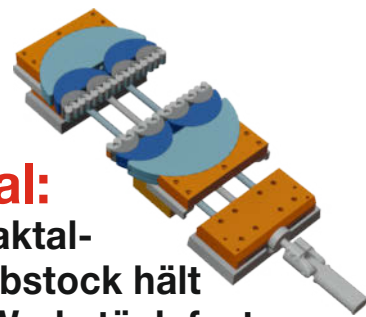




Genial:
DIY-Fraktal-
schraubstock hält
jedes Werkstück fest

Schneller 3D-Drucken

- 3 bis 5 mal schneller bei gleicher Qualität
- Highspeed-Drucker: AnkerMake M5 im Test
- So beschleunigen Sie Ihren eigenen Drucker



Smart Home

- Stylus-Timer für Plattenspieler
- E-Ink-Display fürs Smart Home
- Sauerteig aus dem IoT-Einmachglas

Know-how

- Lego-Roboter mit Python steuern
- Arduino im Browser simulieren
- Übersicht: Software Defined Radio

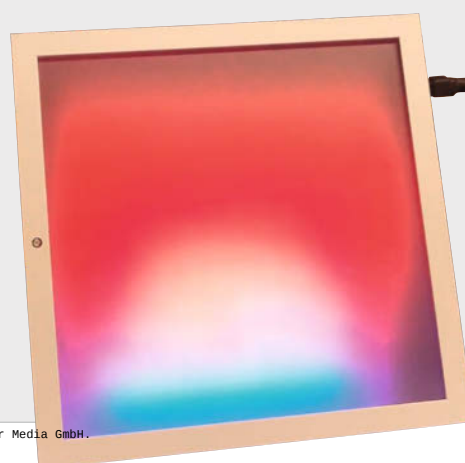
Smarte Gasuhr

- Gaszähler per Sensor auslesen
- ESP8266 sendet Daten per MQTT
- Verbrauch per App kontrollieren



Lava-Lampe

- LED-Streifen im IKEA-Rahmen
- Dekorativ und meditativ
- Programmierbare Effekte



3/23
1.6.2023
CH CHF 26.50
AT 14,90
Benelux 15,90
€ 13,50



Deutschlands größtes Maker-Treffen

Hannover

Maker Faire®

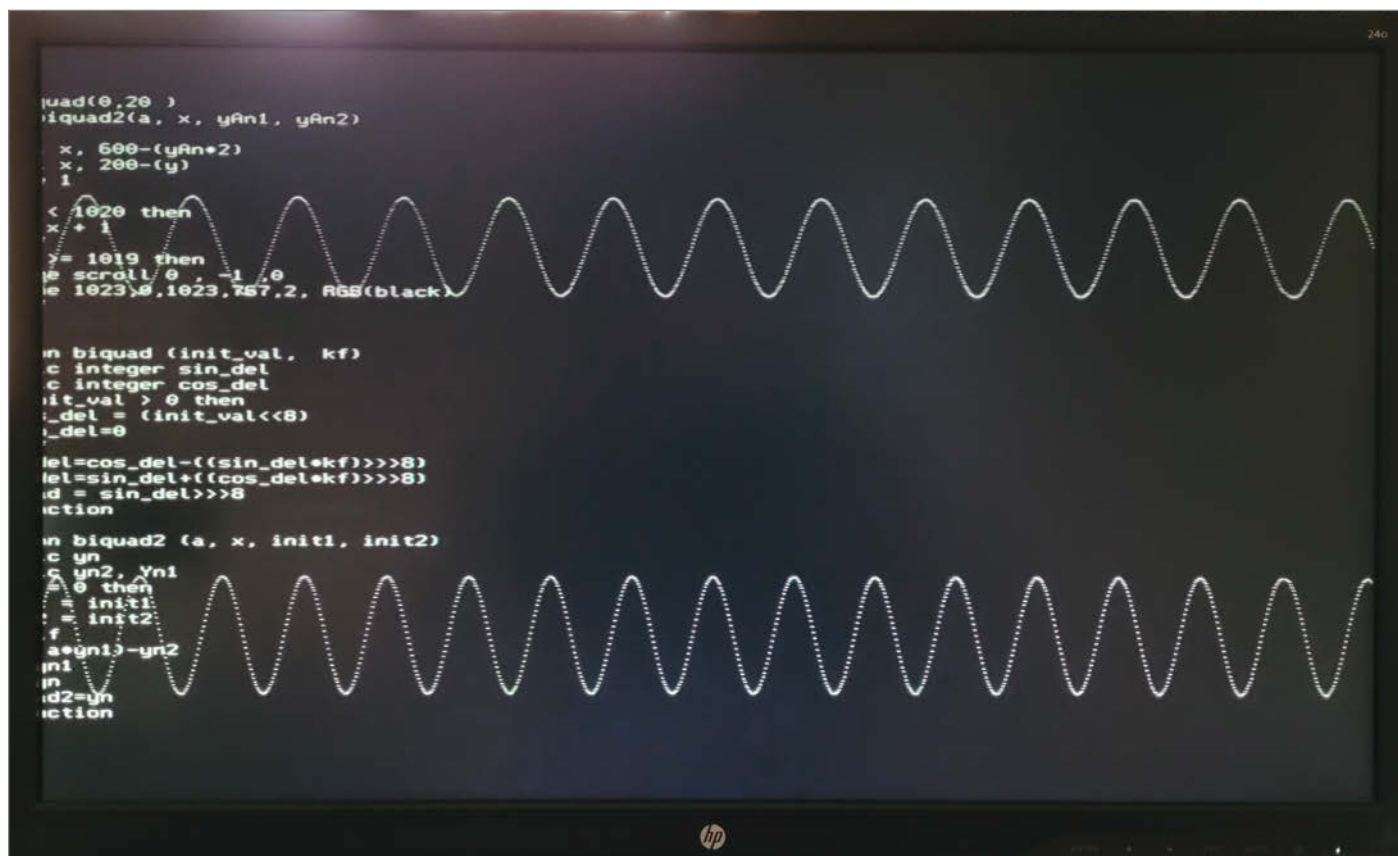
LAST CALL!

Melde dich mit
deinem Projekt
bis zum 11. Juni
kostenfrei als
Aussteller an.

19.–20. Aug.

Hannover Congress Centrum

maker-faire.de



Keep it simple and stupid

Neulich habe ich ein Projekt spaßeshalber auf dem BASIC-Computer Color Maximite 2 (CMM2, vorgestellt in Make 5/20) programmiert (auf S. 113 haben wir mit dem Agon Light einen ähnlichen Ansatz getestet). Ich wollte mit einem Stromsensor INA219 die Stromaufnahme eines Odroid XU4 vom Booten bis zum Betrieb protokollieren und auch gleich grafisch darstellen, um für den Betrieb mit einem Akkupack den richtigen Step-Down-Wandler auszuwählen. PC-Unterstützung ist im CMM2 gleich integriert und dank des PIXEL-Befehls war der Stromverlauf schnell geplottet.

Ich war verblüfft, wie schnell und einfach ich zum Ziel kam. Auf Grundlage von Carsten Meyers Artikel „Digital filtern und oszillieren“ aus der Make 6/22 habe ich gleich noch mehrere Arten von Tiefpassfiltern durchprobiert und die resultierenden Messkurven parallel dargestellt. Und wo ich schon dabei war, habe ich noch mit digitalen Oszillatoren rumexperimentiert. Was soll ich sagen, ich hatte schon sehr lange nicht mehr soviel kreativen Spaß beim Programmieren – ohne Compiler und Linker, ohne Fehlermeldungen wegen fehlender Module oder nachzuinstallierender Crates.

Trotz allen Spaßes hatte ich einen negativen Gedanken im Kopf: Hauptsache, das sieht jetzt keiner, dass ich in BASIC programmiere! Warum? Weil BASIC unter vielen Programmierern verpönt ist. Es gilt als niedere Sprache für blutige Anfänger, lässt unstrukturierten Spaghetti-Code zu, kann keine Rekursion (auch wenn das eigentlich nur Schulbuchglück ist) und so weiter. Ich kann zwar C und Pointer, kenne viele Sprachkonstrukte in C++ bis Version 11 und nutze in

Python Iteratoren, Lambda-Funktionen sowie Dekoratoren und lerne gerade Rust.

Aber wie sagt man so schön: Richtig angekommen bin ich in diesen Sprachen nie. Ich bekenne öffentlich: Ich liebe BASIC, am besten in Kombination mit Assembler. Ich weiß, dass es vielen Lesern als Hobby-programmieren ähnlich geht: Die Sehnsucht nach Einfachheit einer Sprache, um sich auf das eigentliche Problem konzentrieren zu können, statt erst komplizierte Konstrukte einer Sprachen bändigen zu müssen.

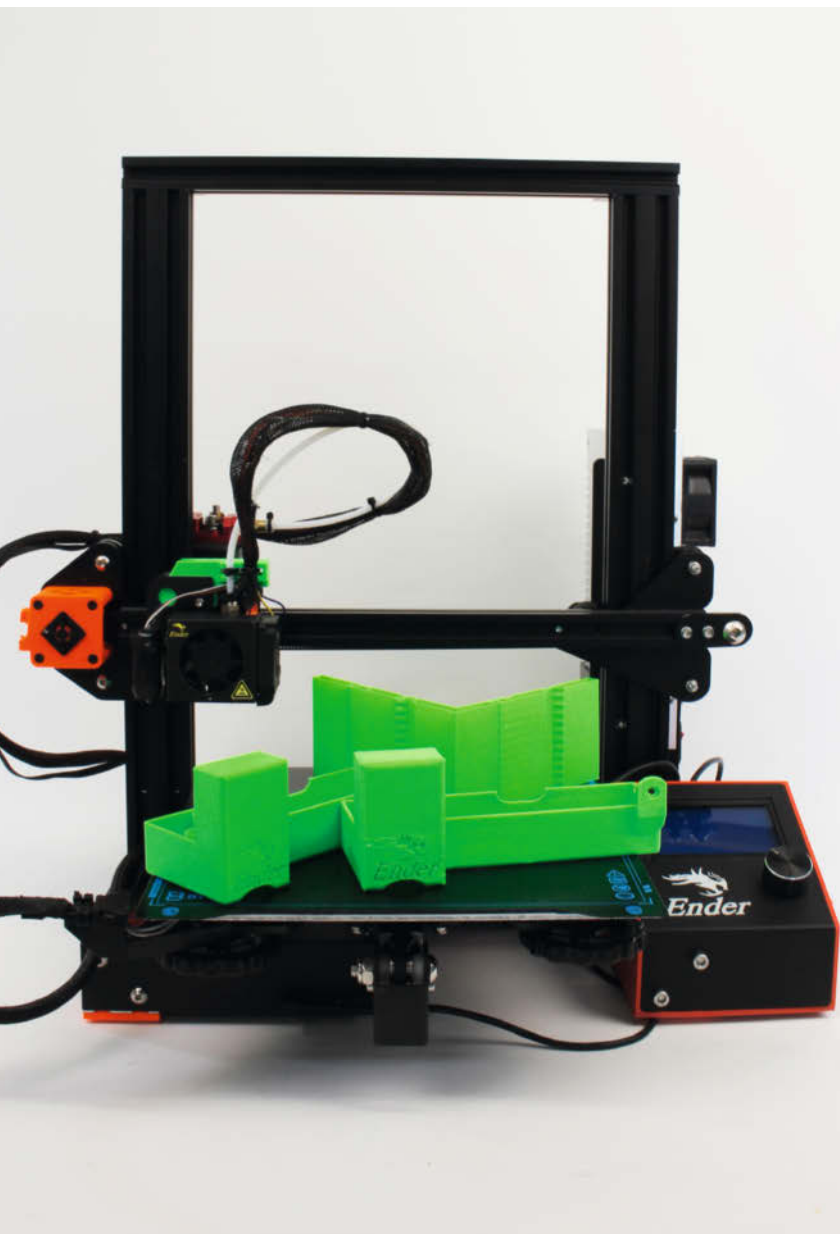
In welcher Sprache programmieren Sie am liebsten? Welche würden Sie gerne hier in der Make erklärt bekommen? Oder wünschen Sie sich Best Practices zu C oder Python? Schreiben Sie uns – abschätzige Blicke wegen Ihrer Lieblingssprache brauchen Sie von uns nicht zu befürchten. Apropos öffentliche Geständnisse: Wussten Sie, dass das Wort „Professor“ aus dem Lateinischen „pro fateri“ für „sich bekennen“ abgeleitet ist?

Happy Hacking!

► make-magazin.de/x3gj

Daniel Bachfeld
Daniel Bachfeld

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



Schneller 3D-drucken

Zehn oder zwanzig Stunden warten auf einen 3D-Druck? Das kann bald vorbei sein: Der neue AnkerMake M5 ist mehrfach schneller als die bislang im Maker-Bereich verbreiteten Drucker – wir haben uns im Test davon überzeugt. Falls Sie lieber Ihrem vorhandenen 3D-Drucker Beine machen wollen, zeigen wir, wie man die Firmware Klipper darauf installiert und mit ein paar Probedrucken nicht nur deutlich mehr Tempo, sondern auch bessere Druckqualität aus Ihrem Schätzchen herausholt.

- 8** Highspeed-Drucker: AnkerMake M5 im Test
- 10** So beschleunigen Sie Ihren eigenen Drucker mit Klipper

Inhalt

Smarthome

Vinyl-Fans aufgepasst: Der Stundenzähler (vornehm: Stylus-Timer) verhindert Schäden an Ihren Musikschätzen durch uralte Plattenspieler-Nadeln. Fürs leibliches Wohl sorgen der Sauerteig aus der IoT-Brutanlage und das E-Ink-Klima-Display fürs Smarthome, das Sie über Temperaturen und Luftfeuchtigkeiten stromsparend informiert.

- 42** Sauerteig aus dem IoT-Einmachglas
- 48** E-Ink-Display fürs Smarthome
- 66** Stundenzähler für Plattenspielernadeln



- 3** Editorial
- 6** Leserforum, Comic
- 8** **Test: 3D-Drucker AnkerMake M5**
- 10** **Werkstatt: 3D-Drucker-Tuning, Teil 3: Schneller drucken mit Klipper**
- 18** Maker Faire: Jetzt noch für Hannover anmelden
- 20** **Projekt: Smarter Gaszähler mit Smarthome-Integration**
- 27** In eigener Sache: Exklusive Online-Artikel für Make-Abonnenten
- 28** Test: Agon Light – der schnellste 8-Bit-Computer der Welt
- 36** **Projekt: Der LavaFrame**
- 42** **Projekt: Sauerteig aus dem IoT-Einmachglas**
- 48** **Projekt: E-Ink-Display fürs Smarthome**
- 54** Community-Projekt: THE 100: Der schnellste 3D-gedruckte Drucker der Welt
- 56** Community-Projekt: Ikea-Hack: Hochbett zu Hochbeet
- 58** Projekt: Der X-Wing zum Anbeißen – ein Geburtstagstraum für Star-Wars-Fans

Know-how

Völlig losgelöst von Tablet und Cloud werden Lego-Roboter, wenn PyBricks ihnen Python bringt. Steht Ihnen der Sinn eher nach Arduino- oder ESP-Projekten, simulieren und testen Sie die mit Wokwi vor dem Zusammenbau. Außerdem zeigen wir die Grundlagen für Radio-Empfang per USB-Dongle.

- 72** Übersicht: Software Defined Radio
- 78** Pybricks: Python steuert Lego-Roboter
- 88** Wokwi: Arduino im Browser simulieren



Smarter Gaszähler

Die Angst vor der nächsten Gasrechnung treibt viele täglich zum Ablesen der Gasuhr in den Keller. Unser smarter Gaszähler erspart Ihnen das Treppensteigen. Er zeigt Ihnen die Verbrauchsdaten jederzeit in einer App an oder übermittelt die Werte per MQTT an Ihr Smart-home.

- 20** Smarter Gaszähler mit Smarthome-Integration



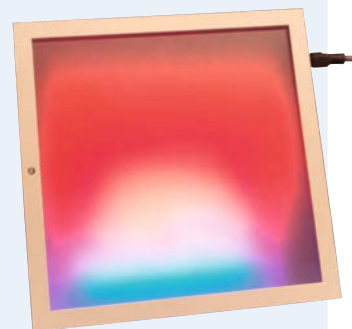
- 60** Projekt: KI für die Modelleisenbahn, Teil 1
- 66** Projekt: Stylus-Timer für Plattenspiellernadeln
- 72** Report: Software Defined Radio (SDR) – ein Überblick
- 78** Workshop: Pybricks: Python steuert Lego-Roboter
- 84** Tipps & Tricks
- 88** Workshop: Mikrocontroller-Projekte simulieren mit Wokwi
- 94** Workshop: Adafruit GFX Library, Teil 2: Bitmaps erstellen und animieren
- 100** Reingeschaut: Glasfaseranschluss
- 102** Werkstatt: Fraktaler Schraubstock aus dem 3D-Drucker
- 108** Test: Hightech-Messgerät für große Flächen
- 110** Kurzvorstellungen: Mikrocontroller-Boards, Mini-USB-Tastatur, Balkonkraftwerk-Buch, E-Ink-Display mit NFC, 3D-Drucker
- 114** Impressum/Nachgefragt

Themen von der Titelseite sind rot gesetzt.

