DEUTSCHLANDS GEFÄHRLICHSTES DIY-MAGAZIN

c't Make:

Make:

Datenpetze:
Deye-Wechselrichter absichern

Oszi-Röhre emulieren

- Vektordisplay mit Laser
- Galvos mit ESP32 steuern
- Erlenmeyerkolben als Röhre

Modulare Frontplatten

- Schnell 3D-gedruckt
- Einfach anpassbar
- Beispiel: Laborverstärker

CNC-Rahmen für Oberfräse

- Einfach aufsetzen und fräsen
- Auch für Laser & Dispenser
- Mit Arduino und CNC-Shield

Know-how

- Dreh-Encoder verstehen
- Speicherverbrauch bei Arduinos

Werkstatt

- Gratis: Schaltungen simulieren
- Amtlich lasergravieren

4 192314 113500 03

3/24
31.5.2024
CH CHF 26.50
AT 14,90
Benelux 15,90
€ 13,50

Ein Veranstaltungsformat der Make:
make-magazin.de

10 JAHRE

Maker Faire®

HANNOVER

17.–18. Aug.

Hannover Congress Centrum

maker-faire.de

UNTER DER SCHIRMHERRSCHAFT VON
BETTINA STARK-WATZINGER MDB

Bundesministerium
für Bildung
und Forschung

sponsored by

BASIC SOLAR

Achtsamkeit

Bild: Shutterstock: Dimedrol68

Wir machen neuerdings immer freitags kleine Umfragen in unserem WhatsApp-Channel (Details zu unseren vielen Social-Media-Auftritten finden Sie auf S. 37). Kürzlich fragte meine Social-Media-Kollegin Dunia unsere Community: „Was sind eure größten Hindernisse beim Maken?“ Zur Wahl standen „Beschaffung von Materialien“, „Technische Schwierigkeiten“, „Zeitmanagement“, „Mangel an Fachkenntnissen“ sowie „Platzmangel / Werkstattorganisation“.

Das Votum war zwar nicht repräsentativ, aber eindeutig: Zeitmanagement ist offenbar mit weitem Abstand für viele das größte Problem beim Nachgehen des Hobbys nach Feierabend. Das überrascht mich nicht, denn das höre ich seit Jahren auch immer wieder in persönlichen Gesprächen auf Maker Faires. Mich überrascht eher, dass das Thema „Ich hab zu wenig Zeit für Hobbys“ offenbar epidemisch wird. Und das, obwohl die freie Zeit nach Feierabend in den letzten Jahrzehnten eigentlich immer mehr wird. Aber wo geht die ganze Zeit hin? Bei den meisten geht sie fürs Fernsehen, Streaming, Surfen im Internet, Social Media drauf: Konsum von mehr oder minder nutzloser Information.

Leider gehts mir seit Corona ähnlich, ich kämpfe gegen die Verlockung inhaltsleerer Ablenkung vieler Plattformen an. Nicht, dass ich unter dem Fear of Missing Out (FOMO) leide, also der Angst, etwas zu verpassen. Aber ich bin schon von Berufs wegen ein neugieriger Mensch, der sich gerne inspirieren lässt. Leider ist das Internet mittlerweile so gebaut, dass man, selbst wenn man nur kurz was schauen wollte, vom Hölzchen aufs Stöckchen kommt und sich stundenlang verliert.

Damit mich mein Kleinhirn nicht ständig in die Aufmerksamkeitsfalle tappen lässt, führe ich mir nun nach Feierabend immer wieder vor Augen, was mir eine nachhaltigere Entspannung und Befriedigung bringt. Ist es das flüchtige Vergnügen beim Binge-Watching einer Serie auf Netflix und das Durchscrollen auf Instagram? Oder das Erfolgserlebnis beim Aufbau eines Projekts und Verstehen eines Programmiertricks in C? Ich denke, die Antwort ist klar.

Um gegen den Sirenengesang der Social-Media-Plattformen zu bestehen und wieder in den Zustand einer bewussten Entscheidung zu kommen, muss man allerdings üben. Achtsamkeit ist eines der Stichwörter. Das klang für mich vor einigen Jahren noch eher nach Klangschalentherapie. Mittlerweile weiß ich, dass es nichts mit Esoterik und eigentlich nur mit Disziplin zu tun hat und dass es bei der Fokussierung und der Konzentration enorm hilft. Sie werden lachen, seitdem ich alle Romane von Karsten Dusse rund um „Achtsam morden“ gelesen habe, gelingt mir das sogar ganz gut.

Was bedeutet das jetzt alles für Sie, lieber Make-Leser? Halten Sie öfter inne: Denken Sie weniger darüber nach, ob und wie Sie Ihre Zeit besser aufteilen können. Füllen Sie die Zeit lieber sinnvoller mit Ihren Projekten. Legen Sie das Smartphone zur Seite und sich selbst nicht auf die Couch vor den Fernseher. Gehen Sie stattdessen in die Werkstatt oder Ihr Hobbylabor und genießen genau diesen Erfolg.

Ach, eine Bitte noch: Kündigen Sie nicht gleich alle Social-Media-Kanäle der Make, nutzen Sie sie einfach bewusster!

Happy Hacking

Daniel Bachfeld

▶ **make-magazin.de/xf29**

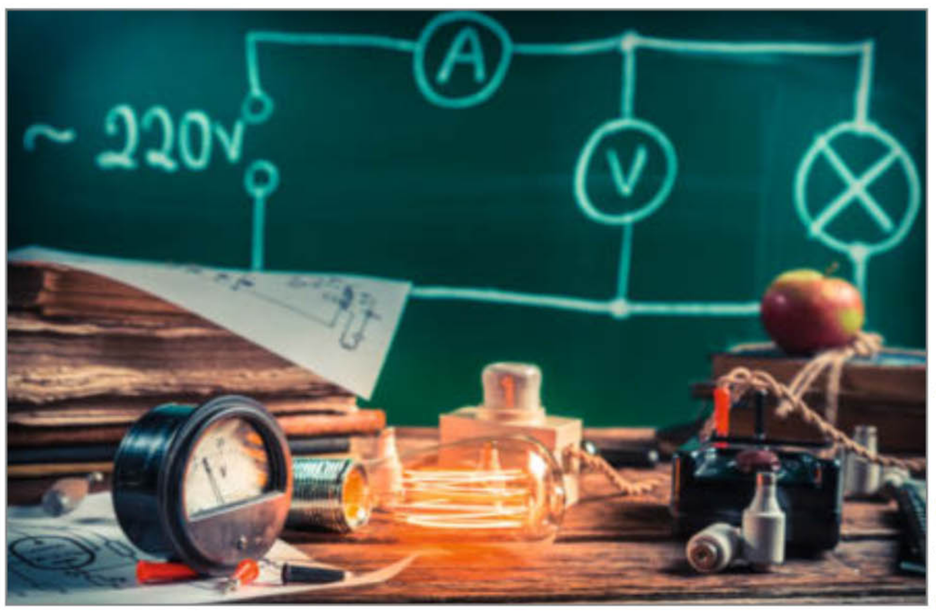
Oszi-Röhre emulieren

Laser- statt Kathodenstrahl: Einst mussten alte Oszilloskop-Röhren herhalten, um daraus nostalgisch wirkende Uhren, Spiele und Ähnliches zu bauen. Moderner gehts mit Erlenmeyerkolben aus dem Chemiebedarf, Laserpointer und Galvo-Scannern. Einfacher nachzubauen ist dieses Projekt auch noch.

Inhalt

Know-how/Werkstatt

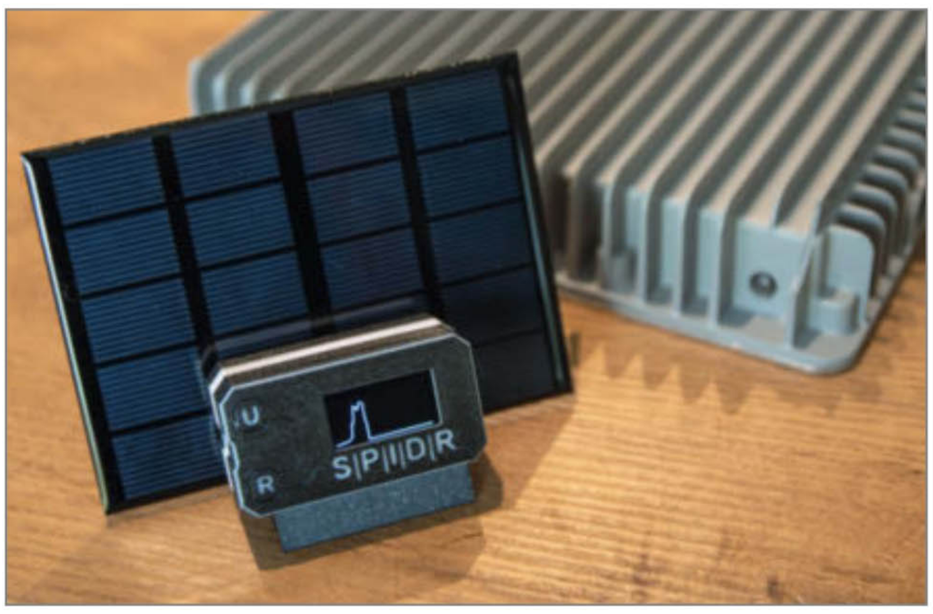
Lasergravuren sind beeindruckend, wenn man den Strahlkünstler optimal einstellt. Der Workshop zeigt, wie Sie das herausfinden. Elektronikentwicklern sagen wir, was Schaltungssimulatoren taugen, wozu der Speicher in Mikrocontrollern benutzt wird und wie Sie Drehgeber richtig einsetzen.



Shaiith / Shutterstock.com

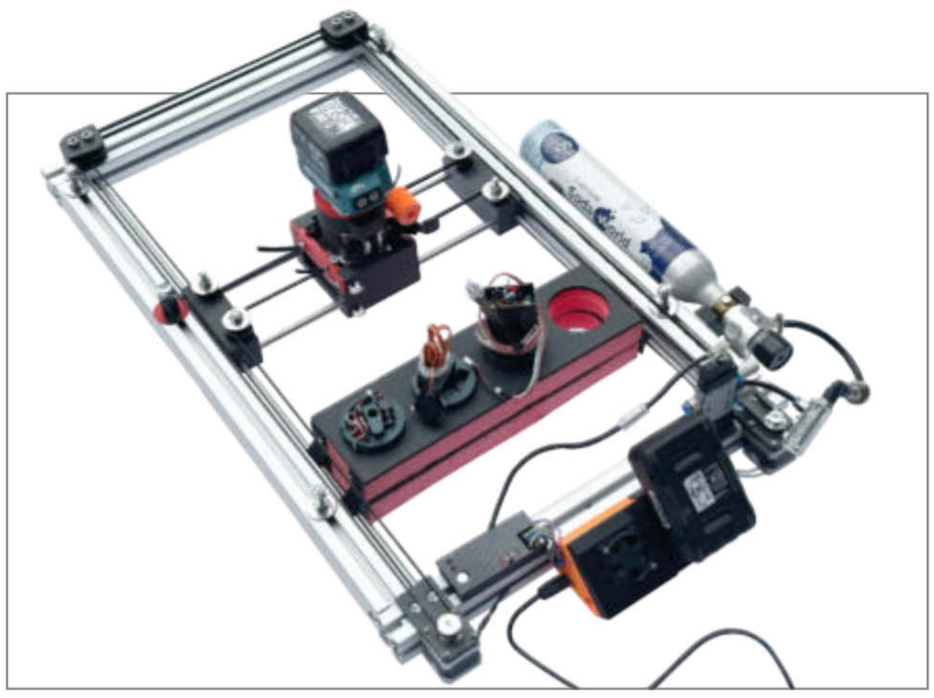
Deye-Wechselrichter sicher machen

Balkonkraftwerke sind in, sparen sie doch so manchen Euro ein. Doch die Wechselrichter (Inverter) solcher Anlagen können recht geschwätzig sein und eigentlich geheimzuhaltende Daten in die unkontrollierbare Cloud senden oder sich von dort manipulieren lassen. Hier erfahren Sie, wie Sie Geräte des Herstellers Deye vom internationalen Datenverkehr sicher trennen.



CNC-Rahmen für Oberfräse

Mit einer Oberfräse lassen sich schöne Sachen formen, so man eine ruhige Hand hat oder entsprechende Schablonen zur Führung des Werkzeugs herstellt. Das geht auch einfacher: In diesem X-Y-Positionierer wird das Elektrowerkzeug exakt vom Computer geführt. Mobi-C ist aus Alu- und 3D-Druckteilen leicht nachzubauen, komplett akkubetrieben und einfach transportierbar.



Themen von der Titelseite sind rot gesetzt.

Modulare Frontplatten

Schalttafeln wie in einem Kraftwerk aus kleinen, kachelähnlichen Funktionseinheiten für Schalter, Potis, Anzeigeleuchten, Messinstrumente und was sonst noch für die Anlagensteuerung und -überwachung nötig ist, machen Eindruck. Mit OpenSCAD konstruieren Sie Ihre 3D-Druck-Schaltwarte ganz schnell.

Leserforum

Berichtigungen

ATtiny statt Arduino, Make 2/24, S. 72

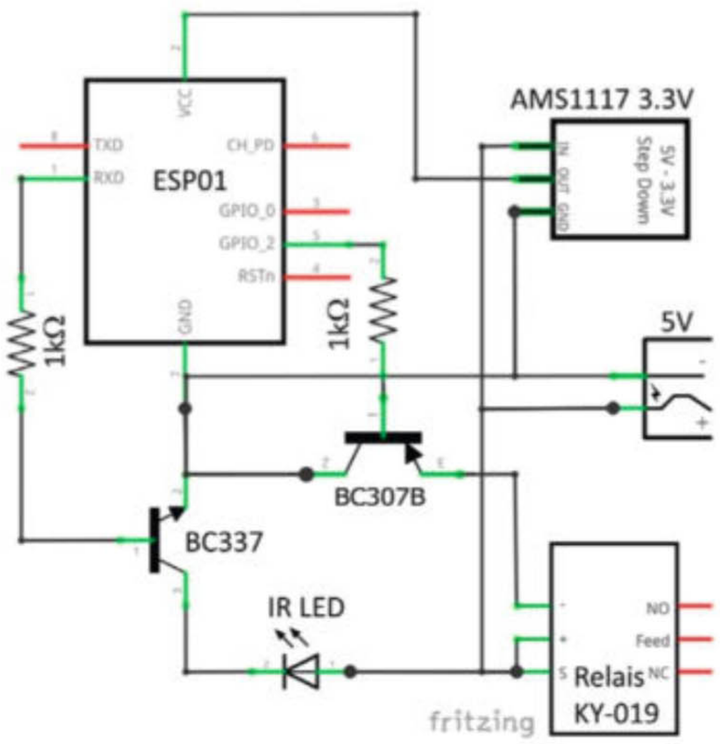
Der Schaltplan zum Projekt ist richtig, allerdings haben sich beim Fritzing-Diagramm zwei Fehler eingeschlichen: Die Lampe wird von GND nach GND geschaltet statt nach 12 V. Die Kathode der LED ist mit 12 V statt mit GND verbunden. Wir haben ein korrigiertes Schema im Github-Repository des Projekts abgelegt.

Memory Maps verstehen, Make 2/24, S. 122

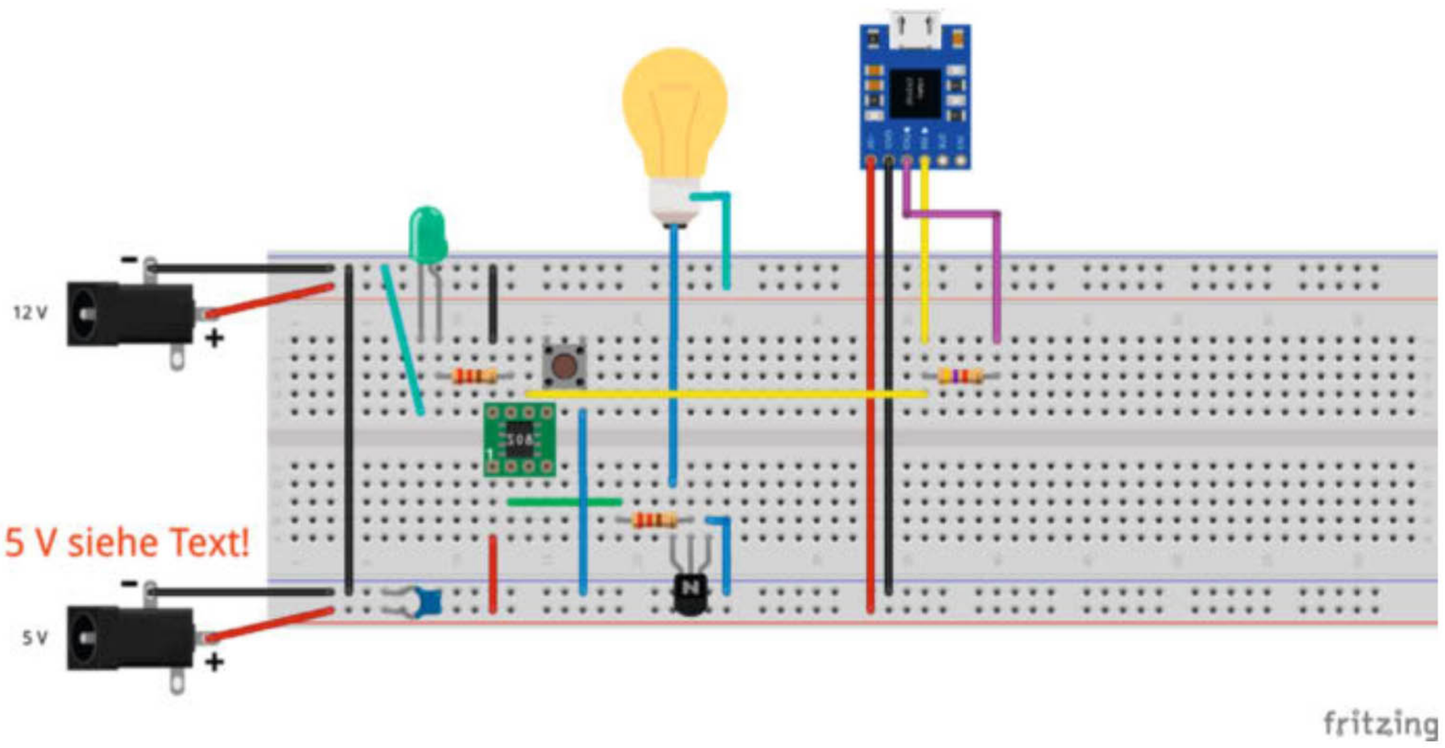
Die im Artikel beispielhaft genannten Speicherstellen 0x7FFF entsprechen dezimal 32767 beziehungsweise 0x8000 entspricht 32768. Zudem sind in Bild 3 in der Tabelle A14 und A15 vertauscht. In der untigen Abbildung ist es richtig.

A15	A14	IC
0	0	ROM
0	1	ROM
1	0	RAM
1	1	I/O

Universal-IR-Fernbedienung, Make 1/24, S. 72



Der korrigierte Schaltplan für die IR-Fern-bedienung aus Make 1/24



Bei Bild 5 auf Seite 75 ist leider in der Schaltplanansicht die Transistor-Beschriftung B0338 falsch, es handelt sich um einen BC337. Dafür wurde für den anderen Transistor BC307B -versehentlich der Typ NPN statt PNP verwendet. Hier drucken wir nochmal den korrigierten Schaltplan, er findet sich auch im Github-Repository zum Projekt.

Smart Home zu Ende denken

Frühjahrsputz im Smart Home, Make 2/24, S. 56

Ein echt lehrreicher Artikel – so, zumindest so ähnlich, sollte man das regelmäßig machen. Aber: Die ganze Sache mit dem Smart Home selbst aufbauen und programmieren ist nicht zu Ende gedacht. Wie am Anfang des Artikels geschrieben wurde, ist „Schatzi" nur rudimentär in die „never ending Story" eingebunden. Das eigentliche Problem entsteht doch erst dann, wenn der Admin des Smart Homes das Zeitliche segnet oder gesundheitlich nicht mehr in der Lage ist, eines der unzähligen Skripte anzupassen. Dann steht der Rest der Familie ziemlich ratlos da, weil hier keiner helfen kann – wer will / kann sich in überschaubarem Zeitrahmen und Kosten in das Ganze reindenken und ggf. einen Fehler finden? Die meist mega guten Foren – in unserem Fall NodeRed – helfen bestimmt gerne: wenn man die richtigen Fragen stellt. Aber wer von den verbliebenen Nutzern kann das? Was passiert, wenn die Bude verkauft werden soll und der Käufer nicht zu den Smart-Home-Nerds gehört und mit der ganzen Technik auf Kriegsfuß steht? Alles wieder rausreißen und verschrotten? Von meiner Seite besteht ein großes Interesse an einer solchen Diskussion und einer möglichen Lösung.

Hubertus

Wartungsfeindlich

Editorial, Make 2/24, S. 3

Ihr Editorial spricht mir so sehr aus der Seele, dass ich Ihnen schreiben muss. Gratulation zu diesem Text! Als Mitglied eines lokalen Repair-Cafés weiß ich zu gut, wovon Sie sprechen, wenn Sie das Wort wartungsfeindlich verwenden. Allzu oft scheitert eine Reparatur, weil sich Geräte nicht öffnen lassen. Unsere Welt erstickt im Plastik- und Elektronikschrott. Vielleicht hilft die Make hier entgegenzuwirken. Nehmen Sie die Themen Reparieren und Upcycling häufiger auf. Ich freue mich drauf.

Björn Flach

Kein Schaltplan

DIY-Röhren-Vorverstärker, Make 2/24, S. 12

Nachdem ich letztes Jahr mein Make-Abo gekündigt hatte, habe ich mir probehalber die Ausgabe 2/24 am Kiosk gekauft und wollte den Artikel DIY-Röhrenverstärker durchlesen. Leider wurde ich sehr enttäuscht, da noch nicht einmal der Schaltplan veröffentlicht wurde. Was macht ein Maker ohne Schaltplan, eigent-

lich nichts. Ist dies ein reiner Werbeartikel, um den Bausatz zu kaufen? Auch im Web zum Artikel war nichts zu finden.

Uwe Lüders

Der Schaltplan ist auf der Projektseite zu finden, die wir in unserer Kurzinfo zum Artikel verlinkt haben.

Irritiert

Task-Reminder, Make 1/24, S. 8

Ihr Artikel hat mich durchaus ein wenig irritiert. Es sieht für mich erst mal so aus, als würde die Front ein LCD-Display bilden, links mit einem Kubus drauf und rechts zwei QR-Codes. Und unten zwei herausragende Acrylstreifen. Wie kommen da die Bildchen drauf? Per CNC-Fräse? Und einen 3D-Drucker braucht es auch noch? Gefällt mir diesmal gar nicht.

Dietrich Jordan

Lizenzfragen

Lasern mit Lightburn, Make 1/24, S.24

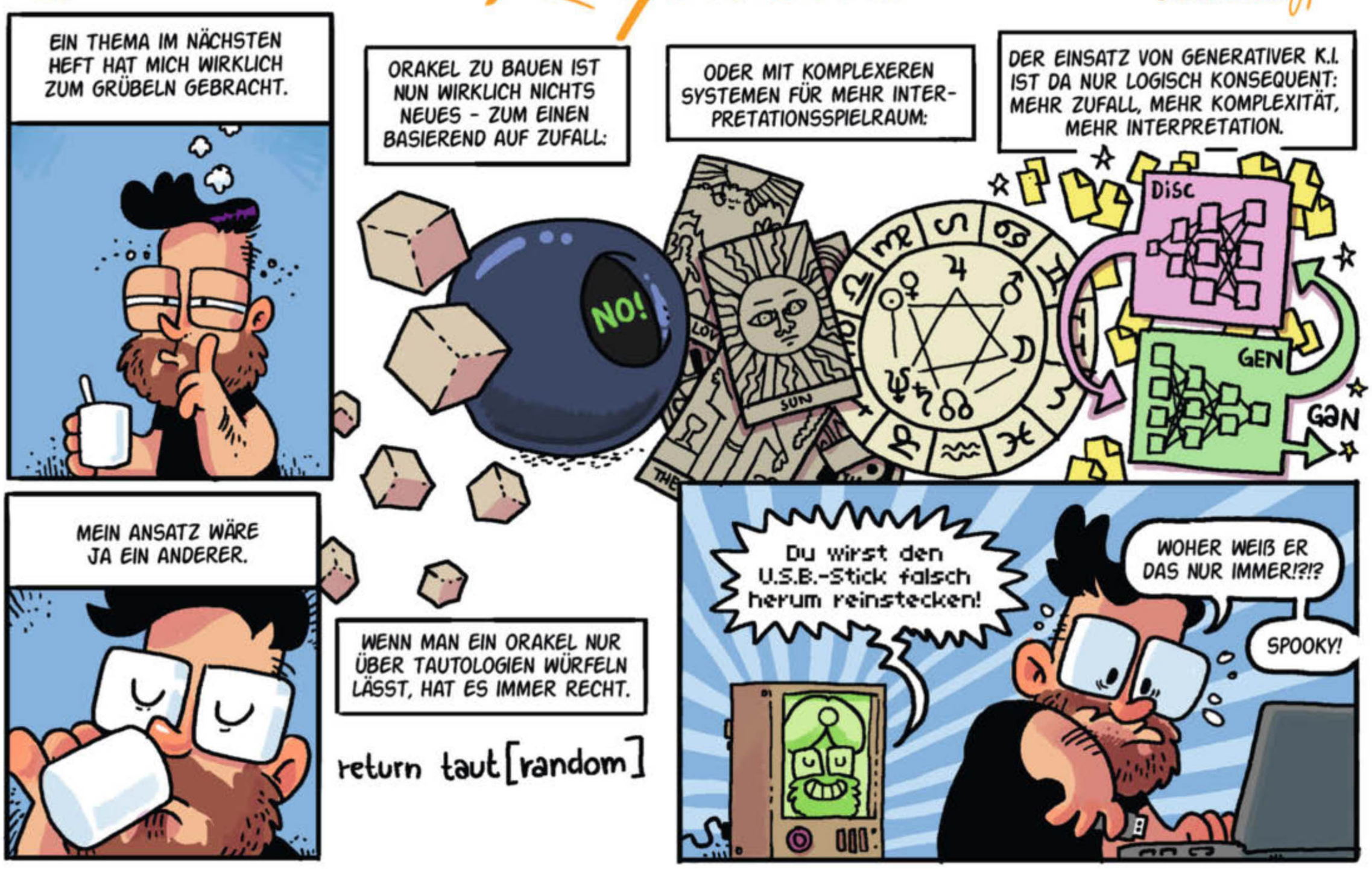
Vielen Dank für Ihre hilfreichen und informativen Artikel! Ich habe eine Frage zur Lizenz von LightBurn: Es gibt bei der Lizenz für GCode-Controller (z. B. Marlin) den Hinweis, dass Personen mit Emblaser bitte eine Lizenz von DarklyLabs kaufen sollen. Da diese für drei Jahre statt für ein Jahr gültig ist und nur 90 Dollar kostet, stellt sich mir die Frage, ob diese Lizenz trotzdem auch für andere GCode-Controller funktioniert. Könnten Sie das vielleicht testen? Der DarklyLabs-Support konnte mir da leider keine Auskunft geben.

Helge Ahrens

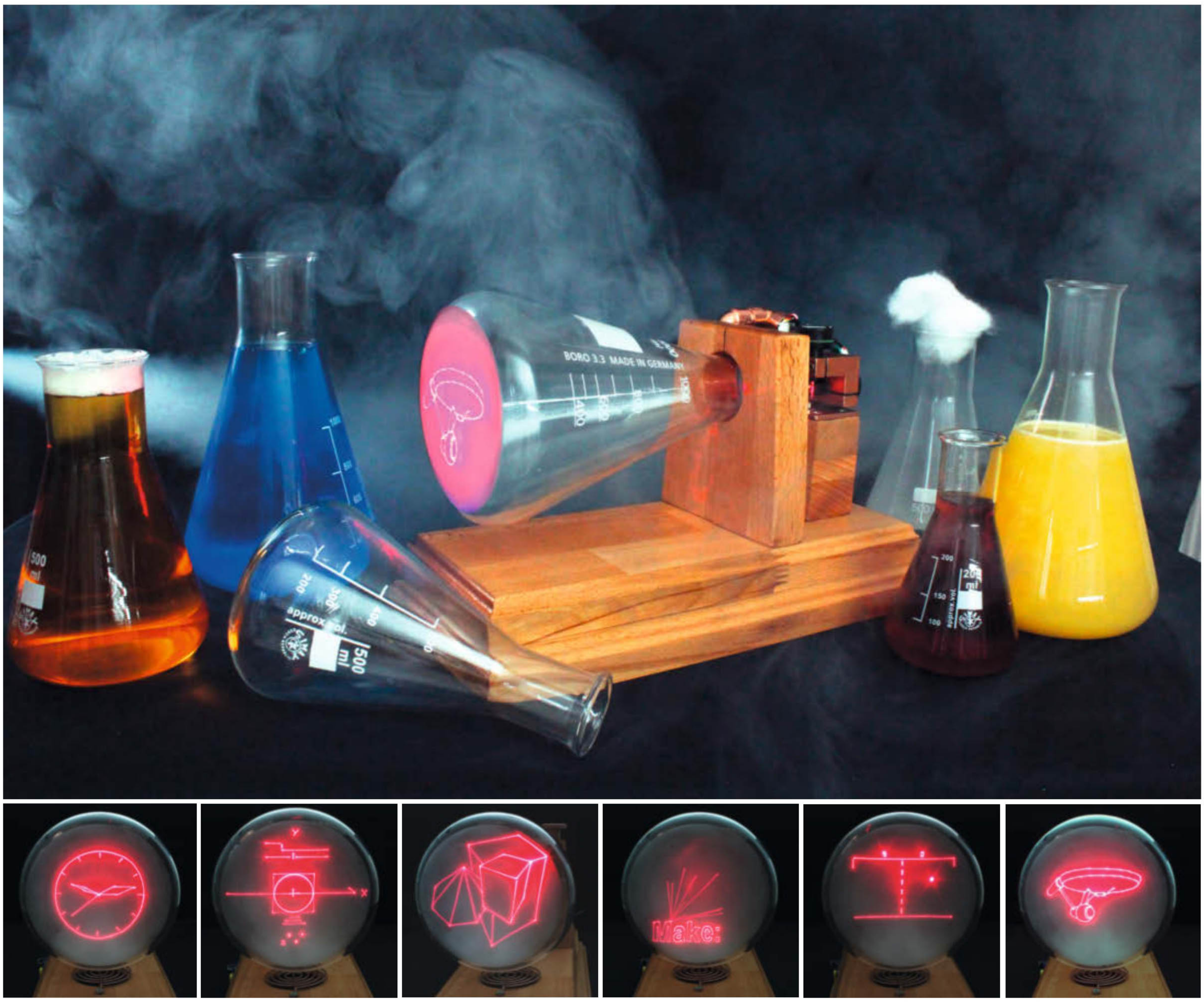
Ob ein mittels Emblaser-Lizenz freigeschaltetes Lightburn auch mit anderen Lasern wie beispielsweise Marlin-basierten Geräten läuft, weiß ich leider auch nicht. Was die Lizenzlaufzeit angeht, gibt es aber wohl ein Missverständnis: Haben Sie Lightburn, mit welcher Lizenz auch immer, freigeschaltet, bleibt das Programm unbegrenzt lang lauffähig. Lediglich die Update-Möglichkeit ist begrenzt auf ein Jahr, d. h., nach einem Jahr können Sie mit der dann aktuell auf Ihrem Computer vorhandenen Lightburn-Version beliebig lang weiterarbeiten. Möchten Sie weiterhin Updates erhalten, gibt es dafür eine Verlängerungslizenz, deren Preis meines Wissens nach unter 30 Euro / Jahr liegt. Notwendig zum Betrieb des Lasers ist die aber nicht.

Kolophonium

von und mit @beetlebum



Dank seines selbstgebauten KI-Orakels ist unser Zeichner seiner Zeit voraus. Wie man selbst eines baut, steht dann in der Make 4/24.



Die Erlenmeyer-Laserröhre

Nein, das ist keine antike Oszilloskop-Röhre, die hier als Vektordisplay arbeitet: Stattdessen zeichnet ein Laserpointer die Grafiken auf den mattierten Boden eines Erlenmeyerkolbens aus dem Chemielabor. Dieser selbst gebaute Bildschirm lässt sich zudem mit beliebigen eigenen Grafiken beschicken und sogar Retrospiele laufen darauf, wenn man die passenden Controller bastelt.

von Ulrich Schmerold

Januar 2024, in der Make-Redaktion: Der stellvertretende Chefredakteur Peter König bearbeitet gerade den Artikel zu Pepper's Ghost für die Ausgabe 1/24 und hat für eigene Experimente mit der dort beschriebenen optischen Illusion auch einen Erlenmeyerkolben auf seinem Schreibtisch stehen. Irgendwann fiel ihm ein, woran er beim Anblick des Kolbens schon die ganze Zeit denken musste: Legt man ihn auf die Seite, erinnert er stark an eine antike Oszilloskop-Röhre (Braunsche Röhre, Kathodenstrahlröhre), wie sie etwa als Anzeige für den Mini-Asteroids-Automaten von Carsten Meyer dient (c't Hacks 3/12, S. 110). Man müsste den Boden des Erlenmeyerkolbens jetzt nur matt bekommen und dann von der Öffnung her mit einem Laserpointer ein Bild auf den Boden projizieren. Aber kann man das spezielle Laborglas sandstrahlen, um die matte Fläche zu bekommen? Passt die Technik zum Steuern des Laserstrahls überhaupt durch den engen Hals des Kolbens? Und entsteht da am Ende überhaupt ein ansprechendes Bild?

Weil ich schon einige Erfahrungen mit Lasern und deren Ablenkung über sogenannte Galvos habe, bat Peter mich in einer Mail um eine Einschätzung und schickte seine Ideenskizze mit (Bild 1). Und er fragte mich, ob ich Lust hätte, diese vage Vision einmal praktisch auszuprobieren. Da ich sofort von dieser originellen Idee begeistert war, sagte ich gleich zu. Weil die ursprüngliche Idee von Peter stammt, den ich sehr schätze und der viele meiner Artikel betreut hat, möchte ich ihm dieses Projekt widmen.

Kurzinfo

- **Retro-Vektordisplay bauen und Anzeigen programmieren**
- **Laser-Galvos mit dem ESP32-WROOM-32 ansteuern**
- **Experimente mit verschiedenen Kolbenfüllungen**

Checkliste

Zeitaufwand:
ein bis zwei Wochenenden

Kosten:
ab 100 bis 150 Euro

Programmieren:
solide Grundkenntnisse beim Umgang mit Arduino IDE und ESP32

Unfallgefahr:
verantwortungsvoller Umgang mit Lasern nötig (siehe Kasten)

Mehr zum Thema

- Ulrich Schmerold, Sandstrahlkabine für Zuhause, Make 2/16, S. 120
- Ulrich Schmerold, Laserharfe, Make 1/17, S. 56
- Ulrich Schmerold, Spielen ohne Grenzen, Make 5/17, S. 28
- Ulrich Schmerold, Wertewandel, Make 5/17, S. 40

Alles zum Artikel im Web unter *make-magazin.de/xhvy*

Material

- **Galvanometer**, auch Laserscanner genannt
- **Dioden-Laser** 5 mW, Farbe nach Geschmack
- **Erlenmeyerkolben** aus Glas, 1000 ml, Enghals
- **ESP32-WROOM-32**
- **Netzteil** 5 V
- **Spannungswandler** SIM2-0512D
- **2 Operationsverstärker** LM358
- **Trimmpotentiometer** 2 × 10 kΩ, 2 × 100 kΩ
- **Potentiometer** 10 kΩ für die Taktrate
- **Widerstände** 4 × 47 kΩ, 2 × 33 kΩ, 4 × 22 kΩ, 2 × 10 kΩ
- **Kondensator** 470 µF
- **Taster**
- **Optionale Elektronikteile** laut Schaltplan
- **Lochrasterplatinen,** Kabel und Befestigungsmaterial
- **Holz** für den Sockel und die Halterungen

Werkzeug

- **Sandstrahlpistole** mit Kompressor und optional Sandstrahlbox
- **Bohrmaschine** oder besser Ständerbohrmaschine, dazu ein Satz Standardbohrer
- **Spezialbohrer** 45-mm-Forstnerbohrer, 12-mm-Bohrer
- **Lötstation** und Zubehör
- **Handwerkzeug** zur Holzbearbeitung
- **Maker-Werkzeug** für Elektronikarbeiten

Kolben mattieren

Die erste Frage, ob man einen Erlenmeyerkolben aus Laborglas sandstrahlen kann, ist schnell beantwortet: Ja, das funktioniert. Idealerweise hat der Maker dazu eine Sandstrahlbox in der Werkstatt stehen, vielleicht sogar eine selbst gebaute, wie in der Make 2/16 beschrieben (siehe „Mehr zum Thema" in der Kurzinfo). Wer keine Möglichkeit zum Sandstrahlen hat, kann mit selbstklebenden Mattierungsfolien oder auch aufgeklebtem Transparentpapier aus dem technischen Zeichnungsbedarf experimentieren. Ob das gewählte Material genügend mattiert, kann man dadurch abschätzen, dass man in einem abgedunkelten Raum den Kolben auf den Tisch

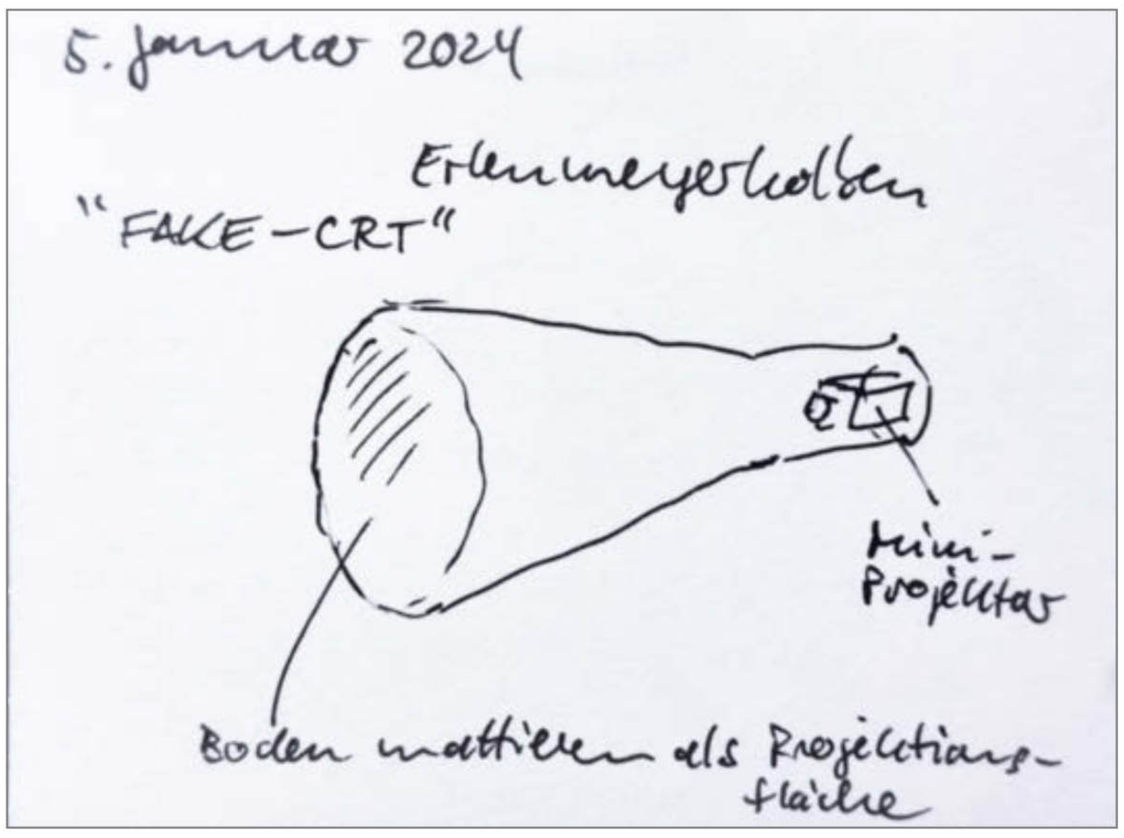


Bild 1: Die erste Skizze zum Projekt sieht noch keinen Laser, dafür als Platzhalter einen„Mini-Projektor" im Hals des Kolbens vor

Bild 2: Ein Stück HT-Rohr begrenzt die Wirkung des Sandstrahlers, das rotweiße Tuch schützt den Rest des Kolbens.

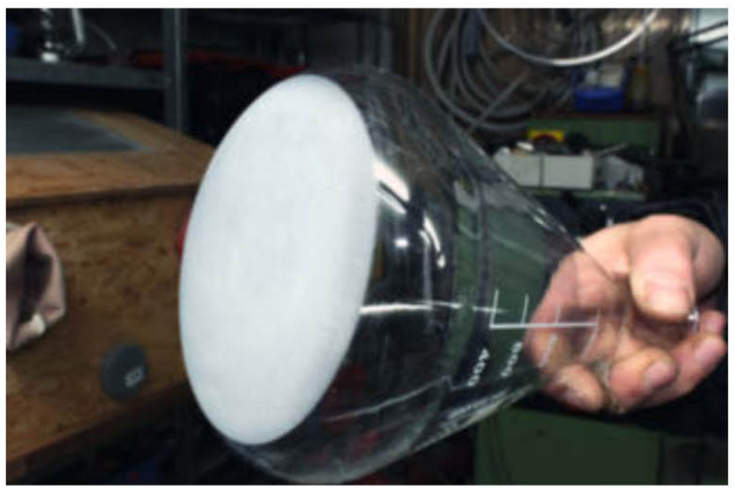


Bild 3: Mit dem HT-Rohr-Trick wird das Ergebnis perfekt!

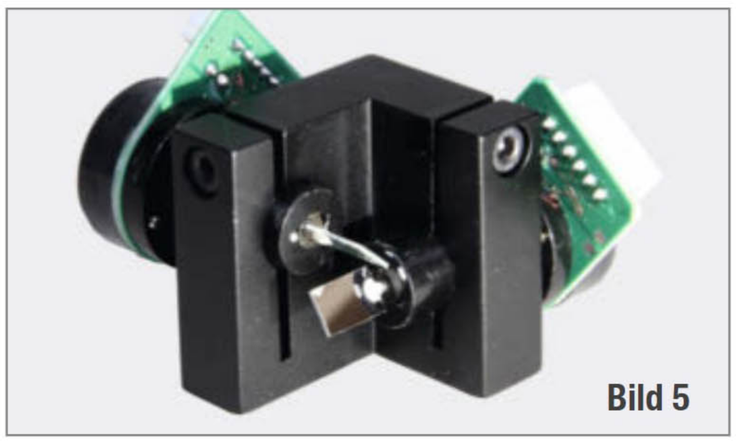


Bild 4: Von links nach rechts: Erlenmeyerkolben unbearbeitet, von innen sandgestrahlt, nach Abkleben von außen sandgestrahlt

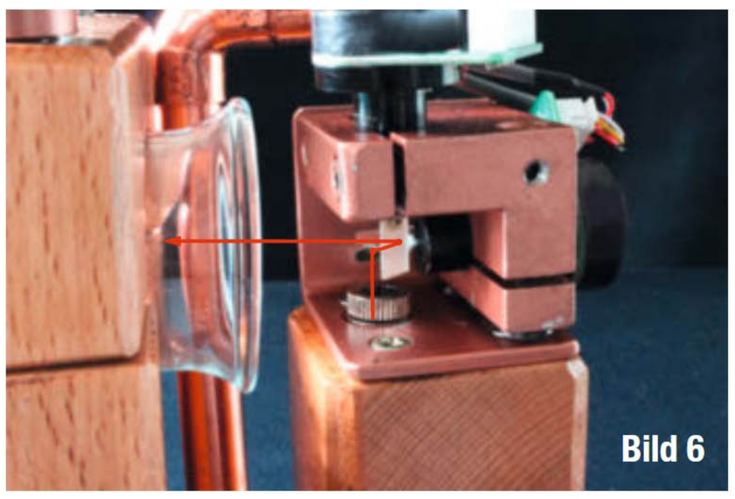
legt, mit einem starken Laserpointer durch den Hals in Richtung Boden strahlt und ein paar Zentimeter dahinter ein Blatt Papier hält. Erscheint dort nur ein diffuser Widerschein, sollte die Streuwirkung ausreichen.

Zum Sandstrahlen wählt man nach Möglichkeit einen sehr feinen Sand – z.B. Edelkorund 0,09 bis 0,125 mm oder Granatsand 0,01 bis 0,06 mm. Damit erhält man eine schönere, feine Anzeigefläche.

Prinzipiell lässt sich der Kolben von innen oder von außen sandstrahlen. Mein erster Versuch, dem Kolben von innen einen matten Boden zu geben, ergab allerdings ein unbefriedigendes Ergebnis: Der Erlenmeyerkolben wurde insgesamt matt und unansehnlich.

Beim zweiten Versuch entschied ich mich, die Außenseite zu behandeln, und schützte vorher alle Oberflächen, die klar bleiben sollten. Um einen schönen runden Anzeigebereich offenzulassen, verklebte ich als Begrenzung ein kurzes Stück von einem 100-mm-HT-Abflussrohr. Den Rest wickelte ich in einen Stofffetzen ein (Bild 2).

Dies ergab ein perfektes Ergebnis, bei dem alle Oberflächen klar und durchsichtig blieben und nur eine matte runde Anzeigefläche am Boden des Kolbens entstand. Das sah schon perfekt wie eine Braunsche Röhre aus (Bild 3).



Laser lenken

Mit dem Thema Galvo-Scanner (oder auch Galvanometer oder Laserscanner genannt, Bild 5) habe ich mich bereits 2017 mit meinen Projekten „Laserharfe“ und „Spielen ohne

Der Erlenmeyerkolben

Der Erlenmeyerkolben wurde 1860 von dem deutschen Chemiker Emil Erlenmeyer (1825 – 1909) entwickelt. Er erkannte, dass ein Glasgefäß mit zylindrischem Hals im Gegensatz zu dem bis dahin verbreiteten Becherglas besser zum Schwenken von Flüssigkeiten geeignet ist, da damit weniger Flüssigkeit verschüttet (oder verschüttelt) wird. Im Unterschied zu Reagenzgläsern hat der Erlenmeyerkolben einen flachen Boden und kann so stabil abgesetzt werden.

Heute werden die Erlenmeyerkolben weltweit in Laboren für Chemie, Medizin und Industrie eingesetzt und sind so verbreitet, dass ihre charakteristische Form auch immer wieder in Piktogrammen genutzt wird, die diese Branchen symbolisieren sollen. Erlenmeyerkolben gibt es inzwischen sowohl aus Glas als auch aus Kunststoff und in verschiedenen Ausführungen:

- Kolben mit Enghals (DIN 12380 / ISO 1773)
- Kolben mit Weithals (DIN 12385)
- mit und ohne Bördelrand
- mit und ohne aufgedruckter Teilung (Skala)

Dünnwandige Erlenmeyerkolben dürfen keinem Vakuum ausgesetzt werden, da sie sonst implodieren. Auch randvoll mit Flüssigkeit gefüllte Exemplare sind mit Vorsicht zu behandeln, wie sich bei diesem Projekt herausstellte … Mehr dazu steht in einem ergänzenden Online-Artikel, zu erreichen über den Link in der Kurzinfo. Kunststoffkolben brechen zwar nicht, sind für dieses Projekt allerdings ungeeignet.

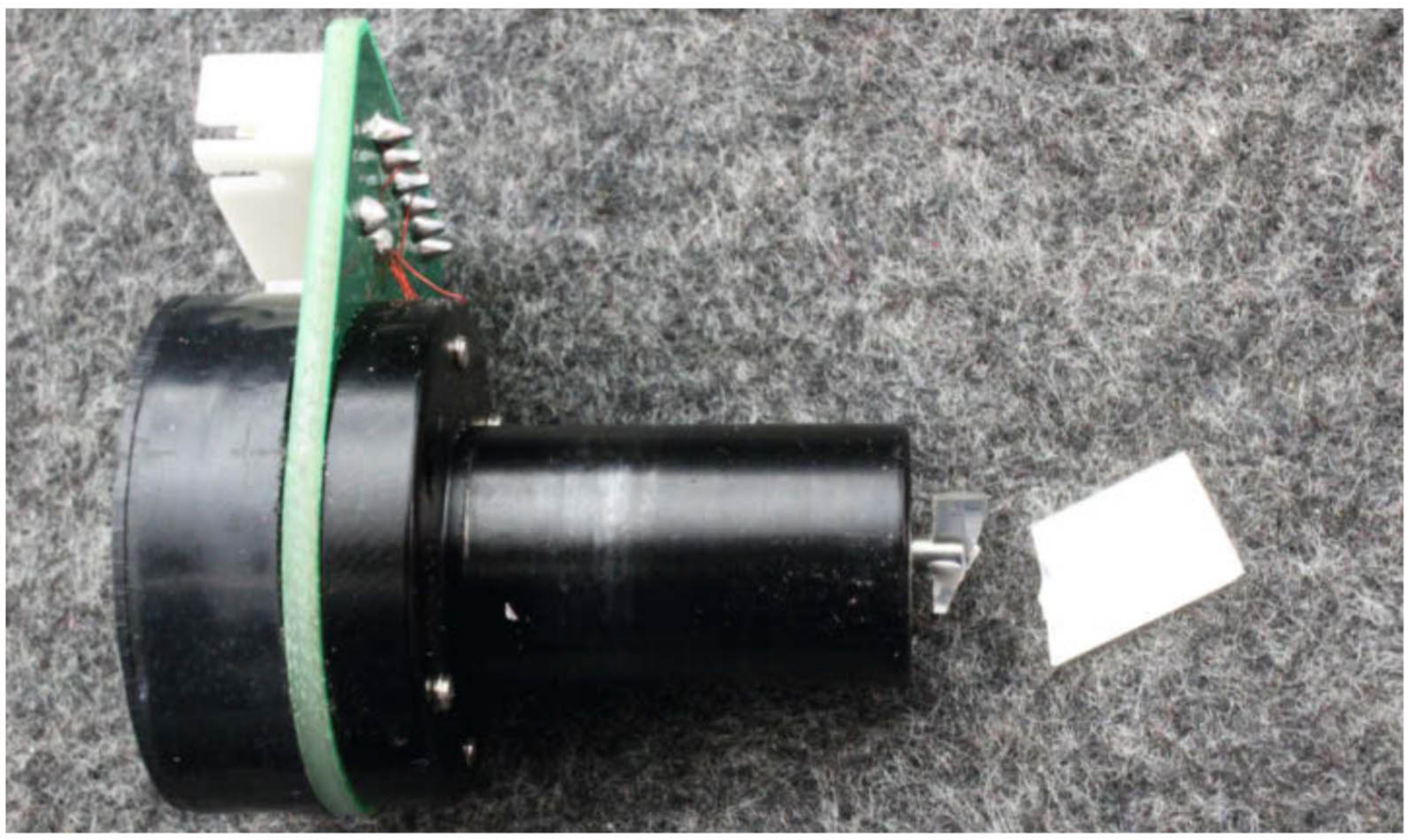


Bild 8: Ein zerbrochener Galvo-Spiegel lässt sich nicht mehr reparieren …

Bild 9: … besser, man schützt das empfindliche Teil durch eine Aluminiumblende.

Grenzen“ beschäftigt. Diese Artikel kann man über den Link in der Kurzinfo im Volltext kostenlos lesen und darin sind auch die Software für die Anzeige beschrieben, die jetzt modifiziert auch auf der Erlenmeyerröhre läuft. Somit konzentriere ich mich hier auf das, was sich seit damals geändert hat. Deshalb nur ganz kurz: Ein Galvo-Scanner verfügt über zwei drehbare Spiegel, die einen eintreffenden Laserstrahl gezielt in X- und Y-Richtung ablenken, um damit zweidimensionale Vektorgrafiken auf eine Projektionsfläche zu zeichnen. Bei meinem Laserspiel-Projekt diente eine Wand als Projektionsfläche, hier ist es hingegen der mattierte Boden des Erlenmeyerkolbens. Solche Laser-Galvos gibt es bereits ab 50 Euro zu kaufen, vernünftige können aber auch schon fast 100 Euro kosten.

Geändert habe ich im Vergleich zu meinem Laserspiel-Projekt den Verlauf des Laserstrahls: Der Laser wird jetzt nicht neben dem Galvo-Block positioniert, sondern unter dem Block im Holzsockel versteckt. Der Laserstrahl trifft also von unten auf den Y-Spiegel. Dieser spiegelt den Strahl auf den X-Spiegel, der ihn dann in den Erlenmeyerkolben umlenkt. Auf Bild 6 habe ich den Verlauf des Laserstrahls eingezeichnet. Nicht irritieren lassen: Auf dem Bild ist der Galvo-Block nicht mehr schwarz, sondern bereits kupferfarben lackiert.

Um das Geräusch, das der Galvo-Scanner im Betrieb verursacht, so weit wie möglich zu dämpfen, habe ich sowohl zwischen Galvo-Block und Befestigungswinkel als auch zwischen Befestigungswinkel und Schraube je einen 6-mm-O-Ring verbaut (aus dem Sanitärbereich des Baumarkts, Bild 7). Das Geräusch war anschließend deutlich leiser.

Da es mir bei diesem Projekt leider passiert ist, möchte ich noch einmal eine ernst gemeinte Warnung abgeben: Die Spiegel des Galvos sind extrem empfindlich! Damit sie möglichst

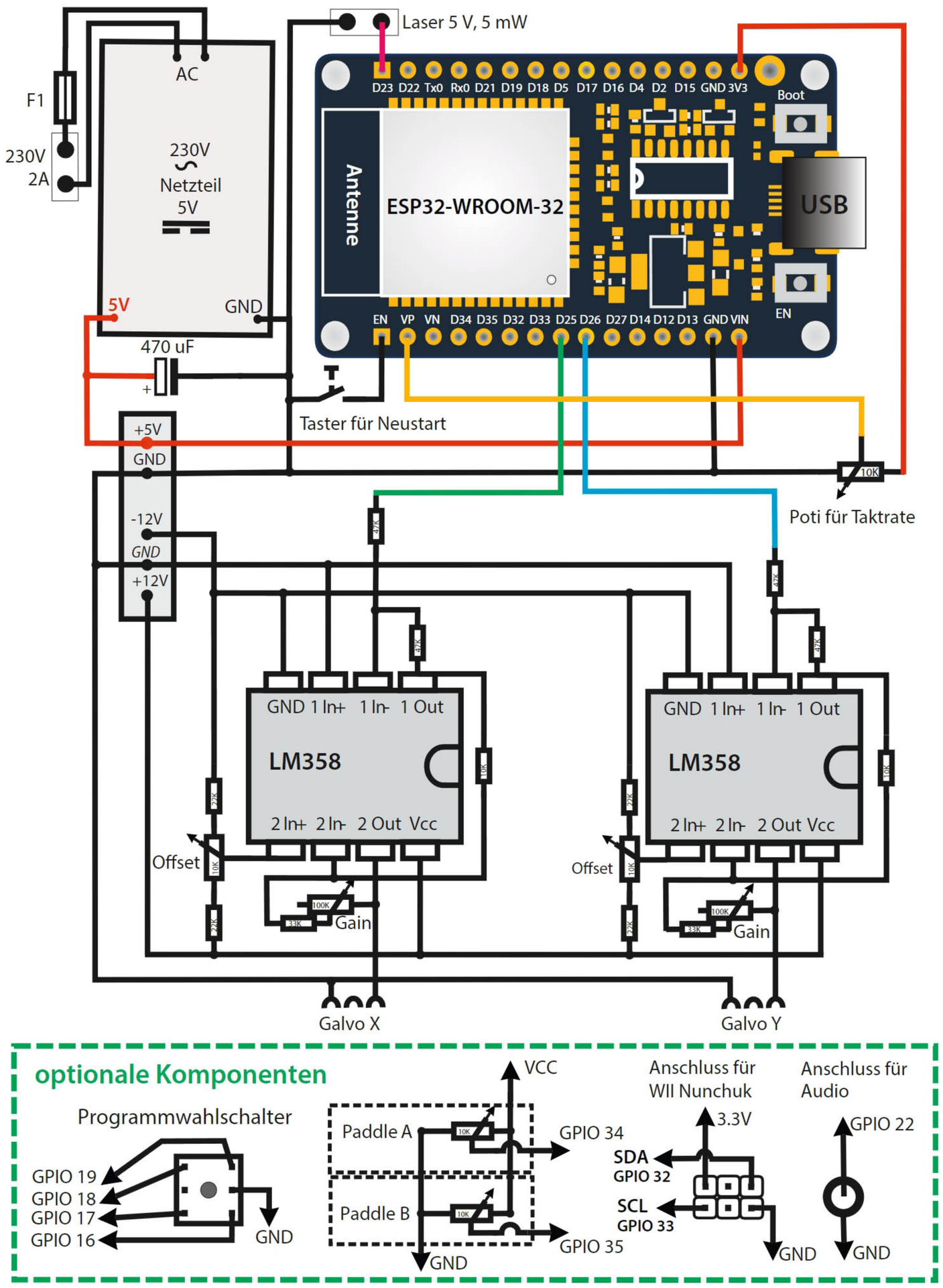


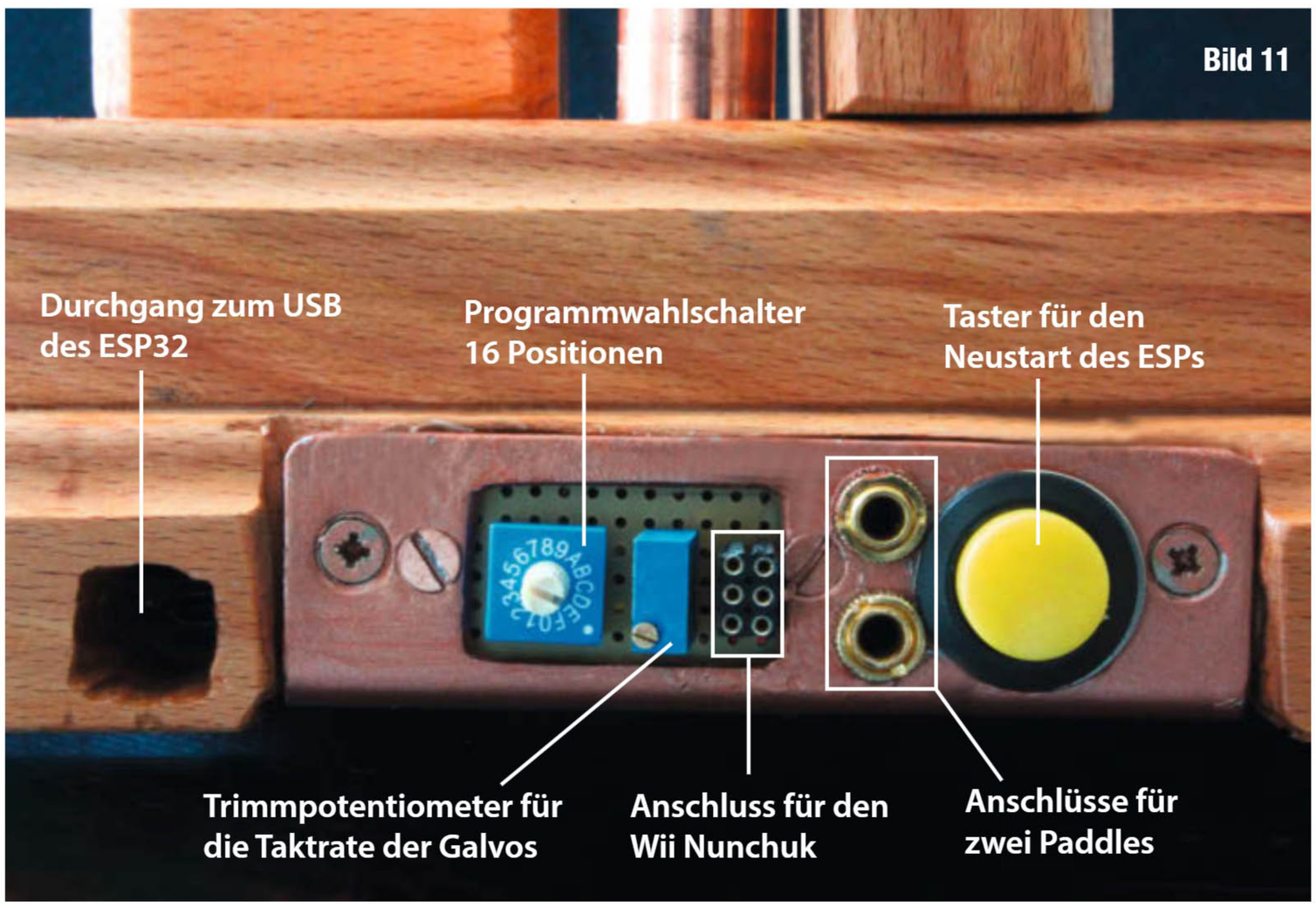
Bild 10: Der Schaltplan für die Erlenmeyerröhre

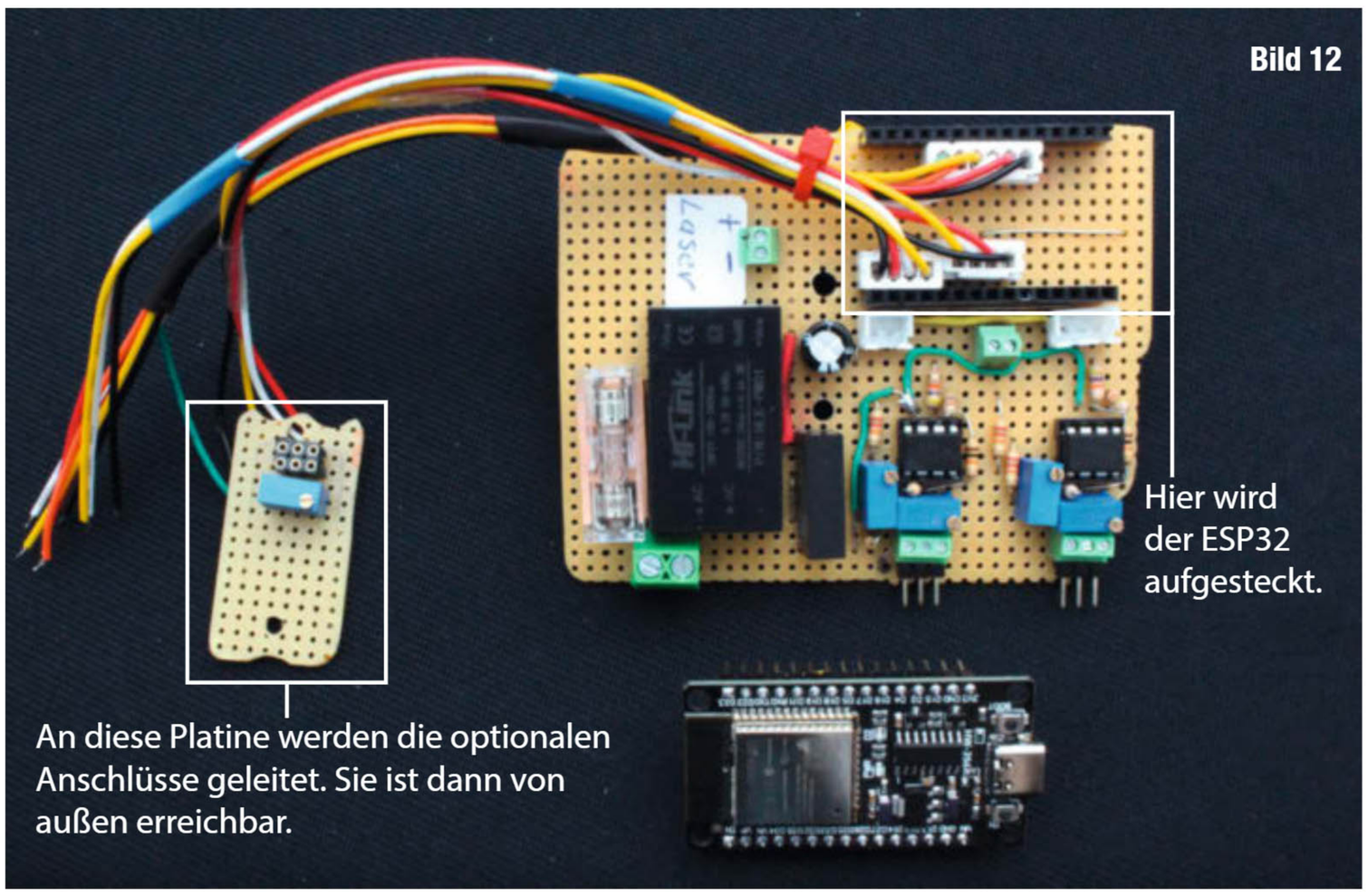
Vorsicht, Laser!

Dieses Projekt setzt einen besonders verantwortungsvollen Umgang mit Lasern voraus. Bei einem „normalen“ Laserprojektor, wie er etwa bei Lasershows zum Einsatz kommt, schaut man nur auf die Reflexion des Laserlichts auf der Projektionsfläche (etwa einer Zimmerwand oder Leinwand). Das Licht gelangt also nur indirekt ins Auge. Bei diesem Projekt hingegen blickt man im Prinzip direkt in Richtung des Lasers, nur durch die mattierte Bodenfläche des Erlenmeyerkolbens abgeschirmt. Diese darf deshalb auf keinen Fall Lücken aufweisen!

Zwar wirkt die sandgestrahlte Fläche als Diffusor, wodurch das gebündelte Laserlicht zerstreut wird und sich seine Energie auf eine größere Fläche verteilt. Außerdem bleibt der Strahl durch die Galvo-Bewegungen normalerweise nicht länger an einer Stelle stehen. Aber wenn die Leistung des Lasers zu hoch ist, dringt am Ende möglicherweise doch zu viel Laserlicht durch die matte Fläche des Kolbens. Dadurch kann das Auge irreparabel geschädigt werden. Ich rate deshalb dringend davon ab, über eine Laserleistung von 5 mW zu gehen!

Da auch solche 5-mW-Laser (Klasse 3R) bei falscher Handhabung Augenschäden verursachen können – etwa, wenn man direkt in den Strahl blickt –, erfordert der Nachbau dieses Projekts und sein Betrieb große Vorsicht und Erfahrung beim Umgang mit Lasern und erfolgt auch auf eigene Gefahr! Handelt verantwortungsvoll und lasst die Finger vom Nachbau, wenn ihr euch nicht firm genug fühlt; ihr sollt ja auch in Zukunft weiterhin problemlos Make-Artikel lesen können. Und: Denkt bitte auch an Kinder und Haustiere, die die Gefahren nicht selbst einschätzen können!





wenig Masse besitzen und dadurch leichter und schneller bewegt werden können, sind sie sehr dünn. Schon leichte Berührungen können zum Bruch führen, das Ergebnis zeigt Bild 8.

Anschließende Klebeversuche am Spiegel führen meist nicht zum Erfolg, da sich dadurch die Masse erhöht, Spiegelfläche verloren geht und ein genau achsengerechtes Ankleben kaum möglich ist. Die Folge ist dann nicht nur ein finanzieller Schaden, sondern auch eine nervige Unterbrechung des Projekts.

Aus Angst, dies könnte noch einmal passieren, etwa wenn ein interessierter Besucher die Spiegel mit den Fingern begutachten will, habe ich später noch eine Blende aus Aluminium gefertigt und damit die Galvo-Spiegel geschützt (Bild 9). Der Nebeneffekt: Die Blende stellt sicher, dass Laserstrahlen nur durch das runde Loch in Richtung Kolbenboden austreten und im Fall einer Fehlfunktion nicht überall in die Umgebung geschickt werden. Wie groß das Loch sein muss und wo es genau sitzt, muss man durch Experimente ermitteln. Wie alle Metallteile habe ich auch diese Blende mit einem Kupfer-Lackspray lackiert, um einen einheitlichen Look des Erlenmeyerkolbens zu erreichen.

Digital zu analog

Um den Galvo-Scanner anzusteuern, werden immer noch analoge Werte benötigt, also echte Spannungswerte, damit die Spiegel sich an die entsprechenden Positionen bewegen – und zwar rasend schnell, denn man will ja keinem Laserpunkt beim Wandern zuschauen, sondern die Grafiken müssen so schnell komplett gezeichnet werden, dass das Auge sie als stehendes Bild wahrnimmt. PWM-Signale (Puls-Weiten-Modulation) sind dafür nicht geeignet.

Bei meinem Laserspiel-Projekt entschied ich mich noch für die Verwendung zweier R2R-

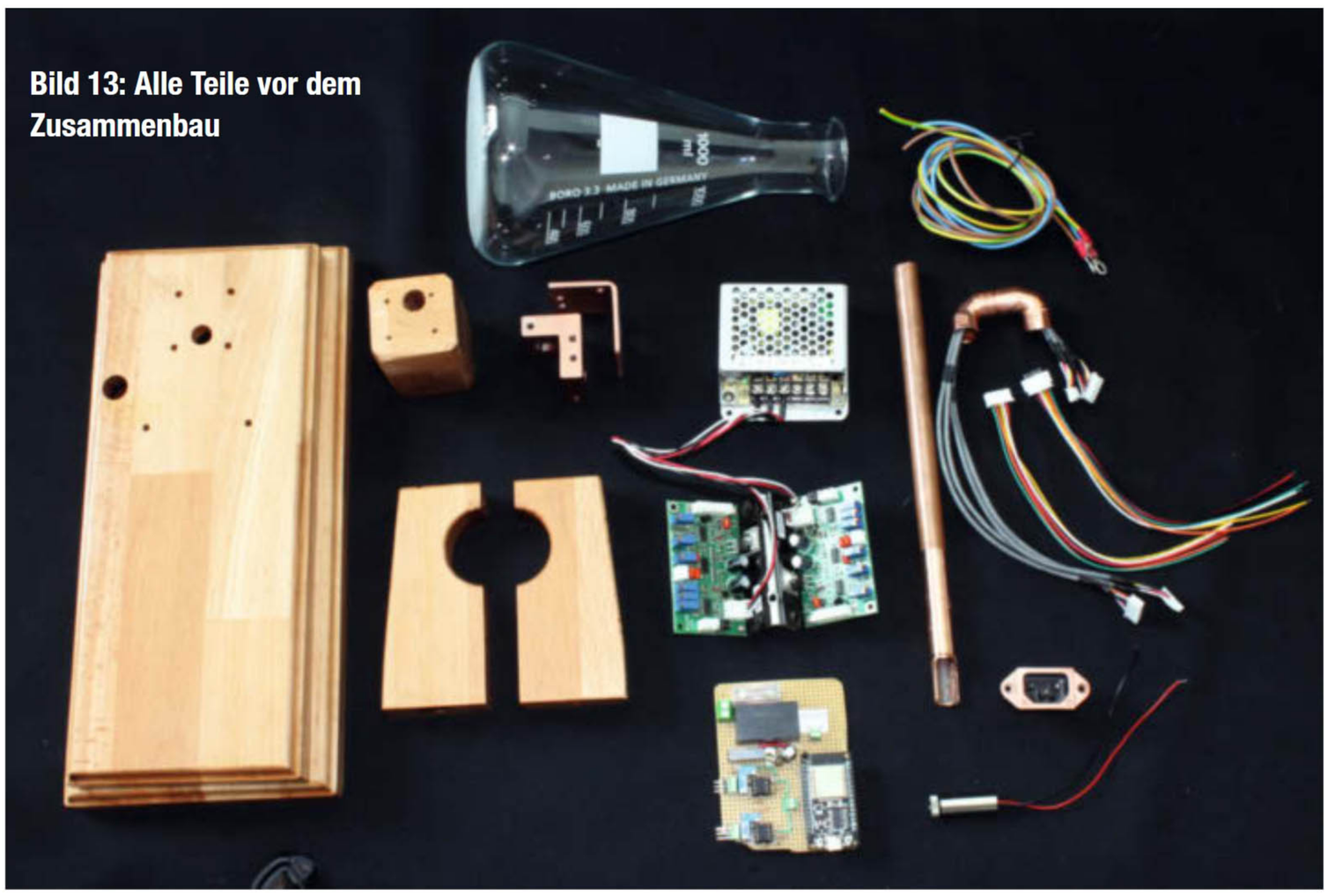


Bild 13: Alle Teile vor dem Zusammenbau


Bild 14

Netzwerke in Verbindung mit einem Arduino MEGA als beste und schnellste Lösung. Diese R2R-Technik war 2017 vor allem in puncto Geschwindigkeit den seinerzeit zur Verfügung stehenden ICs zur Digital-Analog-Wandlung (DAC) weit überlegen. Der Preis dafür bestand aber im hohen Schaltungsaufwand für dieses Widerstandsnetzwerk.

Inzwischen haben sich die Voraussetzungen geändert: Der ESP32-WROOM-32 verfügt heute über zwei echte DAC-Kanäle, die recht schnell einen digitalen Wert in eine analoge Spannung umwandeln können. Noch dazu ist der ESP32-WROOM-32 bereits für unter 5 Euro zu haben, also ein echtes Schnäppchen. Zudem sind dann auch noch zahlreiche nützliche Funktionen wie WLAN, Bluetooth, I^2C, SPI usw. mit an Bord, die zu verschiedensten Darstellungen auf dem Erlenmeyerkolben-Display einladen und ungeahnte Möglichkeiten eröffnen (dazu später mehr).

Die Schaltung

Immer noch ist es jedoch erforderlich, die vom ESP erzeugten analogen Signale (0 V bis 3,3 V) auf den Level anzuheben, den die Galvo-Elektronik erwartet: –5 V bis +5 V. Dazu verwende ich wieder die schon 2017 beschriebene Verstärkerschaltung mit den zwei Operationsverstärkern LM358 (Bild 10).

Die beiden DAC-Ports des ESP32WROOM-32 sind an den Anschlüssen GPIO 25 und GPIO 26 festgelegt. Deren Signale werden jeweils von einem LM358 auf ca. ± 9 V angehoben. Mit jeweils zwei Trimmpotentiometern lassen sich für jeden Galvo-Kanal die Ausrichtung (Offset) und die Größe der Auslenkung (Gain) der Galvo-Spiegel justieren. Die dafür benötigte Eingangsspannung von ± 12 V wird von einem HM-0512D aus der Netzteilspannung (HLK-PM01) von 5 V gewandelt.

Die weiteren Komponenten wie der Programmwahl-Schalter, zwei Paddel-Anschlüsse (Drehregler), ein Wii-Nunchuk-Anschluss und ein Anschluss für einen Soundverstärker sind rein optional und müssen (erst mal) nicht bestückt werden – sie sind Erbstücke des Laserspiel-Projekts und im älteren Artikel ausführlich beschrieben. Reine Bildschirmausgaben wie die analoge Uhr, die Grafik der NCC-1701-Enterprise und die weiteren Test-Anzeigen funktionieren auch ohne diese optionalen Komponenten.

Den Laser habe ich direkt an den GPIO 23 des ESP angeschlossen. Da ein ESP32 problemlos 20 mA über einen GPIO-Port verträgt, wobei selbst der verwendete 5-mW-Laser nur 10 bis 15 mA aus dem Port saugt, sollte dies kein Problem darstellen.

Prinzipiell könnte auch ein kleiner RGB-Laser für farbige Bilder verbaut werden, deshalb habe ich auch die entsprechenden Funktionen im Programm belassen (Download der Software über den Link in der Kurzinfo). Für mich standen dabei jedoch Aufwand und Nutzen in keinem guten Verhältnis mehr (auch wenn dies sicherlich ansprechend aussehen würde).


Bild 15

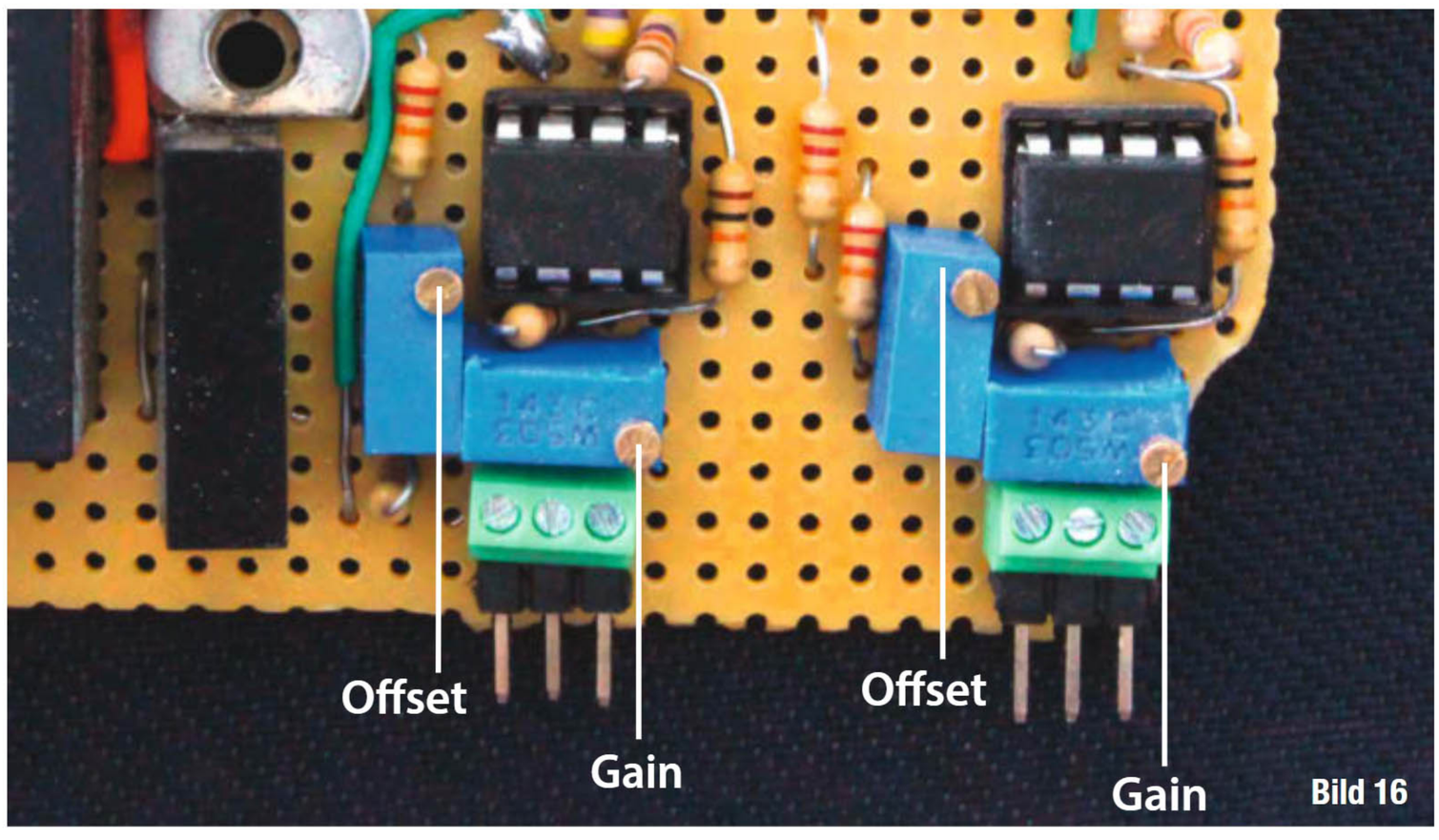


Bild 16

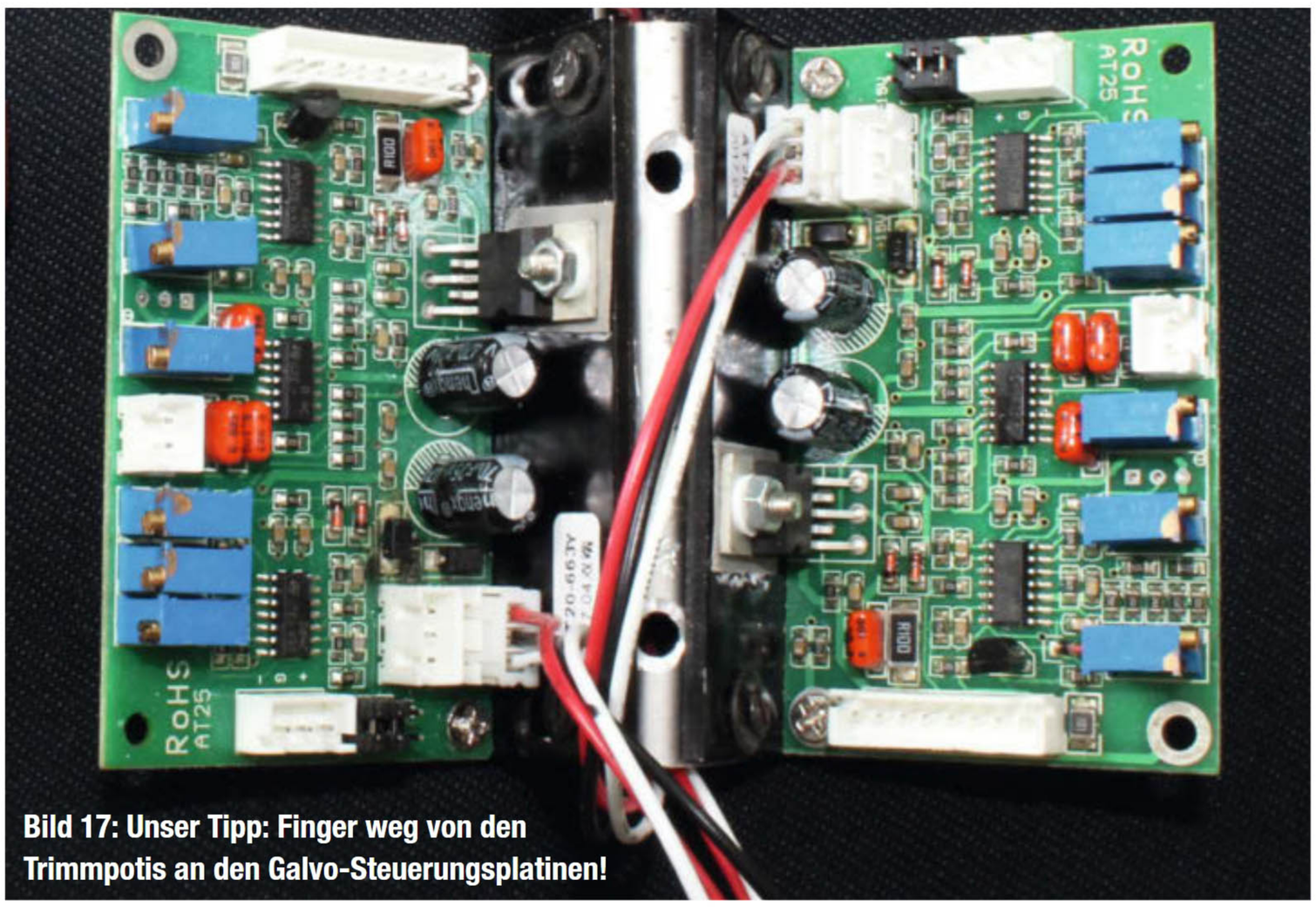


Bild 17: Unser Tipp: Finger weg von den Trimmpotis an den Galvo-Steuerungsplatinen!

Komfortfunktionen

Für mich ist das Sahnehäubchen des Erlenmeyerröhren-Projekts, dass die Elektronik mit einem ESP32 gesteuert wird und damit WLAN und Bluetooth ebenso frei Haus für eigene Projekte genutzt werden können wie I²C, SPI und mehr.

In den Quelltext habe ich bereits eingefügt, dass sich der ESP Uhrzeit und Datum über NTP aus dem Internet holt. Somit stimmt die angezeigte Uhrzeit immer auf die Sekunde genau. In meinem Projekt „Spielen ohne Grenzen" von 2017 war dies noch ein Manko, da dort zusätzlich noch ein RTC-Modul verbaut werden musste und die Uhrzeit dann manuell einzustellen war.

Da der damalige Arduino MEGA natürlich auch eine ganz andere Pinbelegung hatte, musste ich für den ESP nun alle Anschlüsse neu vergeben (wie oben aus dem Schaltplan zu ersehen). Zudem funktionierten einzelne Bibliotheken wie die Wii-Nunchuk-Bibliothek von 2017 nicht mehr auf dem ESP und mussten ersetzt werden. Die neue Bibliothek (esp32-wii-nunchuk-main.zip) ist auf das I²C-Protokoll aufgesetzt und nutzt die GPIO-Ports 32 und 33, die nichts anderes als SDA und SCL und somit einen I²S-Bus darstellen. Alle benötigten Dateien, den Quellcode und die verwendeten Bibliotheken gibt es wie immer zum Download über den Link in der Kurzinfo.

Als neue Funktion lese ich die Werte eines Trimmpotentiometers über den analogen Port 13 ein und steuere damit die Interrupt-Taktrate für den Galvo-Scanner. Dies bringt den Vorteil, dass nicht mehr umständlich über die Arduino IDE und den Programm-Upload ein Wert in den ESP eingetragen werden muss, sondern nur das Trimmpotentiometer zu verstellen ist. Der Effekt kann dann nach etwa fünf Sekunden beobachtet werden – das Potentiometer wird vom ESP nicht häufiger abgefragt, um keine Rechenzeit zu vergeuden.

Übrigens: Steht das vom Laser erzeugte Bild auf dem Kopf oder wird es seitenverkehrt dargestellt, so muss einfach nur an der entsprechenden Verbindungsleitung zwischen Galvo-Steuerung und LM358-Schaltung der Stecker um 180 Grad gedreht werden. Vermutlich wird dann anschließend das Bild zwar richtig dargestellt, ist aber in X- oder Y-Richtung verschoben. Dies wird dann mit den Offset-Trimmpotentiometern korrigiert.

Gestaltungsfragen

Alle (optionalen) Anschlüsse habe ich auf ein kleines Panel geleitet und dieses auf der Rückseite des Sockels angebracht. Dort stehen nun auch das Trimmpotentiometer für die Taktrate und eine Taste für den Neustart des ESP zur Verfügung (Bild 11).

Noch ein Tipp: Ich wollte die Steuerungsplatine so klein wie möglich gestalten und habe so alle Bauteile dicht aneinander platziert. Dadurch gestaltete sich die Verschaltung auf der Rückseite aber als echte Frickelarbeit! Die Leitungen und Lötpunkte liegen so dicht beieinander, dass es allein durch Lötspritzer und Flussmittel zu Fehlkontakten kam. Da zuletzt kein Platz mehr übrig war, musste ich die Stecker für Wii-Nunchuk, Paddles usw. sogar unter der ESP-Platine positionieren (Bild 12) …

Beim Nachbau sollte man zwischen den Bauteilen besser mehr Platz lassen und die Lochrasterplatine so groß dimensionieren, dass sie gerade noch ins gewünschte Gehäuse passt – bei mir war das der Gerätesockel, aber das kann man ja auch großzügiger gestalten.

Zusammenbau

Beim Sockel, der bei mir gleichzeitig das Gehäuse für die Elektronik bilden soll, hatte ich kurz überlegt, ob ich nicht alles aus Acrylglas fertige. Dagegen sprach jedoch, dass der Aufwand um einiges größer gewesen wäre und ich dann nichts hätte verstecken können. So entschied ich mich, die Reste meiner 27-mm-Buchenholz-Tischplatte zu verwerten, indem ich zwei Buchenholzlagen übereinanderleimte

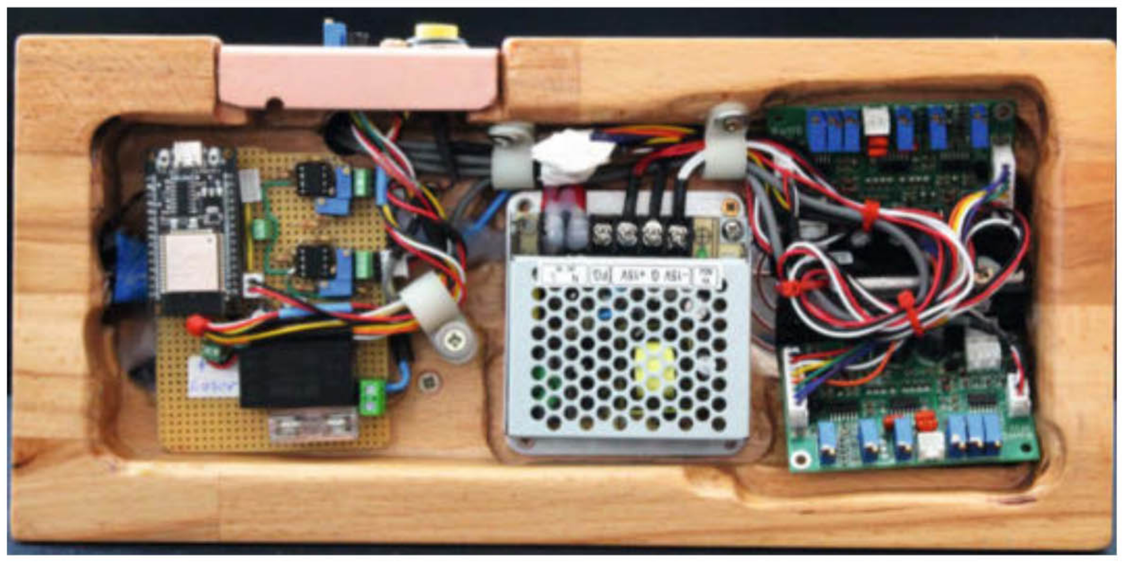


Bild 19: Blick von unten unter den Sockel: links die Platine mit dem ESP, in der Mitte das Netzteil, rechts die Steuerplatinen für die Galvos

Bild 18: Unter dem Gitter befindet sich eine Öffnung für die Kühlung der Platinen, die sich bei Bedarf auch noch mit einem Lüfter bestücken lässt.

(Bild 13, links). Die untere Lage hatte ich vorher mit der Stichsäge entkernt, sodass nur noch ein 25 mm breiter Rahmen übrigblieb. Aus der oberen Platte habe ich mit der Oberfräse so viel Holz ausgefräst, dass sich später alle elektronischen Komponenten leicht einbauen ließen.

Der Halteblock (50 × 50 × 70 mm) für den Galvo-Scanner und den Laser habe ich aus drei Lagen der besagten Buchenplatte zusammengeleimt und ihn später auf 70 mm Höhe abgesägt (Bild 14). Achtung: Die 12-mm-Bohrung für den Laser wird nicht mittig gesetzt, da der Galvo den Laserstrahl versetzt ausgibt. Meine Bohrung befindet sich 22 mm von dem einem und 13 mm vom anderen Rand entfernt.

Den Halterahmen für den Kolben hatte ich im ersten Versuch senkrecht geteilt, so ist er noch auf Bild 13 zu sehen. Dies erwies sich aber als unpraktisch, da sich dann der Glaskolben nur entfernen ließ, wenn auch eine Schraube an der Unterseite des Sockels entfernt wurde. Beim zweiten Versuch teilte ich das Loch für den Kolben (45 mm) waagerecht und schraubte in das Unterteil zwei M8-Stehbolzen ein, mit denen dann das Oberteil befestigt wird (Bild 15).

Die meiste Arbeit beim Zusammenbau der Anlage verursacht die Ausrichtung des Laserstrahls. Zuerst hatte ich den Erlenmeyerkolben und den Galvo-Scanner genau in einer Flucht verschraubt. Das war voreilig, denn es stellte sich heraus, dass zwischen Lasereintritt in den Galvo-Scanner und Laseraustritt ein deutlicher Versatz besteht, wodurch mein Laserstrahl den Kolbenhals streifte. Der Versuch, mit brachialer Gewalt den Laserblock in die gewünschte Richtung zu drücken, scheiterte, und ich musste wieder einiges demontieren und neue Schraubenlöcher bohren. Besser, man ermittelt durch Probieren zuerst die beste Position und bohrt erst dann.

Anschließend mussten an der selbst gelöteten Platine der Gain (die Größe des Bildes) und der Offset (die Lage des Bildes) justiert werden (Bild 16).

An den mitgelieferten Steuerungsplatinen der Galvos sind jeweils 6 Trimmpotentiometer angebracht (also insgesamt 12 Stück), die geradezu danach schreien, eingestellt zu werden (Bild 17). Ich rate jedoch dringend davon ab, denn dadurch können diese auch überlastet und zerstört werden.

Bild 22

Die Steuerungs-Platinen der Galvos sind bei diesem Projekt ohnehin die am stärksten mit Wärme belasteten Bauteile. Bei den ersten Versuchen wurden sie so heiß, dass ich lieber noch mit dem Forstnerbohrer ein (passives) Lüftungsloch gebohrt habe (Bild 18). Nun hält sich die Temperatur in akzeptablen Grenzen. Sollten die Platinen dennoch noch einmal zu heiß werden, werde ich der passiven Lüftung noch einen aktiven Lüfter spendieren.

Nach dem kompletten Zusammenbau sieht es im „Gehäuse“ ganz schön kompliziert aus (Bild 19). Dies liegt zum überwiegenden Teil an den zahlreichen Kabeln der Galvo-Steuerung. Also keine Angst! Es sieht komplizierter aus, als es wirklich ist.

Auf keinen Fall dürfen die Gerätefüße vergessen werden. Steht das Gehäuse direkt auf dem Tisch, findet keine Luftzirkulation statt, und die Galvo-Steuerungen sterben womöglich den Hitzetod. Ich habe Gummifüßchen verwendet, die gleichzeitig Vibrationen entkoppeln (Bild 20). Der Erlenmeyerkolben soll ja auch alltagstauglich sein und sein Geräusch niemanden nerven.

Das Endergebnis kann sich durchaus sehen lassen, wie man in Bild 21 sieht.

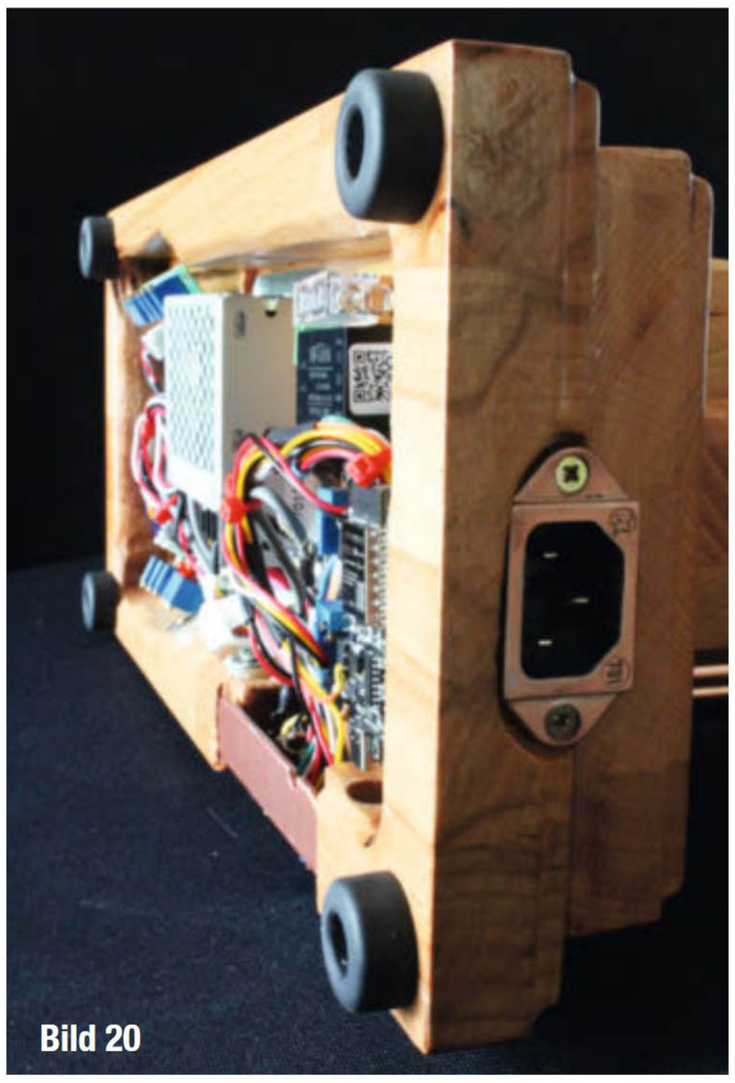


Bild 20

Experimente

Aber damit wollte ich es noch lange nicht bewenden lassen … Beim Anblick des Titelbilds zu diesem Artikel könnte man meinen, dass die bunten Erlenmeyerkolben nur zur Zierde auf dem Foto sind. Aber weit gefehlt! Wenn ich schon einen Kolben aus der Chemie in der Mache habe, warum ihn dann nicht auch mit chemischen Flüssigkeiten füllen und deren optischen Effekt nutzen?

Spannende Flüssigkeiten zum Experimentieren sind z. B. Wasser mit einem Spritzer Bohrwasser, stark verdünnte Milch, verdünnter Kirschsaft, Gingerale oder Tonicwater. Einfach mit allen Flüssigkeiten ausprobieren, die gerade greifbar sind. Doch es geht noch mehr: Als Erweiterung kann man am oberen Ende des Kolbens einen Ring aus RGB-LEDs so platzieren, dass der Laserstrahl immer noch hindurchpasst, und damit weitere Lichteffekte erzeugen. Der Kolben kann dann z. B. gleichzeitig als Ambient-Light und als Uhr dienen (Bild 22). Mehr dazu steht online (Link in der Kurzinfo).

Interessant könnte es auch sein, den roten Laser gegen einen schwachen UV-Laser auszutauschen. Zusammen mit fluoreszierender oder phosphoreszierender Flüssigkeit würde ich interessante Effekte erwarten. Deshalb bin ich sehr gespannt, was ihr aus diesem Projekt macht.

—pek

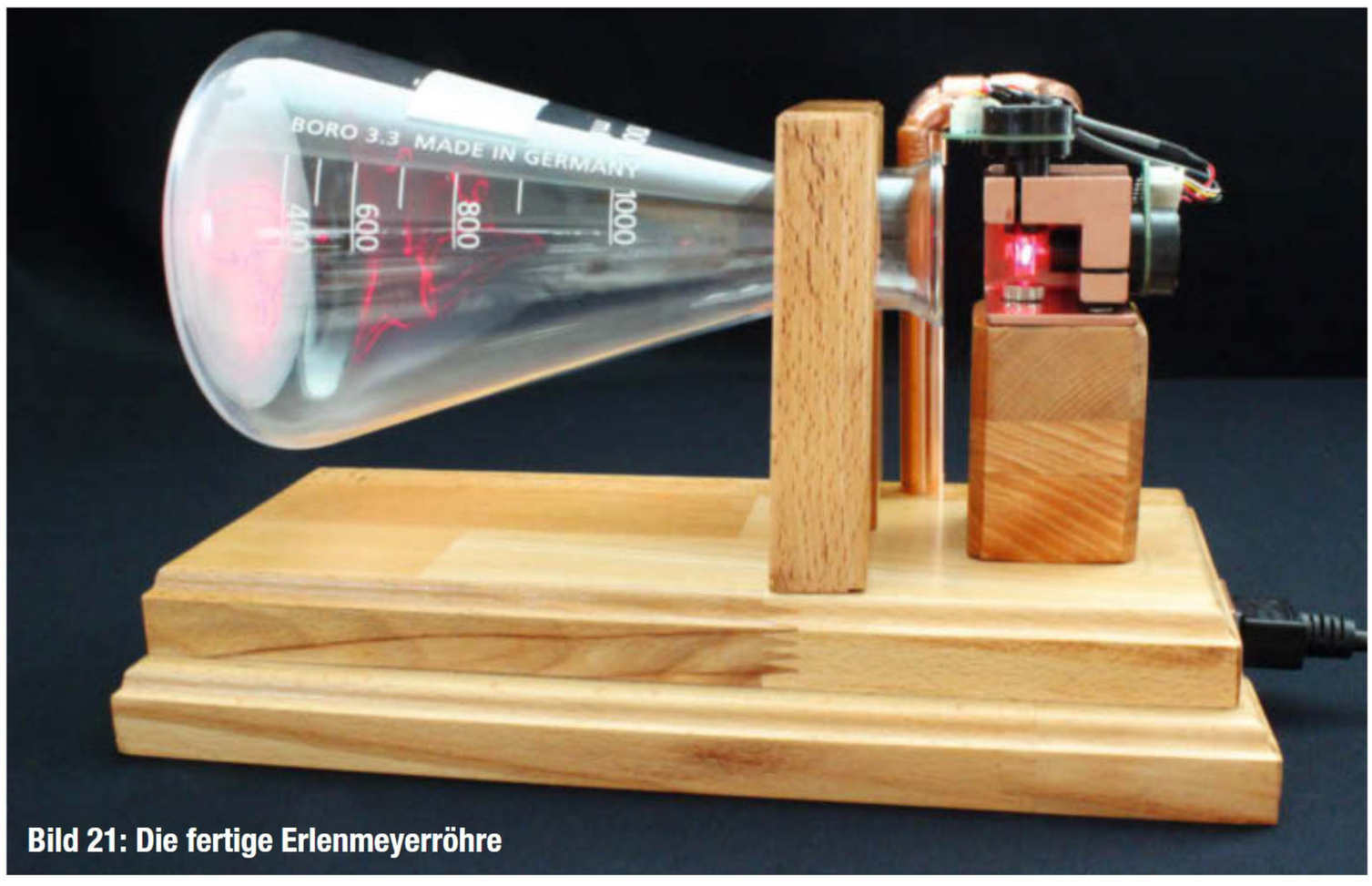
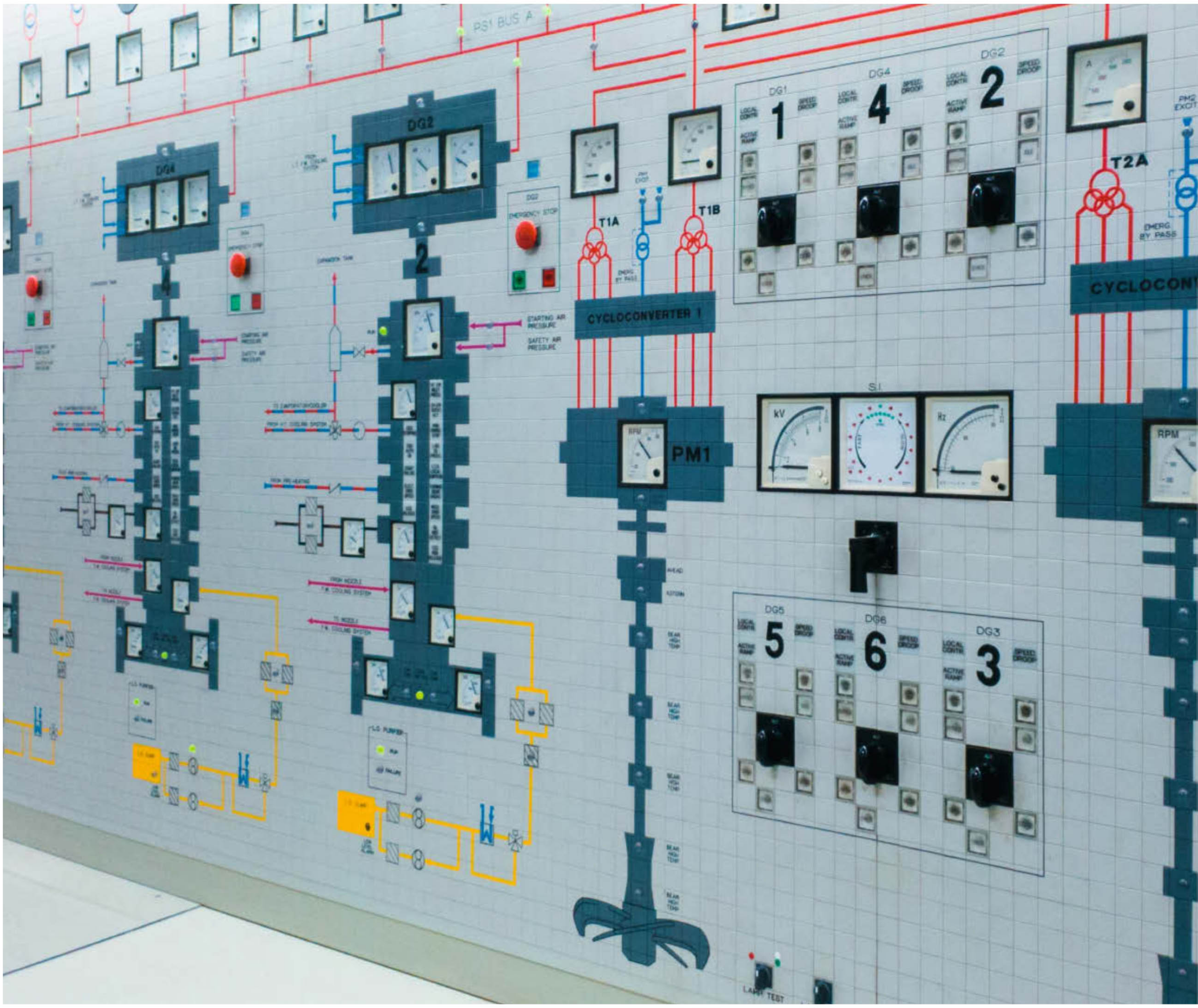


Bild 21: Die fertige Erlenmeyerröhre

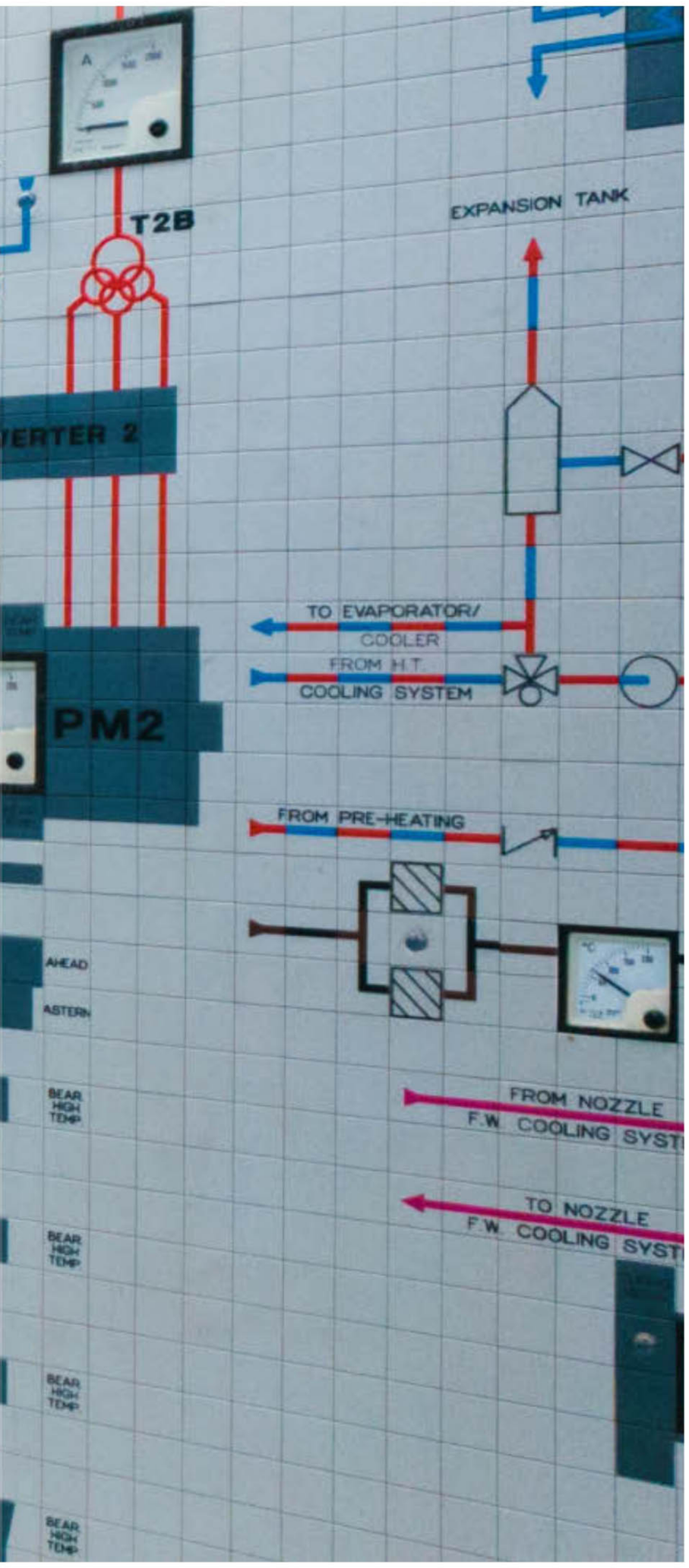


Ihor Koptilin / Shutterstock.com

Frontplatten mit System

Ein Mosaiksystem ist eine modulare Lösung für Frontplatten von Selbstbaugeräten und Prototypen. Mit OpenSCAD geplant und parametriert, sind die Frontplatten extrem flexibel einsetzbar.

von Gerd Michaelis



Kurzinfo

- Mit OpenSCAD modulare Systeme designen
- Struktur und Arbeitsweise in OpenSCAD
- Tipps zur praktischen Umsetzung an Beispielen

Checkliste

Zeitaufwand:
ab 2 Stunden

Kosten:
ab 10 Euro

3D-Druck:
Grundkenntnisse

Entwerfen:
Maße und Toleranzen

CAD:
Programmieren in OpenSCAD

Mehr zum Thema

- Peter König, et al., Gratis-3D-CAD für Maker, Make 4/22, S. 76
- Gerd Michaelis, Stella Risch, 3D-Drucke clever kleben, Make 6/20, S. 114
- Kurt Diedrich, DIY-Gehäuse schnell gebaut, Make 5/21, S. 112

Alles zum Artikel im Web unter
make-magazin.de/xqp2

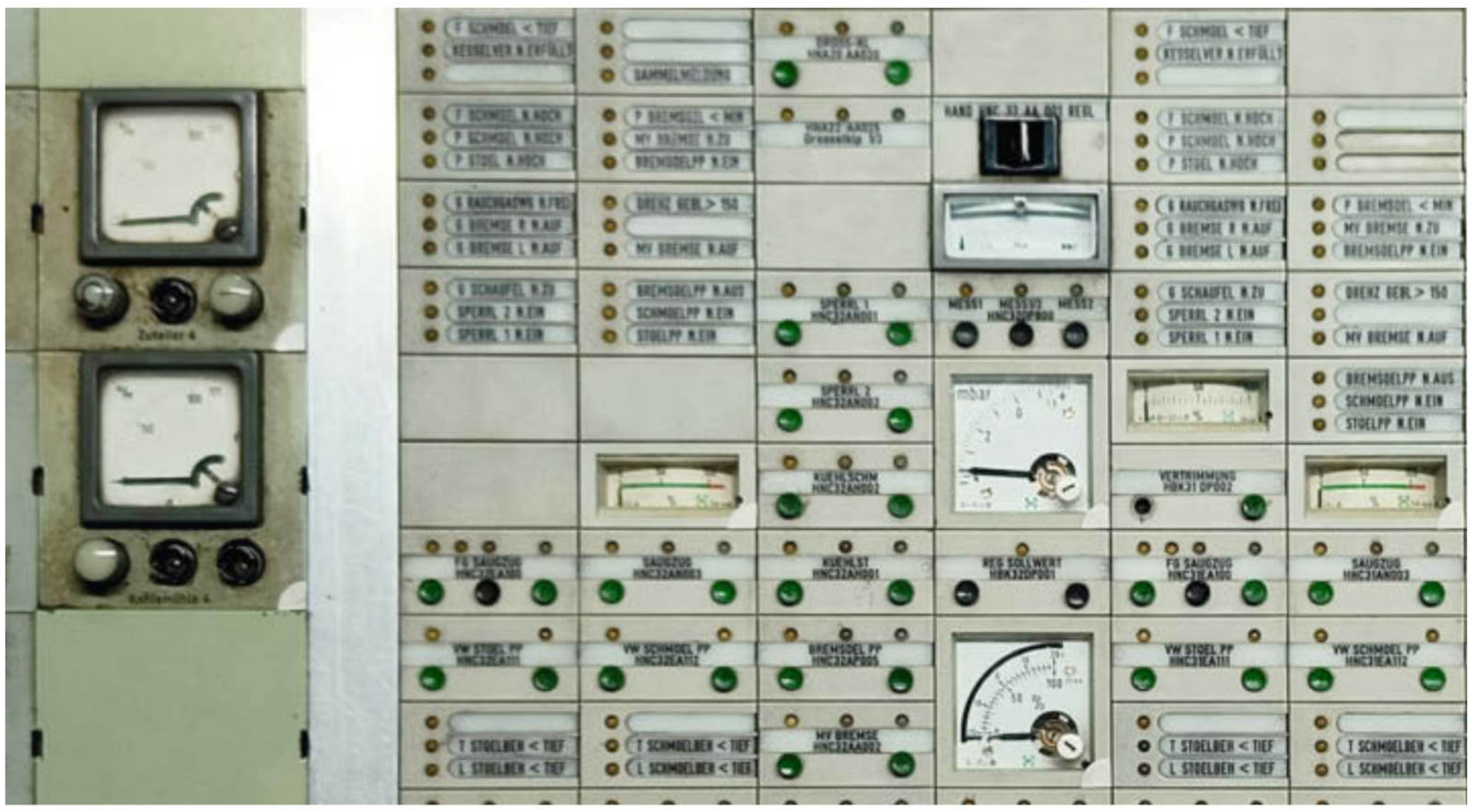


Bild 1: Mosaik-Panel in der DASA in Dortmund

Wer kennt ihn nicht, diesen unangenehmen Zustand in einem Hobby-Elektro(nik)-Bauprojekt: Die Schaltung ist getestet und funktioniert, jetzt fehlt zum guten Schluss nur noch eine Frontplatte für das Gehäuse. Hier stehen viele Werkstoffe (diverse Metalle, Kunststoffe, Holzmaterialien, Glas …) und noch viel mehr Vorgehensweisen bei der Bearbeitung zur Wahl, die aber alle entweder mühsam, schmutzig und fehlerträchtig sind (wie Bohren, Aussägen, Feilen) oder kostspielig und langwierig (wie die NC-Verfahren Fräsen, Lasern, Laserbeschriften).

Auch ich stand oft vor genau diesem Problem. Durch Anpassung von Installationsboxen, Altgehäusen, Blechkisten oder sogar von Saftkästen (siehe „Mehr zum Thema“ in der Kurzinfo) hatte ich zwar immer irgendwie ein Gehäuse für meine Schaltung, aber eben selten eine Frontplatte. Ich empfand es auf die Dauer als sehr mühsam, immer wieder neue Wege zu beschreiten, um an eine halbwegs ansehnliche Front zu kommen. Da muss es doch etwas Besseres geben! Und nachdem meine 3D-Drucker schließlich etwas Sinnvolles zu tun haben müssen, hat es sich angeboten, ein für alle Mal eine saubere, flexible und wieder verwendbare Lösung mit System zu konzipieren.

Mosaik-System

Die Idee dazu ist Mosaiktafeln entlehnt. Die wenigsten Leser dieses Hefts werden die noch selbst in freier Wildbahn gesehen haben, aber diese Übersichtswände waren vor der Zeit der Großbildschirme, Beamer und Videoleinwände für lange Jahre die einzige Möglichkeit, große und komplexe Systeme in Schaltwarten und Leitstellen abzubilden, also beispielsweise ein Versorgungsnetz für Strom, Gas oder Wasser oder einen Streckenplan bei der Bahn oder den städtischen Verkehrsbetrieben. Diese Mosaiktafeln heißen deswegen so, weil man in ein festes Metallraster kleine quadratische Elemente einclipsen kann, die dann entsprechend bedruckt oder beschriftet sind. Enthalten sind auch Anzeigen (Lämpchen, magnetische Stellungsanzeigen) oder Bedienteile (Taster, Schalter). Diese Elemente sind eben wie kleine regelmäßige Steinchen in einem großen Mosaik, daher der Name. Auf der zugänglichen Rückseite dieser Anzeigewände lassen sich die Elemente dann verdrahten.

