtige USB-Programmiergerät aus, unter *Device* den *ATmega328P* und unter *Interface ISP*. Klicken Sie schließlich auf die Schaltfläche *Apply*. Um zu überprüfen, ob alles korrekt eingerichtet ist und das Microchip Studio eine Verbindung zum Zielgerät herstellen kann, klicken Sie auf die Schaltfläche *Read* unter *Device Signature*. Wenn alles in Ordnung ist, werden die Gerätesignatur und die Zielspannung gelesen und im Dialogfeld *Device Programming* angezeigt.
Klicken Sie in der linken Spalte des Dialogfelds *Device Programming* auf *Memories* und im *Memories*-Abschnitt *Flash* auf die Schaltfläche *Read...*, um zu einem Ordner zu navigieren, in dem die HEX-Datei gespeichert werden soll, die aus dem Flash-Speicher des ATmega328P ausgelesen wird. Geben Sie unten im Dialogfeld *Save As* einen beliebigen gültigen Namen Ihrer Wahl der HEX-Datei ein, die gespeichert werden soll. Klicken Sie abschließend auf die Schaltfläche *Save*. Das *Device Programming Utility* liest den Inhalt des ATmega328P-Flash-Speichers und speichert ihn in besagter HEX-Datei. Diese HEX-Datei kann jederzeit verwendet werden, um die ursprüngliche Firmware auf dem (oder einem) ATmega328P wiederherzustellen. Die HEX-Datei enthält den gesamten Inhalt des Flash-Speichers, einschließlich aller Sketches des Users, die zum Zeitpunkt der Erstellung der HEX-Datei im Flash-Speicher programmiert waren.

Die Bootloader-Firmware Optiboot

Bei aktuellen Arduino-Uno-Boards hat im ATmega328P-Mikrocontroller der Bootloader Optiboot einen älteren Bootloader ersetzt. Optiboot begnügt sich mit einem 512-Byte-Segment des Flash-Speichers, während der ältere Bootloader 2 K des Flash-Speichers beanspruchte. Das bedeutet, dass zusätzliche 1,5 K Flash-Speicher für Anwender-Sketches zur Verfügung stehen.
Der C-Quellcode von Optiboot und die HEX-Dateien befinden sich im Ordner der Arduino-IDE unter

```
arduino-1.8.13\hardware\arduino\avr\bootloaders\optiboot
```

wobei der Hauptordner einen Namen trägt, der der Version der Arduino-IDE entspricht. Die gleichen Dateien finden Sie auch auf den GitHub-Seiten von Arduino [1].
Das Optiboot-Projekt auf GitHub finden Sie unter *github.com/Optiboot/optiboot* und das informative Optiboot-Wiki unter *github.com/Optiboot/optiboot/wiki*.
Optiboot lässt beim Starten die gelbe L-LED des Arduinos blinken und beginnt dann entweder mit der Ausführung des aktuell geladenen Sketches oder wartet, wenn es von der Arduino-IDE zurückgesetzt wird, darauf, dass ein neuer Sketch geladen wird. Detaillierte Informationen finden Sie im Optiboot-Wiki [2].

Wiederherstellen des Bootloaders

Der Optiboot-Bootloader kann mit dem Microchip Studio und einem externen USB-Programmiergerät auf dem ATmega328P wiederhergestellt oder auf einen neuen leeren ATmega328P-Mikrocontroller geladen werden, wie es bereits in dem Kapitel für den ATmega16U2 beschrieben wurde. Die Verbindung vom Microchip Studio zum ATmega328P folgt der gleichen Methode, die für die Sicherung des Bootloaders aus dem Abschnitt „Sicherung der ATmega328P-Firmware mit dem Microchip Studio“ verwendet wurde.
Um den Optiboot-Bootloader in den ATmega328P auf dem Arduino Uno zu laden, schließen Sie das USB-Programmiergerät wie oben beschrieben an den ICSP-Header am Rand der Platine an. Öffnen Sie das *Device Programming Utility* im Microchip Studio und schließen Sie das Programmiergerät und den ATmega328P auf die gleiche Weise an. Klicken Sie in der linken Spalte des Dialogfelds *Device Programming* auf *Memories*, dann im Abschnitt *Flash* des Dialogfelds auf den Knopf *[...]*. Es erscheint ein Dialogfeld, mit dem Sie zu der HEX-Datei navigieren, die in den ATmega328P geladen werden soll. Die HEX-Datei zum Wiederherstellen des Bootloaders kann entweder die HEX-Datei sein, die beim Sichern des Bootloaders erstellt wurde (wie in Abschnitt 5.4.1 beschrieben) oder die HEX-Datei aus

```
arduino-1.8.13\hardware\arduino\avr\bootloaders\optiboot
```

die für den Arduino Uno *optiboot_atmega328.hex* lautet. Wählen Sie diese HEX-Datei im Dialogfeld aus und klicken Sie dann auf die Schaltfläche *Open*. Zurück im Dialogfeld *Device Programming* klicken Sie im Abschnitt *Flash* auf die Schaltfläche *Program*. Wenn alles richtig eingestellt ist, wird jetzt der Flash-Speicher mit der ausgewählten HEX-Datei programmiert.
Wenn der Bootloader auf einen neuen und leeren ATmega328P geladen wurde, müssen, bevor Sie fortfahren, die Fuses in diesem Controller so wie im nächsten Abschnitt beschrieben gesetzt werden. Wenn die Fuses bereits korrekt eingestellt sind, schließen Sie das Dialogfeld *Device Programming* in Microchip Studio und ziehen den Netzstecker des Arduino Uno. Trennen Sie das USB-Programmiergerät vom Arduino und schalten Sie den Arduino erneut ein. Wenn die Fuses korrekt eingestellt sind, können Sie den neu geladenen Bootloader testen, indem Sie die Arduino-IDE öffnen und einen Testsketch wie gewohnt über das normalen USB-Kabel auf den Arduino laden.

Fuse-Einstellungen für den ATmega328P

Kurz zusammengefasst, müssen auf einen neuen und leeren ATmega328P zunächst der Optiboot-Bootloader geladen werden und anschließend die ATmega328P-Fuses für die Verwendung im Arduino Uno korrekt eingestellt werden. Die Fuse-Werte können mit dem *Device Programming Utility* in Microchip Studio ausgelesen und eingestellt werden. Um die Fuses auszulesen und zu setzen,

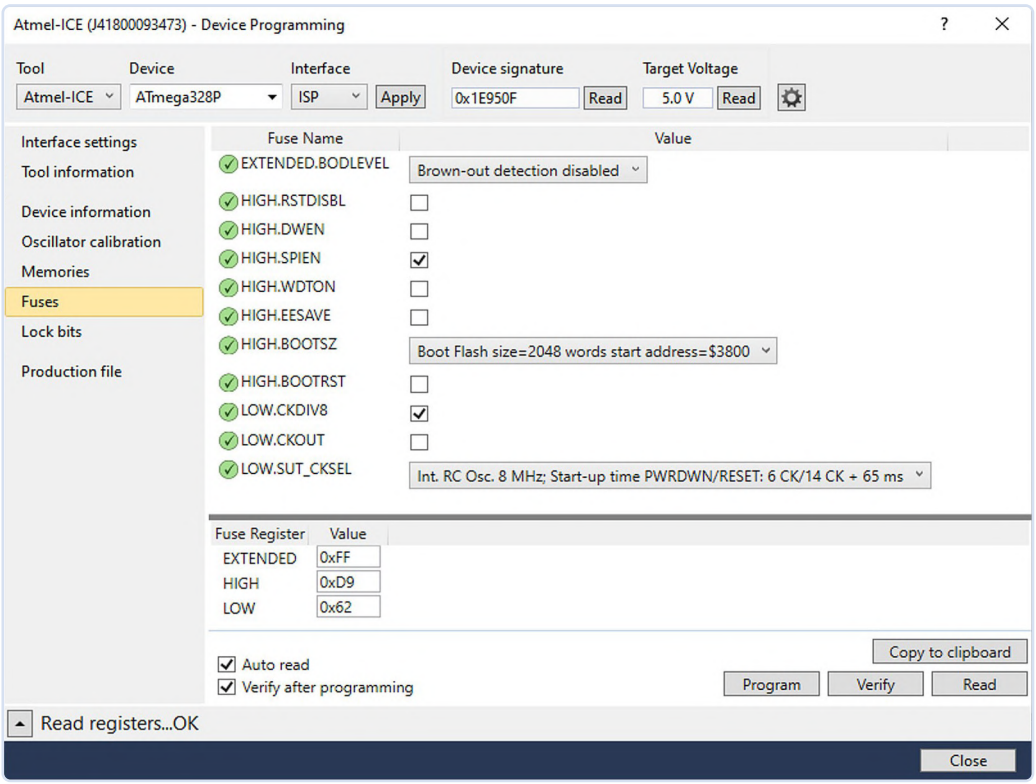


Bild 1. Voreingestellte Fuses bei einem neuen und leeren ATmega328P.

schließen Sie zunächst ein externes USB-Programmiergerät an den ICSP-Header am Rand des Arduino Uno an, öffnen im *Microchip Studio* das *Device Programming Utility*, wählen, wie bereits beschrieben, die richtigen Einstellungen für das Tool, den Controller und die Schnittstelle und klicken auf die Schaltfläche *Apply*, um die Verbindung herzustellen. Klicken Sie nun in der linken Spalte des Dialogfelds *Device Programming* auf den Eintrag *Fuses*.
Bild 1 zeigt, was Sie erwarten können, wenn Sie sich die Fuse-Werte eines neuen und leeren ATmega328P-Mikrocontrollers im Dialogfeld *Device Programming* ansehen. Die Standardwerte der Fuse-Register sind wie folgt.

EXTENDED	0xFF
HIGH	0xD9
LOW	0x62

Diese Fuses müssen auf die in **Bild 2** und **Tabelle 1** angegebenen Werte geändert und programmiert werden.

