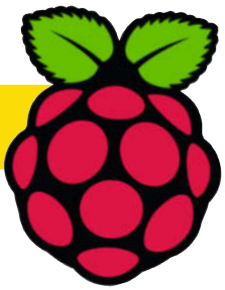
DEUTSCHLANDS GEFÄHRLICHSTES DIY-MAGAZIN

c't Make:

Make:

Doppel-Jubiläum:
10 Jahre
Raspberry Pi und Make



Pneumatik für Maker

- Stärker als Elektromotoren
- Aufbau und Funktion im Detail erklärt
- Praktische Anwendungen: Werkstücke einspannen, greifen, saugen

Für Einsteiger

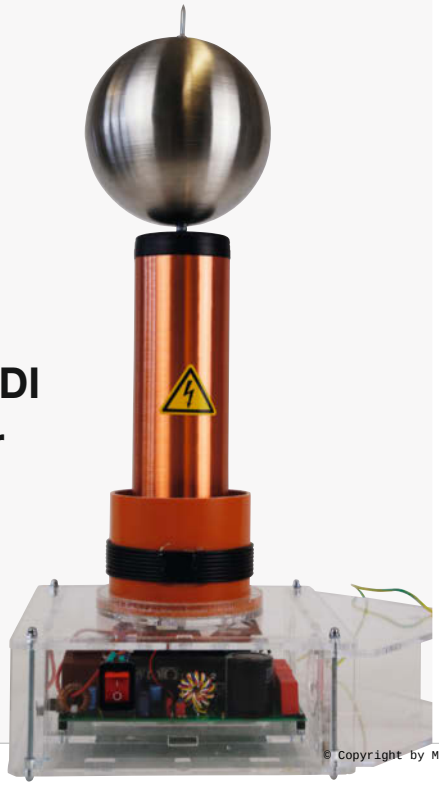
- Powerbox: Strom für unterwegs
- Futuristisch: animiertes Treppenlicht
- Bilderrahmen mit farbigem ePaper

Nachgelegt

- Logger für Taupunktlüfter
- CNC-Fräse mit WLAN
- Octoprinter zieht ins Smart Home

Tesla-spule

- Aufbau und Funktion
- Abspielen von Musik über MIDI
- Praxistipps für den Eigenbau



Energiemessung

- Smarte Stromzähler auslesen
- IR-Lesekopf im Eigenbau
- Mit Wallboxen koppelbar

2/22
7.4.2022
CH CHF 25.80
AT 14,20
Benelux 15,20
€ 12,90

www.make-magazin.de

Sammlerstück gesucht

Na, sind Sie auch von Anfang an dabei? Herzlichen Glückwunsch! Dann lesen Sie seit genau 10 Jahren die *Make* und deren Vorgängerin *c't Hacks*. Dann haben Sie sicher auch alle Ausgaben aufgehoben und die stehen fein säuberlich sortiert im Regal – oder? Gehen Sie mal dorthin und ziehen die *c't Hacks 4/12* hervor und schlagen Sie diese auf Seite 23 auf ….

Wie? Ausgerechnet diese Ausgabe fehlt Ihnen? Das ist jammerschade, denn diese Ausgabe mit dem roten Cover war ein wirklich legendäres Heft. Ob wir uns heute noch den damals gedruckten Artikel zur *Kernfusion im Kinderzimmer* trauen würden? Oder die Anleitung, wie man seinen Goldfischen mit ein paar einfachen DNA-Schnitten Bioluminiszenz implantiert, sodass sie im Dunkeln leuchten? Oder das *Reflow*-Löten im Toaster? Unvergessen auch die Enthüllung, dass der damals noch junge Raspberry Pi möglicherweise den *Eurovision Song Contest* gehackt hat (siehe Link unten).

Dummerweise haben wir das Heft selbst nicht mehr: Alle noch vorhandenen Exemplare im Verlag sind einem Wasserschaden zum Opfer gefallen und die einzige Backup-DVD mit den Druckdaten ist nicht mehr lesbar. Schade, dass es auch bei Ihnen fehlt, denn wir hätten es Ihnen gerne um den Preis eines lebenslangen *Make*-Frei-Abos abgekauft …

Lassen Sie sich nicht verladen: Die Wahrheit hinter der mysteriösen *Hacks-Ausgabe 4/12* ist viel prosaischer. Es hat dieses Heft nie gegeben. Die erste Ausgabe der *c't Hacks* erschien im Oktober 2011 und es war offen, ob und wann ein nächstes Heft erscheint. Als Jahr wurde allerdings 2012 aufgedruckt, damit die Kioske nicht schon zum Jahreswechsel das Magazin aussortieren. Für das erste *reguläre* Hacks-Jahr 2012 standen dann zwei Ausgaben im Plan, für das nächste vier – die neue Redaktion musste auch erst mal aufgebaut werden. Und so kam es, dass der Nummer 1/12 aus dem Jahr 2011 im Jahr 2012 nur die beiden Hefte 2/12 und 3/12 folgten und dann direkt die 1/13, mit der es dann vierteljährlich weiterging.

Eine prima Vorlage, um stets auf dieses mysteriöse Heft zu verweisen, wenn mal ein Artikel zu einem grundlegend spannenden Maker-Thema dringend vermisst wird. „Macht doch endlich mal was zu *Software-Defined Radio* auf dem Multimeter!" – „Alter Hut, hatten wir schon in der Hacks 4/12." „Wie geht *Energy Harvesting* unter Überlandleitungen?" – „Steht ausführlich in Ausgabe 4/12 beschrieben …" Wer diesen Trick durchschaut, gehört wirklich zum harten Kern der Leserschaft.

Gesichert ist jedoch eines: Mit diesem Heft halten Sie garantiert die 50. reguläre *Make*-Ausgabe in den Händen. Anfang 2015 erschien unser erstes Heft, auf dem nur noch *Make* als Titel stand. Im ersten Jahr gab es sechs Ausgaben, seitdem jeweils sieben. Und ganz egal, wann Sie zum Kreis unserer Leserinnen und Leser hinzugestoßen sind, Sie finden ab Seite 8 einen Rückblick auf disruptive Entwicklungen, Stolpersteine und Glücksgriffe, nicht nur von uns, sondern auch aus dem *Raspberry-Pi*-Projekt, das zufällig ebenso alt ist wie wir.

Wir wünschen viel Spaß beim Lesen – dieser Ausgabe und der nächsten fünfzig … oder in den kommenden zehn Jahren!

Peter König

▶ make-magazin.de/xjag

Sagen Sie uns Ihre Meinung!
mail@make-magazin.de

Inhalt

Pneumatik für Maker

Es müssen nicht immer Drähte und Motoren sein: In vielen Fällen kann der Einsatz pneumatischer Bauteile Vorteile bringen – wenn Dinge mit ordentlich Kraft und Geschwindigkeit bewegt, gegriffen oder eingespannt werden sollen. Wir haben uns an dieses Thema herangewagt und genau so viel an Grundlagen zusammengetragen, wie Sie für Ihr erstes eigenes Pneumatik-Projekt brauchen.

44 Pneumatik: Grundlagen für Maker

Energiemessung

Smarte Stromzähler lassen sich mit einer Fotodiode auslesen – das macht sich unser Projekt-Update zur Wallbox-Ladeautomatik zu Nutze, um den Einbau-Aufwand in möglichst engen Grenzen zu halten. Sie sparen sich damit nicht nur den teuren Ladestrom aus dem Netz, sondern auch die Installation eines zusätzlichen Energiezählers.

52 Photovoltaik an der E-Auto-Wallbox – reloaded

3 Editorial
6 Leserforum
8 Report: Doppelgeburtstag: 10 Jahre Make und Raspberry Pi
12 Projekt: Digitaler Bilderrahmen mit ePaper
18 Report: Musikalischer Tesla-Spulen-Bausatz
24 Werkstattberichte: Neues aus der Szene, Comic
26 Projekt: Der Rasenkabelfinder
34 Projekt: Smarte Beleuchtung für die Treppe
38 Projekt: Romménator
44 Pneumatik: Grundlagen für Maker
52 Projekt: Photovoltaik an der E-Auto-Wallbox – reloaded
58 Projekt: Bordnetz – Strom Ahoi!
62 Make Education: Smartphone-Roboter für den Unterricht
68 Community-Projekt: Schachspiel für Sehbeeinträchtigte
70 Community-Projekt: Ergonomische Split-Tastatur im Eigenbau

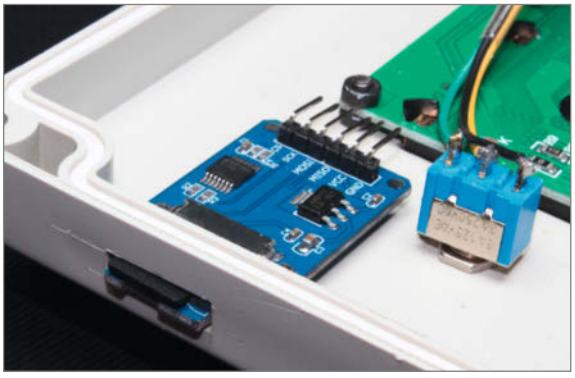
Nachgelegt

Maker-Vorhaben werden selten so richtig fertig – manchmal fällt einem erst im Nachhinein ein, was es noch zu verbessern oder zu ergänzen gäbe. Das geht uns genauso, und deshalb finden Sie hier einige Updates zu beliebten Projekten – zum Beispiel einen Datenlogger-Zusatz für unsere geniale Anti-Schimmel-Taupunktlüftung.

82 Logging-Funktion für das Taupunkt-Lüftungssystem

96 CNC-Fräse mit WLAN

102 Octoprinter zieht ins Smart Home



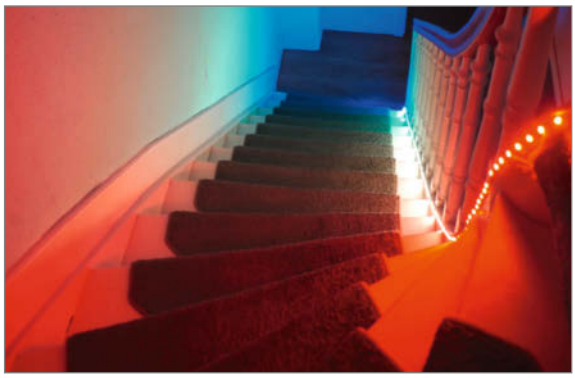
Für Einsteiger

Langweilige Basteleien für Anfänger waren gestern: Unsere eindrucksvolle, farbig animierte Treppenbeleuchtung kann jeder bauen, der schon einmal einen Arduino in der Hand gehalten hat. Auch der fortschrittliche ePaper-Bilderrahmen erfordert dank vorgefertigter Module nur einen überschaubaren Zeit- und Löt-Aufwand.

12 Bilderrahmen mit farbigem ePaper

34 Futuristisch: Animiertes Treppenlicht

58 Powerbox: Strom für unterwegs



72 Community-Projekt: Der Granulat-Extruder

74 Reingeschaut; Skat Champion

76 Projekt: Die animatronische Posteule, Teil 2

82 Projekt: Logging-Funktion für das Taupunkt-Lüftungssystem

86 Workshop: KI für den ESP32

94 Tipps & Tricks: Lego-Steine als Endmaße

96 Werkstatt: Netzwerk-Fräse

102 Werkstatt: 3D-Drucker im Smarthome

106 Workshop: Raspberry-Pi-Gehäuse mit FreeCAD, Teil 4

110 Kurzvorstellungen: 3D-Drucker, Mikrocontroller-Module, runde Displays, Wärmebildkamera, Smartwatch, Sneaker zum Selbernähen, Buch und Podcast

114 Impressum/Nachgefragt

Themen von der Titelseite sind rot gesetzt.

Teslaspule

Nichts für Angsthasen: Bevor der uns gelieferte Bausatz einer großen Tesla-Spule einwandfrei lief, hatten wir mehrere Opfer zu beklagen – dicke Leistungstransistoren, viele Sicherungen und die USB-Schnittstelle eines anwesenden Rechners. Doch das spektakuläre Resultat entschädigte für die Mühen: Funken machen Musik!

18 Musikalischer Tesla-Spulen-Bausatz

Leserforum

Falscher Autor

Die animatronische Posteule, Teil 1, Make 1/22, S. 8

In der Kurzinfo zum Artikel ist unter der Rubrik *„Mehr zum Thema“* für den *„Report: Animatronik“* in Make 4/20 versehentlich der falsche Autor angegeben – der Artikel stammt von Daniel Springwald.

Falsche Länge

Akkupacks selbst reparieren und bauen, Make 1/22, S. 94

Im Absatz „Geeignete Akkus“ ist die Länge von Standard-Zellen vom Typ 18650 fälschlicherweise mit „650mm“ angegeben. Richtig sind natürlich 65mm, sonst würden man solche Akkus ja kaum irgendwo unterbringen können.

Arduino in Gefahr

Der Taupunktlüfter, Make 1/22, S. 22

Mit Interesse habe ich Ihr Taupunkt-Lüftungssystem studiert und bin gerade dabei, es nachzubauen. Einzig schade sind die fehlenden oder falschen Angaben für die Verdrahtung des manuellen ON/OFF-Schalters. Wird der wirklich verdrahtet wie in Abbildung 13 zu sehen ist und in der Beschreibung darauf verwiesen wird, ist der Arduino kaputt, weil ein digitaler Output mit Masse bzw. +5V verbunden wird (Kurzschluss). Also bitte nicht nachmachen! Schließt man das Relais über einen Widerstand mit 470 Ohm bis 1 KOhm an Pin 6 an und verbindet den Schalter zwischen Relais und Widerstand wie angegeben mit GND/+5V, sollte das Relais in der unverbundenen Mittelstellung des Schalters nach wie vor im Automatikbetrieb funktionieren (hochohmiger Transistoreingang), aber der Arduino im „ON“- oder „OFF“-Betrieb nicht zerstört werden.

B. Hasubek

Unserem Arduino hat diese Beschaltung nicht geschadet, aber tatsächlich besteht dabei Zerstörungsgefahr. Mit einem Widerstand von 470 Ohm wie beschrieben funktioniert der Lüfter prima; im GitHub-Repository zum Projekt ist jetzt ein Schaltplan zu sehen, der sowohl den Modusschalter als auch den von Ihnen vorgeschlagenen Widerstand enthält.

Menschlicher Faktor

Toll, dass der Lüfter den Keller automatisch trockenlegt! Was aber, wenn das menschliche Element verhindert, dass man a) Entlüftungsbohrungen in Wand oder Acrylglasscheiben vornimmt und b) Probleme mit dem Konzept des „warme Luft im kalten Keller macht nix trocken!“ hat? ... Nun, man baut das gesamte Projekt fast genauso auf, verzichtet aber auf den Lüfterteil und schaltet stattdessen zwei farbige LEDs, die man in einen angebohrten Tennisball steckt. Und dann nimmt man sich den menschlichen Faktor zur Brust und klärt ihn über diese Lüftungsampel auf:

– Rot: Fenster und Kellertür bleiben ZU!
– Grün: Kannste aufmachen.

Funktioniert seit mindestens vier Jahren exzellent in meinem Haushalt und der Keller ist auch deutlich trockener geworden.

Denise Gutsche

Danke für die Rückmeldung! Interessanterweise haben uns diverse Leserinnen und Leser geschrieben, dass sie sich ein ähnliches System bereits selbst ausgedacht haben und schon länger betreiben. Das Problem (wie auch seine Lösung!) scheinen also verbreitet aufzutauchen …

Temperatur als Kriterium

Ich habe mir die Make gekauft, weil ich auf dem Titel das Projekt zur Kellertrocknung sah und schon seit Jahren was sehr ähnliches in meinem Keller installiert habe. Im Unterschied zum Artikel aber ohne Ventilator, weil mir der Einbau zu kompliziert erschien und durch den Ventilator im Winter der Keller stark runtergekühlt wird, auch wenn er steht. Aber auch bei mir vergleicht ein Mikrocontroller (ATMega, Eigenbau) den Taupunkt innen und außen, und über eine Ampel (rot-gelb-grün) meldet er an, dass es angebracht wäre, die Kellerfenster und Kellertüre zu öffnen, weil es draußen trockener als drinnen im Keller ist.

Ich habe aber noch ein zweites Kriterium für die Ampel drin: die Temperatur. Ich schalte nur auf grün, wenn es draußen wärmer und (absolut) trockener ist als drinnen, weil ich mir den Keller nicht noch zusätzlich abkühlen will. Ignoriert man die Temperatur, dann läuft der Ventilator im Winter quasi dauernd, und (die abgeführte feuchte Luft muss ja von außen ersetzt werden) kühlt das Haus ab, entweder direkt den Keller oder auch das Erdgeschoss.

Joachim Stegmaier

Im Code zu dem Make-Projekt werden für die Temperatur in den Zeilen 27 und 28 Minimalwerte definiert. Sinkt die Temperatur drinnen oder draußen unter die dort festgelegten Schwellwerte, läuft der Lüfter nicht, insofern kann man hier durchaus eine Untergrenze für die Abkühlung vorgeben. Eine Abfrage, ob es draußen wärmer ist als drinnen, lässt sich leicht hinzufügen, falls gewünscht.

Kontakt zur Redaktion

Leserbriefe bitte an:

heise.de/make/kontakt/

Wir behalten uns vor, Zuschriften unter Umständen ohne weitere Nachfrage zu veröffentlichen; wenn Sie das nicht möchten, weisen Sie uns bitte in Ihrer Mail darauf hin.

Sie haben auch die Möglichkeit, in unseren Foren online über Themen und Artikel zu diskutieren:

www.make-magazin/forum

www.facebook.com/MakeMagazinDE

www.twitter.com/MakeMagazinDE

instagram.com/MakeMagazinDE

pinterest.com/MakeMagazinDE

youtube.com/MakeMagazinDE

Korrekturen

Manchmal unterläuft uns ein Fehler, der dringend korrigiert gehört. Solche Informationen drucken wir weiterhin auf den Leserbriefseiten im Heft, aber seit Ausgabe 1/17 finden Sie alle Ergänzungen und Berichtigungen zu einzelnen Heft-Artikeln auch zusätzlich über den Link in der Kurzinfo am Anfang des jeweiligen Artikels.

Falsches MP3-Modul?

Carl: Eine stabile DIY-Musicbox für Kinder, Make 4/21, S. 16

Im Artikel zum MP3-Player mit Arduino steht bei *Material Elektronik* unter anderem *DFPlayer Mini mit AA20HFJ648-94 Chip*. In Ihrer Einkaufliste unter dem Kurzinfo-Link verlinken Sie jedoch auf ein MP3-Modul mit einem inkompatiblen Chip (GD3200B). Leider ist uns das erst jetzt nach unzähligen Stunden aufgefallen!

Nico

Das tut uns leid, jedoch haben wir mit einem Modul exakt aus dieser Quelle den Player im Vorfeld in der Redaktion nachgebaut und getestet. Vielleicht sind dem Händler da einige abweichende Module dazwischengeraten. Unser Autor hat sich mit dem Problem inzwischen beschäftigt und erläutert genaueres dazu im GitHub-Repository zum Projekt (siehe Link). Er arbeitet auch an einer Code-Version, die mit den GD3200B-Chips mit ein paar Tricks klarkommt.

▶ **https://github.com/jandelgado/carl**

Noch schicker

DIY-Gehäuse schnell gebaut, Make 5/21, S.112

Zwei spontane Verbesserungsvorschläge:

– Wenn man die Seitenwände an den Deckel leimt und die Seitenwände von unten durch den Boden verschraubt, fallen die hässlichen Schrauben auf dem Deckel weg.

– Wenn man in Boden, Seitenwände und Deckel mit der Kreissäge eine Nut schneidet, kann man vor den Aluwinkel eine Plexiglasscheibe mit entsprechenden Bohrungen für Buchsen und Knöpfe setzen, die die gedruckte Frontbeschriftung schützt. Der Aluwinkel dient dann nur als Montagewinkel; die Muttern, die Buchsen und Schalter halten, werden durch das Plexiglas verdeckt. Das Fotopapier mit der gedruckten Beschriftung kann man mit Sprühkleber von hinten an der Plexiglasplatte ankleben. Besonders professionell wirken Kontroll-LEDs, die von hinten an die Plexiglasscheibe, mit Sekundenkleber in eine kleine Sackbohrung geklebt sind. Man kann auch die komplette Rückseite der Frontbeschriftung schwarz lackieren, Aussparungen an der Beschriftung lassen und diese von hinten beleuchten – das erleichtert eine Bedienung der Geräte im Dunkeln.

Martin Karlein

Doppelgeburtstag

Once upon two times: 10 Jahre Make und Raspberry Pi. Wir geben einen Rückblick auf disruptive Entwicklungen, Stolpersteine und Glücksgriffe.

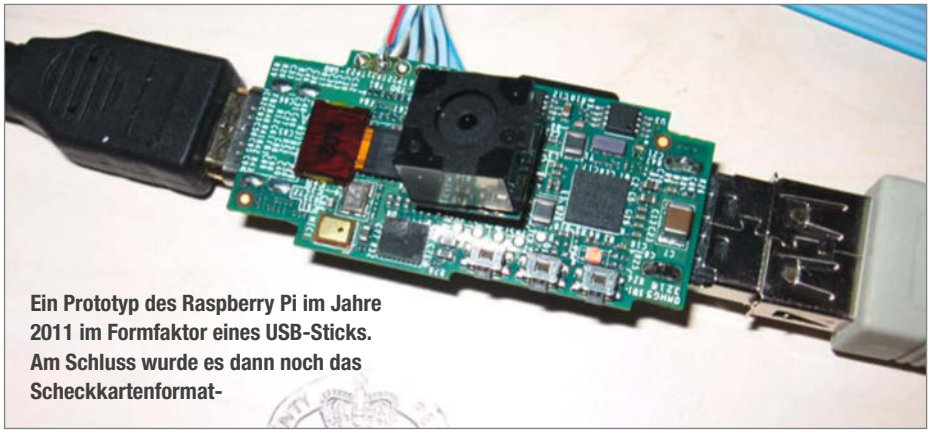
Alles zum Artikel im Web unter make-magazin.de/xchv

von Daniel Bachfeld

Sowohl der *Raspberry Pi* als auch die *Make* entstanden 2011 aus der gleichen Idee: Menschen wieder für den kreativen Umgang mit Technik zu begeistern, inklusive selber programmieren, löten, tüfteln und allem drum und dran. Die *Raspberry Pi Foundation* nahm sich den *BBC Micro* aus den 1980ern zum Vorbild, um einen Miniaturrechner für 25 Dollar zu bauen, der „finanziell benachteiligten" Schulkindern den Umgang mit Computern ermöglichen sollte. Erklärtes Ziel: Mehr Ingenieur-Studenten an die Unis.

Die c't ließ sich unterdessen vom weltweiten *Maker Movement* mitreißen, angetrieben durch die ersten, leidlich funktionierenden 3D-Drucker, von einsteigerfreundlichen *Arduinos* und überall entstehenden *Makerspaces*. In der Redaktion flammten wohlige Erinnerungen an die alten Tage, frühere Einplatinencomputer, *c't-Lab* und den *c't-Bot* wieder auf. In der Aufbruchstimmung starteten wir den Leser-Wettbewerb „Mach flott den Schrott" (MfdS), der sich um das Re- und Upcycling ausrangierter Elektronik-Komponenten zu neuen Projekten drehte.

Und dann entspann sich rund um alte c't-Projekte, MfdS, 3D-Drucker und Arduinos der Gedanke, daraus ein Sonderheft und auf *heise online* einen eigenen Channel zusammenzurühren: *c't Hardware Hacks*. Erklärtes Ziel: Vorhandene Leser zu inspirieren, getreu unserem Motto „Kreativ basteln mit Technik" mal wieder was selber zu machen – und vielleicht eine ganz neue Zielgruppe für sich zu gewinnen. Um es vorweg zu nehmen: Sowohl der Pi als auch c't Hardware Hacks wurden Erfolgsgeschichten, zumindest in ihren Nischen.



Ein Prototyp des Raspberry Pi im Jahre 2011 im Formfaktor eines USB-Sticks. Am Schluss wurde es dann noch das Scheckkartenformat-

Raspberry Pi Foundation

2012

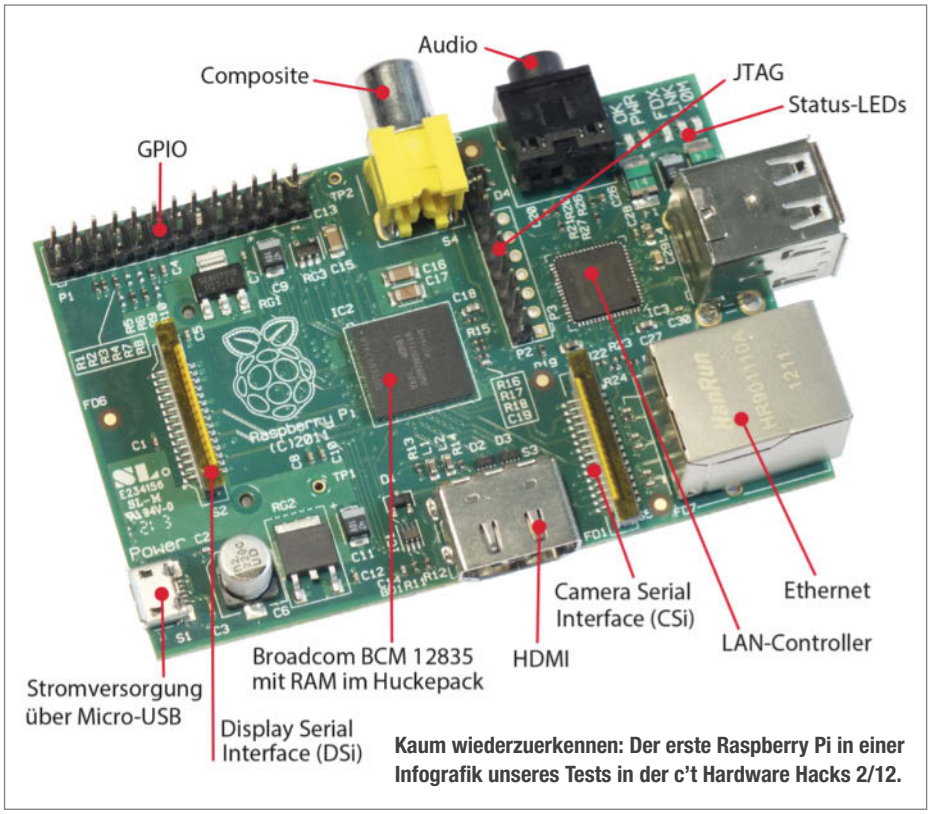
Wenn doch aber alles schon 2011 losging, warum feiern wir dann die Geburtstage erst 2012? Weil 2011 quasi erstmal nur die Prototypen vorgestellt wurden. „In Serie" gingen der Pi und das kurz „Hacks" genannte Magazin eigentlich erst 2012.

Bereits der Verkaufsstart des Raspberry Pi war denkwürdig: Er begann am 29. Februar 2012, also an einem Schalttag, und war eigentlich auch gleich wieder vorbei. Am Morgen brach unter dem Ansturm williger Käufer zunächst erst das Blog der Raspberry Pi Foundation zusammen und nach der offiziellen Lieferbarkeitsankündigung auch die Webseiten der beiden britischen und einzigen Pi-Distributoren *Farnell* und *RS Components*.

Insgesamt standen 10.000 Pi zur Verfügung, allerdings als B-Version für 35 Dollar mit 2 × USB und Ethernet. Die 25-Euro-Version A ohne Ethernet sollte später folgen. Trotz 10 Dollar mehr war bei Farnell die zugeteilte Liefermenge schon nach rund 20 Minuten ausverkauft, RS Components hatte gar nicht erst angefangen, den Pi anzubieten. Interessenten mussten sich dort auf eine Benachrichtigungsliste eintragen und auf weitere Informationen warten. Aktuell sieht es mit dem Raspberry Pi 3 und 4 ähnlich aus: Wegen der Halbleiterkrise gibt es monatelange Wartezeiten.

Da sowohl Farnell als auch RS Components zumindest in Deutschland und Österreich anfangs zunächst keine Privatleute belieferten, sondern nur Firmenkunden mit Gewerbe-

Hinweis zum Einsatz des Gehirns. Mittlerweile liefern wir sicherheitshalber etwas detailliertere Hinweise wie „Achtung 230V, bitte nicht reinfassen"



Kaum wiederzuerkennen: Der erste Raspberry Pi in einer Infografik unseres Tests in der c't Hardware Hacks 2/12.

Kaum zu glauben, die ersten Projekte mit dem Raspberry Pi erschienen erst in den Ausgaben 3/13 und 4/13. Hier der Druckuck, der Tweets per Thermoprinter ausgibt.

Eines der beliebtesten und häufig nachgebauten Projekte: Der Teehase. Sein Ohr tunkt in den Beutel Arduino-gesteuert ins Wasser.

Im Make-Labor trainiert Deutschlands gefährlichste DIY-Redaktion zwischen Tesla-Spulen, Aceton-Dämpfen und Heißklebepistolen.

schein, gab es einige Verwirrung, wie man als privater Maker an ein Pi gelangen konnte. RS Components machte für den Raspberry Pi jedoch eine Ausnahme, aufgrund der großen Nachfrage sollte jeder Interessent aber nur einen Pi bekommen.

Für die Hacks 2/12 nahm der Autor (dieses Artikels) den Pi 1 B unter die Lupe. Man bedenke: Der Pi hatte nur einen ARM-Kern mit schlappen 700MHz und 256MByte RAM und war für Desktop-Anwendungen so gut wie unbrauchbar. XMBC (mittlerweile in Kodi aufgegangen) konnte über HDMI aber schon mal Filme abspielen. Das allgemeine Fazit damals: „Mit der verfügbaren Software hat der Raspberry derzeit noch Probleme. Man muss viel Vorarbeit erledigen, bis alles einigermaßen läuft. Von daher ist fraglich, ob Einsteiger oder Kinder damit klarkommen. Bislang dürften eher Alpha-Geeks mit fundierten Linuxkenntnissen mit der Plattform glücklich werden".

Zweckentfremdet

Was ursprünglich als Lockmittel für Schüler und Studenten für MINT-Themen gedacht war, schlug in die Maker-Community ein wie Vibranium in Wakanda. Von den bislang über 40 Millionen verkauften Exemplaren landete ein Großteil auf den Schreibtischen von Bastlern, die den scheckkartenkleinen Computer nicht zum Lernen einsetzten, sondern sich ihre Träume von mobilen, stromsparenden und leistungsfähigen Projekten erfüllten.

Am Anfang erwartete die Foundation, dass die meisten Anwender Computerspiele programmieren und sich nur am Rande mit den GPIOs beschäftigen würde. Deshalb zielte auch die softwaremäßige Grundausstattung des Pi mit Bibliotheken wie *Pygames* in diese Richtung. In der Praxis stellt sich aber heraus, dass viel mehr Anwender lieber Roboter mit dem Pi basteln. Offenbar ist das Bewegen von Pixeln auf einem Bildschirm weniger faszinierend als das Bewegen physischer Dinge, wie Eben Upton in einem Interview im *The-Verge*-Podcast „Decoder" konstatierte. Es sehe so aus, als hätte man eine Generation von Mechatronik-Ingenieuren inspiriert.

Nicht zu vergessen: Auch als Smart-Home-Zentrale, Druckserver, Adblocker, NAS, VPN, Konsolen-Emulator und Mediaplayer ist der Raspberry Pi weit verbreitet und hat eine Zielgruppe erreicht, dem das Kopieren einer fertigen Firmware auf eine SD-Karte und anschließendes Konfigurieren als Akt des Selbermachens völlig ausreicht. Sogar auf der Raumstation ISS kreist ein Pi in der Erdumlaufbahn.

Ökosystem

Zehn Jahre nach dem Pi-Start hat sich ein großes Ökosystem rund um die Hardware und Software des Pi entwickelt. Die Belegung

Freut sich schon 2018 auf die Maker Faire ISS: der deutsche Astronaut Matthias Maurer, hier mit Make-Autorin Kathrin Grannemann

seiner GPIO-Leiste ist mittlerweile Industriestandard, Erweiterungsboards mit allen erdenklichen Zusatzfunktionen, HATs genannt, machen zusammen mit freier Software den Bau selbst hochkomplexer Projekte zum Kinderspiel.

Andere Hersteller surfen auf der Pi-Welle mit und versuchen mit Alternativ-Boards wie *Pines*, *Odroids* und *Banana Pis* ein Stück vom Kuchen abzubekommen. Oft ist deren Hardware zwar leistungsfähiger, meist ist der Software-Support aber mangelhaft und nur von kurzer Dauer. Hier zeigt sich, dass die Raspberry Pi Foundation sich von einem Hardware- zu einem Softwarehersteller transformiert hat. Laut Eben Upton hat die Zahl der für die Foundation arbeitenden Softwareentwickler die der Hardwareentwickler weit überholt. Man sei mittlerweile eher eine Linux-Computing-Firma.

Seit dem *Pi Pico*, dem Mikrocontroller-Board auf Basis eines ARM-Cortex M0, ist man de facto nun sogar ein echter Chiphersteller, der andere Firmen mit seinem SoC beliefere. Auch wenn die Foundation derzeit gute Umsätze mit ihren Produkten macht, braucht man britischen Medienberichten zufolge weiteres Kapital zur Entwicklung neuer Produkte. Ein Börsengang 2022 sei denkbar.

Marktbegleiter

Auch die c't Hacks hat vom Boom des Pi profitiert, der in vielen im Magazin beschriebenen Projekte die zentrale Rolle spielte und weiterhin spielt. Glücklicherweise haben wir das Heft von Anfang thematisch viel breiter aufgestellt als andere, monothematische Pi-Magazine. So begleiteten wir neben Pi und Arduino die

Einführung vieler spannender Mikrocontroller (*ESP8266, ESP32, Micro:bit, Espruino*), gaben Orientierungshilfe bei 3D-Druckern, zeigten, wie man Drohnen baut und konzipierten diverse Selbstbaufräsen. Dabei haben es einige unserer eigenen Projekte durchaus zu Kultstatus geschafft.

Die „Sperrholzfräse" aus Make 1/14 , von c't-Urgestein Carsten Meyer (aka *Fräsen-Meyer*) konzipiert, wird seit etlichen Jahren in abgewandelter Form in Workshops nachgebaut. Aus Standardbauteilen von *Igus* konstruiert und mit einer eigenen, in unserem Shop erhältlichen Steuerung brachten wir Anfang 2017 den Bauvorschlag für die „MaXYposi"-Fräse ins Heft. Die Resonanz war überwältigend. Leider hatten wir die Support-Anfragen von Lesern unterschätzt und wurden quasi überrollt und konnten nur sehr verzögert reagieren. Sie können es sich denken: Auch die anschließende Leser-Kritik war (zurecht) überwältigend. Seitdem sind wir mir derart komplexen Projekten eher zurückhaltend.

Namenswechsel

Ende 2014 wechselten wir in vollem Galopp den Namen und benannten uns in *Make* um. Der Grund: 2013 holten wir erstmals das amerikanische Format der *Maker Faires* nach Hannover. Ein Tag im Sommer, eine Halle plus etwas Außengelände, erwartete Besucher 1500. Am Ende kamen fast 5000. Für uns ein Riesenerfolg, den wir 2014 mit zwei Tagen, zwei Hallen und 10.000 Besuchern sogar toppen konnten. Allerdings passten c't Hacks und Maker Faire aus unserer Sicht begrifflich nicht zusammen und so lizenzierten wir den Namen der US-Make. Zudem wechselten wir von vier Ausgaben auf sechs pro Jahr, 2017 dann auf sieben.

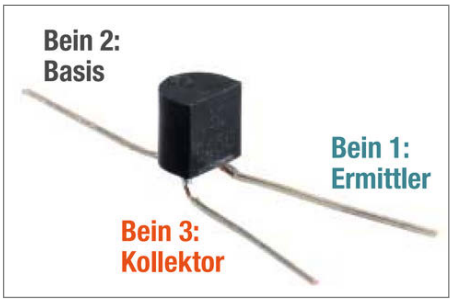
Zum Namenswechsel erreichten uns seinerzeit diverse Leserbriefe, die uns nun als Marionetten amerikanischer Imperialisten sahen. Warum man denn nicht beim alten Namen ohne Anglizismen hätte bleiben können und ob uns unsere Inhalte nun diktiert würden. Inwiefern *c't Hardware Hacks* (*c't* steht eigentlich für *computing today*) nun deutscher als *Make* sein soll, haben wir ehrlich gesagt nie ganz verstanden. Das c't prangt zwar immer noch ganz klein vor der Make auf dem Cover, ist aber nur ein zarter Hinweis für den Kiosk-Besitzer, in welche Abteilung er die Make sortieren sollte. Und zum Inhalt: Eigentlich ist es mittlerweile eher anders herum – die US-Kollegen übernehmen gerne und erfurchtsvoll „german engineering article" unserer Autoren in ihr Heft.

Mitunter tun sich einige mit dem Begriff *Maker* immer noch schwer, insbesondere was das vermeintlich zu niedrige Wissensniveau angeht. Projekte von Makern seien, so liest man häufig in den Foren auf heise online, nicht fachgerecht, gefährlich, sinnlos, oberflächlich und wenig verstanden. *Ja, nee, is klar,* denken wir uns dann immer. Weil man bei Hobby-Projekten natürlich immer die aktuellen VDE-Richtlinien einhält, jedes Mitglied einer Softwareklasse mit Vornamen kennt und mit dem Analog-Oszilloskop (nur von *Tektronix* und bitte nicht digital!) jeden Pegel auf Korrektheit geprüft hat. Manche scheinen zudem die Make mit dem vor knapp 25 Jahren eingestellten, eher an professionelle Leser gerichteten Heise-Magazin *Elrad* zu verwechseln. Wünsche, wir sollen uns an dessen Ingenieurs-Niveau orientieren, ehren uns. Aber dies wollen und können wir nicht erfüllen, weil das nicht unser Ziel ist. Dafür gibt es andere Magazine.

Dennoch hat sich in den letzten 10 Jahren der Begriff *Maker* ins Positive gewandelt. Maker stehen mittlerweile für Inspiration, Kreativität, Technikliebe, Offenheit und Zielstrebigkeit. Und wir glauben, dass wir mit unserem Magazin und den Maker Faires einen Anteil am guten Ruf der Maker hatten und weiterhin haben. Wir schaffen es, Menschen wieder für Technik-Themen und das Selbermachen zu begeistern. Nicht selten berichten uns Aussteller auf Maker Faires, wie sie im Jahr zuvor noch als Besucher fasziniert von den vielen Projekten und dem leichten Einstieg gewesen seien und gedacht hätten, „das kann ich auch". Genau! Wie Sie!

In 10 Jahren ist die Make zu einer wachsenden Marke mit vielen Ausprägungen wie Sonderheften, Specials, Veranstaltungen, Webinaren und Bausätzen geworden. Und wir wollen uns weiter entwickeln und digitaler werden und unser Angebot um Anleitungsvideos, Podcasts und eigene Maker-Komponenten ergänzen. Wie Jubilare so sagen: Auf weitere 10 Jahre. —*dab*

Das gab Ärger: Der Baumwoll-Lappen als Lieblingswerkzeug erntete 2014 viel Kritik. Eigentlich geht es in der Rubrik aber nicht um die tollsten Werkzeuge, sondern die emotionalste Bindung.



Epic fail in Heft 1/15: Kommissar Ermittler statt Transistor-Emitter

Maker im Weltall: Nicht nur der Pi war hoch unterwegs, auch unser Maskottchen Makey schwebte zur Maker Faire Hannover 2018 (fast) im Orbit.

Digitaler Bilderrahmen mit ePaper

Wir bauen einen Farbfotorahmen, der tagsüber jede Stunde das Bild wechselt, Speicherplatz für zehntausende Urlaubsbilder hat und dabei eine Batterielaufzeit von mehreren Monaten aufweist.

von Florian Sommer

Florian Sommer

In vielen Urlauben sind bei mir als Hobbyfotografen über die Jahre zigtausende Fotos zusammengekommen. Einzelne, tolle Bilder habe ich ausgedruckt, aber grob 98 Prozent fristen ein betrübliches Dasein im Datengrab Festplatte. Allerdings stecken auch in den nicht so gut gelungenen Fotos unschätzbare Erinnerungen an leckeres Essen oder lustige Zwischenfälle. Es wäre toll, an diese Momente ab und zu erinnert zu werden. Perfekt wäre ein digitaler Bilderrahmen. Die gibt es inzwischen wie Sand am Meer – groß, klein, hochauflösend und mit Onlinezugang. Eines jedoch eint die bisher am Markt verfügbaren Produkte: Sie fressen sehr viel Strom. Ein dauerhaft betriebener Rahmen kann bei 20 Watt Leistungsaufnahme locker 50 Euro Stromkosten pro Jahr verursachen. Daher habe ich ein stromsparendes Exemplar selbst gebaut.

Grundlage des Projekts ist ein *ePaper*-Display. Auch als *eInk* bekannt, sind sie seit vielen Jahren in eBook-Readern im Einsatz. Der Clou: Die letzte Darstellung bleibt erhalten, wenn das Display vom Strom getrennt wurde. Seit Mitte des Jahres 2020 gibt es neben den bekannten Graustufen-Displays auch ePaper, die Farben darstellen können: schwarz, weiß, grün, blau, rot, gelb und orange. Dabei handelt es sich um diskrete Farben, denn jeder Pixel kann genau eine der Farben annehmen. Abstufungen gibt es nicht. Die Displays haben eine Auflösung von maximal 600 × 448 Pixeln und kosten um die 50 Euro. Das klingt etwas happig, ist aber nach einem Jahr wegen der geringeren Energiekosten wieder eingespart. Dafür hat man ein Projekt, das viel Freude bereitet und welches es bis jetzt noch nirgends zu kaufen gibt.

Um schöne Darstellungen von Urlaubsfotos für das Display zu generieren, sind noch ein paar Softwaretricks aus grauer Vorzeit nötig. Mit einem sparsamen Mikrocontrollerboard, dem neuen ESP32-S2 und einer LiPo-Batterie – die mittels USB-C Buchse komfortabel geladen wird – entsteht so ein Bilderrahmen, der tagsüber regelmäßig das angezeigte Foto wechselt, Speicherplatz für zehntausende Bilder hat und eine Batterielaufzeit von mehreren Monaten aufweist. Meinen Bilderrahmen habe ich im Juni fertiggestellt und geladen. Ende März ging dieser Artikel in den Druck. Geladen werden musste der Rahmen bis dahin nicht.

Kurzinfo

» **Zusammenbau des Bilderrahmens**
» **Algorithmen zur Bildkonvertierung**
» **ESP32-S2 einsetzen**

Checkliste

Zeitaufwand:
5 Stunden

Kosten:
110 Euro

Programmieren:
Grundkenntnisse Arduino/ESP und Python

Material

» **Waveshare 7-Farbdisplay** (5,65 inch, 600 × 448 Pixel, ohne PCB)
» **Unexpected Maker ESP32-S2 Feather Board**
» **Adafruit eInk Feather Friend** (32kB RAM)
» **microSD-Karte** (FAT32 formatiert)
» **LiPo-Batterie** (3,7 Volt, 2-Pin JST-Stecker)
» **1-poliges Kabel**
» **Bilderrahmen** (10cm × 15cm)
» **Passepartout**

Werkzeug

» **kleiner Schlitzschraubendreher**
» **Lötkolben und Lötzinn**
» **Seitenschneider**
» **Pinzette**
» **Raspberry Pi (oder Linuxrechner)**

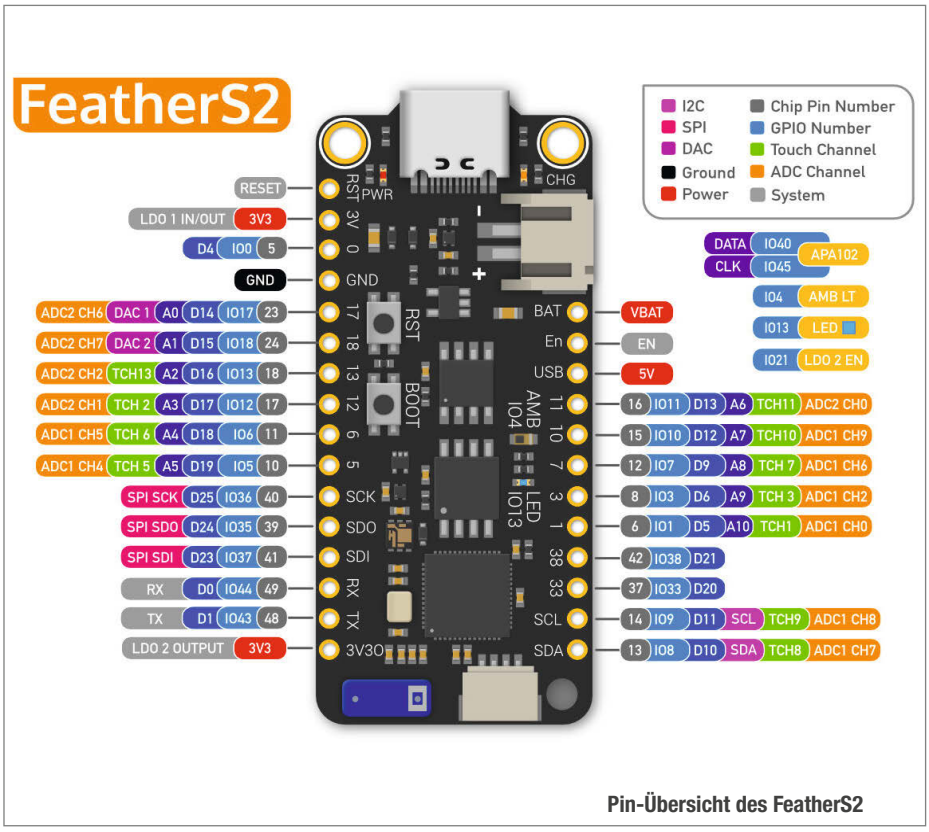
Mehr zum Thema

» Wasili Adamow, Patrick Beedgen, Marc Hassenzahl, Kirstin Kohler, Eva Lenz, Thies Schneider, Digitales Familientreffen, Make: IoT Special 2016, S. 104
» Lukas Scheffler, Digitaler Bilderrahmen mit Messenger-Funktion, Make 5/19, S. 32
» Sven Klomp, Bilderrahmen mit Überraschungseffekt, Make 2/21, S. 42

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

Bauteile

Aufgepasst: Das 7-Farbdisplay von *Waveshare* muss vorsichtig behandelt werden. Sowohl die Glasfläche wie auch das Flachband-Anschlusskabel sind sehr empfindlich. Zu beziehen war es lange nur direkt bei Waveshare aus Asien, was circa einen Monat Lieferzeit bedeutet, aber inzwischen haben es verschiedene deutsche Elektronikdistributoren wie Elektor



Pin-Übersicht des FeatherS2

Unsere Bilder sollen am Ende nur mit diesen sieben Farben dargestellt werden.

Bild-Konvertierung

```
convert original.jpg -channel luminance -auto-level -modulate 120,200
-gravity center -resize 600x448^ -extent 600x448 -background white -dither
FloydSteinberg -define dither:diffusion-amount=75% -remap eink-7color.png
-depth 4 -type Palette BMP3:dithered.bmp
```

picture_display.py

```
import os
from convert_bmp import convert_bmp

def dither_picture(n, file):
    systemcall = f"convert '{file}' -channel luminance -auto-level
-modulate 120,200 -gravity center -resize 600x448^ -extent 600x448
-background white -dither FloydSteinberg -define dither:diffusion-
amount=75% -remap eink-7color.png -depth 4 -type Palette
BMP3:dithered/{n}.bmp"
    os.system(systemcall)
    convert_bmp(f"./dithered/{n}.bmp", f"./converted/{n}.bmp")

path = './'
files = [f for f in os.listdir(path+'original/') if
f.lower().endswith('jpg')]
if not os.path.isdir(path+'dithered/'):
    os.mkdir(path+'dithered/')
if not os.path.isdir(path+'converted/'):
    os.mkdir(path+'converted/')
for n,file in enumerate(files):
    dither_picture(n, path+'original/'+file)
```

convert_bmp.py

```
import numpy as np

def splitByte(b):
    lowerhalf = b & 15
    upperhalf = (b >> 4) & 15
    return [upperhalf, lowerhalf]

def convert_bmp(filename, outfile, headerlength=118):
    file = open(filename, 'rb')
    data = file.read()
    file.close()
    databytes = [b for b in data][headerlength:]
    origpicture = np.array([[hex(sb) for sb in splitByte(b)] for b in
databytes])
    newpicture = origpicture.copy()
    newpicture[origpicture == '0x6'] = '0x1'
    newpicture[origpicture == '0x1'] = '0x4'
    newpicture[origpicture == '0x5'] = '0x3'
    newpicture[origpicture == '0x3'] = '0x6'
    newpicture[origpicture == '0x4'] = '0x5'
    newpicture = [list(e) for e in list(newpicture)]
    newpicture = [int(f"{entry[0][-1]}{entry[1][-1]}", 16) for entry in
newpicture]
    header = list(data[:headerlength])
    newdata = header + newpicture
    file = open(outfile, 'wb')
    file.write(bytearray(newdata))
    file.close()
```

und Eckstein im Programm. Als Mikrocontroller nutze ich das neue ESP32-Board *FeatherS2* von *Unexpected Maker*, der auf möglichst niedrigen Stromverbrauch geachtet hat. Das Board ist mit 8MB PSRAM ausgestattet, das wir für die Bilddarstellung benötigen, und mit dem Feather-Format für Mikrocontroller von Adafruit kompatibel. Der FeatherS2 kann mit *MicroPython* oder C(++) über die Arduino-IDE programmiert werden. Ich habe letzteres gewählt, da die nötigen Libraries aktuell nur dort bereitstehen. Zu beziehen ist der FeatherS2 zum Beispiel bei berrybase.de, die Bezugsquellen finden Sie wie die Software und weitere Anleitungen über den Downloadlink in der Kurzinfo.

Um den Mikrocontroller mit dem Display zu verbinden brauchen wir noch ein Aufsteckboard, den *Adafruit eInk Feather Friend*. Er besitzt 32kB RAM zur Pufferung von Bilddaten, was für unser großes Farbdisplay aber längst nicht ausreichen würde. Neben dem Flachbandkabelanschluss ist ein Slot für microSD-Karten verbaut. Da der eInk Feather Friend mit Pinheadern geliefert wird, kann man ihn platzsparend auf den Mikrocontroller löten. Dabei sind die Boards durch Isolierband oder Tesafilm elektrisch zu isolieren, da sie ansonsten direkt aufeinander liegen. Weil die Boards immer nur wenige Sekunden pro Stunde aktiv sind und im *DeepSleep* kaum Wärme erzeugen, ist das problemlos möglich. Wer den Aufbau wieder trennbar halten und stecken möchte, sollte zusätzlich Pinheader-Sockets bestellen.

Mein Akku hat eine Kapazität von 2500mAh, wobei sicher weniger ginge. Dann sind nur noch ein Kabel, der Bilderrahmen und ein Passepartout aus weißer Pappe nötig, mit dem die Displayränder versteckt werden.

ImageMagick vorbereiten

Wer in den 90er Jahren bereits am Computer spielte, erinnert sich vielleicht an Point-and-Click-Adventures à la *Monkey Island*. Dabei wurden auf Maschinen, die gerade mal 16 oder 256 Farben beherrschten, anspruchsvolle Hintergrundmalereien dargestellt. Zum Einsatz kam eine Technik namens Dithering, genauer gesagt *Floyd-Steinberg-Dithering*, das die Farbtiefe eines Bildes verringert. Komplexe Farbbereiche, wie ein Gebiet in hellbeige, werden beim Dithering durch mehrere benachbarte Pixel approximiert. Ein weißer, ein gelber und ein oranger Pixel beieinander sehen aus ausreichender Entfernung beige aus. Diese Vollfarbbilder sind zwar nicht so brilliant wie ein LCD oder OLED-Display oder gar ein echtes Foto, dennoch können sie sehr schöne Erlebnisse abbilden.

Beim Dithern habe ich mich an eine Anleitung von Adafruit gehalten. Das Tool, mit dem ich aus normalen JPG-Bildern geditherte Bitmaps erstelle, heißt *ImageMagick*. Es wird

1 Vollfarbbild

2 gedithertes Bild

3 gedithertes Falschfarbenbild

über die Kommandozeile bzw. Eingabeaufforderung bedient und ist für Windows, macOS, iOS und Linux kostenlos verfügbar. Auf Windows wird es mit einem Installer installiert, bei macOS mit der beliebten Paketverwaltung *Homebrew* und auf dem Raspberry Pi oder Ubuntu über die Kommandozeile (die Tasten Strg+Alt+T):

```
sudo apt update
sudo apt install imagemagick
```

Um ImageMagick zu nutzen, ruft man unter Windows am besten die Suche über die Tasten Windows+R auf und sucht dort *cmd*, bzw. auf dem Mac das Terminal. Nötig ist außerdem ein Kalibrationsbild (*eink-7color.png*), das ImageMagick verrät, welche Farben es zum Dithern verwenden darf. Dieses finden Sie wie Anleitungen zur Kommandozeile über den Downloadlink in der Kurzinfo.

Beim Dithern erstellen wir ein Bitmap-Bild mit 4 Bits pro Pixel (`-depth`). Die Displayfarben dürfen dabei keine Abstufungen aufweisen und pro Pixel kann nur eine Farbe zur Zeit dargestellt werden. Um die sieben Farben des Displays zu kodieren, brauchen wir nur 3 Bits pro Pixel: 2^3 ergibt 8, also bereits einen Zustand mehr als nötig.

Bilder konvertieren

Das Kalibrationsbild muss zusammen mit dem zu bearbeitenden Bild (zum Beispiel *original.jpg*) in einem Ordner abgelegt werden, etwa *ImageMagick*. In allen Betriebssystemen kommt man über den Befehl `cd` in der Kommandozeile an diesen Ordner, etwa `cd C:\Benutzer\Benutzername\ImageMagick` unter Windows. Nun kann man das Bild mit ImageMagick mit dem Befehl im Listing „Bild-Konvertierung“ in ein gedithertes Bild *dithered.bmp* konvertieren. Unter Windows muss man `convert` durch `magick` ersetzen.

Mit `-remap` binden wir das Kalibrationsbild ein und geben mit `-dither` den Floyd-Steinberg-Algorithmus vor. Meine weiteren Einstellungen für Kontraste (`dither:diffusion-amount`) und Farbsättigung habe ich durch Ausprobieren optimiert und die Zielauflösung des Displays (`-resize`) gesetzt.

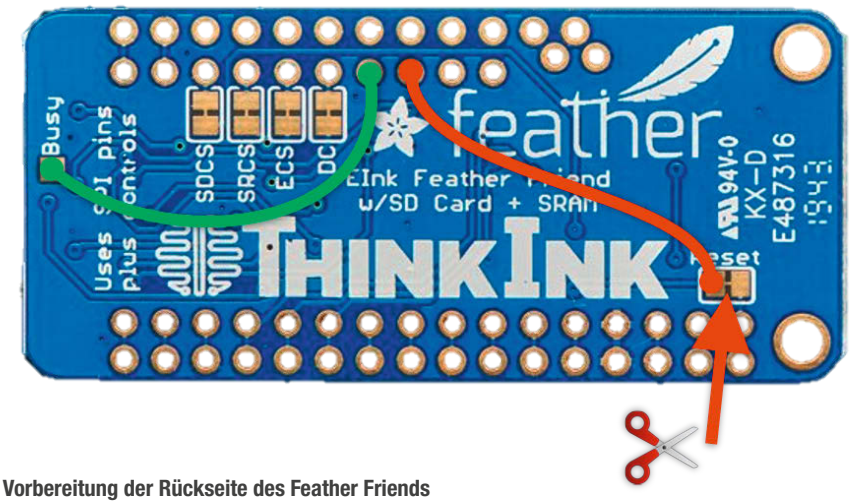
Um mehrere Bilder in einem Verzeichnis auf einmal zu konvertieren, habe ich das Python-Skript `picture_display.py` erstellt. Eine Windows-kompatible Version davon ist auf Github (siehe Downloadlink). Dazu ist eine Python-Installation mit dem Paket numpy nötig – in der beliebten Windows-Distribution Anaconda wird es direkt mitinstalliert. Das Skript wird dann in der Kommandozeile aufgerufen mit `python3 picture_display.py`. Dafür habe ich die zu konvertierenden Bilder in dem Unterordner `original` abgelegt (siehe Listing).

Nun werden alle Bilder in `original` bearbeitet und die Ergebnisse im Ordner `dithered` gespeichert, welcher gegebenenfalls angelegt wird. Im Code wird außerdem die Funktion `convert_bmp()` aus dem Skript `convert_bmp.py` importiert, das im gleichen Ordner liegen sollte. Da der Output von ImageMagick eine andere Bitreihenfolge für die Einzelfarben vorsieht als das Waveshare-Display erwartet, dreht `convert_bmp()` die Bits entsprechend um. Es tut dies für alle Bilder im `dithered`-Verzeichnis und speichert sie im Ordner `converted`.

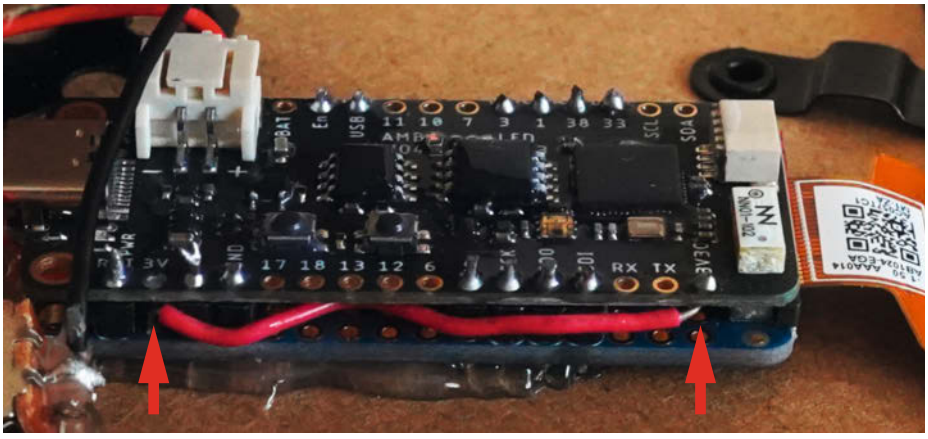
Das Ergebnis des Ditherns und Konvertierens ist im Bildervergleich zu sehen. Kontraste und Farbsättigung sind nach dem Dithern (Bild 2) absichtlich zu hoch, um die geringe Farbbrillanz des Displays zu kompensieren. Bild 3 zeigt die Konvertierung in ein Falschfarbenbild, damit das Display die Farben richtig zuordnet. Zuletzt ist das Bild im elektronischen Bilderrahmen zu sehen 4.

Sind alle Bilder konvertiert, kopieren wir sie auf die microSD-Karte. Mindestens 500 Bilder sollten es sein, sonst wiederholen sie sich sehr schnell. Da ein bearbeitetes Bild mit 132KB wirklich klein ist, passen sehr viele Bilder auf die microSD-Karte. Im Skript werden die Bilder

4 Bild auf ePaper-Display



Vorbereitung der Rückseite des Feather Friends



Anschluss der Peripheriespannung

gleich umbenannt. Der Dateiname wird einfach hochgezählt. Zum Schluss erstellen wir noch eine Textdatei *anzahl.txt* auf der microSD-Karte. Hier schreiben wir die Anzahl an Bildern als fünfstellige Zahl hinein. Sind es insgesamt zum Beispiel 732 Bilder auf der Karte, steht in *anzahl.txt* `00732`. Diese microSD-Karte kommt später in den Feather Friend.

Aufbau

Als Erstes bereiten wir den Adafruit Feather Friend vor und trennen die mit *Reset* markierten Ports auf der Rückseite mit einem Cuttermesser vorsichtig voneinander. Anschließend löten wir folgende Pins mit dünnen, isolierten Drähten zusammen: *Busy* mit dem 7. Pin von links in der zweiten Reihe (neben *DC*, grüne Linie im Bild) und den Pin rechts daneben mit *Reset* (rote Linie). Leider sind die Pins an den Stiftleisten nicht beschriftet. Beim Anlöten des Kabels die Trennung an *Reset* beachten und nicht wieder verbinden.

Auf die Oberseite des Feather Friends kleben wir isolierendes Band (zum Beispiel Tesafilm), um es vom FeatherS2 elektrisch zu trennen. Diesen legen wir mit der Oberseite nach oben und den großen Löchern übereinander auf den Feather Friend. Dann verlöten wir die Boards an den folgenden Pins des FeatherS2 (die Pins des Feather Friends sind unbeschriftet): *RST*, *0*, *GND*, *5*, *SCK*, *SDO*, *SDI*, *EN*, *USB*, *3*, *1*, *38*, *33*. Zum Verlöten nutze ich gekürzte Pinheader, die auch als Abstandshalter dienen. Alternativ kann der Aufbau mit verlöteten Pinheadern und -Sockets einfach gesteckt werden.

Von FeatherS2-Pin *3V3O* löten wir schließlich ein Kabel zum Feather Friend Pin, der unter dem *3V*-Pin des FeatherS2 liegt (siehe Bild). Über den *3V3O*-Pin versorgt der ESP seine Peripherie mit Spannung. Man kann ihn per Software an- und ausschalten und den Feather Friend somit komplett stromlos legen.

Jetzt schließen wir das Flachbandkabel des Displays über die Buchse des Feather Friends an. Allerdings ist der schwarze Plastik-Arretierhebel des Aufsteckboards sehr fragil. Am besten zieht man ihn vorsichtig mit einem kleinen Schlitzschraubenzieher hervor, steckt das Kabel ein und drückt den Arretierhebel wieder ein.

Wir kleben das Display dann auf der Vorderseite des Bilderrahmengrunds mit Teasafilm auf und bringen das Passepartout darüber an, ebenfalls mit Tesafilm. Das Flachbandkabel biegen wir vorsichtig auf die Rückseite um und kleben dort die Elektronikboards zum Beispiel mit Heißkleber auf.

Den Akku kleben wir ebenfalls mit Heißkleber auf die Rückseite. Dabei reicht ein kleiner Klebefleck, sonst wird der Wärmeeintrag zu groß. Anschließend stecken wir ihn über den JST-Anschluss an den FeatherS2 an. Nun kann der fertig präparierte Bilderrahmengrund in den Bilderrahmen gesetzt werden, wobei wieder auf das Flachbandkabel zu achten ist.

Die vorbereitete MicroSD-Karte stecken wir mit einer Pinzette ein und laden den Akku mit einem USB-C Kabel über den FeatherS2. Beim Laden leuchtet beim FeatherS2 eine magentafarbene LED dauerhaft. Ist der Akku voll, blinkt diese. Den Code spielen wir gleich über die Arduinoumgebung auf den Mikrocontroller auf. Wer möchte, kann den ESP zum Schluss noch mit Schutzlack versiegeln oder zwischen Batterie und FeatherS2 einen kleinen Schalter zum Ein- und Ausschalten des Displays setzen.

Vorbereiten der Arduino-Umgebung

In der Arduino-Umgebung muss zunächst die Unterstützung für den FeatherS2 installiert werden. Eine Anleitung dazu finden Sie online, siehe Link in der Kurzinfo. Dabei muss die Version 2 des Pakets von Espressif genutzt werden.

Unter *Werkzeuge* / *Board* / *ESP32-Arduino* wählen wir dann *UM FeatherS2* aus. Außerdem brauchen wir noch die Bibliothek *GxEPD2*, um das Display anzusteuern. Die suchen und installieren wir über *Sketch* / *Bibliothek einbinden* / *Bibliotheken verwalten*. Hierbei werden automatisch zwei weitere, benötigte Bibliotheken von Adafruit installiert (*Adafruit_BusIO* und *Adafruit_GFX_Library*).

Vor dem Aufspielen eines Sketches muss der FeatherS2 in den Boot-Modus versetzt werden: Wir schließen das USB-C Kabel an und halten die Boot-Taste auf dem Board gedrückt, während wir kurz die Reset-Taste betätigen.

Programm

Den Arduino-Sketch *epd_randomfoto_sd.ino* habe ich mit vielen Kommentaren versehen und nutze verschiedene Beispiele aus der *GxEPD2*-Bibliothek. Wer den Rahmen einfach nur nachbauen möchte, muss unter *Werkzeuge* / *Port* noch den passenden COM-Port auswählen und den Sketch hochladen. Für alle anderen erkläre ich die wichtigsten Punkte hier noch einmal.

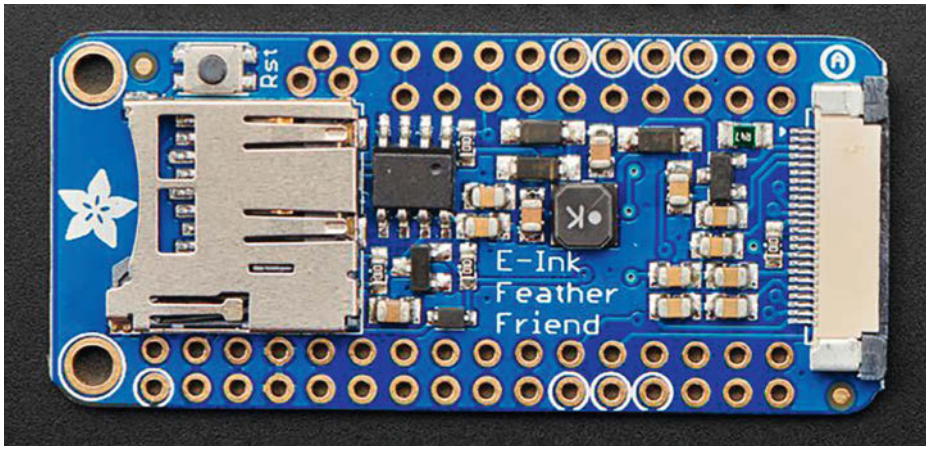
Nach dem Einbinden der benötigten Bibliotheken setzen wir eine Reihe an `#defines`, um das Display zu definieren. Auch für den FeatherS2 setzen wir spezifische `#defines`: `ledPin 13` und `LDO2 Pin 21` (Pin 3V3O). Wie oft das Bild auf dem Display geändert werden soll, genauer gesagt, wie lange der ESP zwischen den Bilderwechseln in den Schlafmodus wechselt, wird bei `TIME_TO_SLEEP` in Sekunden angegeben. Mit `RTC_DATA_ATTR` setzen wir dann in der Echtzeituhr einen persistenten Speicher für eine Zufallszahl. Dieser wird im batterieschonenden Tiefschlafmodus des Mikrocontrollers nicht gelöscht und kann beim Erwachen ausgelesen werden. Mit der Zufallszahl wird beim Aufwachen immer ein anderer *Seed*, also ein Anfangswert, für einen *Random Number Generator* gesetzt, damit die Bilder stets zufällig ausgewählt werden. Der ganze Code spielt sich übrigens in `void setup()` ab, um Strom zu sparen. Die Arduino-Methode `loop()` wird nie erreicht.

Den Umgebungslichtsensor des FeatherS2 lesen wir mit `if (analogRead(4) < 150)` aus. Wenn es zu dunkel ist, wird die Funktion `mysleep()` aufgerufen, die den Controller für die Länge von `TIME_TO_SLEEP` in den Schlaf schickt. Dabei ist er extrem sparsam. Somit wird nachts, wenn es eh zu dunkel zum Anschauen von Bildern ist, kein unnötiger Strom verbraucht. Danach konfigurieren wir `LedPin` und `LDO2 Pin` als Ausgänge und schalten sie ein. Die Display-Elektronik wird nun mit Spannung versorgt.

Die zuvor erstellte Datei *anzahl.txt* auf der SD-Karte lesen wir in den Zeilen von `myFile = SD.open("/anzahl.txt");` bis `myFile.close()`aus und hinterlegen den Wert in der Integer-Variablen `n`. Danach initialisieren wir den *Random Number Generator* mit der Variablen aus dem Speicher der Echtzeituhr `randomSeed(randomState);`. Es wird mit `long randNumber = random(1, n+1);` eine Zufallszahl zwischen 1 und der Anzahl an Bildern generiert und mit `randomState = random();` eine neue Zufallszahl abgespeichert. Dies könnte man sich sparen, wenn man die Bluetooth- oder WLAN-Funktionen des Microcontrollers als Entropiequelle für den Random Number Generator nutzt. Aus Stromspargründen habe ich diese aber deaktiviert.

Jetzt verwenden wir die ermittelte Zufallszahl `randNumber` für das Auswählen der Bilddatei auf der microSD-Karte. Um sie in den Arbeitsspeicher des ESP zu kopieren, benötigen wir ihre Größe (4Bit pro Pixel + Header): `size_t fileSize = 134518;`. Damit reservieren wir Arbeitsspeicher `unsigned char * downloaded_file = (unsigned char *) malloc(fileSize);` und kopieren danach die Daten von der Karte `myFile.read(downloaded_file, fileSize);`.

Mit `display.init(115200);` initialisieren wir das Display. Dabei wird automatisch eine SPI-Verbindung erzeugt, die jedoch die falschen Standard-Pins verwendet. Deswegen wird SPI nochmal beendet (`SPI.end();`) und mit den korrekten Pins für das FeatherS2-Board neu gestartet (`SPI.begin(36, 37, 35, 34);`). Anschließend wird eine Sekunde gewartet.

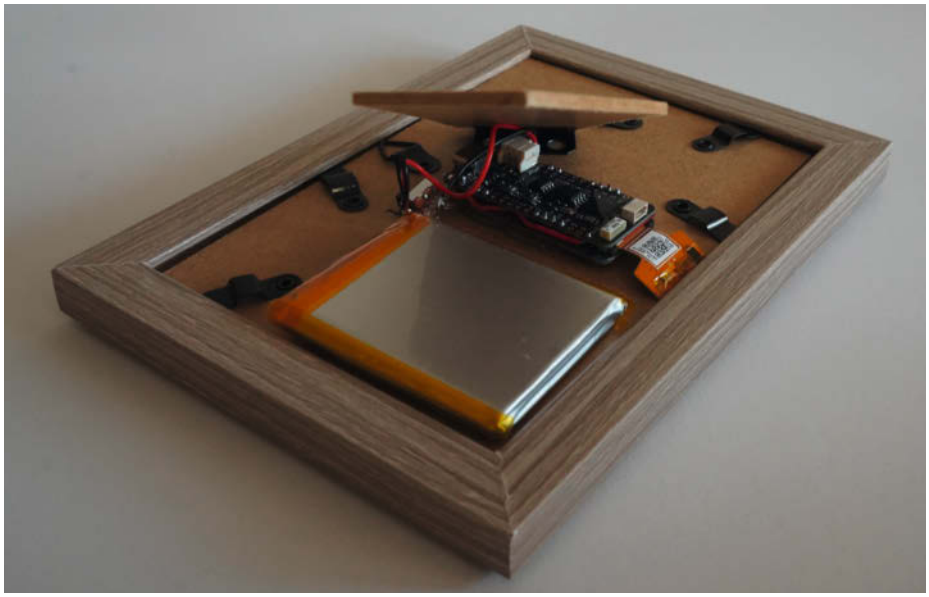


Vorsicht beim Einstecken des Flachbandkabels an der rechten Seite des Feather Friends

In den folgenden Zeilen lesen wir den Header der Bilddatei aus. Nötig sind hier nur `imageOffset`, `width` und `height`, die weiteren Angaben werden erst gebraucht, wenn man etwa ein Display mit anderer Auflösung oder anders geditherte Fotos nutzt. Auf dem Display vorhandene Darstellungen ersetzen wir mit `display.clearScreen();` durch Weiß. Dies dauert ein paar Sekunden. Nach einer Pause von einer weiteren Sekunde übermitteln wir das Bild ans Display – das dauert ebenfalls einige Sekunden. Nun leeren wir den Speicherpuffer, schalten das Display aus und rufen die Funktion `mysleep()` auf. Sie deaktiviert erst die Peripheriespannungsversorgung über den Pin LDO2, dann die LED. Schließlich wird ein Timer auf eine Stunde gestellt und der Microcontroller in den `deep_sleep` versetzt, was alle Elektronik bis auf die Real Time Clock und deren Memory ausschaltet. Nach einer Stunde wird der Chip reaktiviert und führt das Skript erneut aus, wobei `RTC_DATA_ATTR long randomState` den Wert besitzt, den wir zuvor hinterlegt haben. Es wird also ziemlich sicher ein neues Bild ausgewählt und dargestellt.

Ausblick

Die Kombination aus FeatherS2 und dem Adafruit eInk Feather Friend bietet viele Möglichkeiten. Es gibt neben dem vorgestellten Farbdisplay auch große eInk-Displays, die zwar nur Graustufen anzeigen können, dafür aber eine viel höhere Auflösung haben. Der FeatherS2 ist mit Bluetooth und WiFi außerdem sehr gut geeignet, um statt Bildern von der SD-Karte aus dem Internet per API dynamische Daten, zum Beispiel den Google Kalender, zu ziehen und diese auf dem Display darzustellen. *—hch*



Die fertige Rückseite des Bilderrahmens

Musikalischer Tesla-Spulen-Bausatz

Hochspannend, das ist wohl das Adjektiv, das eine Teslaspule am besten beschreibt. Und wenn sie auch noch Musik abspielen kann, hat sie sogar etwas künstlerisches. Wer sagt da nicht schnell: Haben wollen! Aber wie ist das mit dem Bauen bei solchen Spannungen? Ist das von jedem zu bewältigen? Wir haben ausprobiert, eine MIDI-fähige Teslaspule aus einem Bausatz zum musikalischen Leben zu erwecken und schildern hier unsere Erfahrungen.

von Heinz Behling

Gleich eines vorweg: Mit etwa 600 Euro war der Komplett-Bausatz *DRSSTC* von *Lessinger Elektronik* nicht gerade ein Taschengeld-Artikel. Das Verb „war" ist übrigens wörtlich zu nehmen, denn kurz vor Fertigstellung dieses Artikels wurde der Lessinger-Internetshop geschlossen. Der hier geschilderte Bausatz ist daher nicht mehr so ohne weiteres erhältlich. Über den Kurzinfo-Link finden Sie jedoch die Adresse des Herstellers. Vielleicht kann er ja auf Nachfrage doch noch das ein oder andere liefern. Außerdem gibt es inzwischen auch im fernöstlichen Online-Handel diverse Tesla-Spulen-Bausätze. Auch dafür habe ich einige Adressen beispielhaft bereitgestellt. Die hier geschilderten Erfahrungen dürften auch bei Verwendung anderer Bausätze interessant sein.