Lavalampe

Hier wird's psychedelisch: Der LavaFrame lässt das in den 60er Jahren des vorigen Jahrhunderts wegen seiner Wirkung auf Körper und Geist so heiß begehrte Multicolor-Beleuchtungsgerät wieder auferstehen. Allerdings ist es jetzt eckig und kostet nur einen Bruchteil des Geldes vom Mathmos-Original, weil es aus einem IKEA-Bilderrahmen selbstgebaut ist.

- 36** LavaFrame



Leserforum

Wirklich neu?

Mit dem Make-Abo Artikel hinter der Paywall bei heise online lesen

Ich bin Make-Abonnent und freue mich natürlich auf jede neue Ausgabe. Bei der Make 2/23 musste ich dann sehr genau hinsehen, ist die wirklich neu? Doch dann dämmert es mir, die Aufmacher-Projekte auf der Titelseite habe ich schon in heise+ gesehen, wohl-gemerkt vor dem Erscheinungstermin der aktuelle Make. Muss das wirklich sein? Ich begrüße es ausdrücklich, dass die Magazin-Artikel früher oder später auch auf heise+ zu lesen sind, aber doch nicht vor dem Erscheinungstermin des Heftes!

Oliver Bayer

Sie haben recht: Wir veröffentlichen viele Artikel aus dem kommenden Heft vorab online hinter der Paywall, sobald sie fertig sind. Wir verstehen das als Angebot und Service für unsere Abonnenten, die zu diesen Artikeln ohne Aufpreis Zugang haben (siehe auch Seite 27). Aber vielleicht ist unsere Perspektive und damit die Sichtweise ja auch verzerrt. Insofern würden wir gerne die Frage insgesamt in die Runde aller stellen, die die Make lesen: Stört oder freut es Sie, dass Sie als Abonnent so viele Artikel vorab lesen können? Schreiben Sie uns Ihre Meinung an mail@make-magazin.de.

Frischer Code

KI auf der Schreibmaschine, Make 2/23, S. 28

Aus Versehen war der Code für die Ansteuerung der erika 3004 im GitHub-Repository zum Zeitpunkt des Erscheinens des Hefts fehlerhaft – das tut uns leid und er wurde mittlerweile durch eine verbesserte Version ersetzt. Offen gestanden hat es die Redaktion allerdings überrascht, wie viele Make-Leser dieses Projekt nicht nur als Anregung genommen haben, über die API von ChatGPT eigene Projekte künstlich intelligenten Text ausgeben zu lassen, sondern tatsächlich genau so eine selbsttippende Schreibmaschine nachbauen wollten. Dafür ein großes Kompliment in die Runde der Nachbauer!

Wer weitere Fragen zum Projekt hat, schließt sich am besten über den Issue-Tracker im GitHub-Repository mit den anderen kurz, die ihrer Erika oder auch einer ähnlichen Schreibmaschine mit I/O-Schnittstelle das KI-Schreiben beibringen wollen oder das schon geschafft haben:

► make-magazin.de/xbbd

Fasse Dich kurz!

Flusspegel messen mit der Citizen Science Box, Make 1/23, S. 54

Das LoRaWAN-Pegelmesser-Citizen-Science-Projekt hat mich voll begeistert. Welch ein

tolles und umfassendes Projekt, in dem MINT, Community, Energie-Effizienz der Hardware und das Thema Umwelt stimmig zusammen kommen! Lediglich die TTN-Payload erscheint mir mit 24 Bytes recht üppig gewählt. Braucht man wirklich Vcc, Gehäuse-Temperatur, Feuchtigkeit, cycle und on-time jeweils in 24 Bit Genauigkeit? Meines Erachtens würde dafür jeweils ein Byte (0...255) reichen. Das LoRaWAN-Netz ist darauf angewiesen, dass die Payload möglichst klein gehalten wird. Daher wird in der TTN public community die uplink airtime auf 30 Sekunden pro 24 Stunden begrenzt. „Fasse Dich kurz“ bei der Payload kommt letztlich auch dem Energiebedarf des LoRaWAN-Moduls zugute.

Hans-Martin Hilbig

Die im Beispiel übertragenen Daten waren eher als Blaupause gedacht, die an die eigenen Bedürfnisse angepasst werden können (und sollen!). Je nach Anwendungen sind auch unterschiedliche Strategien denkbar und sinnvoll, etwa nur dann den Pegel zu senden, wenn ein bestimmter Schwellwert überschritten wurde und ansonsten nur einmal pro Tag ein Lebenszeichen zu schicken. In unserem Szenario hat sich eine Meldung alle 15 Minuten für den Start als gute Frequenz herausgestellt, hier lernt man einiges über die Dynamik des Gewässers – einige zeigen durch „Schwallbetrieb“ oder durch regionale, saisonale Gegebenheiten unterschiedliche Dynamiken.

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.

Showstopper Big Sur

Eigenbau-Tracker für Apples Find-My-Netzwerk, Make 1/23, S. 48

Mit Begeisterung habe ich mich auf Ihren Artikel gestürzt und OpenHaystack installiert. Leider bin ich gescheitert: Trotz Deaktivierung des Gatekeepers hat die Aktivierung nicht funktioniert. In der Beschreibung wird Big Sur erwartet. Ich habe ein MacBook Pro M1 mit dem OS Ventura 13.0.xx. Können Sie bestätigen, dass das Programm nur und ausschließlich auf Big Sur läuft? Falls ja, wird es ein Update geben?

Matthias Spinnler

In der Tat läuft OpenHaystack nur unter der älteren Version von macOS (laut Github soll es auch unter macOS 12 laufen). Da die Software

als Proof-of-Concept an einer Uni entstanden ist, können wir leider nicht sagen, ob es irgendwann ein Update geben wird.

Blinder Fleck bei (Free) CAD

Technische Zeichnungen mit FreeCAD, Teil 3, Make 2/23, S. 106

Ich nutze hin und wieder FreeCAD – zwischenzeitlich hatte ich auch ein wenig mit AutoCAD Fusion 360 gemacht – und ich habe noch einen Hinweis für technische Zeichnungen, die mir schon einige Probleme bereitet haben. Eine Modell wird für eine technische Zeichnung im ersten und einfachen Schritt auf eine

der drei Ebenen XY, XZ oder YZ projiziert. Alle Winkel und Maße gelten für diese Projektion. Bei einfachen geometrischen Formen ist das in der Regel kein Problem, aber schon bei einer einfachen Pyramide stimmen dann die Maße und Winkel nicht mehr. Möchte man eine Pyramide aus einer quadratischen Bodenplatte und vier identischen Seitenteile z.B. aus Holz bauen, klappt das auf diesem Weg nicht (auch die Gehrungen werden falsch angegeben).

Eine saubere Lösung habe ich leider nicht gefunden, bei FreeCAD baut man mit einem Objekt ein einzelnes Bauteil. Für eine Pyramide müsste man dann zwei (eigentlich fünf, aber da die Seiten symmetrisch sind, braucht man drei davon nicht wirklich separat) Komponenten einzeln erstellen und die Seitenwand am Ende auf eine Ebene legen, die Pyra-

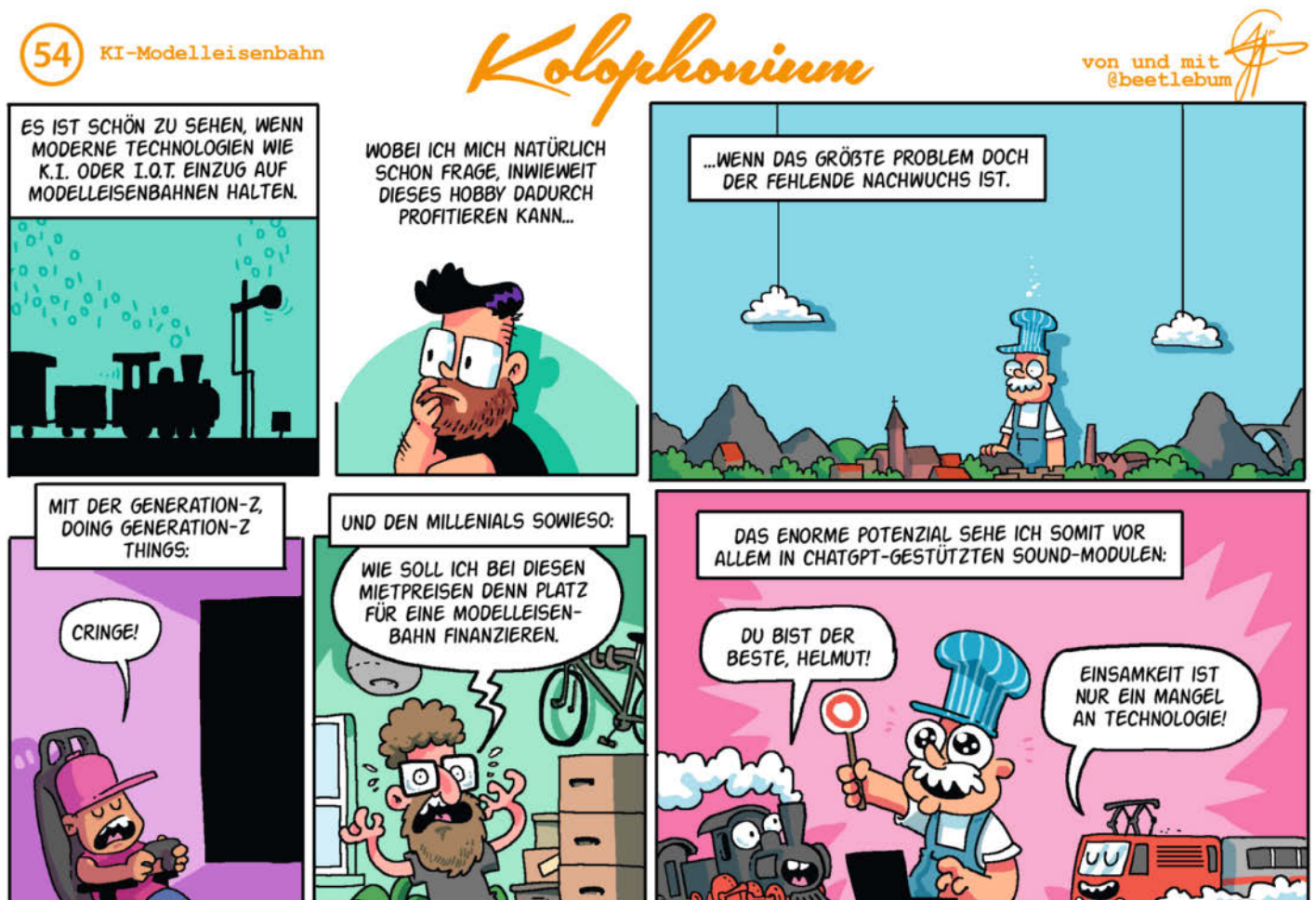
mide also aufklappen, damit die Projektion von oben funktioniert. Bei der Ansicht auf eine Kante stimmen die Maße wegen der doppelten Gehrung aber wieder nicht.

Fusion 360 kann das leider genauso wenig – FreeCAD ist damit also nicht allein. Hier ist es einfacher, direkt in der Zeichnung zu messen und die Werte auf die technische Zeichnung zu schreiben. Dafür nutze ich dann die *Manipulator Workbench*.

Den Thread aus dem FreeCAD-Forum zu diesem Thema kann man unter dem Link nachlesen; dort gibt es auch Fotos und die FreeCAD-Datei meines Leuchtturm-Projekts, bei dem das beschriebene Problem aufkam.

Nils Winkler

► make-magazin.de/xbbd



Highspeed-Drucker: AnkerMake M5

Von der chinesischen Firma Anker kennt man Powerstations und -banks, Solarkraftwerke oder Ladegeräte. Und jetzt bauen die auch 3D-Drucker? Wir waren skeptisch, nach dem ersten Test aber überzeugt: Der AnkerMake M5 ist schick, zuverlässig und schnell.

von Peter König

Vor rund einem Jahr lief die Kickstarter-Kampagne für den AnkerMake M5 und versprochen wurde einiges: Druckgeschwindigkeit von 250mm/s und damit etwa fünfmal so schnell wie die 3D-Drucker, die normalerweise in Maker-Werkstätten stehen, amtliches Metallchassis, eine Überwachungskamera mit KI, die den Druck abbricht, falls das Live-Bild zu sehr von der STL-Vorlage abweicht, und gemessen daran mit rund 700 Euro gar nicht so teuer. Auf den ersten Bildern sah das Gerät zudem so schick aus, als hätte man es nicht fotografiert, sondern in 3D-Software gerendert.

Die Erwartungen waren also hoch, als eine Woche vor Redaktionsschluss dieses Hefts das Testgerät eintraf. Der erste Eindruck: Es sieht auch in echt aus wie gerendert. Alle Teile stecken in gerundeten Gehäusen, Schrauben sind kaum zu sehen, Gummi ummantelt Kabelstränge, Motoren und Elektronik verstecken sich komplett im Gehäuse. Das ist ein Produkt, kein Bastelgerät.

Schnellstart

Ein wenig schrauben muss man aber, denn der Drucker wird gut verpackt in drei Baugruppen geliefert: Man schraubt den Portalrahmen mit acht Gewindeschrauben von unten durch die Bodenplatte fest, dann stößelt man vier Kabel ein, sichert sie mit Schrauben, klickt einen Deckel drauf und dann muss man nur noch den Filamenthalter an den Rahmen schrauben – das ist in einer Viertelstunde erledigt. Das bis 100°C beheizbare Bett levelt der Drucker beim ersten Einschalten selbst. Magnetisch haftet darauf ein Federstahlblech mit beidseitig rauer und mit PEI beschichteter Oberfläche.

Am bequemsten nimmt man den Drucker über die kostenlose App des Herstellers für iOS und Android im Betrieb, für die man ein Benutzerkonto anlegen muss. Die App fin-

det dann den Drucker in der Nähe und über sie bringt man den M5 ins lokale WLAN.

Wer gleich loslegen will, findet in der App schon ein paar vorgeladene Beispielvorlagen.

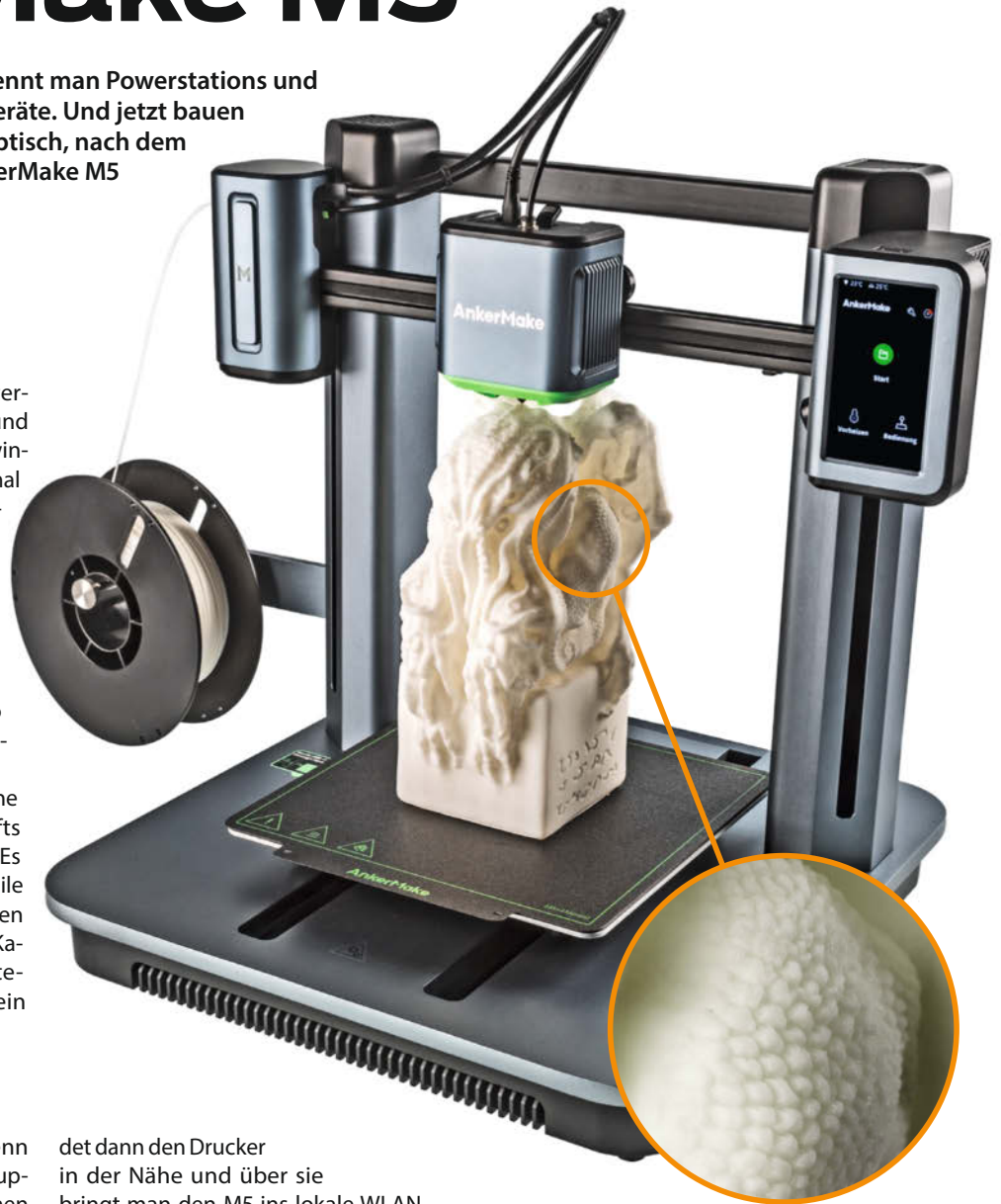
Erst muss man aber noch das Filament (1,75mm) einschieben, wobei klar wird, dass im Gehäuse links am Rahmen der Filamentsensor sitzt, der Vorschubmotor aber direkt im Kopf (Direct Drive). Dadurch und durch die maximale Düsentemperatur von 260°C verarbeitet der M5 nicht nur PLA, sondern etwa auch ABS und das flexible TPU.