Aus dem realen Leben sind solche Tafeln heute nahezu vollständig verschwunden, obwohl es laut meiner Internetrecherche noch Hersteller dafür gibt, wie z. B. STRADA electro-

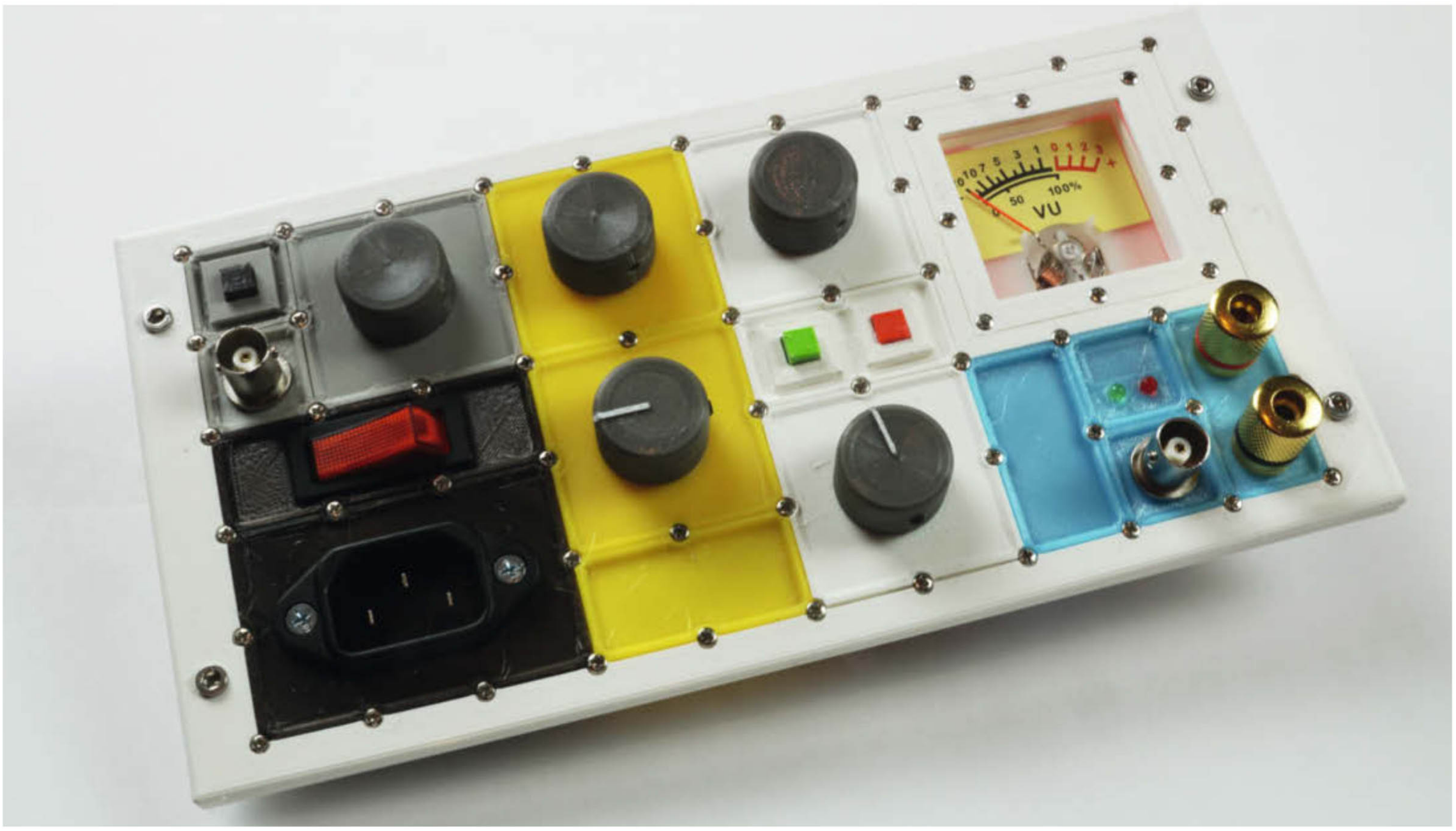


Bild 2: Die Frontplatte meines Laborverstärkers war das erste Projekt mit dem System.

nic in Cham, MLP in Berlin und ZPAS in Polen. Ein Besuch auf deren Websites (siehe Kurzlink) lohnt sich und ist eine kleine Zeitreise. Die heutigen dynamischen Visualisierungen lassen sich mit Videowänden sehr viel besser oder überhaupt nur so darstellen – und das ist meist auch noch preiswerter und zudem ungleich flexibler.

Meine Projekte benötigen solche Flexibilität nicht; meine Frontplattenlösung besteht genau wie eine Mosaiktafel aus einem Rasterrahmen und darin einsetzbaren Elementen, aber alles stets anpass- und 3D-druckbar. Wie beim großen Vorbild auch, können die Einsätze in ihrer Größe und Form variieren, damit nicht nur kleine Schalter, Steckbuchsen und LEDs, sondern auch größere Komponenten darin untergebracht werden können, wie etwa analoge Anzeigeinstrumente oder ganze TFT-Displays. Aus dem Raster müssen dafür dann nur die nicht benötigten Stege herausgetrennt werden – oder sie werden gar nicht erst gedruckt.

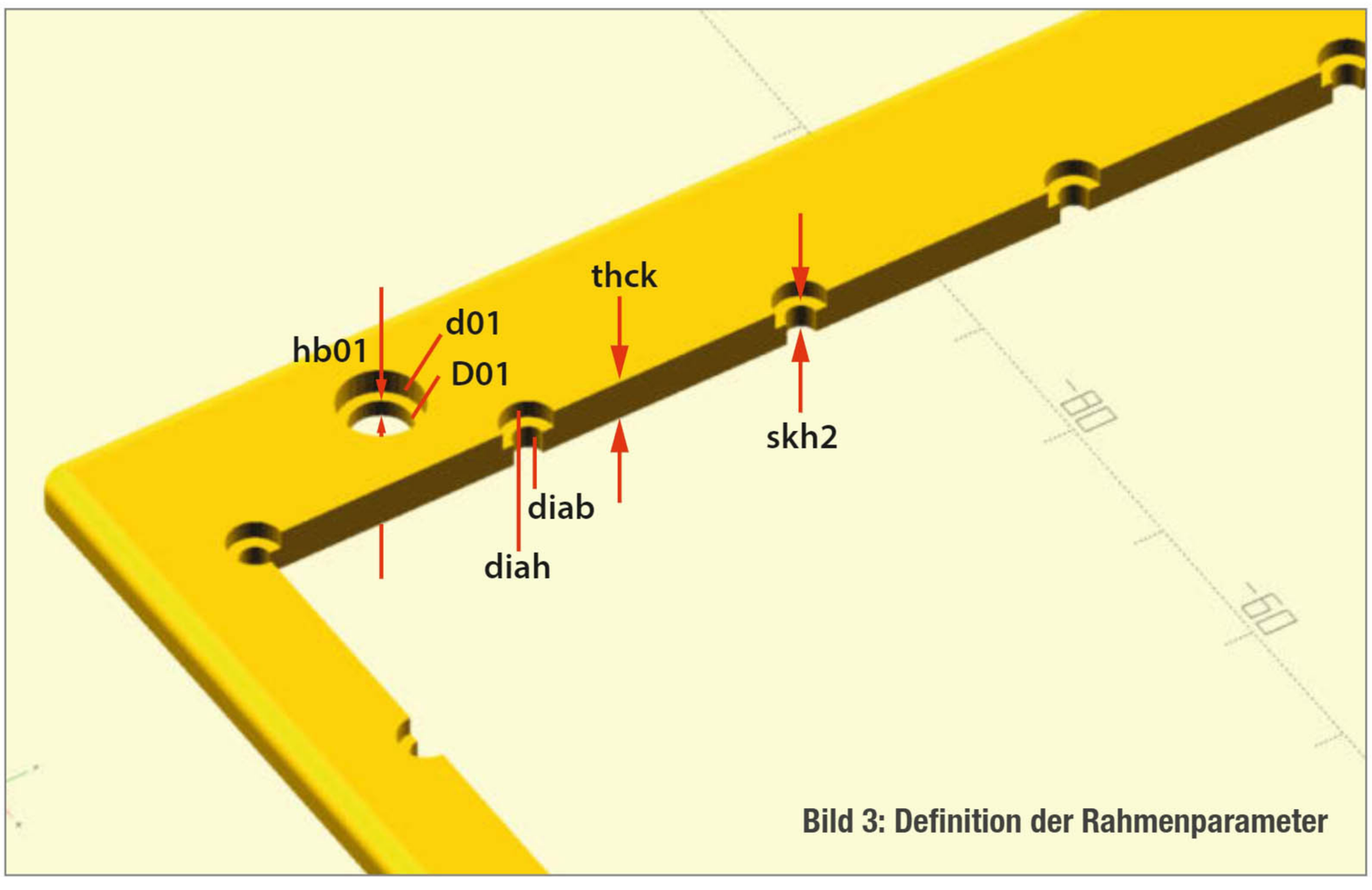


Bild 3: Definition der Rahmenparameter

Im Gegensatz zu einer reinen Anzeigewand muss das jeweilige Frontplattensystem-Element recht fest im Rahmen sitzen, wenn man einen Hebelschalter, Buchsen oder Polklemmen einbaut. Um diese Kräfte aufzunehmen, verwende ich viele kleine Schrauben, die die Elemente an ihren Ecken oder Kanten im Raster fixieren. Ich habe meine Module für selbstschneidende 2-mm-Schrauben entworfen, damit möglichst wenig Platz für die Befestigung verloren geht. Im OpenSCAD-Skript ist dies jederzeit leicht anpassbar. Ursprünglich beinhaltete das Skript auch eine Option für leichter erhältliche M3-Schrauben, aber erstens sind deren Schraubenköpfe meinem Gefühl nach für das von mir gewählte Rastermaß viel zu groß und zweitens wollte ich das lästige und unschön schmierende Gewindebohren in PLA vermeiden. Man sollte sich ausreichend große Mengen der jeweiligen Schrauben besorgen, denn es werden etliche benötigt!

OpenSCAD

Ich habe mir drei OpenSCAD-Skripte (siehe Kurz-Link) angelegt, mit denen die Stützstruktur (Raster.scad), der Rahmen (Rahmen.scad) und die Elemente (FrontElement_nxm.scad) schnell konstruiert werden können. Alle Dimensionen, insbesondere die Außenmaße (Breite b und Höhe h) sind über Parameter frei definierbar, damit man das Frontplattensystem an beliebige Gehäusemaße anpassen

kann, innerhalb derer das Raster dann zentriert, und gegebenenfalls mit Offsets (x_off und y_off) versehen, positioniert wird. Siehe Bilder 3 und 4.

Weiterhin wird in den Skripten das Rastermaß `segment` in horizontaler und vertikaler Richtung in Millimetern definiert, sowie die Größe des Rasters in Breite und Höhe als ganzzahlige Faktoren („x“, „y“) angegeben. Wenn man beispielsweise eine Front für eine 19-Zoll-Box braucht, deren Höheneinheit (1HE) also 44 mm beträgt, dann könnte sich mit etwas Rahmenüberstand oben und unten sowie zwei Segmenten übereinander (y=2) eine Segmentkantenlänge von 19 mm als günstig erweisen. Dies weicht gegenüber dem bei Mosaiktafeln üblichen Wert von 25 mm ab, was man für Hobbyprojekte aber vernachlässigen kann. Bei 2HE oder 88 mm Fronthöhe sind dann vier Segmente übereinander und bei 3HE oder 132 mm sechs Segmente übereinander möglich. Auf der Breite von 19 Zoll (483 mm Außenmaß, etwa 450 mm lichte Weite) lassen sich in der Praxis und unter Beachtung der Einbausituation dann 21 solcher Segmente bequem nebeneinander platzieren.

„The sky is the limit“, bei 3D-Druckern aber ist der Druckraum dafür in der Regel zu klein. Deshalb gibt es die für 19 Zoll angepassten Rahmen- und Rastervorlagen auch aufgeteilt als linke und rechte Hälfte, die jeweils gerade so eben auf ein Druckbett von 250 mm Breite passen.

Die Front besteht aus dem Raster und einem darübergelegten Rahmen. In beiden SCAD-Dateien können über die Parameter `xnn`, `ynn`, `dnn`, `Dnn` und `hbnn` (nn=1...16) unterschiedliche Bohrungen definiert werden, damit man die Front ans Gehäuse anschrauben kann – oder zusätzliche Elemente anbringen kann, etwa Griffe. Die Nummerierung der Bohrungen ist in beiden Skripten identisch, man kann sie also leicht hin- und herkopieren – für nicht benötigte Bohrungen kann man einfach Werte außerhalb der Frontfläche eingeben.

Das Raster hat auf der Unterseite Stege zur Stabilisierung der Front. Der Wert für `intr`, also die Tiefe der Stege, sollte bei großen Fronten erhöht werden. Damit ragt das Raster zwar weiter in das Gehäuseinnere hinein, aber die Front wird wesentlich belastbarer. Einbauteile mit erhöhten Kraftauswirkungen wie z. B. Polklemmen sollten aus demselben Grund eher am Rand und möglichst nicht in der Mitte der Front angeordnet werden.

Das Raster wird (Bild 4) seitenrichtig definiert und erst beim Aufruf der Darstellung (F5 für die Vorschau oder F6 zum Rendern) für die einfache Druckbarkeit 180 Grad um die y-Achse gedreht. Dabei muss man noch die Dicke „thck“ des umlaufenden Randes berücksichtigen, damit die flache Oberseite auf der x-y-Ebene liegt. Aber selbst wenn man das vergisst, korrigieren viele Slicer das

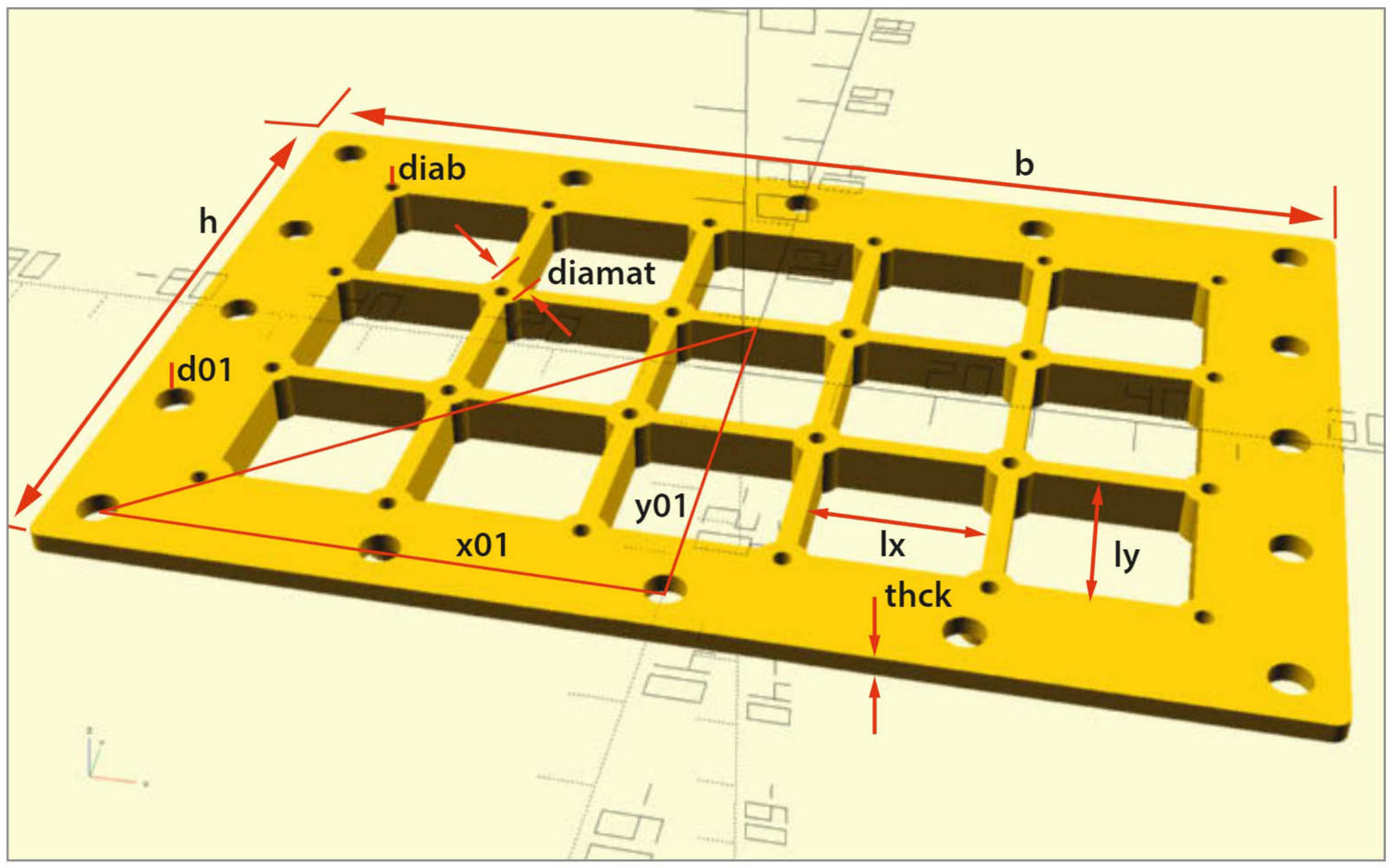


Bild 4: Das Raster und die passenden Werte

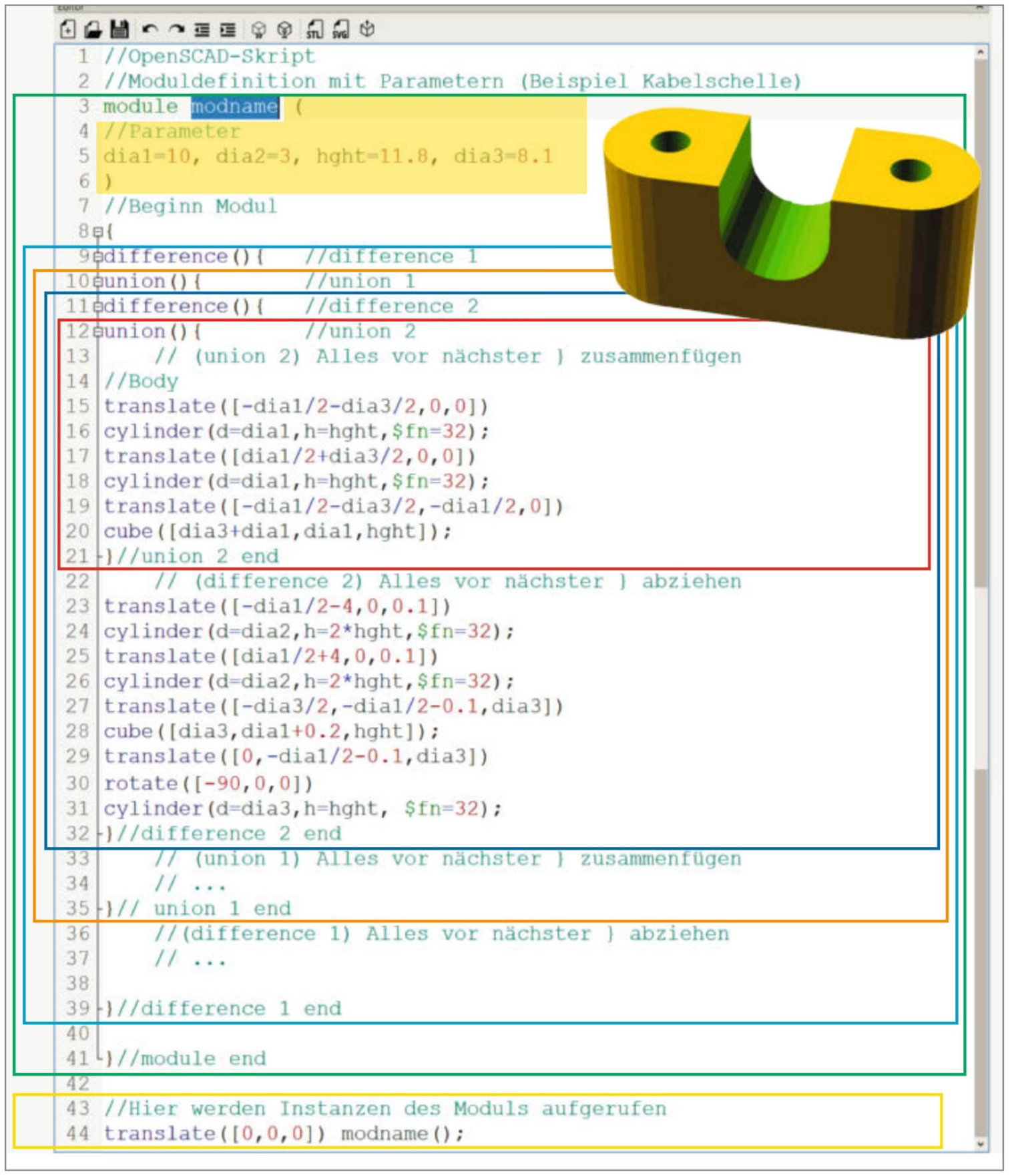


Bild 5: Die Struktur der SCAD-Dateien

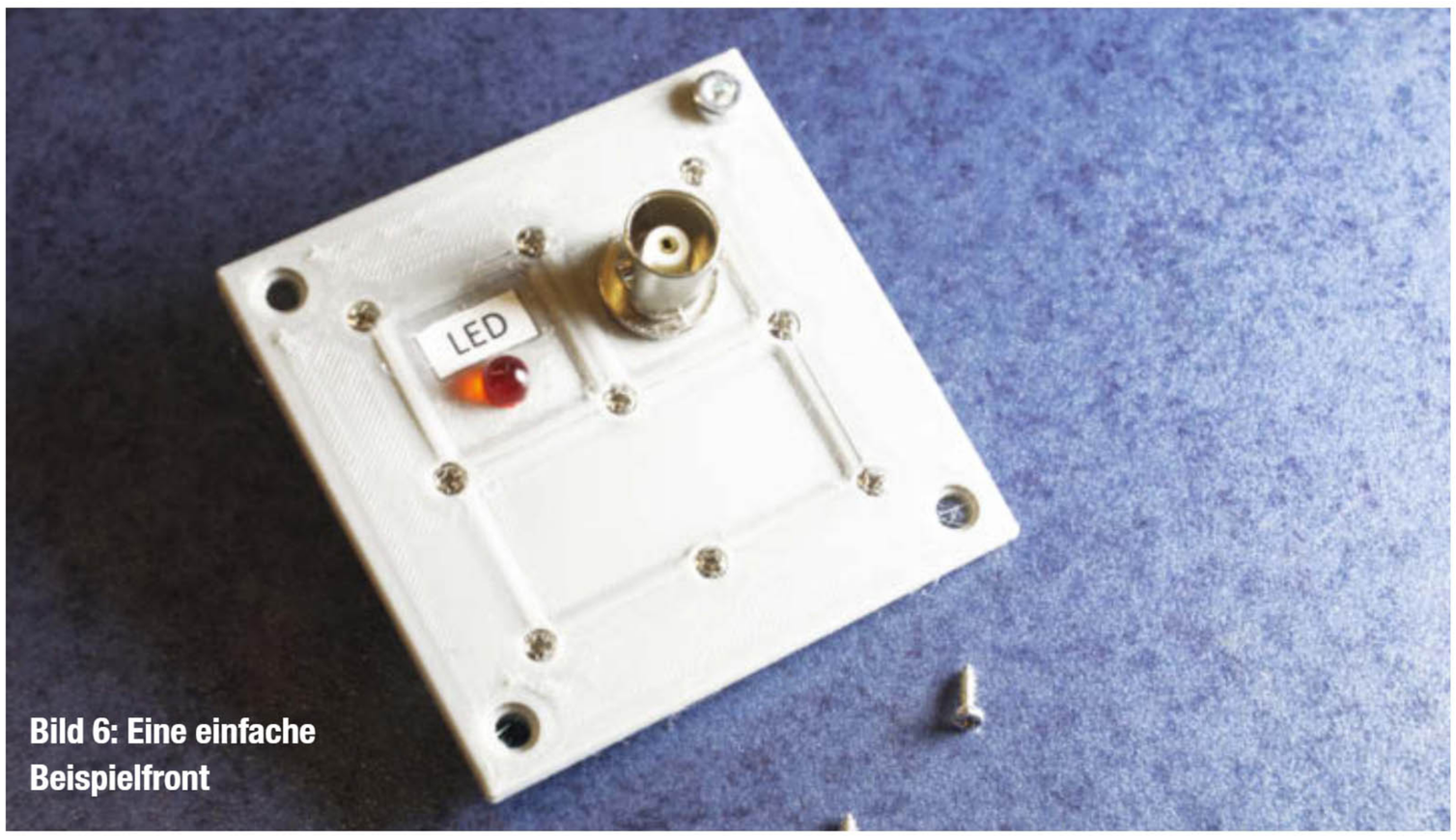


Bild 6: Eine einfache Beispielfront

entweder selbstständig oder bieten eine Funktion für die Verschiebung entlang der z-Achse an.

Struktur der OpenSCAD-Skripte

Alle OpenSCAD-Skripte sind ausgehend von derselben Kopiervorlage (Bild 5), die ich immer und für alles verwende, gleich aufgebaut: Innerhalb der Moduldefinition `module(){...}` (grüner Rahmen) werden

– zunächst die Parameter zwischen den runden Klammern definiert (gelb markiert),

dann von innen nach außen:

– mit der Funktion `union(){...}` Elemente zusammengefügt („union2“, innerer roter Rahmen)
– mit `difference(){...}` Elemente davon subtrahiert („difference2“, innerer blauer Rahmen)
– mit `union(){...}` wieder Elemente dazugefügt („union1“, äußerer roter Rahmen)
– mit `difference(){...}` noch mal welche abgezogen („difference1“, äußerer blauer Rahmen)
– das Modul selbst zur Berechnung aufgerufen (gelber Rahmen).

In dem Beispiel aus Bild 5 wird eine Kabelschelle erstellt. Sie setzt sich aus zwei stehenden Zylindern mit einem Quader dazwischen, wovon dann zwei stehende und ein liegender Zylinder sowie ein weiterer Quader abgezogen wurden, zusammen.

Als geometrische Formen werden in den Skripten zur Frontplattenerstellung nahezu ausschließlich Quader `cube([x,y,z])` und Zylinder `cylinder (d,h,$fn)` benötigt, welche mit dem vorangestellten Befehl `translate([x,y,z])` verschoben und mit `rotate([x,y,z])` gedreht werden können. Darüber hinaus gibt es nur die logische Entscheidung `if(bedingung){...}` und die Schleife `for(i=[anfang:schritt:ende]){...}`, damit die Anpassungen und wiederkehrende Aufgaben leicht erledigt werden können.

Die einzige kompliziertere Form ist ein Polyeder, welches über seine Eckpunkte „points“ und die dazwischen aufgespannten Flächen „faces“ definiert wird. Diese umgekehrte stumpfe Pyramide wird oben von dem quaderförmigen Frontplatteneinsatz subtrahiert, um einerseits Material zu sparen und andererseits die Frontplatte nicht zu dick werden zu lassen, denn die meisten Bedien- und Anzeigeelemente sind für Metallfrontplatten entworfen worden, die nicht dicker als etwa 2 mm sein dürfen. Durch diese Subtraktion entsteht ein abgeschrägter Rahmen, der dem Element trotz der geringen Plattendicke („plate“) eine hohe Stabilität verleiht und wegen fehlender scharfer Innenecken auch noch gefällig aussieht.

Die Höhe des Rahmens „thck“ und seine Breite oben und unten („wo“,„wu“) lassen sich individuell nach Vorliebe wählen. Da Rahmen und das Frontelement in separaten Dateien definiert sind, muss man darauf achten, dass die Dicke („thck“) in den Skripten für Rahmen und Element identisch angegeben wird, weil beide auf dem Raster aufliegen und oben abschließen sollen. Aus demselben Grund sollten auch die Kantendefinitionen „chmfr“ (Radius) und „chmtyp“ (Kantenform) für angefaste oder gerundete Kanten in den Skripten für Rahmen und Raster übereinstimmen.

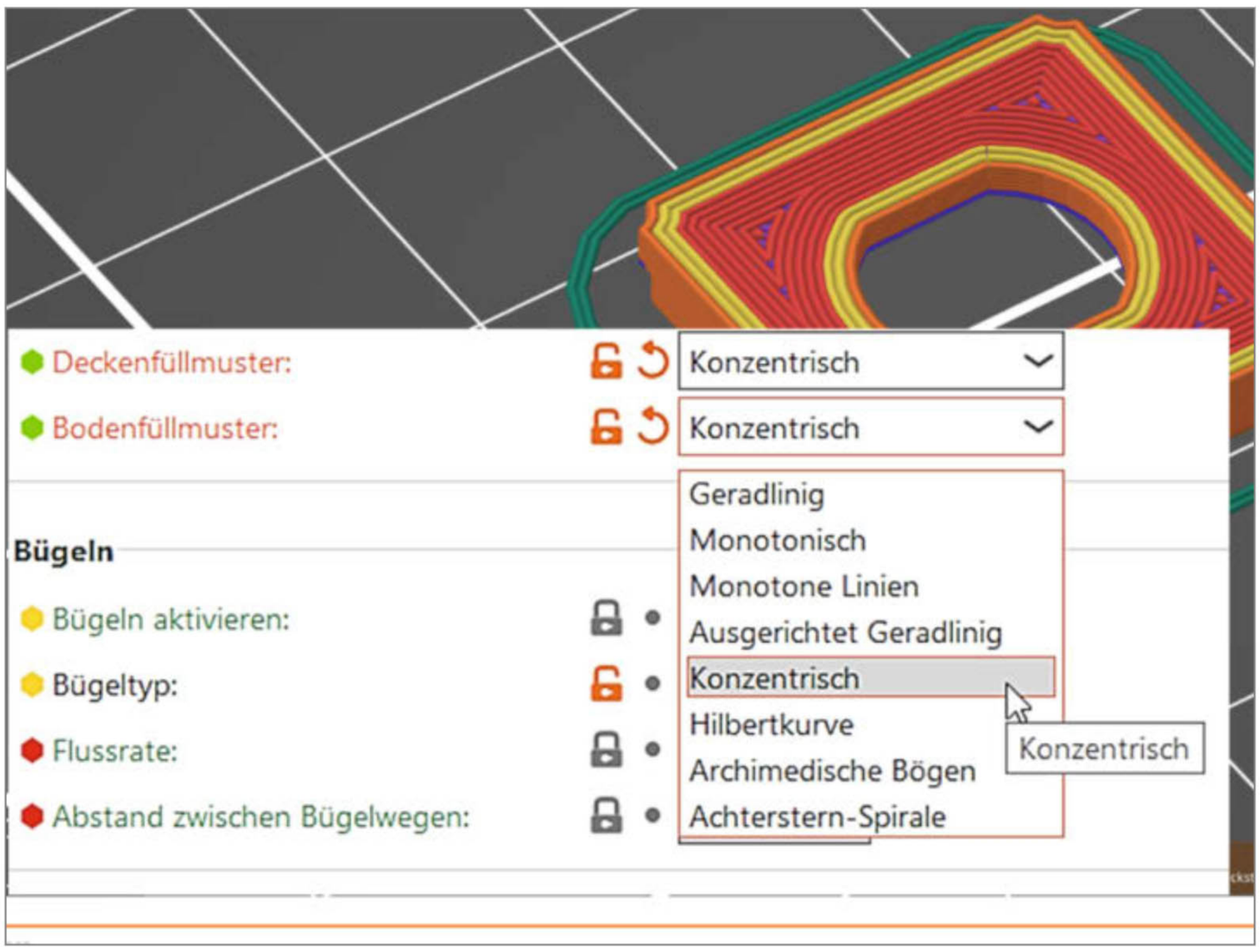


Bild 7: Oberflächenkosmetik im Slicer

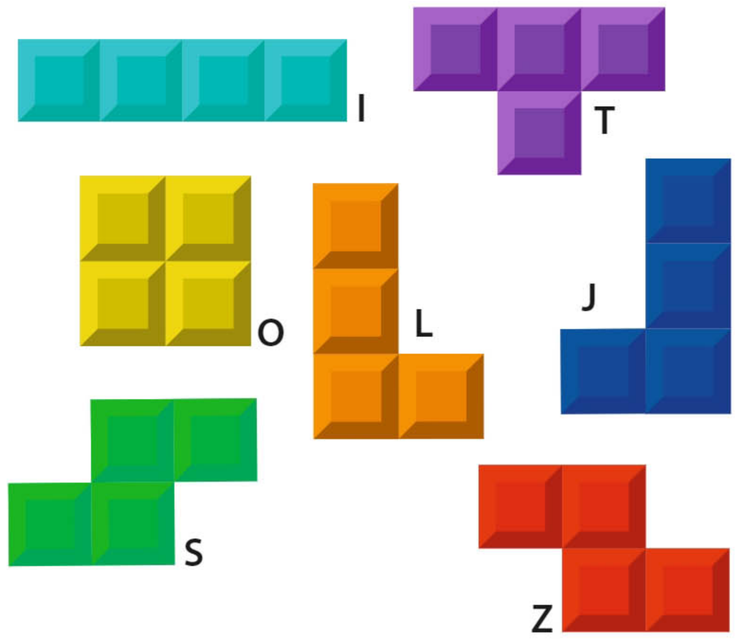


Bild 8: Game on!

Der Parameter „segment“ ist im Raster und im Rahmen mit 19 mm definiert, bei den Elementen eine Winzigkeit weniger, damit die Elemente wegen der unvermeidlichen Ungenauigkeiten der 3D-gedruckten Kanten nicht im Rahmen klemmen. Die Maße der Schrauben sind in den Skripten für den Rahmen und die Elemente identisch.

Ausschnitte

Was jetzt noch fehlt, sind die spezifischen Bohrungen und Ausschnitte für die jeweiligen Bedien-, Anzeige- und Anschlusselemente, die ein Gerät üblicherweise nun mal so benötigt. Diese lassen sich einfach innerhalb der Schachtelung „difference 2“ eintragen und damit sozusagen digital aus der Frontplatte heraussubtrahieren.

In manchen Fällen muss wieder was hinzugefügt werden, was noch in „union 1“ erfolgen muss. Im Kommentar unter „difference 2“ wird darauf hingewiesen. Beispiele sind in der Textdatei „Ausschnitte.txt“ (Download über Kurzlink) für Schalter, Taster, BNC-Buchsen usw. zu finden, um nur einige häufig vorkommende zu nennen.

Auch Halterahmen für LEDs oder Digitalanzeigen kann man so drucken, um diese Bauteile dann in einem Ausschnitt der Frontelemente zu befestigen. In der Textdatei ist angegeben, an welcher Stelle die Anweisungen jeweils in das OpenSCAD-Skript eingefügt werden müssen. Im Skript „FrontElement_nxm.scad“ für die Frontplattenelemente sind viele davon bereits integriert, allerdings auskommentiert. Wenn man die Zeichen „/*“ für den Beginn und „*/“ für das Ende des Kommentarbereichs („//“ kommentiert eine einzelne Zeile aus) entfernt, werden die dazwischenliegenden Befehle aktiv. Mit der Zeit wächst so eine ganze Bibliothek von benötigten Elementausschnitten heran und man kann dann für ein nächstes Projekt wieder schnell darauf zurückgreifen.

Eine weitere angenehme Eigenschaft dieses Frontplattenkonzepts ist, dass man auch später noch jederzeit einzelne Elemente wieder lösen und aus der Front herausnehmen kann, um entweder eine defekte Komponente oder das ganze Element zu tauschen; man muss dafür nicht das Gerät zerlegen, sondern nur ein paar Schrauben herausdrehen. Einzige Voraussetzung ist, dass die jeweilige Komponente innen mit ausreichend langen Leitungen angeschlossen ist. Das sollte in den allermeisten Projekten möglich sein – nur wenige Hobbybastler dringen in Frequenzbereiche vor, in denen die Signallaufzeiten, Dämpfungen oder Abstrahlungen aufgrund von Leitungslängen nicht mehr vernachlässigbar und inakzeptabel sind.

In Bild 6 erkennt man die zusammengesetzte Beispielfront mit den zwei 1×1-Ele-

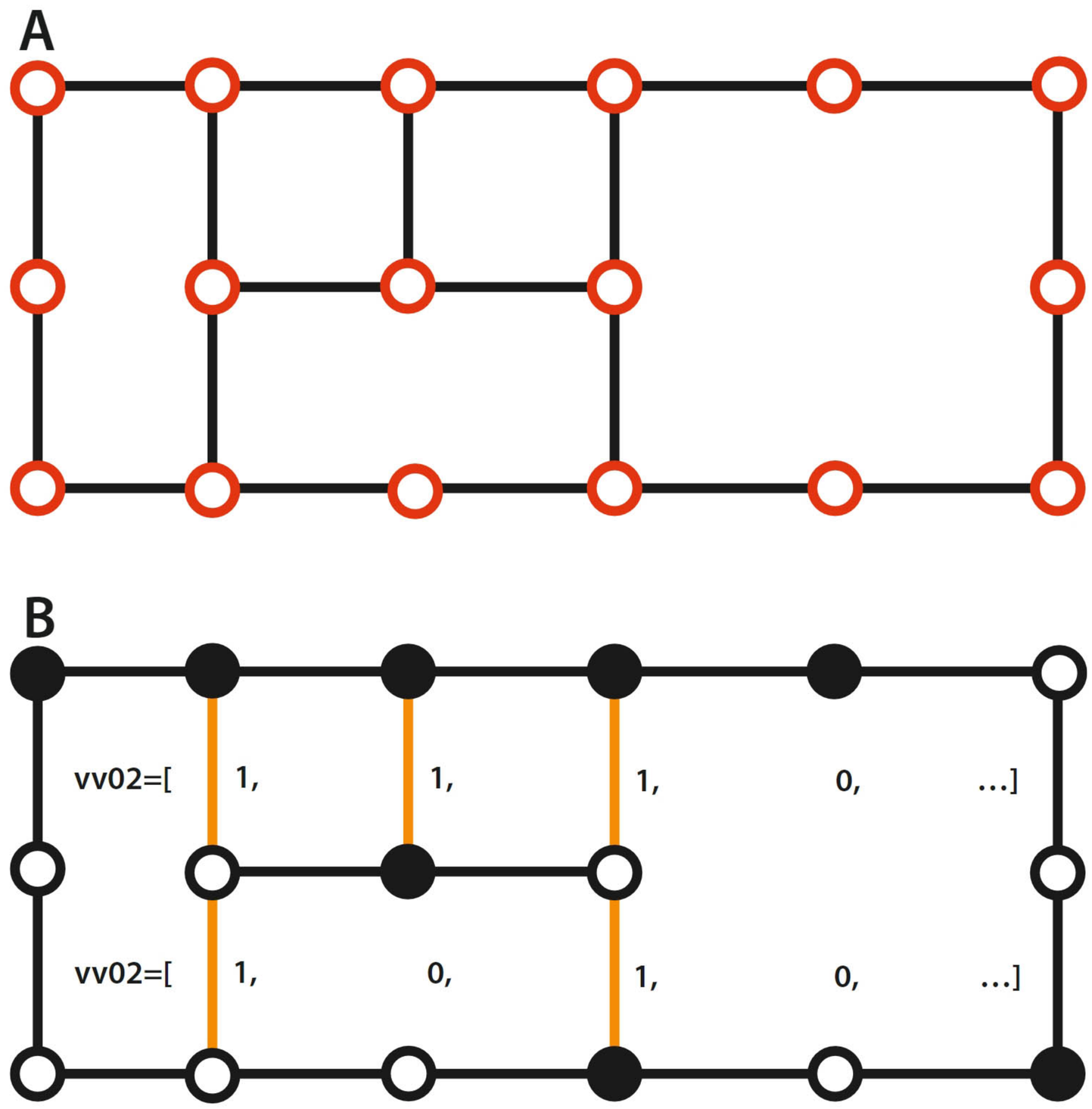


Bild 9: Vektordefinition

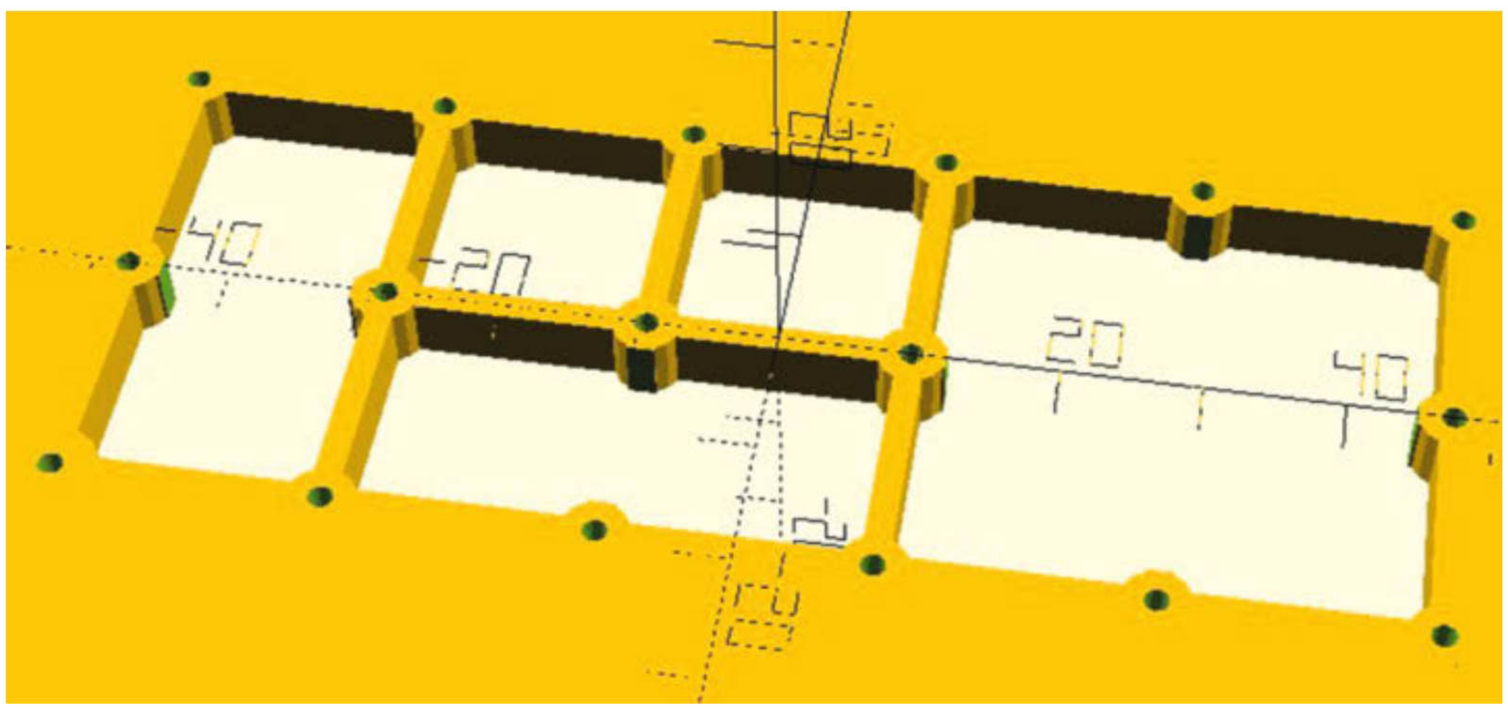


Bild 10: Das Resultat der Vektoren in OpenSCAD

menten und unten dem 2×1-Element sowie den verwendeten Schrauben. Die Oberflächen lassen sich durch geeignete Druckeinstellungen noch verschönern, indem man entweder die Anzahl der Wandschleifen erhöht, und so das schräge Füllmuster vermeidet, oder statt der Voreinstellung „Monotonisch" ein anderes Oberflächenmuster wählt, in der Abbildung etwa „Konzentrisch", wie es der Orca- und Prusa-Slicer anbieten (Bild 7).

Für das Foto des Testdrucks habe ich als Filamentfarbe Hellgrau gewählt, um die Formen hervorzuheben. Eine günstigere Farbe für die Frontplatte ist Weiß – unter anderem deswegen, weil darauf der Hintergrund von Beschriftungen (weißes Beschriftungsband mit schwarzem Text oder mit dem Laserdrucker auf weißes Papier gedruckte Potenziometer-Skalen) nicht so auffällt.

Aber eine Gerätefront kann auch optisch strukturiert und aufgepeppt werden, wenn man bewusst verschiedenfarbige Elemente kombiniert. So kann man etwa unterschiedliche Bereiche für Anzeige-, Bedienungs- oder Anschlusselemente beziehungsweise Funktionsbereiche farblich kenntlich machen. Ganz mutige Fans könnten sogar auf die Idee kommen, die farbigen Tetris-Elemente (Tetrominos, siehe Kurzlink und Bild 8) nutzen zu wollen, müssten dafür aber mein Elemente-Skript anpassen, weil dieses zurzeit nur die Tetrominos „I" (x=1, y=4) und „O" (x=2, y=2) unterstützt.

Strukturdefinitionen

Wenn im Raster bestimmte Streben oder Schraubenkörper nicht benötigt werden, dann wäre es schade, sie erst zu drucken und dann herauszuschneiden. Das kann man auch anders erledigen, nämlich schon in der SCAD-Datei. Dazu sind drei ausreichend große Vektoren definiert, je einer für die vertikalen Streben „vv", einer für die horizontalen Streben „vh" und einer für die Schraubenkörper „sm". Letzterer ließe sich auch durch eine logische Operation aus den anderen beiden Vektoren erzeugen; wer also Lust auf etwas boolesche Knobelei verspürt, darf sich gerne betätigen. Standardmäßig enthalten die Vektoren an allen Positionen eine „1"; das heißt, alle Streben und Schraubenkörper werden gedruckt. Trägt man jedoch eine „0" an einer Position ein, dann entfällt das jeweilige Objekt. Bei der Definition einer Frontplatte (Bild 9 A) muss man also zuerst festlegen, welche Elemente in welcher Größe und Ausrichtung wohin platziert werden sollen. Daraus ergibt sich, welche Schraubenkörper und welche Streben benötigt werden und welche nicht – letztere werden mit „0" eingetragen.

Um den zugehörigen Vektor „vv" für die vertikalen Streben zu erhalten, ordnet man den benötigten Streben eine „1" zu (Bild 9 B). In der Parameterdefinition liest sich das dann folgendermaßen:

```
...

vv02 = [1,1,1,0,1,...,1,1],
vv01 = [1,0,1,0,1,...,1,1],

...
```

Der Vektor „vh" für die horizontalen Streben ermittelt sich sinngemäß wie in Bild 9 C dargestellt zu:

```
...

vh02 = [1,1,1,1,1,...,1,1],
vh01 = [0,1,1,0,0,...,1,1],

...
```

Und der Vektor „sm" für die Schraubenkörper (Bild 9 D) sieht wie folgt aus:

```
...

sm02 = [1,1,1,1,1,...,1,1],
sm01 = [1,1,1,0,1,...,1,1]
```

Das Resultat sehen Sie in Bild 10.

Bild 11: In der Front unterzubringende Elemente

Beispiel Laborverstärker

Das Konzept des modularen Frontplattensystems lässt sich gut anhand eines konkreten Projekts veranschaulichen. Es soll ein Laborverstärker realisiert werden, der Spannungssignale von DC bis 100 kHz mit einstellbaren Offsets und Verstärkungen in einem weiten Bereich versehen kann, mit Amplituden- und Polaritätsanzeigen und allem Firlefanz – ein Laborgerät also. Die Basis dafür ist eine Schaltung aus „P. Horowitz, W. Hill: The Art of Electronics, Cambridge University Press, 3rd edition, 2015, Seite 274" – übrigens ein sehr empfehlenswertes Buch für alle, die sich für elektronische Komponenten und raffinierte Schaltungen jedweder

Art interessieren und der englischen Sprache halbwegs mächtig sind.

Im Gegensatz zu der überwiegenden Mehrzahl meiner sonstigen Elektronikprojekte habe ich diesmal mit der Frontplatte begonnen; die Platine wird dann bei JLCPCB gefertigt, selbst bestückt und abgeglichen. Wie man aus der Projektbeschreibung schon erahnen kann, erfordert so eine Schaltung einiges an Anschluss- und Bedienelementen, ist also wie gemacht für das Frontplattenkonzept – und nicht zuletzt war das auch der Anlass, dieses seit längerer Zeit vor sich hin schwelende Verstärkerprojekt endlich mal anzugehen.

Der erste Schritt ist die Festlegung aller unterzubringenden Komponenten (Bild 11) und deren Dimensionen, woraus sich die Größe der jeweiligen Rasterelemente bestimmt. Daraus resultieren dann nicht nur das einfachste 1×1-Element (z. B. für LEDs, BNC-Buchsen, Tastschalter), sondern auch größere Formate wie 1×2 für zwei Taster und ein Polklemmenpaar, 2×2 für Drehschalter und Potenziometer, 2×3 für den Kaltgerätestecker, 1×3 für den Netzschalter (okay, der hätte auch knapp in ein 1×2-Element gepasst und vertikal neben dem Kaltgerätestecker platziert werden können), und 3×3 für ein VU-Meter, das später noch mit einer Volt- und dBV-Skala versehen werden soll.

Hat man alles festgelegt, geht es an die Anordnung der Rasterelemente auf der Front, was sich mit einer einfachen Handskizze (Bild 12) oder noch besser mit aus Papier oder Karton ausgeschnittenen Elementen erledigen lässt. Zugegebenermaßen ist man durch das Raster nicht ganz so frei wie bei einer gebohrten Front, aber es gibt ja viele andere Vorteile dieses Konzepts, die das mehr als aufwiegen – einer davon ist, dass es eine halbwegs geordnete Struktur erzwingt.

In der Skizze der konkreten Elemente (Bild 13) habe ich auch gleich die Definitionen der Zwischenstege („vv“,„vh“) und Schraubenaufnahmen („sm“) in der Rasterplatte vorgenommen, wie im allgemeinen Teil beschrieben. Die zugehörigen Vektoren (gekürzt um nichtrelevante 1-er) sind folgende:

```
vv05 = [1,0,1,0,1,0,1,0,0,1,...],
vv04 = [1,0,1,0,1,0,1,0,0,1,...],
vv03 = [0,0,1,0,1,0,1,0,0,1,...],
vv02 = [0,0,1,0,1,0,1,1,1,1,...],
vv01 = [0,0,1,0,1,0,1,1,1,1,...],
...
vh04 = [1,0,0,0,0,0,0,0,0,0,...],
vh03 = [1,1,1,1,1,1,1,0,0,0,...],
vh02 = [1,1,1,0,0,1,1,1,1,1,...],
vh01 = [0,0,0,1,1,0,0,0,1,0,...],
...
sm04 = [1,0,1,0,1,0,1,0,0,1,...],
sm03 = [1,1,1,1,1,1,1,0,0,1,...],
sm02 = [1,1,1,0,1,1,1,1,1,1,...],
sm01 = [0,0,1,1,1,0,1,1,1,1,...]
```

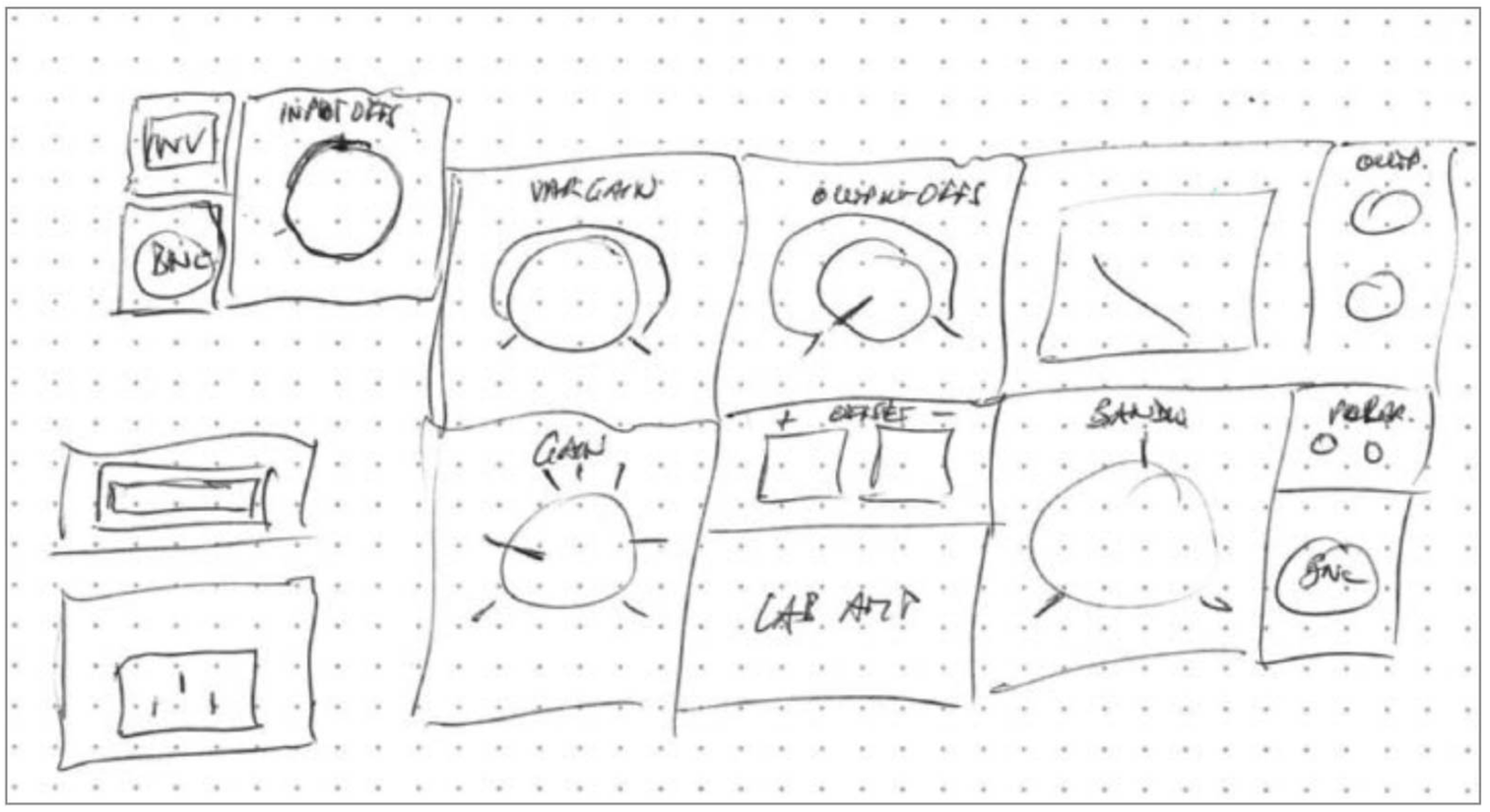


Bild 12: Scan meiner groben Handskizze zur Anordnung

Wegen der Frontplattengröße von 5×10 Elementen wurde die Rastertiefe mit `intr=6` etwas größer gewählt, damit die Front ausreichend stabil wird und sie sich beim Betätigen eines Elements in der Mitte nicht so leicht durchbiegt. Aus demselben Grund wurde die Elementdicke auf 3 mm erhöht. Damit habe ich erst mal Probedrucke (Bild 14) angefertigt, um zu sehen, ob alle Teile passen. Es kann sinnvoll sein, bei langen Seitenkanten noch Schrauben in den Rahmen zu setzen, damit sich dieser beim Festziehen der Elementschrauben nicht nach außen wegdrückt. Bei meinen 2-mm-Schrauben war das nicht der Fall, weil die Unterseiten der Schraubenköpfe ausreichend eben sind; bei Senkkopfschrauben würde das aber ganz sicher zu einem Problem werden, weswegen ich diese ausdrücklich nicht empfehle.

Einbauten

Der Einbau der Komponenten in die Frontelemente ist mit verschiedenen Mitteln zu realisieren. Der Netzschalter ist ein Snap-In-Typ, der nur in die passende Aussparung eingedrückt wird und dort einrastet. Der Kaltgerätestecker wird mit zwei M3-Schrauben fixiert. Die BNC-Buchsen sind innen verdeckt verschraubt und die Bohrung mit den Abflachungen verhindert ein Verdrehen, soweit völlig problemlos.

Drehschalter und Potis haben aber Außengewinde. Die Befestigungsmutter und das überstehende Gewinde außen sehen nicht ganz so ästhetisch aus, und man kann nun versuchen, dies mit einem klobigen Potiknopf irgendwie zu verdecken. Zusammen mit der Skala wird das Element dann aber recht groß und ich wollte zudem die Potiknöpfe – egal wie groß und für welchen Achsendurchmesser – in einem einheitlichen Look selbst drucken können: Unten liegende Hohlräume sind da eher ungünstig.

Daher habe ich mir eine unsichtbare Halterung hinter dem Frontelement überlegt. Sie besteht aus einer Platte, in der das Poti oder

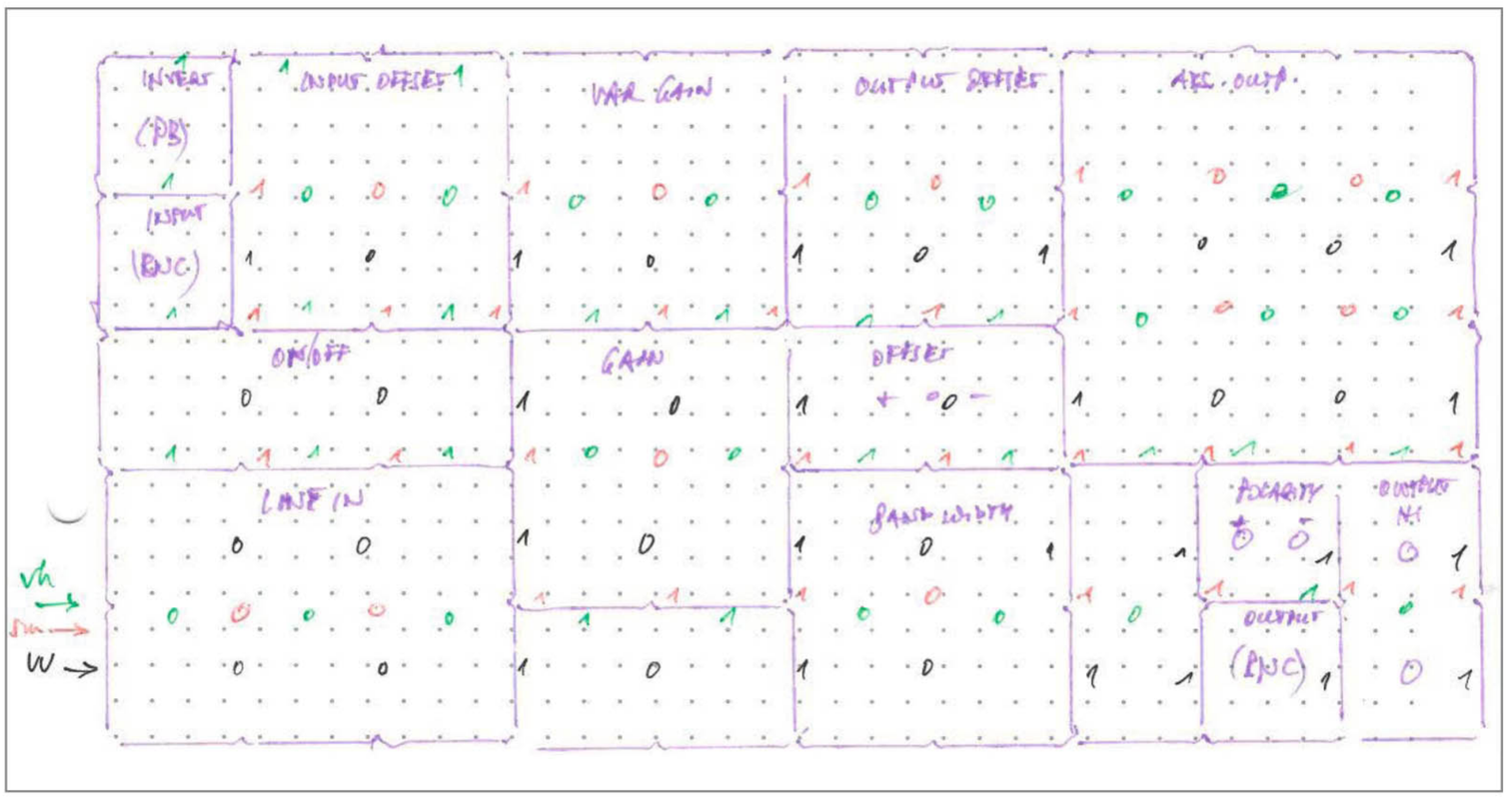


Bild 13: Verfeinerte Skizze mit der die Vektoren bestimmt werden.

Bild 14: Probe-Frontplatte mit teilweise eingesetzten Elementen

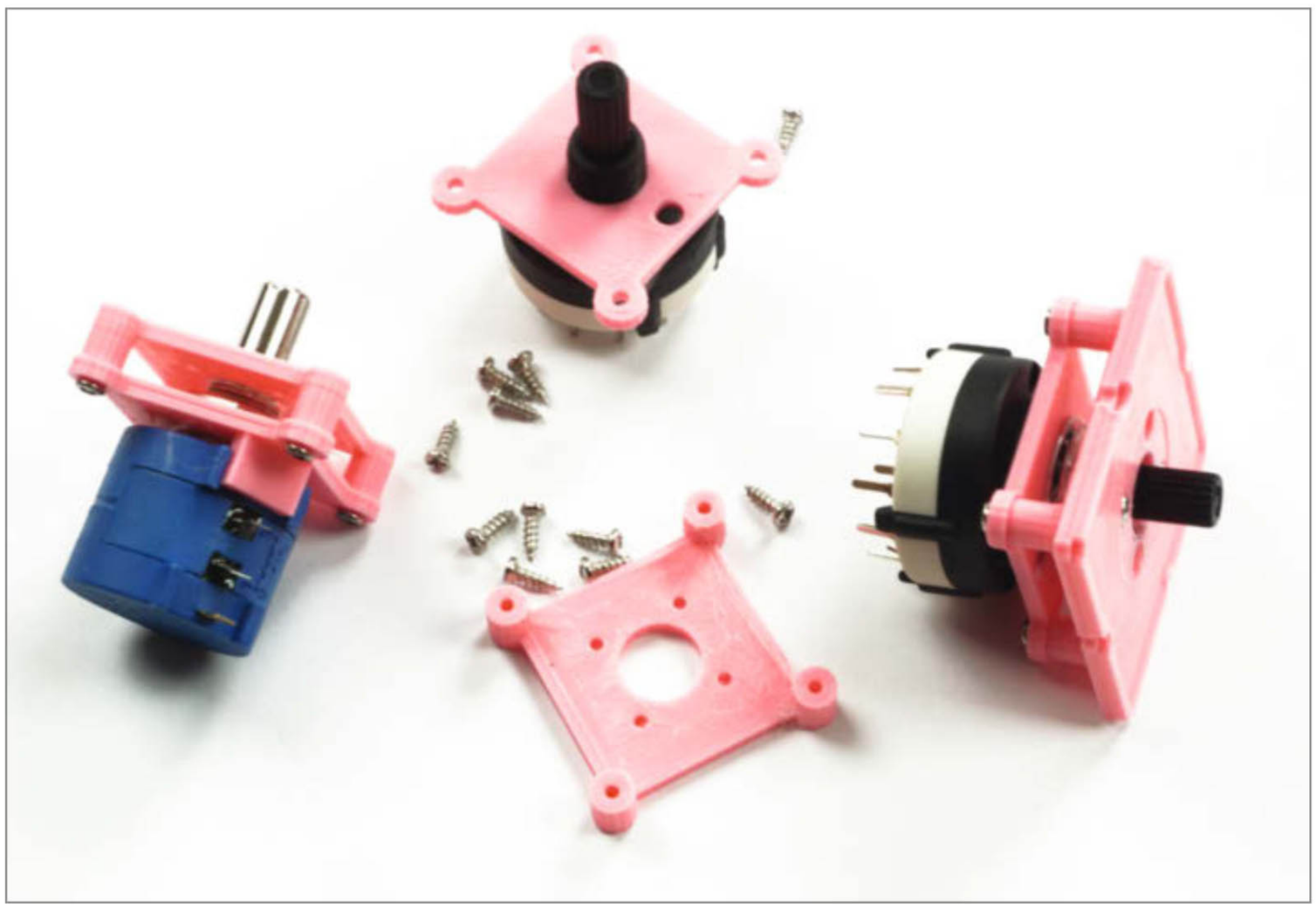


Bild 15: Komponenten für Drehschalter und Potis

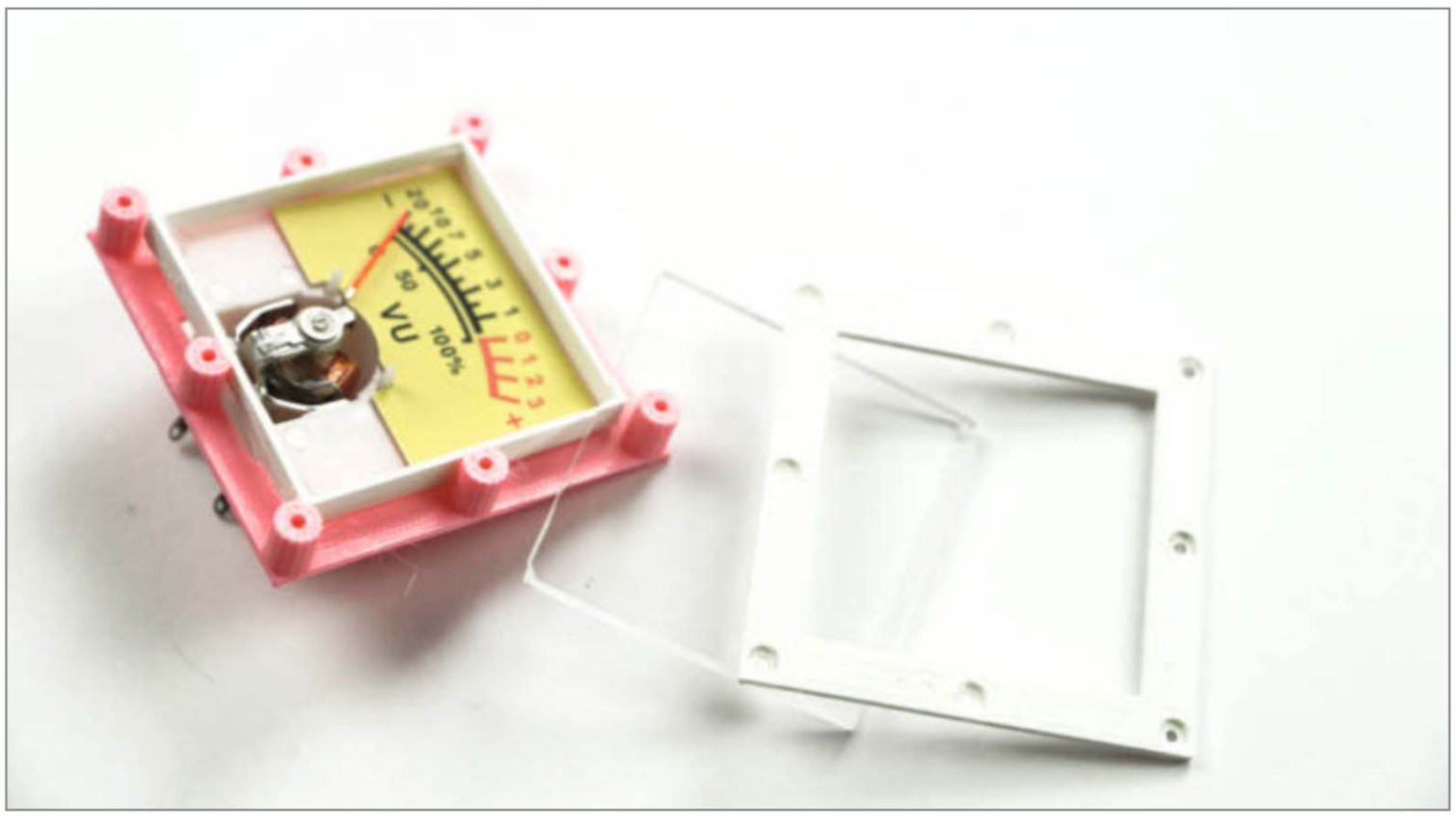


Bild 16: Einbaurahmen für VU-Meter

der Schalter mit der Mutter befestigt wird; eine Verdrehsicherung ist hier ebenfalls berücksichtigt. Um die Mutter herum ist genügend Platz zum Festziehen mittels Schraubenschlüssel. Die Optik ist dabei nebensächlich, denn diese Ebene liegt unsichtbar hinter der Front. Im Frontelement selbst wird eine Bohrung für die Achse benötigt. Die Platte (siehe auch Bild 15) wird mittels einer dazwischenliegenden Haltestruktur mit vier Stützen in der notwendigen Höhe an der Front befestigt, damit die Achse nicht zu weit herausteht. Die dafür erforderlichen Schrauben können sehr nah an der Potiachse liegen und deshalb von einem relativ kleinen und einfachen Potiknopf vollständig abgedeckt werden. Die OpenSCAD-Dateien für die Potiknöpfe heißen Potiknopfxxda,b.scad (wobei xx der Durchmesser des Knopfes ist und a,b der Durchmesser der Achse) oder Potiknopfxxda,b_NoInd.scad, wenn sie keinen Indikatorstrich haben.

Ähnlich ist auch das Anzeigeinstrument hinter bzw. in der Front befestigt und nicht davor. Im Frontelement befindet sich lediglich ein Ausschnitt, der das VU-Meter seitlich führt und einen schmalen Abdeckrahmen aufnimmt, der dafür sorgt, dass das Instrument nicht nach vorne herauskippt. Dahinter wird eine weitere Struktur mit dem Rahmen durch das Frontplattenelement hindurch verschraubt. So wird das Instrument auch nach hinten abgestützt. Die VU-Meter sind eine preiswerte Variante einer analogen Anzeige. Mich hat die abgestufte runde Erhebung auf der Scheibe gestört, deshalb habe ich stattdessen eine passend zugesägte Makrolonscheibe eingesetzt (Bild 16) und mit einem Rahmen auf den richtigen Abstand zur Mechanik des Instruments gebracht, dadurch ist die Skala jetzt viel leichter ablesbar. Und natürlich kann man auch Digitalanzeigen einbauen; entsprechende Beispiele für 3- und 4-stellige Panelmeter sind in den OpenSCAD-Skripten aufgeführt und für andere Größen leicht anpassbar.

Druckschalter beziehungsweise -taster können etwas problematisch sein, denn die sind oft für Printmontage vorgesehen, so wie der 8 × 8-mm-Tastschalter in Bild 17. Nachdem ich keine eigenen Platinen dafür anfertigen wollte, musste eine andere Lösung her. Die Tastenkappe habe ich passend zu einer Hülse gedruckt, welche in das Frontplattenelement eingesetzt wird. Der Taster wird dann von hinten eingeschoben. In meinem Fall waren nur ein paar Feilenstriche nötig, damit das passte: Der Taster ist leichtgängig und die Hülse steckt genauso fest im Frontplattenelement wie der Taster in der Hülse. Falls die Toleranzen nicht genau stimmen, kann man Hülse und Taster entweder einkleben bzw. mit einem 3D-Druck-Stift einschweißen (siehe „Mehr zum Thema“) oder mit Schrauben fixieren. Also auch hier eine (bis auf den Rahmen der Hülse außen) fast unsichtbare Befestigung.

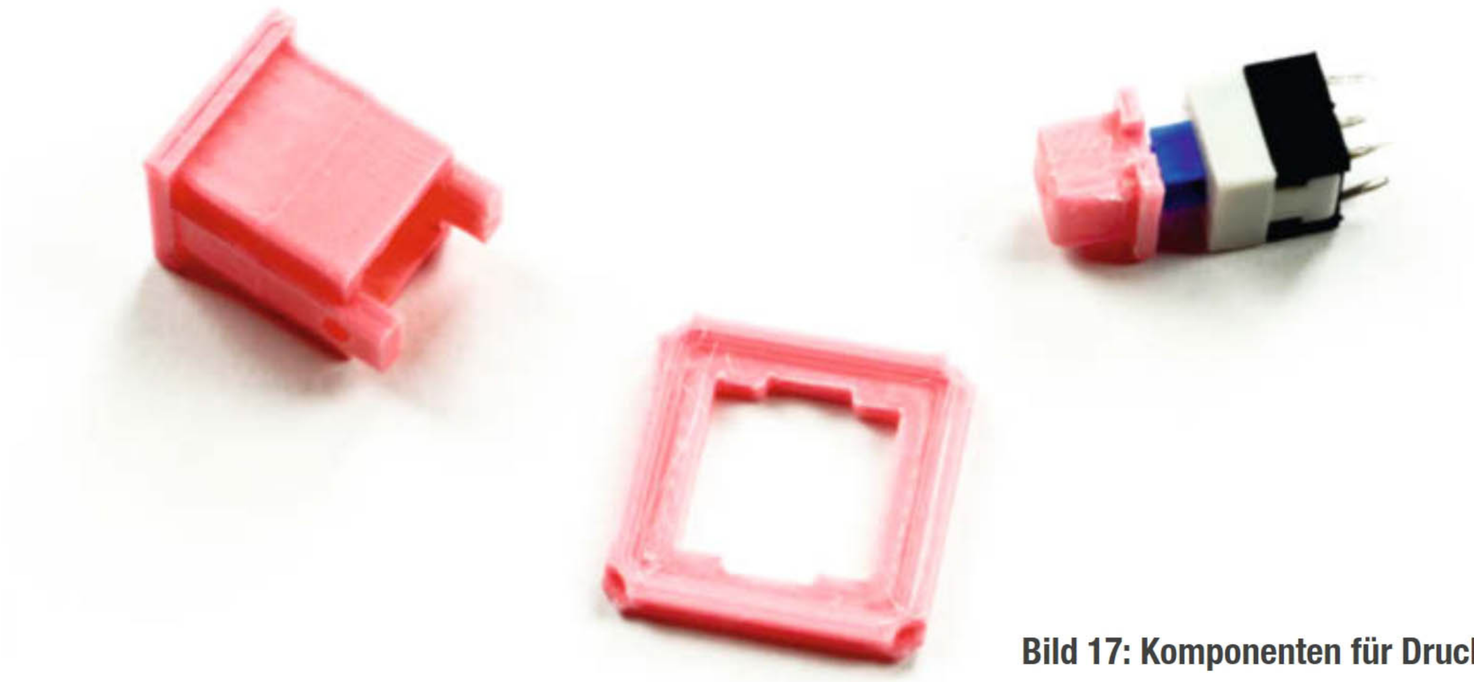


Bild 17: Komponenten für Drucktaster

Einzelne LEDs sind etwas schwierig einzubauen; sie lassen sich zwar von hinten in ein passendes Loch stecken, aber ebenso leicht wieder herausdrücken. Die sicherste Lösung dieses Problems besteht darin, mit einem 3D-Druck-Stift von hinten eine Stützstruktur mit gleichem Material um die LED herum- und an das Frontelement anzuschweißen. Muss die LED getauscht werden, kneift man die Struktur mit einem Seitenschneider samt LED weg oder druckt das Element neu.

Das Ergebnis

Die finale Front (Bild 2) wurde dann zwar in weißer Farbe gedruckt, man könnte sich aber auch ein helles Grau vorstellen. Die eingesetzten Elemente lassen sich farblich abstimmen, um eine Zusammengehörigkeit optisch darzustellen. Ich habe das hier mal mit dem Eingangsbereich (grau), Potis und Schaltern für Offset und Bandbreite (weiß), Verstärkung (gelb) sowie für die netzseitigen (schwarz) und ausgangsseitigen Komponenten (hellblau) so gemacht. Wenn man genügend viele Filamentfarben zur Verfügung hat, sind derlei kreativen Ideen kaum Grenzen gesetzt.

Auf der Innenseite der Front sind die Stege und die eingesetzten Elemente (Bild 18) mit den Bauteilanschlüssen der Komponenten gut zu sehen. Die Stabilität ist gut, da bewegt sich nichts beim Drücken und Drehen.

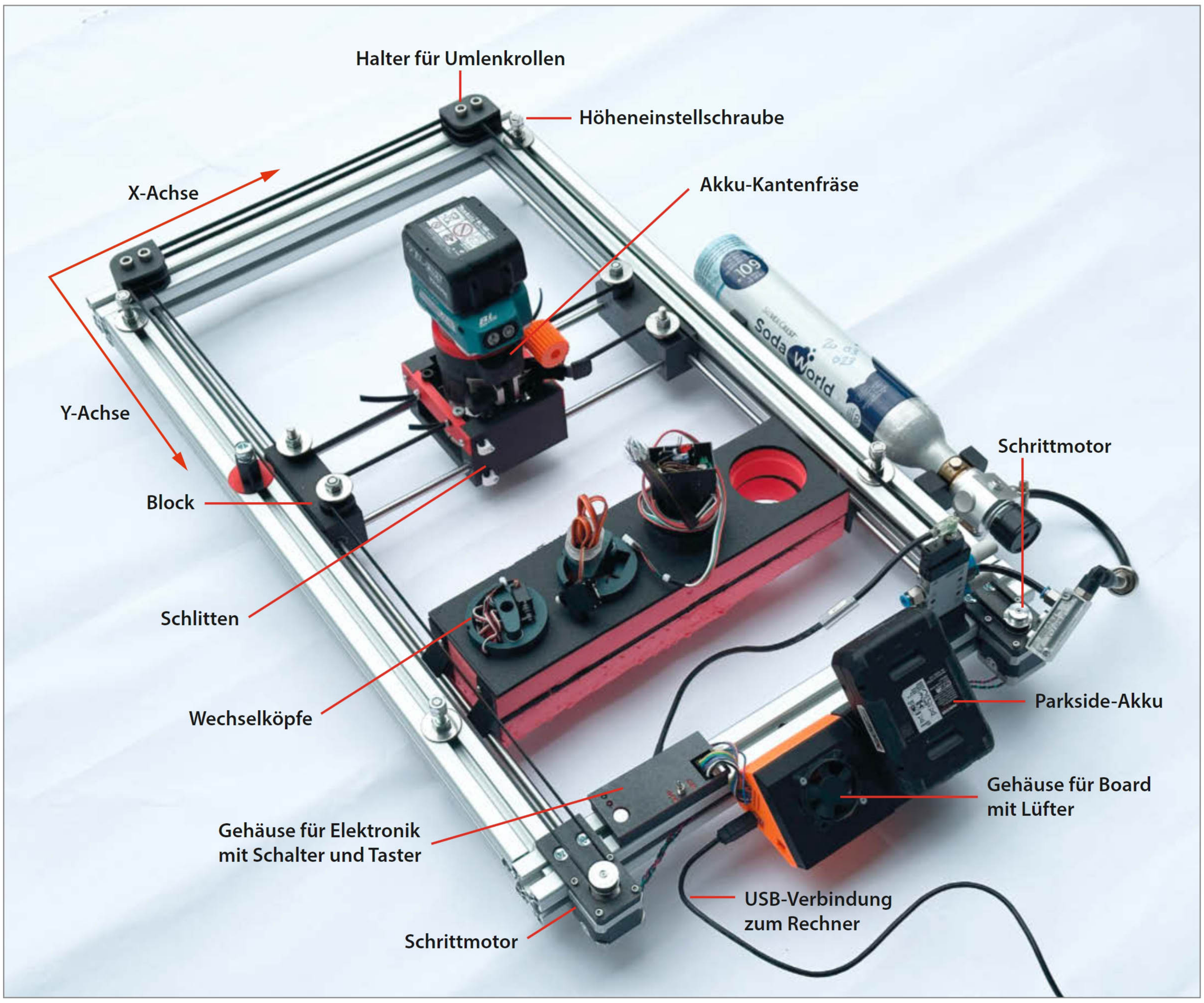
Wer einen Mehrfarbdrucker hat, der kann auch die Beschriftung und Skalen gleich mitdrucken. So etwas steht mir nicht zur Verfügung, deshalb werde ich die Beschriftung mit aufgeklebten Folien bzw. Beschriftungsstreifen ausführen oder schnell mit permanentem Folienstift auf die Elemente draufmalen; die kann ich ja jederzeit einzeln tauschen, wenn es mir nicht mehr gefällt. —*caw*

Bild 18: Innenansicht der fertigen Frontplatte

Mobi-C, die mobile CNC-Fräse

Dieser handliche X-Y-Positionierer für die Oberfräse ist schnell und einfach nachgebaut – dank fertig auf Länge bestellter Aluprofile und Verbindungsteilen aus dem 3D-Drucker. Das Ganze ist leicht zu transportieren, wird komplett mit Akkus betrieben und fräst auch in Werkstücke, die viel größer sind, als der Rahmen selbst.

von Bernd Heisterkamp



Diverse YouTube-Videos zeigen tolle Konstruktionen zur Herstellung von Frässchablonen, an denen man eine Oberfräse entlangführen kann. Die Ergebnisse sind schick, aber die Schablonen sind leider entweder sehr eingeschränkt in der Form, die man damit fräsen kann, oder sie geraten mechanisch sehr aufwendig. Daher dachte ich mir, dass eine wirklich flexible Führung für meine Akku-Kantenfräse doch lieber gleich eine mobile CNC-Steuerung sein sollte.

Hinzu kam, dass wir im Fablab diverse alte 3D-Drucker zerlegt haben und ich so an einen guten Vorrat von Schrittmotoren, Gleitlagern und Führungsstangen gelangen konnte. Unter Einsatz solcher wiederverwendeten Teile schont die selbst gebaute mobile CNC-Maschine daher Umwelt und Geldbeutel – man kommt dann günstiger weg als in der Kurzinfo angegeben.

Mein Projektziel war schnell definiert: Es sollte eine mobile, das heißt komplett akkubetriebene, solide CNC-Fräse werden, die nicht zu kompliziert aufgebaut ist und auch mal auf dem Motorroller transportiert werden kann. Zudem wollte ich meine Kantenfräse, eine Makita DRT50Z, weiterhin auch manuell nutzen können, sie sollte also leicht ein- und auszubauen sein. Im Verlauf des Projekts ist dann auch noch der Wunsch dazugekommen, die Fräse durch ein Schleppmesser ersetzen zu können, um Folien oder Pappe zu schneiden – und so fort.

Steckbrief: Mobi-C

Herausgekommen ist die Mobi-C. In der Grundversion ist das ein reiner X-Y-Positionierer ohne Z-Achse. Die in den Schlitten eingespannte Kantenfräse wird von Hand sowohl an- und abgeschaltet als auch in der Höhe verstellt. Man kann aber dennoch damit schrittweise in die Tiefe fräsen oder Traversen von einer Tasche zur anderen fahren. Dazu wird das CAM-Programm so vorbereitet, dass es zwischendrin stoppt und per LED anzeigt, was jetzt manuell zu tun ist. Anschließend drückt man auf einen Taster und das Programm läuft weiter. Wie ein komplettes Fräsprojekt mit der Mobi-C abläuft, beschreiben wir online (siehe Link in der Kurzinfo).

Weitere Merkmale: Die Mobi-C besteht aus einem offenen Rahmen – dadurch kann man sie direkt auf das Material legen, das man bearbeiten will, selbst wenn das viel größer ist als der Rahmen selbst. So lassen sich zum Beispiel auch Vertiefungen in fest eingebaute Holzteile fräsen. Unter Umständen reicht es sogar schon, den Rahmen beim Fräsen einfach mit den Händen auf dem Werkstück festzuhalten (Video siehe Kurzinfo), oder man zwingt ihn fest. Abenteuerlustige können durchaus mal probieren, was in eine Tür zu fräsen, ohne sie auszuhängen ...

Kurzinfo

- **CNC-Rahmen für die Oberfräse mit Arduino und CNC-Shield bauen**
- **Erweiterbar für Laser, Lötpasten-Dispenser und Schneidplotter-Messer**
- **Mobil dank Akkubetrieb**

Checkliste

Zeitaufwand:
etwa zwei Wochenenden (ohne Druckzeit)

Kosten:
bis 280 Euro (ohne Fräse)

3D-Druck:
Rahmenteile aus PLA drucken

Material

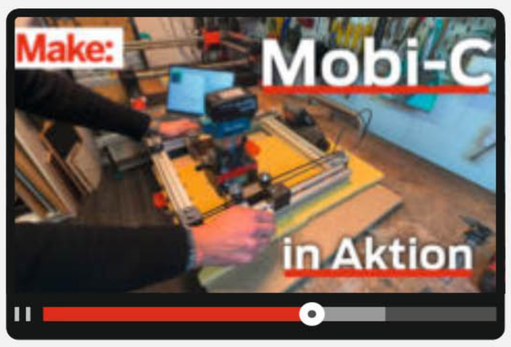
- **Komplette Materialliste** online über den Link

Werkzeug

- **Ständerbohrmaschine** und Satz Metallbohrer, speziell Durchmesser 7 und 8 mm
- **Muttergewindeschneider M8**
- **Lötstation** und Zubehör
- **Multimeter**
- **Schraubstock**
- **Fön** zum Erwärmen von Metall- und 3D-Druck-Teilen
- **Übliches Maker-Handwerkzeug**

Mehr zum Thema

- Ulrich Schmerold, Gewindeschneiden in Metal, Make Sonderheft 2021, S. 44
- Gratis-3D-CAD für Maker, Make 4/22, S. 76
- Peter König, Grafik für Maker, Make 2/21, S. 76
- Peter König, Grafik-Tricks für Maker, Make 3/21, S. 98
- Video: ESTLCAM-Tutorial: CNC-fräsen für BEGINNER (Teil 2) auf dem YouTube-Kanal von Make
- Video: Die Fräse in Aktion



Alles zum Artikel im Web unter
make-magazin.de/x24x

Mobil ist die Fräse auch deshalb, weil sie komplett durch Akkus gespeist wird: Die Kantenfräse bringt ihren eigenen Akku für ihre Spindel mit, die Schrittmotoren werden über einen handelsüblichen Parkside-Werkzeugakku am Rahmen der Mobi-C versorgt. Der Arduino schließlich bezieht seinen Strom über das USB-Kabel vom angeschlossenen Laptop aus. Der wird für den Betrieb ohnehin benötigt, um die vorbereiteten G-Codes zuzuspielen.

Auf der Grundlage der hier beschriebenen Grundversion kann man aber natürlich auch noch eigene Erweiterungen wie eine Z-Achse

Bild 1: So werden die Rahmenteile verbunden.

Bohrmaße

Buchstabe	Bedeutung	Durchmesser	Tiefe	Bemerkungen
G	Gleitstange	8 mm	20 mm	Bohrung bis zur Mitte des Profils; das bemerkt man, weil das Profil hohl ist.
R	Rahmen	8 mm	40 mm	Bohrung geht durch, daher gibt es auf der gegenüberliegenden Seite ebenfalls ein Loch.
J	Justierschrauben	7 mm	40 mm	Hier muss man anschließend noch ein M8-Gewinde hineinschneiden.

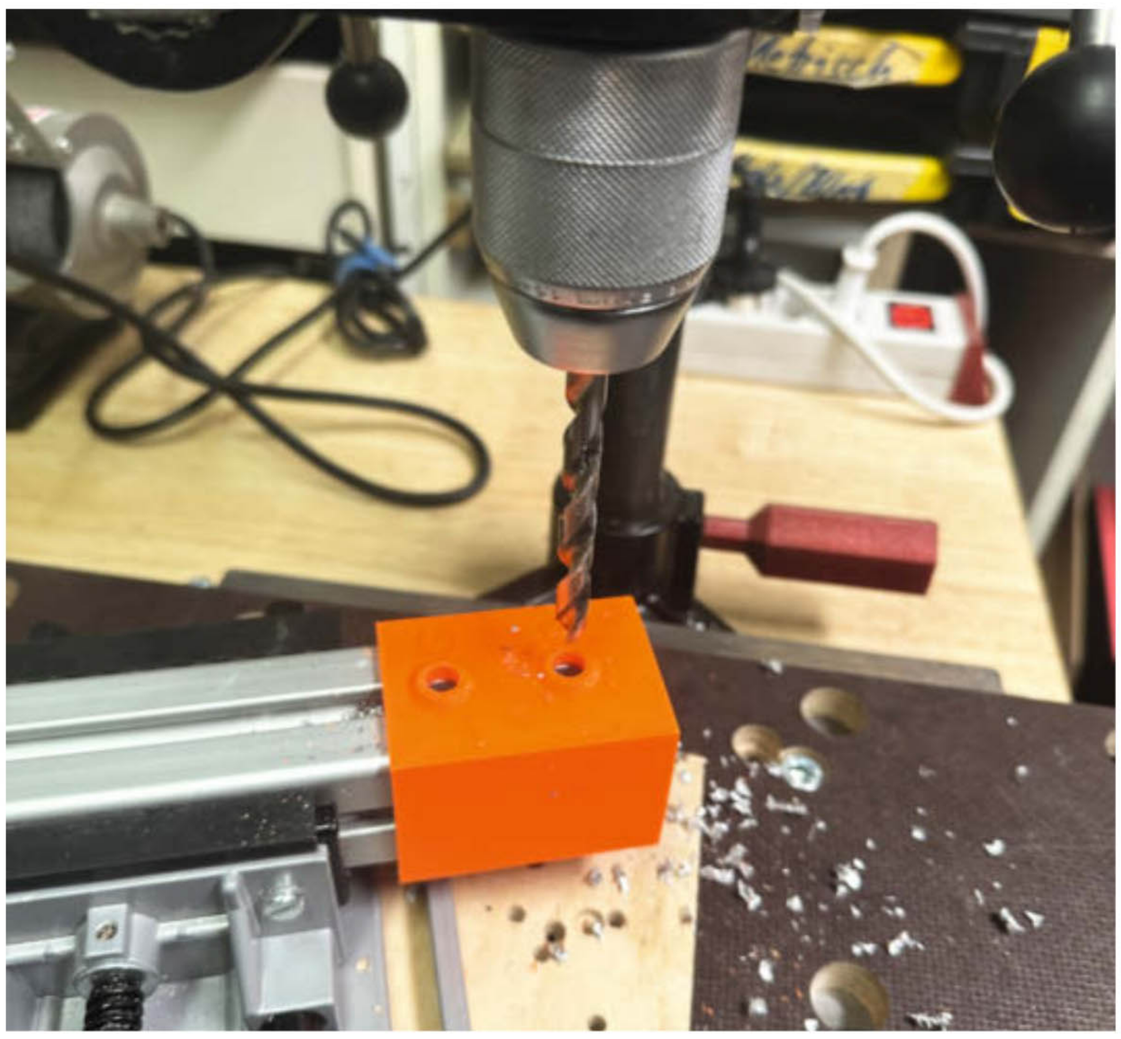


Bild 2: Bohren mithilfe der Schablone aus dem 3D-Drucker

Bild 3: Gedruckte Blöcke zur Aufnahme der X-Achse mit Gleitlagern, die entlang der Gleitstangen der Y-Achse laufen

3D-Druck-Teile

Als Material für alle 3D-Druck-Teile hat sich PLA bewährt, bei 0,2 mm Schichtdicke. Als Infill reichen 15 Prozent, außer bei den Motorhaltern, die brauchen 80. Im GitHub-Repository (siehe Link in der Kurzinfo) gibt es Screenshots, die zeigen, wie die Teile auf der Druckplatte angeordnet und orientiert werden sollten. Support ist nur an wenigen Stellen nötig: unter den Rundungen der Umlenkrollengehäuse sowie beim Parkland-Akkuadapter. Auch hierfür gibt es Screenshots im Github-Repo.

entwickeln und umsetzen; das verwendete CNC-Shield hat dafür noch Kapazitäten. Auf dem Titelbild des Artikels ist meine aktuelle Ausbaustufe zu sehen: Neben der Kantenfräse habe ich noch drei weitere Module zur Hand, die ich per Plug & Play wechseln kann: Einen Laser, einen Plotter-Einsatz für Schneidemesser oder Stifte sowie eine Spritze etwa für Kleber, Lötpaste oder Farbe. Die wird per Druckluft aus einer Soda-Stream-Flasche betrieben und nutzt ein ebenfalls über den Akku betriebenes Festo-Ventil.

Doch diese Erweiterungen sind Stoff für gegebenenfalls nachfolgende Artikel. Meine hier für den Nachbau beschriebene Mobi-C in der zweiten Grundversion hat einen Arbeitsbereich von etwa 40 × 21 cm (siehe Spalte „Beispiellänge“ in der Tabelle). Das erlaubt, etwa längere Schriftzüge ohne Versatz zu fräsen oder Regalbretter stirnseitig zu bearbeiten. In gewissen Grenzen lässt sich die Größe aber auch an eigene Bedürfnisse anpassen, siehe zur Berechnung und zu den empfohlenen Obergrenzen der Längen die Tabelle. Die Obergrenzen sind dabei kein fixes Limit, sondern eine Empfehlung, bis zu welchen Längen aufgrund der bisherigen Erfahrungen keine Probleme zu erwarten sind.

Beim Blick in die Tabelle fällt auf, dass die X-Achse relativ kurz ist (das hat konstruktive Gründe, dazu später mehr) und der Arbeitsbereich in dieser Richtung noch deutlich kürzer ausfällt. Das sind die Abstriche, die man machen muss, um eine leicht zu bauende und leicht zu transportierende Fräse zu bekommen. Dass sich die Mobi-C aber auch für Werkstücke nutzen lässt, die viel größer sind als sie selbst, macht das ein Stück weit wett:

Variable Bauteile und Arbeitsbereich

Beschreibung	Beispiellänge (mm)	Max. Länge (mm)	Berechnung
Aluprofil Richtung X-Achse	470	500	
Aluprofil Richtung Y-Achse[1]	520	600	
Gleitstange Richtung X-Achse	344	374	Profillänge X – 126 mm
Gleitstange Richtung Y-Achse	560	640	Profillänge Y + 40 mm
Zahnriemen (insgesamt)	4460	4900	4 × (Profillängen X + Y) + 500 mm
Arbeitsbereich X-Richtung	216	246	Profillänge X – 254 mm
Arbeitsbereich X-Richtung	402	482	Profillänge Y – 118 mm

[1] M8-Gewinde auf beiden Stirnseiten

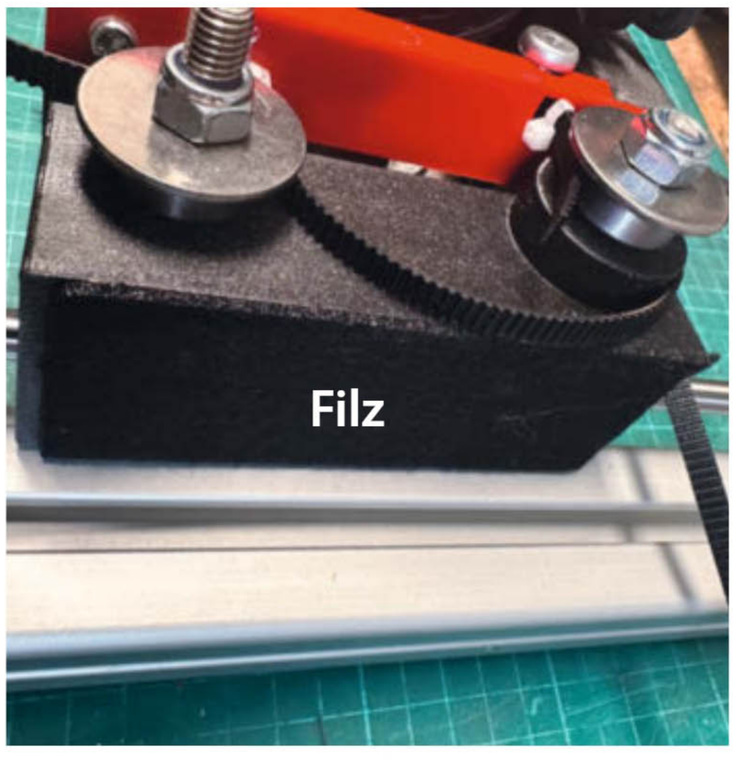


Bild 4: Filz an der Außenseite des X-Achsen-Blocks als Gleithilfe

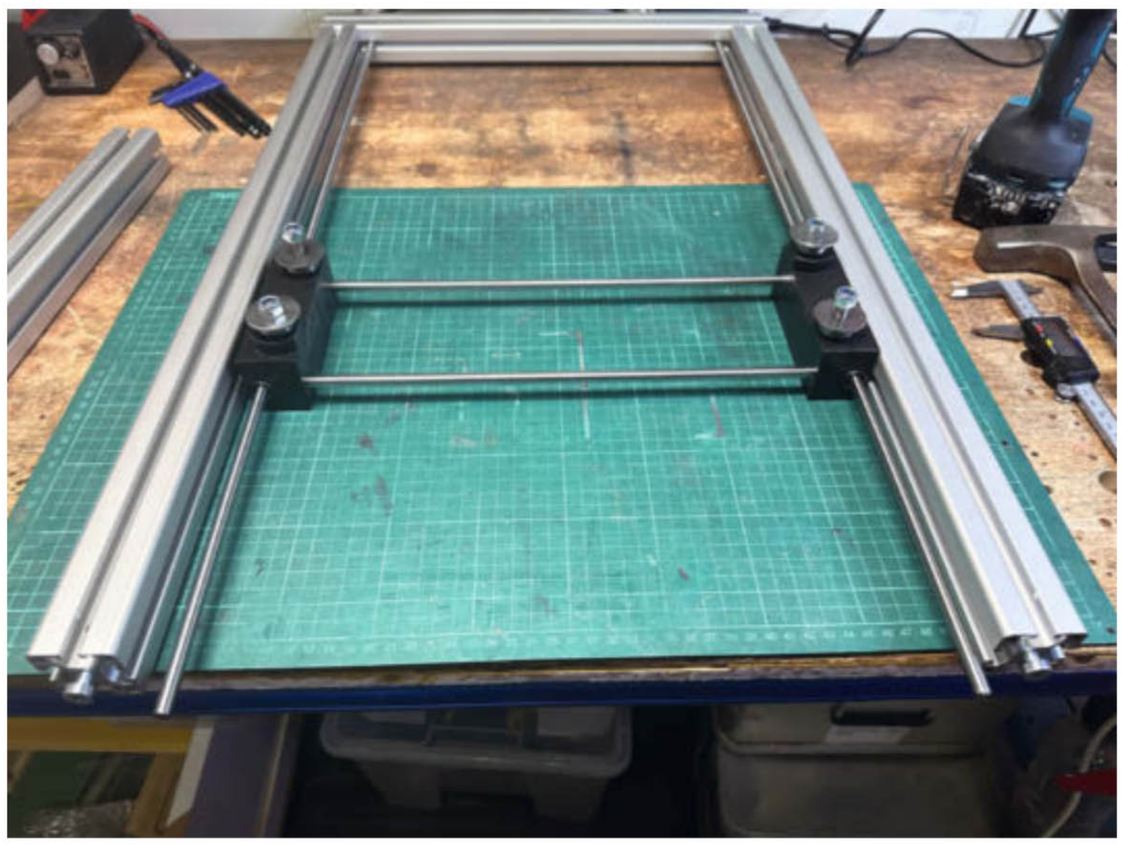


Bild 5: Probehalber zusammenstecken ist immer eine gute Idee, letztes Rahmenteil aber noch nicht verschrauben!

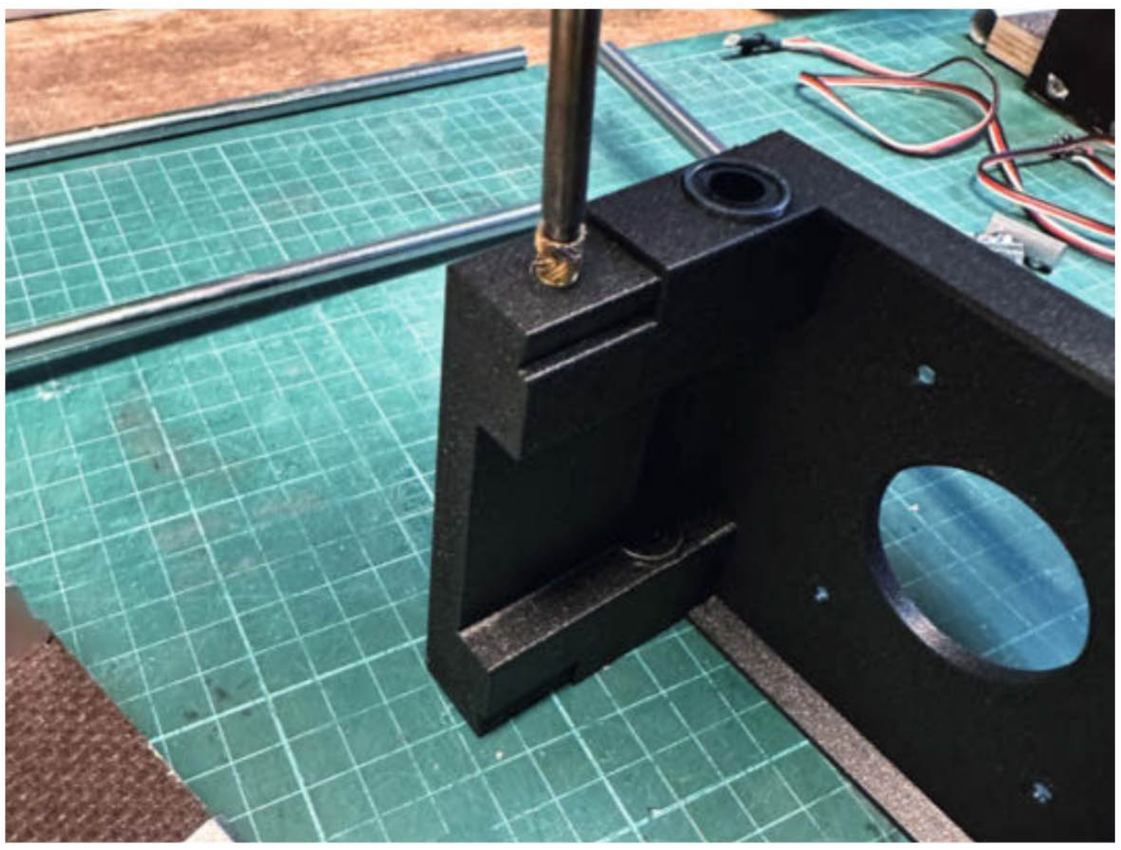


Bild 6: Einschmelzen der Gewindebuchsen in den auf der Seite liegenden Schlitten

Man kann die Fräsarbeit in kleine Portionen aufteilen und die Fräse zwischendrin neu platzieren.

Die Größen der 3D-Druck-Teile verändert sich durch Änderungen am Arbeitsbereich nicht. Die STL-Dateien gibt es über den Link in der Kurzinfo zum Download aus dem Github-Repository zum Projekt, dort gibt es auch genauere Hinweise zu den Druckparametern der einzelnen Teile.

Der Rahmen

Das Titelbild dieses Artikels zeigt den prinzipiellen Aufbau der Mobi-C: Den Kern bildet ein stabiler Rahmen aus vier Aluminiumprofilen mit einem Querschnitt von 40×40 mm, der die Arbeitsfläche umschließt. Die Aluprofile des Rahmens haben jeweils auf einer Stirnseite ein 8-mm-Gewinde. Diese Profile kann man fertig auf Länge zugeschnitten samt Gewindebohrung bestellen, das vereinfacht einen präzisen Aufbau erheblich (Bezugsquellen siehe Link). Damit man mit dem Inbusschlüssel das Querprofil mithilfe einer flachköpfigen Innensechskantschraube festschrauben kann (Bild 1), muss in dieses Profil auf der gegenüberliegenden Seite eine Bohrung angebracht werden, aber hier kommt es dann nicht auf den halben Millimeter an.

Weitere Bohrungen werden für den Einsatz der 8-mm-Gleitstangen der Y-Achse benötigt. Um alle Bohrungen möglichst präzise hinzubekommen, habe ich eine Bohrschablone erstellt, die man in 3D druckt und die dann einfach über das Profilende der Rahmenteile in X-Richtung geschoben werden kann (Bild 2). Details zu den Bohrungen führt die Tabelle auf.

Über die Gleitstangen der Y-Achse laufen auf jeder Seite zwei Gleitlager, die in die horizontalen Bohrungen eines ebenfalls 3D-gedruckten Blocks eingepasst werden (Bild 3). Diese Blöcke hatte ich bei meinem ersten Prototypen noch aus je zwei Holzbrettchen hergestellt, aber die 3D-Druck-Teile der aktuellen Version sind präziser, platzsparender und leichter herzustellen. Von unten werden in den Block zwei Schrauben M8×60 geführt, die als Achsen für die Riemen-Umlenkrollen (Kugellager) dienen. Von oben fädelt man dann je ein Kugellager und dann je eine kleine und eine große Unterlegscheibe darauf, gekrönt von einer selbstsichernden Mutter. Unter das Kugellager kommt keine Scheibe, da im 3D-Druck-Teil rund um das Loch kleine Wülste vorgesehen sind, damit das Kugellager auch ohne Scheiben frei läuft.

Die beiden Gleitstangen der X-Achse werden einfach in die Bohrungen an der Seite des Blocks gesteckt, eine Verschraubung ist nicht notwendig.

Die jeweilige Außenseite der Blöcke liegt am Alurahmen an, dadurch hat sie auch bei längeren Y-Achsen nur wenig Spiel. Damit sie gut am Rahmen gleitet, habe ich ein Stück Filz auf die Außenseite jedes Blocks geklebt (Bild 4). Diese Stabilisierung hat man bei der X-Achse nicht, daher sollten die Gleitstangen in X-Richtung auch unter einer Länge von 40 cm bleiben, ansonsten hat man zu viel Spiel in Y-Richtung (Bild 5).

Der Schlitten

Weiter geht es mit dem Schlitten für die Kantenfräse. Nach dessen 3D-Druck müssen die Gleitlager in die horizontalen Bohrungen dafür eingepasst werden. Ich habe dazu das Druckteil mit vorgesetzten Gleitlagern im Schraubstock eingespannt und diese dann vorsichtig eingepresst. Ist die Aussparung zu klein, hilft es, die Gleitlager mit einem Föhn zu erwärmen.

Sind alle vier Lager eingepasst, testet man den leichten Lauf der Gleitstangen durch beide Seiten. Bei mir musste ich den Schlitten

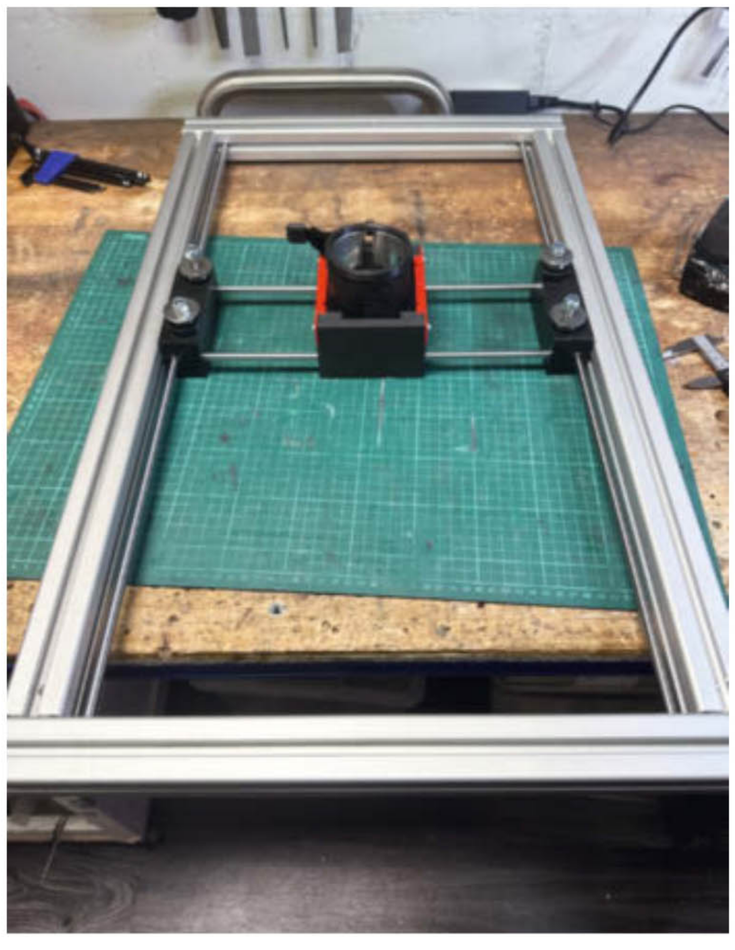


Bild 7: Bitte einmal in X- und Y-Richtung probe-rutschen!

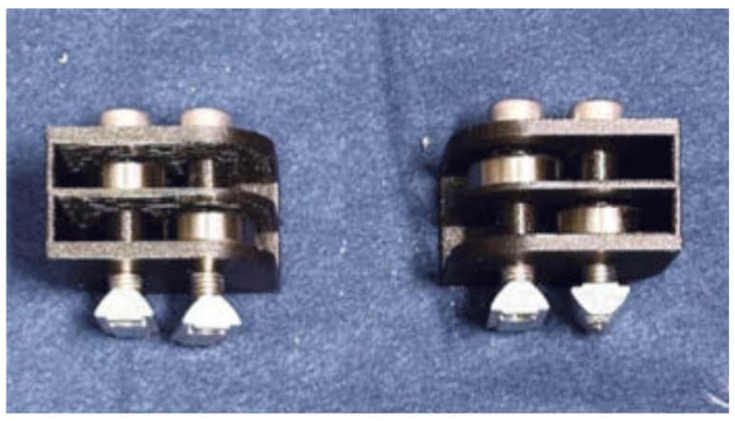


Bild 8: Die Halter der Umlenkrollen

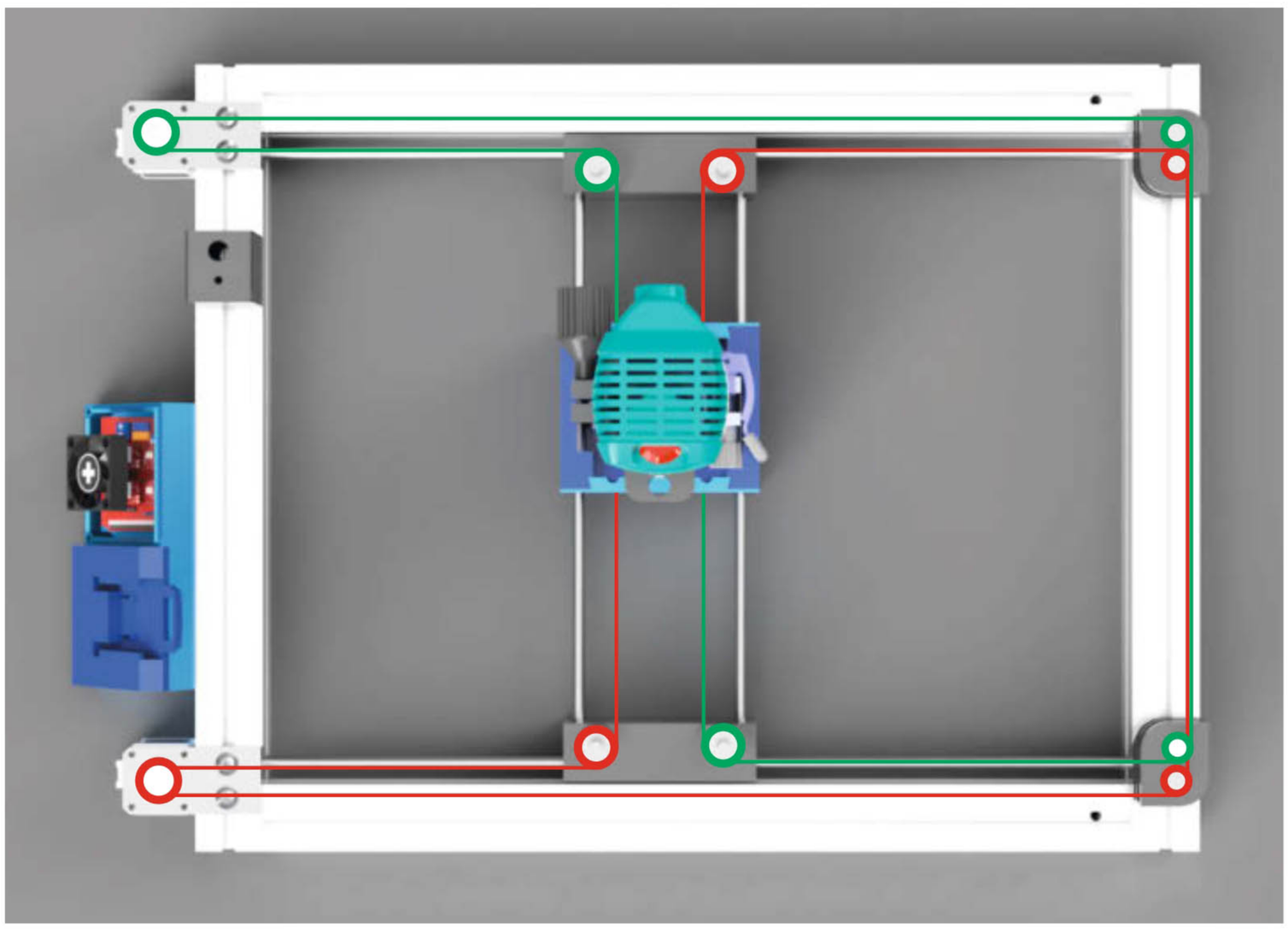


Bild 9: Schema der Zahnriemenführung

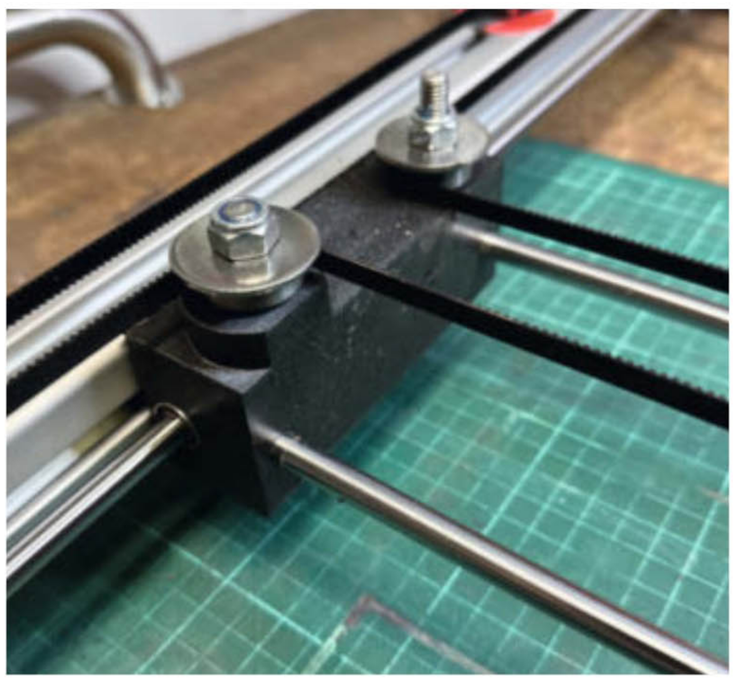


Bild 10: Die Zahnriemen laufen in unterschiedlichen Ebenen.

auf der Unterseite etwas erwärmen und ihn dann mehrmals auf den Gleitstangen hin- und herbewegen, um die Reibung zu minimieren. Zum Schluss werden noch die Gewindebuchsen mit dem Lötkolben eingeschmolzen (Bild 6). So lassen sich die Befestigungsplatten (auf Bild 7 gut erkennbar, weil in rot gedruckt) für die Zahnriemen leicht an- und abbauen.

Nach diesem Arbeitsschritt sollte man unbedingt den Rahmen vorläufig zusammenbauen und testen, ob der Schlitten in X- und Y-Richtung leicht bewegt werden kann (Bild 7). Dass sich die X-Achse dabei etwas verdrehen kann, ist zu diesem Zeitpunkt normal – wenn später erst die Zahnriemen gespannt sind, ist die Konstruktion sehr stabil.

Motoren und Riemen

Für den Antrieb des Schlittens werden auf der einen Seite des Rahmens die Motorenhalter und auf der anderen Seite die Umlenkrollen angebracht. In der aktuellen Version 2 meiner Mobi-C, die hier beschrieben wird, habe ich die Halter für die Umlenkrollen und die Motoren aus PLA gedruckt. Durch die Dicke von 8 mm und 80 Prozent Infill im Bereich des Übergangs zwischen Motor und Rahmen sind diese stabil genug. Die Achsen der Umlenkrollen sind wieder M8-Schrauben, die in die Profile eingelegte Nutensteine geschraubt werden. Die gedruckten Gehäuse dienen dazu, dass die Zahnriemen nicht verrutschen. Au-

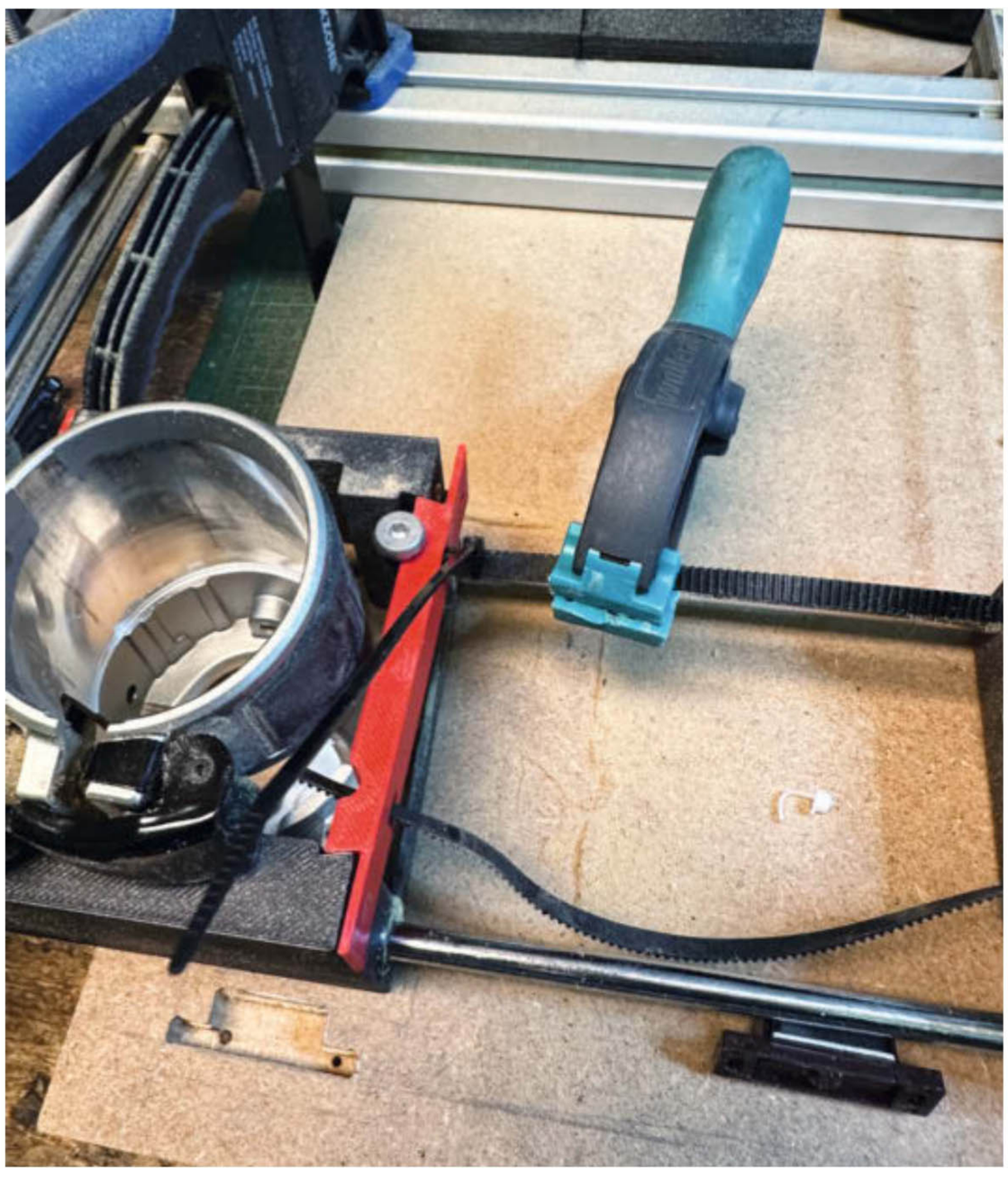


Bild 11: Beim Dosieren der Spannung für den Zahnriemen hilft es, an der senkrechten Schraube am Schlitten zu drehen (links vom roten 3D-Druck-Teil).