Der kostspielige Bausatz enthält schon die fertig gewickelte Sekundärspule, sodass man sich nicht mehr mit dem Wickeln von 1400 Windungen hauchdünnen Kupferdrahts quälen muss. Außerdem sind im Bausatz die Platinen und Bauteile für die Elektronik der Spule (1) und die des MIDI-Konverters enthalten (2). Der hat die Aufgabe, Daten von einem MIDI-fähigen Geräte (das kann ein Computer, ein Keyboard oder ähnliches sein) entgegen zu nehmen, daraus die Steuerimpulse für die Teslaspule zu formen und die an die Spulenelektronik zu senden. Dazu später noch mehr.

Kurzinfo

- **Teslaspule aus Bausatz aufbauen**
- **MIDI-Anschluss für Musikwiedergabe**
- **Inbetriebnahme und Fehlersuche**

Checkliste

Zeitaufwand:
8 Stunden

Kosten:
650 Euro

Material

- **Bausatz für Teslaspule Musical-Mini-DRSSTC**
- **Plexiglas für Spulen-Gehäuse** 5mm dick, ca. 0,5m²
- **Metallgehäuse für den MIDI-Interrupter** zum Beispiel Conrad Best.-Nr. 522945-YD
- **USB-MIDI-Kabel**
- **MIDI-Ausgabe-Software** zum Beispiel MIDI-OX
- **Kunststoff-Kleber**

Werkzeug

- **Lötkolben** 30W, feine Spitze
- **Seitenschneider**
- **Schraubendreher**
- **bei Selbstbau-Gehäuse: Lasercutter**
- **Bohrmaschine und Bohrer für Metallbearbeitung**
- **Oszilloskop**

Mehr zum Thema

- Ulrich Schmerold, Einfache Teslaspule, c't-Hacks 2/14, S. 48
- Dieter Hoffmann, Continous-Wave-Teslatrafo, c't-Hacks 2/14, S. 40
- Dieter Hoffmann, Tesla-Trafo mit Funkenstrecke, c't-Hacks 2/14, S. 58

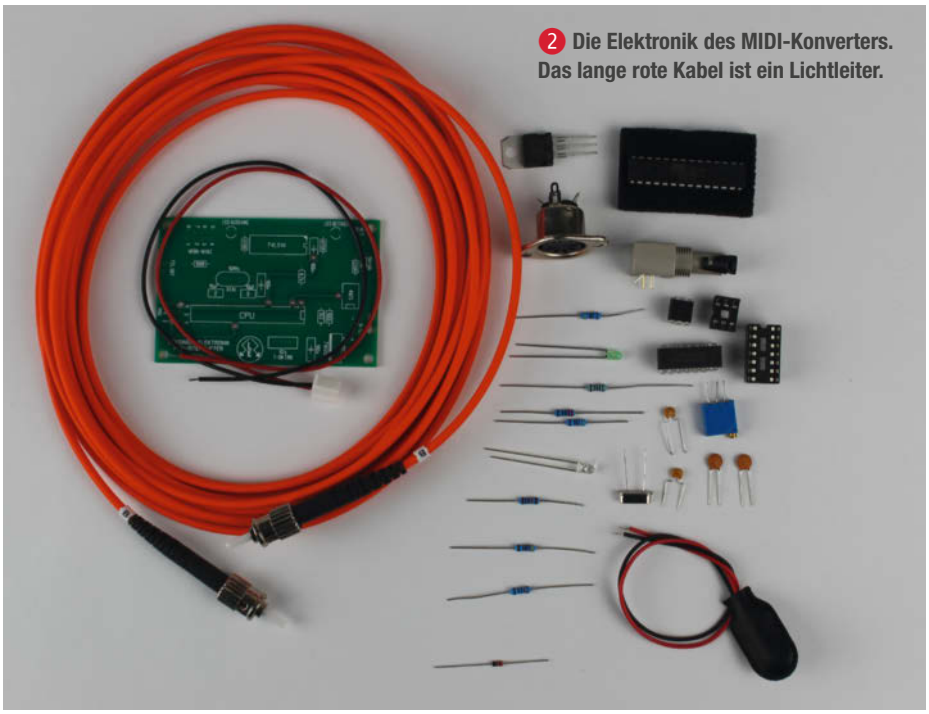
Alles zum Artikel im Web unter
make-magazin.de/xwy9

Funktionsprinzip Tesla-Spule

Eine Tesla-Spule ist im Prinzip ein Hochspannungs-Transformator. Allerdings ließen sich mit dem bei diesem Bausatz benutzten Wicklungsverhältnis von 8 zu 1400 aus der Eingangsspannung von 230V (ja, die Primärspule wird mit Netzspannung betrieben) nur etwa 40.000V gewinnen. Die Tesla-Spule schafft aber deutlich mehr und zwar dadurch, dass Primär- und Sekundärspulen auf ihrer Resonanzfrequenz betrieben werden. Die Sekundärspule zum Beispiel bildet nämlich zusammen mit der als Kondensator wirkenden Kugel am oberen Ende (siehe Titelbild des Artikels) einen Schwingkreis. Wenn man diesem im Rhythmus seiner Eigenfrequenz immer wieder Energieimpulse zuführt (durch kurzzeitiges Einschalten des Stroms in der Primärspule), schaukelt sich die Spannung im Schwingkreis auf und kann durchaus mehrere 100 Kilovolt erreichen.

Diese Frequenz wird bestimmt durch die Anzahl und den Durchmesser der Spulenwindungen, die Drahtstärke, der Kapazität der Kugel am Kopf der Spule und ähnlichem und ist schwierig zu berechnen. Eine Ansteuerung mit einer voreingestellten Frequenz führt daher meist nicht zum gewünschten Ergebnis. Die Steuerelektronik dieses Bausatzes ist so beschaffen, dass durch Rückkopplung die Steuerimpulse genau mit dieser Resonanzfre-

(1) Die Spulenelektronik ist gar nicht so umfangreich.



2 Die Elektronik des MIDI-Konverters. Das lange rote Kabel ist ein Lichtleiter.

quenz erfolgen. Eine Frequenzeinstellung ist nicht notwendig, das regelt die Elektronik automatisch.

Die Ausgangsspannung der Spule wird vor allem durch die anliegende Last begrenzt. Diese Last besteht bei der Tesla-Spule aus dem Entladungsfunken. Der ist nichts anderes als ein Strom, der durch die infolge der hohen Spannung ionisierte Luft fließt. Die Spannung wird dabei vor allem von der Länge dieser Funkenstrecke bestimmt: Ein langer Funken hat einen höheren elektrischen Widerstand und belastet daher die Stromquelle nur wenig. Die Spannung ist daher relativ hoch. Ein kürzerer Funken entspricht einem niedrigeren Widerstand. Durch den fließt ein höherer Strom, der die Quelle stärker belastet und somit zum Absinken der Spannung führt. Dazu mehr im Kapitel *Zweiter Start*.

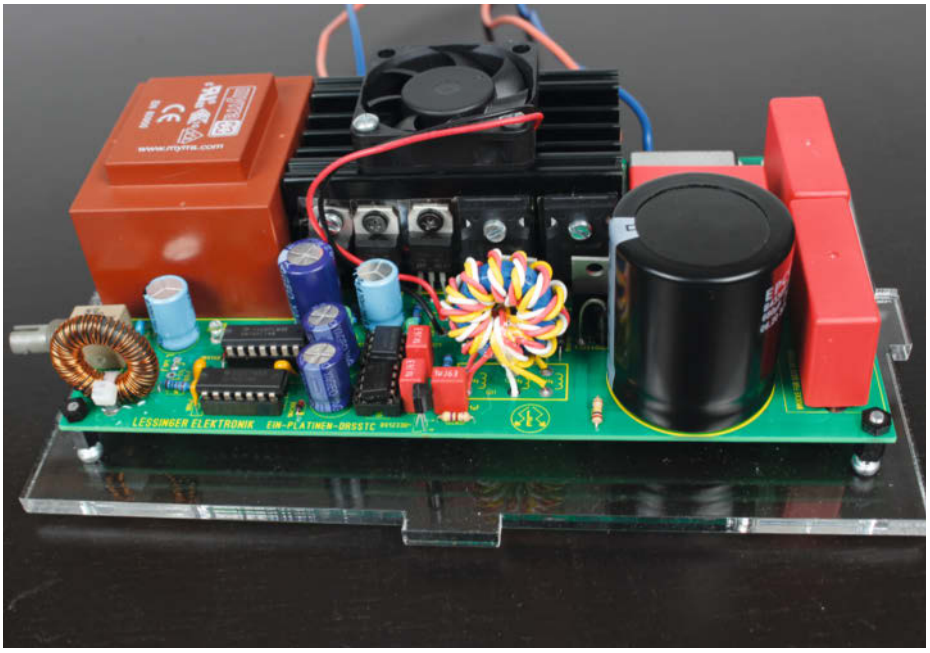
Der Bau

Dem Bausatz lag eine sehr ausführliche Anleitung bei, die selbst unerfahrenen Gelegenheitslötern den Nachbau ermöglicht. Lediglich die Angabe der Widerstand-Farbcodes fehlte, sodass ein Multimeter beim Finden der richtigen Bauteile helfen musste. Die beiden Platinen waren schnell bestückt. Bei der Spulen-Elektronik mussten aber doch zwei kleine Spulen auf Ringkernen selbst gewickelt werden. Entsprechende Drähte lagen in ausreichender Länge bei. Eine Spule hatte 60 Windungen aus 0,5mm-Kupferdraht. Durch deren Ringkern wurde später die Masseleitung der Sekundärspule geführt. Die induzierte Spannung dient als Rückkoppelung und sorgt dafür, dass die Tesla-Spule mit ihrer Resonanzfrequenz angesteuert wird.

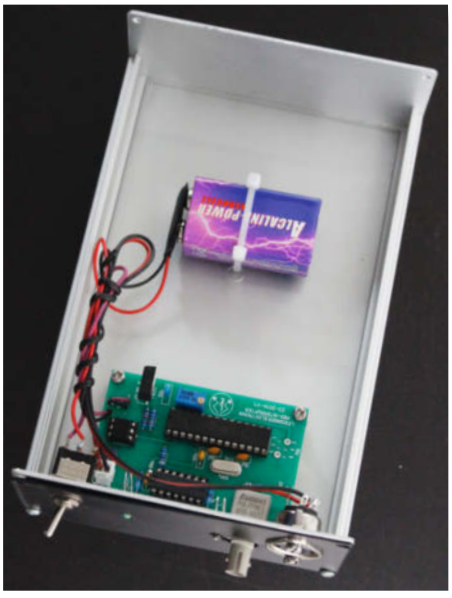
Die zweite Spule hatte zwölf Windungen dreier verdrillter, isolierter Drähte. Über diese Spule werden die Impulse an die beiden Schalttransistoren im Primärkreis der Spule verteilt. Das Wickeln beider Spulen war einfach und ohne zusätzliches Werkzeug möglich 3.

Nachdem die Platinen fertig bestückt waren, wurde handwerkliches Geschick nötig. Dem Bausatz lagen nämlich keine Gehäuse bei, weder für die Spule noch für den MIDI-Konverter. In der Anleitung fanden sich lediglich Empfehlungen für Fertiggehäuse. So wurde insbesondere für den Konverter ein geschlossenes Metallgehäuse verlangt, damit die von der Spule erzeugten Störimpulse nicht in die Konverter-Elektronik einstreuen und ungewollte Impulse zur Spule gesendet werden. Passende Metallgehäuse gibt es bei den üblichen Elektronik-Händlern. Man muss sie allerdings mechanisch bearbeiten, um Öffnungen für Schalter, LEDs und Buchsen zu schaffen 4.

Für die Spule hab ich ein Gehäuse aus Plexiglas entworfen (Schnittdateien über den

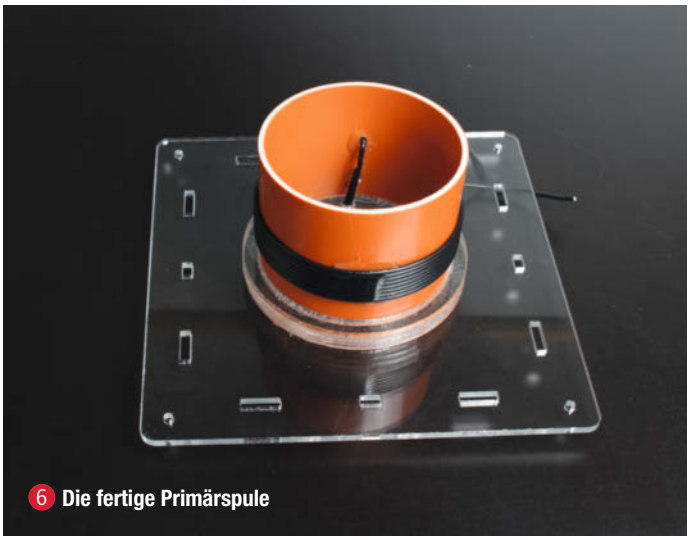


3 Die fertige Platine der Tesla-Spule mit den beiden selbstgewickelten Ringkern-Spulen



4 Der MIDI-Konverter im Metallgehäuse: Die Stromversorgung erfolgt mit einer 9V-Blockbatterie.

5 Beim Selbstbau-Plexiglas-Gehäuse wurde für gute Belüftung gesorgt. Oben die ringförmige Befestigung zum Einkleben der Primär- und Sekundärspulen.



6 Die fertige Primärspule

Kurzinfo-Link erreichbar), das per Lasercutter geschnitten wurde 5. In der Bauanleitung wurde aber auch ein Fertiggehäuse der Firma *Bopla* (Typ UM52011) empfohlen, das jedoch ebenfalls noch mit Öffnungen für Buchsen, Schalter sowie Spulenhalterung versehen werden müsste. Die Spule selbst müsste per Kabelbinder befestigt werden.

Schließlich musste noch eine dritte Spule selbstgewickelt werden: die Primärspule. Sie sollte aus acht eng aneinander liegenden Windungen Kupferlitze mit 1,5mm² Querschnitt bestehen und auf ein Stück HT-Rohr von etwa 10cm Länge gewickelt werden, das im Bausatz enthalten war. Für die beiden Enden der Wicklung habe ich direkt über bzw. unter der Spule je ein Loch gebohrt und die Leitungen nach innen geführt. Die Wicklung selbst wurde mit 2-Komponenten-Kleber am HT-Rohr fixiert. Das HT-Rohr wurde dann ebenfalls mit 2K-Kleber in der ringförmigen Halterung auf dem Plexiglas-Gehäuse befestigt 6.

Die Halterung oben auf dem Gehäuse hat innen eine Nut 7, in die die Sekundärspule mit dem roten Massekabel nach unten eingeklebt wurde. Zur Durchführung der Kabel beider Spulen waren im Gehäusedeckel passende Bohrungen vorgesehen.

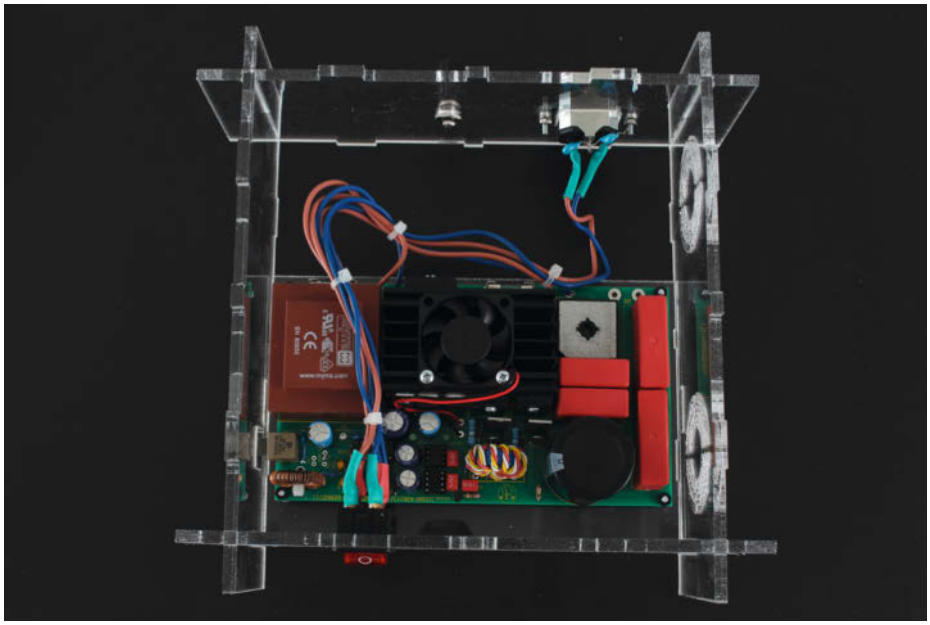
Schließlich wurden ins Gehäuse noch ein Kippschalter an der Vorderseite 8 sowie eine Kaltgeräte-Buchse und ein Erdungsanschluss eingebaut. Dieser Anschluss wird intern mit dem Schutzleiter der Kaltgeräte-Buchse und dem Masseanschluss der Sekundärspule verbunden.

Mit vier 10cm-Gewindestangen und je zwei Hutmuttern wurde das Gehäuse verschlossen – fertig 9.

Abgleich des MIDI-Konverters

Bevor ich mit der Tesla-Spule Musikdateien im MIDI-Format spielen konnte, musste die

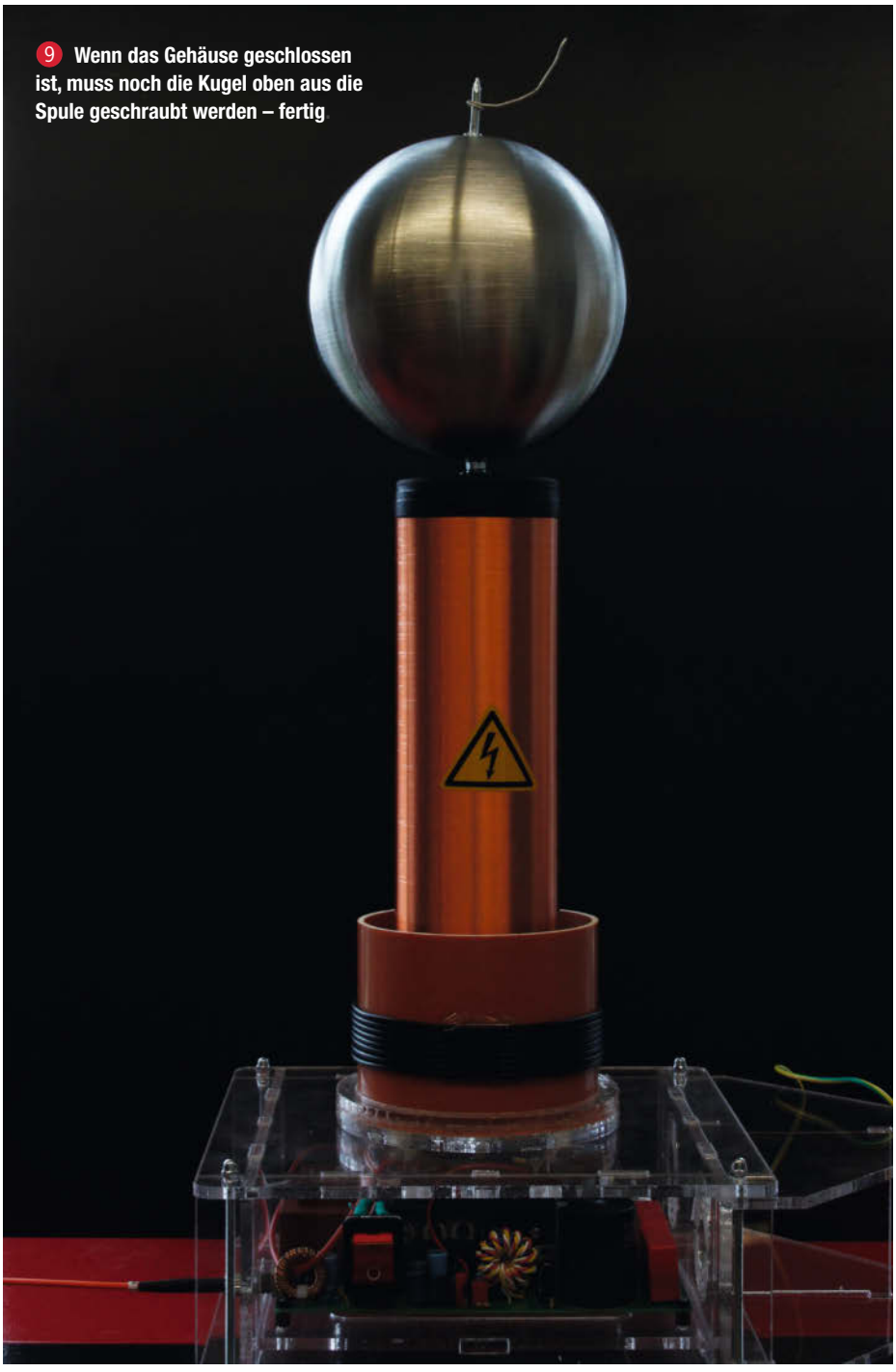
7 In diese ringförmige Nut passt die Sekundärspule, die ebenfalls eingeklebt wurde.



8 Vorn wurde ein Kippschalter für die Netzspannung, hinten eine Kaltgerätebuchse eingebaut.

Länge der Impulse, mit der die Spulenelektronik angesteuert werd, im MIDI-Konverter eingestellt werden. Zum Abgleich soll ein Oszilloskop benutzt werden. Notfalls geht es auch ohne. Die Impulslänge (sie entspricht der Zeit, in der der Primärstrom eingeschaltet ist) soll maximal 90 Mikrosekunden betragen. Dazu musste der Konverter mit MIDI-Daten versorgt werden, die ich vom PC aus mithilfe des im Software-Paket *MidiOX* enthaltenen Players *MidiBar* überspielen wollte. Allerdings lag ein entsprechendes Anschlusskabel (USB auf DIN-Buchsen 10) dem Bausatz nicht bei. Per Amazon ist das aber schnell zu beschaffen (siehe Kurzinfo-Link).

Die Impulslänge war mit dem Trimmpoti auf der Konverterplatine schnell eingestellt. Sie wurde am Pin 10 des 74LS14 mit dem Oszilloskop gemessen. Steht keines zur Verfügung, könnte man laut Bauanleitung auch die blaue LED des Konverters zum behelfsmäßigen Abgleich verwenden. Sie sollte beim Abspielen der MIDI-Musik so schwach wie möglich blinken. Falls man die Länge zu groß einstellt, besteht die Gefahr, dass die Schalttransistoren der Spule durch zu hohe Ströme beschädigt werden.



9 **Wenn das Gehäuse geschlossen ist, muss noch die Kugel oben aus die Spule geschraubt werden – fertig**

Erster Start

Die Spule fand im Make-Labor auf dem Lasercutter Platz. Sie wurde per Lichtleiter mit dem MIDI-Konverter verbunden. Nach dem Einschalten der Spule und des Konverters passierte – nichts. Die Spule zeigte lediglich durch den darin laufenden Lüfter und dem Leuchten der LEDs an, dass sie Strom erhielt. Die blaue LED am Konverter demonstrierte den Signalfluss. Aber Blitze oder Geräusche traten nicht auf.

Nach Abebben der ersten Enttäuschung widmete ich mich wieder der Bauanleitung. Vermutlich kommen solche Misserfolge häufiger vor, denn die Anleitung enthält eine ausführliche Fehlersuche mit Angabe von Messpunkten und Oszillogrammen, die dort jeweils abzuleiten sein sollten. Wichtig: Damit es dabei nicht zu einer versehentlichen Zündung der Spule kommen kann, muss die Sicherung für den Primärkreis der Spule (5A) entfernt werden. Außerdem sollte man mit dem Öffnen des Gehäuses zehn Minuten warten, um den Kondensatoren Gelegenheit zu bieten, sich wieder zu entladen. Und man muss sich bewusst sein, dass auf der Platine netzspannungsführende Leiterbahnen vorhanden sind.

Dann ging es gemäß Fehlersuch-Anleitung der Elektronik der Spule mit dem Oszilloskop an den Kragen. Zunächst suchte ich nach Ansteuerimpulsen an den beiden Schalttransistoren. Dort war nichts zu messen.

Dann ging es weiter mit der Fehlersuche nach Anleitung. Allerdings erwies sich diese Anleitung als fehlerhaft, genauer gesagt entsprach der darin enthaltene Schaltplan nicht den Verbindungen auf der Platine. Insbesondere rings um den zur Impulsformung eingesetzten 74LS14 waren die geforderten Signale nicht an den Anschlüssen zu finden, an denen sie hätten liegen sollen. Dieses IC enthält sechs invertierende Schmitt-Trigger 11.

Auf der Platine waren jedoch die Schmitt-Trigger der beiden Anschlussseiten vertauscht. Das in der Fehlersuchanleitung an Pin 12 zu messende Signal lag in der Realität an Pin 4, da statt des Triggers 4 auf der Platine Nummer 2 benutzt wurde. Und noch dazu hatte das Signal eine sehr merkwürdige Form: Statt eines kurzen Rechtecksignals war es ein sehr flaches Dreieck.

Also wurde nochmal der Schaltplan mit der Platine verglichen. Pin 4 war mit einem Kondensator gegen Masse verbunden, der im Schaltplan allerdings fehlte!?! Nachdem der Kondensator entfernt war, stimmte die Impulsform und auch die Ansteuerimpulse an den Schalttransistoren waren vorhanden, sodass ich alles wieder zusammenbaute und auch die zuvor entnommene Sicherung wieder an ihren Platz brachte.

Zweiter Start

Die Tesla-Spule kam wieder auf ihren Platz im Labor und wurde angeschlossen. Der MIDI-

Player startete und – es krachte gewaltig: Der erste Blitz der Spule schlug in die circa 90cm über der Spule hängende Deckenlampe ein, der nächste kurz darauf in die Alu-Fensterrahmen. Nach insgesamt vier Blitzen war Schluss, die Spule schwieg. Ich auch, denn mit so heftigen Entladungen hatte ich nicht gerechnet.

Meine Vermutung war, dass einer oder beide Schalttransistoren ihr Leben ausgehaucht hatten und ich bestellte gleich Ersatz. Bei dieser Gelegenheit erfuhr ich, dass es sich um nicht mehr hergestellte Typen handelte, von denen es nur noch geringe Restbestände gäbe.

Die Zeit bis zur Lieferung nutzte ich, um herauszufinden, was da schiefgegangen sein konnte. Im Grunde hatte die Spule ja funktioniert, nur nicht sehr lange. Als Ursache kamen für mich nur zwei Möglichkeiten infrage: Überlastung durch zu hohe Ströme oder durch zu hohe Spannung. Nun, eine zu hohe Spannung konnte ich vermeiden, indem ich der Spule eine geerdete zweite Elektrode spendierte, die in einem Abstand von etwa 10 bis 15cm von der Spitze auf der Kugel stand. Nach Faustregel heißt es ja: 1mm Funkenstrecke entsprechen ungefähr 1000V Brennspannung. Bei 15cm sollte ich daher bei etwa 150kV landen, eine für Spulen dieser Art durchaus übliche Spannung.

Die Masseelektrode bestand aus einer etwa 50cm langen Gewindestange (M3), die mithilfe zweier Plexiglasteile an der Spule befestigt und über die Erdbuchse an der Rückseite mit dem zweiten Pol der Spule verbunden wurde (siehe Titelbild des Artikels). Am oberen Ende setzte ich außerdem ein etwa 5cm langes Stück Schaltdraht an. Die Transistoren wurden dann getauscht und ebenfalls die Sicherung des Primärkreises.

Dritter Start

Nächster Durchgang: Inzwischen hatte der Ukraine-Krieg begonnen, weshalb ich die ukrainische Nationalhymne als geeignete MIDI-Datei wählte. Spule einschalten, Konverter einschalten und dann im Player-Programm auf den Start-Button geklickt. Sie spielte! Die Hymne erklang in knatterndem Tönen, die durch die kräftigen Blitze, die dadurch verursachte Erwärmung und Ausdehnung der ionisierten Luft und der deshalb entstehenden Schallwellen wohl noch mehrere Räume weiter zu hören war.

Nach mehreren Tesla-Konzerten fand ich dann noch einiges heraus: Zunächst kann die Spule, wie in der Anleitung kurz erwähnt, nur Mono-MIDI-Dateien abspielen. Außerdem nur solche, in denen lediglich ein Instrument (also ein MIDI-Kanal) benutzt wird. Andernfalls spielt sie nur sporadisch einige Töne der Musik oder gar nicht.

Zweitens ist der Untergrund, auf dem die Spule steht, genauer gesagt, dessen Leitfähigkeit, von Bedeutung: Steht sie auf einer metallischen, geerdeten Oberfläche (Metallrahmen des Lasercutters), ist die Zündung des Funkens unsicher und es werden nur sporadisch Blitze erzeugt. Steht die Spule hingegen auf einer isolierten Fläche (Kunststofffenster des Lasercutters), dann zündet sie sicher ohne Unterbrechungen.

Länger als fünf bis zehn Minuten sollte man sie nicht betreiben. Sie verweigert dann den Dienst. Nach einer Abkühlpause funktioniert sie aber wieder. Diese Zeit könnte man vielleicht durch eine aktive Belüftung des Gehäuses verlängern. Ausprobiert habe ich das aber nicht.

Ach, und noch etwas: Die Länge des Lichtleiters (ca. 4m) zwischen der Spule und dem MIDI-Konverter sollte man komplett ausnutzen, um soviel Abstand wie möglich zwischen Spule und ansteuerndem Computer zu bringen. Ich hatte nur etwa 2m Abstand eingehalten, was beim Betrieb der Spule zu wilden Mauszeigerbewegungen am PC führte. Offenbar streute da ein starkes Signal ins Mauskabel ein. Nach einigen gespielten Stücken streikte der USB-Anschluss, an dem die Maus hing, endgültig. Die Maus funktionierte jedoch noch an einer anderen USB-Buchse. Man kann sich damit also das Motherboard eines PCs bleibend schädigen!

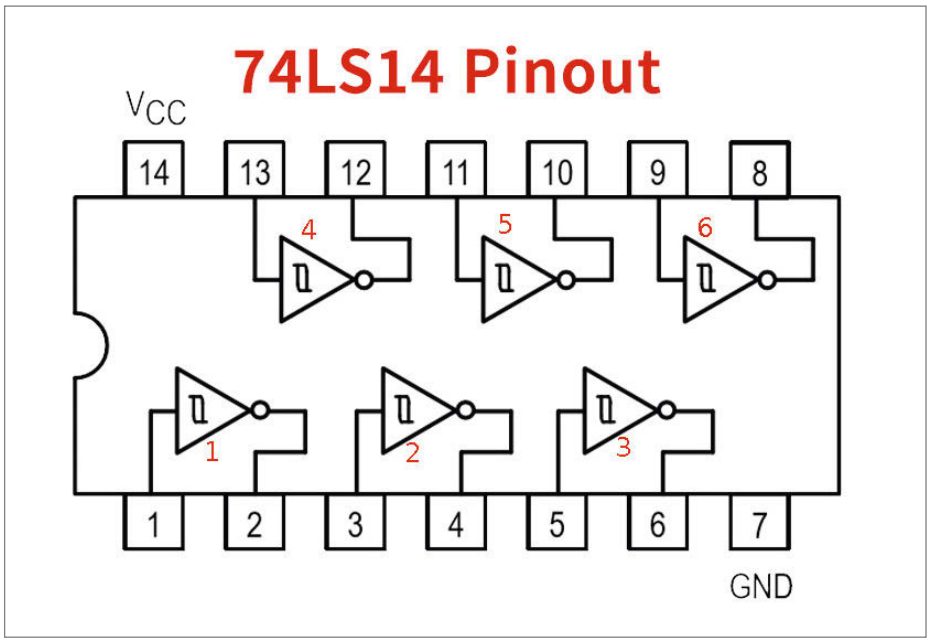
Fazit

Der Nachbau einer solch leistungsstarken Tesla-Spule ist sicher nichts für Anfänger. Selbst, wenn es die Unstimmigkeiten zwischen Anleitung, Schaltplan und Platine nicht gäbe, wäre eine recht wahrscheinliche Fehlersuche nur mit Oszilloskop und einigem Fachwissen

10 Solch ein USB-MIDI-Kabel braucht man, um MIDI-Daten vom PC auf den Konverter zu überspielen.

möglich. Dazu kommt, dass hier extrem hohe Spannungen vorkommen können und die Primärspule direkt an der Netzspannung liegt. Bei der geringsten Beschädigung der einfach isolierten Litze der Spule läge die Netzspannung offen. Ehrlich gesagt: Wagen Sie sich an solch ein Projekt nur, wenn Sie über Fachwissen verfügen.

Falls Sie aber unbedingt mit Tesla-Spulen experimentieren möchten: Es gibt da wesentlich ungefährliche Varianten, die von Batterien oder einem Niederspannungs-Netzteil versorgt werden (siehe in der Kurzinfo angegebene Artikel). Falls Sie bei denen einmal mit der Hochspannung in Berührung kommen, ist das in der Regel ungefährlich. Es sei denn, Sie tragen beispielsweise einen Herzschrittmacher oder ähnliche Implantate. *—hgb*



11 Im Schaltplan wurden die Schmitt-Trigger 1 und 2, auf der Platine aber 4 und 5 benutzt.

Umzug in Ingolstadt

In Ingolstadt zieht der *brigk Makerspace* um. Ab dem 6. Mai wird er im *Kavalier Dalwigk* zu finden sein.

brigkmakerspace.digital

Maker-Termine

Open Hardware Summit
22. April
online

2022.oshwa.org

Hackathon für Assistierende Technologien
22. – 23. April
Happylab Wien

wbt.wien/hackathon

Preisverleihung Coding Da Vinci Ost³
30. April
Filmpalast Görlitz

codingdavinci.de

Coding da Vinci Baden-Württemberg
7. Mai – 24. Juni
online

codingdavinci.de

Open Education Day
14. Mai
PHBern

openeducationday.ch

Diese und weitere Termine stehen auch laufend aktualisiert in unserem Kalender auf der Webseite unter: www.heise.de/make/kalender/

Veranstalten Sie selbst?

Tragen Sie Ihren Termin in unsere Kalender ein oder schicken Sie uns eine E-Mail an:

mail@make-magazin.de

Bild: Fablab Altmühlfranken

Fablab in Altmühlfranken geht in Betrieb

Nach viel Vorlauf sind die Räume nun fast fertig

Seit Sommer 2019 hat das Fablab Altmühlfranken mit einer Leichtbauhalle in der Industriestraße 21a eine Heimat – jetzt geht es dort endlich in den Betrieb. Aufgrund der Pandemie gingen die Umbauarbeiten deutlich langsamer voran als geplant. So wurden bereits im Sommer 2020 eine neue Zwischenebene, Wände und Fenster eingebaut. Dank tatkräftiger Ausbauhelferinnen und -helfer konnte der weitere Innenausbau von der Elektroinstallation bis zu Montagearbeiten dann selbst durchgeführt werden. Durch Sach- und Geldspenden ist die Liste der geplanten Maschinen inzwischen weitgehend abgearbeitet und umfasst neben Holz-, Metall- und Elektronikgeräten einen Laser-Cutter und Resindrucker.

Nun soll es zunächst mit Veranstaltungen für die Mitglieder losgehen. Außerdem müssen die letzten Maschinen eingerichtet und Einweisungen erarbeitet werden. Später sollen dann OpenLab-Termine für alle Interessierten dazu kommen. Auch die drei *Repair Cafés* in Altmühlfranken werden nun vom Fablab getragen. —*hch*

▶ **fablab-altmuehlfranken.de**

Open Hardware in Blievenstorf

Open Source Ecology Germany eröffnet eine neue Werkstatt

An einem Waldrand zwischen Hamburg und Berlin entsteht gerade das OpenEcoLab Blievenstorf, eine Lehr- und Open-Hardware-Werkstatt des Netzwerks *Open Source Ecology Germany* (OSEG). Seit Jahren wurde an der Ausstattung und den Räumen gearbeitet. Von der Elektrik über Beleuchtung bis hin zur Einrichtung mit Maschinen und Werkzeugen haben die Mitglieder die Umbauten zuletzt selbst gestemmt. Nun ist das Lab fertig vorbereitet und kann hoffentlich bald eröffnet werden. Neben einem Forschungs- und Entwicklungsraum für Open-Source-Hardware soll dort auch Platz für Schulungen und Wissensaustausch sein.

Den Aufbau des Blievenstorfer Labs unterstützten außerdem verschiedene Werkstätten des Netzwerks: Der Stuttgarter Hobbyhimmel schickte 3D-gedruckte „Halte-Nupsis“ und in Nieklitz wurden Werkzeugwand-Module gefräst. Geplant ist auch eine Kooperation mit dem Stuttgarter Organisationsprojekt OSSSO und die Mitarbeit am Handbuch für Offene Werkstätten. Damit das Lab zumindest im Sommer dauerhaft besetzt ist, sollen Werkpraktika und ein Freiwilligendienst angeboten werden, Kurse und Workshops sind ebenfalls in Planung.

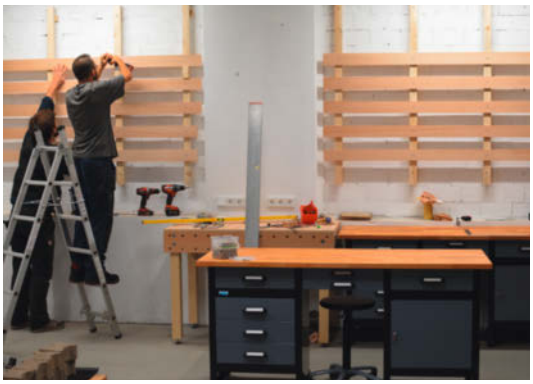


Bild: OpenEcoLab

OSEG ist Teil eines internationalen Netzwerks zum Aufbau einer Open-Source-Ökonomie. Deren Ziele umfassen neben der Produktion und Verteilung offener Designs auch soziale Gerechtigkeit und die Wiederherstellung von Ökosystemen. In Deutschland sind bereits sieben Werkstätten dabei. —*hch*

▶ **blog.opensourceecology.de**

Förderung für Open Hardware

Am 15. April startet der Prototype Fund Hardware

Bereits seit sechs Jahren fördert die *Open Knowledge Foundation* Deutschland (OKFN) mit dem *Prototype Fund* die Entwicklung offener Softwareprojekte, die dem Gemeinwohl dienen sollen. Ab diesem Jahr wird das Programm endlich ausgeweitet. Der *Prototype Fund Hardware* wird offene Hardwareprojekte unterstützen. Die Rahmenbedingungen bleiben dabei gleich: Die Projekte werden über einen Zeitraum von sechs Monaten gefördert, in denen aus einem Konzept ein Prototyp entstehen soll. Die Dokumentation zum Nachbau der Geräte muss dann unter einer Open-Source-Lizenz veröffentlicht werden. Der Fokus liegt ebenfalls auf Objekten, die im Sinne des „Public Tech" die Zivilgesellschaft voranbringen und Probleme lösen. Als Beispiele führt die OKFN etwa den offenen Lasercutter *Lasersaur* und den DIY-Laptop *MNT Reform* an.

In der ersten Runde werden dafür sechs Mal 9.5000 Euro Förderung vergeben, die an Einzelpersonen oder kleine Teams gehen können. Die Bewerbungsphase wird am 15. April beginnen und bis zum 15. Juni laufen. Anschließend wird eine Jury die Vorschläge sichten und eine Auswahl treffen. Das Geld kommt vom Bundesministerium für Forschung und Bildung. Einblicke in die Hardware-Community gibt es jeden letzten Freitag im Monat bei der *Open Hardware Happy Hour*. Die Termine und alle Infos zur Bewerbung gibt es auf der Webseite zum Programm. —*hch*

▶ **hardware.prototypefund.de**

Makerspace Nürtingen

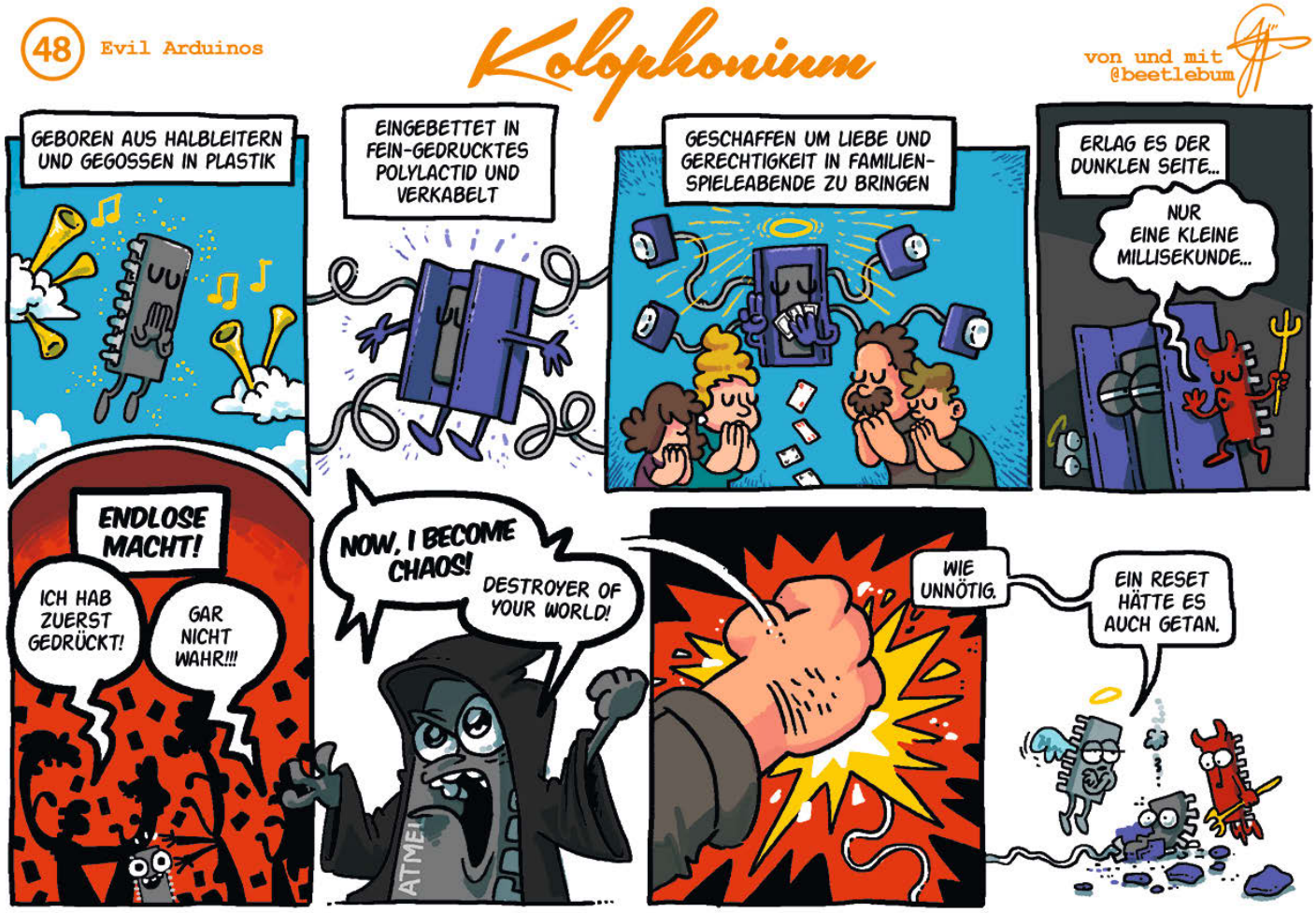
Aus einer FaceShield-Aktion in Nürtingen ist eine Maker-Community entstanden, die einen Makerspace auf die Beine stellen will.

open-diy-projects.com/makerspace

Umzug in Coburg

Die Coburger *Creapolis* zieht auf dem alten Schlachthof in neue Räume. Mitte Februar begann der Umzug in die alte Kühlhalle.

creapolis-coburg.de



Der Rasenkabelfinder

Die meisten Mähroboter benötigen ein Kabel im Rasen als Spielfeldbegrenzung. Bei meinem Versuch, den so präparierten Rasen zu lüften und zu vertikutieren, habe ich dieses Begrenzungskabel zerstört. Eine technische Lösung musste her, um das in Zukunft zu verhindern – und hier ist sie.

von Ralf Stoffels

Sei es aus Technikbegeisterung oder einfach nur Faulheit – es gibt verschiedene Gründe, sich einen Mähroboter anzuschaffen. Im Vergleich zu manchen anderen technischen Haushaltshilfen spart der tatsächlich viel Zeit, da das Mähen und die Entsorgung des Grasschnitts entfällt. Legt man Wert auf einen wildkräuterfreien Rasen unter Verzicht auf Gift, dann muss man den Rasen von Zeit zu Zeit einmal durchlüften. Das sollte man natürlich auch nicht übertreiben, weil dabei Kleininsekten und Mikroorganismen aus dem Boden geholt werden – besser als die chemische Keule dürfte das aber allemal sein. Ich habe mich auch für diese Tätigkeit für eine Maschine entschieden, die den Boden mit drehenden Messern auflockert.

Allerdings ist solch ein Rasenlüfter leider der natürliche Fressfeind des Begrenzungskabels des Mähroboters – das liegt prinzipbedingt nicht am Rand des Rasens, sondern etwa 20 Zentimeter vom Rand entfernt in der Rasenfläche. Um dieses nicht zu zerhäckseln, habe ich meinen Rasenlüfter so erweitert, dass er das Kabel erkennt, genau wie der Mähroboter.

Kurzinfo

- Signale für den Mähroboter analysieren
- Zwei Versionen: Warnlicht mit Akkubetrieb oder 230V-Version für automatische Abschaltung
- Arduino statt Oszilloskop verwenden

Checkliste

Zeitaufwand:
8 Stunden für den direkten Nachbau, bei eigenen Experimenten durchaus etliche Wochenenden

Kosten:
ab 5 Euro (LED-Version),
ab 30 Euro (230V-Version)

Material

- **Stückliste für die Elektronik**
  siehe Link

Alles zum Artikel im Web unter
make-magazin.de/xxc2

Werkzeug

- **Lötkolben**
- **3D-Drucker** (optional) für Spulenkörper und Gehäuse
- **Oszilloskop** (optional)

Mehr zum Thema

- Ralf Stoffels, DC-Motoren steuern, Make 6/16, S. 86
- Video: Der Rasenkabelfinder in Aktion

Mähroboternavigation

Wenn der Mähroboter das Begrenzungskabel erkennen kann, dann sollte das eine selbstgebaute Elektronik ja auch bewerkstelligen können. Es ist allerdings erstaunlich wenig im Netz über das Funktionsprinzip von Mährobotern zu finden. Daher bin ich meinem mit dem Oszilloskop zu Leibe gerückt.

Zunächst war mein Fokus darauf gerichtet, das Kabel selbst zu finden. Mit einer Empfangsspule und einem *BitScope* am Laptop ging ich auf die Suche nach Signalen des Kabels (1). Es ist offensichtlich, dass man hier nach etwas sehr niederfrequentem suchen muss. Die Bundesnetzagentur hätte vermutlich etwas dagegen, wenn jeder in seinem Garten einen Radiosender mit einer riesigen Antenne betriebe ...

Allerdings beruht das Prinzip nicht auf der Detektion des Kabels, sondern darauf, ob sich der Mähroboter innerhalb oder außerhalb der Leiterschleife befindet. Probieren Sie es mal aus: Sobald sich der Roboter außerhalb befindet, weigert er sich zu starten. Das kann auch innerhalb der Rasenschleife passieren, wenn man das Kabel versehentlich verpolt.

Der genaue Blick aufs Oszilloskops zeigt den Unterschied: Liegt die Testspule (hier Schaltlitze mit ca. 50 Windungen) *außerhalb* des Rasenkabels, so zeigt das Oszilloskop einen positiven Impuls, direkt gefolgt von einem negativen (2).

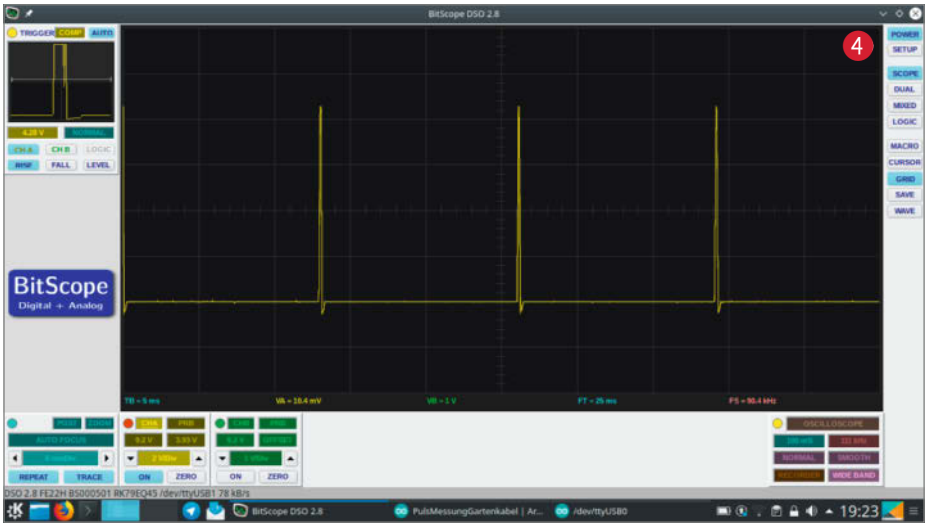
Legt man die Spule in den Bereich *innerhalb* der Rasenkabelschleife, so kehrt sich die Reihenfolge der Pulse um (3). Somit ist also permanent detektierbar, ob man sich innerhalb oder außerhalb des Begrenzungskabels befindet. Da die magnetische Flussdichte innerhalb der Schleife nur moderat abfällt, funktioniert das auch noch in der Mitte größerer (aber noch für Gärten üblicher) Rasenflächen gut.

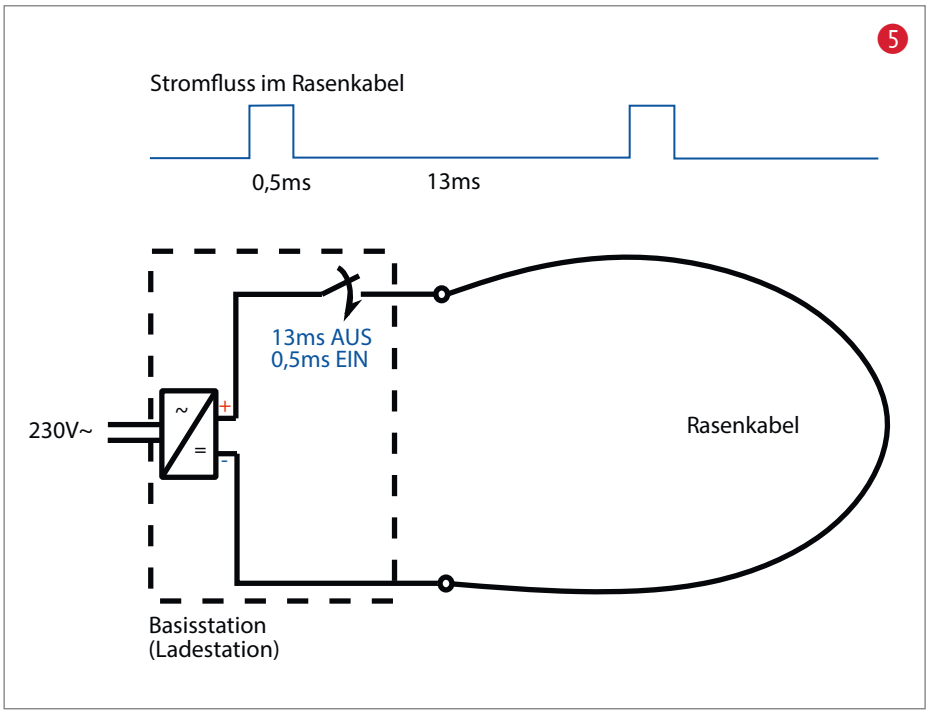
Misst man zusätzlich noch die elektrische Spannung am Kabelanschluss der Ladestation, dann erschließt sich das Funktionsprinzip komplett. Das Oszilloskopbild zeigt kurze Spannungspulse mit einer Breite von circa 0,5 Millisekunden im Abstand von 13 Millisekunden (4).

Falls Sie kein Oszilloskop besitzen: Einen Workaround auf Arduino-Basis beschreibt der Kasten *Arduino statt Oszilloskop* am Ende des Artikels.

Der Rasen als Magnet

Das Bild (5) zeigt vereinfacht das Prinzip der Schaltung in der Ladestation des Mähroboters.





Durch das kurze Anlegen einer ungefährlichen Gleichspannung an das Rasenkabel wird auf der gesamten Fläche des Rasens ein Magnetfeld erzeugt, dessen Richtung, wie im Beispiel des Schnittbilds 6, nach unten zeigt (A). Außerhalb der begrenzten Rasenfläche zeigen die Feldlinien entsprechend nach oben (B). Menschen, die Angst davor haben, von jedweden elektrischen oder magnetischen Feldern durchdrungen zu werden, mähen ihren Rasen also besser von Hand.

Man könnte nun dieses Magnetfeld zum Beispiel mit einem Hall-Sensor detektieren, der Magnetfelder in eine proportionale Spannung wandelt. Da das Feld aber nicht statisch ist, sondern alle 13,5 Millisekunden ein- bzw. ausgeschaltet wird, kann eine einfache Spule verwendet werden, in der zu den Zeitpunkten des Schaltens eine Spannung induziert wird.

Die Spannung ist proportional zu der Änderungsgeschwindigkeit des Magnetfeldes, zur durchdrungenen Fläche und zur Anzahl der Windungen der Empfangsspule. Die Änderung des Magnetfeldes erfolgt sprunghaft, was die kurzen, hohen Impulse auf dem Oszilloskop erklärt. Je größer man die Fläche der Empfangsspule macht, desto unschärfer wird die Erkennung des Übergangs von innerhalb nach außerhalb des Begrenzungskabels. Wählt man den Durchmesser also möglichst klein, braucht man viele Windungen, um eine gute Empfindlichkeit zu erreichen.

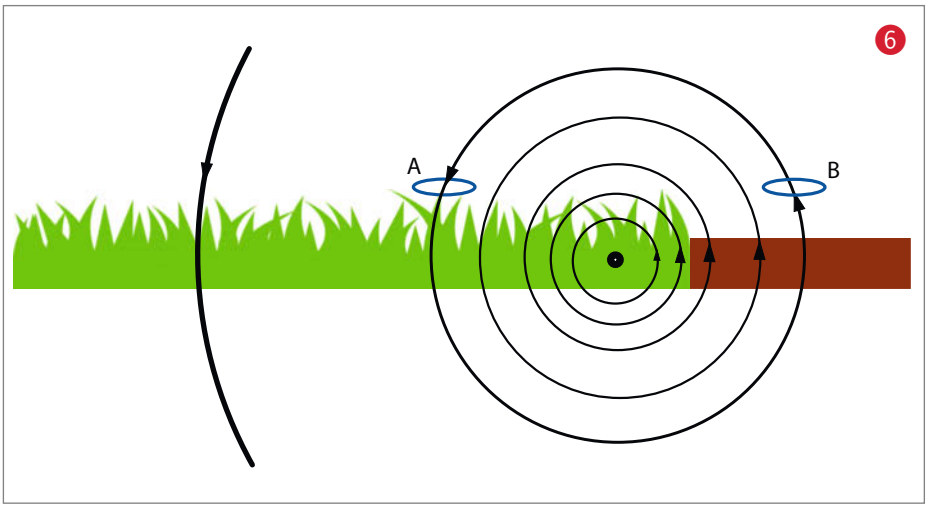
Relaisspule – keine gute Idee

Was liegt also näher, als in der Bastelkiste ein altes Relais 7 zu suchen, das seiner schaltenden Funktionen beraubt wird und nun als Empfangsspule dienen soll?

Leider war das ein Fehlschlag, der meine Schaltungsentwicklung noch einmal zurückwarf. Der Grund: Der Eisenkern des Relais muss jedes Mal magnetisiert und entmagnetisiert werden. Hierbei stellen sich Hystereseeffekte

ein, die das Signal so sehr verzerren, dass von meinem originalen Zweipulssignal nichts mehr übrig blieb. Wenn man also eine fertig gewickelte Magnetspule verwenden möchte, so sollte diese keinen massiven Eisenkern enthalten. Ich habe mich für das Selbstwickeln entschieden und den Wickelkern mit einem Spulendurchmesser von 20mm auf dem 3D-Drucker hergestellt (Download siehe Link in der Kurzinfo). Kunststoffwickelkerne gibt es aber auch im Elektronikhandel oder es findet sich etwas im Haushalt, was man zweckentfremden kann.

Bei meinem Aufbau habe ich mit 100 Windungen eine gute Empfindlichkeit erzielt.



6

Analogelektronik

Die Aufgabe der Elektronik besteht darin, festzustellen, ob zuerst ein positiver und dann ein negativer Spannungspuls empfangen wurde, oder ob die Pulsfolge umgekehrt abläuft 8. Die Abfolge der Pulse kann übrigens durch Drehung der Empfangsspule um 180 Grad umgekehrt werden, was gleich noch wichtig wird.

Die typische Lösung, die dem Maker dazu in den Sinn kommt, ist natürlich ein Arduino oder ein ähnlicher Mikrocontroller. Das Messproblem hier ist allerdings recht simpel und lässt sich mit wenigen analogen Bauteilen ebenfalls zuverlässig lösen. Für diesen Weg habe ich mich entschieden 9.

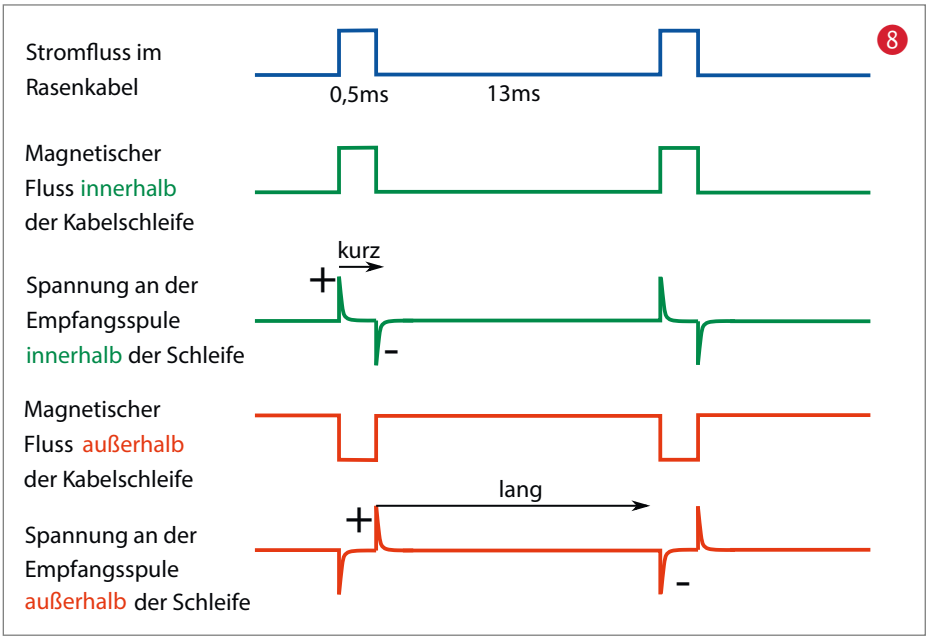
Ein Vierfach-Operationsverstärker reicht aus, um das Signal zunächst in zwei Stufen zu verstärken und dann mit einem Komparator das Ausgangssignal zu erzeugen, das anzeigt, ob sich der Empfänger innerhalb oder außerhalb des Rasenkabels befindet. Der Komparator hat hierzu eine positive Rückkopplung. Diese sorgt dafür, dass das Ausgangssignal bei einem positiven Impuls auf einer positiven Spannung „einrastet" – so lange, bis ein negativer Impuls die Spannung auf die negative Seite „einrasten" lässt. Die Schaltung „merkt" sich sozusagen die Richtung des Impulses, bis der nächste Impuls kommt.

Es ergibt sich so eine Pulsweitenmodulation. Innerhalb des Rasenkabels sind die Pulse nur 0,5 Millisekunden lang, außerhalb beträgt ihre Länge 13 Millisekunden. Mit einem einfachen Tiefpass aus einem Widerstand und einem Kondensator integriert die Schaltung dieses Signal. Wenn sich der Empfänger *außerhalb* des Begrenzungsbereichs befindet, ist das Signal überwiegend *high* mit kurzen *low*-Impulsen. Somit wird der Ausgang des Tiefpasses eine Spannung nahe der Versorgungsspannung annehmen. *Innerhalb* des Begrenzungsdrahts kehrt sich das Signal um (kurze *high*-Impulse und lange *low*-Phase), wodurch der Tiefpass eine Spannung in der Nähe von 0 Volt erzeugt.

Bei der Auswahl des Operationsverstärkers ist nur wichtig, dass er schnell genug ist. Ein sehr preisgünstiger TL074 hat bei mir ausgereicht. Der beliebte LM324 ist zu langsam.

Für den Aufbau der Schaltung habe ich eine Lochrasterplatine verwendet, auf die ich


7



8

Vorsicht mit Netzspannung

An die 230V-Version sollten sich nur Bastler herantrauen, die einschlägige Erfahrungen mit der Arbeit an Netzspannung haben. Insbesondere im Garten kann Netzspannung tödlich sein, weil hier Feuchtigkeit und guter Kontakt zur Erde hinzukommen.

Durch ein sicher isolierendes Gehäuse muss auch unbedingt sichergestellt werden, dass Benutzer des Geräts nicht gefährdet werden. Das Gerät sollte nur für den eigenen Bedarf genutzt werden.

die Spule aus 0,5-Millimeter-Kupferlackdraht mit Sekundenkleber aufgeklebt habe 10. Platziert man die Bauteile sehr nah am Operationsverstärker-IC, kommt man mit sehr wenigen Drähten auf der Rückseite aus 11.

LED-Variante

Mit der beschriebenen Schaltung lässt sich sehr einfach eine Anzeige bauen. Hierzu schließt man eine sehr helle LED, die noch im hellen Sonnenlicht erkennbar ist, direkt an den Ausgang des Tiefpasses an. Man hat die Wahl, ob es eine grüne sein soll, die innerhalb der Kabelschleife leuchtet, oder eine rote, die anzeigt, dass man sich außerhalb befindet. Die Schaltfunktion lässt sich umkehren, indem man die Empfangsspule um 180 Grad dreht.

Ein 3D-gedrucktes Gehäuse ermöglicht die Montage am Rasenlüfter 12 13. Die Druckdateien stehen unter dem Link in der Kurzinfo zum Download bereit.

Bei meinen Experimenten habe ich den Sensor mit Klettband am Rasenlüfter befestigt.

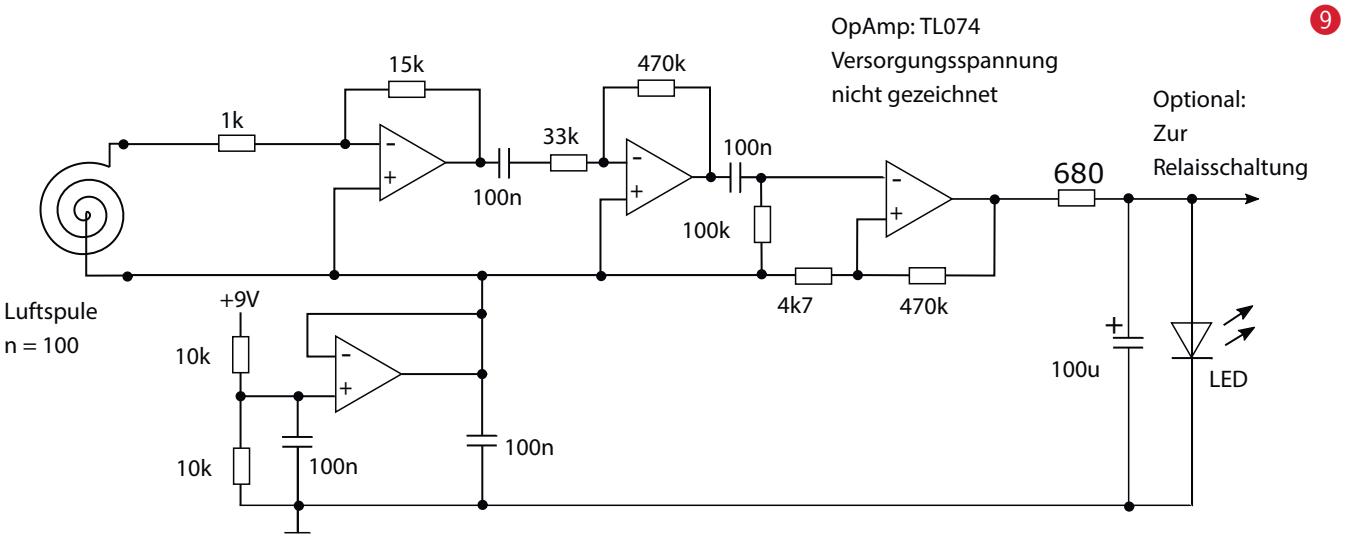
Viel Spannung und Strom

Da man so eine Warn-LED leicht übersieht, sollte mein Rasenlüfter abgeschaltet werden, sobald der Sensor das Rasenkabel überschreitet.

Hierzu habe ich eine Schaltbox gebaut, mit einer 230V-Zuleitung und geschalteter Steckdose, in die der Rasenlüfter eingesteckt wird 14. So musste ich am Gerät selber nichts verändern. Die Box hängt mit Kabelbindern befestigt an der Querstange des Rasenlüfters 15.

An der Sensorplatine habe ich gegenüber der LED-Variante fast nichts verändert. Das Ausgangssignal wird nun mit einem abgeschirmten Kabel vom Sensor an der Front des Rasenlüfters zur Schaltbox geführt. Wichtig ist dabei, dass die LED entfällt, da diese die Amplitude des Ausgangssignals zu sehr limitieren würde. Die 9V-Batterie kann auch entfallen, da die Box die Betriebsspannung der Schaltung mit einem kleinen Schaltnetzteil aus der Netzspannung erzeugt. Das Kabel braucht also drei Adern: 9V, GND und Signal.

Die erste Version meiner Schaltung enthielt ein mechanisches Relais, das für 230V und 16A spezifiziert war. Also etwa für das, was eine Haushaltssteckdose an Leistung bereitstellt. Der Rasenlüfter ist mit 1600W spezifiziert. Also genug Reserve – dachte ich. Schaltet man eine induktive Last aus, also den Motor dieses Geräts, dann wird jedoch zum Abschaltzeitpunkt eine hohe Spannung induziert. Das von mir gewählte Relais war für diesen Zweck ungeeignet, die Kontakte waren schon nach wenigen Versuchen verbrannt, was einen Kontaktwiderstand im eingeschalteten Zustand hervorrief samt jeder Menge Verlustleistung



9

im Relais, was das baldige Ableben des selbigen zur Folge hatte.

Die finale Version hat daher ein 16A-*Solid-State*-Relais, das aber zusätzlich noch durch einen 375V Varistor gegen Überspannung im Schaltzeitpunkt geschützt wurde 16. Wichtig ist, ein Solid-State-Relais zu verwenden, das eine eingebaute sogenannte *Snubber-Schaltung* hat, die dafür sorgt, dass Spannungsspitzen, bedingt durch die induktive Last des Motors, unterdrückt werden.

Tücken

Leider kann dieser Artikel keine hundertprozentig nachbausichere Anleitung geben, da jeder Mähroboter vermutlich ein wenig anders funktioniert. Er soll viel mehr motivieren, selbst zu experimentieren und hoffentlich über anfängliche Hürden hinweg helfen.

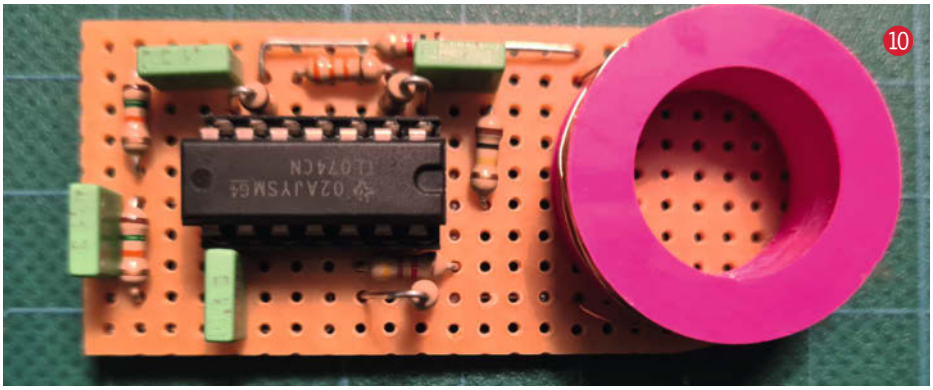
Zunächst mal muss man dafür sorgen, dass die Rasenkabelschleife überhaupt aktiv ist. Bei unserem *Landroid* von *Worx* geschieht das dadurch, dass der Robi **nicht** auf der Ladestation steht; herumfahren muss er dazu nicht. Das mag bei anderen Fabrikaten und Modellen allerdings anders sein.

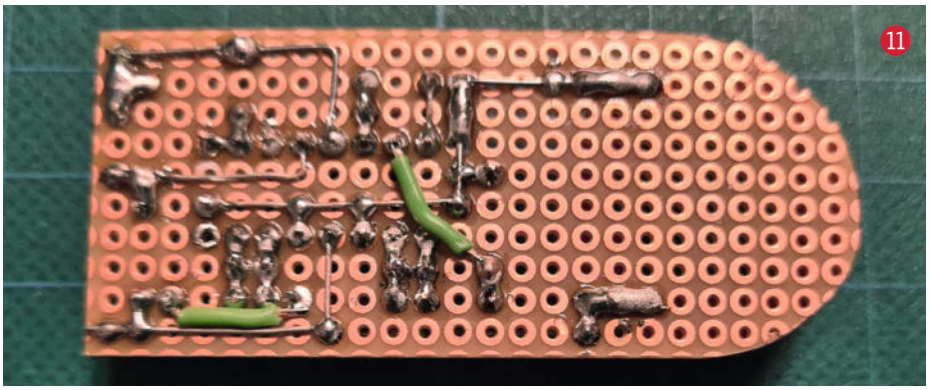
Auf dem Titelbild zum Artikel sieht man die Version mit zwei Sensoren am Rasenlüfter. Leider musste ich auf eine der beiden Empfangsspulen verzichten, weil sie sich im Streufeld des Elektromotors befand. Dadurch war keine Detektion des Kabelsignals mehr möglich. Da man aber weiß, wo ungefähr das Kabel liegt, kann man auch mit einem Sensor immer noch sicher verhindern, das Kabel zu erwischen 17.

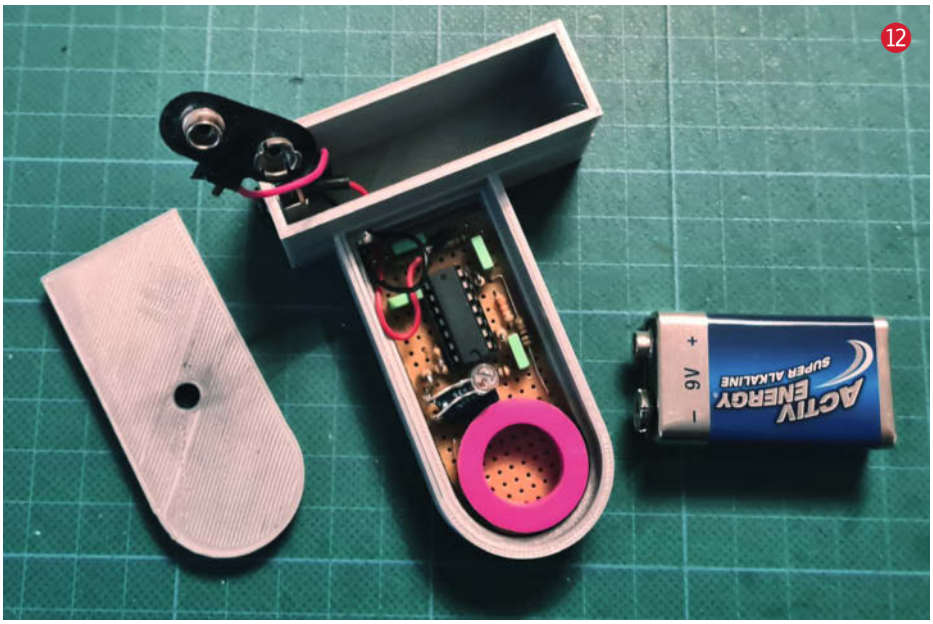
Experimente mit ferromagnetischer Abschirmung stehen noch aus.