Fuse	Ändern auf:
EXTENDED.BODLEVEL	Brown-out detection at VCC=2.7V
HIGH.EESAVE	(optional, siehe nächste Seite zur Erläuterung)
HIGH.BOOTSZ	Boot Flash size=256 words; start address=$3F00
HIGH.BOOTRST	aktiviert
LOW.CKDIV8	nicht aktiviert
LOW.SUT_CKSEL	Ext. Crystal Osc. 8.0- MHz; Start up time PWRDWN/ RESET: 16K/14 CK + 65ms

Wenn diese Einstellungen vorgenommen wurden, lauten die Werte der Fuse-Register am unteren Rand des Dialogfelds:

EXTENDED	0xFD
HIGH	0xDE (EESAVE gesetzt) oder 0xD6 (EESAVE nicht gesetzt)
LOW	0xFF

Es zeigte sich, dass auf älteren Arduino Uno R3 Boards die Checkbox EESAVE aktiviert war, aber auf neuen Arduino Uno R3 Boards nicht. Wenn die EESAVE-Optionsbox (HIGH.EESAVE fuse) im Dialogfeld aktiviert ist, schützt es den EEPROM-Speicherinhalt, wenn der Chip gelöscht wird.
Nachdem Sie die Änderungen vorgenommen haben, klicken Sie auf die Schaltfläche *Program* am unteren Rand des Dialogfelds. Dadurch werden die neuen Fuse-Werte in den ATmega328P programmiert. Klicken Sie auf *OK* in der Warn-Dialogbox, um fortzufahren. Schließen Sie das Dialogfeld *Device Programming*, trennen Sie den Arduino Uno von der Stromversorgung und entfernen Sie dann das USB-Programmiergerät. Schließen Sie den Arduino Uno an den USB-Port des Computers an und laden Sie probehalber einen Sketch aus der Arduino-IDE, um sicherzugehen, dass der Bootloader funktioniert und die Fuse-Einstellungen korrekt sind.

Die RESET-EN-Brücke

Einige der AVR-USB-Programmierer haben Debugging-Fähigkeit, nämlich der AVR-Dragon (der von Microchip nicht mehr verkauft wird) und der Atmel-ICE (der ältere USB-Programmierer ersetzt). Beachten Sie, dass der AVRISP-mkII (der ebenfalls nicht mehr von Microchip verkauft wird) keine Debugging-Fähigkeiten hat, sondern nur ein reines Programmiergerät ist. Dies gilt auch für die vielen gut erhältlichen Hobby-Programmierer wie den USBasp und den USBtinyISP. Debugging-Fähigkeiten beziehen sich auf die

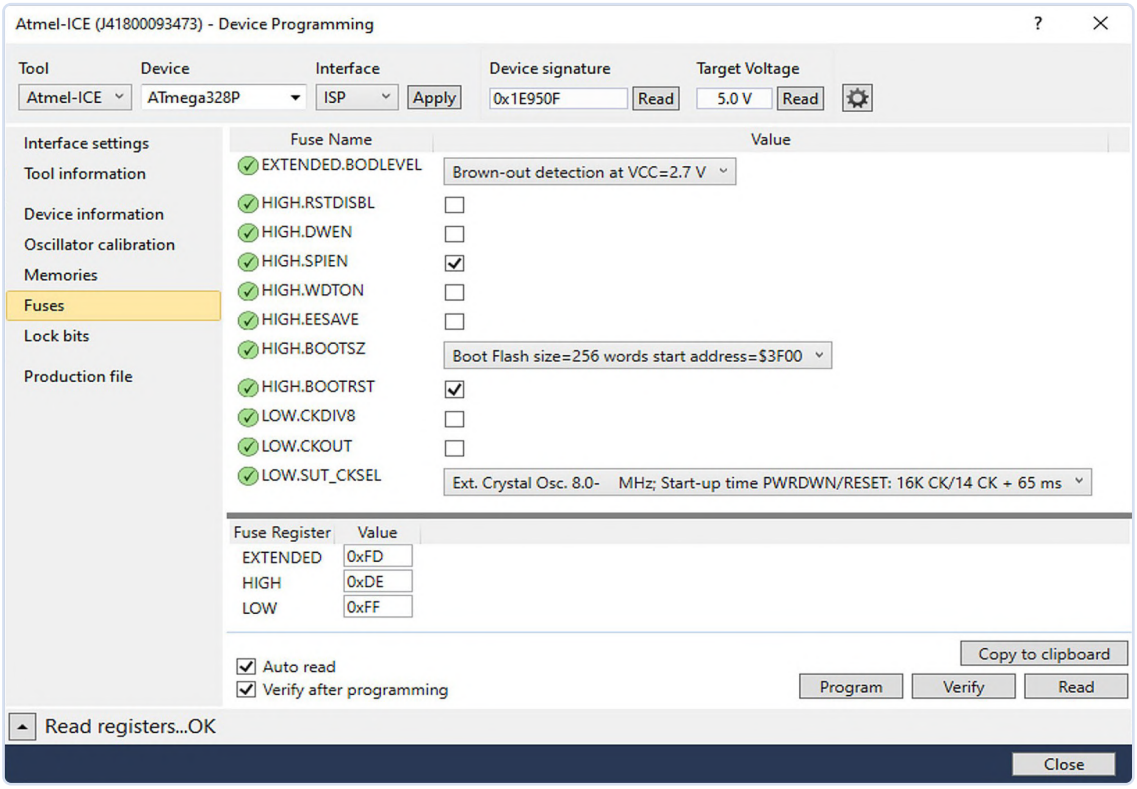


Bild 2. Fuse-Einstellungen bei einem ATmega328P für den Arduino Uno.

Tabelle 1. Fuse-Einstellungen des ATmega328P im Arduino Uno

Fuse-Name	Wert
EXTENDED.BODLEVEL	Brown-out detection level at VCC=2.7 V
HIGH.RSTDISBL	nicht aktiviert
HIGH.DWEN	nicht aktiviert
HIGH.SPIEN	☑
HIGH.WDTON	nicht aktiviert
HIGH.EESAVE	nicht aktiviert bei neuen Boards
HIGH.BOOTSZ	Boot Flash size=256 words, start address=$3F00
HIGH.BOOTRST	☑
LOW.CKDIV8	nicht aktiviert
LOW.CKOUT	nicht aktiviert
LOW.SUT_CKSEL	Ext.Crystal Osc. 8.0 MHz; Start-up PWRDWN/RESET: 16K CK/14 CK + 65 ms

Fähigkeit des USB-Programmierers/Debuggers, in Verbindung mit einer Software wie dem Microchip Studio zu arbeiten und dem Benutzer zu erlauben, auf den Haupt-Mikrocontroller des Arduino zuzugreifen und den Inhalt des Speichers und der internen Register einzusehen, sowie in Einzelschritten durch den Quellcode zu gehen und Haltepunkte einzufügen. Diese Fähigkeit ist aber in der Arduino-IDE der um Zeitpunkt der Erstellung dieses Artikels aktuellen Version 1.8.13 nicht vorgesehen.

Um die Debugging-Fähigkeiten des USB-Programmierers/Debuggers zu nutzen, wird dieser in den ICSP-Header eingesteckt, wie es normalerweise zum Programmieren des Flash-Speichers des Ziel-AVR-Controllers oder zum Auslesen von Fuse-Werten gemacht wird. Ein C- oder C++-Programm kann dann mit Microchip Studio in den AVR geladen werden. Microchip Studio kann dann zum Debuggen des Ziel-AVRs verwendet werden.

Das Debugging-System auf dem ATmega328P-Mikrocontroller ist als *debugWIRE* bekannt und ermöglicht es der Debugging-Software, sich über den ICSP-Header mit dem Ziel-Mikrocontroller zu verbinden, und zwar mit einem USB-Programmiergerät, das über Debugging-Fähigkeiten verfügt. Damit *debugWIRE* auf dem ATmega328P auf Arduino-Uno-Boards funktioniert, muss die Reset-Schaltung vom Reset-Pin getrennt werden. Arduino-Uno-Boards besitzen eine Lötverbindung mit der Bezeichnung RESET-EN, die aus zwei Lötpads besteht, die durch eine Leiterbahn verbunden sind. Die Leiterbahn zwischen den Lötpads muss durchtrennt werden, um

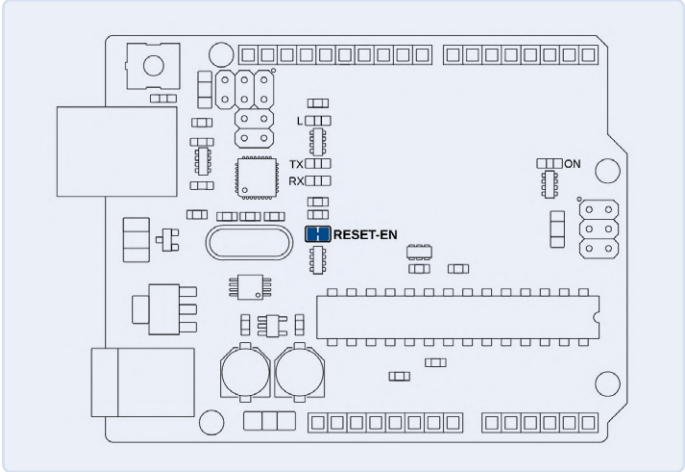
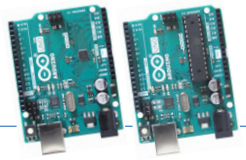


Bild 3. Lage der Lötbrücke RESET-EN bei einem Arduino Uno.



für die Verwendung von debugWIRE einen Teil der Reset-Schaltung vom Reset-Pin zu trennen. Die Verbindung kann später durch Zusammenlöten der beiden Lötpads wieder hergestellt werden. **Bild 3** zeigt die Lage der RESET-EN-Lötpads auf dem Arduino Uno. Diese Funktion wird nur von fortgeschrittenen Arduino-Benutzern verwendet, die das Board normalerweise mit einfachem C und Microchip Studio programmieren. Wer daran interessiert ist, die Sprache C zu erlernen, um Arduino-Boards zu programmieren, findet im Elektor-Buch *C Programming with Arduino* (ISBN 978-1-907920-46-2) des gleichen Autors ausführliche Informationen. Dieses Buch enthält auch ein Kapitel über das Debugging von Arduino-Uno- und Mega2560-Boards mit dem Atmel Studio, das inzwischen durch das Microchip Studio ersetzt wurde.

Alternative Firmware-Programmiermethoden

Alternative Methoden zur Programmierung des Bootloaders auf einem Arduino Uno finden Sie auf dem Arduino-Playground unter dem Abschnitt Bootloader [3]. Dort wird auch die Verwendung alternativer Programmiergeräte und eines zweiten Arduinos als Programmiergerät beschrieben, um einen Ziel-Arduino zu programmieren.

Der Einsatz eines Arduinos als In-System Programmer (ISP) wird ebenfalls auf der Arduino-Website beschrieben [4]. Kapitel 2, Abschnitt 2.8 des Buches enthält weitere Informationen zum Laden von Sketches in den Arduino Uno mit einem externen Programmiergerät und der Arduino-IDE. Im gleichen Abschnitt geht es auch um die Wiederherstellung des Bootloaders im Haupt-Mikrocontroller mit einem externen Programmiergerät und der Arduino-IDE. Schließlich finden Sie unter [5] weitere Informationen zur DFU-Programmierung des ATmega16U2.