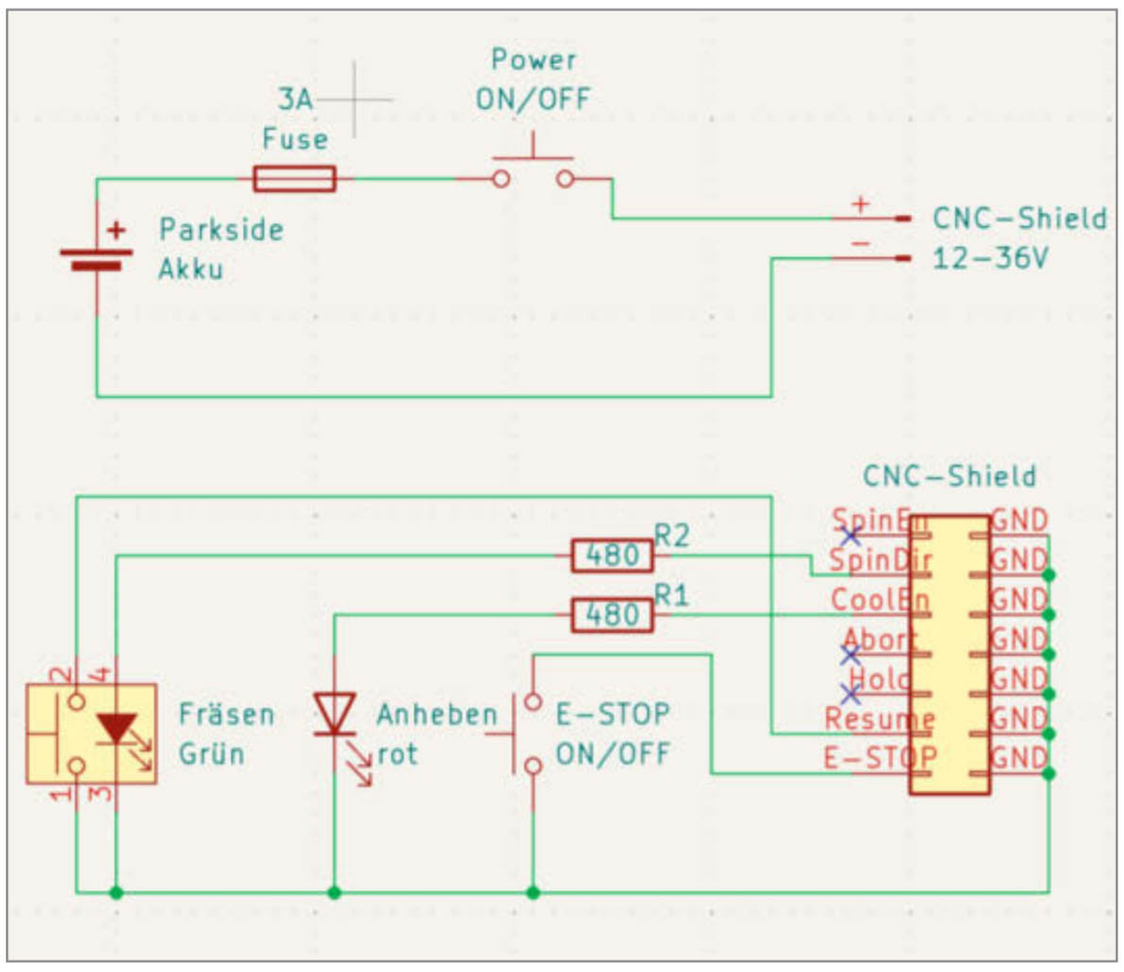


Bild 12: Der Schaltplan

ßerdem sind sie so gestaltet, dass es keine weiteren Unterlegscheiben bedarf (Bild 8).

Bild 9 zeigt die Führung der Zahnriemen. Diese müssen in horizontaler und vertikaler Richtung jeweils genau parallel laufen, damit sich die Gesamtlänge des Zahnriemens bei der Verschiebung des Schlittens nicht ändert. So wird auch klar, weshalb die Umlenkrollen mit zwei leicht versetzten Achsen auf dem Rahmen montiert sind.

Der im Schema rot eingezeichnete Zahnriemen liegt eine Ebene tiefer als der grüne, daher berühren sich die Zahnriemen auf der Stirnseite des Rahmens nicht und laufen auch auf den X-Achsen-Blöcken auf unterschiedlichen Höhen (Bild 10). Wer sich für eine detailliertere Beschreibung des sogenannten CoreXY-Zahnriemenantriebs interessiert, findet bei den Links zum Artikel weitere Informationen.

Sind die Umlenkrollen montiert, sind die Motoren an der Reihe. Auch für sie gibt es 3D-gedruckte Befestigungsteile. Auf die Motorachsen schraubt man je ein GT2-Antriebsritzel. Achtung: Hierbei liegt bei einem Motor der Zahnkranz des Ritzels oben, beim anderen unten, da die beiden Zahnriemen in unterschiedlichen Ebenen laufen!

Anschließend können die Zahnriemen eingefädelt und gespannt werden. Die Spannung erfolgt, indem man die Zahnriemen wie in Bild 11 gezeigt durch die Löcher der Befestigungsplatten am Schlitten führt und auf der Gegenseite ein ebenfalls 3D-gedrucktes Spannwerkzeug durch die Öse steckt. Dieses trägt den Kopf einer Inbusschraube, hat am Schaft aber statt eines Gewindes senkrechte Nuten, in die die Zähne des Riemens passen. Das Foto entstand allerdings, bevor ich auf dieses Helferlein kam und stattdessen noch eine normale Schraube zum Spannen benutzte – geht auch.

Jetzt mit einer Flachzange das Ende des Zahnriemens ziehen und zugleich die Schrauben oder Spannwerkzeuge in den Ösen etwas drehen, dadurch kann die Spannung sehr fein justiert werden. Temporär dann den Zahnriemen mit einer Federklemme fixieren und anschließend einen Kabelbinder für die permanente Fixierung anbringen.

Hat man kontrolliert, dass sich der Schlitten frei bewegen lässt, die Zahnriemen eine gute Spannung haben – das heißt: dass sich die Schrittmotoren bei Verschiebung des Schlittens mit der Hand sauber drehen, ohne dass der Zahnriemen über das Ritzel springt –, ist der mechanische Aufbau fast fertig. Es fehlen nur noch die Gewindestangen, die es erlauben, den Rahmen in der Höhe zu justieren, doch dazu später mehr.

Elektronik

Zur Steuerung von Mobi-C verwende ich einen Arduino Uno mit CNC-Shield. Die beiden Schrittmotoren werden mit A4988-Treiberbausteinen angesteuert, die direkt auf das CNC-Shield gesteckt werden. Der ziemlich übersichtliche Schaltplan ist in Bild 12 zu sehen. Während der Arduino Uno über die USB-Schnittstelle per Kabel vom angeschlossenen Computer mit Strom versorgt wird, erhält das CNC-Shield seine Spannung direkt von einem Parkside-Werkzeugakku (Typbezeichung X 20 V Team, 20 V, 44 Ah, gibt es bei Lidl). Da Kurzschlüsse an Akkus fatale Folgen haben können, wird in die Zuleitung noch ein Schalter und eine Schmelzsicherung gebaut. Generell sollte man aber beim Aufbau der Schaltung erst einmal mit einem Labornetzteil arbeiten und die Stromaufnahme kontrollieren, noch bevor man den Akku anschließt.

Ein Lüfter ist für das CNC-Shield dringend notwendig, da die Schrittmotortreiber bei einer Überhitzung den Dienst verweigern (im Schaltplan ist er nicht eingezeichnet). Ich habe einen 12-V-PC-Lüfter verwendet. Dessen Betriebsspannung ist für den Akku zwar etwas zu klein, aber für den Kurzzeitbetrieb funktioniert das meiner Erfahrung nach. Besser ist natürlich der Einsatz eines Lüfters mit 18 bis 20 V Betriebsspannung.

Das CNC-Shield hat digitale Ein- und Ausgänge für die Spindel- und Programmsteuerung (Bild 13). Den Ausgang für die Spindelrichtungssteuerung nutze ich für etwas anderes als eigentlich gedacht, nämlich für die Ansteuerung einer grünen LED, die direkt in einen Taster integriert ist. Sie zeigt später an, dass die Fräse heruntergelassen werden muss, um den Fräsvorgang fortzusetzen. Der Kühlflüssigkeitsausgang (den wir hier auch nicht brauchen) steuert eine rote LED, die anzeigt, dass die Fräse hochgestellt werden muss, damit die CNC eine Traverse fahren kann, ohne Material wegzunehmen. Der Taster mit der grünen LED darin wird an den Eingang „Resume“ angeschlossen, dies erlaubt später das Fortsetzen des unterbrochenen Programms. Zudem gibt es noch

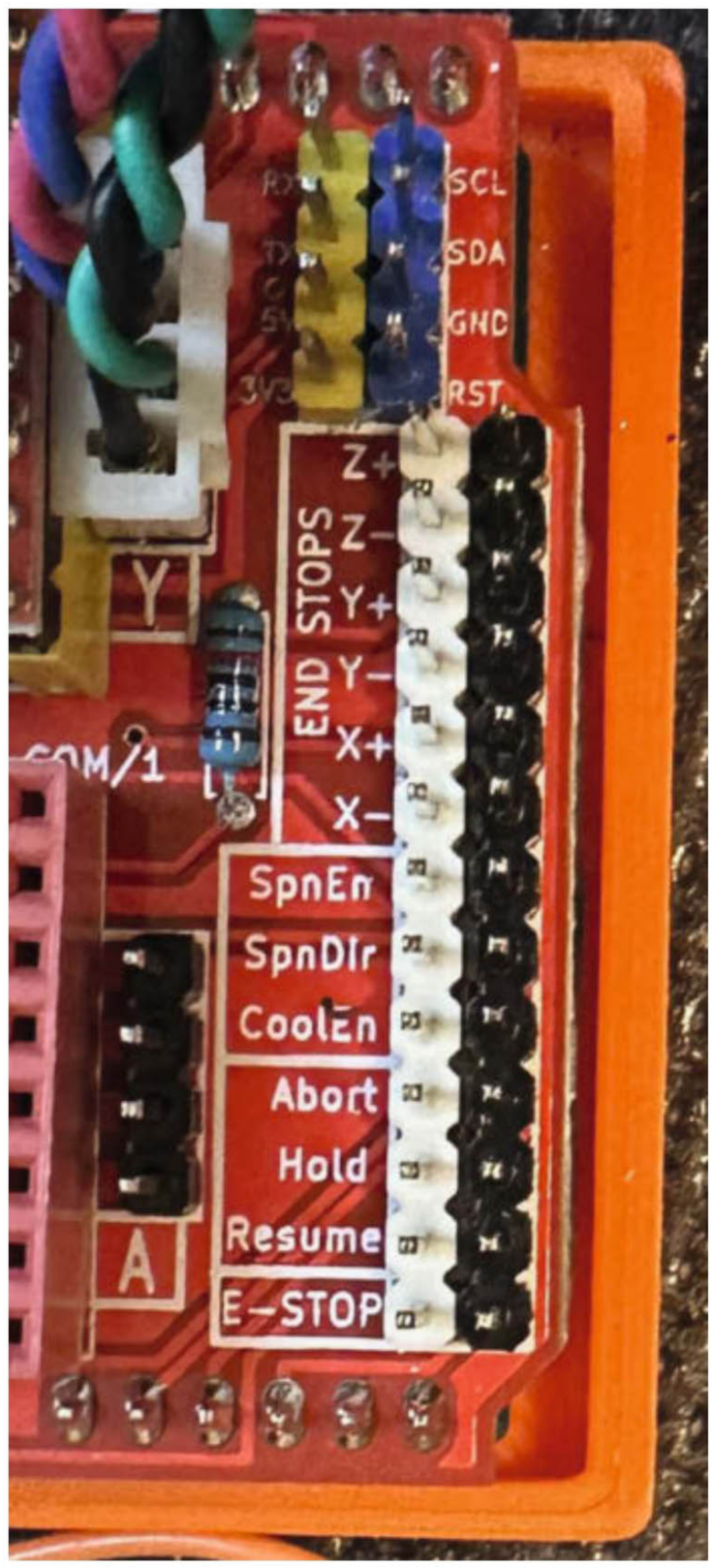


Bild 13: Die Ein- und Ausgänge des CNC-Shield

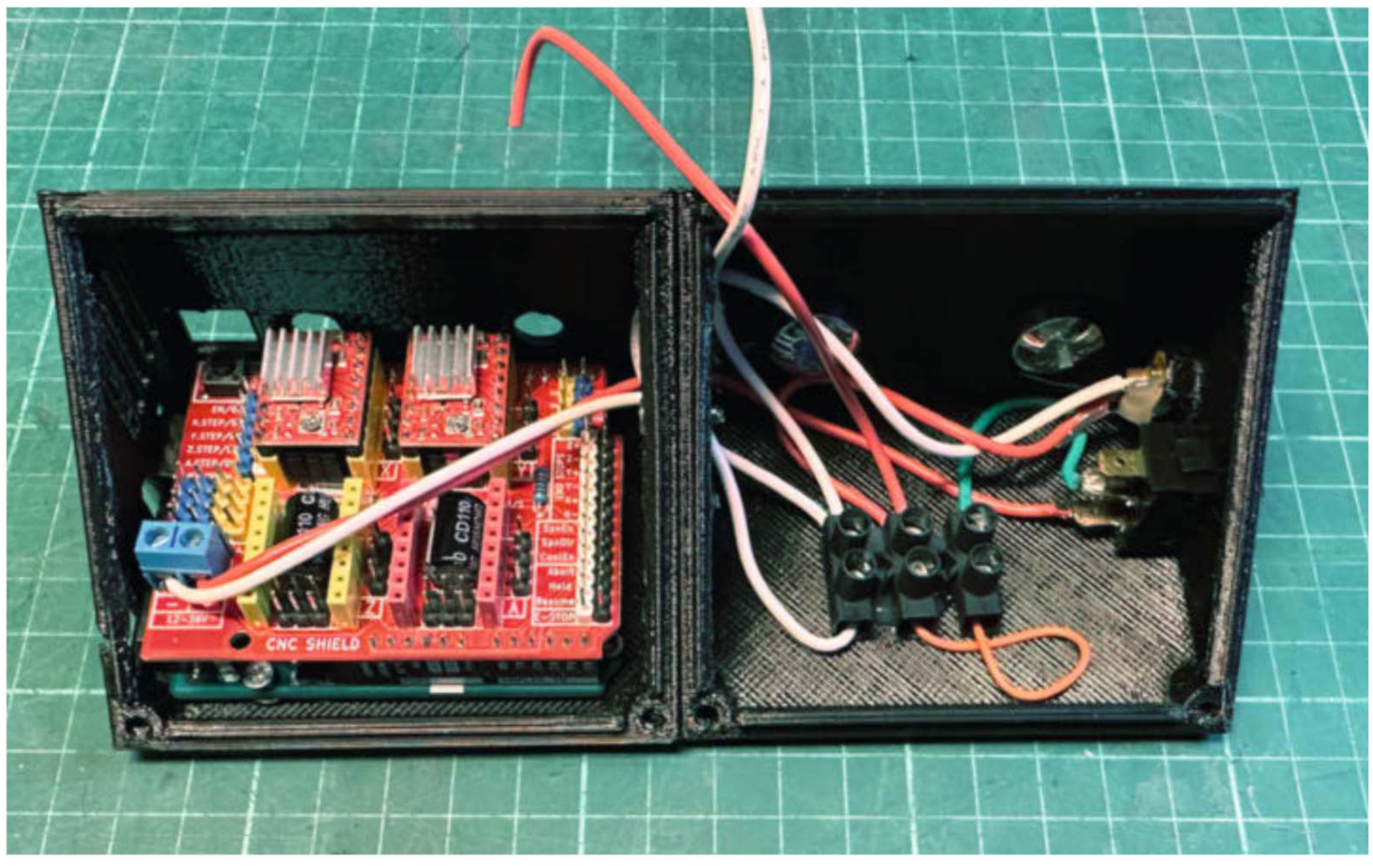


Bild 14: Links das Gehäuse für Arduino und CNC-Shield, rechts das für den Akkuanschluss

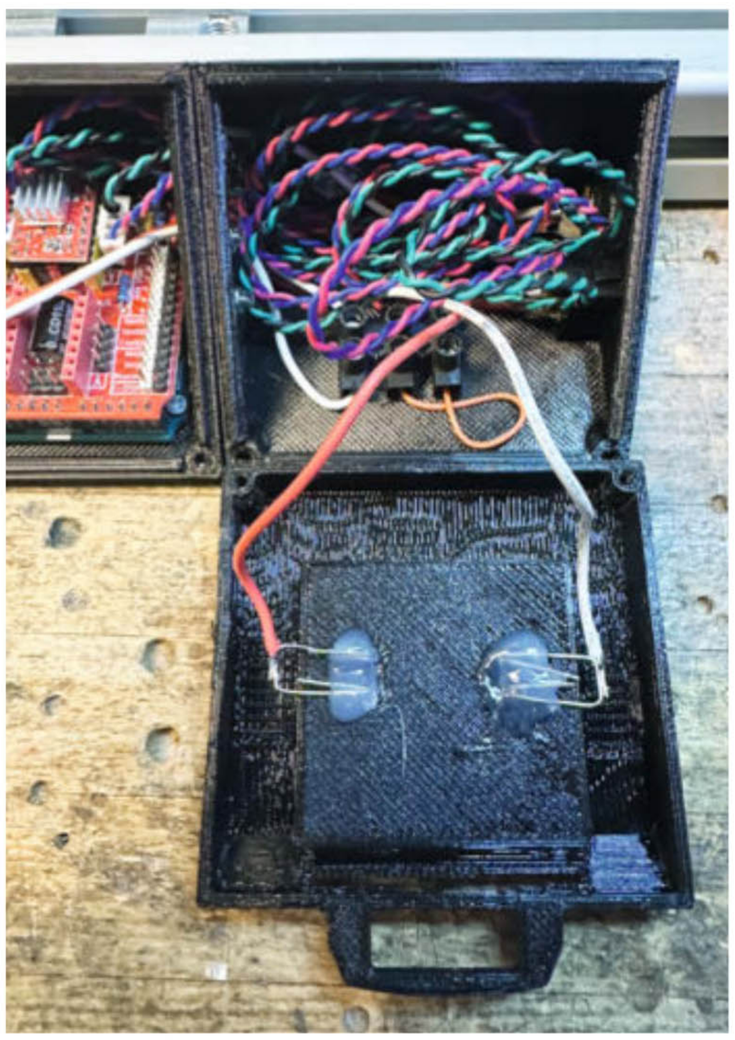


Bild 15: Büroklammern als Akkukontakte

einen Schalter, der an den E-STOP-Eingang (Emergency Stop) angeschlossen wird: Er erlaubt die Deaktivierung der CNC-Steuerung und ermöglicht somit, den Schlitten auch bei eingeschalteter Stromversorgung von Hand zu verschieben.

Für das Gehäuse der Elektronik und des Akkuadapters habe ich eine Vorlage von Printables.com für meine Anforderung modifiziert (siehe Link). Da das Gehäuse für den Arduino bereits etwas klein ist, habe ich nur den Akkuadapter an den Alurahmen geschraubt und die beiden Gehäuse mit zwei M4-Schrauben untereinander verbunden (Bild 14), die Löcher dafür musste ich allerdings noch in die 3D-Druck-Teile bohren. Als Kontaktfedern für den Parkside-Akku dienen zwei Büroklammern, die mit Heißkleber im Deckel des Akkuadapter-Gehäuses festgeklebt werden (Bild 15).

Schalter, Taster und LEDs sitzen in einem dritten Gehäuse, einer einfachen Box, die über eine Bohrung im Boden mit einem Nutenstein im Aluminiumprofil befestigt ist.

Einstellung der Schrittmotortreiber

Ist die Elektronik fertig, muss noch die Stromregelung der Schrittmotortreiber eingestellt werden. Dazu müssen 20 V am CNC-Shield anliegen, auf dem Arduino muss noch kein Programm laufen. Wer will, kann die Treiber aber auch erst direkt vor dem ersten Testlauf mit G-Code einstellen.

Die Treiberbausteine haben ein kleines Potentiometer, die elektrische Spannung der Einstellschraube des Potentiometers gegenüber Erde (GND) entspricht im Wert dem halben Motorstrom. Die Schrittmotoren sollen mit etwa 1,6 A maximal betrieben werden, daraus ergibt sich eine einzustellende Spannung von 0,8 V. Besonders präzise lässt sich dieser Wert einstellen, wenn man eine Strippe des Voltmeters mit GND verbindet und die andere mit einer Krokodilklemme am Schraubendreher festklemmt. Man kann dann während der Einstellung des Potentiometers exakt die aktuelle Spannung auf dem Voltmeter verfolgen.

Software

Mechanik und Elektronik sind jetzt fertig, nun fehlt noch die Software, damit sich der Schlitten genau wie beabsichtigt bewegt. Um von einer Zeichnung zum Fräsprogramm in Aktion zu gelangen, braucht es für die Mobi-C vier Schritte:

1. Erstellung einer Vektorgrafik
2. Erstellung der Fräspfade (G-Code) mit einer CAM-Software
3. Einbau der Pausen, die es ermöglichen, die Fräse zwischendrin manuell hoch- oder herunterzustellen
4. Steuerung der Maschine mit dem erstellten G-Code

Jeder dieser Schritte wird im Folgenden einzeln erklärt. Einen konkreten Beispielablauf beschreiben wir in einem ausführlichen Online-Artikel (siehe Link in der Kurzinfo).

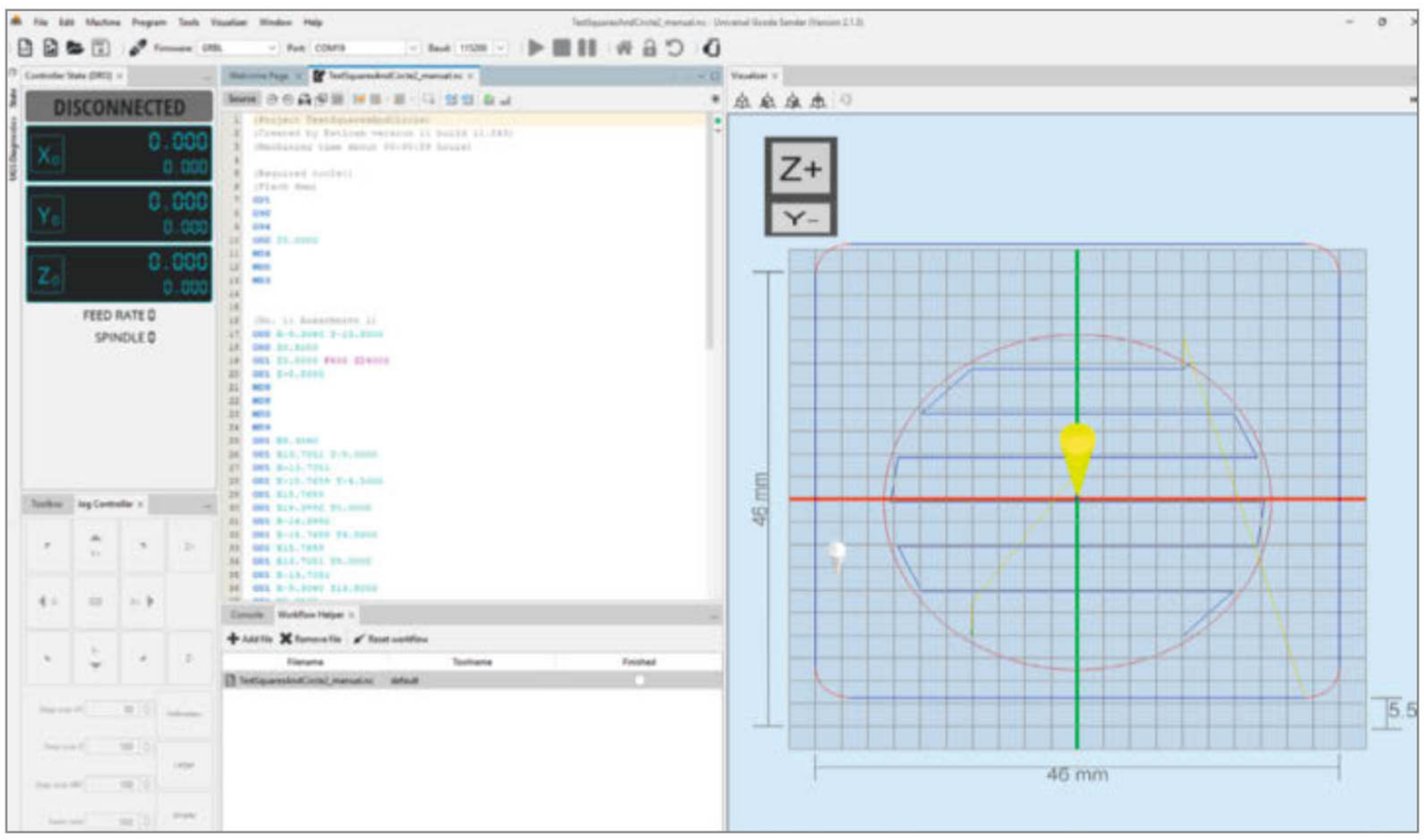


Bild 16: Die grafische Oberfläche des Universal-G-Code-Sender auf dem Windows-Rechner, es gibt auch Versionen für Linux und macOS.

Vektorgrafik

Eine Vektorgrafik kann mit jedem beliebigen Zeichenprogramm erstellt werden, das den Export von DXF- oder SVG-Dateien erlaubt. Ich persönlich nutze gerne die Open-Source-Software Inkscape, die es besonders einfach macht, auch importierte Pixelbilder zu vektorisieren. So können beispielsweise schwarz-weiße Icons mit der Funktion „Bitmap nachzeichnen“ schnell in Zeichnungen umgewandelt werden (siehe „Mehr zum Thema“ in der Kurzinfo). Zur Erstellung von präzisen Zeichnungen mit vordefinierten Dimensionen nutze ich hingegen Fusion 360. Durch die Verwendung von Parametern für Größen und Radien erlaubt es das Programm, Zeichnungen sehr universell zu gestalten und beliebig anzupassen.

Fräspfade

Auch für die Erstellung von Fräspfaden aus der Vektorvorlage hat man sehr unterschiedliche Möglichkeiten, so bieten sowohl Inkscape als auch Fusion 360 dafür eigene Werkzeuge an. Ich habe mich jedoch für das Programm Estlcam entschieden, da es sehr einfach zu bedienen ist und sowohl eine automatische als auch eine manuelle Pfaderstellung erlaubt. Eine Testversion von Estlcam gibt es gratis, eine Lizenz kostet 50 Euro und die Handhabung hat die Make auch schon mal in einem Videotutorial vorgestellt (siehe Link).

Allen Programmen gemein ist, dass sie eine Liste von Anweisung im sogenannten G-Code erstellen. G-Code weist die Steuerung einer CNC-Maschine in Form von einzelnen Befehlen, die meist mit `G` beginnen (daher der Name), an, welche Position mit welcher Geschwindigkeit von der Maschine angefahren werden soll. Weitere Codes, die mit `M` beginnen, steuern zusätzlich das Ein- und Ausschalten der Spindel, der Kühlflüssigkeit und viele weitere Optionen. In diesem Projekt missbrauchen wir die Codes `M03`, `M04`, `M08` und `M09` für die Steuerung der LEDs, dazu gleich mehr.

Einbau von Pausen

Die meisten CNC-Maschinen haben eine gesteuerte Z-Achse: Diese fährt die Spindel herunter, wenn gefräst wird, und hoch, wenn die

Spindel eine Traverse zur nächsten Frässtelle fährt, ohne Material abzutragen. Da bei der Mobi-C in der Grundversion keine Z-Achse vorhanden ist, sondern die Z-Position des Fräskopfes manuell verstellt werden muss, muss der Programmcode vor und nach den Traversen angehalten und der Nutzer über die notwendige manuelle Aktion informiert werden.

Dies geschieht über zwei LEDs. Leuchtet die grüne, muss die Fräse heruntergestellt werden, leuchtet die rote, muss man die Fräse hochstellen. Die grüne LED ist am Ausgang für die Spindelrichtungssteuerung des CNC-Shield angeschlossen. Das bedeutet, dass sie über die Codes `M03` und `M04` ein- beziehungsweise ausgeschaltet werden kann. Die rote LED ist am Digitalausgang für das Kühlmittel angeschlossen, sie wird mit `M08` ein- und mit `M09` ausgeschaltet. Das Programm kann mit dem Befehl `M00` angehalten werden und ein Impuls auf den digitalen Eingang „Resume“ am CNC-Shield, an den bei der Mobi-C der Taster angeschlossen wird, führt das Programm fort.

Das von Estlcam erstellte G-Code-Programm muss deshalb noch so angepasst werden, dass jedes Mal, wenn die Spindel hochgefahren werden muss, eine Pause eingefügt wird und die rote LED leuchtet – an diese Stelle im Programm muss man die Zeilen `M00` und `M08` einfügen. Nachdem der Nutzer durch Tastendruck bestätigt hat, dass die Fräse hochgestellt wurde, geht das Programm weiter, der zusätzlich eingefügte Befehl `M09` löscht dabei zunächst wieder die rote LED. Für das Herunterfahren des Fräsers werden entsprechend die Befehle `M03` (grüne LED an), `M00` (Programm anhalten) und `M04` (grüne LED aus) benötigt.

Eine manuelle Eingabe der Codezeilen wäre zwar möglich, aber in der Praxis natürlich viel zu aufwendig und fehleranfällig. Daher habe ich ein kleines Programm geschrieben, das es mir beim Drücken der Taste F12 erlaubt, eine Datei auszuwählen, die entsprechend den zuvor erklärten Vorgaben modifiziert und dann unter gleichem Namen mit angehängter Zeichenkette „_manual“ gespeichert wird. Das Programm habe ich mit der Windows-Tastenautomatisierung Autohotkey programmiert, einem sehr wertvollen Windows-Helfer. Wer sich dieses Programm nicht selbst installieren möchte, kann einfach für die Anpassung seines G-Codes die exe-Datei transformGCODE.exe herunterladen und ausführen und kommt so zum gleichen Ergebnis.

Steuerung der Maschine mittels G-Code

Im letzten Schritt benötigt man noch ein Programm, das den G-Code an die Steuerung (den Arduino) überträgt und eine Überwachung der Programmausführung erlaubt. Leider kann man hierfür nicht die Steuerungssoftware von Estlcam nutzen, sie unterstützt keine CoreXY-Anordnung. Ich habe mich daher für den Universal-G-Code-Sender (UGS, Bild 16) in Kombination mit der Software GRBL für den Arduino entschieden, beides Open-Source-Softwares (Download siehe Link in der Kurzinfo).

```
145  // Enable CoreXY kinematics. Use ONLY with CoreXY machines.
146  // IMPORTANT: If homing is enabled, you must reconfigure the homing cycle #defines above to
147  // #define HOMING_CYCLE_0 (1<<X_AXIS) and #define HOMING_CYCLE_1 (1<<Y_AXIS)
148  // NOTE: This configuration option alters the motion of the X and Y axes to principle of operation
149  // defined at (http://corexy.com/theory.html). Motors are assumed to positioned and wired exactly as
150  // described, if not, motions may move in strange directions. Grbl requires the CoreXY A and B motors
151  // have the same steps per mm internally.
152  #define COREXY // Default disabled. Uncomment to enable.
```

Bild 17: Diese Zeile muss für GRBL in der Datei config.h einkommentiert werden.

Die GRBL-Software für den Arduino erlaubt es, G-Codes über die serielle Schnittstelle an den Arduino zu schicken, dieser setzt diese dann in die entsprechende Steuerung der Schrittmotoren und digitalen Ausgänge um. Da die Steuerung der Motoren beim CoreXY-System ganz anders als bei klassischer XY-Achsensteuerung ist, muss die Datei Config.h des GRBL-Codes angepasst werden, bevor das Programm für den Arduino kompiliert wird.

Dazu kopiert man das gesamte GRBL-Verzeichnis in das libraries-Verzeichnis der Arduino IDE, dieses befindet sich unter Windows normalerweise in Dokumente/Arduino/libraries. Die Datei config.h kann dann mit Notepad oder einem anderen Editor geöffnet werden. Bild 17 zeigt die notwendige Modifikation in Zeile 152, die vorangestellten Kommentarzeichen `//` müssen entfernt werden. Anschließend startet man die Arduino IDE und wählt unter Beispiele grbl/grblUpload aus. Dies wird dann über einen Klick auf das Symbol mit dem Pfeil nach rechts kompiliert und hochgeladen.

Startet man jetzt auf dem per USB mit dem Arduino verbundenen Rechner den UGS, muss nur noch in der Kopfzeile der COM-Port des Arduino ausgewählt werden und man kann anschließend über das Stecker-Symbol versuchen, eine erste Verbindung mit dem Arduino aufzubauen. Funktioniert das, wählt man über das Machine-Menü den Setup-Wizard und kann die Mobi-C so konfigurieren, dass der Schlitten um genau jene Distanz verfahren wird, die man im Steuerungsprogramm als Strecke vorgibt.

Einzig die Konfiguration der X- und Y-Richtung kann etwas verwirrend sein: Durch die CoreXY-Konfiguration bewirkt ein Häkchen bei der Invertierung der X-Achse auch eine Invertierung der Y-Achse. Ich bin daran fast verzweifelt, bis ich herausgefunden habe, dass ich einfach die beiden Anschlüsse für die Schrittmotorsteuerung vertauschen musste, sodass die Zuordnung der Motoren zu den Achsen umgekehrt wird, um die gewünschte Konfiguration zu erhalten.

Ist die Maschine kalibriert, kann man das erste Programm laden. Zuvor fährt man den Schlitten in UGS an den Anfang der X- und der Y-Achse. Der unter dem Link zu findende G-Code namens FlightTest.nc zeigt beim Ausführen, ob alles richtig eingestellt ist. Wird er gestartet, leuchtet zunächst die rote LED. Drückt man den Taster, fährt der Schlitten etwas vom Rand weg und die grüne LED leuchtet. Drückt man nochmals den Taster, fährt der Schlitten die Form eines Rechtecks mit 30 mm Kantenlänge ab und beendet dann das Programm.

Man kann in UGS den Arbeitsbereich der Maschine definieren (Anleitung siehe Link in der Kurzinfo), aber das ist gar nicht nötig. Ich lade immer zuerst meinen G-Code, platziere den Frässchlitten auf den gewünschten Nullpunkt, setzte in UGS die Position auf 0,0,0 und fahre dann den äußeren Umfang des zu fräsenden Ausschnitts ab, um sicherzustellen, dass der Schlitten dabei nicht an eine Begrenzung stößt.

Ein erfolgreicher FlightTest erlaubt den ersten Frästest mit der Kantenfräse. Wie eingangs erwähnt, nutze ich eine Makita DRT50Z. Die Aussparungen im Schlitten passen genau für den Frässchuh, von dem ich die untere Kunststoffplatte abgeschraubt habe. Ich habe mir einen zweiten Frässchuh gekauft, um zum einen den im Schlitten montierten Frässchuh nicht immer abschrauben zu müssen und zum anderen auch die Tiefeneinstellung der Fräse mit dem zweiten Schuh sauber einstellen zu können.

Ausblick

Natürlich gibt es noch viele Details der Mobi-C zu beleuchten, aber aus Platzgründen haben wir die detaillierte Beschreibung zur Höhenverstellung des Rahmens über dem Werkstück, zur Frästiefen-Einstellhilfe sowie zur Schleppmesser-Erweiterung in einen Online-Artikel ausgelagert (siehe Link in der Kurzinfo). Jetzt freut sich die Make-Redaktion aber erst mal auf Feedback und auf Fotos von Nachbauten oder Modifikationen der Mobi-C und von damit umgesetzten Fräsprojekten – aber natürlich auch über zusätzliche 3D-gedruckte Adapter für andere Fräsen, die wir im Github-Repository zu diesem Projekt sammeln können. Denn die Konstruktion ist ein Work in Progress und wird bei Interesse sicher noch Stoff für weitere Make-Artikel liefern. *—pek*

In der Mitte die Oxocard Connect, links davon zwei Cartridges zur deren Erweiterung. Rechts die Oxocard Science+.

Oxocard Connect und Science+

So groß wie eine Scheckkarte oder sogar nur wie eine größere Briefmarke, aber mit ESP32, Farbdisplay und noch viel mehr ausgestattet: Die Oxocards laden alle ab 14 Jahren zum Programmiereinstieg ein.

von Peter König und Daniel Schwabe

Die Oxocard-Reihe umfasst verschiedene Einplatinencomputer, die von der Schweizer Firma Oxon speziell fürs Programmierenlernen entwickelt werden. Dazu dient die Programmiersprache NanoPy, die syntaktisch an die weitverbreitete Sprache Python angelehnt ist. Eine maßgeschneiderte Entwicklungsumgebung ringsherum samt integrierter Tutorials läuft im Browser. Wir haben die beiden jüngsten Sprösslinge der Familie getestet.

Oxocard Connect

Wie schon bei einigen anderen Produkten der Serie bildet auch bei der Oxocard Connect ein ESP32 den Kern für eine Lernplattform für Programmiereinsteiger ab 14 Jahren. Ausgestattet ist die Connect mit einem WLAN-Modul, 8 MB Speicher und 2 MB RAM. Die grafische Ausgabe übernimmt ein 240 × 240 Pixel großes Display, ein kleiner Joystick bietet Möglichkeiten zur Eingabe. Die Oxocard Connect lässt sich wie die anderen Geräte der Reihe über den Editor im Browser übers WLAN mit neuen, selbst geschriebenen Skripten bespielen.

Anders als die bisherigen Oxocards verfügt die Connect über einen 16-Pin-Steck-Slot, über den sogenannte Cartridges an die Oxocard Connect angeschlossen werden können. Für diese neue Schnittstelle gibt es aktuell vier Erweiterungen im Oxocard-Shop: ein klassisches Breadboard, auf das Bauteile aufgesteckt werden können, einen Sensor zur Distanzmessung via Laser, einen Sensor zur Messung von etwa CO_2 in der Luft und ein sogenanntes Veroboard, auf das sich Schaltungen auflöten lassen. Alle Cartridges verfügen über einen EPROM-Speicher. Darauf befindet sich der Treiber für die Elemente auf der Platine und Demoprojekte, welche die Funktionen der jeweiligen Erweiterung zeigen. Auf diese Speicher kann auch eigener Code aufgespielt werden.

Auch wenn es sich bei den Sensoren um hochwertige Vertreter ihrer Zunft handeln, sind die Breadboard- und Veroboard-Erweiterungen die interessanteren, weil vielfältiger nutzbaren Steckmodule: Sie machen einzelne Pins des Mikrocontrollers zugänglich. Neben fünf Volt, Ground und VDD hat man dadurch Zugriff auf I^2C, SPI, ADC und fünf GPIO-Ports. Das sind alles Verbindungspins, die auch die sonst üblichen Mikrocontroller in freier Wildbahn bieten. Zusätzlich zu den zugänglichen Kontakten sind die GPIO-Pins noch mit LEDs bestückt. Dadurch kann man Codespielereien, die auf blinkenden LEDs aufbauen, ohne ein einziges Bauteil direkt umsetzen.

Die Connect in der Praxis

Die Connect zeigt im Zusammenspiel mit externen Elektrobauteilen ihre Stärke. Dadurch erlaubt sie nicht nur einen Einstieg in die Programmierung, sondern auch in die Elektrotech-

nik. Hat man sich für eines der Beispielprojekte aus dem NanoPy-Editor entschieden, hilft einem das integrierte Schritt-für-Schritt-Tutorial beim Heraussuchen der einzelnen Bauteile und zeigt einem ein Bild der korrekten Verkabelung auf dem Steckbrett. Außerdem wurde auch das integrierte Codelexikon im Editor um Connect-spezifische Befehle erweitert.

Die Oxocard Connect kann man sowohl einzeln (ohne Erweiterungskarten) als auch in einem Set – dem Innovator Starter Kit – kaufen. In diesem Set sind neben der Oxocard auch die Breadboard-Erweiterung und eine Variation an Bauteilen enthalten. Damit lässt sich dann die Programmreihe namens Elektrokurs im NanoPy Editor nachbauen. Dieser Kurs besteht aus 16 kleinen Projekten, die man auf dem Breadboard aufbaut und die einem nach und nach elektronische Schaltungen näherbringen.

Die Beispielprogramme gehen den klassischen Weg von blinkender LED über die Ansteuerung von Servomotoren bis hin zur Verwendung von verschiedenen Sensoren. Das sind alles Anwendungen, die auch in richtigen Maker-Projekten genutzt werden, sodass alle Beispiele auch außerhalb des Lernrahmens von Nutzwert sind.

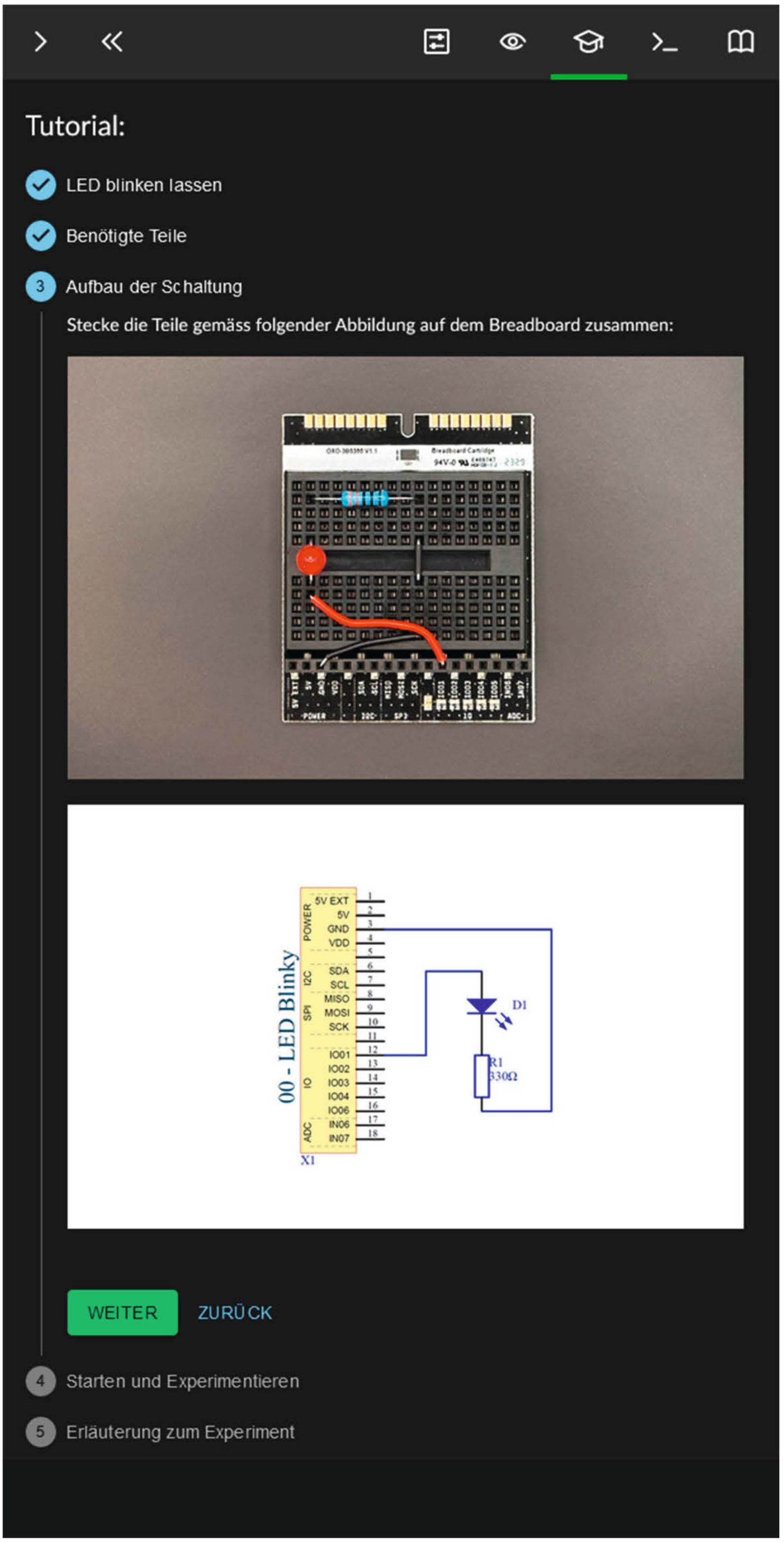
Oxocard Science+

Die Oxocard Science+ gehört in die Board-Familie Oxocard Mini, die es damit jetzt in vier Varianten gibt. Diese haben alle gemein, dass sie auf ihrer hübsch gestalteten Platine von 85 × 55 mm Größe jeweils ein rund 25 mm im Quadrat messendes Farbdisplay mit 240 × 240 Pixeln sowie fünf Taster mitbringen, die als Steuerkreuz mit Mitteltaster angeordnet sind.

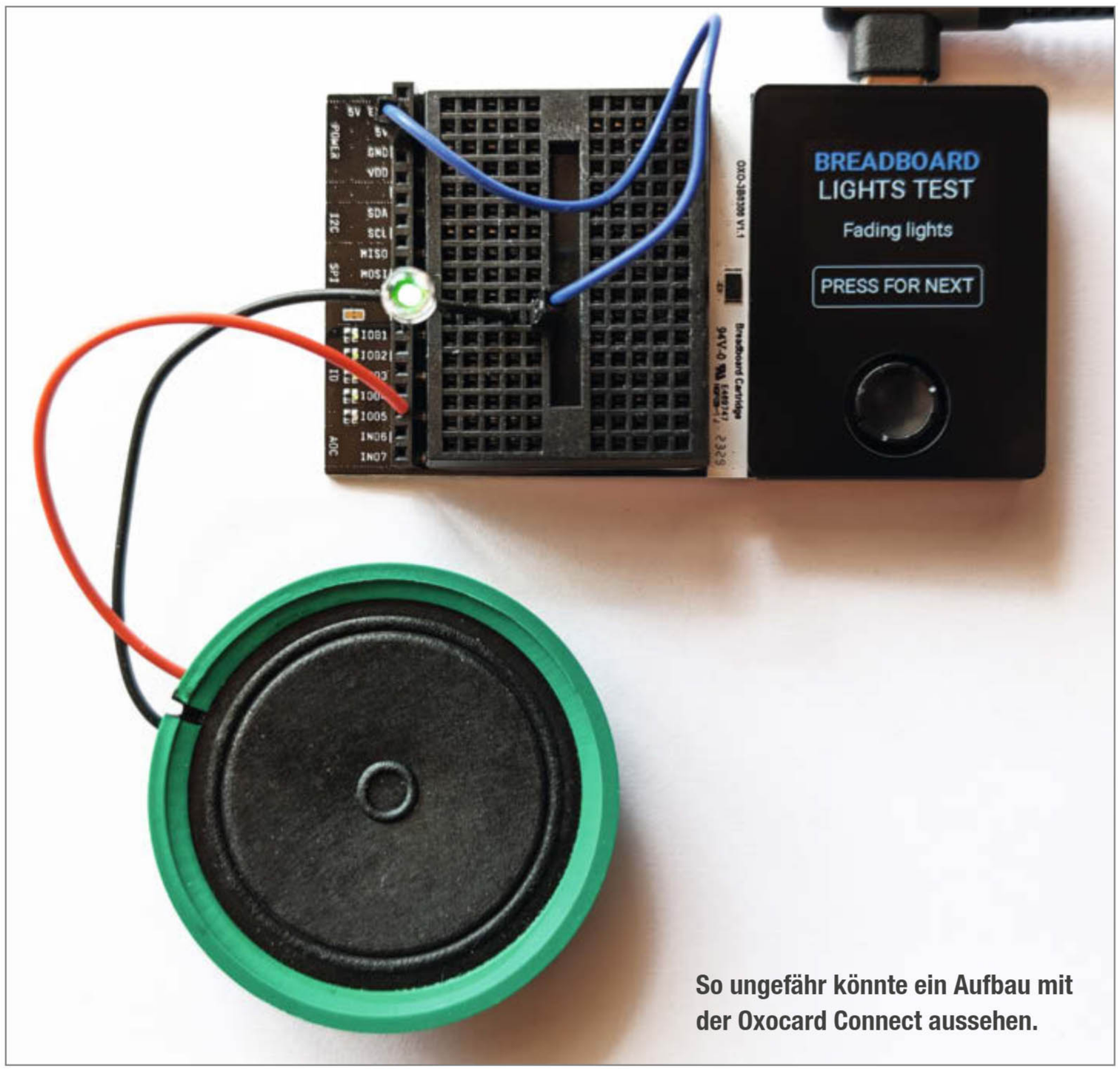
Das Herz der Minis ist ebenfalls ein ESP32 mit 2 MB RAM und 8 MB Flash, ihren Strom beziehen die Cards über eine USB-C-Buchse vom Rechner, aus einem Steckernetzteil oder einer Powerbank – einen Akku gibt es wie bei der Connect auch hier nicht. Außer im grafischen Design der Platine unterscheiden sich die einzelnen Minis durch ihre Sensorausstattung, worin am Ende auch der Grund für die (erheblichen) Preisunterschiede der Cards liegt.

Sensoren für jeden Bedarf

Bisheriges Flaggschiff der Serie war die Oxocard Science (kurz vorgestellt in Make 3/23, S. 113), die neue Science+ bringt verglichen damit nochmals mehr Sensoren an den Start. So sind auf der Platine zwei Luftsensoren von Sensirion verbaut: Der SGP41 liefert Daten zum Gehalt an flüchtigen organischen Verbindungen (VoC) und Stickoxiden (NOX), der SCD41 hingegen misst die CO_2-Menge in der Luft. Die bisher erhältliche Oxocard Science (ohne Plus) hingegen spürt keinerlei NOX auf, benutzt für die VoC-Messung einen Sensor von Renesas (ZMOD4410) und ermittelt aus dessen Werten den CO_2-Gehalt.



Die Schritt-für-Schritt-Anleitungen des Editors wurden mit Schaltplänen erweitert.



So ungefähr könnte ein Aufbau mit der Oxocard Connect aussehen.

Außerdem bietet die Science+ einen Time-of-Flight-Entfernungssensor mit einer Auflösung von 8 × 8 Pixeln (auch der ist als Connect-Cartridge erhältlich) sowie einen Piezo-Lautsprecher. Von der Oxocard Science hat sie außerdem die übrige Ausstattung geerbt: Sensoren für Temperatur, Luftfeuchtigkeit und atmosphärischen Druck, ein Mikrofon, einen 3D-Beschleunigungssensor sowie einen für sichtbares Licht und Infrarot. Mit dieser Ausstattung lassen sich eine Reihe von Messwerten ausgeben, unter anderem die Intensität von Licht und Schall im Raum, aber auch die Lichtfarbe (angezeigt als Anteile von RGB) oder die per FFT (Schnelle Fourier-Transformation) ermittelte dominante Frequenz.

Falls man noch weitere Sensoren oder andere Peripherie anschließen will, gibt es zu beiden Seiten der USB-C-Buchse noch insgesamt sieben GPIO-Kontakte, etwa für die I^2C-Kommunikation, nebst drei für GND, 3,3 und 5 Volt. Die Kontakte sind als Lötpads mit Löchern für optionale Pinleisten ausgeführt.

Programmvielfalt zur Wahl

Die auf der Karte vorinstallierten Beispielprogramme geben bereits eine plastische Vorstellung, was man mit diesen Messwerten so alles anstellen beziehungsweise wie man sie visualisieren kann. Neben zwei Minispielen sind vor allem verschiedene Darstellungen der Sensordaten als Beispiele dabei und manche davon lassen sich direkt in dieser Form im Alltag nutzen: So zeigt der CO_2-Logger neben dem aktuellen Wert in ppm (parts per million) auch den Verlauf des CO_2-Werts für die Minute, Stunde, den Tag und die Woche in vier untereinander angeordneten Balkendiagrammen an.

Andere Visualisierungen sind eher spielerische Demos, etwa die Umsetzung des aktuellen Schallpegels in einen 3D-Stern (DB Crystal) oder die Darstellung der 64 Messwerte des Time-of-Flight-Sensors als 3D-Quader (Depth Camera). Apropros eigene Anwendungen: Auch wenn die Science+ bereits mit vielen nützlichen Programmen ausgeliefert wird, soll sie am Ende wie alle Produkte des Herstellers Oxon dazu animieren, selber Code zu schreiben. Hierfür gibt es in der Online-IDE NanoPy unter „Einstieg/Erste Schritte/OXOCARD Science+“ ein eigenes kurzes Tutorial, das vor allem die Sensoren vorstellt.

Programmieren im Browser

Mithilfe eines umfangreichen Web-Editors, der wie eine „echte“ IDE funktioniert, kann man die Oxocards über WLAN oder ein USB-Kabel bespielen. Für ersteres müssen die Oxocards über ein Handy mit dem normalen WLAN-Netzwerk verbunden werden. Im Web-Editor hat man Zugriff auf alle Beispielprogramme für alle Oxocards, die Schritt für Schritt erklärt werden. Konstanten kann man zudem mittels Schieberegler verstellen. Dadurch kann man vorgefertigte Programme direkt anpassen, ohne im Code etwas zu verändern. Das hat Methode, denn wer mit den Oxocards ins Programmieren einsteigt, kann erst mal die fertigen Beispiele nutzen, dann mit den Konstanten rumspielen und anschließend versuchen, den Code zu modifizieren.

Selbst geschriebene Programme werden in der Cloud des Herstellers gespeichert, können aber auch als Textdatei heruntergeladen werden. Solange man nur mit den Sensoren arbeitet, die die Cards und Cartridges mitbringen, muss man sich auch mit keinen Pinzuweisungen oder Bibliotheken herumschlagen, es ist alles aus einem Guss. Gepaart mit einem durchsuchbaren Lexikon aller Funktionen der Programmiersprache NanoPy bekommt man hier einen wirklich leichten, angeleiteten Einstieg in die Programmierung der Oxocards.

Fazit

Durch die wirklich tolle, lernfokussierte Entwicklungsumgebung und die Erweiterung der Tutorials mit Bildern von Beispielschaltungen wird einem das Erstellen eigener Elektronikprojekte mit der Oxocard Connect einfach gemacht. Im Gegensatz zur Oxocard-Mini-Reihe sehen wir den Einsatz der Oxocard Connect vor allem in Schulen. Im Unterricht könnten neben den Oxocards auch die Bauteile zur Verfügung gestellt werden, damit ohne Probleme das volle Potenzial des Gerätes ausgeschöpft werden kann.

Die Oxocard Science+ bietet mit ihrer erweiterten Sensorausstattung nochmals mehr Möglichkeiten für eigene Projekte als die bisherigen Boards der Oxocard-Mini-Serie, ohne dass man irgendwelche zusätzliche Hardware anschließen müsste. Gleichzeitig zeigen die mitgelieferten Beispielprogramme, wie unterschiedlich sich die gemessenen Daten visualisieren lassen.

Die Oxocard Science+ kostet knapp 120 Euro, Interessenten in der Schweiz bekamen sie bei Druckschluss noch direkt beim Hersteller zum Sonderpreis von 99 Schweizer Franken. Die Connect kostet ohne weiteres Zubehör knapp 50 Euro (39 CHF) und das Innovator Starter Kit inklusive Connect knapp 90 Euro (79 CHF). —*das*

Die Oxocards wurden uns für den Test vom Hersteller zur Verfügung gestellt. Einen ausführlicheren Test mit mehr Bildern lesen Make-Abonnenten online (siehe Link).

Alles zum Artikel im Web unter
make-magazin.de/xjuk

Beliebt auf heise+

Ikea-Hack: LED-Leuchtwürfel in ein smartes Gerät umbauen

Tauscht man den Original-Controller beim Frekvens gegen einen ESP8266 aus, lässt sich die LED-Matrix des Würfels intelligent steuern, etwa als autonome Uhr.

heise+ 05. April 2024, 10:00 Uhr 25

Eigene Domain für zu Hause: Reverse-Proxy auf dem Raspberry Pi einrichten

Mit der Software Caddy lassen sich mit verschiedenen Domains unterschiedliche Server im gleichen Netzwerk aufrufen. Wir zeigen, wie es geht.

heise+ 10. April 2024, 10:00 Uhr 52

Herkömmliche Raumthermostate clever ins Smart Home integrieren

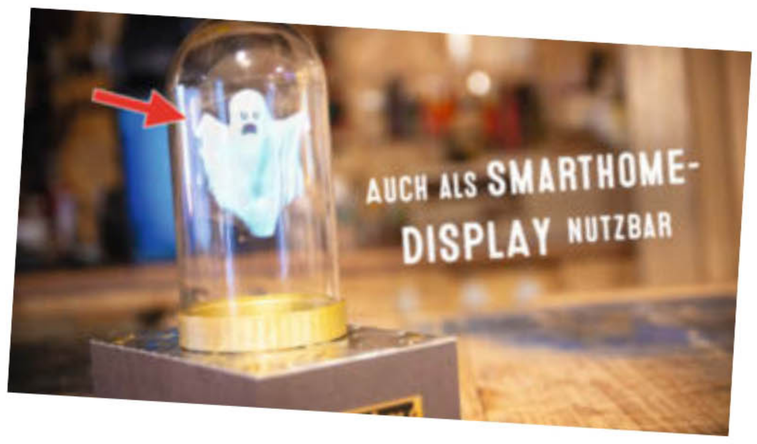
Herkömmliche Raumthermostate steuern die Fußbodenheizung in jedem **Raum** individuell und sorgen so überall für die Wohlfühltemperatur.…

09.04.2024 | Make

So kommen Sie als Make-Abonnent an Artikel hinter der Paywall von heise+: https://heise.de/-7363373

Hologramm als Smart-Home-Display!

Die Idee für dieses Video ist aus dem Artikel „Flaschengeist“ in der Make-Ausgabe 01/2024 entstanden. In dem Artikel wird ein halbtransparentes Bild mit holografischer Anmutung in einen Erlenmeyerkolben projiziert. Johannes baut für dieses Video „Pepper's Ghost“ aus einer Ikea-Glasglocke und einem selbst angefertigten Sockel mit Messingschild eine visuell sehr ästhetische Projektionsbasis. Das fertige Produkt soll allerdings nicht nur schön aussehen, sondern kann auch als praktisches Smart-Home-Display umfunktioniert werden. Egal ob mit dem Handy oder mit einem Display fürs Smart-Home, dieser „Flaschengeist“ kann individuell eingesetzt werden und lässt sich unkompliziert umsetzen. —*dus*



▶ **www.youtube.com/@MakeMagazinDE**

Weitere aktuelle Videos:

Bleib informiert:

www.make-magazin.de

@makemagazinde

Instagram: @makemagazinde
@makerfairedeutschland

Facebook: @makemagazin.de

X (Twitter): @MakeMagazinDE

TikTok: @makemagazinde

Bluesky: @makemagazin.de

WhatsApp Channel:
Make Magazin Deutschland

GitHub: MakeMagazinDE

Threads: @makemagazinde

LinkedIn: linkedin.com/
company/maker-media-gmbh/

Kontakt zur Redaktion
Leserbriefe und Meinungen an:
heise.de/make/kontakt/

Oder diskutiere in unseren Foren online über Themen und Artikel:
www.make-magazin.de/forum

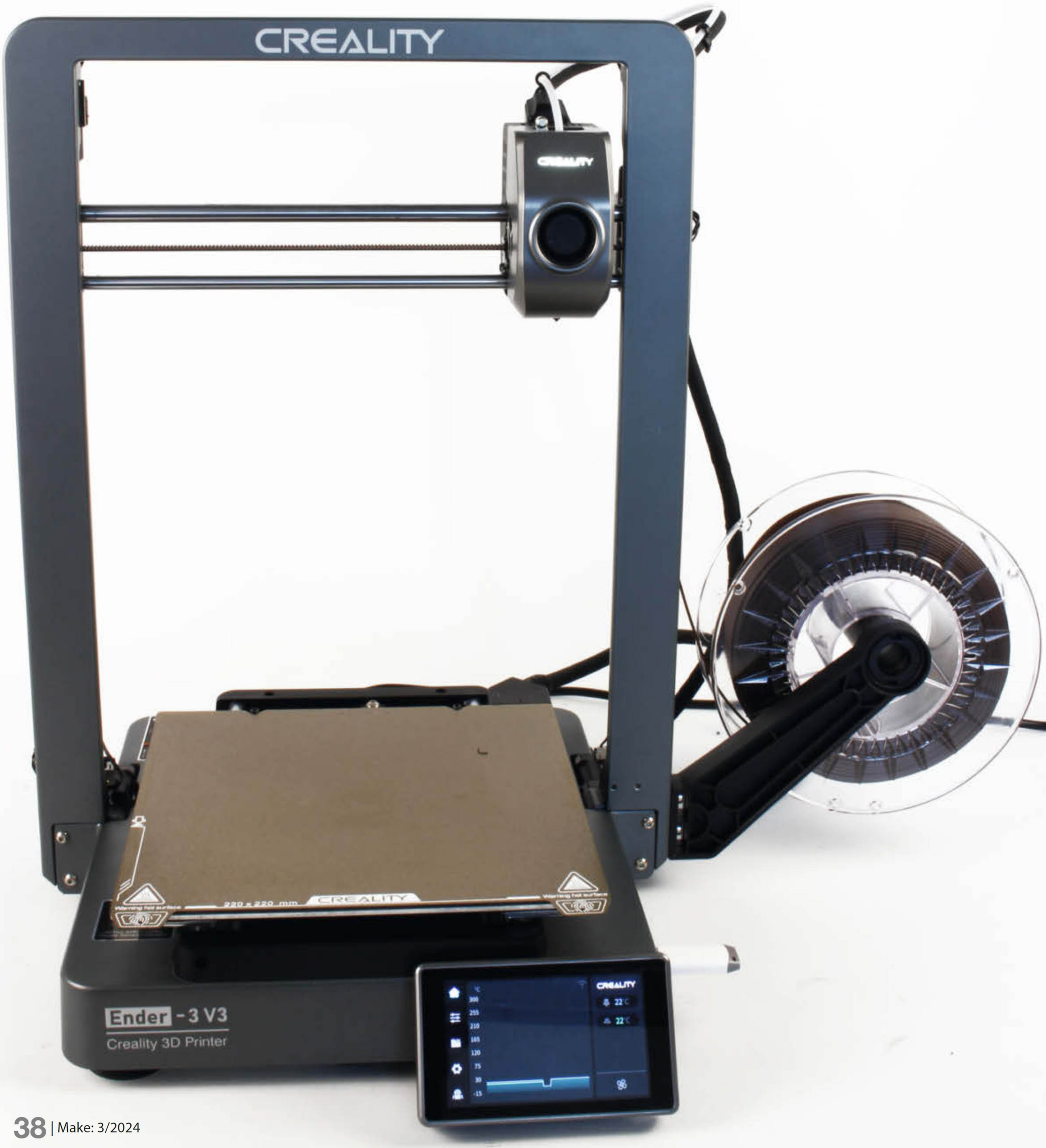
Nichts mehr verpassen:
Abonniere unseren Newsletter!
Weitere Infos unter:
www.maker-faire.de

Wo finde ich die Make am Kiosk?
www.mykiosk.com

Test: Creality Ender-3 V3

Creality hat den Ender-3 völlig überarbeitet: Das Modell Ender-3 V3 soll mithilfe der Firmware Klipper ein echter Hochgeschwindigkeitsprinter sein. Das haben wir natürlich testen müssen.

von Heinz Behling



Kurzinfo

» Neukonstruktion mit auf Klipper basierender Firmware
» Vollautomatische Justierung der Hotend-Position
» Druck mit bis zu 600 mm/s

Mehr zum Thema

» Johannes Börnsen und Pina Merkert: Die Jeder-kann-Drucker, Make 2/24, S. 50
» Heinz Behling: Z-Achsen-Waagesensor, Make 2/23, S. 94
» Heinz Behling: Schneller drucken mit Klipper, Make 3/23, S. 10

Alles zum Artikel im Web unter
make-magazin.de/xm5h

Wie spannt man Ungeduldige auf die Folter? Schenken Sie ihnen einen 3D-Drucker. Die Druckzeiten der meisten Geräte dieser Gattung von zehn und mehr Stunden, selbst bei relativ einfachen Projekten, werden ihn oder sie rasch zur Weißglut bringen.

Die Gründe für das mäßige Tempo liegen zum einen in der Elektronik vieler Geräte, die nicht sonderlich leistungsfähig (schnell) ist, um sich mit hohem Drucktempo bei gleichzeitig guter Druckqualität befassen zu können. Dementsprechend geht die darauf installierte Firmware (meist eine Marlin-Variante) auch gar nicht erst auf das Thema Highspeed ein, sondern lässt es bei recht gemütlichen Werten für Geschwindigkeit und Beschleunigung.

Zum anderen ist die Mechanik vor allem der preiswerteren 3D-Drucker wohl eher in den kaufmännischen Abteilungen der Hersteller entstanden (billige Lager, Wellen mit zu großer Toleranz usw.) als bei den Ingenieuren der Konstruktionsbüros. Elektronik und Firmware (zum Beispiel Klipper) lassen sich zwar relativ einfach austauschen, aber durch die Mechanik sind den Druckern Grenzen gesetzt, die weit unter dem heutzutage Machbaren liegen.

Mit meinem Ender-3 der ersten Serie bin ich inzwischen auch an dieser Grenze angelangt: Alles, was machbar ist, habe ich wohl aus ihm herausgeholt und in der Make mit Ihnen in diversen Artikeln geteilt. Er funktioniert zwar immer noch gut und erheblich besser als ursprünglich, aber inzwischen schiele ich doch neidisch auf die 3D-Drucker der neuesten Generation. Und genau so ein Gerät hat Creality nun auf den Markt gebracht: den Ender-3 V3, nicht zu verwechseln mit den einfacheren Modellen Ender-3 V3 KE und SE, die schon länger angeboten werden.

Die Versprechungen sind groß: Druck mit bis zu 600 mm/s, Beschleunigungswerte von 20 000 mm/s², vollautomatische Installation und Justierung der Druckerachsen, genauere Mechanik und, und, und … Und das bei einem Preis von knapp 400 Euro. Kaum, dass er eine Woche auf dem Markt war, stand er daher auch schon auf meinem Arbeitstisch.

Bausätzchen

Genauer gesagt, lag er zunächst da, denn im großzügig gepolsterten Karton verbargen sich vier Teile, die auf den Zusammenbau warteten: die Grundplattform mit dem Netzteil, der Drucker-Elektronik und dem bis 100 °C beheizbaren Drucktisch, der Portalbogen mit fertig montierter X-Achse und Druckkopf (Hotend und Extruder-Motor), ein Farb-Touchscreen-Display und der Filamentspulenhalter.

Beim Zusammenbau bot sich die Möglichkeit, sich die neue Art des Antriebs von X- und Z-Achse anzuschauen, von Creality CoreXZ genannt. Beide Achsen sind nun über zwei Zahnriemen miteinander gekoppelt, der Antrieb erfolgt gemeinsam über zwei Schrittmotoren am unteren Ende der beiden Portalsäulen (Bild 1). Die Riemen laufen um das Portal komplett herum und überkreuzen sich in der Querstrebe oben. Drehen sich beide Motoren entgegengesetzt, wird die Z-Achse bewegt, der Druckkopf geht also nach oben oder unten. Drehen sich die Motoren gleichsinnig, bewegt sich der Druckkopf auf der X-Achse.

Welche Vorteile bietet das nun? Größter Fortschritt ist der Ersatz der Gewindespindeln für den Z-Antrieb durch den spielfreien Riemenantrieb. Das soll die Genauigkeit der Achse erheblich erhöhen, Creality spricht von 0,25 Mikrometern Positioniergenauigkeit.

Außerdem wird nun der durch den Direktantrieb recht schwere Druckkopf von zwei Riemen und zwei Schrittmotoren bewegt. Dadurch können am Kopf höhere Antriebskräfte wirken, was stärkere Beschleunigung und höheres Tempo ermöglicht.

Der Portalbogen musste mit vier Schrauben pro Seite mit der Grundplattform verschraubt werden (Werkzeug liegt bei, Bild 2). Aber Achtung: Vorher die Netzspannung korrekt einstellen, der Schalter war nach der Montage des Bogens nicht mehr zugänglich. Beide Teile sind nun übrigens aus Alu gegossen und nicht mehr aus mehreren Aluprofilen zusammengeschraubt. Das erspart das Ausrichten, für das früher, wenn man es wirklich genau mit den rechten Winkeln nahm, schon mal die ein oder andere Stunde drauf ging.

Doch weiter mit dem Zusammenbau: Das Displaykabel einzustecken und den Screen

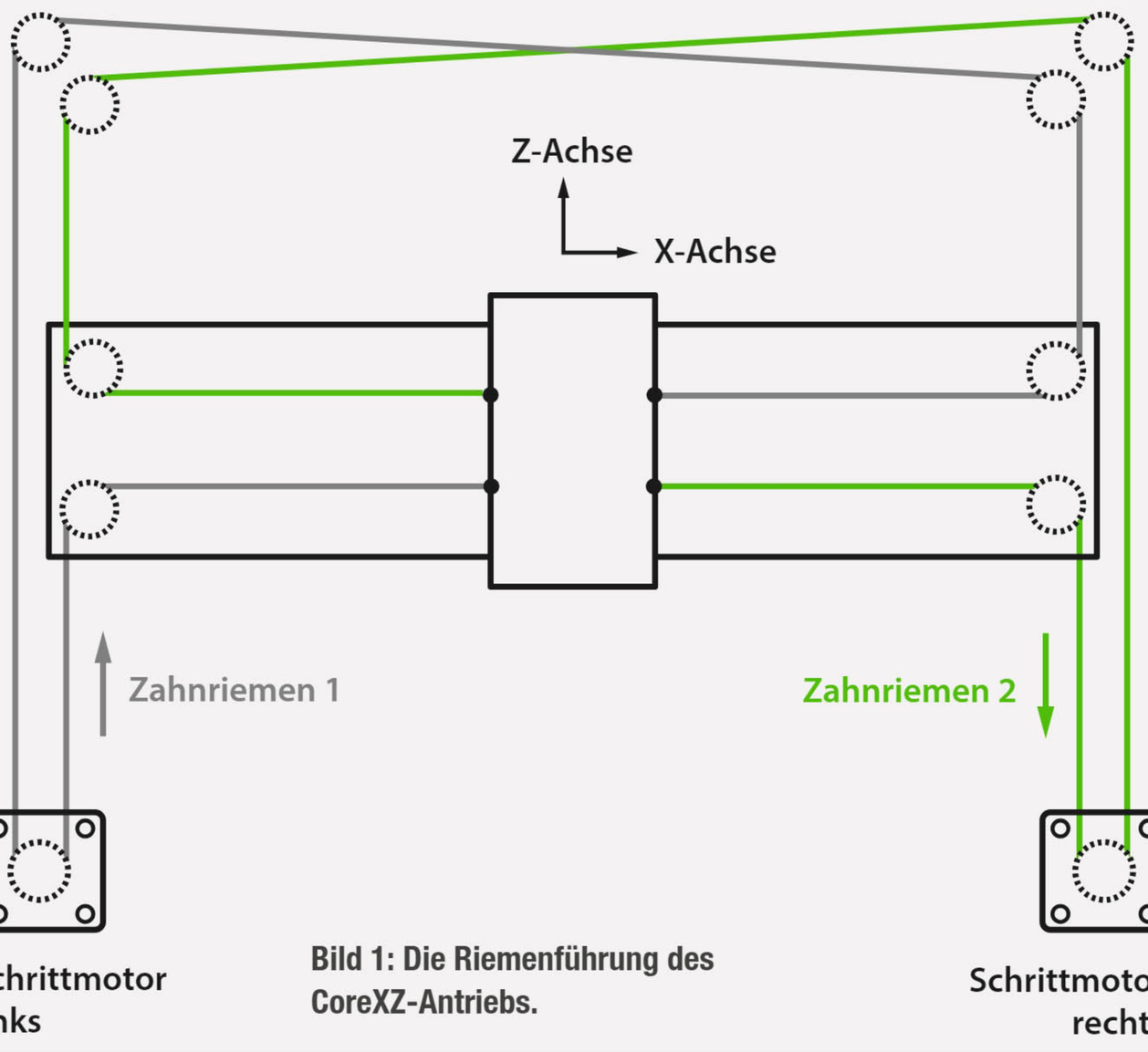


Bild 1: Die Riemenführung des CoreXZ-Antriebs.

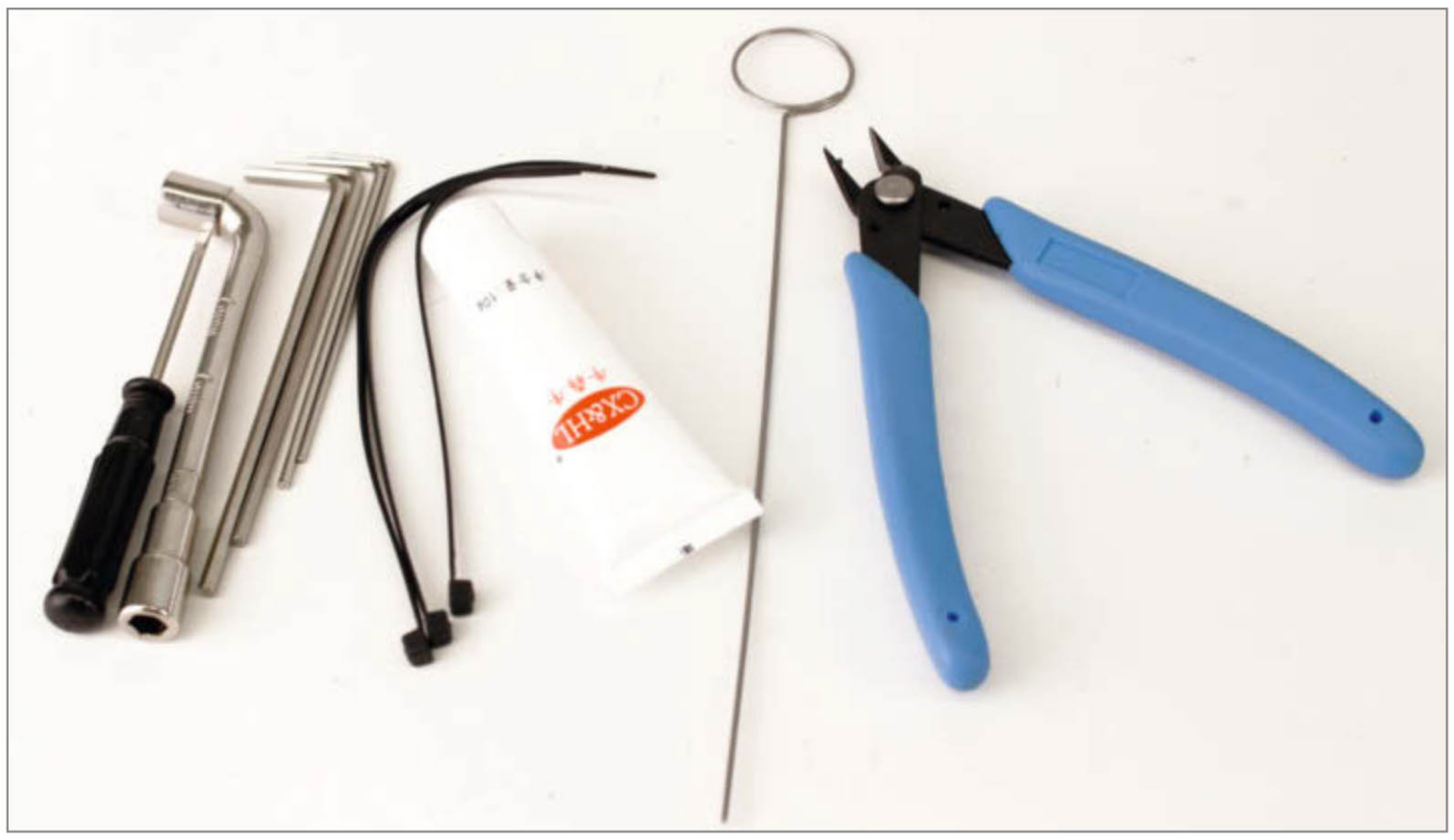


Bild 2: Das Zubehörpaket des Ender-3: In der weißen Tube ist Schmierfett, mit dem man die Stahlwellen des Druckers ab und zu sparsam versorgen sollte.

sowie den Spulenhalter in ihre Halterungen einzurasten war sehr einfach. Die Displayhalterung erschien jedoch recht wackelig, was sich im Laufe des Tests noch verstärkte. Man sollte es vielleicht mit einem Tropfen Heißkleber sichern. Bei dieser Gelegenheit fand sich in der sehr knappen Aufbauanleitung ein ungewollter Witz, denn der Drucker soll weder mit aufgestecktem noch mit nicht aufgestecktem Kabel in Betrieb genommen werden.

Der Spulenhalter rastete an der rechten Seite des Druckers ein (siehe Titelbild). Das machte das Gerät leider sehr breit, sodass an eine Unterbringung in dem vom alten Ender benutzten Ikea-Möbel nicht zu denken war. Selbst wenn man sich für die Filamenthalterung etwas anderes ausdenken würde, gibt es noch einen Grund, der gegen die Schrankunterbringung spricht. Dazu später mehr.

Schließlich waren noch die Kabel am Hotend, den Schrittmotoren und dem Filamentendschalter sowie das Stromkabel einzustecken und dann konnte es losgehen mit der Inbetriebnahme.

Vollautomatik

Nach dem ersten Einschalten begann automatisch der Installationsvorgang mit der Sprachauswahl. Der Drucker beherrscht unter anderem auch Deutsch. Nach dem Akzeptieren der Datenschutzbestimmungen und der Eingabe der WLAN-Daten folgte die Aufforderung, sich per QR-Code und Smartphone mit der Creality-Cloud zu verbinden. Dadurch kann der Drucker später u. a. per Smartphone-App gesteuert werden. Da ich aber kein Verlangen verspürte (und auch heute noch nicht verspüre), meine Projekte auf einem chinesischen Server zu speichern, tippte ich am Touchscreen einfach auf weiter und verzichtete auf die Cloud.

Dann zeigte der Drucker an, dass es eine neuere Firmware-Version online gäbe und fragte, ob die installiert werden solle. Das kann ich nur empfehlen, denn dadurch wird zum Beispiel das Rooting des Druckers möglich, mit dem man nicht nur vollen Zugriff auf die Firmware, sondern auch auf das zugrunde liegende Betriebssystem (Linux) bekommen kann.

Es folgte der Selbsttest und die Kalibrierung. Bei diesem Vorgang, der etwa 15 Minuten dauerte, checkte die Elektronik zunächst alle Komponenten und deren Beweglichkeit. Bei der anschließenden Messung der Resonanzfrequenzen zeigte sich, dass der Drucker unbedingt auf einem sehr stabilen Untergrund stehen muss. Mein zunächst benutzter, einem Blumentopfständer ähnelnder Tisch wanderte bei den heftigen Messschwingungen langsam über den Boden. Die Geräuschentwicklung lag dabei zwischen einem fast unhörbaren Summen bis hin zur unmittelbaren Nähe eines startenden Hubschraubers! Aber diese Messung wird ja nur selten notwendig (beim Tausch beweglicher Komponenten oder der Installation einer neuen Firmware-Version).

Dann wurde mithilfe der fünf Wägesensoren (vier an den Ecken im Drucktisch, einer am Hotend) die Lage der Tischecken vermessen. Außerdem tastete die Düsenspitze in einem 5-×-5-Raster die Oberfläche des Druckbetts ab, um ein Höhenprofil anzulegen.

Während des ganzen Vorgangs prangte auf dem Display des Druckers die Meldung, dass man das Gerät bei der Messung auf keinen Fall berühren soll. Klar, denn das könnte die sensiblen Wägesensoren zu Fehlergebnissen veranlassen. Auch die zum Druckbett und Hotend führenden Kabel sollten sich frei bewegen können, da andernfalls Kräfte auf die Sensoren übertragen werden, die zu Fehlmessungen führen können. Dies ist auch der Grund, warum eine Unterbringung in einer Vitrine schwierig ist, denn es muss nicht nur genug Platz für Drucker und Filament vorhanden sein, sondern auch für die Kabel.

Bild 3: Überhänge, Brücken und auch die spitzen Nadeln druckt der Ender auch bei hoher Geschwindigkeit gut.

Erster Druck

Das Einlegen des Filament war sehr einfach: An der Unterseite des an der rechten Portalsäule angebrachten Filament-Ende-Sensors

musste das Filament in ein kleines Schlauchstück so weit eingeschoben werden, bis es im Hotend ankam (Widerstand). Dann war noch der Arretierhebel am Druckkopf nach links zu schieben. Im Einstellmenü auf dem Touchscreen folgte ein Fingertipp auf Einziehen und der Drucker heizte auf, zog das Filament weiter ein, bis es aus der Düse herausquoll. Der Drucker war somit betriebsbereit. Insgesamt hatte der Zusammenbau und die Inbetriebnahme kaum 40 Minuten gebraucht.

Jetzt fehlte noch ein passender Slicer. Für das bisher von mir verwendete Cura gab es zum Testzeitpunkt noch kein passendes Druckerprofil. Das Zubehörpaket des Ender-3 V3 enthält aber neben diversen Schraubenschlüsseln, Düsenreiniger und Schmierfett auch einen USB-Stick mit diversen Anleitungen und die Slicer-Software Creality Print.

Zum Entpacken war ein RAR-fähiges Programm (auf Windows beispielsweise 7zip) notwendig. Die Installation des Slicer war unspektakulär, er fand den Drucker beim automatischen Scannen im Netz und verband sich mit ihm. Die Bedienung des Slicer war der von Cura nicht unähnlich, inklusive der zahlreichen Einstellungsmöglichkeiten. Druckdaten konnten per Netzwerk auf den Drucker oder per USB-Stick übertragen werden. Per Netzwerk gesendete Daten landeten direkt auf der eMMC-Disk (acht Gigabyte) des Druckers. Der Druckvorgang ließ sich am Slicer oder am Drucker starten. Auch per USB übertragene Druckdaten wurden dann zunächst auf die Disk des Ender kopiert, sodass der Stick während des Druckens nicht im Gerät verbleiben musste.

Auf dem USB-Stick befanden sich aber auch einige Testobjekte, unter anderem Benchy, das kleine Boot, das häufig zum Vergleich der Druckgeschwindigkeit eingesetzt wird und von dem Creality behauptet, dass der neue Ender es in 13 Minuten drucken könne. Das probierte ich zuerst aus: Der Ender begann mit dem üblichen Leveln, also der Bestimmung der Nullhöhe. Da bei diesem Vorgang die Düsenspitze als Sensor benutzt wird, ist es wichtig, dass dort kein Filamentrest haftet, der die Messung verfälschen würde. Daher führte der Ender zunächst eine aufwendige Düsenreinigung mit mehrfachem Aufheizen und Abkühlen sowie Abwischen der Düsenspitze auf dem Druckbett durch. Erst nach circa drei Minuten startete der eigentliche Druck. Das Tempo war schon rein optisch durch die heftigen Bewegungen von Kopf und Bett beeindruckend. Nach 14 Minuten war Benchy fertig.

Bild 4: Der Makey war voll beweglich, die Oberfläche einwandfrei.

Als Nächstes folgte der Test auf Maßhaltigkeit und

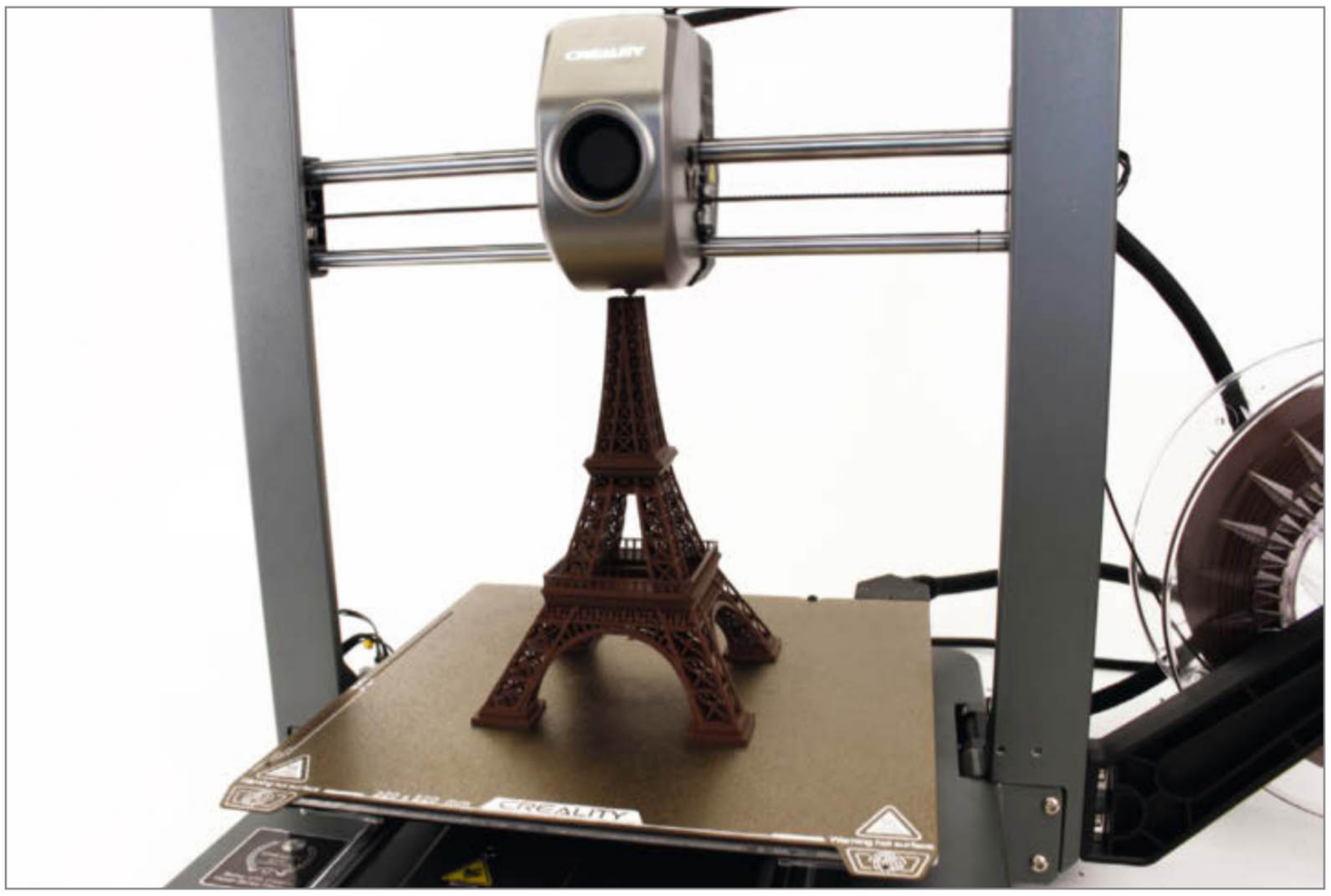


Bild 5: Der Eiffelturm ohne Stützstrukturen gedruckt, brauchte nur etwas mehr als 4 Stunden.

das Verhalten bei ungestützten Überhängen (Bild 3). Das Ergebnis war beeindruckend. Insbesondere die Überhänge konnten sich sehen lassen. Das lag vor allem an der Möglichkeit, im Slicer eine intelligente Lüftersteuerung zu aktivieren. Die achtet auf Überhänge und Brücken und kühlt dann das aus der Düse ausgetretene Filament zusätzlich durch den zweiten, an der Rückseite des Druckkopfes angebrachten Lüfter. Das Filament erstarrt so sehr schnell und hat keine Zeit, sich der Schwerkraft folgend mangels Stütze nach unten zu bewegen.

Beim Druck konnte man den Ender mithilfe der Web-Oberfläche Fluidd überwachen (dazu IP-Adresse und Portnummer 4408 im Browser eingeben).

Zum Check der Genauigkeit und der Oberflächen diente ein Makey-Modell, das ohne zusätzliche Stützen in einem Teil gedruckt wurde, hinterher aber voll bewegliche Arme, Beine und einen drehbaren Kopf haben sollte. Das geht nur, wenn der Drucker sehr genau druckt und die winzigen Zwischenräume im Makey nicht mit Filament verklebt. Diesen Test bestand der Ender problemlos (Bild 4).

Schließlich gab es noch eine große Aufgabe: den Druck eines 24 cm hohen Eiffelturms ohne Stützen. Diesen hatte ich früher auch schon unter anderem einem Anet A8 und meinem alten Ender-3 zugemutet. Sie brauchten dafür jeweils etwas mehr als 15 Stunden (bei 0,2 mm Schichthöhe). Der neue Ender schaffte es in etwas mehr als 4 Stunden und das mit durchaus brauchbarer Qualität. Lediglich die nahezu waagerechten, frei schwebenden Teile an den Bögen im unteren Teil des Turms hatten kleine Durchhänger (Bild 5). Das Gitterwerk druckte er sehr gut und stabil. Trotz der heftigen Drucktischbewegungen und der Höhe des Bauwerks kam es auch nicht zu Problemen mit der Bodenhaftung, obwohl der Turm ohne Brim nur auf seinen relativ kleinen Füßen haftete. Das Druckbett, eine PEI-beschichtete Federstahlplatte, die magnetisch am Drucktisch hält, erwies sich als ausgesprochen praktisch. Nach dem Druck konnten die Objekte durch geringes Biegen problemlos abgenommen werden. In Verbindung mit der genauen Kalibrierung des Hotends haftete jedes Druckobjekt problemlos.

Auf Wunsch auch leise

Beim Druck war das Gerät recht laut, insbesondere schnelle Bewegungen hörten sich an wie ein kreischendes Tier. Das liegt an den hohen Frequenzen, mit denen die Schrittmotoren angesteuert werden. Im Einstellmenü kann man das aber bei laufendem Druck ändern, indem man die Feed-Einstellung auf lautlos stellt. Das ist zwar nicht wirklich lautlos, aber deutlich ruhiger. Die Druckgeschwindigkeit fällt aber auch entsprechend ab.

Als Filamentarten habe ich PLA, ePLA und PETG eingesetzt. Das ePLA, dass ich als Billigfilament online gekauft hatte, erwies sich auf dem alten Ender als problematisch, denn es zog Fäden, die jede Spinne neidisch machen konnten. Der neue Ender kam damit aber problemlos zurecht. Standard-PLA und PETG waren ebenfalls völlig problemfrei. Allerdings muss man im Slicer darauf achten, dass die Vorgabeeinstellungen nach der Installation nicht optimal sind. Bei anderen Layer-Höhen als 0,2 mm wird nämlich auch der erste Layer bereits mit vollem Tempo (bei 0,1 mm immerhin mit 600 mm/s) gedruckt. Folge: Das Filament haftet schlecht auf dem Druckbett. Reduziert man das Tempo des Initial-Layers auf max. 200 mm/s, ist dieses Problem behoben.

Zum Schluss noch eins: Der Ender-3 kann auch mit einer Kamera erweitert werden. Allerdings akzeptiert er nur das Modell Nebula von Creality. Damit soll auch eine KI-gesteuerte Überwachung des Druckvorgangs möglich sein, die beim Auftreten von Fehlern den Druckvorgang abbricht. Im Test habe ich das aber nicht berücksichtigt. Fehler traten auch so nicht auf.

Fazit

Der Creality Ender-3 V3 ist ein hervorragender Drucker, der bei hoher Geschwindigkeit sehr gute Druckergebnisse liefert. Und das bei einem Preis, der anderen Druckerherstellern wehtun wird. Die Geschwindigkeit wird zwar mit einer deutlich höheren Geräuschentwicklung bezahlt, da der Drucker aber nicht mehr über Nacht arbeiten muss, ist dieser Nachteil leicht hinzunehmen. *—hgb*

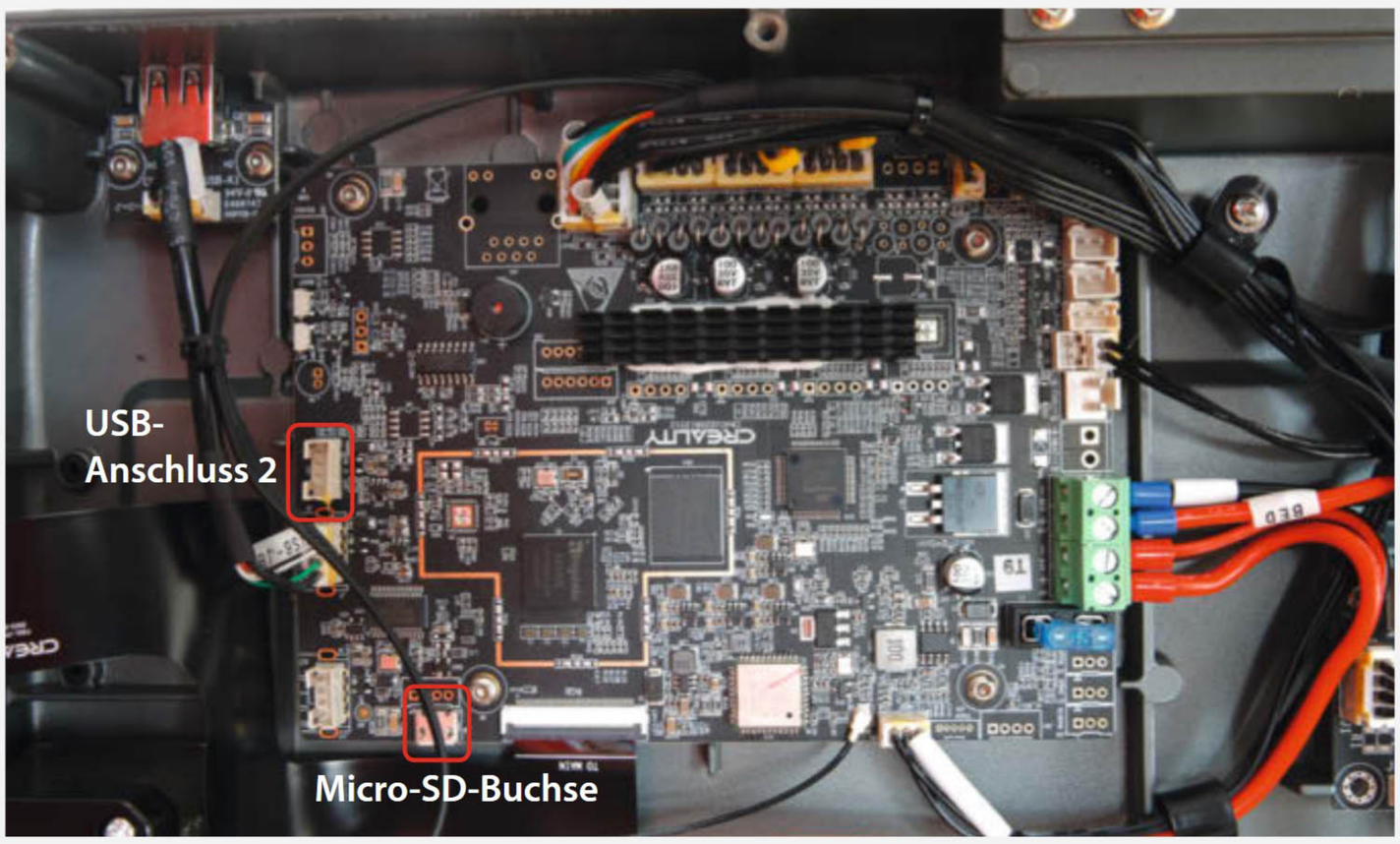
Rooten des Druckers

Im Werkzeugmenü des Druckers gibt es den Punkt „Root-Kontoinformation“. Damit kann man den Root-Zugriff per SSH auf das Betriebssystem des Ender-3 aktivieren. Mit geeigneter Software (zum Beispiel Putty) lässt sich so das Linux des Druckers konfigurieren. Der Benutzername lautet „root“, das Passwort „creality_ender3v3“.

Doch Vorsicht: Falls Sie dabei einen Fehler machen, kann das dazu führen, dass der Drucker nicht mehr startet. Sie sollten also genau wissen, was Sie machen!

Allerdings bietet das vielleicht auch die Möglichkeit, den Ender zur Zusammenarbeit mit Kameras anderer Hersteller zu bewegen. Das Linux bemerkt jedenfalls auch die Verbindung mit anderen Kameras und legt dafür auch ein eigenes Video-Device an. Die Drucker-Firmware benutzt dieses jedoch nicht. Falls Sie eine Lösung dafür finden, teilen Sie uns diese bitte mit.

Auf dem Mainboard im Inneren des Druckers sitzen übrigens noch zwei ungenutzte USB-Schnittstellen. Eine ist wie die nach außen geführte ebenfalls auf eine Stiftleiste geführt und kann genauso genutzt werden, wenn man an die Leiste eine USB-Buchse anschließen würde. Die zweite ist eine Mikro-USB-Buchse. Deren Funktion ist bislang nicht bekannt. Vermutlich dient sie zum Flashen des Betriebssystems aufs Board.



Auf dem Mainboard gibt es ungenutzte USB-Schnittstellen.

Neuer Input für Maker

Make Elektronik Special

Make Elektronik Special bietet einen einfachen und praxisorientierten Einstieg in Transistorschaltungen, die Maker in eigenen Projekten einsetzen können. Das mitgelieferte Experimentierset inkl. Breadboard, Kabeln und 45 Elektronikbauteilen enthält alles, um die gezeigten Schaltungen sofort nachbauen und testen zu können.

Heft + Experimentierset für 44,95 €

shop.heise.de/make-elektronik21

Inklusive Experimentierset und Breadboard

Make Operationsverstärker Special

Das Make-Sonderheft bietet einen praxisorientierten Einstieg in Schaltungen mit Operationsverstärkern inkl. Experimentierset. Will man Sensorsignale verarbeiten oder verstärken, Spannungen überwachen oder Audiosignale filtern: Mit geringem Aufwand und ohne komplizierte Berechnungen setzt man Operationsverstärker ein. Das Heft erklärt, wie alle Schaltungen funktionieren.

Heft + Experimentierset für nur 49,95 €

shop.heise.de/make-opv

Inklusive Experimentierset

Make PI Pico Special

Mit dem Make Special Pi Pico steigen Sie ein in die Welt der Programmierung von ARM-Mikrocontrollern. Make zeigt in dem 64-seitigen Special, welche Entwicklungsumgebungen es für den Rasperry Pi Pico gibt, wie man sie installiert und wie man sie nutzt.

Heft + Raspberry Pi Pico für 24,95 €

shop.heise.de/make-pico

Inkl. Raspberry Pi Pico RP2040

Maker Faire: Jahrmarkt der Erstaunlichkeiten

Final Countdown zum Maker-Faire-Heimspiel

Die Redaktion läuft sich warm für die Maker Faire Hannover 2024. In Anbetracht des zehnten Jubiläums erinnern wir uns an vergangene Maker-Faire-Happenings. Währenddessen ist der Call for Makers natürlich noch weiterhin offen.

von Daniel Schwabe

Die Maker Faire Hannover vom 17. bis 18. August 2024 zeigt sich schon am Horizont. Die Projekte für den Messestand werden bereits vorbereitet und warten auf ihren großen Augenblick auf dem Showfloor. Währenddessen trainiert die Redaktion fleißig das Austeilen von Visitenkarten, Anwerben von neuen Autoren und die Beantwortung von Leserfragen.

Der Ruf nach Makern hallt noch durch die Hallen

Wer auf der Maker Faire seine spannenden Projekte präsentieren möchte, kann noch bis zum 9. Juli 2024 dem Call for Maker folgen und sich als Aussteller anmelden. Für Privatpersonen, offene Werkstätten, Schulen, Hochschulen und Universitäten, Makerspaces, Fablabs sowie gemeinnützige Vereine und Einrichtungen ist das Ganze sogar kostenlos. Alle Infos zum Call for Makers sind auf der offiziellen Maker-Faire-Website zu finden.

Neue Maker Faire in Nordrhein-Westfalen

Am 07. Juni findet von 10 bis 18 Uhr in der Zentralbibliothek Wuppertal die erste Maker Faire in Wuppertal statt.

Die Maker Faire hinter den Kulissen

Dieses Jahr findet die zehnte Maker Faire Hannover statt. In diesen ganzen Jahren seit 2013 hat die Redaktion hinter den Kulissen natürlich schon alle möglichen Maker-Faire-Abenteuer erlebt. Zum Jubiläum plaudert die Redaktion aus dem Werkzeugköfferchen und teilt einige Geschichten, die besonders im Gedächtnis geblieben sind. Auch die diesjährige Maker Faire wird bestimmt wieder erinnerungswürdig! —*das*

Suffering from Success

Die erste Maker Faire Hannover haben wir nur an einem Tag (Samstag) veranstaltet und mit 1500 Besuchern gerechnet. Dazu sollte aus unserer Sicht die wegen ihrer Glasfront auch Glashalle genannte Halle im Hannover Congress Centrum und der angeschlossene Park ausreichen. Es strömten allerdings am Ende 4500 Besucher auf die Veranstaltung, was zunächst dank der Außenflächen und gutem Wetter noch ohne Probleme ablief. Als es jedoch stark zu regnen anfing, drängten alle Besucher in die Halle, sodass es mehr als eng wurde und der Messe-Hausherr kurz davor war, die Halle zu sperren. Glücklicherweise ließ der Regen schnell nach, die Sonne kam wieder und die Situation entspannte sich.

Was sich nicht entspannte, war anschließend das tropische Klima in der Halle. Die war wegen der Glasfront nämlich ursprünglich mal als Ausstellungsort für sonnenhungrige Pflanzen eingesetzt worden. Zusammen mit der hohen Luftfeuchte nach dem Regen und der Sonne sowie mehreren tausend Menschen ergab das eine schwitzige Mischung. Im nächsten Jahr wiederholte sich die Regen-Sonne-Menschen-Mischung abermals und erste Aussteller drohten, nicht noch mal zur Maker Faire zu kommen, wenn sie wieder in der Glashalle stehen müssten. Daraufhin haben wir die Glashalle nur noch für die abgedunkelte Dark Gallery und andere Zwecke genutzt. —*dab*

Wenn Maker einem einen Papageien aufbinden

Eine Maker Faire, bei der öfter mal ein echter Zeppelin über das Messegelände flog, auf der die Dichte an Steampunks und Cosplayern so hoch war wie sonst vielleicht bei der Comic Con, und bei der sich direkt nebenan die Funkamateure trafen und eine ganze Halle mit ihrem Radiofloh-markt füllten – diese spezielle Mischung zeichnete die Maker Faires am Bodensee aus, die 2016 bis 2018 in Friedrichshafen stattfanden.

Da die Messehallen dort etwas außerhalb liegen und seinerzeit am Wochenende mit dem Bus nur sporadisch erreichbar waren, nahmen viele ein Taxi zur Faire – und stürzten manchen Fahrer in Verwirrung: „Sie sind heute der erste normale Mensch, der in meinen Wagen steigt. Ich habe gerade einen Fuchs dahin gefahren! Was ist das eigentlich für eine komische Veranstaltung?“

Keine Panik, es ist nur eine Maker Faire. Aber eben doch eine spezielle, denn der Fuchs war nicht alleine: Es wimmelte von Orks, Computerspielfiguren, Film-Unholdinnen, Steampunks (wobei einige präzisierten, sie seien in Wahrheit Cosplayerinnen, die Steampunks darstellten), diversen Fellwesen, die nur echte Kenner kennen, und sogar versprengten Piraten. Ein wenig wie die konnten wir uns 2017 am Make-Stand fühlen, denn jemand hatte einen ausgewachsenen Papagei mit eingeschmuggelt, der reihum auf den Schultern von allen probesitzen durfte – und ja, ich muss zugeben: Ich habe ihn zuerst für eine perfekt gemachte animatronische Figur gehalten. Dabei war das Jahre vor der Posteule aus Make 1/22 … —*pek*

Bring mich zum Horizont! Käpt'n Olli Hofgärtner vom heise shop auf der Brücke des Maker-Stands.

Man weiß nie, was hinter der nächsten Ecke wartet: Ork-Patrouille auf der Maker Faire Bodensee 2017

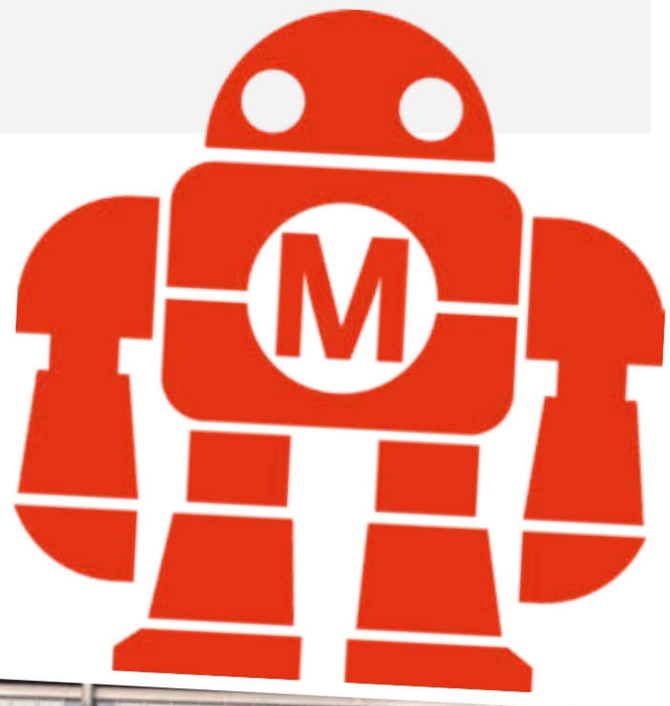
Highscore-Jagd mit Eben Upton

Auf der Maker Faire Hannover 2023 kam Eben Upton, CEO der Raspberry Pi (Trading) Ltd., an unseren Stand. Er hatte kurz zuvor meinem Kollegen Ákos ein Interview gegeben. Im Gespräch über Boards, Pi und Pico sah er unseren MMBasic-Retro-Computer aus der Make 4/23, natürlich mit einem Raspberry Pico. Auf dem VGA-Monitor lief eine Tetris-Variante, programmiert in MMBasic.

Wir haben kurz über den Pico geredet, die PIOs, die die VGA-Ausgabe stemmen, und während Eben noch über PIO, Assembler und Python dozierte, machte er ein Tetris nach dem anderen. Ich hatte schon viele Besucher spielen sehen, aber keiner hat derart schnell viele Punkte erreicht. Und schon gar nicht dabei über komplizierte Themen gesprochen.

Vier Wochen später kam der neue Raspi 5 und wir hatten keine Ahnung und keine Anspielung dazu von Eben Upton im Interview erhalten. —*caw*

Auch am Make-Stand geht es an die Spitze.



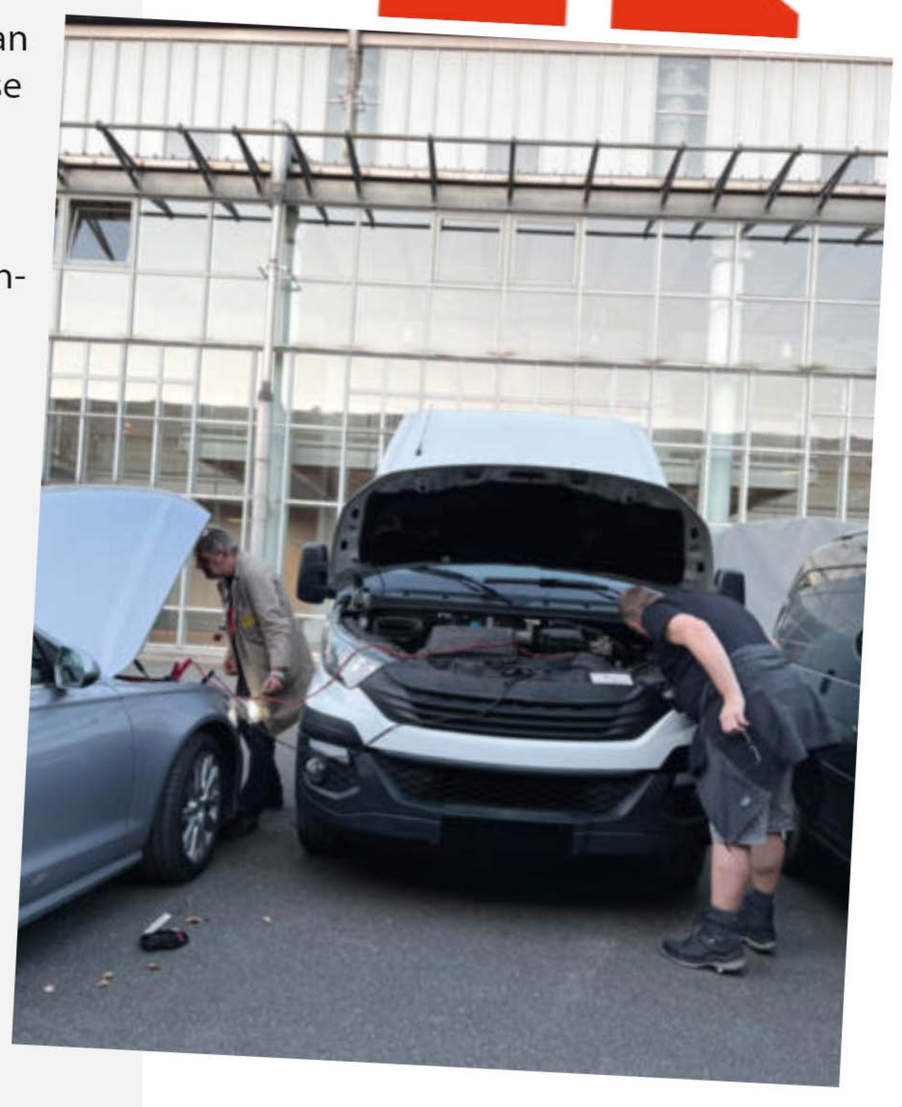
Der Letzte macht das Licht aus

Als Make-Redaktion sind wir auf deutschen Maker Faires in der Regel mit einem Stand vertreten, zeigen euch spannende Projekte aus dem Heft und genießen den Austausch mit Lesern, Autoren und Besuchern. Für den Transport des Standes nutzen wir einen mittelgroßen Sprinter, der nach dem Ausladen geduldig auf dem Parkplatz darauf wartet, dass wir zurückkehren. Aber bei all den Dingen, an die wir logistisch denken müssen, kann schon mal was auf der Strecke bleiben. Doof ist nur, wenn man es selbst ist.

So saß ich also mit Daniel – wir waren beide geschafft von der Maker Faire – im wieder voll beladenen Sprinter und wir wollten uns gerade auf den Weg zum Verlag machen, um alles mit den anderen Kollegen auszuladen. Doch als Daniel den Wagen starten wollte, passierte nichts. Der Grund: Irgendjemand hatte das Licht hinten im Sprinter angelassen, die Tür geschlossen und so ist es niemandem aufgefallen.

Anscheinend konnte es sich auch nicht von selbst ausschalten, wie man es etwa aus seinem smarten Zuhause kennt. Vielleicht hatte jemand auch die Automatik deaktiviert.

Jedenfalls war die Batterie des Sprinters über die zwei Veranstaltungstage komplett entladen und wir brauchten Hilfe. Es war nur keiner mehr da, denn alle hatten sich bereits auf den Weg zum Verlag gemacht. Zum Glück erreichten wir schließlich unseren Kollegen Carsten, der mit PKW und Kabelage zurückkam, um uns zu retten – es lebe die Freisprechanlage. Jetzt mussten wir nur noch eine Anleitung im Internet dafür finden, wie man die Kabel unter der Motorhaube des Sprinters anschließt und konnten schließlich das Fahrzeug wieder erfolgreich in Betrieb nehmen – eigentlich ein bisschen wie immer. —*akf*



Eine Klemme hier, ein paar Funken dort: Maker wissen sich zu helfen!

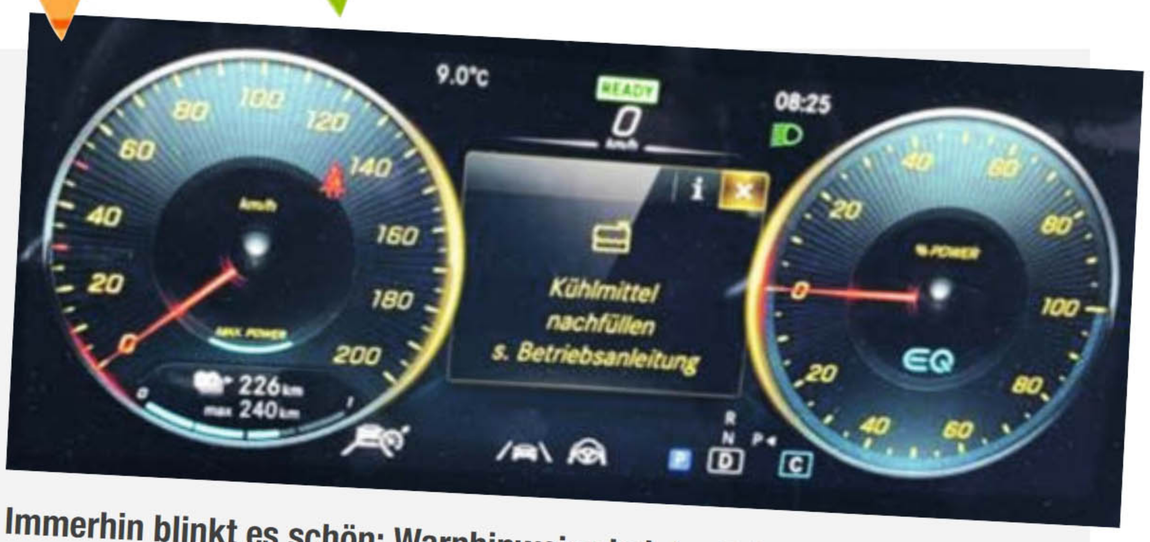
Alle Hände voll zu tun mit Katzenstreu

Die Hand of Man überblickt die Maker Faire

Juni 2017, die dritte Maker Faire Berlin: Am Eingang zur Station am Gleisdreieck wartet direkt das Highlight auf die Besucherinnen und Besucher: Die tonnenschwere und haushohe „Hand of Man“ des Robotik-Künstlers Christian Ristow, die sich überraschenderweise kinderleicht bewegen lässt – man muss lediglich auf dem „Pilotensitz“ Platz nehmen und die eigene Hand in eine Art beweglichen Metallhandschuh stecken, um die Steuerung für die gigantischen Finger viele Meter weiter oben zu übernehmen.

Am Samstagabend dann der Schreck: Die Hand hat ein Leck und verliert Öl in nicht zu ignorierender Menge! Eine der Hauptattraktionen des Festivals nach der halben Zeit stillzulegen kommt aber nicht in Frage. Der Kollege im Team mit dem Feuerwehrhintergrund weiß glücklicherweise ein probates Mittel dagegen: Katzenstreu, in rauen Mengen … und so verbringt er den Samstagabend damit, in Berlin die Zoohandlungen und Baumärkte abzuklappern, um aus „haushaltsüblichen“ Abgabemengen genügend geballte Saugkraft zusammenzuklauben, um auch die gigantischste Hand trockenzulegen. Der Einsatz hat sich gelohnt, die „Hand of Man“ konnte bis Sonntagabend weiter ihre Monsterfinger durch die Berliner Luft wirbeln lassen. —*pek*

Maker Faire Wien 2018



Immerhin blinkt es schön: Warnhinweise bei der Fahrt.