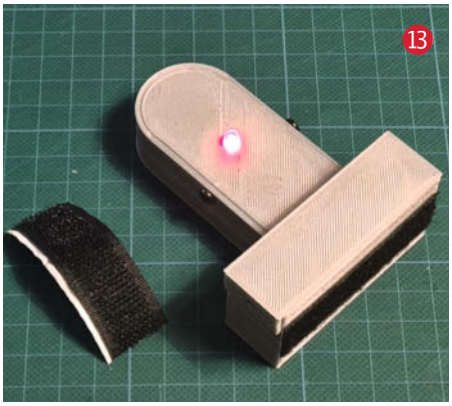
Unsichtbare Kabel

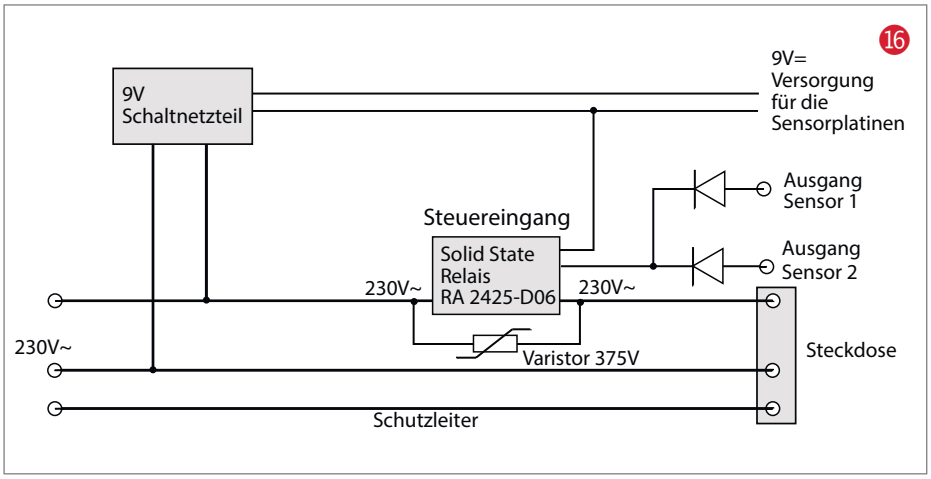
Damit der Mähroboter auch sicher um Inseln im Rasen wie Blumenbeete oder einen Gartenteich herum navigiert, legt man die Zu- und Rückleitung des Begrenzungskabels zu solchen Flächen eng parallel, so dass sich das Magnetfeld aufgrund der gegensinnigen Stromrichtung auslöscht 18. Der Mähroboter ignoriert so das Kabel und fährt ungestört darüber hinweg – der Rasenlüfter leider auch.

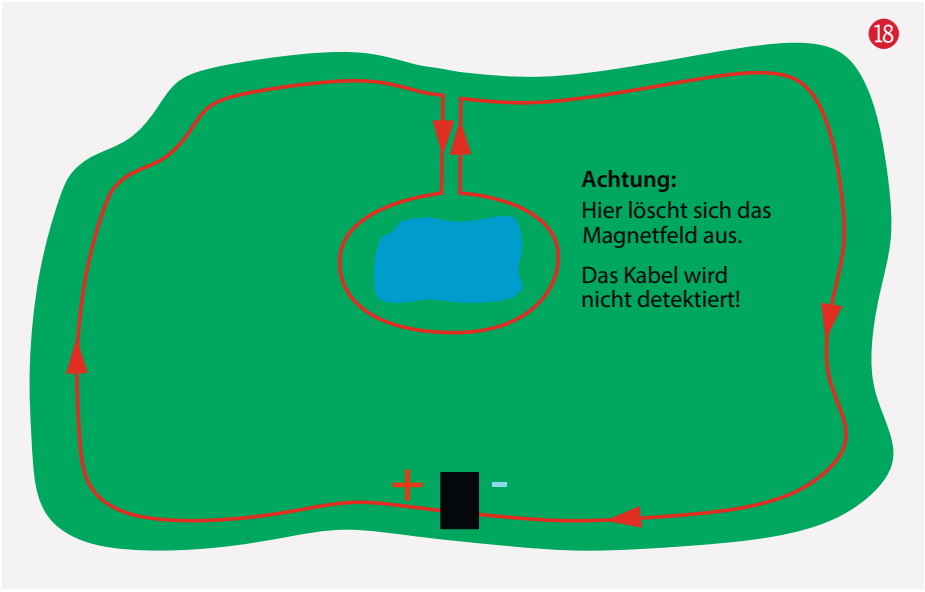












Hier muss man also aufpassen und vielleicht eine Gartenharke auf die Stelle legen, um einen Kabelunfall zu verhindern.

Während viele Mähroboter sich am Begrenzungskabel zur Ladestation „zurückhangeln", haben andere ein zusätzliches Suchkabel, das mitten im Rasen verlegt wird. Das ist eine Stichleitung, die nicht induktiv funktioniert und somit von der hier beschriebenen Schaltung nicht erkannt wird. Der automatischen Abschaltung sind also Grenzen gesetzt, die in der Natur des Verkabelung des Rasen liegen. Dennoch ist die hier beschriebene Automatik eine echte Hilfe bei der Gartenpflege und in Aktion auch im Video zu sehen (siehe Link in der Kurzinfo).

Der Rasenkabelfinder muss übrigens nicht unbedingt an einem Rasenlüfter montiert sein. Er kann natürlich auch dazu verwendet werden, das Kabel einfach nur zu finden, wenn man es anders verlegen möchte oder etwas in dessen Nähe umgraben muss. Leider kann die Schaltung kein unterbrochenes Kabel finden, da für das Messprinzip ja ein Strom hindurchfließen muss. Das ist vielleicht ein Thema für ein weiteres Projekt. *—pek*

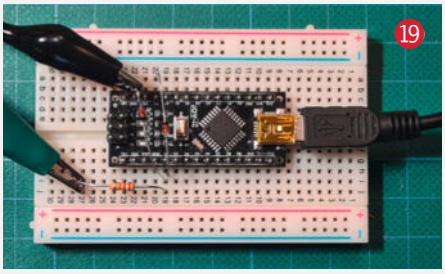
Arduino statt Oszilloskop

Die sicherste Methode, zu prüfen, ob der eigene Mähroboter für die hier beschriebene Schaltung geeignet ist, sind die Oszilloskopmessungen mit Spule wie am Anfang des Artikels beschrieben. Nun gibt es in Maker-Haushalten nicht immer ein Oszilloskop, aber sehr oft einen Arduino.

Mit der `pulseIn()` Funktion eignet sich ein Arduino, um direkt am Kabelanschluss an der Ladestation des Mähroboters die Pulse zu detektieren. Zum Schutz des Arduinos habe ich einen 22-Kiloohm-Widerstand zwischen Messsignal und Arduino-Eingang geschaltet 19.

Zwei Dioden, jeweils in Sperrrichtung gegen GND und +5V geschaltet, sind zwar nicht unbedingt nötig, verbessern aber noch einmal den Schutz des Eingangspins. Mit zwei Kabeln mit Krokodilklemmen kann man diese Testschaltung an die Kabelausgänge an der Ladestation anschließen 20. Das Begrenzungskabel sollte dabei angeklemmt bleiben, um zu verhindern, dass die Ladestation einen Kabelbruch detektiert und abschaltet. Es kann nötig sein, die beiden Klemmen zu verpolen, damit ein Signal sichtbar ist. Bei meiner Ladestation ist freundlicherweise die Klemme des Pluspols rot, während der Minuspol schwarz ist. Auch hier gibt es natürlich keinen Standard.

In meinem Fall liefern die Messungen ungefähr 13000 Mikrosekunden und ca. 500 Mikrosekunden. Den benötigten Arduino-Sketch gibt es über den Link in der Kurzinfo zum Download.



Smarte Beleuchtung für die Treppe

Eine futuristische animierte Treppenbeleuchtung, die nicht nur toll aussieht, sondern auch die Sicherheit auf der Treppe erhöht. Ein tolles Projekt für Einsteiger – Bauaufwand und Kosten halten sich in Grenzen.

von David Traum

Die Treppenbeleuchtung ist ein alltagstaugliches DIY-Projekt und ein Hingucker bei jedem Besuch. Ich fühle mich beim Betreten der Treppe jedes Mal, als ob ich die Brücke eines futuristischen Raumschiffes erklimme. Die grundsätzliche Umsetzung, was Bastelaufwand und die Programmierung angeht, ist auch für Anfänger gut zu meistern. Später kann das Treppenlicht beliebig erweitert werden, um sämtlichen optischen Wünschen und technischen Anforderungen gerecht zu werden.

Kurzinfo

- » Neopixel mit Arduino ansteuern
- » Verdrahtung und Montage
- » Experimentierfeld für RGB-Animation

Checkliste

Zeitaufwand:
ab 2 Stunden

Kosten:
ca. 80 Euro

Material

- » **Arduino** Uno oder vergleichbar
- » **Bewegungsmelder** PIR HC-SR501
- » **NeoPixel/WS2812 LED-Streifen** der benötigten Länge
- » **5V Netzteil** Leistung entsprechend der LED-Zahl
- » **Jumper-Kabel** ggf. Breadboard für Testaufbau
- » **USB-Kabel** zum Anschluss des Arduino

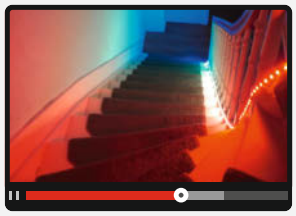
Alles zum Artikel im Web unter *make-magazin.de/xwb9*

Werkzeug

- » **Übliches Maker-Werkzeug** Schraubendreher, Zangen, Cutter-Messer etc.
- » **Bohrmaschine** ggfs. wenn Installation von Kabelkanälen nötig ist

Mehr zum Thema

- » Jan Thar, Unendlichkeitsspiegel in 3D, Make 5/19, S. 52
- » Siegbert Weidl, LED-Cube für 3D-Animationen, Make 4/19, S. 64
- » Elke Schick, Die Prinzessin auf LEDs, Make 3/15, S.16
- » Video: Treppenlicht in Aktion



Planung

Zuerst muss der Standort für die Steuerung, also den Arduino und das Netzteil, gefunden werden. Diese muss nah am Beginn des LED-Streifens positioniert sein, also am unteren oder oberen Ende der Treppe. Dabei muss darauf geachtet werden, wo eine Stromversorgung möglich ist, also eine Steckdose oder Verteilerdose vorhanden ist.

In meinem Fall war unter der Treppe eine Stromversorgung vorhanden und die Elektronik wurde unter der letzten Treppenstufe platziert. Damit die Elektronik nicht optisch stört, kann diese so gut wie möglich versteckt werden. Bei einer offenen Treppe können dazu ein Kunststoffgehäuse und Kabelkanäle genutzt werden, wie sie in jedem Baumarkt erhältlich sind.

Wenn man über eine geschlossene Holztreppe verfügt und gewillt ist, diese zu modifizieren (wie hier bei meiner Treppe), kann man die Elektronik komplett unter der Treppe verstecken und ein horizontales Loch in die unterste Stufe bohren, um die Kabel für den LED-Streifen und den Bewegungsmelder durchzuführen.

Ist die Treppe aus einem soliden Material oder soll sie nicht modifiziert werden, kann der Arduino samt Netzteil auch neben der untersten oder obersten Stufe an der Wand montiert werden. Hier kann für bessere Optik und Schutz auch wieder zu Kabelkanälen und einem Gehäuse gegriffen werden.

In der Planungsphase sollte man auch schon die nötige Länge der LED-Streifen durch Messungen an der Treppe bestimmen und die Kabellängen von LED-Streifen zum Arduino sowie die von Netzteil zum Arduino bzw. zum Stromanschluss möglichst genau ausmessen.

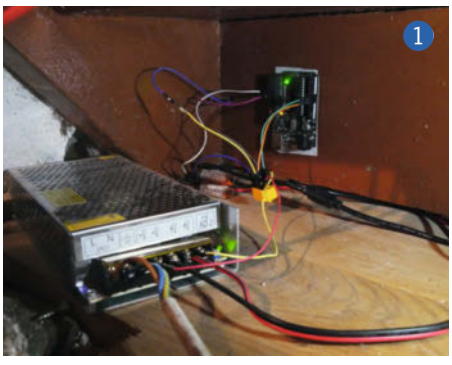
Wir verwenden hier LED-Streifen, bei denen jede einzelne LED komplett unabhängig voneinander in Vollfarben ansteuerbar sind. Bekannt wurden diese LEDs als *NeoPixel*, die technische Bezeichnung ist aber WS2812. Diese LED-Streifen gibt es mit unterschiedlicher LED-Dichte. In meinem Fall habe ich einen Streifen mit 30 LEDs pro Meter verwendet.

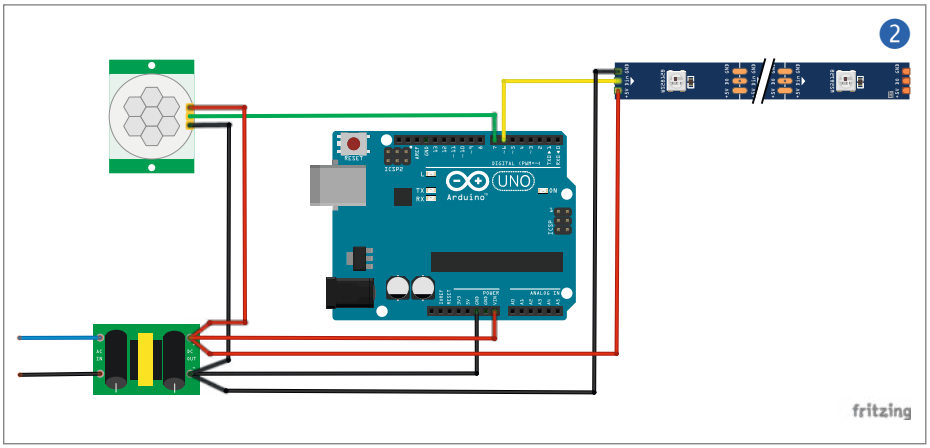
Idealerweise sollte der LED-Streifen selbstklebend sein. Wasserdichte LED-Streifen, welche mit transparentem Kunststoff ummantelt sind, sorgen für einen zusätzlichen Schutz, z. B. beim Staubsaugen.

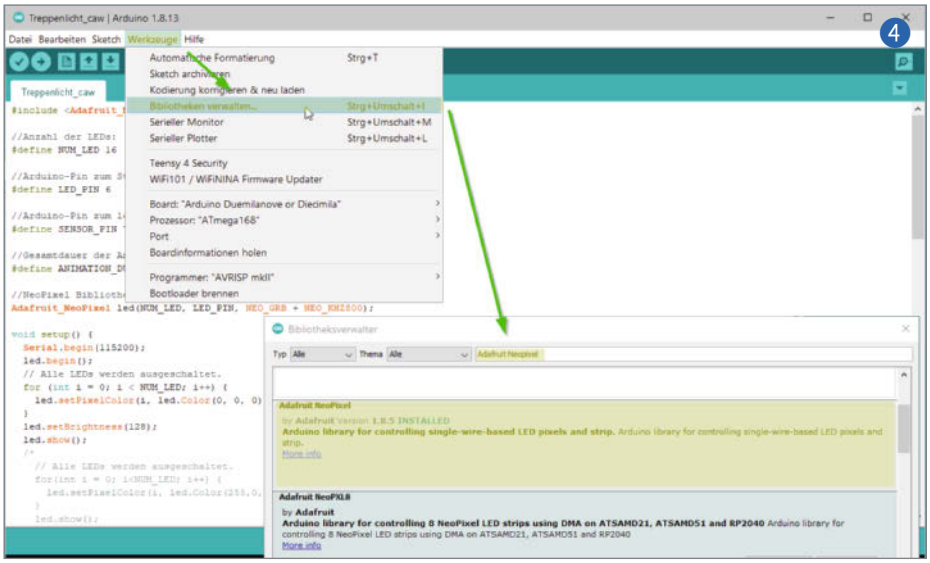
Elektronik

Der erste Testaufbau sollte am Basteltisch erfolgen, damit man nicht zu viel Zeit mit einer möglichen Fehlersuche auf oder unter der Treppe verbringen muss. Die ersten Tests können mit Jumper-Kabeln und Klemmen durchgeführt werden, bei der späteren Montage ist es sicher besser, die Verbindungen durch Löten oder (Schraub-)Klemmen herzustellen.

Das Netzteil sollte auf die Leistung des LED-Streifens abgestimmt sein, d. h. auf keinen







Fall zu wenig, aber wegen der unnötigen Kosten auch nicht zu viel Strom liefern. Der maximale Verbrauch des LED-Streifens ist im Normalfall auf den Produktseiten angegeben. Mit 30 LEDs pro Meter liegt der Verbrauch bei meinem Streifen bei 9,5W pro Meter, insgesamt also knapp unter 50W bei 5m LEDs, falls alle mit der maximalen Helligkeit leuchten. Daher wählte ich ein Netzteil mit 50W (5V bei 10A), so wird der Verbrauch der LEDs und des Arduinos perfekt gedeckt, wobei noch etwas Luft nach oben ist.

Als Erstes wird der Arduino 2 an das Netzteil angeschlossen. Wenn am Netzteil mit offenen Kontakten gearbeitet wird, sollte immer darauf geachtet werden, dass das Netzteil nicht unter Strom steht. Ansonsten kann es schnell zu Schäden an der Hardware oder zu lebensgefährlichen Situationen kommen. Sind Sie unsicher, so benutzen Sie am besten ein gekapseltes Netzteil mit fest angeschlossenem 230V-Stecker, natürlich muss dann eine Steckdose in der Nähe der Treppe vorhanden sein.

Bei einem Netzteil mit zwei getrennten 5V-Ausgängen kann ein Ausgang für den Arduino genutzt werden und einer für den LED-Streifen. Sollte nur ein 5V-Ausgang vorhanden sein, werden sowohl der Arduino als auch der LED-Streifen parallel an demselben Ausgang angeschlossen.

Der Plus-(+) oder V-Out Anschluss des Netzteils wird an den VIN-Pin (Plus) des Arduinos angeschlossen, der Minus (-) oder GND-(Masse)-Anschluss des Netzteils an einen der GND-Pins des Arduinos.

Der LED-Streifen hat drei Anschlüsse. Die VIN(+)- und GND(-)-Anschlüsse werden polrichtig an das Netzteil (nicht an den Arduino) angeschlossen. Der dritte Anschluss dient dazu, die LEDs anzusteuern. Dieser kann an einen beliebigen digitalen Output-Pin des Arduinos angeschlossen werden. Ich verwende in dem hier abgedruckten Programm Pin 6.

Das letzte Bauteil, welches angeschlossen werden muss, ist der Bewegungssensor 3. Der VIN-Pin des Sensors wird an 5V des Netzteils oder Arduinos angeschlossen. Der GND-Pin wird an einen beliebigen freien GND-Pin oder das Netzteil angeschlossen. Das Signal des Sensors wird über den dritten Pin ausgegeben. Bei Erkennung einer Bewegung wird für ein kurzes Zeitintervall eine Spannung von 5V an den Pin angelegt. Dies kann mit dem Arduino ausgelesen werden, im Programm wird dafür Pin 7 genutzt und entsprechend sollte der Arduino auch angeschlossen sein. An den kleinen Potenziometern am Sensor kann außerdem die Dauer der Signalabgabe und die Empfindlichkeit eingestellt werden, damit der Sensor wirklich nur auf Personen vor der Treppe reagiert.

Das Programm

Das Programm für den Arduino kann mit der Arduino IDE erstellt werden, welche frei heruntergeladen werden kann. Den Link und weitere Informationen zur Einrichtung der IDE finden Sie in den Links der Kurzinfo.

Die nötige Funktionalität des Programms (Download-Link in der Kurzinfo) umfasst u. a. das Reagieren auf den PIR-Sensor, die Ansteuerung der LEDs und die Erzeugung einer Farbanimation.

Nachdem die Arduino IDE installiert ist, muss die *Adafruit NeoPixel* Bibliothek 4 hinzugefügt werden. Diese kann über den Menüpunkt *Werkzeuge/Bibliotheken verwalten* in der IDE gesucht und installiert werden. Im Board Menü muss der passende Arduino ausgewählt werden und der richtige serielle Port.

Laden Sie nun das Sketch (Der Quellcode ist hier nur auszugsweise abgedruckt!) *basic.ino* aus meinem GitHub (Anleitung in den Links) und öffnen es in der Arduino IDE. Da Arduino-Files immer in einem Ordner mit dem gleichen Basis-Namen wie der Sketch liegen müssen, fragt die IDE nun, ob der Ordner erstellt werden soll, bestätigen Sie das mit *OK*. Nun können Sie den Sketch übersetzen und auf das Board hochladen. Benutzen Sie dazu den Button mit dem Pfeil nach rechts oder

```
4   #include <Adafruit_NeoPixel.h>
5
6   #define SENSOR_PIN 7 // Pin des Bewegungssensors
7   #define NUM_LED 150 //Anzahl der LEDs
8   #define LED_PIN 6 //Anschlusspin des LED-Streifens
9
10  #define ANIMATION_TIME 2000 //Dauer der Animation
11
12  Adafruit_NeoPixel led(NUM_LED, LED_PIN, NEO_GRB+NEO_KHZ800);
13
14  void setup() {
15    led.begin(); //Einrichten des LED-Outputs
16    led.setBrightness(64); //Maximale Helligkeit
17    led.clear();
18    led.show();
19  }
```

Listing Zeilen 4-19

Strg-U. Nach einer Weile sollten der Prozess erfolgreich beendet sein und wenn das Netzteil angeschlossen ist, können Sie den Näherungssensor auslösen und die Animation des Testprogramms bewundern.

In den Zeilen 1-19 des Codes ist die wichtigste Einstellung (`#define NUM_LED 150`) die Anzahl der LEDs, die sich auf Ihrem LED-Streifen befinden. Wenn Sie vom Schaltplan abgewichen sind, kontrollieren Sie, ob `LED_PIN` und `SENSOR_PIN` zu Ihrem Aufbau passen. Mit dem Wert `ANIMATION_DURATION` bestimmen Sie die Dauer der Animation in Millisekunden, 2000 Millisekunden entsprechen also zwei Sekunden Animationsdauer. Zeile 12 initialisiert die Bibliothek. Die Parameter sollten für alle *NeoPixel* (*WS2812*-LEDs) korrekt sein. Die Library unterstützt aber auch andere LEDs, konsultieren Sie im Zweifel die Dokumentation zur Bibliothek und Ihrem LED-Streifen.

In der *setup()*-Funktion werden die LEDs mittels `led.begin();` in der Library aktiviert. Mit `led.setBrightness(64);` können Sie die maximale Helligkeit der LEDs begrenzen, der Wertebereich beträgt hier 0-255. Der Aufruf von `led.clear();` schaltet eventuell noch von einem vorhergehenden Programmlauf leuchtende LEDs aus. `led.show();` aktualisiert die LEDs.

In den nach Grundfarben getrennten Funktionen in den Zeilen 21-34 werden die RGB-Farben in Abhängigkeit vom Index der LED, also praktisch der Position (0 als erste Position) auf den Streifen, berechnet. Im Listing der Zeilen 26-29 wird die Berechnung exemplarisch am Grünkanal gezeigt. In *basic.ino* wird der rote Farbkanal nicht benutzt. Versuchen Sie doch einmal, die Funktion so zu ändern, dass auch eine Rotkomponente mit berechnet wird, z. B. `return abs(128-((float(index)/NUM_LED) * 255));` oder geben Sie einfach mal eine feste Rotkomponente mit `return 87;` zurück.

Für die eigentliche Animation gibt es zwei Funktionen in den Zeilen 36-54. Die Funktion *turnOn()* aktiviert die LEDs von unten nach oben und *turnOff()* deaktiviert sie. Diese Trennung erleichtert die Programmierung unterschiedlicher Animationen und zeitlicher Abläufe für die beiden Phasen der Animation.

Im Listing der Zeilen 36-44, Zeile 38 wird die Zeit in Millisekunden bestimmt, die zwischen dem Schalten jeder LED gewartet wird. Daraufhin werden alle LEDs nacheinander auf die entsprechende Farbe (Zeile 40) gesetzt, oder auf schwarz (in *turnOff()*, Sketch Zeile 50). Nach dem Ändern der Farbe einer LED muss immer die *show()*-Funktion aufgerufen werden, um die Änderungen an die LEDs zu senden.

In der Hauptschleife `loop()` des Arduino Programms (Zeilen 56-63) wird der PIR-Sensor in einer Schleife zyklisch abgefragt. Wenn der Sensor eine Bewegung erkennt, werden die LEDs mittels `turnOn();` angeschaltet. Nach fünf Sekunden (5000ms) werden die LEDs nacheinander wieder ausgeschaltet (`turnOff();`).

```
26  // Liefert Grünwert (0-255) für die LED an der jeweiligen Stelle zurück.
27  int getG(int index) {
28    return ((float)index / NUM_LED) * 255;
29  }
```

Listing Zeilen 26-29

```
36  // Spielt die Aktivierungs-Animation ab.
37  void turnOn() {
38    int delayTime = (ANIMATION_TIME / NUM_LED);
39    for(int i = 0; i < NUM_LED; i++) {
40      led.setPixelColor(i,getR(i), getG(i), getB(i));
41      led.show();
42      delay(delayTime);
43    }
44  }
```

Listing Zeilen 36-44

```
56  void loop() {
57    if(digitalRead(SENSOR_PIN)) { // Signal vom Sensor auswerten
58        turnOn();
59        delay(5000);
60        turnOff();
61    }
62    delay(100);
63  }
```

Listing Zeilen 56-63

Montage

Bei der Montage legt man zuerst die Einzelteile grob entlang der Treppe aus. Ein weiterer Test ist jetzt empfehlenswert. Je nach Anschlussart und Platzierung des Netzteils muss hierzu eventuell die Stromzuführung demontiert werden.

Daraufhin wird der LED-Streifen an der Treppe (5) angebracht. Dabei sollte auf die Beschaffenheit der Treppe geachtet werden. Bei einem glatten Untergrund kann problemlos ein selbstklebender LED-Streifen verwendet werden. Sollte der Untergrund etwas uneben sein oder keine gute Haftung bestehen, kann z. B. mit etwas Heißkleber nachgeholfen werden. Dieser lässt sich im Normalfall auch wieder nahezu rückstandsfrei ablösen, nur bei Tapeten sollte man damit vorsichtig sein.

Der Bewegungssensor muss so montiert werden, dass er in Richtung der Personen zeigt, welche die Treppe betreten. Am besten ist eine Montage an der untersten Stufe. Bei einem Sensor ohne Gehäuse kann entweder eine Befestigung improvisiert werden (z. B. mit Heißkleber oder Draht), ein Gehäuse gekauft oder per 3D-Druck gefertigt werden. Es gibt auch technisch baugleiche Sensoren, welche für einen Aufpreis bereits über ein Gehäuse verfügen.

Zuletzt wird das Netzteil am Stromnetz angeschlossen. Im einfachsten Fall benutzen Sie eine Steckdose, ansonsten beachten Sie unbedingt die Hinweise zu 230V-Spannungen am Beginn des Artikels.

Der hier gezeigte Aufbau läuft bei mir seit etwa zwei Jahren problemlos dauerhaft durch. Der Stromverbrauch liegt im Standby-Zustand bei ca. 60mA, im Jahr kommen so Stromkosten von etwa ein bis zwei Euro zusammen. Bei ständigem Aktivieren der Beleuchtung steigt der Verbrauch natürlich etwas an. —*caw*

Optionale Erweiterungen und Ideen

Um den Sensor besser zu befestigen und optisch an die Treppe anzupassen, kann ein Gehäuse mit einem 3D-Drucker gefertigt werden. Ich habe mir ein Gehäuse von *Thingiverse* aus weißem PLA gedruckt (Link in den Kurzinfos).

Für die Farben könnten auch weitere Sensordaten genutzt werden. Es könnte zum Beispiel das Signal eines Sensors für die Außentemperatur eingespeist werden. So kann ein bläulicher Farbverlauf erzeugt werden, wenn es draußen kalt ist, und ein rötlicher, wenn es warm ist.

Neben weiteren zufällig wechselnden Farbverläufen könnte man auch noch einen Schalter abfragen, der dann den *Partymode* aktiviert und dauernd wechselnde Lichtstimmungen erzeugt oder sogar mit einem Mikrofon auf Musik reagiert.

Romménator

Mit diesem aufladbaren Arduino-Buzzer-System seid ihr bestens für die nächste Runde Rommé gerüstet. Anstatt zu klopfen, wird ab sofort nur noch gebuzzert. Für mehr Fairness und weniger Diskussionen am Spieltisch.

von Tobias Vogel

Die nächste Karte kann alles entscheiden. Während ich die Karo-8 langsam aus meiner Hand auf den Tisch lege, halten meine Mitspieler den Atem an. Es gibt keine Teams, alle sind auf sich gestellt. Und plötzlich, wie aus dem Nichts, durchbricht ein wildes Bollern die angespannte Stille und es schmettert fast gleichzeitig aus allen Richtungen: „Ich habe zuerst geklopft!" Wer recht hat, lässt sich nur schwer mit Gewissheit sagen.

Meine Familie und ich spielen sehr gern das Kartenspiel Rommé, vor allem in der „Klopfen"-Variante, bei der man Karten der vorherigen Spieler vom Tisch nehmen darf, auch wenn man nicht an der Reihe ist. Die erste Person, die auf den Tisch klopft, darf sie beanspruchen, sofern der Spieler, der an der Reihe ist, die Karte nicht gebrauchen kann. Bei einer dieser berüchtigten Spielrunden kam mir der Gedanke, dass man, ähnlich einer Quizsendung, einen Buzzer benutzen könnte, der anzeigt, wer zuerst gedrückt hat, anstatt auf den Tisch zu klopfen. So wäre das Spiel fairer und wir hätten weniger Diskussionen. Mit dem Ziel, das Projekt möglichst einfach zu halten und schnell umzusetzen, entstand der *Romménator* für bis zu sechs Spieler.

Der Aufbau

Als Basis für den Romménator verwenden wir einen Arduino Nano. Er wird mithilfe eines 18650er Lithium-Ionen-Akkus mit Strom versorgt, der mit einem TC4056 Ladereglermodul geladen wird. Das macht den Romménator mobil und erspart uns ein weiteres Kabel auf dem Spieltisch. Damit statt der 3,7V des Akkus 5V am Arduino anliegen, verwenden wir einen dazwischen geschalteten 5V-Step-Up-DC-DC-Wandler. Für die Anbindung der Taster an den Arduino benötigen wir jeweils drei Adern, sodass wir recht dünne Kabel verwenden können. Bei vielen Arcade-Tastern sind die Vorwiderstände für die LEDs bereits eingebaut, sodass wir sie nicht selbst davor löten müssen. Der Schaltplan 1 zeigt die Komponenten der Taster einzeln. In Wirklichkeit sind sie als Module mit vier Pins vereint 2 .

Die passenden Gehäuse für die Taster und die „Romménator-Zentrale" entstanden schnell und einfach in *Tinkercad*. Dem Gehäuse um Akku und Arduino habe ich noch einen Kartenhalter für den Stapel Nachziehkarten oben drauf spendiert, sodass er einen zusätzlichen Zweck erfüllt (siehe Titelbild des Artikels). Mit einer 1mm-Düse auf dem 3D-Drucker Ender 3 waren die Gehäuseteile ruck-zuck gedruckt. Natürlich ist die Qualität mit einer kleineren Düse um Welten besser, dieser Romménator hat aber das Schicksal eines Prototypen und sollte schnell fertig werden. Es können auch andere Materialien oder Kunststoffgehäuse verwendet werden, in die wir die notwendigen Löcher schneiden oder bohren.

Kurzinfo

» **Buzzer-Controller mit Arduino Nano bauen**
» **Arcade-Taster zum Leuchten bringen**
» **etwas über zufällig generierte Zahlen lernen**

Checkliste

Zeitaufwand:
5–6 Stunden (ohne 3D-Druck)

Kosten:
ca. 50 Euro

Mehr zum Thema

» Carsten Wartmann, Bartop Arcade mit Raspberry Pi, Make 4/20, S. 14
» Jan Wegener, Litium-Ionen-Akkus testen und wiederverwenden, Make 2/21, S. 70
» Mark Liebrand, Akkupacks selbst reparieren und bauen, Make 1/22, S. 94

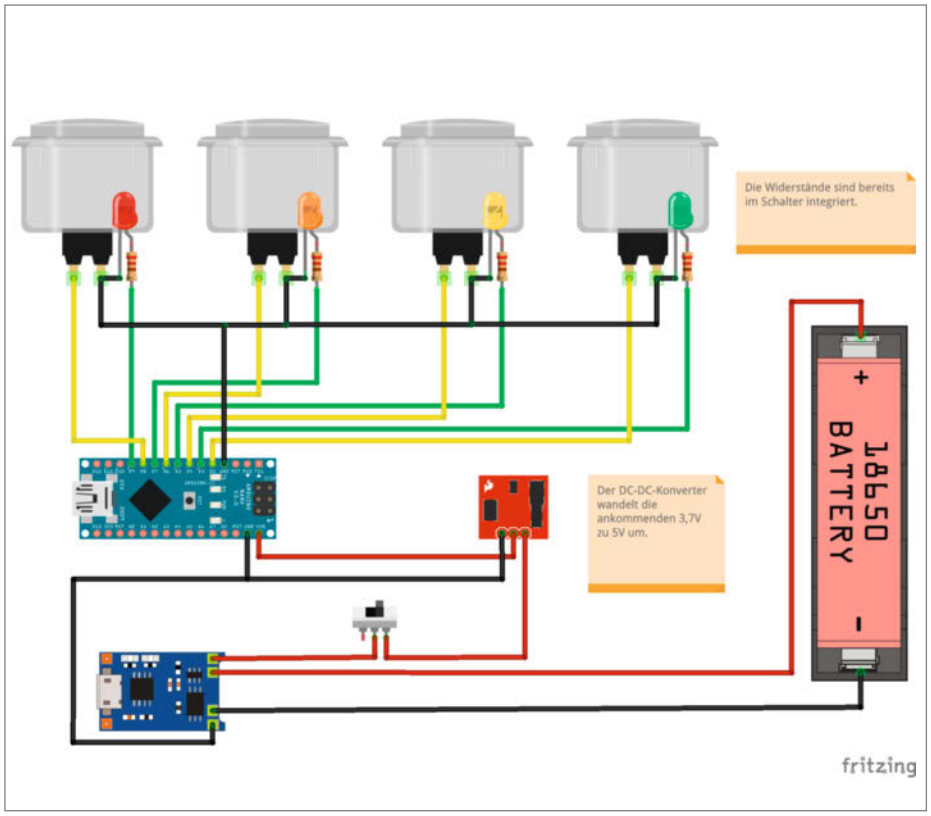
Alles zum Artikel im Web unter
make-magazin.de/xw51

Material

» **Arduino Nano**
» **Arcade-Taster** 4–6 Stück, mit LEDs für 5V, für eine 25mm Bohrung
» **Laderegler TC4056**
» **Lithium-Ionen-Akku 18650**
» **5V Step-Up-DC-DC-Wandler**
» **Akku-Halter**
» **Schiebeschalter**
» **Kabelbinder**
» **Kabel** verschiedenfarbig
» **Alternativ zum 3D-Druck: Gehäuse aus anderem Material,** Abmessungen 91mm × 70mm × 30mm für die Zentrale und je Taster 60mm × 60mm × 54mm

Werkzeug

» **Lötkolben**
» **Heißklebepistole**
» **3D-Drucker**



1 **So wird die Elektronik verbunden.**

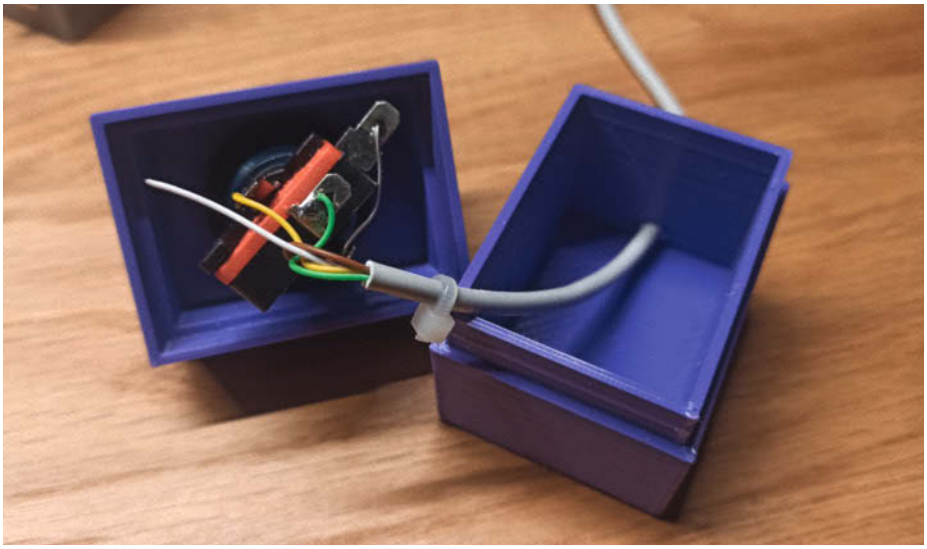
2 Der Arcade-Taster lässt sich aufdrehen. In dem blauen Kunststoffrohr befinden sich die LED und ein Widerstand. Falls eine LED nicht leuchtet, drehen wir das Kunststoffrohr um 180 Grad.

Die Gehäuse für den 3D-Druck stehen im Github-Repository des Projektes als Download bereit.

Um abschließend die Elektronik ordentlich am Boden der Romménator-Zentrale zu befestigen, verwenden wir eine Heißklebepistole 3. Die Arcade-Taster werden von oben in die jeweiligen Gehäuse gesteckt und mit dem Ring von der anderen Seite festgedreht. Dazu müssen wir vorab den Mikroschalter am unteren Ende abnehmen. Für die Zugentlastung der Kabel eignen sich beispielsweise Kabelbinder.

Arduino-Taster-Verbindungen

Arduino Nano	Anschluss am Taster
D2 (gelb)	**Taster 1** (Pin rote Hälfte)
D3 (grün)	**Taster 1** (Pin mittig)
D4 (gelb)	**Taster 2** (Pin rote Hälfte)
D5 (grün)	**Taster 2** (Pin mittig)
D6 (gelb)	**Taster 3** (Pin rote Hälfte)
D7 (grün)	**Taster 3** (Pin mittig)
D8 (gelb)	**Taster 4** (Pin rote Hälfte)
D9 (grün)	**Taster 4** (Pin mittig)



Hier müssen die Kabel am Arcade-Taster angelötet werden. Das braune Kabel wird mit den GND-Pins des Tasters und der LED verbunden.

Den Sketch übertragen

Wenn alles verkabelt ist, übertragen wir als Nächstes den Sketch *Rommenator.ino* auf den Arduino Nano, der alles steuern wird. Ihr findet ihn im Github-Repository (siehe Link in der Kurzinfo). Dazu schließen wir das Board per USB-Kabel an unseren Computer an und starten die Arduino-IDE. In der Menüleiste des Programms unter *Werkzeuge/Tools* wählen wir bei Boards den *Arduino Nano* aus der Liste und anschließend unter dem Menüpunkt *Ports* denjenigen, an dem er angeschlossen ist (z.B. *COM-4*). Danach drücken wir das Icon mit dem Rechtspfeil für die Übertragung auf das Board. Solltet ihr einen günstigeren Arduino-Nachbau verwenden, handelt es sich wahrscheinlich um ein Gerät, das den CH340-Treiber benötigt. Den Downloadlink zu dem Treiber findet ihr in der Kurzinfo. Falls sich der Sketch dennoch nicht übertragen lässt, prüft euer USB-Kabel und wählt in der Arduino-IDE im Menü *Werkzeuge* unter *Prozessor* den Eintrag *ATmega328P (Old Bootloader)* aus. Wenn die Übertragung danach gelingt, verfügt euer Arduino Nano über einen alten Bootloader, den ihr bei Gelegenheit mithilfe einer Anleitung (siehe Kurzinfo) aktualisieren könnt. Weitere Hilfe findet ihr in zahlreichen Foren und auf Webseiten zu diesem Thema. Ein originaler Arduino-Nano erspart euch diese letzten Schritte. Habt ihr alles erfolgreich übertragen, könnt ihr den Romménator direkt in Betrieb nehmen und mit dem Spielen beginnen.

Was macht der Sketch?

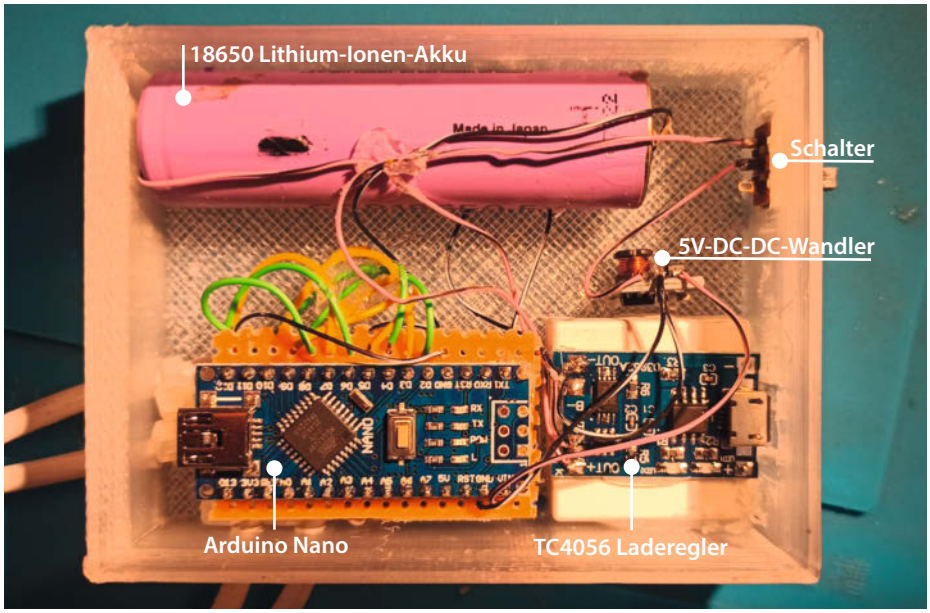
Der Romménator-Sketch steuert die Abfrage der angeschlossenen Taster und zeigt an, wer am schnellsten „geklopft" hat. Zu Beginn leuchten alle Taster. Wird einer gedrückt, blinkt dessen LED zuerst kurz und leuchtet dann eine Zeit lang durchgehend. Die anderen Taster werden daraufhin ausgeschaltet. Nach einer kurzen Wartezeit beginnen alle Taster wieder zu leuchten und es kann erneut „geklopft" werden.

Ähnlich unserem Kartenspiel stellen wir zuerst die Regeln für das Programm auf. Die Funktion `void setup()`, die beim Programmstart oder Reset nur einmal ausgeführt wird, nutzen wir, um Variablen festzulegen und die Pins für die Eingänge und Ausgänge zu initialisieren. Der Befehl `pinMode(2, INPUT_PULLUP)`; legt den digitalen Pin D2 des Arduinos als Eingang fest. Mit `INPUT_PULLUP` aktivieren wir einen eingebauten Pullup-Widerstand, sodass der Eingang permanent ein stabiles `HIGH` Signal erhält. Erst wenn der Taster aktiviert wird, wird der Zustand als `LOW` gewertet. Mit dem einfachen `INPUT` Befehl hätten wir jede Menge Störungen durch ständig wechselnde Zustände zwischen `LOW` und `HIGH` zur Folge gehabt. Die Ausgänge legen wir mit `pinMode(3, OUTPUT)`; fest, hier für den digitalen Pin D3. Am Ende der `void setup()`-Funktion werden alle Ausgänge mit dem Befehl `digitalWrite(3, HIGH)`; auf `HIGH` gestellt, hier der Pin D3. Damit sind schon einmal alle LEDs in den Tastern eingeschaltet.

Als Nächstes definieren wir die Funktion `void klopfwin(int pin)`. Sie zeigt den Gewinner an, der als Erster den Taster gedrückt hat. Hinter dem Namen `klopfwin()`, mit dem wir die Funktion später im Programm aufrufen können, finden wir nun etwas Neues: `(int pin)`. Dies bedeutet, dass beim Aufruf der Funktion die Variable `pin` einen Wert aus derjenigen Funktion erhalten kann, die sie später

aufruft, und zwar von dem Taster, der gedrückt wurde. Deshalb wird auch festgelegt, dass es sich um einen `int` (Integer: eine Ganzzahl) handeln muss. Nach dem Aufruf der Funktion werden erst einmal alle LEDs ausgeschaltet. Der Befehl `digitalWrite(3, LOW);` schaltet z.B. Pin D3 auf `LOW`. Nun soll die LED der schnellsten Person wiederholt blinken. Das erledigen wir mit einer `for`-Schleife, die so lange wiederholt wird, bis ihre Bedingung erfüllt ist. In unserem Fall: `for (int i = 0; i < 10; i++)`. Zu Beginn wird die Variable `i` als `int` auf `0` gesetzt. Nun folgt die Bedingung, dass, solange `i < 10;` ist, `i` bei jedem Durchlauf mit `i++` um eins erhöht und der Vorgang zehnmal wiederholt werden soll.

Bei jedem Schleifendurchlauf wird nun die Variable `pin`, die beim Start der Funktion mit übergeben wurde, auf `HIGH` gesetzt und anschließend nach einer Pause von genau 85 Millisekunden mit dem Befehl `delay(85);` wieder abgeschaltet. Nach einer weiteren Pause geht es mit der Schleife weiter. Sobald die Schleife beendet ist, bleibt `pin` noch einmal 3500 Millisekunden eingeschaltet und geht danach aus. Am Ende der Funktion schalten wir alle LEDs wieder ein und die Funktion ist beendet.



3 In der Romménator-Zentrale wird die Elektronik mit Heißkleber befestigt.

Jetzt fehlt in unserem Sketch noch der Hauptteil, der im `void loop()` dauerhaft erfasst, ob ein Taster gedrückt wurde. Die Zeile `if (digitalRead(2) == LOW {klopfwin(3);}` liest den Eingang D2 aus. Wenn dieser `LOW` ist, wird `klopfwin(3)` aufgerufen und so die 3 als Variable `pin` an `klopfwin(int pin)` übergeben. Trifft dies nicht zu, geht es mit

Rommenator.ino (gekürzt)

```
08   void setup() {

10      pinMode(2, INPUT_PULLUP);
11      pinMode(4, INPUT_PULLUP);
12      pinMode(6, INPUT_PULLUP);
13      pinMode(8, INPUT_PULLUP);

17      pinMode(3, OUTPUT);
18      pinMode(5, OUTPUT);
19      pinMode(7, OUTPUT);
20      pinMode(9, OUTPUT);

24      digitalWrite(3, HIGH);
25      digitalWrite(5, HIGH);
26      digitalWrite(7, HIGH);
27      digitalWrite(9, HIGH);

31   }

33   void klopfwin(int pin) {

35      digitalWrite(3, LOW);
36      digitalWrite(5, LOW);
37      digitalWrite(7, LOW);
38      digitalWrite(9, LOW);

42      for (int i = 0; i < 10; i++) {
43         digitalWrite(pin, HIGH);
44         delay(85);
45         digitalWrite(pin, LOW);
46         delay(85);
47      }

49      digitalWrite(pin, HIGH);
50      delay(3500);
51      digitalWrite(pin, LOW);

53      digitalWrite(3, HIGH);
54      digitalWrite(5, HIGH);
55      digitalWrite(7, HIGH);
56      digitalWrite(9, HIGH);

60   }

62   void loop() {

67      if (digitalRead(2) == LOW) {klopfwin(3);} //Taster 1
68      if (digitalRead(4) == LOW) {klopfwin(5);} //Taster 2
69      if (digitalRead(6) == LOW) {klopfwin(7);} //Taster 3
70      if (digitalRead(8) == LOW) {klopfwin(9);} //Taster 4

74   }
```

Zufällige void loop()-Abfrage

```
randNumber = random(4);
if (randNumber == 0 && digitalRead(2) == LOW) {klopfwin(3);} //Taster 1
if (randNumber == 1 && digitalRead(4) == LOW) {klopfwin(5);} //Taster 2
if (randNumber == 2 && digitalRead(6) == LOW) {klopfwin(7);} //Taster 3
if (randNumber == 3 && digitalRead(8) == LOW) {klopfwin(9);} //Taster 4
```

`if (digitalRead(4) == LOW) {klopfwin(5);}` weiter, bis alle Eingänge abgefragt wurden. Am Ende beginnt die Funktion `void loop()` erneut.

Fairer durch Zufall?

Da das Programm sequenziell abläuft, entscheidet bei gleichzeitigem Drücken, welcher Taster als Nächstes abgefragt wird. Nimmt man an, dass Taster 1 und 3 gleichzeitig gedrückt wurden und in genau diesem Moment das Programm die Zeile `if (digitalRead(4) == LOW) {klopfwin(5);}` für Taster 2 abarbeitet, kommt in der nächsten `if`-Anweisung der Taster 3 und gewinnt. Ist das Programm nun aber in der Zeile mit dem Befehl

Der Romménator im Einsatz. Alle Taster warten darauf, gedrückt zu werden.

`if(digitalRead(8) == LOW) {klopfwin(9);}` für Taster 4, gewinnt Taster 1.

Auch wenn man dies aufgrund der Geschwindigkeit des Arduinos sicher vernachlässigen darf, stellt sich die Frage, ob der Romménator durch das Einfügen einer zufälligen Abfrage vielleicht noch ein wenig fairer gestaltet werden kann. Die Programmänderungen dazu halten sich in Grenzen. Im Folgenden werden die Anpassungen erläutert. Den fertigen Sketch *Rommenator_rnd.ino* könnt ihr ebenfalls im Github-Repository zum Projekt herunterladen.

Vor `void setup()` wird zuerst die Variable `randNumber` als `int` deklariert. Da eine computergenerierte, zufällige Zahl mit einem Algorithmus erzeugt wird und auf eine Berechnung im Algorithmus unter gleichen Voraussetzungen immer das gleiche Ergebnis folgt, verändern wir mit `randomSeed (analogRead(0));` das Muster der Zufallszahl mittels einer „Saat“: dem Rauschen am offenen Analog-Eingang 0 des Arduinos.

Jetzt müssen wir nur noch das `void loop()` anpassen, damit die Taster zufällig abgefragt werden. Mit der neuen Zeile `randNumber = random(4);` wird eine Zufallszahl zwischen 0 und 3 generiert. Die Zahl in der Klammer gibt immer den Maximalwert -1 an. Danach ändern wir die `if`-Abfragen. Diese werden mithilfe der Variable `randNumber` nur noch dann wahr, wenn die Zufallszahl mit der vorgegebenen übereinstimmt. In jedem Fall ist die Abfrage jetzt zufälliger, aber ist sie auch fairer? Immerhin kann niemand garantieren, dass alle Taster gleich oft abgefragt werden. Den Praxistest hat das Programm jedenfalls mit Bravour bestanden und mit dem Wissen, dass niemand bevorzugt wird, fühlt sich auch niemand mehr benachteiligt. *—akf*

Noch mehr Stoff für Maker

2012–2014

2015 Heft 1-6

2016 Heft 1-7

2017 Heft 1-7

2018 Heft 1-7

2019 Heft 1-7

2020 Heft 1-7

2021 Heft 1-7

Lies sie alle!

shop.heise.de/make-magazin

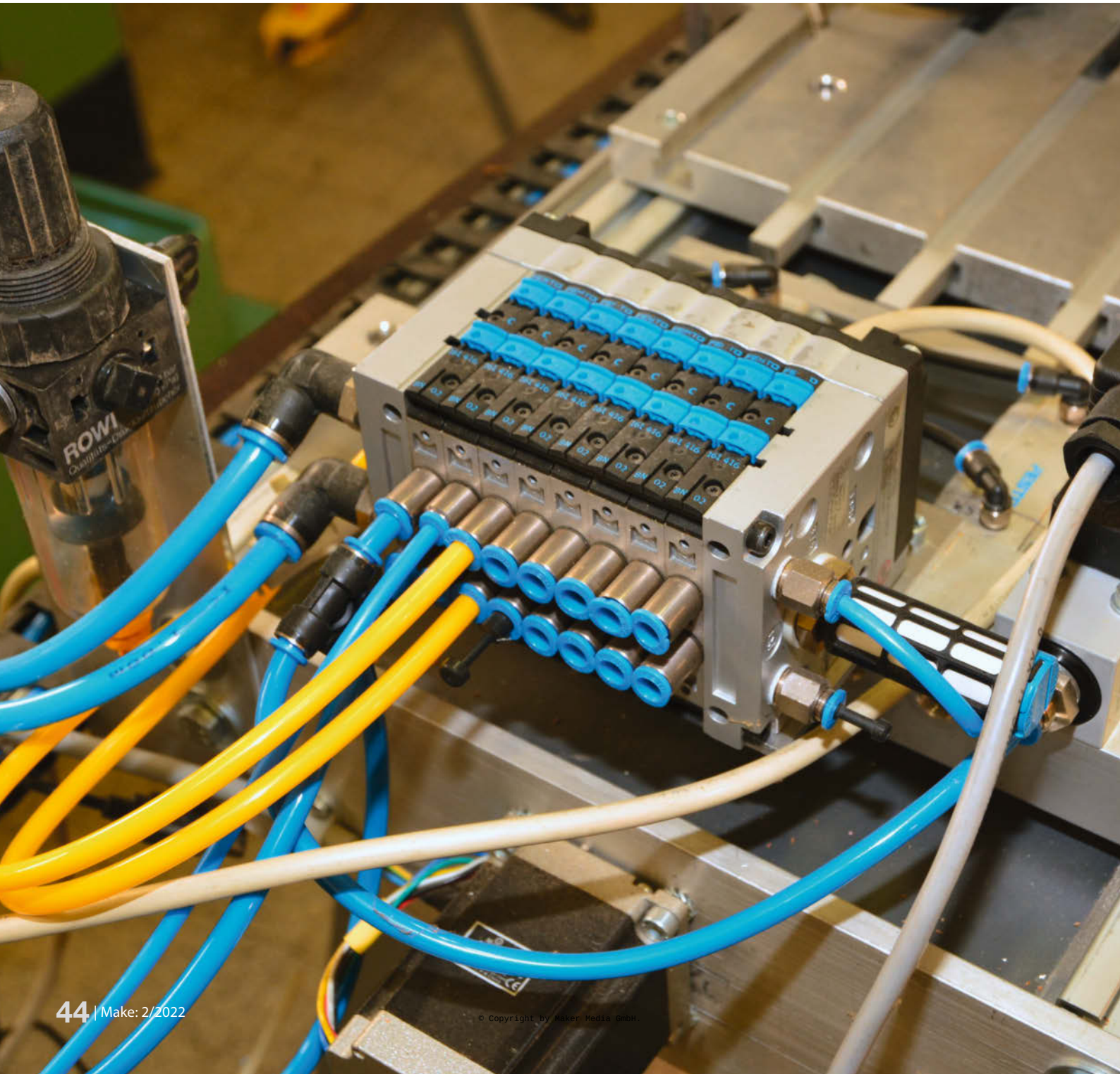
Generell portofreie Lieferung für Heise Medien- oder Maker Media Zeitschriften-Abonnenten oder ab einem Einkaufswert von 20 €.
Nur solange der Vorrat reicht. Preisänderungen vorbehalten.

heise Shop

Pneumatik: Grundlagen für Maker

Per Druckluft gesteuerte Aktoren wie Zylinder und Greifer sind im neuzeitlichen Maschinenbau so selbstverständlich wie Motoren, Servos und Relais in der Elektrotechnik. Immer wenn etwas mit ordentlich Kraft und wenig Aufwand bewegt, gespannt oder gegriffen werden soll, spielt die Pneumatik ihre Vorteile aus – auch für den Maker, so er die hier vorgestellten Grundlagen verinnerlicht hat.

von Carsten Meyer



Ein ganzer Industriezweig beschäftigt sich vorrangig mit Pneumatik im Maschinenbau – exemplarisch sei hier der Marktführer *Festo* genannt, dessen Lieferprogramm so umfangreich ist wie der Quelle-Katalog zu besten Zeiten. In der Tat werden sehr viele Bewegungsabläufe in der Kinematik von industriellen Maschinen nicht von Motoren, Servos und Zugmagneten gesteuert, sondern von pneumatisch gesteuerten Zylindern, Greifern, Saugern, Linearschlitten und rotierenden Aktoren. Ob die Maschine nun Pralinenschachteln verpackt oder Halbleiter-Wafer sortiert, immer ist Druckluft im Spiel.

Vorteil pneumatischer *Handhabungstechnik* ist der einfache Aufbau der Aktoren: Ein simpler Zylinder mit Steuerventil ist nun mal zuverlässiger als Elektromotoren mit komplizierter Servo-Elektronik und noch dazu deutlich billiger. Bei diesen Maschinen kommt es meist darauf an, bestimmte Endpositionen anzufahren, und hier lassen sich die Geschwindigkeit und Kraft, mit der dies geschieht, über vergleichsweise billige Durchfluss- und Druckbegrenzer in weiten Bereichen einstellen; soll der zurückzulegende Weg begrenzt werden, sieht man schlicht einen Anschlag vor.

Die Einfachheit der Aktoren und ihre vergleichsweise schnelle und kräftige Arbeitsweise macht die Sache auch für Maker interessant: Beispielsweise ließen sich animatronische Projekte, die lineare Bewegungen erfordern, mit Druckluft sehr viel einfacher realisieren, wenn Platz für einen kleinen Kompressor oder einen Druckluft-Vorratstank vorhanden ist. Leider braucht man dazu etwas Hintergrundwissen – wir versprechen aber, die Theorie so kurz wie möglich abzuhandeln und uns rasch den Bauteilen zuzuwenden.

Kurzinfo

- Anwendungen, Begriffe, Schaltplansymbole und Bauteile
- Kompressoren und Zubehör
- Mehrwegeventile und Ventilinseln
- Zylinder, Greifer und Drehantriebe

Mehr zum Thema

- Florian Schaeffer, Druckluft auf Vorrat, Make 2/19, S. 116

Alles zum Artikel im Web unter *make-magazin.de/xu4h*

Fluidtechnik

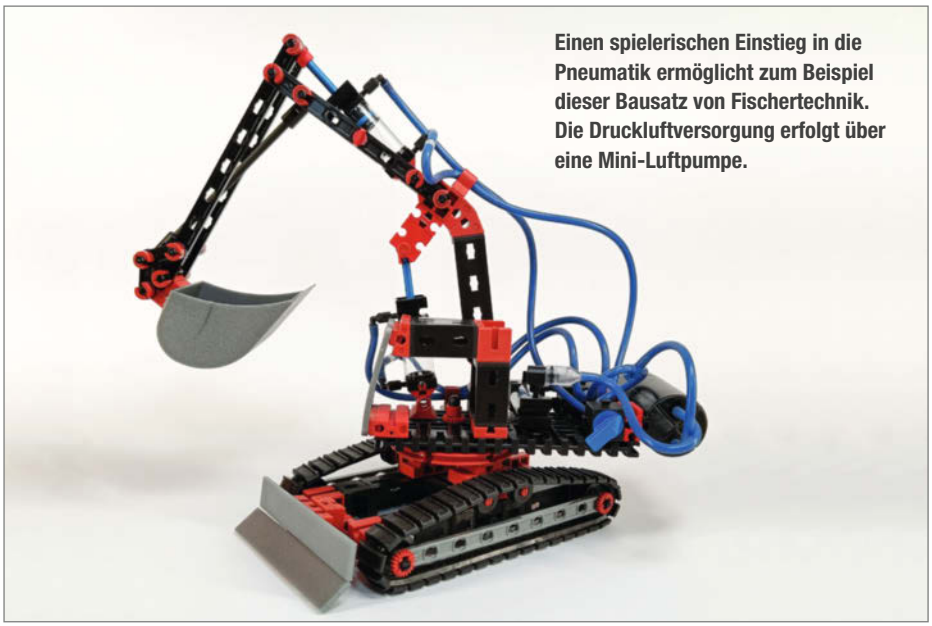
Die Pneumatik ist ein Teilbereich der Fluidtechnik, die man eigentlich eher mit Hydraulik assoziiert. In der Tat sind viele Elemente wie Zylinder und Ventile hier wie dort vorhanden, allerdings mit dem großen Unterschied, dass Gase (wie eben Druckluft) komprimierbar sind und sich deshalb „elastisch" verhalten. Man kann deshalb nicht wie in der Hydraulik über die Zuflussmenge den exakten Weg eines Aktors bestimmen (man denke zum Beispiel an die Hydraulik-Zylinder eines Baggers), sondern nur die Kraft, mit der er wirkt. Ein Zylinder, der mit Druckluft beaufschlagt wird, fährt immer bis zur Endstellung, auch wenn man die Luftzufuhr mitten in der Bewegung abschaltet: Das Bestreben von Gasen, sich ungehemmt ausdehnen zu wollen, gilt natürlich auch im Innern eines Druckluft-Zylinders.

Die der Pneumatik zugrundeliegende Physik lässt sich unter Einbeziehung thermodynamischer Prozesse beliebig verkomplizieren – darum soll es im folgenden aber (zum Glück) nicht gehen. Wir wollen Ihnen lieber einige Bauelemente der technischen Pneumatik und ihre praktische Anwendung vorstellen. Für überschlägige Berechnungen kommt man mit wenigen sehr einfachen Formeln aus, ähnlich dem Ohm'schen Gesetz in der Elektrotechnik. Das Gesetz von Boyle-Mariotte etwa sagt aus, dass das Produkt aus Gasvolumen und Druck eine Konstante ist; der Druck verhält sich demzufolge umgekehrt proportional zum Volumen.

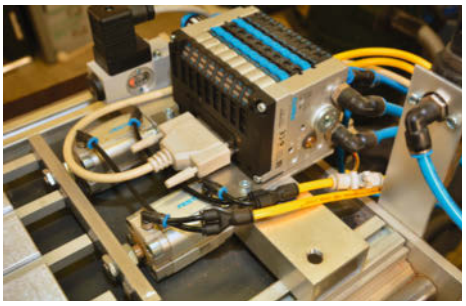
In der Praxis bedeutet dies, dass ein Gas, das auf die Hälfte seines ursprünglichen Rauminhalts zusammengepresst wird, nun den doppelten Druck aufweist oder eben auch, dass ein Liter Druckluft von 6 Bar das Bestreben hat, sich auf 6 Liter auszudehnen. Wir werden in unserem Beitrag die praktische Einheit *Bar* verwenden (1 Bar entspricht etwa dem atmosphärischen Luftdruck in Meereshöhe) und nicht das zum internationalen Einheitensystem gehörende, aber eher sperrige *Pascal* (1 Bar = 100.000 Pascal, mit 1 Pascal = 1 N/m^2).

Warnhinweis

Die Zerstörungskraft von Druckluft wird gern unterschätzt, vor allen, wenn größere Mengen schlagartig frei werden: Nicht umsonst tragen Kessel und viele Pneumatik-Bauteile ein Prüfzeichen. Bauartveränderungen (zum Beispiel Anschweißen einer Halterung an einen Kessel) sind immer gefährlich! Auch abrutschende Schläuche können umherschlagen und Schäden verursachen. Schwere Verletzungen entstehen schlicht durch Dummheit, etwa wenn man versucht, den Körper mit einer Druckluftpistole von Spänen zu befreien. Augen- und Trommelfellschäden gehören dazu, aber auch potentiell tödliche Embolien, wenn Druckluft unter die Haut und in den Blutkreislauf gelangt. Und natürlich besteht eine Quetschgefahr an Aktoren, die beachtliche Kräfte entwickeln können.



Einen spielerischen Einstieg in die Pneumatik ermöglicht zum Beispiel dieser Bausatz von Fischertechnik. Die Druckluftversorgung erfolgt über eine Mini-Luftpumpe.



Pneumatische Steuerung an einer CNC-Fräse: Die von einem Arduino gesteuerte Ventilinsel gibt Druckluft für die luftgelagerte Spindel samt Werkzeugwechsler, die Werkstückeinspannung und die Minimalmengenschmierung frei.

Luftquelle

Für Maker-Zwecke reicht schon ein kleiner Airbrush-Kompressor mit 5-Liter-Kessel, denn bei den meisten Pneumatik-Aktoren ist der Luftbedarf eher gering (dazu später). Lediglich der Druck muss stimmen: 5 bis 6 Bar sollte das Gerät schon liefern können, wenn die eingesetzten Antriebe ihre spezifizierte Kraft abgeben sollen. Die Druckluft sollte wasser- und ölfrei sein. Das ist bei ganz billigen Heimwerker-Kompressoren oft nicht der Fall: Die haben keinen *Wasserabscheider*, sind typischerweise mit einem schnell laufenden Motor mit Membranverdichter ausgestattet und nicht nur wenig leistungsfähig, sondern auch unfassbar laut (oft mehr als 95dBA). Gute Kompressoren arbeiten dagegen mit einem (oft über Treibriemen untersetzten) Induktionsmotor mit Kolbenverdichter, der mit deutlich geringerer Drehzahl auskommt und deshalb nur mit rund 85dBA nervt.

Zu den leisesten Exemplaren (deutlich unter 70 dBA) gehören Verdichter, die dem Aggregat in Kühlschränken ähneln; man findet sie oft in kleinen Airbrush-Kompressoren. Motor und Kolbenverdichter sind hier gekapselt und laufen in einem Ölbad. Im Betrieb sind sie kaum lauter als ein normaler Kühlschrank, fördern aber nur bescheidene Luftmengen. Trotzdem: Solange man keine *Saugdüsen* oder druckluftbetriebenen Werkzeuge verwendet, sind sie für die allermeisten Pneumatik-Projekte völlig ausreichend.

Ein Wasserabscheider ist eine Art Filter mit einer Klarsicht-Kapsel am Kompressor, in der sich die Luftfeuchte als Tröpfchen niederschlägt. Hat sich genügend Wasser angesammelt, kann es nach unten über eine Ventilschraube abgelassen werden. Für Experimente kommen Sie auch ohne Wasserabscheider aus, aber im Dauerbetrieb soll er verhindern, dass sich Kondenswasser in den Aktoren und Ventilen ansammelt.

Sehr leise Kompressoren arbeiten mit einem gekapselten Verdichter ähnlich dem Kältemittel-Kompressor in Kühlschränken. Links am Gerät erkennt man den Druckminderer mit Wasserabscheider.

Wasserabscheider sind oft mit einem Druckminderer und einem Manometer kombiniert. Den Druckminderer braucht man ohnehin, weil der Druck im Kessel deutlich über den benötigten 6 Bar liegt. Der Verdichter fördert gemeinhin einen gewissen Vorrat (7 bis 10 Bar) in den Kessel, bevor ihn ein *Druckschalter* unterbricht; ohne Vorrat und Druckminderer würde der Luftdruck bei Entnahme stark schwanken. Die Differenz von Kessel-Nenndruck zum Entnahme-Druck multipliziert mit dem Kesselvolumen ergibt dann die Luftmenge, die bei konstantem Druck entnommen werden kann, ohne dass der Verdichter „nachpumpen" muss. Beispiel: Mit 9 Bar auf dem Kessel und 6 Bar Entnahmedruck kann ein 20-Liter-Kessel 60 Liter Druckluft liefern.

Praktische Anwendung

Vor einiger Zeit haben wir unsere alte Alu-Fräse (erschienen in *c't Hacks* 1/2013) mit einer luftgelagerten Frässpindel aufgerüstet, was eine intensive Beschäftigung mit dem Thema Pneumatik nach sich zog: Da sie zum Betrieb Druckluft für die Lagerung und den Werkzeugwechsler (Automatic Tool Changer, kurz ATC) benötigt, haben wir einen Kompressor angeschafft und diverse Ventile zur Ansteuerung von ATC und Minimalmengenschmierung angebaut. Ein *Druckschalter* sorgt dafür, dass der empfindliche Spindelmotor nicht eingeschaltet werden kann, wenn die Druckluft für die Spindellager nicht wenigstens 4 Bar aufweist – ohne die würde er sonst augenblicklich zerstört.

Zwei Festo-*Zylinder* dienen nun zum schnellen Spannen der Werkstücke (Platinen); Drosselventile begrenzen dabei die Geschwindigkeit, mit der die selbst gefeilten Spannpratzen an das Werkstück fahren. Die Druckluft für die Zylinder, den Werkzeugwechsler und die ebenfalls mit Druckluft zerstäubende Minimalmengenschmierung (zur Alu-Bearbeitung) wird über eine (natürlich gebraucht erworbene) *Ventilinsel* geschaltet. All diese Bauteile stellen wir auf den folgenden Seiten vor.

Luft statt Elektronen

Wie in der Elektrotechnik die Drähte Elektronen transportieren, geschieht die „Stromversorgung" und die Übertragung von Befehlen in der Pneumatik über Schläuche, je nach Luftbedarf des versorgten Aktors in verschiedenen Durchmessern. Ein de-facto-Industriestandard sind die zuerst von Festo produzierten Polyurethan-Schläuche (PU) mit Außendurchmessern von 2 bis 16 Millimetern. Etwas weicher (und weniger druckbeständig) sind Pneumatik-Schläuche aus Polyethylen (PE), sie eignen sich besonders für bewegliche Baugruppen. Für den Standard-Betriebsdruck von 6 Bar eignen sie sich aber ebenfalls gut als Universal-Schlauch. Eher störrisch sind die „halbstarren" Schläuche aus Polyamid (PA), die Sie meiden sollten.

Im Unterschied zu den Druckluftschläuchen mit Gewebeverstärkung sind Pneumatik-Schläuche außenkalibriert, das heißt, hier ist der Außendurchmesser definiert und innerhalb geringer Toleranzen konstant. Das ist für die verwendeten Druckluft-Steckverbinder wichtig, denn diese dichten an der Außenfläche und nicht innen. Für Maker-typische Projekte dürfte ein kleiner Vorrat mit den Durchmessern 4, 6 und gegebenenfalls auch 8 Millimeter ausreichend sein; andere Plastikschläuche (z.B. aus der Aquaristik) sind nicht geeignet!

Damit ein PU-Schlauch beim Verlegen nicht abknickt und den Durchfluss blockiert, ist ein minimaler Biegeradius zu beachten.

Der durchflussrelevante Biegeradius, unter dem der Querschnitt schon etwas eingeengt wird, beträgt rund das Fünffache des Schlauchdurchmessers. Beispiel: Bei einem 6mm-Schlauch behindert eine Schleife von 6cm Durchmesser den Durchfluss gerade noch nicht. Wenn es enger zugeht, muss man wohl oder übel einen Eckverbinder vorsehen.

Auch wenn sich Druckschwankungen in Luftleitungen mit Schallgeschwindigkeit (343m/s) ausdehnen, ist für die Betätigung von Aktoren eher die Fließgeschwindigkeit entscheidend. Die Fachliteratur ist sich hier weitgehend uneinig und geht für Druckluft von 10 bis 30m/s aus. Letztendlich aber gilt: Um einen großen Zylinder rasch zu füllen und eine besonders schnelle Bewegung zu erzielen, ist ein dickerer Schlauch sinnvoll – hier wird bei gleicher Fließgeschwindigkeit schlicht mehr Luft transportiert.

Steckverbinder und Adapter

Die von Heimwerker-Kompressoren bekannte *Druckluft-Schnellkupplung* (siehe Bild), die beim Abkoppeln des Steckers automatisch den Luftaustritt unterbricht, ist in der industriellen Pneumatik eher selten anzutreffen. Auch für ein Pneumatik-Projekt werden Sie lediglich einen Schnellkupplungs-Adapter benötigen, also einen Stecker mit Schlauchanschluss für die vergleichsweise dünnen PU-Schläuche; ansonsten verwendet man die billigeren und kleineren Steckverbinder, Winkel und Verteiler nach Festo-QS-Bauart, die es in kompatibler Form auch von etlichen anderen Herstellern gibt.

Nur bei besonderen Anforderungen und in älteren Installationen verwendet man die klassischen Verschraubungen, bei denen der Schlauch über eine Gewindemuffe auf die Schlauchtülle geklemmt wird. Üblich sind die Schlauchverschraubungen eigentlich nur noch bei Adaptern wie etwa zur Druckluft-Schnellkupplung.

Im Druckluft-Steckverbinder sorgt ein den eingesteckten Schlauch umschließender O-Ring für die Dichtheit. Das Herausrutschen verhindert ein Ring aus Federstahl-Widerhaken, der sich in den Schlauch krallt. Um die Verbindung wieder trennen zu können, gibt es einen Kunststoffring, der bei kräftigem Hineindrücken den Federstahl-Ring entlastet und damit das Herausziehen des Schlauches ermöglicht. Schlauch-an-Schlauch-Steckverbinder gibt es gerade und abgewinkelt, als Verteiler mit drei oder mehr Anschlüssen in T- oder Y-Form und als Adapter für unterschiedliche Schlauchdurchmesser.

Ventile und Aktoren haben oft keinen direkten Schlauchanschluss, sondern ein Gewinde mit metrischen oder zölligen Durchmessern; hier ist eine *Steckverschraubung* zum Schlauchanschluss nötig. Kleine Anschlüsse sind mit einem metrischen Gewinde (M3 bis M7), größere mit einem Zollgewinde (1/8" bis 1", siehe Tabelle) versehen. Ein Außensechskant ermöglicht das Einschrauben mit einem passenden Maulschlüssel. Bei kleineren Verschraubungen ist dagegen ein Innensechskant-Schlüssel nötig, der in die Schlauchöffnung gesteckt wird. Die Zollgewinde sind mit den in der Sanitär- und Heizungstechnik verwendeten identisch, für feste Druckluftinstallationen kann man daher auch die Rohre und Fittings aus dem nächsten Baumarkt verwenden – so man mit Lötbrenner und Hanf umzugehen weiß.