210345-02

Anmerkung der Redaktion: *Dieser Artikel ist ein Auszug aus dem Buch* Ultimate Arduino Uno Hardware Manual, *neu formatiert und leicht bearbeitet, um den Standards und dem Seitenlayout der Zeitschrift 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 ursprünglichen Buch beziehen. Der Autor und der Herausgeber haben jedoch ihr Bestes getan, um solche Fälle zu vermeiden, und helfen bei Rückfragen gerne weiter. Kontaktinformationen finden Sie unter* Fragen oder Kommentare?

Haben Sie Fragen oder Kommentare?
Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Schicken Sie eine E-Mail an den Autor unter warwsmi@axxess.co.za oder an die Redaktion unter editor@elektormagazine.com.

Ein Beitrag von
Text und Grafiken: **Warwick A. Smith**
Redaktion: **Jan Buiting**
Übersetzung: **Rolf Gerstendorf**
Layout: **Sylvia Sopamena**

WEBLINKS

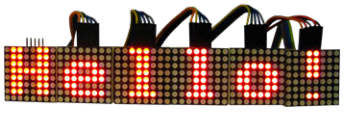
[1] ArduinoCore-avr: https://www.github.com/arduino/ArduinoCore-avr/tree/master/bootloaders/optiboot
[2] Optiboot-Wiki : https://www.github.com/Optiboot/optiboot/wiki/HowOptibootWorks
[3] Arduino-Bootloader: https://playground.Arduino.cc/Main/ArduinoCoreHardware/#Bootloader
[4] Arduino als ISP: https://www.arduino.cc/en/Tutorial/BuiltInExamples/ArduinoISP
[5] Device Firmware Update des ATmega16U2: https://www.arduino.cc/en/Hacking/DFUProgramming8U2

MicroPython für den ESP32 und Co.

Teil 2. Matrix-Displays einfach ansteuern

Von **Dr. Günter Spanner** (Deutschland)

Nachdem im ersten Artikel der Serie die Entwicklungsoberfläche und einfache Befehle im Vordergrund standen, sollen im zweiten Teil Praxisanwendungen vorgestellt werden. Unter Verwendung geeigneter Programmbibliotheken wird so etwa der Aufbau und die Programmierung großflächiger LED-Punktmatrixanzeigen zum Kinderspiel. Neben Lauftexten können diese auch zur Anzeige der Uhrzeit oder der lokalen Temperatur dienen.



Das Mini-Display aus dem ersten Artikel [1] war eher für kleinere Geräte geeignet. In diesem Beitrag soll gezeigt werden, wie man mit Python-Kenntnissen auch eine großflächige Anzeige realisieren kann. Das Mittel der Wahl sind in diesem Fall LED-Matrixelemente mit jeweils 8 x 8 = 64 einzelnen LEDs. Durch Aneinanderreihen mehrerer Einheiten können so nahezu unbegrenzt große Displays aufgebaut werden. Diese eigenen sich beispielsweise als „Live-Ticker“ für Börsenkurse oder auch als Anzeigen für Spielstände auf dem heimischen Sportplatz. Mit wenigen Python-Anweisungen entstehen so großflächige Zeit- oder Temperaturanzeigen, die bei vielen Gelegenheiten sehr werbewirksam zum Einsatz kommen können.

Punktmatrix-Displays

Punktmatrix-Anzeigen erlauben neben der Darstellung von Ziffern und Buchstaben auch die Ausgabe von beliebigen Symbolen und einfachen Grafiken. Damit sind sie den klassischen Sieben-Segment-Displays überlegen, die nur die Ausgabe der Ziffern von 0 bis 9 erlauben. Zudem sind Sieben-Segment-Anzeigen auf die einmal festgelegten Größen beschränkt, Punktmatrizen dagegen beliebig skalierbar.

Darüber hinaus sind im Vergleich zu OLED-Displays wesentlich größere Helligkeiten erzielbar. Mit LED-Matrizen können Anzeigenflächen mit mehreren Metern Länge und Breite erreicht werden. Diese Display-Variante wird daher häufig auch für großflächige Anzeige- oder Werbetafeln an belebten Plätzen, Bahnhöfen oder Flughäfen, in öffentlichen Bussen und Zügen und so weiter eingesetzt. In den Börsensälen der Welt zeigen sie als sogenannte „Live-Ticker“ die aktuellen Wertpapierkurse an.

Bild 1. LED-Punkt-Matrizen sind in Asien sehr beliebt.

Im asiatischen Raum sind Dot-Matrizen übrigens besonders beliebt, da sich damit auch fernöstliche Schriftzeichen problemlos darstellen lassen, wie **Bild 1** zeigt.

Ansteuerung

LED-Matrizen mit 5 x 7 = 35 LEDs sind weit verbreitet. Aber auch die verschiedensten anderen Bauformen sind erhältlich. Häufig sind Elemente mit 8 x 8 LEDs anzutreffen. Die direkte Ansteuerung von 8 x 8 Punkten würde 65 Leitungen erfordern, da 64 Anodenanschlüsse und eine gemeinsame Kathode anzuschließen wären. Bei einer Matrix kommt man mit wesentlich weniger Verbindungen aus. Hier sind jeweils 8 LEDs in einer Spalte und einer Zeile zusammengeschaltet. So sind nur noch 8 + 8 = 16 Verbindungen erforderlich. **Bild 2** zeigt das Prinzip für eine 3x4-Matrix. Statt 13 Anschlüssen für eine Einzelansteuerung der LEDs sind hier nur 7 Verbindungen notwendig.

Um zum Beispiel die 2. LED in der 2. Zeile in einer Matrix zu aktivieren, müssen bis auf den zweiten alle Zeilenanschlüsse auf HIGH gelegt werden. Der zweite Anschluss dagegen muss GND-Potential führen. Bei den Spalten darf ausschließlich in der zweiten Spalte High-Potential anliegen.

Die direkte Ansteuerung von Punktmatrizen bindet einen großen Teil der Ressourcen des Controllers. Sollen zusätzlich noch Sensorwerte erfasst oder Aktoren angesteuert werden, stößt selbst ein leistungsfähiger ESP32-Controller rasch an seine Grenzen. Die Ansteuerung von größeren Displays mit hundert oder mehr LEDs würde ebenfalls schnell zum Problem werden, einerseits wegen der erforderlichen Rechenleistung, andererseits aufgrund der beschränkten Anzahl der verfügbaren Pins an einem Controller. Zudem würde sich hier auch die relativ geringe Ausführungsgeschwindigkeit von Python-Code negativ auswirken.

Es ist daher ratsam, kostengünstige Displaytreiber einzusetzen, wie etwa den MAX7219. Diese Bausteine verfügen über eine SPI-kompatible Schnittstelle und können so mit lediglich drei digitalen Pins Displays mit bis zu 8 x 8 = 64 Matrixelementen ansteuern. Das SPI-Interface (**S**erial **P**eripheral **I**nterface) hat weite Verbreitung gefunden und kommt insbesondere in der Unterhaltungselektronik häufig zum Einsatz. Das Bussystem und insbesondere seine Verwendung unter MicroPython sind in einem Elektor-Buch des Autors [2] detailliert beschrieben.

Die Treiber sind zusammen mit dem eigentlichen Matrix-Element als komplette Module verfügbar. **Bild 3** zeigt ein Beispiel.

Der Anschluss der Treiberbausteine an den ESP32 ist sehr einfach. Lediglich die drei SPI-Pins sind mit dem Controllerboard zu verbinden:

- MAX7219_data (DIN) Port D02
- MAX7219_load (CS) Port D05
- MAX7219_clock (CLK) Port D04
- MAX7219_GND ESP32 GND

Aufgrund des relativ hohen Stromverbrauchs empfiehlt es sich, die Spannungsversorgung extern zur Verfügung zu stellen. Dabei ist darauf zu achten, dass die Versorgungsspannung der Module bei etwa 3,3 V bis 4 V liegen sollte, um die Kompatibilität mit dem ESP32 zu gewährleisten. Auf diese Weise erhalten die Module einerseits noch die laut Datenblatt minimal zulässige Versorgungsspannung.

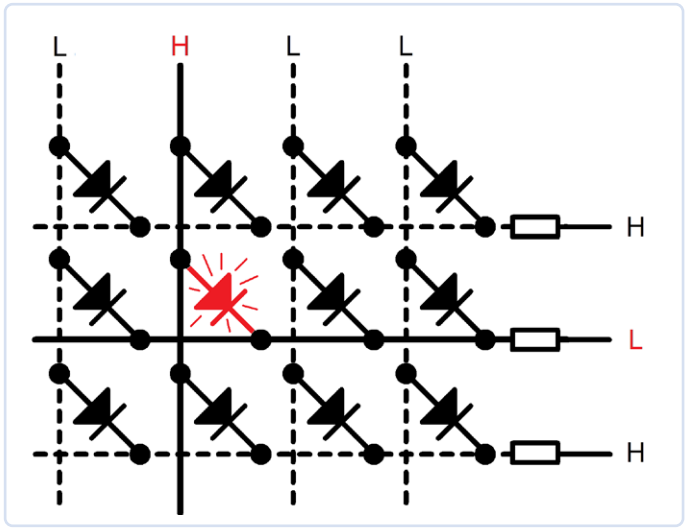


Bild 2. Prinzip der Punktmatrix-Ansteuerung.