Scheibchenweise

Von AnkerMake gibt es eine spezielle Software für Windows und macOS zum Download, die

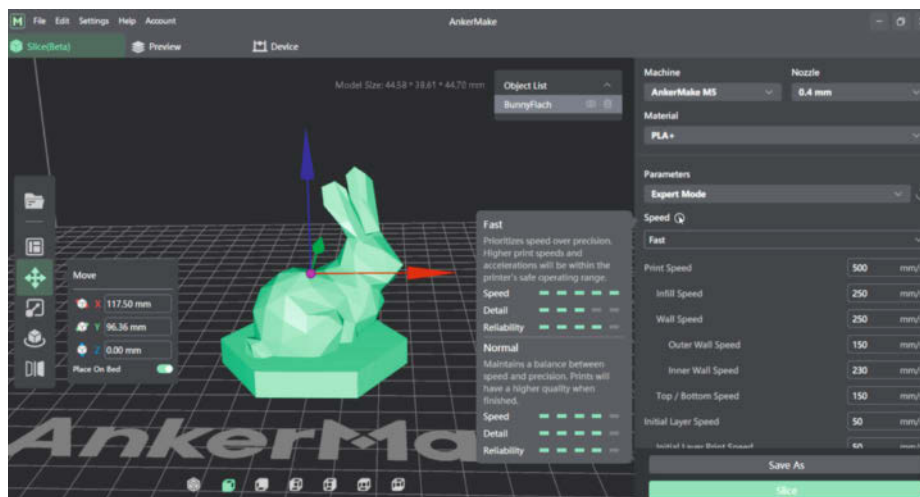
Slicer und Drucküberwachung vereint. Loggt man sich mit seinem Nutzerkonto dort ein, kann man Druckaufträge direkt aus dem Slicer auf den Drucker schicken, dort das Live-Bild der Kamera verfolgen, den Rechner runterfahren und von unterwegs in der App den Druckfortschritt weiter überwachen.

Das ist bequem, aber es gibt Alternativen, etwa Tools und Anleitungen, wie man der Drucker auch mit dem Prusa Slicer und Cura benutzt – dann muss man aber auf die automatische Fehlererkennung durch die Kamera verzichten und die G-Code-Dateien per USB-Massenspeicher auf den M5 bringen. Kleiner Haken: Der Drucker hat nur eine USB-C-Buchse





Roter Hase: Schnellmodus aus PLA, rund 15 Minuten Druckzeit, Kupferhase: Normalmodus aus BioFusion, 30 Minuten (Höhe jeweils 45mm)



Der Slicer erlaubt Zugriff auf viele Parameter, enthüllt aber auch Ungereimtheiten bei den Tempoangaben im Schnellmodus.

und USB-C-Sticks sind noch selten. Im Test funktionierte ein älterer Stick mit einem Adapter aber prima.

Bisher haben wir ausschließlich mit dem AnkerMake-Slicer gearbeitet (die Cloud ist schon sehr bequem), aber die „intelligente“ Kameraüberwachung abgeschaltet. Die verlangsamt die Vorbereitung in der Software und auch den Druck erheblich, weil bei den ersten Schichten zwischendrin der ganze Druckkopf samt Kamera nach oben gefahren wird, um das Ergebnis aus höherer Warte zu beäugen, bevor es zur nächsten Schicht geht. Da die Druckbetthaftung aller bisher von uns getesteter Materialien (PLA, PLA+, PETG, TPU, extrudr BioFusion, sogar ABS, wenn man mit Raft druckt) bislang stets so bombig war, dass es keine Fehldrucke gab, erwies sich die Überwachung für uns als verzichtbar.

Tempofrage

Kurz vor unserem Test gab es von AnkerMake noch ein Firmware-Update (sowas kann man mit einem Touch auf den Screen am Drucker übers Netz einspielen), das sogar ein Drucktempo von 500mm/s möglich machen soll. Wählt man im Slicer für das Material PLA+ den Modus *Fast*, taucht bei den Parametern *Travel Speed* und *Print Speed* tatsächlich dieser Wert auf, beim Druck prangt er auch stolz auf dem Display des Geräts. Schaut man sich im Slicer aber die Werte für einzelne Arten von Materialauftrag an, liegen die Angaben für *Infill*, *Outer* und *Inner Wall*, *Top*, *Bottom*, *Initial Layer* allesamt bei maximal 250mm/s (siehe Screenshot). Da darf man sich schon fragen, welche Teile des Modells wirklich mit 500mm/s gedruckt werden.

In der App gibt es sogar ein fertig geslictes *Benchy*, das in 17 Minuten und 40 Sekunden fertig sein soll – wie der Hersteller in einem Blogbeitrag schreibt (siehe Link), reizt das al-

lerdings den M5 aus und lässt ihn unter „Voll-last“ laufen, was zu einer „gewissen Instabilität“ führen soll. Jenseits solcher Rekordversuche sind die real 250mm/s des *Fast*-Modus aber immer noch ziemlich schnell. Bei 0,25mm Schichtdicke braucht der Drucker für die 25cm hohe *Cthulhu*-Statue auf unserem Foto knapp 12 Stunden, die Dresdner Frauenkirche war bei gleicher Höhe nach 9 Stunden fertig (Foto siehe Link). Da druckt man gerne mal größer. Maximal passen Objekte mit 23,5cm Grundfläche im Quadrat und 25cm Höhe in den Bau-raum.

Der *Fast*-Modus steht zwar nur für das vorgefertigte Materialprofil *PLA+* zur Verfügung, im Test ließ sich damit aber auch normales PLA prima drucken. Andere Materialien verarbeitet der M5 im *Normal*-Modus, der beim Infill ebenfalls auf 250mm/s kommt und die äußeren Wände mit immer noch 150mm/s aufbaut. In der Praxis macht das gar nicht so viel Unterschied: So dauerte der Druck unseres Test-Makeys (0,15mm Schichtdicke) im *Normal*-Modus nur knapp ein Viertel mehr Zeit als im *Fast*-Modus. Außerdem war die Qualität sichtbar besser, vor allem an scharfen Kanten, bei denen der Druckkopf des M4 bei Höchstgeschwindigkeit offenbar überschwingt und zur Wulstbildung neigt.

(Zwischen-)Fazit

Auch wenn der Drucker wirklich schnell ist und seit seiner Ankunft in unserer Redaktion praktisch Tag und Nacht druckt, ist eine Woche wenig Zeit für einen intensiven Test. Wir testen weiter – was wir zwischen Druckschluss und Erscheinungstermin dieses Hefts noch herausgefunden haben, lesen Sie deshalb in einer Online-Fortsetzung (siehe Link).

Was wir jetzt schon sagen können: Drucken mit dem AnkerMake M5 macht wirklich Spaß, weil das Gerät prima und unkompliziert arbei-



Früher war der bewegliche, in einem Stück gedruckte Test-Makey für viele 3D-Drucker eine harte Nuss, der AnkerMake M5 schafft ihn sogar im Schnellgang.

tet, trotz hohem Tempo überraschend gute Ergebnisse liefert und bisher mit jedem Material aus unserem 1,75mm-Fundus klarkam. Einen Haken hat er allerdings: Er ist ziemlich laut. Wer schnell druckt, muss auch viel pusten, damit das Material schnell aushärtet, um die nächste Lage draufschichten zu können.

Bei Redaktionsschluss variierten die Preise für den Drucker bei verschiedenen Händlern zwischen rund 720 Euro und 850 Euro. Stand heute können wir sagen: Die Investition lohnt sich, wenn man vor allem drucken und nicht am Drucker basteln will.

—pek

Alles zum Artikel
im Web unter
make-magazin.de/xp3t

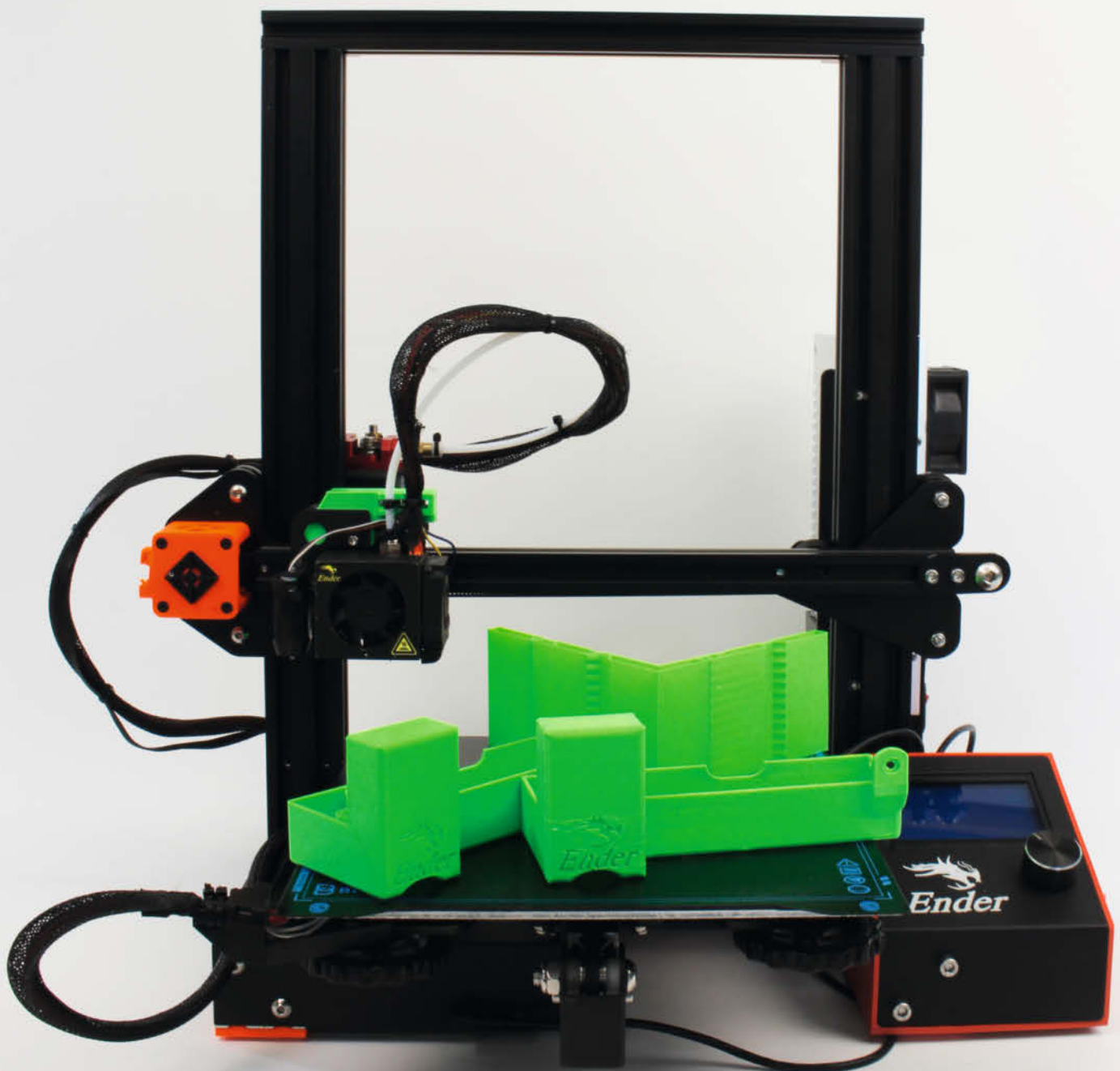


3D-Drucker-Tuning, Teil 3

Schneller drucken mit Klipper

In dieser Folge der Tuning-Reihe geht es ans Tempo: Mit der Drucker-Firmware Klipper gibt der Drucker so richtig Gas, ohne die dabei sonst üblichen Qualitätseinbußen. Dazu ist allerdings etwas Testarbeit nötig.

von Heinz Behling



In dieser dritten Folge der Tuning-Anleitung kommen wir zum Thema Geschwindigkeit. In diesem Punkt sind gerade preiswerte 3D-Drucker nicht gut aufgestellt. Preiswert heißt in diesem Zusammenhang: Alle Geräte, die unter 2000 Euro Verkaufspreis liegen. Inzwischen gibt es allerdings auch 3D-Drucker, die serienmäßig mit Klipper arbeiten, etwa den Creality K1, der unter 700 Euro kostet. Allerdings liegen uns mit diesem Gerät noch keine Erfahrungen vor, da die Auslieferung erst Mitte Mai beginnt.

Der geringe Preis bisheriger Geräte wird stets durch eine einfach aufgebaute Mechanik erkauft (zum Beispiel bei der Lagerung des Drucktisches auf Aluprofilen mit Laufrollen) und oft auch noch durch Verwendung von Teilen minderer Qualität. Einfache Mikroschalter oder Touch-Stifte als Homing-Sensoren mit geringer Wiederkehrgenauigkeit nach längerem Gebrauch senken Kosten, erfordern aber häufiges Nachjustieren. Das haben wir in den ersten beiden Folgen dieser Tuning-Reihe aber schon beseitigt.

Lager, die nur mit reichlich Spiel auf ihren Wellen laufen, verhindern, dass der Drucker senkrechte Flächen wirklich eben druckt (siehe Kasten „Lagerspiel und ähnliche Fehler“). Riemenantriebe, die wie Gitarrensaiten Resonanzfrequenzen aufweisen und bei ungünstigen Bewegungsfolgen anfangen, stark zu schwingen, sind weitere Beispiele für konstruktive Mängel. Das kompensieren die Hersteller, indem sie die maximale Druckgeschwindigkeit der Geräte gegenüber der theoretisch möglichen in der Firmware des Geräts deutlich reduzieren. Verändern Sie diese Grenzen und drucken deutlich schneller, zeigen sich die typischen Auswirkungen dieser Schwingungen **1**.

Die herstellerbedingte verringerte Geschwindigkeit macht übrigens auch die Verwendung billiger Mainboards möglich, deren Schrittmotor-Treiber die Motoren nicht optimal ansteuern können und daher ebenfalls ein Tempolimit darstellen. Dieses Manko wurde aber bereits in der ersten Folge unserer Tuning-Reihe beseitigt.

Dem geht es in dieser Folge nun an den Kragen: Das Mittel der Wahl dazu heißt *Klipper*. Diese 3D-Drucker-Firmware unterscheidet sich aber stark von den gängigen Firmwares wie etwa *Marlin*, denn Klipper ist zweigeteilt: Der kleinere Teil wird im 3D-Drucker installiert und kann nichts weiter, als Steuerbefehle für die Motoren und Heizungen entgegenzunehmen und Sensorsignale (u. a. Positionen und Temperaturen) nach außen weitergeben. Da die Drucker-Elektronik nichts anderes mehr machen muss (vor allem keine zeitraubende Display-Ansteuerung, keine Interpretation von G-code-Befehlen), erfolgt dies sehr schnell.