Die Steckverschraubungen gibt es zum platzsparenden Einbau auch abgewinkelt; bei diesen Ausführungen kann der Kunststoffteil nach dem Festziehen noch verdreht werden, damit der Schlauch auch von einer passenden Seite eingeführt werden kann. Die Abdichtung erfolgt entweder über einen O-Ring oder bei den größeren Ausführungen auch alternativ durch eine PTFE-Beschichtung des Gewindes. Wir empfehlen die O-Ring-Dichtung, weil sich diese Steckverschraubungen mehrfach wiederverwenden lassen.

Wer sich ernsthaft mit Pneumatik-Projekten beschäftigt, wird auch bei den Steckverbindern einen kleinen Vorrat anlegen. Das ist allerdings, wenn man nicht gerade auf dem Flohmarkt oder bei *eBay* auf ein lohnendes Konvolut stößt, nicht gerade billig: Jeder einzelne Verbinder schlägt mit ein bis zwei Euro

Druckluftschläuche für die Automatisierungstechnik (links) unterscheiden sich deutlich von den gewebeverstärkten Schläuchen, die man für Druckluftwerkzeuge benutzt (ganz rechts).

Bei Druckluft-Schnellkupplungen zum Anschluss an den Kompressor oder an ein Druckluft-Netz sind 7,2 und auch 5mm Nennweite (Innendurchmesser) gängig.

Außendichtende Pneumatik-Steckverbinder, die kompatibel zum Festo-QS-System sind, gibt es auch preiswert von Zweitherstellern. Die Steckverbindung kann durch beherzten Druck auf den (bei Festo immer blauen) Kunststoffring wieder gelöst werden.

Steckverschraubungen zum Schlauchanschluss an Aktoren und Ventile: Vorn die kleineren metrischen Gewinde M3, M5 und M7, hinten die zölligen Ausführungen 1/8", 1/4" und 1/2" mit verschiedenen Dichtungen.

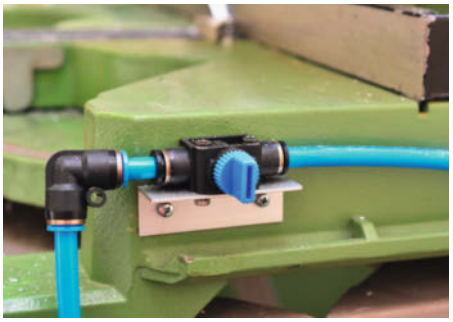
Zollgewinde

Gewindegröße	Durchmesser Außengewinde	Durchmesser Innengewinde
1/8"	9,73mm	8,85mm
1/4"	13,16mm	11,89mm
3/8"	16,66mm	15,39mm
1/2"	20,95mm	19,17mm
3/4"	26,44mm	24,66mm
1"	33,25mm	30,93mm

Durchmesser zylindrischer Whitworth-Rohrgewinde nach ISO 228; die in Zoll angegebenen Gewindegrößen entsprechen **nicht** dem jeweiligen Außendurchmesser, sondern sind historisch vom Innendurchmesser alter Gusseisen-Rohre abgeleitet.

Druckminderer für die Montage an der Maschine (rechts) und verschiedene Drosselventile zum Einfügen in eine Schlauchverbindung sowie zum direkten Anschrauben an einen Pneumatik-Aktor

Einzelne, größere Mehrwege-Ventile werden elektrisch gern über Würfelstecker angeschlossen, die es in verschiedenen Standard-Größen und auch mit LEDs zur Anzeige des Schaltzustands gibt.



Handbetätigtes 3/2-Wege-Ventil an einer Aluminium-Kreissäge. Hier wäre auch ein reines Absperrventil ausreichend, da kein Aktor angeschlossen ist, sondern nur eine Luftdüse für die Nebelkühleinrichtung.

Pneumatisch betätigte Presse mit als Fußschalter ausgeführtem 4/2-Wege-Ventil: Unbetätigt wird der Kolben und der Pressstempel nach oben gedrückt. Der 80mm-Kolben liefert bei 6 Bar eine Kraft von rund 3000N.

zu Buche – deutlich mehr als der Lötklecks oder die Schraubklemme bei den Drahtverbindungen des Elektronikers.

Druckminderer und Drosselventile

Den *Druckminderer* oder *Druckregler* haben wir schon im Kasten über die Kompressoren kennengelernt. Es gibt auch besonders kompakte Ausführungen für die Montage in der Maschine – wenn etwa ein pneumatisches Bauteil einen geringeren als den Betriebsdruck benötigt. Manchmal wird er bei Zylindern und Greifern verwendet, um deren Kraft zu begrenzen. Ein Zylinder, der nur mit 2 statt 6 Bar beaufschlagt wird, hat dann halt nur ein Drittel seiner Nennkraft. Druckminderer (oder Druckregler) sind relativ aufwendige und teure Bauteile; wirtschaftlicher ist es in jedem Fall, den Durchmesser des Wirkzylinders in einem Aktor nach der benötigten Kraft zu bemessen (dazu später).

Häufiger sind *Drosselventile*: Man benötigt sie, um die Geschwindigkeit eines Aktors einzustellen. Ohne Drosselventil würde ein mit Nenndruck beaufschlagter Zylinder schlagartig in seine Endstellung fahren, was nicht nur laut, sondern auch wenig materialschonend ist. Drosselventile erkennt man an der Einstellschraube, mit der sich der Durchfluss-Querschnitt verengen lässt; je weiter sie im Uhrzeigersinn eingedreht wird, desto geringer ist der Durchfluss. Im Unterschied zu Druckminderern verringern Drosselventile nicht den erreichten Enddruck – die Kraft eines Zylinders bleibt also vollständig erhalten, auch wenn er sich nun langsamer bewegt.

Drosselventile werden gern mit einer Steckverschraubung zur direkten Montage am Aktor kombiniert. Damit die Luft aus einem Zylinder, der sich zurückbewegen oder entlastet werden soll, schnell entweichen kann, sind sie üblicherweise mit einer *Rückschlagventil*-Funktion ausgestattet: Die Luftmenge wird hier nur in der Richtung für den Arbeitshub begrenzt, in der anderen Richtung findet keine Drosselung statt. Die Arbeits-Durchflussrichtung wird in Form eines Pfeiles oder durch Beschriftung der Anschlüsse mit IN und OUT gekennzeichnet.

Reine *Rückschlagventile* sind sozusagen die Dioden der Pneumatik: Sie lassen die Luft nur in einer Richtung passieren, in der anderen ist der Durchfluss gesperrt. Da sie im Innern nur aus einer beweglichen Kugel mit Dichtsitz bestehen, sind sie preiswert und klein. Drosselventile mit Rückschlag-Funktion lassen sich übrigens als reine Rückschlagventile verwenden, wenn man die Stellschraube für die Durchfluss-Regulierung komplett hineindreht.

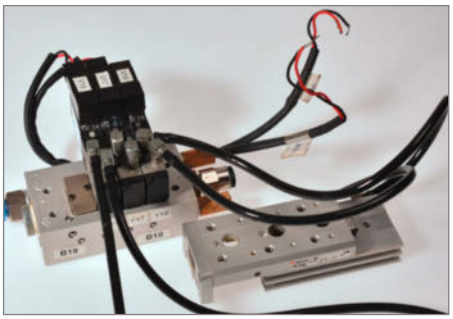
Ebenfalls zu den Drosselventilen gezählt werden die kleinen Schalldämpfer aus Sinterbronze oder Kunststoff, die an den Luftauslässen von Mehrwege-Ventilen montiert werden. Ohne sie würde sich die beim Entlasten eines Aktors schlagartig frei werdende Druckluft recht geräuschvoll äußern. Sie sind mit einem Schraubgewinde versehen und werden meist direkt am Auslass eines Mehrwege-Ventils montiert.

Mehrwege-Ventile

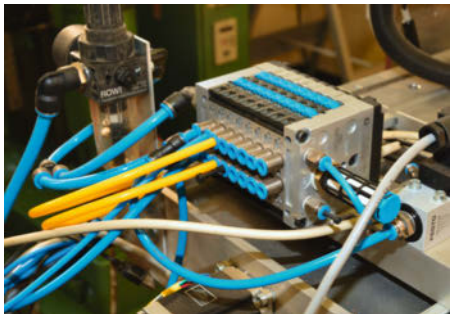
Mit reinen Absperrventilen nach dem Wasserhahn-Prinzip kommt man in der Pneumatik nicht allzu weit: Sperrt man die Luftzufuhr zu einem Zylinder einfach ab, ändert sich an seinem Zustand gar nichts: Der Druck im Innern bleibt ja nach wie vor bestehen, und ohne Leckage würde er auf alle Zeiten so verharren. Will man den Zylinder zurückbewegen oder wenigstens entlasten, sodass er beispielsweise von einer Feder (der berühmten Zylinder-Rückholfeder) zurückbewegt werden kann, muss man dafür sorgen, dass die auf den Kolben wirkende Druckluft im abgeschalteten Zustand auch wieder entweichen kann.

Pneumatik-Ventile sind daher fast immer Mehrwege-Ventile, vergleichbar mit einem Umschalter in der Elektro-Technik: Sperrt man hier die Druckluftzufuhr ab, wird gleichzeitig ein Weg vom Zylinder zu einem dritten Anschluss geöffnet – die Luft im Zylinder kann entweichen und gibt ihn frei. Ein solches Ventil nennt man 3/2-Wege-Ventil, weil es drei Anschlüsse und zwei Schaltstellungen besitzt, während ein einfaches Absperrventil ein 2/2-Wege-Ventil wäre. Bei einfachen handbetätigten Ventilen ist der Entlüftungsausgang oft verborgen und nicht als Anschluss herausgeführt, die Druckluft entweicht hier zum Beispiel durch eine Öffnung im Schaltknebel.

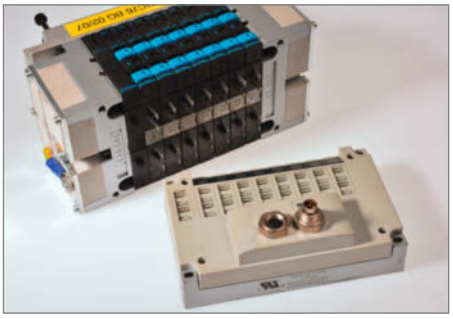
Magnetisch betätigte Mehrwegeventile spielen insbesondere bei einer elektronischen Ansteuerung eine große Rolle. Ein sehr häufiger Anwendungsfall ist ein Aktor (Zylinder oder Greifer), der vor- und zurückbewegt werden soll. Hier wirkt die Druckluft abwechselnd auf beide Seiten des Kolbens; die nicht



Mini-Ventilinsel zum Anschluss von insgesamt drei pneumatischen Aktoren. Auch hier sind 24V-Spulen zur Ansteuerung eingesetzt. An eines der Ventile ist ein Linearschlitten angeschlossen.



Festo-Ventilinsel CPV-10 in unserer Maschine, bestückt mit acht 5/3-Wege-Ventilen Mit den blauen Schiebern auf der Oberseite kann man die Ventile im Notfall oder zum Test auch von Hand betätigen.



Ventilinsel mit Spulenansteuerung über Feldbus-Anschluss: Wenn man das aufgeschraubte Feldbus-Modul entfernt, sind die Anschlüsse der Magnetventile einzeln zugänglich.

beaufschlagte Seite muss daher entlastet sein. Zur Ansteuerung eines einzelnen Zylinders würde man also zwei 3/2-Wege-Ventile benötigen, von denen immer nur eines gleichzeitig eingeschaltet ist und das jeweils andere die auf der Gegenseite enthaltene Luft entweichen lässt.

Schaltet man (versehentlich) beide Ventile ein, ergibt sich ein indifferenter Zustand: Der Kolben übt dann (fast, siehe unter *Zylinder*) keine Kraft aus, weil er auf beiden Seiten den gleichen Druck „sieht"; er saust keinesfalls, wie man vielleicht unbedarft vermuten könnte, in eine Mittelstellung.

Um die „Verschlauchung" einfacher zu gestalten, bietet die Industrie kombinierte Wegeventile an, die zwei 3/2-Ventile zu einem 4/3- oder 5/3-Wegeventil zusammenfassen. Das magnetisch gesteuerte 5/3-Wegeventil ist das Standardbauteil zur Ansteuerung von Aktoren schlechthin, es lohnt daher, etwas näher darauf einzugehen.

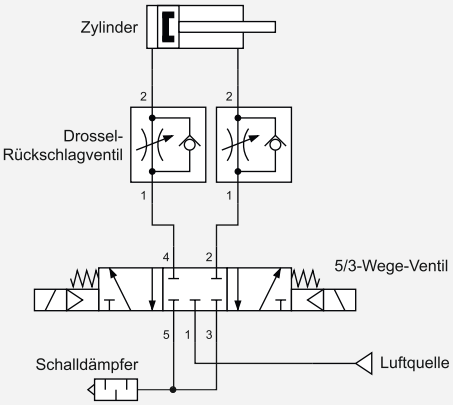
Bei den 5/3-Wege-Ventilen wird bei Betätigung der Luftstrom zur einen Seite des Kolbens (zum Beispiel Ausgang 2) freigeschaltet, während die andere Seite (hier dann Ausgang 4) entlüftet wird. Die Abluft-Anschlüsse 3 und 5 können offen bleiben, werden aber üblicherweise an einen Schalldämpfer angeschlossen und dürfen auch zusammengeführt werden.

Pneumatik-Symbole

Auch in der Pneumatik gibt es genormte Schaltzeichen – wie sollte man auch sonst den Plan einer Maschine erstellen? Während einige Symbole mit etwas Fantasie noch selbsterklärend sind (z.B. bei Zylindern, Rückschlag- oder Drosselventilen), sind insbesondere die Schaltzeichen für die Wegeventile erklärungsbedürftig.

Bei einem Wegeventil gibt es zwei oder drei Schaltstellungen, die durch Quadrate dargestellt werden. Bei einem Ventil mit zwei Schaltstellungen gilt das Quadrat für die unbetätigte Grundstellung, bei dem die Nummern der Anschlussleitungen eingezeichnet sind. Das linke Quadrat stellt den betätigten Zustand dar, wobei Pfeile die Richtung des Luftstroms anzeigen. Ventile mit *pneumatischer Vorbetätigung* arbeiten nur mit einem Luftstrom in Pfeilrichtung.

Bei Ventilen mit drei Schaltstellungen gilt das mittlere Quadrat für die Grundstellung, links und rechts davon sind die Stellungen mit Flussrichtung für eine Betätigung mit dem angrenzenden Schalter (von Hand, per Druckluft oder elektromagnetisch betätigt) dargestellt. Das T-förmige Symbol steht für eine abgesperrte Leitung.

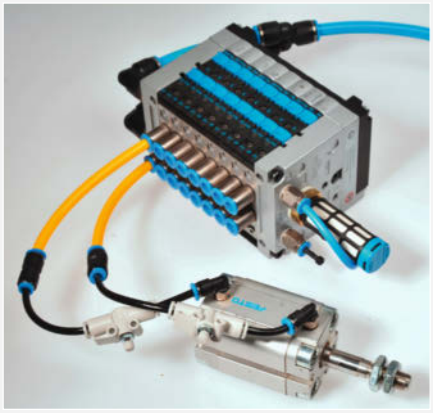


Typische Pneumatik-Schaltung mit Mehrwege-Ventil, Drosselventilen und einem Zylinder als Aktor.

Die Nummerierung der Anschlüsse folgt einem festen Schema, wobei die genannten Buchstaben bei älteren Ventilen anzutreffen sind:

- 1, 11 oder P – Hauptluftanschluss, Eingang Druckluftversorgung vom Kompressor
- 2, 4 oder A, B – Arbeitsleitungen zum Aktor (Zylinder, Greifer)
- 3, 5 (ggf. auch 7) oder R, S, T – Entlüftungsleitungen, ggf. zum Luftaustritts-Schalldämpfer
- 12, 14, 16 oder Y, Z – Steueranschlüsse, pneumatisch oder elektrisch
- 84 – Getrennter Steuerluft-Anschluss bei pneumatisch vorgesteuerten Ventilen

Zum Kennenlernen der Symbole und zum Zeichnen eigener Pneumatik-Schaltpläne eignen sich zum Beispiel der kostenlose Online-Editor *PneuDraw* von SMC (siehe Link im Info-Kasten), mit dem wir auch den abgebildeten Plan erstellt haben.



Praktische Ausführung der nebenstehenden Schaltung: Sie bewegt einen Zylinder hin oder her, wenn die Ventilspulen angesteuert werden.

Verschiedene Pneumatik-Standardzylinder: Als häufig eingesetzte Bauteile findet man eine Fülle von verschiedenen Ausführungen.

Linearschlitten von den Herstellern Festo und SMC: Der Schlitten läuft kugelgelagert und hat sehr wenig seitliches Spiel.

Präzise Greifer und Drehantriebe gehören zu den teuren Bauteilen in der Handhabungstechnik. Für den Maker lohnt hier nur der Gebraucht-Kauf.

Saugdüse als eigenständiges Bauteil: Über ein Venturi-Röhrchen erzeugt sie aus Druckluft einen Unterdruck, oft benötigt zum Anheben von Verpackungsgut und Bauteilen mit Saugern. Die „verbrauchte" Luft entweicht über einen Schalldämpfer.

Im Ruhezustand ist der Aktor drucklos, während bei einem 5/2-Wege-Ventil mit nur zwei Schaltstellungen eine Seite des Aktor-Kolbens immer mit Druck beaufschlagt ist; es kennt keine drucklose Ruhestellung.

Beim 4/3-Wege-Ventil gibt es nur einen gemeinsamen Entlüftungsanschluss, was in der Praxis kaum einen Unterschied zum 5/3-Wege-Ventil macht. Manuell betätigte Ventile zur Ansteuerung eines doppeltwirkenden Zylinders (siehe unten) sind oft 4/2-Wege-Ventile: Es gibt einen Lufteinlass, einen (möglicherweise verborgenen) Luftauslass, zwei Anschlüsse für den Aktor und natürlich zwei Schaltstellungen (vor und zurück).

Um die Ansteuerleistung bei elektrisch betätigten Ventilen zu verringern, sind sie oft *pneumatisch vorgesteuert*: Das Magnetventil gibt hier nur einen Hilfsluftstrom frei, der dann das Hauptventil betätigt. Manchmal gibt es für die Steuerluft einen getrennten Anschluss, der in der Praxis aber mit dem Hauptluftanschluss zusammengelegt werden darf. Wichtig: Vorgesteuerte Ventile arbeiten nur in jener Durchflussrichtung, die im Schaltsymbol durch einen Pfeil gekennzeichnet ist. Auf dem Ventil selbst ist der Druckluft-Eingang mit 1, 11 oder P gekennzeichnet (siehe auch Kasten *Pneumatik-Symbole*).

Achten Sie bei der Beschaffung darauf, ob das Ventil monostabil oder bistabil arbeitet. Monostabile Ventile lassen nur Luft durch, so lange eine Steuerspannung anliegt, während bistabile Ventile ihren letzten Schaltzustand beibehalten; hier genügt ein kurzer Steuerimpuls. Bistabile Varianten sind beim 5/2-Wege-Ventil häufig anzutreffen; eine kurze Ansteuerung (100ms sind völlig ausreichend) wirft dann den Aktor in die eine oder andere Endstellung. Als Ansteuerspannung für die Magnetventile benötigt man gemeinhin 24V Gleichstrom, größere Einzelventile können auch mit 230V-Spulen ausgerüstet sein.

Ventilinseln

Vorgesteuerte Ventile fasst man gerne zu sogenannten *Ventilinseln* zusammen, wenn gleich mehrere Aktoren unabhängig voneinander elektrisch betätigt werden sollen. Ventilinseln sind modular aufgebaut: Auf den Endplatten findet man die zu einer Art „Bus-System" zusammengelegten Versorgungs- und Entlüftungsanschlüsse, die einzelnen Ventilplatten enthalten die Magnetventile und Anschlüsse für die Aktoren.

Üblich sind Ventilinseln für vier bis acht Aktoren; bei *eBay* werden sie oft günstig angeboten, wenn sie mit einem (für Maker-Zwecke wenig brauchbaren) *Feldbus*-Modul ausgestattet sind. Das Feldbus-Modul lässt sich aber recht leicht entfernen, die Magnetanschlüsse sind dann einzeln zugänglich (siehe Bild). Beliebt sind die CPV-Ventilinseln von Festo, es gibt sie mit 10 und 14mm breiten Ventilmodulen; letztere bieten mehr Luftdurchlass. Solange Sie keine lebensgroßen Androiden bauen möchten, sind die kompakten CPV-10-Inseln für Maker-Zwecke völlig ausreichend.

Ansonsten geschieht der elektrische Anschluss über D-Sub-Steckverbinder, an deren Pins die Spulenanschlüsse einzeln anliegen; die Belegung findet man im Datenblatt des Herstellers. Wenn die Magnetspulen bereits mit einer Freilaufdiode zur Unterdrückung von Abschalt-Impulsspitzen und/oder Anzeige-LEDs ausgestattet sind, ist natürlich auch die angegebene Polung wichtig.

Üblich ist auch hier eine Spulenspannung von 24V; die Stromaufnahme ist mit 0,5W oder rund 20mA pro Spule beim CPV-10 recht gering, die größeren CPV-14-Ventile benötigen 0,7W oder knapp 30mA. Die Ansteuerung beispielsweise von einem Arduino kann also über die billigen Treiber-ICs ULN2003 (sieben Treiber) oder ULN2803 (acht Treiber) erfolgen, die ausreichend spannungsfest sind. Die kleinen Ventile in den Ventilinseln arbeiten ziemlich flott: Schaltzeiten von 10 bis 20ms sind die Regel.

Achten Sie bei einer Beschaffung von Gebrauchtteilen auf die Bestückung: Ventilinseln können durchaus auch gemischt mit verschiedenen Ventilen und funktionslosen Blindplatten ausgestattet sein. Unbenutzte Luftausgänge von 5/2-Ventilen kann man gegebenenfalls mit einem Verschlussstopfen blockieren, da hier sonst ungehindert Luft austritt.

Aktoren

Kommen wir endlich zum interessanten Teil: Zylinder, Linearschlitten, Schwenkantriebe und Greifer gehören zu den Aktoren, die einen Luftstrom in eine Bewegung umsetzen. Zu den Arbeitspferden der Pneumatik gehören doppeltwirkende Zylinder mit Kolbenstange, die im Prinzip wie die Antriebszylinder einer Dampflok funktionieren: Ein Kolben kann von beiden Seiten mit Druckluft beaufschlagt werden und bewegt sich dann vorwärts oder zurück. Seltener sind einfachwirkende Ausführungen, bei denen nach Entlüftung eine Feder den Kolben in die Ausgangsposition zurückholt.

Schon kleine Zylinder können beachtliche Kräfte entwickeln: Ein Kolben mit 40mm Durchmesser übt bei 6 Bar Druck eine Kraft von 754N aus, was ausreicht, um einen Finger schmerzhaft zu quetschen. Die entwickelte Kraft lässt sich recht einfach über die Kolbenfläche in Quadratzentimetern ($\pi \cdot d^2 / 4$, also etwa $0{,}785 \cdot d^2$ mit d = Durchmesser) multipliziert mit dem Nenndruck in Bar berechnen. Der Druckluftbedarf fällt nur bei größeren Zylindern und dauernder Bewegung nennens-

wert ins Gewicht, er berechnet sich aus der Kolbenfläche multipliziert mit dem Hub. Der genannte 40mm-Zylinder verbraucht für einen 100mm-Hub also bescheidene 125 Kubikzentimeter Druckluft.

Die Zylinder gibt es auch in Kombination mit einem Linearschlitten als kompakte Einheit; um die Bauhöhe gering zu halten, verwendet man hier Doppelzylinder. Die auf sehr präzisen Linearkugellagern laufenden Pneumatik-Schlitten sind als Neuteil sehr teuer, selbst gebraucht sind sie kaum unter 50 Euro zu bekommen. Auch Schwenkantriebe und Greifer sind ab Werk unerschwinglich. Schwenkantriebe führen eine winkelbegrenzte Drehbewegung aus und verlangen wie die Linearschlitten und Greifer eine aufwendige und teure Fertigung.

Saugdüsen

Vakuumsaugdüsen sind Vorrichtungen, die mit einer *Venturi-Düse* aus einem Druckluftstrom einen Unterdruck erzeugen; es gibt sie als Stand-Alone-Bauteil, aber auch als Modul für Ventilinseln, in Kombination mit einem Magnetventil. Bei Bestückungs- und Verpackungsmaschinen sind sie häufig anzutreffen, um mittels Sauger Gegenstände anzuheben und abzulegen. Saugdüsen erreichen trotz des einfachen Prinzips einen Unterdruck bis auf 15 Prozent des Außendrucks.

Da sie eingeschaltet einen ständigen Luftstrom zur Funktion benötigen, ist ihr Luftverbrauch relativ hoch; man muss mit 30 bis 40 Litern pro Minute rechnen. Modelle mit *Abwurfimpuls* geben beim Abschalten des Unterdrucks einen kurzen Überdruckimpuls auf die Saugleitung, sodass ein Sauger das angehobene Gut (zum Beispiel ein SMD-Bauteil bei einem Bestückungsautomaten) auch wieder zuverlässig abwirft.

Logikfunktionen

Tatsächlich gibt es auch in der Pneumatik Bauteile, die eine Logik-Funktion ausführen: Ein UND-Ventil gibt nur dann einen Luftstrom frei, wenn an beiden Eingängen Überdruck ansteht. Schon ein einfaches T-Stück ist natürlich eine Art ODER-Gatter, denn am Ausgang erscheint Druckluft, sobald an einem der Eingänge Überdruck herrscht. Echte ODER-Ventile sind allerdings mit Rückschlagklappen ausgerüstet, damit die Druckluft auch wirklich nur am Ausgang ankommt und nicht etwa durch den anderen Eingang entweicht, wenn dort kein „Signal" anliegt. Die Bauteile sind kaum größer als die Schlauch-Steckverbinder selbst.

Inverter kann man mit pneumatisch angesteuerten Mehrwege-Ventilen realisieren: Statt einer Magnetspule mit Anker ist hier ein kleiner pneumatischer Aktor eingebaut. Ein solches „pneumatisches Relais" kann bei entsprechender Beschaltung nur dann Druckluft freigeben, wenn am Eingang *kein* Überdruck herrscht.

Sogar Zeitglieder sind pneumatisch realisierbar: Im Lieferprogramm von Festo finden sich Exoten wie ein Verzögerungsventil, das erst nach einstellbarer Zeitdauer den Durchfluss freigibt, oder auch ein Kurzimpulsventil, das mit einsetzendem Überdruck am Eingang nur einen kurzen Luftimpuls am Ausgang erzeugt. Mit den genannten Bauteilen lassen sich sogar RS-Flipflops und einfache Automaten und Schaltwerke realisieren, obwohl man komplexere Steuerungsaufgaben natürlich lieber auf die Ansteuerungselektronik auslagern wird – das ist deutlich billiger.

Druckschalter

Damit die ansteuernde Elektronik feststellen kann, ob sich in einem bestimmten Bereich des Systems überhaupt Druckluft befindet, benötigt man Sensoren, die auf Überdruck reagieren. Im einfachsten Fall sind das sogenannte *Druckschalter*, die bei Erreichen eines (einstellbaren) Druckniveaus einen Stromkreis schließen oder öffnen. Druckschalter findet man zum Beispiel in jedem Kompressor, sie schalten den Motor ab, sobald der Kessel-Nenndruck erreicht ist.

Druckschalter zum Einbau in das Pneumatik-System sind deutlich kleiner; im Unterschied zum Kompressor-Druckschalter können sie keine größeren Leistungen schalten. Man verwendet sie gern für Überwachungszwecke – wenn etwa eine Maschine nicht versehentlich ohne vorhandene Druckluft anlaufen darf.

Spezialisten

Zu den aufwendigeren (und dementsprechend teuren) Bauteilen der Pneumatik gehören *Druckverstärker* oder *Druckbooster*, die über eine Doppelkolben-Einheit den Betriebsdruck vervielfachen können. Im Unterschied zu den Saugdüsen arbeiten sie nicht verschleißfrei. Man benötigt sie manchmal, wenn ein Aktor bei begrenztem Zylinderdurchmesser durch Verdopplung des Drucks (und damit auch des Luftverbrauchs!) eine höhere Kraft liefern soll.

Druckluftmotoren sind das pneumatische Pendant zum Elektromotor; man kennt sie zum Beispiel aus der Autowerkstatt, wo sie als Druckluft-Schlagschrauber zum Lösen und Befestigen der Radbolzen dienen. Druckluftmotoren können als Kolben- oder Lamellenmotor ausgeführt sein, letztere ähneln vom Aufbau her einem Wankelmotor. Der Gesamt-Wirkungsgrad ist eher bescheiden und der Luftverbrauch hoch; ein 180W-Motor konsumiert etwa 330 Liter Druckluft pro Minute. Druckluftmotoren benötigen zur Schmierung einen Ölnebel in der Druckluft.

Druckschalter öffnen oder schließen einen Stromkreis, sobald der eingestellte Nenndruck erreicht ist; hier eine Ausführung mit Würfelstecker-Anschluss.

Neben den Mehrwege-Ventilen, die den Luftstrom ganz oder gar nicht durchlassen, gibt es noch sogenannte *Proportionalventile*. Das sind Sonderausführungen elektrisch angesteuerter Ventile, wie sie beispielsweise für Dosierungsaufgaben gebraucht werden. Im Prinzip arbeiten sie wie ein Druckregler, nur dass der ausgangsseitige Druck nicht manuell eingestellt wird, sondern von einer analogen Steuerspannung abhängt. Als Spezialisten sind sie relativ selten und teuer. In Verbindung mit einem Istwert-Geber (z.B. Linear-Potentiometer) und einem Regelkreis ermöglichen sie aber einen lastunabhängigen „Teilausschlag" eines Zylinders bis zu einer beliebigen vorgegebenen Position.

Manchmal wird der genaue Messwert eines Über- oder Unterdrucks verlangt – zum Beispiel, wenn man feststellen muss, ob ein Sauger tatsächlich einen Gegenstand aufgenommen hat oder nicht. Dann benötigt man Drucksensoren, die eine Ausgangsspannung (oder auch einen Strom) analog zum anliegenden Absolutdruck liefern.

Damit hätten wir die wichtigsten pneumatischen Bauteile auch schon vorgestellt; wie in der Elektronik finden sich am Markt von jedem Bauteil unzählige Versionen verschiedener Leistungsklassen, wobei sich unser Artikel natürlich auf die für Maker interessanten Ausführungen beschränkte. Für einen Einstieg und als Ideenlieferant für eigene Projekte dürften die vermittelten Grundlagen indes reichen. —*cm*

Anwendung eines Druckluft-Motors in einer kleinen Bohrmaschine.

Photovoltaik an der E-Auto-Wallbox – reloaded

Unser Beitrag über das sogenannte Überschussladen im Herbst letzten Jahres hatte eine erfreulich große Resonanz – so groß, dass sich der Autor nochmals Gedanken über eine einfachere und billigere Lösung gemacht hat. Mit dem neuen Bauvorschlag erspart man sich sogar Eingriffe in die Elektro-Installation.

von Uwe Rohne

In der Make-Ausgabe 5/21 haben wir beschrieben, wie eine Wallbox das E-Auto immer genau mit der Leistung lädt, die die Photovoltaik-Anlage (PV) zur Verfügung stellt, das sogenannte Überschussladen. Die jeweils aktuelle Leistung haben wir dabei unabhängig vom verwendeten Wechselrichter der PV mit dem Energie-Messmodul *Shelly 3em* gemessen. Die Vorteile des Shelly-Systems sind die Genauigkeit, die Datenübertragung via WLAN und die freie Bestimmung des Messpunktes, in unserem Fall also hinter der PV und dem Haushalt, aber vor dem Anschlusspunkt der Wallbox.

Nachteile sind der Preis von rund 100 Euro für den Shelly-Sensor und die notwendige Installation über einen Fachbetrieb; da können schon mal zwei Stunden Arbeit zusammenkommen. Um das Geld dafür zu sparen, wäre also eine Lösung perfekt, die ohne ein spezielles Messsystem auskommt. Genau diesen Lösungsansatz möchten wir mit diesem Artikel vorstellen.

Dabei ist die Lösung so simpel, dass man erst einmal darauf kommen muss: In jedem Zählerschrank ist ein Energiemesser (umgangssprachlich Zähler) vom Netzversorger verbaut. Bei Einsatz von PV-Anlagen sind es sogenannte digitale Zweirichtungszähler. Diese Zähler zeigen nicht nur die Energieeinspeisung und den Verbrauch an, sondern bei aktuellen Systemen auch die momentane Leistung.

Die Verbrauchswerte können nun auch über die drahtgebundene *S0-Schnittstelle* (nicht zu verwechseln mit ISDN) und alternativ über eine Infrarot-Sende-/Empfangseinheit namens D0 ausgelesen werden. Ist der Zähler bereits verbaut, kommt man an die S0-Schnittstelle ohne Zerstörung einer Plombe nicht mehr heran. Es bleibt also die IR-Schnittstelle. Einfache elektronische Zähler „funken" ihre Daten auch über blinkende LEDs; unter den Zweirichtungs-Zählern sind sie allerdings nicht zu finden.

Bei der IR-Übertragung gibt es leider unterschiedliche Protokolle und Übertragungsgeschwindigkeiten. Verwendet werden textbasierte Protokolle nach IEC 62056-21 und SML (Small Message Language), wobei die SML-Daten meist zyklisch gesendet werden. Bei einigen Modellen muss die Ausgabe der Werte explizit angefordert werden, dabei werden Halbduplex-Asynchron-Verfahren mit Geschwindigkeiten von 300 bis 9600 Baud und 7 oder 8 Bits pro Byte benutzt.

Digitaler Logarex-Zähler mit aufgesetztem IR-Schreiblesekopf

Der Metallring am Zähler zum Aufsetzen des Sensors. Deutlich sind die Empfangs- und Sendediode zu erkennen.

Kurzinfo

- **Energie-Überschuss einer Photovoltaik-Anlage über den Zählerstand bestimmen**
- **Auslesen moderner Energiezähler mit Infrarot-Schnittstelle**
- **Wallbox-Ladestrom mit ESP8266 drahtlos einstellen**

Checkliste

Zeitaufwand:
4 Stunden

Kosten:
50 Euro

Löten:
Bestückung einer einseitigen Platine

Maschinen:
Akkuschrauber, Heißklebepistole

Material

- **ESP8266-Modul** WEMOS D1 Mini
- **Elektronische Bauteile** laut Stückliste (siehe Link)
- **Platine** gemäß Layout (siehe Link)
- **Micro-SD-Karte** mit Adapter
- **PET-Flaschenverschluss**
- **Neodym-Ringmagnet** R-27-16-05-N
- **Aufputz-Verteilerdose** als Gehäuse

Mehr zum Thema

- Uwe Rohne, Photovoltaik an der E-Auto-Wallbox, Make 5/21, S. 20

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

Zähler-Telegramm

```
/LOG5LK1XXXXXXXXX<\r><\n>
<\r><\n>
1-0:96.1.0*255(001LOG0XXXXXXXX)<\r><\n>
1-0:1.8.0*255(000471.6452*kWh)<\r><\n>
1-0:2.8.0*255(000528.9178*kWh)<\r><\n>
1-0:0.2.0*255(ver.03,432F,20170504)<\r><\n>
1-0:96.90.2*255(0F66)<\r><\n>
1-0:97.97.0*255(00000000)<\r><\n>
!<\r><\n>
```

Typisches Datentelegramm eines Logarex-Zählers, 1.8.0 = Verbrauch, 2.8.0 = Einspeisung. Diese so genannten OBIS-Kennzahlen sind internationaler Standard und in der Norm IEC 62056-61 festgeschrieben.

Leistungsberechnung über Zählerstand

Die momentan dem Lichtnetz entnommene (oder zugeführte) Leistung lässt sich durch Differentation der Zählerstände über die Zeit berechnen. Klingt kompliziert, ist aber ganz einfach: Der Zählerstand wird in regelmäßigen Abständen ausgelesen, die Differenz aus den Ablesungen (in kWh) wird dann durch das Zeitintervall (in h) geteilt – man erhält somit den Verbrauch in kW.

$$\frac{(\text{Zählerstand}_{\text{neu}} - \text{Zählerstand}_{\text{alt}})\ \text{kWh}}{\text{Zeit in h}} = \text{Leistung in kWh}$$

Da 1h = 3600 Sekunden = 3600 × 1000 Millisekunden sind, ergibt sich

$$\frac{(\text{Zählerstand}_{\text{neu}} - \text{Zählerstand}_{\text{alt}}) \times 1000\text{W} \times 3600 \times 1000\ \text{Millisekunden}}{\text{Zeit in Millisekunden}} = \text{Leistung in W}$$

Weil die Wallbox ohnehin nur in Schritten von 1 Ampere oder 230W eingestellt werden kann, ist die Genauigkeit des Verfahrens ausreichend. Bei Zählern mit nur drei Nachkommastellen sollte der Messzyklus allerdings mindestens 50 Sekunden betragen.

Gängig ist ebenso, die Verbrauchswerte ganz simpel durch eine Impulsanzahl zu kodieren; dann ändert sich das Blinken der IR-LED deutlich mit dem aktuellen Verbrauch. Ältere Digitalzähler geben 500 oder 1000 kurze IR-Lichtimpulse je kWh ab, vergleichbar mit der roten Markierung auf der silbernen Scheibe bei den alten elektromechanischen Ferraris-Zählern.

IR-Test

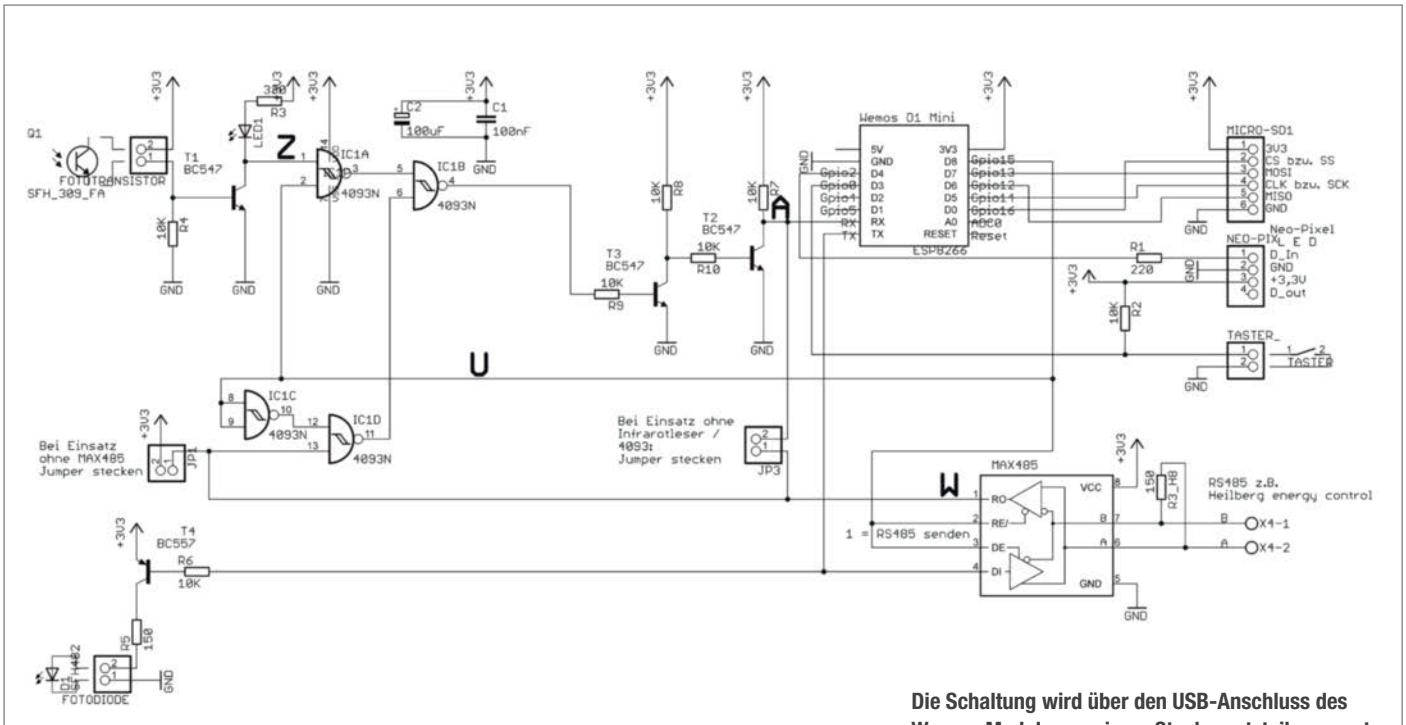
Das menschliche Auge kann infrarotes Licht nicht sehen. Es gibt aber einen einfachen Trick, infrarote Strahlung zu erkennen: Wenn man eine Smartphone-Kamera auf eine Infrarotlichtquelle hält, ist diese auf dem Display zu sehen. Probieren Sie es einmal mit einer IR-Fernbedienung aus, bevor sie zum Zählerschrank gehen.

Ob es sich um die Lichtimpuls-Kodierung handelt oder um ein zyklisch versandtes Datenpaket, lässt sich testen, indem man entweder alle Verbraucher ausschaltet – zum Beispiel über den FI-Schalter – oder indem man einen leistungsstarken Verbraucher (z.B. Fön, Staubsauger, Heizlüfter) einschaltet. Ändert sich die Dauer des IR-Lichtimpulses merklich, zeigt der Zähler analog zur roten Markierung der Ferraris-Zähler an. Bei dieser Art von Zählern muss der aktuelle Stand explizit mit einem kurzen Datentelegramm angefordert werden.

Bleibt hingegen der Rhythmus identisch, spricht sehr viel dafür, dass der Zähler zyklisch SML-Klartext-Daten sendet. Sollte gar nichts zu sehen sein, kann es daran liegen, dass der Zähler noch mit einer PIN geschützt ist. Dieses wird aus Datenschutzgründen gemacht, da die Zähler oft frei zugänglich verbaut werden. Seine PIN erhält man mit einer kurzen Anleitung vom Netzbetreiber.

Ein typischer Vertreter moderner E-Zähler ist das weit verbreitete Modell der tschechischen Firma Logarex (siehe Bild). Der Zähler sendet jede Sekunde seine zwei Zählerstände (Einspeisung und Verbrauch) mit einer Genauigkeit von vier Stellen hinter dem Komma. Wenn man diese Zähler im Sekundenabstand auswertet, passiert gerade bei kleinen Leistungen nicht viel. Erhöht man jedoch den Abstand auf 10 Sekunden, kann man die gemittelte Leistung mit einer Abweichung von maximal 35 Watt errechnen. Das ist das, was wir brauchen.

Um nun eine Wallbox zu steuern, müssen wir mehr tun – denn in dem Moment, in dem wir die Wallbox aktivieren und das Aufladen

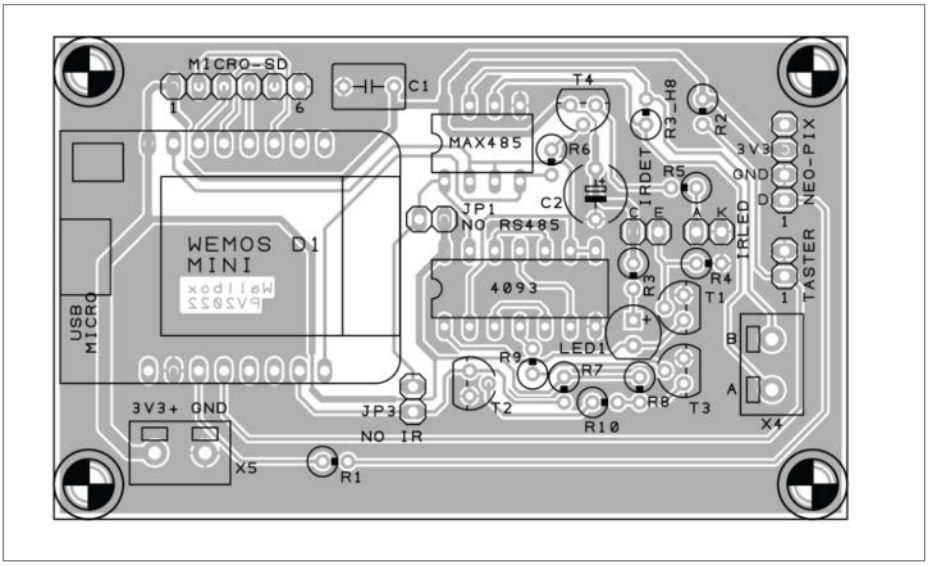


Die Schaltung wird über den USB-Anschluss des Wemos-Moduls aus einem Steckernetzteil versorgt.

des E-Autos beginnt, läuft dieser Verbrauch ja auch über den Zähler. Wir können nicht unterscheiden, ob der Durchlauferhitzer, der Herd oder eben die Wallbox die aus den Datentelegrammen errechnete Leistung zieht. Jetzt greifen wir zum zweiten Trick: Unsere Wallbox kann ja die jeweils aktuelle Leistung auf Anforderung unserer Steuerung mitteilen. Jetzt müssen wir noch von der über den Zählerstand ermittelten Leistung jene abziehen, die uns die Wallbox mitteilt, und schon ist unsere Steuerung fertig – zumindest in der Theorie.

Unser Bauvorschlag verwendet als Zähler des Netzbetreibers das weit verbreitete Modell *Logarex LG13BE* und die Wallbox *go-eCharger* oder *Heidelberg energy control*; andere Zähler sind wahrscheinlich ebenso gut geeignet, erfordern aber unter Umständen kleine Anpassungen der ESP8266-Firmware. Zu tun bleibt der Bau eines Schreib-/Lesekopfes und das Programmieren einer Steuerung mit dem ESP8266.

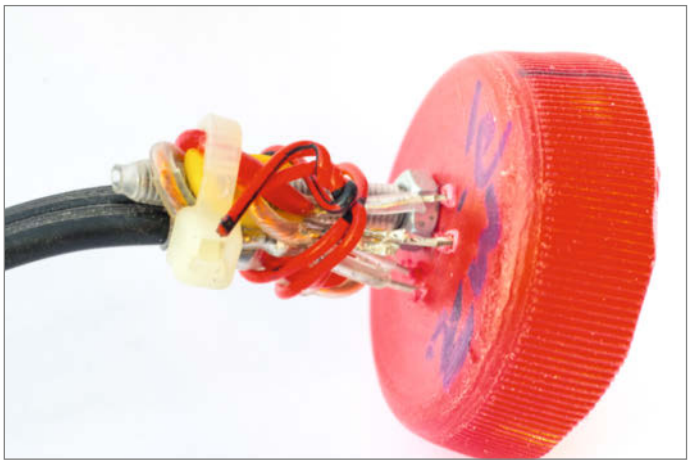
Rund um die IR-Empfangs- und Sende-LED am Zähler befindet sich ein Metallring, an den normalerweise kommerzielle Lesegeräte magnetisch angeheftet werden. Um unsere selbstgebaute IR-Einheit zu fixieren, verwenden wir einen runden Neodym-Magneten



Bestückungsplan: Um kompakte Abmessungen zu erreichen, werden die Widerstände stehend montiert. Der MAX485 muss nur für die Heidelberg-Wallbox bestückt werden (siehe Text).

R-27-16-05-N, wobei 27 den Außendurchmesser kennzeichnet und 16 den Innendurchmesser. Als improvisiertes Gehäuse haben wir den Drehverschluss einer Getränkeflasche verwendet. In den Deckel haben wir neben vier 1mm-Löchern für die Dioden- und Fototransistoranschlüsse in der Mitte ein weiteres 3mm-Loch gebohrt, um hier eine lange M3-

Unser Sensor besteht aus IR-Fototransistor, IR-LED und einem Ringmagneten in einem Getränkeverschluss, vergossen mit Heißkleber. Die Bauteile ragen noch minimal aus der Klebemasse heraus.



Die Beinchen der Bauteile werden durch kleine Löcher im Deckel gesteckt und auf der Rückseite mit dem Anschlusskabel verlötet. Eine Isolierung mit Schrumpfschlauch ist ratsam.

Schraube als Zugentlastung zu montieren. Nachdem die Dioden eingesetzt sind, werden diese (wie auch die Schraube und der Neodym-Magnet) mit einer Heißklebepistole oder auch Silikon-Dichtmasse im Deckel fixiert. Achten Sie darauf, dass die Diode und der Fototransistor noch „sehen" können.

Die Schaltung

Es erreichten uns einige Zuschriften mit Fragen zur Stromversorgung der Schaltung aus Make 5/21. Wir nutzen für die benötigten 3,3V den Wemos D1 Mini, der ja über eine Micro-USB-Schnittstelle mit 5V versorgt wird und einen Wandler auf 3,3V auf der Platine hat. Dies gilt auch für die hier vorgestellte neue Schaltung; wir haben der Platine zusätzlich einen 3,3-Volt-Eingang (X5) spendiert, der alternativ mit einem 3,3V-Schaltnetzteilmodul (2 bis 3 Watt Leistung sind ausreichend) verwendet werden kann.

Werfen wir einen Blick auf das Schaltbild: Der Transistor T1 verstärkt das IR-Signal. Ein nachgeschalteter Schmitt-Trigger (4093) sorgt für ein sauberes Ausgangssignal. Der Zustand *kein Licht* entspricht hierbei dem Stopp-Pegel und vorhandenes IR-Licht somit dem Start-Pegel. Um den Seriell-Eingang auch für andere Quellen nutzen zu können, ist eine Logik zum Abschalten vorgesehen. Diese Logik ist bei der Verwendung der Heidelberg-Wallbox nötig, da die Soft-Serial-Schnittstelle des ESP8266 nur mit einem Pin funktioniert, der hier schon durch die SD-Karte belegt ist. Leider schafft es der Ausgang des Bausteins 4093 nicht, den Rx-Pin zuverlässig auf Low-Pegel zu ziehen, da er gegen einen 470-Ohm-Widerstand am USB-Empfangsbaustein des Wemos-Moduls arbeiten muss. Daher haben wir die zwei Transistoren T2 und T3 als zusätzliche Treiberstufe spendieren müssen.

Zusammenbau der Platine

Die Platine ist einseitig ausgeführt und kommt sogar ohne Drahtbrücken aus. Die benötigten Teile haben wir in der Stückliste aufgeführt, die Sie (wie die Firmware-Sourcen auch) unter dem Link im Kurzinfo-Kasten finden. Soll die Steuerung mit einem go-eCharger betrieben werden, braucht der Transceiver-Baustein MAX485 nicht bestückt zu werden. In diesem Fall ist JP1 zu stecken (oder durch eine eingelötete Drahtbrücke zu ersetzen). Die Infrarot-Sendediode D1 und der davor geschaltete Transistor T4 wiederum wird bei einem Logarex-Zähler nicht benötigt, da dieser die Datentelegramme ohne Anforderung zyklisch ausgibt. Darüber hinaus gelten die Aufbauhinweise aus der Make 5/21.

Ob die IR-Empfangsdiode Lichtsignale empfängt, ist am rhythmischen Flackern der Kontroll-LED auf unserer Platine zu sehen. Für einen ersten Test sollte es relativ dunkel sein oder die IR-Empfangs-LED mit einem Tuch abgedeckt werden. Jetzt die LED vor eine Schreibtischlampe halten: Die Kontroll-LED sollte jetzt leuchten und damit den Lichtempfang bestätigen. Jetzt den Test mit einer IR-Fernbedienung bei gedämpften Licht wiederholen. Klappt auch das, ist es Zeit, das Testprogramm (*Infrarotkopf_pruefprogramm.ino*) auf den ESP8266 zu laden. Den Flashvorgang des Prozessors haben wir bereits beim vorangegangenen Artikel in der Make 5/21 ab Seite 26 beschrieben. Jetzt müssen wir unseren IR-Adapter am Zähler mittels Magneten befestigen. Mit dem Smartphone haben wir ja zuvor gesehen, wo genau die Sende-LED innerhalb des Metallringes im Zähler sitzt.

Mit einem seriellem Terminalprogramm, zum Beispiel dem Monitor der Arduino-IDE, sehen wir nach der Auswahl der verwendeten COM/USB-Schnittstelle und dem Einstellen der Geschwindigkeit von 9600 Baud jetzt die Datentelegramme im Klartext. Die Daten der Anzeigezeile *1.8.0* entsprechen dem Verbrauch, die Daten der Zeile *2.8.0* der Einspeisung. Alternativ zur Arduino-IDE können ebenso die kostenlosen Terminalprogramme *HTerm* oder *Teraterm* zum Einsatz kommen. Auch hier sind natürlich die COM-Schnittstelle und die Geschwindigkeit einzustellen, außerdem 8 Datenbits, keine Parität und ein Stoppbit (9600 8N1). Sollte alles soweit funktionieren, hat unser Prüfprogramm ausgedient und wir können die eigentliche Software (*pv_charger_ohne_shelly.ino*) flashen.

Zu guter Letzt

Die Bedienung, die Beschreibung des SD-Kartenhalters, den Inhalt der auf der Karte abgelegten *config.txt* sowie Hinweise zur Fehlersuche entnehmen Sie bitte dem ersten Teil aus der Make 5/21. Leider stand uns für diesen zweiten Teil unseres Walbox-Hacks keine Heidelberg-Box zur Verfügung, wir sind uns aber ziemlich sicher, dass auch die Heidelberg-Box mit der Schaltung funktioniert und sind auf die Erfahrungen unserer Leser gespannt.

Und auch für diesen Bauvorschlag gilt nachfolgender Hinweis: Die Zahl und die Varianten der am Markt erhältlichen E-Autos nimmt ständig zu. Ob die von uns skizzierte Lösung möglicherweise den Vorgaben des Herstellers widerspricht, können wir in diesem Artikel nicht bewerten. Ziehen sie also im Zweifelsfall die Bedienungsanleitung ihres Autos und natürlich der Wallbox zu Rate. Für den Fall, dass das Protokoll Ihres Zählers anders aufgebaut ist, finden Sie in den Sourcen entsprechende Hinweise; unser Prüfprogramm hilft dann bei der Analyse der Zähler-Daten. —*cm*

heise Academy

JETZT KOSTENLOS TESTEN

DIE NEUE LERNPLATTFORM FÜR IT-PROFESSIONALS

Wir machen IT-Weiterbildung digital

IT-Kurse aus der Praxis
Lerne in Online-Kursen und -Trainings, wie Techniken funktionieren und wie du Aufgaben löst.

Triff erfahrene IT-Experten
Profitiere von der Erfahrung unserer IT-Experten und hole dir hilfreiches Praxiswissen aus erster Hand.

Lerne, wie es für dich passt
Nutze das Kursangebot überall und auf jedem Gerät und lerne immer dann, wenn du es brauchst.

Übungen zum Ausprobieren
Probiere das gelernte Wissen selbst aus – mit Beispielaufgaben, Coding-Segmenten und Praxisübungen.

Überprüfe dein neues Wissen
Teste das Gelernte mit interaktiven Quizzes und löse die Programmieraufgaben deiner Trainer spielerisch.

Individuelle Lernumgebung
Lerne in deinem eigenen Tempo, inklusive Notizen, Transkript und Fragen-Modul.

Hier geht's zu deiner Weiterbildung: **heise-academy.de**

Bordnetz: Strom Ahoi!

Es ist praktisch, Strom in Form einer Akkubox an Bord zu haben: sei es auf dem Boot, im Auto oder für die Kühlbox auf der Fahrradtour. Wir stellen hier eine günstige, sichere und einfach nachbaubare Variante vor.

von Alexander Moser

Vor kurzem hat der Autor dieser Zeilen ein sündhaft teures Segelboot erworben. Es war nicht zuletzt deswegen so teuer, weil es aufblasbar ist und auf engstem Raum in einem Kleinwagen Platz findet (ein Smart!). Für das neue Hobby wäre nun noch ein kleiner Elektromotor praktisch. Er hilft dem frisch gebackenen Matrosen bei der Fahrt aus dem Hafen, im Kanal oder bei Flaute. Für die Stromversorgung muss also eine aufladbare 12V Batterie her, die möglichst in einer wasserdichten Box, ebenfalls auf kleinstem Raum, untergebracht werden muss.

Da das Boot nun schon so viel gekostet hat, muss diese Lösung so günstig wie möglich sein. Die Box soll also eher in der Region um die 100 Euro liegen. Nach Möglichkeit sollte der Akku über ein Solarpanel aufgeladen werden können. Dafür kann man etwa die Solarpaneele eines Wohnmobils nutzen. Ein Solar-Laderegler soll also die Aufladung über Paneele oder eine andere Stromquelle übernehmen. Der verwendete Solar-Laderegler bietet zwei USB-Ausgänge und kostet nur rund 34 Euro. Als Ladebuchse käme hier ein sogenannter *SAE*-Anschluss (1) in Frage, der auch in Wohnmobilen verwendet wird. Je nach Anwendungszweck (s.u.) kann aber auch ein anderes System verwendet werden.

Weitere Ausgänge, wie eine KFZ-Steckdose und USB-Buchsen, wären wünschenswert. So kann die Box auch für andere Anwendungen genutzt werden um z. B. an Land eine 12V-Wasserpumpe für einen Druckreiniger zu betreiben, oder einen Kompressor, um ein Schlauchboot oder die Luftmatratze aufzupusten. Die USB-Buchsen und die 12V-KFZ-Steckdose laden Smartphones und versorgen Aktivboxen etwa auf dem Bollerwagen. Eine Kühlbox wird an 12V angeschlossen und mit einem Wechselrichter für die KFZ-Buchse kann man sogar 230V-Wechselstrom bereitstellen, auch wenn man diese Möglichkeit nur im Notfall wählen sollte, falls es keine Möglichkeit gibt, die 12 oder 5 Volt zu benutzen.

Die Batterie soll zusätzlich über eine Sicherung mit Notausschalter abgesichert werden. Damit das Ganze in der Box Halt findet und nicht verrutscht, wurde ein Halter für den 3D-Drucker in CAD erstellt, der alternativ auch aus ein paar Holzleisten, Brettchen oder Hartschaum gebaut werden kann.

Als Gehäuse für die Apparatur eignet sich eine preiswerte transparente Kunststoffbox aus dem Baumarkt. Sie ist recht kompakt und kann einen zyklenfesten Standard-Blei-Gel-Akku mit 12V und 12Ah (150mm × 98mm × 95mm, 3,3kg) aufnehmen. Diese Akku-Technik hat keinen so hohen Energiegehalt wie Lithium-Akkus, ist aber bewährt und sicherer

Kurzinfo

» **Flexible Stromversorgung für unterwegs**
» **Hochstromverkabelung im Griff**
» **Lade- und Sicherheitstechnik**

Checkliste

Zeitaufwand:
etwa 4 Stunden

Kosten:
ca. 110 Euro

3D-Druck:
optional

Löten:
Lötkolben ab 60W oder Lötstation

Material

» **Bleigel-Akku** 12V, 12Ah
» **Solar-Laderegler** Eco Worthy
» **Sicherung** mit Not-Aus-Schalter
» **Solarpaneleingang** SAE-Buchse
» **Adapterkabel** von MC4 zu SAE
» **Buchse** für 12V-KFZ-Ausgang
» **Hochstrombuchse** für 12V-Außenbordmotor (je nach Einsatzgebiet der Box)
» **2-adriges Kabel** 2–3m, 2,5mm²
» **Schrumpfschlauch**
» **Box mit Deckel** etwa Dirk XXS 4,5l von Hornbach
» **Doppelklebeband**
» **Akkuhalter** 3D-Druck oder gebastelt
» **Flachstecker und Gabelkabelschuhe** oder löten

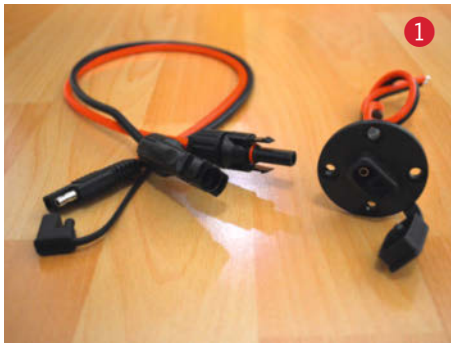
Werkzeug

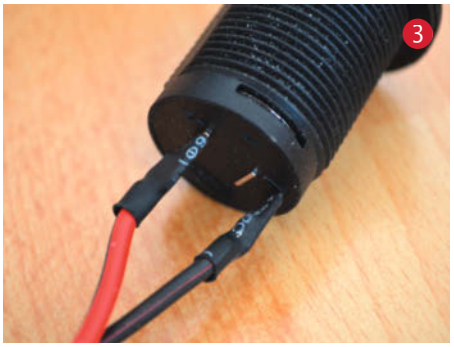
» **Bohrmaschine** oder Akkuschrauber
» **Lochsägen** Durchmesser passend zu den Durchführungen
» **Schraubendreher**
» **Seitenschneider**
» **Abisolierwerkzeug**
» **Lötequipment** für dicke Kabel (ab 60W)
» **Crimpzange** wenn Quetschverbinder verwendet werden

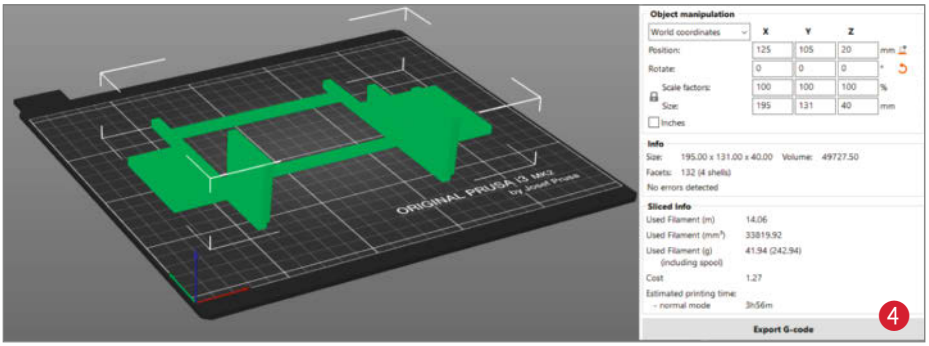
Mehr zum Thema

» Photovoltaik auf dem Balkon, Sebastian Müller, Make 1/22, S. 38
» KRITIS Powerbox, Daniel Jedecke, Make 5/21, S. 72
» E-Bike mit Akkuschrauberantrieb, Christian und Finn Koubek, Make 1/20, S. 80
» Video: Bau und Anwendung der Akkubox

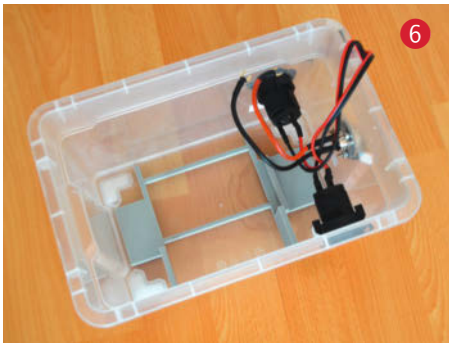
Alles zum Artikel im Web unter
make-magazin.de/xppq

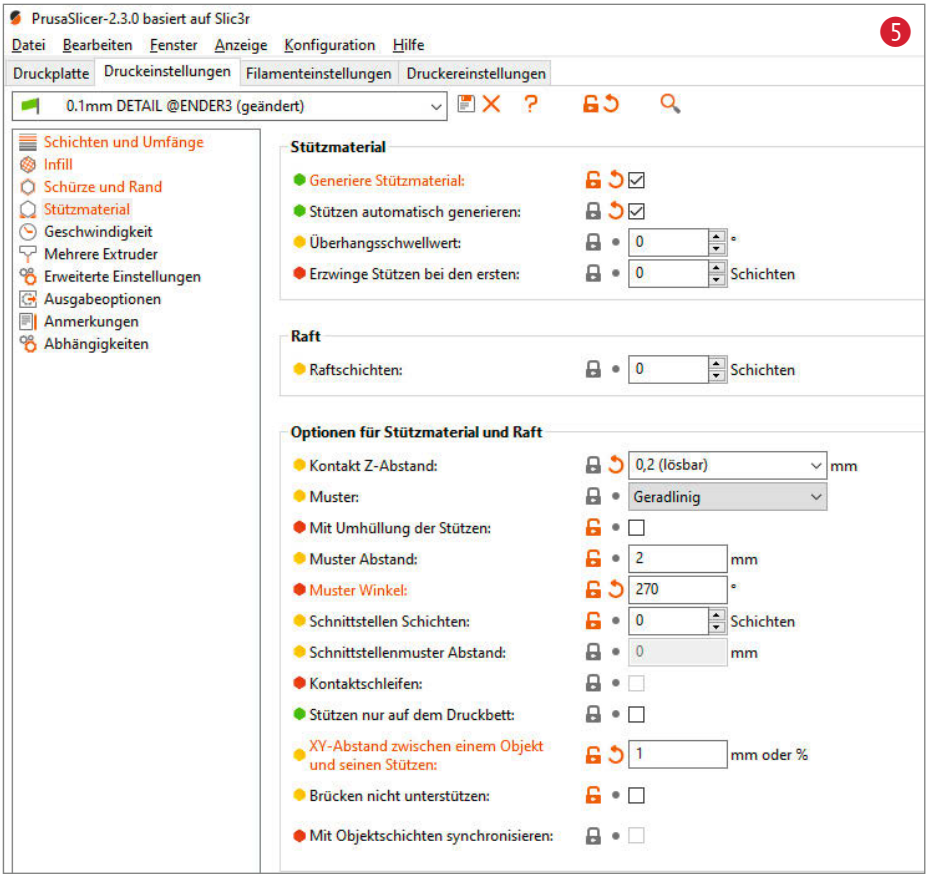




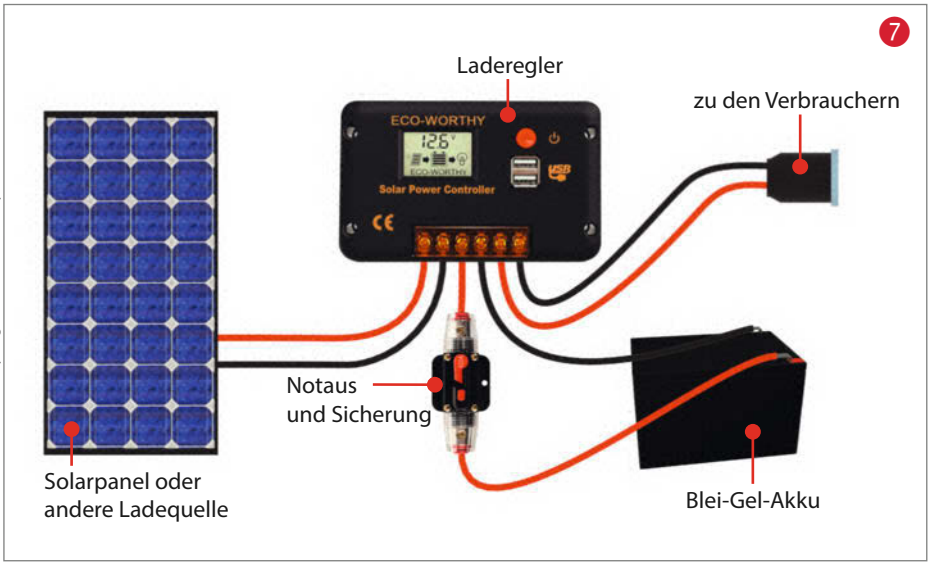


4


6



5



7

Schalter RED WOLF, EcoWorthy Regler, Solar Textur tonytextures.com

im alltäglichen Gebrauch. Da die Wände der Box flexibel sind, können die Knöpfe des Ladereglers und des Notausschalters auch im geschlossenen Zustand betätigt werden.

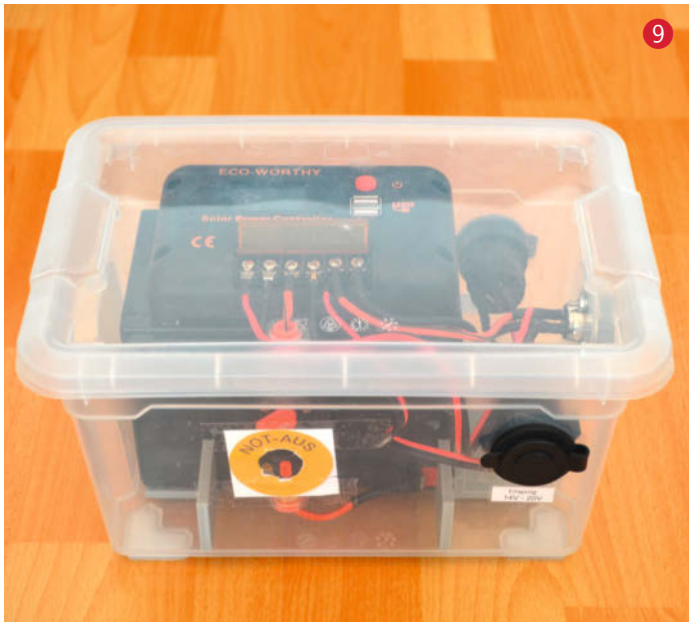
In die Box werden zunächst die Löcher für die Anschlüsse gebohrt. Auf einer der kurzen Seiten zeichnet man hier zunächst ein Kreuz in der Höhe von 8cm vom Boden und 7cm vom Rand aus an. Die Markierung dient als Mittelpunkt für den Bohrer einer Lochsäge. Das Loch ist für den Anschluss zum Außenbord-Elektromotor gedacht und muss zu dem Einbaumaß des 2-poligen Hochstrom-Einbausteckers passen, meist 19mm. Die Mittelpunkte für den (Solar-)Ladeanschluss und die KFZ-Steckdose werden gegenüberliegend auf der langen Seite angebracht. Das Loch für den Ladeanschluss sollte bei Verwendung einer *SAE*-Buche ca. 20mm Durchmesser haben. Für die Entnahmeseite über eine KFZ-Steckdose braucht man etwa 30mm Öffnung. Der Mittelpunkt der Bohrung für den Außenbordmotorstecker liegt bei je 4cm Abstand von der Schmalseite und in einer Höhe von 8cm vom Boden aus gemessen. Die Box sollte nun wie in Abbildung 2 aussehen.

Nun können die Kabel an die Buchsen gelötet werden. Zweifarbiges Kabel ist für die richtige und wichtige Zuordnung der Polung besonders angebracht. Der Kabelquerschnitt sollte mindestens 2,5mm² betragen. Für den 2-poligen Hochstromstecker reichen 15cm Kabellänge. Für die Buchsen an den langen Seiten sind 30cm Kabel ausreichend.

Wer die Ausrüstung dafür hat, kann auf die Kabelenden Gabelkabelschuhe und im Falle des Akkus Flachstecker crimpen. Man kann die Kabelenden aber auch verdrillen und in den Klemm-Anschlüssen des Ladereglers festschrauben. Alle Lötstellen und sonstige offene Kontaktmöglichkeiten müssen zur Sicherheit mit Schrumpfschlauch 3 überzogen werden, die hier möglichen Ströme bergen ein erhebliches Gefahrenpotential.

Bevor man jetzt die Buchsen in die Box schraubt, kann man noch feuchtigkeitsgeschützt von innen Aufkleber für die Beschriftung der Buchsen anbringen. Nachdem die Buchsen eingebaut sind, legt man nun den Akkuhalter in die Box. Den Link zu der Druck-

8



9

datei *Boxhalter.stl* finden Sie in der Kurzinfo zum Artikel. Das Akkuhalter-Modell hat eine Grundfläche von 131mm × 195 mm und eine Höhe von 40mm. Somit passt er auf die Druckfläche der meisten 3D-Drucker wie der *Ender-3*-Serie, *Anycubic Mega S* oder dem *Prusa i3* 4. Weiterhin sind in den Links fertige *3MF*-Dateien mit allen Einstellungen für den *PrusaSlicer* und *Cura* zu finden.

Es reicht eine Düse mit 0,4mm bei einer Schichtdicke von 0,2mm und 20% Infill. Unten hat das Model ein paar Aussparungen, um genau in die von mir verwendete Box zu passen. Es empfiehlt sich also, das Modell mit Stützmaterial zu drucken, welches anschließend wieder entfernt werden muss. Das Stützmaterial 5 sollte 1mm XY-Abstand und 0,2mm Z-Abstand vom eigentlichen Objekt haben, damit es nicht zu schwer lösbaren Verbindungen kommt. Um das Fädenziehen zu begrenzen, hat sich ein Wert von 10mm für *Retraction* bei meinen Druckern bewährt, dies ist aber sehr modellabhängig und muss für Ihren Drucker eventuell anders gewählt werden.

Nachdem die Buchsen eingebaut sind, kann man nun schon mal den Akkuhalter 6 in die Box legen.

Jetzt kann der Akku nach dem Schema aus Bild 7 vorbereitet werden. Der Solar-Laderegler wird mit doppelseitigem Klebeband am Akku festgeklebt. Die Klebestelle sollte später wieder gelöst werden können, falls der Akku mal verschlissen ist und getauscht werden muss. Beim Aufkleben muss man darauf achten, dass die Oberseite mit den Kontakten des Akkus zu einem hin zeigt und die Pole auf der rechten Seite liegen 8. Anschließend kann der ausgeschaltete Notausschalter auf die Seite mit den Polen geklebt und mit dem Pluspol des Akkus verdrahtet werden. Der Ausgang des Notausschalters wird mit dem Pluspol des Akkuanschlusses am Solar-Laderegler verdrahtet. Der Minuspol des Akkus wird direkt mit dem Minuspol des Solar-Ladereglers verdrahtet.