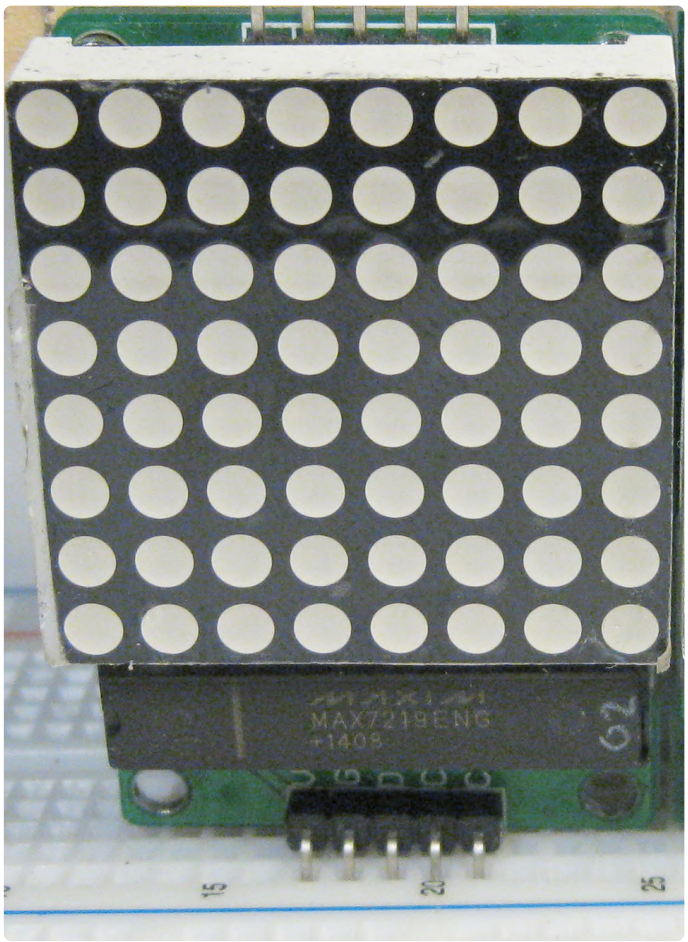


Bild 3. Einzelnes LED-Matrix-Modul.

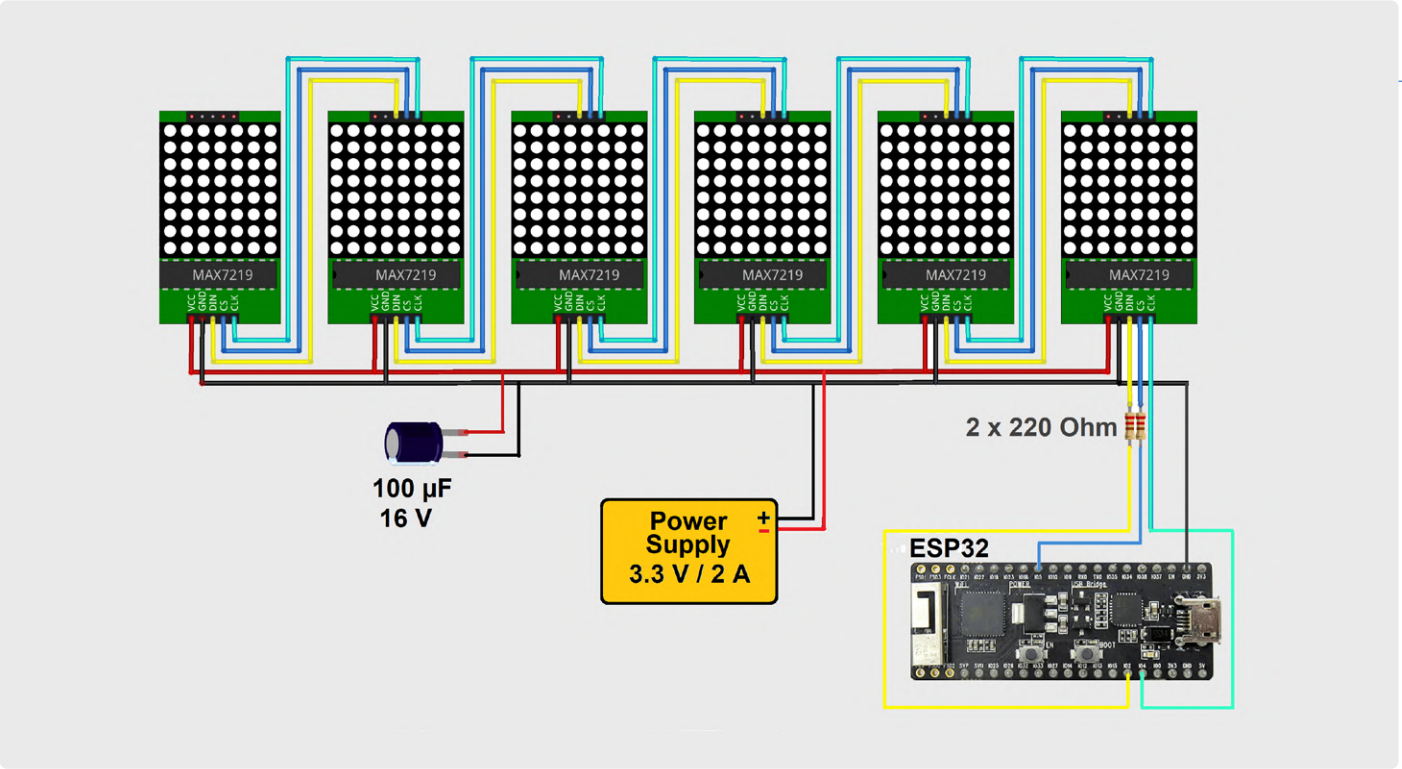


Bild 4. LED-Matrix-Modul-Array.

Andererseits wird eine Überlastung der Controller-Eingänge vermieden. Alternativ kann auch ein bidirektionaler 3,3/5-V-Pegelwandler eingesetzt werden.
Zudem sind ausreichend Abblockkondensatoren vorzusehen. Unerlässlich sind auch niederohmige VCC- und Masseleitungen. Werden diese Hinweise berücksichtigt, können nahezu beliebig große Displays aufgebaut werden. Der Controller wird kaum mehr belastet, da nur noch einzelne Kommandos über den SPI-Bus gesendet werden müssen. Zudem bleiben noch ausreichend Pins frei, um externe Sensoren oder andere Peripherie zu betreiben. Der Realisierung von großflächigen Anzeigen zu Werbezwecken oder bei Sportveranstaltungen steht damit nichts mehr im Weg.
Bild 4 zeigt das vollständige Schaltbild zum Aufbau eines 6-stelligen Matrix-Displays mit einem ESP32-Controller-Modul.

Mit der passenden Library zum Erfolg

Ein Link für eine geeignete Python-Library zur Ansteuerung der Maxim-ICs findet sich im Downloadpaket [4]. Nachdem die Treiber-Datei *Max7219.py* auf den Controller geladen wurde, stehen die folgenden Anweisungen zur Verfügung:

```
spi = SPI(1, baudrate=10000000, polarity=1, phase=0,
          sck=Pin(CLK), mosi=Pin(DIN))
ss = Pin(CS, Pin.OUT)
```

Für die oben angegebene Pinbelegung ist
CLK = 4,
DIN = 2,
CS = 5
zu setzen. Dann kann ein Display-Objekt erzeugt werden:

```
display = max7219.Matrix8x8(spi, ss, MN)
```

wobei für MN die Anzahl der verwendeten Matrixelemente einzusetzen ist. Anschließend können mit den Befehlen in **Listing 1** Texte und Grafiken entworfen werden.
Damit lassen sich werbewirksame Anzeigen erstellen, die auch noch aus mehreren Metern Entfernung gut lesbar sind.

LED-Matrix in Aktion

Der folgende Code zeigt ein Anwendungsbeispiel für ein Display

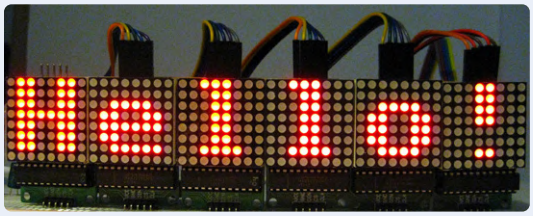


Bild 5. Punktmatrix-Display mit sechs 8x8-Matrix-Elementen.

Bild 6. Punktmatrix-Display mit zwölf Matrix-Elementen.

Listing 1. Display-Befehle

```
display.pixel(x,y,1)          # set a pixel at the coordinate x, y
display.pixel(x,y,0)          # Delete a pixel at the coordinate x, y
display.hline(x,y,l,1)        # horizontal line from x, y - length l
display.vline(x,y,l,1)        # vertical line from x, y - length l
display.line(a,b,c,d,1)       # line from a, b to c, d
display.rect(a,b,c,d,1)       # rectangle with corners a, b, c, d
display.text('Text',x,y,1)    # text at position x, y
display.scroll(x,0)           # scroll by x pixels
display.show()               # Refresh display
```

mit sechs 8x8-Matrixelementen:

```
# LED_matrix_test.py

import max7219
from machine import Pin, SPI

spi = SPI(1, baudrate=10000000, polarity=1, phase=0,
             sck=pin(4), mosi=pin(2))
ss = Pin(5, Pin.OUT)

display = max7219.Matrix8x8(spi, ss, 6)
display.text('Python',0,0,1)
display.show()
```

Nach dem Laden des Programms auf den ESP-Controller wird der Text auf dem Display ausgegeben (**Bild 5**). Die Anzeige kann durch weitere Matrixmodule problemlos erweitert werden. **Bild 6** zeigt ein Array mit 12 Elementen. Es sind jedoch nicht nur statische Anzeigen und Texte darstellbar. Auch bewegte Grafiken sind in Python problemlos umzusetzen. Der nächste Abschnitt zeigt ein entsprechendes Beispiel.

Laufschriften und Börsenticker

Über die Scroll-Funktion können Laufschriften und Ähnliches erzeugt werden. Damit lassen sich auch auf kleinen beziehungsweise kurzen Displays längere Texte ausgeben. Das Programm in **Listing 2** lässt den Schriftzug „Python“ von rechts nach links über ein Dot-Matrix-Display laufen. Die einzelnen Buchstaben werden dabei über

```
display.text('_',40,0,1)
```

am rechten Rand der Anzeige erzeugt. Dann werden diese über die Funktion `moveLeft` um den Wert `pixelDistance` nach links verschoben:

```
for i in range(8):
    display.scroll(-1,0)
    sleep(speedFactor)
```

Die Ablaufgeschwindigkeit kann über den Wert `speedFactor` verändert werden. In einem YouTube-Video [3] ist das Display live in Aktion zu sehen.