Der zweite Teil von Klipper findet seinen Platz auf einem deutlich leistungsfähigeren

Kurzinfo

- » Mehrfach schneller Druck mit Raspberry Pi/Octopi und Klipper
- » Messung der mechanischen Resonanzen des Druckers
- » Ausgleich der Resonanzen und dadurch mögliche höhere Druckgeschwindigkeit

Checkliste


Zeitaufwand:
ein bis zwei Tage


Kosten:
ab 70 Euro (bei Verwendung eines Raspberry Zero 2, Stand Mai 2023)

Material

- » Raspberry Pi (WLAN-fähiges Modell)
- » zum Raspberry passendes Netzteil
- » Speicherkarte 32 GB oder mehr

Werkzeug

» Messschieber

Mehr zum Thema

- » Heinz Behling, 3D-Drucker-Tuning: Sensor-loses Homing, Make 1/23, S. 78
- » Heinz Behling, 3D-Drucker tunen: Z-Achsen-Waagesensor, Make 2/23, S. 94

Alles zum Artikel
im Web unter
make-magazin.de/x2xh



Computer, nämlich einem Raspberry Pi. Der liest aus den Druckdateien die jeweiligen G-code-Befehle und steuert die Motoren und Heizungen des Druckers an. Der besondere Gag dabei: Klipper optimiert dabei die Motoransteuerung. Die Firmware beherrscht nämlich die sogenannte Resonanz-Kompensation. Bei diesem Verfahren werden die Resonanzfrequenzen des Druckers durch Vermessen von Probedrucken ermittelt und als Resonanzprofil auf dem Raspberry gespeichert.

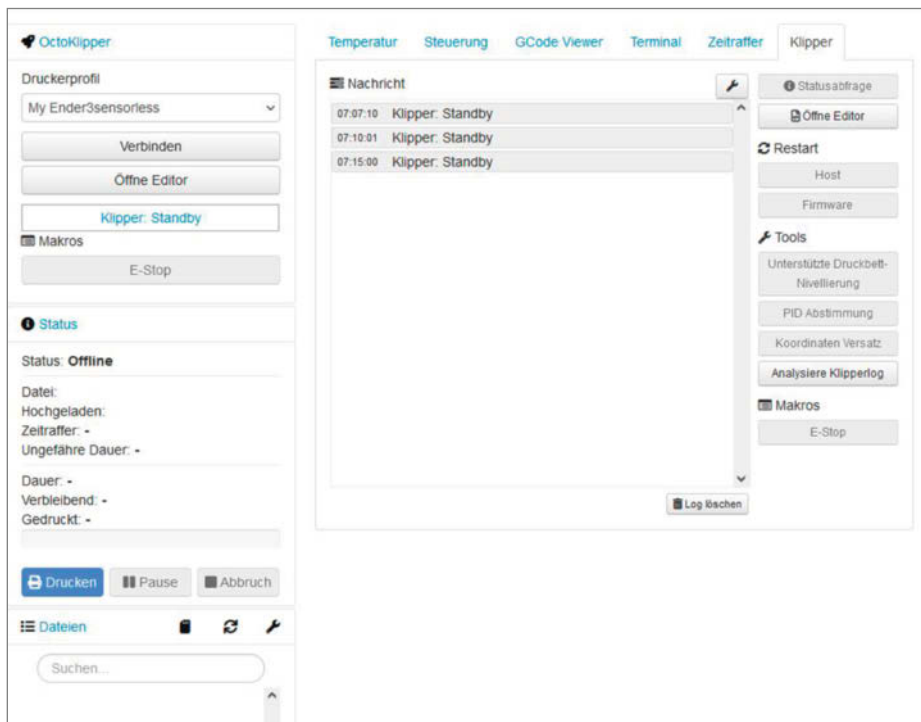
Dieses Resonanzprofil wird dann im späteren Druckbetrieb vor allem bei höherer Geschwindigkeit berücksichtigt: Bemerkt Klipper Motorbewegungen, die zu Resonanzschwingungen führen, dann ändert es die Steuerbefehle so, dass dem entgegengewirkt wird. Überspringen, wie in Bild **1** zu sehen, wird rechtzeitig durch entgegengesetzte Motorbewegungen, veränderte Beschleunigungswerte und sogar Anpassung der Motorströme verhindert oder zumindest erheblich verringert. Je nach Druckermodell können mit diesem Verfahren Geschwindigkeitssteigerungen auf das drei- bis fünffache erreicht werden. Statt nach 10 Stunden Druckzeit ist dann ein Job schon in 120 Minuten erledigt.

Diese Anleitung zeigt wieder am Beispiel des bereits umgebauten Creality Ender 3, wie man Klipper installiert. Aber weder der Umbau noch der Druckertyp sind zwingende Voraussetzung für Klipper. Falls Sie also den nicht ganz einfachen Umbau scheuen, können Sie auch serienmäßige 3D-Drucker mit Klipper betreiben. Auf Github (siehe Kurzinfo-Link)

können Sie nachschauen, ob es für Ihren Drucker bereits eine Konfigurationsdatei gibt. Aber selbst, wenn Sie dort nicht fündig werden, kann etwas googlen helfen: Vielleicht hat ja schon jemand dasselbe Gerät Klipper-fähig gemacht.



1 So sieht das Druckergebnis aus, wenn man mit der vierfachen der vom Hersteller vorgesehenen Geschwindigkeit druckt.



2 Die Web-Oberfläche von OctoKlipper

Der richtige Pi

Ein Raspberry Pi ist Voraussetzung für Klipper. Aber die sind im Moment teuer: Selbst ein Zero wird im Moment für etwa 70 Euro gehandelt, ein Raspi 4 kostet 180 bis über 200 Euro. Falls Sie bereits einen Raspi als Octoprint-Druckserver benutzen, können Sie den weiterverwenden. Im Folgenden verwenden wir nämlich eine Distribution für den Raspi, die nicht nur Klipper, sondern auch Octoprint enthält. Sie behalten also Ihren Druckerserver und bekommen Klipper dazu. Somit fallen keine zusätzlichen Kosten außer jenen für den Beschleunigungssensor an.

Auf Octoprint.org wird von der Verwendung eines Raspberry Pi Zero der ersten Generation abgeraten. Aus eigener Erfahrung weiß ich, dass Octoprint durchaus auf diesem Zero läuft, wenn man auf den Betrieb einer Kamera zur Druckerbeobachtung verzichtet. Unter Klipper sieht die Sache etwas anders

aus: Insbesondere die Resonanz-Kompensation (auch *Input shaper* genannt) kann den Zero 1 überfordern. Laut Meldungen in diversen Internet-Foren soll es mit dem Zero 2 besser funktionieren. Dies habe ich aber nicht testen können. Falls es einer von Ihnen ausprobiert, schicken Sie uns bitte einen Erfahrungsbericht.

Mein Raspi war ein 3B, der selbst mit Kamerateilbetrieb und Videoaufnahme einen eher unterforderten Eindruck machte, was Temperaturmessungen des Prozessors bestätigten. Auch dessen Vorgänger sollten ohne Probleme mit Octoprint und Klipper arbeiten. Tipp: Auch der Tastatur-Computer *Raspberry Pi 400* ist geeignet und meist besser und preiswerter verfügbar.

Installation OctoKlipperPi

Für den Raspberry Pi gibt es eine OctoPi-Variante namens *OctoKlipperPi*, die bereits

Klipper und alles für diese Kombination erforderliche inklusive des Compilers zur Herstellung des druckerspezifischen Firmware-Teils enthält. Dieses Betriebssystem basiert auf einem für 3D-Druck optimierten *Raspbian*, enthält neben Klipper und dem 3D-Druck-Server *OctoPrint* auch das *OctoKlipper-Plugin*, das beides miteinander verbindet, und schließlich einen mjpeg-Streamer, mit dem Sie den Druck per Kamera überwachen und Zeitraffer-Videos aufnehmen können. Wie bereits erwähnt sollten Sie den Kamera-Betrieb bei Verwendung eines Raspberry Zero nicht nutzen, da er den Prozessor relativ stark belastet und recht viel Arbeitsspeicher braucht. Beides benötigen wir künftig aber für Klipper, um den Drucker auf Trab zu bringen.

OctoKlipperPi ist als Speicherkarten-Image verfügbar, das mit dem Tool *Pi Imager* installiert werden kann (Download-Adresse siehe Kurzinfo-Link). Dazu wird eine zum benutzten Raspberry Pi passende Speicherkarte benötigt (32 GB).

Nach dem Start des Pi Imagers gelangen Sie mit Klicks auf *Other specific-purpose OSs* und *3D-Printing* zur Auswahlliste der 3D-Betriebssysteme. Dann können Sie nach Klick auf *OctoKlipperPi* die aktuelle Version *OctoKlipperPi (stable)* wählen. Anschließend ist noch die richtige SD-Karte einzustellen und Sie können das Image auf die Karte schreiben. Das dauert ein paar Minuten, da das Image erst vom Server überspielt werden muss.

Da der Pi Imager die Speicherkarte nach dem erfolgreichen Schreibvorgang automatisch abmeldet, müssen Sie sie einmal kurz aus dem Kartenleser herausziehen und wieder einstecken, um sie erneut anzumelden. Wechseln Sie dann in die *boot* genannte Partition der Karte. Dort finden Sie eine Datei mit dem Namen *octopi-wpa-supPLICANT.txt*. Öffnen Sie diese mit einem geeigneten Texteditor (Wordpad ist nicht geeignet, empfehlenswert ist Notepad++).

In der Datei gibt es drei Blöcke für die gängigen WLAN-Zugriffsverfahren (WPA/WPA2, WEP und Open/unsecured für ungesicherte Netze). In den Zeilen unterhalb des von Ihrem WLAN benutzten Verfahrens entfernen Sie die #-Zeichen und ersetzen *put SSID here* beziehungsweise *put password here* jeweils durch Ihren WLAN-Namen und das entsprechende Passwort. Entfernen Sie außerdem das #-Zeichen vor *#country=DE # Germany*. Speichern Sie die Datei und melden Sie sie vom System ab.

Stecken Sie die Speicherkarte in den Raspberry und schließen Sie den Drucker per USB-Kabel und das Raspi-Netzteil an. Schalten Sie auch den Drucker ein. Nach einigen Minuten ist der Druckerserver einsatzbereit und dessen Web-Oberfläche steht zur Verfügung. Dazu muss im Browser meist nur *octopi.local* eingegeben werden. Sollte Ihr Browser damit nicht klarkommen, brauchen

3 Kompilier-Anweisungen

```
# This file contains common pin mappings for the BIGTREETECH SKR mini
# E3 v2.0. To use this config, the firmware should be compiled for the
# STM32F103 with a "28KiB bootloader" and USB communication. Also,
# select "Enable extra low-level configuration options" and configure
# "GPIO pins to set at micro-controller startup" to "IPA14".
# The "make flash" command does not work on the SKR mini E3. Instead,
# after running "make", copy the generated "out/klipper.bin" file to a
# file named "firmware.bin" on an SD card and then restart the SKR
# mini E3 with that SD card.
```

Sie die IP-Adresse des Raspi, die Sie auf der Web-Oberfläche des Routers nachschlagen können.

Zunächst müssen Sie im Installations-Wizard einige Fragen beantworten und jeweils mit *Weiter* zum nächsten Fenster wechseln. Unter anderem müssen Sie einen Benutzernamen und ein Passwort eingeben und sich bitte auch merken! **Achtung:** Im Fenster *Konfiguriere dein Druckerprofil* gibt es vier Registerreiter. Sie müssen alle vier abarbeiten und u. a. die Maße der Druckfläche eingeben. Schließlich erscheint das übliche Octoprint-Fenster, dass sich nun aber OctoKlipper nennt und einige zusätzliche Felder enthält ②.

Klipper für Ihren Drucker

Dann geht es mit der Kompilierung des druckerspezifischen Klipper-Teiles weiter. Dazu braucht man eine Konfigurationsdatei für das jeweilige Gerät. Falls es den Drucker in mehreren Varianten gibt (beispielsweise wie bei dem hier verwendeten Ender 3 mit einem nicht serienmäßigen Mainboard), muss auch die Konfigurationsdatei genau zu dieser Variante passen. Auf Github gibt es eine Menge an Konfigurationsdateien. Über den Kurzinfo-Link gelangen Sie zur entsprechenden Download-Adresse. Für den in den ersten beiden Folgen der Tuning-Reihe bereits umgebauten Creality Ender 3 gibt es auf der Projektseite eine entsprechende Konfigurationsdatei für den sensorlosen Ender mit dem Mainboard *SKR Mini E3 V2*. Auch eine Version ohne Wägebalken-Sensor habe ich dort gespeichert. Das hatte leider seinen Grund in einem der letzten Klipper-Updates: Seitdem sind negative Werte für den Z-Sonden-Versatz nicht mehr möglich. Den braucht dieser Sensor aber. Sobald ich eine Methode gefunden habe, um diese Klippe zu umschiffen, werde ich sie in der Make mitteilen.

Die Konfigurationsdatei selbst ist kein Bestandteil des Klipper-Firmware-Teils für den Drucker. Die Dateien enthalten aber am Anfang jeweils Anmerkungen, was beim Kompilieren zu beachten ist. Öffnen Sie daher Ihre Konfigurationsdatei in einem Editor wie Notepad++ und schauen Sie sich den Anfang der Datei an. Dort finden Sie die beim Kompilieren zu beachtenden Mainboard-spezifischen Einstellungen wie Prozessortyp, Angaben über den Bootloader, GPIO-Ports und ähnliche Werte. ③.

So benutzen Sie die: Sie brauchen eine SSH-Verbindung zum Raspberry, zum Beispiel mit dem Tool Putty. Nach dessen Start geben Sie als Hostname `octopi-local` oder die IP-Adresse des Raspi ein. Nach einem Klick auf *Open* und bestätigen der Key-Übernahme melden Sie sich mit dem Usernamen `pi` und dem Passwort `raspberrypi` an. Mit `cd /klipper` wechseln Sie das aktuelle Verzeichnis.

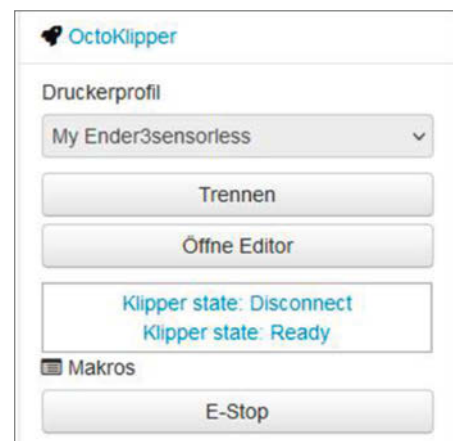
④ Vor dem Kompilieren muss dies eingestellt werden.

Geben Sie dann `make menuconfig` ein. Im nun folgenden Einstellungsmenü setzen Sie den Markierungsbalken zunächst auf *Enable extra low-level configuration options* und tippen auf die Leertaste. In den eckigen Klammern vor dem Eintrag muss ein Stern erscheinen.

Wechseln Sie dann auf die jeweils darunterliegenden Zeilen und wählen Sie nach einem Tipp auf die Eingabetaste jeweils die Einstellung gemäß dem Header der Konfigurationsdatei aus. Im Falle des modifizierten Ender 3 muss das wie in Bild ④ aussehen.

Ist alles eingestellt, tippen Sie auf `q` und bestätigen das Speichern der geänderten Datei mit `y`. Jetzt ist alles zum Kompilieren bereit. Das erfolgt mit dem Befehl `make`. Daraufhin rauschen etliche Meldungen durch das Terminal-Fenster. Zum Schluss ist die Firmware fertig und liegt im Unterverzeichnis `out`. Sie heißt *klipper.bin*.

Diese Datei muss nun vom Raspi auf Ihren PC und von dort auf eine leere Speicherkarte kopiert werden. An die Dateien auf dem Raspi gelangen Sie zum Beispiel per SSH mit *FileZilla* (Download-Adresse siehe Kurzinfo-Link). In FileZilla sind Server-Adresse, Usernamen und Passwort identisch mit denen in Putty. In der rechten Hälfte des Programmfensters klicken Sie sich bis zum Verzeichnis `/home/pi/klipper/out` durch. In der linken Hälfte wählen Sie das Verzeichnis auf Ihrem PC, in das Sie die Firmware-Datei kopieren möchten. Per Rechtsklick auf *klipper.bin* und Herunterladen wird die Datei kopiert. Im Zielverzeichnis müssen Sie diese dann in *firmware.bin* umbenennen. Beenden Sie FileZilla und kopieren Sie die *firmware.bin* auf eine leere Speicherkarte.



⑤ Im Menü von OctoKlipper findet sich auch ein Editor für die Drucker-Konfiguration.