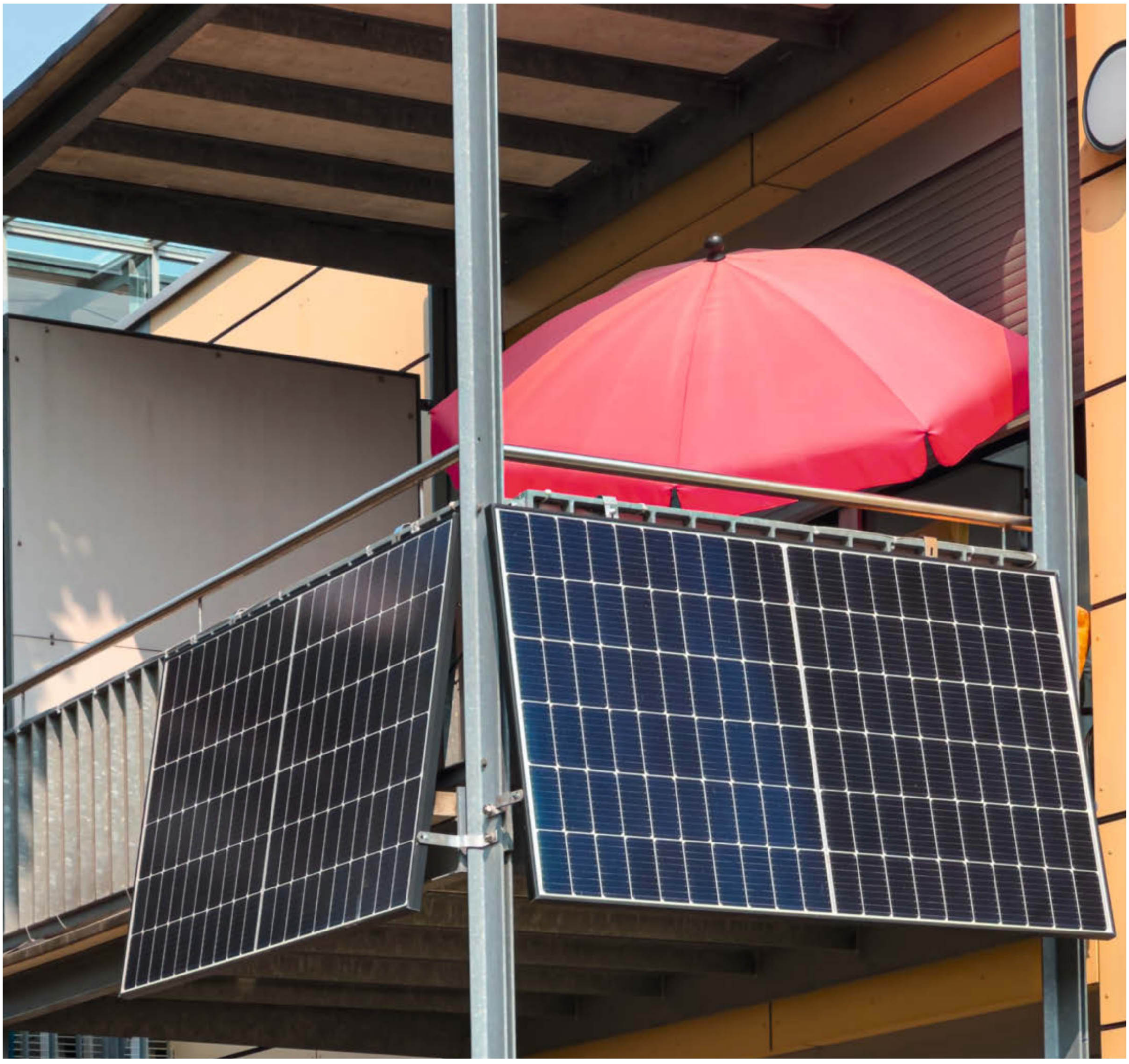
Hannover, 6 Uhr Freitag morgens: Mein Kollege Florian und ich steigen in den Mercedes-Miettransporter, etwa eine Tonne Ausrüstung an Board, bestes heißes Wetter. Wenig später, A7 bei Kassel: Auf dem Display des Transporters erscheint: „Kühlwasser überprüfen!“ Nächster Rastplatz: Behälter nahezu leer, also aufgefüllt. Göttingen, kaum eine Stunde später: „Kühlwasser überprüfen!“ Behälter wieder fast leer …!?! Anruf bei Autovermietung Hoffmann, Problem geschildert, Lösungsvorschlag: „Wir schicken einen Fahrer mit einem neuen Fahrzeug. Entladen Sie schon mal auf dem Parkplatz.“ Erst auf den Einwand, dass wir unsere Sachen auf Paletten haben und keine Ameise oder Gabelstapler zur Verfügung steht, folgt: „Wir rufen zurück.“ Kurze Zeit später: „Fahren Sie nach Würzburg in die Mercedes-Werkstatt. Dort soll das Fahrzeug überprüft werden.“

Mercedes-Werkstatt um kurz vor 12: „Bitte warten Sie noch einen Moment, wir haben im Augenblick keinen Arbeitsplatz frei.“ Kaum 2,5 Stunden später beschäftigt sich (endlich) jemand mit dem Wagen. Nach einer halben Stunde die Diagnose: „Das können wir so nicht feststellen. Das ist ja alles so eng im Motorraum, da müssten wir ja zunächst einiges ausbauen, um das Leck zu finden.“ Lösungsvorschlag: „Fahren Sie weiter. Wir geben Ihnen einen Kanister mit Wasser mit.“ Weiterfahrt um ca. 15:30 nach Wien. Alle 100 km Wasser nachfüllen, einmal tanken. Wasserverbrauch ist gefühlt höher als der Dieselverbrauch. Kurz vor Mitternacht Ankunft in Wien. Montagmorgen, Rückfahrt nach Hannover: Dieseltank, Kühlwasserbehälter und Kanister sind voll. Übliches Spiel: Alle 100 bis 150 km Wasser nachfüllen. Am späten Nachmittag, A7, Kasseler Berge. Der Mercedes hat inzwischen sämtliche Fehlermeldungen zum Besten gegeben, die in seinem Motorkontrollgerät gespeichert sein dürften. Bergauf wirds kritisch. Pling, der Alarm-Gong, Fehlermeldungen interessieren uns jetzt nicht mehr. Tempo geht runter. Beladene 30-Tonner überholen uns bergauf mühelos… Am Abend endlich in Hannover… Transporter läuft (noch)… —*hgb*

Dem Solarinverter auf den Zahn gefühlt

Strom von der Sonne ist eine prima Sache. Aber will man, dass die Balkonanlage Daten ins Internet sendet oder gar von dort gesteuert und umprogrammiert werden kann? Sicher nicht! Hier gibt es eine Lösung für solch unsichere Anlagen, für die nur ein ESP-Board benötigt wird.

von Roland Marx



Solarinverter (Wechselrichter) sind mittlerweile weit verbreitet. Nachdem sie für private Haushalte sogar von der Umsatzsteuer befreit wurden, bekommt man sie für deutlich unter 200 Euro. Die Leistungsgrenzen liegen meist zwischen 600 und 800 W und sie haben zwei **M**aximum-**P**ower**p**oint-**T**racker (MPPT).

Die Digitalisierung verschlafen haben die Entwickler der meist aus China stammenden Geräte sicherlich nicht. Die Geräte kommen daher entweder mit einem proprietären Protokoll und Funkmodul oder sogar direkt mit integriertem WLAN-Modul. Damit kann man dann die Daten des Inverters leicht per App einsehen und seiner Investition beim Amortisieren zusehen. Doch auch unabhängig von der finanziellen Ausbeute ist es spannend, zuzuschauen, wie da in den großen Platten geräuschlos aus Sonnenlicht auf einmal brauchbarer Wechselstrom produziert wird.

Doch so praktisch das mit App, lokaler Webseite und Cloud auch ist, ich habe schon vor Jahren Andeutungen gelesen, dass die Hersteller es mit dem Datenschutz wohl nicht so ernst nehmen. Leider habe ich solche Hinweise stets ohne konkrete Informationen und Beweise gefunden.

Da ich mich beruflich mit der Security von IoT und Embedded-Systemen befasse, hat mich das nicht losgelassen und das Projekt Solarhacking wurde gestartet. Konkret ging es bei mir um den Mikrowechselrichter Deye SUN600G3, der bei meinem Solarbalkonset dabei war. Er gehört zu den WLAN-fähigen Geräten. Die App und das Cloudportal habe ich von Anfang an nicht verwendet. Im Rahmen meiner Analysen habe ich zumindest das Cloudportal ausprobiert und kann dadurch mehr zur Funktionsweise erzählen.

Kurzinfo

- **ESP-Board als Sicherheitssperre zwischen Inverter und Smart-Home-WLAN**
- **Auslesen von Daten aus der Inverter-Webpage**
- **Display als Infozentrale für die Energieerzeugung**

Checkliste

Zeitaufwand:
1 Stunde

Kosten:
ab ca. 20 Euro

Mehr zum Thema

- Roland Marx: Solar-Inverter-Sicherheit auf dem Prüfstand, Online-Artikel auf heise+
- Uwe Rohne: Photovoltaik an der E-Auto-Wallbox, Make 5/21, S. 20
- Guido Burger, Klaus-Uwe Gollmer: Der Solartisch und die grüne Steckdose, Make 4/22, S. 10

Alles zum Artikel im Web unter
make-magazin.de/xh9f

Material

- **eines der folgenden Boards:**
- Wemos Lolin S2 Pico ESP32-S2 (128 × 32)
- Wemos Lolin32 ESP32 OLED (128 × 64)
- NodeMCU ESP8266 OLED HW-630 (128 × 64)
- Heltec ESP8266 NodeMCU (128 × 32)
- Heltec Wifi Kit 32 v3 ESP32 (128 × 64)

Werkzeug

- **3D-Drucker für optionales Gehäuse**
- **Software** Arduino IDE getestet mit Version 2.3.2

Folge den Daten

Wie kommen die Daten eigentlich vom Inverter auf das Handy? Bei sehr vielen IoT-Produkten sind die Bits und Bytes nicht besonders faul. Ganz im Gegenteil: Auch wenn der Inverter und das Handy nur wenige Meter voneinander entfernt sind, reisen die Daten auf dem Weg von dem einen zum anderen oft erst um den halben Globus herum zu einem Server in Werweißwo und wieder zurück (Bild 1)!

Sie werden vom IoT-Gerät an die Hersteller-Cloud geschickt und dort gespeichert. Die App oder das Web-Dashboard ruft sie dann von dort ab und zeigt sie an. Gefragt, ob das okay ist, wird man selten. Bei meinem Inverter wurden die Daten 2022 noch in die USA geschickt. Mittlerweile steht der Server in Frankfurt bei AWS, was aus Datenschutzgründen auch dringend angeraten ist.

Usability vs. Privacy?

Doch warum sind die Datenwege so komplex? Vermutlich erhofft sich der Hersteller, aus den Daten Informationen auszulesen, und auch Serviceangebote spielen sicher eine Rolle. Der trivialste Grund lautet aber Usability, also die Einfachheit bei der Installation – für jeden Skill-Level.

Dazu muss der Nutzer dem Gerät nur den Zugang zum WLAN geben. Die App kann er leicht installieren und auch einen Account in einem Webportal anzulegen, stellt heute kein Problem mehr dar. Einzig das Verknüpfen des

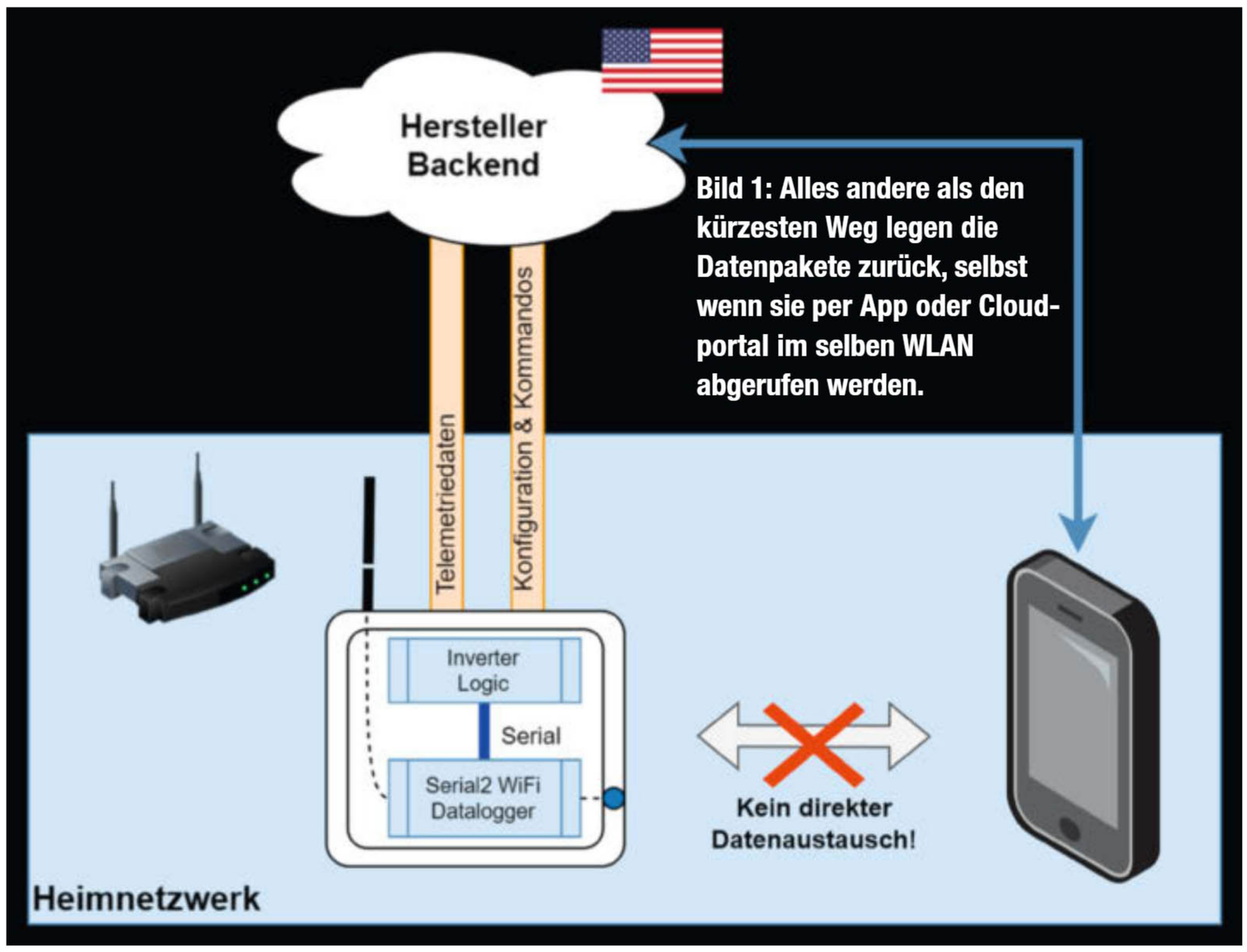


Bild 1: Alles andere als den kürzesten Weg legen die Datenpakete zurück, selbst wenn sie per App oder Cloudportal im selben WLAN abgerufen werden.

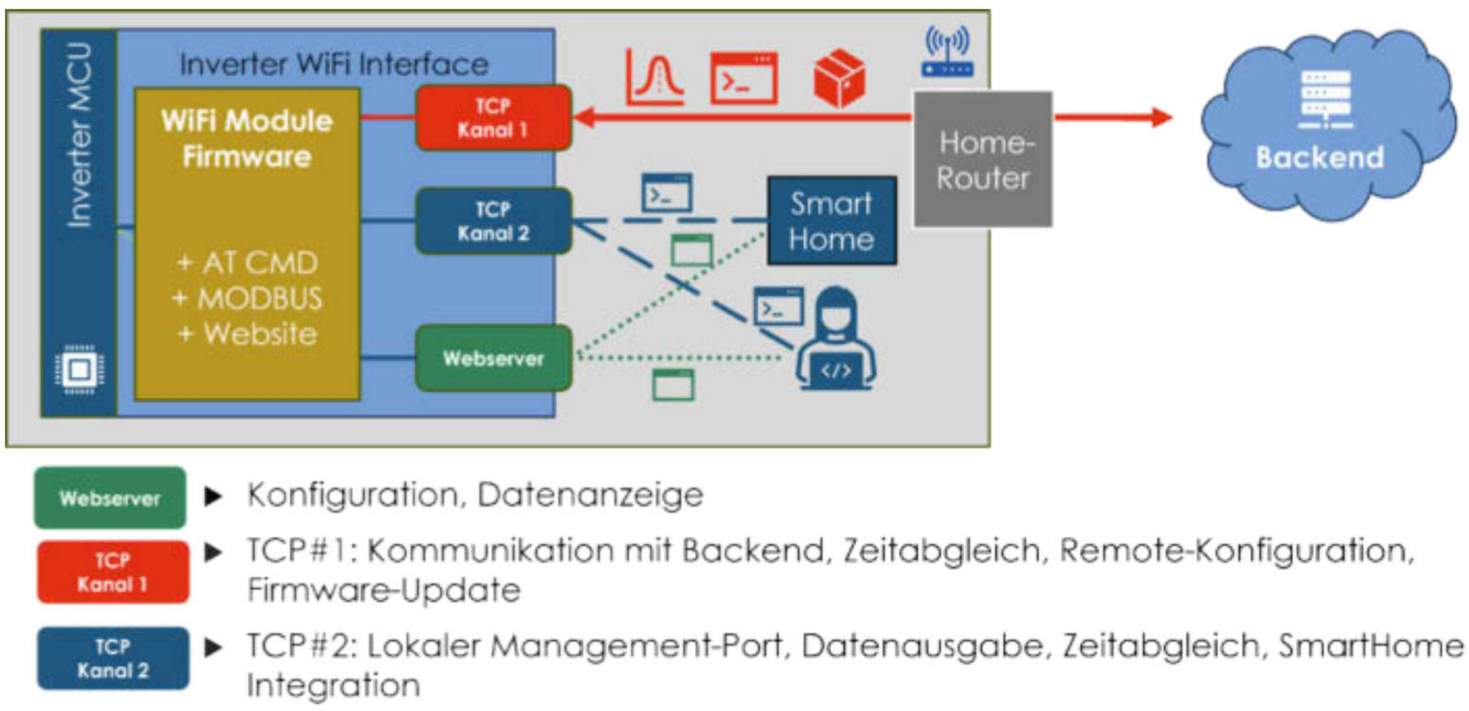


Bild 2: Inverter-Komponentenansicht: Drei Services bietet das integrierte WLAN-Modul.

Geräts mit dem Account anhand der Seriennummer ist zu meistern. Was im Einzelnen danach an Daten übertragen wird, erfährt man kaum. Im Online-Artikel (siehe Kurzinfo-Link) habe ich beschrieben, wie man das mit einfachen Mitteln herausfinden kann.

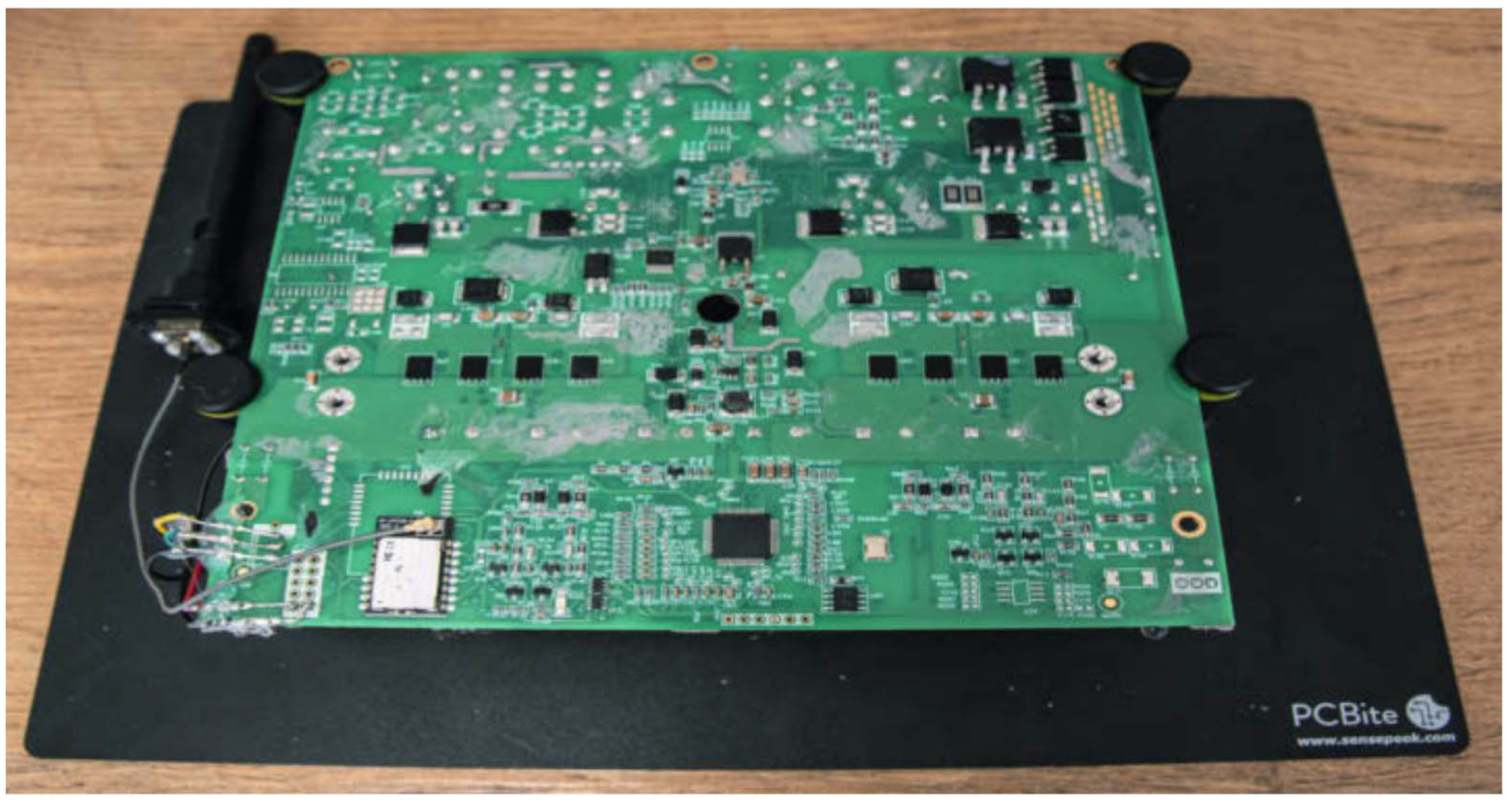
Möchte man die Daten nur lokal verarbeiten und direkt auf dem Handy im Heimnetz anzeigen, müsste man zunächst die IP-Adresse des Inverters herausfinden und in die App eintragen. Noch besser wäre, wenn man es dann noch schafft, die IP statisch zu konfigurieren. So ein Setup ist für den versierten Make-Leser wohl keine Hürde, aber wie viele Personen in eurem Bekanntenkreis kennt ihr, die das wohl hinbekommen, ohne euch zu fragen?

Und selbst wenn das geklappt hat – auf die Daten kann man so nur zugreifen, wenn man zu Hause ist. Zum Posen auf der Party, was man heute wieder aus seinen 4-m²-Zellen erwirtschaftet hat, benötigt man noch eine VPN-Verbindung oder ein extern erreichbares Smart-Home-System.

Wer macht hier eigentlich was?

Im Rahmen der Mitteilung von Security-Verbesserungsvorschlägen an den Hersteller habe ich mir die Frage gestellt, wer da jetzt eigentlich für was die Verantwortung trägt, denn wir haben nicht direkt mit dem Solarinverter-Hersteller Deye Kontakt aufgenommen, sondern durch einen Zufall mit dem Backend-Betreiber. Der nahm die Erkenntnisse ernst und bemühte sich, Updates herauszubringen. Kleiner Spoiler: Die intern verbaute Hardware ist selbst Teil des Problems und nicht alles ist mit Softwareupdates zu fixen.

Vom Dienstleister, der unter anderem das Solarman-Backend betreibt, habe ich erfahren, dass er nicht nur für Deye, sondern auch für eine Vielzahl anderer Gerätehersteller die Digitalisierung übernimmt.



Roland Marx

Bild 3: Inverterboard-Rückseite freigelegt: Unten links befindet sich das WLAN-Modul von HiFlying, das die Webseite und alles andere bereitstellt. Es spricht mit dem eigentlichen Herz des Inverters, dem schwarzen Chip unten in der Mitte.

Um ihre elektrischen Geräte IoT-fähig zu machen und dadurch besser verkaufen zu können, setzen viele Hersteller auf Rundumsorglos-Digitalisierungspakete von Drittanbietern. Solche Pakete bestehen aus den Backends, den Apps für Endanwender / Verkäufer / Installateure und auch den WLAN-Modulen, die man in seine Elektronik einbauen kann. Die WLAN-Modul-Firmware muss dann noch an das eigene Produkt angepasst werden und los geht's! Im Fall von Deye SUN600 sind dies WLAN-Module des chinesischen Herstellers HiFlying, Typ HF LPT 230-0. Zu diesem Modul findet man online beim Hersteller viele Dokumentation, und so habe ich mich mit seinen Fähigkeiten auseinandersetzen können.

Im Blockschaltbild (Bild 2) sieht man links die Inverter MCU (**M**icro **C**ontrol **U**nit), die sich mit der Elektronik um das Wechselrichten kümmert. Zum Arbeiten benötigt sie viele Parameter, die je nach Land und Typ des Inverters variieren und deswegen auch nachjustiert werden müssen. Das WLAN-Modul spricht Inverter-intern mit der MCU und auf der anderen Seite per WLAN mit der Außenwelt.

Das Modul besitzt drei wichtige Kommunikationsschnittstellen. Direkte Bekanntschaft hat man in der Regel schon mit dem Webserver für die Konfiguration gemacht. TCP-Kanal 1 versucht alle fünf Minuten, Daten an das Backend zu senden und von dort Rückantworten und Befehle entgegenzunehmen. TCP2 ist eine lokale und eher passive Schnittstelle, auf die man sich per UDP verbinden kann. Hierüber kann das Gerät viel tiefgreifender als über TCP1 konfiguriert und administriert werden. Sowohl TCP1 als auch TCP2 hören indes auf einen AT-Befehlssatz, der sehr viele Funktionen anbietet. Ein sehr hilfreiches Repo, das sich dieser Schnittstelle widmet, gibt es auf Github (Adresse siehe Kurzinfo-Link). Ich habe einige Funktionen von dieser Go-basierten Lösung auf den ESP portiert.

Tiefer Einblick in die Komponenten

„Erst wenn du es öffnest, gehört es dir!" Das hatte ich mir jedenfalls noch gedacht und wollte einen Blick auf die Innereien meines Inverters werfen. Doch nachdem die rückseitige Metallplatte nur langsam mit Spreizen und viel Kraft zu öffnen war, wurde meine Neugier jäh enttäuscht, denn eine graue Wand aus einer Masse mit einer Konsistenz ähnlich wie Radiergummi verbarg alle Blicke ins Innere – Randnotiz: Das ist wohl auch der Grund, weswegen das fehlende NA-Schutzrelais nicht so schnell auffiel.

Da das Risiko beim Entfernen der Vergussmasse hoch ist, den Inverter zu beschädigen,

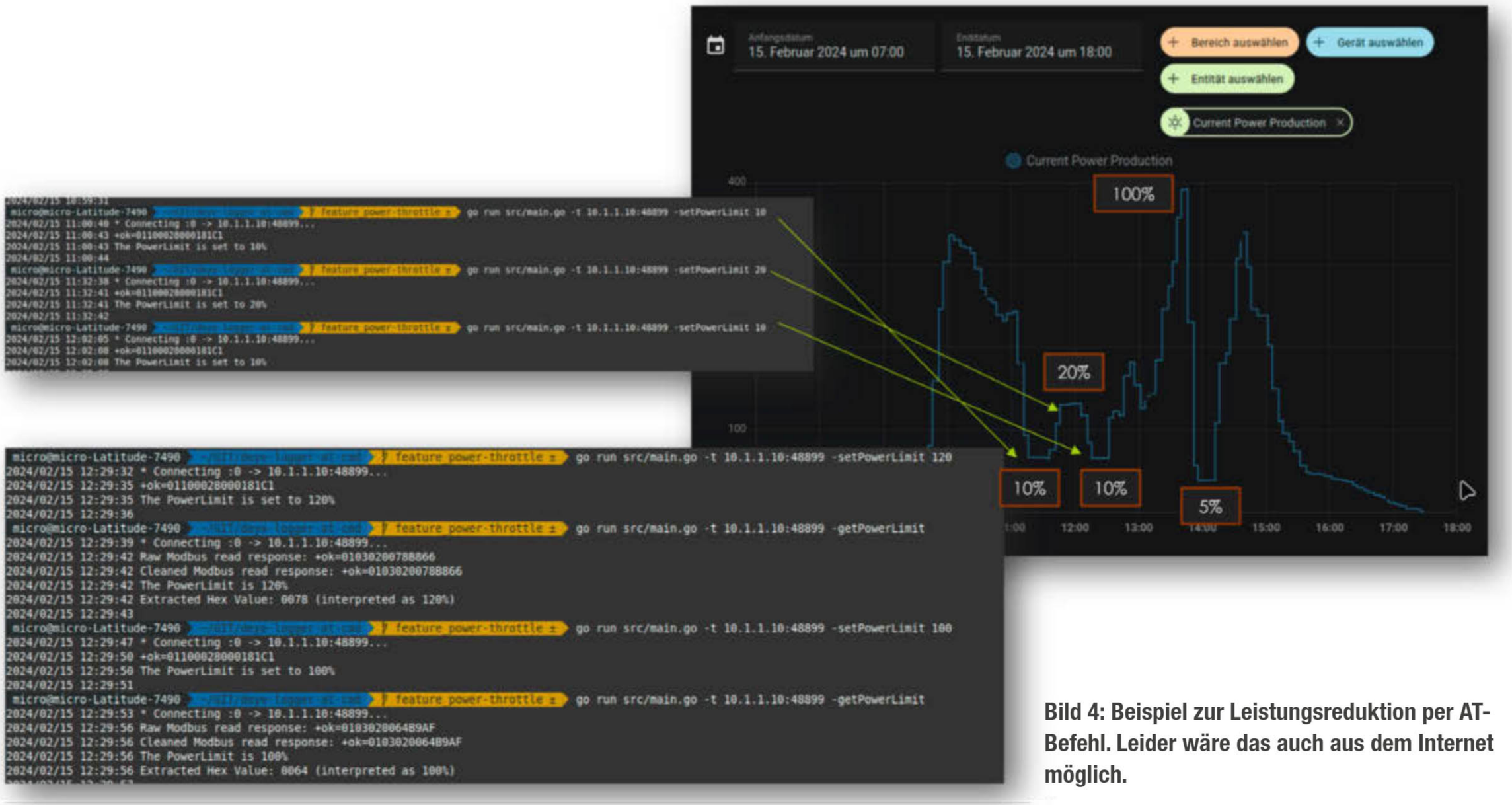


Bild 4: Beispiel zur Leistungsreduktion per AT-Befehl. Leider wäre das auch aus dem Internet möglich.

habe ich mir als Hacker anders geholfen: Ich besorgte mir günstig einen defekten Inverter und buddelte den Schatz aus (Bild 3). Das macht aber wenig Spaß und dauert selbst mit Übung etwa 30 bis 45 Minuten.

Sicherheitsanalyse in Eigenregie

Welche Daten genau ausgetauscht werden und ob das Ganze in Bezug auf Datenschutz, Security und Safety auch gut gemacht wird, hat mich schon länger interessiert. Bereits mit sehr einfachen Mitteln gelang mir ein Packet Capture und ich konnte die Kommunikation analysieren. Wie man selbst schnell herausfinden kann, was die Geräte (nicht nur Inverter oder Geräte dieses Herstellers) im eigenen Netz mit wem sprechen und wie man sich mit einer FritzBox gut absichern kann, ist in unserem ergänzenden Online-Artikel nachzulesen (Link in der Kurzinfo).

Die wichtigsten Punkte, die ich im Rahmen meiner Hobby-Analysen gefunden habe, sind:

- Die Geräte senden und empfangen alle Daten komplett ohne jegliche Verschlüsselung: Alle Daten sind im Klartext prinzipiell les- und manipulierbar. Die Server verwenden auch keine Zertifikate, um sich auszuweisen.
- Software-Updates können remote eingesteuert werden: Die Downloadserver werden nicht anhand von Zertifikaten überprüft. Eine kryptografische Signatur der Firmware gibt es nicht.
- AT-Befehle können auch aus der Ferne an die Inverter gesendet werden und haben einen (zu) weitreichenden Funktionsumfang.
- Zwei sehr kritische AT-Funktionen sind das Auslesen und Setzen von lokalen Zugangsdaten: Es ist möglich, die Zugangsdaten des Heim-WLANs remote auszulesen. Auch die eingestellten Zugangsdaten für die Webschnittstelle können gelesen und die Leistung des Inverters herauf- oder herabgesetzt werden (Bild 4). Diese Funktion macht angesichts der später erlaubten 800 W sicherlich Sinn, jedoch sollte die Möglichkeit, das zu verändern, gut geschützt sein, denn damit könnte man die Anlage praktisch abschalten oder per Überlastung beschädigen.

Risikoabschätzung: Ist das jetzt schlimm?

Diese Frage ist gar nicht so einfach zu beantworten. Für jeden Einzelnen scheint das finanzielle Risiko auf den ersten Blick klein zu sein, falls man denn mehr als 100 bis 200 Euro für Ersatz des Inverters und die zusätzlichen Stromkosten während des Anlagenausfalls verschmerzen kann. Ob das auch bei den Folgen einer Veröffentlichung des eigenen Standorts (sofern richtig angegeben) samt WLAN-Name und -Passwort in einem öffentlichen Forum so bleibt, muss jeder für sich entscheiden.

Prinzipiell kann man wohl davon ausgehen, dass der Hersteller oder auch nur der Backend-Betreiber sich seiner „Macht“ vermutlich bewusst ist. Allerdings sind in Zeiten von aufflammenden Krisenherden auch digitale Bedrohungen längst gang und gäbe. Über solch einen Hersteller und seine Geräteschar könnte man schnell einen Teil der Solarproduktion eines Landes lahmlegen oder etwa die europäischen Netze mit ungewöhnlichen Schwankungen belegen.

Bei 1000 Wechselrichtern wäre das sicher kein Grund zur Beunruhigung, jedoch wächst

Bild 5: Links zwei Tage mit, rechts ohne Internetverbindung. Ohne setzt sich der Zähler für die täglich erzeugte Energie nicht zurück.

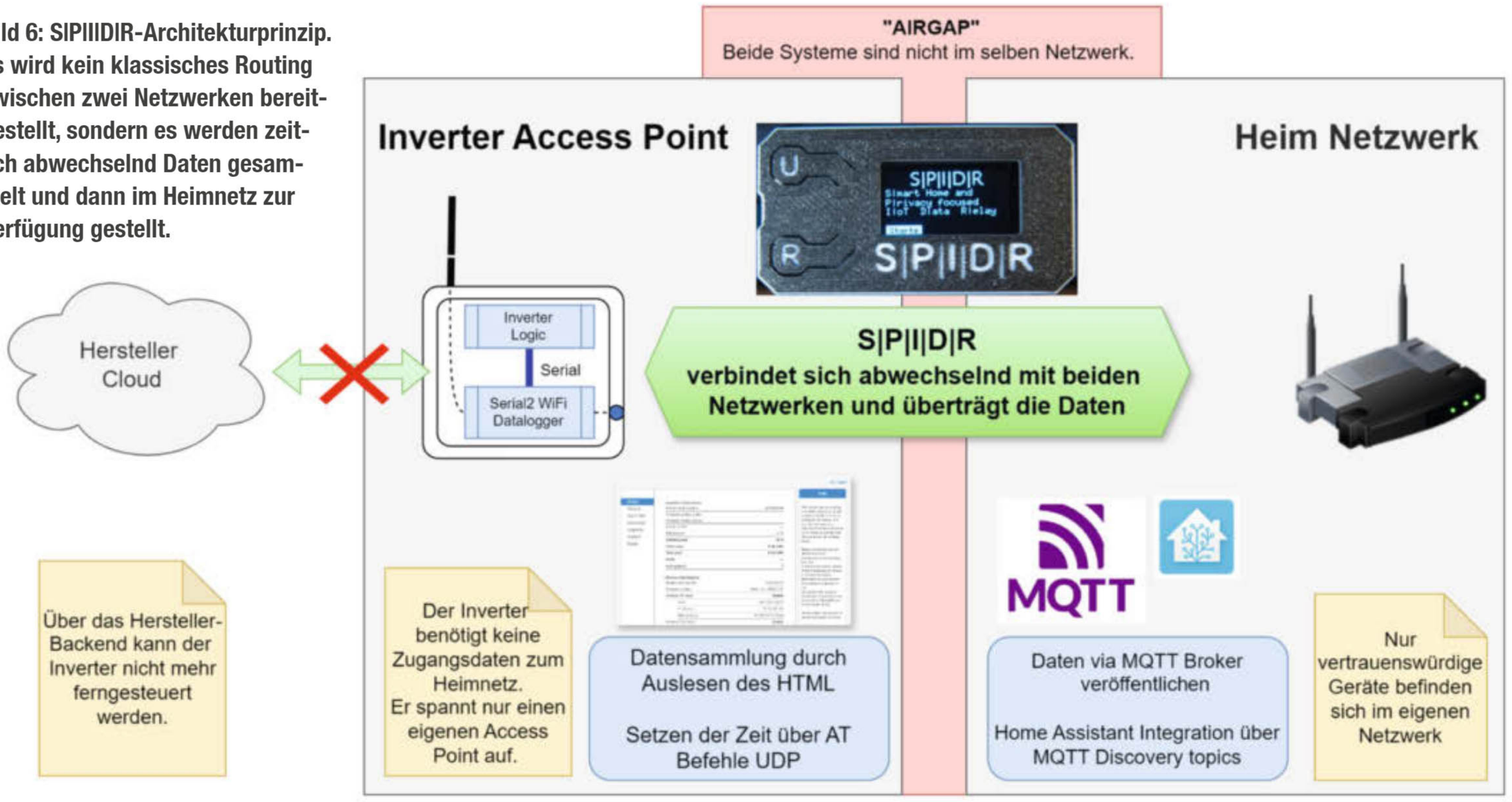


Bild 6: SIPIIIDIR-Architekturprinzip. Es wird kein klassisches Routing zwischen zwei Netzwerken bereitgestellt, sondern es werden zeitlich abwechselnd Daten gesammelt und dann im Heimnetz zur Verfügung gestellt.

die Anzahl der Wechselrichter gerade stark an und das Risiko wächst somit stetig weiter. Allein im Jahr 2023 seien laut einer auf zdf.de erschienenen Meldung mehr als eine Million neuer Balkonsolaranlagen ans Netz gegangen. Das entspricht dann mehr als 600 Megawatt Nennleistung.

Selbst wenn ein Angriff auf diese Infrastruktur nicht durch den Hersteller selbst oder einen Nationalstaat ausgeübt würde, so gibt es auch noch das große Risiko, dass durch ein zu schwach gesichertes Backend auch „ganz gewöhnliche Hacker“ ohne besondere Hilfsmittel Zugriff auf die Funktionen der Inverter und z. B. WLAN-Passwörter samt Standortdaten bekommen könnten. Dann gäbe es vielleicht keinen volkswirtschaftlichen Schaden, aber es könnte durch Missbrauch ein großer persönlicher Verlust entstehen.

Hinweis: In verantwortungsvoller Rücksprache mit dem Hersteller werden voraussichtlich einige der Probleme noch behoben, die größten – die nicht verschlüsselte Kommunikation und fehlende Zertifikate – werden mangels Hardwarefähigkeiten im WLAN-Modul jedoch nicht behebbar sein.

Schutzmaßnahmen

Was kann man tun, wenn man das Gerät nicht ersetzen möchte, es von der Hersteller-Cloud unabhängig betreiben will, aber Zugriff auf alle notwendigen Daten und Funktionen beibehalten möchte? Es gibt da viele Optionen:

Kindersicherung: Viele Router haben die Möglichkeit, Geräten beschränkte Internetrechte oder -zeit zu geben. So ist es möglich, dass ein Gerät zwar im WLAN funken und sich mit anderen Geräten austauschen, sich aber nicht mit dem Internet verbinden kann. Wie man das einrichtet, ist auch im oben erwähnten Online-Artikel nachzulesen.

Nachdem ich dies ausprobiert hatte, gab es leider ein Komfortproblem. Der Inverter zählt die insgesamt erzeugten kWh sowie die täglich erzeugten kWh stetig hoch. Bekommt der Inverter keinen Zugang zum Internet, zählt leider auch der Tageszähler munter weiter nach oben, wird also nie resettet. Grund hierfür ist, dass der Inverter den Tageszähler nur zurücksetzt, wenn er seine Zeit mit dem Backend synchronisiert (Bild 5).

Die Ursache ist verständlich. Das WLAN-Modul im Inverter wird nur von der DC-Seite (Solarmodule) mit Spannung versorgt. Nachts vergisst es also seine Zeit. Nach dem Hochfahren ist die Zeit intern immer auf eine Art Default-Wert gesetzt. Wenn der Inverter von Default auf einen beliebigen anderen Zeitwert wechselt, wird auch der Zähler zurückgesetzt.

Wen der stetig steigende Tageszähler nicht stört, der kann die Daten recht einfach ins eigene Smart Home transferieren. Eine Möglichkeit für Home Assistant, die ich länger eingesetzt habe, ist das Plug-in home-assistant-omnik-inverter, das die Daten per Screenscraping vom Inverter abgreift.

Firmwareanpassung in Eigenregie: Da wir genügend Informationen sowie Tools vom WLAN-Modul-Hersteller beziehen können und auch Referenzfirmware verfügbar ist, wäre auch die Anpassung der Firmware selbst denkbar. Man könnte sie so umfunktionieren, dass sie nur noch Daten sendet, aber keine Befehle von der Cloud mehr entgegennimmt. Die Befehle zum Herausrücken der WLAN-Credentials könnte man auskommentieren und laut Manual ist sogar ein MQTT-Client in der Firmware möglich!

Aber: Da es sich bei dem Inverter um ein CE-konformes Gerät handelt, ist diese Option zwar für den interessierten Hacker reizvoll, jedoch aus rechtlicher Sicht und Safety-Betrachtung keine Lösung.

Lokales Inverter-Backend: Das Projekt deye-microinverter-cloud-free ist auch eine schöne Option. Hier wird das Backend simuliert, die Zeit synchronisiert und die Daten in das Smart-Home-System integriert.

Allein die Komplexität hat mich hier etwas abgeschreckt, da es einen weiteren Service irgendwo in meinem Netzwerk oder auf meinem HA-Server als Docker-Container bedeutet hätte, um dem man sich auch ab und an mal kümmern muss.

Airgapping und ESPHome oder Tasmota: ESPHome und Tasmota sind geniale Smart-Home-Projekte mit extrem guter Usability. Ich habe überlegt, ob ich sie nicht auch für die Security-Verbesserungen der Solaranlage einsetzen kann.

In meinem Konzept gibt es aber eine Anforderung, die beide Projekte nicht in ihrer DNA haben: Airgapping. Das bedeutet auf

Deutsch so viel wie „Luftspalt“ und ist im Security-Kontext das komplette Trennen von Systemen. In meinem Beispiel bedeutet das, dass der Inverter gar nicht erst in mein Heim-WLAN gelassen werden soll, ich aber dennoch seine Daten abgreifen möchte.

Dies könnte man z. B. durch einen speziellen Router umsetzen, der mit beiden Netzwerken reden kann und nur ganz bestimmte Nachrichten und Kommunikation erlaubt. Das wäre einerseits kein direktes Airgapping mehr, andererseits wäre so ein Router nicht mehr ganz so günstig.

Ich habe daraufhin geschaut, was mit ESP8266s und ESP32s möglich ist. Und leider musste ich feststellen, dass diese sich zwar in ein Netzwerk einwählen und parallel auch einen Accesspoint aufmachen, sich aber nicht in zwei Netzwerken gleichzeitig als Station anmelden können.

Das ist aber auch gar nicht so schlimm. Der Inverter selbst sammelt die Daten nicht im Sekundentakt. Nur etwa alle fünf Minuten stehen neue Messwerte bereit, um abgeholt zu werden. Der erste Stresstest mit einem ESP8266 bestand aus dem Versuch, sich immer wieder im Wechsel in meinem Heim-WLAN und dann im Inverter anzumelden. Das funktionierte ohne Probleme und Projekt **S|P|I|D|R** war geboren (Bild 6).

Das Projekt S|P|I|D|R

Das Projekt macht folgendes: Der ESP baut eine Verbindung zum Accesspoint des Inverters auf. Aus der lokalen Webseite des Inverters liest die Software dann die Daten des Inverters aus. Anschließend wird die Verbindung mit dem Inverter gekappt und das Board meldet sich als Client am Heim-WLAN an. Dort kann es dann die Inverter-Daten z. B. an das Smart Home weiterleiten. Schließlich wird auch die WLAN-Verbindung wieder gekappt und danach beginnt der Prozess von vorn. Es besteht also nie gleichzeitig eine Verbindung zum Inverter und zum WLAN/Internet.

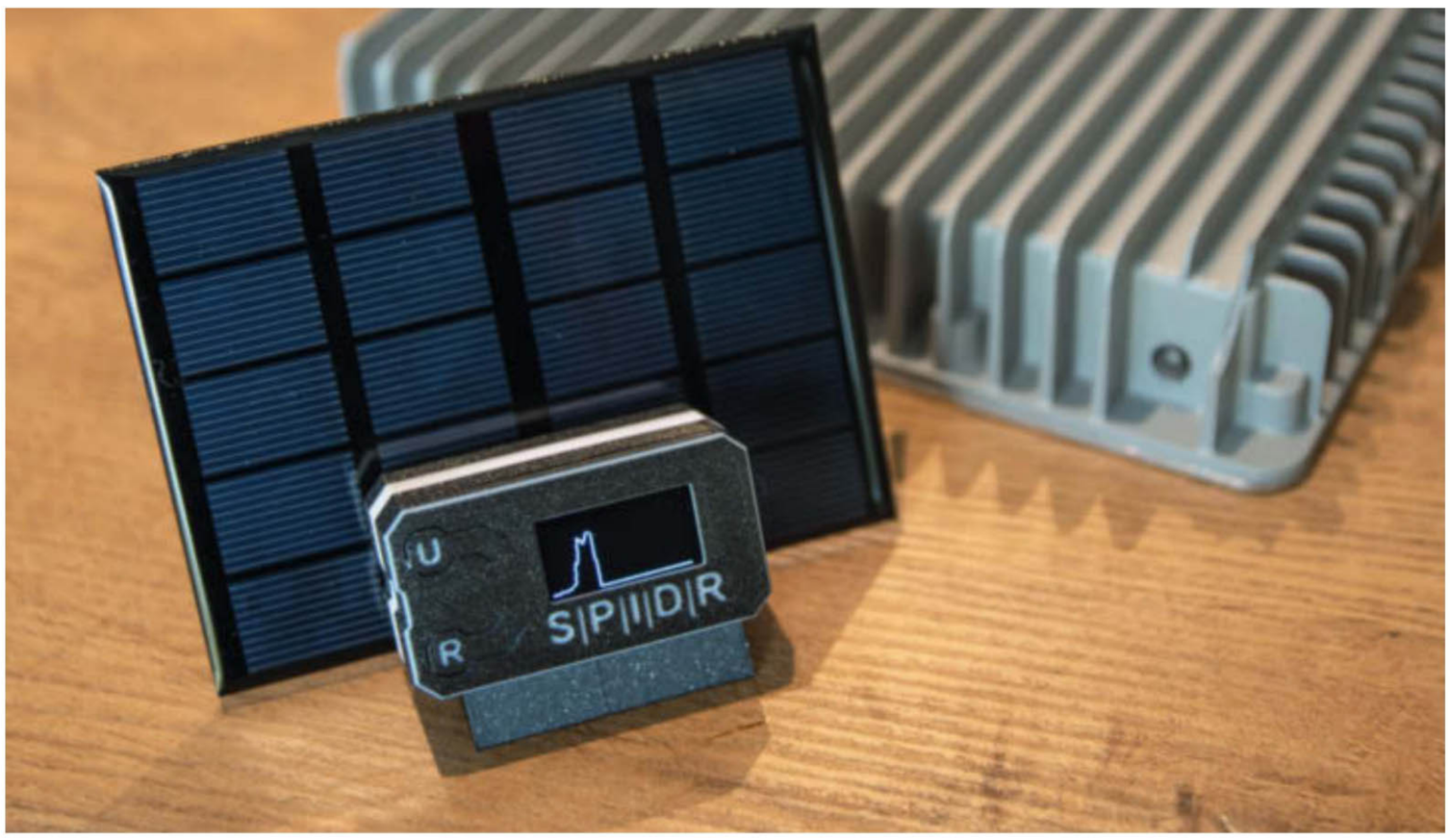
Der Projektname steht für **S**mart Home and **P**rivacy focused **I**oT **D**ata **R**elay. Gestartet mit dem Namen **deye-esp-mqtt-bridge** dachte ich zunächst an eine Nischenlösung für nur einen Solarhersteller. Später fand ich heraus, dass eben nicht nur Deye diese WLAN-Module bezieht, sondern zig andere Hersteller. Auch deutsche Firmen wie Bosswerk / Green Akku labeln Produkte unter anderem Namen und bringen sie hier in Verkehr. Die Technik ist oft dieselbe.

Sicherheit und Usability gleichzeitig umzusetzen ist schwer und oft genug wirft man Herstellern mangelnde Sicherheitsvorkehrungen vor. Beide unter einen Hut zu bekommen war mein wichtigstes Ziel. Idealerweise sollte sich jeder solch ein kleines Gerät leisten und es selbst ohne Vorkenntnisse zu Hause installieren und konfigurieren können.

Auch ohne Smart Home mach das Projekt Spaß. Die Daten bleiben zu Hause und auf dem Display (Bild 7) bekommt man jede kWh auch ohne App mit.

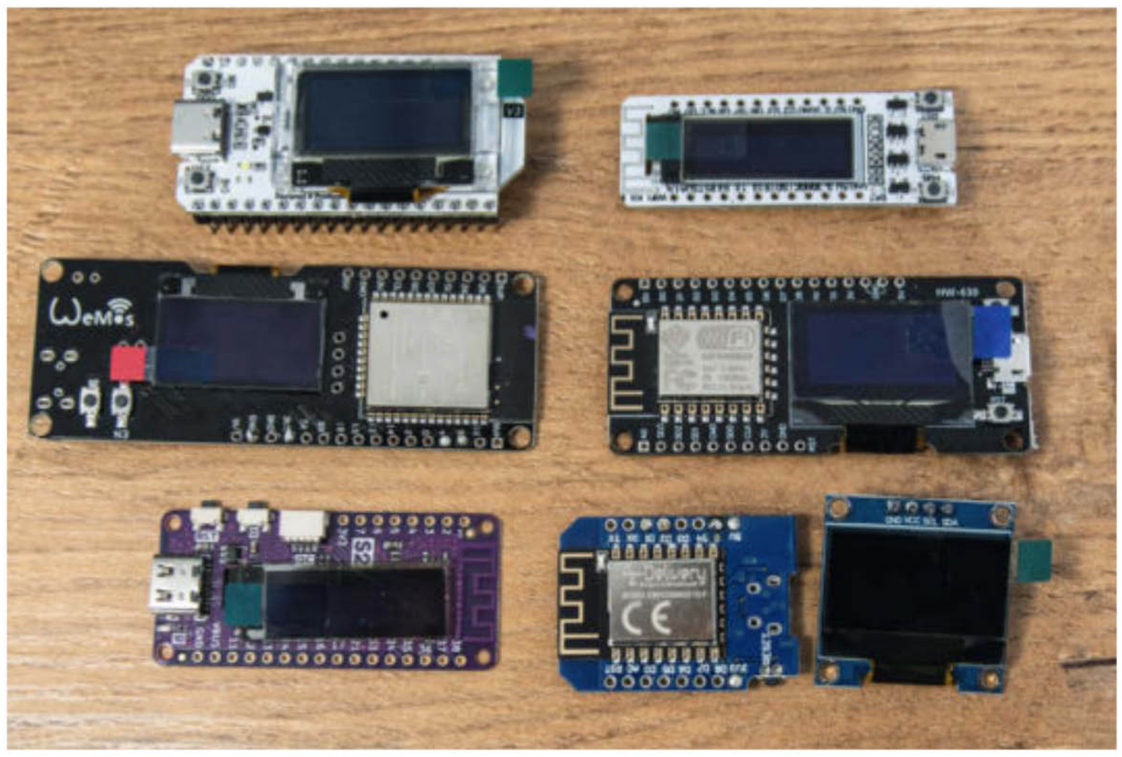
Meine Anforderungsliste enthielt diese Punkte:

– Usability: Jeder soll das Projekt nutzen können.



Roland Marx

Bild 7: Das Display als Infozentrale



Roland Marx

Bild 8: Von oben links nach unten rechts: HeltecWifi Kit 32, Heltec ESP8266, Wemos Lolin32, NodeMCU ESP 8266 OLED, Wemos Lolin S2 Pico, ESP 8266 (Wemos D1 mini Clone) und Display einzeln

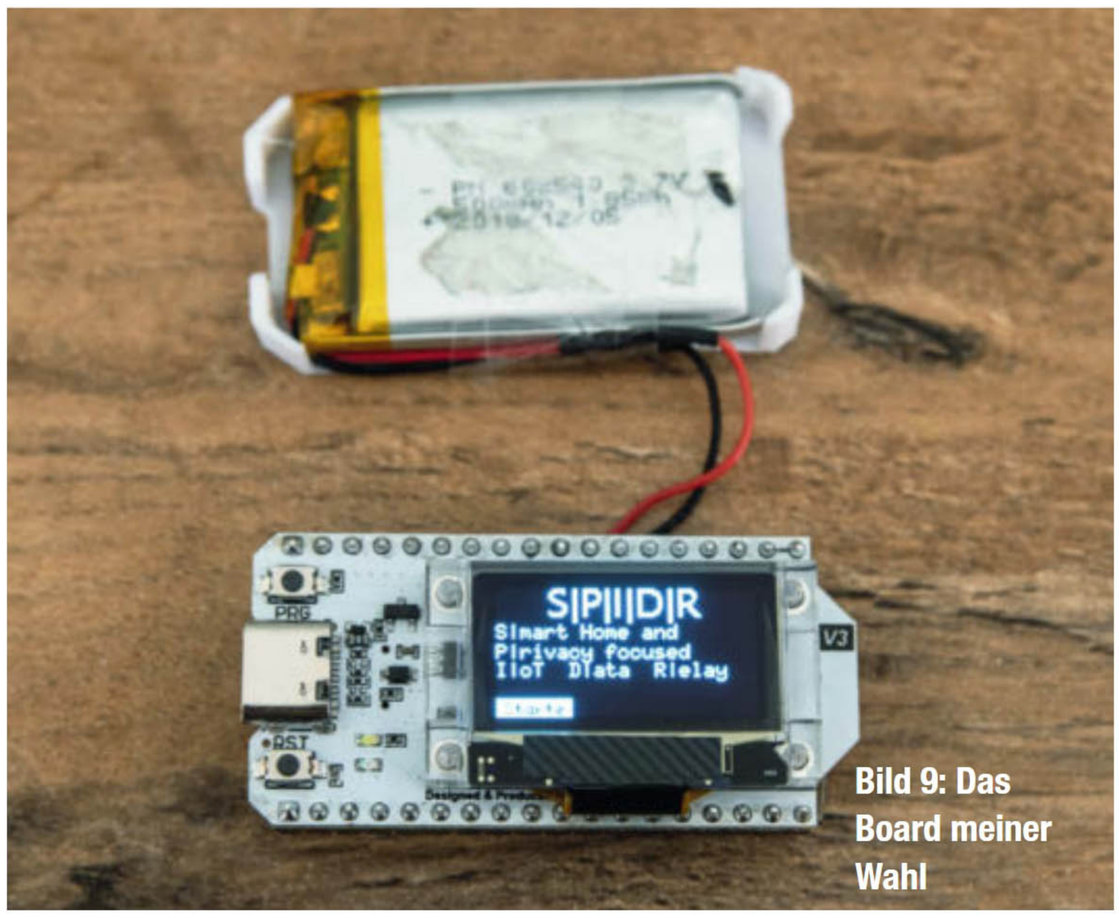


Bild 9: Das Board meiner Wahl

Roland Marx

Update

Die aktuellen Versionen des für diesen Artikel verwendeten Inverters der Firma Deye sind mit bei Tests der Stiftung Warentest im April 2024 im vorderen Feld dabei gewesen. Jeder, der solche Inverter betreiben möchte, kann dies nun auch einfach und cybersicher tun.

- Stand-alone: auch ohne MQTT als Display und Datenlogger
- Secure: Inverter gar nicht ins Heimnetz lassen müssen – Airgap
- Data Relay: Daten vom Inverter sammeln und an MQTT-Broker übergeben
- Integration: in verschiedene Smart-Home-Systeme (Beispiel mit Home Assistant, HA)
- Modularität & Erweiterbarkeit: andere Geräte, mehr Funktionen …

Auf welcher Hardware läuft S|P|I|D|R? Die Hardware sollte preiswert und der Energieverbrauch klein sein. Ich löte zwar gerne, dennoch habe ich mich wegen der Usability und Massentauglichkeit auch mit fertigen Modulen samt Display beschäftigt.

Aktuell läuft das Projekt auf ESP8266- sowie ESP32-Boards (Bild 8). Als Display habe ich mich auf OLEDs mit 128 × 32 oder 128 × 64 Pixel fokussiert. Die Wemos- und Heltec-Boards verfügen auch über Li-Ion-Batteriemanagement-Chips. Einige Boards mit integrierten Displays habe ich auch konkret getestet und bereits einige Anpassungen und defines dafür erstellt. Die finden sich im Sketch in der Datei config.h. Dort ist auch das verwendete Board zu definieren. Mehr dazu finden sie auf der Projektseite in GitHub (Adresse siehe Kurzinfo-Link).

Folgende Boards wurden getestet:

- Heltec Wifi Kit 32 v3 ESP32 (128 × 64)
- Heltec ESP8266 NodeMCU (128 × 32)
- Wemos Lolin S2 Pico ESP32-S2 (128 × 32)
- Wemos Lolin32 ESP32 OLED (128 × 64)
- NodeMCU ESP8266 OLED HW-630 (128 × 64)

Um die Usability noch weiter zu erhöhen, werden vorgefertigte Binaries für ausgesuchte Boards auf Github zur Verfügung gestellt. Diese können dann nach einem initialen Flashen leicht online geupdatet werden. Die Konfiguration erfolgt wie bei Tasmota und WLED per Webbrowser.

Meine Hardware-Empfehlung: Das Heltec WiFi Kit 32 (Bild 9) ist zwar mit 15 bis 20 Euro nicht besonders günstig, allerdings bietet es viele Vorteile:

- Der ESP32 bietet im Vergleich zum ESP8266 Performance-Reserven.
- Der ESP32 bietet auch dank FreeRTOS bessere Multitasking-Optionen als der ESP8266.
- Das Display ist mit Abstandhalter vom PCB erhoben. Das macht das Casedesign deutlich besser.
- Zwei Taster, die von vorne bedienbar sind (User und Reboot)
- Das Board kann per USB ohne Tastendruck geflasht werden (das war nicht bei allen Boards der Fall).
- Das Board hat einen eingebauten Akkuladeregler.
- OpenSource-Gehäuse gibt es zahlreich auf Printables.com, auch im S|P|I|D|R-Design mit und ohne Batterie/LED (siehe Kurzinfo-Link).

Wer kann oder sollte das Projekt benutzen? Für Deye, Bosswerk, Omnik und andere Inverter sollte das Projekt direkt funktionieren. Eine Liste mit kompatiblen Geräten werde ich auf der GitHub-Seite von S|P|I|D|R pflegen. Der Sketch für die Arduino IDE sowie weitere Projektinformationen sind über den Kurzinfo-Link zu erreichen.

Direkt oder mit kleinen Anpassungen nutzbar sollte das Projekt all denen helfen, denen der Screenshot der Webseite des HiFlying-WLAN-Moduls bekannt vorkommt (Bild 10). Wenn es noch dazu ein Solarinverter mit 600 bis 800 W ist, sind die Chancen hoch, dass es „out of the box“ klappt.

Mit der Zeit ist die Code-Basis von S|P|I|D|R stark angewachsen. Ich habe das meiste bereits in einzelne Klassen verpackt, die sich um die Anzeige, die MQTT-Verbindung, den Webserver und das Ansteuern und Auslesen des Inverters kümmern. Die wichtigsten Dateien und deren Funktion:

- **spidr.ino**: Hauptprogramm, steuert den Ablauf und die State Machine
- **config.h**: enthält wichtige Settings wie den Board-Typ
- **DisplayManager.cpp:** Convenience-Funktionen für das Display
- **EnergyDisplay.cpp:** Display-Steuerung, zeigt abwechselnd unterschiedliche Werte an
- **Inverter.cpp:** parst die Daten aus dem HTML und stellt die Werte des Inverters bereit
- **InverterUPD.cpp:** kommuniziert per UDP mit dem Inverter, um die Zeit zu setzen
- **MQTTManager.cpp**: sendet die Daten an MQTT-Broker und so auch an Home Assistant
- **PreferencesManager.cpp:** speichert alle Parameter
- **SerialCaptureLines.cpp**: sendet Log-Nachrichten auf UART und in der WebSerial-Konsole
- **WebPages.h:** der HTML-Sourcecode und Templates für die Webseiten
- **WebServerManager.cpp:** Webserverfunktionen, Anzeige und Parameter schreiben

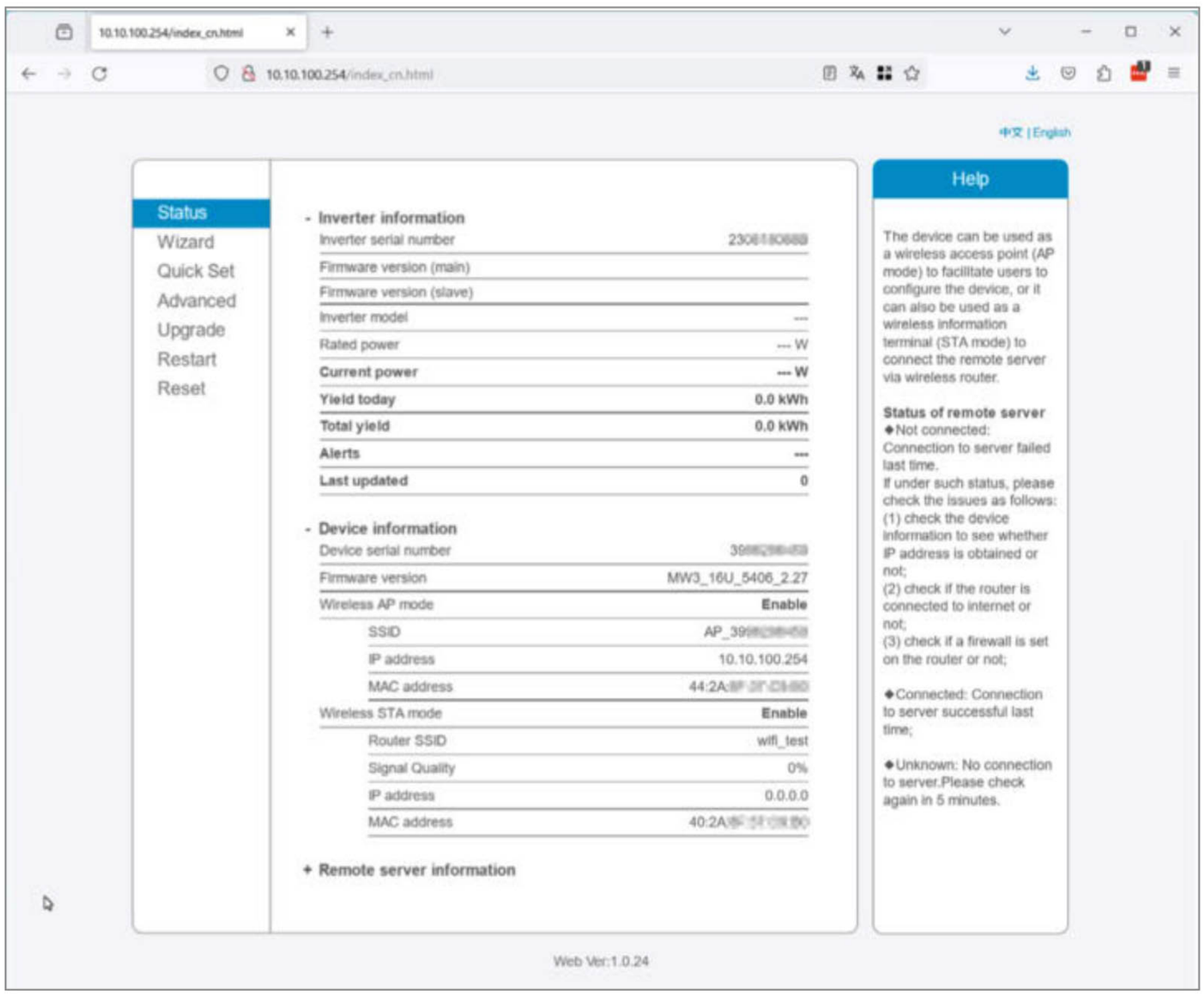


Bild 10: Wem diese Webseite bekannt vorkommt, der hat ein HiFlying-WLAN-Modul und kann das Projekt nutzen.

Ablaufplan

Der Ablauf von S|P|I|D|R passte ursprünglich locker in den Main Loop und die Setup-Funktion. Einen entscheidenden Umbruch gab es mit dem Integrieren des Webservers. Denn der

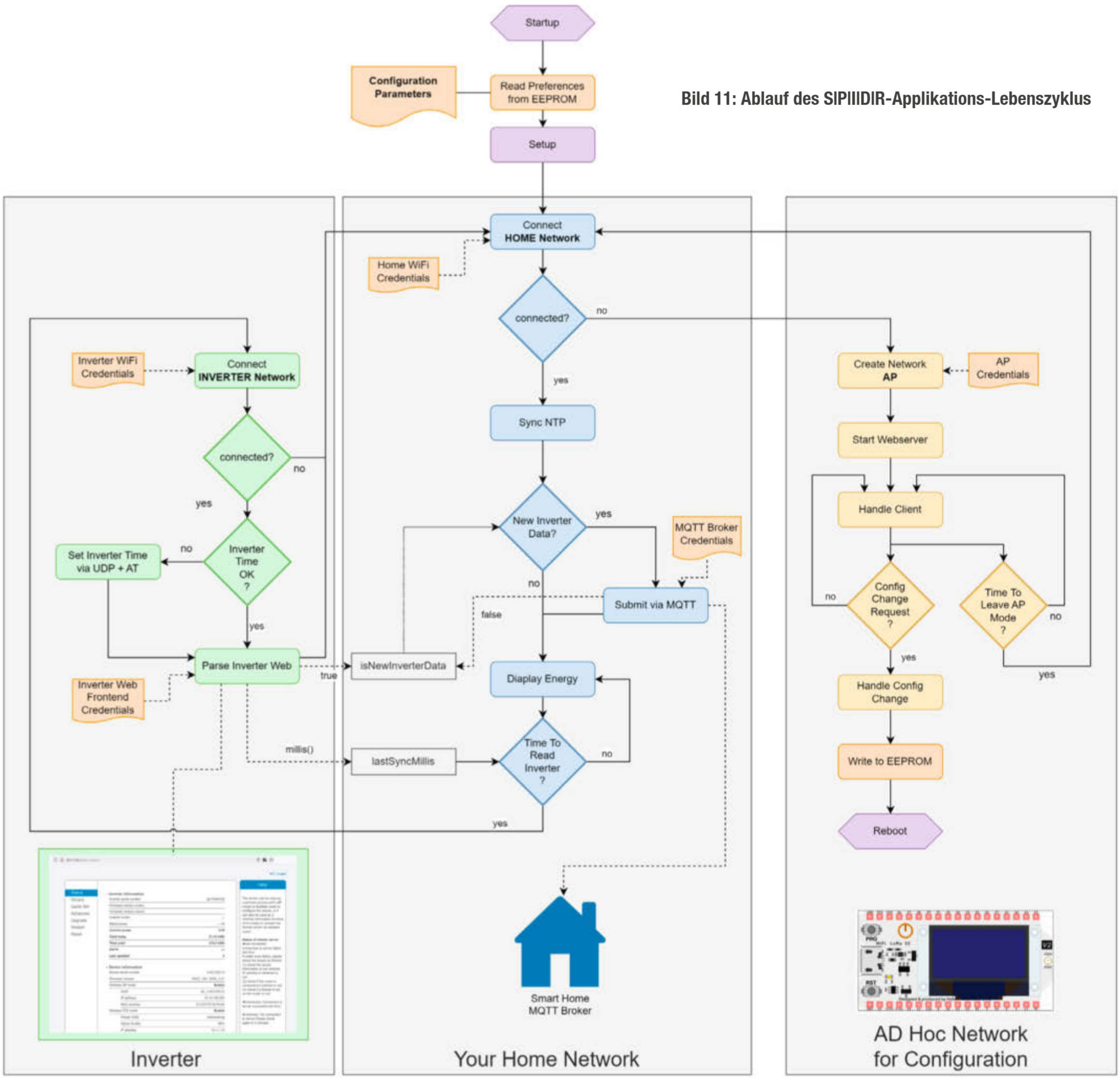


Bild 11: Ablauf des SIPIIIDIR-Applikations-Lebenszyklus

Webserver muss immer wieder gehandelt werden. Das bedeutet, er schaut nach, ob jemand eine Seite ausgeliefert bekommen will, um sie dann zu übergeben.

Ursprünglich war es egal, wie lange welcher Task dauerte und ob man eine Display-Routine hat, die den Kontrollfluss blockierte. Nun musste alles darauf umgebaut werden, dass im Heimnetzwerkmodus alle paar Millisekunden der Webserver angestupst wurde. Perfekt ist das leider noch nicht, was man an stellenweise langen Antwortzeiten des Servers merken kann.

Ein Betriebssystem wie FreeRTOS würde das Verhalten sicher verbessern, war auch bereits kurz in der Codebase. Dann stellte ich leider fest, dass ESP8266 keine FreeRTOS-Unterstützung haben, und ich bin auf einen State-Machine-Ansatz gewechselt, um die Hardware-Kompatibilität aufrecht zu halten. Hinzu kamen auch noch einige Ausnahmen wie das nächtliche Nicht-Vorhandensein des Inverters und dann auch ein Rücksetzen von im ESP gespeicherten Variablen.

Um auch selbst nicht den Überblick zu verlieren, habe ich zunächst den Programmfluss visualisiert (Bild 11). Er stimmt mit dem aktuellen Code des Projekts bis auf kleine Änderungen, wegen denen ich aber keinen neuen Ablaufplan zeichnen wollte, überein. Für das Verständnis des Programmablaufs reicht er völlig aus.

Im Quellcode des Programms ist viel kommentiert. Über den Kurzinfo-Link erreichen Sie auch eine GitHub-Seite mit ausführlichen deutschsprachigen Codebeispielen. Wer möchte, kann sich den Text in der Arduino IDE anschauen und mit dem genauen Ablauf vertraut machen. Im Folgenden wird nur der Teil genauer beschrieben, der aus der Weboberfläche des Inverters die wichtigen Daten extrahiert.

ScreenScraping – Werte aus der Weboberfläche ablesen

Eine Möglichkeit, die Daten vom Inverter zu bekommen, ist über sein Webinterface. Nach der Anmeldung mit Benutzername und Passwort kann man auf der Seite `index_cn.html` alle relevanten Werte sofort sehen (Bild 12).

Neben der Startseite des Routers gibt es noch eine weitere Seite, die im Hintergrund geladen wird und in Wirklichkeit alle Werte per JavaScript zur Anzeige bringt, die Seite `status.html`. In deren Quellcode findet sich neben einigem HTML auch folgende Sektion

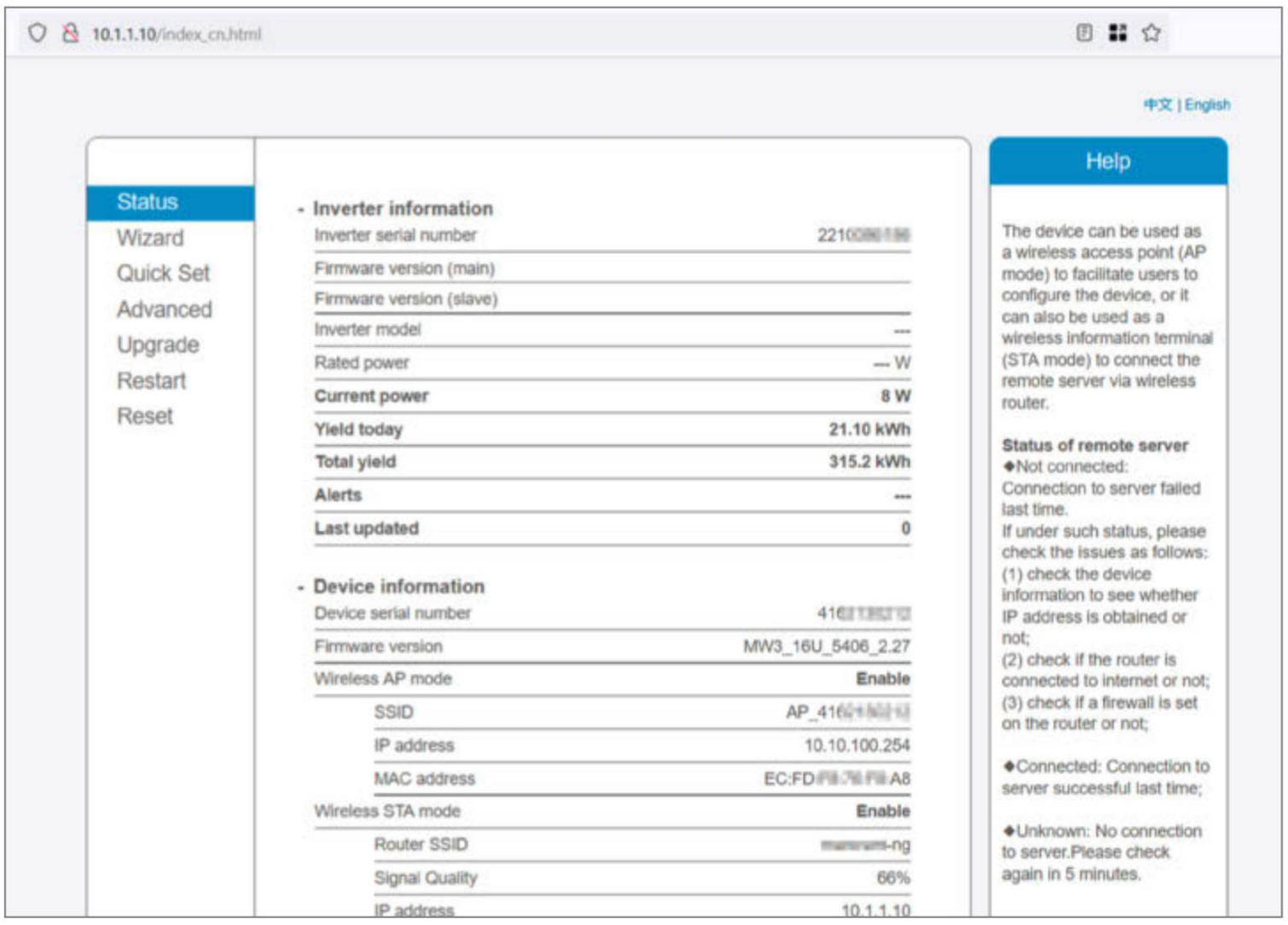


Bild 12: Die Weboberfläche des Inverters

(Bild 13), die alle gewünschten Werte mit einer einfachen Struktur beinhaltet.

Um das Screenscraping umzusetzen, müssen wir also folgende Schritte meistern:

1. die passwortgeschützte Seite `status.html` mit einem Webclient herunterladen
2. die wichtigen Parameter aus dem HTML herausfiltern
3. die einzelnen Werte z. B. von Text in Fließkommazahlen umwandeln und in lokalen Variablen speichern

Das Lesen der `status.html` und die Reduzierung auf alles, was mit „`var `“ anfängt, wird von der Funktion `readInverterDataFromWebInterface()` erledigt, die direkt im Mainfile `spidr.ino` definiert ist.

Die folgenden Teile befinden sich in `Inverter.cpp`. Nach dem Lesen wird die Funktion `inverter.updateData(response)` aufgerufen und das html-Fragment als `response` übergeben. In der `updateData()`-Funktion des Inverters wird nun das eigentliche Interpretieren vorgenommen und die Werte werden, bei erfolgreichem Parsen, als Klassenvariablen gespeichert.

In der `extractVariables()`-Funktion werden alle Informationen aus dem HTML-Code extrahiert.

Die `extractFloat()`-Funktion selbst ruft die Unterfunktionen `extractFloatValue()` und `extractAndValidateString()` auf, die den String nach dem Marker zurückgibt. Das Ergebnis wird von ihr als `float` konvertiert und zurückgegeben.

Die Anzeige mit dem OLED-Display

Viele kennen vermutlich die Adafruit_SSD1306-Library, da sie meist benutzt wird, um OLED-Displays anzusteuern. Ich selbst habe sie mit verschiedenen No-Name-Displays verwendet und zudem verschiedene ESP-Boards mit und ohne aufgelötete OLED-Displays benutzt.

Mit der Adafruit-Library hatte ich nur mäßigen Erfolg. Manche Displays bekam ich nicht ans Laufen und stellenweise ging es nur mit merkwürdigem Verhalten. Beispielsweise konnte ich das blaugelbe Display, das eigentlich 128×64 Pixel besitzt, nur als 128×32-Display ansprechen, wobei jede zweite Zeile dunkel blieb.

Im Tal der Tränen habe ich die Bibliotheken U8g2 und U8x8 kennen und schätzen gelernt. Beide sind auch Libraries, die sehr gut mit SSD1306-OLEDs umgehen können. U8x8 arbeitet auf Basis einzelner Zeichen und Zeilen und ist recht schlank. Man kann dank besonderer Fonts auch Icons auf den Screen zaubern. U8g2 hingegen hat noch mehr Funktionen und erlaubt auch das Zeichnen von Elementen und die Ansteuerung auf Pixelebene.

Was muss ich wissen, um zu starten? Das einfachste ist die Displayauflösung. Diese muss der Library natürlich mitgeteilt werden. Sie beträgt bei meinen OLEDs entweder 128×32 oder 128×64 Pixel.

Komplizierter ist da schon der I²C-Bus. Hier benötigt jedes Gerät eine Adresse. Die meisten Displays haben diese entweder fest vergeben oder man kann sie mit einer Lötbrücke noch anpassen. Oft steht sie hinten auf der Platine. Manchmal stimmt diese aber auch nicht mit der tatsächlichen Adresse überein! Hier hilft leider manchmal nur ausprobieren. Das Praktische an den U8x8- und U8g2-Libs ist, dass sie die I²C-Adresse in der Regel selbst herausfinden können. Es ist also oft nicht notwendig, diese zu spezifizieren.

Tipp: Möchte man auf mehreren Displays unterschiedliche Inhalte anzeigen, müssen die Adressen unterschiedlich sein oder man muss einen Multiplexer einsetzen.

Als Letztes muss man noch wissen, wie die I²C-Schnittstelle des Displays physikalisch am ESP verdrahtet ist. Bei Boards, die mit einem fest verlöteten Display daherkommen, ist es stellenweise sehr schwer, an korrekte Informationen zu den verwendeten SDA- und SCL-Pins zu kommen. Hat man ESP und Display separat verbunden, weiß man das natürlich. Für alle Boards, die ich ausprobiert habe, habe ich die Pins in den U8g2-Konstruktoren eingetragen.

```
// Auf der HTML Seite sind die Werte in diesem Fall sehr leicht auszuelesen:
/* Auszug ...
<script type="text/javascript">
var webdata_sn = "2200000000                                    ";
...
var webdata_now_p = "550";
var webdata_today_e = "2.0";
var webdata_total_e = "308.1";
...
var cover_mid = "4100000000";
var cover_ver = "MW3_16U_5406_2.27";
...
*/
```

Bild 13: Hier stehen die Daten, die wir brauchen.

Der Displaymanager

Nachdem ich viel mit den Displays ausprobiert hatte, war ich selten wirklich zufrieden. Die reine Darstellung von Zeichenketten mit einer Schriftgröße war nicht wirklich hübsch und ich musste viel Code duplizieren, um andere ähnliche Anzeigen zu realisieren.

Hinzu kam, dass mit immer mehr Funktionen auch die Komplexität zunahm und ich besser über Zustand und die einzelnen Schritte und deren Erfolg informiert werden wollte.

Also erstellte ich einen Displaymanager, dem man nur noch Daten zur Verfügung stel-

len musste. Unter anderem kümmert sich der Displaymanager darum, dass die Aufrufe den Programmfluss nur minimal blockieren und nur Änderungen übertragen werden, wenn das notwendig ist.

Das Big-Number-Display: Es dient der Anzeige der Leistung, der Gesamt- und Tagesenergiemenge. Um eine gute Lesbarkeit zu erreichen, wollte ich, dass die Zahl möglichst groß dargestellt wird und auch die Einheit sowie eine Beschreibung angezeigt werden.

Die Anzahl der benötigten Zeichen und damit auch der benötigte Platz variieren leider stark. Extrembeispiele sind „0W“ und „300,0 kWh“. Bei statischen Schriftgrößen passt entweder nicht mehr alles auf den kleinen Screen oder es wird alles nur klein dargestellt, auch wenn genügen Platz vorhanden wäre. Hier wollte ich keinen großen Kompromiss eingehen und habe mich für eine dynamische Anpassung der Fonts entschieden.

Das Action-Display: Diese Funktion ermöglicht es, schnell Statusmitteilungen anzuzeigen. Es gibt sowohl eine große Überschrift, Details und Parameter als auch einen Bereich für das Ergebnis und Details. Die Daten, die anzuzeigen sind, werden mit einem eigenen Datentypen `ActionData` vereinfacht (Bild 14).

Das Graph-Display: Auch eine schöne Anzeige des Energieverlaufs durfte nicht fehlen. Zwar handelt es sich hier noch um eine sehr frühe und einfache Form, durch die Kapselung im Displaymanager kann man sie aber später sehr einfach erweitern.

Grundlage für die Kurve sind P(t)-Datenpunkte, die in der Inverter-Klasse in einem Ringpuffer `powerData` abgelegt vereinfacht (Bild 15).

MQTT und die Smart-Home-Integration

Zwar ist die lokale Anzeige schon recht mächtig mit der Power- und Graph-Anzeige, und wer ein Smart-Home-System im Einsatz hat, wünscht sich sicher, die Daten hier auch schnell integrieren zu können. Ich selbst nutze schon einige Jahre Home Assistant und bin immer sehr zufrieden gewesen, da es dank einer großen Community für fast alles schon Integrationen gab.

Bei Home Assistant geht sowohl das Hosten eines MQTT-Brokers als auch das Erstellen von neuen Sensoren und das Übermitteln derer Daten sehr einfach. Den Broker kann man einfach auf dem gleichen System wie Home Assistant betreiben oder einen externen definieren. Wie man das genau konfiguriert, ist in der HA-Dokumentation gut erklärt.

Aus Sicht des ESPs befindet sich der MQTT-Broker im lokalen Netzwerk und er verfügt über einen Nutzernamen und Passwort, um sich dort anzumelden und die Daten zu publishen. Um den Solarinverter automatisch im Home Assistant wiederzufinden, kann man die Funktion MQTT Discovery nutzen. Sie arbeitet relativ einfach. Neben den reinen Sensordaten muss man auf ein bestimmtes Topic die Metadaten des Geräts und der Sensoren publishen. Hat man alles korrekt gemacht, taucht der Inverter direkt bei den anderen Geräten auf.

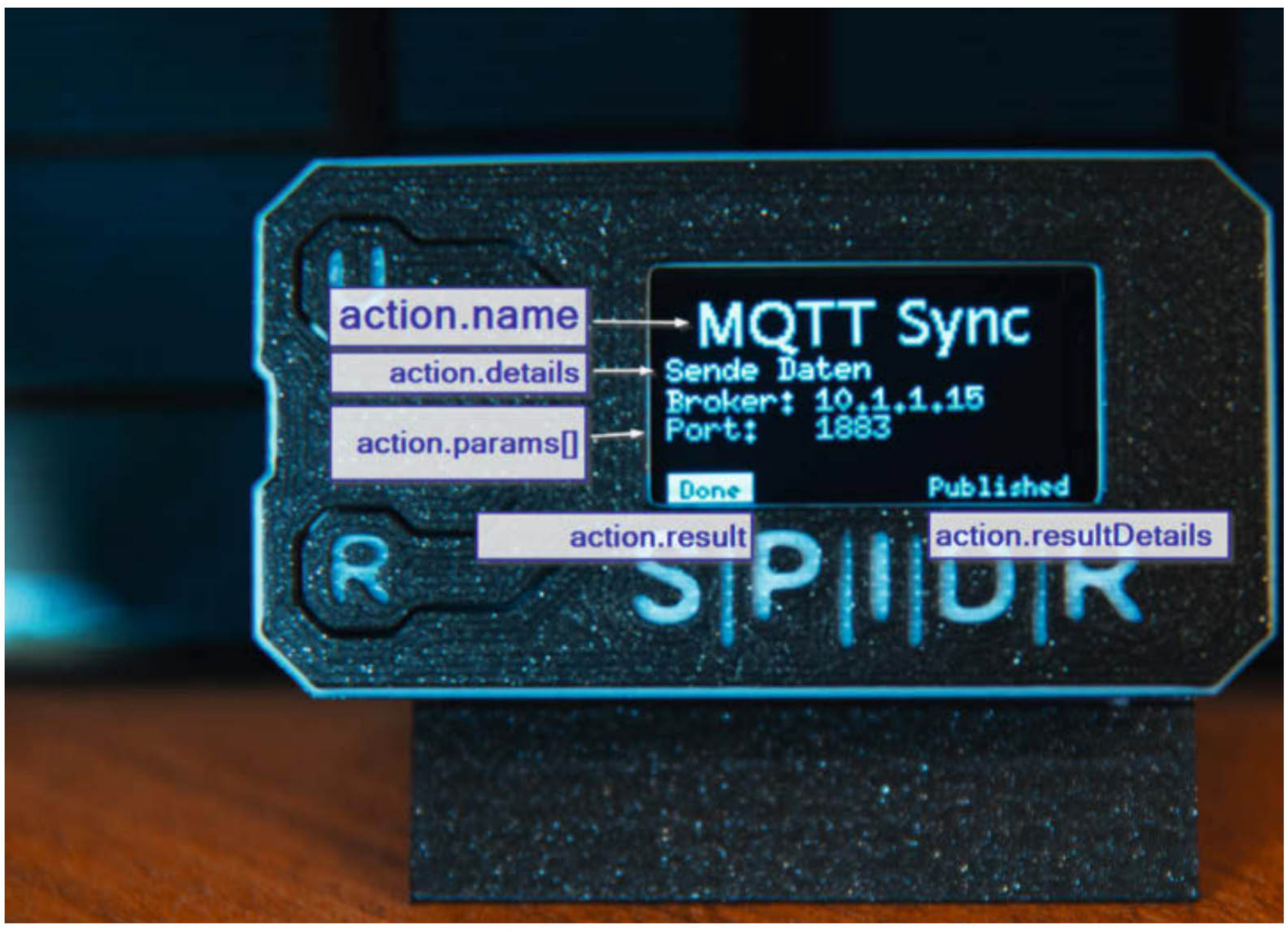


Bild 14: Das Action-Display stellt alle wichtigen Inverter-Daten dar.

S|P|I|D|R-Webserver

Aus der Idee heraus, S|P|I|D|R möglichst einfach und benutzbar zu machen, kam ich zu dem Schluss, dem System eine Weboberfläche zu verpassen.

Durch den Server wird es möglich, dass man ein fertiges Board mit OLED bestellt und direkt mit einem vorgefertigten Binary flasht. Die Konfiguration, die man sonst im Quellcode vornehmen müsste, und auch das Kompilieren und Installieren der Arduino IDE selbst kann einem so erspart werden. Ein Feature, was ich bei WLED sehr zu schätzen weiß.

Auch Stand-alone ist es praktisch, die aktuellen Daten auf dem Inverter sehen zu können. Zu Debug-Zwecken ist auch ein Serial Output vorhanden.

Das Ganze ist natürlich erst rund mit einem schicken Gehäuse. Ich habe für das Heltec WiFi Kit 32 ein schönes Gehäuse auf Printables gefunden und noch etwas angepasst. Auf Printables gibt es die STL- und Fusion360-Dateien dazu (siehe Link in der Kurzinfo).

I think I S|P|I|D|R!

Für das Nachbasteln kann man entweder das GitHub-Projekt verwenden oder auch vorkompilierte Images für verschiedene Systeme direkt flashen und per Webbrowser einrichten. Kein Löten, kein Kompilieren. Mit dem Tasmonizer oder esptool geht das auch super einfach.

Die Entwicklung hat sehr viel Zeit und Mühen gekostet und bietet eine gute Grundlage für Erweiterungen und zusätzliche Projekte – mit und ohne Airgap. Es würde mich freuen, wenn sich Mitstreiter finden, die Security, Privacy und Usability optimieren wollen und ihre Ergebnisse für alle kostenlos zur Verfügung stellen. *—hgb*

Bild 15: Die verschiedenen Displays

Pixel-Lampe mit WLED

Mit dem WLED-System und einem ESP kann man LED-Streifen ganz leicht ansteuern. An die Wand geklebt wirkt das 1D-Feuerwerk jedoch etwas verloren. Wieso also nicht eine zweidimensionale Pixel-Lampe bauen, die ordentlich Retrocharme in die Bude bringt? Der Wow-Effekt ist garantiert.

von Wolfgang Ziegler

Als Kind der Achtzigerjahre habe ich noch lebhafte Erinnerungen an Arcade-Spielhallen und die unzähligen Stunden, die ich in solchen verbracht habe. Einer der Klassiker übte schon immer eine besondere Faszination auf mich aus: Space Invaders.

Kürzlich entdeckten meine Kinder Bügelperlen als neues Hobby für sich und als wir nach passenden Motiven suchten, boten sich selbstverständlich Sprites aller Art an. Es dauerte nicht lange, da war der erste Bügelperlen-Space-Invader (Bild 1) fertiggestellt – mein erstes haptisches Erlebnis mit dieser Spezies.

Daraus entstand die Idee, das Motiv für ein Projekt im größeren Maßstab zu verwenden – eine Pixel-Lampe, die man sich an die Wand hängen kann. Dank des hauseigenen 3D-Druckers sollte die Herstellung keine große Herausforderung darstellen. Farbige LEDs könnten dem Projekt dann noch das gewisse Etwas verleihen. Vielleicht ließen sich diese sogar mit einem Mikrocontroller ansteuern, um auf diese Weise spannende Lichteffekte zu erzielen? Doch Schritt für Schritt.

Kurzinfo

- 2D-Pixel-Lampe mit WLED bauen
- Einzelne LEDs mit JSON steuern
- WLED-Befehle mit Python ausführen

Checkliste

Zeitaufwand:
1 Tag

Kosten:
20 bis 50 Euro

Material

- ESP8266 oder ESP32, z. B. NodeMCU
- 5-V-Netzteil ab 3 A
- 46 5V-NeoPixel-LEDs, Typ WS2812
- Kondensator mit 1µF
- Transparentes Filament
- Kabel für die LED-Verbindungen

Werkzeug

- 3D-Drucker
- Lötkolben und -zinn
- Seitenschneider

Mehr zum Thema

- Carsten Wartmann, Disco is back – Lichtshow mit WLED, Make 2/23, S. 8
- Ulrich Schmerold, Tetris-Klon mit LED-Streifen, Make 1/15, S. 12
- Heinz Behling, Intelligentes Heim mit Home Assistant, Make 1/21, S. 100

Alles zum Artikel im Web unter
make-magazin.de/x7an

Gedruckte Pixel-Boxen

Die einzelnen Pixel des Sprites habe ich in TinkerCAD als Würfel mit vier Zentimetern Kantenlänge modelliert, wobei ich sie innen hohl und einseitig offen gelassen habe, um die LEDs unterbringen zu können. Ihr findet einen Link zu den Modellen in der Kurzinfo.

Es dauert eine Weile, bis 46 Stück davon gedruckt sind. Ich empfehle, im Slicer mehrere Würfel auf einmal auf der Druckplattform zu platzieren. Als Filament habe ich transparentes PETG gewählt, um später möglichst spannende Lichteffekte zu erhalten (Bild 2). Solltet ihr kein transparentes Filament haben, könnt ihr ersatzweise auch weißes verwenden. Dieses hat sich in einem früheren Experiment auch als praktikabel erwiesen.

Die Würfel habe ich mithilfe einer Heißklebepistole zusammengesetzt. Lediglich die oberste und unterste Reihe musste ich abwandeln, da in der originalen Grafik einige Pixel nur mit den Ecken aneinanderstoßen und sich so nicht gut verkleben lassen. Testweise habe ich versucht, diesen Effekt mithilfe eines leeren (also später unbeleuchteten) Würfels zu simulieren. Dieser erwies sich aber als störend, da er von den umliegenden Pixeln immer noch stark durchleuchtet und somit sichtbar wurde. Aus reinem Pragmatismus habe ich mich letztlich für eine Variante mit Einrückungen entschieden (Bild 3).

Verkabeln und löten

Die ursprüngliche Idee, einen LED-Streifen zu verwenden, hatte ich bereits in einer früheren Phase verworfen, da die Abstände zwischen den LEDs nicht passten und deren korrekte Unterbringung in den einzelnen Pixeln schlichtweg zu mühsam geworden wäre. Stattdessen habe ich einzelne NeoPixel-Module (WS2812) miteinander verdrahtet (Bild 4). In die Wände zwischen den einzelnen Pixeln habe ich mithilfe eines ausrangierten Lötkolbens Löcher geschmolzen und kurze Kabelstücke gezogen, um die 46 einzelnen LEDs an Ort und Stelle miteinander zu verlöten (Bild 5). Dieser Arbeitsschritt nahm mit Abstand den größten Teil der Aufbauzeit in Anspruch. Wer keine Löcher in den Kunststoff schmelzen will, kann auch die Würfel mit Löchern verwenden, die im GitHub-Repository des Projekts liegen.

Wie man die LEDs aneinanderreiht, ist im Grunde nicht wichtig. Man sollte allerdings die Polung (5 V zu 5 V sowie GND zu GND) beachten und dass man das Ausgangssignal DOUT mit DIN der folgenden LED verbindet (Bild 6). Am Ende ergibt sich ein zusammenhängender LED-Streifen, auf dem die meisten Effekte schon spektakulär aussehen werden. Möchte man seine Pixel-Lampe später auch mit den 2D-Effekten bespielen, die WLED bereithält, muss man seine Kabelführung etwas planen, wie es im Kasten „Löchrige Matrix“ beschrieben ist.

Verbindung zum ESP

Als Mikrocontroller, der die Ansteuerung der LEDs übernimmt, habe ich ein NodeMCU-Board mit ESP8266-Prozessor gewählt, da sich dieser bereits in der heimischen Bastelschublade befand. Ähnliche Boards mit

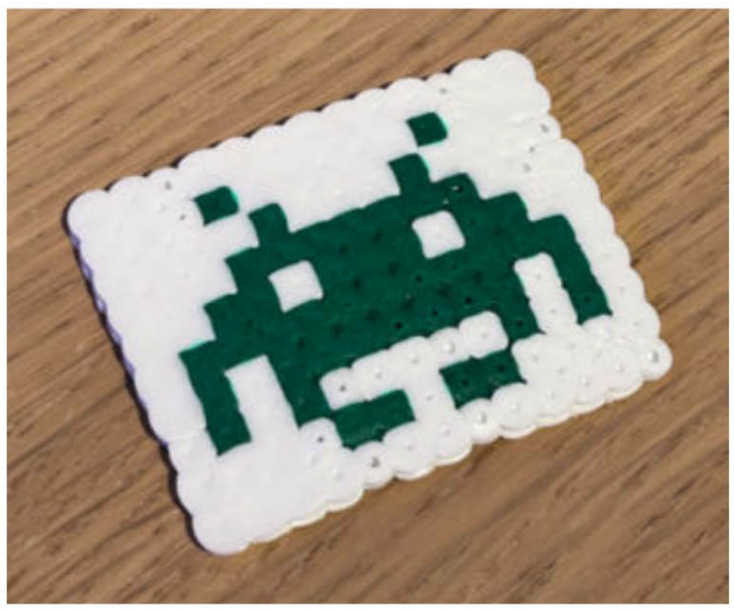


Bild 1: Ein gebügelter Space Invader

Bild 2: Gedruckte Pixel-Boxen aus transparentem Filament

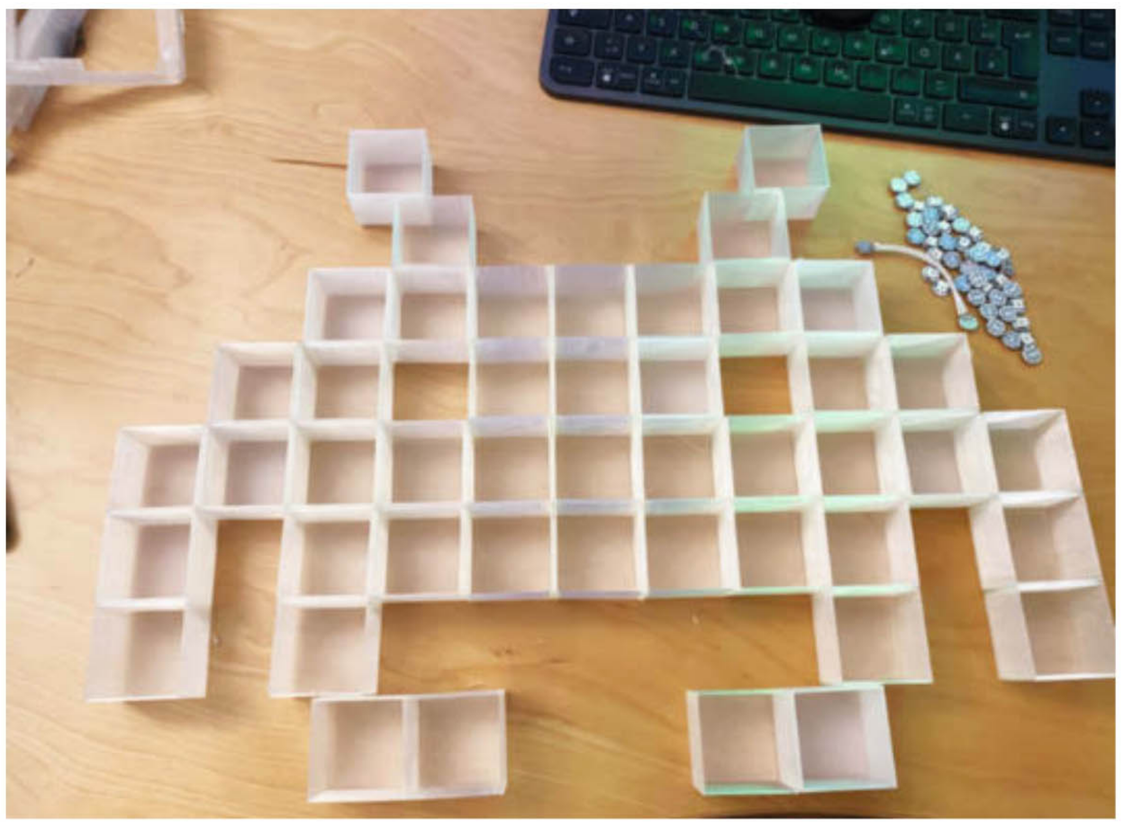


Bild 3: Die zusammengeklebte Lampe und die LEDs rechts daneben

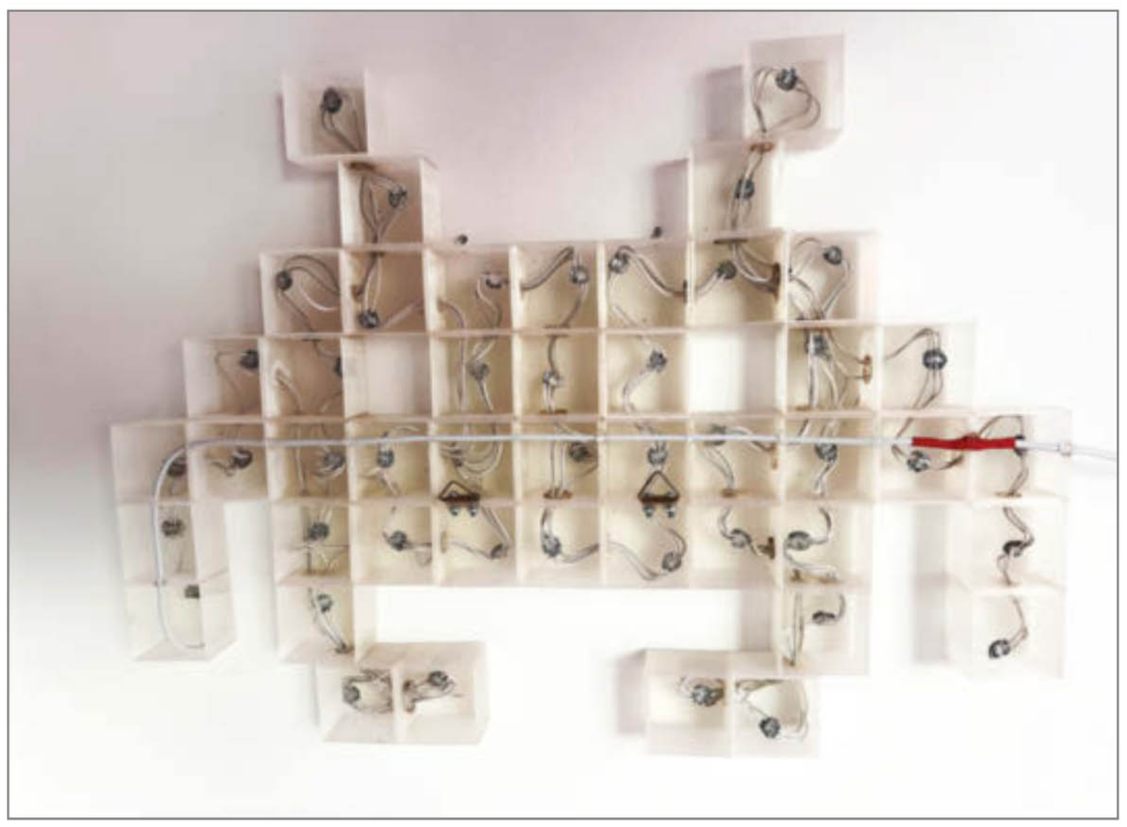


Bild 5: Zum Schluss sind LEDs und Kabel in den Boxen versteckt.

Bild 4: Das Verbinden der einzelnen LEDs dauert eine Weile.

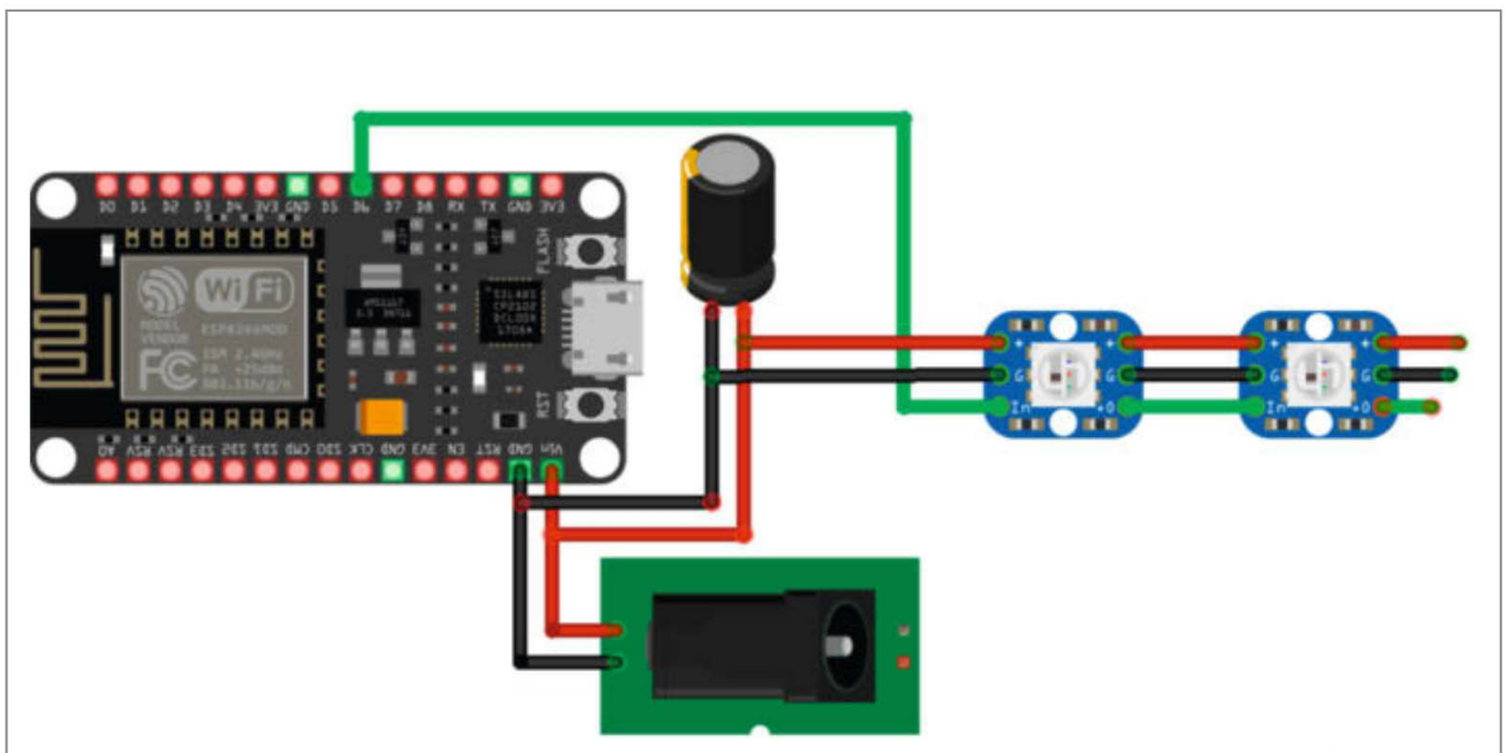


Bild 6: Der Mikrocontroller und die LEDs erhalten Strom von einem 5-V-Netzteil. Ein 1µF-Kondensator glättet auftretende Stromspitzen.

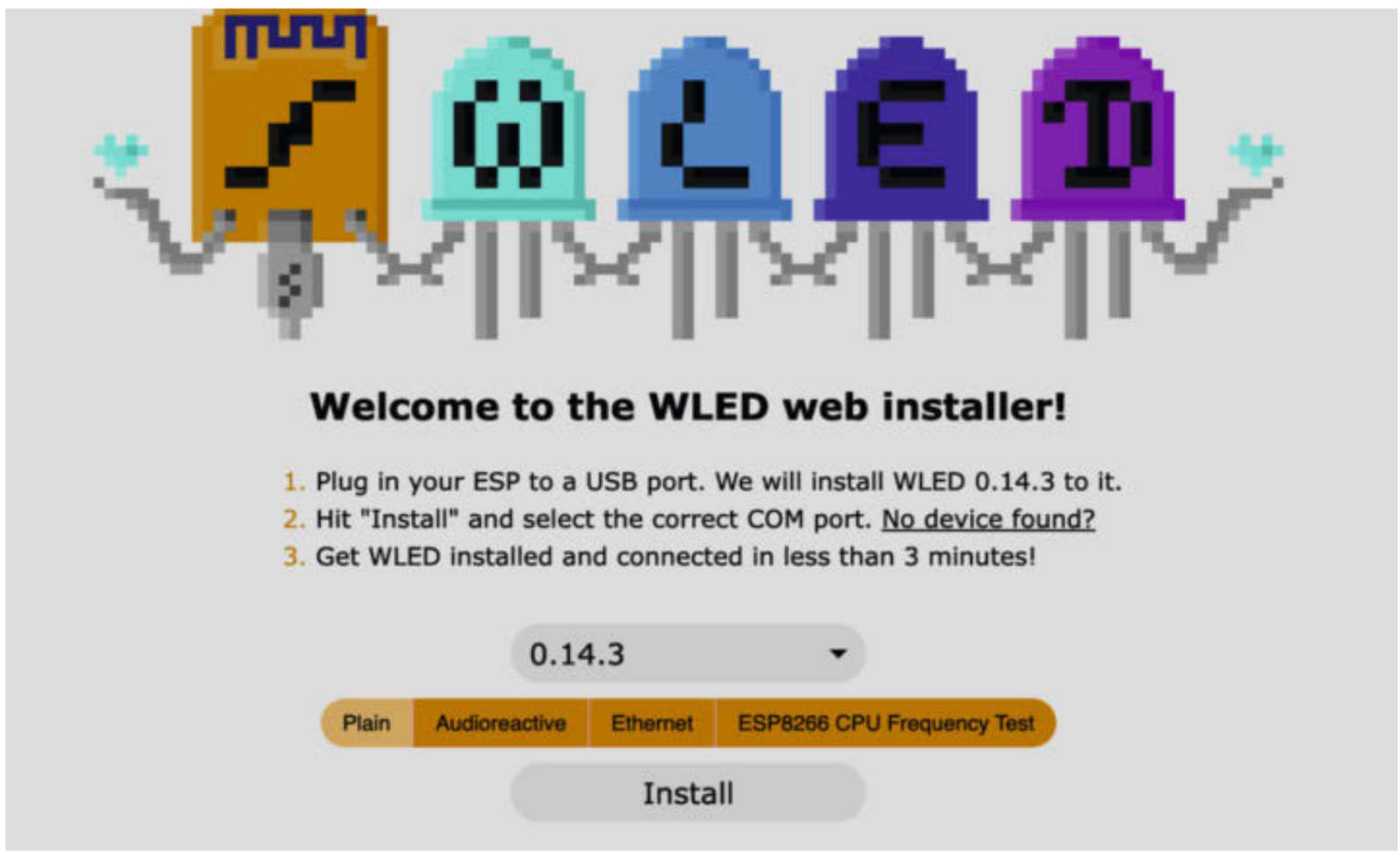


Bild 7: Die Installation von WLED funktioniert über einen Web-Flasher im Browser.

ESP8266 oder ESP32 eignen sich aber natürlich ebenso gut.

Das Bild 6 zeigt, wie der Mikrocontroller, die Stromversorgung und die LEDs verbunden werden müssen. Da sowohl das Mikrocontroller-Board als auch die WS2812-Module 5 Volt benötigen, können diese mit derselben Stromquelle versorgt werden. Man sollte lediglich darauf achten, ein Netzteil zu verwenden, das ausreichend Strom liefert. Wenn wir von maximal 55 mA pro LED ausgehen und den Strombedarf des Mikrocontrollers mit 500 mA dazurechnen, benötigen wir etwas über 3 Ampere. In der Praxis hat sich jedoch ein 2-A-Netzteil als vollkommen ausreichend erwiesen, da meine LEDs selbst bei weißem Licht mit maximaler Intensität keine 55 mA erreichten. Außerdem lässt sich die maximale Stromstärke in der WLED Steuerungssoftware begrenzen (dazu mehr im nächsten Abschnitt). Diese Funktion benötigt man z. B., falls man den ESP über ein

WLED mit Access Point verwenden

WLED kann man auch ohne ein verfügbares WLAN steuern. Dazu erstellt das System automatisch den Access Point WLED-AP. Dessen Passwort lautet wled1234 und die IP-Adresse, die man im Browser eingeben muss, heißt 4.3.2.1. Auf dem erscheinenden Willkommensbildschirm kommt man mit „To the controls“ direkt zur Steuerung. Außerdem kann man hier auch nachträglich seine WLAN-Zugangsdaten eingeben. Das ist praktisch, falls man die Lampe verschenken will.

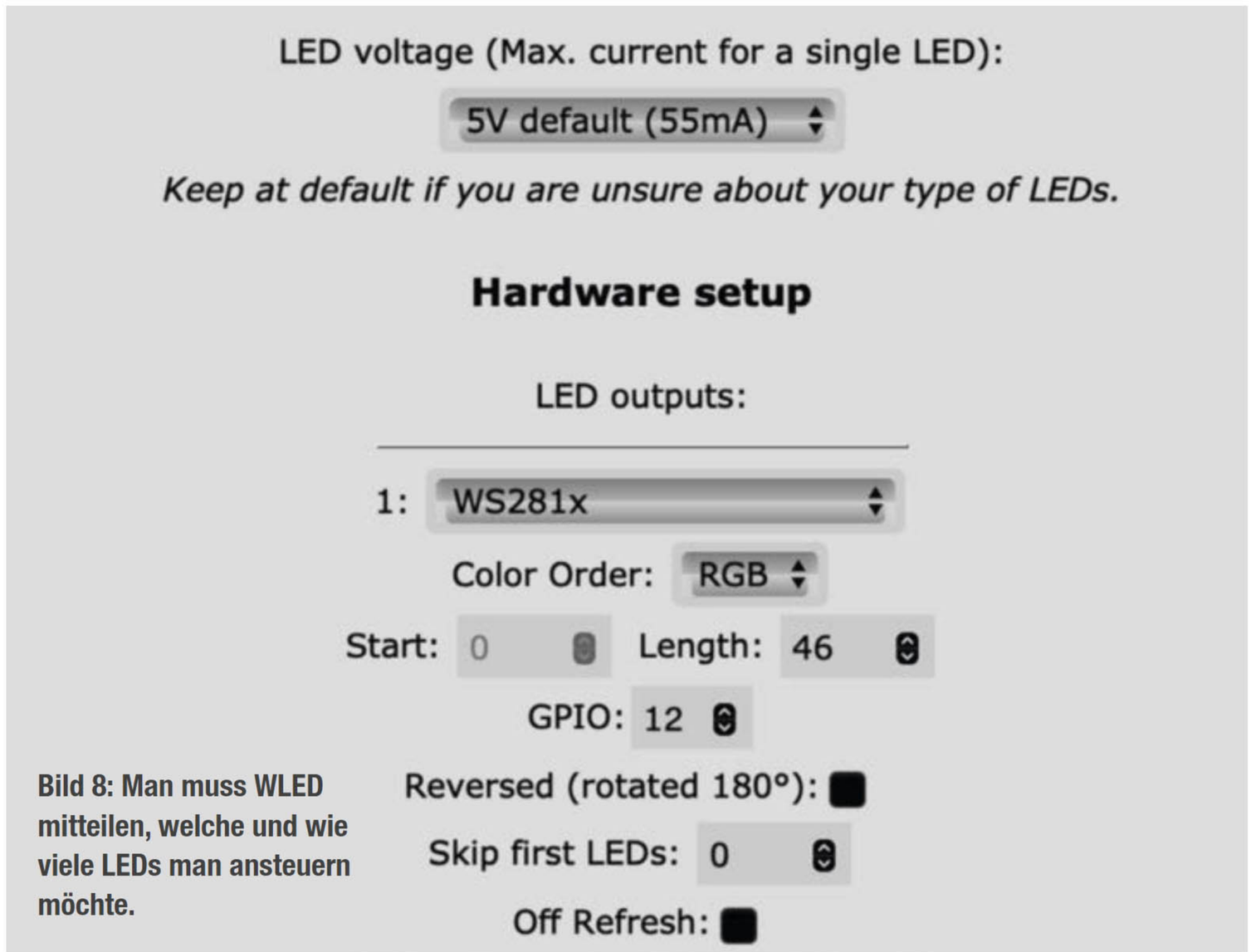


Bild 8: Man muss WLED mitteilen, welche und wie viele LEDs man ansteuern möchte.

USB-Netzteil betreibt und die LEDs ihren Strom vom Mikrocontroller-Board erhalten – den man nicht mit mehr als 1 A belasten sollte.

Weiter gilt es noch zu beachten, welchen Pin (hier: D6/GPIO12) man für die Datenleitung, also die logische Ansteuerung der LEDs, verwendet. Hier muss man unbedingt zwischen den Pin-Beschriftungen des Boards und den GPIO-Pin-Bezeichnungen des Mikrocontrollers unterscheiden. Für das verwendete NodeMCU-Board findet ihr die Tabelle mit den entsprechenden Pinouts hinter einem Link in der Kurzinfo.

Firmware flashen

Um die LEDs mit dem Mikrocontroller anzusteuern, verwende ich die frei verfügbare Firmware WLED. Durch dessen Verwendung ergeben sich schlagartig zahlreiche Vorteile:

- WLED ist mit einer großen Anzahl von verschiedenen Mikrocontrollern, Boards und LEDs getestet und kompatibel. Die tatsächlich verwendete Hardware lässt sich sehr einfach konfigurieren.
- Die Installation der Firmware erfolgt im einfachsten Fall direkt über den Webbrowser.
- Es gibt eine große Anzahl an vordefinierten LED-Effekten.
- WLED-Installationen kann man über den Webbrowser oder Apps für iOS und Android ansteuern.

Für die Installation öffnet man in Chrome den Web-Installer von WLED (siehe Link in der Kurzinfo), schließt das ESP-Board über ein USB-Kabel am Computer an, wählt die neueste stabile Firmware-Version (z. B. 0.14.3) aus und klickt auf „Installieren“ (Bild 7). Hinweis: Bei manchen ESP-Boards muss man vor dem Flashen die Boot-Taste gedrückt halten.