Listing 2. Laufschrift

```
# LED_matrix_ticker.py

import max7219
from machine import Pin, SPI
from time import sleep

spi = SPI(1,baudrate=10000000, polarity=1, phase=0,
    sck=Pin(4), mosi=Pin(2))
ss = Pin(5,Pin.OUT)
speedFactor=0.05
pixelDistance=7
display = max7219.Matrix8x8(spi, ss, 6)

def moveLeft(Pixel):
    for i in range(Pixel):
        display.scroll(-1,0)
        sleep(speedFactor)
        display.show()

while True:
    display.text('P',40,0,1)
    moveLeft(pixelDistance)
    display.text('y',40,0,1)
    moveLeft(pixelDistance)
    display.text('t',40,0,1)
    moveLeft(pixelDistance)
    display.text('h',40,0,1)
    moveLeft(pixelDistance)
    display.text('o',40,0,1)
    moveLeft(pixelDistance)
    display.text('n',40,0,1)
    moveLeft(pixelDistance)
    display.text(' ',40,0,1)
    moveLeft(pixelDistance)
```

Temperaturanzeige im Großformat

Viele Apotheken, Banken oder Einzelhändler versuchen die Aufmerksamkeit potentieller Kunden auf sich zu lenken, indem sie mit großflächigen Temperatur- oder Zeitanzeigen werben. Aber auch bei Sportveranstaltungen, Messen, Ausstellungen oder für

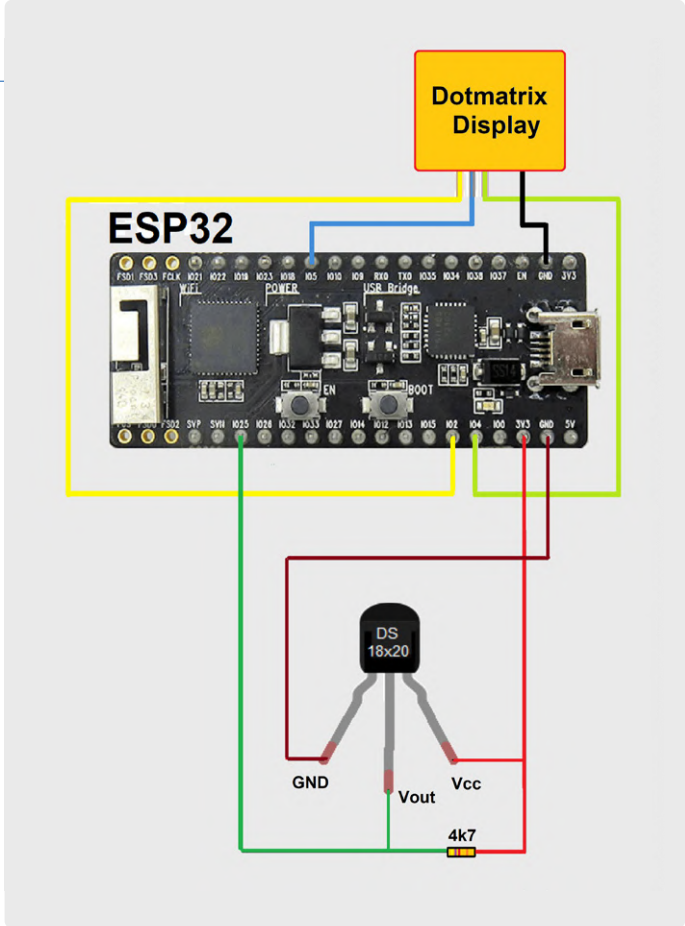


Bild 7. DS18x20-Thermosensor am ESP32-Controller.

FabLabs sind derartige Displays immer ein „Hingucker“. Um etwa eine Temperaturanzeige zu realisieren, muss man den oben vorgestellten Aufbau nur noch um einen entsprechenden Sensor erweitern. Eine besonders geeigneter Variante ist hier der DS18x20 von Maxim Integrated (ehemals Dallas). Mit der passenden Python-Library ist das Auslesen des Sensors problemlos möglich.
Diese Sensoren kommunizieren über den One-Wire-Bus und belegen damit nur einen einzigen I/O-Pin des ESP32. Zudem ist für diese Sensorreihe eine fertige Python-Library verfügbar. Zusätzlich zum Sensor selbst ist lediglich ein Pull-up-Widerstand von 4,7 kΩ erforderlich. Bei Verwendung des ESP32-internen Pull-ups kann sogar dieser entfallen. Der Sensor verfügt über die folgenden Leistungsmerkmale:

- Versorgungsspannung: 3,0 V bis 5,5 V
- Temperaturbereich: 55 °C … + 125 °C
- Messgenauigkeit: ± 0,5 °C (- 10 °C bis + 85 °C)
- Auflösung: 9 Bit, entsprechend ca. 1/10 °C
- Messperiodendauer: 750 ms (max.)

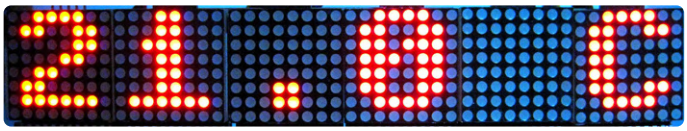


Bild 8. Ein Mega-Temperatur-Display.

Über One-Wire-Bus kann nicht nur ein einzelner Sensor pro Pin ausgewertet werden. Durch das spezielle Protokoll des One-Wire-Bussystems lassen sich nahezu beliebig viele Temperatursensoren parallel über einen einzigen Controller-Pin abfragen. Im Folgenden soll allerdings nur ein einzelner Sensor verwendet werden. **Bild 7** zeigt den Anschluss des DS18x20 an den ESP32. Ein Auswertungsprogramm für den DS18x20 zeigt die erfassten Temperaturwerte auf der Konsole an:

```
# DS18x20_TempSens_demo.py

from machine import pin
import onewire
import ds18x20
import time

ow = onewire.OneWire(Pin(25))  # init one wire bus
ow.scan()
ds=ds18x20.DS18X20(ow) # create ds18x20 object

while True:
    units=ds.scan() # scan for ds18x20 units
    ds.convert_temp() # convert temperature
    for unit in units:
        print(ds.read_temp(unit)) #display
    time.sleep(1)
    print()
```

Die Module zum Auslesen des Sensors sind wieder standardmäßig in der MicroPython-Firmware verfügbar. Die Erweiterung des Programms auf die LED-Matrixanzeige ist mit wenigen Zeilen erledigt, siehe **Listing 3**. **Bild 8** zeigt das Ergebnis. Weitere Details zum Anschluss verschiedener Sensoren unter anderem für Lichtstärke, Luftfeuchtigkeit oder Magnetfelder sind im Buch [2] zu finden.

Animierte Grafiken

Neben Buchstaben und numerischen Daten lassen sich auf der LED-Matrix auch Grafiken und sogar Animationen präsentieren. Das Programm in **Listing 4** zaubert eine beliebige Pixelgrafik auf das Display.
Die Grafik wird dabei über die Bitmap `icon` erzeugt. Punkte, die später leuchten sollen, sind mit einer „1“ zu markieren. Dunkle Punkte erhalten eine „0“:

```
[0, 0, 0, 0, 0, 0, 0, 0],
[0, 1, 1, 0, 0, 1, 1, 0],
[0, 1, 1, 0, 0, 1, 1, 0],
[0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 1, 1, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0],
[0, 1, 0, 0, 0, 0, 1, 0],
[0, 0, 1, 1, 1, 1, 0, 0],
```

Listing 3. Temperatur-Anzeige

```
# DS18x20_to_LED_matrix.py

import max7219
from machine import Pin, SPI
import onewire
import ds18x20
import time

# pin assignment for LED matrix
spi = SPI(1, baudrate=10000000, polarity=1, phase=0, sck=Pin(4), mosi=Pin(2))
ss = Pin(5, Pin.OUT)

ow = onewire.OneWire(Pin(25))                    # init one wire bus
ow.scan()
ds=ds18x20.DS18X20(ow)                           # create ds18x20 object

while True:
    units=ds.scan()                              # scan for ds18x20 units
    ds.convert_temp()                            # convert temperature
    for unit in units:
        T=ds.read_temp(unit)
        output="T = {:5.1f} °C"                  # generate formated string
        print(output.format(T))

        output=str(T)
        output="{:4.1f} C"                       # generate formated string for LED matrix
        display = max7219.Matrix8x8(spi, ss, 6)
        display.text(output.format(T),0,0,1)     # write temperature to LED matrix
        display.show()
    time.sleep(1)
```

Auf diese Weise lassen sich beliebige Icons erzeugen. Die Enumerate-Funktion sorgt für die Umsetzung der Bitmap in eine anzeigefähige Grafik:

```
for y, row in enumerate(icon):
    for x, c in enumerate(row):
        display.pixel(x, y, c)
```

Damit ergibt sich **Bild 9** auf der LED-Matrix. Mit der Scroll-Funktion in der Hauptschleife kann das so erzeugte Bild dann über die Anzeigeeinheit hinweg bewegt werden (siehe auch im YouTube-Video [3]).

Zusammenfassung und Ausblick

Nachdem im ersten Artikel der Serie [1] die elementaren Befehle vorgestellt wurden, konnten diese nun im zweiten Teil in verschiedenen Praxisanwendungen eingesetzt werden. Mächtige Libraries machen es möglich, bereits mit einigen wenigen Programmzeilen eindrucksvolle Projekte umzusetzen. Dies bestätigt die bemerkenswerte Leistungsfähigkeit von MicroPython auch für Controller-Anwendungen. Weitere Informationen und viele Praxisprojekte sind im Buch „MicroPython für Mikrocontroller“ [2] zu finden. Dort werden unter anderem auch die Ansteuerung von Servos, die drahtlose Datenübertragung via RFID, das MQTT-Protokoll sowie die Übertragung von Sensorwerten in das Internet ausführlich behandelt.

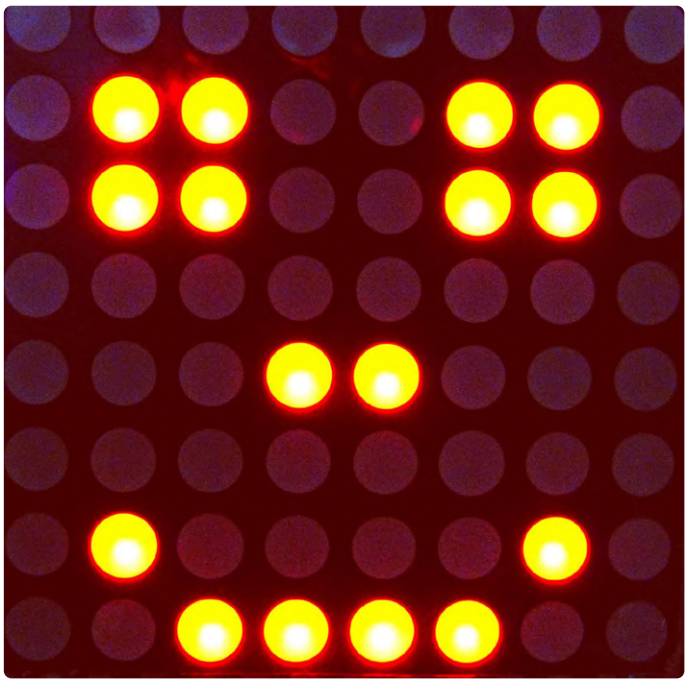


Bild 9. Selbstdefinierte Pixelgraphik auf dem Dot-Matrix-Display.