Bevor Sie die neue Firmware auf den Drucker installieren, schauen Sie in dessen Menü nach, welcher Z-Sonden-Versatz eingestellt ist. Notieren Sie sich den Wert! Legen Sie dann die Speicherkarte in den wieder ausgeschalteten Drucker und schalten Sie ihn erneut ein. Wundern Sie sich nicht: Das Display des Druckers bleibt von nun an leer, es wird von Klipper nicht benutzt. Die Bedienung des Druckers erfolgt von nun an ausschließlich per OctoKlipper-Web-Oberfläche.

Im (hoffentlich noch nicht geschlossenen) Putty-Terminalfenster geben Sie dann den Befehl

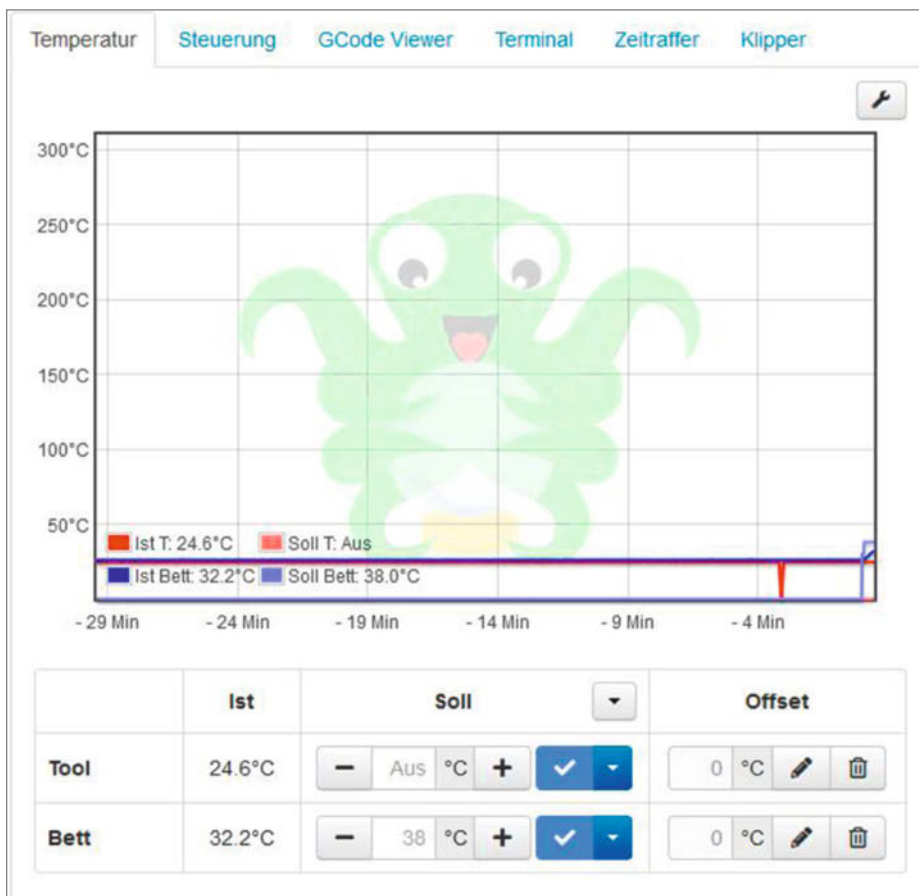
```
ls /dev/serial/by-id/*
```

ein. Sie erhalten dann eine längere Pfadangabe zurückgeliefert, die für den per USB

angeschlossenen Drucker steht. Das sieht dann etwa so aus:

```
/dev/serial/by-id/usb-Klipper_stm32f103
xe_35FFD7054E4B323517881143-if00
```

Die Angabe kann bei Ihrem Drucker anders lauten. Ersetzen Sie die in der Drucker-Konfigurationsdatei im Abschnitt [mcu] enthaltene Angabe hinter serial: durch diesen Pfad.



6 Sobald Sie im unteren Auswahlfeld eine höhere Bett-Temperatur gewählt haben, muss die blaue Kurve anfangen zu steigen.

```

Temperatur  Steuerung  GCode Viewer  Terminal  Zeitraffer  Klipper

Recv: OK B:21.5 /0.0 T0:19.2 /0.0
Send: M105
Recv: ok B:21.5 /0.0 T0:19.1 /0.0
Send: G28
Recv: ok
Send: RESTART
Recv: ok
Recv: // Klipper state: Disconnect
Send: M105
Recv: ok T:0
Recv: // Klipper state: Ready
Send: G28
Recv: !! Endstop y still triggered after retract
Recv: ok
Send: M105
Recv: ok B:21.6 /0.0 T0:19.1 /0.0
Send: M105
Recv: ok B:21.5 /0.0 T0:19.2 /0.0

```

7 Endstop still triggered... deutet auf eine falsche Einstellung beim sensorlosen Homing hin.

Nun muss diese Datei noch auf den Raspi gelangen. Da macht es uns OctoKlipper einfach, denn es besitzt einen eigenen Editor für die Konfigurationsdatei. Sie starten ihn per Klick auf **Öffne Editor** in der Menüleiste 5. Per Copy & Paste kopieren Sie dann den Inhalt der Konfigurationsdatei in das OctoKlipper-Editorfenster. Mit **Speichern & Schließen** beenden Sie den Editor. Die folgende Frage beantworten Sie mit **Neustart**.

Jetzt folgt der erste Test der Kommunikation zwischen Raspi und Drucker. Stellen Sie im Temperatur-Bereich des OctoKlipper-Fensters eine höhere Temperatur für das Druckbett ein. Der Drucker sollte dann mit dem Heizen beginnen und die Temperaturkurve im Diagramm entsprechend steigen 6.

Falls das nicht der Fall ist, kontrollieren Sie den Pfadseintrag in der Konfigurationsdatei mit dem Editor. Prüfen Sie außerdem, ob das USB-Kabel am Raspi und am Drucker korrekt eingesteckt ist und ob es wirklich ein USB-Datenkabel ist. Es gibt auch reine Ladekabel, denen die Datenleitungen fehlen (nicht lachen, eigene Erfahrung).

Der zweite Test betrifft das Homing: Im Terminalfenster von Octoprint geben Sie G28 ein. Bei diesem Test sollten Sie den Finger stets am Schalter des Druckers haben. Falls beispielsweise beim Homing der Z-Achse der Druckkopf nicht rechtzeitig stoppt, schalten Sie den Drucker aus, bevor die Druckdüse das Druckbett beschädigt. Der Drucker sucht nun die Nullpunkte aller drei Achsen, beginnend mit der X-Achse. Der Druckkopf sollte zur linken Seite fahren bis zum Anschlag und nach kurzer Zeit wieder ein kleines Stück zurück. Danach setzt sich der Drucktisch nach hinten in Bewegung bis zum Anschlag. Falls die entsprechenden Teile aber dauerhaft an die Anschläge rattern, müssen die entsprechenden Einstellungen in der Konfigurationsdatei geändert werden. Das ist auch der Fall, wenn OctoKlipper meldet, dass einer der Achsensensoren dauerhaft getriggert ist 7. Dies gilt für den in den ersten beiden Folgen umgebauten Ender3. Ist Ihr Drucker nicht auf sensorloses XY-Homing umgebaut und/oder hat er keinen Wägebalken als Z-Achsen-Sensor, dann funktioniert die entsprechende Achse ohne weitere Einstellungen.

Als Nächstes öffnen Sie den Editor. In der Konfiguration suchen Sie nach dem Abschnitt [probe]. Falls der nicht existiert, suchen Sie nach [bltouch]. In dem jeweiligen Abschnitt müssen Sie den Wert hinter z_offset: = durch den zuvor bei Ihrem Drucker abgelesenen Z-Sonden-Versatz ersetzen. Kam es beim sensorlosen Homing zum Rattern am Anschlag, dann suchen Sie in der Konfiguration auch noch nach dem Abschnitt [stepper_x] beziehungsweise [stepper_y], je nachdem, welche Achse Probleme machte. In diesen Abschnitten ist die Zeile driver_SGTHRS: 50 für die Anschlag-

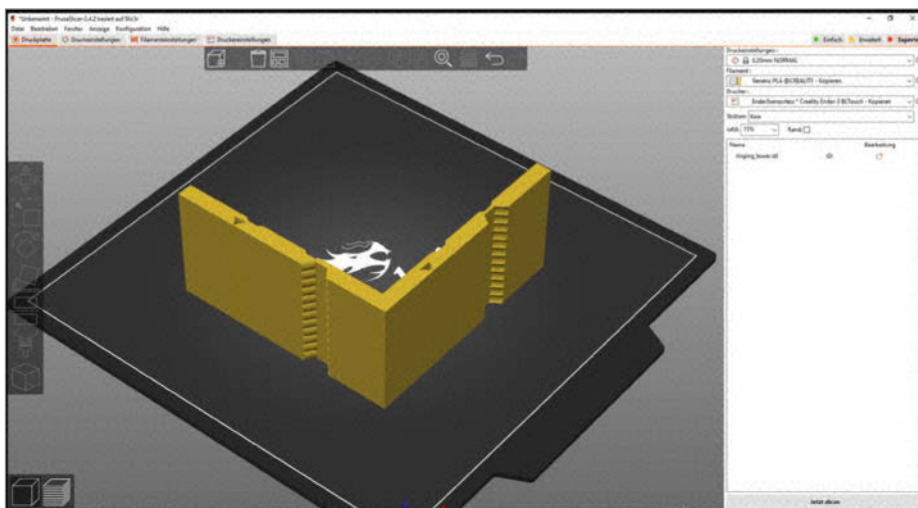
erkennung zuständig. Bei einem Anschlagratern müssen Sie den Wert etwas verringern, bei dauerhaft getriggertem Sensor hingegen erhöhen. Danach die Datei mit q und y speichern, einen Neustart veranlassen und erneut das Homing testen. Jetzt ist Ihr Drucker wieder einsatzbereit. Sie können bereits wieder mit ihm drucken.

Resonanz-Kompensation

Doch wir wollen ihn ja noch auf Geschwindigkeit tunen, genauer auf hohe Druckqualität bei hoher Druckgeschwindigkeit. Schnell drucken konnte er schon vorher, doch wie in Bild ① zu sehen ist, war das Ergebnis noch verbesserungsfähig. Besonders hinter den eingedruckten Buchstaben (hinter bezieht sich hier auf die Richtung des Druckkopfes beim Druck) zeigen sich merkwürdige Rillen und Konturen. Dieses *ringing* (oder auch *echoing*, *ghosting* oder *rippling* genannte Phänomen sind ungewollte Wiederholungen insbesondere von rechtwinkligen Kanten, die durch Nachschwingen des Hotends oder des Drucktisches bei starken Richtungsänderungen entstehen. Besonders die per Riemen angetriebenen Teile des Druckers (beim Ender also die Y- und die X-Achse) verhalten sich dann wie Gitarrensaiten: Die Richtungsänderung entspricht dem Zupfen der Saite, die so lange mit ihrer Resonanzfrequenz nachschwingt, wie es die innere Reibung zulässt.

Klipper besitzt eine Funktion, die solchen Schwingungen entgegenwirken kann. Sie heißt *input shaper*. Deren Funktionsprinzip ist vereinfacht dargestellt so: Beim Druck eines Objekts werden die Bewegungsbefehlsfolgen zunächst daraufhin analysiert, ob sie eine schwingungsauslösende Richtungsänderung enthalten. Falls dies der Fall ist, werden die Bewegungsbefehle und die in der Firmware eingestellten Werte für die Beschleunigung bzw. das Abbremsen der Mechanik zu Beginn bzw. am Ende einer Bewegung so geändert, dass eine Art Gegenschwingung zum Nachschwingen entstehen wird. Resonanzschwingung der Mechanik und von Klipper erzeugte Gegenschwingung heben sich im Idealfall völlig auf.

Damit dies auch funktioniert, braucht die Funktion die Resonanzfrequenzen. Im Falle des Ender 3 sind dies die Frequenzen für die Y- und die X-Achse. Die Z-Achse (also die Höhenverstellung des Druckkopfes) spielt keine Rolle, da deren Bewegungen nur sehr gering



⑧ Das Test-Objekt mit scharfen Kanten und Schlitten, hier im Prusa-Slicer

und selten sind. Außerdem ist der Spindelantrieb dieser Achse ohnehin nicht sehr schwingungsanfällig.

Die Resonanzfrequenzen der Achsen sind von einigen konstruktiven und Geräte-individuellen Faktoren abhängig: So spielt die Länge der Zahnriemen und deren Material eine Rolle. Die Riemenspannung wirkt sich ebenfalls stark auf die Resonanz aus. Außerdem spielt das Gewicht des Druckkopfes bzw. des Drucktisches eine wichtige Rolle. Auch die Ströme, mit denen die Schrittmotoren der Achsen angetrieben werden, wirken sich ebenso aus wie die Reibung der ganzen Mechanik. Alles in allem sind die Resonanzfrequenzen daher keine Werte, die für alle Exemplare eines Druckertyps gleich sind, sondern eine sehr individuelle Angelegenheit. Das bedeutet: Damit input shaper richtig arbeiten kann, müssen wir zunächst diese Frequenzen herausfinden.

Dazu fertigen wir einen Probedruck an. Das Testobjekt (Download-Adresse siehe Kurzinfo-Link) enthält einige Herausforderungen für die Druckermechanik: Scharfe Kanten, dünne Schlitten und Wellenmuster ⑧. Die dünnen Schlitten dienen übrigens dem Erkennen eines weiteren Übels bei hohen Druckgeschwindigkeiten: *smoothing*. Wird eine Außenecke gedruckt, muss der Druckkopf/Drucktisch rechtzeitig vor dem Eckpunkt zum Stillstand kommen. Andernfalls schießt der Kopf/Tisch infolge der Elastizität des Antriebs (Riemen!) über das Ziel hinaus und es bilden sich statt scharfen Ecken überstehende Rundungen.

Das Abbremsen der beiden Achsen muss so erfolgen, dass dieses Überspringen nicht eintritt. Infolge der Masse von Kopf und Tisch kann man diese Teile nicht beliebig schnell beschleunigen bzw. abbremsen (was ja nur eine Beschleunigung entgegen der Bewegungsrichtung darstellt). Ist die in Klipper eingestellte maximale Beschleunigung (max_accel) zu groß, wird zu heftig gebremst und *smoothing* tritt ein. Auch den Wert für die maximal zulässige Beschleunigung können wir am Testdruck ablesen.

Laden Sie die Druckdatei in Ihren Slicer. Achten Sie darauf, dass keinerlei die Druckqualität beeinflussende Plugins oder Funktionen eingeschaltet sind. Diese Druckereinstellungen sind erforderlich:

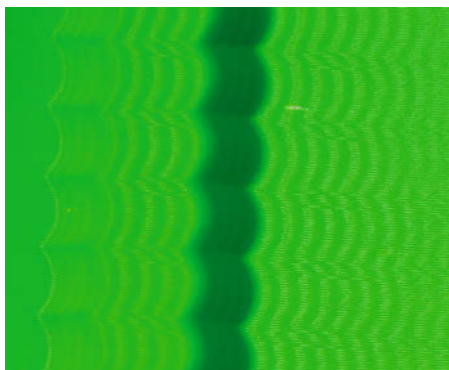
- Layerhöhe: 0,2mm
- Wandstärke: 1 Linie
- Füllung: 0%
- Geschwindigkeit für die Wände und Außenkonturen: 100mm/s

Slicen Sie die Druckdatei damit. Bevor Sie den Druck beginnen, müssen Sie im Terminal-Fenster von OctoKlipper noch nacheinander die Befehle aus Listing ⑨ eingeben. Starten Sie dann den Druck.

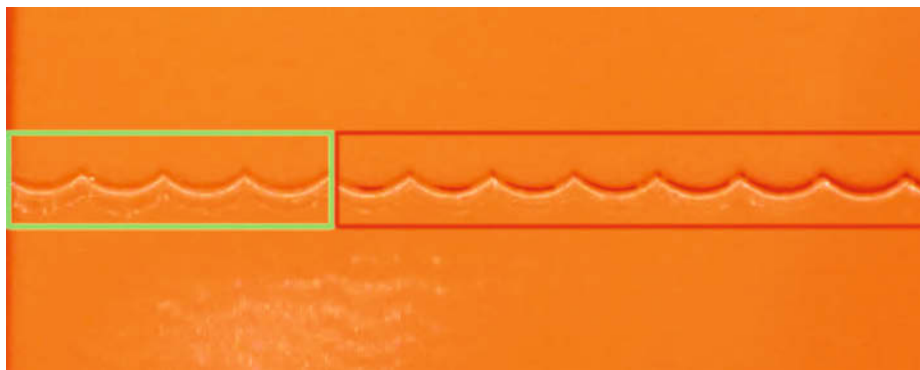
Nach dem Druck schauen Sie sich die Kanten des Druckteils genauer an: Sie werden die typischen Wiederholungen der Konturen finden. Aus dem Abstand der Wiederholungen (also der Wellenlänge der Schwingung) und der Druckgeschwindigkeit lassen sich die Frequenzen berechnen. Da sich aber eine ein-

⑨ Befehle vor dem Testdruck

```
SET_VELOCITY_LIMIT ACCEL_TO_DECEL=7000
SET_PRESSURE_ADVANCE ADVANCE=0
SET_INPUT_SHAPER SHAPER_FREQ_X=0 SHAPER_FREQ_Y=0
TUNING_TOWER COMMAND=SET_VELOCITY_LIMIT PARAMETER=ACCEL START=1500 STEP_DELTA=500 STEP_HEIGHT=5
```



10 So sieht das Wellenmuster auf dem Testdruck aus.



12 Hier sind vier Wellen mit geschlossenem Spalt. Die Beschleunigung beim letzten davon betrug 3000mm/s².



11 Mit einem Bleistift auf Papier durchgerieben erkennt man die Schwingungen meist deutlicher.

zelne Schwingung nur sehr ungenau messen lässt, messen Sie die Gesamtlänge möglichst vieler hintereinander liegenden Schwingungen 10.