Am Ende des Installationsprozesses gibt man noch seine WLAN-Zugangsdaten ein und schon kann man über die IP-Adresse des ESP auf das Web-Interface zugreifen. Diese lässt

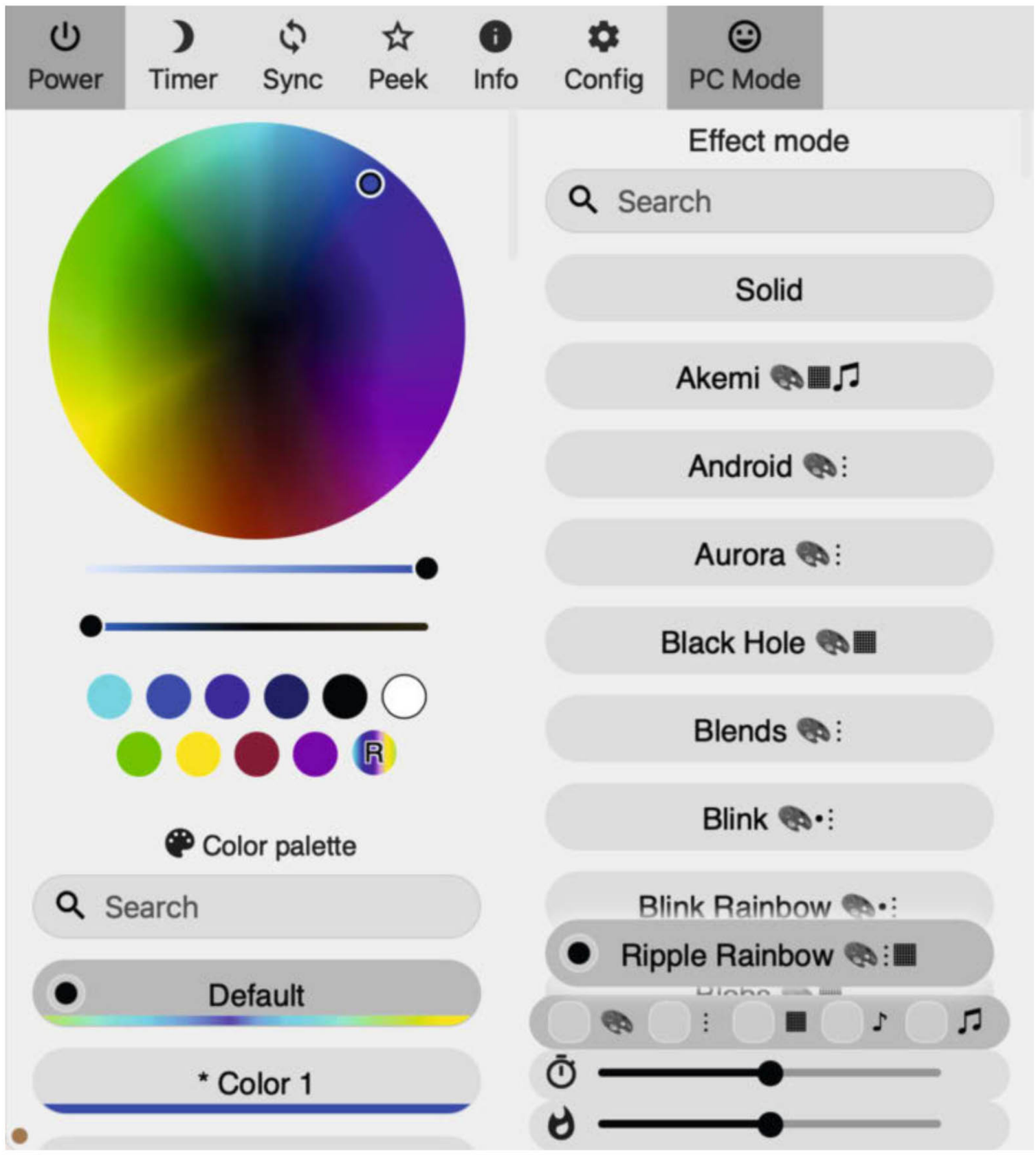


Bild 9: Auf der WLED-Benutzeroberfläche kann man zahlreiche Lichteffekte individualisieren.

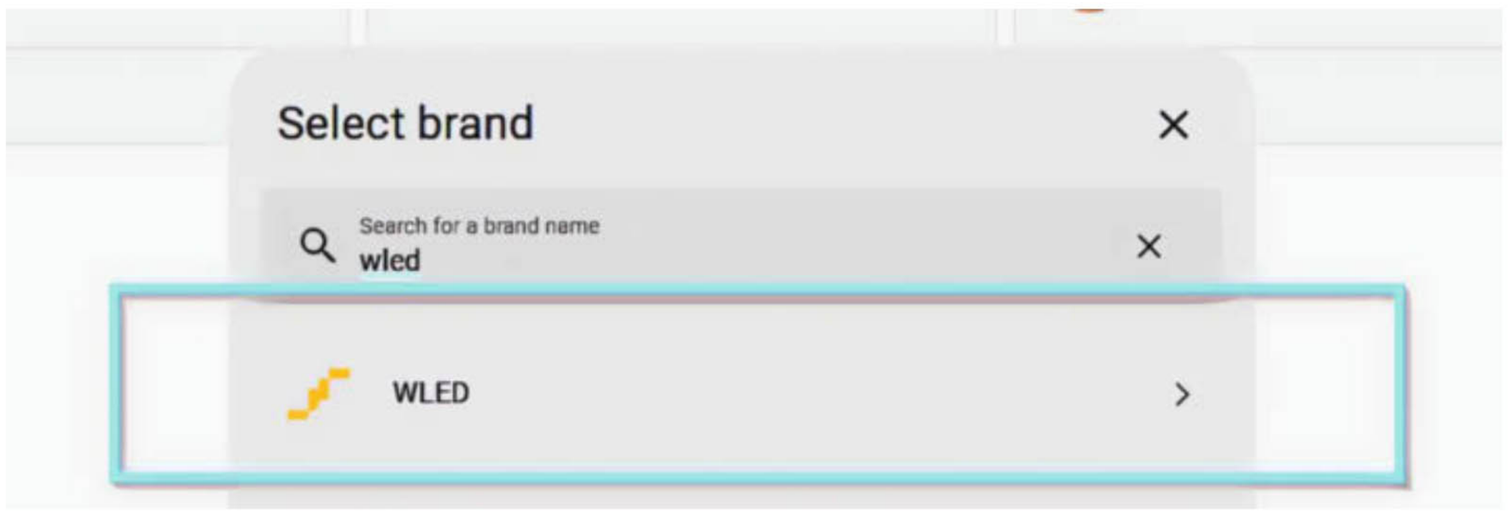


Bild 12: Home Assistant besitzt eine WLED-Integration, mit der man die Pixel-Lampe leicht einbinden kann.

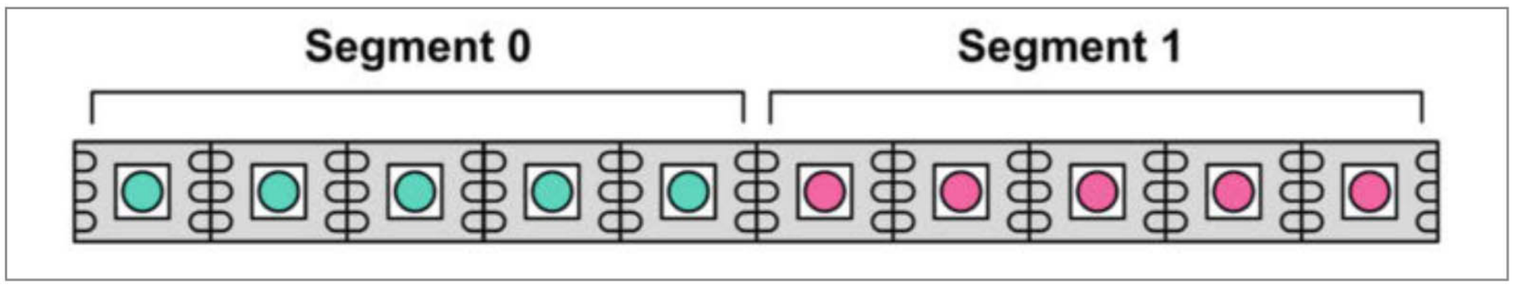


Bild 13: In WLED kann man einen Streifen segmentieren und den Leuchteffekt individuell einstellen.

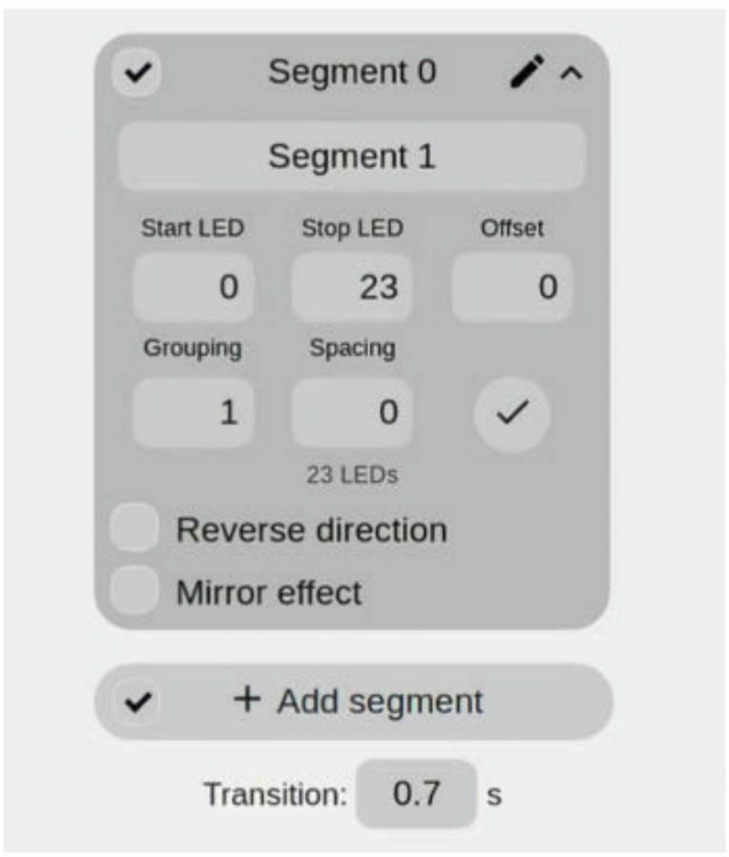


Bild 14: Mit Segmenten kann man einen längeren LED-Streifen unterteilen.

sich z. B. über den Router oder die WLED-Smartphone-App herausfinden, die WLED-Geräte automatisch erkennt – vorausgesetzt man befindet sich mit dem Smartphone im selben Netzwerk.

Um die LEDs richtig anzusteuern, muss man WLED noch über den Menüpunkt„Config/LED Preferences“ konfigurieren (Bild 8). Hier kommen die oben bereits erwähnten Einstellungen zur Anwendung:

- Bei „LED voltage“ ist standardmäßig 5 V ausgewählt. Das kann man so lassen.
- Darunter befindet sich die Rubrik „Hardware Setup“. Dort stellt man zunächst den Typ der WLEDs (WS281x) ein sowie ihren Farbmodus (z. B. RGB oder GRB). Die richtige Option hängt von den jeweiligen LEDs ab.
- Im nächsten Feld trägt man bei Length die Anzahl (46) der LEDs ein.
- Als GPIO-Pin nimmt man denjenigen, an dem das Datenkabel des Streifens angeschlossen ist (bei mir 12).
- Zum Schluss kann man noch einmal nach oben scrollen und den Punkt „Enable automatic brightness limiter“ aktivieren (falls inaktiv). Darunter lässt sich der maximale Stromverbrauch des Systems regulieren. So kann man auch schwächere Netzteile verwenden, indem WLED die Lichtstärke auf den eingestellten Maximalwert drosselt.
- Speichern (ganz oben) nicht vergessen!
- Damit ist die Konfiguration auch schon abgeschlossen und man kann die umfangreiche WLED-Effektpalette an der fertigen Pixel-Lampe ausprobieren (Bild 9).

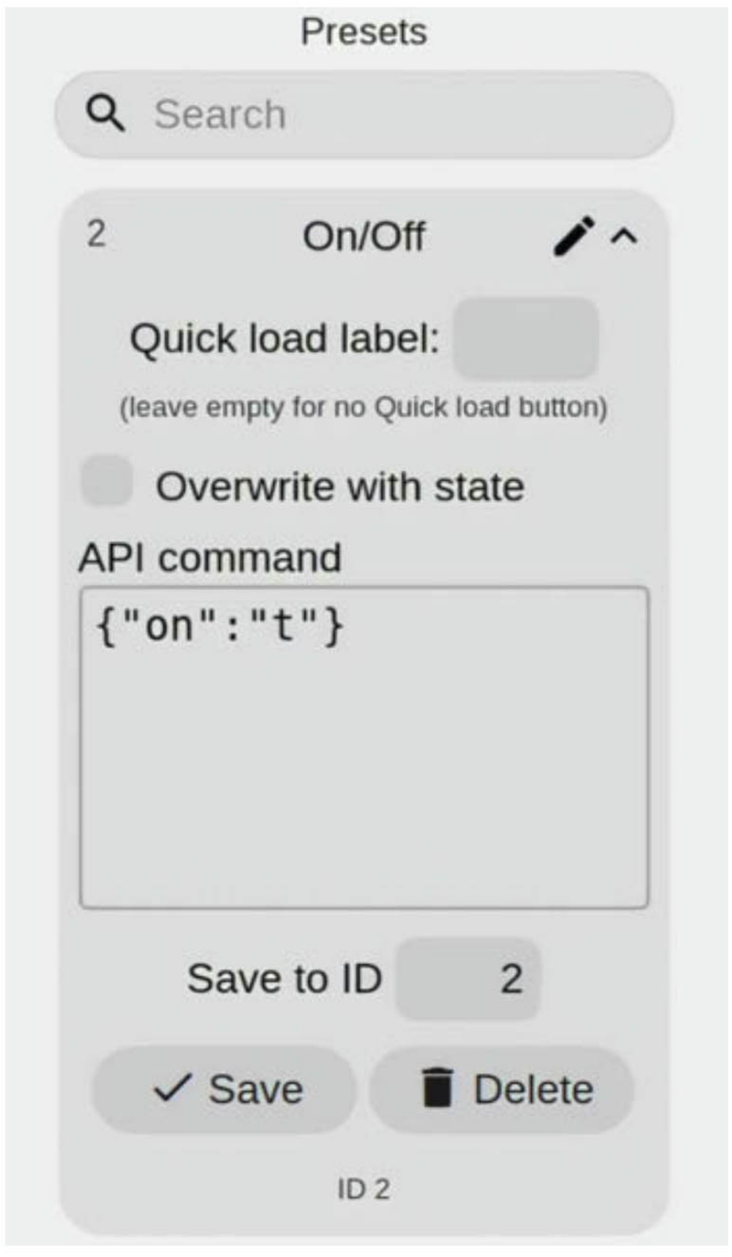


Bild 15: Mit einem JSON-Befehl lassen sich LEDs auch einzeln ansteuern.

Home Assistant Integration

Wer zu Hause ein Smart Home mit Home Assistant betreibt, kann die Lampe auch dort einbinden. Dafür bietet das System praktischerweise eine Integration für WLED an, die den Prozess stark vereinfacht. Auf der Web-Oberfläche von Home Assistant fügt man das WLED-Gerät einfach über den Menüpunkt „Add Integration“ hinzu (Bild 12). Danach kann man die Lampe direkt von Home Assistant aus steuern. Selbstverständlich lassen sich damit verschiedenste Automatisierungen umsetzen.

In meinem Zuhause schaltet sich die Lampe beispielsweise durch einen Bewegungsmelder ein. Zusätzlich werden abhängig von der Tageszeit jeweils unterschiedliche Farbschemata und Effekte verwendet. Der Fantasie sind hier keine Grenzen gesetzt.

Abschnitte steuern

Die vorgefertigten Effekte und Animationen, die WLED standardmäßig anbietet, reichen im Normalfall aus, um beeindruckende Resultate zu erzielen. Doch was ist, wenn man gern die volle Kontrolle über die einzelnen Pixel-Farben der Lampe haben will – etwa weil man einen 8-Bit-Mario gebaut hat, der eigentlich bunt sein soll? Auch das ist möglich und wie so oft führen mehrere Wege zum Ziel.

Ein einfacher, aber auch umständlicher Ansatz führt über das Segmente-Feature von WLED. Dabei lassen sich die einzelnen LEDs in logische Abschnitte (Segmente) unterteilen, die man separat mit Lichteffekten bespielen kann (Bild 13). Sie können an beliebigen Positionen starten und enden und lassen sich auch benennen (Bild 14).

Wenn wir versuchen, die einzelnen Pixel mit diesem Ansatz anzusteuern, stoßen wir allerdings schnell an Soft- und Hardware-Grenzen: Die Funktion ist nämlich in erster Linie für LED-Streifen, also sequenzielle Abschnitte konzipiert und nicht unbedingt für komplexere Gebilde. Insgesamt bräuchten wir 46 Unterteilungen, um jede LED separat anzusprechen. Dies führt aber zum einen das Segment-Konzept ad absurdum und zum anderen ist die Funktion typischerweise für 10 bis 20 Segmente gedacht – was nicht zuletzt mit den begrenzten Hardware-Ressourcen des verwendeten Mikrocontrollers zu tun hat.

Pixelgenau mit JSON

Also muss eine andere Lösung her. Diese findet man in Form der WLED JSON WEB API

Löchrige Matrix

In WLED kann man eine Pixel-Lampe mit beliebiger Form auch als 2D-Matrix definieren. Wenn man für den Sprite – wie in meinem Projekt – nicht alle Pixel benötigt, lässt sich diese Matrix nämlich auch mit Löchern aufbauen, d. h., man lässt die LEDs, die nicht leuchten, einfach aus.

Dazu muss man sich als Erstes überlegen, wie man die LED-Matrix aufziehen will, z. B. dass sie sich von links unten nach links oben schlängelt (Bild 10). Sobald man den Mikrocontroller mit WLED geflasht und eingerichtet hat (siehe Abschnitt „Firmware flashen"), geht man in der WLED-Benutzeroberfläche auf „Config/2D Configuration" und ändert „1D Strip" in „2D Matrix". In den Optionen, die daraufhin erscheinen, legt man den Rahmen und die Ausrichtung seiner Matrix fest (Bild 11).

Damit die Pixel innerhalb dieses Rahmens korrekt angesteuert werden, muss man schließlich eine Konfigurationsdatei hochladen, die mithilfe eines Arrays immer von links oben nach rechts unten (auch bei einer vertikalen Verkettung) zeilenweise bestimmt, an welcher Stelle sich LEDs befinden und wo nicht (siehe Listing). Dabei steht eine 1 für aktive LEDs, 0 für inaktive (z. B. wenn man eine vollständige Matrix verwendet und LEDs ausblenden will) und -1 kennzeichnet eine Lücke. Die fertige Datei speichert man als 2d-gap.json ab, lädt sie ganz unten auf der Matrix-Einstellungsseite hoch und speichert die Konfiguration. Danach kann man ein paar der 2D-Effekte ausprobieren, etwa Swirl, GEQ oder Ripple. Falls die Animationen in eine andere Richtung laufen sollen, lassen sich die X- und Y-Achse bei den Segment-Einstellungen (Bild 13) mit der Option „Reverse" umkehren.

```
[-1,-1, 1,-1,-1,-1,-1,-1, 1,-1,-1,
 -1,-1,-1, 1,-1,-1,-1, 1,-1,-1,-1,
 -1,-1, 1, 1, 1, 1, 1, 1, 1,-1,-1,
 -1, 1, 1,-1, 1, 1, 1,-1, 1, 1,-1,
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
  1,-1, 1, 1, 1, 1, 1, 1, 1,-1, 1,
  1,-1, 1,-1,-1,-1,-1,-1, 1,-1, 1,
-1,-1,-1, 1, 1,-1, 1, 1,-1,-1,-1]
```

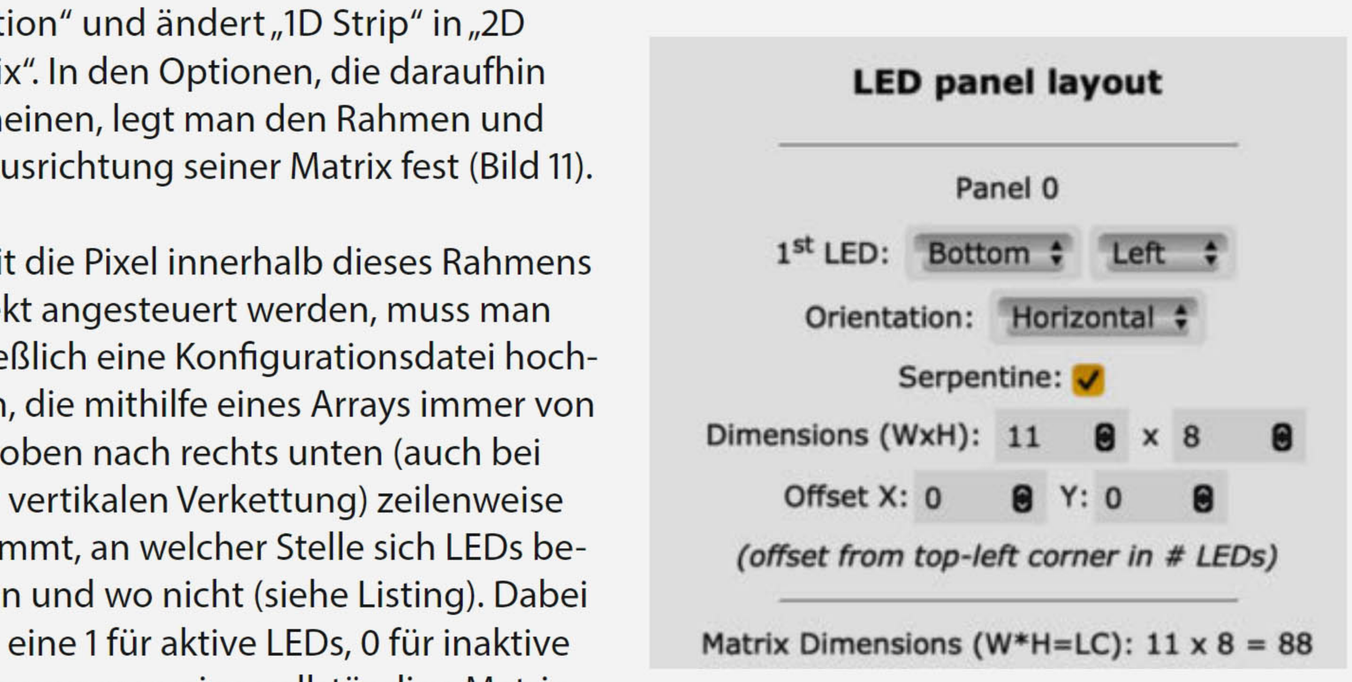


Bild 11: Die WLED-Optionen für eine 2D-Matrix

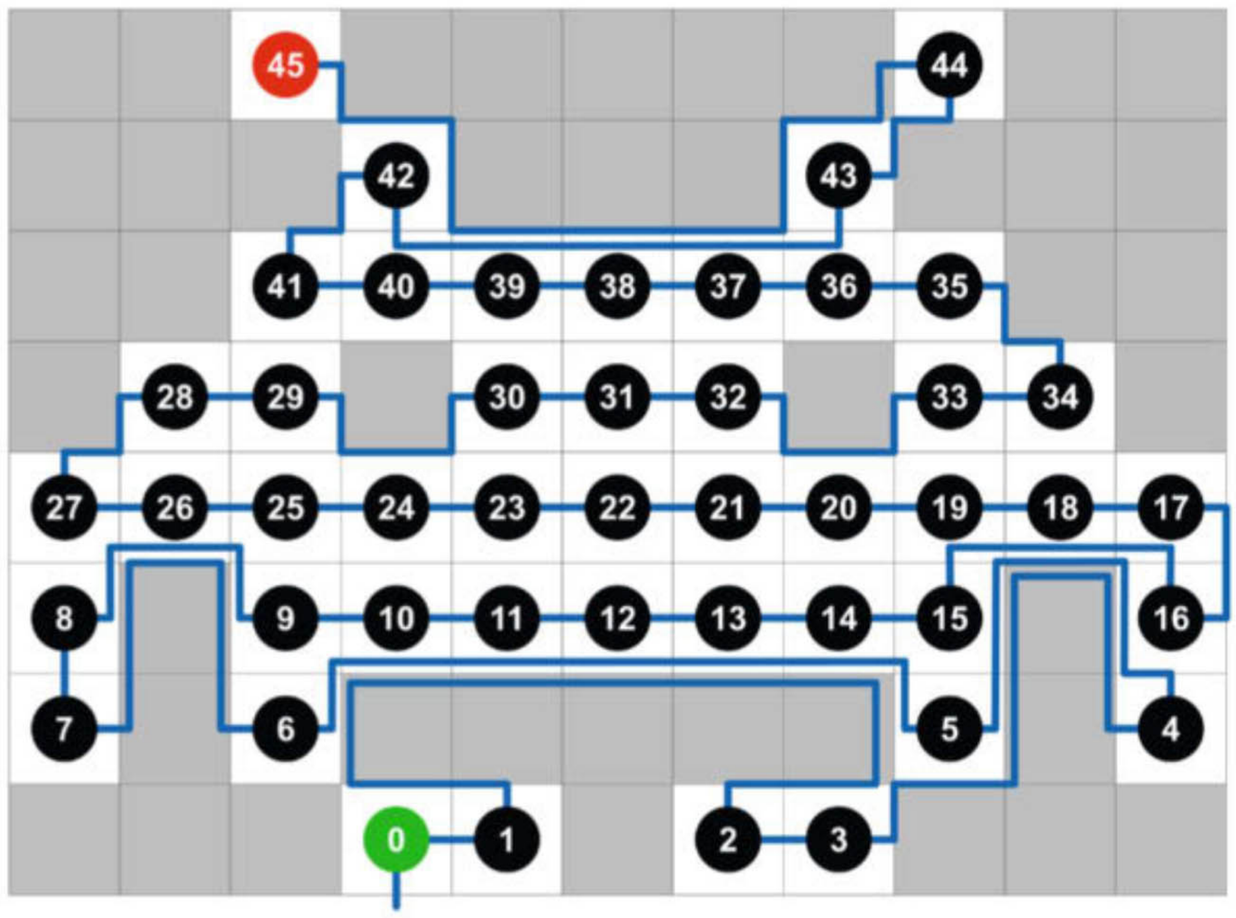


Bild 10: Mit dieser LED-Verkettung kann man die Pixel-Lampe auch als 2D-Matrix verwenden – trotz der Lücken.

wled.py

```
import requests
import random

wled_api_url = 'http://192.168.1.245/json'
num_leds = 46

flat_list = [i if i % 2 == 0
             else format(random.randint(0, 0xFFFFFF), '06x')
             for i in range(num_leds * 2)]

wled_json_data = {
    "seg": {
        "i": flat_list
    }
}

requests.post(wled_api_url,
              headers={'Content-Type': 'application/json'},
              json=wled_json_data)
```

(siehe Link in der Kurzinfo). Kurz erklärt, stellt uns diese Web-Schnittstelle sämtliche Funktionen, die man auch über die WLED-Benutzeroberfläche erreicht, als Kommandos zur Verfügung (und noch mehr). Die Dokumentation der JSON API mag aufgrund ihres Umfangs anfangs etwas komplex und vielleicht sogar einschüchternd wirken, jedoch lässt sie sich sehr schön experimentell erkunden. Auch hierfür bietet das Web-Interface von WLED den perfekten Einstieg. Über sogenannte Presets lassen sich nämlich ebensolche JSON-Kommandos absetzen und ohne zusätzliche Hilfsmittel ausprobieren.

Im einfachsten Fall (Bild 15) setzen wir lediglich das Kommando `{"on": "t"}` ab, was die LEDs ein- oder ausschaltet – das `"t"` steht für toggle. Wollen wir nun einzelne LEDs ansteuern, so lässt sich das über das Attribut `seg` umsetzen. Folgendes Kommando setzt z. B. das erste Pixel auf Rot, das zweite auf Grün und das dritte auf Blau.

```
{"seg":{"i":[0, "FF0000",
             1, "00FF00",
             2, "0000FF"]}}
```

Zur Definition der Farben muss man die hexadezimale Farbnotation verwenden, wie man sie aus dem Web-Umfeld oder anderen Programmierumgebungen kennt (siehe Link in der Kurzinfo).

Befehle mit JSON-Tool senden

Bereits auf Bild 14 lässt sich unschwer erkennen, dass diese Form der feingranularen LED-Ansteuerung schnell unhandlich wird, da der Platz in der entsprechenden Textbox doch recht begrenzt ist. Aus diesem Grund sollte man für längere JSON-Kommandos lieber ein zusätzliches Werkzeug benutzen. Ich verwende dafür das kostenfreie Insomnia (siehe Link in der Kurzinfo), doch auch Postman oder SoapUI sind beliebt.

Nachdem man das Programm installiert und gestartet hat, wird man aufgefordert, sich mit einem Account anzumelden. Es gibt aber im unteren Bereich des Login-Fensters die Option „Use the local Scratch Pad“. Mit ihr kann man das Programm auch erst mal ohne Account verwenden.

Danach erstellt man über die Schaltfläche „New HTTP Request“ ein neues Dokument. Um die Pixel-Lampe anzusteuern, muss man danach im oberen Arbeitsbereich zuerst den Befehl GET auf PUT umstellen und die IP-Adresse des Mikrocontrollers mit der Ergänzung /json eingeben (Punkt 1 in Bild 16). Lautet die IP-Adresse der Lampe wie im Fall des Autors 192.168.1.245, so ist die entsprechende JSON-API-URL also http://192.168.1.245/json. Danach ändert man unter der URL den Menüpunkt Body in JSON, gibt seine Befehle in das Programmierfeld ein und klickt auf den Send-Button.

Neben dem größeren Eingabefeld bietet Insomnia noch weitere Vorteile: JSON-Kommandos lassen sich benennen, speichern und später wiederverwenden (Punkt 2). Zudem stellt das Programm die JSON-Daten mit Syntax-Highlighting ansprechend dar (Punkt 3) und man kann sie auf Knopfdruck formatieren, um eine übersichtliche Darstellung zu erhalten (Punkt 4). Zu guter Letzt gibt Insomnia auch noch die Antwort des Mikrocontrollers aus (Punkt 5), was bei Fehlern viel Zeit ersparen kann.

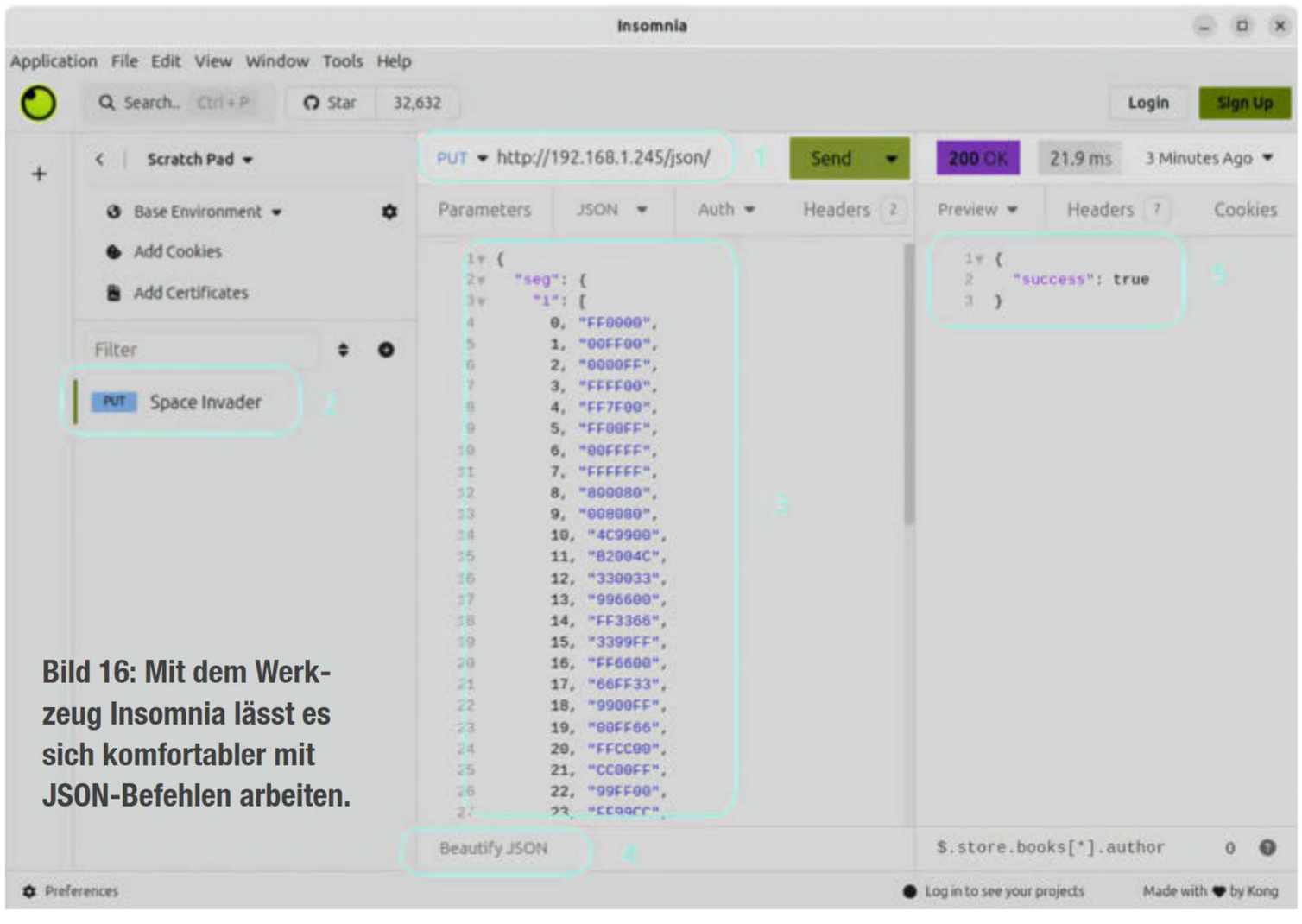


Bild 16: Mit dem Werkzeug Insomnia lässt es sich komfortabler mit JSON-Befehlen arbeiten.

Mit Python programmieren

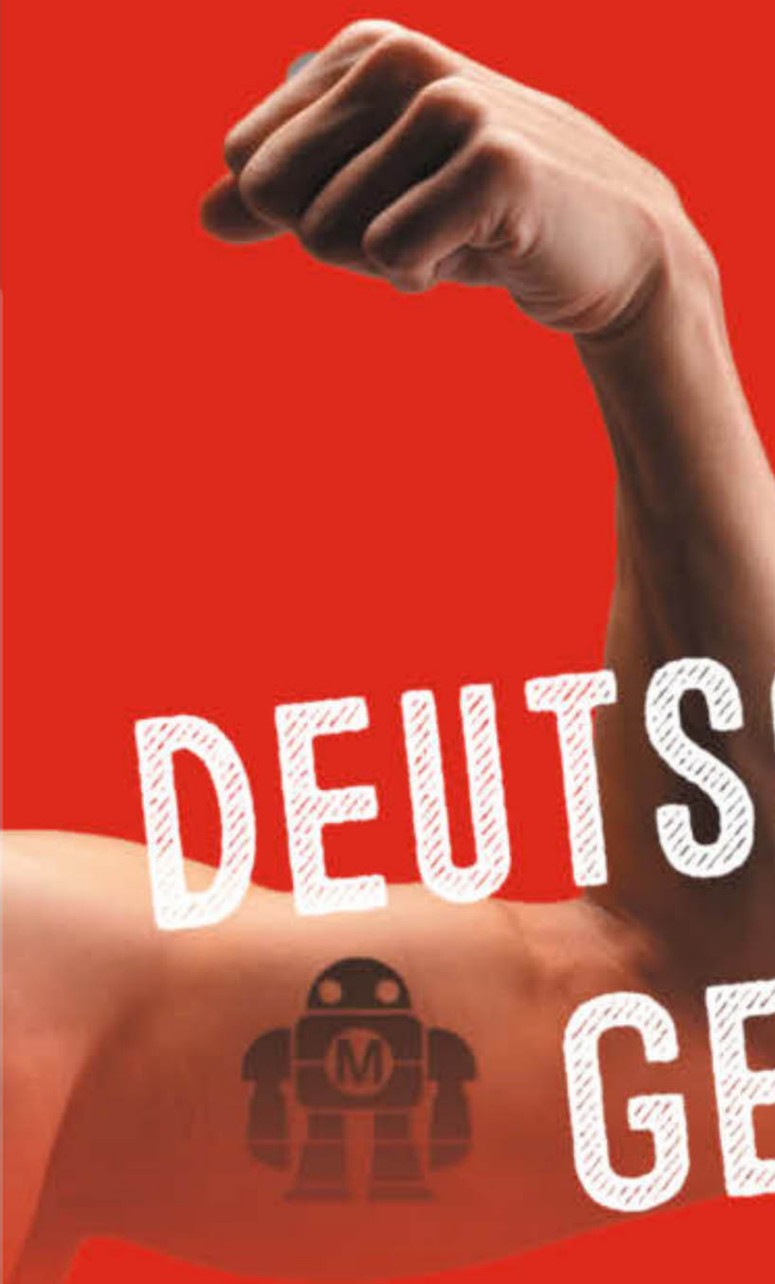
Wer noch einen Schritt weitergehen möchte, kann – entsprechende Programmierkenntnisse vorausgesetzt – die JSON-API natürlich auch per Code ansprechen. Das Listing wled.py zeigt ein einfaches Python-Programm, das alle LEDs auf einen zufälligen Farbwert setzt. So leuchtet die Pixel-Lampe jedes Mal unterschiedlich, sobald man das Programm ausführt.

Es lohnt sich

Mit diesem Wissen und dem Programmfragment gewappnet, sollte es nun ein Leichtes sein, der Pixel-Lampe einen individuellen Stempel aufzudrücken. Vielleicht lässt man ihn in den Farben seines Lieblingsvereins leuchten oder die Farbstimmung je nach Tageszeit und Wetter variieren.

Abschließend lässt sich sagen, dass das Projekt bisher bei Videotelefonaten im Homeoffice schon oft für reges Interesse und Gesprächsstoff gesorgt hat. Es lässt sich mit vertretbarem Zeitaufwand (und etwas Durchhaltevermögen beim Löten) umsetzen, ist ein echter Hingucker. *—akf*

Make:



DEUTSCHLANDS GEFÄHRLICHSTES ABO-ANGEBOT*

*VON DEUTSCHLANDS GEFÄHRLICHSTEM DIY-MAGAZIN (LAUT LESERN)

2× Make testen mit über 30 % Rabatt

Jetzt bestellen: make-magazin.de/abo-angebot

Warum eigentlich gefährlich?
Laut Lesern sind wir das "gefährlichste DIY-Magazin" Deutschlands. Das ist aber natürlich nur Spaß! Sie können unser Magazin ganz unbesorgt lesen und 2 Ausgaben als Heft + digital testen, zusätzlich erhalten Sie ein Geschenk Ihrer Wahl – klingt doch eigentlich ganz ungefährlich.

infoservice@maker-media.de +49 541/80 009 125 make-magazin.de/abo-angebot

Der Brewintosh

Ein originaler Macintosh Plus konnte aufgrund schwer zu beschaffender Ersatzteile nicht ohne Modifikationen zum Laufen gebracht werden. Für die Übergangszeit wurde ein originalgetreuer Macintosh Plus aus dem 3-D-Drucker entwickelt. Dieses Projekt stellte den Maker vor vielfältige Herausforderungen, um dem Original in jeder Hinsicht gerecht zu werden.

von Kevin Noki

Zuerst wurde jeder einzelne Bereich und Winkel des Macintosh Plus sorgfältig vermessen. Diese Messungen übertrug ich in ein CAD-Programm, um das 3D-Modell zu erstellen. Für den Druck nutze ich einen modifizierten Ender-3, mit dem ich das Gehäuse in nur vier Teilen drucken konnte. Nach dem Druck trug ich Kunststoffspachtel auf und schliff die Oberflächen glatt.

Für die Montage der Gehäuseteile setzte ich Metallpins in vorgefertigte Löcher und verklebte mit dickflüssigem Sekundenkleber. Jedes Teil wurde wiederholt geschliffen, um eine perfekte Oberfläche zu erzielen. Nachdem die äußeren Teile vorbereitet waren, grundierte und lackierte ich sie anschließend in der charakteristischen beigen Farbe des Macintosh Plus. Einige Teile des Gehäuses sollten ein mattes Finish bekommen. Um die glänzenden Teile zu schützen, wurden sie mit auf dem Schneidplotter gefertigten Folien maskiert.

Für das Innenleben zerlegte ich einen älteren 10-Zoll-Bildschirm. Die CCFL-Röhre ersetzte ich durch LED-Streifen, gesteuert von einem Dimmermodul, damit ist die Hintergrundbeleuchtung wie beim Original mit einem Drehknopf regelbar. Für die Stromversorgung sorgt ein modifiziertes Laptop-Netzteil sowie ein USB-Hub mit Netzteil. WAGO-Verbinder wurden verwendet, um alle Komponenten an die Kaltgerätebuchse anzuschließen.

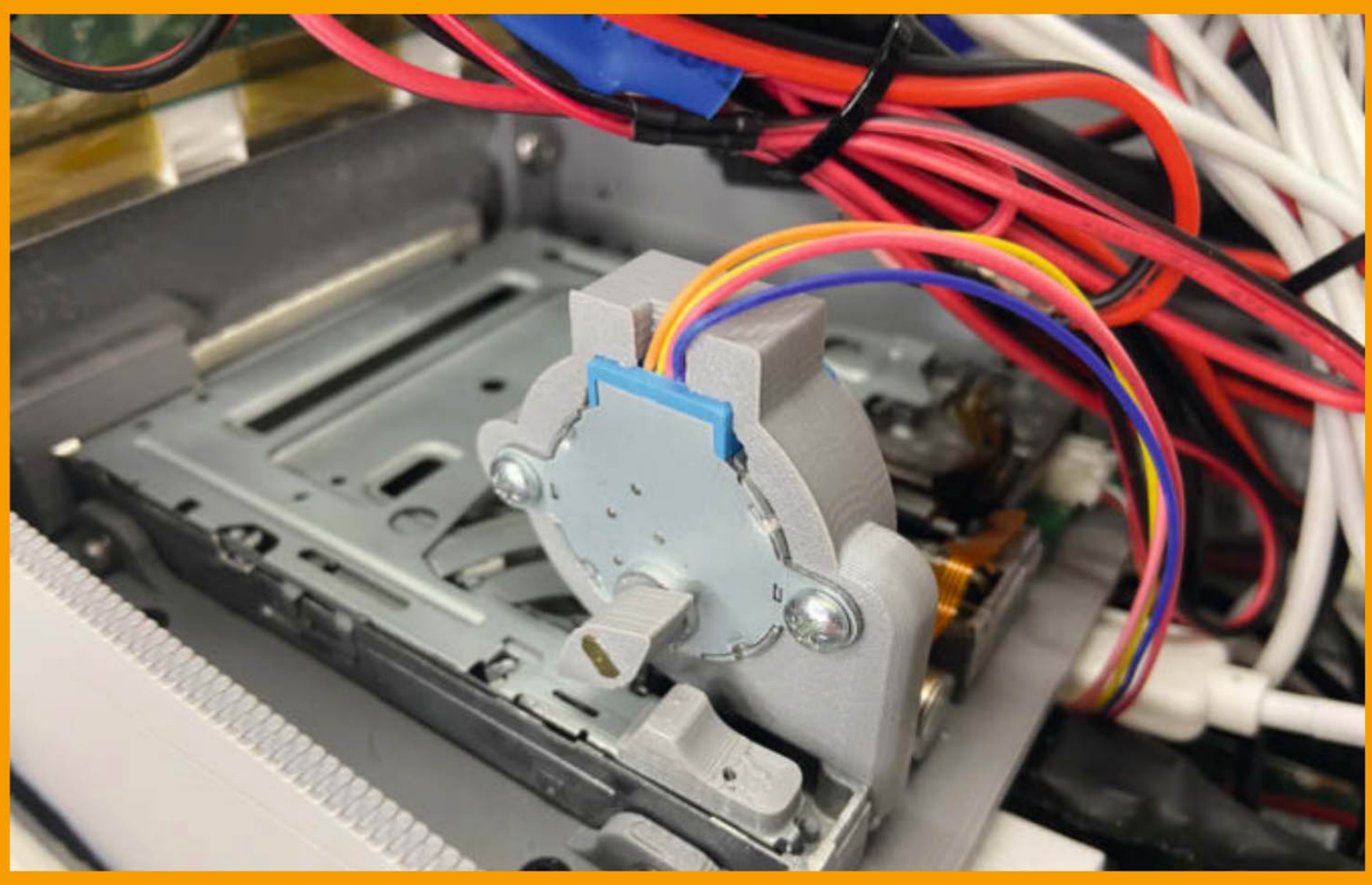
Der Anschluss von Originaltastatur und -maus wurde mithilfe eines Teensy Entwicklungsboards und TMK-Keyboard-Firmware verwirklicht. Der Macintosh-Startton wird über einen eingebauten Lautsprecher abgespielt. Eine weitere Herausforderung bestand darin, das USB-Floppy-Laufwerk mit einem vom Arduino gesteuerten Schrittmotor auszustatten, der es ermöglicht, Disketten softwaregesteuert auszuwerfen.

Für den letzten Schliff wurden Aufkleber und Schriftzüge gestaltet. Das 3-D-Logo entstand, indem ich es mit einer Schicht UV-Resin überzog. Der Brewintosh-Schriftzug ist auf einem Resin-Drucker gedruckt und wurde dann in einem Laminiergerät mit silberner Folie versehen. Der Brewintosh läuft auf dem Open-Source-Emulator „mini vMac“ mit dem originalen Betriebssystem von damals. Durch Anpassungen am Code des Emulators kann von HFS-formatierten Disketten gebootet und gelesen werden. Eine SD-Karte dient zum Datentausch mit der moderneren Welt.

Warum Brewintosh? Es ist eine Kombination aus „Homebrew“ (engl. Eigenbräu) und „Macintosh“. In den 70er-Jahren war der „Homebrew Computer Club“ in Kalifornien ein Verein von Bastlern, die ihre eigenen Computer gebaut und in wöchentlichen Treffen präsentiert haben: darunter auch Steve Jobs und Stephen Wozniak mit ihrem Apple-1. —*caw*

▶ youtu.be/7N9oz4Ylzm4

Das fertige Innere des Brewintosh ist fast so komplex wie das Original.



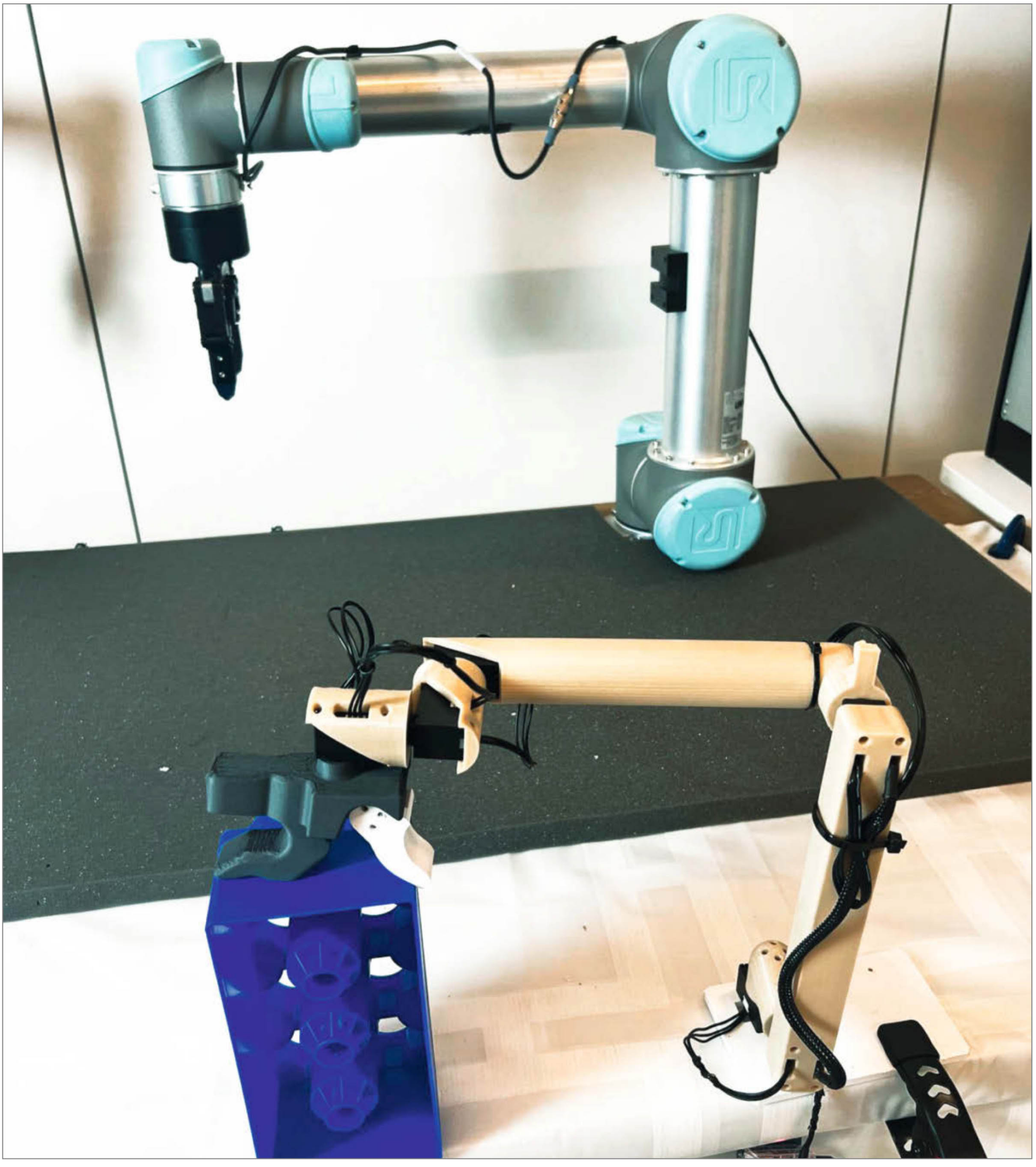
Der vom Emulator gesteuerte, automatische Disketten-Auswurf per Schrittmotor.

Liebe zum Detail auch auf der Rückseite: Aufwändig gefertigte Aufkleber

Roboterarme intuitiv steuern

Robotern komplexe Bewegungsabläufe beizubringen, erfordert meist selbst viel Übung und ist kostspielig. Das Projekt GELLO vereinfacht diesen Prozess und macht ihn für jeden zugänglich.

von Ákos Fodor



Alle Bilder: Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin and Pieter Abbeel (University of California, Berkeley)

Ein Team an der University of California in Berkeley hat eine Fernsteuerung für Telepräsenz-Roboterarme entwickelt. Sie nennt sich GELLO (General, Low-Cost, and Intuitive Teleoperation Framework for Robot Manipulation) und sieht aus wie eine verkleinerte Version der zu steuernden Arme. Mit dem Design lassen sich selbst komplexe Abläufe intuitiv auf den Roboter übertragen, denn dieser bewegt sich genau wie der menschliche Arm, der ihn lenkt.

Wie gut das System funktioniert, haben die Entwickler in einer Versuchsreihe getestet, bei der die Teilnehmer ohne technische Anleitung Aufgaben absolvieren sollten. Dazu gehörte etwa, ein Toast aus einer Verpackung zu holen und zu toasten oder ein Tuch zusammenzufalten.

Die Roboterarme durften dabei nicht aneinanderstoßen, sonst galt der Test als fehlgeschlagen. Verglichen mit anderen Eingabemethoden (VR-Controllern und 3D-Mäusen), gelangten die Probanden nicht nur schneller zum Ziel, sondern machten auch weniger Fehler.

GELLO ist Open Source und lässt sich mithilfe eines 3D-Druckers und DYNAMIXEL-Motoren (XL330-M288-T für den Arm und XL330-M077-T für den Greifer) zusammenbauen. Zurzeit unterstützt das Projekt folgende Roboterarme: The Franka (Franka Robotics), UR5 (Universal Robotics), xArm7 und Lite6 (uFactory) und AR4 (Annin Robotics). Für diese Modelle bieten die GELLO-Entwickler herunterladbare 3D-Daten sowie eine passende Software auf GitHub an.

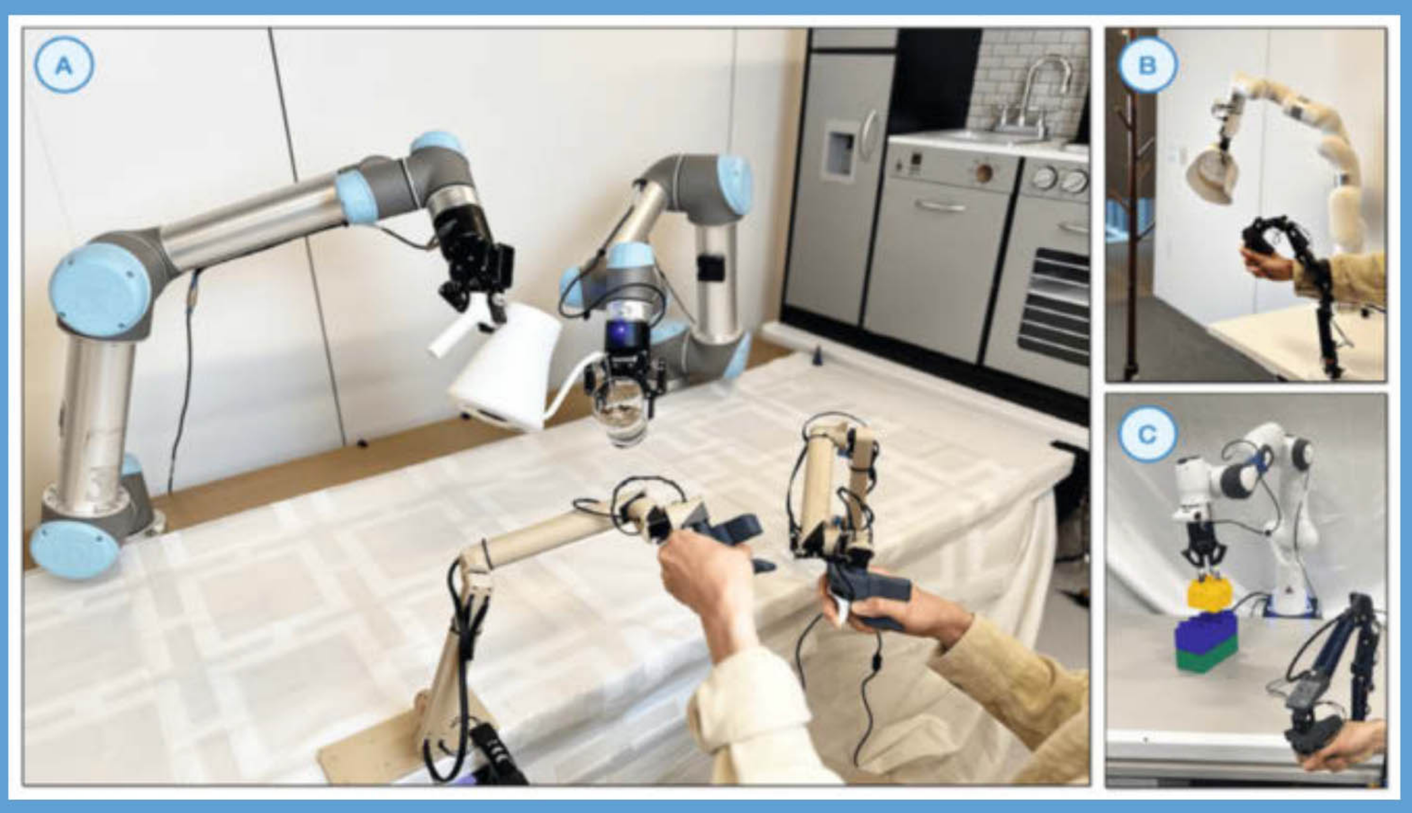
Der Roboterarm und die Steuereinheit kommunizieren über einen Computer. Wie man die Python-Software mithilfe von Docker installiert und konfiguriert, wird Schritt für Schritt erklärt. Für einen ersten Test lässt sich das System sogar mit einem virtuellen Roboterarm simulieren.

Die Entwickler heben in ihrem Paper immer wieder den günstigen Preis des GELLO als Feature hervor. Tatsächlich lässt sich das Gerät für ca. 300 Euro nachbauen – für einen der unterstützten Roboterarme sollte man aber noch einen fünfstelligen Betrag bereithalten.

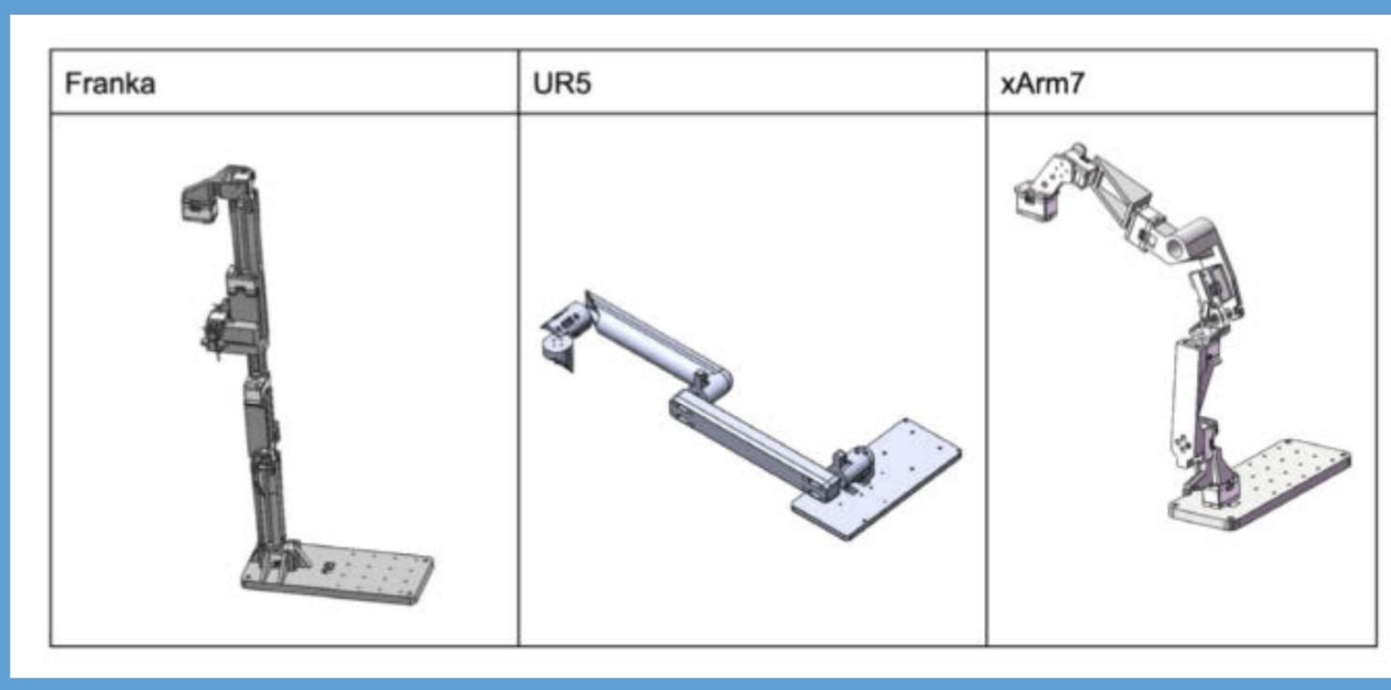
Allerdings lässt sich GELLO auch mit anderen Roboterarmen verwenden, sofern sie dem Aufbau der teuren Modelle stark genug ähneln. Oder man macht es wie der GitHub-User Alexander Koch: Mit seinem low_cost_robot hat er das GELLO-System so weit vereinfacht, dass man für knapp 400 Euro sowohl einen ausführenden als auch einen steuernden Roboterarm bauen kann. Wie ein Video zeigt, sind zwei davon wie ihre großen Vorbilder auch imstande, Kleidung zu falten.

Mehr Informationen zu GELLO und dem low_cost_robot findet ihr über den folgenden Link. —*akf*

▶ **make-magazin.de/xw9j**



Die Praktikabilität von GELLO auf dem Prüfstand: Tee zubereiten (A), einen Hut aufsetzen (B) und Klötzchen stapeln (C).



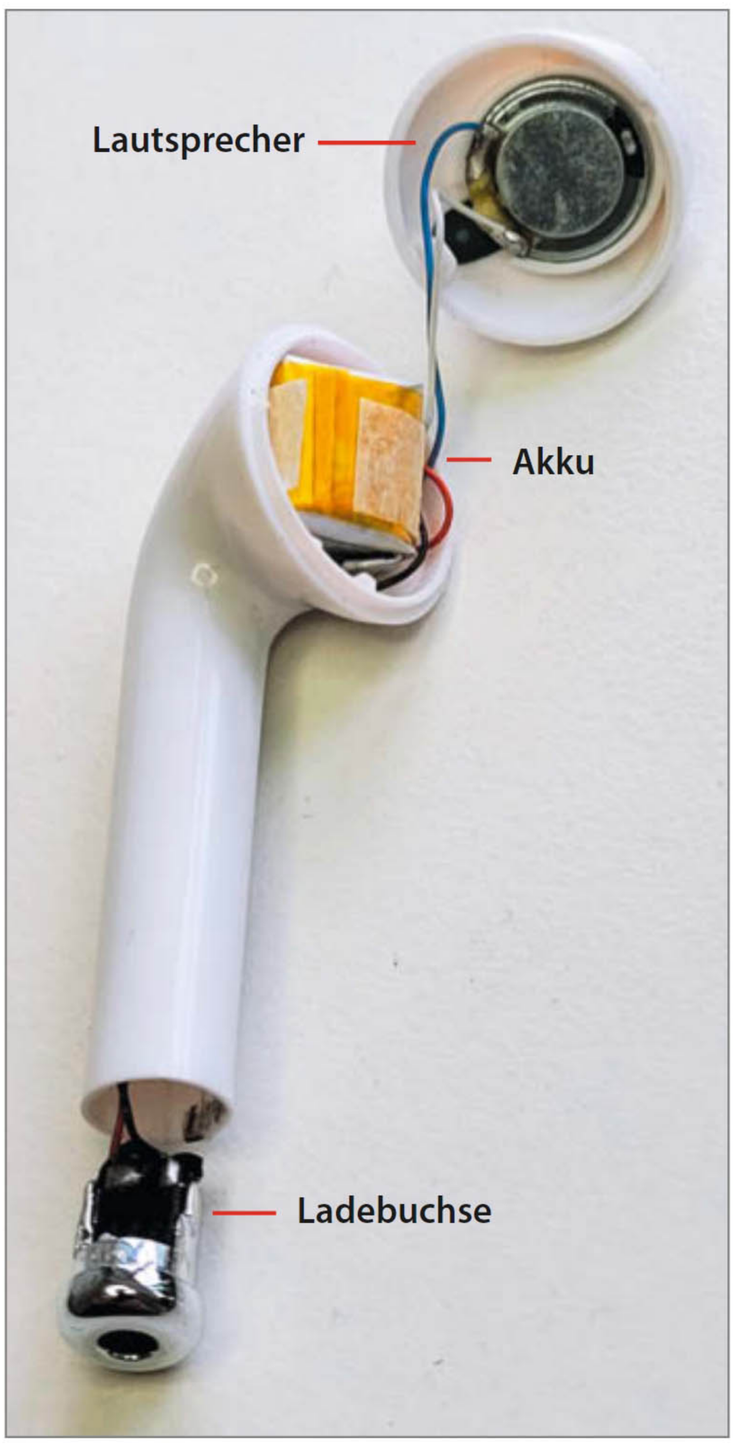
Für GELLO haben die Entwickler verschiedene kommerzielle Roboterarme miniaturisiert, die sich komplett 3D-drucken lassen.

Über die frei verfügbare Software lassen sich die Roboterarme konfigurieren und steuern.

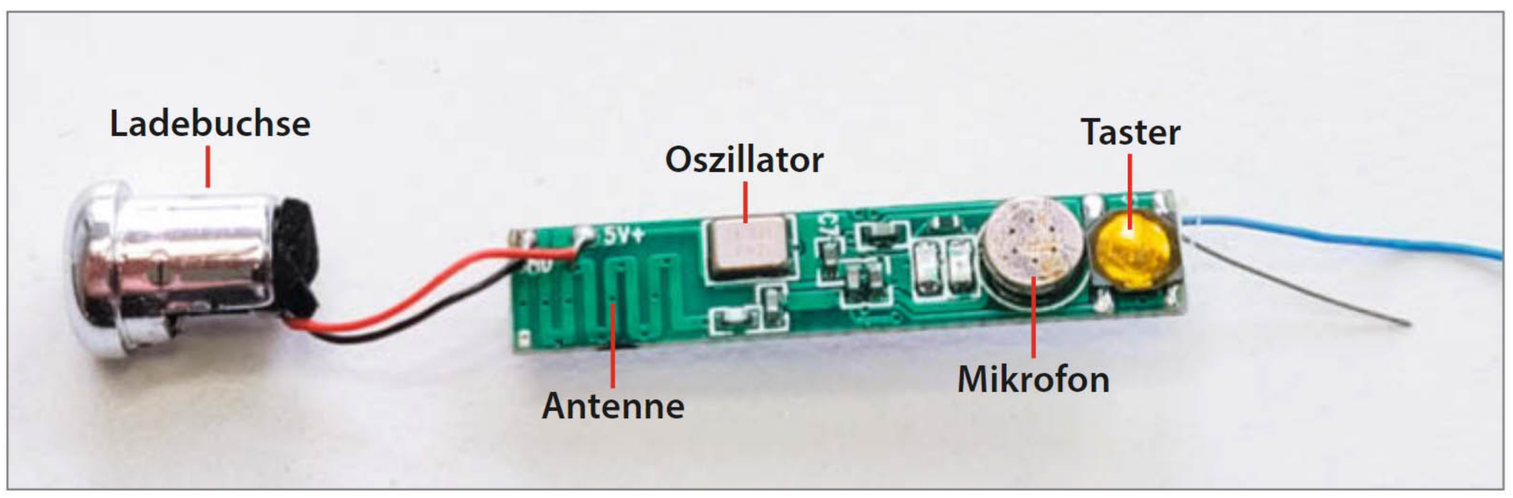
True-Wireless-Kopfhörer

Wireless Earbuds haben sich im Stadtbild durchgesetzt. Kaum jemand rennt noch verkabelt durch die Straßen und klassische Over-Ear-Kopfhörer dienen eher zum Aufhübschen des Outfits. Um kabellos Musik über die kleinen Stöpsel ins Ohr zu übertragen, muss man Technik auf kleinstem Raum unterbringen. Wir haben uns das mal (unter der Lupe) angeschaut.

von Daniel Bachfeld



Die eine Seite der Platine trägt den zentralen SoC für Bluetooth- und Audio-Funktionen. Rechts am Ende sind die Anschlüsse für den Lautsprecher und den Akku zu sehen.



Die andere Seite der Platine enthält das Mikrofon, einen Taster und einen Oszillator für den SoC.

Die True-Wireless-Kopfhörer des Anbieters Denver (Kostenpunkt 10 Euro) koppelt man per Bluetooth 5.0 mit einem PC, dem Smartphone oder einem Tablet. Rund drei Stunden sollen die Winzig-Akkus mit ihren 40 mAh durchhalten, um die Buds anzutreiben. Die Miniaturlautsprecher leisten 8 dBm, was nur 6,4 mW entspricht. Klingt wenig, direkt im Hörgang reicht es aber für die meisten Menschen vollkommen aus, um Musik laut abzuspielen.

Über das integrierte Mikro kann man aufnehmen, etwa um hands-free zu telefonieren oder an einer Videokonferenz teilzunehmen. Die im Kopfhörer verbaute Leiterplatte trägt einen Bluetooth-Transceiver mit integriertem Verstärker sowie eine Akku-Ladeschaltung. Nach dem Anschalten koppeln sich die zwei Ohrhörer zunächst untereinander. Nur einer der Ohrhörer stellt dann zu anderen Geräten eine Verbindung her, um das Pairing anzustoßen. Die rote und die blaue LED links neben dem Mikrofon signalisieren dabei den Status. Über die Minitaster kann man die Musikwiedergabe auf dem Abspielgerät stoppen und starten oder Anrufe entgegennehmen.

Die Ladestation nimmt beide Hörer auf und lädt sie über die Buchsen. In der Station selbst sind eine Ladeelektronik und ein Akku mit 200 mAh enthalten. Über die verwendeten ICs in den Hörern und der Ladestation gib es keine Datenblätter. Die bestückten Platinen sollten sich aber dennoch leicht in eigene Projekte integrieren lassen, wenn man denn der Billig-Buds überdrüssig geworden ist. —*dab*

▶ make-magazin.de/x2m3

Der Akku ist kleiner als ein Fisherman's Friend, liefert aber vermutlich mehr Energie. Ich werde ihn in eigenen LoRaWAN-Experimenten einsetzen.

Die Ladestation arbeitet auch mobil: Ein 200-mAh-Akku kann die Earbuds unterwegs laden. Der Original-Akku macht schon dicke Backen!

Elektroniklabor für den Küchentisch

Dieser Koffer vereint alles, was man für kleinere Elektronikprojekte oder -reparaturen benötigt. Einfach aufklappen und loslegen – Labornetzteil, Multimeter, Lötkolben und Co sind direkt einsatzbereit. Ein Bauvorschlag zum Anpassen.

von Johannes Börnsen

Die LED-beleuchteten Herdplatten unserer Spielküche leuchteten nicht mehr. Ich kramte also den Karton mit Multimeter, Universalnetzteil und allerlei Handwerkzeug wie Abisolierzange, Schrumpfschläuchen und Co hervor und verteilte alles auf dem Küchentisch, ehe ich mit der eigentlichen Fehlersuche beginnen konnte. Während ich noch eine Mehrfachsteckdose suchte, um das Netzteil und den Lötkolben betreiben zu können, grübelte ich über eine Lösung, die mir derartige Aufbauzeit in Zukunft ersparen würde. So entstand die Idee für dieses mobile Elektroniklabor im Koffer, in dem alles Platz hat, was ich für kleinere Reparaturen und Basteleien benötige.

Ich hab' noch einen Koffer im Keller

Zunächst begab ich mich auf die Suche nach einem geeigneten Gehäuse. Inspiriert vom Experimentierkoffer meines Kollegen Carsten Wartmann, der dafür einen alten Mikrofonkoffer mit einem Breadboard, Arduino, Jumperkabeln und allerlei Schaltern und Anschlüssen ausgebaut hat (siehe Artikel in Make 02/2023), schaute ich zunächst meine Sammlung ehemaliger Werkzeugkoffer durch. So richtig gut wollte davon jedoch nichts passen. Auch eine Suche im Netz brachte nicht das, was ich im Kopf hatte. Also entschied ich mich dazu, selbst einen passenden Koffer zu bauen.

Vollgepackt mit guten Sachen

Labornetzteil und Multimeter wollte ich im oberen Bereich verbauen. Dadurch sind sie in Sichthöhe und können dauerhaft verkabelt bleiben. Das bedeutet aber auch, dass die beiden Geräte möglichst flach sein sollten, damit das obere Fach nicht zu groß wird. Beim Labornetzteil entschied ich mich für ein Gerät von Ruzizao. Es liefert maximal 30 V und 5 A. Das reicht für meine Zwecke aus, zufrieden bin ich mit dieser Wahl aber nicht ganz. Es passt zwar in das obere Fach, der Lüfter läuft jedoch dauerhaft und die Einstellungen sind umständlich vorzunehmen, weil das Display die Werte ausblendet, solange an den Knöpfen gedreht wird.

Das Multimeter ist ein Owon XMD1041, was bisher seine Aufgaben klaglos erledigt hat und ehrlicherweise mehr kann, als ich nutzen werde. Aber es passt vom Formfaktor und ist nicht auf Batterien angewiesen. Es ist nur wenige Zentimeter tief, sodass die nötige Verkabelung hinter dem Multimeter stattfinden konnte. Neben diesen beiden Geräten habe ich eine kleine Schublade verbaut, in der sowohl die passenden Kabel für Netzgerät und Multimeter als auch ein kleines Akku-Oszilloskop seinen Platz finden.

Die Stromverteilung für den ganzen Koffer befindet sich hinter dem Multimeter. Es ist nur wenige Zentimeter tief und mit einem Lochblechstreifen gesichert.

Weniger ist mehr

Die erste Version meines Koffers wurde deutlich zu breit. Ich hatte sie an eine 60 Zentimeter lange Lampe angepasst, die ich ebenfalls verbauen wollte. Das hätte auch gut zum geplanten Standort des Koffers gepasst. Jedoch war der Rohbau schon ohne Innenleben so schwer und sperrig, dass ich seine Breite kurzerhand auf 45 Zentimeter reduzierte. Das reicht gerade aus, um Netzteil, Multimeter und Oszi-Schublade nebeneinander anzuordnen. Auch so ist der Koffer kein Leichtgewicht. Da er bei mir in der Regel jedoch keine größeren Strecken zurücklegen muss, sondern in der Werkstatt direkt neben der Werkbank wohnen und bei Bedarf nur auf die Werkbank gehoben wird, macht das nichts.

Wenn der Koffer mobiler sein soll, würde ich statt der vergleichsweise schweren Multiplex-Bauweise doch eher einen fertigen Kunststoffkoffer nehmen oder einen Selbstbau aus Tischlerplatten oder Pappelsperrholz überlegen. Auch mit einem Steckernetzteil mit einstellbarer Spannung und einem mobilen Multimeter ließe sich einiges an Gewicht sparen. Eine bemaßte Skizze meines Koffers finden Sie über die Short-URL am Ende des

Eine kleine Schublade enthält nicht nur ein einfaches Oszilloskop, sondern auch sämtliche Kabel und Zubehör für die Geräte des oberen Fachs.

Handwerkzeug und Lötkolben finden im unteren Teil Platz. Zubehör und Verbrauchsmaterial lagern in kleinen Schubladen. Die hinter Riffelblech versteckte Leuchte lässt sich nach vorne schwenken …

Artikels. Dahinter verstecken sich außerdem auch Links zu den von mir verwendeten Komponenten.

Im unteren Teil des Koffers lagern Werkzeug und Verbrauchsmaterial. Neben Spitz- und Kombizange finden dort eine Crimp- und eine Abisolierzange, Cutter, Schere, Bleistift, Filzmarker, ein Pinzettenset und ein Entlötkolben in zwei herausnehmbaren Boxen Platz. Dahinter versteckt sich eine Mehrfachsteckdose mit USB-Anschlüssen, um beispielsweise einen Arduino unkompliziert mit Strom versorgen zu können.

Lötkolbenupgrade

Zunächst hatte ich nur einen einfachen Lötkolben mit digitaler Temperaturregelung in den Koffer gelegt. Der Unterschied zur Lötstation war mir nach den ersten Einsätzen aber doch zu gravierend, sodass ich einige der Kleinteilfächer wieder ausgebaut habe und nun eine kleine Lötstation mit im Koffer steht. Für diese habe ich einen eigenen Lötkolbenhalter mit eingebauter Absaugung gedruckt. Ein Video dazu finden Sie auf unserem YouTube-Kanal über die Short-URL am Artikelende.

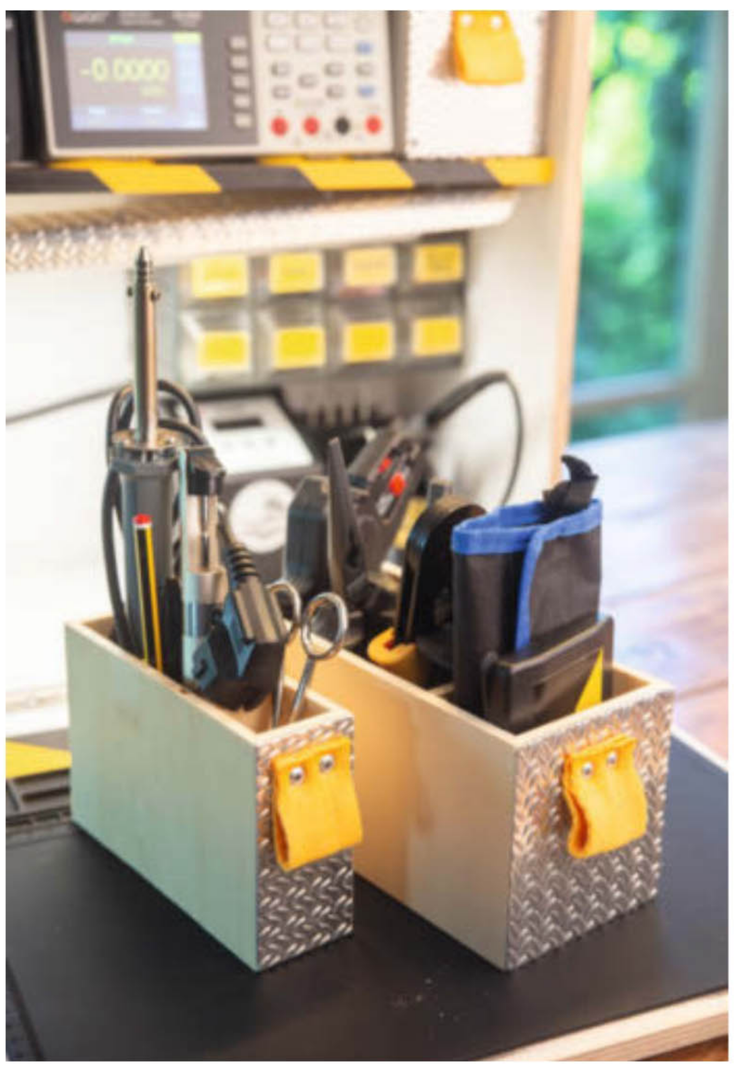
In den Kleinteilfächern finden sich häufig benötigte Dinge wie Schrumpfschlauch, Entlötlitze, Flussmittel, ein paar Elektroklemmen, Steckernetzteiladapter und oft benutze USB-Kabel. Außerdem habe ich zwei verschiedene Schraubenziehersets mit im Koffer: Für größere Schrauben gibt es ein Set mit VDE-Schraubenzieher-Aufsätzen von Wiha, die sich in einen Handgriff stecken lassen. Für kleine Schrauben ist ein Set von iFixit praktisch, beispielsweise um Smartphone-Gehäuse zu öffnen.

Bitte anpassen!

Dieser Koffer ist natürlich auf meinen Einsatzzweck hin optimiert. Das Gewicht ist nicht zu verachten und die Werkzeugauswahl vermutlich für Ihre Projekte nicht passend. Die grundsätzliche Idee, für einen bestimmten Fall wie die Reparatur des eingangs erwähnten Kochfeldes alle benötigten Werkzeuge in einer Box so anzuordnen, dass sie direkt einsatzbereit sind, hat sich jedoch definitiv bewährt. Apropos Kochfeld: Offenbar war Flüssigkeit über die Schalter gelaufen, sodass sie komplett verrostet waren. Das restliche Kochfeld auf Funktion zu überprüfen, die alten Schalter aus- und neue einzulöten war mithilfe des Koffers schnell erledigt. —*jom*

▶ **make-magazin.de/xbnr**

… um den Arbeitsplatz zu beleuchten. Die Lötmatte hat nummerierte Vertiefungen für kleine Bauteile. Legt man Schrauben bei der Demontage eines defekten Gerätes in der richtigen Reihenfolge ab, erleichtert das den anschließenden Zusammenbau.



Herausnehmbare Boxen platzieren die Werkzeuge in Reichweite.

Ein Veranstaltungsformat der Make: make-magazin.de

Maker Faire®

Das Format für Innovation & Macherkultur

Die nächsten Events

Maker Faire®
7. Juni 2024
WUPPERTAL

HANNOVER
Maker Faire®
17.-18. August 2024

Maker Faire®
8. SEPT 2024
SCHWEDT/O.
UCKERMARK

Maker Faire®
12. Oktober 2024
SINDELFINGEN

Maker Faire®
9. November 2024
SALZBURG

... weitere folgen.

maker-faire.de

Optimal gravieren mit Lightburn

Ob Hightech-Kunst oder nur einfaches Markieren: Bilder mit dem Lasercutter auf Holz, Kunststoff, Glas oder anderes Material zu gravieren, ist zwar je nach Vorlage etwas kniffelig, aber mit dem Wissen aus diesem Workshop kommen Sie sicher zum Ziel.

von Heinz Behling

Lasercutter können nicht nur schneiden, sondern auch hervorragend gravieren. Bilder, Beschriftungen und Symbole lassen sich damit recht eindrucksvoll in verschiedenste Materialien einbrennen. Die Helligkeitsunterschiede der Vorlagen werden dabei zu entsprechend mehr oder weniger starken Brandspuren. Voraussetzung ist aber, dass man die richtigen Einstellungen findet. Und die sind von der jeweiligen Maschine, dem benutzten Material (Holz, Plexiglas, Pappe usw.) und dem zu gravierendem Motiv abhängig.

Dieser Workshop zeigt Ihnen, was Sie in welcher Reihenfolge unternehmen müssen, um optimale Gravuren herzustellen. Zunächst einmal werden wir herausfinden, was Ihr Lasercutter kann. Dann geht es darum, welche Leistungs-Geschwindigkeits-Kombination für das jeweilige Material die beste ist. Und schließlich zeige ich Ihnen den Unterschied zwischen den Bildmodi, die die Lasersoftware Lightburn zur Umsetzung von Bildern in ein Lasergravur-Raster anbietet.

Eins vorweg: Sie werden Probematerial brauchen, um diverse Testgravuren durchzuführen, und zwar für jedes von Ihnen gewünschte Material. Reststücke reichen dabei völlig aus. Keine Angst: Haben Sie für einen Stoff erst mal die richtigen Werte herausgefunden, müssen Sie diese Tests künftig nicht erneut durchführen, solange die Materialqualität sich nicht wesentlich ändert. Einmal ermittelte Werte können in Lightburn nämlich in der (allerdings gut versteckten) Materialbibliothek gespeichert und dann bei Bedarf per Mausklick wieder abgerufen werden. Das gilt übrigens nicht nur fürs Gravieren, sondern auch fürs Lasercutten.

Kurzinfo

- **Lasercutter fürs Gravieren justieren**
- **Materialwerte ermitteln und in Materialbibliothek eintragen**
- **Ditheringverfahren vergleichen**
- **Graviertricks**

Material

- **Probe(Rest)-Stücke aller gewünschten Materialien**

Software

- **Lightburn**

Mehr zum Thema

- Heinz Behling: Material für Lasercutter, Make 2/24, S. 92
- Heinz Behling: Lasern mit Lightburn, Make 1/24, S. 24
- Heinz Behling: Autofokus nachrüsten, Make 2/24, S. 100

Alles zum Artikel im Web unter *make-magazin.de/xtmh*

Bild 1: So sehen Geisterbilder in Gravuren aus.

Laser-Check

Beim Gravieren wird der Laserkopf auf der X-Achse zeilenweise von links nach rechts geführt, anschließend um eine Zeilenhöhe (Linienintervall) auf der Y-Achse versetzt und dann für die nächste Zeile von rechts nach links bewegt. Die Intensität des Laserstrahles wird dabei gemäß der Helligkeit des entsprechenden Pixels im Vorlagebild gesteuert. Der Laser graviert also auf dem Hin- und Rückweg. Wichtig dabei ist, dass das Linienintervall der Breite des Laserstrahls entspricht, damit es nicht zu Überschneidungen (doppeltes Brennen) oder Zwischenräumen kommt.

Außerdem müssen die Links- / Rechts-Zeilen genau über den Rechts- / Links-Zeilen liegen. So sollte es idealerweise bei einem intakten Lasercutter sein – ist es aber manchmal nicht. Dann bekommt man als Ergebnis eine Gravur, in der alle Konturen doppelt vorhanden sind (ein sogenanntes Geisterbild, Bild 1). Ist der Versatz zwischen den beiden Zeilen nur gering, äußert sich das als Unschärfe in der fertigen Gravur, allerdings nur in horizontaler Richtung. Vertikal ist die Schärfe hingegen optimal. Diesem Problem widmen wir uns zuerst.

Dazu sind ein bis mehrere Testgravuren erforderlich. Über den Kurzinfo-Link können Sie ein Testbild herunterladen. Es enthält ein Quadrat mit einer Spirale (Bild 2).

Importieren Sie diese Datei in Lightburn. Die Größe des Bildes stellen Sie auf 50 × 50 mm ein. Als Laser-Parameter für mein Testmaterial (Finnpappe) habe ich bei meinem 40-W-CO^2-Laser 140 mm/s und 25 Prozent Leistung eingestellt. Je nach Gerät und Material sind hier andere Werte notwendig. Es ist dabei nur wichtig, dass man später das Testbild auf dem Material erkennen kann. In den Lasereinstellungen muss außerdem das bidirektionale Scannen eingeschaltet sein (Bild 3).

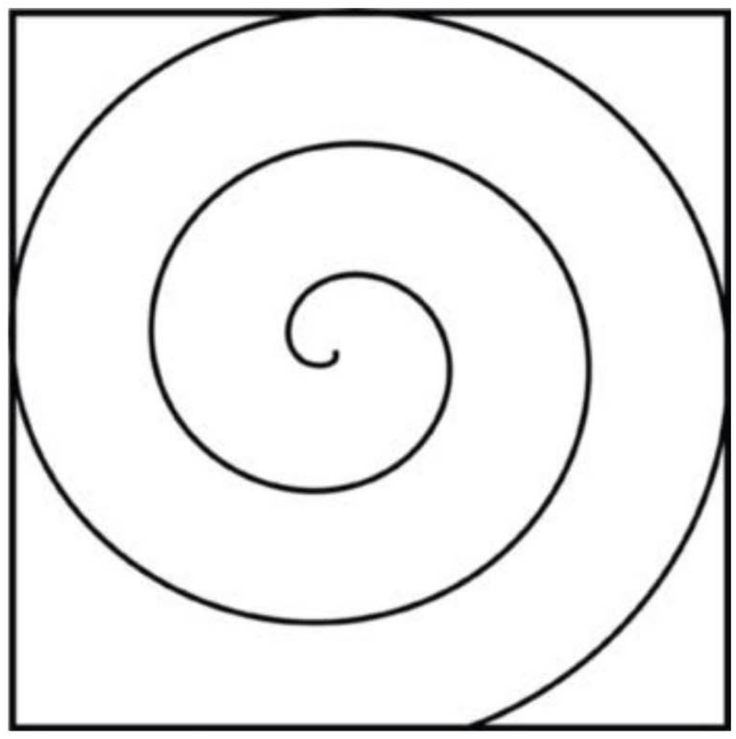


Bild 2: Mit diesem Testbild wird der Laser überprüft.

Gravieren Sie dann und schauen Sie sich das Ergebnis an. Sollte der Lasercutter den Gravierdienst verweigern und sich im Display mit der kryptischen Meldung „Nicht genug erw. Sper Bitte ESC“ beschweren, dann müssen Sie das zu gravierende Motiv anders platzieren. Damit der Laser eine Zeile richtig gravieren kann, muss der Laserkopf seine endgültige Geschwindigkeit bereits erreicht haben. Dafür braucht er vor der Zeile Platz zum Beschleunigen. Sie müssen also links der Gravur je nach Gerät etwa 5 bis 10 mm Platz lassen. Dasselbe gilt rechts der Gravur, denn da muss der Laserkopf abbremsen und in die Gegenrichtung beschleunigen. Positionieren Sie also im Falle dieser Meldung die Gravur gegenüber dem Nullpunkt des Laserkopfes entsprechend. Die Gravur sollte aussehen wie in Bild 4, mit gleich breiten horizontalen und vertikalen Linien; und vor allem mit nur einer(!) Spirale. In diesem Fall ist der Check bestanden und Sie können mit dem Ermitteln des Linienintervalls weitermachen.

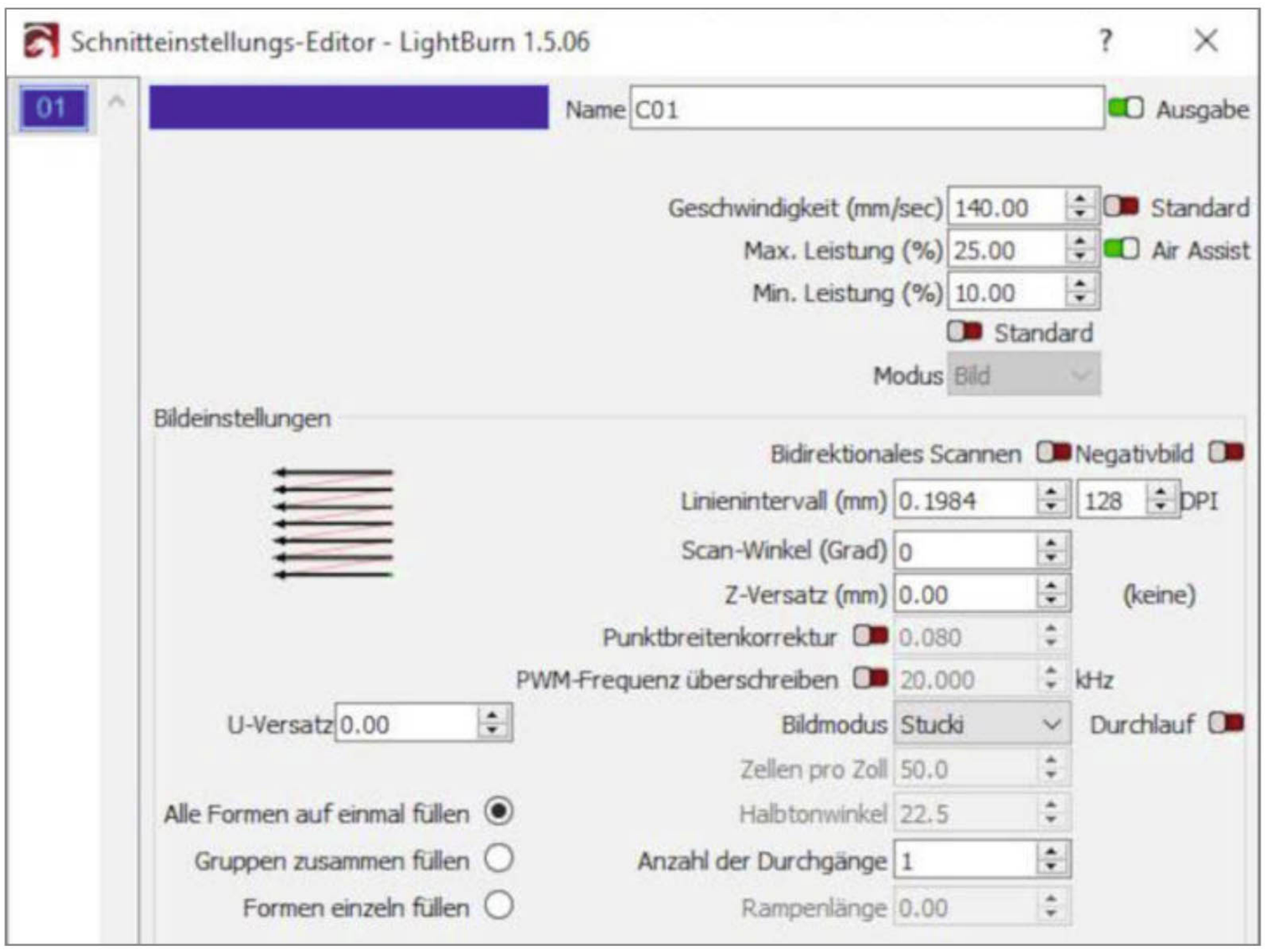


Bild 3: Achten Sie darauf, dass der Schieber neben „Bidirektionales Scannen“ grün ist.

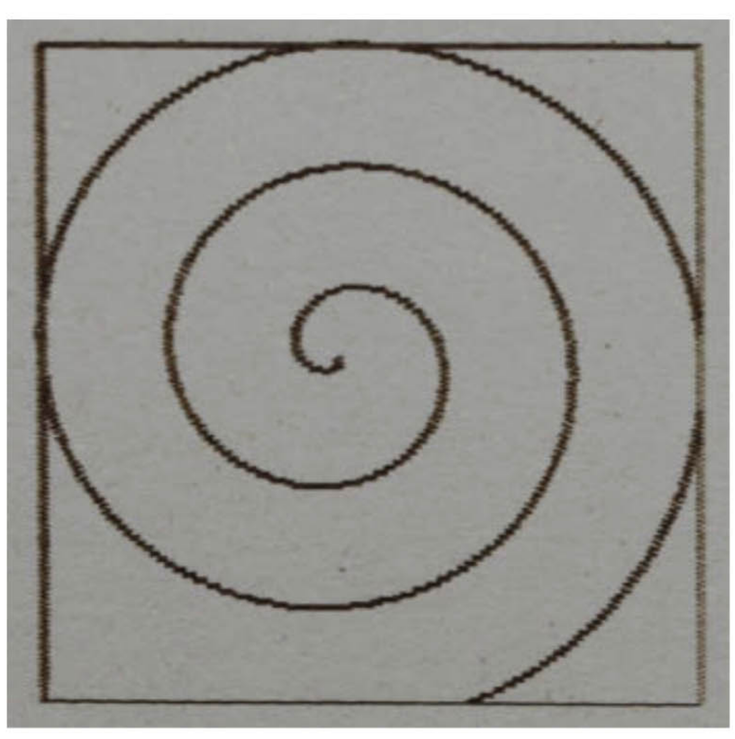


Bild 4: Ist mit dem Laser alles in Ordnung, erscheint die Spirale so.

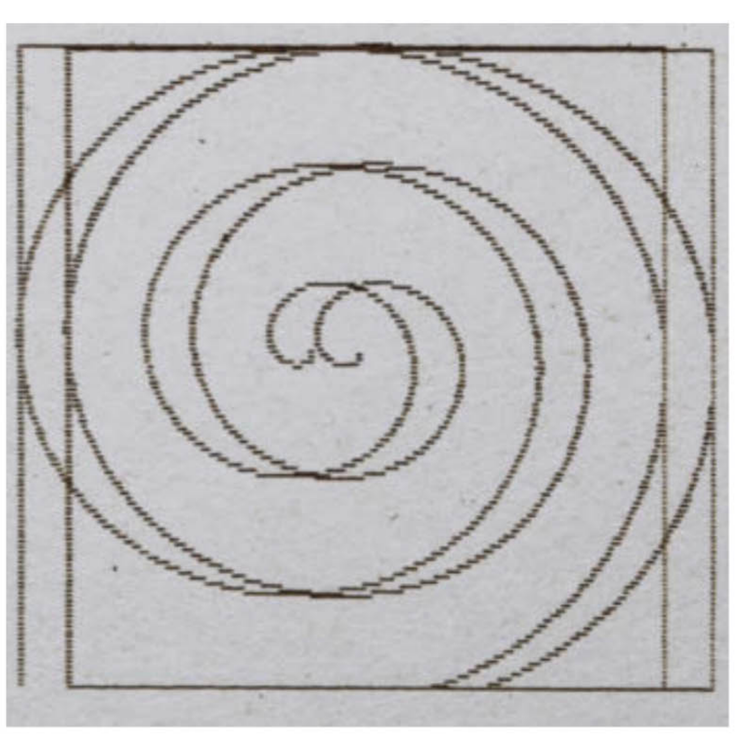


Bild 5: Hier sitzen die Bildzeilen nicht richtig untereinander. Das kann ein mechanischer Fehler sein.

Ähnelt das Ergebnis aber eher Bild 5, erscheinen also zwei Spiralen nebeneinander, hat Ihr Laser eventuell ein mechanisches Problem, das auf jeden Fall behoben werden muss. Oft ist zum Beispiel der Zahnriemen der X-Achse locker, also bitte nachspannen (siehe Handbuch des Herstellers). Es kann auch an einem lockeren Zahnrad auf der Achse des dazugehörenden Schrittmotors liegen (Schrauben mit Inbus-Schlüssel festdrehen). Falls das bei Ihrem Gerät zutrifft, beheben Sie diese Fehler. Wiederholen Sie dann die Gravur der Spirale. Ist dann alles in Ordnung, können Sie den Check abschließen und sich bereits der Ermittlung des Linienintervalls widmen.

Finden Sie beim Gerät jedoch keinen mechanischer Fehler, das Testbild wird aber trotzdem nicht korrekt graviert, liegt das Problem tiefer im System. Das kann ein Firmware-Fehler sein (Update) oder eine schlechte Einstellung der Ansteuerung der Treibermodule für die Schrittmotoren. Sie sollten aber nun nicht in der Firmware herumstochern. Das kann im schlimmsten Fall dazu führen, dass der Lasercutter jeglichen Dienst verweigert. Stattdessen bieten sich zwei Lösungen an: das bidirektionale Scannen beim Gravieren auszuschalten (dann dauert die Gravur zwar länger, enthält aber garantiert kein Geisterbild). Das ist in diesem Fall meine Empfehlung.

Oder Sie kompensieren den Fehler per Software, also in Lightburn. Allerdings ist das ein aufwendiges Verfahren, denn für jede Gravurgeschwindigkeit muss ein eigener Korrekturwert ermittelt werden. Lightburn versetzt dann künftig die Hin- und Rückzeile der Gravur

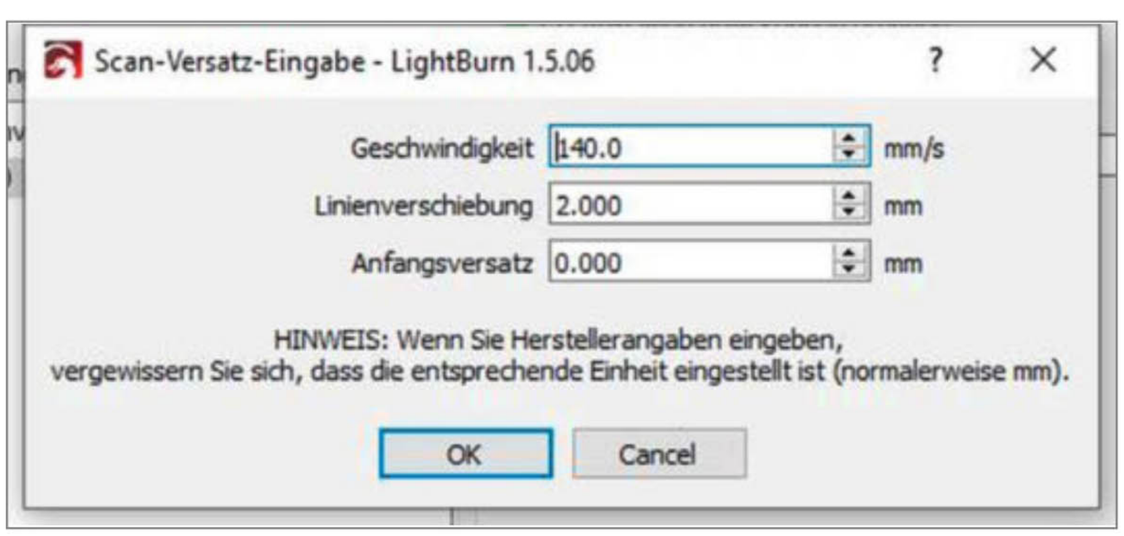


Bild 6: Der Scan-Versatz gleicht die Fehler des Lasercutters aus.

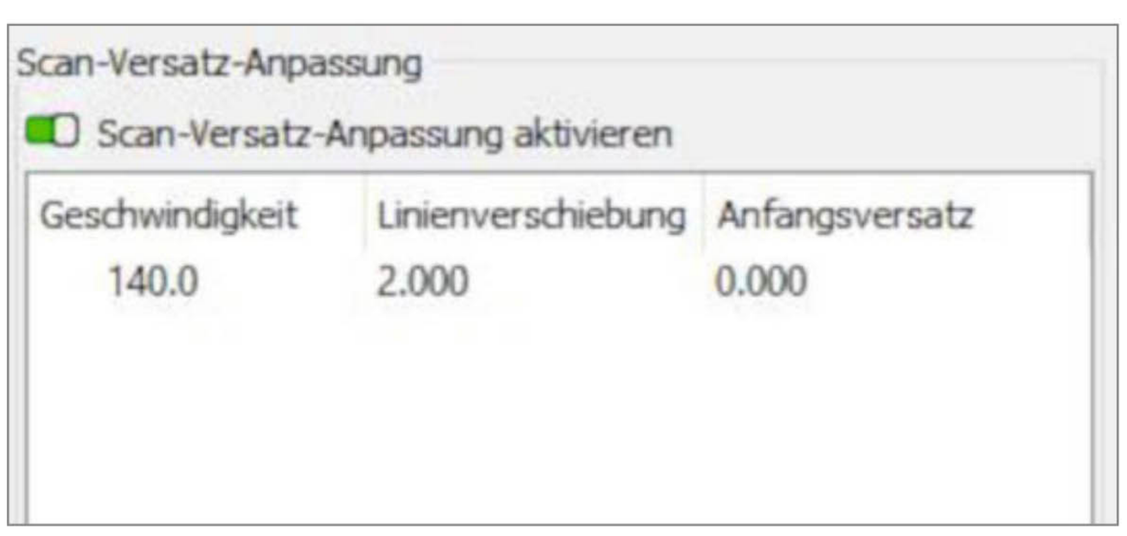


Bild 7: Ist dieser Schieber nicht grün, wird der Scan-Versatz nicht benutzt.

entsprechend in die zum Gerätefehler entgegengesetzte Richtung. Beides kompensiert sich gegenseitig und die Gravur ist wieder korrekt. Das Ermitteln dauert aber recht lange und kostet einiges an Testmaterial. Ein weiterer und meiner Meinung nach größerer Nachteil: Auf dem Display des Lasercutters wird das zu gravierende Bild künftig mit Geisterbild angezeigt. Falls Sie das aber für die doppelte Geschwindigkeit in Kauf nehmen möchten, zeige ich Ihnen, wie es geht.

Messen Sie dazu zunächst in der Testgravur den Abstand der beiden Teilbilder (am besten an den senkrechten Linien) möglichst genau aus (idealerweise mit einem Messschieber, zur Not geht ein Lineal auch). Sie erhalten dann zum Beispiel 4 mm Versatz für die beiden Bilder.

In Lightburn gehen Sie unter Bearbeiten in die Geräteeinstellungen. Unter Scan-Versatz-Anpassung klicken Sie auf Hinzufügen. Im folgenden Einstellungsfenster setzen Sie dann als Linienverschiebung die Hälfte des zuvor gemessenen Linienabstands ein (2 mm) und als Geschwindigkeit die beim vorherigen Gravieren benutzte, also in diesem Beispiel 140 mm/s (Bild 6).

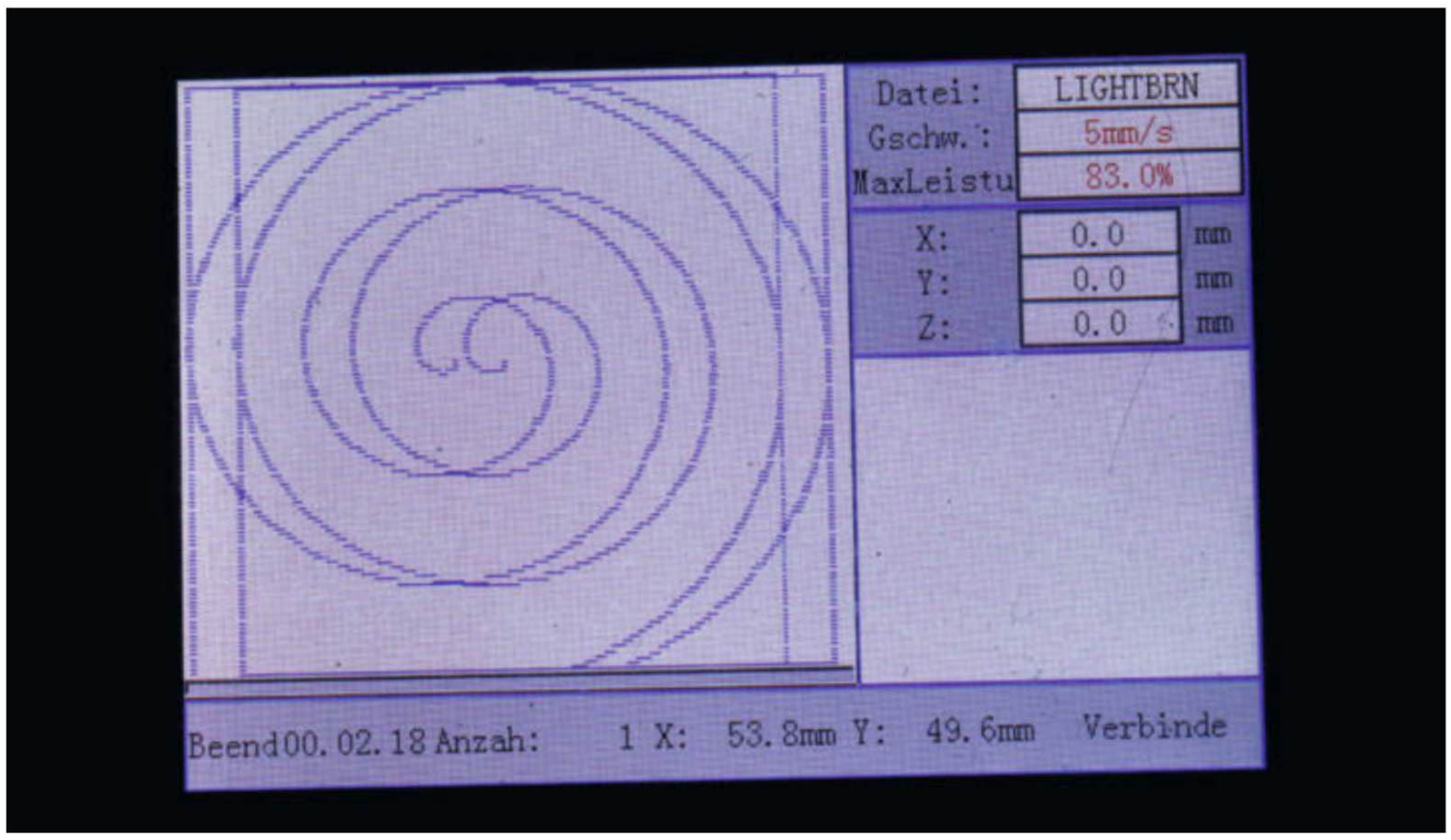


Bild 8: Sie sind nicht betrunken: Das Doppelbild erscheint durch die Fehlerkompensation.

Nach Ihrem O. K. müssen Sie den Scan-Versatz noch einschalten. Das geschieht mit dem Schieber (auf Grün stellen, Bild 7).

Jetzt müssen Sie die Testgravur mit der Spirale wiederholen. Auf dem Display des Lasercutters erkennen Sie sofort, dass die Gravur per Software korrigiert wurde, da das Gravurmotiv doppelt erscheint (Bild 8).

Ist der Fehler beseitigt, ist alles o. k., aber nur für diese Graviergeschwindigkeit. Ist der Abstand zumindest geringer geworden, müssen Sie den Scan-Versatz ein wenig variieren.

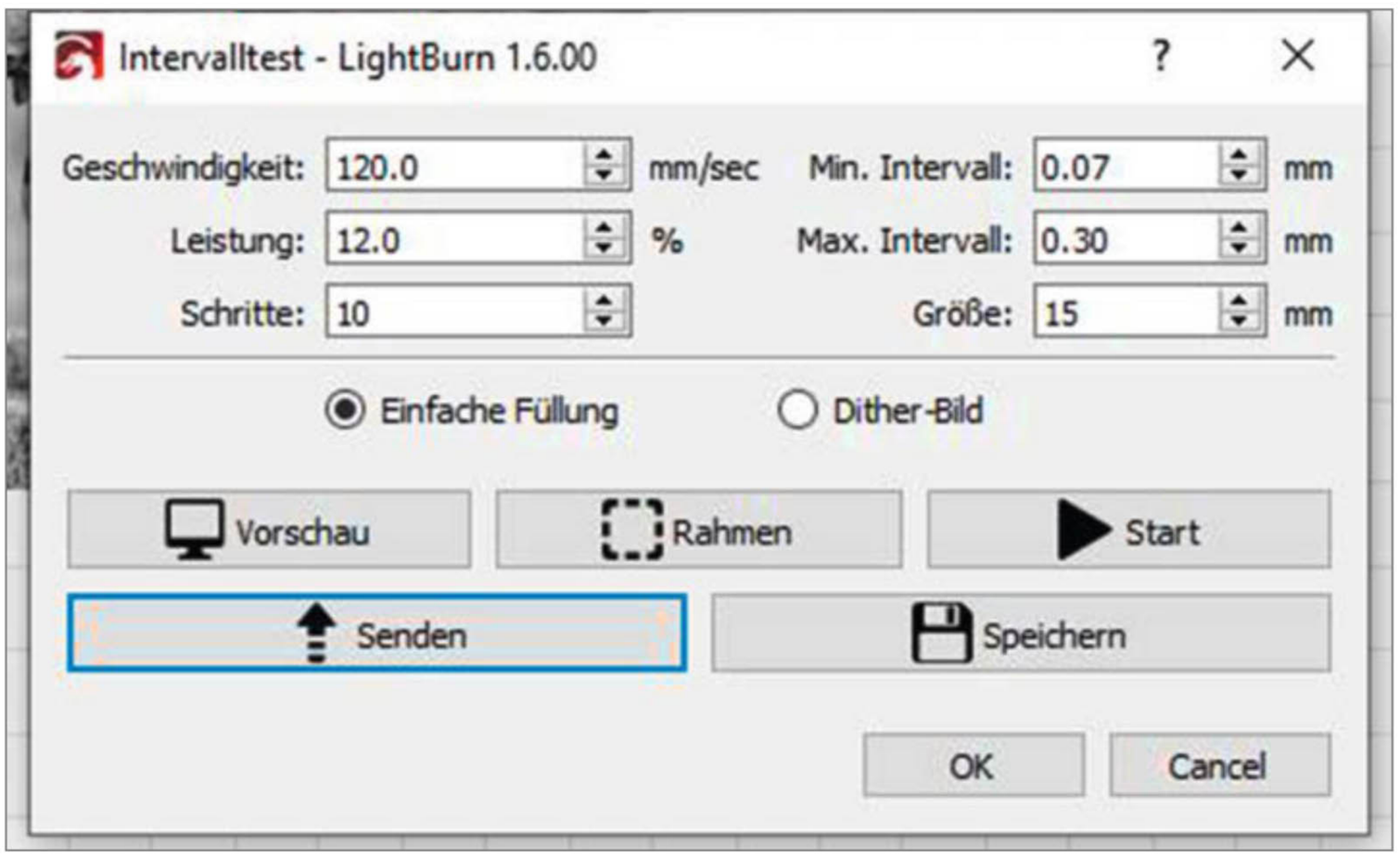


Bild 9: Mit diesem Tool ermitteln Sie den richtigen Zeilenabstand.

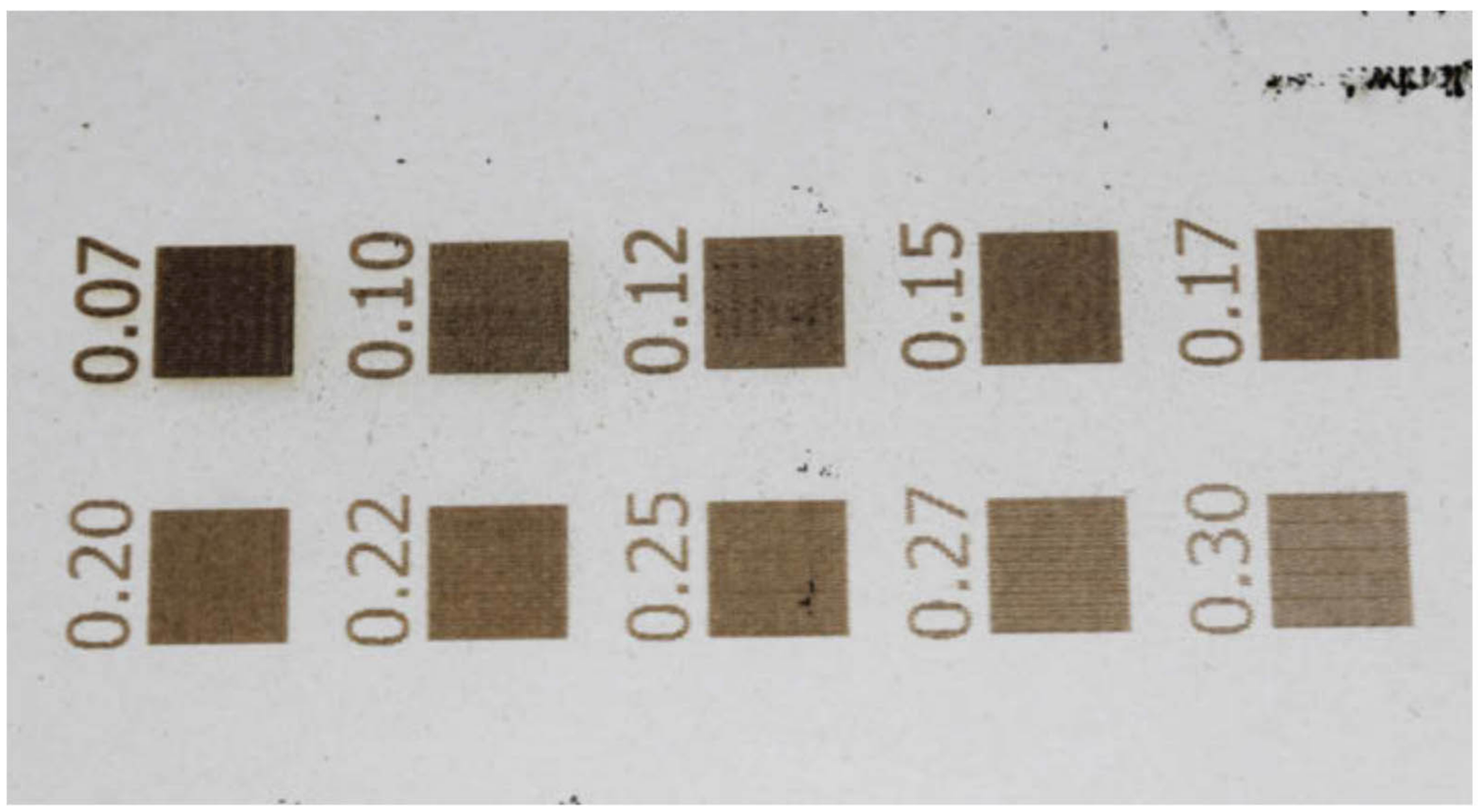


Bild 10: So sehen die Testfelder beim Intervalltest aus

Hat er sich jedoch deutlich vergrößert, muss als Scan-Versatz der negative Wert (also -2 statt 2 mm) benutzt werden.

Das wiederholen Sie dann für andere Graviergeschwindigkeiten. Ich empfehle den Bereich von 100 bis 200 mm/s mit jeweils 10 mm/s Abstand bei einem 40-W-Laser, aber das ist stark von der jeweiligen Maschine abhängig.

Optimales Linienintervall

Das Linienintervall bestimmt, wie nahe die Zeilen eines Bildes beim Gravieren beieinanderliegen. Idealerweise sollte der Zeilenabstand dem Durchmesser des Laserstrahls auf dem Gravurmaterial entsprechen. Dieser Wert ist in Lightburn standardmäßig auf 0,1 mm eingestellt, ein Wert, der meist zu klein ist. Die Software enthält aber ein Testtool, mit dem der optimale Abstand einfach bestimmt werden kann. Sie finden es unter Laserwerkzeuge/Intervalltest. Stellen Sie die Werte darin entsprechend Bild 9 ein.

Der Test ergibt eine Gravur mit unterschiedlichen Intervallen (Bild 10).

Sie sollten sich das Resultat mit einer Lupe genau anschauen und das Feld suchen, das weder helle Zwischenräume zwischen den Zeilen hat (zu großes Intervall, Bild 11)noch dunkle Linien aufweist (zu kleines Intervall und daher überlappende Zeilen mit doppelt gravierten Bereichen, Bild 12).

Das Feld mit dem richtigen Linienintervall liegt dann in der Mitte, hier sind es 0,15 mm (Bild 13).

Damit haben wir die materialunabhängigen Werte ermittelt. Nun widmen wir uns dem Material, das Sie künftig benutzen möchten. Auch davon brauchen wir zum Test einige Proben.

Bild 11: Die helleren Linien zeigen einen zu großen Zeilenabstand.

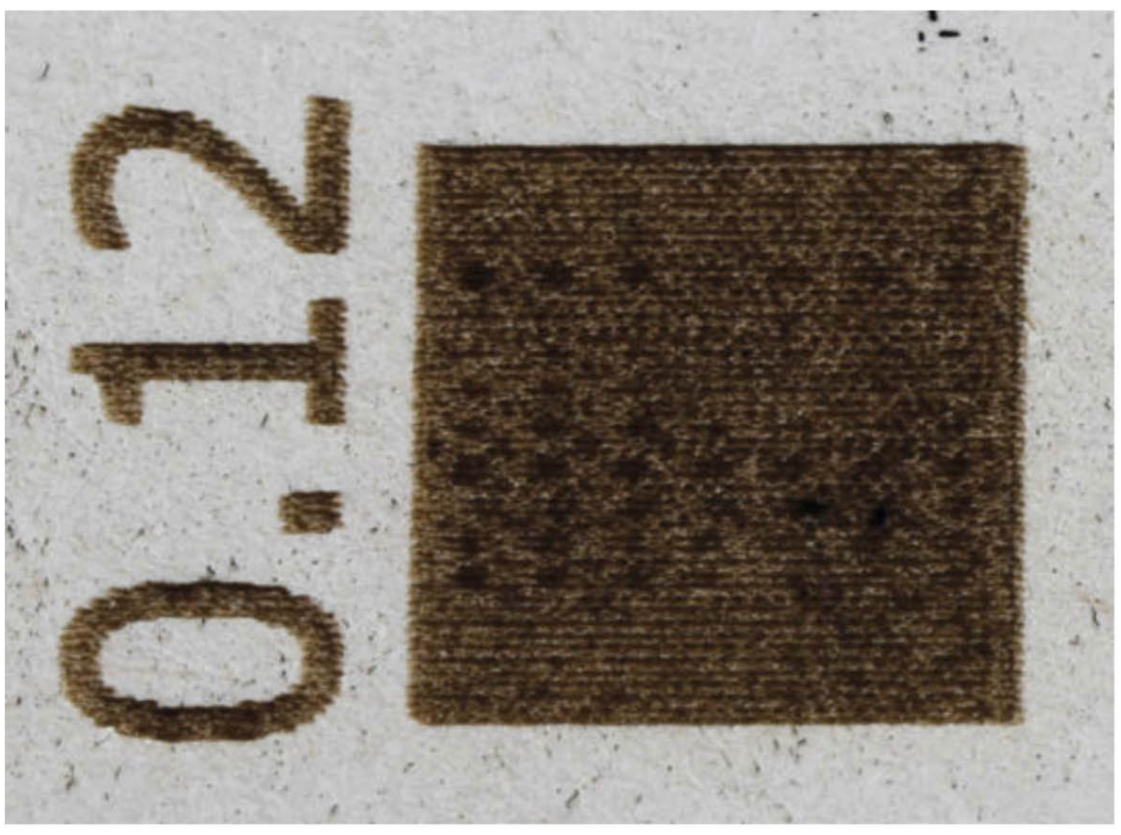


Bild 12: Hier tauchen dunkle Linien auf, die auf zu kleinen Abstand hinweisen.