Listing 4. Pixelgrafik

```
#LED_matrix_icon.py

import max7219
from machine import Pin, SPI
from time import sleep

# sck -> serial clock - CLK; mosi -> master out slave in - DIN
# ss -> chip select - CS
spi=SPI(1,baudrate=10000000,polarity=1,phase=0,sck=Pin(4),mosi=Pin(2))
ss=Pin(5,Pin.OUT)

display=max7219.Matrix8x8(spi,ss,6)
display.brightness(0)

icon = [
    [0, 0, 0, 0, 0, 0, 0, 0],
    [0, 1, 1, 0, 0, 1, 1, 0],
    [0, 1, 1, 0, 0, 1, 1, 0],
    [0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 1, 1, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0],
    [0, 1, 0, 0, 0, 0, 1, 0],
    [0, 0, 1, 1, 1, 1, 0, 0],
]

for y, row in enumerate(icon):
    for x, c in enumerate(row):
        display.pixel(x, y, c)

display.show()

while True:
    for i in range(40):      #move square left to right
        display.show()
        display.scroll(1, 0)
        sleep(0.05)

    for i in range(40):      #move square right to left
        display.show()
        display.scroll(-1, 0)
        sleep(0.05)
```

Zudem werden über die Kombination von Python mit dem rasch wachsenden Feld des Maschinellen Lernens und der Künstlichen Intelligenz in naher Zukunft Anwendungen ermöglicht, die bislang undenkbar waren. Neue Controller-Generationen wie etwa der Kendryte K210 erlauben beispielsweise bereits die direkte Auswertung von Audio- und Kamerasignalen. Zusammen mit Python-basierten Bildverarbeitungsalgorithmen stehen dem Entwickler so vielfältige und zukunftsweisende Wege offen.

210179-B-02

Sie haben Fragen oder Kommentare?
Sie haben technische Fragen oder Kommentare zu diesem Artikel? Schreiben Sie uns unter redaktion@elektor.de!

Ein Beitrag von
Text und Bilder: **Dr. Günter Spanner**
Redaktion: **Jens Nickel**
Layout: **Harmen Heida**

WEBLINKS

[1] **Dr. Günter Spanner, „MicroPython für den ESP32 und Co.“, Elektor 7-8/2021**: www.elektormagazine.de/200179-02
[2] **Dr. Günter Spanner, „MicroPython für Mikrocontroller“, Elektor Verlag**: www.elektor.de/micropython-fur-mikrocontroller
[3] **Demo-Video auf YouTube**: www.youtube.com/watch?v=5uligjyKMEk
[4] **Software-Download**: www.elektormagazine.de/210179-B-02

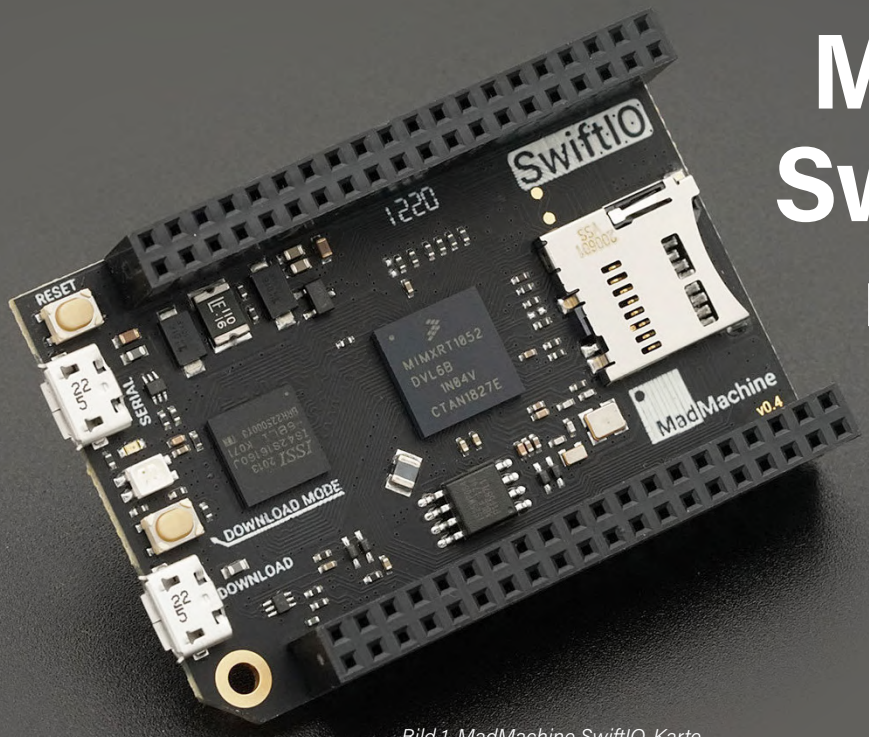


Bild 1. MadMachine SwiftIO-Karte

MadMachine SwiftIO-Karte

Moderne Sprache trifft moderne Hardware

Von **Mathias Claußen** (Elektor)

Die auf Apple-Geräten etablierte Programmiersprache Swift kann nun auch auf einer MCU eingesetzt werden. Wir werfen einen Blick auf das SwiftIO-Board, das einen ARM Cortex-M7-Prozessor und 32 MB RAM enthält.

Die MadMachine SwiftIO ist ein Entwicklungsboard im Beagle-Bone-Black-Formfaktor. Was das Board für mich so attraktiv macht, ist die Tatsache, dass hier eine interessante Hardware mit einem ungewöhnlichen Software-Framework kombiniert ist. Der Code kann in Swift geschrieben werden, einer Programmiersprache, die auf Apple-Geräten gut etabliert ist.

Die Hardware

Während Swift auf die Apple-Hardware abzielt, soll das SwiftIO-Board (**Bild 1**) von MadMachine die Sprache Swift in die Embedded-Welt bringen.

- NXP i.MX RT1052 Crossover-Prozessor mit Arm® Cortex®-M7-Kern (600 MHz)
- 8 MB externer QSPI-Flash
- 32 MB SDRAM
- Micro-SD-Kartensteckplatz, unterstützt Standard- und High-Capacity-SD-Karten

Mit dem i.MX RT1052 erhält das Board eine kräftige MCU, deren Eigenschaften in **Bild 2** zu sehen sind. Das Board bietet zusätzlich:

- 46 GPIO (general-purpose I/O)
- 12 ADC-Kanäle (12 bit)
- 14 PWM-Kanäle
- 4 UART
- 2 CAN (CAN 2.0B)
- 2 I²C (3,4 MHz max. Takt)
- 2D-Grafik-Engine
- RGB-Display-Ausgang (1366 x 768 WXGA max)
- Kameraeingang (CSI 8,16,24 Bit)

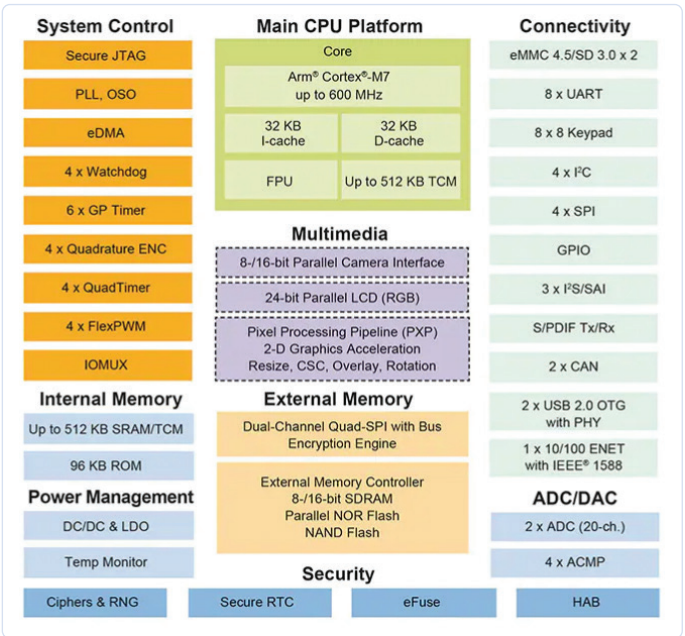


Bild 2. Übersicht i.MX 1052 (Quelle: NXP)

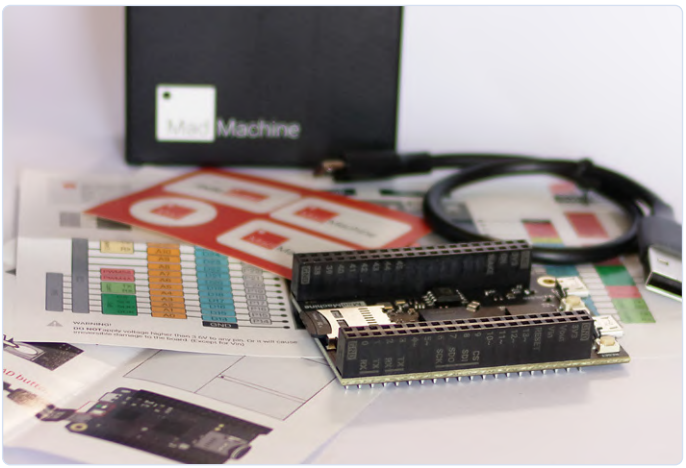


Bild 3. Inhalt der Box

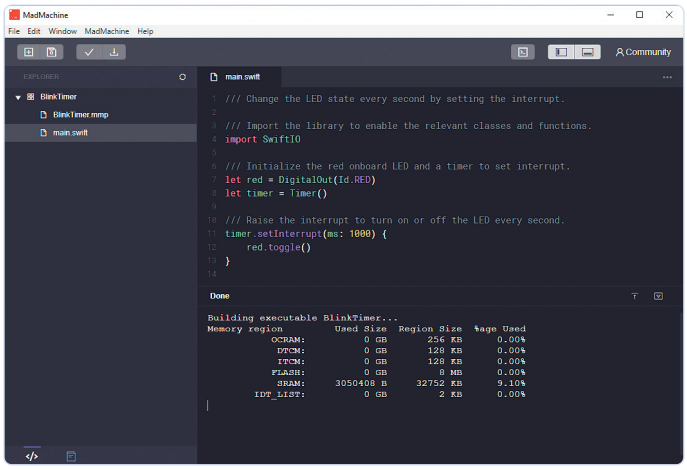


Bild 4. MadMachine-IDE

Dies ist keine vollständige Auflistung der Funktionen, soll aber einen guten Eindruck über die Hardware vermitteln, die Sie erhalten. Auch das Paket ist in der Größe kompakt. Sie erhalten ein USB-Kabel, einen Satz Aufkleber und das SwiftIO-Board selbst mit einer eingesteckten SD-Karte (**Bild 3**). Alles, was Sie brauchen, um loszulegen.