Notieren Sie die gemessene Länge, die Schwingungsanzahl und die Achsenbezeichnung. Falls Sie Probleme haben, die Schwingungen direkt auf dem Kunststoff erkennen zu können, hilft ein Trick: Legen Sie ein Stück Papier auf den Bereich mit den Schwingungen und reiben Sie mit einem Bleistift darüber: Das Schwingungsmuster überträgt sich so als Graphit-Muster aufs Papier und ist besser zu erkennen und zu messen 11. Das führen Sie für die X- und Y-Achse durch, die Seiten des Testobjekts sind entsprechend gekennzeichnet. Bei meinem Drucker habe ich bei der X-Achse 7 Schwingungen über einer Länge von 10,18mm und für die Y-Achse 6 Schwingungen mit 7,76mm Länge gemessen.

Mit der Formel $F = V \times N/D$ (V: Druckgeschwindigkeit 100mm/s, N: Anzahl der Schwin-

gungen, D: gemessene Länge) erhalten Sie die Resonanzfrequenz F der jeweiligen Achse. Das ergibt für die X-Achse

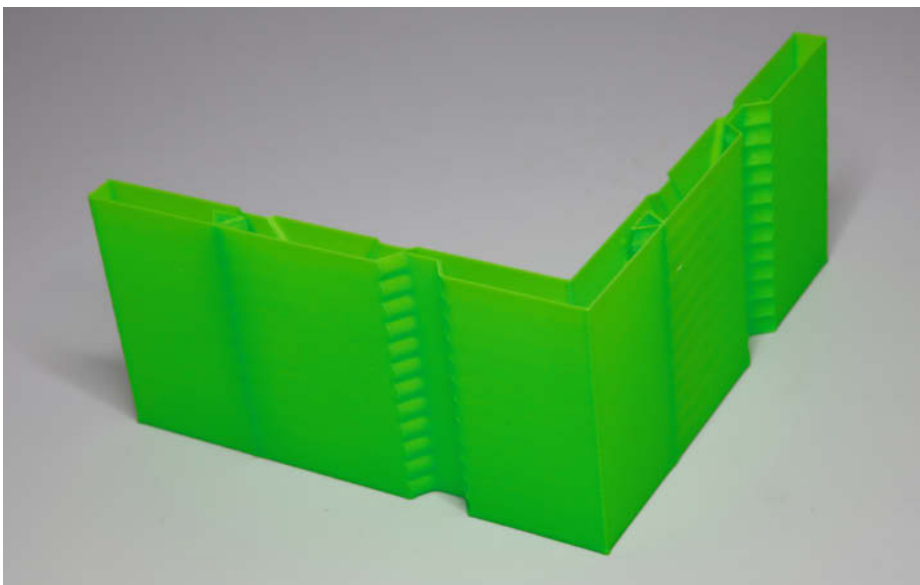
$$F(X) = 100\text{mm/s} \times 7/10,18\text{mm} = 68,8 \text{ Hz}$$

Für die Y-Achse ergibt sich entsprechend 77,3Hz. Beide Werte müssen nun ihren Platz in der Konfigurationsdatei finden. Öffnen Sie in OctoKlipper also erneut den Editor. Suchen Sie darin den Abschnitt mit der Überschrift [input_shaper]. Darunter ersetzen Sie die beiden dort schon vorhandenen Frequenzen durch die von Ihnen ermittelten. In meinem Beispiel sieht das so aus:

```
[input_shaper]
shaper_freq_x: 68.8
shaper_type_x: mzv
shaper_freq_y: 77.3
shaper_type_y: mzv
```

Sie sehen dort aber noch zwei weitere Zeilen für den shaper_type. Die input-shaper-Funktion besitzt mehrere Algorithmen für den Schwingungsausgleich mit den Bezeichnungen zv, mzv, zvd, ei, 2hump_ei und 3hump_ei. Der Standard ist mzv. Das sollten Sie auch zunächst beibehalten.

Das Testobjekt enthält auch ein senkrechtes, wellenförmiges Spaltmuster 12. Der Spalt ist 0,15mm breit. Das ist etwas weniger, als solch ein Drucker wie der Ender 3 drucken kann. Der Spalt sollte also bei korrektem Druck gerade noch geschlossen sein. Wundern Sie sich nicht, wenn der Spalt seine Breite geändert hat. Beim Testdruck ändert sich nämlich jeweils nach 5mm Druckhöhe die Beschleunigungseinstellung für die X- und Y-Achse, beginnend bei 1500mm/s² in Schritten zu je 500mm/s². Schauen Sie nach, bei welcher Welle (von unten gezählt) der Spalt gerade noch zu ist, zum Beispiel wie im Bild beim vierten. Der wurde mit einer Beschleunigung von 3000mm/s² gedruckt. Diesen Wert müssen Sie in der Konfigurationsdatei im Abschnitt [printer] hinter max_accel einsetzen.



13 Glatt, ohne Wellen: So sollte der Testdruck nach all den kryptischen Einstellungen aussehen.



14 Deutlich verbessert: Mit input shaper kamen keine lästigen Nachschwingungen hinter der Schrift mehr vor. Die etwas unsaubere Rundung vorne kommt daher, dass ohne Stützstrukturen gedruckt wurde.

Jetzt führen Sie erneut einen Druck des Testobjekts durch. Diesmal aber nach einem Reset des Druckers (im OctoKlipper-Terminal-Fenster RESTART eingeben) und ohne die vorgeschalteten Befehle aus Listing 9. Das bewirkt, dass diesmal input shaper mit den neuen Einstellungen benutzt wird.

Sind im neuen Testdruck keine Nachschwingungen mehr enthalten **13**, müssen Sie auch nichts mehr ändern. Gibt es noch

störende Wellen, führen Sie den Test nochmal durch, verwenden als Vorschaltbefehl SET_INPUT_SHAPER SHAPER_TYPE=EI. Führt auch EI nicht zum Erfolg, checken Sie die anderen zuvor aufgeführten Algorithmen bis zum bestmöglichen Ergebnis.

Zum Schluss habe ich dann nochmal die Abdeckung aus Bild 1 gedruckt, wieder mit 300mm/s. Das Ergebnis war jetzt erheblich besser **14**. —hgb

Lagerspiel und ähnliche Fehler

Diese ganze Tuning-Anleitung kann nicht zum Erfolg führen, falls Ihr Drucker mechanische Probleme wie ausgeschlagene Lager hat. Auch Zahnriemen, die irgendwo schleifen, wirken sich sehr nachteilig auf das Druckergebnis aus. Wenn deren Zahnung irgendwo reibt, bringt auch das den Mechanismus zum Schwingen. Solche Schwingungen aber kann input shaper nicht beseitigen.

Verdächtig ist auch, wenn die beim Testdruck herausgefundenen Resonanzfrequenzen sehr niedrig sind (25Hz oder weniger). Die rühren dann von einem wackeligen Rahmen oder unsicherem Stand des Druckers her. Und beim Thema Rahmen: Kontrollieren Sie, falls der Drucker

einen wackeligen Stand auf eigentlich ebenem Untergrund hat, ob die Rahmentheile wirklich rechtwinklig zueinander verschraubt sind. Da erlebt man so manche Überraschung!

Gerne werden als Nachrüstteile für preiswerte 3D-Drucker „verbesserte“ Riemenspanner verkauft, meist aus stabilem Metall gefertigt. Die verleiten dazu, die Riemen sehr fest zu spannen. Dann aber verhalten die sich erst recht wie Gitarrensaiten und schwingen munter drauflos. Die Zahnriemen sollten so gespannt sein, dass sie spielfrei laufen, aber bitte nicht mehr. Das führt außer zu lästigen Schwingungen auch noch zu erhöhtem Verschleiß.

OXOCARD SCIENCE

ENTDECKE UND ERFORSCH DEINE UMWELT

Programmierbare Multisensorplatte mit hochauflösendem Farbscreen, fünf Tastern und vielen Sensoren.

Die sieben eingebauten Umweltsensoren liefern zwölf Sensorwerte, mit denen du experimentieren kannst.

- | | |
|-------------------------------|----------------------|
| • Temperatur | • Luftqualität (IAQ) |
| • Luftdruck | • Licht |
| • Feuchtigkeit | • Infrarot |
| • VoC | • Dezibel |
| • CO ₂ (berechnet) | • Frequenzen |
| • Ethanol (berechnet) | • Beschleunigung |

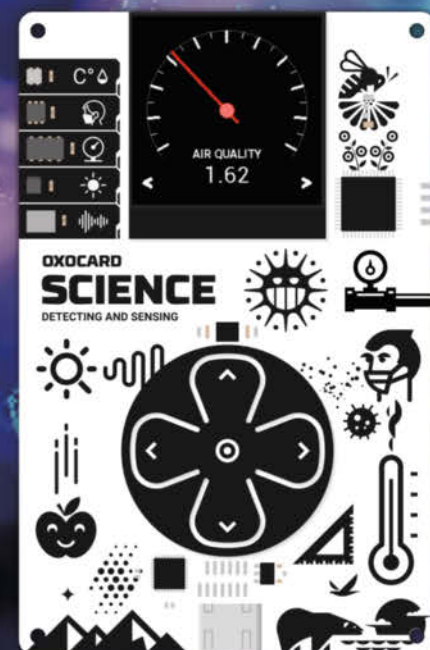
INFOS UNTER:

OXOCARD.CH



Jetzt im Heise-Shop bestellen

ORIGINALGRÖSSE – 55×85mm

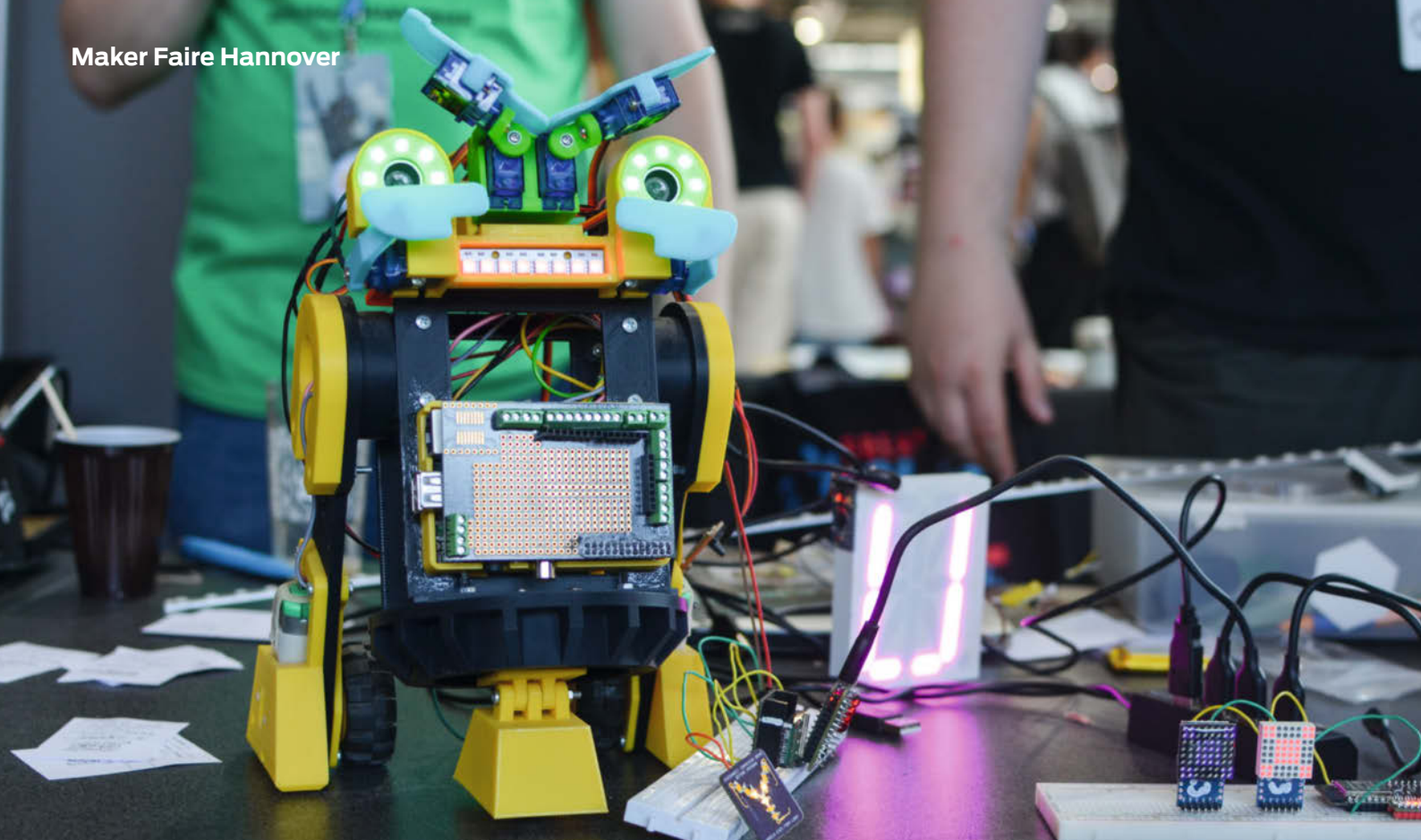


MAKER-FREUNDLICH



14+ 14-99





Last Call for Maker – noch bis zum 11.06.

Die Maker Faire Hannover sucht dich und dein Projekt. Melde dich und dein Projekt noch bis 11. Juni auf unserer Homepage an und sichere dir deinen Platz auf der größten Maker Faire im deutschsprachigen Raum. Wir freuen uns auf dich!

von Leonhard Ure



Wir laden dich ein bei der 9. Ausgabe der Maker Faire Hannover am **19. und 20. August** dein Projekt in einer von drei Hallen aus- und vorzustellen! Wir bieten die perfekte Plattform für alle, die sich für den kreativen Umgang mit Technik begeistern. Digitale Pioniere finden hier Orientierung und Inspiration und können sich mit Gleichgesinnten vernetzen. Zusätzlich zu den beiden Veranstaltungstagen freuen wir uns auf das **Get-Together** am **Abend des 19. August** mit dir. Die perfekte Gelegenheit, andere Maker und ihre Projekte in entspannter Atmosphäre bei einem Bier und Snack kennenzulernen.

Make Education

Auch wird das Thema *Make Education* immer präsenter. Hier ist *learning by making* das Motto. Wir sind weiterhin auf der Suche nach spannenden und innovativen Education-

Projekten, die vor allem den Nachwuchs begeistern. Spannende Schulprojekte gibt es schon lange auf der Maker Faire zu bewundern. Seit letztem Jahr haben wir bei der Maker Faire Hannover zusätzlich einen **Lehrertag** im Angebot, der auch in diesem Jahr wieder stattfindet, und Bundesbildungsministerin Bettina Stark-Watzinger konnten wir auch wieder als Schirmherrin gewinnen.

Forschen und zeigen

Zu den diesjährigen Neuerungen gehört, dass wir nun auch für Forschungsinstitute sowie Hochschulen und Universitäten eine Plattform sein wollen, auf der sie ihre zukunftsweisenden Projekte und Technologien kostenfrei der Öffentlichkeit vorstellen können. So könnt ihr ganz einfach und unkompliziert eure Forschung unseren erwarteten 15.000 Besucherinnen und Besuchern präsentieren.