Material-Parameter ermitteln

Hier geht es darum, für jedes Material herauszufinden, bei welcher minimalen Geschwindigkeits-Leistungs-Kombination der Laser gerade anfängt, eine Gravur zu verursachen und bei welcher Kombination der maximale Gravureffekt eintritt, eine weitere Steigerung also keine zunehmende Verfärbung des Materials mehr bringen würde, nur mehr Rauch. Bei Glas, Plexiglas und anderen durchsichtigen Kunststoffen hingegen sind das Trübungen durch Materialabtrag statt Verfärbungen.

Den folgenden Test müssen Sie für jedes Material durchführen. Über den Kurzinfo-Link können Sie Dateien mit Testmustern herunterladen. Beginnen sollten Sie mit der Datei „GravTest_150_1_25.lbrn2“. Öffnen Sie die Datei in Lightburn und gravieren Sie sie auf eine Platte des Materials, dass Sie testen möchten. Die Datei enthält 25 Felder, die bei einer Geschwindigkeit von 150 mm/s mit 1 bis 25 Prozent Laserleistung graviert werden. Das Ergebnis sieht dann beispielsweise wie in Bild 14 aus.

Ist bei diesem Test der Abstand zwischen minimalem und maximalem Effekt weniger als etwa 7 Prozent, führen Sie einen Test mit höherer Geschwindigkeit durch. Entsprechende Testdateien stehen zum Download bereit. Tritt hingegen auch bei 25 Prozent Leistung noch keine kräftige Färbung auf, probieren Sie den Test mit geringerer Geschwindigkeit. Sie könnten zwar auch mit höherer Leistung gravieren, doch davon rate ich ab. Bei vielen Lasern steigt mit der Leistung auch der Durchmesser des Laserstrahls an, sodass die Gravur letztendlich Schärfe verlieren würde. Falls Sie es trotzdem probieren möchten: Auch dafür habe ich Dateien vorbereitet.

Haben Sie eine gute Geschwindigkeits-Leistungs-Kombination gefunden, tragen Sie diese Werte in die Materialbibliothek von Lightburn ein. Vermutlich ist Ihnen das zugehörige Einstellungsfenster in der Lasersoftware noch nie begegnet, denn es liegt, gut versteckt, hinter dem Laserfenster (Bild 15).

Schließen Sie das Laserfenster, und die Bibliothek kommt zum Vorschein (Bild 16).

Allerdings gibt es da einen kleinen Bug in Lightroom: Per Klick auf Neu kann man keine neue Bibliothek anlegen. Wohl aber durch Übernahme von Werten aus einem bereits vorhandenen Layer. Da das auch später die übliche Methode ist, zeige ich Ihnen hier, wie das geht. Importieren Sie ein beliebiges Bild. Klicken Sie im Fenster Schnitte/Ebenen doppelt auf den dabei angelegten Layer-Eintrag. Das Einstellungsfenster erscheint.

Geben Sie darin die zuvor ermittelten Werte für die maximale und die minimale Leistung ein (Bild 17).

Im Bibliotheksfenster klicken Sie auf „Neu aus Ebene erstellen“ und geben dann einen aussagekräftigen Materialnamen sowie eine Beschreibung ein. Sie können auch eine Materialstärke einstellen (Bild 18).

Mit Speichern und Eingabe eines Namens für die neue Bibliothek schreiben Sie die Daten auf die Festplatte. Künftig können Sie dann die Parameter aus der Bibliothek übernehmen. Dazu müssen Sie nur einen Layer markieren, in der Bibliothek das Material wählen und auf Zuweisen klicken. So einfach kann die Welt sein …

Der richtige Bildmodus

Jetzt ist es Zeit, mit den Bildern zu arbeiten, die Sie gravieren möchten, und zwar auf dem

Bild 13: Im Feld zwischen den beiden schon erwähnten ist das Zeilenmuster am gleichmäßigsten.

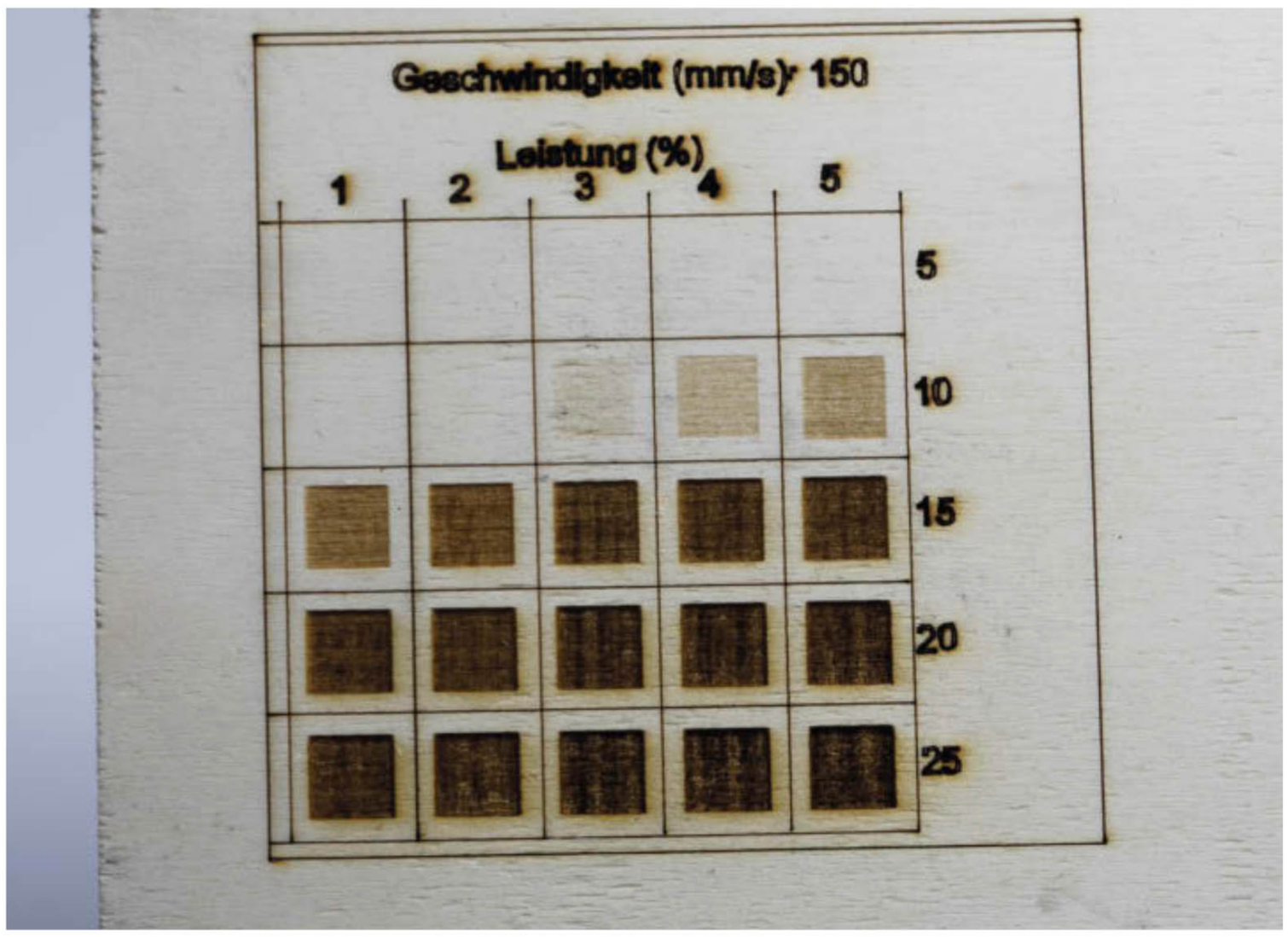


Bild 14: Erst ab 8 Prozent Laserleistung beginnt die Gravur. Der maximale Effekt tritt bei 18 Prozent ein.

Bild 15: Hinter der Lasersteuerung versteckt sich ein weiteres Fenster, …

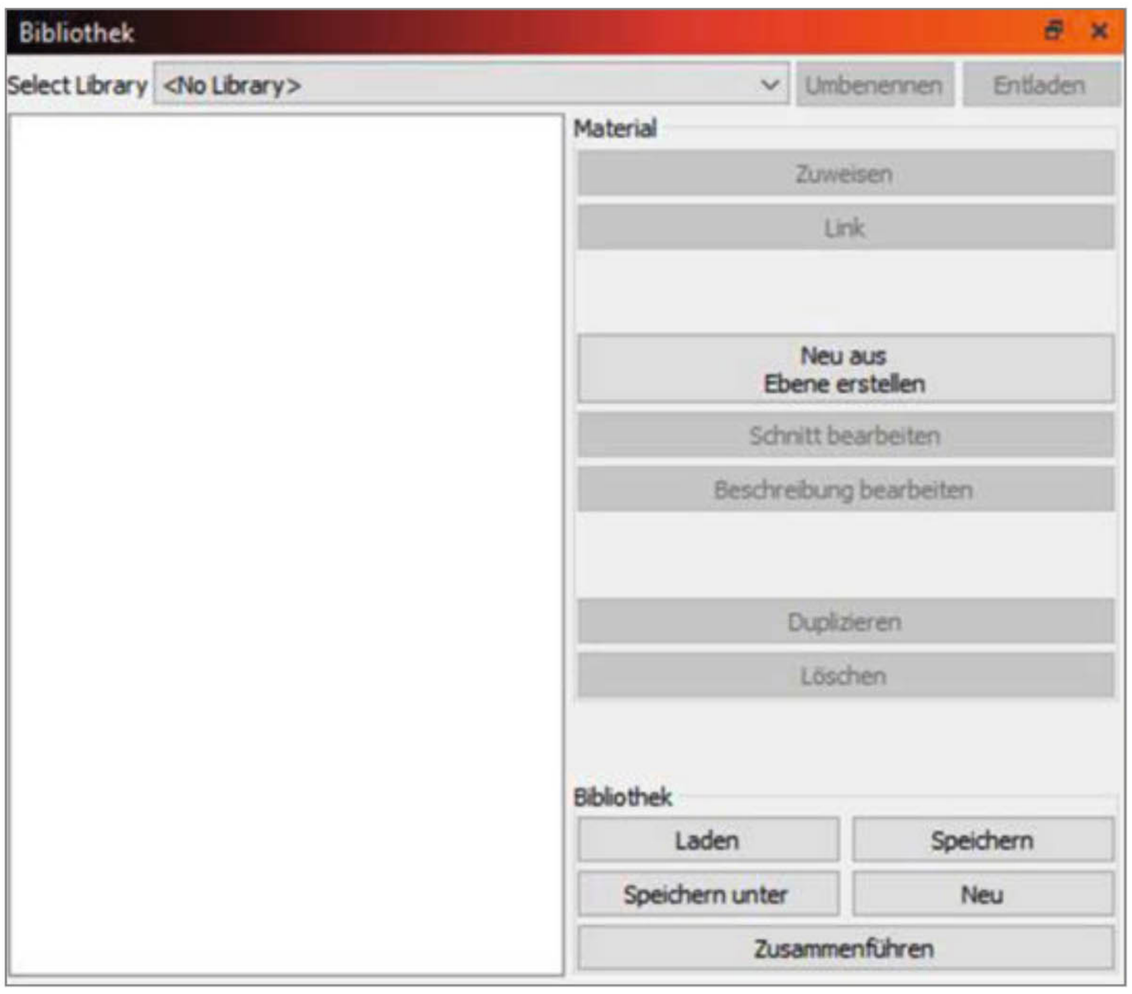


Bild 16: … die Bibliothek, die bei den meisten wohl noch völlig leer sein dürfte.

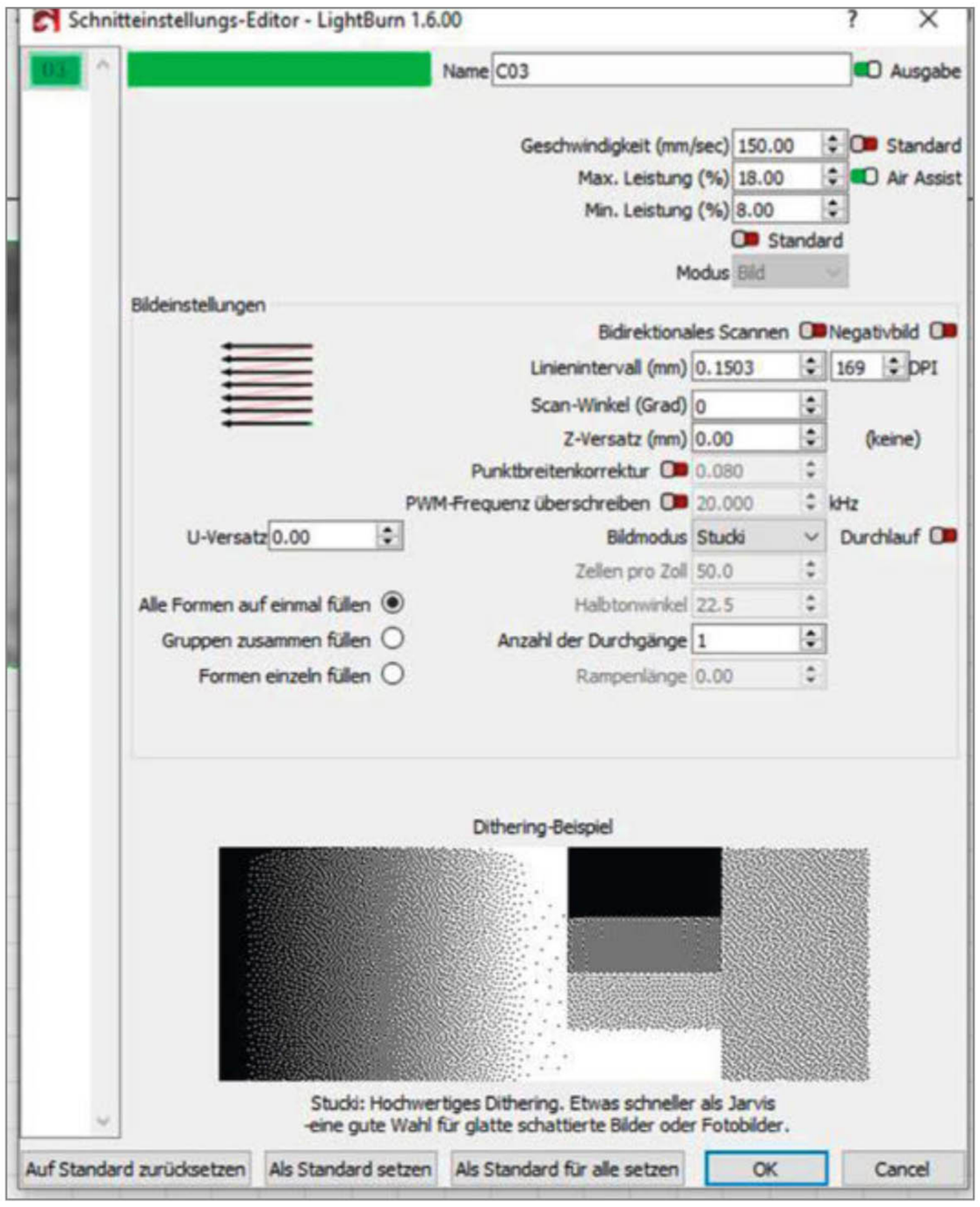


Bild 17: Stellen Sie den Layer auf die ermittelten Werte ein.

Der Laser-Tipp

Den Artikel über Material für Lasercutter nahm Make-Abonnent Rüdiger Mohnen zum Anlass, uns einen raffinierten Trick zuzusenden: Glas lässt sich doch mit Dioden-Lasern gravieren und zwar, nachdem man es mit Zinkspray (Rostschutz aus dem Autohandel) beschichtet hat. Allerdings sollte man unbedingt richtig auf die Glasoberfläche fokussieren.

Nach der Gravur kann die Beschichtung je nach Spray mit Spiritus oder Farbverdünner wieder abgewaschen werden. Das Ergebnis kann sich sehen lassen.

Besten Dank für diesen Tipp.

Übrigens funktioniert das auch auf Metall, sogar noch besser als der im genannten Artikel erwähnte Penaten-Creme-Trick, da sich das Spray gleichmäßiger auftragen lässt.

Das Laser-Zink-Spray

Ein Glas per Zinkspray-Verfahren mit dem Dioden-Laser graviert.

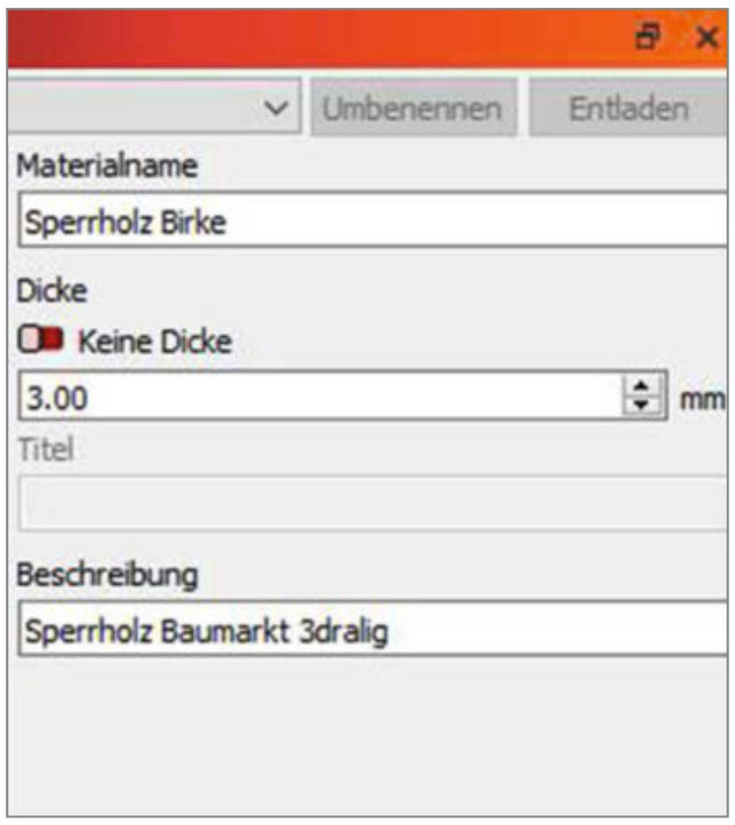


Bild 18: Achten Sie auf einen genauen Materialnamen.

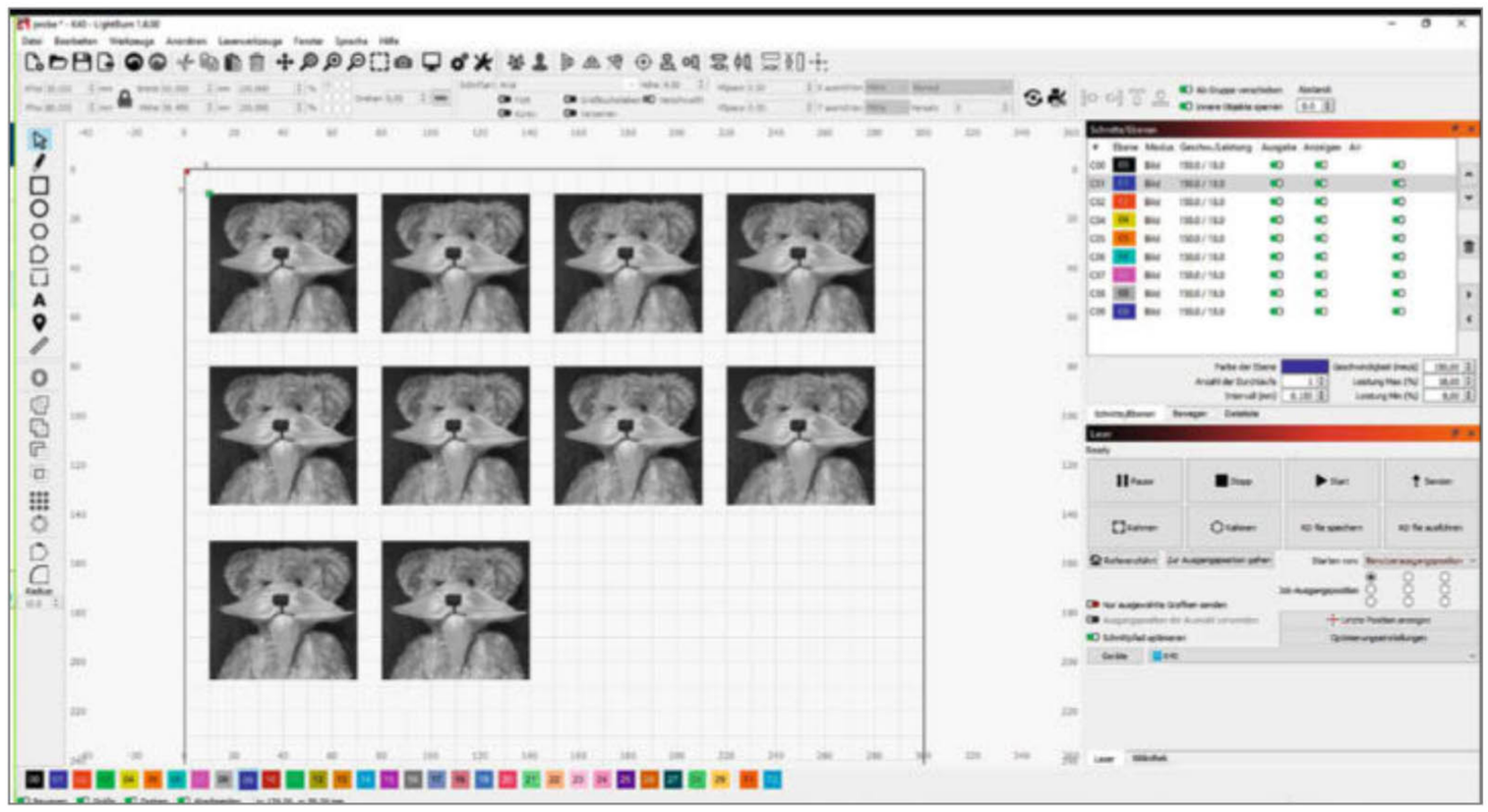


Bild 19: 10 Bilder, 10 Layer, jeder mit eigenem Bildmodus

Material, auf dem das Endergebnis schließlich landen soll. Dennoch brauchen Sie auch hier zunächst noch eine Probeplatte.

Lightburn kann Bilder mit unterschiedlichen Verfahren in ein Pixelmuster umwandeln, das der Laser dann graviert. Die Gründe für diese Umwandlung: Zum einen haben die Bilder meist eine erheblich höhere Auflösung als die Gravur, die der Laser herstellen kann. Zum anderen müssen die Farb- und Helligkeitsabstufungen in Punkte unterschiedlicher Helligkeit, Durchmesser oder in unterschiedliche Punktedichten umgerechnet werden. So machen es übrigens auch Tageszeitungen beim Druck von Bildern.

Zehn verschiedene Verfahren, die sogenannten Bildmodi, stellt Lightburn dafür im Einstellungseditor für Bilder bereit. Aber welcher ist der Beste? Da eine Empfehlung abzugeben, wäre doch sehr vermessen, denn das hängt vom jeweiligen Bild und vom gewünschten Ergebnis ab. Aber Sie können sich relativ schnell eine Vergleichsübersicht erstellen: Nehmen Sie einen annähernd quadratischen Ausschnitt aus dem Bild, das letztendlich graviert werden soll (im Beispiel muss mal wieder mein Teddy Willy herhalten).

Diesen Ausschnitt importieren Sie in Lightburn. Setzen Sie seine Größe dann auf etwa 60 × 60 mm. Diesen Ausschnitt duplizieren Sie dann jeweils mit STRG + D und ordnen die Kopien neben und untereinander an je nach Größe Ihrer Probeplatte. Sie brauchen zehn Kopien.

Jeder Kopie weisen Sie nun in der Farbauswahlleiste am unteren Rand des Lightburn-Fensters einen eigenen Layer zu. Wählen Sie dann nacheinander im Schnitt-/Ebenen-Fenster jeden Layer per einfachen Mausklick an,wählen Sie in der Bibliothek das gewünschte Material und klicken Sie auf „Zuweisen“.

Anschließend wählen Sie noch mal jeden Layer, diesmal aber per Doppelklick an. Im Einstellungsfenster weisen Sie dann jedem Layer einen der zehn Bildmodi zu (Bild 19).

Bild 20: Willy in den zehn Bildmodi graviert (von oben links nach unten rechts): Schwelle, Geordnet, Atkinson, Dithering, Stucki, Jarvis, Newsprint, Halbton, Sketch und Graustufen.

Danach gravieren Sie die Datei und können am Ergebnis (Bild 20) die Auswirkungen der Bildmodi vergleichen.

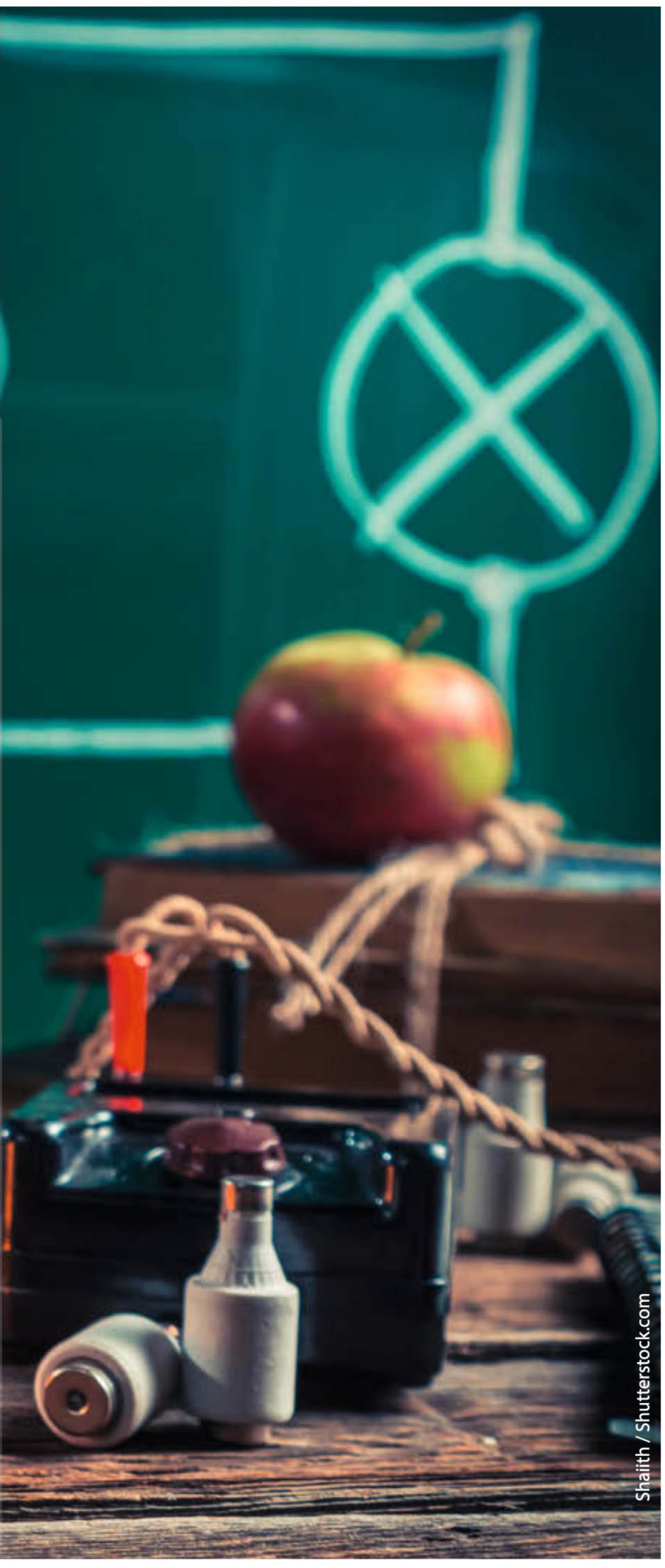
Suchen Sie sich einen für Ihre Vorlage und Ihre Wünsche geeigneten Bildmodus aus. Laden Sie dann das komplette Bild und stellen Sie die Parameter wie beschrieben ein. Übrigens: Oft ist es hilfreich, vor dem Importieren in Lightburn das Bild mit einem Grafikprogramm wie Gimp oder Irfanview in eine Schwarz-Weiß-Version umzuwandeln. Damit kann man meist erheblich besser abschätzen, wie die Gravur später aussehen wird. Das Bild sollte außerdem einen normalen Kontrast haben, also vom satten Schwarz bis zum leuchtenden Weiß. Flaue Bilder können Sie mit den erwähnten Programmen in der Kontrast- und Helligkeitseinstellung entsprechend korrigieren, bevor sie in Lightroom landen.

Jetzt haben Sie alle Werkzeuge für gute Lasergravuren an der Hand. Trotzdem werden Sie zu Beginn noch das ein oder andere Mal mehrere Versuche brauchen, um ein optimales Ergebnis zu erzielen. Diese Erfahrungen müssen Sie aber selbst sammeln. Viel Spaß dabei! *—hgb*

Schaltungen simulieren

Elektronik-Simulatoren helfen den Profis beim Entwickeln von Schaltungen. Aber auch der Hobbyist und Maker kann davon profitieren, ist doch eine Simulation schneller erstellt als eine Breadboard-Schaltung oder ein Prototyp auf Lochraster. Und in der Lehre und beim Lernen spart man Bauteile und Ressourcen.

von Carsten Wartmann


Shaiith / Shutterstock.com

Kurzinfo

» Open Source und Freeware
» von Anfänger bis Profi
» Mainstream- bis Spartensoftware

Mehr zum Thema

» Ákos Fodor, WLAN in Wokwi nutzen, Make 1/24, S. 201
» Ákos Fodor, Mikrocontroller-Projekte simulieren mit Wokwi, Make 3/23, S. 88
» Andreas Gräßer, Simulieren statt Löten - mit LTspice, Make 2/21, S. 118

Alles zum Artikel im Web unter
make-magazin.de/xzvr

Simulatortypen

Wir haben die Simulatoren im Artikel grob in zwei Kategorien eingeteilt: interaktiv und nicht interaktiv bzw. zeitbasiert.

Beim interaktiven Programmen startet man den Simulator und kann dann sozusagen mit der simulierten Schaltung interagieren. Wie auf der Werkbank kann man Messen, sei es mit einem simulierten Multimeter, Oszilloskop oder auch einen erzeugten Ton abhören. Natürlich ist es auch während der laufenden Simulation möglich, die Werte von Bauteilen zu ändern (auch bei „fixen" Widerständen etc.) und bei einigen Simulatoren (hier im Test CuircuitJS) während die Simulation läuft, neue Komponenten einbauen oder die Schaltung zu erweitern. Je nach Schaltung und CPU-Geschwindigkeit läuft die Simulation langsamer, in Echtzeit oder sogar schneller als in der realen Welt.

Beim nicht interaktiven oder zeitbasierten Simulator lässt man zuerst den Simulator eine bestimmte Zeitspanne berechnen oder gibt einen Spannungsverlauf vor, der sozusagen in die Schaltung gefüttert wird. Die umfassendste Simulation ist die Transientenanalyse. Damit erhält man eine komplette Datenreihe von Spannungen und Strömen, inklusive der Einschwingvorgänge, an allen Komponenten (Knoten) der Schaltung. Solche Vorgänge sind in interaktiven Simulatoren oft nur schwierig zu erfassen. Zusammen mit anderen Simulationsarten (etwa Parameter-, DC- und Frequenz-Sweep) kann die Schaltung bis ins Detail untersucht werden. Die Daten kann man dann visualisieren und bearbeiten. Nach jeder Änderung an der Schaltung oder den Werten der Bauteile wird dann die Simulation wieder berechnet.

Mithilfe von Simulationen können lineare und nichtlineare, analoge und digitale Schaltungen simuliert werden. Möchte man in das Thema einsteigen, so hat man die Qual der Wahl zwischen zahlreichen kostenlosen und natürlich auch kommerziellen Programmen und Diensten. In diesem Artikel geben wir einen Überblick über die Arten (siehe Kasten Simulatortypen) und Klassen (von spielerisch bis professionell) der Simulatoren.

Oft steht und fällt der Einsatz eines Simulators mit den virtuellen Komponenten und Bauteilen, die für ihn zur Verfügung stehen. Für die Spice-Varianten (siehe Kasten „Würzig? Spice!") unter den Simulatoren gibt es professionelle Bibliotheken, die zum Teil direkt vom Hersteller an den Originalteilen erstellt und vermessen wurden. Die fachgerechte Nutzung und Integration ist dann allerdings oft auf einem höherem Anwenderniveau.

Ebenso wichtig ist eine einfache und effiziente Bedienung der Programme. Bei einem Wechsel des Simulators kann bei komplexen Schaltungen ein Export sehr hilfreich sein.

Für viele Einsteiger-Simulatoren steht nur eine begrenzte Anzahl von Standardbauteilen zur Verfügung. Bei etwas komplexeren Simulatoren ist es jedoch möglich, die Parameter der Bauteile in weiten Grenzen, etwa nach einem Datenblatt, anzupassen und so zum Ziel zu kommen. Ein nicht vorhandener Chip kann als eigenes Bauteil aus Standardkomponenten „nachgebaut" werden.

Bei der Auswahl der Testkandidaten haben wir versucht, aus dem dynamischen Feld eine Auswahl zu treffen, die jeweils einen Nutzertyp anspricht. Fritzing taucht hier z.B. nicht auf, weil es bei den Simulationen sehr schwach aufgestellt ist. Tinkercad Circuits ist von der Zielgruppe her ein geeigneter Ersatz. Allerdings läuft Fritzing lokal auf dem Rechner und ist Open Source, während Tinkercad (Autodesk) serverbasiert ist (siehe auch Kasten „Cloud, Freeware oder Open Source").

Sicher haben Sie, lieber Leser, ein eigenes Lieblingsprogramm oder einen Tipp; zögern Sie nicht, uns eine E-Mail zu schreiben.

Falstad/CircuitJS

CircuitJS wurde ursprünglich von Paul Falstad als Java Applet entwickelt und dann von Ian Sharp nach Javascript portiert, wodurch es nun praktisch in jedem Browser läuft. Das

Würzig? SPICE!

Der Elektronik-Simulator SPICE (Simulation Program with Integrated Circuit Emphasis) ist ein Standardwerkzeug zur Simulation von elektronischen Schaltungen. Es wurde in den 1970er-Jahren an der University of California, Berkeley, entwickelt und der Quellcode freigegeben.

SPICE ermöglicht es, elektronische Schaltungen zu modellieren und das Verhalten von Komponenten wie Widerständen, Kondensatoren, Transistoren, Dioden und vielen anderen Bauteilen zu simulieren und zu analysieren. Es kann sowohl im Zeit- als auch im Frequenzbereich arbeiten und hilft somit, das Verhalten von Schaltungen unter verschiedenen Betriebsbedingungen zu verstehen.

Neben den ursprünglichen Spice-Versionen gibt es auch eine Vielzahl von kommerziellen und Open-Source-Ablegern, die zusätzliche Funktionen und Benutzerfreundlichkeit bieten. Einige der bekanntesten Spice-Versionen sind:

- PSpice: Eine populäre kommerzielle Version von SPICE, von Cadence Design Systems. PSpice bietet erweiterte Funktionen wie parametrische Analyse, Sensitivitätsanalyse und eine grafische Bedienoberfläche.
- LTspice: Eine kostenlose Spice-Implementierung von Analog Devices, die speziell für die Simulation von Schaltungen mit Linear Technology-Bauteilen entwickelt wurde. LTspice bietet eine grafische Oberfläche und umfangreiche Bauteilbibliotheken.
- Ngspice: Ein Open-Source-Projekt, das den SPICE3-Code erweitert und verbessert. Ngspice zielt darauf ab, eine moderne, plattformübergreifende Spice-Implementierung bereitzustellen.

Projekt ist auf GitHub aktiv und steht unter der GPL-Lizenz (General Public License). Damit kann es lokal mit Internetzugang von Pauls Server, auf dem eigenen Server oder auch als Standalone-Version (App/Exe) für gängige Betriebssysteme eingesetzt werden.

Die Bedienung von CircuitJS (Bild 1) kommt etwas altbacken daher. Im Gegensatz zu anderen Simulatoren finden sich hier auch modernere Konzepte und die rechte Maustaste ruft ein Kontextmenü für das Bauteil unter der Maus auf. An einer leeren Stelle im Schaltplan erscheint das Hauptmenü Zeichnen. Weiterhin gibt es Auswahlrahmen und man kann eine Auswahl komplett verschieben. Allerdings muss man sich die unterschiedlichen Modi zum Ziehen (mindestens fünf) mit jeweils eigenen Tastenmodifikatoren merken.

CircuitJS ist ein interaktiver Simulator (siehe Kasten „Simulatortypen“). Wenn man den Stromkreis schließt und den Simulator startet (RUN/STOP-Taste), fließt der simulierte Strom. Natürlich nicht in Echtzeit, sondern abhängig von der eingestellten Simulationsgeschwindigkeit und der Rechenleistung des PCs. Der Stromfluss wird durch gelbe Pixel visualisiert, positive Spannungen sind durch die grüne

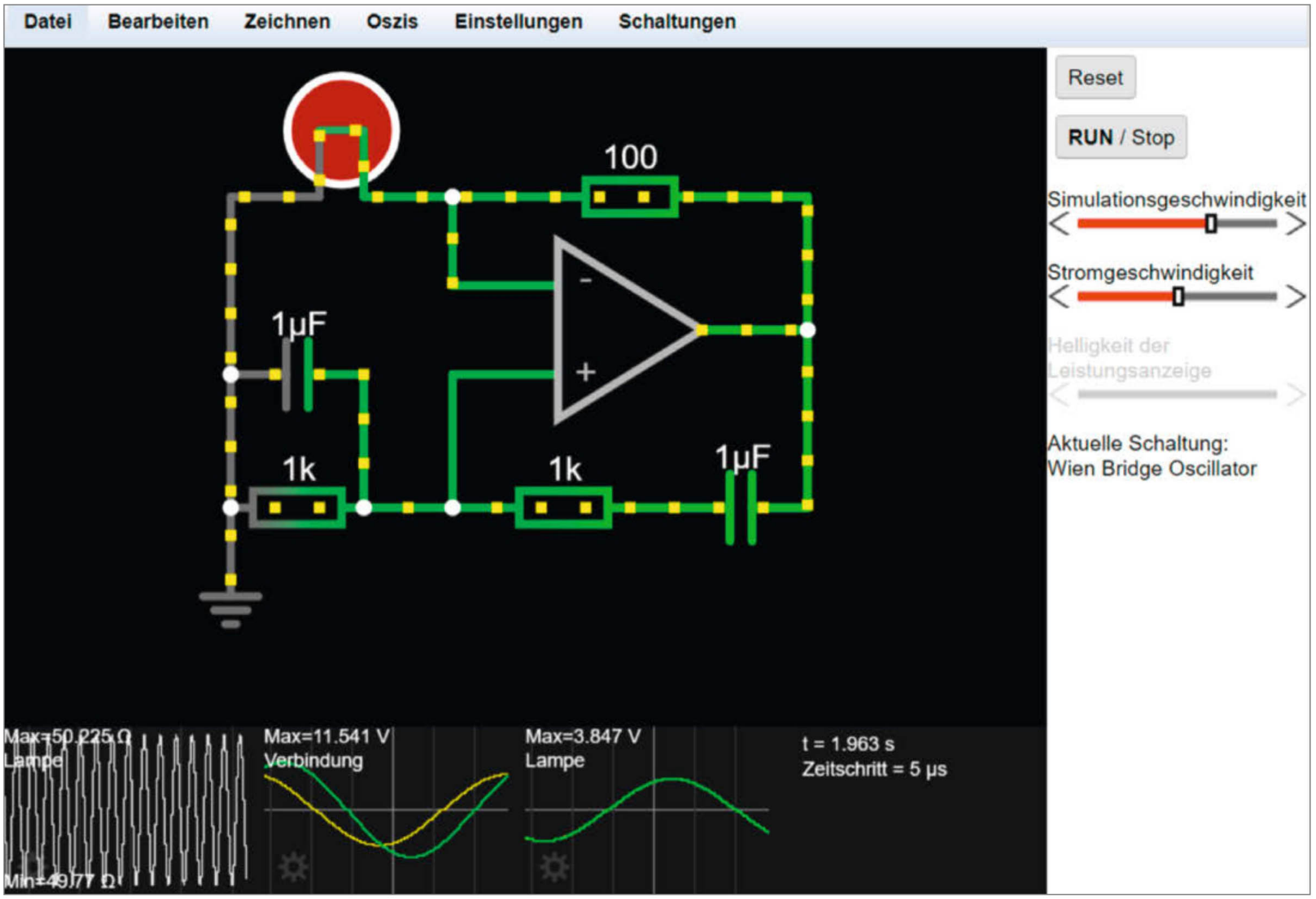


Bild 1: CircuitJS bei der Arbeit. Der Schaltplan ist frei zoombar.

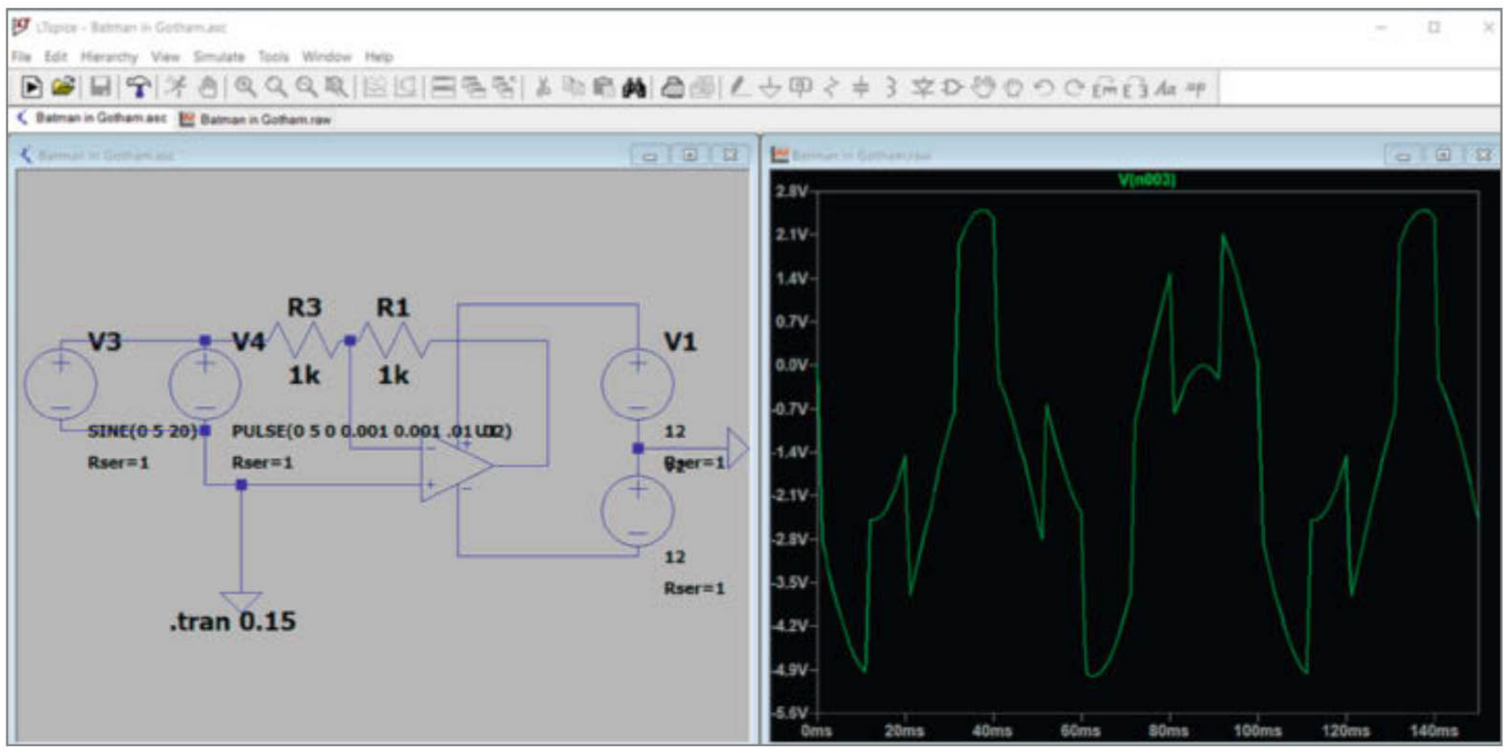


Bild 2: LTSpice simuliert. Erkennen Sie Batman in Gotham?

CircuitJS Lokalisierung

CircuitJS bietet die Oberfläche in vielen Sprachen an. Einige Menüpunkte und Texte sind jedoch bisher nicht übersetzt. Den Knopf „RUN / Stop“ oder auch „Reset“ übersieht man fast, dann tauchen aber auch Menüpunkte wie „View in New Scope“ auf. Dies ist bekannt und dank Open Source auch leicht zu beheben, nur hat sich bisher noch niemand für diese Sisyphus-Arbeit gefunden.

Farbe der Leitungen gekennzeichnet, rot bedeutet negative Spannung und grau ist der Erdungspegel (Ground-Level). So kann man auf den ersten Blick erkennen, was im Stromkreis passiert.

Die Beispiele unter dem Menüpunkt Schaltungen lesen sich, als wären die Schaltungen aus einem Elektronikbuch dort abgelegt worden. Hier kann man ausgiebig stöbern, Schaltungen ansehen oder als Grundlage für eigene Entwürfe verwenden. Online finden sich weitere interessante Schaltungen, wie etwa die Simulation des Ur-Pong-Videospiels.

Fährt man mit der Maus über eine Komponente, sei es ein Draht oder ein Bauteil, werden unten rechts aktuelle Spannungen, Ströme, Leistungen und mehr angezeigt. Noch genauer und vor allem für Wechselspannungen wichtig, sind die Oszilloskop-Ansichten (rechter Mausklick auf die Komponente und dann „View in …“, siehe Kasten CircuitJS Lokalisierung), die man entweder unten im Fenster oder frei in der Schaltung verteilen kann.

Über das Menü Zeichnen oder die rechte Maustaste werden die Komponenten ausgewählt und durch Drücken, Halten und Ziehen der linken Maustaste die Anschlüsse platziert. Die wichtigsten Komponenten lassen sich mit einfachen Tastendrücken auswählen. So ist etwa ein Widerstand mit R und anschließendem Ziehen und Platzieren mit der Maus schnell in die Schaltung eingebaut. Die Taste C wählt Kondensatoren, Shift-C Elektrolytkondensatoren. W wählt den wichtigen Draht (Wire), also eine Verbindung zwischen Komponenten. Die Navigation in größeren Schaltungen erfolgt mit der mittleren Maustaste und dem Scrollrad (oder den +/- Tasten) .

CircuitJS bietet eine gute Auswahl an Standardbauteilen, komplexe Chips sucht man eher vergebens. Es gibt einige, wie den 555-Timer oder RAM-Chips, aber die Auswahl ist nicht sehr groß. Ist das Passende nicht vorhanden, muss man sich das Gewünschte als „Subcircuit“ selbst definieren oder aus dem Netz laden.

Die Grundbauteile sind dann auch erst einmal „Standard“, möchte man einen speziellen Transistor definieren, so ruft man das Bearbeiten-Fenster auf und kann nun Grundparameter wie Hfe (grob die Stromverstärkung) einstellen. Über „Create New Model“ können dann mit dem Datenblatt in der Hand auch ganz spezielle Komponenten definiert werden.

Durch den einfachen Zugang, die schnelle Arbeitsweise und die brauchbaren Ergebnisse ist CircuitJS für viele Dinge im Maker-Alltag einsetzbar. Man bekommt ein Feedback ohne die Simulation manuell starten zu müssen, ideal für die Lehre, YouTube und die schnelle Schaltung zwischendurch. Man sollte sich aber nicht täuschen lassen, in den ausufernden Menüs sind noch viele Möglichkeiten versteckt, wie die Soundausgabe und das „Simulieren“ von echten Daten, die z. B. mit einem digitalen Oszilloskop aufgezeichnet wurden.

LTspice

LTSpice (siehe auch „Mehr zum Thema“) ist eine kostenlose Spice-Variante von Linear Technology (jetzt Analog Devices). LTspice basiert auf dem „Ursimulator“ SPICE (siehe Kasten„Würzig? SPICE!“). Im Gegensatz zur Urvariante gibt es eine grafische Oberfläche zum Zeichnen von Schaltplänen und zum Duchführen vonSimulationen. LTSpice ist kostenlos und für Windows und MacOS verfügbar. Unter Linux kann die Windows-Version mit Wine passabel benutzt werden.

Man zeichnet seine Schaltpläne, simuliert sie und kann dann mit der virtuellen Messspitze im „Waveform Viewer“ die Vorgänge an beliebigen Stellen der Schaltung grafisch darstellen (Bild 2).

Die Bedienung von LTspice ist recht altmodisch und heutzutage ungewohnt. Die vielen verfügbaren Tutorials (auch beim Hersteller, siehe Link in Kurzinfo) helfen hier ungemein beim Einstieg. Hat man sich aber erst einmal an das Konzept gewöhnt, lässt es sich effektiv arbeiten und letztlich hat LTSpice einen Standard gesetzt, an dem sich auch andere Programme orientieren.

Neben den wenigen Grundkomponenten finden man hinter dem Komponenten-Symbol (sieht aus wie ein Logikgatter) tausende andere Komponenten. Dort verstecken sich auch die so wichtigen Spannungsquellen unter „voltage“. Mit ihnen lassen sich Gleich- und Wechselspannungen und verschiedene Wellenformen erzeugen, aber auch eigene Daten einspeisen.

Für viele Komponenten, die nicht in LTSpice enthalten sind, gibt es online Komponentenmodelle, oft direkt vom Hersteller. Die Installation und Verwendung ist jedoch nicht ganz unkompliziert und die Konvertierung aus anderen Spice-Varianten teilweise schwierig. Damit arbeitet man dann aber in der Simulation mit Komponenten, welche die Realität so gut wie möglich beschreiben und so für sehr exakte Modellierungen der realen Vorgänge sorgen. Aber auch hier muss man viel Wissen über Spice mitbringen. In den Programmdialogen wird oft einfach ein SPICE-Befehl abgefragt.

Das Programm ist für den Profi, den angehenden Elektroniker und den ambitionierten Maker. Hier geht praktisch alles, auch externe Daten und Messreihen können integriert werden. Den Luxus auch Arduino&Co zu simulieren, bietet LTSpice allerdings nicht. An die leicht angestaubte Programmoberfläche kann man sich gewöhnen oder sie umgehen, wenn man sich bereits mit KiCAD (siehe nächster Abschnitt) auskennt; dieses E-CAD integriert ngSpice in seinen Schaltplaneditor.

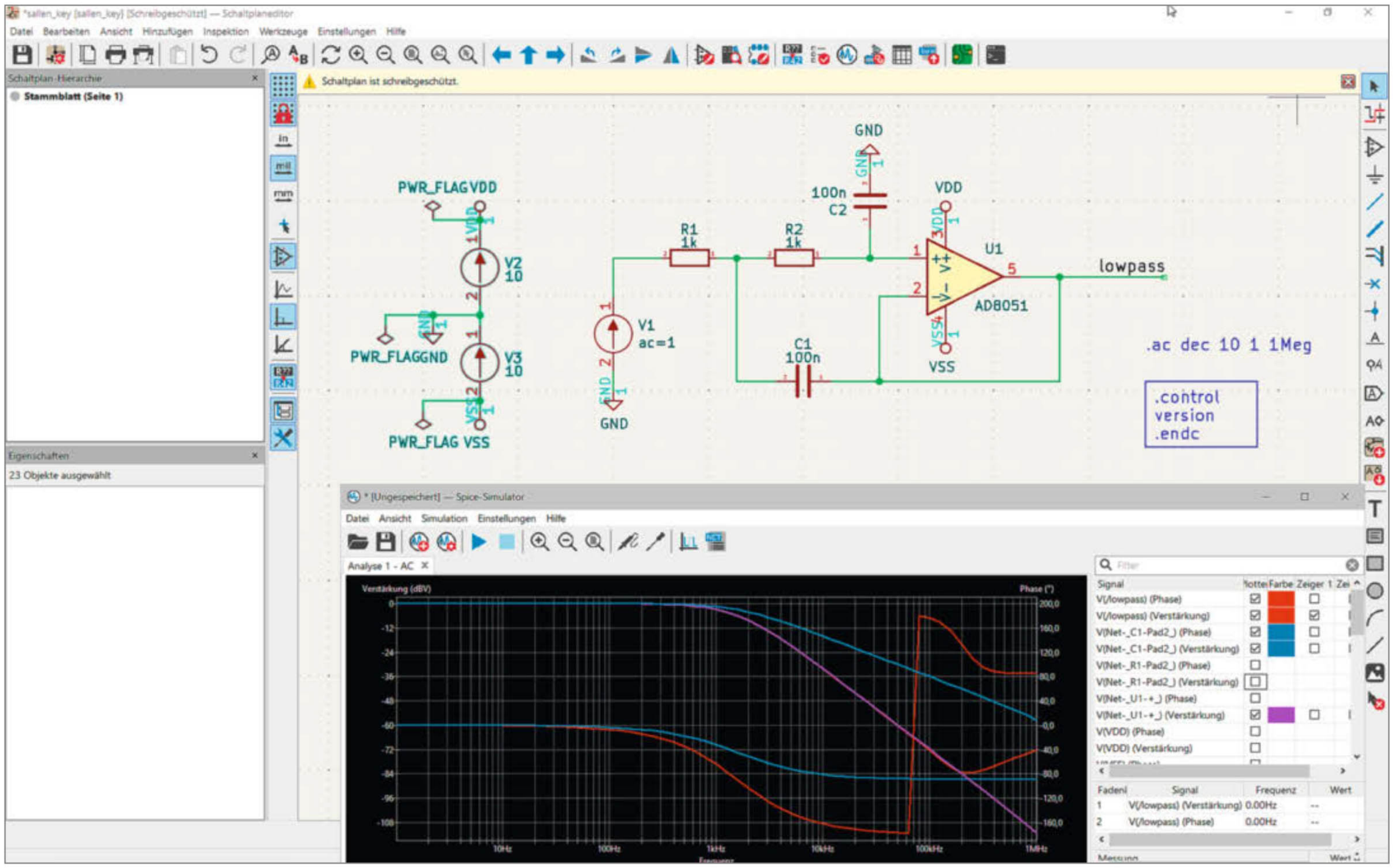


Bild 3: Hier kann es eng werden. KiCAD fühlt sich auf Multimonitorsystemen oder Breitbildschirmen besonders wohl.

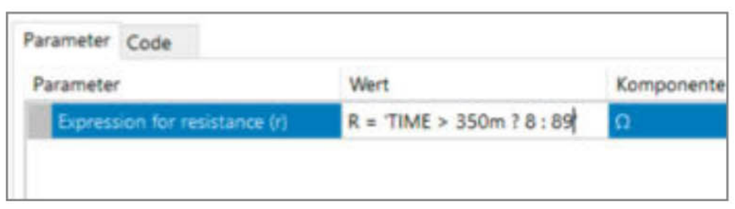


Bild 4: SPICE-Code um nach 350 ms den Widerstandswert zu ändern.

KiCAD mit ngSpice

KiCAD ist eine sehr komplexe und leistungsfähige Open-Source-Software zur Erstellung von Platinen und einhält einen ngSpice-Simulator. Es ist für alle gängigen Plattformen (Linux, Windows, macOS, Quellcode) verfügbar.

Es nur als Elektroniksimulator zu verwenden, ist aufgrund der Einarbeitungszeit eher eine Verschwendung. Benutzt man es ohnehin für seine Leiterplatten, ist die Integration von ngSpice aber einwandfrei gelöst und man kann problemlos (Bild 3) zwischen Schaltplan, Simulation und PCB-Design wechseln.

Der Schaltplan-Editor ist mächtig, auch hier sollte man sich die Tutorials anschauen, um das Potenzial auszuschöpfen. Das Simulatorfenster kann geöffnet bleiben, man ändert Verbindungen oder Bauteilwerte in der Schaltung und startet die Simulation neu. Nicht so schnell wie die interaktiven Simulatoren, aber effektiv. Die Simulation ist wie bei LTSpice sehr gut, aber es gelten die gleichen Einschränkungen wie dort.

Nicht vergessen sollte man, dass die riesige Funktionalität und Komplexität von Spice (Bild 4) nicht wirklich versteckt oder durch die GUI vereinfacht wird: Ein solides Wissen drüber, wie man Befehle, Parameter und Spice-Programmcode in Spice schreibt, ist unabdingbar.

Tinkercad (Circuits)

Tinkercad läuft auf von Autodesk gehosteten Servern im Browser und ist mit Anmeldung über Google-, Apple-Account oder per E-Mail und Passwort kostenlos nutzbar. Neben der 3D-Modellierung kann Tinkercad auch Codeblöcke erzeugen, ist also praktisch ein OpenSCAD (siehe in diesem Heft S. 16) auf Blockly-Basis. Im Schaltkreisteil („Circuits“) besteht dann die Möglichkeit für eine in diesem Artikel interessante Elektronik-Simulation.

Hier baut man elektronische Schaltungen mit typischen (Starter)-Maker-Komponenten (Bild 5) auf. Alle Elemente, die angeboten werden („Komponenten Alle“), sind auch in der Simulation verfügbar. Leider gibt Tinkercad nicht viele Informationen zu den Bauteilen. Pinbelegungen muss man durch Mouseover über den Pins ablesen, zudem sind die deutschen Übersetzungen oft merkwürdig. Also ist es fast immer nötig ein zweites Browserfenster geöffnet zu haben, in welchem man das Datenblatt des Bauteils sieht. Über die Autodesk-Cloud kann man seine Entwürfe mit der Community teilen und deren Projekte ausprobieren. Für alles, was über die Funktionen von „Schaltkreise“ hinausgeht, verweist Autodesk in seiner FAQ auf Fusion 360, das ebenfalls Spice benutzt.

Wir haben vergebens eine Hilfe-Funktion gesucht; sicher gibt es viele Tutorials, aber wenn man eine schnelle Hilfe zu Tasten benötigt, wäre das super. In der Werkzeugleiste kann man die wichtigsten Tastenbefehle als Hinweise zu den Schaltflächen sehen.

Die Schaltplanansicht produziert einen lesbaren Schaltplan, allerdings muss man nehmen, was die Automatik aus dem Breadboard macht: Ein Editieren ist in dieser Ansicht nicht möglich. Der Schaltplan kann als PDF heruntergeladen werden. Zusätzlich gibt es noch eine Materialliste, die man als Tabelle im CSV-Format exportieren kann.

Man muss übrigens kein Breadboard nehmen: Um mal schnell eine Schaltung auszuprobieren, kann man die Komponenten auch frei verdrahten. Bedienelemente oder das Multimeter, kann man im Simulationsmodus einfach mit der Maus bedienen. Das Multimeter wird leider im Schaltplan einfach weg-

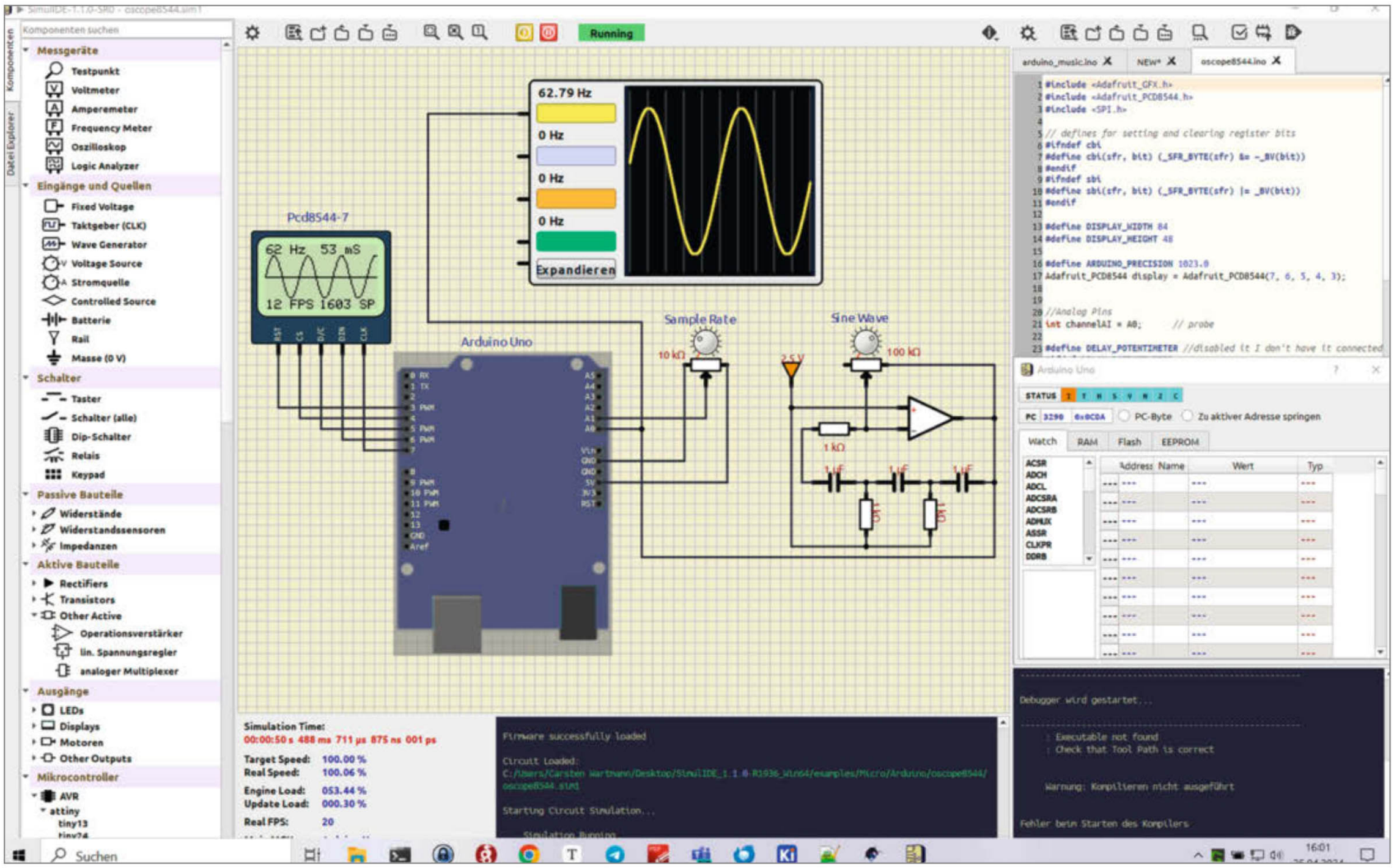


Bild 6: Arduino, MCU-Monitor, analoge Schaltung, Display und Oszilloskop in einer Simulation

gelassen, was besonders bei Strommessungen verwirrend ist und einen nicht geschlossenen Schaltkreis verursacht.

Die Werte von Komponenten wie Widerständen oder Kondensatoren lassen sich in der laufenden Simulation ändern: Wird dadurch etwa der Strom durch eine LED zu hoch, sieht man ein Ausrufezeichen an der LED, dass beim Überfahren mit der Maus noch weitere Informationen preisgibt. Damit ist der Simulator in Tinkercad „Schaltkreise“ nur halb interaktiv: Bauteile hinzufügen oder löschen stoppt die Simulation. Die Einstellmöglichkeiten der Oszilloskope sind bedauerlicherweise sehr rudimentär; dynamische Vorgänge sind nur schwer zu erfassen.

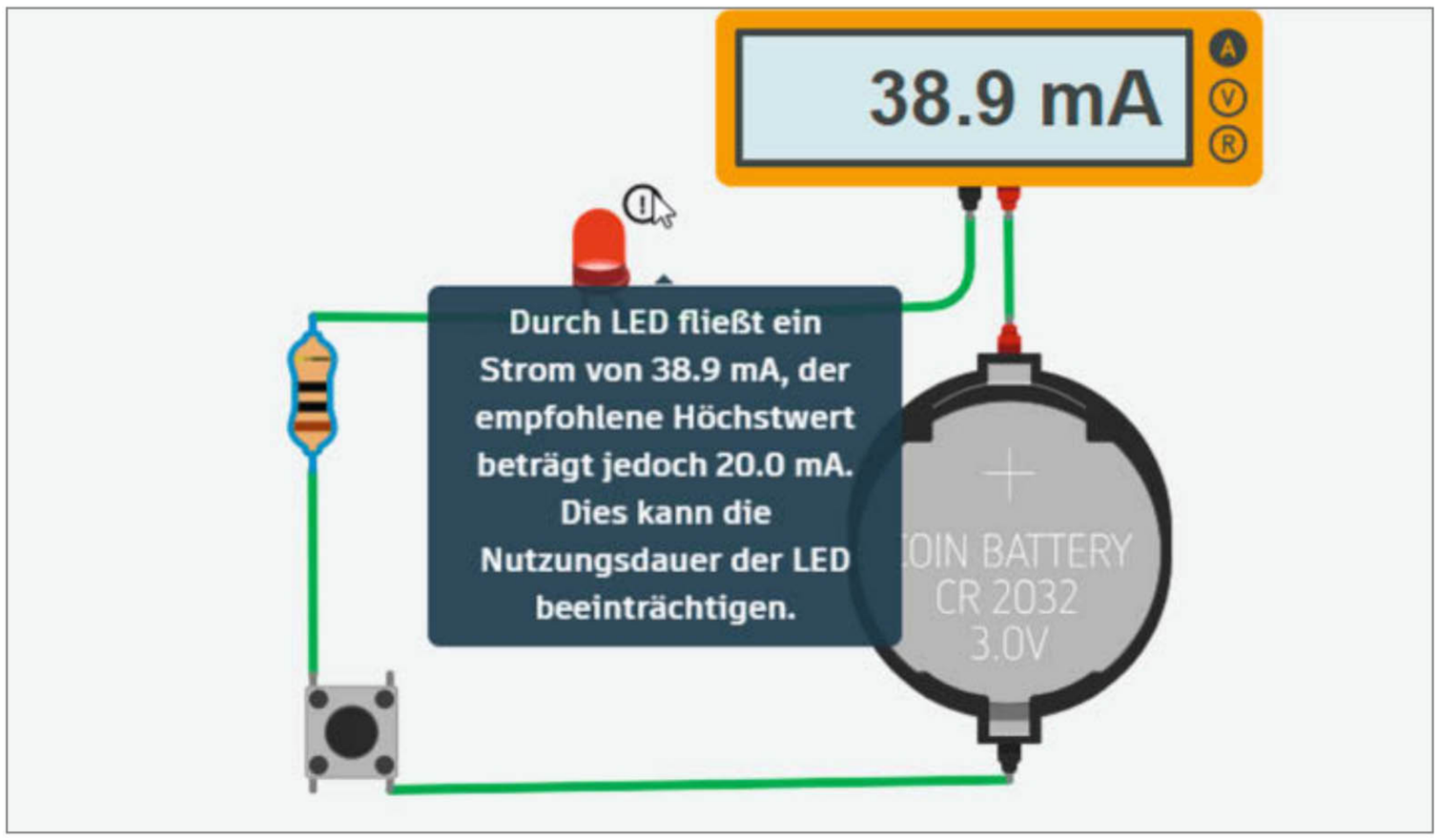


Bild 5: Grafisch ansprechene Simulation in Tinkercad-Schaltkreise

Wunderbar ist hingegen die Anbindung der beiden verfügbaren Mikrocontroller Arduino und Micro:bit gelungen. Beide lassen sich mit grafischen Blöcken oder Arduino C++ bzw. Python für den Micro:bit programmieren. Beim Start der Simulation wird der Code sofort ohne merkliche Verzögerung ausgeführt und für den Micro:bit erscheint eine Oberfläche, mit der man die Sensoren (Lage, Kompass, Temperatur, etc.) des Micro:bit beeinflussen kann.

Alles in allem ist Tinkercad „Schaltungen“ ein prima Simulator für Einsteiger, um mal in die Materie hineinzuschnuppern und schnell analoge Schaltungen zusammen mit Mikrocontrollern auszuprobieren.

SimulIDE

SimulIDE ist ein Open-Source-Projekt, das sich in den letzten Jahren schnell entwickelt hat. Inzwischen ist die Version 1.1.0 aktuell. Es gibt Versionen für Windows, Linux und macOS. Die Installation ist einfach, den Ordner im Zip-Archiv irgendwo entpacken, SimulIDE starten. Warum man bisher so wenig von dieser Software gehört hat? Wir wissen es nicht.

Eine flinke Oberfläche (Bild 6), auf Qt basierend, lässt die Schaltplanerstellung schnell

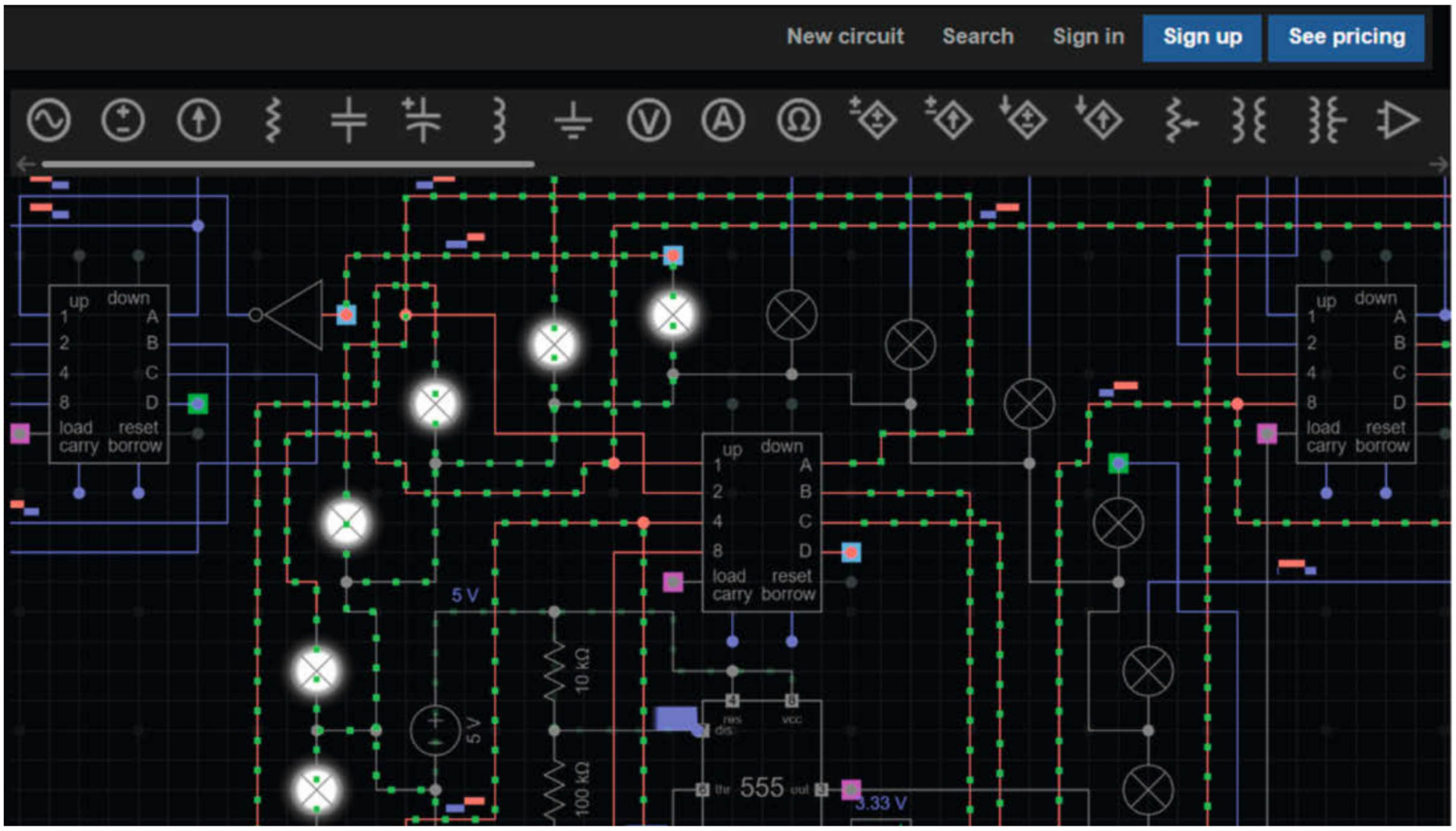


Bild 7: Every Circuit: Komplexe Simulationen mit großer Community

Cloud, Freeware oder Open Source

Das wichtigste Kriterium für einen Simulator ist natürlich dessen Leistungsfähigkeit, kann ich meine Schaltung auch so simulieren, dass mir die Ergebnisse weiter helfen? Gibt es die nötigen Bauteile und ist die Bedienung so, dass man schnell zum Ziel gelangt?

Daneben gibt es aber noch andere Kriterien: Wie gut und langfristig sicher ist der Support der Hersteller? Das kann eine Firma sein oder bei Open Source die Entwickler. Gute und vollständige Dokumentation sind besonders für Einsteiger und Gelegenheitsnutzer wichtig. Support bedeutet auch, dass der Simulator langfristig verfügbar ist, wird ein Cloud-Dienst oder ein Freewareprogramm eingestellt, sind oft nicht nur die eigenen Schaltungen weg, sondern auch die Arbeit, die man in das Erlernen des Programms gesteckt hat. Bei Open Source kann man noch selbst Hand anlegen, um Fehler auszumerzen oder gar die Entwicklung übernehmen.

Viele kostenlose Cloud-Dienste bezahlt man letztlich mit seinen Daten, seien es Google-, Apple- oder Autodeskaccounts oder dem Anschauen von Werbung.

per Drag&Drop aus der gut gefüllten Bauteileliste von der Hand gehen. Die Simulation ist mit einem Knopfdruck gestartet und kann über Bedienelemente in der Schaltung beeinflusst werden. Der Simulator wurde laut Entwickler auf Geschwindigkeit getrimmt.

Der Simulator ist interaktiv, man kann alle Bauteile während der Simulation in ihren Parametern ändern. Beim Einbau neuer Komponenten stoppt die Simulation allerdings. Das Oszilloskop scheint auf den ersten Blick etwas einfach gestrickt, ein Klick auf den Button „Expandieren“ bringt dann die ganze Oberfläche mit Volts/Div, Timebase, Trigger etc. eines 4-Kanal-Oszilloskops hervor.

Für die Ausgabe gibt es simulierte Displays, Lautsprecher (die den Ton natürlich über den Computer ausgeben), LEDs, LED-Matrizen, Grafikdisplays, (Stepper)-Motoren und Servos, die die Drehung grafisch anzeigen und vieles mehr.

Damit nicht genug: Auch komplexe binäre Schaltungen können simuliert werden, die Beispiele reichen von einfachen Logikgattern über Addierer, Uhren, Speicher bis hin zu einer aus einzelnen Logikbausteinen aufgebauten ALU-CPU.

Wenn das nicht reicht, kommen noch diverse Mikrocontroller (MCUs) wie Arduinos, „nackte“ ATmegas, PICs, aber auch Z80 und MC6502 hinzu. Dazu eine ganze Reihe Sensoren (auch I^2C oder SPI), Shields und sonstiges Maker-Zeug. Die MCUs können direkt aus SimulIDE programmiert werden (Arduino, GC Basic, PIC asm, AVR asm, etc.) oder man lädt eine fertige Firmware in den simulierten Mikrocontroller. Zum Debuggen gibt es den MCU-Monitor, mit dem man dem Prozessor auf die Bits, Register, RAM und ROM schauen kann. Und es gibt einen Debugger für den Code.

SimulIDE ist auf jeden Fall einen Versuch wert, es bildet gut ab, was man so im Maker-Alltag braucht.

Spezielles und Interessantes

Ein kompletter interaktiver Simulator in 3D-Grafik auf Steam: Das ist der CRUMB Circuit Simulator (Android, iOS, Windows, macOS, 10 Euro). Da macht das Experimentieren noch mehr Spaß. Die Bauteileauswahl (etwa nur ein Arduino) ist eher beschränkt und da es weder Schaltplan noch Stücklisten gibt, bleibt es auch beim Spielen. Die Simulationsergebnisse passen dank Spice-Unterbau gut.

Geht es eher darum, typische Maker-Projekte zu entwerfen, also fertige Sensorboards oder Bauteile auf Breadboards mit Arduino&Co zu steuern, so sind die kostenlosen Web-Apps Wokwi und das schon etwas ältere circuit.io einen Blick wert. Über Wokwi, das eher auf Mikrocontroller ausgerichtet ist, haben wir bereits mehrere Artikel (siehe „Mehr zum Thema“ und Links) im Make-Magazin veröffentlicht.

Every Circuit (Bild 7) ist ein Cloud-Dienst. Im kostenlosen Bereich kann man nicht viele Elemente zusammenstellen, aber beliebig komplexe Schaltungen aus der riesigen Community anschauen. So sieht man, was der Dienst zu bieten hat und ob sich die einmaligen 15 US-Dollar lohnen.

Die interaktiven Simulatoren (Bild 8) von PhET (Physics Education Technology, University of Colorado Boulder) sind eine tolle Sache, gerade für jüngere Maker. Man kann lernen, wie man Gesetze von Ohm, Faraday und Coulomb anwendet oder einfach nur „John Travoltage“ mit statischer Elektrizität quälen. Das macht allen Spaß. In den Gleich- und Wechselstromsimulatoren kann frei mit Bauteilen experimentiert werden, macht man Fehler, gehen die Bauteile in Flammen auf.

QUCS – Quite Universal Circuit Simulator: Auch wenn die Versionsnummer nichts erwarten lässt, ist diese Software ein Edelstein. Sie kann analoge und digitale Schaltungen mit einer guten, wenn auch altmodischen grafischen Oberfläche simulieren und die Ergebnisse ansprechend darstellen. Und gut dokumentiert ist es auch. Warum wird das so tolle Projekt hier nur kurz erwähnt? Die Version 0.0.20 liegt seit vier Jahren als Release Candidate vor.

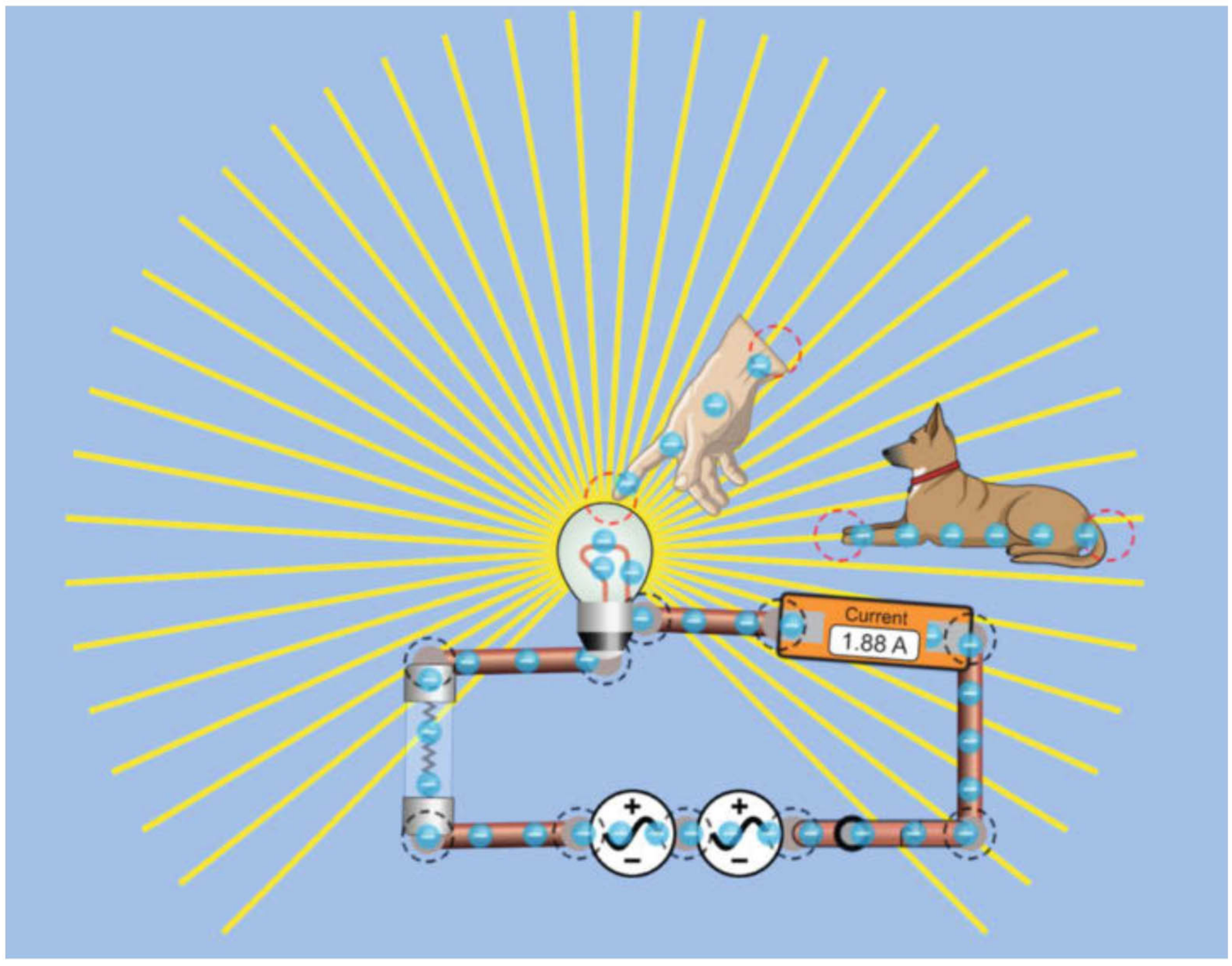


Bild 8: Keine Angst dem Hund passiert nichts!

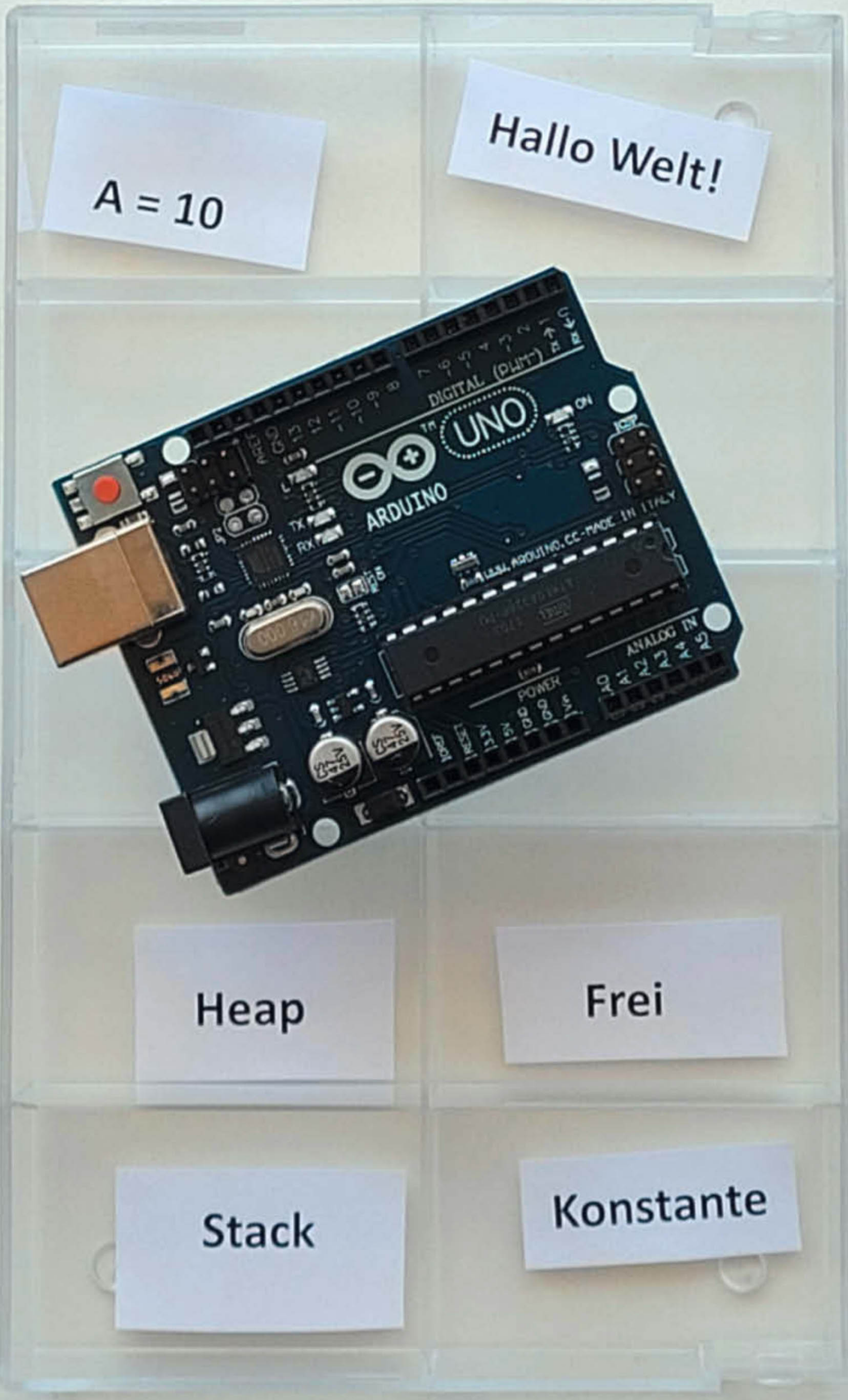
Für die reine Simulation von digitaler Logik gibt es auch einige Programme. Das ältere und beliebte „Logisim“ ist Open Source, wird aber nur noch in teilweise weiterentwickelt. Ein moderner Nachfolger soll die Open-Source-Software „Digital“ von H. Neemann sein, sicher sehr leistungsfähig und mit modernem Unterbau, aber die GUI in Java-Optik ist eher schwer zu ertragen.

—*caw*

Speicherverbrauch in Mikrocontrollern

In der Make 2/24 haben wir erklärt, an welchen Stellen im Adressraum eines Mikrocontrollers RAM, ROM und I/O liegen. Nun erklären wir, wo im RAM Programme welche Art von Daten ablegen.

von Daniel Bachfeld



Wenn man in der Arduino IDE einfache Anwendungen wie LED-blinken-lassen und Servo-Motor-ansteuern programmiert, muss man sich in der Regel nicht mit Speicheraufteilungen beschäftigen. Bei längeren und komplexeren Programmen kann es aber passieren, dass die Ressourcen des gewählten Mikrocontrollers nicht ausreichen. Passt mein Programm in den Flash? Reicht der RAM, um Bilddaten zur Laufzeit zwischenzuspeichern? Nicht selten stürzt ein Programm erst nach einer gewissen Zeit ab, weil sein RAM voll ist.

Manche Programmierer probieren dann einfach einen Mikrocontroller mit mehr Flash und / oder RAM, manche optimieren hingegen den Speicherverbrauch ihres Programms. Bei Letzterem muss man verstehen, in welchem Teil des Speichers das Programm und die statischen oder dynamischen Daten abgelegt werden.

Kurzinfo

- Verteilung von Daten im Speicher
- Tools zur Speicheranalyse
- RAM sparen mit PROGMEM

Alles zum Artikel im Web unter *make-magazin.de/xduj*

Segment

Wir beschäftigen uns hier zunächst mit der Speicheraufteilung des RAM. Übersetzt man in der Arduino IDE für den Arduino UNO den unter den Beispielen mitgelieferten Blink-Sketch,

```
void setup() {
    pinMode(LED_BUILTIN, OUTPUT);
}

}void loop() {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(1000);
    digitalWrite(LED_BUILTIN, LOW);
    delay(1000);
}
```

meldet die IDE (respektive der Linker) einen Platzverbrauch im Flash von 924 Byte und 9 Byte im „Dynamischen Speicher“ durch globale Variablen. Mit dynamisch ist hier nicht ein dynamischer RAM-Baustein gemeint, sondern ein freier Speicherbereich im RAM selbst, der wachsen und schrumpfen kann.

Der Hinweis „Leaving 2039 bytes for local variables“ bezieht sich auf Variablen im RAM innerhalb von Funktionen. Er besagt, dass wir zur Laufzeit des Programms noch 2039 Byte der insgesamt 2048 Byte des ATmega-SRAM für Variablen innerhalb von Funktionen nutzen können, dazu gleich mehr.

Die Formulierungen der Arduino IDE sind im Prinzip leichter verständlichere Übersetzungen aus der Informatikwelt. Dort teilt man Speicher in die Segmente Text, BSS, Data, Stack und Heap auf. Damit sind Adressbereiche gemeint, die eine spezielle Bedeutung haben.

Das Text-Segment bezeichnet den Speicherbereich des eigentlichen Codes, also das Programm, plus aller später zu kopierenden Konstanten und Variablen, dazu gleich mehr. Text liegt üblicherweise im Flash, also einem separaten Bereich im Mikrocontroller. In BSS sind Variablen abgelegt, die zwar im Programm deklariert sind, aber denen anfangs noch kein Wert zugewiesen wurde oder der Wert 0. BSS bedeutet in ausgeschriebener Form Block Starting Symbol und stammt aus der Frühzeit der Softwareentwicklung. Leichter zu merken ist zwar better save space, was aber leider nicht auf den Platz im RAM zutrifft, sondern nur auf die vom (Compiler und) Linker erzeugte Objekt-Datei (.elf). In Kasten „Urknalltheorie“ erklären wir kurz die ELF-Datei und was im Mikrocontroller eigentlich passiert, bevor der Sketch im Arduino gestartet wird.

Im Segment Data liegen alle Variablen (und Konstanten), die mit einem anderen Wert als 0 initialisiert wurden. Auch Zeichenketten liegen dort.

Global

Bei den in BSS und Data abgelegten Variablen handelt es sich um sogenannte globale Variablen. Das bedeutet, jeder Programmteil, auch Funktionen, können diese Variablen „sehen“. Man spricht in diesem Zusammenhang auch von Scope. Zudem sind diese Variablen während der gesamten Laufzeit des Programms vorhanden, das nennt man Lifetime. Globale Variablen und Konstanten (zumindest beim Arduino) belegen immer Speicherplatz im RAM.

Globale Variablen werden beim Arduino **vor** den Funktionen `setup()` und `loop()` deklariert (und ggfs. definiert). In normalen C-Programmen schreibt man globale Variablen vor `main()`. Zwar handelt es sich bei der Programmierung mit Arduino eigentlich auch um ganz normale C/C++-Programme, ihnen fehlt aber offenbar die `main()`-Funktion? Nein, denn im Hintergrund gibt es sie trotzdem, versteckt vor den Augen des Anwenders in den Untiefen der Arduino-Pfade in der Datei main.cpp:

```
int main(void)
{ init();
  initVariant();
#if defined(USBCON)
        USBDevice.attach();
#endif
  setup();
  for (;;) {
        loop();
        if (serialEventRun) \
                serialEventRun();
  }
  return 0;
}
```

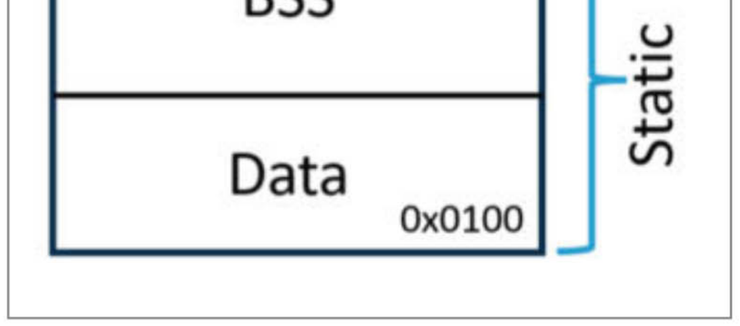


Bild 2: Die Aufteilung des RAM in Segmente, Heap und Stack. Auf dem PC liegt unterhalb von Data das Text-Segment für den Code.

```
Sketch uses 924 bytes (2%) of program storage space. Maximum is 32256 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables. Maximum is 2048 bytes.
```

Bild 1: Die Arduino IDE macht nur rudimentäre Angaben zum Speicherverbrauch eines Sketches.

Speicherverbrauch Blink.ino

```
avr-size.exe Blink.ino.elf
   text    data     bss     dec     hex filename
    924       0       9     933     3a5 Blink.ino.elf
```

Sie initialisiert die serielle Schnittstelle, ruft einmalig die `setup()`-Funktion auf und dann in einer Dauerschleife `loop()`. Zudem checkt sie regelmäßig, ob Daten über die serielle Schnittstelle rein- und rausgehen sollen. Das erklärt zum einen, warum selbst minimalistische Sketche vergleichsweise lahm sind: Die Überwachung der seriellen Schnittstelle bremst. Zudem erklärt es, warum sogar ein leerer Sketch immer 9 Byte verbraucht: Drei Variablen für den Timer0 und die Funktion `millis()` belegen den Platz.

Mit der Befehlszeilen-Toolchain des Arduino kann man sich genauer anschauen, in welchem Bereich das Programm seine Variablen ablegt. Die Toolchain liegt in der Regel unter

C:\Users\Anwender\AppData\Local\Arduino15\packages\arduino\tools\avr-gcc\7.3.0-atmel3.6.1-arduino7\bin\

wobei hier die Versionsnummer abweichen kann. Die Sketches liegen unter

C:\Users\Anwender\AppData\Local\Temp\arduino\sketches\

gefolgt von einer kryptischen Zeichenkette. Liegen dort mehrere Ordner, findet man anhand des Datums und der Uhrzeit schnell heraus, welche der aktuell übersetzte Sketch ist.

Mit dem Befehl `avr-size` kann man sich die Verteilung der Segmente in der ELF-Datei anschauen, wie im Kasten „Speicherverbrauch Blink.ino“ zu sehen. Die ELF-Datei beschreibt den Aufbau des Speichers, wie er im laufenden Betrieb des Arduino aussehen soll (siehe auch Kasten „Urknalltheorie“).

Die Größe der ELF-Datei ergibt sich aus der Summe von Text, Data und BSS. Hier sehen wir, dass die Variablen in BSS liegen und in der Tat werden sie in den eingebundenen Bibliotheken mit 0 initialisiert.

Will man die Verteilung der Variablen im BSS genauer aufdröseln, nimmt man das Tool `avr-objdump`. Als Argumente gibt man `-C` für decode, `-t` zum Anzeigen der Symbol Table und `-j .bss` zur Anzeige nur des BSS-Segments an.

Im Kasten „BSS-Segment Blink.ino“ sieht man prima die drei Variablen und wie ihr Speicherplatz zusammen 9 ergibt. Und man sieht, dass der BSS-Bereich bei Adresse 0x100 startet (das Prefix 00800 ignorieren wir hier mal) und bei 0x109 endet! Das liegt daran, dass der SRAM im ATmega328 hardwaretechnisch erst bei 0x100 beginnt (Vergleich Bild 3).

Fügt man in den Blink-Sketch vor `setup()` die Zeile `int global = 1;` ein, so erscheint die Variable mit ihren 2 Byte im Datensegment. Hinweis am Rande: Man muss mit der Variable im folgenden Code noch irgendwas anstellen, etwa inkrementieren. Andernfalls optimiert der Compiler sie einfach weg, weil sie augenscheinlich keine Funktion hat.

Das Data-Segment liegt vor dem BSS-Segment, sodass sich nun alles verschiebt. BSS beginnt nur erst ab 0x102 und geht bis 0x010b (siehe Kasten „Datensegment Blink.ino“).

Lokal

So weit zu den beim Start in den Speicher eingemeißelten Variablen. Platzsparender als globale Variablen sind lokale Variablen. Sie haben eine begrenzte Sichtbarkeit und Lebensdauer und werden innerhalb von Funktionen deklariert. Sie werden erst beim Aufruf der jeweiligen Funktion angelegt, und zwar auf dem sogenannten Stack. Der Stack ist der dynamische Teil des Speichers und wächst vom oberen Ende gegen das untere Ende des Speichers wie ein Stalaktit an der Decke einer Tropfsteinhöhle. In unserem Arduino-Beispiel beginnt der Stack bei 0x8FF (dezimal 2023), also der letzten Zelle des 2048 Byte kleinen RAM des ATmega328.

Anders als bei globalen Variablen kann man schwer voraussagen, wie groß der Stack zur Laufzeit wird. Sein Füllstand hängt vom Programmablauf ab und der kann ja durchaus verschlungen sein.

Nehmen wir mal an, wir rufen eine C-Funktion `eins()` ohne weitere Parameter auf. Das Programm legt zuerst die Rücksprungadresse auf den Stack, also quasi den Weg zurück ins Hauptprogramm. Im Hintergrund sichert das Programm zwar noch andere Daten auf den Stack, aber zum besseren Verständnis lassen wir das mal weg.

Nun deklarieren wir in `eins()` eine Variable der Länge `uint16_t`, also 2 Byte. Die Variable legt das Programm auf dem Stack ab. Anschließend erniedrigt es den Stack Pointer um 2. Genau genommen subtrahiert es die Größe der Variablen vom Stack Pointer, was 2301 respektive 0x8FD ergibt. Ab dieser Adresse können weitere Daten gespeichert werden.

Deklarieren wir eine weitere Variable, so wächst der Stack immer weiter nach unten in Richtung Heap. Innerhalb der Funktion lassen sich alle lokalen Variablen ändern. Auch die globalen Variablen lassen sich von `eins()` aus ändern.

Die in `main()` deklarierten (lokalen) Variablen sind indes für die Funktion `eins()` nicht sichtbar. Wichtig zu wissen ist, dass auch `main()` in C-Programmen eine Funktion ist. Das heißt, in ihr deklarierte Variablen sind auch lokale Variablen, auf die nur `main()` selbst zugreifen kann. Aus `main()` heraus aufgerufene Funktionen können nicht auf sie zugreifen.

Endet die Funktion, so addiert das Programm die Größe aller zuvor angelegten Variablen wieder zum Stack Pointer dazu, der Stack schrumpft. Prinzipiell liegen die Werte der Variablen allerdings noch im Speicher, da sie ja nicht wirklich gelöscht wurden. Aber das stört nicht weiter.

Zuletzt liegt noch die Rücksprungadresse auf dem Stack, die nun eingelesen und angesprungen wird, womit der Stack Pointer aber-

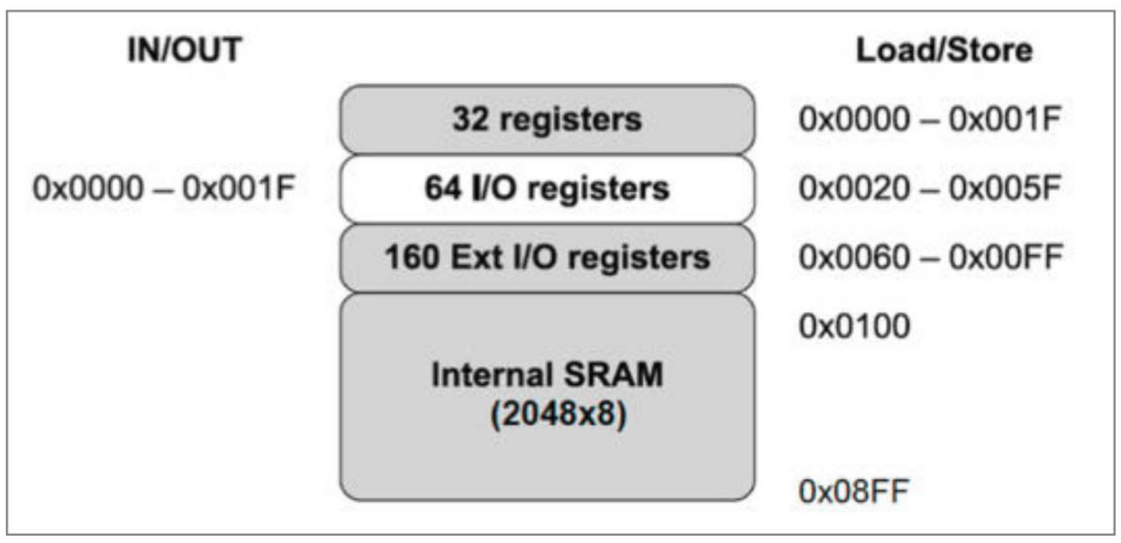


Bild 3: Der RAM im ATmega beginnt erst ab Adresse 0x0100. Davor liegen noch diverse I/O-Register.

BSS-Segment Blink.ino

```
avr-objdump.exe -C -t -j .bss Blink.ino.elf

Blink.ino.elf:     file format elf32-avr

SYMBOL TABLE:
00800100 l    d  .bss   00000000 .bss
00800105 l     O .bss   00000004 timer0_overflow_count
00800101 l     O .bss   00000004 timer0_millis
00800100 l     O .bss   00000001 timer0_fract
00800109 g       .bss   00000000 __bss_end
00800100 g       .bss   00000000 __bss_start
```

mals um 2 Byte (so lang ist die Adresse) verändert wird und man wieder in `main()` landet.

Würde aus der Funktion `eins()` eine andere Funktion `zwei()` aufgerufen, so würde es gleich ablaufen: Rücksprung sichern, Variablen erzeugen, Stack Pointer erhöhen. Und bei Ende der Funktion rollt sich quasi alles wieder auf.

Zwar sind lokale Variablen platzsparender, weil sie nach der Rückkehr aus einer Funktion gelöscht sind. Wenn sie allerdings in `main()` bzw. in `loop()` deklariert sind, liegen sie zur gesamten Laufzeit des Programms auf dem Stack. Erst wenn `main()`, also das ganze Programm, beendet wird, würde auch dieser Platz freigegeben. Nur interessiert das dann keinen mehr. Insofern macht es letztlich keinen Unterschied, ob die Variablen in BSS bzw. Data oder dem Stack liegen.

Daten-Segment Blink.ino

```
\avr-objdump.exe -C -t -j .data Blink.ino.elf

Blink.ino.elf:     file format elf32-avr

SYMBOL TABLE:
00800100 l    d  .data  00000000 .data
00800100 l     O .data  00000002 global
00800102 g       .data  00000000 __data_end
00800102 g       .data  00000000 _edata
00800100 g       .data  00000000 __data_start

avr-objdump.exe -C -t -j .bss Blink.ino.elf

Blink.ino.elf:     file format elf32-avr

SYMBOL TABLE:
00800102 l    d  .bss   00000000 .bss
00800107 l     O .bss   00000004 timer0_overflow_count
00800103 l     O .bss   00000004 timer0_millis
00800102 l     O .bss   00000001 timer0_fract
0080010b g       .bss   00000000 __bss_end
00800102 g       .bss   00000000 __bss_start
```

Haufen

Praktischerweise muss man sich um die Verwaltung der Daten auf dem Stack keine Gedanken machen, sie kommen und gehen. Allerdings ist der Stack unpraktischer, wenn es um zusammenhängende Speicherbereiche geht, deren Größe sich erst zur Laufzeit des Programms ergibt, beispielsweise der Image-Buffer der ESP32-CAM. Man kann diverse Auflösungen und Farbtiefen einstellen, deren Speicherbedarf sich erst im Betrieb ableiten lässt. Für die dynamische Belegung oder Allozierung (engl. Allocation) dient im Speicher der Heap.

Benötigt man einen Speicherbereich (Block) auf dem Heap, kann man ihn mittels der C-Funktion `malloc()` zur Laufzeit anfordern. Dazu muss man ihr sagen, wie viel Platz man benötigt. Im Beispiel

```
uint8_t *heap = (uint8_t *)malloc(10);
```

fordern wir 10 Byte an. `malloc()` liefert eine Adresse für einen C-Pointer auf den Datentype `uint8_t` zurück, hier `heap`. Ein Pointer ist ein Konstrukt der Sprache C, um gezielt auf Speicheradressen und ihre Inhalte zugreifen zu können. Die Zeichenkette `(uint8_t *)` vor `malloc()` passt dabei den Typen des übergebenen Werts an, man spricht hier auch von Casting. Ohne die Zeichenketten geht es auch, allerdings warnt die Arduino IDE stets, die Typen würde nicht zueinanderpassen.

Der Pointer `heap` zeigt auf das erste Element und könnte nun beispielsweise wie ein Array behandelt werden. Über die Array-Elemente `heap[0]` bis `heap[9]` ließen sich Daten auf den Heap schreiben. Leider wird es hier für viele Programmierer unübersichtlich, weil man sich neben Arrays noch etwas tiefer mit Pointern und Pointerarithmetik auseinandersetzen muss, wenn man alles selber programmieren will. Alternativ kann man in C++ mit dem `new`-Operator Speicher für Objekte anfordern.

Bei Projekten für kleinere Arduinos wie dem UNO wird man Code mit `malloc()` eher selten begegnen. Der ESP32 ermöglicht mit seinem viel größeren Speicher erheblich komplexere Anwendungen, in denen Rege Gebrauch von `malloc()` gemacht wird.

Freigeben

Um den mit `malloc()` reservierten (allozierten) Speicher wieder freizugeben, ruft man die Funktion `free()` auf. Benötigen wir die im obigen Beispiel angelegten 10 Byte nicht mehr, so löscht `free(heap)` den Eintrag in der Verwaltung. Das Problem: Der Speicher ist zwar freigegeben (dealloziert), der Pointer `heap` zeigt aber immer noch dahin. Nutzt man den Pointer

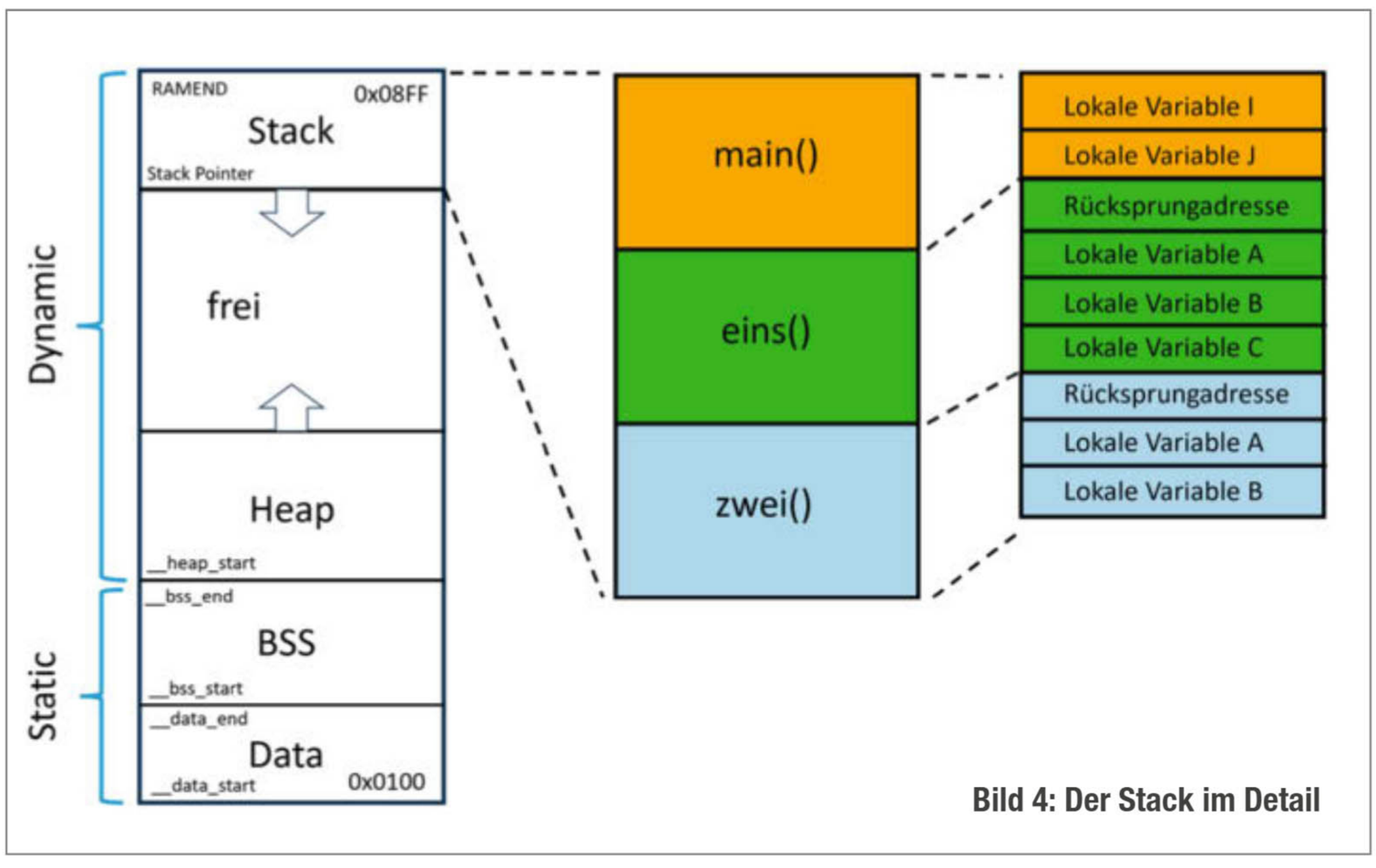


Bild 4: Der Stack im Detail

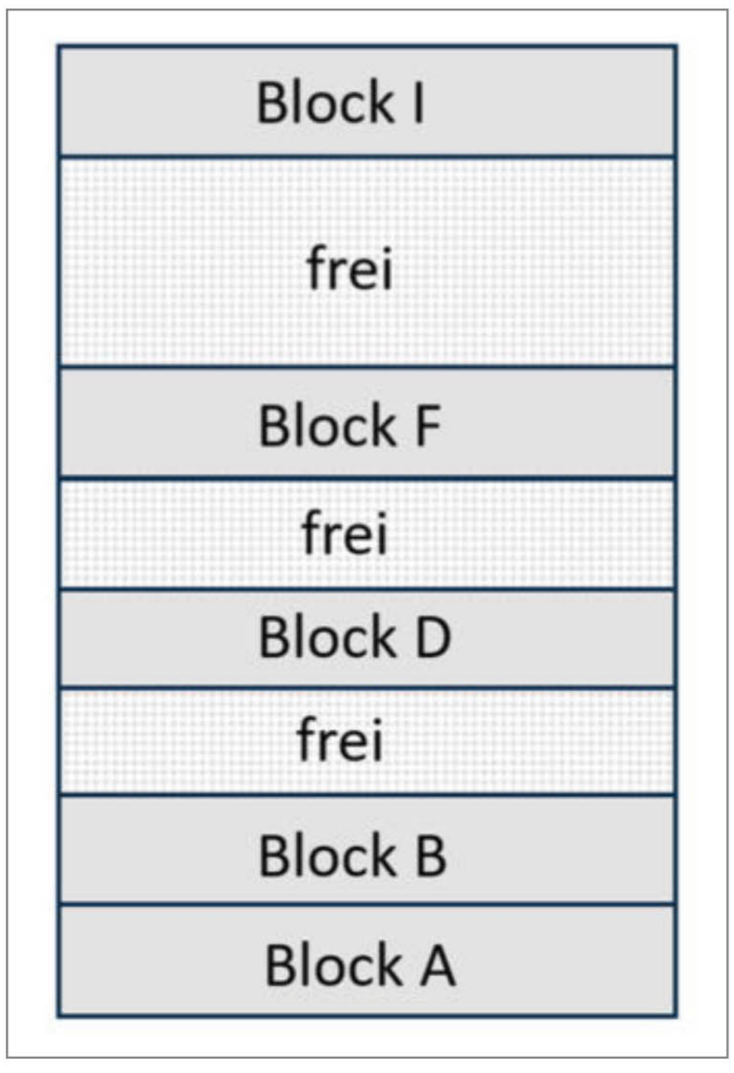


Bild 5: Werden Speicherbereiche dealloziert, entstehen Lücken im Heap.

```
PROGRAM MEMORY

avr-size.exe -C Patchwork.ino.elf
AVR Memory Usage
----------------
Device: Unknown

Program:    1612 bytes
(.text + .data + .bootloader)

Data:        201 bytes
(.data + .bss + .noinit)


Mit PROGMEM


avr-size.exe -C Patchwork.ino.elf
AVR Memory Usage
----------------
Device: Unknown

Program:    1614 bytes
(.text + .data + .bootloader)

Data:        191 bytes
(.data + .bss + .noinit)
```

später fälschlicherweise, kann er Daten überschreiben. Man spricht hier auch von Dangling Pointern. Sie gefährden die Zuverlässigkeit und Sicherheit von Systemen.

Das Programm verwaltet die belegten Bereiche im Heap über verkettete Listen. Jeder Block hat einen Header mit Größe sowie vorhergehendem und folgendem Bereich. Denkt man genauer darüber nach, kommt man schnell darauf, dass man bei unserem 10-Byte-Beispiel insgesamt mehr Speicher verbraucht. Erstens benötigen wir einen 16-Bit-Pointer, um die Adresse des jeweiligen Blocks im Heap zu speichern. Zweitens enthält der Block noch 2 Byte für die verkettete Liste: Größe, Zeiger auf vorherigen Block und folgenden Block. Insgesamt kommt man so auf 14 Byte. Das ganze lohnt sich also nur, wenn man größere Blöcke reserviert.

Es gibt noch einen weiteren Nachteil des Heaps: Wiederholtes Allozieren und Deallozieren führt in der Regel zu einer Fragmentierung des Heaps: Bestimmte Bereiche sind freigegeben, andere sind noch belegt. Man erinnere sich an die nach einer gewissen Nutzungszeit fragmentierten Festplatten unter alten Windows-Versionen, die keinen größeren, zusammenhängenden Speicherplatz mehr anbieten konnten.

Speicherlecks

Ein noch viel größeres Problem in Zusammenhang mit Heaps in der Praxis ist aber, dass Programmierer vergessen, die `free()`-Funktion einzusetzen, wenn der Speicher nicht mehr benötigt wird. So wird etwa bei wiederholtem Aufruf einer Funktion immer wieder neuer Speicher reserviert, aber vor Verlassen der Funktion nicht freigegeben. Nach und nach wächst der Heap wie ein Stalagmit in der Tropfsteinhöhle von unten dem Stack (Stalaktit) entgegen. Berühren sich beide, ist der Speicher voll. Das Programm stürzt in der Regel ab bzw. macht seltsame Dinge. Da man über die Zeit den Speicher schwinden sieht, spricht man auch von einem Speicherleck (Memory Leak), als würde er auslaufen.

Praktischerweise kann man zur Laufzeit seinen Speicherverbrauch kontrollieren. Die Arduino-Lib MemoryUsage hilft, Speicherlecks auf die Spuren zu kommen. Sie zeigt Anfang und Ende von Stack und Heap an, sodass sich Speicherfresser aufspüren lassen.

Verschieben

Wenn so wenig RAM in Mikrocontrollern vorhanden und der Flash eher üppig ist, lohnt es sich, zu überlegen, welche Teile vielleicht im Flash verbleiben sollen und nicht ins RAM kopiert werden! Wir erinnern uns: Im Flash liegt sowieso alles, vom Code bis zu den später benutzten Variablen, Konstanten und Zeichenketten. Man muss dem Compiler bzw. dem Linker eigentlich nur sagen, welche Teile beim Start NICHT ins RAM kopiert werden sollen.