Die IDE

Wenn es um die Softwareentwicklung geht, liefert MadMachine eine IDE für SwiftIO. Diese IDE kann auf deren Website heruntergeladen werden und sieht im Prinzip wie in **Bild 4** aus. Die IDE ist für Windows und macOS verfügbar, was eine gute Nachricht für unsere Apple-Nutzer ist.

Wenn Ihnen die IDE nicht gefällt, können Sie die mitgelieferten CLI-Tools für dieses Board verwenden oder diese mit XCode oder Visual Studio Code für Ihr nächstes Projekt kombinieren. Ein Tutorial zur Verwendung des CLI-Tools wird von MadMachines bereitgestellt.

Nicht neu flashen, nur kopieren

Das Board enthält im angeschlossenen SPI-Flash einen Bootloader, der Ihr Programm beim Booten in das SDRAM kopiert und von dort aus ausführt. Das macht die Neuprogrammierung des Boards einfach. Der Onboard-Programmierer ermöglicht das Schreiben Ihres Codes auf die angeschlossene Micro-SDCard, die als Massenspeichergerät angezeigt wird. Außerdem können Sie die SDCard einfach entfernen und Ihre neue Firmware direkt darauf schreiben.

Das SwiftIO Framework

Was meine Aufmerksamkeit erregte, war die verwendete Software. Das SwiftIO-Framework ist in **Bild 5** zu sehen und nutzt das darunter liegende Zephyr-RTOS. Der Zugriff auf die gesamte Low-Level-Hardware wie GPIOs, Analogfunktionen, PWM, I²C und so weiter ist abstrahiert, so dass es einfach sein sollte, unsere eigene Software zu schreiben, ohne sich mit allen Details der i.MX 1052-MCU auseinandersetzen zu müssen. Außerdem kümmert sich die Software um den SDRAM-Zugriff und die Initialisierung beim SPI-Flash-Setup. Als Programmiersprache für dieses Board wird Swift verwendet, das auf Apple-Systemen weit verbreitet ist und hauptsächlich im Apple-Ökosystem zu finden ist. Swift ist aber nicht auf Apple-Systeme beschränkt und kann auch für Linux und seit neuestem auch für Windows zur Entwicklung von Anwendungen verwendet werden. Da Swift eine Programmiersprache ist, die moderne Elemente und Konzepte bietet, ist die Verwendung auf einer MCU eine perfekte Ergänzung, um Ihr Swift-Wissen zu erweitern.

Wenn Sie sich fragen, wie SwiftIO-Code aussieht: **Listing 1** bringt eine LED zum Blinken, **Listing 2** demonstriert, wie man PWM verwendet, um die Helligkeit einer LED zu steuern. Weitere kompliziertere Beispiele stehen auch zur Verfügung, so dass Sie aus dem Stand beginnen können, SwiftIO zu erforschen.

Support und weitere Dokumentation

Für dieses Board können Sie ganz einfach Support bekommen: Gehen Sie einfach auf den Discord-Server von MadMachines und stellen Sie Ihre Fragen in den entsprechenden Chats. Wie auf jeder Plattform sollten Sie eine freundliche und höfliche Formulierung verwenden; niemand möchte Trolls um sich haben. Auch tiefergehende technische Fragen können gestellt werden. Wenn Sie irgendwelche Dokumente vermissen, kann es helfen, einfach nachzufragen.

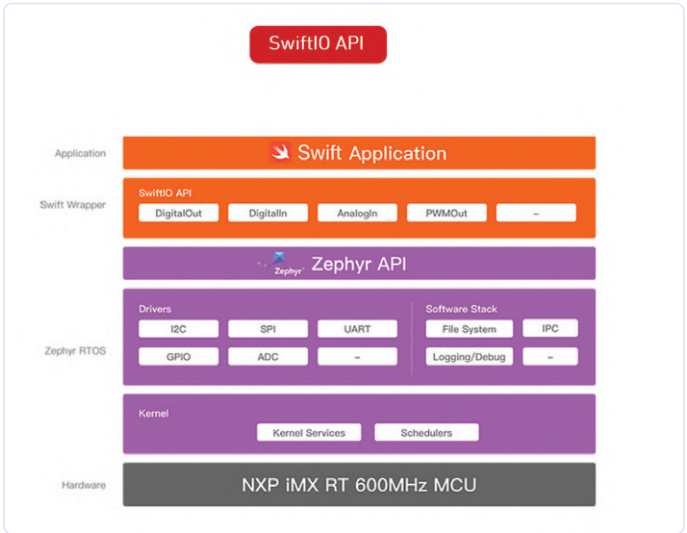


Bild 5. Softwarestack (Quelle: MadMachine.io)

Listing 1. Blink-Demoprogramm

```
/// Change the LED state every second by setting the interrupt.
/// Import the library to enable the relevant classes and functions.
 import SwiftIO
/// Initialize the red onboard LED and a timer to set interrupt.
 let red = DigitalOut(Id.RED)
 let timer = Timer()
/// Raise the interrupt to turn on or off the LED every second.
 timer.setInterrupt(ms: 1000) {
     red.toggle()
 }

 while true {
}
```

Abschließende Gedanken

Wenn Sie ein Arduino-kompatibles Board suchen oder etwas, das von Haus aus mit C/C++ verwendet werden kann, ist dieses Board nichts für Sie. Außerdem hat das Board einen höheren Preis als die meisten der üblichen Entwicklungsboards. Wenn Sie aber bedenken, dass Sie einen Support haben, die Entwicklung auf allen wichtigen Betriebssystemen möglich ist und Sie eine moderne Programmiersprache verwenden können - mit einem Board, das sehr gut *out of the box* funktioniert - liegt es an Ihnen, Ihre Prioritäten zu bestimmen.

210297-02

Listing 2. PWM-Demoprogramm

```
/// Brighten or dimming the LED by changing the duty cycle of PWM signal.
/// Import the library to enable the relevant classes and functions.
 import SwiftIO
/// Initialize the PWM pin the LED is connected to, with other parameters set to default.
 let led = PWMOut(Id.PWM0A)
/// Initialize a variable to store the value of duty cycle.
 var value: Float = 0.0
/// Change the brightness from on to off and off to on over and over again.
 while true {
     // Brighten the LED in two seconds.
     while value <= 1.0 {
         led.setDutycycle(value)
         sleep(ms: 20)
         value += 0.01
     }
     // Keep the duty cycle between 0.0 and 1.0.
     value = 1.0
    // Dimming the LED in two seconds.
     while value >= 0 {
         led.setDutycycle(value)
         sleep(ms: 20)
         value -= 0.01
     }
     // Keep the duty cycle between 0.0 and 1.0.
     value = 0.0
 }
```


Ein Beitrag von

Design und Text: **Mathias Claußen**
Redaktion: **Jens Nickel**
Übersetzung: **Vasileios Laskaridis**
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 mathias.claussen@elektor.com oder kontaktieren Sie Elektor unter editor@elektor.com.

WEBLINKS

[1] **MadMachine**: www.madmachine.io/

[2] **MadMachine CLI How-To**: https://resources.madmachine.io/ about-our-project#how-does-the-building-procedure-work

[3] **MadMachine Discord-Server**: https://discord.gg/zZ9bFHK

[4] **MadMachine-Resourcen**: https://resources.madmachine.io/

Aus dem Leben gegriffen

Eine elektronische On/Off-Beziehung

Foto: Ilse Joostens.

Von **Ilse Joostens** (Belgien)

In den 70er Jahren hatten wir noch einen Schwarz-Weiß-Röhrenfernseher der Firma ACEC [1], auf dem eine schwenkbare Zimmerantenne stand, ein konisches Ding mit zwei Armen. Als ich eines Tages von der Schule nach Hause kam, fand ich meinen Vater im Wohnzimmer vor, eifrig hinter dem Fernsehgerät beschäftigt. Die hintere Abdeckung war abgeschraubt und auf dem kleinen Tisch daneben lagen verschiedene Röhren. Irgendwann schaltete er den Fernseher wieder ein, um ihn zu testen, und ich erinnere mich noch an das geheimnisvolle Glühen der Röhren. Ich fand das sehr clever von ihm und habe deshalb voller Bewunderung zugeschaut. Damals wusste ich noch nicht, dass er eigentlich gar keine Ahnung davon hatte und nur einer detaillierten Anleitung folgte, die mein Onkel, der Aufzugsmechaniker war, ihm gegeben hatte. Die einzelnen Röhren waren auch ziemlich ehrfurchtgebietend. Es sah fast so aus, als ob sich darin, hinter dem Glas, kleine Satelliten oder andere Raumfahrzeuge befanden.

Knackende Geräusche und wütende Bauern

In meiner zarten Jugend hatte ich verschiedene Interessen und Hobbys, wie das Knüpfen von Teppichen, das Bestimmen von Wildpflanzen und das Experimentieren mit Lampen und Batterien. Ich war auch sehr an Wissenschaft interessiert und verschlang damals so manches „Was Ist Was“-Buch. Irgendwann bekam ich von meinem Onkel, also dem Aufzugsmechaniker, einen Karton voller Dinge, darunter auch ein paar altmodische Telefonhörer, noch mit Kohlemikrofon. Ich spielte natürlich sehr gerne Streiche und mit einer Batterie konnte man aus einem kleinen Lautsprecher, der einem solchen Telefonhörer entnommen war, ganz schön viele Knackgeräusch herausholen. Mit ein paar Metern Kabel dazwischen, um die Reichweite zu verlängern, konnte ich so manchen Menschen einen großen Schrecken einjagen. Heutzutage wäre das sehr viel schwieriger, aber zu jener Zeit, als die Tiere noch sprechen konnten, waren die meisten Menschen nicht an vieles gewöhnt.

Wir lebten in einer ländlichen Gegend und hinter unserem Garten befanden sich Wiesen, die von einem Elektrozaun

umgeben waren. Das lud natürlich zum Experimentieren ein, und ich amüsierte mich köstlich, indem ich alles Mögliche an den Elektrozaun anschloss, einschließlich eines VU-Meters, das ich aus einem alten Tonbandgerät gerettet hatte und dessen Zeiger sich fröhlich im Rhythmus der Hochspannungsimpulse hin und her bewegte. Allerdings dauerte es nicht lange, bis ein wütender Bauer an der Haustür klingelte, um mit meinen Eltern zu sprechen, was das vorzeitiges Ende meiner Experimente bedeutete.