3D-Druck auf der Maker Faire Hannover 2022

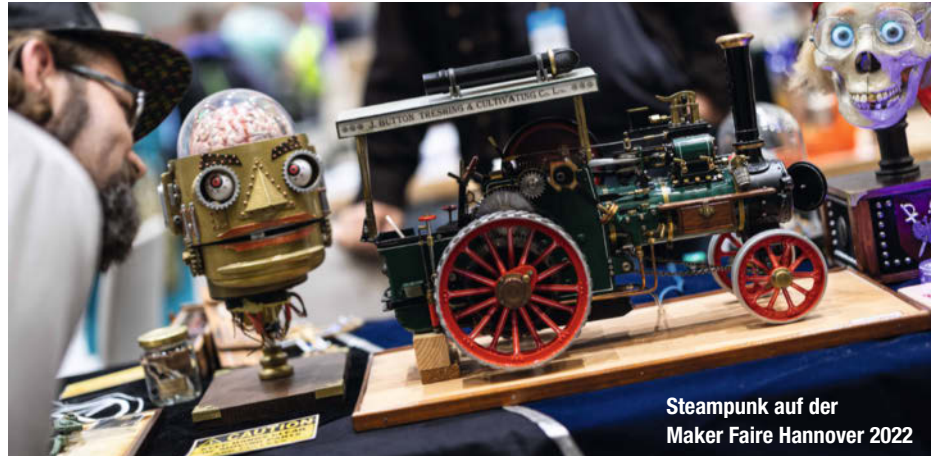
Treffpunkt der Community

Die Maker Faire Hannover ist wie jedes Jahr der Ort, an dem sich die deutsche Maker- und Maker-Influencer-Community trifft. Präsentiert werden Projekte zu verschiedenen Themen wie 3D-Druck, Robotik, Programmierung, Upcycling, Arbeiten mit CNC-Fräsen und Lasercuttern, Cosplay, innovativem Handwerk, Lichtkunst in der Dark Gallery und vielem mehr. Was kannst du der Welt präsentieren?

Nur dank euch gibt es die Maker Faire, nur durch euch wird sie lebendig, nur durch euch wird unsere gemeinsame Sache größer, denkwürdiger und nachhaltiger. Hier bist du Teil der Förderung von Austausch, Gedanken und Ideen. Begeisterung und Inspiration gleichermaßen. Nutze auch du die Gelegenheit, dich mit anderen Makern, Unternehmen, Start-ups, Künstlern und Kreativen aus ganz Deutschland und der Welt in Präsenz zu vernetzen. So kannst du Teil der Community und der Zukunft werden.

Jetzt anmelden!

Unser Aufruf richtet sich an alle, die eine brillante Idee oder ein fertiges Projekt in der Werkstatt stehen haben, welches sie in diesem Sommer im August tausenden Besuchern der Maker Faire Hannover präsentieren möchten.



Steampunk auf der Maker Faire Hannover 2022

Jetzt ist eure letzte Chance dabei zu sein: **Am 11. Juni endet unser „Call for Makers“.**

Du fragst dich, was du tun musst, um kostenfrei teilzunehmen? Füll ganz einfach über die Kurz-URL unten unser „Call for Makers“-Formular auf der Maker-Faire-Homepage aus, sag uns was du brauchst, und wir kümmern uns um den Rest!

Hast du noch weitere Fragen? Dann schau auf unserer Homepage bei unseren FAQs vorbei oder kontaktiere uns direkt unter: leou@maker-faire.de

Wenn du dich angemeldet hast, melden wir uns mit Informationen zu deiner Standfläche, dem Hallenplan und dem Ablauf vor Ort bis Ende Juni bei dir. Werde Teil einer großartigen Gemeinschaft, erlebe viel Inspiration, unser Aussteller Get-Together am Samstagabend und zwei Tage lang Spaß und Begeisterung mit unseren Besuchern! Wir freuen uns auf dich!

—pek

► make-magazin.de/xmub



Dein Einstieg in Embedded Systems mit RISC-V und ESP32-C3

Neu und groß im Kommen – RISC-V ist eine auf freier Technologie basierende Befehlsatzarchitektur. Dieses Buch vermittelt dir grundlegende Konzepte und Funktionsweisen von Mikrocontrollern. Schritt-für-Schritt-Anleitungen und zahlreiche Abbildungen führen dich anschaulich an das Thema heran, Hands-On-Projekte helfen dir, das Gelernte zu vertiefen und bereiten dich auf das selbstständige Programmieren mit Mikrocontrollern vor. Zum Einsatz kommt das Entwicklungsboard ESP32-C3. Vorkenntnisse in der Programmiersprache C und RISC V-Assembler erleichtern die Umsetzung der Übungen.

2023 · 346 Seiten · 39,90 €
ISBN 978-3-86490-937-5



2023 · 236 Seiten · 24,90 €
ISBN 978-3-86490-914-6



2023 · 282 Seiten · 29,90 €
ISBN 978-3-86490-913-9



3. Auflage · 2022 · 366 Seiten · 36,90 €
ISBN 978-3-86490-869-9

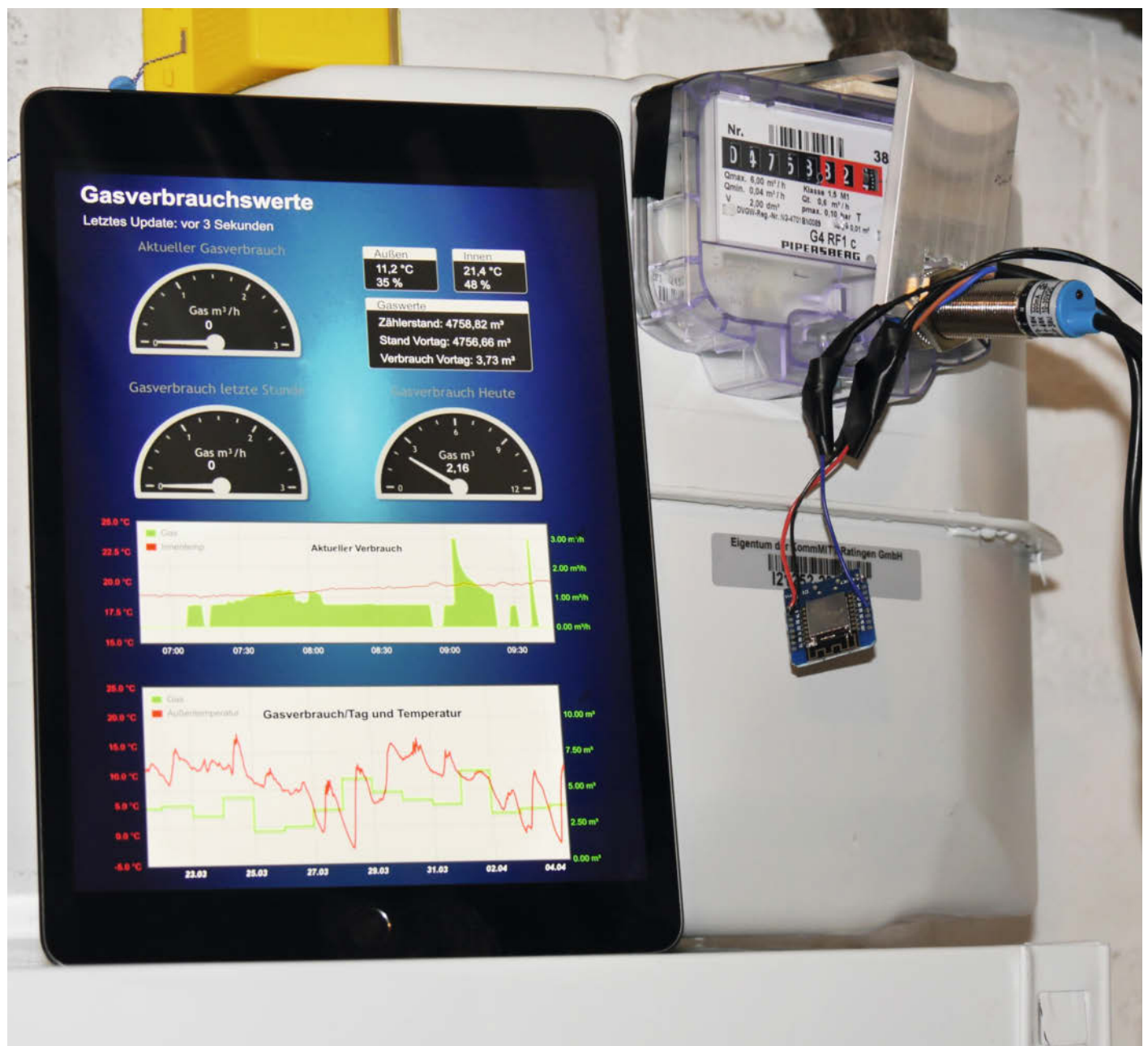


dpunkt.verlag

Smarter Gaszähler mit Smarthome-Integration

Aufgrund der angespannten Energiesituation ist es sinnvoll, den aktuellen Gasverbrauch zu messen und eine direkte Rückmeldung über die Wirkung von Energiesparmaßnahmen zu erhalten. Um nicht täglich den Gaszähler ablesen zu müssen, wurde ein Smart Meter mit Web-Interface und Datenspeicherung entwickelt, das kommerziellen Produkten in nichts nachsteht.

von Andreas Koritnik



Viele Gaszähler sind bereits für den Anschluss einer elektronischen Verbrauchsmessung vorbereitet. Im einfachsten Fall genügt ein einfacher Magnetschalter (Reedkontakt), der durch einen rotierenden Magneten im Gaszähler betätigt wird. Bei Gaszählern ohne rotierenden Magneten wird ein induktiver Näherungsschalter verwendet, der auf ein rotierendes Metallplättchen im Anzeigefeld des Zählers reagiert. Eine Umdrehung (Magnet oder Plättchen) entspricht einem bestimmten Verbrauch (bei meinem Zähler 0,01 Kubikmeter).

Welche Variante vorliegt, muss man durch eine Web-Suche mit der Hersteller-/Typbezeichnung selbst herausbekommen oder eventuell mit einem Reed-Kontakt ausprobieren. Ich habe einen Gaszähler vom Hersteller *Pipersberg* (Typ: *G4 RF1c*). Dieser Zählertyp ist für das Auslesen mittels induktivem Sensor vorbereitet (*Cyble Sensor Encoder*). Der Hersteller bietet dafür unterschiedliche Lösungen an (433 MHz, *Wireless Mbus*, *Open Collector*).

Die Nutzung der teuren Hersteller-Encoder ist aber für Maker nicht notwendig. Ein ESP-8266-Mikrocontroller, der passende Sensor und eine Stromversorgung reichen völlig für die Realisierung eines eigenen Projekts. Damit hat man dann ein schönes Smart Meter mit Web-Oberfläche zur Bedienung und zum Ablesen der Verbrauchswerte. Noch eleganter ist aber die Integration in eine Smarthome-Lösung wie *ioBroker* auf z. B. einem Raspberry Pi inklusive Archivierung und Visualisierung. Beide Ausbaustufen werden in diesem Artikel beschrieben.

Hardware

Die Hardware ist, wie in der Schaltung ① zu sehen, relativ schnell aufgebaut. Im einfachsten Fall bei einem Zähler mit rotierendem Magnet werden lediglich eine 5-Volt-Stromversorgung, der Reedkontakt und zwei Kabel benötigt. Ein Reedkontakt ist ein einfacher magnetischer Schalter, der in einem Magnetfeld schließt und ohne Magnetfeld wieder öffnet.

Kurzinfo

- » ESP basierend, Anzeige der Verbrauchswerte über Web-Interface
- » Übertragen der Verbrauchswerte mit MQTT z. B. an ioBroker
- » Visualisierung und Speicherung der Verbrauchsdaten

Checkliste



Zeitaufwand:
2 Stunden



Kosten:
20 Euro (ohne Raspi)

Material

- » ESP8266 D1 Mini
- » LJ18A3-8-Z/BX-Sensor oder Reed-Kontakt
- » Netzteil 12 Volt oder 5 Volt (bei Reedkontakt)
- » Step-Down-Wandler MP1584EN, auf 5V, bei Induktivsensor
- » Kabel
- » Gehäuse für ESP und ggfs. Raspi
- » Raspberry Pi ab Version 3B, für ioBroker

Werkzeug

- » Lötutensilien
- » Multimeter
- » Werkzeug zum Basteln einer Sensorbefestigung

Mehr zum Thema

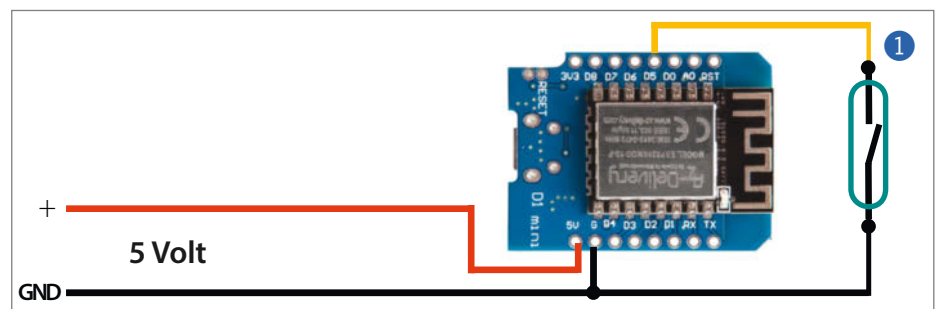
- » Josef Müller, ESP32CAM liest Wasseruhr mit KI, Make 2/21, S. 14
- » Andreas Hitzig, Smart-Home zentral steuern mit Raspberry Pi und ioBroker (heise+Ab erforderlich)

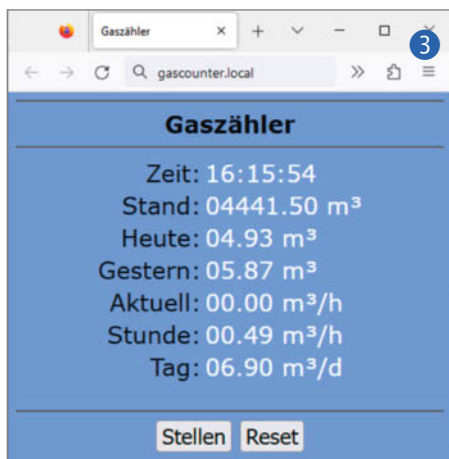
Alles zum Artikel
im Web unter
make-magazin.de/xrte



Die Spannungsversorgung des ESP kann entweder über die USB-Buchse oder direkt durch Anschluss eines Netzteils an die Pins 5V und GND erfolgen. Der Anschluss des Reedkontaktes erfolgt an den Pins GND und D5. Die Polung spielt hierbei keine Rolle.

Soll ein Zähler, der einen induktiven Sensor benötigt, abgefragt werden, ist etwas mehr Aufwand erforderlich ②. Der Sensor benötigt je nach Typ mindestens 6 oder sogar 12 Volt. Eine 12-Volt-Versorgung reicht aber in jedem Fall aus, 500mA Stromstärke sind genug, mehr





schadet aber auch nicht. Der Sensor wird an 12V (braun) und GND (blau!) angeschlossen. Das 12V-Netzteil versorgt auch den Step-Down-Spannungswandler, der dann die 5V für den ESP erzeugt.

In meinem Bauvorschlag verwende ich die auf meiner GitHub-Seite aufgelisteten Bauteile (siehe Kurzlink) und einen kleinen regelbaren Spannungswandler, dessen Ausgangsspannung vor dem Anschluss an den ESP8266 auf fünf Volt eingestellt werden muss! Das geht ganz einfach mit einem kleinen Schraubendreher und einem Multimeter. Das kleine Potentiometer wird so eingestellt, dass am Ausgang 5 Volt anliegen. Dann kann der Ausgang des Spannungswandlers mit den Pins 5V und GND des D1 Mini verbunden und die Schaltung in Betrieb genommen werden.

Der Ausgang des Sensors (schwarz) wird dann mit Pin D5 verbunden. Mehr ist nicht erforderlich. Durch Konfiguration in der Software wird Pin D5 mit einem internen Pull-Up-Widerstand beschaltet. Entweder durch den Reedkontakt oder den *Open-Collector*-Ausgang des Sensors wird der Pin bei jedem Impuls nach GND gezogen. Die Software zählt dann einfach die Impulse und berechnet die Verbrauchswerte.

Einstellungen

```
#define USE_MQTT          true // Set to false if no MQTT to be used

#define TIME_24_HOUR      true
#define NTP_SERVER "de.pool.ntp.org"
// Western European Time
#define TZ_INFO "WEST-1DWEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00"

// Wifi
const char *ssid = "ssid"; // Set WLAN name
const char *password = "password"; // Set password

// MQTT
#if USE_MQTT == true
WiFiClient Mqtt;
PubSubClient client(Mqtt);
const char* mqtt_server = "192.168.0.71";
#endif
```

Interrupt-Behandlung

```
void IRAM_ATTR handleInterrupt() {
  noInterrupts();

  if (millis() - Last_int_time > 100) { // Entprellen 100 ms

    DeltaCounter += GasDelta; // Gas COUNT hochzählen
    DayCounter += GasDelta;
    GasCounter += GasDelta;

    unsigned long TempVal = millis(); // Zeit für Mittelwert
    PeriodCount = TempVal - StartValue;
    StartValue = TempVal;
    Last_int_time = millis();
  }
  interrupts();
}

// aus Setup()
...
pinMode(Gas_Counter_Pin, INPUT_PULLUP);
attachInterrupt(digitalPinToInterrupt(Gas_Counter_Pin),
                handleInterrupt, FALLING);
...
```

Software

Das komplette Programm ist auf meiner GitHub-Seite verfügbar (Link siehe Kurzinfo). Zum Programmieren wird eine aktuelle Arduino-IDE (1.8.x oder 2.1.x) benötigt. In den Voreinstellungen der IDE Version 1.8.x muss die URL für den ESP8266 zu den Board-Administrator-URLs hinzugefügt werden. Ab der Version 2.x werden die ESPs bereits direkt unterstützt.