Der so vorbereitete Akku wird nun in die Box auf den Akkuhalter gelegt und die weitere Verdrahtung kann vorgenommen werden. Die Kabel des Hochstromanschlusses und der KFZ-Steckdose werden an den Ausgang des Solar-Ladereglers angeschlossen. Den Ausgang des hier benutzten Solar-Ladereglers erkennt man an einem Glühbirnen-Symbol, auch hier muss die Polung beachtet werden. Zuletzt werden die Kabel für den SAE-Anschluss mit dem Solarpanel-Eingang des Ladereglers (Solarpanel-Symbol) verdrahtet.

An dem SAE-Anschluss kann anstelle eines Solarpanels auch eine andere Stromquelle passend zur Spannung, die der Laderegler erlaubt (14V-20V), angeschlossen werden. Nun noch mit dem Deckel die Box verschließen 9, Akku laden und es kann losgehen mit dem ersten Einsatz 10 unserer kompakten Powerbox. Wenn Sie noch weitere praktische oder coole Erweiterungen und Ideen haben, freuen wir uns über eine E-Mail an: **make@make-magazin.de** —*caw*


10

Smartphone-Roboter für den Unterricht

Holt die eingestaubten Smartphones aus der Schublade, jetzt wird gebastelt! Dieser Roboter besteht aus Pappe, leitfähigem Klebeband und einfacher Elektronik und ist ein tolles Projekt für Schulen und Familien.

von Michael Scheuerl

Ein programmierbarer Roboter aus einfachen Materialien und ohne Arduino oder andere Mikrocontroller – mit diesem Ziel habe ich die Entwicklung meines Smartphone-Roboters angefangen. Denn er soll nicht nur *für* das Kinderzimmer, sondern *im* Kinderzimmer gebaut werden. Und dabei die Kreativität von Kindern und Erwachsenen beflügeln, statt viele, komplizierte Bauteile zu erfordern und im schlimmsten Fall noch Müll zu produzieren.

Der Smartphone-Roboter ist modular aufgebaut, damit die Kinder schnell ein Erfolgserlebnis haben und hoch motiviert bleiben. Er kann ohne spezielle oder gar gefährliche Werkzeuge gebaut werden und arbeitet mit geringen Strömen, um Gefahren bei Kurzschluss auszuschließen. Außerdem ist er so günstig, dass man ihn auch mit einer Gruppe von Kindern in einem Klassenzimmer bauen kann.

In der Basisversion fährt er nur geradeaus, weil die beiden Getriebemotoren direkt an den Batterien hängen. Sie laufen bereits bei geringen Strömen, daher reichen Knopfzellen oder kleine Solarzellen aus und der Nachbau mit Kindern ist ungefährlich. Als optionale Upgrades habe ich eine kabelgebundene Fernsteuerung, Solarzellen für den Betrieb mit Sonnenenergie, leuchtende Antennen und eine schöne Karosserie entwickelt.

Mit einem Smartphone wird der Roboter in der „Luxus-Version“ schließlich programmierbar. Eine Lichtsensor-Transistor-Schaltung steuert die Geschwindigkeit: Die Lichtsensoren liegen direkt auf dem Bildschirm des Smartphones und erkennen dessen Helligkeit, von der die Geschwindigkeit der Motoren abhängt. Alles andere erledigt das Smartphone. Es kann mit einem *Text-To-Speech*-Modul sprechen oder über den *Voice Assistant* auf Befehle hören. Programmiert wird der Smartphone-Roboter über die blockbasierte Umgebung *MIT App Inventor*. Damit werden Apps so wie mit der beliebten Programmierlernumgebung *Scratch* erstellt.

Alle nötigen Links, die Druckvorlagen, Bezugsquellen für die Teile und eine Anleitung als Video findest du, wenn du meine Webseite unter „Alles zum Artikel im Web“ in der Kurzinfo aufrufst.

Kurzinfo

- **Basis-Papproboter bauen**
- **Modulare Erweiterung: Fernsteuerung, Solarzellen, Smartphone**
- **Programmierung mit MIT App Inventor**

Checkliste

Zeitaufwand:
ab zwei Stunden

Kosten:
20 Euro

Mehr zum Thema

- Elke Schick, Roboter für Kinder, Make Sonderheft Roboter 2019, S. 90
- Florian Schäffer, Maker-Apps selber machen, Make 3/18, S. 34

Alles zum Artikel im Web unter
make-magazin.de/x34f

Material

- **ausgedruckte Vorlagen** siehe Downloadlink
- **2 Getriebe-Solarmotoren** Sol-Expert G243 oder Solarbotics GM17
- **4 Knopfzellen** CR2032
- **2 Maulklemmen**
- **leitfähiges Klebeband**
- **Klebeband**
- **Deckel einer PET-Flasche**
- **Maulklemme**
- **2 NPN-Transistoren** 2N3904BU
- **2 Lichtsensoren**
- **2 LEDs** optional
- **Litze in rot und schwarz** optional
- **kleine Solarzellen** optional

Werkzeug

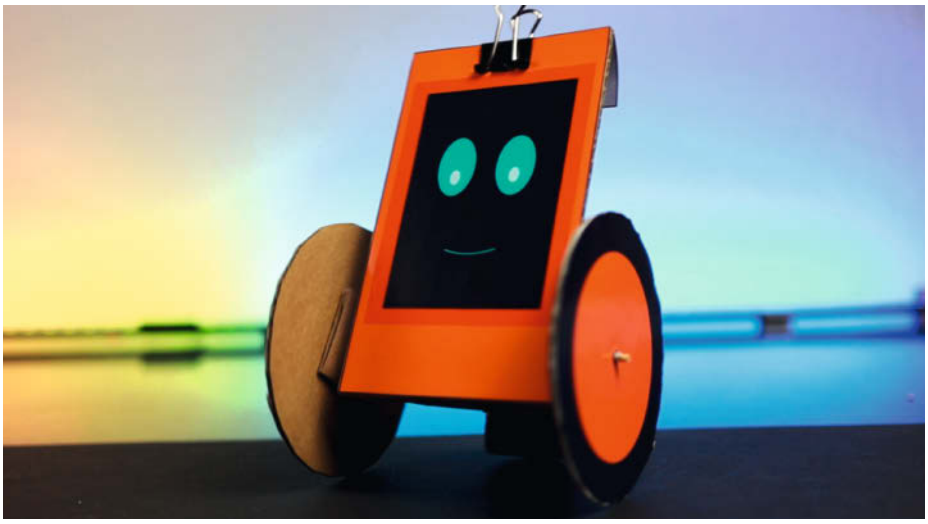
- **Heißkleber**
- **Schere**
- **Hammer**

Basis-Roboter bauen

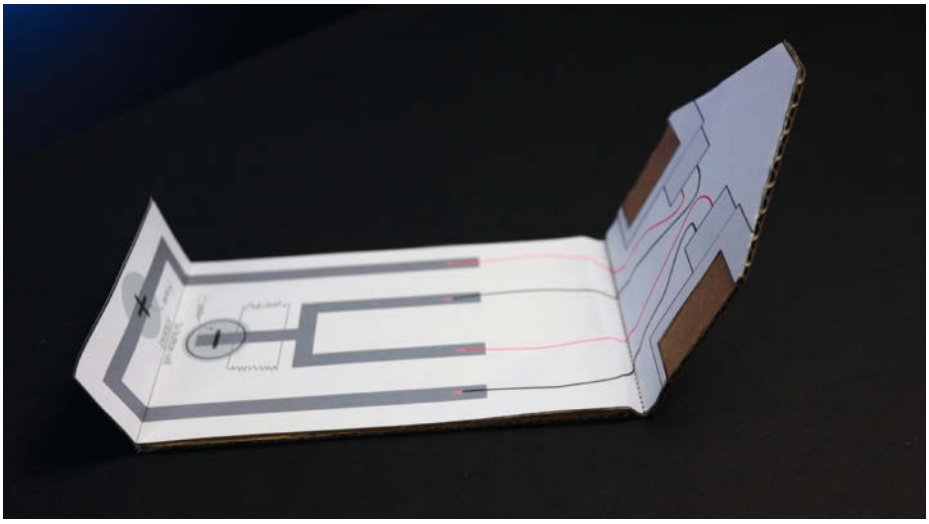
Die Vorlagen für den Roboter und die Räder druckst du am besten auf dickem Papier mit 160 Gramm Stärke. Klebe sie nach dem Ausschneiden auf ein dünnes Stück Pappe und schneide die Pappe ebenfalls zurecht. Der untere Streifen, an dem die Batterie eingeklemmt wird, sollte nicht auf Pappe geklebt werden. Falte anschließend das Fahrgestell in der Mitte an der gestrichelten Linie.

Beklebe die eingezeichneten Leiterbahnen mit dem stromleitenden Klebeband. Ich habe leitfähiges Nylon-Tape genutzt, dessen Kleber ebenfalls leitet. Es ist sehr robust und einfacher zu nutzen als leitfähiges Kupferband. Wenn die Klebeflächen nicht leiten, musst du dein Klebeband an den Ecken falten, um den Stromkreis geschlossen zu halten. An Übergängen oder wenn das Klebeband reißt, kannst du es auch doppelseitig legen. Bleibt der Stromkreis an einer Stelle offen, kann der Roboter nicht fahren. Dort, wo später die Knopfzellen platziert werden, kleben wir einen kleinen Streifen gewöhnliches Klebeband. Es dient als Isolierung, da an dieser Stelle sonst der Stromkreis überbrückt werden könnte.

Wer die Motoren von *SolExpert* gekauft hat, muss sie anhand der mitgelieferten Anleitung zusammenbauen. Dies geht ganz einfach, die Zahnräder, der Motor und das Gehäuse werden per Hand zusammengesteckt. Um ein Zahnrad auf die Motorwelle zu stecken, ist



Die Basisversion des Roboters



Mit einer Pappe und der Vorlage geht es los. Auf den Aussparungen der Vorlage werden später die Motoren aufgeklebt.

noch ein Hammer nötig. Die Übersetzung kannst du beim Zusammenbau selbst bestimmen, ein guter Wert ist 1:81. Damit sich die Stangen im Roboter nicht anstoßen, sollten die Wellen eher weiter vom Motor weg zeigen. Später kannst du über die anderen Übersetzungen mit der Kraftübertragung und Geschwindigkeit experimentieren. Ich habe Motoren von *Solarbotics* verwendet – sie werden fertig geliefert, der Shop sitzt aber in den USA.

Schaue nun, wie du die SolExpert-Motoren einbauen möchtest: Du kannst die Motorwelle näher oder weiter weg vom Boden aufkleben, je nachdem, ob du die großen oder kleinen Räder möchtest. Außerdem müssen die Wellen in einer Linie liegen, sonst lässt sich der Roboter später nicht ordentlich steuern. Klebe die Motoren dann fest, zum Beispiel mit Heißkleber. Knicke das lange Stück Pappe, bis es die Motoren berührt, und klebe es ebenfalls an den Motoren fest. Dann müssen noch die Motorkabel auf die Leiterbahnen aus Klebeband geklebt werden, am besten mit weiterem Klebeband. Die Vorlage zeigt, welches Kabel wohin kommt. Dreht sich der Roboter später im Kreis, sind die Kabel eines Motors vertauscht. Falls er rückwärts fährt, sind die Kabel von beiden Motoren vertauscht.

Schiebe die Pappräder mittig auf die Motorwelle und klebe sie fest. Bei den Solarbotics-Motoren ist ein Zahnstocher als Verlängerung der Achse hilfreich. Wichtig ist, dass der Kleber nicht über die Welle in den Motor gelangt und diesen verklebt. Als „drittes Rad" dient die Kappe einer PET-Flasche. Sie gleitet über den Boden und stabilisiert den Roboter. Das Gesicht klebst Du auf die Vorderseite der Pappe.

Klemme zwei CR2032-Knopfzellen mit der Maulklemme oben am Roboter fest. Dazu musst du die Pappe an der gestrichelten Linie knicken. Wie die Knopfzellen orientiert sein sollen, zeigt die Vorlage. Jetzt sollte der Roboter direkt losfahren.

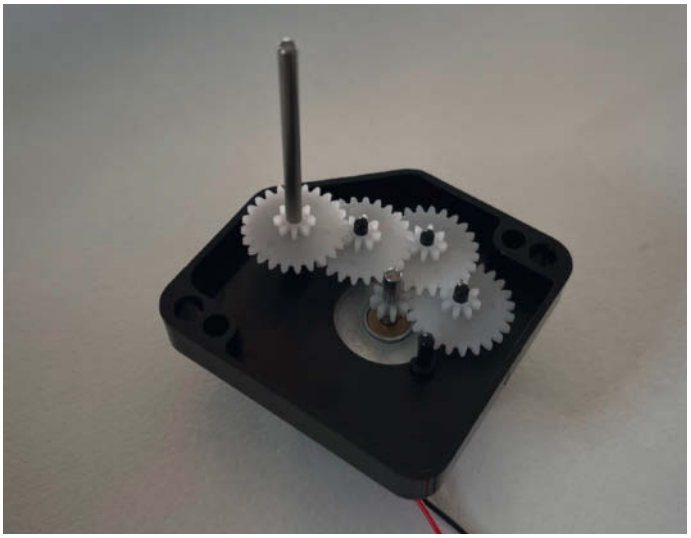
Fehlersuche

Falls er stehen bleibt, ist der Stromkreis nicht geschlossen. Eine häufige Fehlerquelle ist die Verbindung der Motorkabel zum Schaltkreis. Die Motorkabel haben nur wenige Millimeter abisolierte Litze am Ende. Dies stellt oft keinen guten Kontakt her und ein oder beide Motoren funktionieren nicht zuverlässig. Drücke fest auf die Klebestellen und prüfe, ob das hilft. Dann kannst Du die Kabelenden abnehmen und die Plastikhülle mit einer Zange vorsichtig entfernen, um mehr blanke Litze zu bekommen und diese wieder anzukleben.

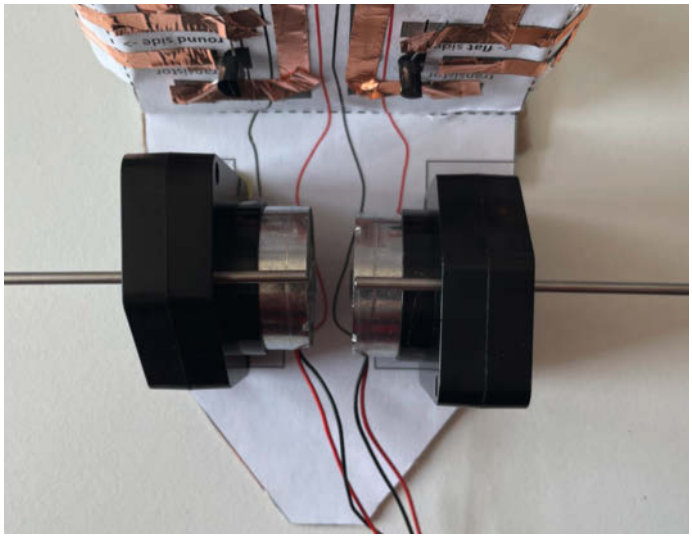
Kontrolliere dann, dass die Klebebänder nirgendwo eingerissen sind und alle Ecken und Anschlüsse Kontakt haben. Vielleicht sind auch die Batterien leer und müssen durch neue ersetzt werden. Sollten sich die Batterien etwas warm anfühlen, gibt es irgendwo einen Kurzschluss. Berühren sich die Leiterbahnen an einer Stelle, wo sie es nicht sollten? Ist der Streifen normales Klebeband neben den Batterien angebracht? Schließlich können die Räder das Problem sein. Sind sie blockiert? Oder drehen sich die Räder, wenn du den Motor anhebst, aber auf dem Boden fährt der Roboter nicht? Dann ist die Übersetzung der Getriebemotoren zu gering.

Fernsteuerung

Für die Verkabelung der Fernsteuerung benötigst du zwei rote und zwei schwarze Litzen von mindestens 1,5 Meter Länge. Wichtig ist,



Bei den SolExpert-Motoren sollte beim Zusammenbau die Motorwelle weit nach oben herausstehen.



Damit der Roboter gut zu steuern ist, müssen die Wellen der Solarbotics-Motoren vor dem Kleben auf eine Linie ausgerichtet werden.

dass die Kabel sehr flexibel und dünn sind. Die Enden aller Kabel müssen abisoliert werden. Die Druckvorlage hat zwei Teile: Oberseite und Unterseite. Beklebe beide zuerst mit stromleitendem Klebeband und zwei Streifen gewöhnlichem Klebeband. Die Unterseite wird auf Pappe geklebt und ausgeschnitten. Auf die andere Seite der Pappe klebst du die große Fläche der Oberseite. Die übrigen Teile der Oberseite kannst du anschließend umschlagen, zusammenkleben und dann nochmals zur anderen Seite umknicken. So ergeben sich unten ein „Batteriefach" für zwei Stapel Knopfzellen und oben zwei Schalter.

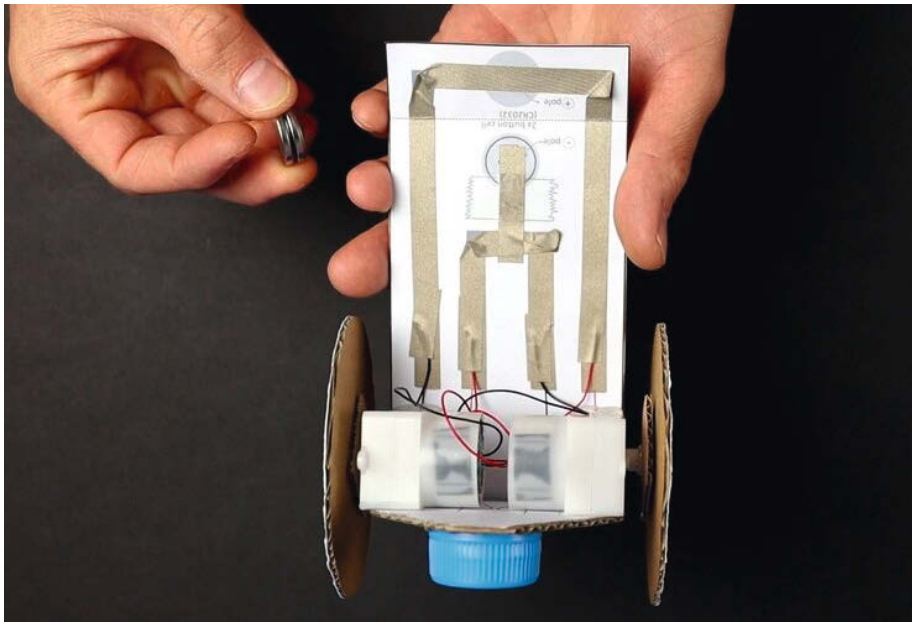
In der Vorlage ist markiert, wie der Schaltkreis der Fernbedienung mit dem Schaltkreis des Roboters zu verbinden ist. Klebe die Kabel wieder mit leitfähigem Klebeband auf. Wenn alles richtig verkabelt ist, aktiviert der rechte Schalter den rechten Motor und der linke den linken. Nun musst du den ursprünglichen Batteriehalter des Roboters abschneiden und den Stromkreis so unterbrechen. Wir benötigen jetzt zwei separate Schaltkreise, um die Motoren zu steuern. Die Batterien sowie zwei weitere Knopfzellen wandern in die Fernbedienung. Ein Stapel wird pro Motor benötigt. Teste nun wieder den Roboter. Aufgepasst: Da er auch die Kabel ziehen muss, drehen die Räder schnell durch. Dann kannst du ihnen mit einer Spur Heißkleber rundherum etwas mehr „Grip" verpassen.

Solarantrieb

Für diese Variante kannst du die passende Vorlage ausdrucken oder eine vorhandene Fernsteuerung umbauen, indem du das Batteriefach entfernst und die Leiterbahnen auf Ober- sowie Unterseite mit einem Streifen leitfähigem Klebeband verbindest. Der Roboter funktioniert schon mit einer Solarzelle, mehr Spaß hat man aber mit zwei. Die Kabel der Solarzellen verbindest du, wie in der Vorlage angegeben: schwarze Kabel nach unten und rote Kabel nach oben. Dann geht's raus in die Frühlingssonne, um den Roboter zu testen.

Karosserie-Upgrade

Der Roboter kann noch schicker aussehen. Drucke dafür die Karosserie-Vorlage auf Fotopapier aus. So glänzt sie, als wäre der Roboter lackiert und die Farben wirken viel intensiver. Wichtig ist, dass man vor dem Zusammenbau alle Faltkanten vorfalzt. Dazu nutzt du ein Lineal und den Bügel der Maulklemme oder den Rücken eines Butterbrotmessers. Damit fährst du mit genug Druck alle Faltkanten entlang. Lege am besten eine Matte oder Pappe unter. Wenn die Kanten gut gefalzt sind, kannst du sie ganz einfach falten und die ausgeschnittenen Teile an den Laschen aneinander kleben.



Die Rückseite des Papproboters: Die Leiterbahnen aus Klebeband müssen durchgängig Kontakt haben. Es muss unter dem Flaschendeckel etwas Platz zum Boden sein. Hier sind die Solarbotics-Motoren verbaut.

Leuchtende Antennen

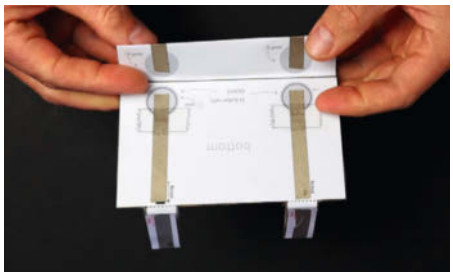
Die leuchtenden Antennen bestehen aus einem einfachen Papierstromkreis und zwei LEDs. Wer weiße LEDs verwendet, kann diese mit buntem Papier umwickeln, um farbige Antennen zu bauen. Wenn man das farbige Papier einölt, leuchtet das Papier noch besser. Die Antennen steckst du durch einen Schlitz in der Karosserie oder befestigst sie direkt am Basis-Roboter.

Schaltkreis testen

Das letzte Upgrade, der Smartphone-Roboter, arbeitet mit einer Lichtsensor-Transistor-Schaltung. Um das Prinzip zu erforschen und zu verstehen, kannst du meinen Demonstrations- und Teststromkreis nutzen. Klebe den Stromkreis wie in der Vorlage angegeben und schaue, wie er funktioniert. Der Transistor öffnet und schließt den Stromkreis zum Motor. Aktiviert wird er über den Lichtsensor, der mit der Basis des Transistors verbunden ist. Fällt Licht auf den Sensor, wird dessen Widerstand sehr klein und Strom kann zum Transistor fließen, der so den Stromkreis zum Motor „öffnet". Ohne Licht „schließt" der Transistor den Stromkreis. Er fungiert in dieser Schaltung also als Schalter. Mehr Licht führt dabei zu einem stärkeren Stromfluss und somit zu einer höheren Drehgeschwindigkeit des Motors. In der Vorlage gibt es noch eine Lücke für einen optionalen Widerstand. Mit ihm kann man den Schwellenwert verändern, wann der Transistor den Stromkreis „öffnet".

Smartphone-Roboter

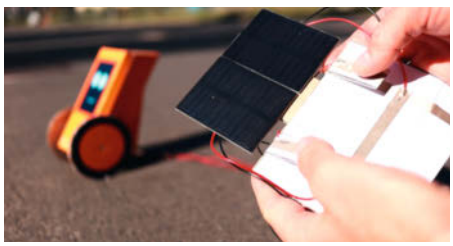
Schneide die Vorlage aus und falte alle Faltkanten vor. Klebe anschließend den Stromkreis.



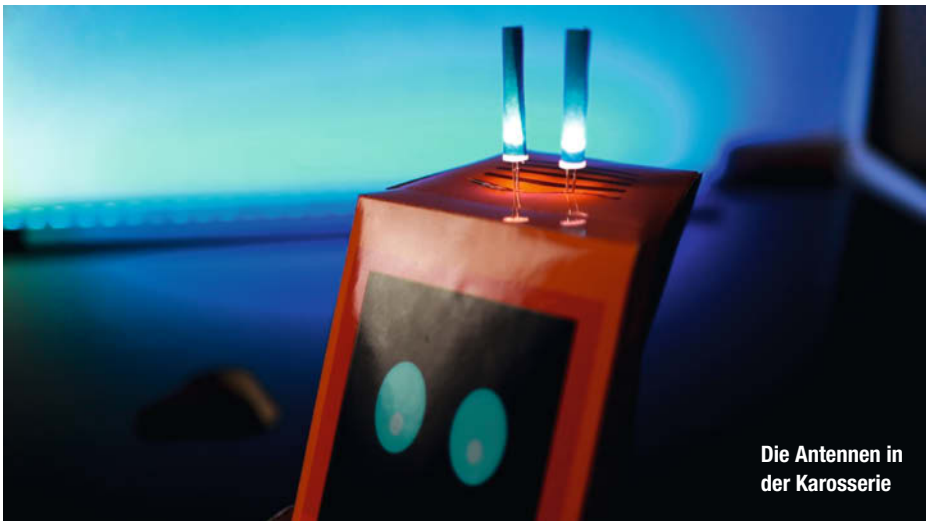
Die Fernbedienung wird oben und unten zurechtgebogen.

An den Batteriehaltern sind wieder kurze Streifen normalen Klebebands nötig. Dann biegst du die Beinchen der Transistoren zurecht und klebst jedes Beinchen mit dem leitfähigen Klebeband an die angegebene Stelle. Die Transistorgehäuse haben eine runde und eine flache Seite. Die Vorlage zeigt, wie die Transistoren eingeklebt werden müssen.

Die Lichtsensoren werden durch ein Loch gesteckt, damit die Sensoren auf die Seite mit dem Smartphone zeigen, und die Beinchen ebenfalls angeklebt. Die Sensoren solltest du mit transparentem Klebeband überkleben,



Wichtig ist, die Fernsteuerung in die Sonne zu halten.



Die Antennen in der Karosserie

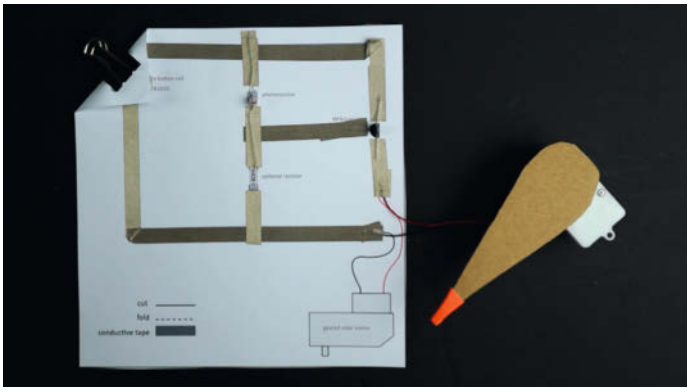
sonst berühren sie später den Screen. Dann löst du die Motorkabel vom Schaltkreis und klappst das Fahrgestell auf. Dafür löst du die Klebebefestigung der Motoren. Das untere Ende des neuen Schaltkreises ist nicht mehr nötig und kann abgeschnitten werden. Klebe nun den neuen Schaltkreis über den alten.

Die Motorkabel werden wie in der Vorlage gezeigt neu verbunden. Beide Motoren müssen mit dem Minuspol und jeweils einem der beiden Transistoren verbunden werden. Klebe dann die Vorderseite des Fahrgestells wieder an die Motoren. Dort wo bisher das Robotergesicht war, platzierst du nun dein Smartphone und biegst die Streifen mit den Lichtsensoren vorsichtig um das Smartphone und klebst sie mit Klebeband aneinander. Damit das Smartphone nicht nach unten durchrutscht, kommt noch ein kleines Stück Pappe an die Unterseite des Fahrgestells. Da Streulicht und Licht von außen unseren Roboter stören können, solltest du die Rückseite der Sensoren mit normalen Tape bekleben und kannst noch eine rechteckige Verkleidung (Druckvorlage siehe Kurzinfo) davor anbringen.

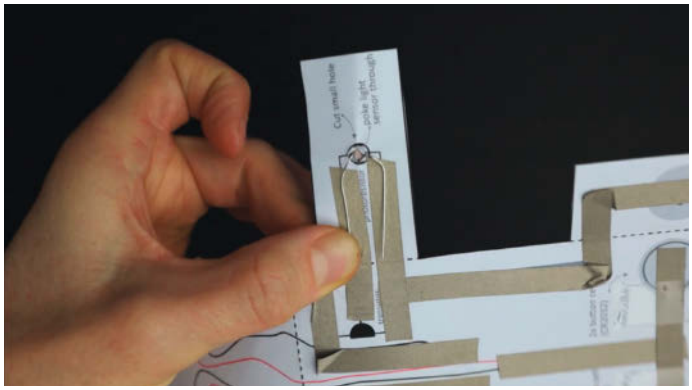
Losfahren

Klemme auf beiden Seiten zwei Knopfzellen fest und halte den Roboter in helles Licht. Bewegen sich beide Motoren? Wenn ja, kann es weitergehen. Zum Testen habe ich ein Video vorbereitet (siehe Links in der Kurzinfo). Es zeigt dort, wo die Lichtsensoren sind, weiße und schwarze Flächen. Starte das Video am besten in einer Wiederholungsschleife und schiebe das Handy in den Roboter. Wenn du das Zimmer abdunkelst, sollte das Smartphone mit dem Video den Roboter steuern: geradeaus, Kurve rechts, Kurve links und dann stopp.

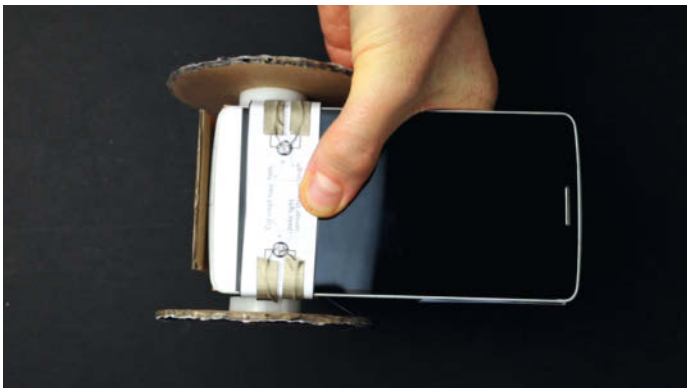
Wenn die Motoren sich auch ohne Smartphone und in hellem Licht nicht drehen, sind entweder die Batterien nicht korrekt angeschlossen, die Motorkabel nicht gut mit dem Stromkreis verbunden oder es gibt im Stromkreis einen Kurzschluss. Wenn der Roboter fährt, aber keine Kurven macht, wird schwarz („kein Licht“) nicht erkannt. Dann kannst du die Helligkeit des Smartphonescreens verringern oder das Zimmer besser abdunkeln. Es kann auch sein, dass zu viel Licht von der einen Seite des Screens zur anderen Seite herüber leuchtet. Dann kannst du ein kleines



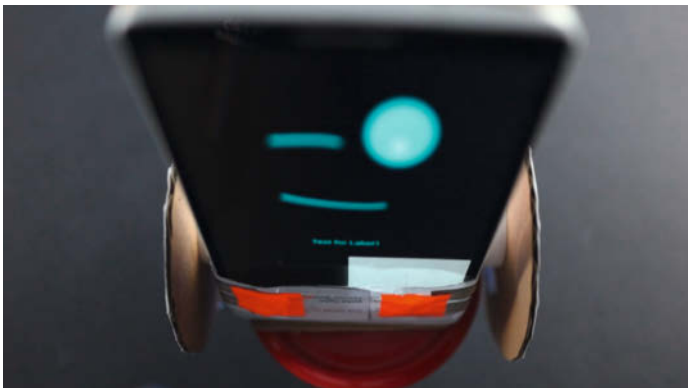
Der Test-Schaltkreis



Einsetzen des Lichtsensors und des Transistors



Die Pappstreifen mit den Lichtsensoren werden vorsichtig umgebogen. Unter dem Smartphone sorgt eine Pappe für Halt.



Unter dem roten Klebeband sitzen die Lichtsensoren, die auf die Helligkeit des Smartphone-Screens reagieren.

Stück Papier zusammenrollen und zwischen die beiden Lichtsensoren schieben. Stoppt der Roboter nicht, wird ebenfalls schwarz nicht erkannt. Sollte der Roboter sehr langsam sein, kannst du die Getriebeübersetzung in den Motoren ändern. Überprüfe vorher noch, ob die Transistoren richtig eingeklebt sind.

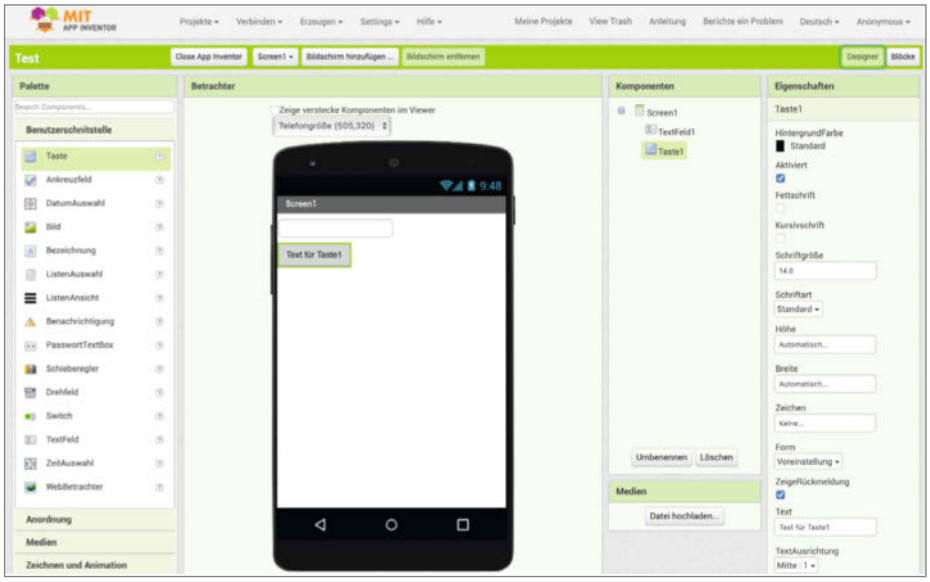
Programmieren

Als Erstes müssen wir den *MIT App Inventor* mit dem Handy verbinden und eine kleine Testapp aus Text und einem Button übertragen. Besuche dazu an einem PC die Webseite von MIT App Inventor. Der App Inventor fragt standardmäßig nach einem Google-Account, funktioniert aber auch ohne. Dann bekommst du einen Code, den du notieren musst, um deine Projekte später wiederzufinden – wenn du etwa an einem anderen Computer arbeitest. Auf dem Handy musst du die *Companion*-App installieren. Im Google Play Store heißt sie „MIT AI2 Companion", in Apples App Store ist sie als „MIT App Inventor" zu finden.

Nach dem Einloggen oder Start auf dem Computer erstellst du über *Start a Blank Project* ein neues Projekt und gibst einen *Projektnamen* an. Oben rechts kannst Du dann die Sprache von *English* auf *Deutsch* (oder eine andere Sprache) ändern. Für einen ersten Test fügst du ein Textfeld und einen Button zu deiner App hinzu. Klicke dafür links im Fenster unter *Benutzerschnittstelle* auf *Textfeld* und ziehe es auf die große Fläche. Genauso fügst du dann eine *Taste* ein.

Oben rechts wechselst du zur Programmierung auf *Blöcke*. Nun findest du links den Kasten *Blöcke*. Unter *Screen1* kannst du auf *Taste1* klicken, um die möglichen Befehle zu sehen. Ziehe *Wenn Taste1 Klick, mache* in das Programmierfeld. Mit einem Klick auf *Textfeld1* bekommst du den Befehl *setze TextFeld.Textfarbe auf*. Die Farben gibt es unter *Eingebaut*. Achtung: Solange keine Farbe eingesetzt ist, siehst du neben dem Code eine Fehlermeldung. Jetzt hast du ein Programm, das auf Knopfdruck die Textfarbe des Textfeldes ändert.

Um das Programm auf das Handy zu übertragen, muss es im selben WLAN wie der Computer sein. Wähle nun im App Inventor *Verbinden / AI Companion* aus. Dann siehst du einen QR-Code, den du mit der Companion-App auf dem Handy scannen kannst, sobald du ihr den Zugriff auf die Kamera erlaubt hast. Nach dem Scan startet deine neue App und du hast eine direkte Verbindung. Gib auf dem Handy etwas im Textfeld ein und drücke den Button, um die eben programmierte Testapp auszuprobieren. Du kannst auch deine App verändern und alles auf dem Handy testen, ohne den Code noch einmal scannen zu müssen. Aktualisiere ihn einfach über *Verbinden / Refresh Companion Screen*.

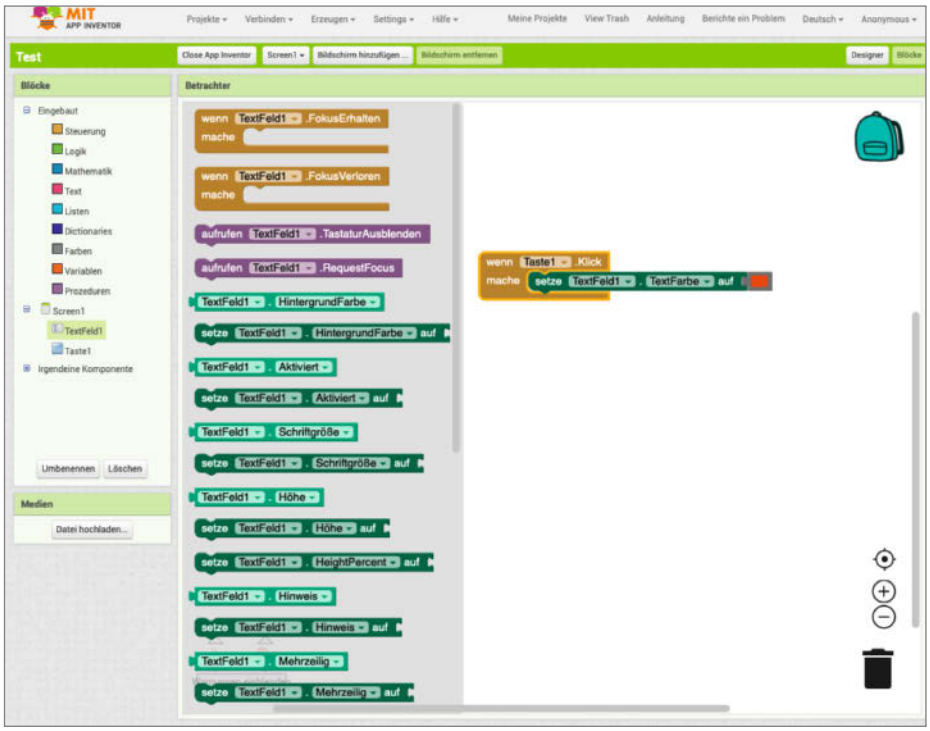


So soll die Test-App aussehen.

Viele Möglichkeiten

Jetzt geht es los mit der Roboterprogrammierung! Du kannst den Roboter herumfahren lassen, indem du schwarze und weiße Rechtecke dort einblendest, wo die Lichtsensoren sitzen. Oder du bringst dem Roboter mit dem Text-to-Speech-Modul Sprache bei. Mit diesem Modul kannst du einen beliebigen Text eingeben und den Roboter auf Befehl sprechen lassen. Durch das Einfügen von selbst erstellten oder gemalten Grafiken kann der Roboter verschiedene Gesichtsausdrücke zeigen. Ein Beispielprogramm und vorbereitete Gesichter stelle ich online bereit.

Mit dem Speech-Recognition-Modul lässt du den Roboter auf Sprachbefehle reagieren. Dann startet der Voice Assistant des Betriebssystems und wandelt gesprochene Sprache in Text um. Mit einem zweiten Handy kannst du über *TinyWebDB* Variablen austauschen und so den Roboter von überall auf der Welt fernsteuern. Wie das geht, erkläre ich auf meiner Webseite (siehe Link in der Kurzinfo). Welche Ideen hast du? *—hch*



Der Quellcode besteht aus nur zwei Befehlen.

Schachspiel für Sehbeeinträchtigte

In einem Projekt an der HBK Essen wurde ein Schachspiel im bunten Materialmix von Beton und Holz entwickelt. Hierbei wurde auf Inklusion und Nachhaltigkeit geachtet.

von Kerstin Ogrissek

Ich studiere im dritten Semester an der *HBK Essen* digitales Produktdesign. Im Sommersemester 2021 verwirklichten wir in der Lehrveranstaltung *Design für Alle* ein Projekt zum Thema *Schachspiel und Inklusion bei Sehbeeinträchtigung*. Mein Schachspiel ist eine Kombination aus 3D-Druck, Silikon- und Betonguss, Lasercut und Holzbearbeitung. Die entworfenen Figuren wurden anfangs in PLA gedruckt. Dies ist bei 32 Figuren allerdings sehr zeitaufwändig und nicht ressourcenschonend.

So entstand die Idee, eine Gussform zur Nutzung mit verschiedenen Materialien, insbesondere Beton, zu erstellen. Diese Gussformen können immer wieder verwendet werden. Die Figuren wurden mit *Fusion 360* designt. Für die Erstellung der Gussformen wurden die Bauern viermal pro Form und alle anderen Figuren einmal für eine Figurensatzform gedruckt.

In Zusammenarbeit mit dem FabLab und der Abform-Werkstatt wurde aus den Positiven eine Negativform in Silikon erstellt, um diese als Gießform für Beton zu nutzen. Die Negativform besteht aus zwei Teilen, um die Gussfiguren später leicht entformen zu können. Dabei sind diverse Trennmittel hilfreich, die vor dem Gießen aufgetragen werden. Die fertigen Silikon-Negativformen wurden in Kästen mittels Gipsguss stabilisiert und sind nun für den Betonguss verwendbar.

Jetzt kann der pigmentierte Beton gegossen werden. Nach 1–2 Tagen Abbindezeit können die Figuren entformt werden und die Formen sind nach einfachem Abwaschen für den nächsten Guss bereit. Die fertigen Figuren wurden mit Farbvertiefer versiegelt.

Somit fällt bei erneutem Guss kein unnötiger Müll an und das Verfahren ist nachhaltig. Da die bereits für die Formerstellung gedruckten PLA-Figuren bei meinem Schach das gegnerische Figurenset bilden, ist auch schon bei meinem ersten Guss kein Plastikmüll entstanden. Ein anderer spannender Effekt, den ich so erzeugen konnte: Aufgrund der unterschiedlichen Materialien sind die Figurensets unterschiedlich schwer und Menschen mit Sehbeeinträchtigung können nachvollziehen, welche Figuren zu welchem Spieler gehören.

Eine weitere Funktion, die ich für Menschen mit Sehbeeinträchtigung im Design der Figuren umgesetzt habe: Es wurden tastbare Symbole unter die Figuren gesetzt. Durch die von mir entwickelte transportable, an den Fronten offene Schachbox, ist es so möglich, die Figuren auch von der Unterseite her mit den Händen zu berühren. So können die Spieler Schachzüge erfühlen, ohne den aktiven Spieler bei seinem Zug zu beeinträchtigen. Die Schachbox wurde aus leichtem Pappel-Massivholz in unserer Holzwerkstatt gebaut. Das Holz war sehr gut zu lasern und zu bearbeiten. —*caw*

▶ heise.de/s/Mkgr

Einfüllen von Betonguss in die stabilisierten Latex-Formen

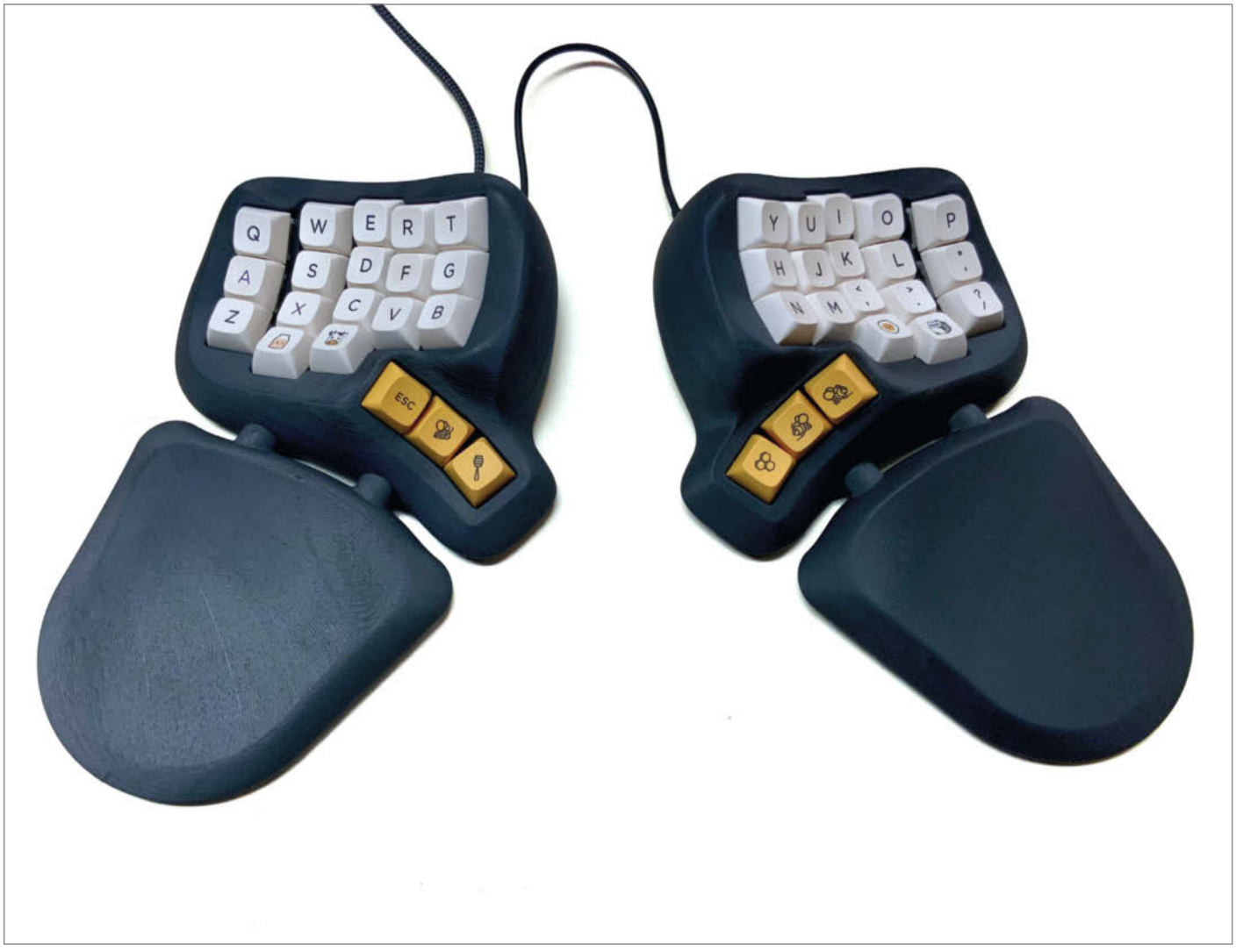
Fertige Beton-Figuren

Das zusammenklappbare Untergestell der Schachbox

Ergonomische Split-Tastatur im Eigenbau

Als Basis für die Tastatur dienen zwei der günstigen Raspberry-Pico-Boards. Die Firmware ist per einfachem Texteditor oder einem grafischen Tool anpassbar.

von Andreas Känner



Selbstbautastaturen liegen im Trend. Zubehörteile wie Tasten, Tastenkappen, Kabel und Mikrocontroller sind mittlerweile gut verfügbar und es gibt bereits viele Bauanleitungen im Internet. Aber trotz langer Recherche fand ich keine Tastatur, die genau meinen Wünschen entsprach und habe deshalb die *PicoSplit*-Tastatur entworfen. Sie besteht aus zwei Hälften, die sich individuell anordnen lassen, um Ermüdungserscheinungen und Verkrampfungen zu verhindern. Sie ist ergonomisch geformt, hat abnehmbare Handballenauflagen, lässt sich auf magnetisch haftenden Unterlagen fixieren und alle Plastikteile lassen sich auf einem herkömmlichen 3D-Drucker drucken.

Die *PicoSplit* ist eine Tastatur mit nur 40 Tasten. Jede Taste ist aus der Grundposition alleine über Fingerbewegungen erreichbar. Die Handballen müssen praktisch nie bewegt werden. Zum Vergleich: Eine normale PC-Tastatur hat zwischen 83 und 108 Tasten. Damit man alle üblichen Zeichen tippen kann, sind die Tasten der *PicoSplit* mehrfach belegt.

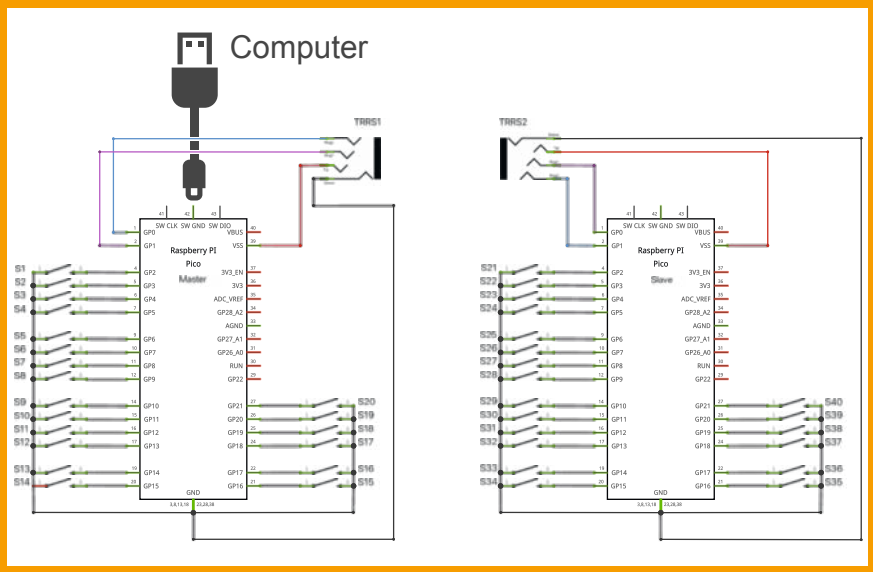
Die zwei Herzen der *PicoSplit* sind *Raspberry-Pi-Pico*-Mikrocontroller, die in *CircuitPython* programmiert sind. Das hat den Vorteil, dass man keine Entwicklungsumgebung braucht, um das Tastaturlayout zu ändern oder um die Firmware selbst zu erweitern. Ein einfacher Texteditor genügt! Per Tastendruck meldet die Tastatur einen USB-Speicher am Rechner an, auf dem der Quellcode und die Konfigurationsdateien liegen.

Da die Tasten der *PicoSplit*-Tastatur nicht in einer Ebene liegen, müssen sie von Hand verdrahtet werden. Ich habe eine Unterstützung in die Firmware eingebaut, um hier etwas Zeit zu sparen: Man gibt in der Konfigurationsdatei lediglich an, welche Pins der Raspberry Pi Pico für die Tasten verwenden soll. Welche Taste man dann an welchen Pin lötet, spielt keine Rolle. Die Zuweisung der Tasten kann man anschließend mit einem grafischen Programm vornehmen. Das Programm ist in *HTML* und *Javascript* geschrieben und ist Bestandteil der Firmware. Die Firmware unterstützt Ebenen, um Tasten mehrfach zu belegen. Tasten können abhängig davon, wie lange man sie gedrückt hält, unterschiedliche Aktionen auslösen. Drückt man eine Taste nur kurz, dann wird der entsprechende Buchstabe klein ausgegeben. Hält man sie länger gedrückt, bevor man loslässt, so wird der Buchstabe groß ausgegeben. *Modifier* lassen sich auf Tasten legen, die ansonsten Buchstaben ausgeben. Verschiedene weitere Funktionen sind verfügbar. Die Firmware lässt sich leicht an eigene Tastaturen anpassen und ich habe sie unter einer Open-Source-Lizenz auf *GitHub* veröffentlicht. —*caw*

▶ **kaenner.de/picosplit**

Der erste Prototyp als mobile Schreibmaschine

Alle nötigen Bauteile als Tastaturpuzzle ausgelegt



Der sehr übersichtliche Schaltplan

Der Granulat-Extruder

Bereit für die nächste Stufe? Dieses nachhaltige Projekt hinterfragt den Status Quo des Filamentdrucks und zeigt euch, wie ihr mit selbst recyceltem Material vorzeigbare 3D-Drucke erzielt.

von Ákos Fodor

Bilder: Norbert Heinz

Wohin mit den übrig gebliebenen Fehldrucken und alten Filamentresten? Das fragen sich viele Maker und Norbert Heinz scheint eine Lösung gefunden zu haben, die ihm, nach eigener Aussage, kaum jemand zugetraut hat. Der Entwurf eines kostengünstigen, leichten und zuverlässigen Granulat-Extruders für 3D-Drucker hat ihm gewiss ein paar schlaflose Nächte beschert. Doch der in Version 3 befindliche Prototyp überzeugt bereits jetzt mit zuverlässigen Druckergebnissen bis 30mm/s und wird stetig von ihm weiterentwickelt.

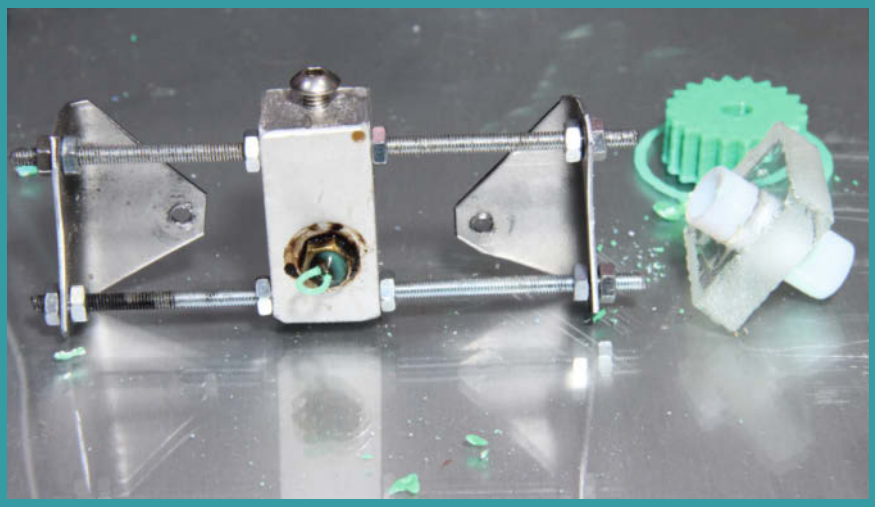
Das für den 3D-Druck recycelte Granulat entsteht durch das Schreddern alter Druckteile oder Filamente in einem Mixer, wird anschließend gesiebt und danach in den Extruderbehälter gekippt, wo es mithilfe einer Schnecke zum Hotend befördert wird. Auch mit Zucker hat der Erfinder bereits erfolgreich experimentiert, da es in den meisten Haushalten bereits als Granulat vorrätig ist und sich durch Erhitzen verarbeiten lässt.

Hinter der vermeintlich simplen Funktionsweise stecken ein ausgeklügeltes Zusammenspiel aus verschiedenen Systemen und viele praktische Entwicklungsversuche. Norbert Heinz zeigt uns auf seiner Webseite einerseits den Aufbauprozess und hilft uns gleichzeitig mit seinen detaillierten Erläuterungen zum Entstehungprozess, die Funktionsweise des Extruders und die ihr zugrundeliegenden Entscheidungen zu verstehen. So erklärt er z.B. die Verwendung des Glases im Hotend, weshalb er eine Holzschraube anstatt eines Holzbohrers für die Schnecke gewählt hat oder wie die Materialzufuhr im Detail funktioniert, was wiederum mit den Herausforderungen des Materials selbst zusammenhängt.

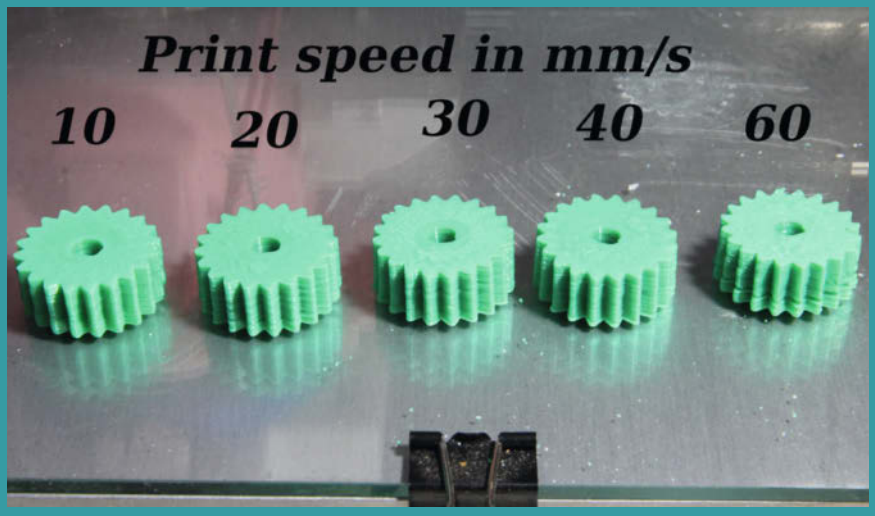
Obwohl die Anleitung wie auch das Projekt vor allem für fortgeschrittene Maker geeignet sind, können Technikbegeisterte jeder Altersstufe etwas aus den Erklärungen lernen. Bei der Gelegenheit lohnt es sich ebenfalls, einen Blick auf die vielen anderen Projekte zu werfen, die alle, dem Open-Source-Gedanken folgend, für jeden frei zur Verfügung stehen.

Mit dem Granulat-Extruder gibt uns Norbert Heinz ein Werkzeug an die Hand, das die Maker-Community unabhängiger von der Filamentindustrie machen kann. Den ersten Schritt Richtung Nachhaltigkeit ist er bereits für uns gegangen, der zweite Schritt liegt in unserer Verantwortung. Denn wirklich wertvoll wird das Projekt vor allem dann, wenn wir die Chance ergreifen, unser Materialwissen zu erweitern, um bewusster mit Kunststoffen umzugehen. Wir wünschen viel Spaß beim Nachbau, Experimentieren und Lernen. *—akf*

▶ make-magazin.de/xrwa



Das Hotend und seine Befestigung im Detail: Die meisten Bauteile sind von Hand gefertigt, so auch das Glaselement rechts.



Probedrucke mit einer 1mm-Düse und unterschiedlichen Geschwindigkeiten

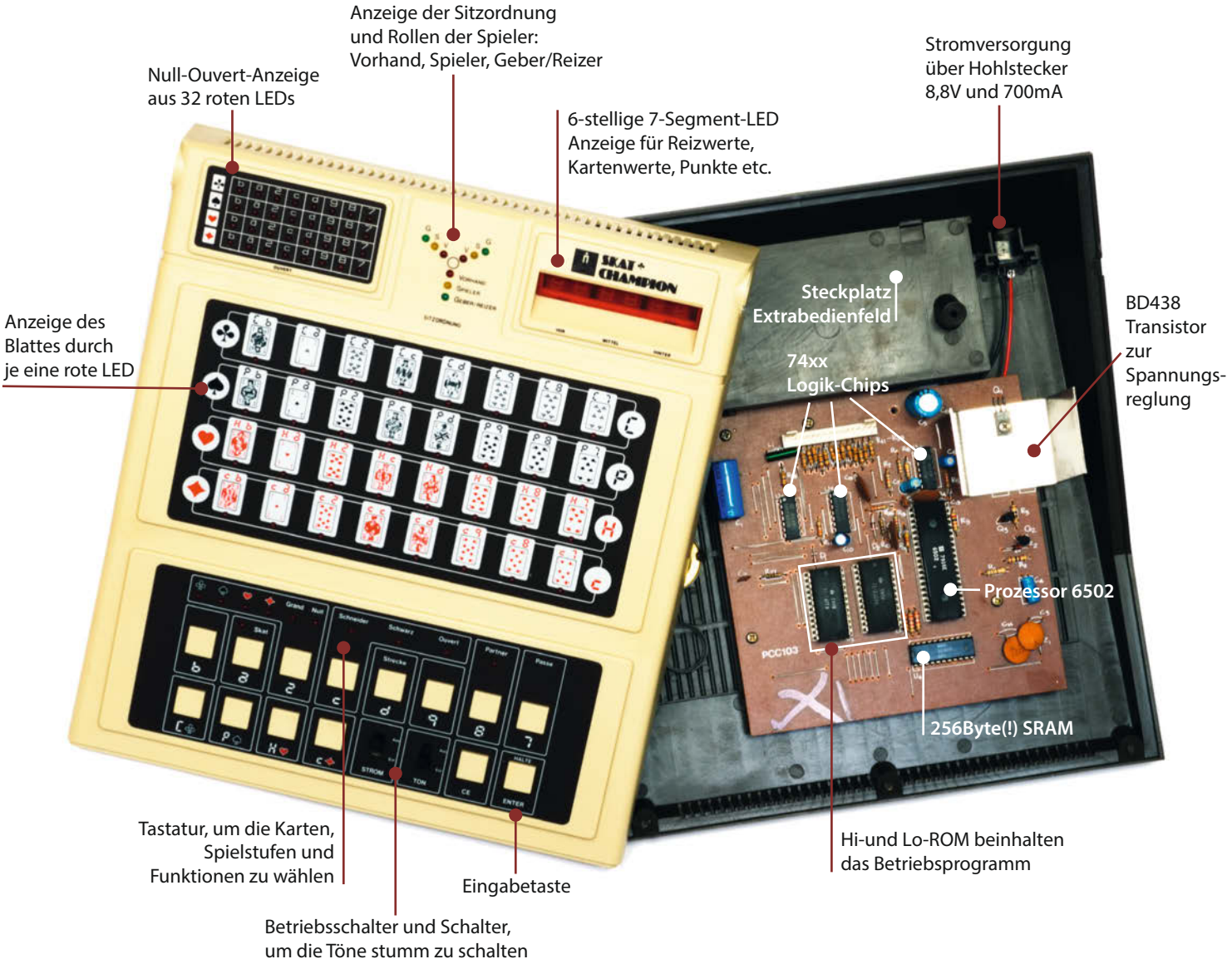
Selbst Zucker lässt sich verarbeiten, allerdings mit weniger elastischen Materialeigenschaften.

Reingeschaut: Skat Champion

Wir haben einen Skatcomputer von Anfang der 1980er Jahre aufgeschraubt und zeigen, was in ihm tickt und piept.

von Carsten Wartmann

Alles zum Artikel im Web unter *make-magazin.de/x2p1*



Entwickelt wurde der *Skat Champion* von den Firmen *Novag Industries Ltd.* und *Scientific Systems Ltd. (heute Saitek)*, Marktreife erlangte er 1980 und kostete zu Anfang 398 DM. Der Computer ersetzt in der Basisausführung zwei der drei Skatspieler. Braucht man nur den „dritten Mann", so gab es ein Zusatzgerät, das in einen Erweiterungssteckplatz eingesteckt wurde, ein weiteres Bedienterminal für den zweiten menschlichen Spieler bot und durch eine Sichtblende das Spicken verhinderte.

Viel Medienecho erhielt der Skat Champion seinerzeit in Deutschland, unter anderem im *Spiegel*, *Hamburger Abendblatt* und der *Welt*. Der Skat Champion hält sich streng an die Wettkampfregeln des *Deutschen Skatverbandes*, die Regeln, die das gesellige Spiel ausmachen, beherrscht er nicht. Seine Spielstärke wurde von den Testpersonen des *Spiegel* als „eher durchschnittlich" und „phantasielos" benannt.

Damals war die Fülle von Tasten, LEDs und Segmentanzeige schon eine Herausforderung für den Spieler. Viele Tasten sind doppelt belegt und so gibt es, bis das tatsächliche Spiel startet, elf Tasten zu drücken, wie der *Spiegel* seinerzeit zählte. Ebenso gewöhnungsbedürftig ist es heute für uns, als Maus- und Touchscreen-verwöhnte Generation, die LED-Segmentanzeige abzulesen.

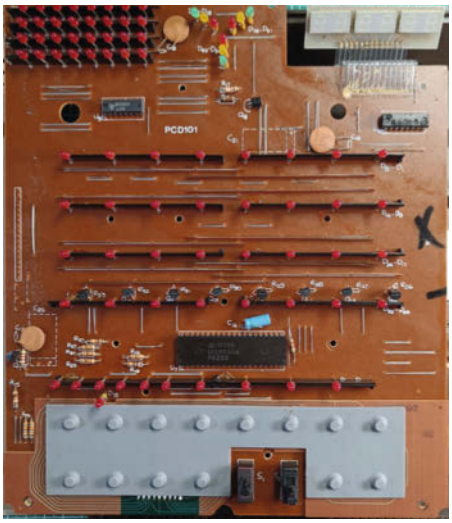
Was dieses fast antik wirkende Gerät noch heute interessant macht, ist der technische Aufbau und dass es als eines der ersten Geräte dieser Art in Serie produziert wurde. Im Gegensatz zu den auch heute noch verbreiteten Schachcomputern hat sich der Skatcomputer allerdings nie durchsetzen können.

Der jüngste Chip in meinem Gerät stammt laut *Datecode* aus Kalenderwoche 43 des Jahres 1980. Als Herz des Skatcomputers schlägt ein 6502A-Prozessor mit knapp 1MHz Takt, allerdings nicht vom ursprünglichen Entwickler *MOS*, sondern von *Scientific Systems Ltd.* in Lizenz produziert. Diese Prozessorreihe war ein wesentlicher Motor der 8-Bit-Revolution: Von Apple I bis III, über verschiedenste Commodore-Computer bis zu Ataris 8-Bittern war der 6502 in Varianten verbaut, etwa im C64 als MOS6510.

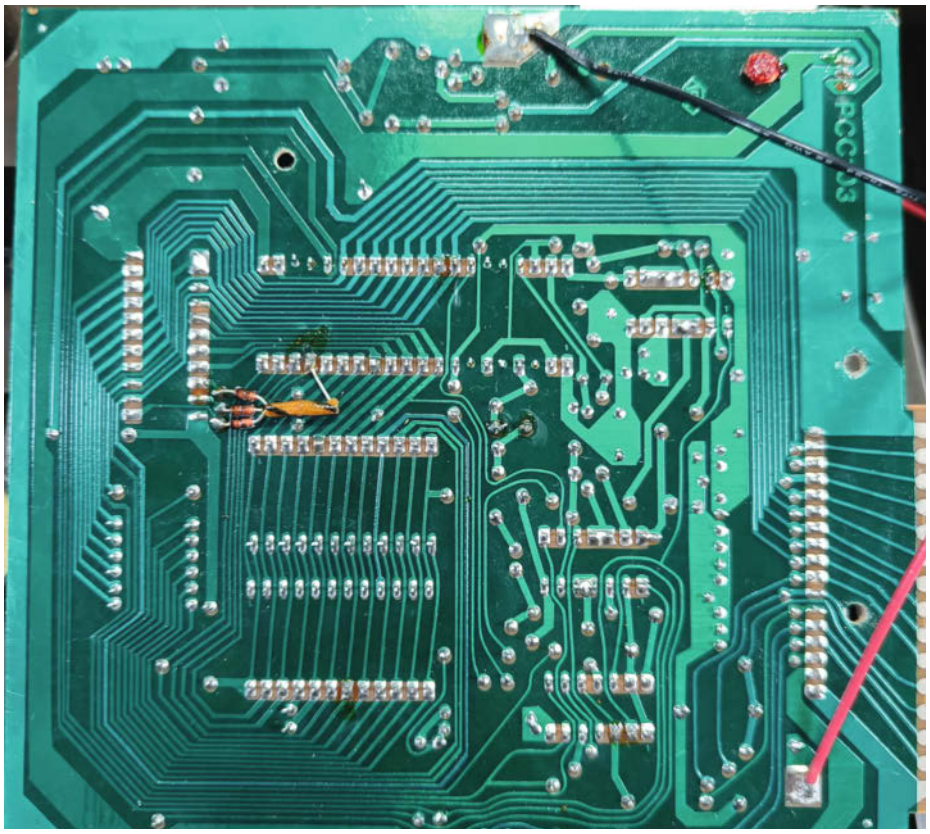
Pik Bube, Creutz (Kreuz) 7 und Kreuz Bube

Sein Programm bekommt er aus zwei ROMs, die 16kByte fassen, dabei stehen ihm magere 256 Bytes RAM-Speicher zur Verfügung. Sonst besteht die Hauptplatine nur noch aus drei weiteren Logik-Chips der 74xx-Reihe, aber wie so oft bei Vintage-Geräten dürfen ein paar *Bodges* auf der Platine nicht fehlen: Diese Hacks korrigieren Fehler auf dem PCB, hier in Form eines nachträglich angebrachten Kondensators und drei Dioden, die wohl ein paar Signale (A0-A2) vom SRAM zusammenfassen und zu Pin A8 am Prozessor gehen.

Auf der Platine des Bedienpanels nehmen natürlich die 84 LEDs und das 6-stellige LED-7-Segment-Display den größten Raum ein. Zusätzlich müssen 14 Tasten abgefragt werden, daher übernimmt der programmierbare Eingabe/Ausgabebaustein (I/O) *INS8255N* diese mühevolle Aufgabe. Der Prozessor muss dann nur noch seine Daten an den IO-Chip übergeben und die Ergebnisse abholen, auch das war damals ein Novum, bis dahin wurden für solche Aufgaben viele TTL-Logic-Chips benötigt. Daneben finden sich nur noch zwei BCD-Dekodierer des Typs *DM74145N*, die Teile der LEDs ansteuern. —*caw*



Viele LEDs und Drahtbrücken auf der vermutlich per Hand gelöteten Platine



Unterseite der einseitig beschichteten Platine mit Bodges

Die animatronische Posteule, Teil 2

Mit Geduld und einer ruhigen Hand nehmen wir die kleine Eule in Betrieb: von der Einrichtung des Systems über die Kalibrierung der Servos bis hin zum Gehäuse mit Plüschüberzug.

von Ákos Fodor

Im ersten Teil im vorigen Heft habt ihr bereits auf engstem Raum erfolgreich Servo-Motoren mit Elektronik und ausgedruckten Teilen verschraubt. Jetzt erwarten euch ein feinteiliger Augenmechanismus, das schrittweise Kalibrieren der Servos, das schützende Gehäuse samt Plüschüberzug und die Anpassung der Programmierung an eure Bedürfnisse. Die 3D-Daten sowie Skripte findet ihr im Github-Repository zum Projekt. Ergänzend unterstützt euch ein Aufbauvideo (Link in der Kurzinfo). Habt Spaß, lernt und lasst euch Zeit.

Kurzinfo

- **Stromversorgung und Headless-System-Setup**
- **Servos kalibrieren und verbinden**
- **Audiowiedergabe mit pysinewave und pydub**
- **Mailabruf mit IMAPClient**

Checkliste

Zeitaufwand:
3–5 Stunden

Kosten:
15 Euro

Material

- **2 Lüfter,** 5V, 30mm × 30mm × 8mm
- **M2- und M3-Schrauben** in verschiedenen Längen
- **Kabel**
- **Doppelseitiges Klebeband**
- **Zahnstocher und Schaschlikspieße**

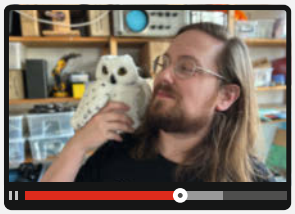
Alles zum Artikel im Web unter
make-magazin.de/x4ru

Werkzeug

- **Lötkolben**
- **Computer**
- **3D-Drucker**
- **Skalpell oder Cutter**
- **Seitenschneider**
- **Fingernagelschere**

Mehr zum Thema

- Ákos Fodor, Die animatronische Posteule, Teil 1, Make 1/2022, S. 8
- Video: Eulen-Aufbautipps, Teil 2

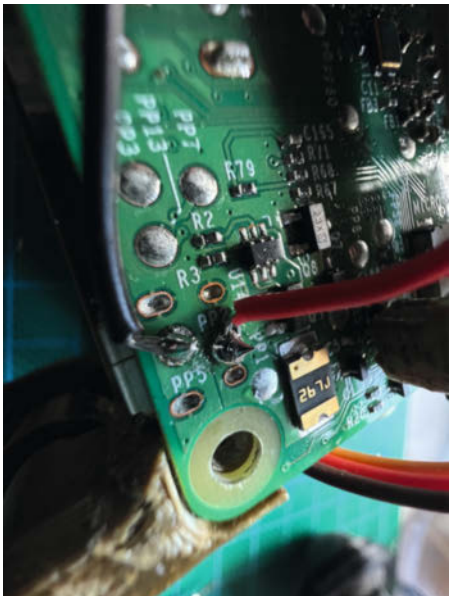


Strom

Der Raspberry Pi ist das einzige Bauteil, das wir bislang noch nicht mit Strom versorgt haben. Wie beim Audioausgang bringen wir mit dem Lötkolben zwei 20cm lange Kabel an der Rückseite des Mikro-USB-Anschlusses an. Dazu lösen wir kurz das Rückseitenbauteil, um besser an die Kontakte zu gelangen und löten ein schwarzes Kabel an Pin PP5 (GND) und ein rotes an Pin PP2 (5V) 1. Danach prüfen wir unter guten Lichtbedingungen, ob die Lötstellen sauber sind. Am besten kürzt ihr im Vorfeld die abisolierten Kabelenden auf 2-3mm, dann liegt nachher weniger frei. Ihr könnt eure Lötarbeit zusätzlich mit einem Tropfen Heißkleber versehen, nur um sicherzugehen, dass sich später beim Zusammenbau nichts aus Versehen löst. Zum Abschluss klemmen wir alle losen Kabel in die *DeLock*-Kupplung.