Der Süßwarenladen

Als ich von der Grundschule in die weiterführende Schule wechselte, zog ich von einem ländlichen Nest in ein trostloses und verschlafenes Provinzstädtchen. Der Zufall wollte es, dass ich jeden Tag auf meinem Schulweg an einem Spielzeugladen und einem Geschäft für elektronische Bauteile vorbeikam. Obwohl beide Läden praktisch nebeneinander lagen, muss ich gestehen, dass ich in den ersten Jahren immer nur den Spielzeugladen betrat. Irgendwann jedoch war das Schaufenster des Elektrofachgeschäfts voll mit Bausätzen und ich wurde davon angezogen wie von einem Süßwarenladen. Der erste Bausatz, den ich damals kaufte, war ein einfacher Dämmerungsschalter, der einen LDR, ein paar Transistoren, etwas passiven Kleinkram und ein Relais enthielt. Als das Ding gebaut war, war ich stolz wie ein Pfau, dass ich das Relais zum Klicken bringen konnte, indem ich mit einer Taschenlampe den LDR beleuchtete oder nicht beleuchtete. Mission accomplished! Seitdem habe ich, mit wechselndem Erfolg, viele weitere Bausätze aufgebaut, obwohl sich das meist darauf beschränkte, der Bauanleitung zu folgen, ohne wirklich zu verstehen, was ich da tat. In unserem Dorf konnte ich auch so einen EX-System-Experimentierkasten der japanischen Firma Gakken mit „Denshi Blocks“ [2][3] ergattern. Ich hatte eine sehr vergnügliche Zeit, in der ich unzählige seltsame Geräusche und ungewollt „illegale“ Mittelwellen-Radiosendungen produzierte.

Das Gakken-Experimentiersystem.

Während meiner letzten Jahre auf der Oberschule verlagerten sich meine Interessen mehr in Richtung Chemie und ich richtete mir zu Hause ein kleines Labor ein. Das war eigentlich etwas unverantwortlich, denn irgendwann hatte ich genug giftiges Zeug, um die gesamte Einwohnerschaft unserer Stadt mehrfach um die Ecke zu bringen. Nicht, dass ich jemals das Ganze in die Luft gesprengt hätte oder so. Das Schlimmste, was mir passiert ist, war ein Stopfen, der während einer Destillation herausgesprungen ist, was dazu führte, dass eine Fontäne aus kochend heißem, billigem Rotwein an die Decke geschossen ist. Ich benötigte nämlich für ein Experiment reines Ethanol und das war, auch wegen der Verbrauchssteuer, recht teuer. Tja, wer billig kauft, kauft zweimal, heißt es manchmal.

Foto: Shutterstock.

Eine Hassliebe

Alles in allem fand ich Chemie doch etwas zu gefährlich und potentiell explosiv für ausgiebige Experimente zu Hause. Nach der Schule war ich eher an einem Studium in Richtung Elektronik interessiert. Aufgrund der damals sehr begrenzten Auswahlmöglichkeiten wurde es Elektromechanik, sehr gegen meinen Willen. Aus Mangel an anspruchsvollen Jobs in der Elektronikbranche in Belgien bin ich schließlich in der Welt der Informations- und Kommunikationstechnik gelandet, weshalb Elektronik wieder im Hintergrund verschwand.
Nach einer Karriere als Analytischer Programmierer und darauffolgender Systemadministration bin ich vor etwa zehn Jahren in die Selbstständigkeit als Elektronikentwickler gewechselt. Der Job eines Systemadministrators ist eben nicht auf Rosen gebettet, und schon gar nicht für ein Nervenbündel wie mich.
In der Zwischenzeit habe ich einige nette Projekte in dieser Zeitschrift veröffentlicht, aber mit der zunehmenden Konkurrenz aus Fernost und steigender Wertschätzung für niedrige Preise hat sich meine Liebe zur Elektronik wieder deutlich abgekühlt. Ich bewege mich jetzt viel eher in Richtung einer meiner anderen Leidenschaften, der Chemie, aber diesmal in Form von alten fotografischen Techniken [4][5] und dem Beizen von Holz mit Eisenacetat. In der Schule habe ich Geschichte und Ästhetik gehasst, aber seit ein paar Jahren sind sowohl Geschichte als auch Kunst zu zwei neuen Leidenschaften geworden. Vielleicht auch, weil ich jetzt ein paar Jahre älter bin und Menschen sich halt ändern. Vielleicht wird meine Liebe zur Elektronik eines Tages wieder zurückkehren, wer weiß…

210341-03

Ein Beitrag von

Autor: **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? Dann schreiben Sie eine E-Mail an die Elektor-Redaktion unter redaktion@elektor.de

WEBLINKS

[1] ACEC: www.radiocollection.be/fr/acec1968_fr.html
[2] Gakken EX-System: www.petervis.com/electronics-lab/gakken-ex-system/gakken-ex-system.html
[3] Gakken EX-System: https://en.wikipedia.org/wiki/Gakken_EX-System
[4] Cyanotypie: https://de.wikipedia.org/wiki/Cyanotypie
[5] Salzpapierdruck: www.alternativephotography.com/a-dash-of-salt/

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!

Velleman VTSS230 – 2-in-1 SMD-Heißluftstation

Preis: 82,50 €

www.elektor.de/19833

Envox EEZ Bench Box 3 (BB3) 4-ch – Modulare Test- und Messlösung

Preis: 899,00 €

Mitgliederpreis: 809,10 €

www.elektor.de/19825

M5Stack UnitV2 KI-Kamera für Edge Computing

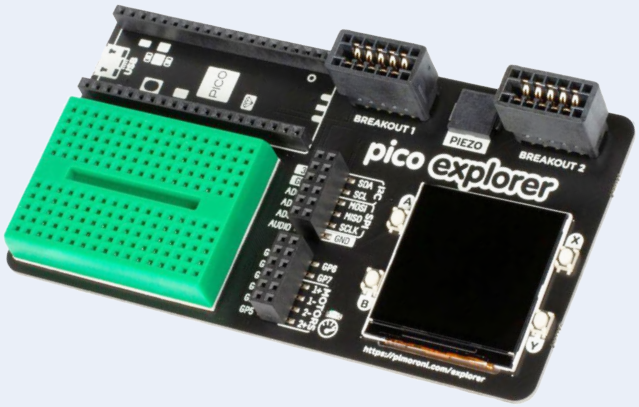
Preis: 79,95 €
Mitgliederpreis: 71,96 €

www.elektor.de/19782

Phiz 3D-Laserscanner für Smartphone

Preis: 349,00 €
Mitgliederpreis: 314,10 €

www.elektor.de/19708



Pimoroni Raspberry Pi Pico Explorer Base

Preis: 29,95 €
Mitgliederpreis: 26,96 €

www.elektor.de/19767

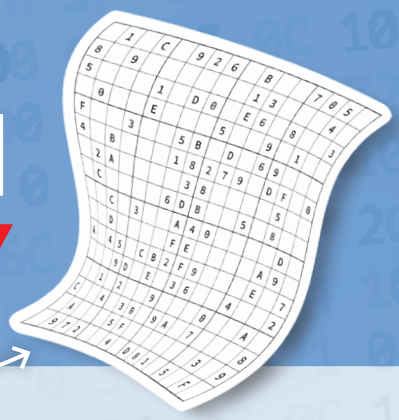
JOY-iT LCR-T7 Multifunktions-Komponententester

Preis: 24,95 €
Mitgliederpreis: 22,46 €

www.elektor.de/19709

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. Oktober 2021.

DIE GEWINNER DES HEXADOKUS AUS DER AUSGABE JULI/AUGUST STEHEN FEST!

Die richtige Lösung ist: **ADFB3**

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!

A			5				6	4				1			8
D		7		4			2	3			5		6		B
		C		B						F	A		E		
	E		0		C		8	1		6		4		D	
2	0	E	3		B	C			A	1		6	D	4	7
	5	4				F			0				3	1	
6		9	B		2					C		A	8		0
				9	4					7	8				
				C	9					8	E				
E		5	F		D					A		8	B		4
	1	B				8			6				9	A	
7	6	8	C		1	E			9	D		3	F	2	5
	B		2		E		F	C		5		9		8	
		0		D	6					4	3		2		
8		A		1			C	D			2		4		6
5			7				4	A				C			3

C	4	8	F	2	E	0	3	D	6	B	1	7	A	5	9
5	3	9	D	F	1	7	8	0	C	A	E	4	6	2	B
6	A	0	1	4	D	9	B	5	7	2	8	C	E	3	F
B	2	E	7	A	5	6	C	3	4	9	F	8	0	D	1
D	B	2	5	0	9	1	6	4	F	3	7	E	C	8	A
E	F	6	3	B	8	2	4	C	A	0	5	9	D	1	7
8	9	4	C	3	7	A	E	1	B	D	6	2	F	0	5
7	0	1	A	5	C	F	D	E	2	8	9	B	3	4	6
F	D	3	9	C	6	B	5	2	8	1	A	0	4	7	E
2	6	7	0	8	A	D	F	B	3	E	4	5	1	9	C
A	5	B	4	7	2	E	1	F	9	C	0	D	8	6	3
1	8	C	E	9	3	4	0	6	5	7	D	A	B	F	2
0	C	F	6	E	B	5	9	7	D	4	3	1	2	A	8
9	E	A	2	1	4	3	7	8	0	6	C	F	5	B	D
3	1	5	B	D	0	8	A	9	E	F	2	6	7	C	4
4	7	D	8	6	F	C	2	A	1	5	B	3	9	E	0

Treten Sie jetzt der Elektor Community bei!

Jetzt elektor GOLD membership Mitglied werden!

- Komplettes Webarchiv ab 1970
- 6x Elektor Doppelheft (Print)
- 9x Digital (PDF) inkl. Elektor Industry
- 10% Rabatt im Online-Shop und exklusive Angebote
- Elektor Jahrgangs-DVD
- Mit Tausenden von Mitgliedern des Online-Labors gemeinsam entwickeln mit Zugang zu über 1.000 Gerber-Dateien und direktem Kontakt zu unseren Experten!
- Veröffentlichen Sie Ihr eigenes Projekt oder verkaufen Sie direkt über unseren Shop!

Auch erhältlich

Die digitale elektor GREEN membership Mitgliedschaft!

- Zugang zu unserem Webarchiv
- 10% Rabatt in unserem Online-Shop
- 6x Elektor Doppelheft (PDF)
- Exklusive Angebote
- Zugang zu über 1.000 Gerber-Dateien

www.elektor.de/mitglied

elektor design › share › sell

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