Warum aber kopiert der Startup-Code beim Arduino dann überhaupt Konstanten in den RAM? Weil der Compiler und Linker den einfachsten Weg wie auf einem PC nehmen und alle Daten gleich behandeln, egal ob Variable oder Konstante. Technisch gesehen ist ein Zugriff auf Daten im Flash zur Laufzeit für ein Programm nämlich erheblich aufwendiger. Der ATmega beruht auf der Harvard-Architektur und bringt für das Lesen und Speichern von Flash-Daten dedizierte Assembler-Befehle Load Programm Memory (LPM) und Store Program Memory (SPM) mit. Die laufen aber langsamer als die normalen Befehle Load Direct From SRAM (LDS) und Store Direct to SRAM (STS). Funfact: PIC16-Mikrocontroller

```
1   const char Text[] = "Hallo SRAM";
2   const PROGMEM char Zeichenkette[] = "Hallo Flash" ;
3   const uint8_t Konstante = 10;
4   uint8_t Variable = 0;
5   uint8_t * heap;
6
7   void setup() {
8     static uint32_t i_will_survive = 0;
9     i_will_survive++;
10  }
11
12  void loop() {
13    heap = (uint8_t *)malloc(10 * sizeof(uint8_t));
14    uint32_t test = 0;
15    Serial.print(Zeichenkette);
16    Serial.print(Text);
17    Variable++;
18    uint16_t p = (uint16_t) &Konstante;
19    Serial.print(p);
20  }
```

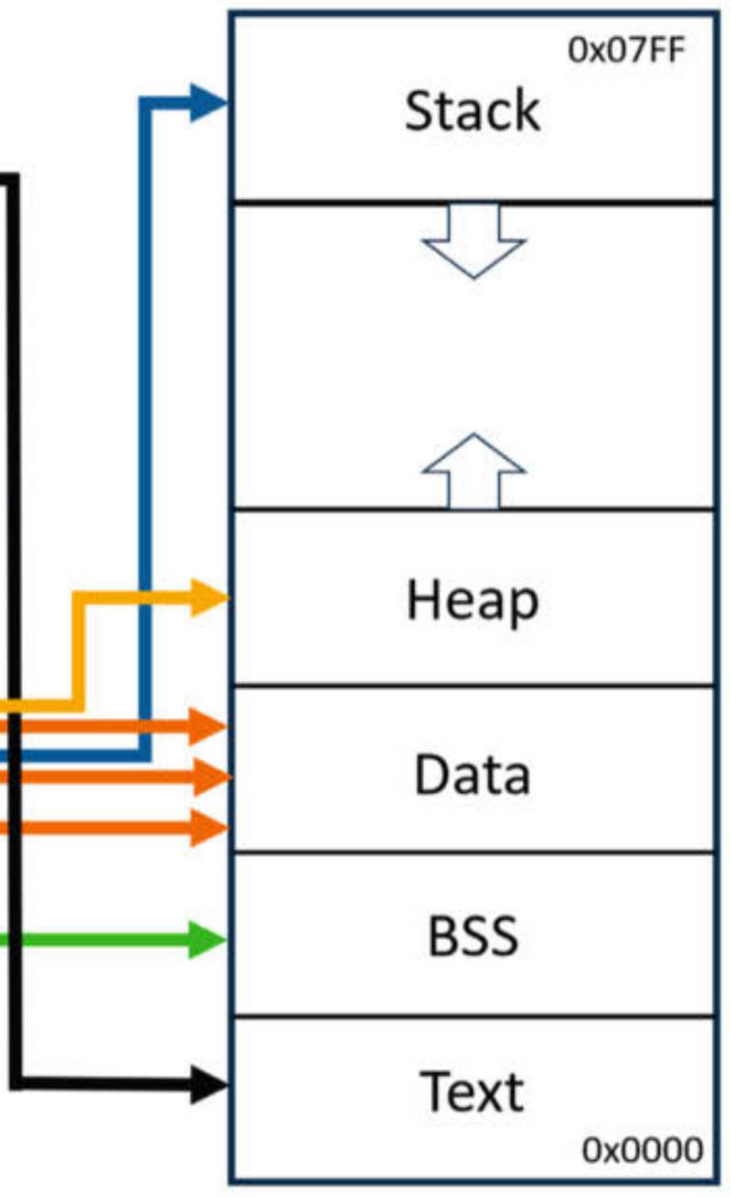


Bild 6: An diesen Stellen landen die Variablen, Konstanten und Zeichenketten im Speicher. Im Listing ist ein klassischer Fehler versteckt, der zu einem Speicherleck führt!

haben gar keinen dedizierten Befehl zum direkten Lesen von Daten aus dem Flash.

Man muss dem Compiler in der Arduino IDE also explizit mitteilt, dass er Konstanten und insbesondere speicherplatzfressende Zeichenketten für Text auf Displays oder das Debuggen über die serielle Schnittstelle nicht ins RAM kopieren soll. Vielmehr soll er auf die ohnehin schon im Flash vorhandenen Daten zugreifen.

PROGMEM

Die Arduino-Entwickler haben dafür den Modifizierer `PROGMEM` (Program Memory) eingeführt, den man seinen Konstanten hinzufügt. Abhängig von der Version des Compilers funktionieren folgende Kombinationen, wobei `{}` die Zeichenkette enthält:

```
const char Text[] PROGMEM = {};
const PROGMEM char Text[]  = {};
```

Sobald eine Konstante auf diese Weise modifiziert wurde, kann man nur noch mit speziellen Funktionen auf sie zugreifen, die in den Program Space Utilities (pgmspace.h) der Arduino-Bibliothek zu finden sind. Um auf Teile der Zeichenkette in `Text[]` zuzugreifen, würde etwa die Funktion

```
pgm_read_byte(&Text[i]);
```

dienen. Für Insider: Genau genommen handelt es sich um Makros, hinter denen eingebetteter Assemblercode für den LPM-Befehl steckt.

Achtung: Derart modifizierte Konstanten müssen als globale Konstanten definiert sein! Sie funktionieren nicht innerhalb von Funktionen, es sei denn, man stellt ihnen den Schlüsselwert `static` voran, also `const static char`.

Bei aktuelleren Boards mit ARM-Mikrocontrollern muss man seinen Sketches in der Arduino IDE nicht mehr explizit mitteilen, dass Konstanten aus dem Flash gelesen werden sollen. Gleiches gilt für ESP-Prozessoren. PROGMEM ist nur für AVR-Prozessoren noch sinnvoll.

Das F()-Makro

Um den Umgang mit Zeichenketten bei der Ausgabe über die serielle Schnittstelle zu vereinfachen, gibt es zusätzlich das `F()`-Makro. Es kapselt das ganze PROGMEM-Geraffel rund um eine Zeichenkette, sodass der Befehl `Serial.print(F("Hello World"));` ausreicht, um RAM zu sparen.

An dieser Stelle gibt es oft das Missverständnis, `F()` würde Zeichenketten vom RAM in den Flash kopieren. Nein, sie sind bereits im Flash und die Binärdatei, die man auf den Arduino flasht, wird durch `PROGMEM` und `F()` auch nicht größer – mal abgesehen von wenigen Bytes durch die aufwendigeren Lesebefehle.

Mit dem Tool `avr-size` kann man sehr gut beobachten, welchen Effekt der Einsatz von PROGMEM hat. Im Kasten „PROGRAM MEMORY“ sieht man die Belegung des Listings in Bild 6 vor und nach dem Einsatz von PROGMEM.

Mit PROGMEM vor `Text[]` verschwindet die Zeichenkette aus dem Data-Segment, das um 10 Byte kleiner wird. Die 2 Byte mehr im Program-Segment rühren von der zusätzlichen Leseoperation her.

Patchwork

Den Abschluss dieses Artikels soll das Bild 6 bilden, in dem zu sehen ist, an wie vielen Stellen Daten verteilt im Speicher liegen können.

Mit `avr-objdump` und der ELF-Datei kann man sich die künftige Verteilung im Speicher anschauen. Im Kasten „Patchworkspeicher“ sieht man die über die Segmente verteilten Daten. Die Zeichenkette `Text[]` liegt in Data, genauso wie die Konstante. Die mit PROGMEM modifizierte `Zeichenkette[]` liegt im Text-Segment.

Die mit 0 definierte Variable landet in BSS, zusammen mit der Variable `i_will_survive`. Müsste sie aber als lokale Variable der Funktion `setup()` nicht eigentlich auf dem Stack landen? Nein, da ihr das Schlüsselwort `static` vorangestellt ist, landet sie in BSS und überlebt dort das Ende der Funktion. Der in `loop()` mit `malloc()` reservierte Speicher zeigt auf den Heap und die Variable `test` liegt auf dem Stack. —*dab*

Patchworkspeicher

```
avr-objdump.exe -C -t -j .data Patchwork.ino.elf

Blink2.ino.elf:     file format elf32-avr

SYMBOL TABLE:
00800100 l    d  .data  00000000 .data
00800112 l     O .data  00000001 Konstante
00800113 l     O .data  0000000b Text
00800100 l     O .data  00000012 vtable for HardwareSerial
0080011e g       .data  00000000 __data_end
0080011e g       .data  00000000 _edata
00800100 g       .data  00000000 __data_start


SYMBOL TABLE:
0080011e l    d  .bss   00000000 .bss
0080012c l     O .bss   0000009d Serial
00800128 l     O .bss   00000004 timer0_millis
00800127 l     O .bss   00000001 timer0_fract
00800123 l     O .bss   00000004 timer0_overflow_count
0080011e l     O .bss   00000004 setup::i_will_survive
00800122 l     O .bss   00000001 Variable
008001c9 g       .bss   00000000 __bss_end
0080011e g       .bss   00000000 __bss_start#


SYMBOL TABLE:
...
00000072 l     O .text  0000000c Zeichenkette
....
```

Urknalltheorie

Versorgt man den ATmega328 im Arduino UNO mit Strom, so ist der einprogrammierte Sketch nicht das Erste, was ausgeführt wird. Denken wir uns den Bootloader mal weg, so übernimmt anfangs die C Runtime Zero (crt0) die Kontrolle. Sie ist kein Teil des Mikrocontrollers, sondern wird für den Anwender unsichtbar in jeden Arduino-Sketch eingebunden. Sie initialisiert nicht nur wichtige Register, sondern kopiert auch alle Variablen und Konstanten an die richtigen Stellen im RAM (Data und BSS) und ruft dann die Funktion `main()` auf, also das eigentliche Programm. Der Prozess wird mitunter auch C Copy Down genannt.

Woher weiß die Runtime, wohin was im RAM kommt? Beim Upload schreibt man nicht nur seinen Sketch in den Flash, sondern eine Datei im Executable and Linkable Format, kurz ELF. Sie ist quasi ein Abbild des gewünschten Aufbaus des Speichers und den Sektionen Text, Data, BSS. Die ELF-Datei wird vom Linker in der Arduino IDE aus dem Sketch und den Variablen und Konstanten zusammengebaut. Aus ihr heraus kann die Runtime nach dem Start die definierten Variablen aus dem Flash in Data kopieren und die nur deklarierten Variablen in BSS in das RAM kopieren.

Eigene Serverdienste mit einem Klick

Die auf einer grafischen Oberfläche basierende Software Umbrel bietet eine anfängerfreundliche Möglichkeit, Cloud-Dienste wie Home Assistant mit nur einem Klick einzurichten. Dadurch wird es einfach, Software auf einem Raspberry Pi zu verwalten und neue zu installieren, ohne sich durch lange Installationsanleitungen zu lesen.

von Daniel Schwabe

Umbrel wird wie jedes andere Betriebssystem für den Raspberry Pi installiert. Auf der offiziellen Umbrel-Website lädt man sich das Image für die passende Hardware herunter und flasht es dann mit dem Raspberry Pi Imager oder einer ähnlichen Software auf eine Micro-SD-Karte. Wie das funktioniert ist im Artikel „Verbindungen zum Raspberry Pi Server“ in Make 5/23 Seite 72 nachzulesen. Die Links zu allen Ressourcen findet man in der Online-Info.

Nachdem die SD-Karte im Raspberry Pi eingelegt und dieser mit Strom und Internet verbunden wurde, ist unter umbrel.local im Browser die Weboberfläche erreichbar. Und diese Weboberfläche hat es in sich. Statt einer Eingabeaufforderung zeigt sich Umbrel als komplett UI-basiertes System. Mit Icons für einen App-Store, Heimbildschirm und Systemeinstellungen erinnert es mehr an eine Desktopoberfläche. Es hat nichts gemein mit Textkolonnen ausspuckenden Schwarz-Weiß-Texteingaben, die man aus Filmen kennt, wenn wieder mal ein Mainframe gehackt wird.

Über den App-Store (das Tragetaschen-Icon am unteren Ende des Bildschirms) hat man Zugriff auf eine Vielzahl an Serverprogrammen, die praktische Dienste implementieren. Und diese lassen sich wie moderne Handy-Apps mit einem Klick auf Installieren auf den Server aufspielen. Es bedarf keiner Installationsskripte, keiner zusätzlichen Ressourcen, die installiert werden müssen. Nur ein Klick genügt. Im Vergleich zu unserem Artikel „Private Maker-Tools auf dem Pi“ aus Ausgabe 2/24 bedarf es mit einer Umbrel-Lösung keiner Kenntnisse über Docker oder anderes Server-Know-how.

Installation in der Praxis

Die smarte Personenwaage aus der Make 7/23 S. 70 ist ein sehr schönes Projekt (vor allem jetzt, in der perfekten Zeit nach der Oster- und vor der Weihnachtsvöllerei). Um dieses Projekt umzusetzen, installiert man über Umbrel einen Home-Assistant-Server. Dazu sucht man im App-Store nach Home Assistant und installiert es mit einem Klick.

Nachdem der Ladebalken durchgelaufen ist, befindet sich jetzt ein Home-Assistant-Logo auf dem Homescreen. Mit einem Linksklick auf dieses Logo landet man dann auch direkt im Einrichtungsassistenten von Home Assistant.

Mit einem Rechtsklick bekommt man noch weitere Optionen angeboten: Beenden, neu starten oder deinstallieren sind selbsterklärend. Hinter der Fehlerbehebung versteckt sich die Konsolenausgabe des Programms, in dem eventuelle Fehlermeldungen etc. aus den Logdateien ausgelesen werden können.

Installiert man über Umbrel ein Programm und greift danach darauf zu, gibt es drei Szenarien, wie die Software startet:

Kurzinfo

- **Serviceinstallation mit nur einem Klick**
- **Bereitstellung von Programmen über App-Store**
- **Mit Docker erweiterbar**

Checkliste

Zeitaufwand:
15 Minuten

Kosten:
60 Euro

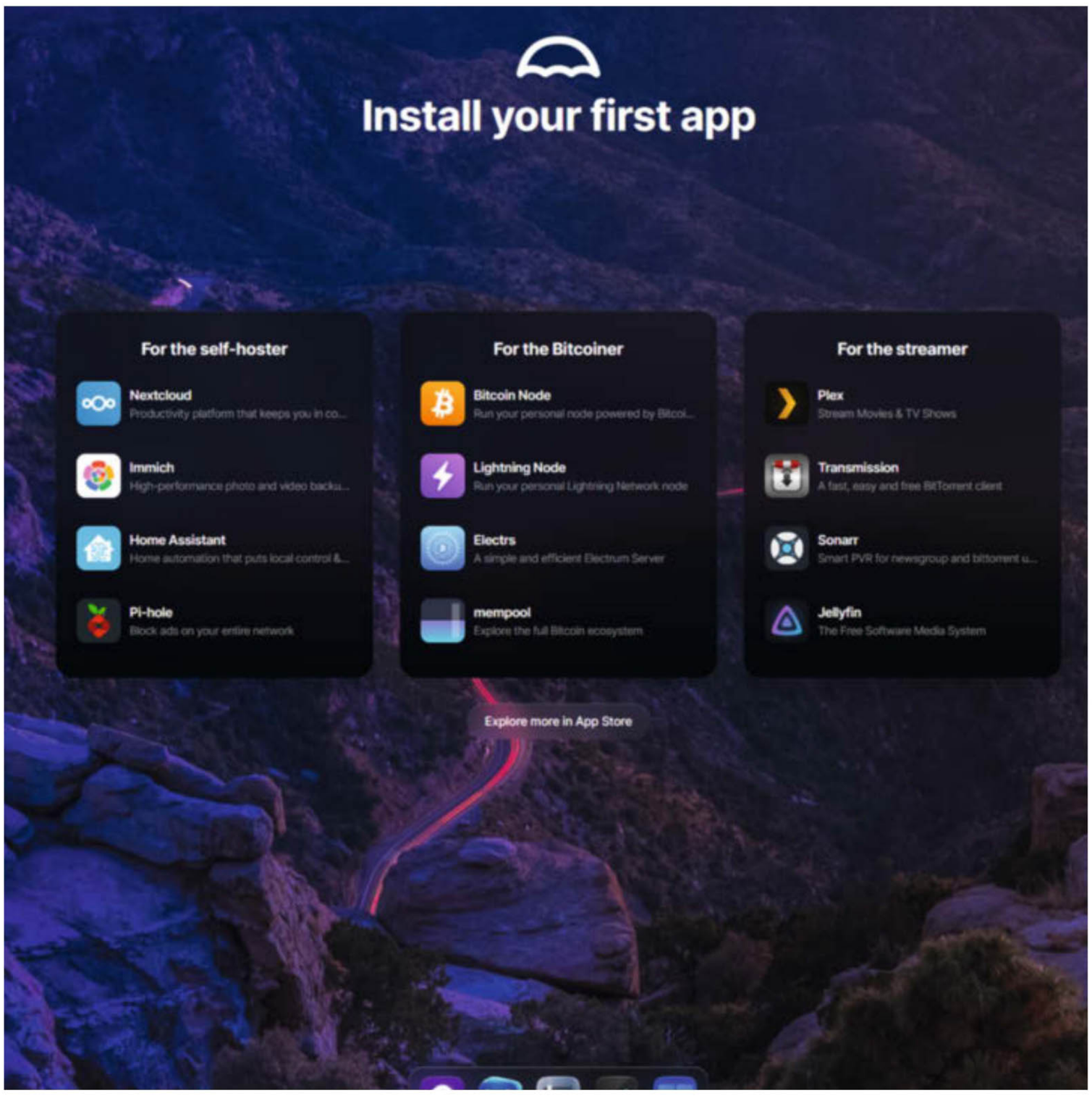
Material

- **Raspberry Pi mit Netzteil**
  Modell 4 oder 5
- **Mikro SD-Kart**

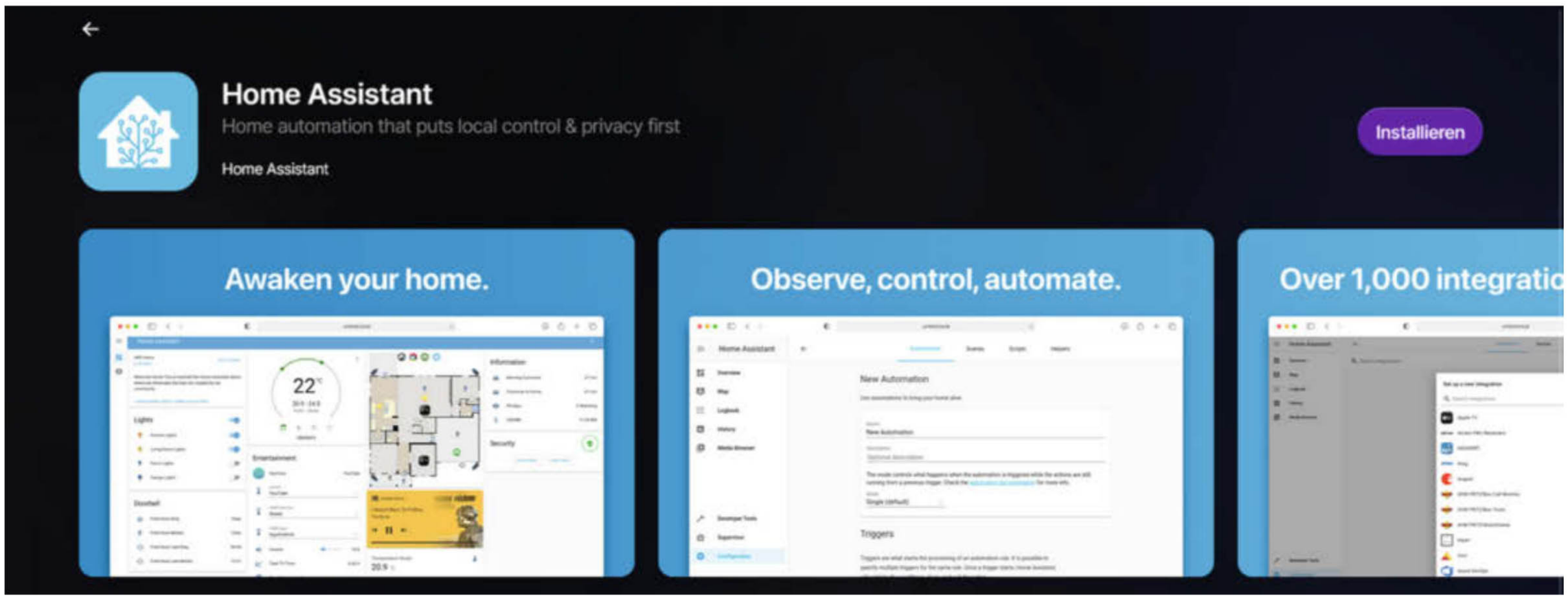
Alles zum Artikel im Web unter
make-magazin.de/x4s3

Mehr zum Thema

- Daniel Schwabe, Private Maker-Tools auf dem Pi, Make 2/24, S. 110
- Daniel Schwabe, Reverse-Proxy auf dem Raspberry Pi, Make 2/24, S. 116
- Video: Docker auf dem Raspberry Pi installieren (Einsteiger-Tutorial)



Icons und Mausunterstützung. Die Bedienoberfläche sieht mehr wie ein Desktop aus und nicht wie eine Serverkonsole. In den Einstellungen kann man sogar ein Hintergrundbild setzen.



Umbrel bietet eine Shopseite mit Bildern der Software und Informationen über die Features der Programme.

- Ein Einrichtungsassistent öffnet sich und führt einen durch die Einrichtung der Software, inklusive dem Anlegen eines Administratoren-Accounts etc.
- Die Software ist fertig eingerichtet und man kann über die Weboberfläche einen neuen Account anlegen. Das funktioniert dann wie bei jeder anderen Website auch. Account-Daten eingeben, Passwort vergeben und man kann loslegen.
- Es gibt festgelegte Daten zum Einloggen. Passwort und Name des Administratorkontos sind bereits angelegt. Diese Daten lassen sich über das Rechtsklickmenü der App auf dem Umbrel-Homescreen anzeigen.

Und das war es auch schon! Eine simple Ein-Klick-Methode, um Serverdienste auf einem Raspberry Pi zu installieren. Wer nicht mehr braucht, kann jetzt aussteigen. Wer noch mehr aus Umbrel herausholen will, kann aber gerne weiterlesen.

Docker-Zugriff

Was wäre ein Raspberry-Pi-Artikel ohne Docker? Da es nicht alle Programme im Umbrel-App-Store gibt, muss man für externe Wünsche selber Hand anlegen. Dafür gibt es auf Umbrel die Möglichkeit, Portainer zu installieren. Was Portainer ist, und wie genau es funktioniert, erklärt Make-Chefredakteur Daniel Bachfeld in dem Video in der Online-Info.

Auf der Shopseite von Portainer fallen als Erstes die Warnhinweise auf. Sie weisen darauf hin, dass Portainer in diesem Szenario nur mit benannten Docker-Volumes benutzt werden soll und dass man beim Installieren einer Software über Portainer auf eventuell von anderen Umbrel-Apps belegte Ports achten soll.

Belegte Ports hängen immer von bereits installierter Software ab. Sollte man aus Versehen einen belegten Port verwenden wollen, wird man von Portainer mit einer großen und gefährlich aussehenden Fehlermeldung darauf hingewiesen, dass etwas nicht stimmt. Hier kann also erst einmal nichts schiefgehen.

Um ein benanntes Volumen in Portainer zu erzeugen, öffnet man die Portainer-Weboberfläche, wählt die lokale Docker-Umgebung aus und klickt auf der linken Seite auf das Menü Volumes. Dort dann auf der rechten Seite auf „Add Volume“.

Im sich öffnenden Assistenten kann im Feld Name ein Volumenname vergeben werden. Die anderen Felder können so bleiben, wie sie sind. Am Ende auf „Create the volume“ klicken und das Volume ist angelegt.

Um jetzt so ein Volume zu verwenden, benutzt man z. B. in einer Docker-Compose-Datei (in Portainer unter Stacks zu finden) den Namen des Volumes wie einen Pfad, auf den dann der Ordner im Container gelegt wird.

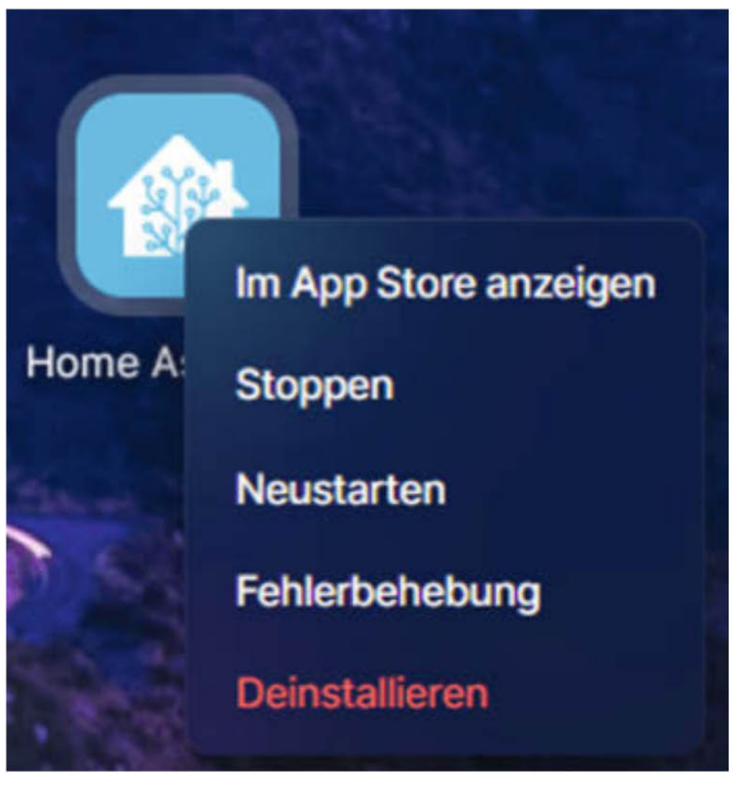
Das ist zum Beispiel ein Stack für einen Apache-Webserver. In Zeile 7 wird festgelegt, dass der Ordner /usr/local/apache2/htdocs im Container auf das Volume MeinVolume gemapped werden soll.

Dazu muss man in den Zeilen 9 bis 11 festlegen, dass das verwendete Volume external ist. Das bedeutet, dass es dieses Volume bereits gibt und es nicht neu erstellt werden muss. Mit dieser Einstellung bleiben die Daten von in Portainer erstellten Containern immer persistent.

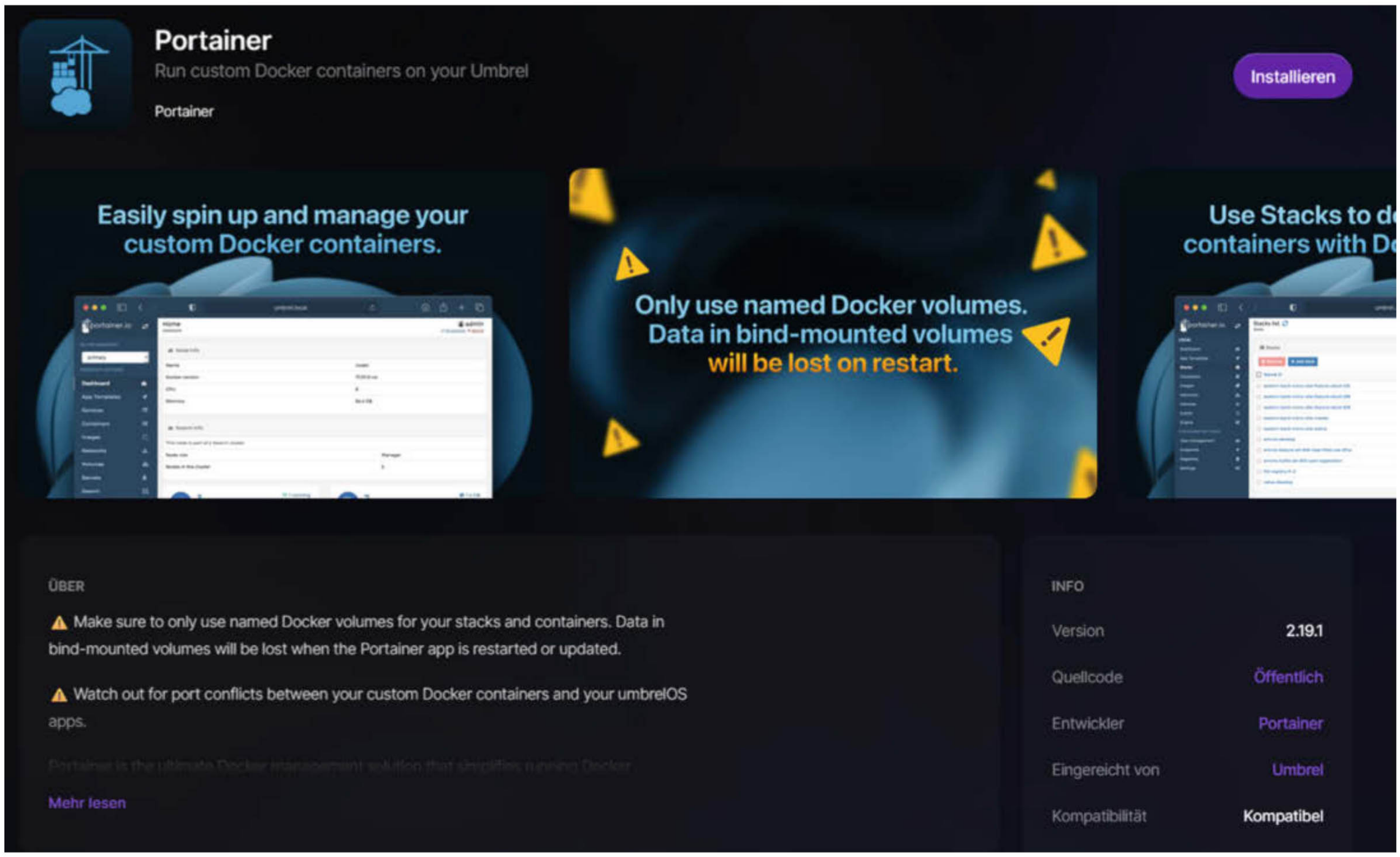
Es wird offiziell: Eine App für Umbrel erstellen

Um eine eigene App für Umbrel zu erstellen, gibt es in der Online-Info eine verlinkte Dokumentation zu dem Thema. Im Endeffekt ist es so: Unter der Haube benutzt Umbrel Docker. (Wer hätte damit gerechnet?) Allerdings ist die Konfiguration ein bisschen komplexer. Als Erstes benötigt jede App eine `umbrel-app.yml`-Datei. Aus dieser Datei werden die Informationen für z. B. den App-Store ausgelesen. Diese Felder müssen alle ausgefüllt sein, sonst kann es zu Problemen kommen. Die wichtigsten Informationen in so einer Datei sind die ID und der Port der Software. Die ID darf nur aus kleinen Buchstaben und Zahlen mit Bindestrichen ohne Leerzeichen bestehen. Als Port wird der Port festgelegt, über den die Software später erreicht werden soll. Das muss nicht derselbe sein, den die Software wirklich benutzt. Denn der wird in einer `docker-compose.yml` festgelegt.

Diese ist genauso aufgebaut wie jede andere Compose-Datei. Jede Software bekommt zusätzlich zu den normalen Containern, aus denen sie aufgebaut wird, noch eine Proxy-App. Diese leitet dann vom Port, der in umbrel-app.yml festgelegt wurde, auf den eigentlichen Port der Software um. Dafür wird folgender Code zu einer Compose-Datei hinzugefügt:



Ein Rechtsklickmenü gibt direkten Zugriff auf wichtige Kommandos für einzelne Serverprogramme.

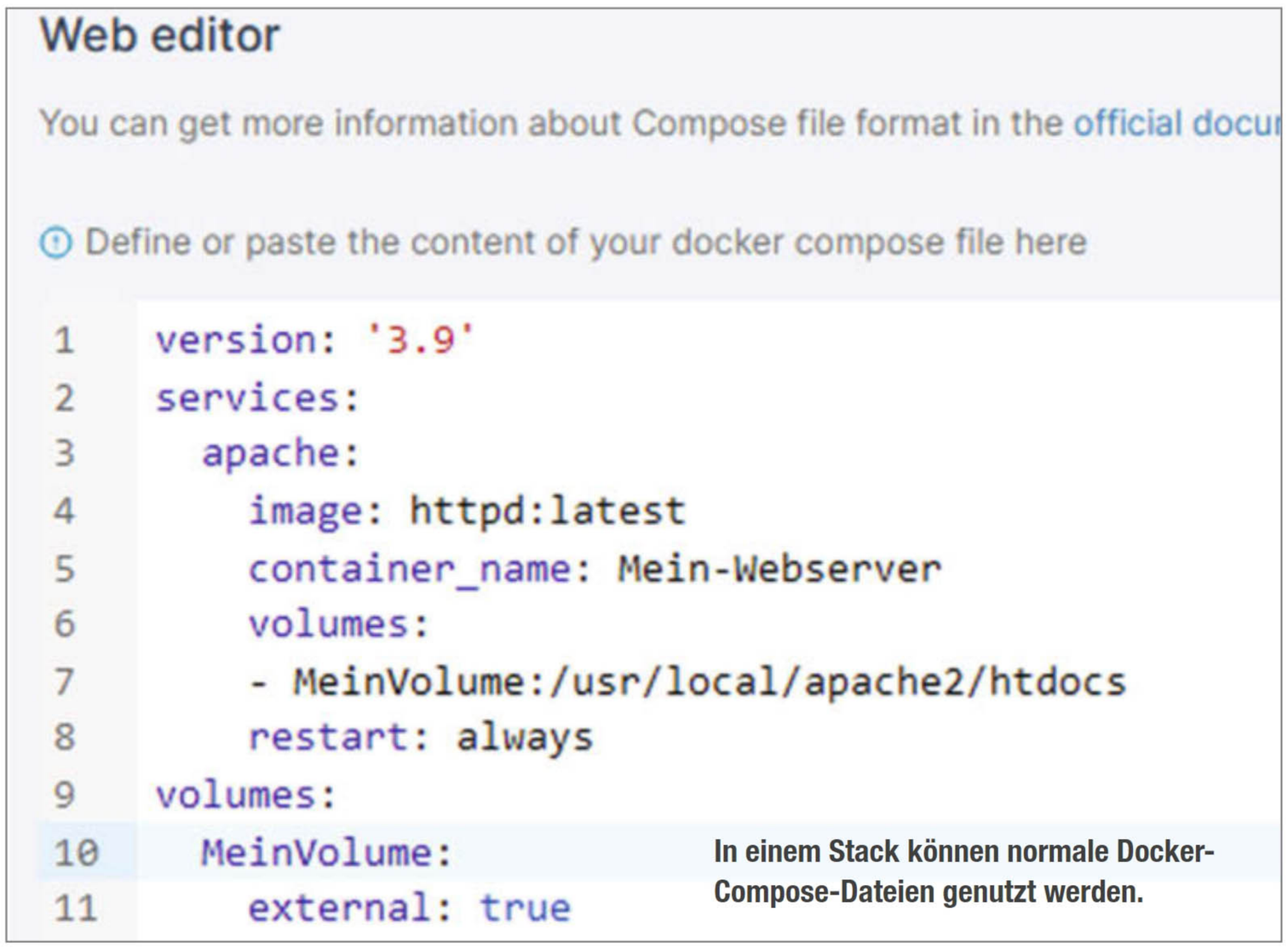


Das Docker-Frontend Portainer ermöglicht eine grafische Verwaltung von Containern über den Browser.

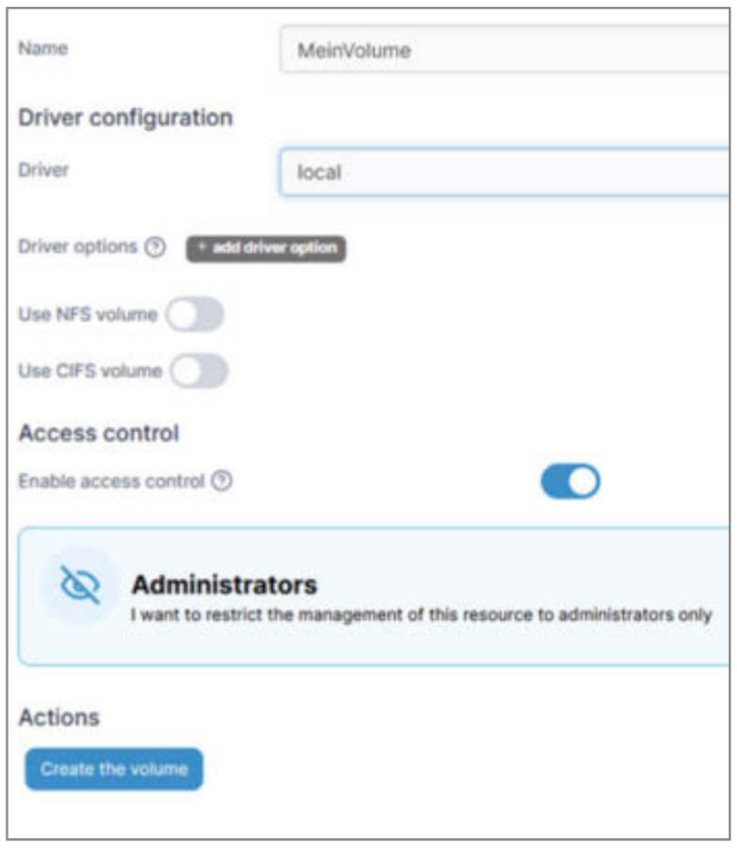
```
app_proxy:
    environment:
      APP_HOST: ID-Der-Software_
{Servicename}_1
      APP_PORT: Echter-App-Port
```

APP_HOST wird ausgefüllt mit der in `umbrel-app.yml` festgelegten App-ID, mit dem Zusatz _{Servicename}_1. Servicename wird ersetzt mit dem Namen des Services, auf den der Netzwerkverkehr umgeleitet werden muss. Im Apache-Beispiel auf der nächsten Seite wäre das dann z. B. apache aus Zeile 3.

Die restliche Konfiguration entspricht der regulären Docker-Compose-Datei einer dockerisierten Anwendung. Muss etwas auf



In einem Stack können normale Docker-Compose-Dateien genutzt werden.



Der Volumenassistent ist übersichtlich. Für den normalen Gebrauch kann man viele Optionen ignorieren.

Umbrel-app.yml Vorlage

```
manifestVersion: 1
id:
category:
name:
version: ""
tagline:
description: >-

releaseNotes: >-

developer:
website:
dependencies:
repo:
support:
port:
gallery:
path: ""
defaultUsername: ""
defaultPassword: ""
submitter:
submission:
```

```
manifestVersion: 1
id: apache
category: developer
name: Apache
version: "1.1"
tagline: Ein Webserver
description: >-
  Ein Webserver
releaseNotes: >-
  Neue Informationen zu der App
developer: Entwicklerin
website: https://httpd.apache.org
dependencies:

repo: https://github.com/apache
support: https://github.com/apache
port: 9001
gallery:
  - 1.jpg
  - 2.jpg
  - 3.jpg
path: ""
defaultUsername: ""
defaultPassword: ""
submitter: Ich
submission: Ich
```

```
version: "3.7"

services:
  app_proxy:
    environment:
      APP_HOST: apache_apache_1
      APP_PORT: 9001

  apache:
    image: httpd:latest
    volumes:
    - ${APP_DATA_DIR}/Apache-Dateien:/usr/local/apache2/htdocs
    restart: always
```

So würde eine Konfiguration für Apache aussehen.

das echte Filesystem gemapped werden, wird dort mit `${APP_DATA_DIR}` zuerst ein data-Ordner im App-Verzeichnis angegeben und danach ein Dateipfad in diesem Ordner. Für das Apache-Beispiel könnte das dann so aussehen: ${APP_DATA_DIR}/Apache-Dateien:/usr/local/apache2/htdocs.

Jetzt erstellt man einen Ordner mit dem ID-Namen der Anwendung. In diesem Ordner liegen direkt die umbrel-app.yml und docker-compose.yml. Dazu müssen dort noch alle referenzierten Ordner angelegt werden. Also hier Apache-Dateien.

Diesen Ordner muss man dann auf den Raspberry Pi in das Verzeichnis /home/umbrel/umbrel/app-stores/getumbrel-umbrel-apps-github-53f74447/ kopieren. Dazu bieten sich unter Linux das Tool scp und unter Windows WinSCP an. Ist der Ordner an der richtigen Stelle, muss man die lokale Umbrel-Website neu laden und im App-Store einfach nach dem Namen der App suchen und sie so installieren. Sollte das nicht klappen, gibt es in den Einstellungen unter „Fehlerbehebung/umbrel OS“ einen Log des Systems, in dem auch Fehlermeldungen beim Installieren von Apps angezeigt werden. Wie jede andere App auch bekommt die neue Software auf dem Homescreen auch ein Icon und kann darüber aufgerufen werden.

Fazit

Umbrel ist wirklich eine sehr einfache Lösung, um Serverdienste zu installieren. Aktuell gibt es bereits viele Maker-relevante Apps im Store. Home Assistant, OctoPrint, VS-Code und Gitea sind nur einige davon. Dadurch, dass der App-Store selbst erweitert werden kann und die Community neue Software hinzufügt, kann das Angebot an Maker-Tools bald steigen. *—das*

Life is what you

Make: it

Hannover

Maker Faire®

17.–18. Aug. 2024

Zeigt eure Projekte!

Private Maker, Bildungseinrichtungen, Makerspaces, offene Werkstätten, Vereine, uvm.

Wir haben für euch kostenfreie Standflächen.

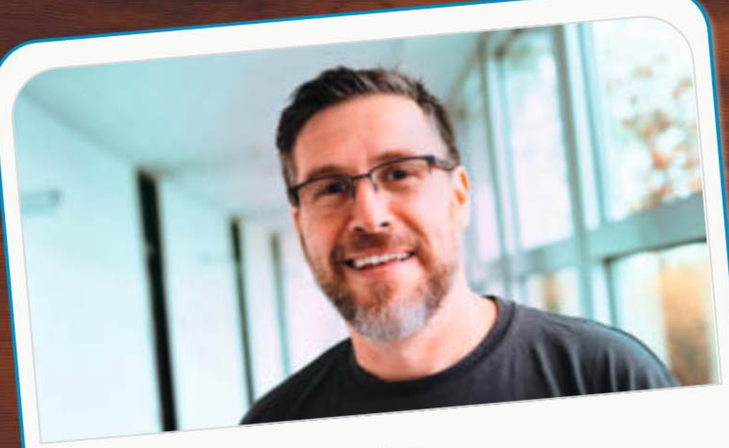
Jetzt informieren und anmelden!

Am 09.06. ist Anmeldeschluss für Aussteller.

maker-faire.de/hannover

Womit beschäftigst du dich gerade?

Unser Team



Daniel Bachfeld
Chefredakteur, dab@make-magazin.de

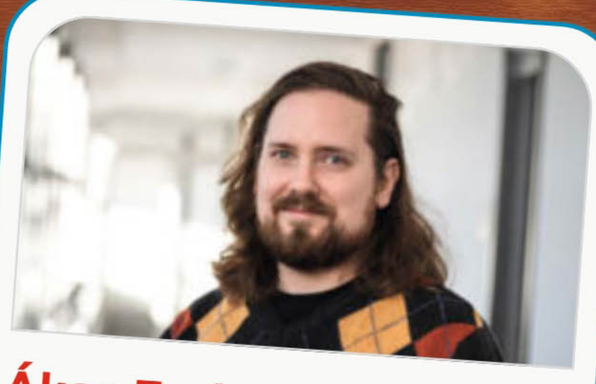
Ich beschäftige mich gerade mit der berührungslosen Füllstandsmessung von Tankinhalten, konkret für meinen Wassertank und die Gasflasche im Wohnwagen. Für kapazitive Messungen für Plastiktanks gibt es ICs wie den FDC1004. Die Lösungen für Gasflaschen funktionieren bei mir nicht zuverlässig. Ich überlege, mal Ultraschallsignale hineinzusenden und die Antwort per FFT und KI auszuwerten.

Peter König
Stellv. Chefredakteur,
pek@make-magazin.de

Gerade plane ich eine minimale Maker-Ecke für zu Hause und überlege, welcher 3D-Drucker sich da am besten einfügt. Bald kann ich die Maschinen der Make-Redaktion nicht mehr nutzen, weil ich die Firma nach diesem Heft verlasse. Maker bleibe ich natürlich lebenslang.

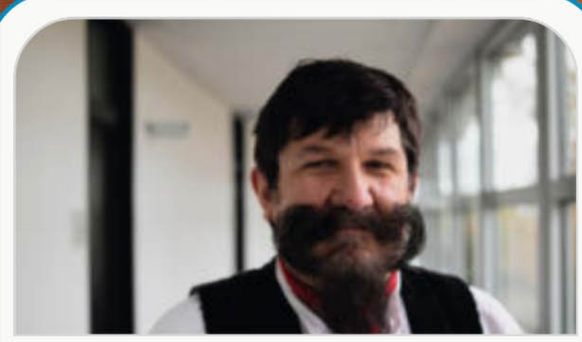
Carsten Wartmann
Redakteur, caw@make-magazin.de

Ich habe auf dem Flohmarkt einen String-Synthesizer von 1973 geschossen. Mit heftigem Wasserschaden. Nach intensiver Reinigung und ersten Reparaturen gehen Netzteil und Hauptoszillator wieder. Jetzt habe ich noch etwa 150 Kondensatoren zu tauschen, um weiterzumachen.



Ákos Fodor
Redakteur, akf@make-magazin.de

Zurzeit baue ich mir aus Einzelteilen einen großen und schnellen Zweitdrucker nach dem CoreXY-Prinzip. Neben den schicken 2020-Aluprofilen verwende ich auch ausrangierte Komponenten eines alten Bettschubsers. Bei 40 cm Bauhöhe kommt natürlich Klipper aufs Mainboard.



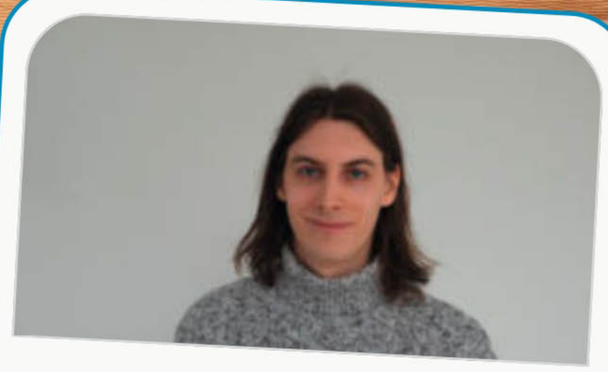
Heinz Behling
Redakteur, hgb@make-magazin.de

Ich checke gerade einen Tipp fürs kommende Heft, den ein Leser mir zum Bearbeiten von Glas mit Diodenlasern gemailt hat. Die Lösung ist einfach und kostengünstig: Zink-Grundierspray.

Johannes Börnsen

Redakteur und YouTube-Host,
jom@make-magazin.de

Ich baue gerade eine Treppe, deren Stufen auf Schränken aufliegen werden. Im Werkstattrundgang-Video (YouTube) kann man mein selbst gebautes Modell davon sehen. Wenn das klappt, will ich mich an einen Stuhl wagen. Das empfinde ich als die Königsklasse des Holzwerkens.



Daniel Schwabe

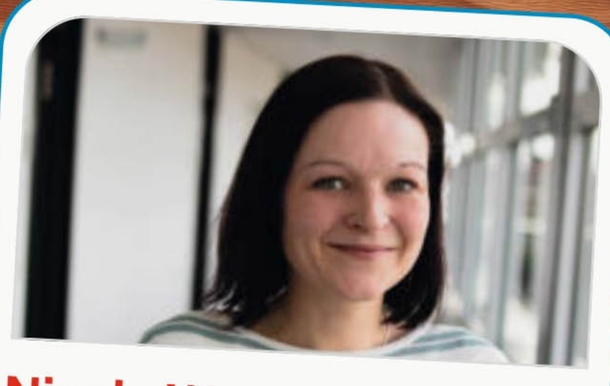
Technical Writer,
das@make-magazin.de

Ich muss zugeben, ich kann leider kein Python. Deshalb beschäftige ich mich aktuell mit dem Programmieren von Python-Programmen auf dem Raspberry Pi – inklusive der Nutzung von aufgesteckten HATs. Dazu noch ein bisschen Live-Videoübertragung und MQTT-Optimierung.

Dunia Selman

Social-Media-Managerin,
dus@make-magazin.de

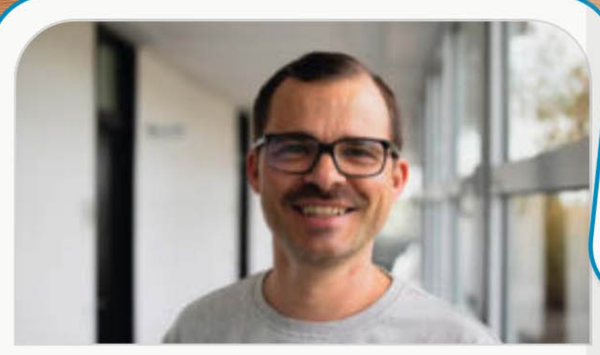
Als Social-Media-Managerin habe ich bisher relativ wenig selbst „gemaked". Nachdem ich kürzlich zum ersten Mal gelötet habe, möchte ich nun auch das 3D-Drucken lernen. Deshalb beschäftige ich mich gerade mit Slicern, Modellen und Ideen, die ich bald umsetzen möchte.



Nicole Wesche

Mediengestalterin,
niwe@make-magazin.de

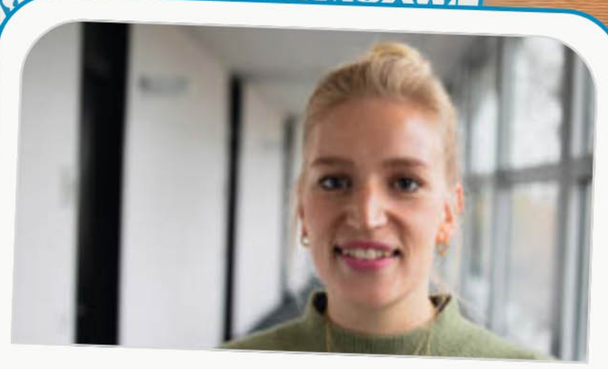
Zum 1. Mai hat die Make ein neues Grafikteam bekommen. In diesem Zuge wollen wir frischen Wind in das Magazin bringen. Ich lerne also gerade viele neue Kollegen kennen. Nebenbei teste ich – in der laufenden Produktion – jede Menge neue Schriften, sortiere Farben und Rubriken um und verschiebe Kästen.



Daniel Rohlfing

Leiter Events und Sales,
dnr@maker-faire.de

Mit der Gewinnung von letzten Partnern für unsere Jubiläumsveranstaltung. Am 9. Juni ist für die Maker Faire Hannover Anmeldeschluss. Zudem kreiere ich die Werbekampagne. Wir hoffen, dass es zur zehnten Maker Faire im Hannover Congress Centrum einen neuen Besucherrekord gibt.



Kristina Fischer

Projektleitung Maker Faire,
krfi@maker-faire.de

Ich mag alles rund um Holz, kreative Inneneinrichtung und clevere Lösungen für den Balkon. Derzeit baue ich an einem magnetischen Regal in Raketenform für die neue Tonie-Sammlung und recherchiere viel nach Palettenmatschküchen und Ikea-DIY-Hacks für das Kinderzimmer.

Louis Behrens

Werkstudent Social Media,
loub@make-magazin.de

Ich bereite derzeit Content für die Maker-Faire-Social-Kanäle vor. Neben DIY-Anleitungen, Memes und spannenden Themen zählen dazu auch Ankündigungen, Videos und wichtige Infos rund um die Messe. Ansonsten studiere ich noch im schönen Lingen und treibe viel Sport.

Nele Krautberger

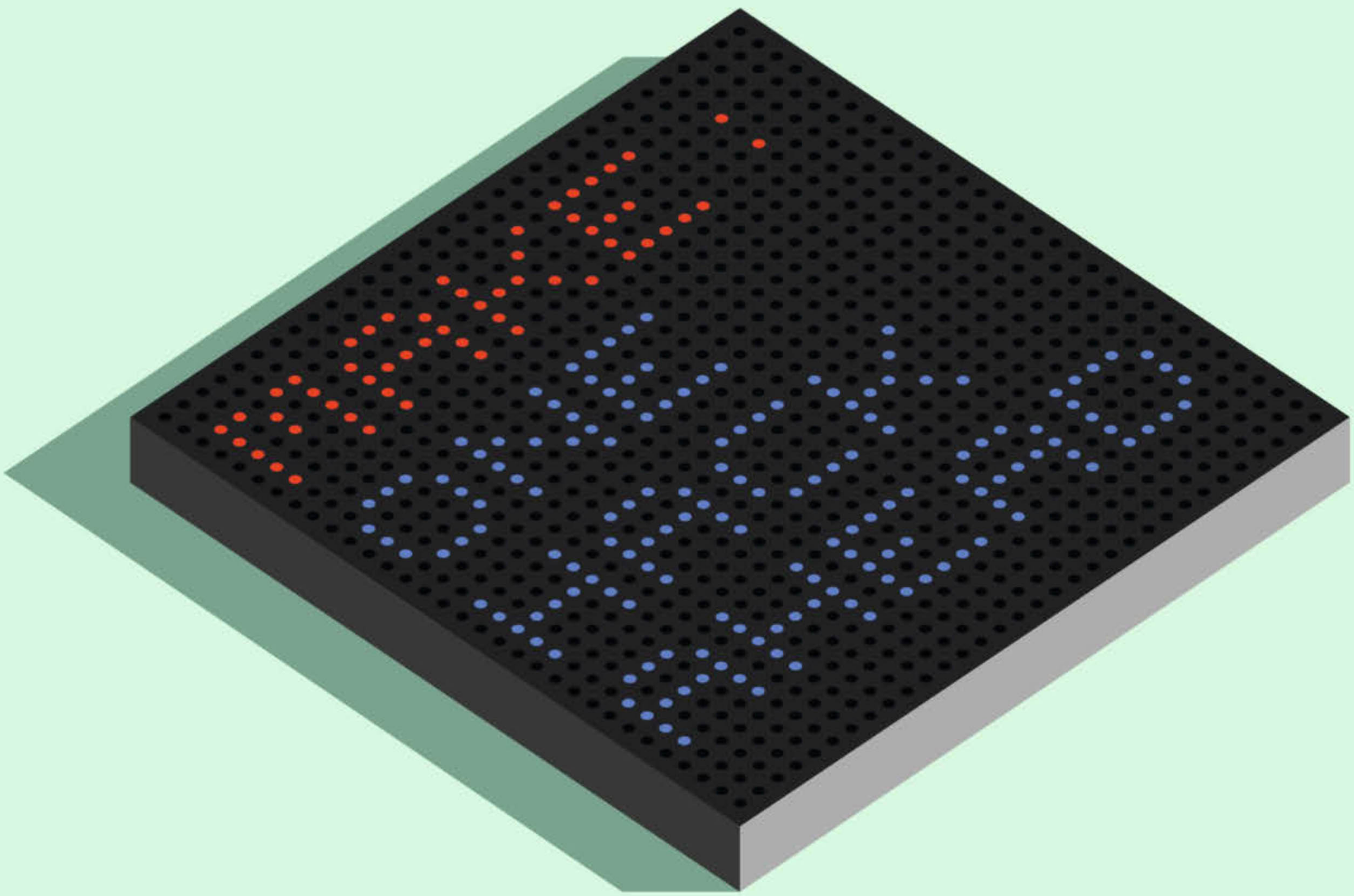
Werkstudentin Maker Faire,
nekr@maker-faire.de

Neben der Maker-Akquise für die Maker Faire fertige ich technische Zeichnungen zu verschiedenen Make-Themen an, z. B. Roboter. Außerdem beschäftige ich mich gerne mit Holz und baue zurzeit an einem Vogelhaus. Als nächstes Projekt habe ich einen Tisch in Planung.

LED-Matrizes mit MicroPython steuern

Wer etwas mit LED-Panels (HUB75) machen will, aber keine Erfahrung hat, findet mit dem Controller-Board Interstate 75 von Pimoroni einen leichten Einstieg in das Thema. Einfach draufstecken, Strom anschließen und nach ein paar Zeilen MicroPython-Code scrollt schon die erste Laufschrift über bis zu vier Panels.

von Ákos Fodor



Kurzinfo

- HUB75-Matrizes programmieren
- Pimoroni Interstate 75 kennenlernen
- Einfache Formen und Laufschriften erstellen

Checkliste

Zeitaufwand:
1 Stunde

Kosten:
40 bis 60 Euro

Material

- **HUB75-Matrix**
- **Pimoroni Interstate 75/ Interstate 75 W**
- **5-V-Netzteil ab 4 A pro Panel**
- **USB-Netzteil für das Interstate 75**

Werkzeug

- **Schraubendreher**

Mehr zum Thema

- Daniel Bachfeld, Pixelart mit Pi, Make 5/21, S. 56
- Ákos Fodor, Adafruit GFX Library: Malen mit Zahlen, Make 2/23, S. 64
- Dr. Ing. Armin Zink, IKEA-Matrix gehackt, Make 6/23, S. 64

Alles zum Artikel im Web unter *make-magazin.de/xsnh*

Wenn man das erste Mal mit einer LED-Matrix experimentieren möchte, stellt sich unweigerlich die Frage, welche man nehmen soll und wie man diese mit Inhalten bespielt. In diesem Workshop erkläre ich, wie man HUB75-Matrizes ganz leicht mit einem Mikrocontroller und MicroPython programmiert. Dafür verwende ich den Controller Interstate 75 von Pimoroni, mit dem selbst Einsteiger schnell ans Ziel kommen – ohne groß zu löten oder sich in den Anschlüssen zu verheddern. Grundkenntnisse im Programmieren sind hilfreich. Zu den gezeigten Funktionen habe ich aber auch Quellcodes in das GitHub-Repository des Projekts hochgeladen, die man sich parallel zu den Erklärungen anschauen und ausprobieren kann.

HUB75-Matrizes

Mit ihren RGB-LEDs kann eine HUB75-Matrix nahezu jede Farbe abbilden und lässt sich zu größeren Displays verbinden. Man findet Panels von unterschiedlichen Herstellern in quadratischem oder rechteckigem Seitenverhältnis mit den Auflösungen von 16 × 32 bis 128 × 128 Pixeln. Für diesen Workshop verwende ich eine Matrix von Waveshare mit 64 × 64 Pixel und P3, d. h. einem LED-Abstand von 3 mm (siehe Link in Kurzinfo), aber auch zwei 64 × 32 Pixel Panels, die ich zu einer längeren Konfiguration miteinander verkettet habe. Laut Pimoroni ist das Interstate 75 mit Panels mit 32 × 32, 64 × 32 und 64 × 64 Pixeln kompatibel – in der Bibliothek, auf die ich gleich noch im Detail eingehe, findet man aber auch einen Verweis auf ein unterstütztes Panel mit 96 × 48 Pixeln.

Rückseitig besitzt eine HUB75-Matrix drei Anschlüsse: je einen 16-poligen für den Dateneingang und einen für die Verbindung zu weiteren Panels sowie einen 4-poligen für die Stromversorgung (Bild 1). Wer mehr über die Arbeitsweise der HUB75-Panels erfahren möchte, findet in der Make 5/21 einen passenden Artikel zu dem Thema (siehe „Mehr zum Thema“ in der Kurzinfo). Für diesen Artikel reicht es aus, wenn man die einzelnen Anschlüsse unterscheiden kann.

Interstate 75

Das Pimoroni Interstate 75, das es auch in einer WLAN-fähigen Variante gibt, läuft mit einem RP2040-Mikrocontroller der Raspberry Pi Foundation und ist auf HUB75-Panels spezialisiert. Ich habe in diesem Artikel die einfache Variante ohne WLAN verwendet. Da man beide Modelle für etwa denselben Preis erhält (ca. 24 Euro), empfehle ich das Interstate 75 W, wenn man später auch Daten aus dem Internet oder dem lokalen Netz (z. B. Smarthome) auf der Matrix ausgeben möchte.

Auf der Rückseite des Boards befindet sich ein 16-poliger Stecker für die Datenübertragung, sodass man ihn einfach auf ein HUB75-Panel aufstecken kann, ohne sich Gedanken über Pinbelegungen machen zu müssen. Hinzu kommen zwei programmierbare Tasten (drei bei der W-Variante), eine NeoPixel-LED, ein Qwiic/STEMMA-Anschluss sowie weitere Pins für Sensoren und Co.

Außerdem lassen sich Panels mit bis zu 64 × 64 Pixeln Auflösung auch direkt über das Board mit Strom versorgen (zumindest bis 3 A). Dafür sitzen zwei Schraubkontakte auf der Platine, die unübersehbar mit + und - markiert sind (Bild 3). Das Board selbst verbindet man über USB mit einer ausreichend leistungsfähigen Stromquelle.

Bei meinen Experimenten mit 64 × 64 und 128 × 32 Pixeln konnte ich die verwendeten Matrizes direkt von meinem Laptop aus programmieren und (z. B. für Laufschriften) gleichzeitig mit genug Strom versorgen. Allerdings habe ich auch nicht alle LEDs eingeschaltet und bei voller Helligkeit leuchten lassen, sodass die Matrizes nicht den maximalen Strom benötigten (bis zu 4 A bei der LED-Matrix von Waveshare).

Der Betrieb über USB muss aber nicht funktionieren, z. B. wenn der Anschluss am PC nur 500 mA liefert. Um eine zuverlässige Stromversorgung sicherzustellen, ist es daher ratsam, die LED-Matrix oder mehrere davon nicht

Bild 1: Die Rückseite eines HUB75-Panels. A ist der Dateneingang, an den man das Interstate 75 anschließen muss. B ist der Datenausgang, über den man (in Pfeilrichtung) weitere Panels verbinden kann. In der Mitte ist der Stromanschluss.

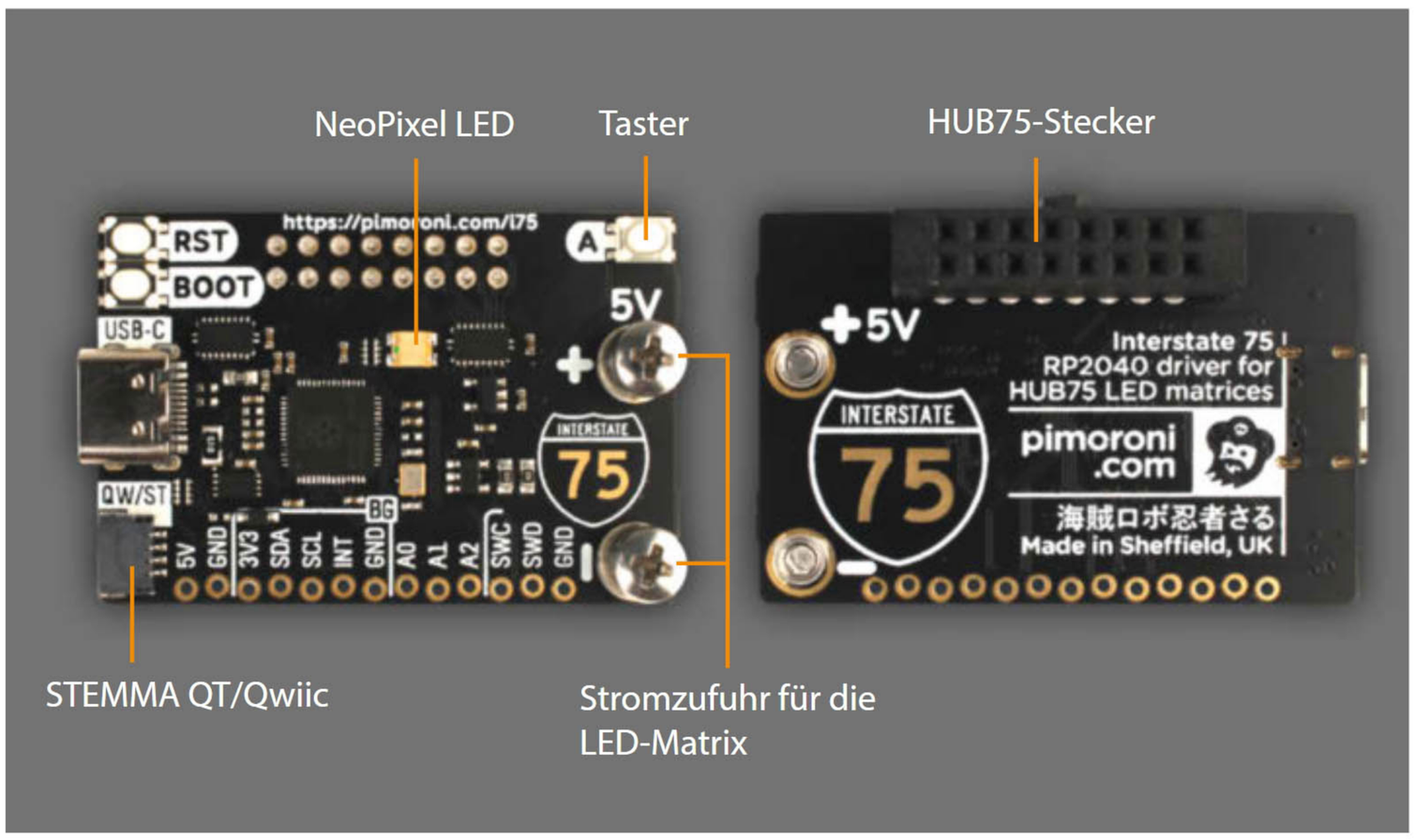


Bild 2: Das Interstate 75 von vorne und von hinten

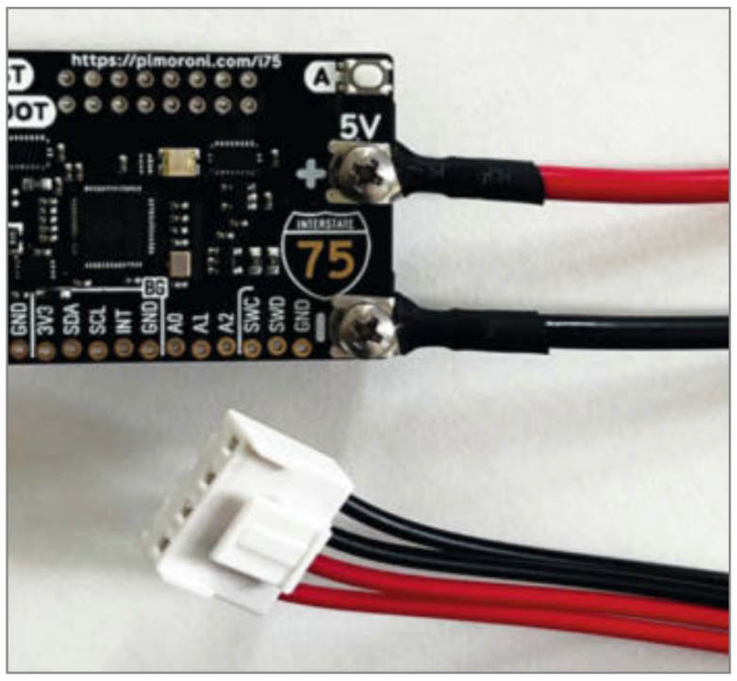


Bild 3: Das Interstate 75 kann über die beiden Schraubkontakte bis zu 3 A Strom liefern – wenn er über USB genug bekommt. Der weiße Stecker kommt auf das Panel.

über die Schraubkontakte des Interstate 75, sondern über eine externe Stromquelle zu versorgen und lediglich das Interstate 75 über USB zu betreiben.

Matrizes verbinden

Gerade Laufschriften profitieren von einer längeren Zeile. Dafür kann man mehrere HUB75-Panels miteinander in Reihe schalten. Die Bibliothek des Interstate 75 ermöglicht Auflösungen bis hin zu 256 × 64 Pixeln, also maximal vier Matrizes mit je 64 × 64 Pixeln. Tatsächlich stößt man hier auf eine erste technische Einschränkung, denn Pimoroni sieht nur eine horizontale Verkettung vor. Nicht wundern: Da man das Interstate 75 rückseitig links auf den Dateneingang steckt und dann nach rechts hin weitere Panels verbindet, sitzt das Board nach dem Aufhängen an einer Wand am rechten Ende, also im letzten Panel – auch wenn man es vielleicht genau andersherum vermuten würde. Diese Besonderheit hat aber keinen Einfluss auf das Koordinatensystem, das in der linken oberen Ecke beginnt.

Wer sich diesbezüglich mehr Flexibilität wünscht, kann das Board auch mit Adafruits CircuitPython programmieren, das noch weitere Steck-Layouts erlaubt (siehe Link in der Kurzinfo). Pimoroni weist allerdings darauf hin, dass man mit Performance-Einbußen rechnen müsse, weil CircuitPython nicht wie das modifizierte MicroPython für den Betrieb von Matrizes mit einem RP2040 optimiert sei.

In Betrieb nehmen

Wie eingangs beschrieben, kann man das Interstate 75 einfach auf den Dateneingang des (ersten) Panels stecken. Um das Board zu programmieren, benötigt man noch die Entwicklungsumgebung Thonny und die aktuelle Firmware für das Interstate 75 (siehe Links in der Kurzinfo).

Sobald alles heruntergeladen ist, hält man mit einem Finger die Boot-Taste des Interstate 75 gedrückt und verbindet das Board über ein USB-Kabel mit dem Computer. Daraufhin erscheint der RP2040 als Laufwerk in der Dateiverwaltung. Um das Board zu flashen, zieht man die UF2-Datei auf das RPI-RP2-Laufwerkssymbol. Danach startet der Mikrocontroller sich neu und man kann zu Thonny wechseln. Dort gibt es ganz rechts unten im Programmfenster das Verbindungsmenü, in dem man das Board mit „MicroPython - Board in FS mode“ auswählt (Bild 4). Jetzt ist man bereit zum Programmieren.

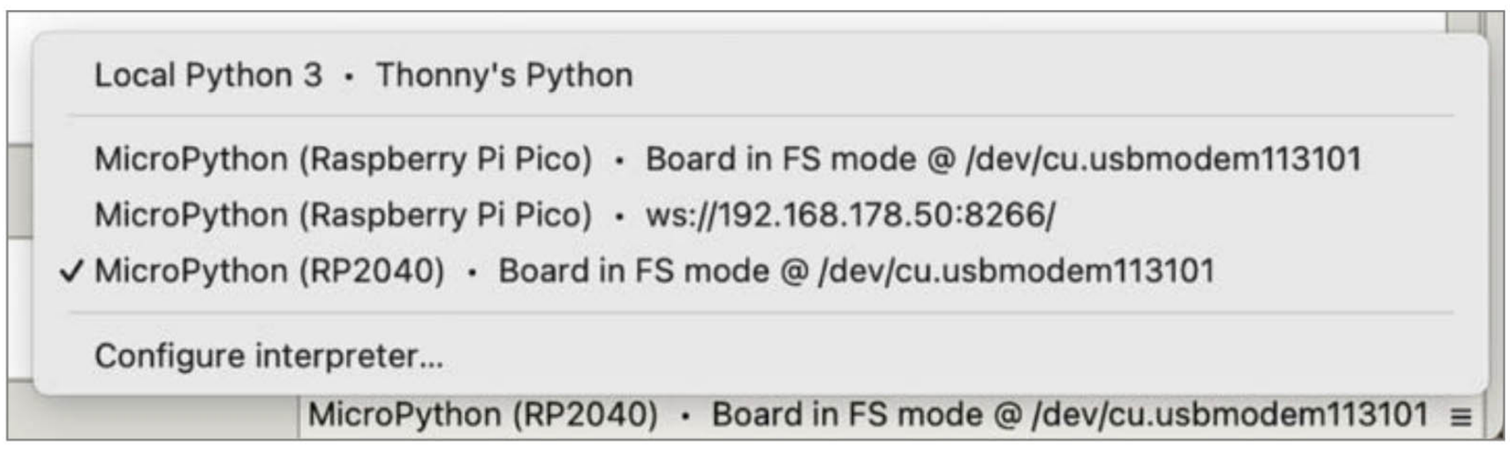


Bild 4: Rechts unten in Thonny wählt man das angeschlossene Board aus.

Erste Schritte auf der Matrix

Pimoroni liefert die Firmware seines Boards mit der gleichnamigen Bibliothek Interstate75 aus. Diese greift wiederum auf eine weitere hauseigene Bibliothek zurück, die sich PicoGraphics nennt und neben HUB75-Matrizes auch alle übrigen Displays von Pimoroni unterstützt.

In PicoGraphics findet man eine Reihe grundlegender Befehle, mit denen man Pixel, Linien, Dreiecke, Rechtecke, Kreise und Polygone zeichnen kann. Hinzu kommen Funktionen, um Texte darzustellen, und man kann sogar Bilder vom Typ PNG abbilden – dazu mehr in einem weiteren Artikel. Allerdings sind nicht alle Features mit HUB75-Panels kompatibel. So lässt sich für das Panel etwa die Helligkeit nicht einstellen, sie ist nur über die RGB-Werte der LEDs regelbar.

Um ein paar Grundfunktionen auszuprobieren, erstellt man zunächst in Thonny ein neues Dokument. Dort importiert man in der ersten Zeile die Bibliothek Interstate75 und legt fest, welche Displayauflösung man verwenden möchte (siehe Listing „erste_grafik.py“). Falls die Auflösung der eigenen LED-Matrix von den hier gezeigten 64 × 64 abweicht, muss man eines der anderen Profile verwenden, die ich in der Tabelle „Display-Profile“ aufgelistet habe.

Als Nächstes erzeugt man im Code eine Display-Instanz und nennt sie z. B. `matrix`. Außerdem benötigt man noch einen Displaypuffer (hier: `buffer`). Dieser merkt sich zunächst alles, was man gezeichnet hat und gibt das Ergebnis erst zum Schluss auf dem Panel aus – das bewirkt eine flüssigere Darstellung.

Danach fragt man die Breite und Höhe des Displays ab, um die Grenzen der LED-Matrix zu definieren. Mit `width` und `height` lassen sich Projekte später auch gut skalieren, da sich die

Display-Profile

DISPLAY_INTERSTATE75_32X32
DISPLAY_INTERSTATE75_64X32
DISPLAY_INTERSTATE75_96X32
DISPLAY_INTERSTATE75_96X48
DISPLAY_INTERSTATE75_128X32
DISPLAY_INTERSTATE75_64X64
DISPLAY_INTERSTATE75_128X64
DISPLAY_INTERSTATE75_192X64
DISPLAY_INTERSTATE75_256X64

Werte dynamisch an das verwendete Display-Profil anpassen.

Um etwas zu zeichnen, muss man zuerst (Stift-)Farben erzeugen, z. B. eine schwarze und eine weiße.

```
black = buffer.create_pen(0, 0, 0)

white = buffer.create_pen(255,
                          255,
                          255)
```

Der Code ist (wie auch später noch im Text) aus Platzgründen teilweise umgebrochen. Befehle, die von einer Klammer umschlossen sind, kann man auch in eine Zeile schreiben. Mit dem vor den Befehl `create_pen()` gehängten `buffer` wird erstmal alles in den Displaypuffer geladen. Die Werte innerhalb der Klammer sind die Helligkeit für Rot, Grün und Blau, jeweils von `0` bis `255` (8-Bit), wobei `0` die kleinste Helligkeit ist und `255` die größte.

Drei Nullen ergeben demnach Schwarz, mit dem man vor der eigentlichen Zeichnung seinen Bildschirm löschen sollte. Dafür aktiviert man zunächst mit

```
buffer.set_pen(black)
```

den schwarzen Stift und füllt den Bildschirm anschließend mit

```
buffer.clear()
```

Danach kann man mit dem weißen Stift z. B. einen (gefüllten) Kreis in die Mitte der Matrix setzen. Dafür wählt man den weißen Stift aus und setzt den Kreis mit dem Befehl `circle()`.

```
buffer.set_pen(white)
buffer.circle(int(width/2),
              int(height/2),
              10)
```

Die ersten beiden Werte in der Klammer des `circle()`-Befehls bestimmen den Mittelpunkt des Kreises auf dem Koordinatensystem. Weil er genau in der Mitte des Panels liegen soll, teile ich die Höhe und Breite der Matrix durch 2 und muss – weil MicroPython das Ergebnis von Divisionen als Gleitkommazahl (`float`) ausgibt – das Ergebnis noch mit `int()` wieder in eine ganze Zahl umwandeln, da das Koordinatensystem nur diese akzeptiert. Der dritte Wert gibt den Radius vor. Zum Schluss muss man noch die gepufferte Grafik auf der Matrix ausgeben. Das geschieht mit

```
matrix.update(buffer)
```

Danach klickt man in der oberen Symbolleiste in Thonny auf den grünen Play-Button und schon erscheint der Kreis. Mit dem roten Stop-Button lässt sich das Programm wieder unterbrechen.

Nach diesem Prinzip kann man schon eine ganze Menge auf der Matrix darstellen (Bild 5).

erste_grafik.py

```
from interstate75 import Interstate75, DISPLAY_INTERSTATE75_64X64

matrix = Interstate75(display=DISPLAY_INTERSTATE75_64X64)
buffer = matrix.display
width = matrix.width
height = matrix.height

black = buffer.create_pen(0, 0, 0)
white = buffer.create_pen(255, 255, 255)

buffer.set_pen(black)
buffer.clear()

buffer.set_pen(white)
buffer.circle(int(width/2), int(height/2), 10)
matrix.update(buffer)
```

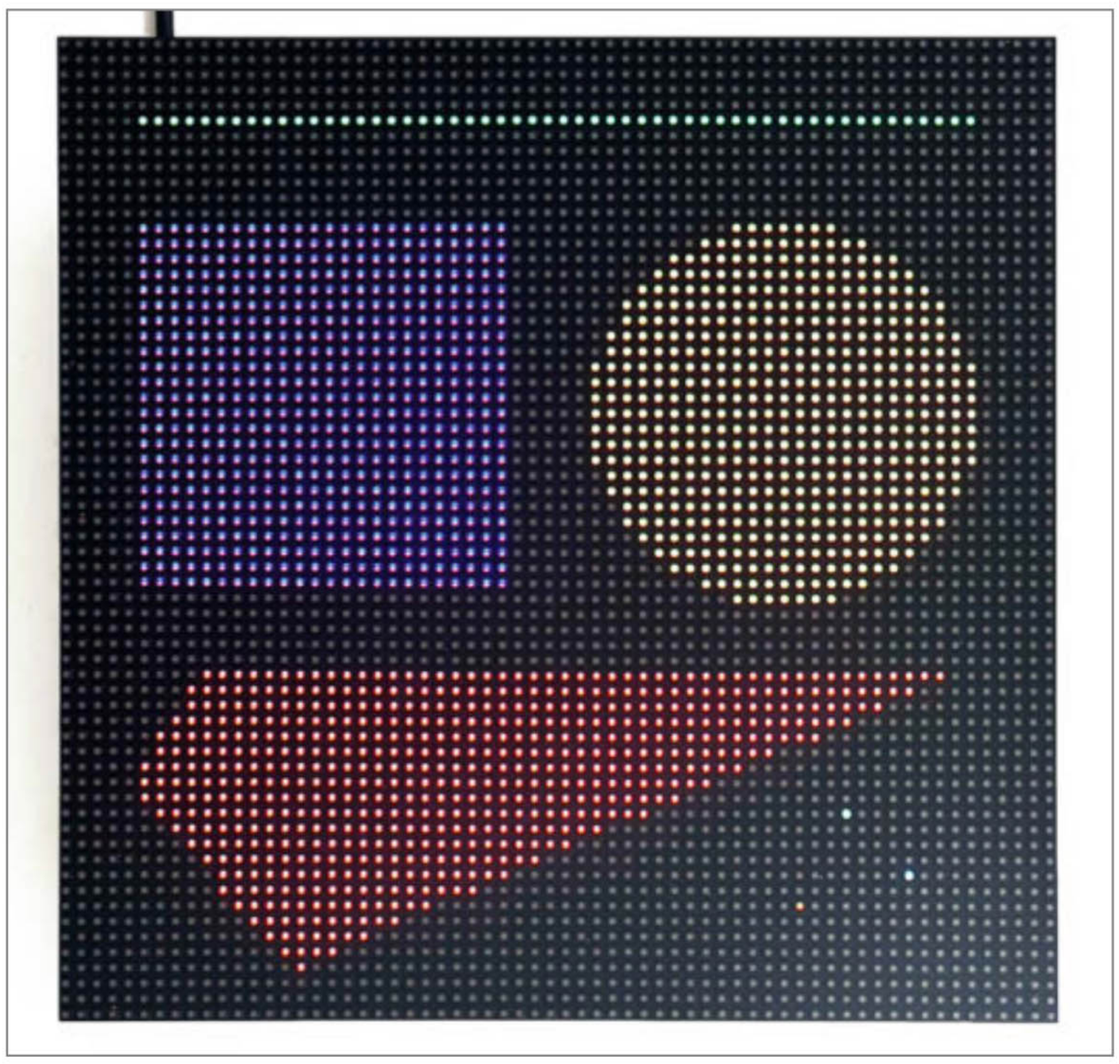


Bild 5: Die Werkzeuge von PicoGraphics: von einzelnen Pixeln bis hin zu Polygonen.

Bild 6: Mit wordwrap kann man Zeilen umbrechen.

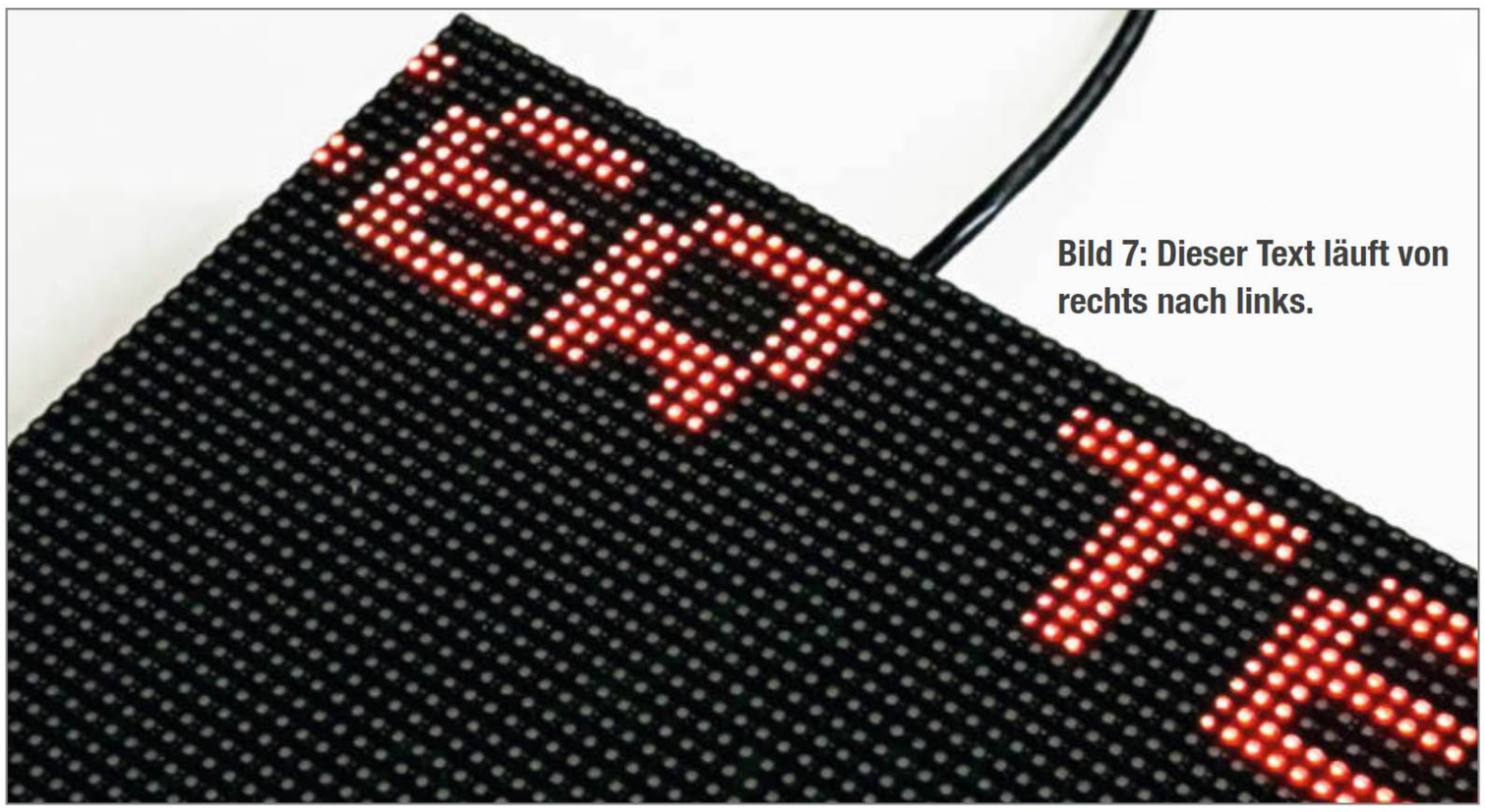


Bild 7: Dieser Text läuft von rechts nach links.

Im Folgenden sind die möglichen Grundformen noch einmal aufgeschlüsselt, wobei `t` für die Linienbreite, `b` und `h` für die Breite und Höhe eines Rechtecks und `r` für den Radius des Kreises stehen. Wer mehr Koordinaten in seinem Polygon benötigt, muss die Liste nur ergänzen.

```
pixel(x, y)
line(x, y, x2, x2, t)
triangle(x, y, x2, y2, x3, y3)
rectangle(x, y, b, h)
circle(x, y, r)
polygon([(x, y),(x2, y2),(x3, y3)])
```

Texte ausgeben

Wer Texte auf der LED-Matrix anzeigen will, kann drei Pixel-Schriftarten nutzen: `bitmap6` (Großbuchstaben), `bitmap8` (Klein- und Großbuchstaben) und `bitmap14_outline` (eine Schrift, die aus einem umlaufenden Rand besteht). Zudem gibt es noch fünf Vektor-Schriften: `sans`, `gothic`, `cursive`, `serif_italic` und `serif`, die man im Vergleich zu Pixel-Schriften frei skalieren und drehen kann.

Um eine Schriftart auszuwählen, schreibt man:

```
buffer.set_font("bitmap8")
```

Wenn man keine Schrift festlegt, verwendet PicoGraphics standardmäßig `bitmap6`. Danach kann man einen Text in den Puffer übergeben.

```
buffer.text("Cola 10 Euro!", 0, 0)
```

Die beiden Werte nach dem String sind die X- und Y-Koordinaten, die wie das Koordinatensystem links oben beginnen.

einfache_laufschrift.py

```
from interstate75 import Interstate75, DISPLAY_INTERSTATE75_64X64

matrix = Interstate75(display=DISPLAY_INTERSTATE75_64X64)
buffer = matrix.display
width = matrix.width
height = matrix.height

text_scale = 2
pos_x = width+3
pos_y = 0
text_string = "Dieser Text bewegt sich von rechts nach links."
text_width = buffer.measure_text(text_string, scale = text_scale)

black = buffer.create_pen(0, 0, 0)
red = buffer.create_pen(255, 0, 0)

while True:
  buffer.set_pen(black)
  buffer.clear()
  buffer.set_pen(red)
  buffer.text(text_string, pos_x, pos_y, scale = text_scale)

  if pos_x > -text_width:
    pos_x -= 1
  else:
    pos_x = width+3

  matrix.update(buffer)
```

Es gibt auch noch weitere Parameter, mit denen man Texte anpassen kann:

- `wordwrap` (Zeilenumbruch, `int`-Wert)
- `scale` (Skalierung, `float`-Wert)
- `angle` (Drehung, `float`-Wert, nur bei Vektor-Schriften möglich)
- `spacing` (Zeichenabstand, `int`-Wert)
- `fixed_width` (Gleiche Breite aller Buchstaben, `bool`-Wert, z.B. True)

Stehen einem wie bei meiner Matrix z.B. 64 LEDs in der Breite zur Verfügung, kann man `wordwrap` auf 64 (oder noch besser `width`) setzen, z. B. so:

```
buffer.text("Dieser Text ist viel zu
             lang für eine Zeile!",
             0, 0, wordwrap = width)
```

Danach brechen alle Wörter, die über die maximale Breite hinausgehen, um in die nächste Zeile (Bild 6). Mit `scale` kann man die Bitmap-Schriften mithilfe ganzer Zahlen vergrößern. Vektor-Schriften lassen sich auch mit Bruchzahlen skalieren – sogar verkleinern, aber hier muss man ein wenig mit der Lesbarkeit experimentieren.

Eine einfache Laufschrift

Lange Texte kann man aber auch anzeigen, ohne sie umzubrechen, indem man sie auch über die Matrix laufen lässt (siehe Listing „einfache_laufschrift.py“). Hierfür reicht es grundsätzlich, die festgelegten X- und Y-Koordinaten des Strings gegen Variablen auszutauschen, die man hoch oder – wenn der Text nach links laufen soll – runterzählt. Diese beiden Variablen nenne ich `pos_x` und `pos_y` und gebe `pos_x` zu Beginn einen Wert, der etwas größer ist als die Breite der Matrix.

```
pos_x = width+3
```

Dieser Wert rückt den Text erst mal nach rechts aus dem Bild. Damit er nach links wandert, benötigt man als Nächstes eine Dauerschleife.

```
while True:
```

In diese fügt man alle seine Textbefehle ein und setzt in `buffer.text()` die Variablen `pos_x` und `pos_y` als X- und Y-Koordinaten ein. Danach könnte man in einer weiteren Zeile `pos_x -= 1` (verkürzt für `pos_x = pos_x - 1`) schreiben, um den Text nach links laufen zu lassen. Jedoch würde er nie wieder rechts auftauchen, nachdem er sich aus dem Bild bewegt hat. Das kann man mit folgender Bedingung lösen:

```
if pos_x > -text_width:
  pos_x -= 1
else:
  pos_x = width+3
```

Jetzt scrollt der Text nach links, solange seine X-Position größer ist als die gesamte Textlänge. Danach wird `pos_x` wieder zurückgesetzt.

laufschrift_mit_timing.py (gekürzt)

```
import time
...
old_ticks = time.ticks_ms()
waiting_time = 200
...
while True:
  if ((time.ticks_ms() - old_ticks) >= waiting_time):
    old_ticks = time.ticks_ms()
    buffer.set_pen(black)
    buffer.clear()
    ...
    matrix.update(buffer)
```

Doch woher bezieht man die in Pixeln gemessene Textlänge `text_width`? Diese lässt sich mit folgendem Befehl ermitteln:

```
buffer.measure_text(text,
                    scale,
                    spacing,
                    fixed_width)
```

Für die einfache Laufschrift reichen die ersten beiden Parameter. Damit sie mit `buffer.text()` identisch sind, empfiehlt es sich, Variablen für den String und die Skalierung zu verwenden. Etwa so:

```
text_scale = 2
text_string = "Von rechts nach links"
text_width = buffer.measure_text(
                    text_string,
                    text_scale)
```

Die Variablen muss man entsprechend auch in `buffer.text()` einsetzen. Jetzt kann man das Programm mit einem Klick auf den grünen Play-Button testen (Bild 7).

Etwas langsamer, bitte

Die Laufschrift bewegt sich jetzt, aber in einer unkontrollierten Geschwindigkeit. Mit der Bibliothek `time` und der Funktion `sleep()` kann man das schnell ändern. Ganz oben im Code fügt man zunächst die Zeile

```
import time
```

ein und schreibt in die letzte Zeile der Dauerschleife

```
time.sleep(0.1)
```

Dadurch legt das Programm nach jedem Durchlauf eine kleine Zwangspause mit 0,1 Sekunden ein.

Besseres Timing

Allerdings hat dieser Quick-Fix auch Nachteile. Sobald man nämlich damit beginnt, mehrere Abläufe parallel zu koordinieren (z. B. zwei Laufschriften mit unterschiedlichen Geschwindigkeiten), treten mit `time.sleep()` Komplikationen auf, da der Befehl das gesamte Programm pausiert. Das macht sich dann z. B. in einer ruckeligen Bewegung bemerkbar.

Eine Möglichkeit, das zu lösen, besteht darin, das Programm nicht anzuhalten, sondern `pos_x -= 1` in bestimmten Zeitintervallen aufzurufen. Dafür liefert `time` die Funktion `ticks_ms()`, die fortwährend die Millisekunden seit dem Einschalten des Mikrocontrollers zählt. Man kann dem Programm also sagen:

- Überprüfe, ob 100 Millisekunden verstrichen sind.
- Setze einen neuen Timer.
- Bewege die Laufschrift.

Um diese Befehlskette als Code umzusetzen, benötigt man zwei Variablen: einen Zeitstempel (`old_ticks`) und die Wartezeit (`waiting_time`). Diese definiert man zunächst in den oberen Zeilen, wo auch die übrigen Variablen stehen und setzt `old_ticks` auf die aktuelle Zeit.

Danach erstellt man innerhalb der Dauerschleife `while True:` eine `if`-Bedingung, die den oben beschriebenen Ablauf abstrahiert (siehe Listing „laufschrift_mit_timing.py“) – und fügt alles, was man zeitlich steuern will, hinein.

Nach diesem Prinzip und indem man die `if`-Bedingungen dupliziert, kann man auch einen weiteren Text in abweichender Geschwindigkeit den Bildschirm entlanglaufen lassen. Im Projekt-Repository liegen dazu Beispiel-Codes sowie eine Erklärung, was man noch beachten muss.

Ich bin fertig, und nun?

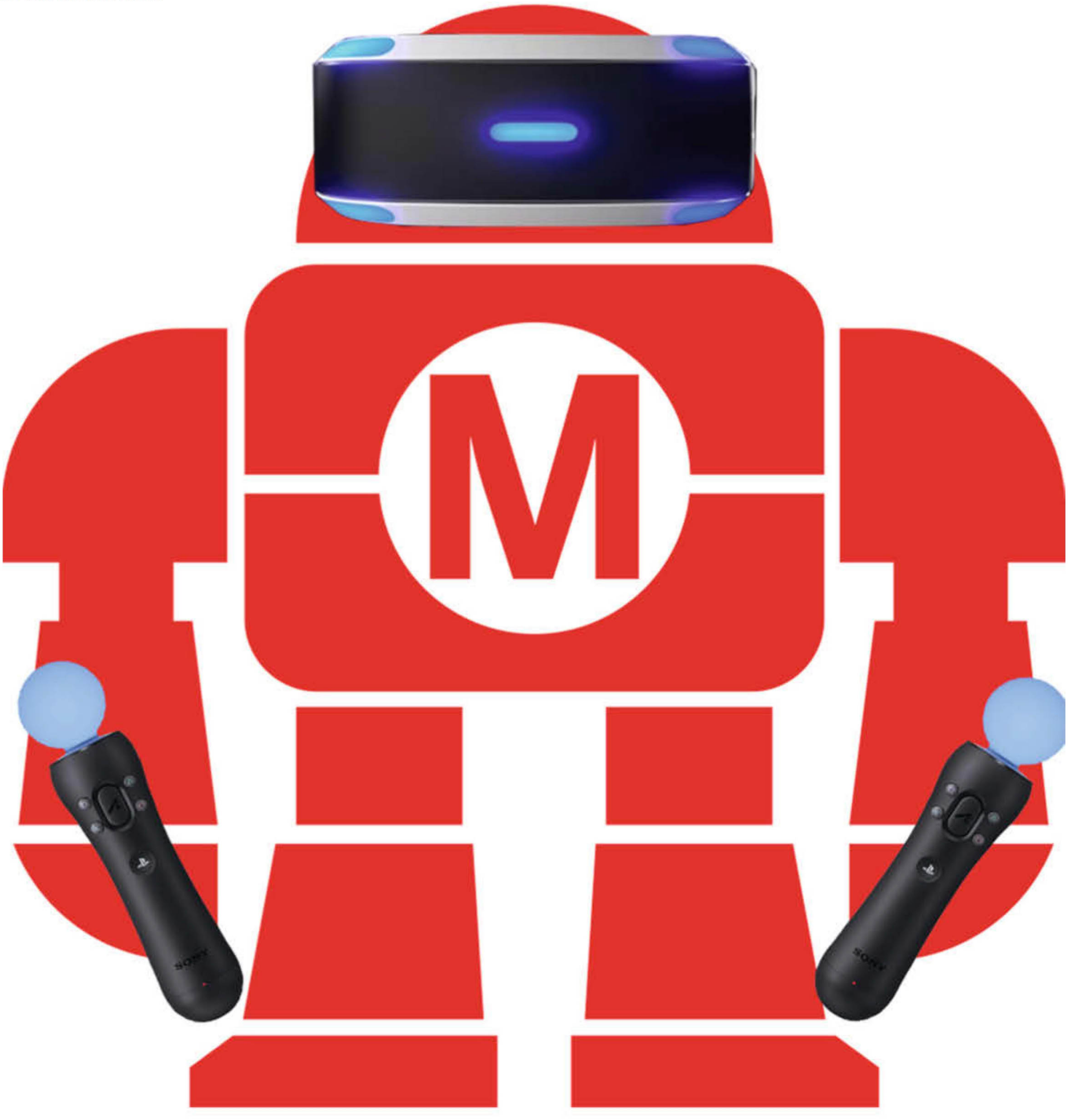
Sobald man das Programm für die LED-Matrix fertiggestellt hat, ist es an der Zeit, es auf das Board zu übertragen. Dazu klickt man in Thonny auf „File/Save as“ und wählt als Zielort das „RP2040 Device“ aus. Als Namen vergibt man main.py, denn diese Datei wird jedes Mal automatisch gestartet, sobald man das Interstate 75 in Betrieb nimmt – auch ohne Computer.

In der nächsten Make zeige ich euch, wie man PNGs und kleine Grafiken als Arrays darstellt und diese animiert. Bis dahin wünsche ich viel Spaß beim Ausprobieren! —*akf*

PSVR 1 unter Windows nutzen

In den letzten 10 Jahren sind Virtual Reality Headsets im Mainstream angekommen. Für viele war das PSVR-System der erste Berührungspunkt mit der neuen Technik. Heute liegt die Brille aber bei vielen eingepackt in einem staubigen Karton im Regal. In diesem Artikel beschreiben wir, wie man der PSVR 1 unter einem Windows PC neues Leben einhauchen kann.

von Daniel Schwabe



An ein Ökosystem gebundene Hardware ist immer suboptimal. Glücklicherweise gibt es aber Möglichkeiten, das PSVR-1-Headset von seinen durch Sony auferlegten Fesseln zu befreien.

Zum Spielen ist die in die Jahre gekommene Hardware zwar nicht unbedingt das Headset der Wahl (sonst würde es auch nicht verstaubt in einer Kiste liegen), aber durch das Sprengen der „Sonys-Ketten“ werden die Einsatzmöglichkeiten des PSVR-Systems um ein Vielfaches größer.

Kurzinfo

- Brechen der Hardwarebindung
- Einfaches Setup
- Öffnen für neue Systeme

Checkliste

Zeitaufwand:
2 Stunden

Kosten:
120 Euro

Alles zum Artikel im Web unter *make-magazin.de/xhj4*

Material

- PSVR 1 Headset
- PS3 Kamera
- iVRy Software
- Trinus PSVR Software

Mehr zum Thema

- Marcus Wengenroth, Mit den Augen des Roboters, Make 6/17, S. 26
- Guido Körber, Alte Joysticks mit USB aufrüsten, Make 6/23, S. 10

Das Headset

Es gibt mehrere Möglichkeiten, ein PSVR-1-Headset unter Windows zu nutzen. Die zwei ausgereiftesten stellen wir hier vor. Beide Programme haben Trial-Versionen, mit denen man herumprobieren kann, ob alles funktioniert und ob es sich lohnt, in eine Vollversion zu investieren.

Für beide Softwarelösungen ist zuerst das Setup der Hardware wichtig. Für die Nutzung am PC wird der PSVR genau so angebunden wie an der Playstation 4. Ein HDMI-Kabel muss von einem zusätzlichen HDMI-Port an die Breakout-Box für das Headset angeschlossen werden. Dies funktioniert auch, wenn am PC ein Displayport mit einem HDMI-Adapter verwendet wird. Wichtig ist, dass diese Breakout-Box einen eigenen Anschluss hat – es dürfen keine Splitter oder ähnliches verwendet werden. Außerdem darf am HDMI-Ausgang der Breakoutbox kein Monitor angeschlossen sein.

Dann wird die Breakout-Box noch über ein Micro-USB-Kabel mit dem USB-Anschluss des Computers verbunden. Zuletzt bekommt die Box noch Strom durch das Netzteil.

Windows sollte nun einige Hardwaregeräte erkennen und auch installieren. Das Headset selbst wird als Monitor erkannt. Auch die Soundausgabe des Headsets (der 3,5 mm Kopfhörerausgang am Headset-Riemen) funktioniert direkt.

iVRy

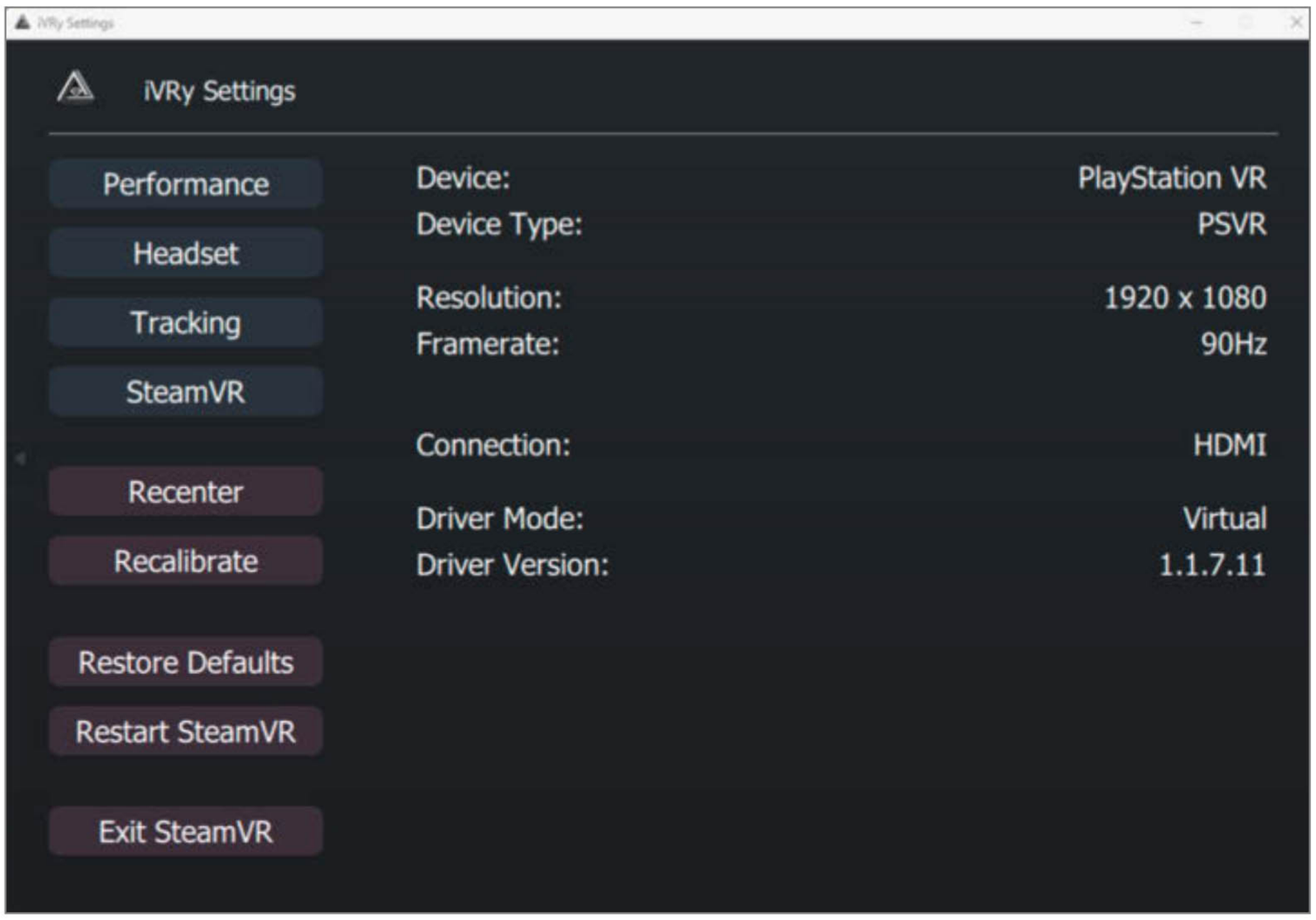
iVRy lässt sich direkt über den Steam-Store installieren. Der Link zum Shop ist in der Online-Info zu finden. Zuerst installiert man die kostenlose Basissoftware. Sobald diese heruntergeladen ist, muss man den PSVR-Zusatzinhalt aufspielen. Davon gibt es, wie gesagt, eine kostenlose Lite-Version. Ist auch dieser Inhalt eingebunden, muss iVRy einmal über die Steam-Oberfläche wie ein normales Spiel gestartet werden. Dadurch werden alle Treiber für die Hardware installiert.

Während der Installation wird man mehrfach gefragt, ob die PSVR-Treiber aufgespielt werden sollen. Bestätigen Sie diese Meldungen duch einen Klick auf „Installieren“. Danach startet auch SteamVR und nach der Kalibrierung sieht man das SteamVR-Home in der PSVR-Brille.

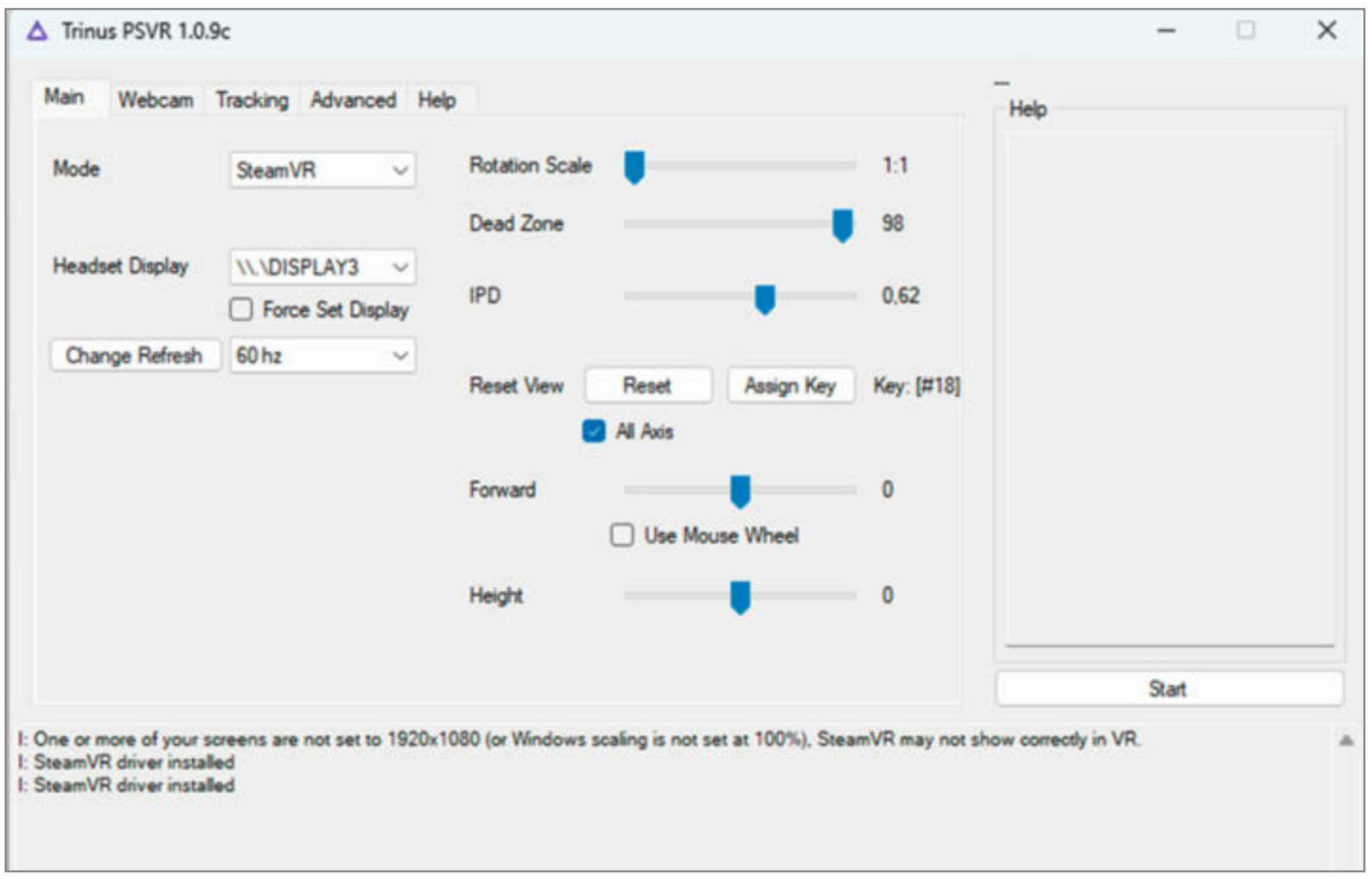
Wenn sich die virtuelle Welt jetzt unkontrolliert um einen dreht, muss man eine zweite Kalibrierung über iVRy durchführen. Dazu klickt man zunächst in der unteren rechten Ecke in der Taskleiste auf den kleinen Pfeil, wodurch weitere Status-Icons sichtbar werden. Anschließend führt man einen Doppelklick auf den iVRy-Button aus. In dem sich öffnenden Fenster ist dann „Recalibrate“ zu wählen und den Anweisungen auf dem Bildschirm zu folgen.

Hardwaredetails

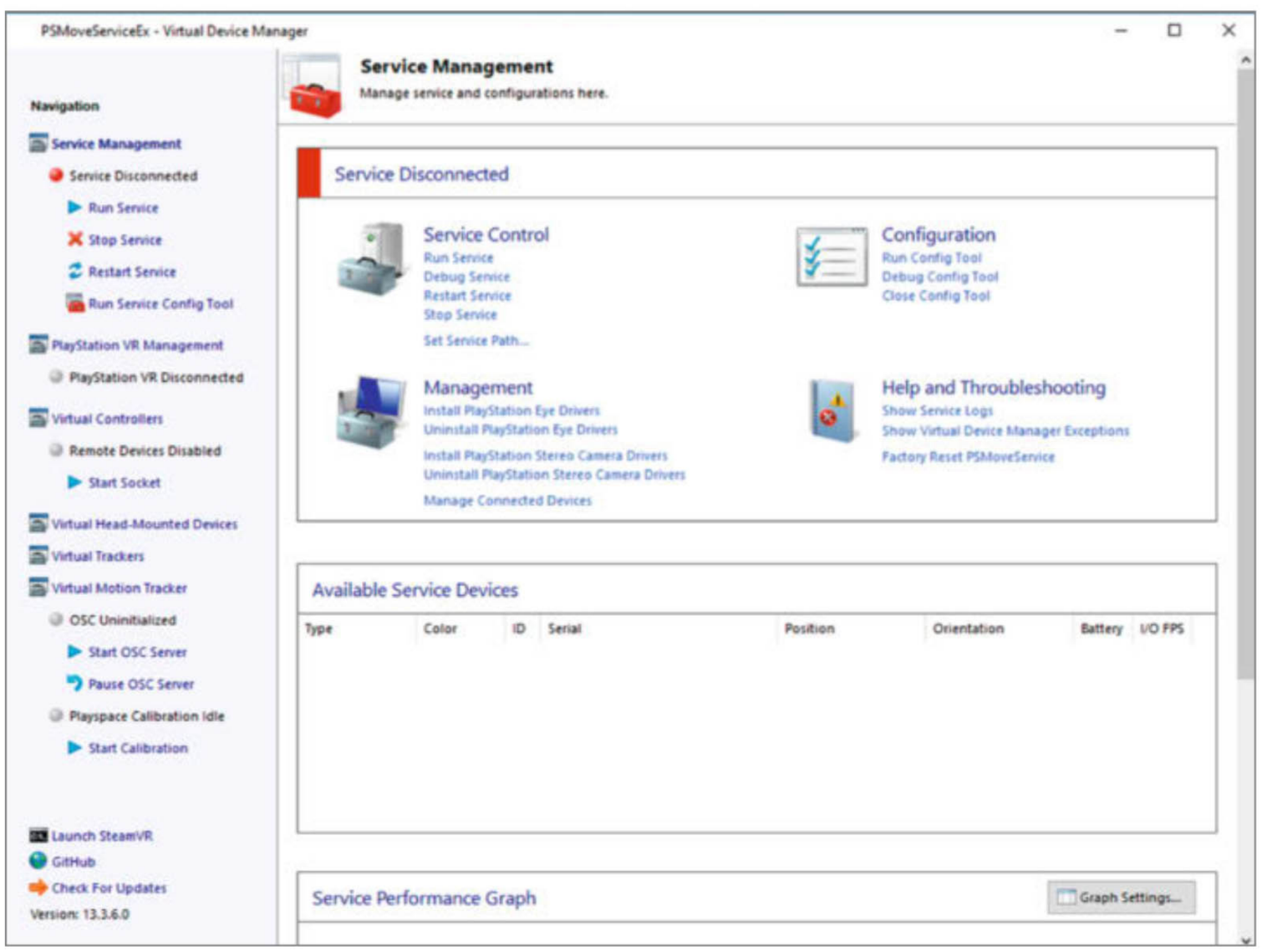
Bildschirmtechnologie	OLED
Bildschirmgröße	5,7 Zoll
Auflösung	1920 × 1080 Pixel
Bildwiederholungsrate	90/120 Hz
Sichtfeld	100°
Sensoren	Gyroskop und Beschleunigungsmesser
Anschlüsse	HDMI, USB, 3,5 mm Klinkenbuchse



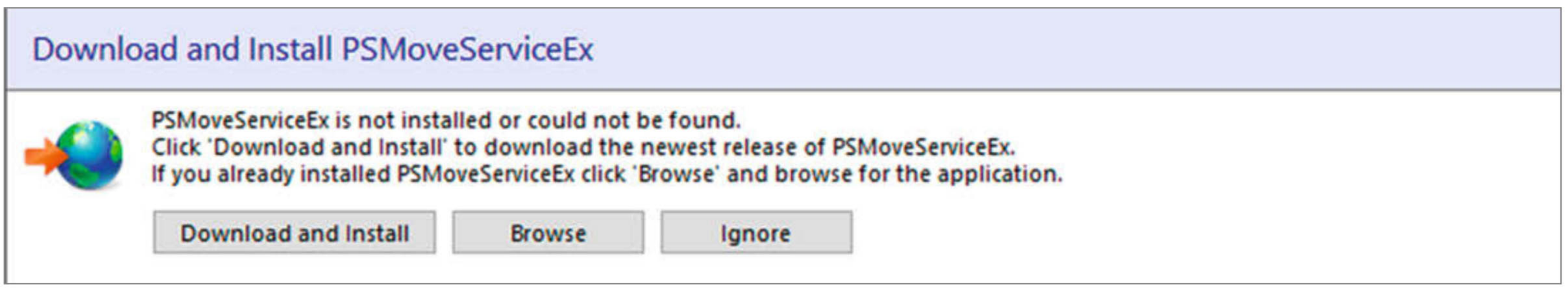
Das iVRy Control-Panel: Hier kann man den Status des VR-Gerätes ablesen.



Trinus PSVR erlaubt freiere Einstellungen. Die wichtigsten sind auf der Main-Seite.



Der PSMove Device Manager bietet leichten Zugang, um PSMove-Geräte einzustellen.



Aus dem Device Manager lässt sich der PSMoveService direkt installieren.

Die Welt bewegt sich jetzt nur noch, wenn man das auch will. Auch wenn es nur die virtuelle ist ...

Im iVRy-Fenster kann man aber noch viel mehr machen. Z. B. kann unter „Performance“ die Bildwiederholungsrate der Headset-Bildschirme von 90 auf 120 Hz gesetzt werden. Dadurch ist die Bewegung bei einem entsprechend starken Rechner wesentlich glatter und sorgt unter Umständen für weniger Schwindel oder Bauchweh.

Trinus PSVR

Die zweite Lösung für das Headset ist TrinusVR. Das ist ein Tool, welches im Gegensatz zu iVRy nicht über den Steam Store installiert, sondern extern bezogen wird.

Auch während der Installation von TrinusVR muss man eine Bestätigung für die PSVR-Treiber geben. Das geschieht ebenfalls mit einem Klick auf den Button „Installieren“ des Pop-ups.

Nachdem alles installiert und TrinusVR gestartet ist, müssen einige Einstellungen vorgenommen werden. Als Erstes wird der Mode auf SteamVR umgestellt. Die Einstellung „Headset Display“ darunter ist auf den Monitor einzustellen, unter dem das VR-Headset von Windows erkannt wird. Das ist in der Regel Monitorzahl + 1. Wenn man zwei Monitore am Computer angeschlossen hat, ist das Display Nummer drei.

Ist dies geschehen, muss das Headset einmal gerade auf den Tisch gelegt und der Startknopf rechts in der Software angeklickt werden. Danach erscheint über dem Hilfefenster eine Statusanzeige, die den Fortschritt der Kalibrierung anzeigt. Ist die Kalibrierung abgeschlossen, kann man über SteamVR starten und landet erfolgreich im Steam Home.

Eine weitere Einstellung im TrinusVR-Fenster ist die Bildwiederholung des Headsets. Die Auswahl dafür ist direkt unter dem Menü, in dem der Bildschirm eingestellt wurde. Sollte diese Funktion ausgegraut sein, muss über die Grafikkartensoftware eine zusätzliche Auflösungskonfiguration für den PSVR-Monitor angelegt werden. Wie das für die verschiedenen Hersteller funktioniert, ist in der Online-Info verlinkt.

Einschränkungen

Mit diesen beiden Optionen funktioniert das PSVR-Headset nun mit SteamVR. Auch Kopfbewegungen werden problemlos getrackt. Allerdings nur, wenn man stationär spielt – also z.B. im Stehen oder im Sitzen. Bewegt man sich im 3D-Raum, bleibt der virtuelle Kopf an Ort und Stelle und wird nicht getrackt. Das ist schade, aber die meisten VR-Programme funktionieren trotzdem problemlos.