Raspberry Pi OS vorbereiten

Als System für die Eule verwenden wir das *Raspberry Pi OS Lite (64-Bit)*, d.h. eine *Headless*-Variante ohne grafische Benutzeroberfläche.



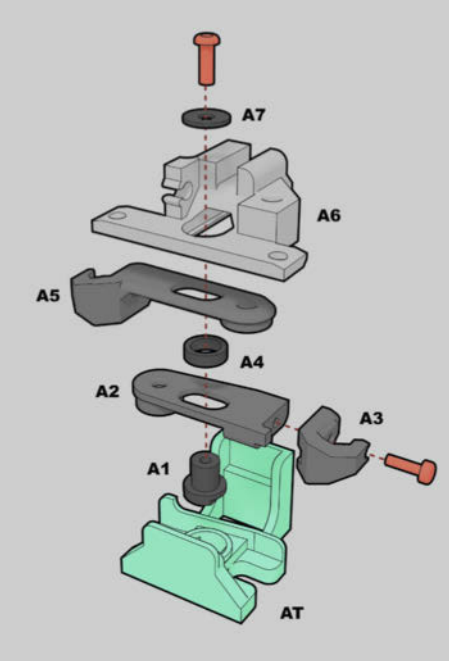
1 Wir befestigen für die Stromversorgung ein schwarzes Kabel an Pin PP5 und ein rotes an Pin PP2.

Ladet euch zunächst den aktuellen *Raspberry Pi Imager* (ab Version 1.7.1, siehe Link in der Kurzinfo) herunter. Dort könnt ihr anschließend aus dem OS-Menü die eben genannte Distribution auswählen. Daraufhin erscheint rechts unten im Programmfenster ein Zahnrad, über das ihr folgende Einstellungen vornehmt:

- Falls sich mehrere Raspberries im Netzwerk befinden, ändert den *Hostnamen*. Meine Eule heißt z.B. *bubo.local* .
- Aktiviert *SSH* und belasst die Einstellung auf *Passwort*.
- Ändert das Passwort des Benutzers *Pi*.
- Aktiviert das Kästchen bei *Wifi einrichten* und gebt die Informationen eures WLANs ein. Falls ihr ein zusätzliches 5GHz-Netzwerk habt, wählt die Zugangsdaten für das 2,4GHz-Netzwerk, da der Raspberry Pi 3B keine 5GHz unterstützt.
- Ändert das *Wifi-Land* auf *DE* sowie unter *Spracheinstellungen* das Tastaturlayout.
- Drückt zum Schluss auf *Speichern*, wählt die zu beschreibende SD-Karte aus und klickt auf *Schreiben*.

Nachdem ihr die SD-Karte in die Eule gesteckt habt, könnt ihr diese endlich das erste Mal einschalten. Gebt dem Pi beim Hochfahren ein paar Minuten Zeit für die Startkonfiguration. Die grüne LED hört irgendwann auf, beschäftigt zu blinken. Um euch via SSH mit der Eule zu verbinden, öffnet auf eurem Computer, der sich im selben Netzwerk befinden muss, die *Powershell* (Windows) oder das *Terminal* (macOS/Linux) und tippt in die Kommandozeile `ssh pi@bubo.local` , wobei *pi* euer Benutzername und *bubo.local* der Hostname des Raspberry sind, die ihr eingestellt habt. Das System wird euch sowohl nach dem Passwort als auch nach der Erlaubnis fragen, eine SSH-Verbindung zu diesem Raspberry herstellen zu dürfen. Nachdem ihr zugestimmt habt, seid ihr mit der Eule verbunden und könnt damit beginnen, die noch fehlenden Pakete für die Konfiguration zu installieren. Gebt dazu folgende Zeilen einzeln ein (in den drei Zeilen, die mit einem Bindestrich enden, gehört die nächste dazu) und bestätigt jeweils mit der Eingabetaste oder kopiert euch die Zeilen aus dem Github-Repository des Projekts:

```
sudo apt update
sudo apt full-upgrade
sudo apt install python3-pip
sudo apt install ffmpeg libavcodec-
extra
sudo apt install i2c-tools
sudo apt install libportaudio2
sudo pip3 install imapclient
```



2 Die Bauteile für den Augenmechanismus werden in das Werkzeug AT gestapelt und von oben mit einer M2-Schraube verbunden.

```
sudo pip3 install smbus
sudo pip3 install adafruit-
circuitpython-lis3dh
sudo pip3 install adafruit-
circuitpython-servokit
sudo pip3 install pysinewave
sudo pip3 install pydub
```

Damit der Raspberry Pi mit dem Servo-Controller kommunizieren kann, müssen wir noch die Schnittstelle I²C aktivieren. Gebt den Befehl `sudo raspi-config` ein, geht auf *Interfacing Options* und wählt dort *I2C* aus, um es einzuschalten. Startet anschließend den Raspberry mit `sudo reboot` neu und verbindet euch erneut mit `ssh pi@bubo.local` und eurem Passwort.

servoCalib.py

```
#!/usr/bin/python3
import time
from adafruit_servokit import \
  ServoKit
from time import sleep

kit = ServoKit(channels = 16)

kit.servo[0].angle = 90
time.sleep(1)
kit.servo[0].angle = None
```

Die Servos kalibrieren

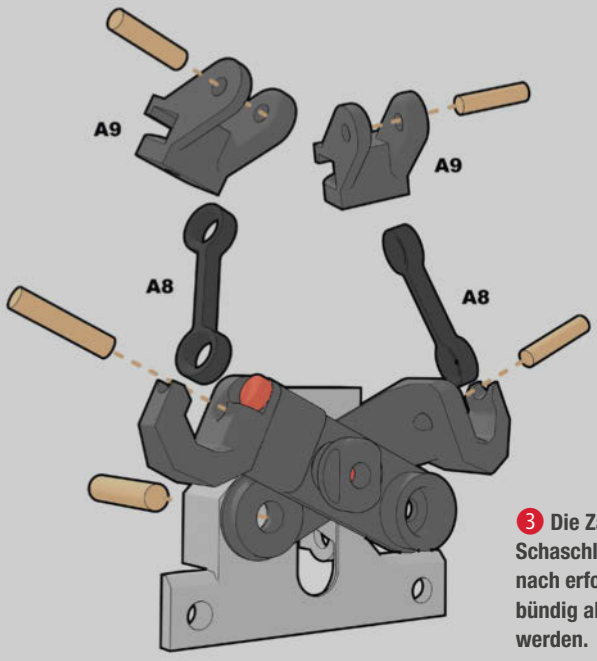
Die von mir verwendeten MG90S-Servo-Motoren haben einen Drehwinkel von 180 Grad. Da wir zum jetzigen Zeitpunkt nicht wissen, in welchem Winkel sie eingestellt sind, führen wir eine manuelle Ausrichtung durch, **bevor** wir sie mit den Mechanismen verbinden. Dazu erstellen wir das Skript *servoCalib.py* mithilfe des kompakten Editors *nano*, den wir über die SSH-Kommandozeile mit dem Befehl `nano` starten können. In das leere Dokument, das nach dem Öffnen erscheint, geben wir den Code aus dem *servoCalib.py*-Listing ein und speichern anschließend mit STRG+S. Achtet darauf, die Dateiendung *.py* mit anzugeben, weil nano sie nicht automatisch ergänzt. Beendet nano mit STRG+X und verwendet `nano servoCalib.py`, um das Skript erneut anzupassen.

Im Skript stellen wir mit `kit.servo[x].angle = 90` den anzusteuernden Servo-Winkel ein. Mit `kit.servo[x].angle = None` wird das PWM-Signal deaktiviert und der Servo entlastet. Anstelle des `x` könnt ihr die Zahl desjenigen Servos eingeben, den ihr ansteuern möchtet. Am Servo-Controller-Board, von oben nach unten gesteckt, sind die ersten vier Servos 0, 1, 2 und 3. Verwendet bitte die oberen vier Anschlüsse, da sich dort beim finalen Zusammenbau durch eine Aussparung im Gehäuse keine Kabel einklemmen werden.

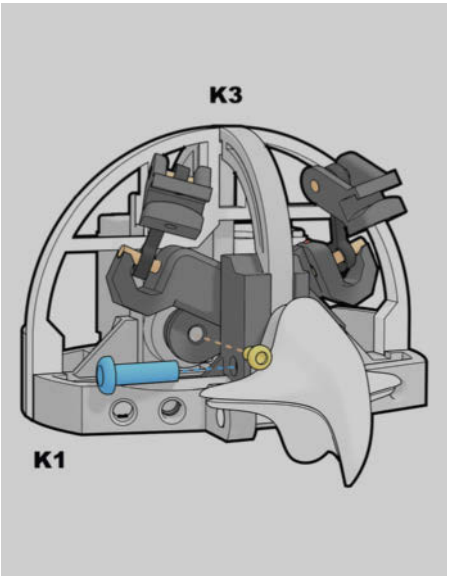
Augen – servo[0]

Dieser Servo steuert beide Augen parallel mithilfe eines Scherenmechanismus, den wir im Vorfeld ausdrucken und zusammengebaut bereitlegen können. Ihr benötigt dafür die mit A gekennzeichneten Bauteile:

- Schraubt als Erstes die beiden Augenmodule von **K1** ab und löst die Schraube, die **K1** und **K3** verbindet, sodass ihr **K3** nach oben herausziehen könnt (siehe Teilebezeichnung erste Anleitung).
- Baut den Scherenmechanismus wie in der Grafik 2 zusammen. Das Werkzeug **AT** hält die einzelnen Bauteile zusammen, sodass ihr von oben alles mit einer **M2 × 6**-Schraube verbinden könnt.
- Klappt die Arme 3 nach oben, befestigt mithilfe von Zahnstochern die Bauteile **A8** und **A9** sowie den Arm **A5** mit einem Schasch-



3 Die Zahnstocher und Schaschlikspieße sollten nach erfolgtem Stecken bündig abgeschnitten werden.



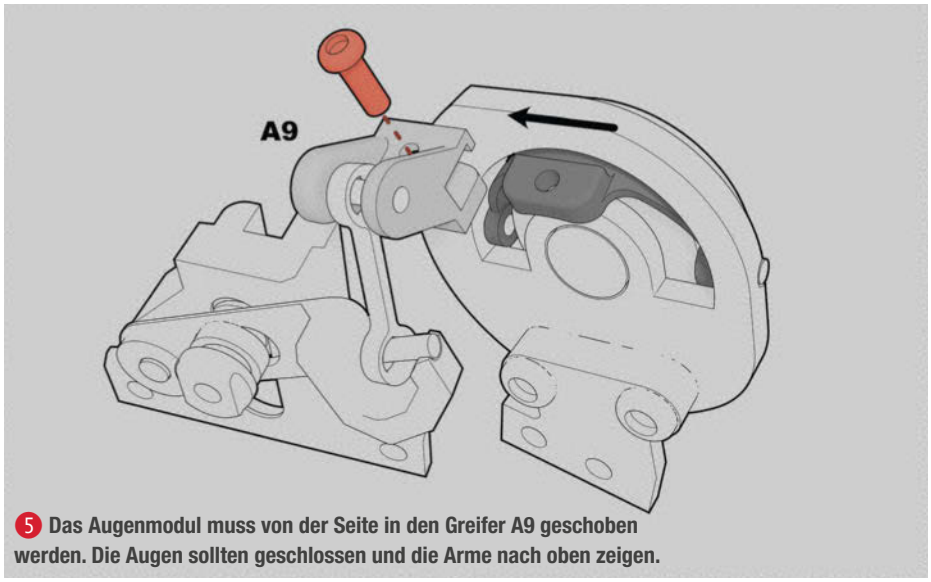
4 Der Augenmechanismus wird mit der Servo-Schraube an der Servo-Welle befestigt. K3 wird von oben in K1 zurückgesteckt und samt Mechanismus mit der M3-Schraube verbunden.

likspieß an **A6** und legt den Mechanismus beiseite.
- Definiert in *servoCalib.py* mit dem Befehl `kit.servo[0].angle = 90` die Ausrichtung des Augen-Servos auf 90 Grad. Führt den Befehl mit `python3 servoCalib.py` aus. Fahrt den Raspberry anschließend mit `sudo shutdown -h 0` herunter und trennt ihn vom Strom.
- Befestigt den Augenmechanismus am Kopf 4. Schraubt zuerst den Arm **A2** mithilfe der **M2,5**-Servo-Schraube in die Servo-Welle, sodass er fest sitzt. Dazu müsst ihr den oberen Schnabel etwas zur Seite drehen. Achtet darauf, die Servo-Welle dabei nicht zu drehen. Steckt dann von oben **K3** wieder auf **K1** und verbindet mit der **M3 × 10**-Schraube **K3**, **A6** und **K1**. Schraubt nicht zu fest.
- Schiebt als Letztes die Augenmodule seitlich auf die Bauteile **A9**, verschraubt sie vorsichtig mit **M2 × 6**-Schrauben und anschließend wieder an **K1**. Dabei verbinden sich die innenliegenden Schrauben mit **A6**. Achtet darauf, dass die **A8**-Verbinder am Ende richtig sitzen 5.
- Fahrt die Eule wieder hoch und verbindet euch erneut über SSH.

Als Nächstes ermitteln wir die Minimal- und Maximalwinkel für die Augenbewegung. Ich habe dafür den Winkel in *servoCalib.py* in kleinen Schritten verändert und den Vorgang wiederholt, bis ich die richtigen Werte hatte. Falls euer Servo fiept und heiß wird, hat er sich möglicherweise verkeilt, weil der Winkel zu weit war. Verringert daraufhin in jedem Fall den Wert im Skript und startet es erneut. Wundert euch nicht, wenn ihr nach dieser Prozedur immer wieder Phantomfiepen habt. Wichtig ist, dass der Motor nach `kit.servo[x].angle = None` kaum zu hören ist. Ihr werdet ein Gefühl dafür entwickeln. Notiert euch die ermittelten Werte und übertragt sie in ein neues Skript: *servoTest.py* . Hier wird nacheinander, mit einer zeitlichen Unterbrechung, zwischen den Minimal- und Maximalwinkeln gewechselt, sodass wir die Augenlider in Bewegung sehen. Experimentiert ein wenig mit den Werten. Um *servoTest.py* zu stoppen, STRG+C.

Schnabel – servo[1]

Die nächste Ausrichtung ist etwas einfacher. Ändert in *servoCalib.py* den Ausrichtungsbefehl auf `kit.servo[1].angle = 90` und führt ihn mit `python3 servoCalib.py` aus. Fahrt den Pi herunter und befestigt anschließend einen der mit dem Servo gelieferten Servo-Arme auf der Servo-Welle. Verbindet anschließend den Schnabel und den Servo-Arm mithilfe eines dünnen Drahtes 6. Beim Ermitteln der Maximalwinkel empfehle ich euch, den Schnabel nicht zu weit zu öffnen, da er sich beim Drehen des Kopfes später mit dem Flügelmechanismus oder der Schulter verkeilen könnte. Merkt euch die Maximalwerte und testet auch hier im Anschluss mit *servoTest.py* die Bewegung des Schnabels.



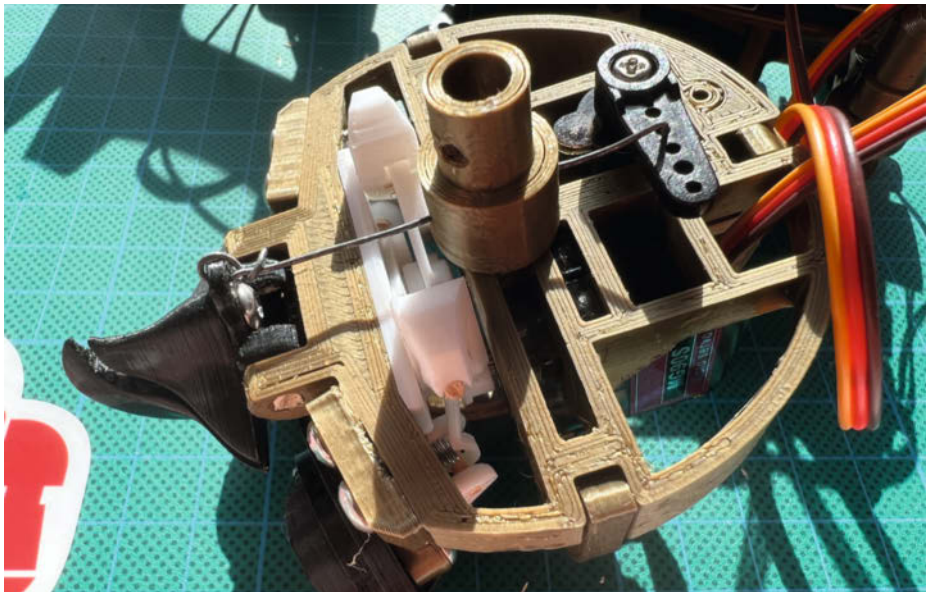
5 Das Augenmodul muss von der Seite in den Greifer A9 geschoben werden. Die Augen sollten geschlossen und die Arme nach oben zeigen.

Nacken – servo[2]

Wir starten wie gewohnt mit einem 90-Grad-Winkel in *servoCalib.py* für `servo[2]`. Nachdem der Winkel eingestellt und die Eule ausgeschaltet ist, steckt ihr den Eulenkopf auf den Hals, sodass der Kopf nach vorne bzw. zu euch schaut. Durch das in **N2** befindliche Loch markiert ihr **N1** mit einem Filzstift, nehmt den Kopf wieder ab und erstellt behutsam ein Loch in **N1**, z.B. mit der Fingernagelschere. Steckt anschließend den Kopf wieder auf **N2** und verschraubt beide Teile durch die Löcher mit einer **M3 × 6**-Schraube. In diesem Fall sind die Maximalwerte 0 und 180 und müssen nicht ermittelt werden.

servoTest.py

```
#!/usr/bin/python3
import time
from adafruit_servokit import \
  ServoKit
from time import sleep

kit = ServoKit(channels = 16)
eyesMin = #ermittelter Minimalwert
eyesMax = #ermittelter Maximalwert

while True:
  kit.servo[0].angle = eyesMax
  time.sleep(0.1)
  kit.servo[0].angle = eyesMin
  time.sleep(2)
  kit.servo[0].angle = None
  time.sleep(1)
```



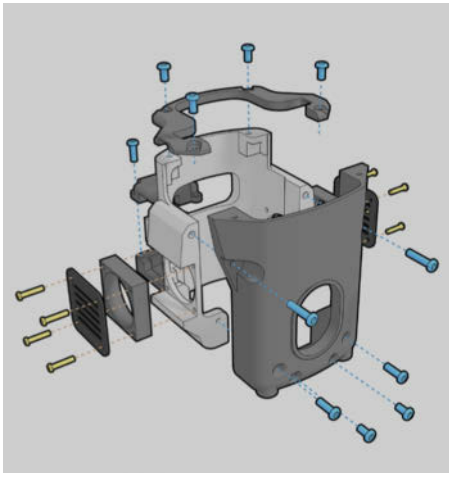
6 Um den Servo zu entlasten, ist der Draht mit dem zweiten Loch verbunden. Dadurch benötigen wir zwar eine größere Drehung, die Bewegung des Schnabels ist jedoch gering.

Flügel – servo[3]

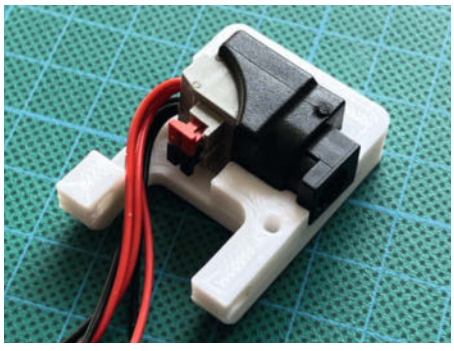
Als Letztes lassen wir die Eule flattern. Mechanisch sind wir auf ca. 90 Grad in der Bewegung beschränkt, von hängend bis angehoben. Für die Ausrichtung wählt ihr `kit.servo[3].angle = 90` und verschraubt, nach dem Herunterfahren, den kleinen Arm **B5** (siehe erster Artikel) so mit der Servo-Welle, dass er nach vorne zeigt, der Flügel also halb angehoben ist. Die Maximalwerte für die Bewegung sollten zwischen 45 und 135 Grad liegen. Je nach Genauigkeit des Servos müsst ihr diese Werte noch minimal anpassen. Vor allem die Drehung nach unten kann zu einem Verkeilen des Arms führen.

soundTest.py

```
#!/usr/bin/python3
import time
from pysinewave import SineWave
from time import sleep

sinewave = SineWave(pitch = 12)
sinewave.play()
time.sleep(2)
sinewave.stop()
```



7 Die Gehäuseteile werden mit M3-Schrauben um das Innenleben befestigt. Am einfachsten ist es, das Innenleben erst in die rückseitige Schale zu stecken.



8 Die DeLock-Kupplung passt nicht durch die Öffnung links unten und sollte zusammengeschraubt werden, bevor das Bauteil mit Schaschlikspießen am Gehäuse befestigt wird.

Audiowiedergabe

Auch wenn uns noch keine Audiodaten vorliegen, können wir mit dem Skript *soundTest.py* einen einfachen Ton für zwei Sekunden abspielen, um den Lautsprecher zu testen. Auf dem PAM-Modul ist ein Lautstärkesteller, der sich mit einem Kreuzschraubendreher verstellen lässt, falls die Lautstärke nicht stimmen sollte. Nutzt zum Ausführen `python3 soundTest.py`.

Zur Wiedergabe unserer finalen Eulen-Sounds verwenden wir die zu Beginn installierte *pydub*-Library für Python. Auf der Webseite *xeno-canto.org* findet ihr eine Fülle von echten Eulenaufnahmen zum kostenlosen Download, die ihr z.B. mit *Audacity* zurechtschneiden könnt. Oder ihr verwendet Sounds, die gar nichts mit Eulen zu tun haben. Da sind eurer Fantasie keine Grenzen gesetzt. Die Audiodateien habe ich mit *Filezilla* über eine gesicherte SFTP-Verbindung auf die Eule übertragen. Nachdem ihr das Programm heruntergeladen und gestartet habt, seht ihr im oberen Fensterbereich die Felder *Server*, *Benutzername*, *Passwort* und *Port*. Als Server verwendet ihr z.B. *bubo.local*, euer Benutzer ist vermutlich *Pi* und das Passwort habt ihr im Kopf. Zum Schluss gebt ihr *22* beim Port ein und drückt auf *Verbinden*. Nach einem kurzen Moment seht ihr links die Inhalte eures Computers und rechts die des Raspberry Pi. Zieht die Audiodateien zum Kopieren von links nach rechts in das Hauptverzeichnis des Pi. Im finalen Skript verwenden wir für jeden Sound eine Variable, wie z.B. `sndHoot = AudioSegment.from_wav('owlHoot.wav')` und können diese mit `play(sndHoot)` abspielen.

Finales Zusammenbauen

Jetzt ist es endlich so weit. Wir verpacken das Innenleben der Eule in ein schützendes Gehäuse. Hierfür benötigt ihr die mit G gekennzeichneten ausgedruckten Bauteile 7. Achtet darauf, die *DeLock*-Kupplung zusammenzuschrauben 8, bevor ihr das Bauteil an der Eulen-Gehäuserückseite befestigt. Für den Aufbau mit Plüsch-Mantel empfehle ich, links und rechts jeweils einen 5V-Lüfter mit der Windrichtung nach außen zu verwenden. Diese lassen sich auch direkt an der Kupplung mit Strom versorgen. Meine sind unterschiedlich stark, sodass ich noch einen Widerstand oder ein Poti dazwischen löten werde, um die Lüfterlautstärke zu regeln. Beim Zusammenbauen des Gehäuses ist es wichtig, dass die Kabel am Raspberry mittig und nicht an den Seiten verlaufen. In den 3D-Daten im Github-Repository findet ihr eine GPIO-Abdeckung, die unsere Crimp-Stecker davor schützt, aus Versehen herauszurutschen und den Kontakt zum Servo-Controller zu verlieren. Die Servo-Motoren-Kabel habe ich beim Zusammenbau durch die Öffnung an der Rückseite gezogen, um den Raspberry einfacher in die Schale schieben zu können. Wendet keine Gewalt an. Wenn etwas klemmt, nehmt das Innenleben wieder heraus und versucht es erneut. Ich weiß, dass es viele Kabel sind.

Wenn schließlich alles verschraubt ist, können wir den Rumpf mit heruntergeklapptem Schwanzteil in den Plüsch zwängen. Damit die Flügel sich nachher frei bewegen können, müssen wir in den Plüsch unterhalb der Schultern noch Löcher schneiden 9. Zieht den Plüsch anschließend bis über die Schultern. Schneidet ebenso Löcher an den Lüftungsschlitzen. Wenn ihr die Abdeckungen in Weiß druckt, fallen sie kaum auf. Alternativ lassen sie sich auch von außen anschrauben, nachdem das Gehäuse im Plüsch sitzt. Befestigt die Füße an den Beinen und verschraubt diese von außen mit dem Rumpf 10. Damit der Plüsch am Kopf hält, verwende ich ein dickes doppelseitiges Klebeband, das ich in kleinen Stücken aufklebe 11. Geschafft! Die Eule ist komplett.

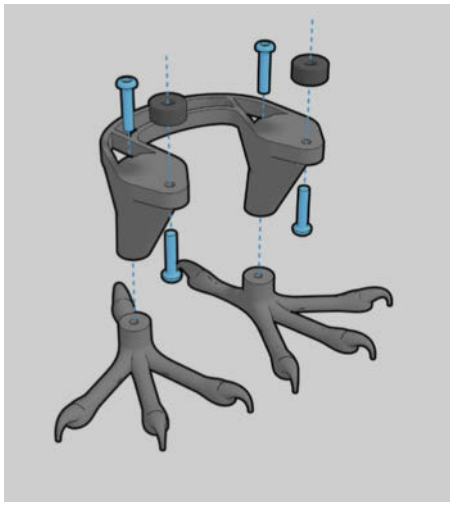
E-Mail-Abfrage vorbereiten

Da sie jetzt flattert und schnattert, ist es an der Zeit, diese Fähigkeiten praktisch zu verknüpfen. Zunächst benötigt ihr einen eigenen E-Mail-Account für die Eule, auf die E-Mails von eurer Hauptadresse weitergeleitet werden. **Achtung:** Verwendet **unter keinen Umständen** euren persönlichen E-Mail-Account in dem endgültigen Skript, da der Benutzername und das Passwort unverschlüsselt gespeichert sind und das Skript im letzten Schritt den gesamten Posteingang löscht. Ein kostenloser Gmail-Account ist schnell erstellt und bietet für die erweiterte Sicherheit *App-Passwörter* an, die wir löschen können, falls wir sie nicht mehr benötigen. Nach der Account-Erstellung klickt ihr rechts oben auf den Kreis mit eurer Initiale, wählt *Google-Konten verwalten* aus und klickt im Menü links auf den Punkt *Sicherheit*. Scrollt danach herunter, bis ihr *Bestätigung in zwei Schritten* seht und aktiviert die *Zwei-Faktor-Authentifizierung*. Erst danach erscheint im Menü Sicherheit der Punkt *App-Passwörter*, über den ihr ein neues Passwort generieren und später im Skript einfügen könnt. Wechselt danach im Browser zu eurem eigentlichen E-Mail-Anbieter und erstellt Regeln für die Weiterleitung von E-Mails. Ich habe z.B. bei GMX eingestellt, dass E-Mails von bestimmten Personen immer auch an die Eulen-E-Mail-Adresse gesendet werden.

Finales Skript

Das Skript *bubo.py* beinhaltet alles Notwendige, um die Eule final in Betrieb zu nehmen.

9 Mit Skalpell und Nagelschere schneiden wir an den Seiten Löcher für die Flügel und die Lüftung. Zieht anschließend den Plüsch über das Schulterteil.



10 So lassen sich die Beine mit den Füßen und dem Gehäuse verbinden. Für die Verwendung ohne Plüsch können die Platzhalter-Ringe weggelassen werden.

11 Gepolstertes doppelseitiges Klebeband hält den Plüsch am Kopf.

Neben der kompletten E-Mail-Abfrage findet ihr einzelne Bewegungen für Augen, Flügel, Schnabel und Kopf sowie Platz für Bewegungsabläufe und die Audioeinbindung, die ihr nach Belieben anpassen könnt.

Die Hauptfunktion `checkIncoming()` verbindet sich regelmäßig mit dem E-Mail-Account der Eule und fragt die im Posteingang liegenden Nachrichten ab. Je nach Absender wird eine individuelle Animation abgespielt. Abschließend löscht das Skript den gesamten Posteingang und prüft nach der Pause `time.sleep(checkFreq)` erneut, ob neue E-Mails eingegangen sind.

Die einzeln angelegten Animationen können zu Bewegungsmustern choreografiert werden. Das Listing *Augen-Animation* zeigt, wie die Augen-Servos sich mithilfe einer `while`-Schleife verlangsamt von ihrer aktuellen Position `eyesPos` zum Zielwinkel `eyesAim` bewegen. Mit der Variable `aniSpeed` werden Pausen zwischen den Winkelschritten eingefügt, die sich auf die Animationsgeschwindigkeit auswirken. Ladet *bubo.py* am besten mit Filezilla in das Hauptverzeichnis der Eule und ergänzt das Skript in nano mit euren Servo- und E-Mail-Informationen.

Autostart und Read-Only

Damit die Eule nach dem Einschalten direkt einsatzbereit ist, geben wir über SSH den Befehl `crontab -e` ein und wählen `1` für nano aus. In dem daraufhin erscheinenden Dokument schreiben wir in die letzte Zeile `@reboot /home/pi/bubo.py`, speichern mit STRG+S und schließen nano mit STRG+X. Als letzten Schritt erlauben wir mit `chmod a+x bubo.py`, dass die Datei beim Neustart ausgeführt werden darf.

Wenn ihr beim Raspberry Pi den Stecker zieht, ohne ihn herunterzufahren, kann das zu einem Fehler auf der SD-Karte führen. Damit das nicht passiert, versetzen wir die Eule in einen *Read-Only-Modus*, bei dem nicht mehr auf die SD-Karte geschrieben wird und sich nach einem Neustart alles auf den voreingestellten Zustand zurücksetzt. Diese Option aktiviert ihr in den Einstellungen über `sudo raspi-config`. Wählt den vierten Menüpunkt *Performance Options* aus und aktiviert das *Overlay File System*. Die Frage nach dem Schreibschutz könnt ihr verneinen. Ab jetzt darf die Eule bedenkenlos von der Steckdose gezogen werden, ohne dass ihr Gefahr lauft, euer System damit zu beschädigen. Bedenkt, dass ihr für spätere Änderungen das Overlay File System deaktivieren und anschließend wieder aktivieren müsst.

Abschließend möchte ich noch darauf hinweisen, dass die animatronische Posteule kein Spielzeug, sondern ein Proof-Of-Concept mit experimenteller Elektronik ist und nicht unbeaufsichtigt verwendet werden sollte. Jetzt aber: Viel Spaß und viel Erfolg beim Nachbau! *—akf*

Augen-Animation

```
def aniEyesClose():
  global eyesPos
  eyesAim = eyesMin

  while eyesPos != eyesAim:
    if eyesPos <= eyesAim:
      eyesPos += 1
    else:
      eyesPos -= 1

  kit.servo[0].angle = eyesPos
  time.sleep(aniSpeed)

  kit.servo[0].angle = None
```

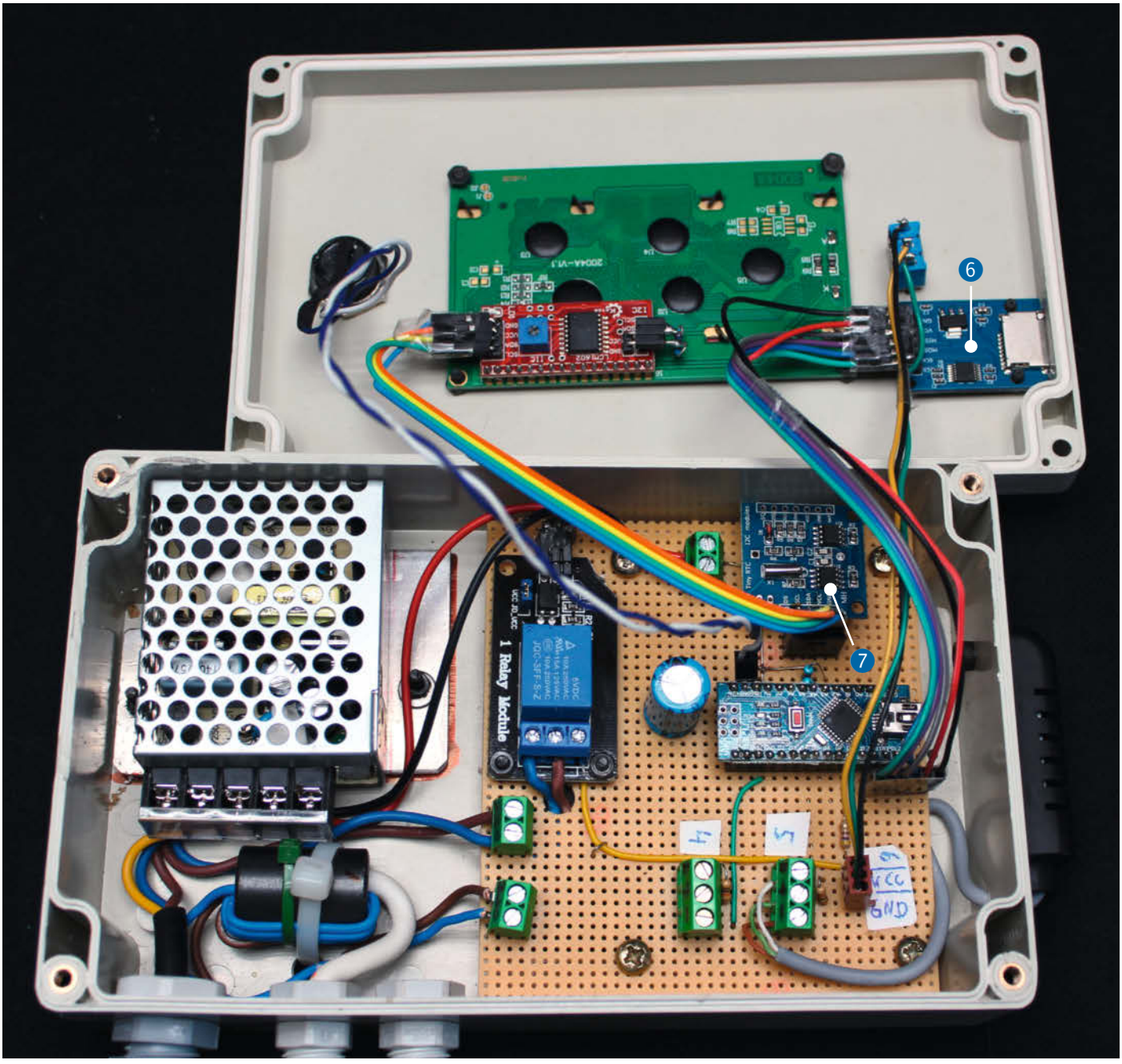
Eintreffende E-Mails

```
def checkIncoming():
  server = IMAPClient(HOSTNAME, use_uid=True, ssl=True)
  server.login(USERNAME, PASSWORD)
  server.select_folder('Inbox')
  person1 = server.search(['FROM','person1@gmail.com'])
  person2 = server.search(['FROM','person2@gmx.de'])

  if(len(person1) > 0):
    aniPerson1()
  elif(len(person2) > 0):
    aniPerson2()

  all = server.search('ALL')
  for mail in all:
    server.delete_messages([mail])

  time.sleep(checkFreq)
```

Logging-Funktion für das Taupunkt-Lüftungssystem

Der Keller wird dank unseres Taupunkt-genau gesteuerten Lüfters aus der vergangenen Make-Ausgabe kontinuierlich trockener – aber wie lange läuft der Lüfter jeweils und wie schnell sinkt dadurch die Luftfeuchtigkeit im Keller? Zeit, dass jemand kontinuierlich mitschreibt. Das erledigt der ohnehin schon eingebaute Arduino doch mit links – oder?

von Ulrich Schmerold



Nach der Veröffentlichung unseres Artikels *Das Taupunkt-Lüftungssystem* in der Make 1/22 waren wir doch einigermaßen überrascht von der Resonanz. Bereits einen Tag nach Erscheinen des Hefts, berichteten stolze Maker, dass ihr System bereits läuft. Es gingen auch jede Menge Fragen nach weiteren Details wie Lüfterleistung, der Verschaltung des Modus-Wahlschalters oder nach der im Artikel als Möglichkeit erwähnten Datenlogging-Funktion. Bei vielen Fragen konnten wir schnell helfen – so entstand die Online-FAQ zum Taupunktlüfter (siehe Link in der Kurzinfo), die bei Bedarf fortlaufend erweitert wird. Das Datenlogging hingegen erfordert etwas mehr Aufwand und kann nicht in wenigen Worten beschrieben werden. Somit haben wir uns entschlossen, eine Lösung dafür in diesem Fortsetzungsartikel zu beschreiben. Wie sich zeigte, war es gar nicht so einfach, ein leicht umzusetzendes und komfortables System zu entwickeln.

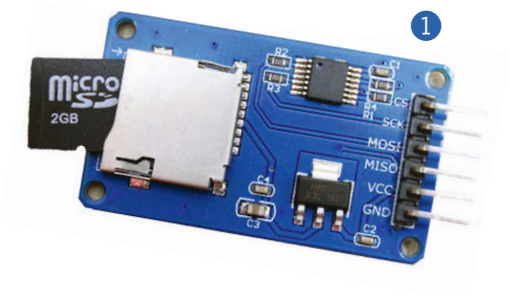
Alles auf eine Karte

Fürs Datenlogging gibt es viele Möglichkeiten – ein Vorschlag war etwa, das Netzwerkprotokoll *MQTT* (Message Queuing Telemetry Transport) zu benutzen. Allerdings braucht man dafür WLAN im Keller und auch der übrige Aufwand ist nicht wirklich nötig, wenn man vor allem sichergehen will, dass alles funktioniert wie gedacht. Dazu muss man die Daten nicht unbedingt in Echtzeit abrufen können, denn die Veränderungen der Feuchtigkeit im Keller vollziehen sich recht langsam und die entstehenden Graphen sind ohnehin erst nach einigen Tagen interessant.

Deshalb entschieden wir uns für eine Aufzeichnung auf Mikro-SD-Karte. Mikro-SD-Kartenleser für den Arduino (1) werden bei eBay schon für unter 3 Euro angeboten. Sie arbeiten mit 3,3V oder 5V und werden über das SPI-Protokoll angesprochen. Der richtige Anschluss an den Arduino ergibt sich aus dem Schaltplan (2).

Bei der Auswahl der SD-Karte ist zu beachten, dass der Leser nur SD-Karten erkennt, die maximal 2GB groß sind. SDHC-Karten sollten nach Angabe des Herstellers auch bis maximal 32 GB funktionieren. Da die Logdaten relativ klein sind, ist es aber eher unwahrscheinlich, dass eine 2GB-Karte nicht ausreicht.

Den Kartenleser haben wir im Gehäusedeckel an der rechten oberen Seite eingebaut (3). Um die Aussparung nicht auf die ganze



(1)

Kurzinfo

- Automatisch Daten protokollieren: Temperatur, Luftfeuchtigkeit, Taupunkt innen und außen, Lüfterlaufzeit
- SD-Kartenleser und Echtzeituhr einbauen
- Darstellung von CSV-Daten mit Freeware

Checkliste

Zeitaufwand:
4 bis 6 Stunden

Kosten:
ca. 20 Euro (wenn das Lüftungssytem aus Make 1/22 bereits vorhanden ist)

Elektronik:
Löten von bedrahteten Bauteilen

Programmieren:
Arduino IDE bedienen

Mehr zum Thema

- Ulrich Schmerold, Das Taupunkt-Lüftungssystem, Make 1/22, S.22
- FAQ zum Taupunkt-Lüftungssystem (Online, siehe Link)
- Tipps bei Speicherproblemen (Online, siehe Link)

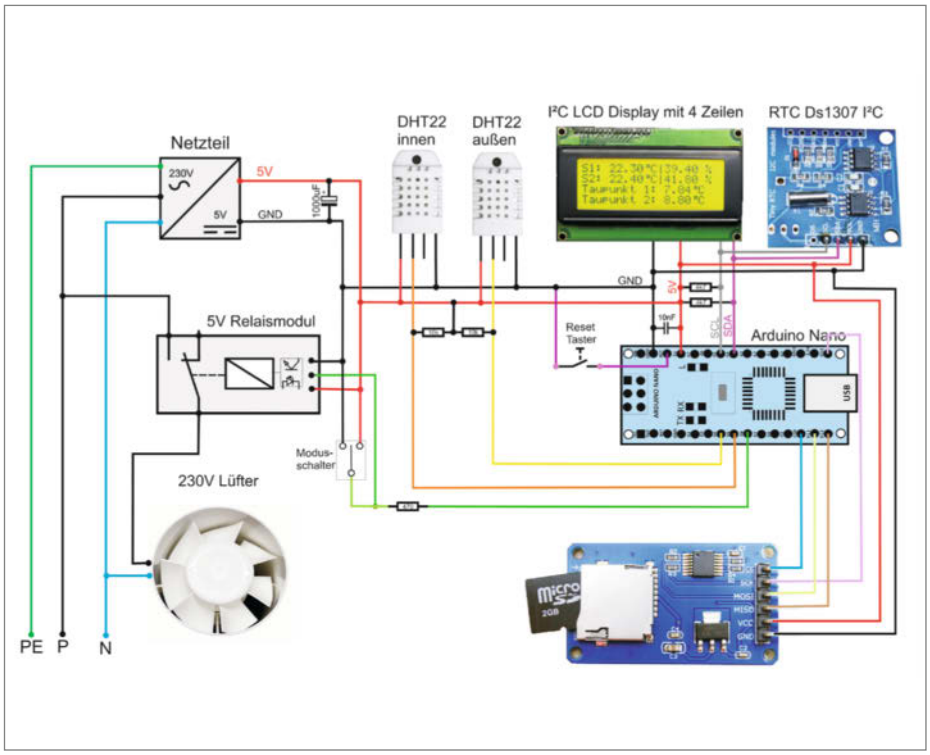
Material

- **microSD-Kartenleser** 5V für Arduino
- **microSD-Karte** bis zu 2GB Kapazität
- **I2C-Real-Time-Clock-Modul** RTC DS1307
- **Li-Ion-Akku-Knopfzelle** LIR2032, 3,6V
- **6 Jumperkabel** female / female
- **10-Pin-Anschluss Stiftleiste** 2,54mm
- **4-Pin-Anschluss Buchsenleiste** 2,54mm

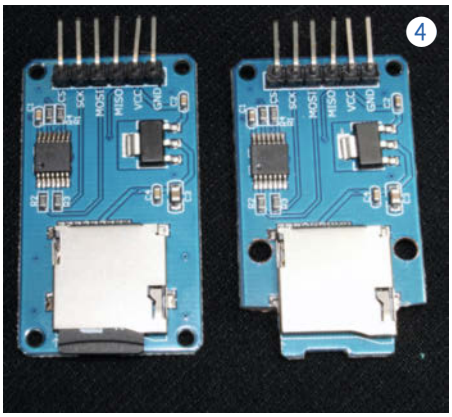
Werkzeug

- **Bohrmaschine** nebst diversen Bohrern
- **Schlüsselfeile**
- **Lötkolben** und Zinn
- **PC** mit Micro-SD-Karten-Leser

Alles zum Artikel im Web unter
make-magazin.de/x1z5



(2) Der komplette Schaltplan zum Taupunktlüfter umfasst hier zusätzlich zur Version aus dem vorigen Heft den Modus-Schalter, den Reset-Taster, die Real Time Clock sowie den SD-Kartenleser.



Länge der Leserplatine feilen zu müssen, haben wir die Platine rechts und links neben der Öffnung für die Karte abgefeilt. Zudem haben wir zwei neue 3mm-Bohrungen für die Befestigung der Platine hergestellt, da ohnehin keine 2mm Schrauben zur Hand waren. Bild 4 zeigt die Platine vor und nach diesen Modifikationen.

Zeit nehmen

Erst jetzt fiel uns aber ein wichtiges Detail auf: Ohne Datum und Uhrzeit macht eine Protokollierung nur wenig Sinn. So kauften wir bei eBay noch ein I²C-Echtzeitmodul (RTC für *Real Time Clock*) vom Typ *RTC DS1307*. Dieses Modul hält mittels Lithium-Ionen-Akkuzelle die Uhr am Laufen, wenn der Arduino die Spannung verliert.

Da wir schon für das LC-Display das I²C-Protokoll einsetzten, ist der Anschluss des RTC-Moduls trivial: Einfach parallel zum Display anschließen und fertig! Auf Bild 5 ist das RTC-Modul zu sehen (der Akku befindet sich auf der Rückseite), davor die Anschlussleiste für das Display. Die I²C-Anschlüsse wurden eins zu eins durchverbunden.

Eine Empfehlung: Kaufen Sie ein RTC-Modul ohne Akku und erwerben diesen separat, da bei unseren Einkäufen die Akkus, die sich bereits in den RTC-Modulen befanden, oft entweder fast oder bereits ganz defekt waren. So verliert dann auch die RTC ihre Zeit, und das ist ja nicht Sinn der Sache.

Auf dem Titelfoto zum Artikel ist der komplette Aufbau des Taupunktlüfters inklusive Datenlogging-Erweiterung zu sehen: Die Anschlüsse für SPI haben wir auf eine 6-polige Anschluss-Stiftleiste geführt und mit Jumperkabel (female/female) zum SD-Kartenleser verbunden 6.

Das RTC-Modul erhielt eine 4-polige Stiftleiste und als Gegenstück die RTC-Platine eine 4-polige Buchsenleiste 7. So lässt sich das Modul bequem herausziehen, falls der Akku dann doch einmal ersetzt werden muss.

Der Arduino-Code

Nachdem alles montiert und verkabelt war, ging es an den Code. Neben den schon für den Taupunktlüfter in der Version ohne Datenlogging verwendeten Bibliotheken werden wie gehabt und im vergangenen Heft beschrieben jetzt noch drei weitere eingebunden:

```
#include <DS1307RTC.h>
#include <SD.h>
#include <SPI.h>
```

Vor der ersten Verwendung der RTC müssen die aktuelle Uhrzeit und das Datum übertragen werden. Hierfür verwendet man am einfachsten den Beispielsketch *SetTime*, der bei der Installation der RTC-Bibliothek *DS1307RTC* mit installiert wurde.

Der Sketch *SetTime* wird dabei einfach auf den Arduino geladen und ausgeführt. Nun werden das Datum und die Uhrzeit des Compilers in die RTC geschrieben.

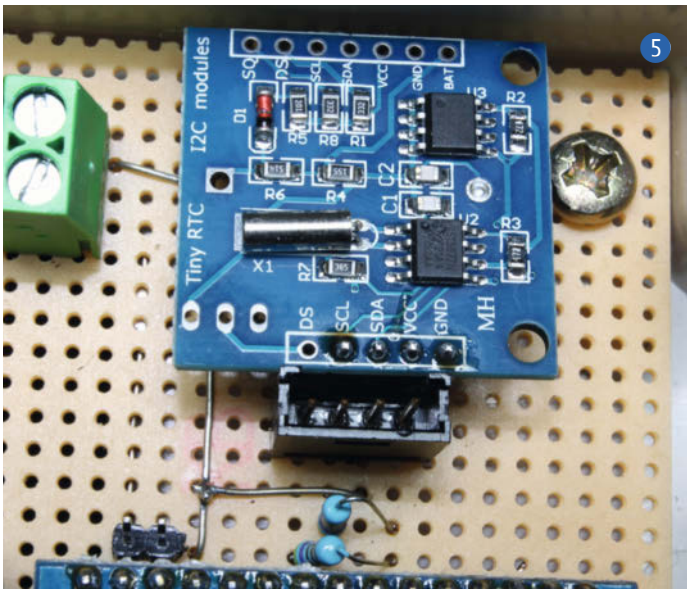
Anschließend kann der Sketch *ReadTest* ausprobiert oder gleich der Sketch *Taupunkt_Lüfter_Datenlogging* (Download siehe Link in der Kurzinfo) auf den Arduino übertragen werden.

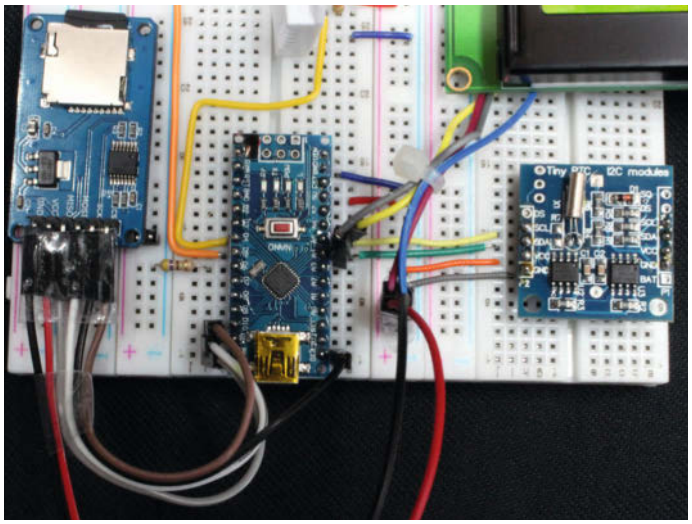
Das Datenlogging-Programm testet zu Beginn, ob die RTC vorhanden ist und ein gültiges Datum sowie eine Uhrzeit besitzt. Anschließend wird getestet, ob eine SD-Karte verfügbar ist. Die Ergebnisse der Tests werden jeweils im LC-Display angezeigt.

Treten dabei Fehler auf, wird das Datenlogging abgeschaltet und nur der Taupunktlüfter betrieben. Deshalb sollte man immer die Reset-Taste drücken, wenn man die SD-Karte vorübergehend zum Auslesen herausgezogen und sie dann wieder eingesetzt hat, damit das System wieder mit aktivierter Logging-Funktion startet. Sollten auch bei den Sensoren Fehler auftreten, so wird das Programm (wie in der bisherigen Version ohne Logger auch) immer wieder neu gestartet, bis wenigstens die Sensoren funktionieren.

Speicherproblem

Als alles fertig programmiert und auf den Arduino übertragen war, kam die Enttäuschung: Das Programm stürzte ständig ab. Auch stundenlanges Suchen und endlose Tests brachten keinen Erfolg. Der Fehler trat auch immer wieder an verschiedenen Stellen des Programms





An dem Breadboard-Aufbau für den Test lässt sich der Anschluss des SD-Kartenmoduls via SPI und der Anschluss des RTC-Moduls via I²C gut erkennen.

auf. Wir waren uns irgendwann zu hundert Prozent sicher, dass der Code fehlerfrei war. Nach langen Recherchen in Arduino-Foren brachte ein Beitrag über die Speichernutzung endlich den Erfolg. Was dahintersteckte und wie wir das Problem lösten, lesen Sie online (siehe Link in der Kurzinfo).

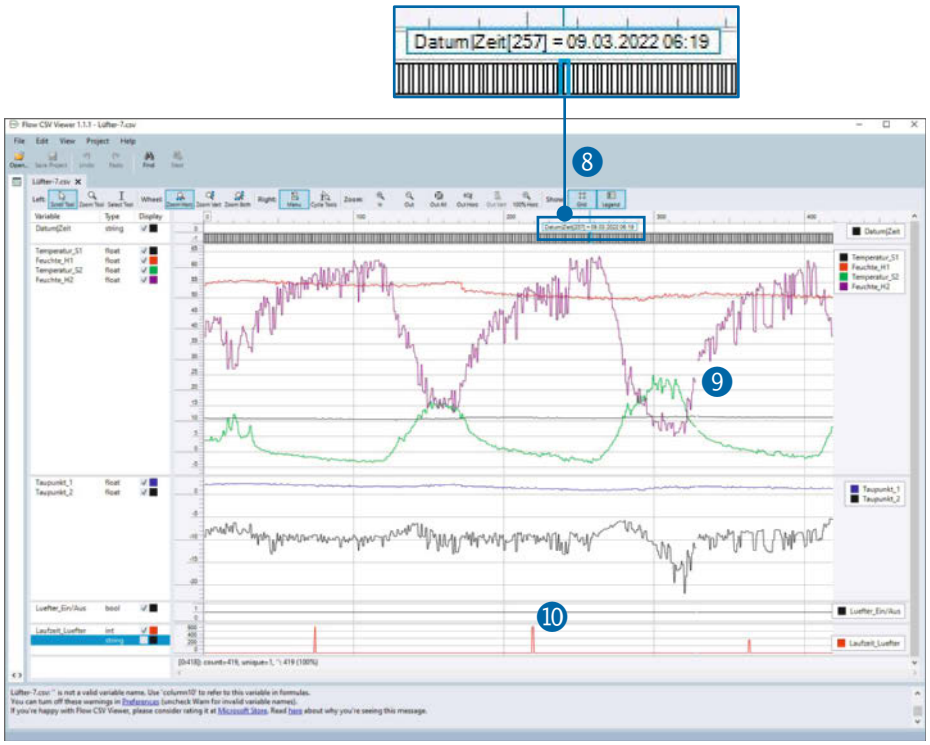
Output

Mit dem speicheroptimierten Code zeichnet der Taupunktlüfter auf seiner SD-Karte folgende Daten auf: Innen-Temperatur, Innen-Luftfeuchte, Innen-Taupunkt, Außen-Temperatur, Außen-Luftfeuchte, Außen-Taupunkt, Lüfter Ein/Aus, Lüfterlaufzeit pro Tag in Minuten. Zusätzlich wird auch gespeichert, wenn die Steuerung neu startet.

Auf einer leeren SD-Karte wird zuerst die Datei *Luefter1.csv* erstellt und darin eine Headerzeile für die Bezeichnung der einzelnen Loggingwerte geschrieben. Den Dateinamen und die Headerzeile kann man selbstverständlich im Quellcode verändern. Der Dateiname muss dabei allerdings dem Format 8.3 entsprechen, also maximal 8 Zeichen vor dem Punkt und 3 Zeichen danach.

Das Standard-Dateiformat .csv (*Comma-separated Values*) ist ein simples Textformat mit Kommas (es können aber auch Strichpunkte sein) zur Trennung der einzelnen Werte. Viele Programme unterstützen dieses Format, etwa auch *Microsoft Excel* oder *Libre/OpenOffice*. Zur schönen Darstellung der Werte unter Windows gibt es aber auch die Freeware *Flow CSV Viewer* (Download siehe Link in der Kurzinfo). Das Programm wird seit 2019 fortlaufend aktualisiert und Updates bereitgestellt. Die aktuelle Version 1.1.1 ist vom 03.03.2022.

Nach der Installation von Flow CSV Viewer lässt sich unsere Datei *Luefter1.csv* problemlos öffnen. Ohne weitere Einstellungen erscheinen sofort die aufgezeichneten Graphen und es



wurden bereits als Bezeichnung die Namen aus der Headerzeile verwendet. Der Umgang mit der Software gelingt rein intuitiv auch ohne das Studium der Help-Dokumente. Es lässt sich bequem per Mausrad in die Graphen rein- oder rauszoomen. Als Bezugspunkt wird die Position gewählt, an der sich gerade der Mauszeiger befindet.

Ganz links lassen sich die Graphen per Drag & Drop neu zuordnen. Standardmäßig werden alle Daten von einem Sensor zusammen dargestellt, aber auf Wunsch kann man so stattdessen die Temperatur- und Taupunktkurven gruppieren. Auch die Farben der Graphen kann man hier ändern.

Das Datum und die Uhrzeit werden in der obersten Zeile angezeigt, sind dort aber nicht lesbar, bis man eine Zelle mit der Maus berührt (8). Wird die Steuerung neu gestartet (oder fällt zeitweise aus), so zeigt sich dies in einer Lücke im Graphen (9).

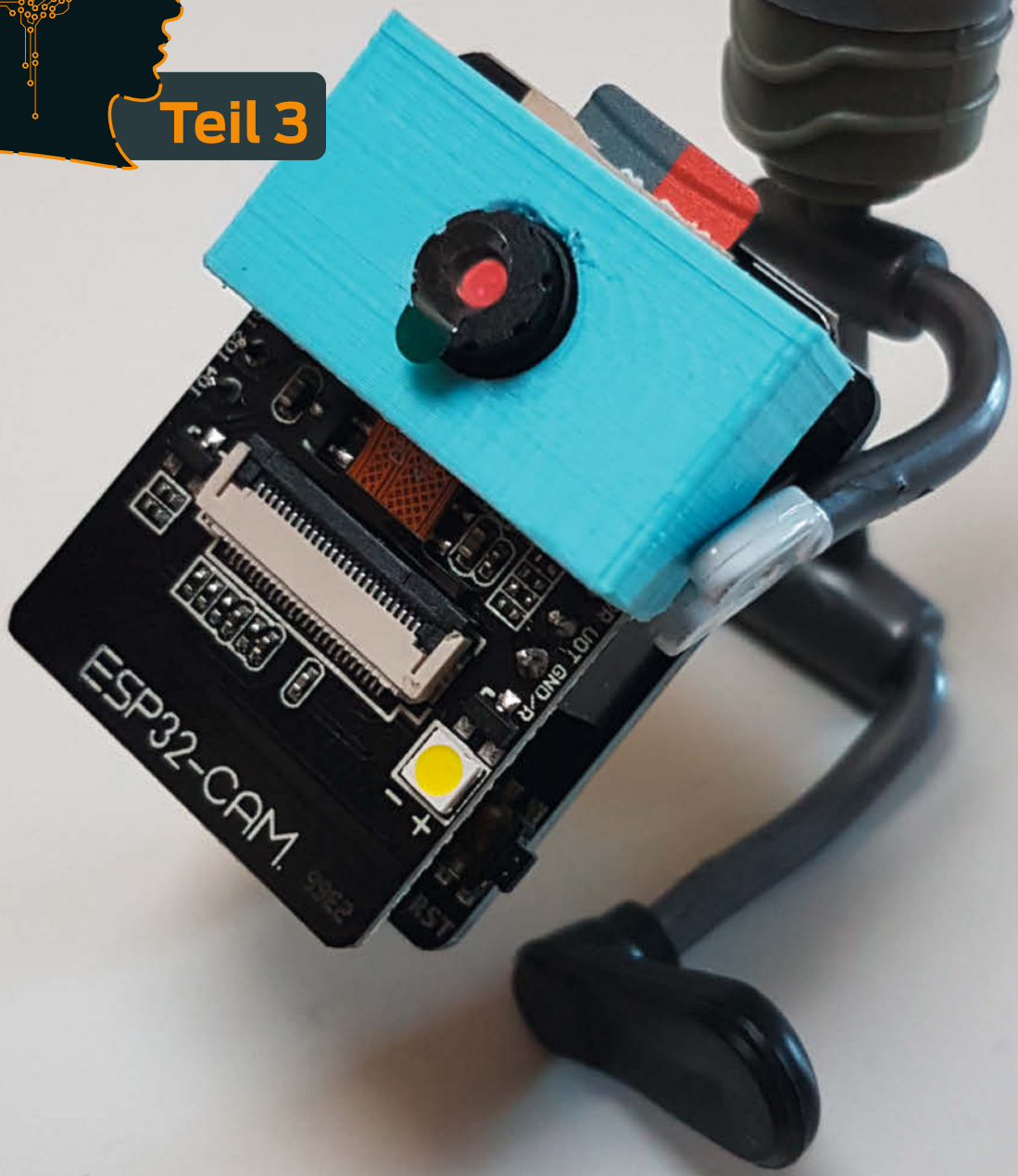
Die Lüfterlaufzeit (rote Kurve ganz unten (10)) wird noch nicht optimal dargestellt. Berührt man die Spitzen der Kurve mit der Maus, so wird jedoch die Laufzeit pro Tag in Minuten angezeigt.

Mit der beschriebenen Logging-Funktion lässt sich jetzt schön verfolgen, ob das Taupunkt-Lüftungssystem arbeitet, wie es soll, und dazu noch dokumentieren, wie der Keller langsam immer trockener wird. Und das alles im Warmen am Rechner in der Wohnung, in den man ab und zu die SD-Karte zum Auslesen steckt. —*pek*

KI für den ESP32

Neuronale Netze auf Mikrocontrollern: Mit TensorFlow Lite und einer einfachen Klasse in C++ lässt sich künstliche Intelligenz einsteigertauglich auf Embedded-Systemen implementieren. Wir zeigen, wie man das mit der ESP32CAM macht.

von Josef Müller



Im dritten und letzten Teil unserer Artikelserie zeigen wir, wie man TensorFlow Lite auf einer ESP32-CAM zur Ziffernerkennung betreibt. Die Kernidee, nämlich die Verwendung von neuronalen Netzen zur Bilddigitalisierung, steckt in nur zwei zentralen Bibliotheken und relativ wenigen Zeilen selbst geschriebenem Code. Davon ausgehend lassen sich leicht eigene Projekte mit eingebetteter KI ableiten.

Bevor es jedoch an die Programmierung (oder an die Erklärung unseres Codes) geht, muss zunächst die Hardware und die Programmierumgebung vorbereitet werden.

Kurzinfo

- **Visual Studio Code installieren, konfigurieren und bedienen**
- **Neuronale Netze in TensorFlow Lite und C++ implementieren**
- **ESP32CAM flashen und neuronales Netz testen**

Checkliste

Zeitaufwand:
1 Stunde

Kosten:
10 Euro

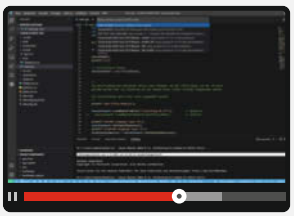
Material

- **ESP32CAM** (AI Thinker)
- **USB2Serial-Wandler** plus Jumper-Kabel
- Alternativ: **Development Board mit CH340**

Alles zum Artikel im Web unter
make-magazin.de/xvc5

Mehr zum Thema

- Josef Müller, KI für den ESP32, Teil 1, Make 6/21, S. 48
- Josef Müller, KI für den ESP32, Teil 2, Make 1/22, S. 74
- Josef Müller, ESP32CAM liest Wasseruhr, Make 2/21, S. 14
- Daniel Bachfeld, Gesichtssteuerung, Make 2/20, S. 16
- Video: Visual Studio Code installieren



Hardware

Typischerweise wird der ESP32 in DIY-Projekten in einem integrierten Modul eingesetzt, auf dem extern notwendige Beschaltungen wie WLAN-Antenne oder Spannungswandler bereits vorhanden sind. Die ESP32CAM hat noch folgende Komponenten integriert:

1. SD-Kartenleser
2. PSRAM (8 MByte, davon 4 MByte effektiv nutzbar)
3. 4 MByte Flash
4. OV2640 Kameraschnittstelle und -modul
5. LED-Flashlight

Insbesondere die Nutzung von SD-Karten und der erweiterte Speicher sind für neuronale Netze sehr von Vorteil. Grundsätzlich lassen sich die neuronalen Netze auch vollständig in der Firmware abbilden. Wenn man jedoch die Größe von neuronalen Netzen mit mehr als 100 000 Knoten anschaut, wird schnell klar, dass die 512kByte SRAM bald an ihre Grenzen kommen. Hier hilft das PSRAM weiter, da es auf einfache Weise den Arbeitsspeicher um 4 MByte erweitert. Hier sei schonmal vermerkt, dass die Unterstützung von PSRAM in der Konfiguration des Compilers aktiviert werden muss.

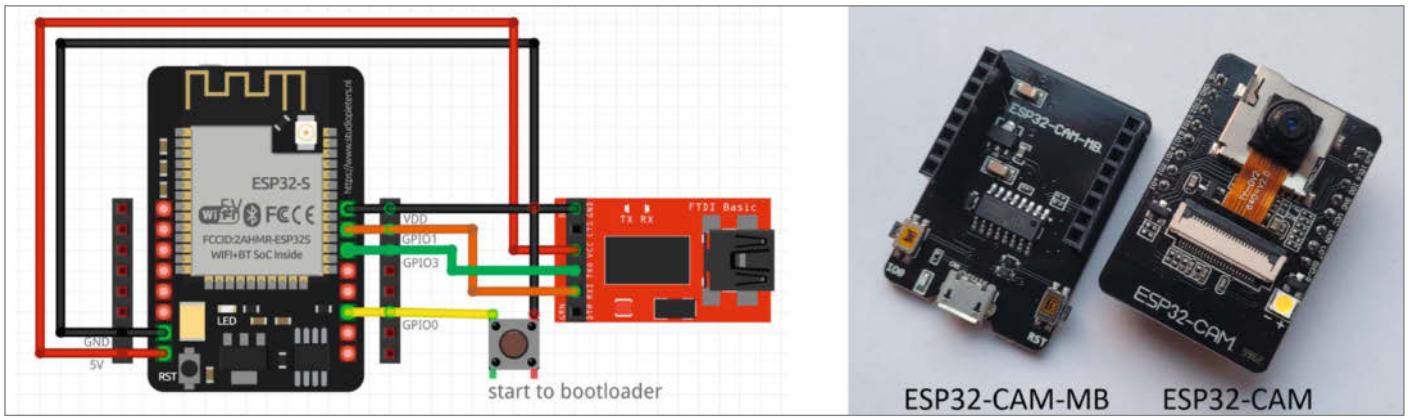
Die SD-Karte dient vor allem als Speichermedium für Beispielbilder und auch zum dynamischen Laden von neuronalen Netzen während des Programmablaufs. Beides wird im Quellcode gezeigt.

Auf das eigentliche Kameramodul, wie man damit zur Laufzeit Bilder aufnimmt und Ziffern erkennt, gehen wir in diesem Workshop ganz am Ende ein. Die Erklärungen zum Steuern der Kamera und dem Aufnehmen und Normalisieren eines Bildes unter C++ würde den Rahmen dieses Artikels sprengen.

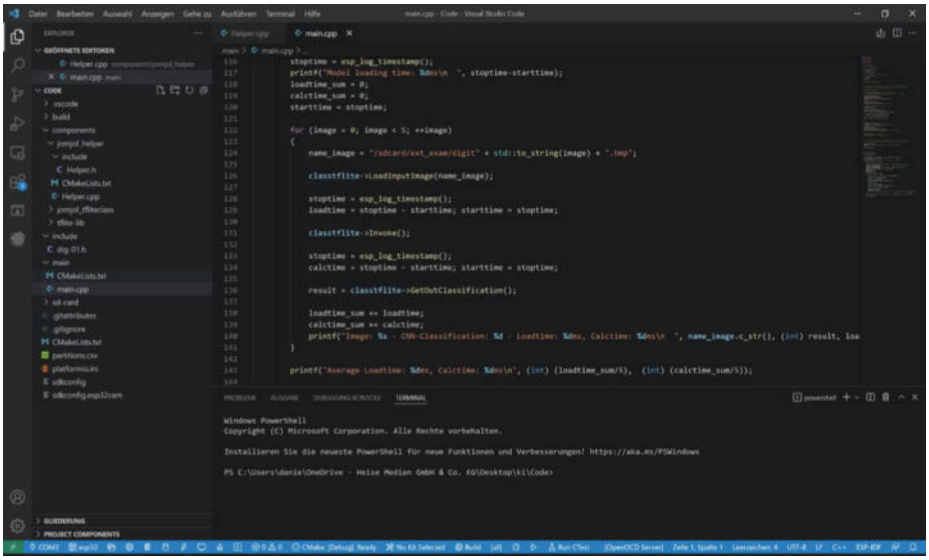
Das ESP32CAM-Modul inklusive der Kamera bekommt man schon für ca. 10 EUR bei den einschlägigen Versendern. Bei günstigen Angeboten findet man in Foren immer wieder mal den Hinweis, dass doch nur 2 MByte PSRAM verbaut sind oder die Kamera Schwierigkeiten macht. Der PSRAM wird daher auch zu Beginn des Programms abgeprüft und ggf. eine Warnung ausgegeben.

Programmier-Interface

Um die Firmware auf den ESP32 zu flashen, ist eine Programmierschnittstelle notwendig. Diese verbindet die UART-Schnittstelle des ESP32 mit der USB-Schnittstelle des Rechners. Dabei gibt es im Wesentlichen zwei Möglich-



Die ESP32CAM lässt sich mit einem USB-zu-seriell-Konverter programmieren. Einfacher macht es das Development Board für die ESP32CAM, das eine Spannungsversorgung und einen Button für Pin IO0 mitbringt.



MIt Visual Studio Code behält man in komplexeren Projekten eine bessere Übersicht als mit der Arduino IDE.

```
ctmake-KI-ESP32-Teil3
├── sd-card
│   └── ...
├── components
│   ├── tfmicro
│   ├── jomjol_helper
│   └── jomjol_tflite_wrapper
├── include
│   └── dig-01.h
├── main
│   ├── main.cpp
│   └── CMakeLists.txt
├── sdkconfig
├── CMakeLists.txt
└── partitions.csv
```

Überblick über die Programmstruktur des Projektes

keiten: Zum ersten gibt es häufig das ESP32-CAM Modul bereits in Kombination mit einem Adapter, der eine Mini-USB-Schnittstelle und darüber auch die Stromversorgung enthält. Man findet es häufig unter der Bezeichnung *ESP32-CAM-MB*. Zum zweiten kann man auch einen FTDI-Adapter mit den entsprechenden GPIOs verbinden. Dann muss man natürlich auch die Spannungsversorgung sicherstellen. Die Kommunikation ist an GPIO1 und GPIO3 angeschlossen.

Um den ESP32 im Downloadmodus zu starten, muss der GPIO0 während des Starts auf Masse-Potential (GND) gelegt werden – im Bild angedeutet durch einen Taster. Beim ESP32-CAM-MB ist dazu ein extra Schalter vorhanden. Wenn der ESP32 autark läuft oder per WLAN/Bluetooth-Schnittstelle angesteuert wird, ist nur noch eine Stromversorgung notwendig. Es wird eine 5V Versorgung empfohlen, da 3,3V teilweise nicht stabil laufen. Damit ist die Hardware für die Programmierung vorbereitet.

SD-Karte

Der SD-Kartenleser nimmt normale Micro-SD-Karten auf. Eigentlich keine große Sache. Leider hat sich jedoch im *AI-on-the-Edge*-Projekt gezeigt, dass nicht alle SD-Karten kompatibel sind. Insbesondere große SD-Karten mit 32/64 GByte in Kombination mit der internen 1-Kanal-Datenverbindung zwischen ESP32 und Kartenleser sind problematisch. Daher wird hier standardmäßig die 4-Kanal-Verbindung verwendet, die dann aber auch den GPIO12 und 13 belegt. Auch der GPIO4 wird verwendet, was eine Doppelbelegung mit dem LED-Flashlight bedeutet. Daher leuchtet im Beispielprogramm das Flashlight bei Lesezugriffen auch auf.

Die SD-Karte muss FAT32 formatiert sein und darf nur eine einzige Partition enthalten. Wenn die Karte sich nicht initialisieren lässt, dann gibt es eine entsprechende Fehlermeldung auf der Konsole. Leider gibt es, wenn auch selten, das Problem, dass die Karte sich initialisieren lässt, dann aber die Dateien nicht lesbar sind. In diesem Fall hilft nur eine andere SD-Karte, und zwar möglichst klein – 4GB-Karten machen typischerweise keine Probleme.

Programmierumgebung

Die Programmierung des ESP32 erfolgt in C/C++, da darin auch die TensorFlow-Lite-Bibliotheken für Mikrocontroller programmiert sind. Als Programmierumgebung insbesondere auch für einfache bis mittel komplexe Projekte wird häufig die Arduino-IDE verwendet. In diesem Projekt wird jedoch eine deutlich leistungsfähigere und umfangreichere Umgebung benötigt.

Als Editor kommt der kostenlose *Visual Studio Code* (VSCode) von Microsoft zum Einsatz, der sich mit zahlreichen Plugins, Compilern und Interpretern erweitern lässt. Mit dem ebenfalls kostenlosen ESP-IDF-Plug-in lassen sich alle Produkte des Herstellers Espressif in VSCode programmieren und flashen. Allerdings stehen hier die einsteigerfreundlichen Bibliotheken und Makros wie in der Arduino-Welt nicht mehr zu Verfügung, die kompliziertere Code-Teile weitestgehend abstrahieren. Die Anleitung zur Installation, Konfiguration und Bedienung von Visual Studio Code haben wir in einen Online-Artikel ausgelagert (siehe Link in der Kurzinfo), zu dem es zusätzlich ein Video-Tutorial gibt.

Das ESP-IDF-Plugin installiert und konfiguriert die Toolchain für das Übersetzen des C-Codes, das Linken und das Flashen. Eine detaillierte Installationsanleitung und ein Video mit einem ersten einfachen Demobeispielprojekt findet sich in unserem begleitenden Online-Artikel (siehe Link). Die hier verwendete Toolchain von Espressif wird in der Version 4.3 verwendet.

Programmstruktur

Wenn nach der Installation das Beispielprogramm läuft, steht dem eigentlichen Projekt nichts mehr im Weg. Zunächst benötigen wir Code aus dem GitHub-Repository (siehe Link). Die wesentliche Programmstruktur ist in der Grafik gezeigt. Neben dem Hauptverzeichnis gibt es vier Unterverzeichnisse:

– *sd-card*
– *components*
– *include*
– *main*

Das Verzeichnis *sd-card* wird nicht zum Kompilieren benötigt, sondern enthält die Dateien, die während der Laufzeit auf der SD-Karte benötigt werden. Der Inhalt des Verzeichnisses muss auf die vorbereitete SD-Karte kopieren werden. Im Verzeichnis *main* findet sich das eigentliche Hauptprogramm. Dazu später mehr, ebenso zum Verzeichnis *include*, welches eine direkt im Code einbindbare Definition des neuronalen Netzes enthält.

Wichtig ist das Verzeichnis *component*. Dort wird der Code für alle verwendeten Komponenten abgelegt. Konkret sind das in diesem Projekt drei Bibliotheken:

jomjol_helper: Diese dient im Wesentlichen dazu, das Hauptprogramm so übersichtlich wie möglich zu halten und typische Standardaufgaben in Funktionen auszulagern, welche dann im Hauptprogramm einfach aufgerufen werden können. Darin finden sich zum Beispiel Hilfsprozeduren zum Initialisieren der SD-Karte oder zum Prüfen des PSRAMs.

jomjol_tfliteclass: Hierin ist eine eigene Klasse in C++ implementiert, die die Ansteuerung der TensorFlow-Lite-Komponenten kapselt und alle sich wiederholenden Abläufe zum Laden, Initialisieren und Verwenden von neuronalen Netzen enthält. In dieser Klasse sind auch einige Hilfsfunktionen implementiert, um die Bilder zu laden oder Informationen über die Ein- und Ausgangslayer des Netzes zu bekommen. Mit dieser Art Wrapper-Klasse lässt sich die komplexe dritte Bibliothek, die eigentliche Bibliothek für die neuronalen Netze, in einer sehr einfachen und schlanken Art und Weise nutzen. Mehr dazu gleich.

tflite-lib: Dies ist der eigentliche Code von TensorFlow. Dieser Code kann aus dem offiziellen TensorFlow-Lite-Code abgeleitet und für die ESP-IDF-Umgebung angepasst werden (Link in der Kurzinfo). Glücklicherweise hat Espressif in seinen eigenen Beispielprogrammen für den ESP32 das bereits für uns erledigt. Dort findet man die Bibliothek im Verzeichnis *component*. Dieses Verzeichnis wird von Espressif regelmäßig aktualisiert. In unserem Projekt-Verzeichnis findet sich die Kopie des ESP-Github-Repos vom Stand Anfang 2022.

Neben diesen Verzeichnissen sind noch drei Dateien von wesentlicher Bedeutung: *CMakeLists.txt*, *sdkconfig* und *partitions.csv*:

1. CMakeListes.txt

Erstere findet sich in jedem Hauptverzeichnis des Quellcodes und steuert die Details für den Compiler für die jeweilige Komponente. Das sollte erstmal so übernommen werden. Wer schon mal unter Linux C/C++-Code übersetzt hat, ist womöglich mit dem Tool `make` in Berührung gekommen. Es steuert anhand der zum jeweiligen Projekt gehörenden Datei *Makefile* alle Befehle, Parameter und Schritte beim Übersetzen und Linken. Auch die Arduino-IDE verwendet im Hintergrund das make-System, um aus Sketchen Binärdateien zu bauen.

`CMake` geht einen Schritt weiter und macht Projekte unabhängig vom Betriebssystem und der Entwicklungsumgebung. Es erstellt aus den Schritten in der Datei *CMakeLists.txt* die für das verwendete Framework passenden Anleitungen, beispielsweise Makefiles. Hier zeigen wir als Beispiel die Datei *CMakeLists.txt* im Verzeichnis *main*, die dem Compiler notwendige Schritte vorgibt.

```
idf_component_register(SRCS \${CMAKE_
SOURCE_DIR}/main/main.cpp`
                    `INCLUDE_DIRS
\${CMAKE_SOURCE_DIR}/include`
                    `REQUIRES jomjol_
tfliteclass jomjol_helper)
```

Dem Compiler müssen verschiedene Dinge vorgegeben werden. Dies geschieht durch den Aufruf von `idf_component_register` mit folgenden Informationen respektive Schlüsselworten: der Pfad zum Quelltext des Haupt-

```
SD-Card using 4-line connection mode. GPIO12/13 is used by SD-Card.
I (1432) gpio: GPIO[13]| InputEn: 0| OutputEn: 1| OpenDrain: 0| Pullup: 0| Pulldown: 0| Intr:0
Name: SU08G
Type: SDHC/SDXC
Speed: 20 MHz
Size: 7580MB
PSRAM initialisiert - freier verfügbarer Speicher: 4192151

Lade tflite-Model

Größe Eingangs-Layer:
Anzahl Eingangsdimensionen: 3
  Größe Dimension 1: 32
  Größe Dimension 2: 20
  Größe Dimension 3: 3


Größe Ausgangs-Layer:
Anzahl Ausgangsdimensionen: 1
  Größe Dimension 1: 11

Lade Bilddaten...
Berechne neuronales Netz ...

Frage Ergebnis ab ...
Filename: /sdcard/digit3a.jpg
 Einzelne Output-Neuronen:
   Neuron #0: 0.0
   Neuron #1: 0.0
   Neuron #2: 0.0
   Neuron #3: 1.0
   Neuron #4: 0.0
   Neuron #5: 0.0
   Neuron #6: 0.0
   Neuron #7: 0.0
   Neuron #8: 0.0
   Neuron #9: 0.0
   Neuron #10: 0.0
  CNN-Klassifizierung: 3


Lade Bilddaten...
Berechne neuronales Netz ...

Frage Ergebnis ab ...
/sdcard/digit1.jpg: CNN-Klassifizierung: 1

Herzlichen Glückwunsch!
```

Über die serielle Schnittstelle gibt die ESP32CAM ihren Status und ihre Ergebnisse aus.

```
Drücke beliebige Taste zum Fortfahren ... (erweitertes Beispiel)

====== Model small.tfl ===================
  Model loading time: 300ms
  Image: /sdcard/digit0.bmp - CNN-Classification: 2 - Loadtime: 10ms, Calctime: 380ms
  Image: /sdcard/digit1.bmp - CNN-Classification: 7 - Loadtime: 10ms, Calctime: 380ms
  Image: /sdcard/digit2.bmp - CNN-Classification: 10 - Loadtime: 10ms, Calctime: 380ms
  Image: /sdcard/digit3.bmp - CNN-Classification: 8 - Loadtime: 10ms, Calctime: 380ms
  Image: /sdcard/digit4.bmp - CNN-Classification: 5 - Loadtime: 10ms, Calctime: 380ms
  Average Loadtime: 10ms, Calctime: 380ms

====== Model middle.tfl ===================
  Model loading time: 1040ms
  Image: /sdcard/digit0.bmp - CNN-Classification: 2 - Loadtime: 0ms, Calctime: 1440ms
  Image: /sdcard/digit1.bmp - CNN-Classification: 7 - Loadtime: 10ms, Calctime: 1440ms
  Image: /sdcard/digit2.bmp - CNN-Classification: 10 - Loadtime: 10ms, Calctime: 1440ms
  Image: /sdcard/digit3.bmp - CNN-Classification: 8 - Loadtime: 10ms, Calctime: 1440ms
  Image: /sdcard/digit4.bmp - CNN-Classification: 5 - Loadtime: 10ms, Calctime: 1440ms
  Average Loadtime: 8ms, Calctime: 1440ms

====== Model large.tfl ===================
  Model loading time: 1760ms
  Image: /sdcard/digit0.bmp - CNN-Classification: 2 - Loadtime: 10ms, Calctime: 1480ms
  Image: /sdcard/digit1.bmp - CNN-Classification: 7 - Loadtime: 10ms, Calctime: 1480ms
  Image: /sdcard/digit2.bmp - CNN-Classification: 10 - Loadtime: 10ms, Calctime: 1480ms
  Image: /sdcard/digit3.bmp - CNN-Classification: 8 - Loadtime: 10ms, Calctime: 1480ms
  Image: /sdcard/digit4.bmp - CNN-Classification: 5 - Loadtime: 10ms, Calctime: 1490ms
  Average Loadtime: 10ms, Calctime: 1482ms
```

Verschiedene neuronale Netze bei ihrer Arbeit

Methoden der Wrapper-Klasse

Name	Funktion
LoadModelFromFile	Netz von SD-Karte laden
LoadModelFromCharArray	Netz aus Header laden
LoadInputImage	Bild von SD-Karte laden
Invoke	Berechnen der Ausgabe (Inferencing)
GetOutClassification	Ausgabe-Neuronen ausgeben
GetOutputValue	Neuron mit dem höchsten Wert ausgeben
GetInputDimension	Größe des Input-Layer ausgeben
GetOutputDimension	Größe der Ausgabeschicht ausgeben

programms *main.cpp* – definiert über `SRCS`, das Verzeichnis der Header-Dateien (`INCLUDE_DIRS`) und weitere wichtige Bibliotheken (`REQUIRES`). Letzteres verweist auf weitere benötigte Komponenten in dem Verzeichnis *component*.

Hier sind es zum einen die Hilfskomponenten `jomjol_helper` und `jomjol_tfliteclass`, die die Ansteuerung des neuronalen Netzes übernehmen. Die Bibliothek `tflite-lib` muss hier nicht angegeben werden, da sie nicht direkt aufgerufen wird, sondern indirekt über die `jomjol_tfliteclass` eingebunden wird, die über ein eigenes CMakeFile übersetzt wird (siehe *CMakeList.txt* im jeweiligen Unterverzeichnis). Kompilierinformationen können auch deutlich komplexer ausfallen – etwa das Makefile für die Komponente *tflite-lib* im entsprechenden Unterverzeichnis.

2. sdkconfig

Wichtig ist auch die Datei *sdkconfig* im Hauptverzeichnis. Dort werden alle Compiler-spezifischen Einstellungen für den ESP32 gespeichert. Zur Konfiguration klickt man in VSCode das Rädchen im linken Bereich des unteren Menübands (siehe Online-Tutorial). Es öffnet sich ein neuer Tab (was beim ersten Mal einige Zeit dauern kann) mit dem Namen *SDK Configuration Editor*. Hier sind zwei Einstellungen wichtig: Aktivierung von PSRAM und die Verwendung einer eigenen Partitionstabelle (siehe nächster Abschnitt). Das ist in unserem Projekt bereits korrekt vorkonfiguriert. Wenn man jedoch die ESP-IDF-Version ändert, aktualisiert oder das Projekt auf eine andere Entwicklungsumgebung wie *PlatformIO* umzieht, müssen diese Einstellungen überprüft und gegebenenfalls wieder angepasst werden.

3. Partitions.csv

In der Datei *partitions.csv* wird eine benutzerdefinierte Partitionierung des Flashspeichers definiert, um die 4 MByte vollständig auszunutzen. Dies wird im Abschnitt `Serial flasher config` im Editor der *sdkconfig* angegeben. Üblicherweise haben ESP32-Module mit 4MB Speicher eine Aufteilung in fünf Partitionen, wozu auch Bereiche für das Update der Firmware per WLAN (OTA) vorgesehen sind. Das verbraucht viel Speicher, den wir in unserem Projekt benötigen, da wir das neuronale Netz direkt in den Programmcode implementieren. Die richtige Partitionierung ist in der Datei schon voreingestellt.

```
//////////////////////////////////////////////////////////////////
////////// 0) Konfiguration und Einstellungen ////////////////////
//////////////////////////////////////////////////////////////////

    // WLan Einstellungen
    const char* WLAN_ssid     = "SSID";
    const char* WLAN_password = "PASSWORD";

    // Bildeinstellungen Bildgröße = QVGA (320x240x3 Farben)
    int Camera_Image_Size_x = 320;
    int Camera_Image_Size_y = 240;
    int Camera_Channels = 3;

    // ROI Einstellungen
    // Das ROI definiert den Ausschnitt aus dem Kamerabild, der zunächst ausgeschnitten
    // und anschließend auf die Inputgröße des neuronalen Netzes reskaliert wird.
    bool ROI_rotate = false;            // true: Drehung des ROIs um 90°
    int ROI_x = 20;
    int ROI_y = 20;
    int ROI_dx = 136;
    int ROI_dy = 205;

    // Dateinamen zum Speichern der Bilder auf der SD-Karte
    std::string image_name_input  = "/sdcard/original.jpg";
    std::string image_name_ROI    = "/sdcard/roi.jpg";
    std::string image_name_resize = "/sdcard/resize.bmp";

    // Einstellungen für das neuronale Netz
    std::string cnn_tflite = "/sdcard/dig-01.tfl";
    int cnn_input_x = 20;
    int cnn_input_y = 32;
```

Für das Beispiel der Live-Erkennung gibt es einen Konfigurationsbereich in main.cpp.

ESP32-Programmierung

Nun sind alle Vorbereitungen getroffen und es geht an die eigentliche Programmierung. Im Prinzip kann man mit der Hilfsbibliothek *jomjol_tflitewrapper* in fünf Schritten einen Programmteil zur Nutzung von neuronalen Netzen erstellen:

1. Erzeugen eines Objekts der Wrapper-Klasse *CTfLiteClass*
2. Laden der Models
3. Laden eines Bildes
4. Berechnen des neuronalen Netzes
5. Auslesen der Ergebnisse

Diese Schritte sind in der Klasse als eigene Methoden implementiert; ihre Namen sind im Prinzip selbsterklärend (siehe Tabelle *Methoden der Wrapper-Klasse*). Eine detaillierte Erklärung finden Sie in *CTfLiteClass.h*.

Ihre Arbeitsweise wollen wir in den nächsten Abschnitten der Reihe nach durchgehen und um zusätzliche Informationen ergänzen. Diese Schritte sind im Hauptprogramm in der `main`-Funktion abgebildet.

Vorbereitung

Damit alle Funktionen zur Verfügung stehen, müssen zunächst zwei Bibliotheken geladen werden:

```
#include "CTfLiteClass.h"
#include "Helper.h"
```

Als erster Schritt werden mit Hilfe von Funktionen aus der Helper-Bibliothek die SD-Karte eingebunden und die Verfügbarkeit des PSRAMs geprüft. Dahinter findet sich eine rudimentäre Fehlerbehandlung. Details dazu finden sich direkt in den Funktionen im Code (*main.cpp*) beziehungsweise in den Ausgaben auf der Konsole.

Im weiteren müssen wir eine Instanz der Klasse `CTfLiteClass` erzeugen, um ein neuronales Netz aufzubauen:

```
neuralnetwork = new CTfLiteClass;
```

Dabei werden im Hintergrund die Variablen initialisiert und ein Speicherbereich reserviert, in dem die *TfLite*-Bibliothek anschließend ihren Daten und Strukturen speichert. Dessen Größe hängen von der Größe des neuronalen Netzes ab. Hier sind zunächst 600kByte voreingestellt. Das ist eher großzügig und ausreichend auch für größere Netzwerke.

Laden des Modells

Als nächstes wird die *tflite*-Beschreibung des neuronalen Netzwerkes geladen. Dort kommt jetzt das Trainingsergebnis des zweiten Teils zum Einsatz. Man hat zwei Möglichkeiten, die

Modellbeschreibung des neuronalen Netzes einzubinden. Entweder man lädt die *tflite*-Datei dynamisch während des Programmablaufs von der SD-Karte:

```
if (!neuralnetwork->LoadModelFromFile
("/sdcard/dig-01.tfl"); // dynamisch
    return;
```

oder man kann es direkt als `char`-Array aus der automatisch erzeugten Header-Datei statisch einbinden:

```
if (!neuralnetwork->LoadModelCharArray
(tflite_model)); // statisch
    return;
```

Beim statischen Einbinden muss man die im zweiten Teil unserer Artikelstrecke erzeugte Headerdatei während bzw. vor dem Übersetzungsvorgang einbinden. Dies geschieht hier durch die Zeile `#include "dig-01.h"`. Darin ist das Modell in einem `char`-Array namens `tflite_model` definiert, das dann direkt eingebunden wird. Das Laden direkt aus einem Array hat den Vorteil, dass man kein Dateisystem benötigt und damit entsprechend einfachere Hardware einsetzen kann.

Der Vorteil des dynamischen Ladens von der SD-Karte liegt zum einen in der Speicherverwaltung, denn er wird erst dynamisch während des Programmablaufes im PSRAM allokiert und kann für andere Aufgaben wieder freigegeben werden. Zum zweiten kann man verschiedene neuronale Netze zur Laufzeit nach Bedarf laden. Man benötigt dafür aber ein Speichermedium, wie hier die SD-Karte. Es lassen sich Daten aber auch über das Netzwerk oder von anderen Speichermedien nachladen.

Sind die *tflite*-Daten geladen, müssen die Strukturen und Speicheranforderungen im vorher reservierten Speicherbereich angelegt und initialisiert werden. Dies übernimmt die interne Funktion `MakeAllocate()` innerhalb der Ladefunktion. Im Code findet man noch zwei weitere Befehle (`GetInputDimension()` und `GetOutputDimension()`), die die Struktur des geladenen neuronalen Netzes auslesen und wiedergeben: am Eingang sind es hier drei (20 × 32 × 3) Dimensionen und am Ausgang nur eine Dimension (mit 11 Neuronen).

Laden des Bildes

Jetzt ist das neuronale Netz bereit, um Bilder zu bewerten. Dazu wird zunächst das Bild geladen – hier von der SD-Karte. In realen Anwendungen könnte hier auch ein Ausschnitt des Kamerabildes des integrierten OV2640-Kameramoduls geladen werden.

```
image_file = "/sdcard/digit3a.jpg";
if (!neuralnetwork->
LoadInputImage(image_file.c_str()));
   return;
```

Hier können sowohl jpg- wie auch bmp-Bilder verwendet werden. Das Bild wird über eine Hilfsbibliothek geladen, die die Bildpunkte Pixel für Pixel an die Eingangsneuronen des ersten Layers anlegt. Dies erfordert etwas C-Zeigerarithmetik und ist innerhalb der *jomjol_tfliteclass*-Bibliothek implementiert.

Im nächsten Schritt wird das neuronale Netz berechnet. Das übernimmt eine einzige Funktion:

```
neuralnetwork->Invoke();
```

Dies ist der rechenintensivste Prozess und kann einen Moment dauern.

Auslesen der Ergebnisse

Als letzten Schritt gibt das Programm die Ergebnisse über die serielle Schnittstelle aus. Um sich diese anzeigen zu lassen, muss man in der unteren Leiste den Punkt *ESP IDF Monitor Device* anklicken. Die Ausgabe des Ergebnisses erfolgt in `main()` auf zwei Arten:

```
for (int _output = 0; _output <
AnzahlOutputNeurons; _output++)
{
    result = neuralnetwork->
 GetOutputValue(_output);
    printf("   Neuron #%d: %.1f\n",
_output, result);
}
```

In der Schleife wird über die Funktion `GetOutputValue(_output)` der Wert jedes einzelnen Neurons ausgelesen und ausgegeben. Die gesuchte Klassifizierung ergibt sich aus dem Neuron mit dem höchsten Aktivitätswert. Will man nur das Neuron mit dem höchsten Wert, genügt die Funktion `GetOutClassification()`, die wir für ein weiteres Bild nutzen:

```
result = neuralnetwork->GetOutClassifica
tion();
```

Damit ist die Basis für die Anwendung zur Bildverarbeitung gelegt. Um diese wenigen Zeilen Code herum kann man jetzt die weiteren Strukturen zur Bereitstellung der Bilder und dem Darstellen der Ergebnisse entwickeln. Zum Beispiel könnte man ein Bild direkt mit der vorhandenen Kamera aufnehmen und Abschnitte daraus an das neuronale Netz übergeben; über Standardbibliotheken auf die Eingangsgröße des neuronalen Netzes skalieren und dann klassifizieren lassen. Das Ergebnis könnte man in einer Webapplikation darstellen oder auch auf die SD-Karte speichern.

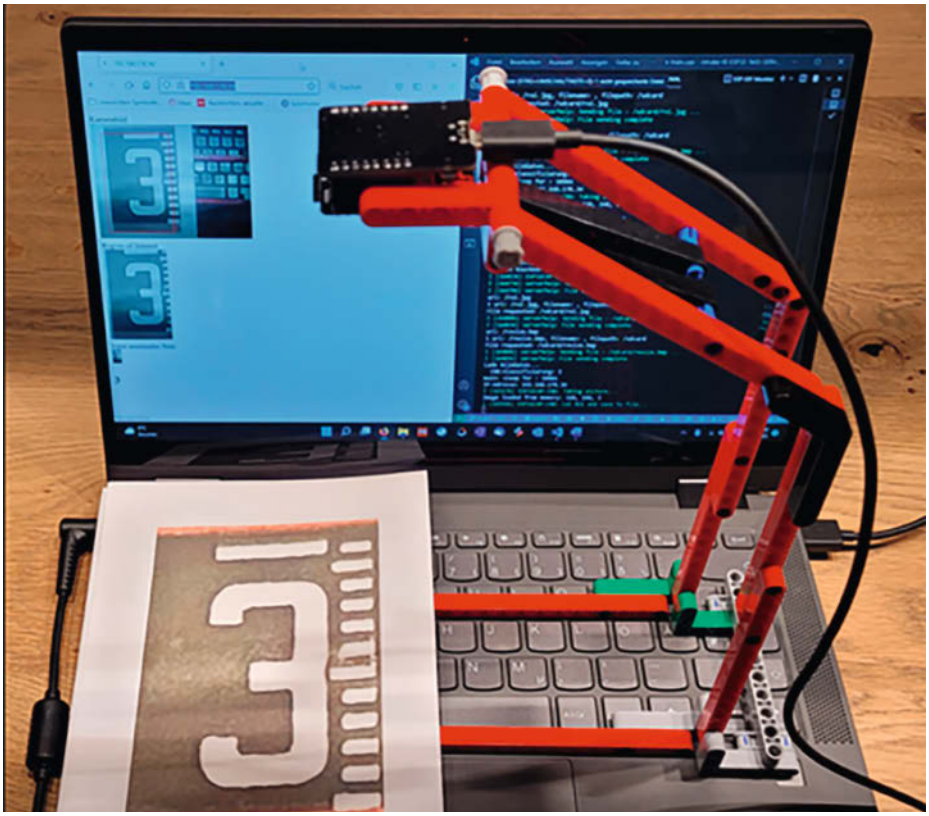
Wie in unserem Online-Tutorial zur Installation von VSCode und der ESP-IDF müssen Sie den gesamten Code durch Anklicken von *ESP IDF Build Project* übersetzen. Das kann einige Zeit in Anspruch nehmen. Anschließend lädt man das Binary auf die Kamera, steckt die SD-Karte mit den *TFlite*-Modellen und den

Die vom Webserver ausgelieferte Seite zeigt das Bild, die Region of Interest und die erkannte Ziffer an.

Bilddateien und resettet. Mit einem Klick auf *ESP IDF Monitor Device* öffnet sich ein serieller Monitor, mit dem sich die Ausgabe des ESP verfolgen lässt.

Dieses Beispiel zeigt, wie man neuronale Netze, die man in einer beliebigen Umgebung trainiert hat, sehr einfach auf dem ESP32 anwenden kann. Das Laden der neuronalen Netze direkt aus dem *tflite*-Format ermöglicht auch eine flexible Anpassung und Tests verschiedener Netzkonfigurationen. So könnte man in einer zweiten Schleife auch unterschiedlich Netzgrößen und Strukturen verwenden und die Ergebnisse oder Performance auf dem ESP32 vergleichen. Aufgrund des erweiterten Speichers ist es möglich, *tflite*-Files, die größer als 1MB sind und nahezu eine Million Parameter enthalten, zu laden und laufen zu lassen. Mit diesen wenigen Zeilen Code hat man schon die Berechnung eines selbst trainierten neuronalen Netzes sowie die Anwendung auf beliebige Bilder ermöglicht.

Im Hauptprogramm wird nach dem ersten Teil durch einen Tastendruck ein zweites, etwas umfangreicheres Beispiel gestartet. In diesem werden der Reihe nach drei neuronale Netze von der SD-Karte geladen, welche eine unterschiedliche Größe haben. In einer Schleife werden dann fünf Bitmaps berechnet respektive erkannt. Sowohl die Zeit zum Laden des neuronalen Netzes wie auch die Berechnungszeit der einzelnen Bilder wird angezeigt. In diesem Beispiel werden zum einen die Vorteile des dynamischen Ladens von neuronalen Netzen gezeigt, wie auch durch die unterschiedlich großen Netze ein Gefühl für die



Der stabile Probeaufbau macht die Ausrichtung robuster und die Erkennung zuverlässiger.

Berechnungsdauer von unterschiedlichen Netzgrößen entwickelt.

Live-Erkennung

Ganz bewusst haben wir aus didaktischen Gründen im ersten Programm auf die Verwendung der Kamera verzichtet, da neben der einfachen Bildaufnahme noch wichtige Zusatzschritte notwendig sind, damit ein reproduzierbares und zufriedenstellendes Ergebnis herauskommt. Dazu gehören unter anderem: Ausrichten des Kamerabildes, Identifikation des Bildausschnitts (in der Regel nicht das gesamte Bild) und Skalierung des Bildes.

Insbesondere das Ausrichten und Identifizieren der sogenannten *Region of Interest* (ROI), welches den Bildausschnitt für das neuronale Netz ausmacht, ist ein ganz zentraler und teilweise sehr aufwendiger Algorithmus. Das wird einem klar, wenn man berücksichtigt, dass die neuronalen Netze typischerweise mit einer Verschiebung von bis zu 20% trainiert wurde, aber bei einer Größe der ROIs von typischerweise 30 - 100 Pixel dies auch nur 6 bis 20 Pixel Toleranz ergibt.

Beim ESP32 kommt noch hinzu, dass er per se erstmal keine graphische Benutzeroberfläche hat, auf der man die Ausrichtung der Bilder kontrollieren und justieren kann. Um dennoch die Kombination der OV2640 Kamera des ESP-32CAM-Moduls und der neuronalen Erkennung in einem Gesamtsystem zu demonstrieren, stellen wir ein Programm-Beispiel bereit, das alle Elemente dazu enthält und auch für das Alignment eine rudimentäre Lösung bietet.

Dazu sind allerdings deutlich mehr und umfangreichere Bibliotheken notwendig als im ersten Beispiel. Eine detaillierte Erklärung im Detail finden Sie im Programmcode. Der Quellcode findet sich ebenfalls auf Github und kann in derselben Umgebung wie auch das erste Beispiel kompiliert werden (siehe Link).

Ausgangspunkt des zweiten Programms ist ein sehr rudimentäres Beispiel aus dem ESP32CAM-Repository von Espressif. Darin wird die Kamera initialisiert und immer wieder ein Bild aufgenommen, jedoch ohne Speicherung. Wir ergänzen dieses Beispiel durch einen WLAN-Client, einen einfachen http-Server, um die aufgenommenen Bilder anzuzeigen sowie Funktionen zur Aufbereitung der aufgenommenen Bilder, damit sie vom neuronalen Netz verarbeitet werden können.

Zunächst werden im ersten Teil des Programms (*main.cpp*) die Variablen initialisiert und damit der Ablauf gesteuert (*0 Konfiguration und Einstellungen*). Hier ist insbesondere die Einstellungen für das WLAN wichtig, damit der ESP32 sich verbinden kann und man im Browser über den http-Server die Kamerabilder kontrollieren kann.

Bildverarbeitung

Der Prozess besteht aus drei Schritten. Die Bildaufnahme: Hier wird das Bild von der Kamera geladen und über eine zusätzliche Bildverarbeitungsbibliothek auf der SD-Karte gespeichert. Die Bildgröße ist auf QVGA festgelegt. In das Bild wird vor dem Speichern noch das ROI gezeichnet, welches im nächsten Schritt weiterverwendet wird. Die Größe und Position des ROIs wird im Konfigurationsteil festgelegt.

Im zweiten Schritt wird mit Hilfe von zusätzlichen Bildverarbeitungsbibliotheken (*CImage*, *stb_image*) aus dem Originalbild zunächst das ROI ausgeschnitten, ggf. um 90 Grad gedreht (konfigurierbar) und auf die Zielgröße für das neuronale Netz (20 x 32 Pixel) skaliert. Beide Bilder (Original ROI und reskaliertes ROI) werden auf der SD-Karte abgespeichert. Im dritten Schritt berechnet das neuronale Netz das Ergebnis und zeigt es auf dem Webserver an.

Das Alignment, also die richtige Orientierung des Bildes, wird nicht durch das Programm gelöst, sondern unter Zuhilfenahme des Anwenders und des http-Servers. Der Server beziehungsweise die ausgelieferte Webseite ist so eingestellt, dass sie sich alle 2 Sekunden selbst aktualisiert und das Bild sowie das Ergebnis darstellt. Anhand des Bildes und mit Hilfe des eingezeichneten ROIs kann der Nutzer die Kamera ausrichten und verschiedene Bilder testen. Beispielbilder sind in einer Word-Datei im Projektordner zum Ausdruck mitgeliefert. Der Webserver wird über seine IP-Adresse im LAN aufgerufen; diese wird über die Monitoringschnittstelle in VSCode zu Beginn jedes neuen Bildverarbeitungszyklusses angezeigt.

Der Server ist über die ESP-IDF eigenen Bibliotheken realisiert und in eine separate Komponente ausgelagert. Er enthält nur einen einzigen Handler, der die Anfragen je nach URL abarbeitet.

Inbetriebnahme

Um das ganze in Betrieb zu nehmen, muss zunächst die Konfiguration angepasst werden. Zunächst reicht es, die SSID und das Passwort des WLANs einzustellen, damit man den Server erreicht. Jetzt kann die Firmware kompiliert und auf das ESP32CAM-Modul geflasht werden. Als nächstes muss noch der Inhalt des Ordners sdcard auf die SD-Karte kopiert werden. Hier enthält er nur ein einziges File nämlich das tflite-File für das neuronale Netz. Jetzt kann der ESP32 in Betrieb genommen werden. Am besten startet man parallel den seriellen Monitor, um zu sehen, ob der ESP ohne Probleme startet. Dort ist auch die IP-Adresse abzulesen.

Damit hoffen wir, dass wir mit den drei Artikeln und begleitenden Online-Tutorials einen ersten Einblick in neuronale Netze und deren Verwendung auf einem günstigen Mikroprozessor gegeben haben. Viel Spaß bei eigenen Projekten und schreiben Sie uns! —*dab*

Digitalisierung leicht gemacht!

Wenn Ihre Bücherregale gerade überquellen, bieten wir Ihnen die ideale Lösung. Das gesammelte Know-how Ihrer Fachmagazine kompakt auf Archiv-Discs und -Sticks gespeichert jederzeit zum Abruf bereit.

SONDERPREISE für Abonnenten

c't Jahrgang 2021

Alle **27 Ausgaben** des c't Magazins 2021 in digitaler Fassung.

Archiv-DVD	**24,90 €**
Archiv-Stick (32 GB)	**34,90 €**

c't Gesamtarchiv 1983 – 2021

Das komplette Archiv des c't Magazins von **1983 bis 2021**. Holen Sie sich 39 Jahre IT-Geschichte mit allen redaktionellen Inhalten bis 2021 nach Hause.

2× Blu-ray	**99,90 €**
Archiv-Stick (64 GB)	**139,90 €**

iX Jahrgang 2021

13 digitale Ausgaben des iX-Magazins 2021 mit allen redaktionellen Beiträgen.

Archiv-DVD	**24,90 €**
Archiv-Stick (32 GB)	**34,90 €**

iX Gesamtarchiv 1988 – 2021

34 Jahrgänge des Magazins für professionelle Informationstechnik, der Pflichtlektüre des professionellen IT-Anwenders.

2× Archiv-DVD	**79,90 €**
Archiv-Stick (64 GB)	**109,90 €**

MIT Technology Review Jahrgang 2021

8 digitale Ausgaben des Magazins MIT Technology Review 2021 mit allen redaktionellen Inhalten.

Archiv-DVD	**24,90 €**

MIT Technology Review Gesamtarchiv 2003 – 2021

19 Jahrgänge der Technology Review zusammengefasst auf zwei DVDs.

2× Archiv-DVD	**59,90 €**

Make Gesamtarchiv 2011-2021

Das komplette Archiv mit **59 Ausgaben** von c't Hardware Hacks über c't Hacks bis zum deutschen Make-Magazin.

Archiv-Stick (32 GB)	**99,90 €**

PORTOFREI shop.heise.de/archive21

Generell portofreie Lieferung für Heise Medien- oder Maker Media Zeitschriften-Abonnenten oder ab einem Einkaufswert von 20 €.
Nur solange der Vorrat reicht. Preisänderungen vorbehalten.

heise Shop

Tipps & Tricks

Lego-Steine haben sich schon immer als Helfer für den Maker bewährt. Durch die gute Maßhaltigkeit eignen sie sich aber auch für Aufgaben, bei denen es auf das sprichwörtliche Mü ankommt.

von Achim Bertram

Beim Bau meiner CNC-Portalfräsmaschine 1 stehe ich immer wieder vor der Herausforderung, mit Heimwerkermitteln Ergebnisse zu erzielen, die ausreichend genau sind, um Sinn und Funktion einer solchen Maschine zu genügen. Eine CNC-Maschine bietet neben der automatisierten Fertigung von Teilen mit komplizierter Geometrie vor allem auch reproduzierbare Ergebnisse. Das kann sie aber nur, wenn die mechanischen Komponenten den Ansprüchen der CNC-Steuerung und dem Steuercode genügen. Dabei sind spielfreie und trotzdem reibungsarm gleitende Führungen und Spindeln unabdingbar. Nun, die gibt es mittlerweile auch für den Heimgebrauch zu moderaten Preisen bei guter Präzision zu kaufen. Um Erfolg zu haben, muss der Maker aber bei der Montage solcher Elemente einige Techniken des Maschinenbaus kennen und anwenden können.

Linearführungen werden fast immer paarweise eingesetzt. Um einen einwandfreien Lauf der durch die Aufbauten miteinander verbundenen Gleiter (Linearwagen) zu gewährleisten, müssen sie mit geringer Lagetoleranz zur theoretischen Achse und zueinander parallel ausgerichtet und so fixiert werden, dass sie nach einer Demontage ohne erneute Justierarbeit wieder montiert werden können. Ebenso wichtig ist eine möglichst geringe Formtoleranz bezüglich Ebenheit der Montagebasis und deren Steifheit gegenüber Verformung bei Belastung, was wie in 2 gezeigt mit einem Haarlineal geprüft werden kann.

Feinmechaniker verwenden zur präzisen Lagepositionierung von Maschinenelementen sogenannte *Parallel-Endmaße*. Das sind Blöcke aus Stahl, Hartmetall oder Keramik, mit sehr hoher Maß- und Formgenauigkeit, in unterschiedlichen Längen und Abstufungen hergestellt, die durch *Ansprengen* ihrer Oberflächen (Haftung durch *Adhäsionskraft*) zu fast beliebigen Maßverkörperungen zusammengesetzt werden können 3.

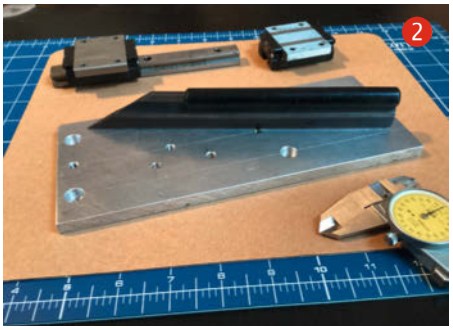
Allerdings werden die allerwenigsten Hobbymechaniker einen Satz Endmaße (min. 47 verschiedene Stücke) ihr Eigen nennen, aber vielleicht gibt es ja Zugang zu einer feinmechanischen Werkstatt, in der man sich helfen lassen kann. Denn der Umgang mit den teuren Stücken erfordert entsprechende Ausbildung und Sorgfalt.

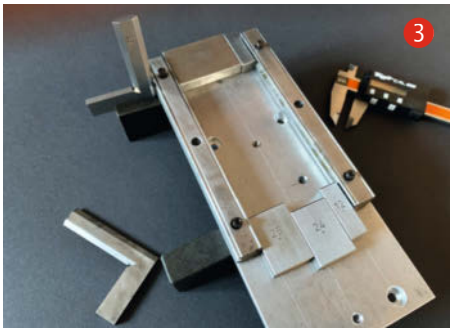
Nun hat fast jeder Maker Zugriff auf Lego-Steine, ob eigene oder die der Kinder oder Enkelkinder. Man mag es kaum für möglich halten, aber diese bunten Klemmbausteine werden mit sehr kleinen Toleranzen gefertigt. Da sich beim Zusammensetzen die Maßabweichungen der einzelnen Steine ja aufsummieren, ist diese hohe Präzision geeignet, um die Bricks als Endmaßersatz im DIY-Bereich zu nutzen. Man erreicht natürlich nicht die hohe Auflösung der Maßabstufungen von Endmaßen, aber darauf kommt es auch nicht an. Es geht nur um die Gleichheit zweier Einheiten, um damit die Parallelität von Bauteilen zu erreichen. Wenn man bereits bei der Konstruktion von Abständen das Lego-Raster berücksichtigt, hat man eine ernstzunehmende

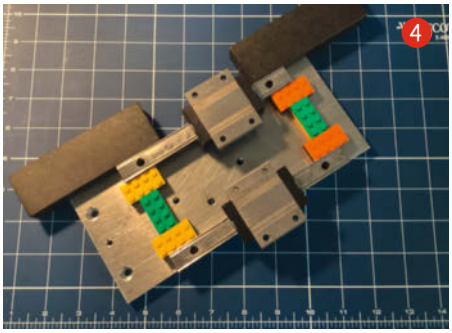
Alternative für präzises Arbeiten mit einfachen Mitteln gefunden 4.

Das Lego-Größensystem

Die Größe von LEGO-Modellen oder Steinen wird nicht in Millimetern angegeben, sondern in der kuriosen Maßeinheit der *Noppen* 5. Das Maß einer Noppe entspricht der Breite des kleinsten Bausteins mit dem Nennmaß 8mm (P=8mm). Die Bezeichnung 2 × 2 bezeichnet also einen quadratischen Stein mit 4 Noppen und der Kantenlänge 16mm. Um die Steine







störungsfrei nebeneinander stecken zu können, werden sie unabhängig vom Noppenmaß mit 0,2mm Untermaß gefertigt. Die Höhe *H* eines Noppensteins ist als 1,2 × P definiert und beträgt also 9,6mm. Mit 3,2mm ist die Höhe *h* einer Platte ein Drittel so hoch. Man kann also mit diesem System Längen im groben 9,6mm-Raster oder im feineren 3,2mm-Raster mit hoher (+/-0,02mm) Genauigkeit nachbilden. Natürlich dürfen Kanten und Oberflächen der Steine nicht beschädigt sein. Als letztes Element einer zusammengesetzten Stange sollte man eine Fliese (glatte, noppenlose Platte) verwenden, denn die Noppenhöhe (1,7mm) passt nicht zum 3,2mm-Raster. Oder man setzt einzelne Bricks der Länge nach zusammen, dann erhält man Maßabstufungen im handlichen 8mm-Raster.

Nach dem Zusammensetzen und Anpressen ist ein Längenvergleich der Stangen zu empfehlen 6. Zum Beispiel durch Messen mit einem Uhr- oder Digital-Messschieber oder per Lichtspalt-Prüfung durch Auflegen eines Haarlineals. Eine mögliche Maßdifferenz von wenigen hundertstel Millimetern kann jetzt noch mit den Blättern einer Fühlerlehre ausgeglichen werden.

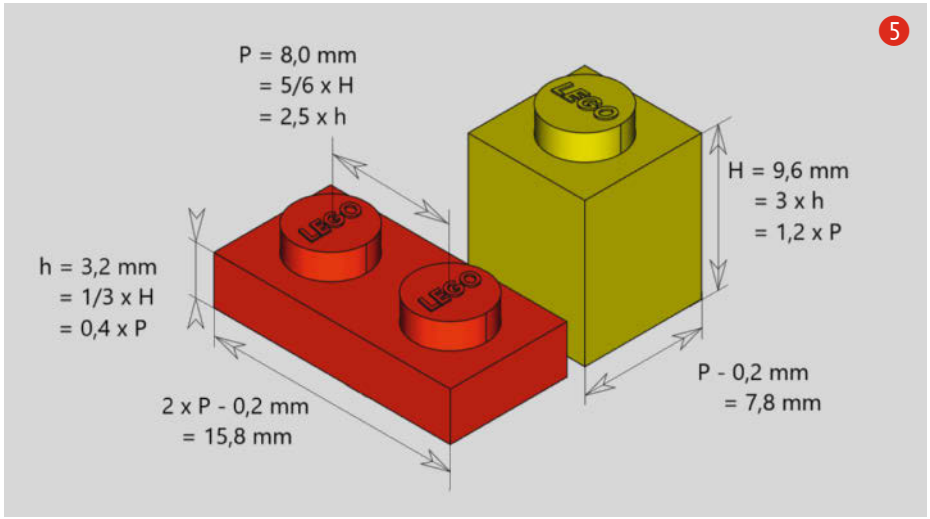
Verschrauben und Verstiften

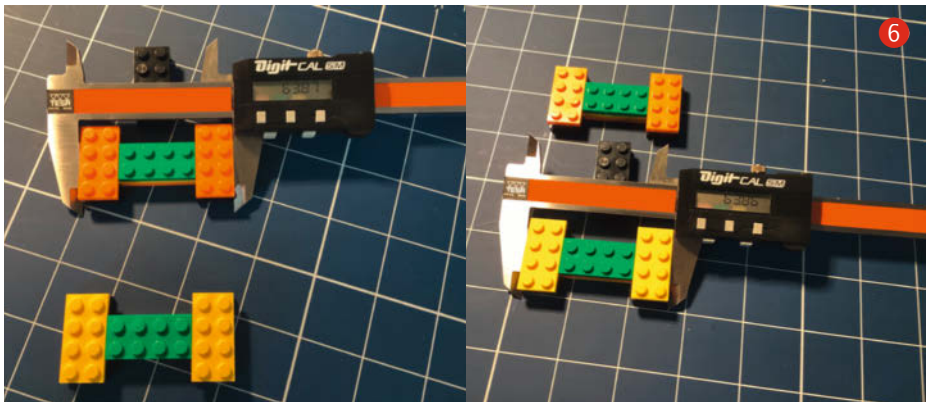
Die Linearschienen werden mit den zur Verfügung stehenden Mitteln, also Lego- oder Endmaße, auf der Trägerplatte nach den rechtwinklig zueinander stehenden Bezugskanten und parallel zueinander ausgerichtet. Dann mittels vorgefertigter Bohrungen miteinander verschrauben. Bohrungen für Verbindungsschrauben sollten in diesem Fall nach Toleranzklasse *c* (grob), mindestens aber nach *m* (mittel) gebohrt werden, damit die Teile genügend Spielraum zum Ausrichten haben.

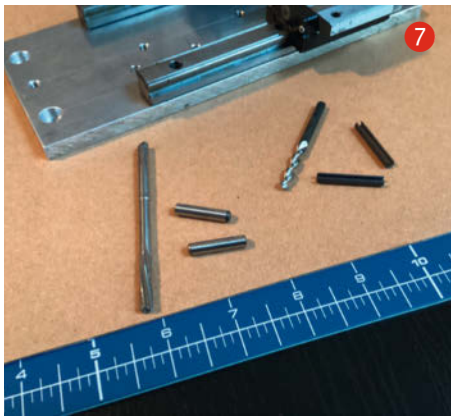
Danach können Schienen und Trägerplatte unverrückbar miteinander verstiftet 7 werden. Durch Stifte werden Werkstücke formschlüssig in radialer Richtung der Stifte verbunden. Sie sind mit Nägeln vergleichbar, haben jedoch weder Kopf noch Spitze. Verschiedene Ausführungen (Zylinder-, Kegel-, Spann- und Kerbstifte) dienen verschiedenen Zwecken. Hier verwenden wir sie, um bei einer Demontage der Linearführungen ohne eine Neujustage auszukommen.

Im Maschinenbau üblich sind Zylinderstifte mit Übermaß (Toleranzfeld *m*) die in Einheitsbohrungen (*H*) gepresst werden. Herausfallen wird durch Kraftschluss verhindert. Zylinderstifte nach *DIN EN ISO 8734* sind gehärtet und geschliffen (*m6*) und dienen zur lagesicheren Verbindung von Teilen, die nie oder nur selten gelöst werden müssen. Sie benötigen geriebene (*H7*) und absolut koaxiale Bohrungen der zu verbindenden Teile. Darum müssen zum Beispiel die Linearschienen mit der Basisplatte als verschraubtes Paket gemeinsam gebohrt und gerieben werden.

Wer sich nicht in der Lage sieht, so eine Verbindung fachgerecht herzustellen, kann hier aber auch mit Spannstiften arbeiten. Spannstifte sind geschlitzte zylindrische Hülsen aus Federstahl die nicht zwingend eine geriebene Bohrung benötigen. Koaxialität (s. o.), ist aber Bedingung. —*caw*







Alles zum Artikel im Web unter
make-magazin.de/xe8w

Machen Sie mit!

Kennen Sie auch einen raffinierten Trick? Wissen Sie, wie man etwas besonders einfach macht? Wie man ein bekanntes Werkzeug oder Material auf verblüffende Weise noch nutzen kann? Dann schicken Sie uns Ihren Tipp – gleichgültig aus welchem Bereich (zum Beispiel Raspberry, Arduino, 3D-Druck, Elektronik, Platinenherstellung, Lasercutting, Upcyling …).

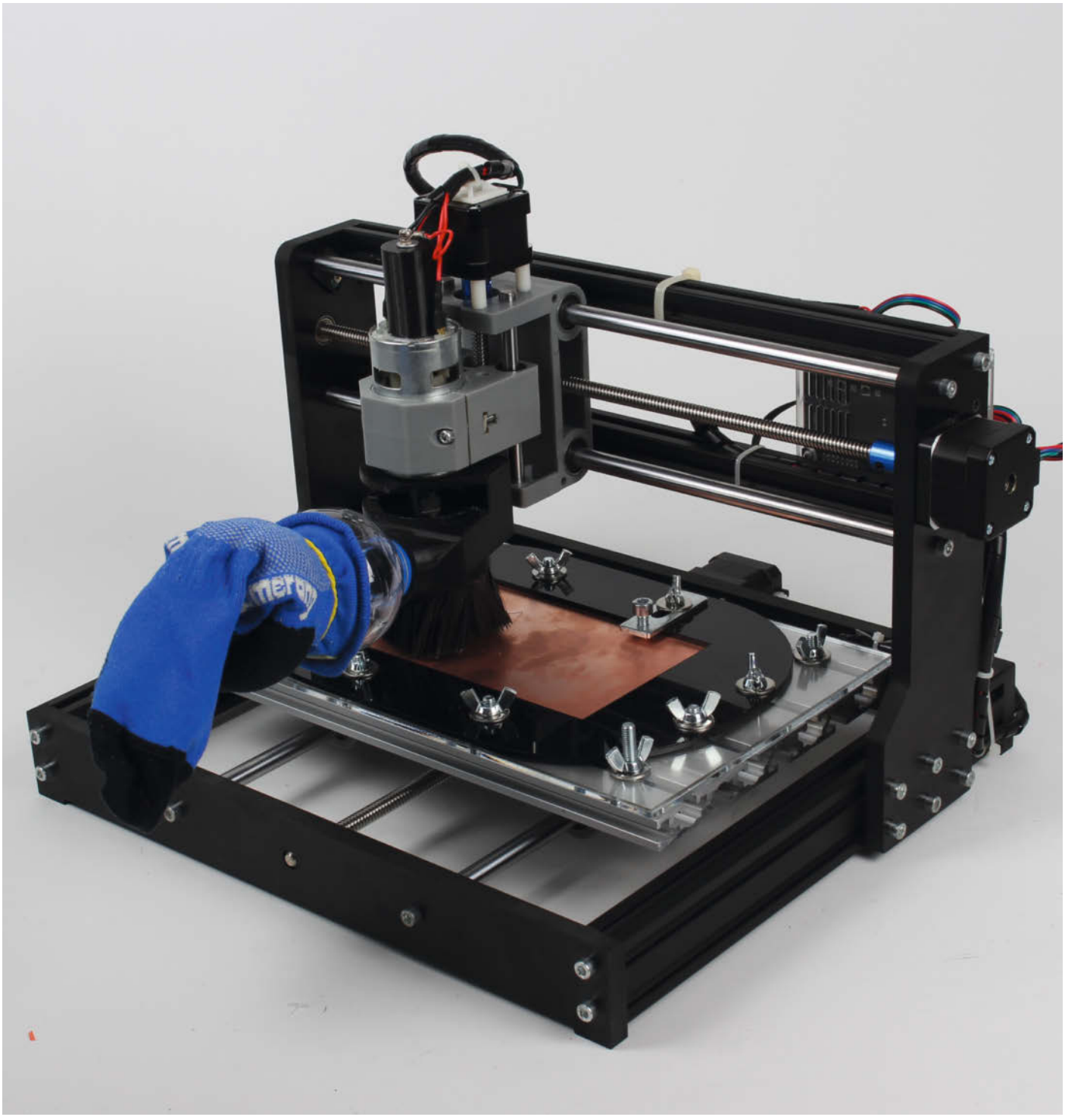
Wenn wir Ihren Tipp veröffentlichen, bekommen Sie das bei Make übliche Autorenhonorar. Schreiben Sie uns dazu einen Text, der ungefähr eine Heftseite füllt und legen Sie selbst angefertigte Bilder bei. Senden Sie Ihren Tipp mit der Betreffzeile *Lesertipp* an:

mail@make-magazin.de

Netzwerk-Fräse

Die kleine Fräse CNC3018 ist ein nützliches Werkzeug beispielsweise zur Platinenherstellung. Doch das USB-Kabel zum steuernden Computer stört. Ersetzen wir es doch durch eine WLAN-Verbindung per ESP8266-Modul.

von Heinz Behling



Laut ist sie, die kleine Fräse. Da möchte man sie nicht unmittelbar neben dem PC-Arbeitsplatz aufstellen. Doch das zur Verbindung erforderliche USB-Kabel darf nicht beliebig lang sein. Meist ist bei 5 bis 10m Länge Schluss mit der sicheren Datenübertragung. Und außerdem liegt die Strippe immer irgendwo im Weg.

Zwar kann man der Fräse die Steuerdateien auch per Speicherkarte übergeben. Dann jedoch lässt sich das *Heightmapping*, also die Korrektur unebener Fräsplatten durch die Steuersoftware, nicht mehr anwenden. Womit wir wieder beim Kabel wären.

Die Fräse selbst wird über eine serielle Schnittstelle gesteuert, die per Chip onboard auf einen USB-Anschluss geschaltet ist (so machen es auch die meisten 3D-Drucker). Von dort gehen die Daten per Kabel an die USB-Buchse des steuernden Computers, wo wiederum ein Treiber die Daten auf einem COM-Port für die Steuersoftware zur Verfügung stellt.

In diesem Artikel soll diese Verbindung zwischen den beiden seriellen Schnittstellen per WLAN hergestellt werden. Die serielle Schnittstelle der Fräse wird dazu auf ein ESP8266-Modul geführt. Das sendet und empfängt die Daten per WLAN an den steuernden Computer, in dem eine Treiber-Software den seriellen Datenverkehr aus dem WLAN auf einen virtuellen COM-Port umleitet. Für die steuernde Software (in diesem Fall *GRBL-Controller* bzw. *Candle*) ändert sich dadurch nichts außer der Nummer des von ihr anzusteuernden COM-Ports.

Kurzinfo

- » ESP-8266-Modul flashen
- » Fräse CNC3018 über WLAN mit Computer verbinden
- » Treiber für virtuellen COM-Port-Treiber installieren und konfigurieren

Checkliste

Zeitaufwand:
1 Stunde

Kosten:
20 bis 50 Euro

Material

- » ESP-01-Modul
- » ESP-01-Breakout-Modul
- » Jumperkabel 4polig

Werkzeug

- » Programmer ESPLink V1.0

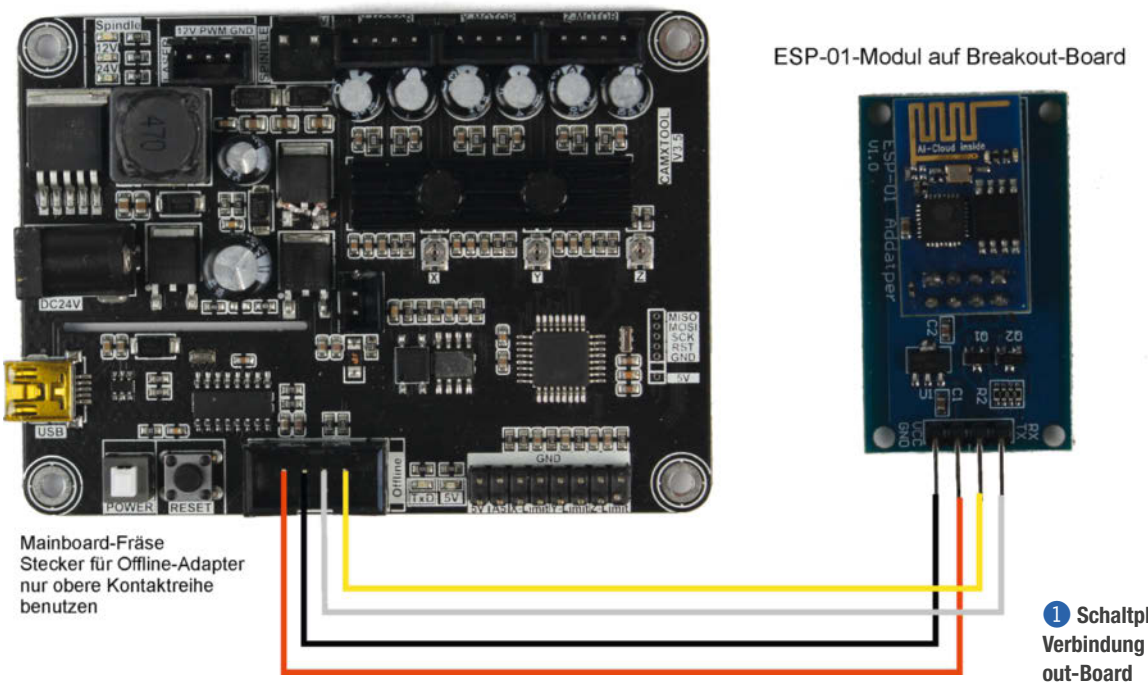
Mehr zum Thema

- » Gustav Wostrack, Der Weg zur Platine Teil 1, Make 6/21, ab Seite 104
- » Gustav Wostrack, Der Weg zur Platine Teil 2, Make 1/22, ab Seite 86
- » Heinz Behling, Billigfräse verbessern, Make 1/22, ab Seite 106

Alles zum Artikel im Web unter
make-magazin.de/xs38

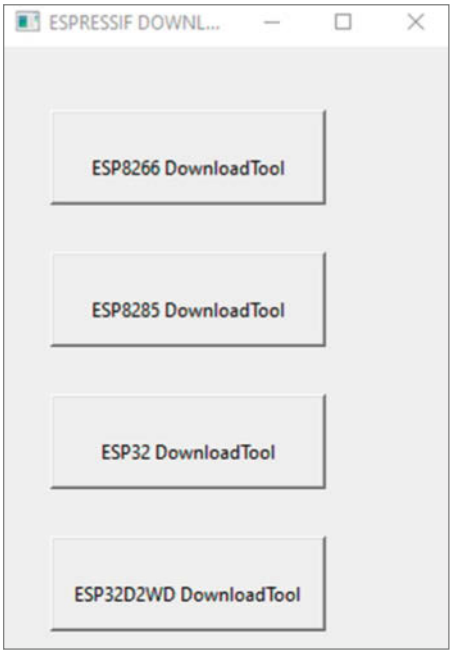
Minimale Hardware

Als zusätzliche Hardware brauchen wir lediglich ein ESP-01-Modul sowie ein dazu passendes Breakout-Board und ein kurzes vierpoliges Jumperkabel. Das Breakout-Board wird entsprechend dem Schaltplan 1 mit dem Steckplatz für den Offline-Adapter auf dem Mainboard der Fräse verbunden. Achtung: Die Reihenfolge der Adern am Stecker zum Breakout-Board entspricht nicht der Reihenfolge am

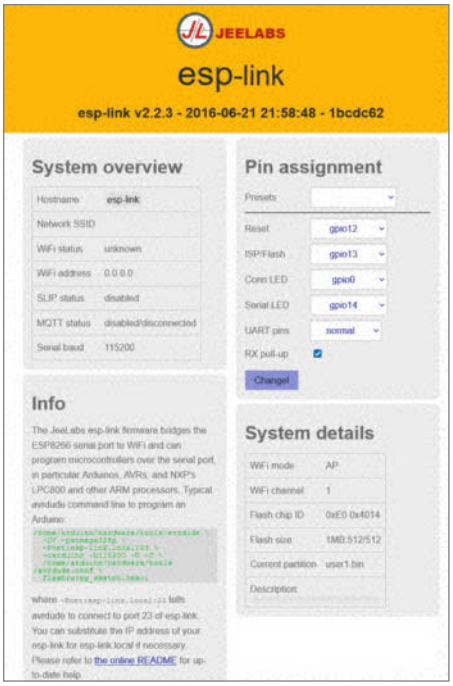


1 Schaltplan für die Verbindung zum Breakout-Board

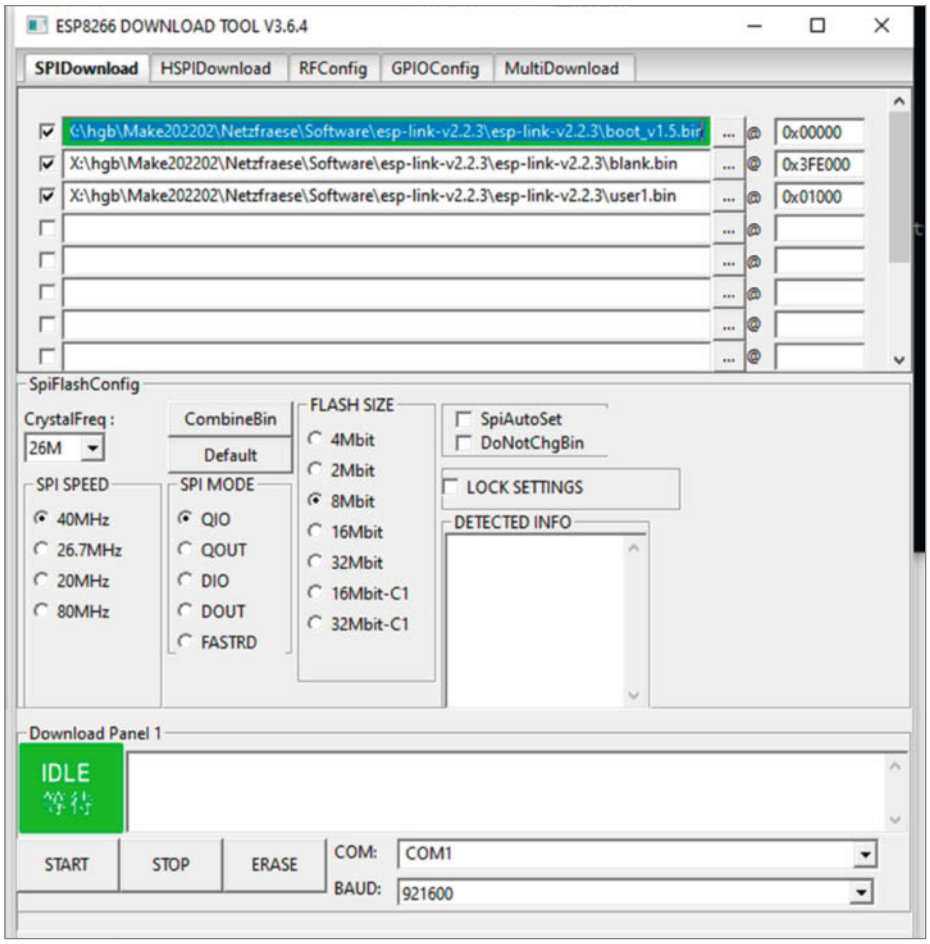
2 Der Programmer für den ESP-01 ist nur so groß wie ein USB-Speicherstick.



3 Hier wählen Sie das Tool für den ESP8266.



5 Beim ersten Start sieht die Konfiguration des ESP-Moduls so aus.



4 Diese drei Bestandteile der ESP01-Firmware müssen Sie im Brennprogramm eintragen.

anderen Ende für den Stecker am Mainboard. Hier müssen die Adern getauscht werden wie im Schaltplan gezeigt!

Die Firmware *ESP-Link* für das ESP-01-Modul wird mithilfe eines kleinen USB-Programmers ins Modul geschrieben. Dazu braucht man außer dem Programmer 2 ein dazu passendes Brennprogramm sowie die Firmware für das ESP-01-Modul. Die Download-Adressen dafür finden Sie über den Kurzinfo-Link. Die heruntergeladenen Archive sind zum Teil .rar- und .tar-Archive. Zum Entpacken eignet sich das Komprimierungsprogramm *7-ZIP*. Dessen Download-Adresse gibt es ebenfalls über den Kurzinfo-Link.

Entpacken Sie zunächst alle Archive. Stecken Sie dann das ESP-Modul in den Programmer. Die Platine des Moduls muss zum USB-Stecker des Programmers zeigen. Anschließend starten Sie das Programm *ESPFlashDownloadTool_v3.6.4.exe*. Wählen Sie dann *ESP8266 DownloadTool* 3.

Die Firmware für das ESP01-Modul besteht aus drei Teilen: *boot_v1.5.bin*, *blank.bin* und *user1.bin*, die jeweils an die richtigen Adressen im Speicher des ESP gebracht werden müssen. Dateispeicherort und Adressen müssen Sie im Fenster des Brennprogramms eintragen 4.

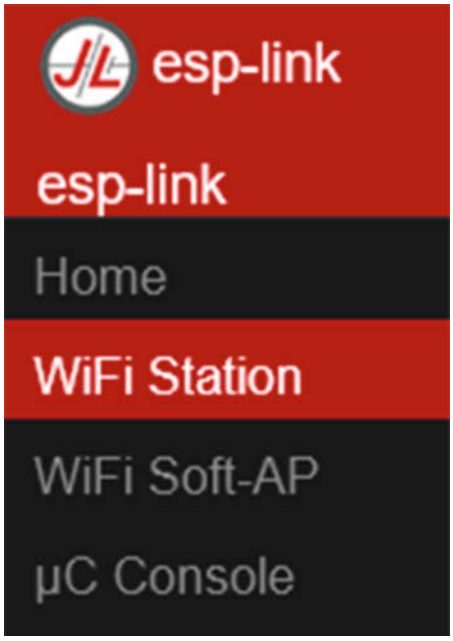
Die Speicheradressen können Sie direkt eingeben, die Speicherorte nach einem Klick auf die drei Punkte rechts des Eingabefeldes. Die Dateien befinden sich im entpackten Firmware-Archiv, das auf Ihrem Computer natürlich anders heißen kann als in diesem Beispiel.

Stecken Sie den Programmer in eine USB-Buchse des Computers und klicken Sie auf *Start*. Falls dann eine Fehlermeldung kommt, ist wahrscheinlich der falsche COM-Port gewählt. In den meisten Fällen ist COM1 aber richtig.

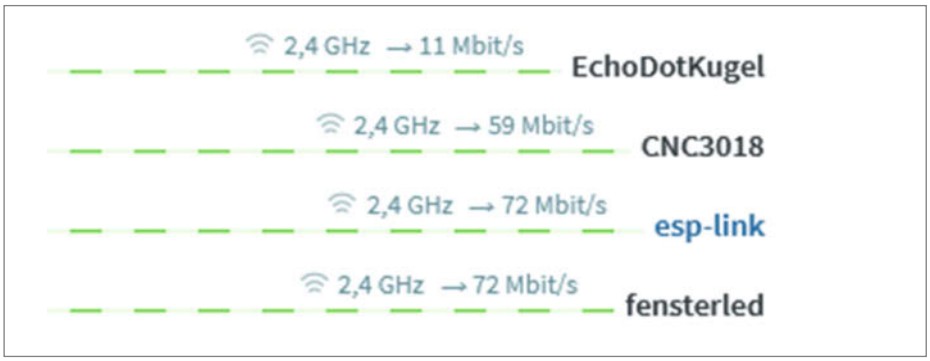
Anschließend stecken Sie das ESP-Modul ins bereits mit der Fräse verbundene Breakout-Board und schalten Sie die Fräse ein.

ESP-Konfiguration

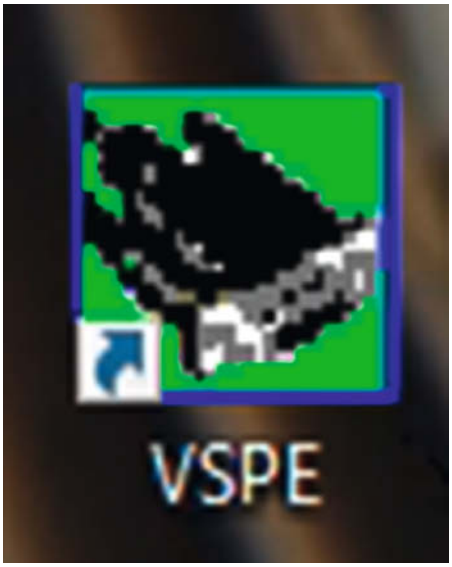
Der ESP01 muss nun noch die Zugangsdaten fürs WLAN erhalten. Das geschieht online. Dazu stellt das ESP-Modul einen Access-Point zur Verfügung, in dass man sich einloggen muss (auf PC, Notebook oder dem Smartphone). Der Name des Access-Points beginnt mit *ESP_* gefolgt von einer Buchstaben-/Zifernkombination, z.B. *ESP_D1BBE2*. Ein Passwort ist zum Einloggen nicht notwendig.



6 Achten Sie darauf, WiFi-Station und nicht WiFi Soft-AP zu wählen.



7 Geschafft: Die Fräse taucht als CNC3018 auf der Router-Homepage auf.



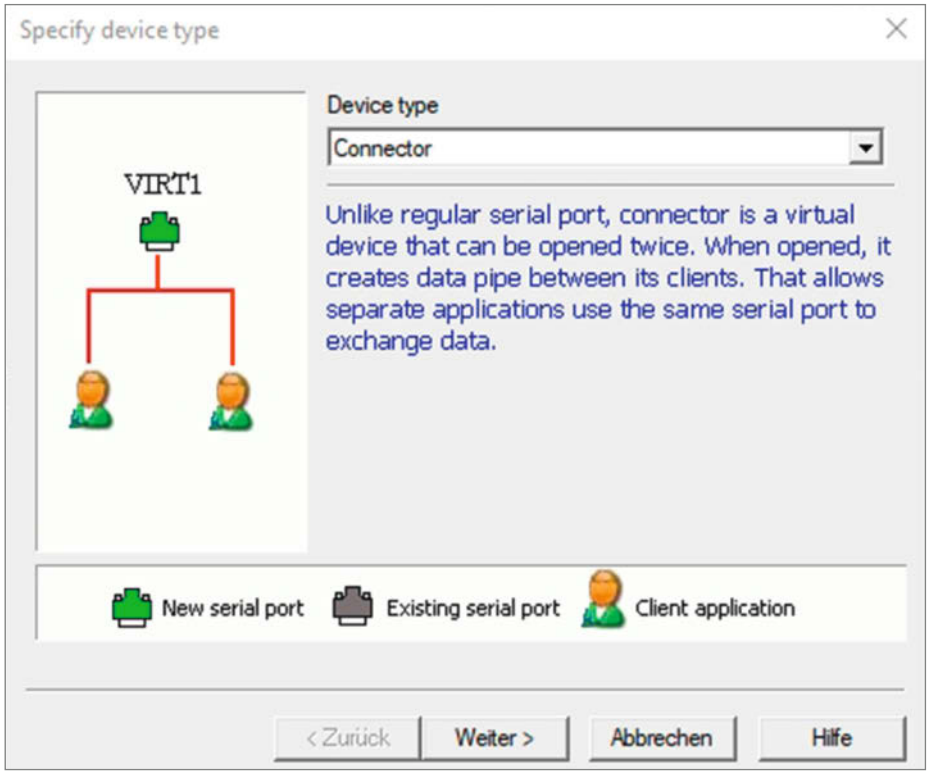
8 Der Virtual Serials Port Emulator erscheint nur als Icon auf dem Desktop, nicht im Startmenü!

9 Sie müssen nun einen virtuellen COM-Port und anschließend einen TCP-Client einrichten.

Anschließend erreichen Sie im Browser über die Adresse `http:192.168.1.4` die Konfigurationsseiten von *esp-link* 5.

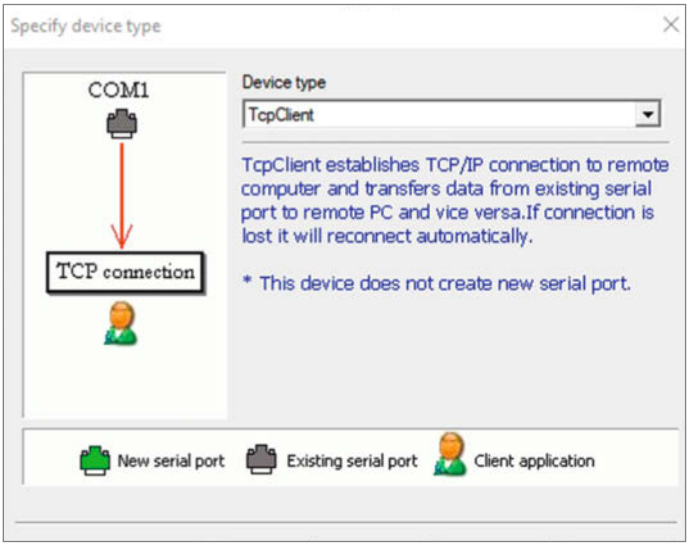
Hier können Sie den Hostnamen des Netzwerkadapters ändern, beispielsweise in CNC3018. Unter diesem Namen erscheint die Fräse dann später im Netzwerk. Links neben den angezeigten Werten klicken Sie im Menü auf *WiFi-Station* 6.

Klicken Sie dann auf *Switch to STA mode*. Jetzt ist ein wenig Geduld angesagt, denn der ESP scannt nun die erreichbaren Funk-Netzwerke. Das kann ein bis zwei Minuten dauern. Danach wählen Sie in der angezeigten Liste

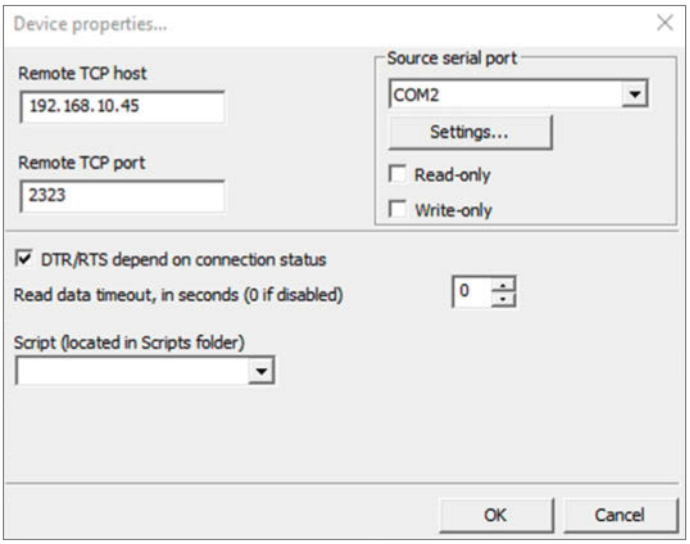


10 So legen Sie einen virtuellen COM-Port an.

11 Erster Teil geschafft: Der neue COM-Port 2 ist bereit.



12 Bitte TcpClient und nicht -Server wählen.



13 Die IP-Adresse erfahren Sie auf der Web-Oberfläche Ihres Routers.

Ihr WLAN aus und geben das Passwort ein. Nach einem Klick auf *Connect* ist der Netzwerk-Adapter einsatzbereit. Sie können auf der Webpage Ihres Routers nachschauen, ob die Verbindung funktioniert 7.

Virtueller COM-Port

Nun müssen Sie dem PC noch beibringen, dass er einen COM-Port einrichtet, der aber statt eines Kabels eine Netzwerkverbindung benutzt. Dazu benutzen wird den *Virtual Serial Ports Emulator* der Firma *Eterlogic*. Es gibt ihn in zwei Versionen: für 32-Bit- und für 64-Bit-Windows-Versionen. Die 32-Bit-Version ist kostenlos. Sie kann auch auf 64-Bit-Systemen zwei Wochen gratis getestet werden, bevor dort ein kostenpflichtiger Registrierungsschlüssel (ca. 30 Euro) notwendig wird. Bei der 32-Bit-Version ist der Registrierungsschlüssel im Download-Archiv enthalten.

Überspielen Sie die für Ihren PC passende Software (siehe Kurzinfo-Link) und entpacken Sie sie auf dem Computer, der später auch die Fräse kabellos steuern soll. Auf dem Desktop erscheint dann das Icon 8.

Starten Sie das Programm und geben Sie den Registrierungsschlüssel ein bzw. klicken Sie auf *Nein*, falls Sie die kostenlose Testzeit nutzen möchten. Im noch leeren Fenster des Emulators klicken Sie auf das Symbol mit dem roten Sternchen 9.

Wählen Sie als *Device type* den *Connector* 10 und nach einem Klick auf *Weiter* die gewünschte Nummer des Port aus, im Beispiel *COM2*.

Sie landen wieder im nun nicht mehr ganz leeren Emulator-Fenster 11.