TrinusVR bietet zwar eine Alpha-Funktion, um die LEDs des VR-Headsets über eine Kamera zu tracken, aber leider funktioniert das nicht zuverlässig und führt durch Bugs und Glitches zu Übelkeit.

Die Controller

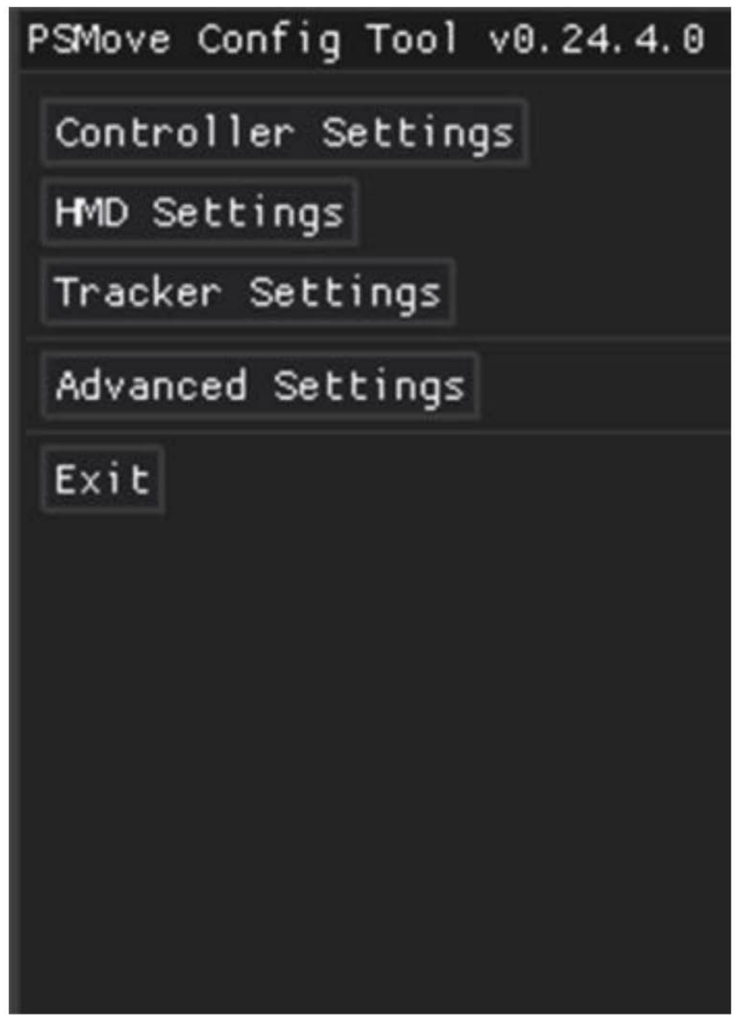
Die PSVR-Controller können auch am PC genutzt werden. Dafür braucht man die Programme PSMoveServiceEx und den dazugehörigen Virtual Device Manager. Der Link dazu findet sich in der Online-Info.

Den Playstation Moce Controller gibt es in zwei Versionen. Die ältere für die Playstation 3 und die neuere für die Playstation 4. Äußerlich unterscheiden sie sich durch den USB-Port. Die älteren haben einen Mini-, die neueren einen Micro-USB. Für VR am PC eignen sich die PS3-Versionen besser, weil sie neben anderen Sensoren auch noch ein Magnetometer eingebaut haben.

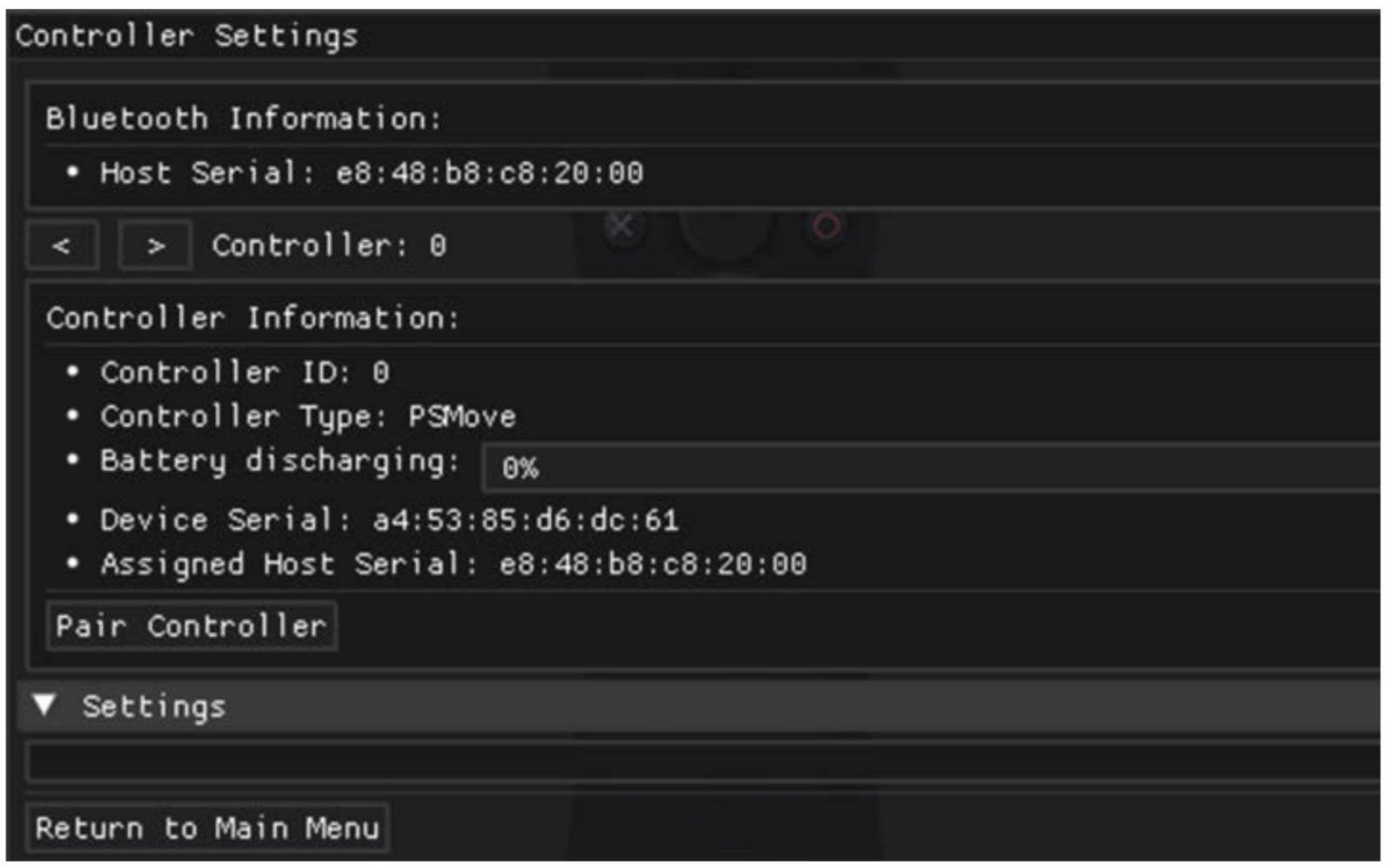
Um mit diesem Programm PSmove-Controller mit dem Computer zu verbinden, braucht man noch einen Bluetooth-Adapter und eine PS3-Eye-Kamera.

Zuerst muss Bluetooth aktiviert und die Kamera per USB angeschlossen werden. Dann stellt man die Kamera z. B. auf den Computerbildschirm oder an eine andere Stelle, von der aus sie das Spielfleld überblickt.

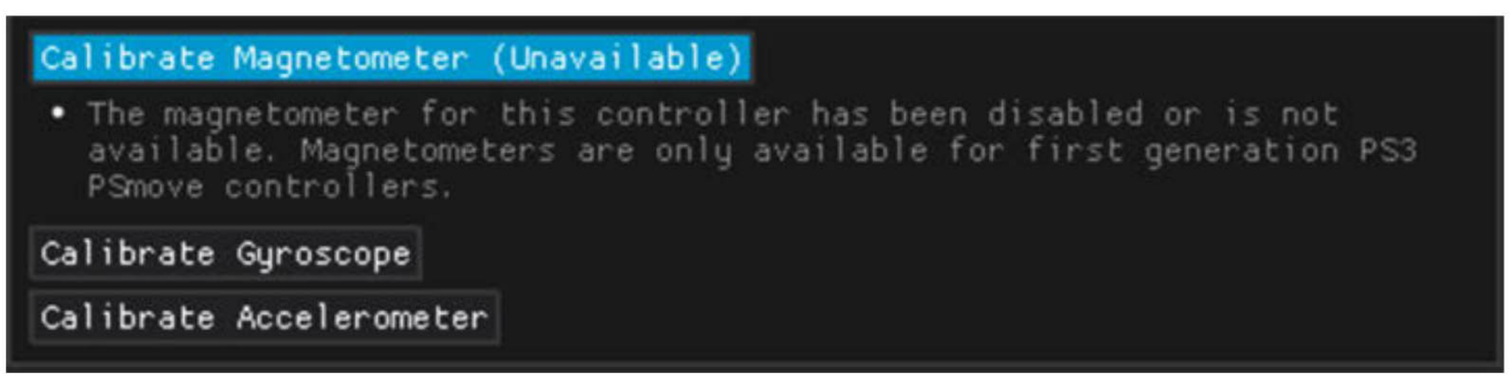
Nachdem die Hardware so eingerichtet ist, laden Sie die beiden Programme herunter und entpacken sie diese. Danach startet man den Virtual Device Manager mit der Anwendung PSMSVirtualDeviceManager.exe.



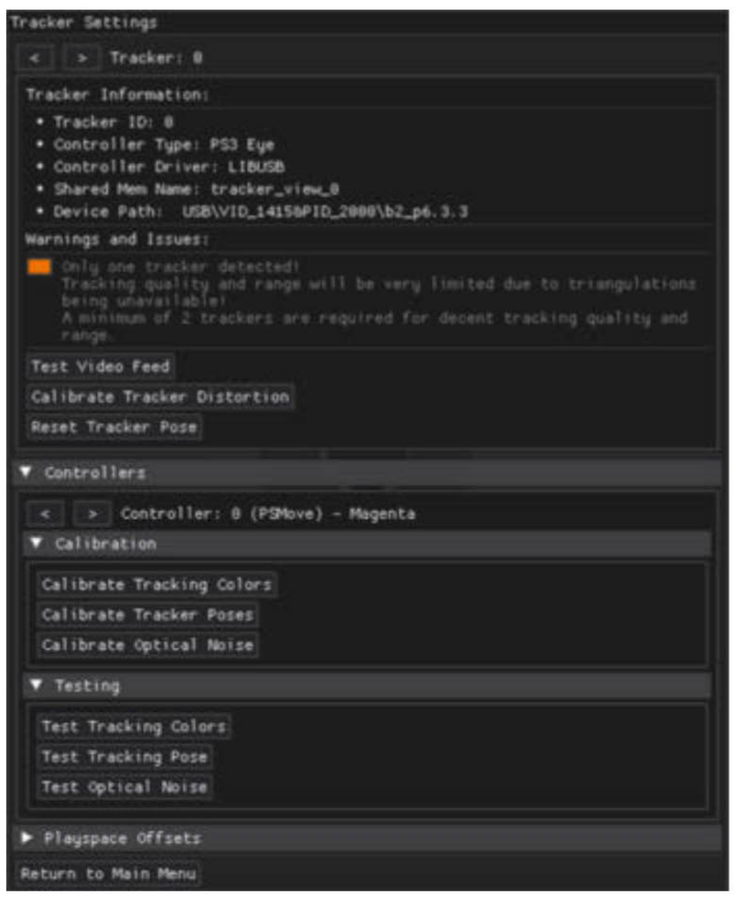
Im PSMove config Tool kann man alle Geräte einzeln einstellen.



In den Controller Settings werden die Gamepads verbunden.



Je nach Ausführung verfügen die PSMove-Controller über unterschiedliche Sensoren.



Die Tracker-Settings sorgen dafür, dass die Kamera die bunte Kugel am Controller-Ende erkennt.

Die Kalibrierung kann automatisch durchgeführt werden.

Im sich öffnenden Fenster erscheint zunächst die Meldung, dass man den PSMoveServiceEx installieren soll. Wenn man diesen bereits heruntergeladen und entpackt hat, kann man hier auf Browse klicken und zum entpackten Ordner navigieren, dort die PSMoveService.exe auswählen und bestätigen. Ansonsten einfach auf den Button „Download and Install“ klicken.

Als nächstes muss der Treiber für die PS3-Kamera installiert werden. Dafür klicken Sie im Bereich Management auf den Text „Install PlayStation Eye Drivers“ und folgen den Bildschirmanweisungen.

Als letztes verbinden Sie noch die PlayStation Move Controller. Dazu muss zunächst ein Controller per USB an den Computer angeschlossen werden. Danach sind im Bereich „Service Control“ auf „Run Service“ zu klicken und unter Configuration „Run Config Tool“ auszuwählen.

Im Menü „Controller Settings“ sollten jetzt einige Informationen zum angeschlossenen Controller angezeigt werden. Klicken Sie unten im Fenster auf „Pair Controller“.

Danach öffnet sich ein Fenster mit einem Ladebalken. Nun muss der Controller vom USB-Kabel getrennt und der Playstation-Knopf am Move Controller mehrmals gedrückt werden, bis die Verbindung hergestellt ist. Auf dem Bildschirm erscheint die Meldung. Unter „Calibration“ müssen alle drei Schritte durchgeführt werden. Wie das funktioniert, wird ausführlich auf dem Bildschirm beschrieben.

Für den verbundenen Controller erscheinen jetzt neue Optionen.

Führen Sie jetzt den gleichen Vorgang mit dem zweiten Controller durch. Achtung – eventuell muss vor dem Pairing in den Controller Settings eine andere Controller-ID ausgewählt werden. Das geschieht mit den beiden Pfeiltasten neben „Controller: 0“. Controller 0 ist der erste, Controller 1 der zweite zu verbindende.

Nachdem beide Controller verbunden sind, kallibrieren Sie noch die PS3-Kamera. Dafür klickt man im ersten Fenster des Config Tools auf „Tracker Settings“.

In diesem Fenster klicken Sie unter „Calibrate Tracking Colors“ anklicken, den ersten Controller (also Nummer 0) vor die Kamera halten und auf Start klicken. Der Controller darf dabei nicht bewegt werden. Wenn die Kalibrierung abgeschlossen ist, wählen Sie mit den Pfeiltasten den zweiten Controller (Nummer 1) aus und wiederholen den Vorgang.

Nachdem alles gespeichert ist, muss nur noch das kostenlose iVRy PSmove-Addon aus dem Store installiert werden. Danach erkennt SteamVR die Controller und sie können im Spiel benutzt werden.

Leider funktionieren die PSmove-Controller im Gegensatz zum Headset nicht so fehlerfrei, wie man es gerne hätte. Der Griff zu externen Lösungen ist hier wahrscheinlich besser.

Wie geht es weiter?

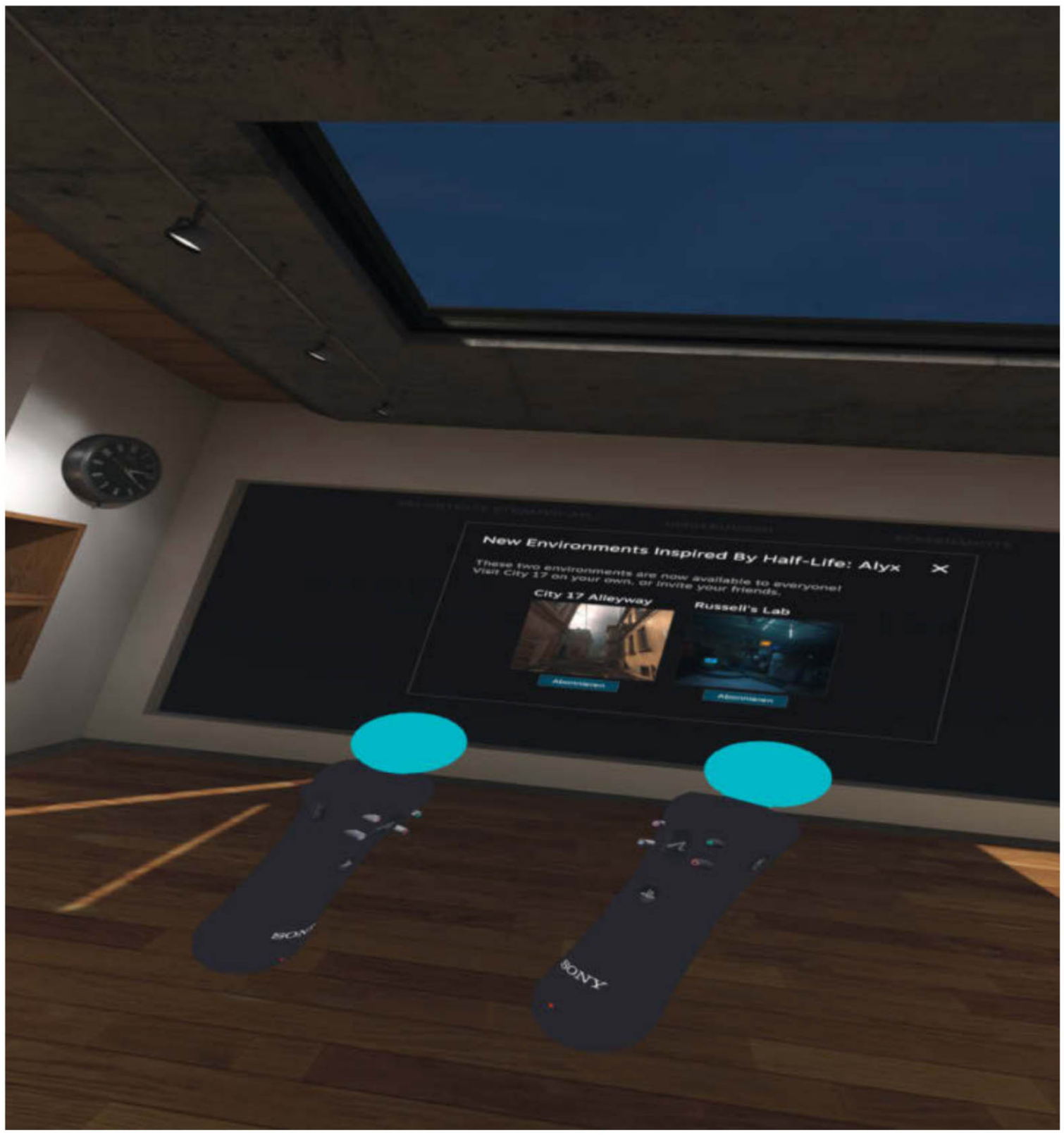
Jetzt kann man das PSVR-Headset im SteamVR-Ökosystem verwenden. Mit Tools wie Revive (Link in der Online-Info) lassen sich auch Programme aus dem geschlossenen Oculus-Store auf anderen Headsets nutzen. Damit steht noch mehr Software zur Verfügung.

Fazit des Autors

Wer kein VR-System besitzt, sollte nicht mit der Möglichkeit, die Ketten eines PSVR-Systems zu sprengen, in das Thema einsteigen. Mittlerweile gibt es auch günstige Möglichkeiten, mit gebrauchten Oculus-Ques-Brillen VR zu erleben.

Allerdings bin ich persönlich Fan davon, obsolete – bzw. in einem veralteten Ökosystem gefangene – Technik noch weiter zu verwenden statt sie wegzuwerfen und zu hoffen, dass der Schrott nicht irgendwann im Magen eines Robbenbabys landet.

Ich besitze noch ein zweites VR-Headset für den PC, nutze aber meine PSVR-Brille gerne zum Arbeiten in einer virtuellen Desktop-Umgebung wie Virtual Desktop Classic. Das Headset ist leichter als mein anderes und benötigt nur eine kurze Kalibrierung, bevor ich es aufsetze. *—das*



Mit leichter Redakteur-Schlagseite befindet man sich jetzt mit PSVR im virtuellen Raum.

Volumio mit Drehgebern erweitern

Die Musik-Distribution Volumio lässt sich leicht über diverse Schnittstellen bedienen – auch stilecht mit Drehknöpfen. Wir zeigen in diesem Workshop, wie man dafür KY-040-Drehgeber in das System einbindet und wie man sie für ein reibungsloses Bedienerlebnis entprellt.

von Andreas Voß

Auch ich gehöre zu den Leuten, die sich einen Raspi-Netzwerkplayer mit Volumio gebaut haben, und bin sehr begeistert davon. Wie das geht, beschreiben die verlinkten Artikel in der Kurzinfo. Wichtig war mir bei meiner Umsetzung, dass ich die Lautstärke und einige andere Funktionen über Drehgeber (engl. rotary encoder) einstellen kann, da ich mit ihnen schneller bin als auf einem Touchscreen am Player oder Smartphone. Auch lässt sich die Wiedergabe unmittelbar stummschalten oder die Stummschaltung wieder aufheben, wenn man einen Drehgeber drückt.

In diesem Artikel erkläre ich am Beispiel eines KY-040, wie ein Drehgeber funktioniert, wie man ihn hardwareseitig entprellt und mithilfe eines Plug-ins in das Musikplayer-Betriebssystem Volumio einbindet. Die Hardware-Lösung lässt sich aber auch für andere Projekte mit Drehgebern verwenden.

Der Artikel setzt voraus, dass Sie bereits einen Raspberry Pi 4 besitzen und wissen, wie man Volumio darauf installiert. Falls Sie meinen Schaltplan verwenden möchten, benötigen Sie außerdem das „Argon Nanosound ONE"-Gehäuse oder müssen die GPIO-Verbindungen entsprechend an Ihren Aufbau anpassen.

Kurzinfo

- **Funktionsweise des Drehgebers KY-040 verstehen**
- **Drehgeber hardwareseitig entprellen**
- **Volumio mit dem Plug-in Rotary Encoder II steuern**

Checkliste

Zeitaufwand:
2 bis 3 Stunden

Kosten:
ca. 30 Euro
optional zzgl. 80 Euro für ein Gehäuse, das in diesem Projekt verwendet wurde

Material

- **3 Drehgeber KY-040** mit Gewinde, Mutter und Widerständen
- **Lochrasterplatine** mit Lötaugen, mindestens Europakarten-Größe
- **2 14-polige IC-Sockel** mit Abblockkondensator, z.B. GS-KO 14P
- **2 ICs** mit invertiertem Schmitt-Trigger, SN74HC14
- **9 Kondensatoren** mit 100 nF, 100 V, RM 7,5, Breite 2,5
- **9 Widerstände** mit 10 kΩ
- **Wannenstecker** 40-polig, gerade, z. B. HAN 540 6324
- **Raspberry Pi GPIO-Kabel**, 40-Pin, 20 cm lang
- **Stiftleiste**, 50-polig, gerade, RM 2,54, z. B. SL 1X50G 2,54
- **Steckbrückenkabel-Set**, 40 cm lang, 3 × 20 Kabel, z. B. DEBO KABELSET23

Optional

- **„Argon Nanosound ONE"-Gehäuse** mit integriertem DAC

Werkzeug

- **Lötkolben und -zinn**
- **Seitenschneider**

Software

- Volumio

Mehr zum Thema

- Dieter Schaurich, Schicker Streamer, Make 4/23, S. 48
- Roman Radtke und Ákos Fodor, HiRes-Audio mit dem Raspberry Pi, Make 6/21, S. 10
- Wolfgang Ziegler, PC-Lautstärkesteller selbst gemacht, Make 2/19, S. 50
- Heinz Behling, Raspberry: GPIO-Pins beim Booten initialisieren Make, Online-Artikel

Alles zum Artikel im Web unter
make-magazin.de/xqz5

Drehgeber vs. Potentiometer

Auf den ersten Blick sieht der KY-040 (Bild 1) fast so aus wie ein Potentiometer. Er ist aber kein variabler Widerstand, sondern erzeugt Impulse, mit deren Hilfe man die Drehrichtung und den gedrehten Winkel bestimmen kann. Da der Drehgeber keine Endpunkte hat, lässt er sich zudem endlos in jede Richtung drehen. Außerdem besitzt er einen integrierten Taster. Dank der fehlenden Kohleschicht, die sich bei Potis gern über die Zeit abnutzt – man kennt das von kratzenden Lautsprecherreglern – ist der KY-040 ideal für den langjährigen Betrieb.

Allerdings kann er sich seine Position nicht selbst „merken", d.h. während der Widerstand eines Poti selbst nach einem Stromverlust unverändert bleibt, ist für den KY-040 prinzipiell jede Position im Stillstand identisch. Erst wenn man ihn dreht, entstehen die Impulse, mit denen man per Software Werte hoch und runter zählen kann (z. B. die aktuelle Lautstärke). Aufgrund der Arbeitsweise nennt man diese Art von Drehgebern daher auch Inkrementalgeber.

Drehungen erfassen

Drehgeber arbeiten mit einem Schaltersystem, das zwei versetzte Signale (hier an den Kontakten CLK und DT) ausgibt, deren Pegel zwischen High und Low wechselt, sobald man den Knopf weit genug dreht. Dieser Wechsel geschieht entweder über magnetische oder optische Trigger – oder wie beim KY-040 über einfache Metallkontakte (Bild 2). Sie verbinden CLK und DT abwechselnd mit GND, sodass mal ein Stromkreis geschlossen wird und dann wieder nicht. Das geschieht um 90 Grad phasenverschoben, d.h. manchmal sind die Werte an den Ausgängen identisch und dann wieder unterschiedlich.

Um nun herauszufinden, in welche Richtung man gedreht hat, kann man auf einen Pegelwechsel von CLK warten und diesen dann mit dem aktuellen Pegel von DT vergleichen. Wenn CLK auf High wechselt und DT zu diesem Zeitpunkt Low ist (oder CLK auf Low wechselt und DT High ist) – also beide unterschiedlich sind –, hat man im Uhrzeigersinn gedreht (Bild 3). Bei einer Bewegung in die andere Richtung sind die Paare immer identisch: CLK wird Low, DT ist bereits Low; CLK wird High, DT ist High (Bild 4). Mit dieser Logik lässt sich dann in einem Programm auch mitzählen, wie weit man gedreht hat und ausrechnen, wie schnell das passiert ist.

Den KY-040 entprellen

Man sollte den KY-040 nicht direkt an die GPIOs des Raspberry Pi anschließen, da die Ausgänge SW, CLK und DT aufgrund der einfachen Metallkontakte prellen. Der Pi erhält

Bild 1: Der Drehgeber KY-040

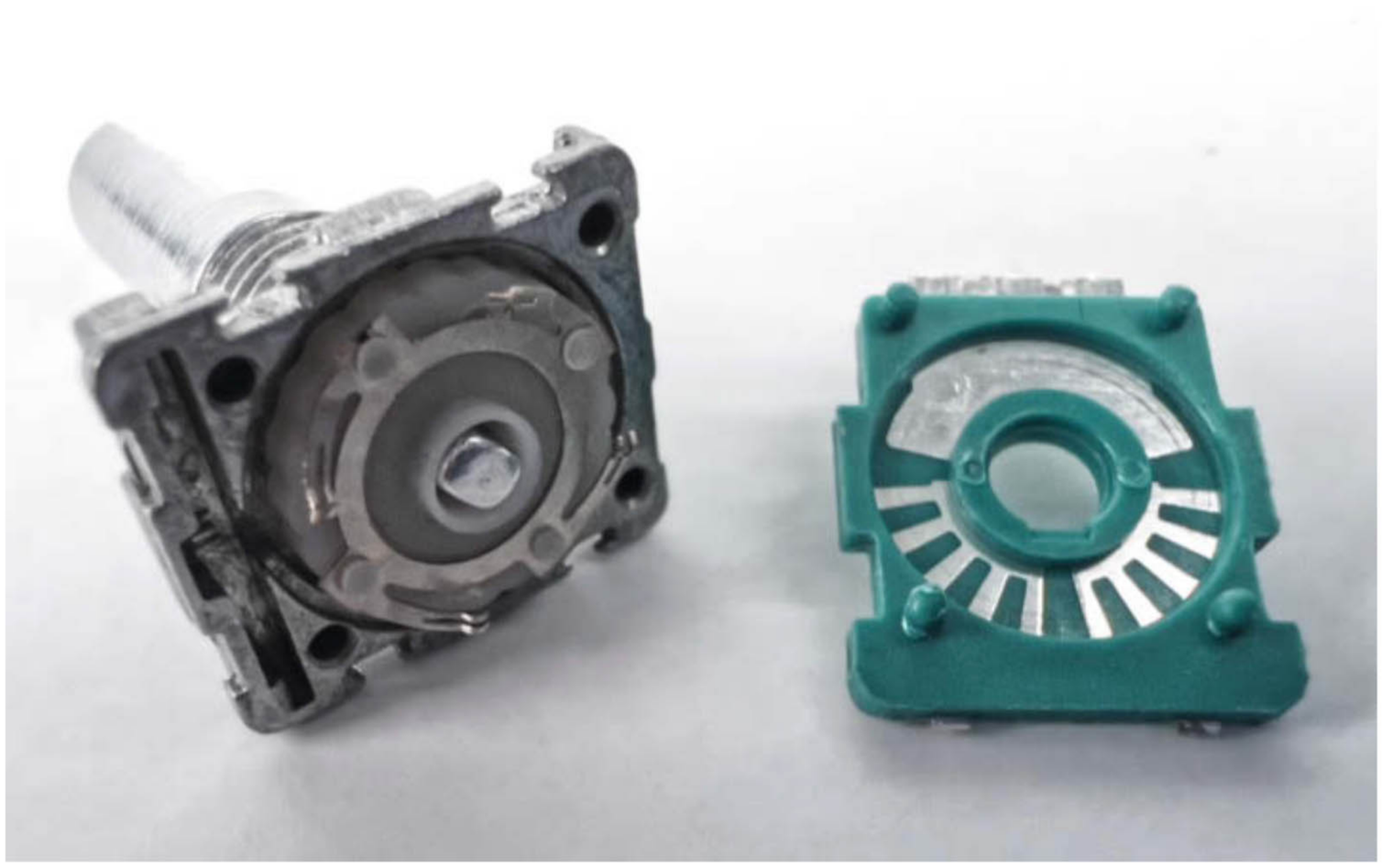


Bild 2: Im Inneren des KY-040 erzeugen einfache Metallkontakte die Impulse.

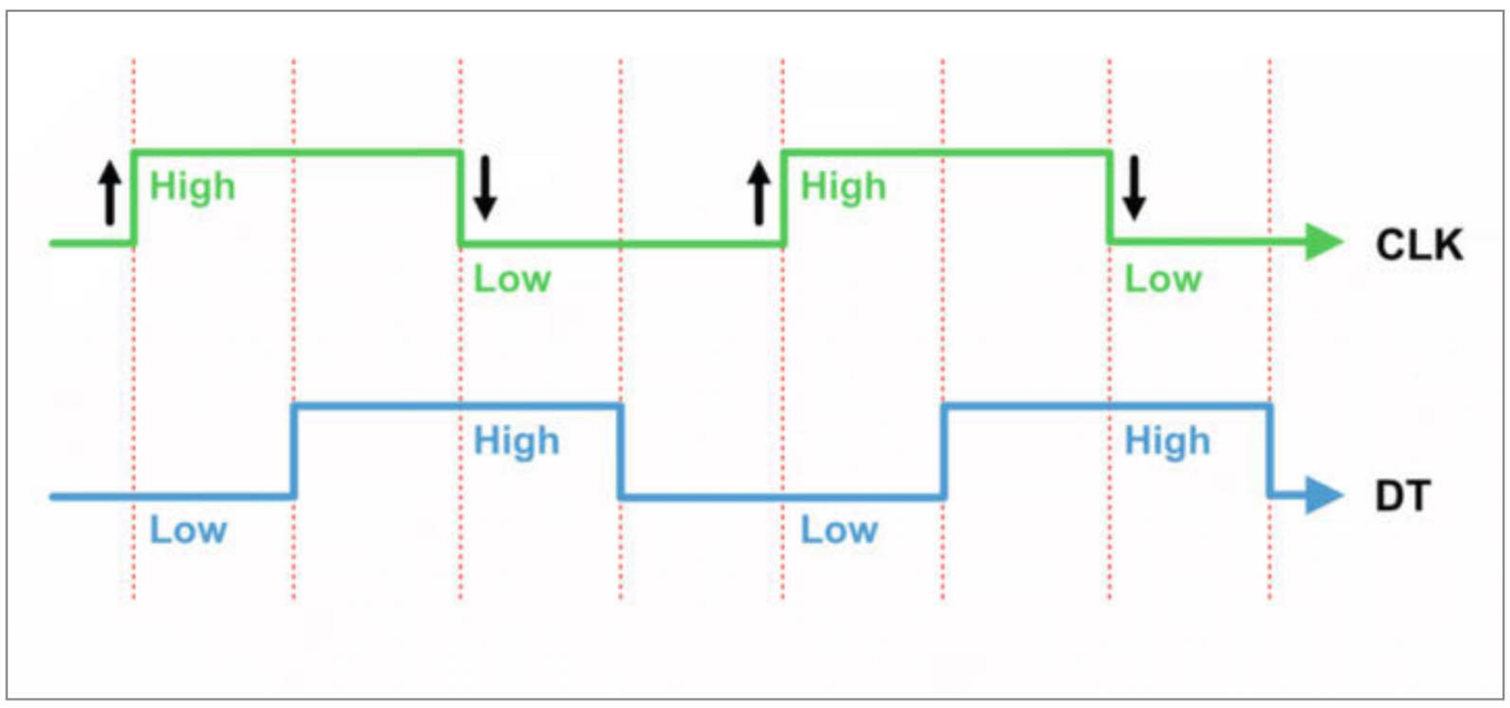


Bild 3: Wenn man im Uhrzeigersinn dreht, ist der Pegel von CLK bei einem Wechsel immer gegenteilig zu DT.

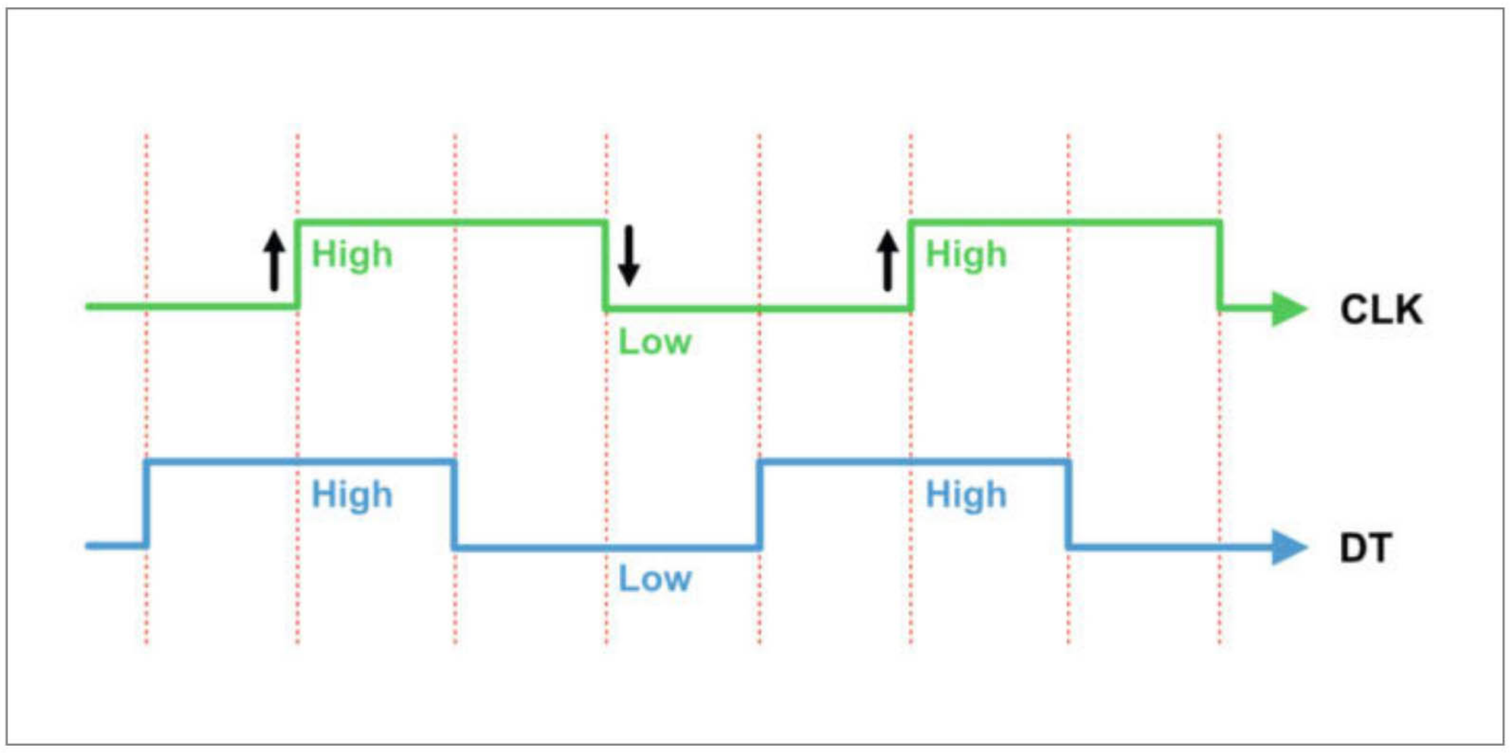


Bild 4: Dreht man in die entgegengesetzte Richtung, sind beide Signale kurzzeitig identisch, sobald CLK seinen Pegel wechselt.

bei einem Pegelwechsel dann mehrere ungewollte Impulse – was zu Fehlfunktionen und einer unzuverlässigen Steuerung führt. Betrachtet man beispielsweise das Ausgangssignal des Tasters (SW) im KY-040, sieht es beim Drücken und Loslassen ungefähr so aus wie in Bild 5 – man kann die zusätzlichen Impulse (Preller) deutlich erkennen.

Dieses Problem lässt sich software- und hardwareseitig beheben. Das von mir verwendete Volumio-Plug-in bietet allerdings nur eine Software-Entprellung für den Taster an. Zudem hat eine Hardwarelösung den Vorteil, dass sie unabhängig von der Software (oder Updates) funktioniert und man sie universell auch in anderen Projekten verwenden kann. Es schadet also nicht, wenn man weiß, wie so etwas geht.

Um die Preller im Drehgeber zu beseitigen, habe ich im ersten Schritt einen zusätzlichen Widerstand und einen Kondensator (RC-Glied) eingesetzt (Bild 6). Danach sind die ungewollten Impulse verschwunden. Bedauerlicherweise erhält man aber kein sauberes Rechtecksignal, sondern eine Entlade- und Ladekurve des Kondensators. Auch das kann wieder zu Problemen führen, wenn man den Drehgeber schnell dreht und die Signale entsprechend schnell nacheinander beim Raspberry Pi ankommen.

Um die Kurven in ein sauberes, rechteckiges Signal umzuwandeln, habe ich daher noch einen zusätzlichen invertierten Schmitt-Trigger eingebaut (Bild 7). Danach ist das Signal so aufbereitet, dass SW dem Raspberry Pi über einen GPIO-Pin ein eindeutiges High- oder Low-Pegel liefert. Auch die Anschlüsse CLK und DT benötigen jeweils eine solche Schaltung. Praktischerweise befinden sich in einem 74HC14-Chip sechs invertierte Schmitt-Trigger, sodass zwei der Chips für drei Drehgeber ausreichen.

Platine für die Drehgeber

Für meinen Netzwerkplayer verwende ich drei Drehgeber, die von den 3,3 V des Raspberry Pi mit Spannung versorgt werden. Um die KY-040 zu entprellen und vernünftig im Gehäuse zu verbauen, habe ich nach dem oben erwähnten Prinzip eine Schaltung entwickelt (Bild 8) und auf einer Lochrasterplatine mit Lötaugen aufgebaut (Bild 9). Bitte beachten Sie, dass die Platine von der Bestückungsseite aus gezeichnet ist. Als Drehgeber empfehle ich die KY-040-Variante mit Gewinde und Mutter sowie drei aufgelöteten Pull-up-Widerständen (siehe Kasten).

Mein Raspberry Pi 4 steckt in einem „Argon Nanosound ONE“-Gehäuse. Es ist aus Aluminium gefertigt, das der Kühlung dient und hat einen eingebauten HiFi-DAC (Digital-Analog-Wandler) für die Audioausgabe. Sollten Sie eine andere Soundkarte benutzen, müssen Sie schauen, welche GPIOs diese für die Kommu-

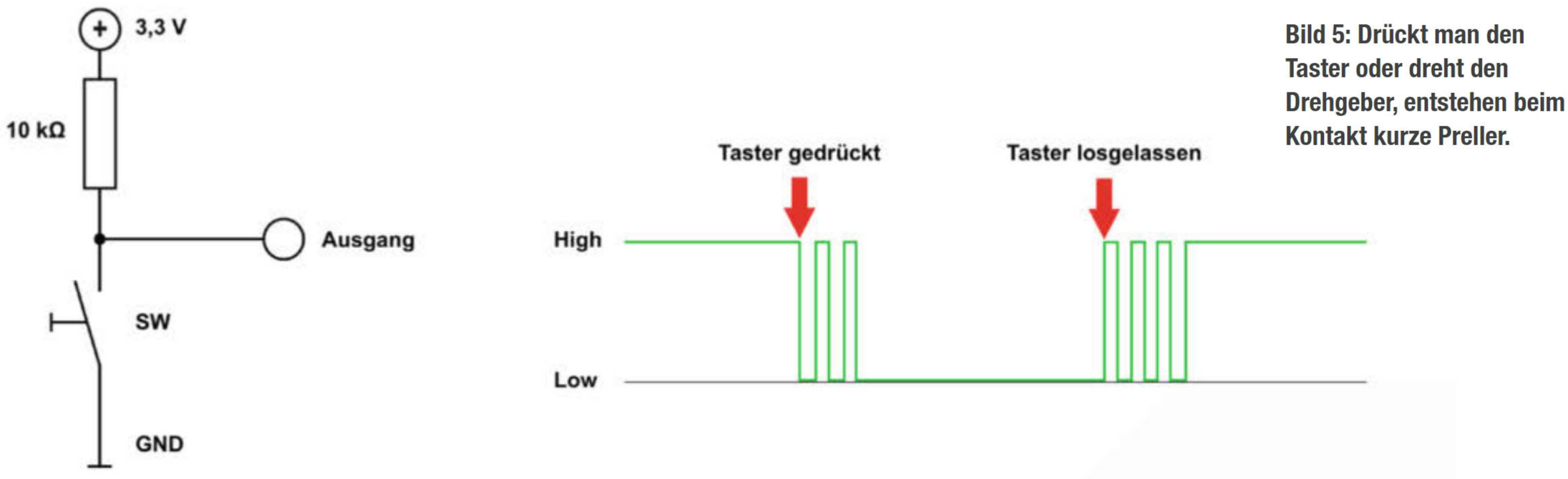


Bild 5: Drückt man den Taster oder dreht den Drehgeber, entstehen beim Kontakt kurze Preller.

Hinweise für den Kauf eines KY-040

Es gibt die kleinen Drehgeber mit Gewinde und ohne. Wenn man sie samt ihrer Platine an der Frontplatte eines Gehäuses anschrauben will, sollte man welche mit Gewinde nehmen. Auch fehlen bei einigen Modellen Pull-up-Widerstände auf den Platinen (Bild 10). Diese muss man dann entweder nachlöten oder die GPIO-Eingänge des Raspberry Pi beim Booten initialisieren und die internen Pull-up-Widerstände einschalten (siehe Link in der Kurzinfo). Einfacher ist es, wenn man gleich Drehgeber mit Gewinde, Muttern und aufgelöteten Pull-up-Widerständen nimmt. Passende Drehknöpfe kann man nach eigenen Vorstellungen dazu kaufen. Da meine Frontplatte aus gebürstetem Aluminium besteht, habe ich mich für gedrehte Aluminium-Knöpfe entschieden, die allerdings nicht ganz billig sind.

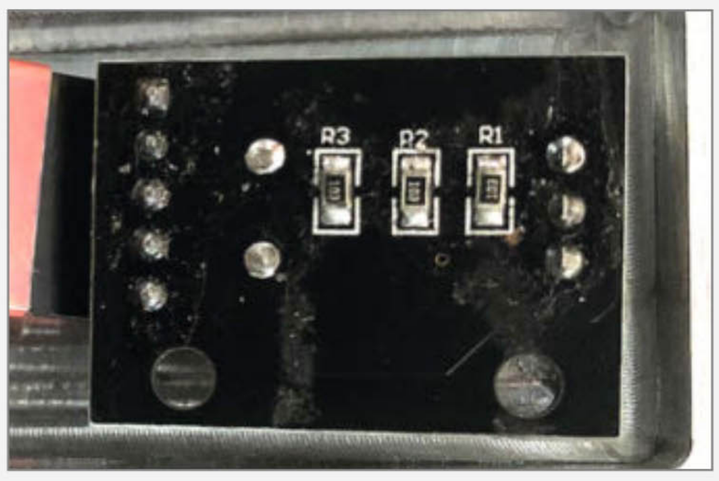


Bild 10: Dieses KY-040-Modell hat je einen Widerstand für CLK, DT und SW verbaut.

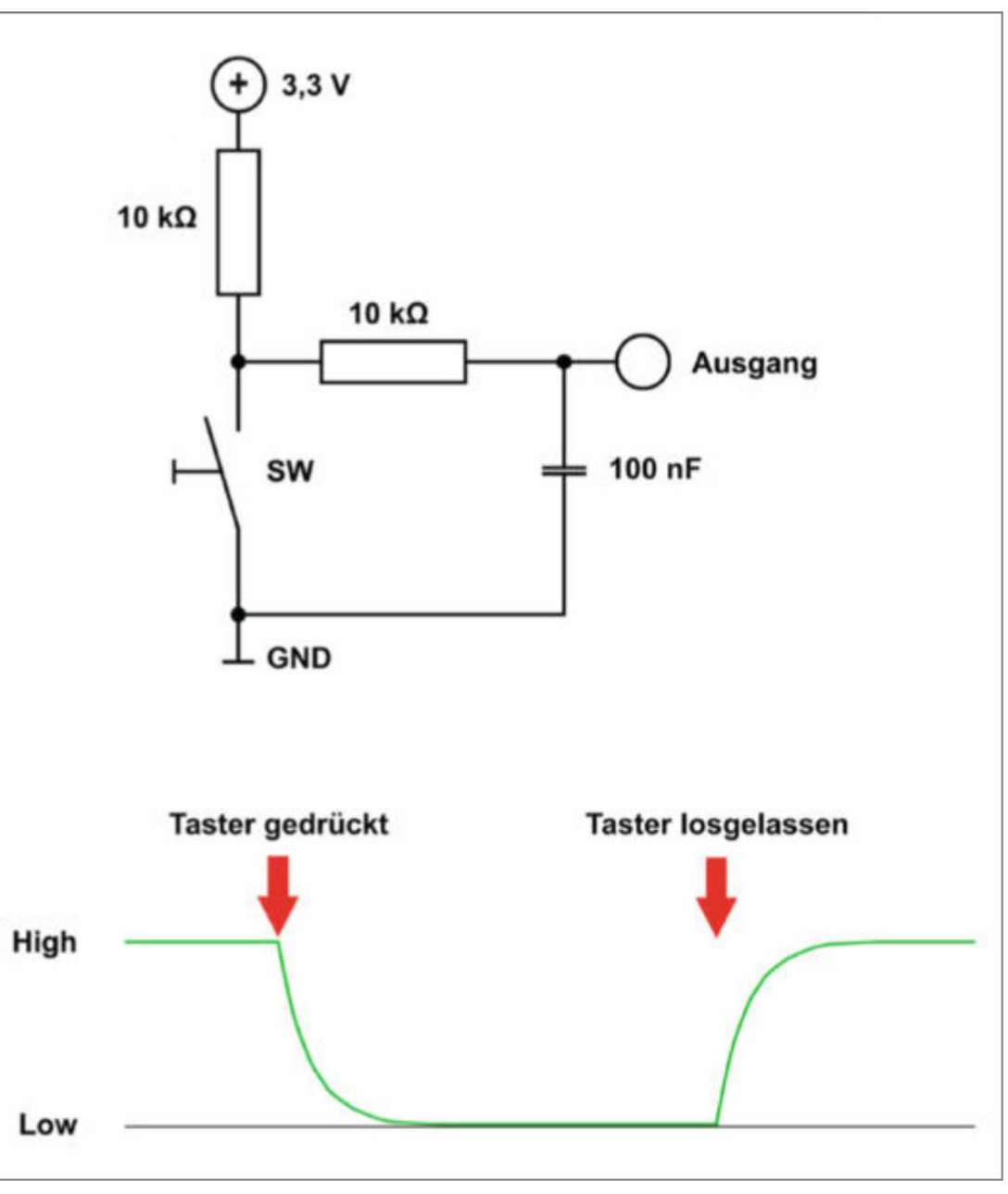


Bild 6: Diese Preller kann man mit einem RC-Glied glätten. Das erhoffte Ergebnis ist aber damit noch nicht erreicht.

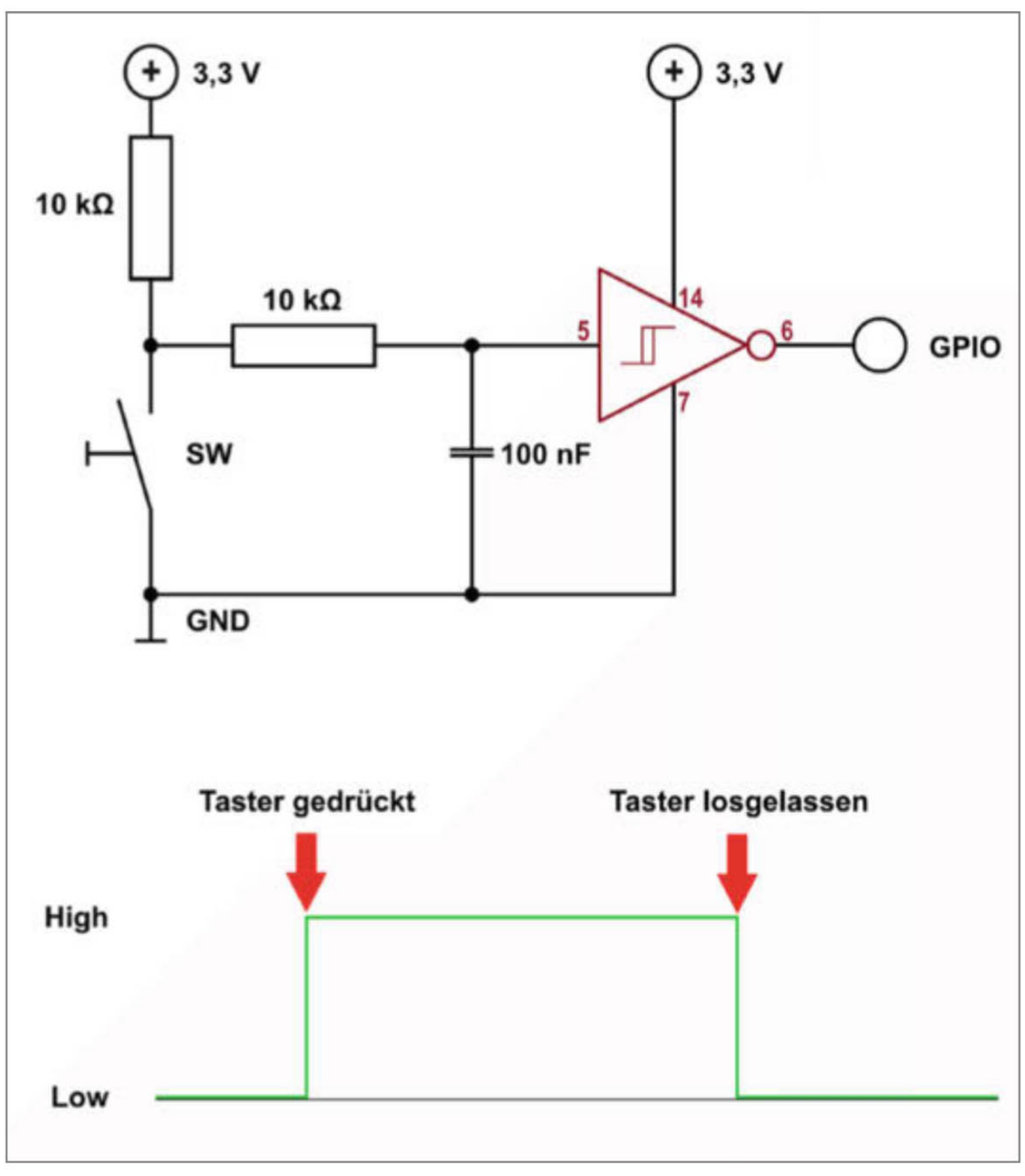


Bild 7: Erst mit einem invertierten Schmitt-Trigger erhält man ein eindeutiges und sauberes Rechtecksignal.

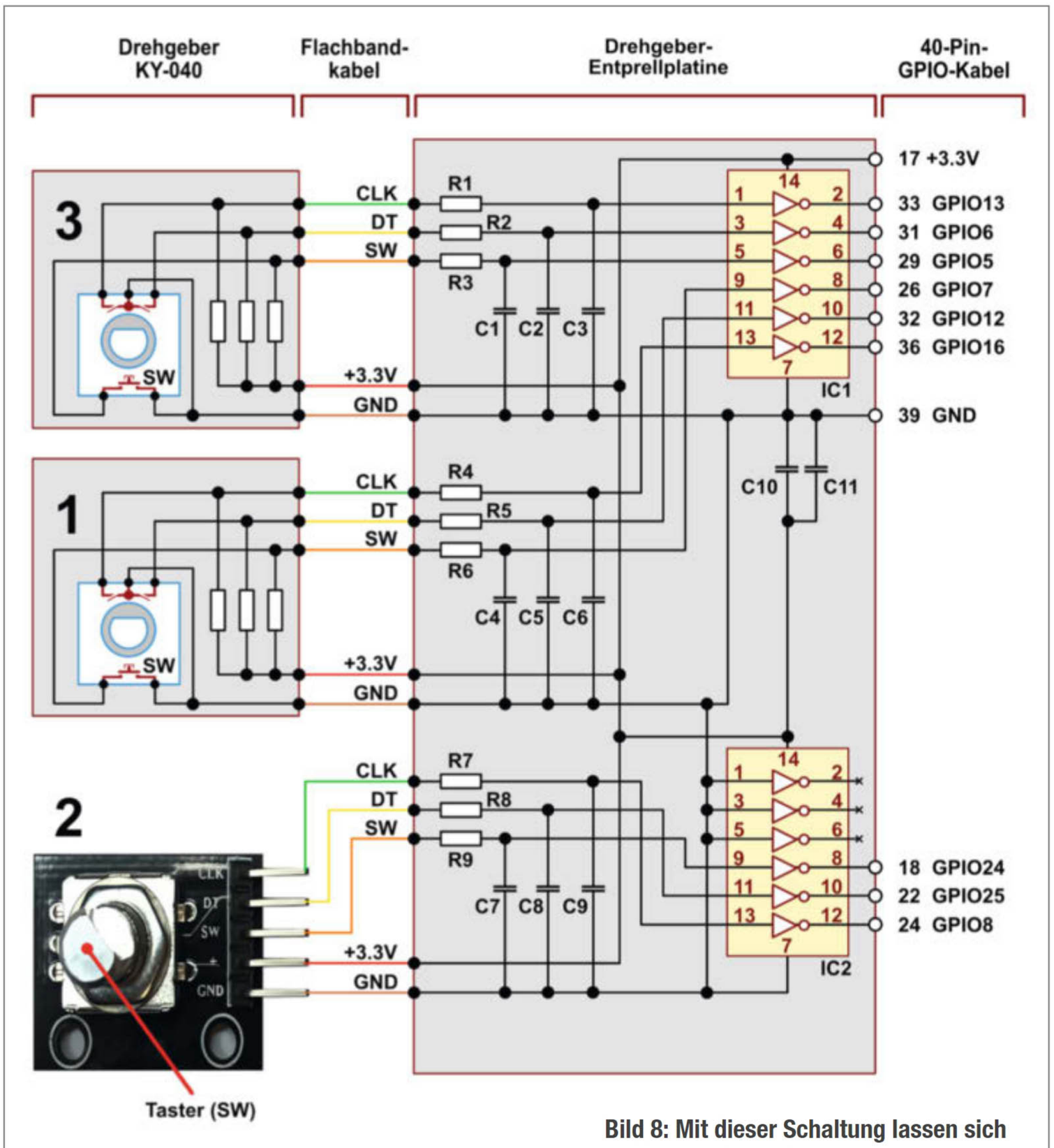


Bild 8: Mit dieser Schaltung lassen sich drei Drehgeber entprellen.

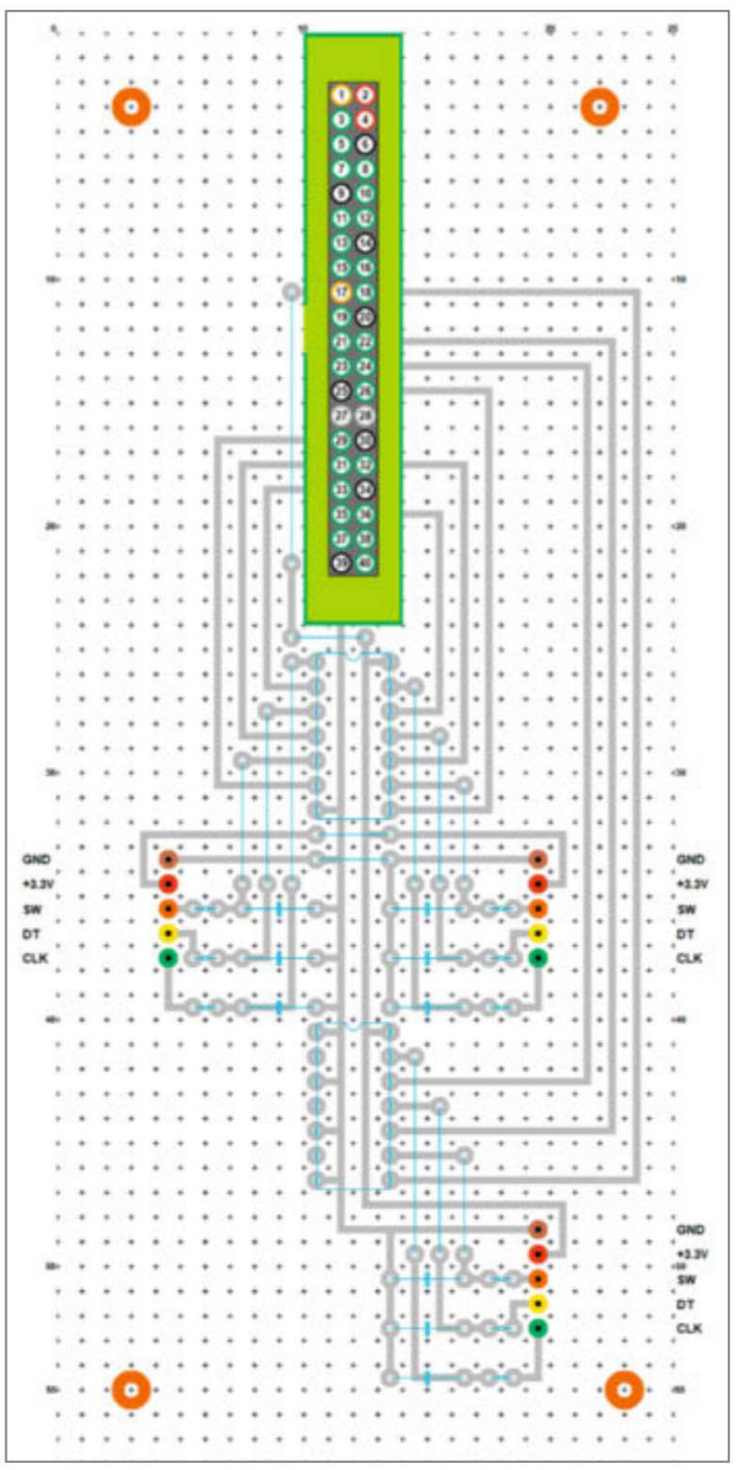


Bild 9: Das Layout für die Lochrasterplatine funktioniert mit dem „Argon Nanosound ONE“-Gehäuse. Die Grafik liegt in höherer Auflösung im GitHub-Repository des Projekts (siehe Link in Kurzinfo).

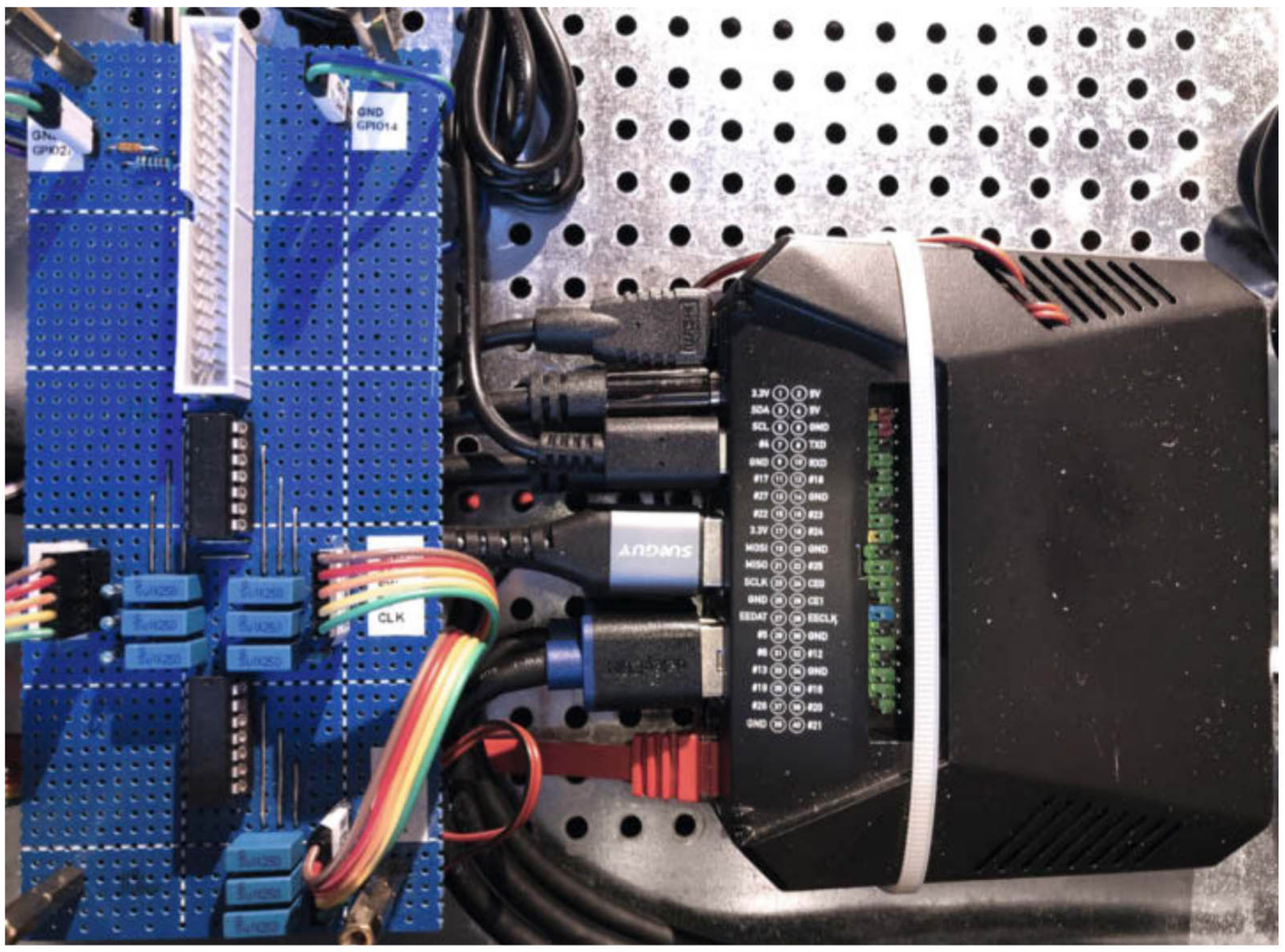


Bild 11: Die Drehgeber sind mithilfe von Steckbrückenkabeln mit der Platine verbunden (links). Jetzt fehlt nur noch das Kabel für den Raspberry Pi.

nikation verwendet und den Schaltplan entsprechend anpassen.

Die Drehgeber habe ich mit der Platine mithilfe von 40 cm langen Steckbrücken-Kabeln (female-female) verbunden. Diese hängen in der Regel als breiter Streifen aneinander. Diesen habe ich in drei schmalere Streifen zerteilt, jeweils mit den Farben Braun, Rot, Orange, Gelb und Grün (Bild 8 und 11). Der Raspberry Pi wird über ein 40-Pin-GPIO-Kabel angeschlossen.

Um die Drehgeber samt Platine an einer 10 mm dicken Frontplatte zu befestigen, habe ich Aussparungen von 8 mm Tiefe fräsen und Löcher bohren lassen – durch die ich die Drehgeber-Achsen stecken und diese dann mit der Mutter anschrauben konnte. Damit die Frontplatte nicht von den Drehknöpfen der Drehgeber zerkratzt wird, habe ich noch 1 mm dünnen Filz dahinter geklebt.

Plug-in für Volumio

Um die Drehgeber mit Volumio zu verwenden, nutze ich das Plug-in „Rotary Encoder II“. Es erlaubt den Einsatz von bis zu drei Stück. Zum Installieren des Plug-ins geht man in der

Volumio-Benutzeroberfläche links oben auf „Einstellungen/Plugins/System Hardware“. Dort sucht man nach „Rotary Encoder II“ und drückt auf den Button Installieren. Danach erhält man recht schnell die Rückmeldung, dass die Installation erfolgreich war und muss das Plug-in als Nächstes noch aktivieren. Anschließend wechselt man auf den Karteireiter „Installierte Plugins“. Dort klickt man auf die Einstellungen des Plug-ins.

Am Anfang sind alle drei verwendbaren Drehgeber noch deaktiviert (Off). Sobald man einen anschaltet (On), öffnet sich ein Konfigurationsfenster mit weiteren Optionen (Bild 12). Hier muss man die GPIOs für A (CLK), B (DT) und den Taster (SW) des jeweiligen Drehgebers eintragen. Sollte ein Drehgeber verkehrt herum funktionieren – das ist mir nämlich passiert –, muss man einfach nur die Werte bei A und B tauschen. Als Periode gibt man für den KY-040 Drehgeber 1/1 an. Bei den Drehfunktionen kann man zwischen vier Optionen wählen: Lautstärke, Titel vor/zurück, Suche im Titel und Websocket Nachricht senden. Letztere lässt sich etwa verwenden, um Kommandos an andere Plug-ins zu senden, sobald man den Drehgeber nach links oder rechts dreht (weitere Infos gibt es über einen Link in der Kurzinfo).

Wenn man Drehgeber verwendet, bei denen nur der Taster prellt, kann man diesen unter Entprell-Zeit per Software entprellen – mit der hier gezeigten Hardware-Lösung ist das nicht mehr notwendig. Schließlich stellt man für den Taster noch ein, wie er bei kurzem, langen oder doppelten Tastendruck reagieren soll, z.B. Play/Pause, Stummschalten oder das System neustarten. Danach ist der Drehgeber einsatzbereit und man kann die beiden anderen konfigurieren. Ich wünsche Ihnen viel Spaß beim Ausprobieren! —*akf*

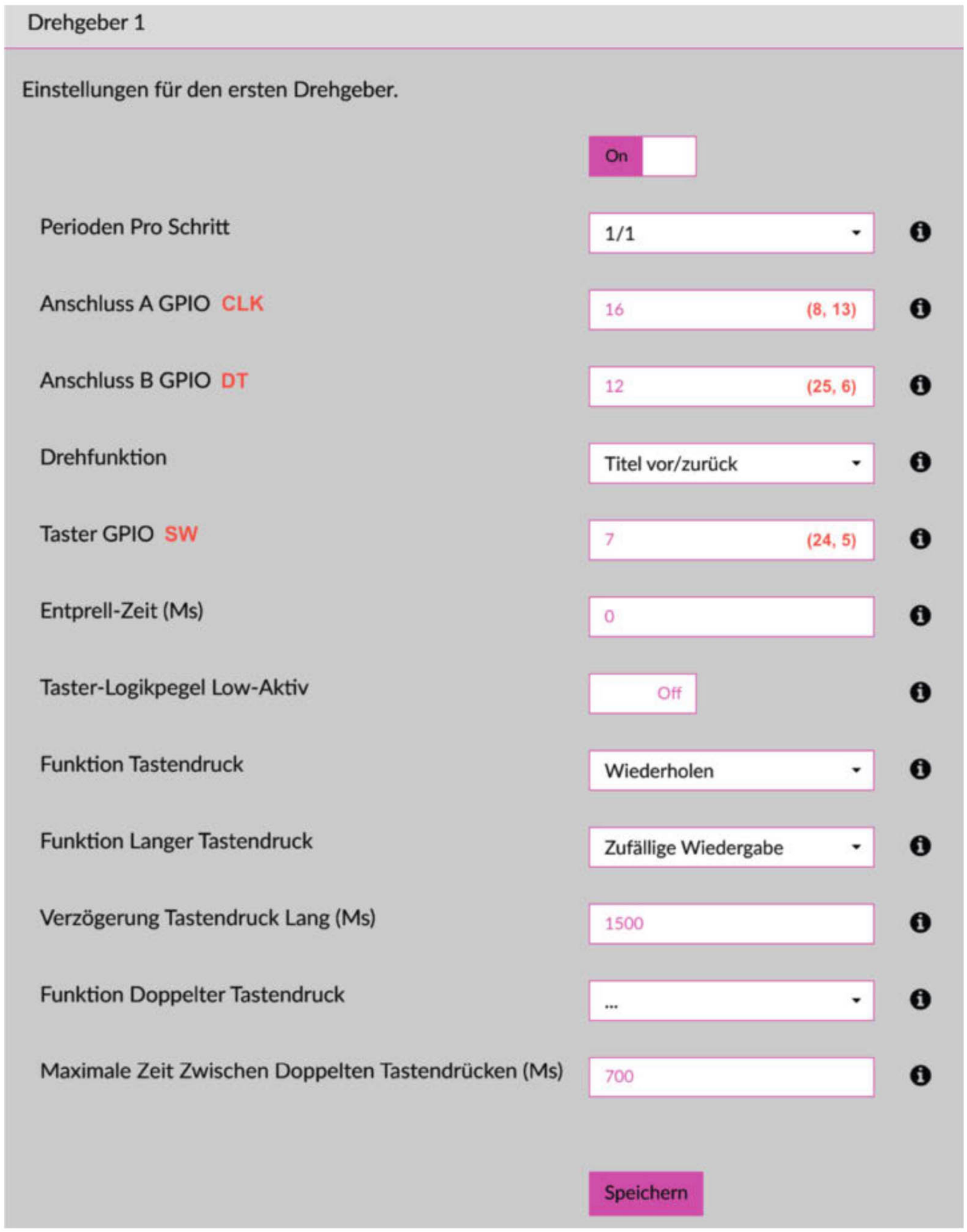
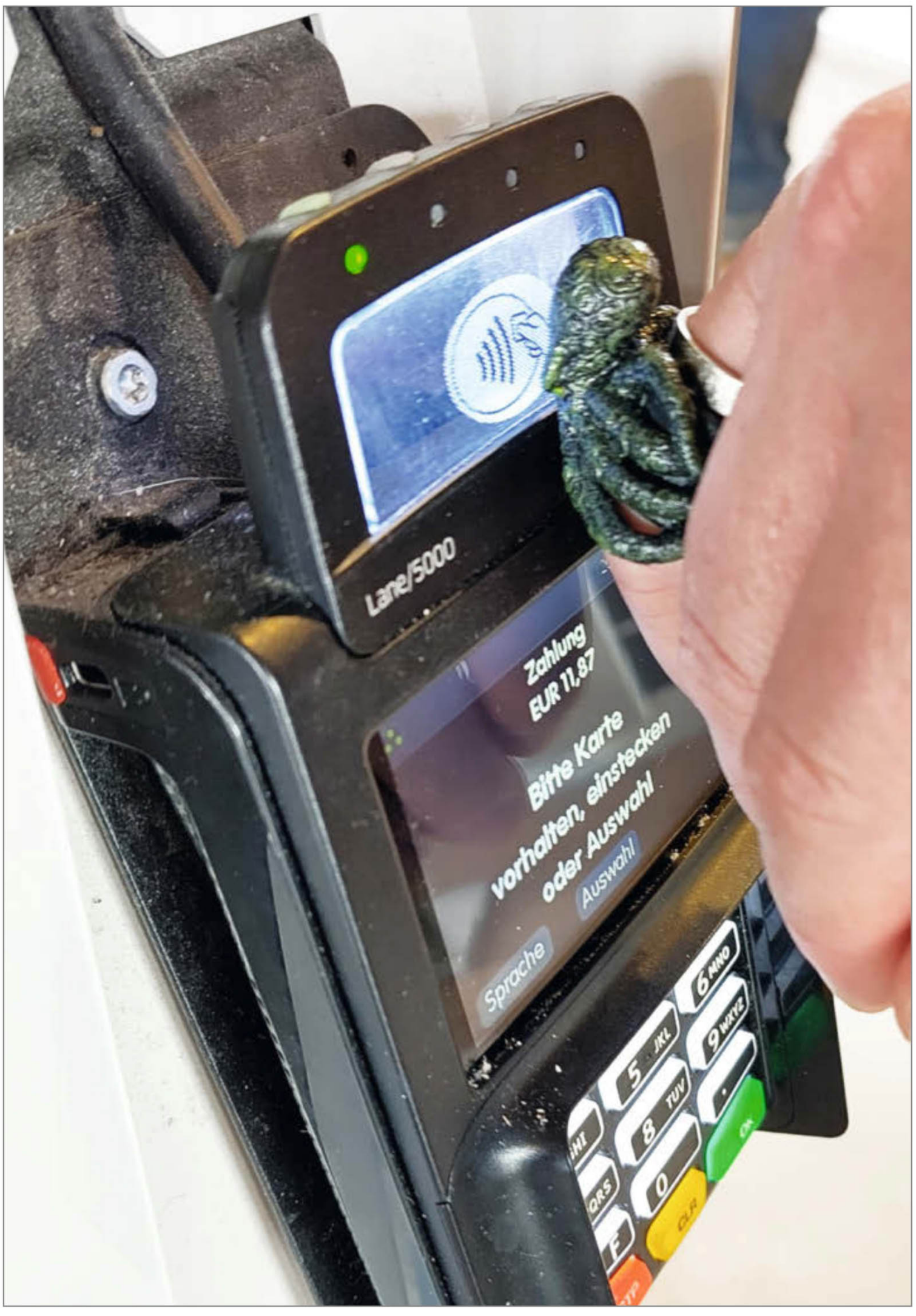


Bild 12: In den Einstellungen des „Rotary Encoder II“-Plug-ins in Volumio kann man jeden Drehgeber einzeln konfigurieren. In den Klammern stehen die Pins für die Drehgeber 2 und 3.

Payment-Ring im Eigenbau

Bezahlen praktisch im Vorbeigehen, ohne den Geldbeutel zu zücken – NFC-Chips, die etwa in Uhren, Armbändern oder Ringen eingebettet sind, machen es möglich. Wer solche Bezahl-Gegenstände selber baut, kann sie frei nach den eigenen Wünschen gestalten und kommt deutlich günstiger weg.

von Robert Kränzlein



Bei den sozialen Medien sehe ich in letzter Zeit immer öfter Werbung für sogenannte Payment-Ringe: Schmuckstücke mit eingebettetem Tag für NFC (Near Field Communication), mit denen man kontaktlos bezahlen kann. Die Idee an sich gefiel mir, aber die fertig angebotenen Ringe selbst sind eher Geschmackssache, weil sie für viele Szenarien meist zu modern gestaltet sind: Marktfrau Lena auf dem Mittelaltermarkt, Vikinger Jorn bei dem Metal-Festival oder ein Cosplayer auf einer Maker-Messe werden diese Ringe als eher unpassend empfinden.

Aber es spielen nicht nur ästhetische Aspekte eine Rolle: Bei mir lag die Motivation, mich mit dem Thema zu beschäftigen, im Wunsch, an Mautstationen nicht mehr die Motorradhandschuhe ausziehen zu müssen, sondern einen Payment-Chip irgendwie in die Rückseite des Handschuhs zu integrieren. Für erste Versuche (und weil gerade Winter war und deshalb keine Motorrad-Saison) beschloss ich, mich aber erst einmal an einen selbst gemachten Payment-Ring zu wagen.

Chip-Suche

Doch wo bekommt man einen passenden Chip her, um ihn in einen selbst gemachten Gegenstand einzubauen, der dem geplanten Zweck und den eigenen Vorstellungen entspricht? Klar, naheliegend ist es, einen fertigen Ring zu kaufen, den Chip herauszuholen und ihn dann neu zu umhüllen.

Die am Markt verfügbaren Payment-Ringe sind oft aus Holz oder aus Keramik, da massives Metall ringsherum das Auslesen des NFC-Chips verhindern würde. Mir ist zwar ein Ring aus Gold bekannt, aber ich vermute jedoch, hier sitzt der Chip im eingearbeiteten Stein (nebenbei kostet der Ring rund 1500 Euro). Bei Holz wäre es sicher machbar, einen solchen Ring Schicht für Schicht abzutragen, um an den innen liegenden Chip zu gelangen. Bei Keramik wird das etwas schwieriger, denn der Chip darf ja nicht beschädigt werden. In bei-

den Fällen ist es ein riskantes und teures Unterfangen, da Holz- oder Keramikringe in der Regel gut über hundert Euro kosten.

Auf der Suche nach möglichst preiswerten Payment-Wearables bin ich dann auf den Hersteller K-Pay gestoßen, der vor allem Armbänder, aber auch ein Silikonbändchen mit integriertem NFC-Chip anbietet. Dieses sogenannte Maverick-Bändchen (Bild 1) ist dafür gedacht, es in selbst geflochtene Armbänder einzufügen, und kostet 19 Euro plus 10 Euro Versand nach Deutschland (Bezugsquelle siehe Link in der Kurzinfo). Das war ein Betrag, bei dem ich es verschmerzen konnte, ein oder zwei Stück bei Experimenten zu zerstören.

Beherzter Schnitt

Das Payment-Bändchen zeigt auf der Vorderseite ein Logo aus dem Buchstaben K, kombiniert mit einem WLAN-Zeichen (Bild 1). Auf der Rückseite zeichnet sich bei genauem Hinsehen ein Rechteck von 30×13 mm ab, das minimal tiefer liegt und eine andere Textur aufweist. Schneidet man an der Kante dieses Rechtecks mit einem scharfen Skalpell entlang – besser leicht außerhalb, keinesfalls innerhalb –, wird schnell klar, dass genau dieser Lappen in der Produktion anfangs offen ist und dann nach Einlegen des RFID-Chips mit Antenne nachträglich verschweißt wird (Bild 2).

Hierbei entsteht scheinbar so viel Hitze, dass sich sogar die Typennummer F8A03 oder FBA03 in den Kunststoff der Vorderseite eindrückt. Der Chip selbst hat eine Seitenlänge von etwas über 5 mm. Das umgebende Feld mit den Maßen 27×13 mm besteht aus einer flexiblen Folie mit einer Antenne (Bild 3). Das Hacking-Werkzeug Flipper Zero identifiziert den NFC-Tag als ISO 14443-4 (NFC-A), wahrscheinlich von Infineon.

Testringe

Im Anschluss habe ich für einen ersten Test einen ganz einfachen breiten Ring in Autodesk Fusion entworfen, an dessen Innenseite eine Aussparung für den Chip mit Antenne vorhanden ist (Bild 4). Bei filamentworld.de habe ich dann 50-Gramm-Samples diverser, zum Teil metallhaltiger, Filamente bestellt und den Ring damit ausgedruckt, um zu sehen, inwiefern den NFC-Chip dadurch abgeschirmt wird (Bild 5).

Das ermutigende Ergebnis: Bei keinem der Filamente, die ich probiert habe, gab es Probleme, den Chip dahinter auszulesen. Vermutlich ist der Metallgehalt dafür zu gering, belegen ließe sich das wohl nur mit aufwendiger

Kurzinfo

- **Bezahlring nach den eigenen Vorstellungen selber in 3D drucken**
- **Flexiblen und günstigen NFC-Chip in einen Fingerring einbetten**
- **Individuell anpassbarer Basisring als OpenSCAD-Programm**

Checkliste

Zeitaufwand:
2 Stunden (ohne Druckzeit)

Kosten:
ab 30 Euro (ohne 3D-Druck-Material)

Material

- **Bezahlarmband**
  hier K-Pay Maverick-Bändchen
- **Filament oder Resin**
  je nach gewünschter Gestaltung

Alles zum Artikel im Web unter
make-magazin.de/xk1a

Werkzeug

- **3D-Drucker** FDM oder Resin, je nach gewünschtem Material und nach Gestaltung
- **Lack und Klebstoff** für die Montage und das Finish

Mehr zum Thema

- Gratis-3D-CAD für Maker, Make 4/22, S. 76
- Stefan Porteck und Jörg Wirtgen, Gute Karten, Drahtloses Bezahlen mit Handys und Uhren, c't 9/21, S. 58
- Markus Montz, Luftbuchungen, Wie das Bezahlen mit Uhr und Smartphone funktioniert, c't 9/21, S. 64

Bild 3: Der entnommene NFC-Chip

Bild 1: Die Vorderseite des Maverick-Bändchens

Bild 2: Aufschneiden auf der Rückseite; das unbeschädigte Band vorne zeigt deutlich die rechteckige Vertiefung.

Bild 4: NFC-Chip im Testring

Labormesstechnik. Da ich diese nicht habe, habe ich es eben einfach exemplarisch ausprobiert.

Getestet wurden die metallhaltigen Materialien Metalfil Kupfer und Protopasta Steel sowie die Metalleffekt-Filamente Silk-PLA Kupfer und Silk-PLA Silbergrau. Die beiden Filamente mit Metallpartikeln (auf Bild 5 in der Mitte) könnte man mit viel Aufwand entsprechend polieren, aber nur bei großen, ebenen Flächen ohne viele Verzierungen. Mit persönlich hat am besten das silberne Silk-PLA gefallen, das fast ohne Nacharbeit einen schönen metallischen Glanz bietet.

Gestaltungsspielraum

Nun sollte aus dem Testring ein passendes Schmuckstück werden. Hier kann jeder selbst kreativ werden oder auf den diversen Plattformen nach einer Vorlage suchen. Ich habe mich für einen Cthulhu-Ring von Thingiverse entschieden (Bild 6, Link zum Download siehe Kurzinfo). Aufgrund der feinen Oberflächendetails habe ich den Ring am Ende doch auf einem Resindrucker gedruckt; letztendlich muss man die Entscheidung über die geeignete Fertigungstechnik für jedes selbst gebaute Payment-Gadget selber treffen.

Das heruntergeladene 3D-Modell – den Schmuckring – habe ich im Slicer so skaliert, dass der durch die Tentakel gebildete ringförmige Hohlraum 2,5 mm größer ist als der des zuvor von mir erstellten Basisrings mit der Aussparung für den NFC-Chip. Legt man beides im Slicer übereinander, überschneiden sich beide Objekte (Bild 7), und das Ganze habe ich dann auf einem SLA-Drucker aus Standard-Resin gedruckt (Bild 8). Wichtig ist, beim Skalieren ausschließlich die Größe des Schmuckrings anzupassen! Der Basisring darf nicht gedehnt werden, da sonst die Aussparung für den NFC-Chip nicht mehr passt. Eventuell nötige Anpassungen am Basisring müssen stattdessen vorab im CAD-Programm vorgenommen werden, dazu gleich mehr. Den fertigen Ring habe ich nach dem Druck entsprechend gereinigt, grundiert und mit Acrylfarben (Vallejo Game Color) lackiert (Bild 9).

Zum Testen genügt es, den NFC-Chip in die Aussparung zu legen und innen ein Stück Tesafilm darüberzukleben. Für die dauerhafte Montage nach einem erfolgreichen Test kommen entweder UV-härtender Sekundenkleber infrage oder eine Einbettung in transparentem Silikon. Die zweite Variante ist unter Umständen weniger dauerhaft, hat aber den Charme, dass der Chip jederzeit wieder in ein neues Schmuckstück, je nach Saison, Geschmack oder Anlass, versetzt werden kann.

Zahlungen freischalten

Bevor wirklich bezahlt werden kann, steht aber noch die Aktivierung der Bezahlfunktion auf dem Programm, die über die NFC-Plattform Fidesmo abgewickelt wird. So kann man nach aktuellem Stand beispielsweise die VISA-Karten einiger Volksbanken, der Aachener Bank, von Comdirect und Consorsbank direkt mit Fidesmo verknüpfen.

Ich musste hingegen erst eine virtuelle Kreditkarte bei VIMpay anlegen. Die Folge waren zwei zu installierende Apps, zwei anzulegende Accounts, zweimal Identifizierung. Dann verschiebt man Geld an VIMpay, um damit anschließend das Wearable aufzuladen. Es kostet ein paar Nerven, bis all dies erledigt ist. Doch ist dieser Prozess erforderlich – auch viele der mir bekannten Payment-Schmuckstücke und z. B. auch die Bezahl-Uhr Swatch-PAY nutzen den gleichen technischen Zahlungsprozess.

Als Tipp an alle, die am Ende doch lieber auf ein fertig erhältliches Payment-Schmuckstück zurückgreifen möchten: Man sollte die FAQs und Angaben zum Payment-Prozess im Vorfeld genau anschauen. So setzt etwa der Anbieter McLear einen britischen Ausweis oder Führerschein für die Identifikation voraus, den hierzulande die wenigsten besitzen dürften. Wer sich tiefer mit dem Thema beschäftigen möchte, für den haben wir in der Kurzinfo zwei Artikel der c't-Kollegen aus deren Heft 9/21 aufgeführt.

Ringbau mit OpenSCAD

Da kaum ein Leser exakt meine Ringgröße hat, habe ich noch einen neuen Grundring entworfen, der mittels Parametern ganz einfach an die eigene Ringgröße angepasst werden kann. Die bekommt man beispielsweise, indem man den Innendurchmesser eines gut passenden anderen Rings mit dem Messschieber ermittelt. Für den anpassbaren Grundring entschied ich mich für OpenSCAD als Software, da diese kostenfrei zur Verfügung steht und auch viel schneller zu installieren und zu starten ist als etwa Fusion.

Aus den Außenmaßen des NFC-Chips ergeben sich Länge und Breite der Aussparung innen am Ring. Allerdings kann SCAD nicht mit Bogenlängen rechnen. Die Länge eines Kreisbogens b ergibt sich aus der Formel:

$$b = 2 \cdot \pi \cdot r \frac{\alpha}{360}$$

Bild 5: Testringe aus den Materialien Silk-PLA Kupfer, Protopasta Steel, Metafil Kupfer und Silk-PLA Silbergrau (von links nach rechts)

Bild 6: Der Cthulhu-Ring von Thingiverse

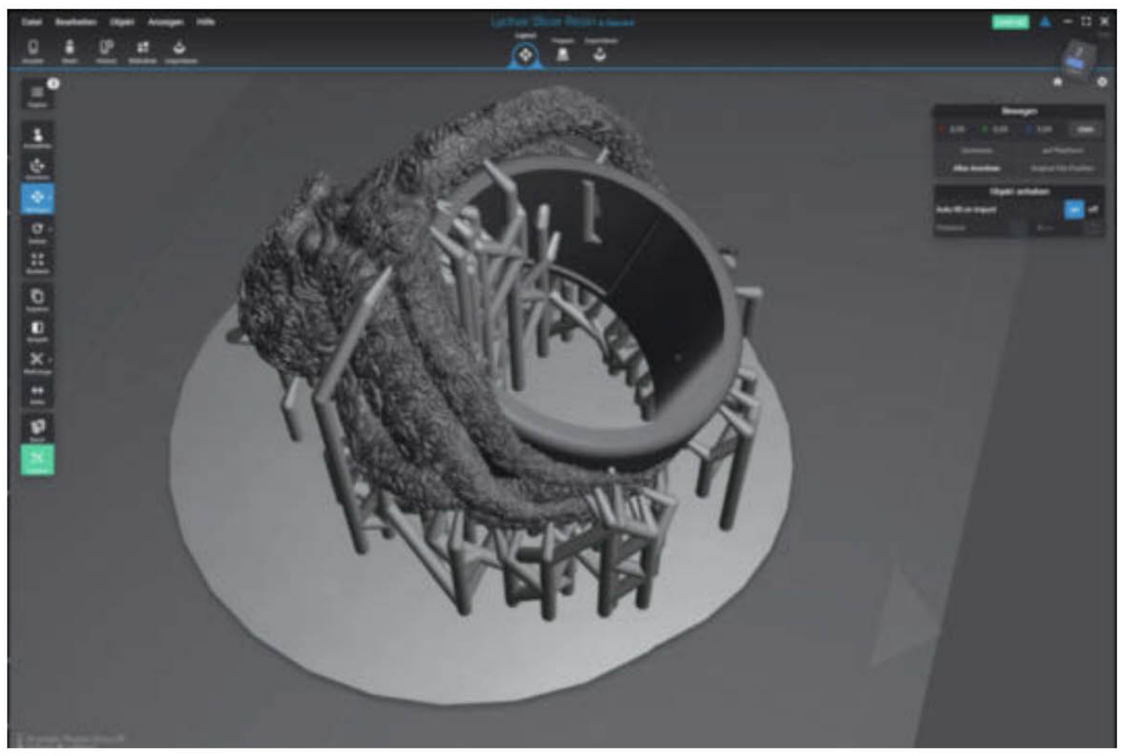


Bild 7: Basisring und Schmuckring werden im Slicer kombiniert.

Bild 8: Der Ring nach dem Druck

Stellt man diese nun um nach dem Winkel α

$$\alpha = \frac{b \cdot 360}{2 \cdot \pi \cdot r}$$

beziehungsweise setzt statt des Radius *r* den ohnehin als bekannt vorausgesetzten Durchmesser *d* ein, ergeben sich dann durch

$$\alpha = \frac{b \cdot 360}{\pi \cdot d}$$

die Winkel für die Aussparungen der Antenne und des NFC-Chips.

Das OpenSCAD-Programm ist ausführlich kommentiert und über den Link in der Kurzinfo herunterladbar. Angepasst werden sollte in erster Linie der Parameter `ring` für den Innendurchmesser. Bei Bedarf nimmt man noch Änderungen bei `breite`, `staerke` und `fase` vor. Änderungen an den anderen Parameter sind nur notwendig, wenn man einen anderen NFC-Chip findet und das OpenSCAD-Programm daran anpassen möchte. Meine Empfehlung: Ich würde immer zuerst einen schnellen Testring drucken, nur um den richtigen Durchmesser zu testen, und zwar aus dem Material und mit dem Drucker, die auch später beim echten Payment-Ring zum Einsatz kommen. Ein halber Millimeter kann bei einem Ringdurchmesser schon entscheidend sein und das fertige Bezahl-Schmuckstück soll am Ende ja weder drücken noch vom Finger rutschen.

Ausblick

Nachdem ich den Winter über nun Ringe gebaut habe, steht das Frühjahr vor der Tür und ich kann mich optimistisch mit den gemachten Erfahrungen an das Thema meines eigentlich gewünschten Payment-Handschuhs wagen. Beim Schreiben dieses Artikels habe ich ganz neu noch den Schlüsselanhänger VIMpayGO entdeckt. Das ist eine Mini-Prepaid-Kreditkarte in einem Plastikhalter (den man natürlich entfernen kann) mit den Maßen 26 × 15 mm. Für Ringe wohl eher nicht geeignet, da sich eine solche Karte nicht biegen lässt wie die flexible Antenne des K-Pay, aber sicher interessant für andere Accessoires. Wie wäre es mit einem Payment-Zauberstab? Dann sind mit einem Schwung auf den Kartenleser und einem „Anapneo" die Halspastillen in der Apotheke bezahlt! —*pek*

Bild 9: Der fertig lackierte Ring, bereit zum kontaktlosen Zahlen

Bild 11: Im Schlüsselanhänger VIMpayGO steckt eine Mini-Prepaid-Kreditkarte, die sich natürlich auch in selbst gebaute Payment-Gegenstände einbetten lässt.

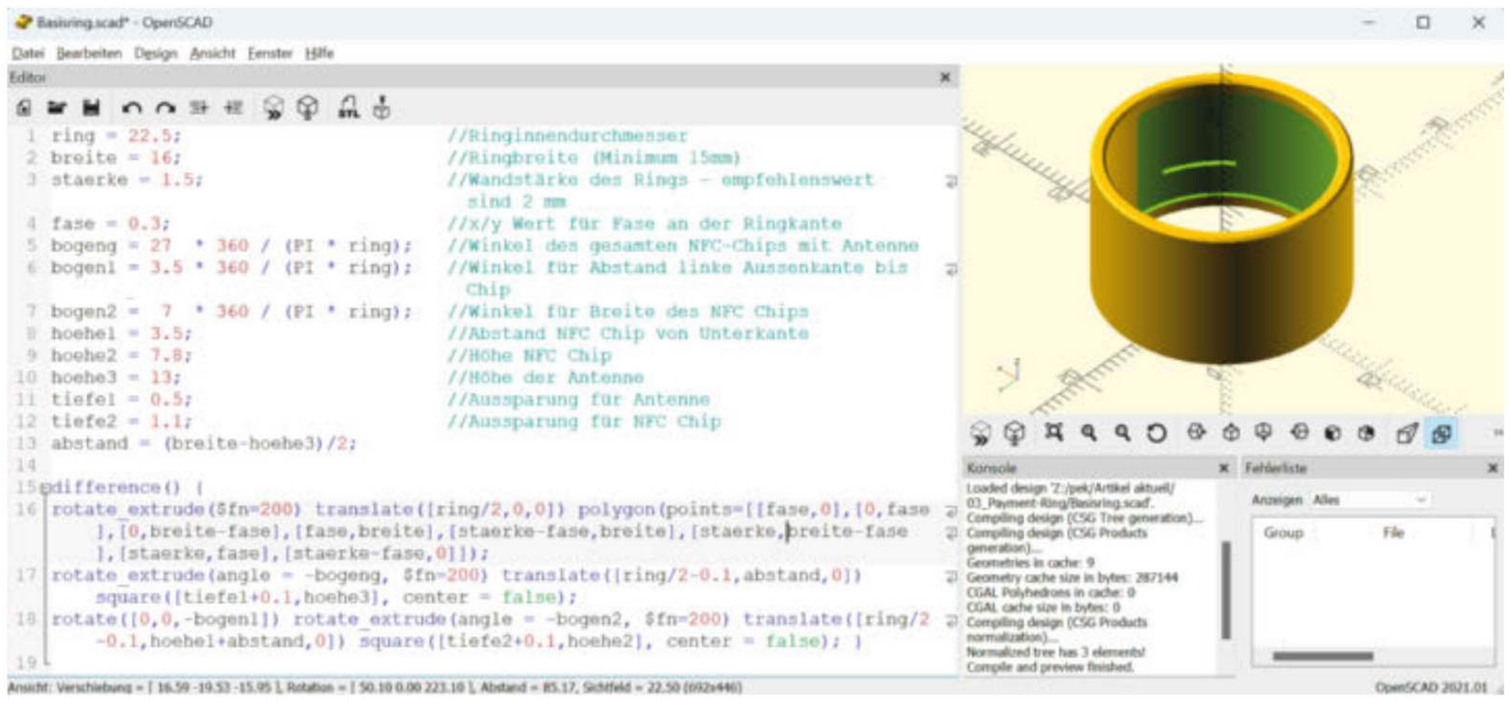
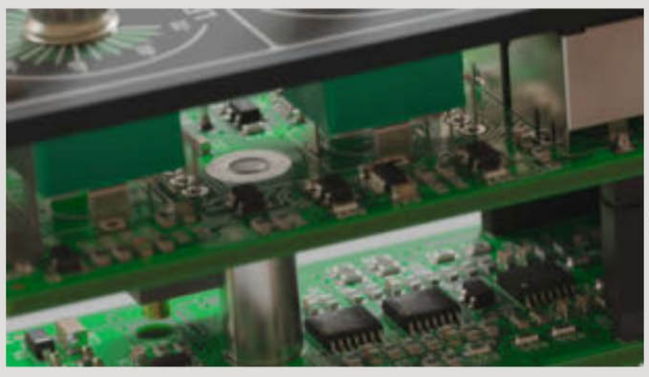


Bild 10: Die Parameter des OpenSCAD-Programms für individuelle Basisringe sind im Code ausführlich kommentiert.

Ausprobiert —von Make:—

pcb2blender

Platinen aus KiCAD fotorealistisch rendern



30350n/pcb2blender

Möchte man für die Dokumentation und Präsentation von Elektronikgeräten die Leiterplatten abbilden, so geschah dies in der Vergangenheit gerne in 2D oder als Foto. Inzwischen sind die 3D-Darstellungen aus E-CADs immer besser geworden. Manchmal muss es aber mehr sein, etwa weil die Auflösung der 3D-Vorschau im E-CAD nicht frei wählbar ist.

Hier kommt das Projekt pcb2blender ins Spiel. Es bringt einen .pcb3d-Exporter für KiCAD und einen Blender-Importer für die .pcb3d-Dateien mit. Das Projekt pcb2blender ist schon etwas länger in der Mache, aber immer noch aktiv entwickelt, wie die Commit-Daten auf GitHub zeigen. Hier lief die Installation auf einem aktuellen Blender 4.1.1 problemlos. Nur für die Installation in KiCAD mussten wir im Test den manuellen Weg gehen, wie auf der GitHub-Seite von pcb2blender dokumentiert. Der Weg über die „Plugin- und Contentverwaltung“ funktionierte nicht, weil die gesuchten Dateien nicht gefunden werden konnten.

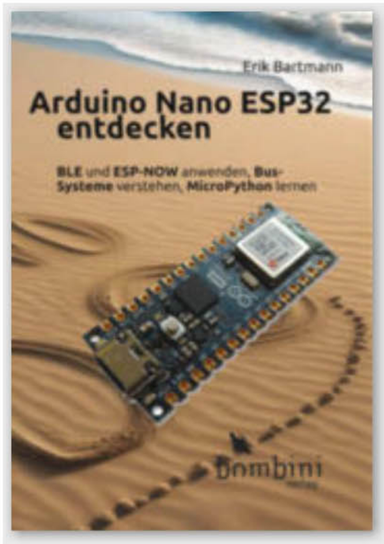
Die Materialien werden automatisch vergeben und sind schon im Viewport von Blender schicker als etwa in KiCAD. Ein Render mit nur einer Lichtquelle ist bereits ein Aha-Moment. Gibt man sich etwas mehr Mühe, so bekommt man nahezu fotorealistische Bilder. Mit den Tools für Animation in Blender steht dann der gerenderten Montageanleitung nichts mehr im Wege. Nur fehlende Blender-Kenntnisse vielleicht. —*caw*

URL	github.com/30350n/pcb2blender
Preis	gratis

Arduino Nano ESP32 entdecken

BLE und ESP-NOW anwenden, Bus-Systeme verstehen, MicroPython lernen

Im Arduino Nano ESP32 steckt so viel Power, dass die Möglichkeiten fast unendlich erscheinen. Nicht umsonst ist der ESP32 die Grundlage vieler Maker-Projekte.



Das Buch „Arduino Nano ESP32 entdecken“ ist ein hervorragendes Werk für Einsteiger. Zum einen beginnt der Autor Erik Bartmann bei den ganz grundlegenden Basics. Was ist der Arduino Nano ESP32? Was ist der Unterschied zu einem Arduino Nano? So ziemlich jede Frage, die bei der Auseinandersetzung mit diesem Board auftauchen kann, wird direkt am Anfang des Buches geklärt. Und das zieht sich über die gesamten 337 Seiten. Jeder Fachbegriff wird mit einfachen Worten erklärt, sobald er das erste Mal Verwendung findet. Dadurch bietet dieses Fachbuch wirklich einen vollumfänglichen Einstieg in die Thematik.

Im gesamten Verlauf des in zwei Teile (Programmierung C++ bzw. MicroPython) aufgeteilten Buches wird man mit farbigen Bildern an die Hand genommen. Jeder Schritt wird mit einem Foto genau dargestellt, jede Codezeile ist abgedruckt. So kann man wirklich nicht aus dem Tritt kommen und weiß immer, was wo geklickt oder eingesteckt werden muss.

Das führt allerdings dazu, dass eine andere Entscheidung für dieses Buch etwas seltsam anmutet: Immer wieder wird auf Websites verwiesen. Anstatt jetzt aber einen QR-Code zu dieser URL abzudrucken, steht der gesamte Link im Fließtext auf den Seiten. Den mehrzeiligen Link zu den Espressif-Docs abzutippen, wirkt dann schon fast steinzeitlich.

Abschließend lässt sich aber sagen, dass „Arduino Nano ESP32 entdecken“ wirklich das perfekte Einsteigerbuch ist, das Schritt für Schritt in alle Möglichkeiten des Arduino Nano ESP32 einführt. —*das*

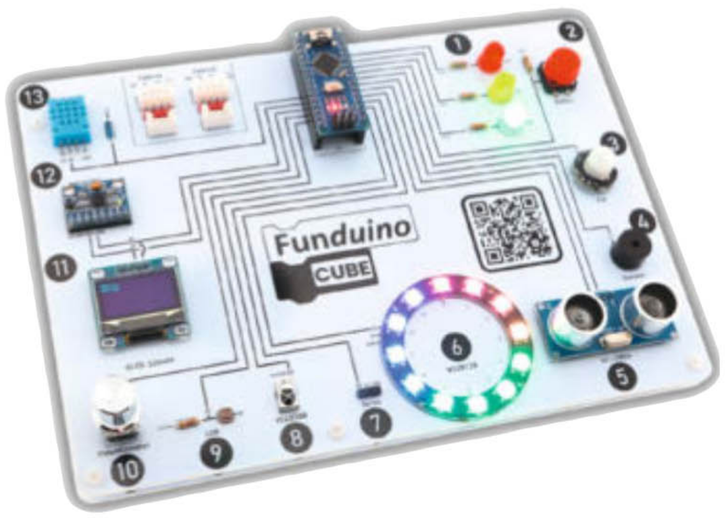
Autor	Erik Bartmann
Verlag	bombini
Umfang	342 Seiten
ISBN	978-3-946496-36-6
Preis	34,95 €

Funduino Cube

Arduino-Lehrsystem

Controller-Board, Sensoren, Tasten und Display – alles, was man für erste Experimente mit dem Arduino braucht, bietet der Funduino Cube auf einem Board an. Das Lehrsystem soll laut Hersteller Funduino für Schüler und Schülerinnen ab der 5. Klasse geeignet sein. Aber auch Erwachsene können sich damit in die Arduino-Welt einarbeiten. Die Zeiten wackeliger Breadboard-Schaltungen selbst für einfache Thermometer-Projekte sind damit beim Lernen erst einmal vorbei.

Auf dem Board arbeitet ein Arduino Nano R3. Ihm zur Seite stehen ein 0,96-Zoll-Display mit 128 × 64 Pixeln, ein LED-Ring mit zwölf RGB-LEDs vom Typ WS2812, drei einzelne RGB-LEDs, ein Ultraschall-Entfernungssensor, Temperatur- und Luftfeuchtigkeitsmesser, ein Lage- und Beschleunigungssensor (ADXL335), Taster, Potenziometer, Infrarot-Fernbedienungssender und -empfänger inklusive Fernbedienungsgeber, ein Piezo-Lautsprecher sowie ein lichtempfindlicher Widerstand (LDR) und ein Bewegungsmelder. Außerdem gibt es einen Anschluss für einen (mitgelieferten)



Funduino GmbH

Servo vom Typ SG90, eine I²C- und eine serielle Schnittstelle.

Programmieren kann man das Board mithilfe der vom Fraunhofer-Institut entwickelten grafischen Programmierumgebung OpenAlberta oder der textbasierten Arduino IDE. Das ebenfalls mitgelieferte 130-seitige Lehrbuch bringt in 13 Lektionen jeweils Code-Beispiele für beide Möglichkeiten. Die Stromversorgung erfolgt über USB. Zusätzlich gibt es weitere wichtige Informationen in der Online-Dokumentation zur Hardware und zur grafischen sowie zur textbasierten Programmierung. —*hgb*

Hersteller	Funduino GmbH
URL	funduinoshop.com
Preis	54,90 €

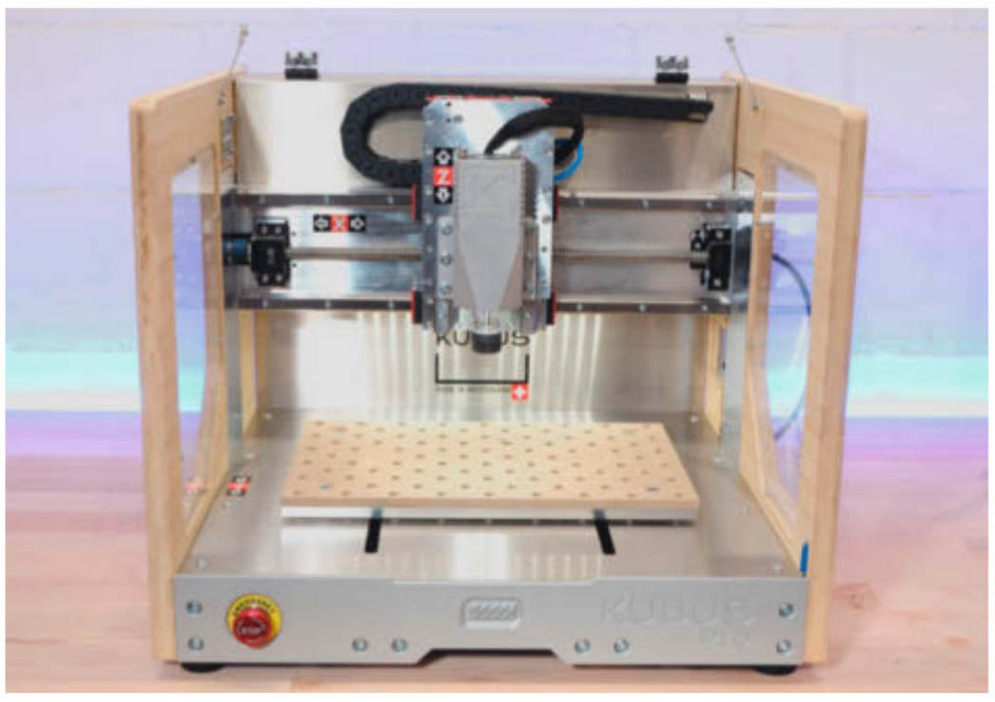
NENO-CNC KUBUS Pro

Geschlossene Desktop-CNC-Fräsmaschine

In den letzten drei Jahren entwickelte eine kleine Schweizer Firma eine neue Generation von Desktop-CNC-Fräsmaschinen: die KUBUS Pro. Als einzige geschlossene Desktop-CNC-Fräsmaschine unter 5000 Euro in Europa kann sie Aluminium schnell und genau bearbeiten.

NENO-CNC GmbH

Laut Hersteller erzielte sie in Tests (3 m/min, 1 mm DOC und 6-mm-Fräser in Vollnut) durch 6082er-Aluminium sehr gute Fräsqualitäten. Als 4-Achsen-Fräsmaschine bietet sie präzises Fräsen mit hoher Oberflächenqualität, ideal für Heimanwender, Kleinbetriebe und Schulen. Der Grundrahmen aus 6082-Aluminium-Präzisionsgussplatten, HGR15-Linearschienen und 12-mm-Kugelumlaufspindeln garantiert Stabilität und Präzision. Mit 350×220 mm Größe bietet der Maschinentisch erstaunlich viel Platz für ein Desktop-Gerät.



NENO-CNC GmbH

Die 500-W-Frässpindel erreicht 18.000 U/min. Eine eigens entwickelte Steuerung auf Basis des Raspberry Pico RP2040 verhindert Schrittverluste und ermöglicht Vorschübe bis zu 7000 mm/min. Sicherheitsfeatures und Unterstützung verschiedener Steuersoftware erhöhen Sicherheit und Flexibilität. Das erhältliche Zubehör wie etwa Spannsysteme erhöht die Anwendungsvielseitigkeit. Die KUBUS Pro ist aktuell im Vorverkauf mit Ermäßigung erhältlich. Im Shop werden auch passende Erweiterungen und Zubehör angeboten. —*caw*

Hersteller	NENO-CNC GmbH
URL	kubuscnc.ch
Preis	ab 3850 €, exkl. USt. / Zoll

Arducam PiINSIGHT

Ausprobiert —von Make:—

Kamera mit KI-Beschleuniger

Die Arducam PiNSIGHT mit eingebautem KI-Beschleuniger ermöglicht die KI-Auswertung von Kamerabildern und -videos. Basis der Kamera ist ein 12-Megapixel-Sensor (Sony IMX378).

Die PiNSIGHT kommt als Modul zum Aufstecken auf einen Raspberry Pi. Das Modul besteht aus einem massiven Alublock, der auch als Kühler des KI-Moduls dient, mit montierter Platine. Der Raspi wird hinten auf das Modul geschraubt. Unten befindet sich ein Stativgewinde. Vorn schaut die Kameralinse heraus, an der Seite gibt es einen USB-C-Port für die Verbindung zum Pi. Die USB-Verbindung ragt allerdings sehr heraus und macht das Ganze etwas unpraktisch, gleichwohl man ja den Kabel-Kraken-Look vom Raspberry Pi gewohnt ist. Allerdings ist der angedachte Anwendungszweck auch eher die Entwicklung von KI-Applikationen und -Hardware und dank des USB-C kann man das Modul räumlich vom Raspi trennen. Praktisch ist dagegen, dass man hinten noch ein LCD-Shield anflanschen kann.

Empfohlen wird ein Pi5, aber beim Test in der Redaktion funktionierte es ohne Probleme auch mit einem Pi4. Der Pi5 bietet allerdings eine schnellere Anbindung an Massenspeicher, wenn viel hochaufgelöstes Video aufgezeichnet werden muss. Die Installation der Toolkits, Abhängigkeiten und Demos auf Pi OS ist schnell mit zwei Zeilen Bash erledigt. Dabei kommt DepthAI als Toolkit zum Einsatz; die Demos sind alle in Python im Quellcode vorhanden. Die PiINSIGHT soll laut Hersteller auch mit OpenVINO (und den vielen Trainingsdaten im „Open Model Zoo") harmonieren, was wir aber im Einzelnen in diesem Kurztest nicht ausprobiert haben.

Insgesamt sind die Demos erhellend und funktionieren, allerdings sollte man keine Plug-and-Play-KI erwarten: Eigene Anwendungen inklusive Datenerfassung, Training, etc. erfordern eine intensive Beschäftigung mit der Materie und profunde Python-Kenntnisse. Die Kamera wird auch ganz bewusst von Arducam als Basis für Proof-of-Concept-Entwicklung (POC) beworben. Arducam hat uns aber versprochen, die Dokumentation noch zu ergänzen und einsteigerfreundlicher zu machen. Außerdem ist ein weiteres Produkt in den Startlöchern, die Arducam Pivistation, die ein mehr in sich geschlossenes System mit verschiedenen Kameras und Objektiven bieten wird, das eher den Normalanwender bedient. —*caw*

Hersteller	Arducam
URL	heise.de/s/rLL37
Preis	100 US-$

IMPRESSUM

Redaktion

Make: Magazin
Postfach 61 04 07, 30604 Hannover
Karl-Wiechert-Allee 10, 30625 Hannover
Telefon: 05 11/53 52-300
Telefax: 05 11/53 52-417
Internet: www.make-magazin.de

Leserbriefe und Fragen zum Heft: info@make-magazin.de
Die E-Mail-Adressen der Redakteure haben die Form xx@make-magazin.de oder xxx@make-magazin.de. Setzen Sie statt „xx“ oder „xxx“ bitte das Redakteurs-Kürzel ein. Die Kürzel finden Sie am Ende der Artikel und hier im Impressum.

Chefredakteur: Daniel Bachfeld (dab)
(verantwortlich für den Textteil)

Stellv. Chefredakteur: Peter König (pek)

Redaktion: Heinz Behling (hgb), Johannes Börnsen (jom), Ákos Fodor (akf), Daniel Schwabe (das), Dunia Selman (dus, Social Media), Carsten Wartmann (caw)

Mitarbeiter dieser Ausgabe: Beetlebum, Bernd Heisterkamp, Robert Kränzlein, Roland Marx, Gerd Michaelis, Kevin Noki, Ulrich Schmerold, Andreas Voß, Wolfgang Ziegler

Assistenz: Susanne Cölle (suc), Martin Triadan (mat)

Layout und Satz: Lisa Hemmerling, Steffi Martens, Nicole Wesche, Heise Medienwerk GmbH & Co. KG

Korrektorat: Dörte Bluhm, Lara Bögner, Marei Stade, Christiane Tümmler, Heise Medienwerk GmbH & Co. KG

Titel: Lisa Hemmerling, Nicole Wesche

Fotografie und Titelbild: Andreas Wodrich

Digitale Produktion: Melanie Becker, Thomas Kaltschmidt, Pascal Wissner

Hergestellt und produziert mit Xpublisher:
www.xpublisher.com

Verlag

Maker Media GmbH
Postfach 61 04 07, 30604 Hannover
Karl-Wiechert-Allee 10, 30625 Hannover
Telefon: 05 11/53 52-0
Telefax: 05 11/53 52-129
Internet: www.make-magazin.de

Herausgeber: Christian Heise, Ansgar Heise

Geschäftsführung: Ansgar Heise, Beate Gerold

Anzeigenleitung: Daniel Rohlfing (-844)
(verantwortlich für den Anzeigenteil),
mediadaten.heise.de/produkte/print/das-magazin-fuer-innovation

Leiter Vertrieb und Marketing: André Lux (-299)

Service Sonderdrucke: Julia Conrades (-156)

Druck: Dierichs Druck + Media GmbH & Co.KG, Frankfurter Str. 168, 34121 Kassel

Vertrieb Einzelverkauf:
DMV DER MEDIENVERTRIEB GmbH & Co. KG
Meßberg 1
20086 Hamburg
Telefon: +49 (0)40 3019 1800
Telefax: +49 (0)40 3019 1815
E-Mail: info@dermedienvertrieb.de
Internet: dermedienvertrieb.de

Einzelpreis: 13,50 €; Österreich 14,90 €; Schweiz 26.50 CHF; Benelux 15,90 €

Abonnement-Preise: Das Jahresabo (7 Ausgaben) kostet inkl. Versandkosten: Inland 80,50 €; Österreich 88,90 €; Schweiz 123.90 CHF; Europa 95,20 €; restl. Ausland 100,80 €

Das Make-Plus-Abonnement (inkl. Zugriff auf die App, Heise Magazine sowie das Make-Artikel-Archiv) kostet pro Jahr 6,30 € Aufpreis.

Make: Nächste Ausgabe erscheint am 26. Juli 2024

Abo-Service:

Bestellungen, Adressänderungen, Lieferprobleme usw.:
Maker Media GmbH
Leserservice
Postfach 24 69
49014 Osnabrück
E-Mail: leserservice@make-magazin.de
Telefon: 0541/80009-125
Telefax: 0541/80009-122

ISSN 2364-2548

Nachgefragt

Wir haben eine 2D-Pixelart-Lampe im Heft! Welche Form / Figur würdest du damit bauen?

Roland Marx
Donauwörth, sichert Deye-Wechselrichter ab S. 46 ab.

Ich würde gerne meine längst überfällige Büro-Status-Lampe realisieren: On Air, Komm rein, DO NOT DISTURB z.B. mit dem Blocker-Lemming, Emojis und Co.

Ulrich Schmerold
Graben, emuliert auf S. 8 mit Erlenmeyerkolben Oszi-Röhren.

Ich würde aus den 2D-Pixelart-Pixeln das Unendlichkeitszeichen bauen, auf dem dann in unendlich wiederkehrenden Folgen die Regenbogenfarben kreisen.

Kevin Noki
Bergheim, zeigt ab S. 66, wie man ein Macintosh-Gehäuse mit dem 3D-Drucker herstellt.

Ich würde daraus ein Retro-Apple-Logo bauen, das perfekt in den Raum passt, in dem meine Apple-Sammlung steht.

Andreas Voß
Schwerte, erklärt auf S. 118 die Funktion von Dreh-Encodern.

Ich würde mit der Lampe den Anführer der Gegner des Retrospiels Galaxian bauen. Dieses Spiel begleitet mich jetzt seit Jahrzehnten und ich bin einfach davon begeistert, dass es das immer noch gibt.

Pressefreiheit ist deine Freiheit.

Du hast die Wahl.

www.mvfp.de

MVFP
Medienverband der freien Presse

AMD Power
@ TUXEDO

TUXEDO
Polaris 17 - Gen5

17,3-Zoll-Linux-Gamer der gehobenen Mittelklasse mit hocheffizientem AMD-Ryzen-Prozessor und schneller NVIDIA-RTX-Grafik.

TUXEDO
Sirius 16 - Gen2

All-AMD-Gamingnotebook der gehobenen Mittelklasse mit hocheffizientem Ryzen 7 8845HS und schneller Radeon RX 7600M XT.

TUXEDO
Pulse 14 - Gen3

Ultra portable CPU-Workstation mit AMD Ryzen 7 7840HS, High-Res-3K-Display und 32 GB LPDDR5-6400-Hocheffizienz-Arbeitsspeicher.

tuxe.do/make0324

Linux kompatibel

Bis zu 5 Jahre Garantie

Sofort einsatzbereit

TUXEDO

Gefertigt in Deutschland

Deutscher Datenschutz

Deutscher Tech Support