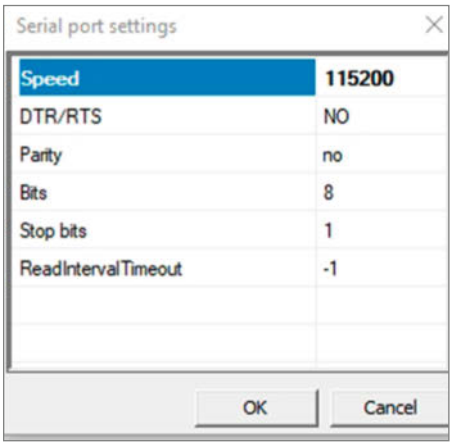
Als Nächstes verbinden wir diesen nur virtuell vorhandenen Port mit einer Netzwerk-(TCP-)Verbindung. Dazu klicken Sie erneut auf das Symbol mit dem roten Sternchen. Diesmal wählen Sie als *Device Type* aber *TcpClient* 12.

Nach *Weiter* geben Sie als *Remote TCP host* die IP-Adresse Ihrer Fräse (siehe Router-Seite) und als *RemoteTCP port 2323* ein und wählen den zuvor eingerichteten COM-Port aus 13.

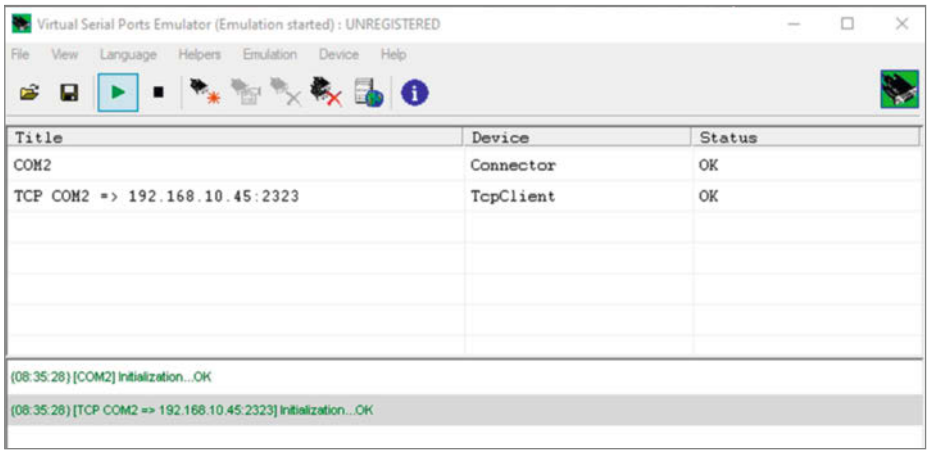
Wählen Sie dann noch *Settings* und danach bei *Speed* 115200 14.

Mit *OK* und *Fertigstellen* gehts weiter. Jetzt checken Sie, ob beide Einträge in der Emulator-Liste enthalten sind und am Ende jeweils ein *OK* steht 15. Falls nicht, haben Sie sich höchstwahrscheinlich bei der IP-Adresse vertippt, den falschen COM-Port gewählt oder die Fräse ist ausgeschaltet.

Jetzt muss nur noch in der Fräsen-Software (in diesem Fall *GRBL-Control* oder *Candle*) dieser Port (COM2) gewählt werden und die Fräse kann wie gewohnt arbeiten. Das USB-Kabel zur Fräse können Sie endgültig entfernen und die Fräse auch weiter entfernt aufstellen. Die Stolperei über das Kabel und die Lärmbelästigung durch die unmittelbar neben dem Computer kreischende Fräse haben ein Ende! —*hgb*



14 Hier muss die gleiche Geschwindigkeit stehen, die auch die Fräse benutzt, also 115200.



15 Glückwunsch: Die Verbindung von der Fräse zum virtuellen COM-Port 2 ist eingerichtet. Die beiden ok in der Status-Spalte zeigen, dass die Verbindung zur Fräse eingerichtet ist.

Maker Faire®

Das Format für
Innovation & Macherkultur

Die nächsten Events

maker-faire.de

3D-Drucker im Smarthome

3D-Drucker sind Geräte, die man überwachen sollte. Schließlich bringen die hohen Temperaturen, mit denen sie arbeiten, durchaus ein gewisses Gefahrenpotential mit sich. Insbesondere daheim und bei Druckvorgängen, die viele Stunden dauern, hat man nicht immer die Zeit dazu. Überlassen wir diese Aufgabe doch unserem Home Assistenten.

von Heinz Behling

Kurzinfo

- OctoPrint-Integration in Home Assistant installieren
- Konfiguration der Integration
- Drucker-Webcam in Home Assistant anzeigen lassen
- Druckerfehler auf dem Smartphone melden

Checkliste

Zeitaufwand:
1 Stunde

Kosten:
0 Euro (zzgl. Kosten für Druckserver)

Material

- **Raspberry Pi** mit Netzteil und Speicherkarte mit *OctoPrint*
- **USB- oder Raspberry-Kamera**

Mehr zum Thema

- Heinz Behling, Komfortdrucker, Make 2/18, ab Seite 120
- Heinz Behling, Intelligentes Heim mit Home Assistant, Make 1/21, ab Seite 100
- Heinz Behling, Heizung unter Kontrolle, Make 2/21, ab Seite 22
- Heinz Behling, Druckserver für mehrere 3D-Drucker, Make 4/20, ab Seite 98

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

Seien Sie mal ehrlich: Sie haben doch sicher auch schon Ihren 3D-Drucker unbeaufsichtigt arbeiten lassen, obwohl in den Bedienungsanleitungen und der einschlägigen Literatur ständig davor gewarnt wird. Wer hat auch schon Zeit/Lust, zehn oder mehr Stunden bei der kleinen Kunststoffspritze auszuharren? Aktuelle Geräte sind zwar recht zuverlässig, trotzdem bleibt da immer so ein mulmiges Gefühl.

In einem Smarthome, das von *Home Assistant* (Installation wurde in Make 1/21 beschrieben) gesteuert und bewacht wird, sollte der elektronische Wächter auch die Aufsicht über den Drucker übernehmen. Dadurch können Sie sich nicht nur jederzeit und überall die aktuellen Druckerdaten (Temperaturen, Druckfortschritt etc.) auf dem Smartphone anzeigen lassen. Der Home Assistant kann Sie auch bei einer Fehlermeldung des Druckers per Smartphone-Alarm darauf aufmerksam machen.

In diesem Artikel erfahren Sie, was Sie dafür an zusätzlicher Soft- und Hardware brauchen und wie Sie alles installieren. Ich setze dabei voraus, dass Sie Ihr Smarthome bereits funktionstüchtig eingerichtet haben. Falls nicht, finden Sie in den Make-Ausgaben 1/21 bis 6/21 eine entsprechende Artikelreihe (erreichbar über den Kurzinfo-Link). Außerdem sollte Ihr 3D-Drucker über einen *OctoPrint*-Druckserver erreichbar sein. Eine Anleitung dazu gab es im Artikel *Komfortdrucker* in der Make 2/18.

OctoPrint-Integration installieren

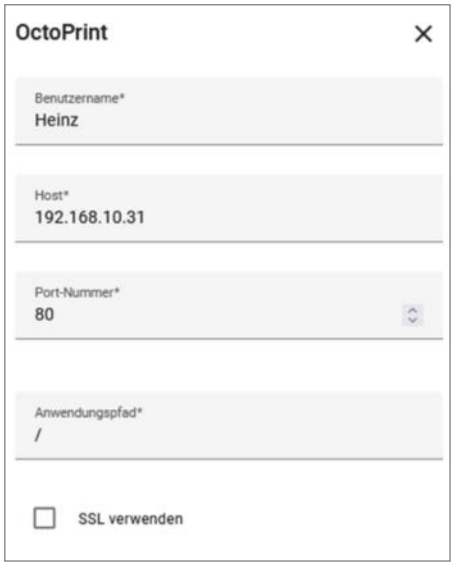
Beginnen wir mit der Installation der OctoPrint-Integration im Home Assistant. Begeben wir uns also im Browser auf *homeassistant:8123*. Dort klicken wir auf *Einstellungen* und auf *Geräte und Dienste*.

Im folgenden Fenster geben Sie als Suchbegriff *Octoprint* ein und klicken dann auf die so gefundene Integration.

Anschließend sind der *Benutzername* für den Drucker sowie im Feld *Host* seine IP-Adresse einzugeben. *Portnummer* und *Anwendungspfad* können Sie einfach übernehmen und dann auf *Absenden* klicken (1).

Home Assistant fordert Sie nun auf, die OctoPrint-Oberfläche des Druckers im Browser zu öffnen (2).

In der Regel erreichen Sie die unter der Adresse *octopi.local*. Dort müssen Sie nämlich



(1) Die IP-Adresse Ihres Druckers können Sie in der Benutzeroberfläche Ihres Routers nachschlagen.

(2) Jetzt ist es Zeit, die OctoPrint-Webseite Ihres 3D-Druckers zu öffnen.

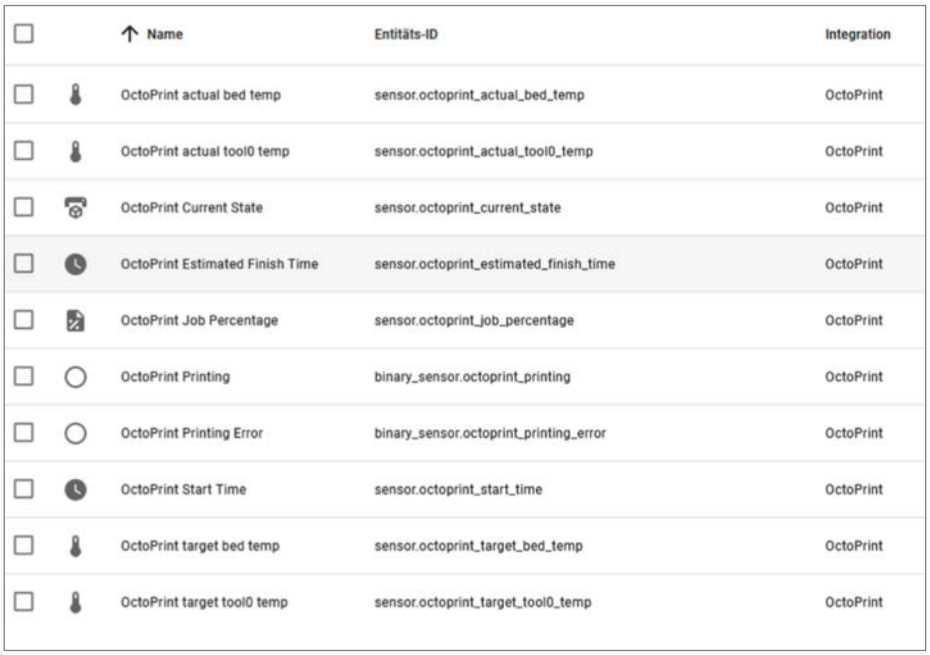
(3) Home Assistant kann erst auf den Drucker zugreifen, wenn es auf der Webseite Ihres Druckers erlaubt wird.

5 Die neue Integration

7 So erhalten Sie einen neuen Bereich für den 3D-Drucker mit passendem Symbol.

den Zugriff auf den Drucker gestatten, indem Sie auf *Erlauben* klicken 3.

Die OctoPrint-Seite darf danach wieder geschlossen werden. Anschließend können Sie in Home Assistant noch den Bereich angeben, in dem Ihr Drucker steht. Ein Klick auf *Fertig* komplettiert die Installation der Integration.



Name	Entitäts-ID	Integration
OctoPrint actual bed temp	sensor.octoprint_actual_bed_temp	OctoPrint
OctoPrint actual tool0 temp	sensor.octoprint_actual_tool0_temp	OctoPrint
OctoPrint Current State	sensor.octoprint_current_state	OctoPrint
OctoPrint Estimated Finish Time	sensor.octoprint_estimated_finish_time	OctoPrint
OctoPrint Job Percentage	sensor.octoprint_job_percentage	OctoPrint
OctoPrint Printing	binary_sensor.octoprint_printing	OctoPrint
OctoPrint Printing Error	binary_sensor.octoprint_printing_error	OctoPrint
OctoPrint Start Time	sensor.octoprint_start_time	OctoPrint
OctoPrint target bed temp	sensor.octoprint_target_bed_temp	OctoPrint
OctoPrint target tool0 temp	sensor.octoprint_target_tool0_temp	OctoPrint

6 Hinter den zehn Entitäten des Druckers stecken wichtige Messwerte und Meldungen.

In der Home-Assistant-Liste erscheint nun die neue Integration mit einem eigenen Fensterchen 5.

Falls Sie mehrere OctoPrint-Drucker in Ihrem Netzwerk installiert haben, erscheint zusätzlich noch ein Fenster, dass Sie auf weitere gefundene Drucker aufmerksam macht. Nach einem Klick auf *Konfigurieren* können Sie die dann ebenfalls einrichten.

Was kann man nun mit der OctoPrint-Integration anfangen? Wie sie sehen, bietet sie zehn Entitäten. Klicken Sie einmal darauf und Sie sehen, welche 6.

Diese Werte und Meldungen sollten Sie sich auf der Übersichtsseite des Home Assistent anzeigen lassen, um Ihren Drucker im Auge behalten zu können. Klicken Sie dazu auf der Übersichtsseite auf die drei Punkte oben rechts und dann auf *Benutzeroberfläche konfigurieren.* Wählen Sie den gewünschten Bereich aus (zum Beispiel Wohnzimmer) oder legen Sie nach Klick auf das Pluszeichen einen neuen an 7.

In diesem Bereich legen Sie dann per Klick auf *Karte hinzufügen* und *Elemente* eine neue Karte an. Geben Sie dann die Entitäten aus Bild 8 ein und klicken Sie auf *Speichern*.

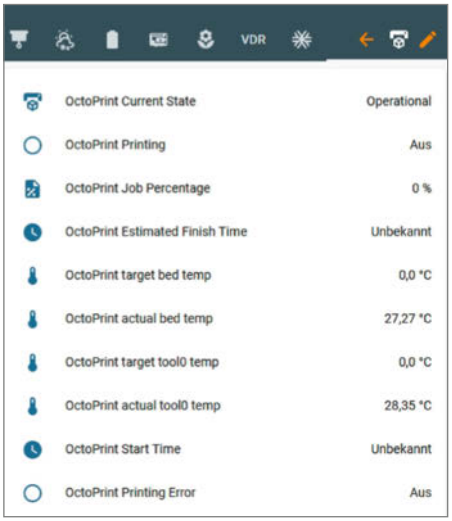
Auf der Übersichtseite sieht das dann so aus wie auf Bild 9.

Kamerabild einfügen

Haben Sie Ihren OctoPrint-Server mit einer Kamera ausgestattet, können Sie auch deren Bild in die Home-Assistant-Übersicht einbauen. Dazu müssen Sie die Datei *configuration.yaml*

```
sensor.octoprint_current_state
binary_sensor.octoprint_printing
sensor.octoprint_job_percentage
sensor.octoprint_estimated_finish_time
sensor.octoprint_target_bed_temp
sensor.octoprint_actual_bed_temp
sensor.octoprint_target_tool0_temp
sensor.octoprint_actual_tool0_temp
sensor.octoprint_start_time
binary_sensor.octoprint_printing_error
```

8 Mit diesen zehn Entitäten haben Sie Ihren 3D-Drucker unter Kontrolle.



9 Der Drucker im Home Assistant

erweitern. Wie das geht, lesen Sie im Smarthome-Artikel in Ausgabe 2/21. Ans Ende der Datei müssen die Zeilen für die Kamera angefügt werden ⑩. Statt *IP-Adresse* müssen Sie natürlich die Adresse Ihres Druckers einsetzen.

Nach dem Speichern der Datei und einem Neustart des Home Assistent kann das Kamerabild in die Übersichtsseite eingefügt werden. Gehen Sie also wieder in den Bereich *3D-Drucker* und klicken Sie erneut auf *Karte hinzufügen*. Diesmal wählen Sie *Bild Glance* aus. In der Kartenkonfiguration tragen Sie dann als Kamera-Entität *camera.octopi* ein ⑪.

Nach dem Speichern steht das Bild dann in der 3D-Drucker-Übersicht bereit.

Alarm auf dem Smartphone

Sich nun nur die aktuellen Werte des Druckers auf dem Bildschirm anzeigen zu lassen, ist aber kein wirklicher Nutzen, da könnten wir auch stundenlang neben dem Drucker sitzen bleiben. Komfortabel wird es erst, wenn wir nun auch den Home Assistent diese Überwachungsaufgabe erledigen lassen.

Im Folgenden zeige ich Ihnen, wie man auf dem Smartphone alarmiert wird, wenn es einen Druckerfehler gibt. Dazu ist auf dem Smartphone die Home-Assistent-App notwendig (gibt es für Android- und Apple-Smartphones). Installieren Sie diese aus dem entsprechenden Store und starten Sie sie. Auf der Startseite wählen Sie *Fortfahren* und geben danach die URL des Home Assistent ein. Das kann die lokale URL sein (also *http://homeassistant.local:8123*) oder eine auch extern erreichbare. Falls Sie zum Beispiel mit der *Home-Assistant-Cloud* arbeiten, geben Sie die entsprechende Adresse ein, die auf *.nabu.casa* endet ⑫.

Danach sind noch *Benutzername* und das *Passwort* notwendig. Ein Fenster weiter wird Ihnen der *Gerätename* des Smartphones genannt. Notieren Sie sich diese Bezeichnung, sie wird noch gebraucht ⑬.

Danach ist die Home-Assistant-App einsatzbereit. Sie zeigt Ihnen die komplette HA-Oberfläche an.

Jetzt brauchen Sie nur noch eine kleine Automatisierung, um die Fehlermeldungen des Druckers auszuwerten. Klicken Sie also auf der HA-Oberfläche des Computers in *Einstellungen* und *Automatisierungen & Szenen* auf *Automatisierung erstellen*, wählen Sie eine *leere Automatisierung* und geben Sie das folgende ein:

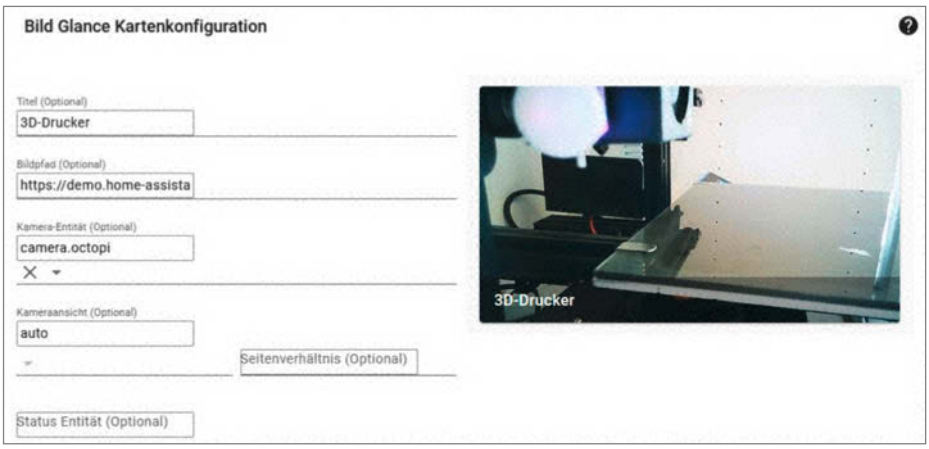
- Name: *Drucker-Alarm per Smartphone*
- Auslösertyp: *Zustand*
- Entität: *binary_sensor.octoprint_printing_error*
- Aktionstyp: *Dienst ausführen*
- Dienst: *Benachrichtigung: Send a notification via mobile_app_Gerätekennung* (Gerätekennung durch die zuvor notierte Kennung ersetzen)
- Message: *Druckeralarm*
- Title: *!!Druckeralarm!!*

Dann noch speichern und Ihr Druckerwächter ist im Dienst. Kommt es jetzt zu einer Fehlermeldung durch den OctoPrint-Server, wird auf dem Smartphone eine entsprechende Meldung erscheinen ⑭.

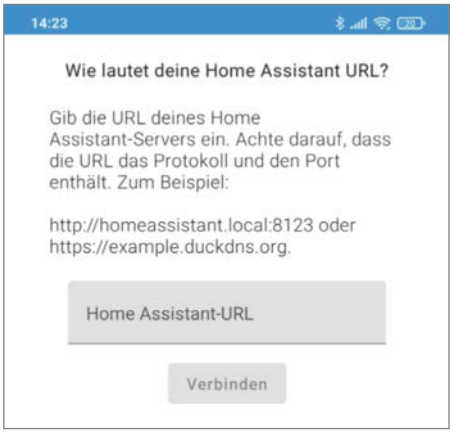
Sie können die Automatik auch noch um weitere Sicherheitsfunktionen erweitern: Falls Sie den Drucker zum Beispiel über eine smarte Schaltsteckdose angeschlossen haben, können Sie im Alarmfall den Drucker damit einfach abschalten. Entdecken Sie einfach die Möglichkeiten. —*hgb*

⑩ Kameraeintrag in configuration.yaml

```
camera:
  - platform: mjpeg
    name: OctoPi
    still_image_url: http://IP-Adresse/webcam/?action=snapshot
    mjpeg_url: http://IPO-Adresse/webcam/?action=stream
```

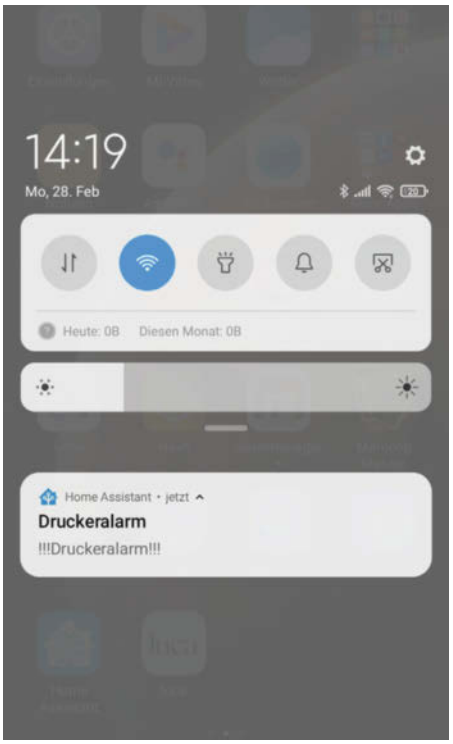


⑪ **So sieht der Eintrag für das Kamerabild aus.**



⑫ **Hier geben Sie die lokale oder die Internet-Adresse Ihres Home Assistent ein.**

⑬ **Die Gerätebezeichnung des Smartphones**

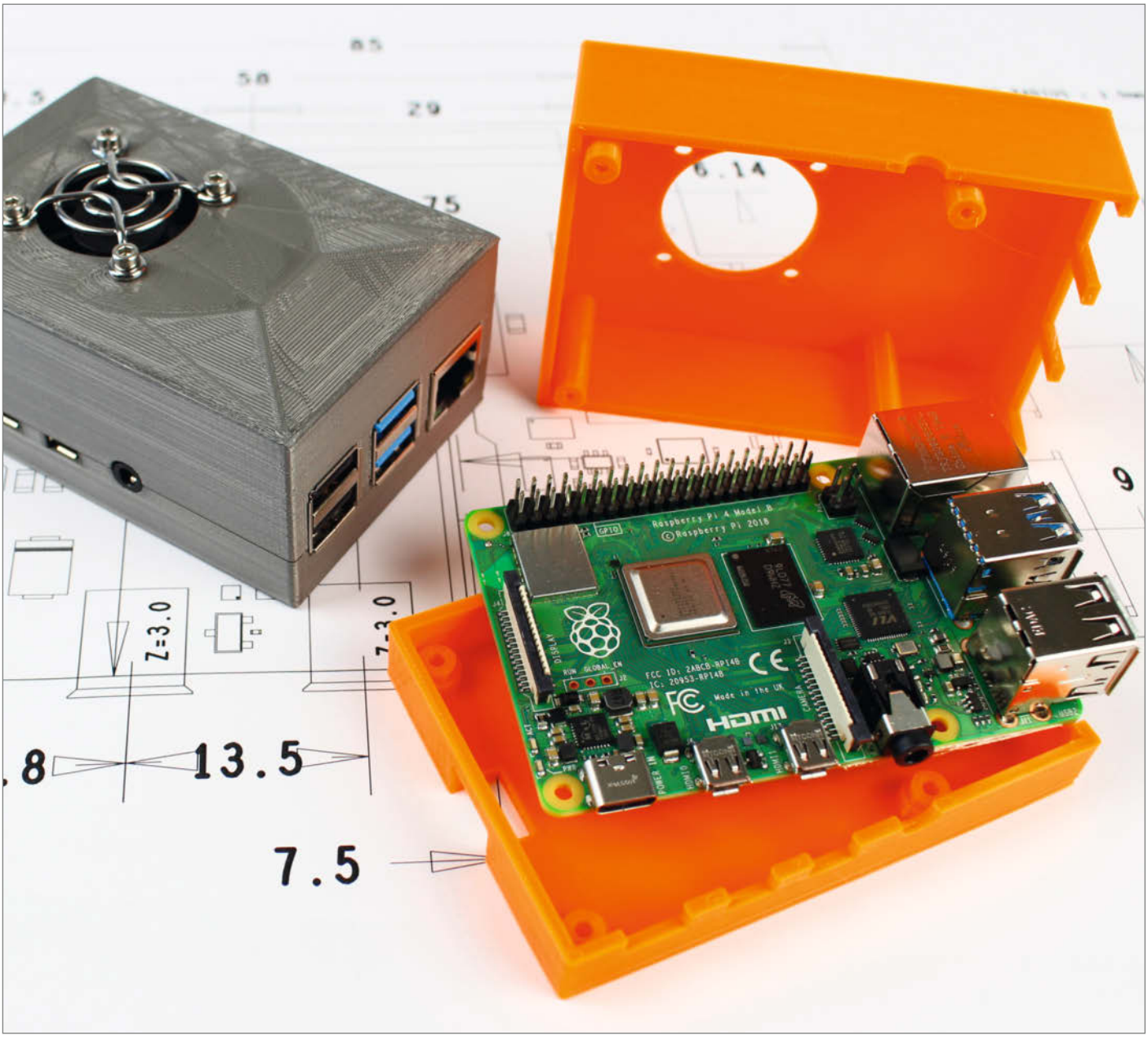


⑭ **Der Drucker gibt Alarm auf dem Smartphone.**

Raspberry-Pi-Gehäuse mit FreeCAD, Teil 4

In unserer Artikelserie zeigen wir, wie Sie mit der kostenlosen Software FreeCAD Ihr eigenes Gehäuse für Einplatinenrechner wie dem Raspberry Pi entwerfen. In den ersten drei Teilen haben wir gezeigt, wie Sie zu einer günstigen Bauform kommen, die Größe des Gehäuses ermitteln und die Maße für die Aussparungen herausfinden können. In diesem vierten Teil ermitteln wir die Position und Abmessungen der Hülsen, mit denen Sie die Platine im Gehäuse befestigen und bereiten die Montage eines Lüfters vor.

von Matthias Mett



Die Grundform des Gehäuses für den Raspberry Pi und die Konstruktion der Öffnungen für die Anschlüsse haben wir in den vorigen Folgen unserer Artikelserie und den dazugehörigen Videos gezeigt.

In dieser Folge ermitteln wir die Position und die Größe der Hülsen, auf denen die Platine im Inneren des Gehäuses sitzen soll (1) (2). Außerdem wollen wir auf der Innenseite der Oberschale einen Lüfter anbringen, wofür neben der eigentlichen Belüftungsöffnung (3) noch Befestigungsbohrungen (4) nötig sind. Die Schrauben sollen dann von außen durchgesteckt werden, durch den Lüfter gehen, der dann mit Muttern fixiert wird. Auch diesmal gibt es für die ganze Konstruktion in FreeCAD ein Schritt-für-Schritt-Video online (siehe Link in der Kurzinfo).

Befestigungshülsen

Für die Position der Befestigungshülsen können wir wieder auf die technische Zeichnung der *Raspberry Pi Foundation* zurückgreifen, die Sie auf der Webseite herunterladen können. Sie finden die technische Zeichnung im Bereich *Raspberry Pi 4 Model B specifications* unter *Raspberry Pi 4 Model B mechanical drawings* (siehe Link in der Kurzinfo).

Die Hülsen in der Unterschale (1) sind kürzer, damit die Öffnungen für den USB-C-Anschluss (5) und die Micro-HDMI-Anschlüsse (6) vollständig unterhalb des Gehäuseschnitts zwischen Ober- und Unterschale liegen. Dementsprechend sind die Hülsen in der Oberschale (2) länger. Stellt man die Oberschale auf die Unterschale, ist zwischen den Hülsen noch genau so viel Platz, dass die Platine dazwischen passt, so sitzt diese später fest im Gehäuse. Von unten werden schließlich Schrauben durch die Unterschale und die Befestigungsbohrungen der Platine in die Oberschale gedreht, sodass die Verschraubung von oben unsichtbar bleibt.

Zwei Bohrungen für die Befestigung sitzen direkt an den beiden Ecken der Schmalseite der Platine mit dem SD-Karten-Halter. Die anderen beiden Bohrungen befinden sich von der anderen Schmalseite zurückgesetzt, hinter den beiden USB-Anschlüssen und dem Netzwerkanschluss. Glücklicherweise sind die Maße für die Mittelpunkte der Bohrungen in der technischen Zeichnung der *Foundation* direkt angegeben: Bei den zwei Bohrungen an den Ecken sitzen sie mit jeweils 3,5mm Abstand von der Längs- und Breitseite; die beiden anderen sitzen von den jeweiligen Längsseiten ebenfalls 3,5mm entfernt und von der Schmalseite mit der SD-Karte gemessen beträgt der Abstand 61,5mm. Alle Bohrungen haben einen Durchmesser von 2,7mm, ihr äußerer Ring (da fehlt der grüne Lack und man sieht das nackte Platinenmaterial) hat einen Durchmesser von 6mm (7).

Kurzinfo

- **Konstruieren eines einfachen Raspberry-Pi-Gehäuses (4, Model B) mit kostenloser Software in 3D**
- **Ermittlung von Maßen aus technischer Zeichnung und direkt an der Hardware**
- **Berechnung und Konstruktion von Befestigungsvorrichtungen für Platine und Lüfter**
- **Schritt-für-Schritt-Anleitung als Video**

Checkliste

Zeitaufwand:
etwa drei Stunden für die komplette Konstruktion

Kosten:
keine für die Konstruktion, je nach Material für den Druck ab 1 Euro mit dem eigenen Drucker und ab 25 Euro beim Dienstleister

Konstruieren:
Arbeit mit CAD-Software, aber keine Vorkenntnisse nötig

Computer:
Desktop oder Notebook mit Windows, macOS oder Linux

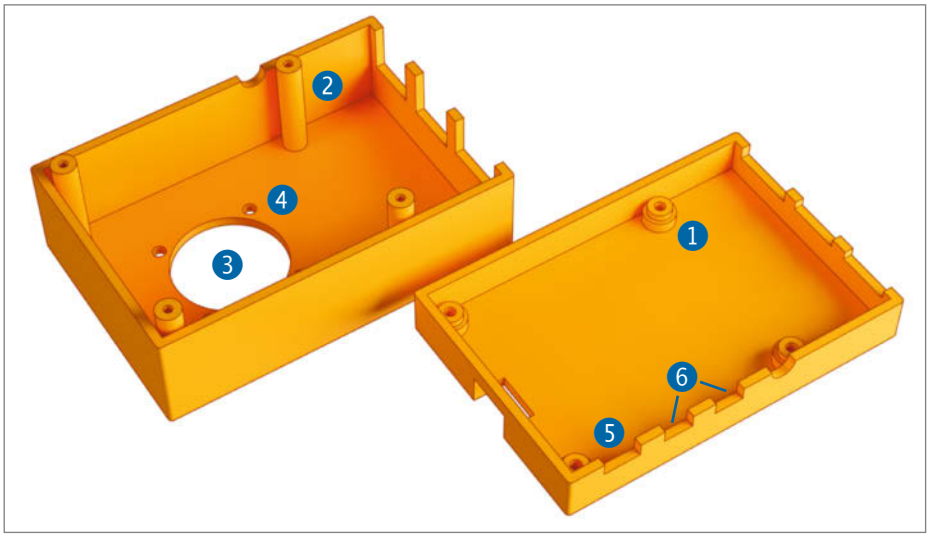
3D-Druck:
mit eigenem Drucker, im Fablab oder beim Dienstleister

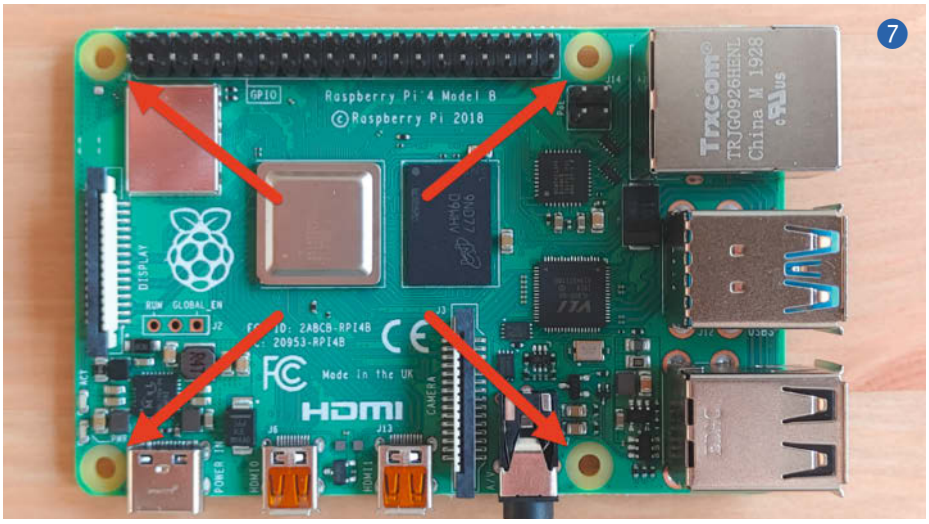
Alles zum Artikel im Web unter
make-magazin.de/x8cu

Mehr zum Thema

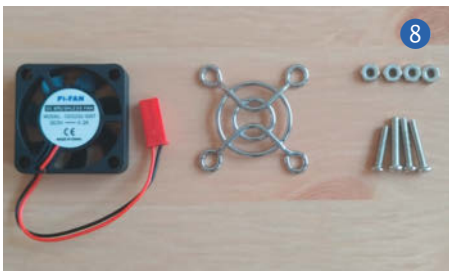
- Matthias Mett, Gehäusebau mit FreeCAD, Teil 1, Make 5/21, S. 116
- Matthias Mett, Gehäusebau mit FreeCAD, Teil 2, Make 6/21, S. 116
- Matthias Mett, Gehäusebau mit FreeCAD, Teil 3, Make 1/22, S. 80
- Matthias Mett, 3D-Entwurf mit FreeCAD, Make 1/21, S. 128
- Matthias Mett, Stempeln mit FreeCAD, Make 4/20, S. 106
- Matthias Mett, Ring frei für FreeCAD, Make 2/20, S. 114
- Billie Ruben, CAD-Designtipps für den 3D-Druck, Make 1/21, S. 138
- Peter König, Gratis-CAD für Maker, Make 4/18, S. 90
- Video: Schritt für Schritt zur FreeCAD-Konstruktion

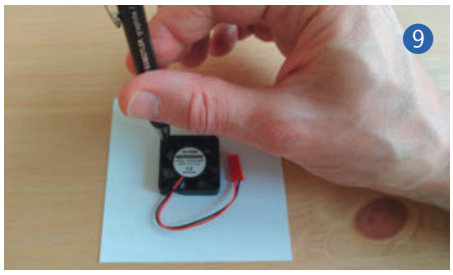
Die Höhe der Hülsen in der Unterschale berechnen wir aus der Höhe des USB-C-Anschlusses von 3,2mm und der Platinendicke von 1,5mm, zusammen 4,7mm. Dies ziehen wir von den 12mm Höhe der Unterschale ab, das ergibt 7,3mm. Hiervon müssen wir aber noch





7


8


9

die Höhe des Bodens von 2,5mm abziehen, was eine Gesamthöhe von 4,8mm für die Hülsen in der Unterschale ergibt, gemessen von der Innenseite des Bodens.

Die Höhe der Hülsen in der Oberschale berechnen wir einmal aus deren Höhe von 26mm minus der Dicke ihres Deckels von 2,5mm. Da die Hülsen um die Höhe des USB-C-Anschlusses von 3,2mm in die Unterschale hineinragen, rechnen wir dieses Maß noch hinzu und kommen auf 26,7mm Höhe, wieder gemessen von der Innenseite des Deckels.

Unsichtbar schrauben

Um die beiden Schalen samt Platine zusammenzuhalten, wollen wir Schrauben von unten durch die Hülsen der Unterschale und die Löcher der Platine durchstecken und in den Hülsen der Oberschale verschrauben – die unteren Hülsen brauchen deshalb eine weitere Bohrung, in die sich die Schrauben einfach einstecken lassen, die oberen eine engere, in die später ein passendes Gewinde geschnitten wird. Zum Versenken der Schraubenköpfe in der Unterschale wollen wir möglichst wenig Aufwand betreiben, welcher beispielsweise bei Schrauben mit Senkkopf notwendig wäre. Daher nehmen wir einfache Zylinderkopfschrauben mit Innensechskant.

Bei einem Bohrungsdurchmesser von 2,7mm in den vorgegebenen Bohrungen der Platine bietet sich eine Schraubengröße von M2,5 an. Bei dieser ist der Schraubenkopf 2,5mm hoch und hat einen Durchmesser von 4,5mm. Dieser sollte sich bei einem Durchmesser der Hülsen von 6mm problemlos versenken lassen. Die Länge der Schraube berechnet sich zum einen aus der Länge der Hülse in der Unterschale (inklusive des Bodens) von 7,3mm; zum anderen soll die Schraube mit genügend Länge auch noch im später geschnittenen Gewinde der Hülse der Oberschale sitzen. Da die Schraube nur bis zur Hälfte ihrer Länge mit Gewinde ausgestattet ist, kommen wir so auf eine gewünschte Schraubenlänge von etwa der doppelten Länge der Bohrung durch die Unterschale, also 14mm. Das Schraubenmaß wäre also M2,5 x 14.

Den Durchmesser der Bohrung in den Hülsen der Unterschale können wir mit 2,7mm von den Platinenlöchern übernehmen, damit sollten sich die Schrauben durchstecken lassen. Da wir in die Hülsen der Oberschale nach dem Druck die Gewinde hineinschneiden wollen (für den direkten Druck sind die M2,5-Gewinde zu fein), benötigen wir hier eine Kernlochbohrung von 2,1 mm. Da beim 3D-Druck solche Löcher in der Regel kleiner ausfallen, bohren wir sie nach dem Druck gegebenenfalls nach.

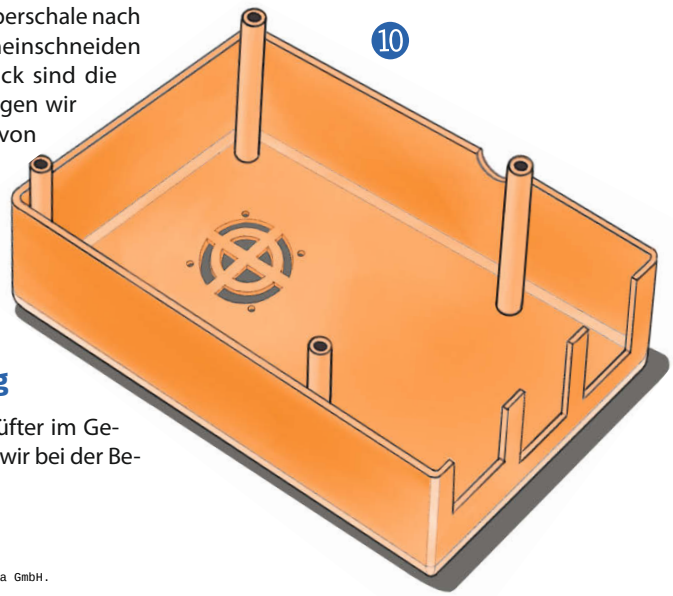
Lüftungsöffnung

Wir wollen außerdem einen Lüfter im Gehäuse platzieren. Daher haben wir bei der Bestellung des Raspberry Pi direkt einen passenden Lüfter mit geordert. Der Lüfter kam mitsamt einem kleinen Metallgitter sowie passenden Schrauben und Muttern 8. Die mitgelieferten Schrauben haben ebenfalls die Maße M2,5 x 14. Bei einer Höhe des Lüfters von 7,8mm lassen sich die Schrauben durchstecken und bei einer Wanddicke der Oberschale von 2,5mm mit den Muttern verschrauben, selbst wenn man das Abdeckgitter einfügt. Auf der Unterseite des Lüfters sind Aussparungen zum Versenken des Schraubenkopfes vorgesehen, was nochmals eine Platzreserve bietet.

Um die Position der Bohrungen im Lüfter herauszufinden, kann man ihn auf ein Blatt Papier legen und mit einem geeigneten dünnen Stift (etwa einem Druckbleistift) durch die Löcher hindurch die Bohrungen anzeichnen, die man anschließend auf dem Papier misst 9. Unser Lüfter hat Bohrungen mit 3,3mm Durchmesser und einem Abstand von 24mm zu den jeweiligen Nachbarn. Der Durchmesser des Lüfterrotors beträgt 28mm.

Im ursprünglichen Entwurf war für den Lüfter ein in die Oberschale integriertes Gitter vorgesehen 10. Da dem Lüfter allerdings ein passendes Gitter beilag, haben wir jetzt nur vier Schraubenlöcher mit einem 28mm großen Loch in der Mitte konstruiert, worauf das Abdeckgitter zu liegen kommt. So sparen wir nochmal eine Menge Konstruktionsaufwand und verwenden alle mitgelieferten Teile.

Mit diesem Teil unserer Artikelserie haben wir jetzt alle Maße ermittelt, um die Hülsen zum Befestigen der Platine und die Bohrungen des Lüfters in FreeCAD zeichnen zu können. Hierzu finden Sie online, wie zu den vergangenen zwei Teilen, eine Schritt-für-Schritt-Anleitung als Video. Dort zeigen wir, wie Sie mit FreeCAD-Werkzeugen wie *Additiver Zylinder* die Hülsen hinzufügen und mittels *Abzuziehender Zylinder* die Bohrungen aussparen und die Konstruktion unseres Gehäuses abschließen. Der 3D-Druck, die Nachbearbeitung und Montage ist dann Thema der nächsten Folge unserer Serie. Jetzt aber erst mal viel Spaß beim Konstruieren! —*pek*


10

betterCode()

API 2022

Die Heise-Konferenz zu Design, Entwicklung und Management von Web-APIs

27. April 2022 – Online

Ob Private oder Public APIs – Software- und Webentwickler müssen schwierige Fragen beantworten: Welche **Architekturparadigmen**, welche **Verfahren**, welche **Protokolle** nutze ich, wann welche besser nicht? Wie gewährleiste ich **Qualität**, **Kompatibilität** und **Sicherheit**?

Die Vorträge und Workshops der betterCode() API bieten konzeptuelles Wissen, neueste Entwicklungen der Web-API-Entwicklung sowie umsetzbares Praxis-Know-how.

Das sind die Themen der betterCode() API am 27. April 2022:

- Was macht »gutes« API-Design aus?
- Geht's auch ohne REST?
- OAuth2- und OIDC-Authentifizierung mit Keycloak
- API Economy: Wie verdient man Geld mit APIs?
- Deklarative APIs am Beispiel Kubernetes
- Interprozesskommunikation (IPC) für Microservices mit gRPC
- Wie wird eine API zum Produkt?

Jetzt buchen – und alle 20 Aufzeichnungen der früheren Auflagen der betterCode() API erhalten.

Jetzt Tickets zum Frühbucher-Rabatt sichern!

Goldsponsor

axway

Veranstalter

iX heise Developer dpunkt.verlag

Workshops: 18. Mai und 23. Mai zu Keycloak, GraphQL und API-Sicherheit

api.bettercode.eu

M5Stack Tough

Robustes ESP32-Modul mit Touchscreen

M5Stack

Von *M5Stack* gibt es nun mit dem *Tough* ein ESP32-Modul für den Einsatz in staubiger oder feuchter Umgebung. Möglich werden soll dies durch ein neues Gehäuse mit Gummidichtungen und wasserdichten Kabeldurchführungen. Die Outdoor-Fähigkeiten unterstreicht auch die UV-Beständigkeit des leuchtend gelb-orangen Gehäuses. „Wasserdicht" heißt in diesem Zusammenhang laut Hersteller *spritzwassergeschützt*; ein Betrieb unter Wasser ist also nicht möglich.

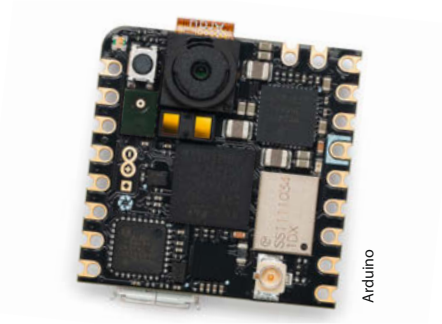
Gegenüber den bisherigen M5Stack-ESP-Modulen wurde auch die Stromversorgung geändert. Nun ist ein 12V-Eingang vorhanden. Das dürfte Probleme mit Spannungsabfällen bei längeren Versorgungskabeln gegenüber der bisherigen 5V-Versorgung deutlich verringern. Kabel können über zwei schraubbare Durchführungen mit Dichtung ins Innere geführt werden. Das zur Programmierung erforderliche USB-Kabel wird bereits mit einem entsprechenden Schraubkopf mitgeliefert und kann ebenfalls spritzwassergeschützt mit dem Modul verbunden werden.

Das Tough-Modul enthält neben dem ESP32 mit Bluetooth und WLAN noch ein Touch-Farb-LC-Display mit 320 × 240 Pixeln, einen per I²S angesteuerten Soundverstärker mit 1W-Lautsprecher sowie einen Mikro-SD-Kartenslot. Der ist allerdings nur nach Öffnen des Gehäuses zugänglich. —*hgb*

Hersteller	M5Stack
URL	make-magazin.de/xga4
Preis	49,90 US-$

Nicla Vision

KI-fittes Kameramodul von Arduino


Arduino

Auf der mit 22,85mm im Quadrat messenden (und damit nur briefmarkengroßen) Platine hat Arduino eine Menge an Rechenleistung und Sensorik etwa für Objekterkennung und -verfolgung untergebracht: Da ist als erstes der Mikrocontroller *STM32H747AII6* mit je einem M7- und M4-32-Bit-Kern. Diese werden mit bis zu 480 bzw. 240MHz getaktet. Für die Verbindung nach außen sorgt ein *Murata-1DX*-Modul mit WLAN 802.11b/g/n und Bluetooth 4.2 BR/EDR/LE (mit Anschluss für eine externe Antenne) sowie eine OTG-fähige USB2-Schnittstelle.

Die Kamera hat einen 2-Megapixel-CMOS-Sensor mit 10-Bit-ADC und soll *TinyML* unterstützen. Direkt daneben hat eine RGB-LED Platz gefunden, die über I²C angesteuert wird und auch als Beleuchtung bei der Bildaufnahme dienen kann. Die Liste der Onboard-Sensoren ist umfangreich: Da gibt es einen Näherungssensor *VL53L1CBV0FY/1* mit einer Reichweite von 4m, der mit unsichtbarem UV-Laserlicht arbeitet. Im Chip namens *LSM6DSOX* sitzt ein 3D-Beschleunigungsmesser sowie ein 3-Achsen-Gyroskop. Hinzu kommt ein Mikrofon sowie ein vorkalibrierter Batterie-Überwachungschip. Für die Datensicherheit sorgt ein *NXP-SE050C2*-Cryptochip.

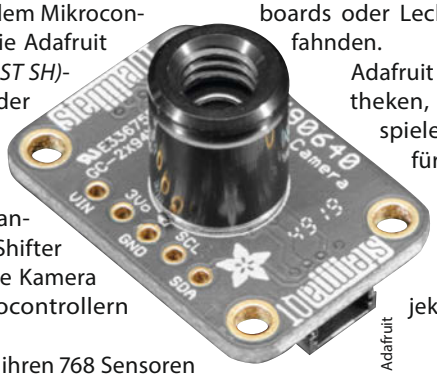
An Schnittstellenanschlüssen bietet das Board UART, I²C und SPI sowie zehn digitale I/O-Pins, zwei analoge Eingänge und 12 PWM-Ausgänge. Allerdings sind die Anschlüsse mehrfach belegt, sodass sich nicht alles gleichzeitig nutzen lässt. Die Spannungsversorgung kann mit 5V (über USB oder VIN-Pin) oder per 3,7V-Akkuzelle erfolgen. Programmieren soll man die *Nicla Vision* beispielsweise in *MicroPython*, etwa wenn man die Plattform *OpenMV* nutzen will, oder über die Arduino IDE in C++. Das Board soll außerdem gut zu den anderen Profi-Modulen der Serien *Portenta* und *MKR* von Arduino passen. —*hgb*

Hersteller	Arduino
URL	store.arduino.cc/products/nicla-vision
Preis	95 €

MLX90640 IR Thermal Camera

Preiswerte Thermokamera-Modul für Experimente

Dieses Breakout-Board von *Adafruit* erlaubt einen günstigen Einstieg in Experimente mit Wärmebildern. Für knapp 70 Euro bekommt man ein Kamera-Modul mit 32 × 24 Pixeln Auflösung und einer effektiven Bildrate von 16Hz. Per I²C-Schnittstelle kommuniziert die Kamera mit praktisch jedem Mikrocontroller, entweder über die Adafruit typischen *STEMMA-QT-(JST SH)*-Kabel oder per Pin-Header zum Auflöten. Das Modul gibt es mit 55° × 35° oder 110° × 70° Sichtfeld. Auf dem Board sind Spannungsregler und Level-Shifter integriert, sodass sich die Kamera an 3,3V- und 5V- Mikrocontrollern betreiben lässt.



Adafruit

Die Kamera misst mit ihren 768 Sensoren Temperaturen von -40°C bis 300°C (+/- 2°C bei 0-100°C) und kann diese Daten etwa 16 mal pro Sekunde an den Controller liefern. Somit kann man dann seine eigenen Anwendungen wie eine Mini-Thermokamera für die Überwachung von Räumen auf Brände im Smarthome oder die Auswertung von Unfallsituationen mittels KI als Projekt angehen. Aber man kann auch einfach nach Wärmebrücken im Haus, defekten Chips auf Motherboards oder Lecks in Wasserleitungen fahnden.

Adafruit stellt wie immer Bibliotheken, Dokumentation, Beispiele und natürlich Videos für Python und die Arduino-IDE zur Verfügung und so gelingt der Einstieg in die Entwicklung eigener Projekte recht einfach. —*caw*

Hersteller	Adafruit
URL	www.adafruit.com/product/4407
Preis	74,95 US-$

Barcode HAT für Pi

Ausprobiert —von Make:—

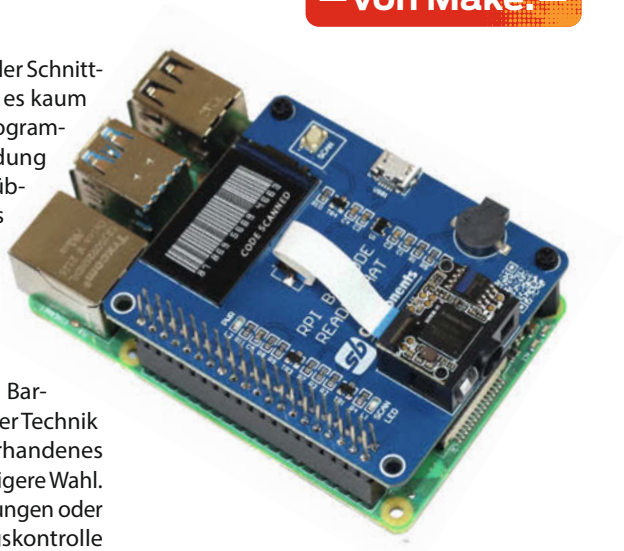
Bar- und QR-Codes einfach einlesen

Der Raspberry Pi *Barcode HAT* (**H**ardware **A**ttached on **T**op) wird auf die 40-polige Anschlussleiste eines Raspberry Pi gesteckt, weitere Anschlüsse sind nicht nötig. Auf dem HAT sind neben dem Scanner noch ein Piepser, ein 1,14"-LCD, ein Micro-USB-Port und ein Button zum Auslösen des Scans verbaut. Durch lange Header-Pins können weitere HATs aufgesteckt werden.

Das Board verwendet das Scannermodul *DE2120 (DYSCAN)* und kann 20 verschiedene Bar- und QR-Codes (1D/2D/QR) einlesen. Das Modul besitzt zwei LEDs, die den Code beleuchten und eine rote „Ziellinie" projizieren. Durch die Kameratechnik ist das Modul auch im Außenbereich bei hellem Sonnenlicht einsetzbar. Wir fanden im Test kein Code, der nicht erkannt wurde, auch wenn bei schlecht gedruckten Codes ab und zu mehrere Versuche nötig waren.

Da das Scanner-Modul per serieller Schnittstelle (UART) kommuniziert, sollte es kaum eine Rolle spielen, mit welcher Programmiersprache man seine Anwendung erstellt. Das Scanner-Modul wird übrigens durch Scannen von Barcodes konfiguriert, der Micro-USB-Anschluss auf den Boards kann so als virtueller serieller Port benutzt werden oder als USB-Keyboard.

Möchte man nur mal ein paar Barcodes scannen oder etwas mit dieser Technik experimentieren, ist wohl ein vorhandenes Smartphone die bessere und günstigere Wahl. Sollen aber eigene mobile Anwendungen oder ortsfeste Systeme für die Zugangskontrolle oder Inventarisierung verwirklicht werden, so bietet das Modul einen niederschwelligen Einstieg mit hohem Nutzwert. —*caw*



Hersteller	SB Components Ltd.
URL	make-magazin.de/xga4
Preis	61 € inkl. Steuern ohne Versand

Bangle.js 2

Ausprobiert —von Make:—

Smartwatch für selbst programmierte Apps

Gordon Williams hat schon 2019 mit seiner *Bangle.js* eine erste preisgünstige Smartwatch angeboten, die sich besonders einfach programmieren ließ. Jetzt ist der Nachfolger *Bangle.js 2* erhältlich, mit deutlichen Verbesserungen: In ihr steckt ein *NordicnRF52840*-SoC (System-on-Chip). Der beinhaltet 256KB RAM und eine *ARM-Cortex-M4*-CPU, die mit 64MHz taktet. Für Programme stehen insgesamt 9MB Flash-Speicher zur Verfügung, von denen 1MB auf dem Chip liegen und 8MB extern angebunden werden.

Die Bluetooth-5.2-Einheit implementiert sowohl *Bluetooth Low Energy* (BLE) als auch *Bluetooth Mesh*. Unterstützt wird das Nordic-SoC von einer Reihe von Komponenten, wie sie größtenteils auch schon in der ersten Version der Bangle.js zu finden waren. Den aktuellen Luftdruck und die Temperatur misst zum Beispiel ein *BMP280* von Bosch. Die aktuelle Position berechnet der GPS/Glonass-Empfänger *AT6558* und den Puls des Trägers ermittelt ein *Vcare VC31*.

Schließlich enthält die Bangle.js 2 noch den 3-Achsen-Beschleunigungssensor *Kionix KX023* und ein 3-Achsen-Magnetometer. Zur Kommunikation mit der Außenwelt dient in erster Linie ein LC-Touch-Display (LPM013-M126) mit einer Bildschirm-Diagonale von 1,3 Zoll und 176 × 176 Pixeln und einer Farbtiefe von drei Bit. Eine dimmbare Hintergrundbeleuchtung verhilft auch bei schlechten Lichtverhältnissen zu einem klaren Bild.

Subtilere Ausgaben erzeugen ein Vibrationsmotor und ein Piezo-Pieper. Der 200mAh-Akku wird per USB aufgeladen und hält im Stand-By für bis zu vier Wochen. Das gilt natürlich nur, wenn Stromfresser wie GPS und der Bildschirm nicht permanent eingeschaltet sind. Ein besonderes Schmankerl für Hardware-Hacker ist der SWD-Port (*Full Serial Wire Debug*) auf der Rückseite der Uhr. Damit können Entwicklerinnen und Entwickler Programme auch in anderen Sprachen als JavaScript entwickeln und ohne Zusatzhardware aufs Gerät transferieren.

Rein äußerlich unterscheidet sich die Bangle.js 2 übrigens deutlich von der Bangle.js. Sie ist merklich kleiner (36mm × 43mm × 12mm) und in den Farben Schwarz, Blau und Pink erhältlich. Wasserdicht ist sie weiterhin, aber nur noch für maximal 30 Minuten in einer Tiefe von bis zu einem Meter.

Für die meisten Menschen dürfte sich die Uhr damit besser für den alltäglichen Gebrauch eignen als der Erstling. Die Hardware kann aber mehr als nur schick die Zeit anzeigen und schreit geradezu nach ein paar praktischen Apps. Die programmiert man für den JavaScript-Interpreter *Espruino* in der passenden IDE, die im Browser auf dem Rechner läuft. Über den Browser installiert man die fertigen Apps dann drahtlos auf der Uhr. Wie das im Detail funktioniert, lesen Sie online (siehe Link). —*Maik Schmidt/pek*

espruino.com

▶ make-magazin.de/xga4

Hersteller	Gordon Williams /Espruino.com
Bezugsquelle	z.B. BerryBase.de
Preis	89,90 €

Artists and Hackers

A Podcast on art, code and community

Sie erfinden ganz neue Programmiersprachen, bauen Chatbots oder arbeiten mit dem „Offline-Internet" aus Kuba: Im Podcast „Artists and Hackers" geht es um Leute, die an der Schnittstelle von Kunst und Technik arbeiten. Sie berichten von ihren Projekten, die manchmal recht künstlerisch und dann doch äußerst praktisch orientiert sind.

Gerade ist die erste Staffel zu Ende gegangen, die einen Schwerpunkt auf das Thema Chatbots legt und ihm drei der elf Folgen widmet. Während etwa die Bots von *QueerAI* und *Bina48* mit queerer Literatur bzw. afroamerikanischer Geschichte trainiert werden, um diese zu bewahren, gibt es mit dem „Trollbot" *Faith* einen Gegenentwurf. Einen Einblick in die Geschichte der Hackerkultur, genauer gesagt, die Entwicklung der ersten Mailbox 1973, gibt es ebenfalls. Der Entwickler von *Community Memory*, Lee Felsenstein, berichtet wie er Technologie weit vor allen Open-Source-Manifesten für alle zugänglich machen wollte.

Mit 10 bis 30 Minuten Länge sind die Folgen schnell durchgehört und bieten einen wirklich breiten Überblick über ganz verschiedene Kunstprojekte. Wer trotzdem keine Zeit hat oder lieber lesen möchte, findet zu allen Episoden ein Transkript auf der Webseite. Die zweite Staffel ist bereits angekündigt. —*hch*

URL	www.artistsandhackers.org

Arduino & Co – Messen, Schalten und Tüfteln

Pfiffige Lösungen mit Arduino Pro Mini und ATmega328-Boards

Das Buch führt in die Programmierung von Mikrocontroller-Boards (mit ATmega328 und 168) mittels der Arduino-IDE ein. Es erklärt die Syntax der Arduino-spezifischen C/C++-Variante, behandelt das Auswerten einer Reihe verbreiteter Sensoren und die Ansteuerung von Motoren und Servos, alles jeweils mit Beispielcode. Es liefert zudem diverse Tipps, die von einschlägigen Einkaufsquellen im Internet (nebst Warnungen vor Fallen bei der Bestellung in Fernost) über den Inhalt eines praktischen Handsortiments an Standardbauteilen für Elektronikbasteleien bis hin zum Betrieb von Schrittmotoren mit Li-Ion-Akkus reichen.

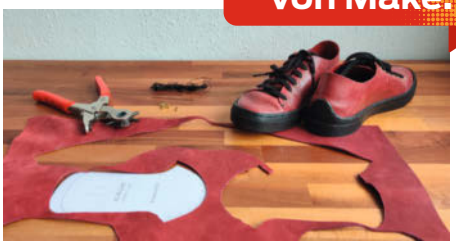
Aber auch, wenn das jetzt erst mal anders klingt: Dieses Buch eignet sich *nicht generell* für Einsteiger. Genauer: Es eignet sich nur für *sehr spezielle* Einsteiger, nämlich solche, die schon einiges mit Elektronik gebastelt haben, aber Nachholbedarf bei Mikrocontrollern und deren Programmierung haben. Und die manche Vorlieben mit dem Autor teilen, der etwa konsequent nicht mit Breadboards arbeitet, sondern lieber lötet (und sei es auf Reißnägeln als Kontaktpunkte), der einen fest verbauten USB-Anschluss auf Board kategorisch überflüssig findet, weil der nur einmal benutzt würde, um den Code aufzuspielen – weshalb das erste Projekt im Buch darin besteht, sich selbst einen Programmieradapter zu löten. Und der im Code seiner *Lithium-Ionen-Akku Test- & Ladestation* den `loop()` komplett mit knapp 500 Zeilen geschachtelter `if-else`-Abfragen füllt, ohne einen einzigen Funktionsaufruf. Kompliziert geht immer. Wer auf anderer Wellenlänge unterwegs ist, tut sich beim Einstieg in die Arduino-Programmierung mit anderen Büchern oder auch unseren Sonderheften zum Thema deutlich leichter. —*pek*

Autor	Robert Sontheimer
Verlag	Elektor
Umfang	331 Seiten
ISBN	978-3-89576-477-6
Preis	34,80 €

SneakerKit

Ausprobiert — von Make:—

Nachhaltige Schuhe zum Selbernähen



Das DIY-Set ist eine nachhaltige (weil reparierbare) Option für selbstgebaute Schnürschuhe ohne Kleber. Die einzelnen Bestandteile können selbst zugeschnitten und zusammengenäht werden. Zur Individualisierung stehen 25 verschiedenfarbige Leder zur Auswahl sowie neun unterschiedlich hohe Schnittmuster und sommerliche Alternativen.

Das *FULL SneakerKit* beinhaltet eine Anleitung, Sohlen und -Einlagen, Leder, eine Loch- und Ösenzange, Ösen, Nadel und gewachsten Faden. Zusätzlich sollte man einen scharfen Cutter, eine Ahle oder Reißzwecke, Malerkrepp und eine Schneideunterlage bereitlegen. Falls man bereits Material zu Hause hat oder seinen Sneaker reparieren möchte, können einzelne Teile bestellt werden. Das Schnittmuster besteht nur aus zwei Teilen. Damit man seine eigenen Ideen umsetzen kann, steht ein Template als PDF oder DXF-Datei zur Verfügung. Dann kann das Zuschneiden in Maker-Manier von einem Laser erledigt werden. Die beiliegende Anleitung ist zwar auf Englisch, kann aber auch als Video nachvollzogen werden. Zum gemeinsamen Basteln werden auf der ganzen Welt Workshops angeboten. Die Schuhgrößen reichen von 36 bis 47. Ein Kit für Kleinkinder ist ebenfalls erhältlich. Einen ausführlichen Testbericht mit Video gibt es online (siehe Link). —*stri*

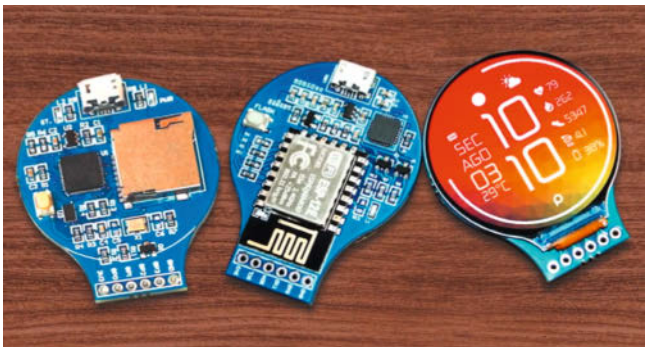
▶ **make-magazin.de/xga4**

Hersteller	SneakerKit
URL	sneakerkit.eu
Preis	89 €

RoundyPi und RoundyFi

Ausprobiert — von Make:

Mikrocontroller-Boards mit rundem Display



Om Singh / Kickstarter.com

Von vorne sehen die beiden Boards praktisch identisch aus: Oben das kreisrunde Display, unten auf einer Platinenzunge eine Reihe aus sechs Lötaugen für die beiliegende Pinleiste. Auf der Rückseite zeigen sich deutlich die Unterschiede: Auf dem *RoundyFi* arbeitet ein ESP-12E und die WLAN-Antenne springt ins Auge. Auf dem *RoundyPi* sitzt der Raspberry-Pico-Chip *RP2040*, der sich aber neben dem Micro-SD-Karten-Slot fast unscheinbar ausnimmt.

Die Displays werden – wie viele andere runde Bildschirmchen – durch den *GC9A01*-Treiber angesteuert. Sie haben einen Durchmesser von 35mm (ringsherum bleibt allerdings stets ein Rand von gut einem Millimeter schwarz) bei einer Auflösung von 240 × 240 Pixeln und einer Farbtiefe von 65K. Je nach Anwendung und Betrachtungsabstand sind damit einzelne Pixel deutlich erkennbar. Die Darstellung ist kräftig und kontrastreich, wenn man direkt senkrecht darauf schaut. Schräg betrachtet fällt die Helligkeit aber schon deutlich ab.

Die sechs Pins beziehungsweise Lötaugen führen vier GPIO-Anschlüsse nach draußen, die übrigen beiden sind für die Stromversorgung und Masse da. Die Betriebsspannung liegt bei 3,3V. Alternativ betreibt man die Boards über die Micro-USB-Buchse auf der gegenüberliegenden Seite der Pinleiste, also sozusagen am Kopfende des Displays. Über USB gelangt auch der Code auf die Boards, wie gewohnt.

Bis Mitte März konnte man sich noch auf *Kickstarter* am Crowdfunding für die beiden Boards beteiligen, inzwischen kann man sie bereits regulär kaufen – und sie sind gar nicht mal viel teuer als in der Kampagne. Einen ausführlichen Testbericht mit Video lesen Sie online (siehe Link). —*pek*

▶ make-magazin.de/xga4

Anbieter	SB Components Ltd.
URL	shop.sb-components.co.uk/products/roundy
Preis	36,95 € (RoundyPi), 42,95 € (RoundyFi)

Anycubic Photon M3 Max

Ausprobiert — von Make:

Resin-Drucker mit Cloud-Anschluss und Harzpumpe

Der chinesische Hersteller *Anycubic* hat mit den Modellen *Photon M3* eine neue Serie von drei Resin-3D-Druckern auf den Markt gebracht, die sich vor allem in der Display-Auflösung und in der maximalen Werkstückgröße unterscheiden. Vom mittleren Modell *Photon M3 Plus* schickte der Hersteller uns vorab ein Testgerät. Dessen 6K-Display belichtet Objekte bis zu einer Grundfläche von 19,7cm × 12,2cm und 24,5cm Höhe. Das geht dennoch fix: Je nach Schichtdicke türmt das Gerät in Z-Richtung bis zu 10cm Materialschichten pro Stunde auf; für die Belichtung einer 0,05mm dünnen Schicht des Standardharzes des Herstellers reichen 1,5 Sekunden. Das monchrome 9,25"-LCD verfügt als Lichtquelle über eine Matrix aus 40 parallel leuchtenden LEDs, die so eingebaut sein sollen, dass nur wenig Energie verloren geht – eine Technik, die die Firma *Anycubic LighTurbo* nennt und auch in die anderen Geräte der Serie eingebaut hat.

Auf der Rückseite des Druckers lässt sich eine Resinflasche einhängen, in die statt des Deckels eine Förderpumpe eingeschraubt wird. Registriert der simple Sensor im Becken einen kritisch niedigen Harzstand, sorgt die Pumpe für Nachschub: Sie presst Luft in die Flasche, sodass Harz durch einen Schlauch ins Becken gedrückt wird. Falls zwischen Pumpendeckel und Flasche Luft entweicht, muss man allerdings mit einem zusätzlichen Dichtungsring arbeiten. Der Vorteil der Pumpe: Auch bei wirklich großen Drucken geht das Material nicht auf halbem Weg aus und auch der Füllstand der Reserveflasche wird per Sensor verfolgt.

Die Fläche des Drucktischs zeigt ein gelasertes Quadratmuster, was für gute Haftung des Werkstücks sorgt – fast zu gut, denn die fertigen Drucke lassen sich zum Teil nur schwer wieder davon lösen (wir experimentieren noch mit einer Reduktion der Belichtungszeit bei den unteren Lagen). Eine gute Haftung ist allerdings bei diesem Drucker wichtig, weil er zum Lösen der jeweils frischen Schicht vom Boden des Harzbeckens einfach den Tisch in Richtung der Z-Achse zieht und dabei die ganze Werkstückfläche auf einmal belastet; das bei anderen Druckern übliche Schwenken des Beckens ist hier nicht vorgesehen.

Die zuvor auf dem Rechner geslicten Druckdaten kommen entweder klassisch per USB-Stick auf den Drucker oder über WLAN bzw. Ethernet-Kabel. Als Austauschplattform hat Anycubic jetzt eine eigene Cloud gestartet, in die man sich sowohl auf dem Slicer auf dem Desktop als auch auf dem Drucker selbst einloggt. Zusätzlich gibt es eine Android-App, über die sich der Druckfortschritt auch unterwegs online verfolgen lässt und von der man auch zuvor vorbereitete Druckjobs starten kann. Eine optionale Kamera soll später zusätzlich den Kontrollblick erlauben, ob das Werkstück noch seiner Vorlage ähnelt oder ob es sich lohnt, zur Harzersparnis den Druck abzubrechen. Einen ausführlichen Testbericht mit mehr Bildern lesen Sie online. —*pek*

▶ make-magazin.de/xga4

Hersteller	Anycubic
URL	www.anycubic.com
Preis	699 US-$

IMPRESSUM

Make: Nächste Ausgabe erscheint am 2. Juni 2022

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), Ákos Fodor (akf), Helga Hansen (hch), Carsten Meyer (cm), Carsten Wartmann (caw)

Mitarbeiter dieser Ausgabe: Beetlebum (Comic), Achim Bertram, Andreas Känner, Matthias Mett, Alexander Moser, Josef Müller, Kerstin Ogrissek, Stella Risch (stri), Uwe Rohne, Michael Scheuer, Ulrich Schmerold, Florian Sommer, Ralf Stoffels, David Traum, Tobias Vogel

Assistenz: Susanne Cölle (suc), Christopher Tränkmann (cht), Martin Triadan (mat)

Leiterin Produktion: Tine Kreye

DTP-Produktion: Martina Bruns, Martin Kreft (Korrektorat)

Art Direction: Martina Bruns (Junior Art Director)

Layout-Konzept: Martina Bruns

Layout: Nicole Wesche

Fotografie und Titelbild: Andreas Wodrich

Digitale Produktion: Kevin Harte, Thomas Kaltschmidt

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: Michael Hanke (-167)
(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: 12,90 €; Österreich 14,20 €; Schweiz 25.80 CHF; Benelux 15,20 €

Abonnement-Preise: Das Jahresabo (7 Ausgaben) kostet inkl. Versandkosten: Inland 77,00 €; Österreich 84,70 €; Schweiz/Europa: 90,65 €; restl. Ausland 95,20 €

Das Make-Plus-Abonnement (inkl. Zugriff auf die App, Heise Magazine sowie das Make-Artikel-Archiv) kostet pro Jahr 6,30 € Aufpreis.

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

Was wünscht Du dem Raspberry Pi zu seinem 10. Geburtstag?

ESP32

China, Allround-Talent in flinken und stromsparenden WLAN-Projekten

Ich wünsche ihm eine bessere Kühlung. Nicht nur, dass die 4-CPU-Kerne sich ganz schön aufheizen, auch der USB- und Netzwerkchip werden ordentlich heiß.

Arduino UNO

Italien, das Urgestein der Maker-Boards. Langsam und veraltet, aber robust und beliebt

SD-Karten als Massenspeicher sind eine Frickellösung. Der soll sich endlich mal integrierten eMMC-Flash anschaffen. Oder zumindest einen Slot dafür anbauen.

Nano-Axe

Hannover, der anspruchslose Minimalist für die BASIC-Anhänger unter den Makern

Ich wünsche dem Raspberry eine weiterhin wachsende Community und noch mehr Softwareportierungen von Intel-Plattformen auf ARM.

BBC micro:bit

London, dank der eingebauten Sensoren und LEDs in Schulen sofort startklar

Der Pi hat so viele Funktionen, aber keinen einzigen A/D- oder D/A-Wandler dabei. Wenn die Foundation das nachrüsten würde, könnte man noch spannendere Experimente machen.

Inserentenverzeichnis

Binker Materialschutz GmbH, Lauf 29
dpunkt.verlag GmbH, Heidelberg 55
Reichelt Elektronik GmbH & Co., Sande 2
Rheinwerk Verlag GmbH, Bonn 7
Make:markt 41

Make:

2× Make testen und über 9 € sparen!

Ihre Vorteile:

- ✓ **GRATIS dazu:** Make: Tasse
- ✓ Zugriff auf Online-Artikel-Archiv*
- ✓ Jetzt auch im Browser lesen!
- ✓ Zusätzlich digital über iOS oder Android lesen

Für nur 16,10 € statt 25,80 €

* Für die Laufzeit des Angebotes.

Jetzt bestellen: make-magazin.de/miniabo

leserservice@make-magazin.de +49 541 80 009 125 make-magazin.de/miniabo