Zusätzlich müssen zwei weitere Bibliotheken installiert werden. Dies geht ganz einfach mit dem Bibliotheksmanager (*Werkzeuge/Bibliotheken verwalten...*). Hier einfach nacheinander die folgenden Bibliotheken durch Eingabe der Namen suchen und installieren:

- ArduinoJson
- PubSubClient (von Nick O'Leary)
- die weiteren Bibliotheken aus den `#includes` sollten bereits installiert sein.

Damit ist die IDE vorbereitet und das Programm kann geladen werden. Dazu das gesamte Repository von GitHub als ZIP-Datei herunterladen und in ein lokales Verzeichnis entpacken. Dann mit der IDE im Ordner *Gascounter* die Datei *Gascounter.ino* öffnen.

In den ersten Zeilen des Programms (Kasten *Einstellungen*) müssen einige Anpassungen vorgenommen werden. Der Wert für `USE_MQTT` legt fest, ob MQTT zum Senden von Daten an *ioBroker* verwendet werden soll. Falls keine Integration gewünscht wird, bitte `true` durch `false` ersetzen. Die entsprechenden Programmteile werden dann ausgeblendet (Stichwort: bedingte Kompilierung, `#if/#endif`).

Das Programm holt sich die aktuelle Zeit per NTP (**N**etwork **T**ime **P**rotocol) aus dem Internet. Die aktuelle Einstellung ist für Deutschland und berücksichtigt die Sommer-/Winterzeitumstellung.

Ab dem Kommentar `//Wifi` wird die SSID und das Passwort für das lokale WLAN eingetragen. Bei Verwendung von MQTT muss zusätzlich die IP-Adresse des Raspberry Pi unter `mqtt_server` eingetragen werden.

Damit sind die Anpassungen abgeschlossen. Das so geänderte Programm kann nun gespeichert und per USB auf den angeschlossenen ESP8266 übertragen werden. Dazu muss zuvor in der IDE unter *Werkzeuge* der USB-Port und das Board *LOLIN(WEMOS) D1R2 & mini* angegeben werden. Spätere Änderungen können auch über WLAN erfolgen (Stichwort: OTA, **O**ver **T**he **A**ir). Dabei wird in der Arduino IDE nicht der serielle USB-Port ausgewählt, sondern der angezeigte Name (Gascounter) und die IP-Adresse.

Web-Interface

Nachdem das Programm gestartet wurde, kann mit einem Webbrowser auf den Gas-

zähler zugegriffen werden. Sofern der Client *mDNS (Bonjour)* unterstützt, z. B. ein iPhone/iPad oder ein aktuelles Android-Gerät, kann der Zugriff durch Eingabe von *Gascounter.local* in der Adresszeile des Browsers erfolgen. Falls der Client kein mDNS unterstützt, genügt es, die in der Arduino-IDE angezeigte IP-Adresse einzugeben. Alternativ kann sie auch über die Konfigurationsmenüs des Routers in Erfahrung gebracht oder eine feste Adresse vergeben werden.

Im Client sollte nun die Weboberfläche wie in 3 angezeigt werden. Der aktuelle Gaszählerstand ist allerdings noch null. Wenn später alles am Gaszähler montiert ist, kann mit dem Button **Setzen** einmalig der aktuelle Zählerstand eingegeben werden (bitte mit Punkt statt Komma). Von nun an zählt der Gaszähler alle Impulse und errechnet die Verbrauchswerte. Die einzelnen Anzeigen haben folgende Bedeutung:

Zeit zeigt die aktuelle Uhrzeit. *Stand* zeigt den aktuellen Zählerstand, *Heute* den kumulierten Verbrauch des aktuellen Tages, *Gestern* den Gesamtverbrauch des Vortages, *Aktuell* den aktuellen Verbrauch in Kubikmeter pro Stunde. *Stunde* zeigt den Verbrauch der letzten Stunde und *Tag* den durchschnittlichen Verbrauch pro Tag seit dem Einschalten des Zählers.

Impulsmessung mit Interrupt

Die Messung der Impulse vom Sensor erfolgt über die Interrupt-Funktion des ESP8266 (Kasten *Interrupt-Behandlung*). Dazu wird im Programm die Funktion `handleInterrupt()` verwendet.

In der Funktion `setup()` wird Pin *D5* als `INPUT_PULLUP` definiert und die Interrupt-Behandlung für diesen Pin mit der Funktion `handleInterrupt()` aktiviert. Mit `FALLING` wird festgelegt, dass die Funktion beim Wechsel von HIGH nach LOW aufgerufen wird.

In der Funktion `handleInterrupt()` werden nun bei jedem Impuls die Verbrauchswerte hochgezählt und die Zeit zwischen zwei Impulsen gemessen. Um zu verhindern, dass bei einem Prellen des Reedkontaktes mehrere Impulse gezählt werden, wird nach einem



Interrupt mindestens 100ms gewartet, bis der nächste Impuls gezählt wird.

Montage des Sensors

Je nach Ausführung des Gaszählers und der Funktionsweise der Übertragung ist der Einbau des Sensors unterschiedlich kompliziert. In der Regel wird ein Reedkontakt an der Unterseite des Anzeigefeldes des Zählers montiert. Im Internet finden sich für viele Modelle auch Angaben zur Position und teilweise auch Informationen zum 3D-Druck eines geeigneten Gehäuses.

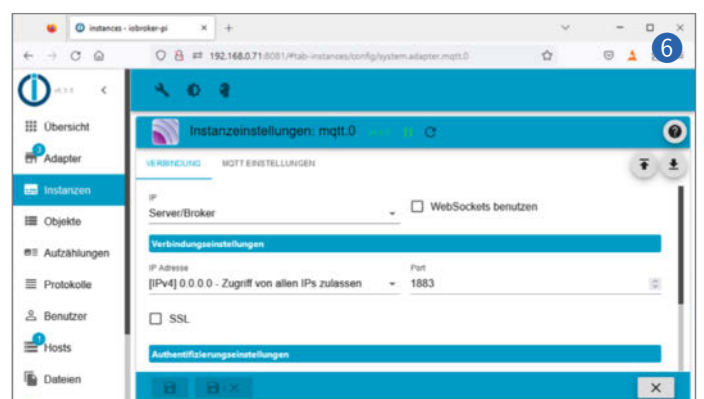
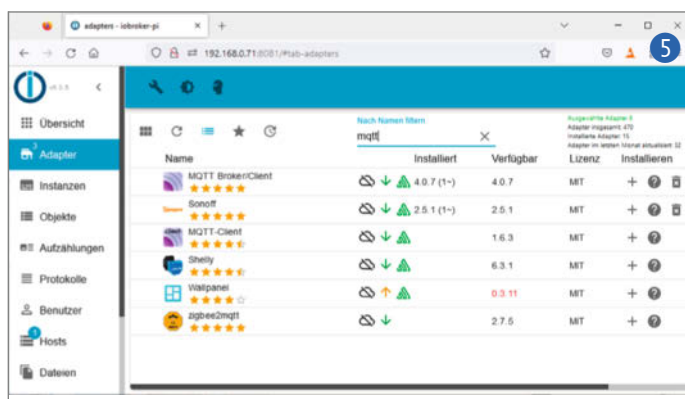
In meinem Fall war es jedoch notwendig, einen induktiven Sensor zu verwenden. Zur Montage 4 genügt ein einfacher Winkel (Kunststoff oder Aluminium). In ein Ende des Winkels wird ein Loch mit dem Durchmesser des Sensors gebohrt. Nun wird der Sensor eingesetzt und mit den mitgelieferten Muttern festgeschraubt. Dann wird der Winkel mit dem Sensor so auf dem Zählwerk montiert, sodass das Sensorfeld zur Hälfte auf das sich drehende Zählwerk zeigt. Die Position ist richtig,

wenn die rote LED des Sensors bei jeder Umdrehung einmal aufleuchtet und wieder erlischt. Ich habe die Montage des Winkels am Zählwerk einfach mit Klebeband gelöst. Es geht sicher eleganter, aber für mich war es der einfachste Weg.

Wenn alles installiert ist und funktioniert, kann der aktuelle Zählerstand einmalig über den Web-Client eingegeben werden. So hat man jetzt einen intelligenten Gaszähler mit Fernablesung und ein paar Verbrauchsdaten. Am Anfang hat mir diese Lösung auch gereicht. Aber nach einiger Zeit und täglicher Eingabe der Verbrauchsdaten in Excel kam der Wunsch nach einer automatischen Speicherung und Visualisierung der Verbrauchsdaten auf.

Integration in ioBroker

Da ich noch einen unbenutzten Raspberry Pi herumliegen hatte, habe ich mich auf die Suche nach geeigneter Software gemacht und bin schließlich auf das Programm *ioBroker* gestoßen (Links in der Kurzinfor). Das ist eine



Open-Source-Smarthome-Umgebung, die die Integration zahlreicher Komponenten und Sensoren ermöglicht und läuft lokal ohne Cloudzwang. Damit ioBroker flüssig läuft, wird mindestens ein Raspberry 3B benötigt. Neben der Open-Source-Version gibt es auch eine Pro-Version mit erweiterter Cloud-Integration, wenn die Anforderungen steigen oder die Installation im kommerziellen Umfeld eingesetzt werden soll.

Unter anderem wird auch *MQTT* unterstützt. MQTT (**M**essage **Q**ueuing **T**elemetry **T**ransport) ist ein Protokoll für die Maschine-zu-Maschine-Kommunikation und arbeitet nach dem Publisher/Subscriber-Prinzip. Für den Gaszähler verwende ich ioBroker als MQTT-Server. Der ESP8266 wird als Client (Publisher) betrieben.

MQTT wird auch von anderen Smarthome-Programmen wie z. B. *Home Assistant* unterstützt.

Falls bereits eine Smarthome-Lösung im Einsatz ist, können die Daten natürlich auch über MQTT in diese Lösungen integriert werden.

Das Senden eines Wertes per MQTT in das Programm ist dank der Bibliothek *PubSubClient* sehr einfach:

```
snprintf(Text, sizeof(Text), "%8.2f",
        GasCounter);
client.publish("Count", Text);
```

Zuerst wird der zu sendende Wert als Text in einem Array formatiert, gespeichert und dann einfach mit `client.publish()` gesendet, wobei sowohl der Name (Count) als auch der formatierte Wert übergeben werden.

Um ioBroker auf den Raspberry zu bekommen, gibt es zwei Möglichkeiten: Per Image auf einen frischen Raspi oder per Skript auf einen bereits laufenden Raspi. Alternativ gibt es auch Windows-Installer und Docker-Container.

Wenn bereits ein funktionsfähiger, aktueller Raspberry im lokalen Netzwerk vorhanden ist, kann das Programm einfach mit dem Befehl

```
curl -sLf https://iobroker.net/
install.sh | bash -
```

installiert werden. Die Installation kann einige Zeit in Anspruch nehmen! Eventuell müssen noch weitere Softwarepakete installiert werden, um ein funktionsfähiges System zu bekommen. Wer mit Linux nicht so vertraut ist (apt und bash), dem empfehle ich die Variante mit dem fertigen Image. Damit funktioniert es auf jeden Fall wie hier beschrieben.

Wenn Sie sich für eine Komplettinstallation entscheiden, wird das Image ganz normal auf die SD-Karte geschrieben. Zum Beispiel mit dem *win32diskimager* oder dem *Raspi Imager* (Links siehe Kurzinfor). Damit landet ein für ioBroker optimiertes System auf dem Raspi, das natürlich ganz normal mit anderen Diensten erweitert werden kann. Die Vorgehensweise ist wie folgt:

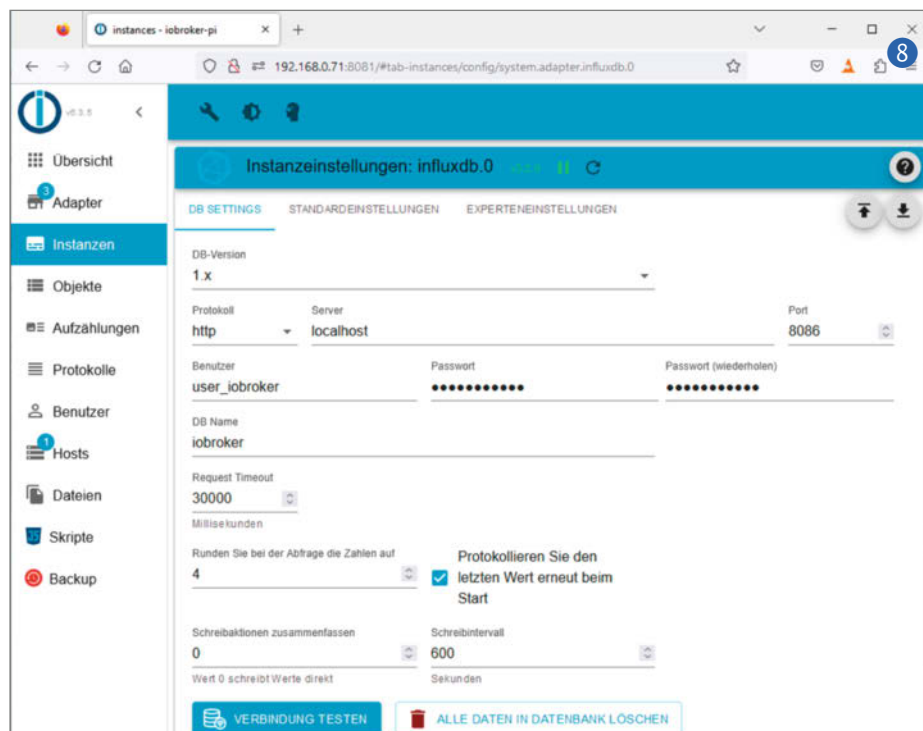
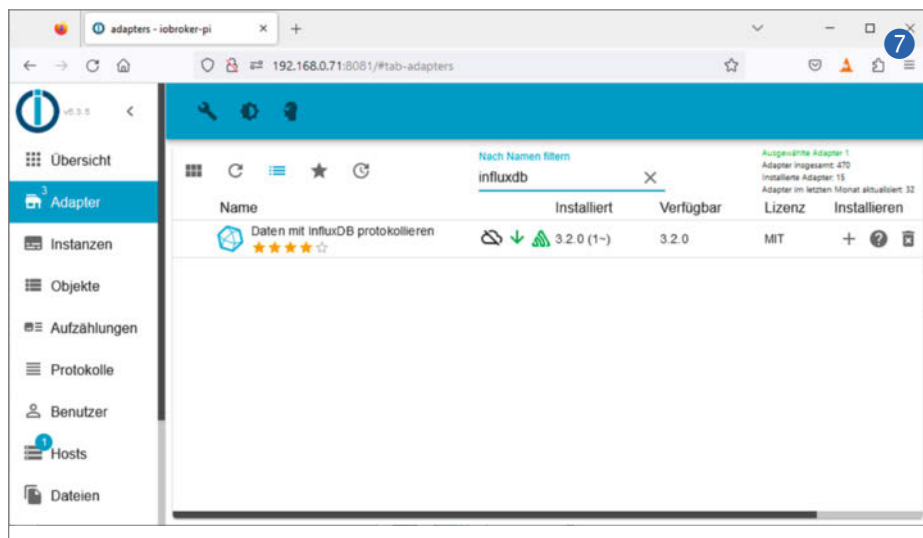
- Raspberry an Tastatur und Monitor anschließen
- die mit ioBroker beschriebene SD-Karte einstecken und den Raspberry mit Strom versorgen
- Einloggen und Benutzername/Passwort ändern
- Konfiguration des Raspberry mit `sudo raspi-config` starten.
- WLAN in *System Options* und Landeseinstellungen in *Localisation Options* konfigurieren.
- Bei Bedarf (sehr zu empfehlen) in *Interface Options* SSH erlauben. Mit SSH kann man später auch ohne Tastatur und Bildschirm auf den Raspberry zugreifen (z.B. mit *Putty*).
- Nach einem Neustart (`sudo reboot`) neu einloggen.

Jetzt aktualisieren wir das System mit den beiden Befehlen:

```
sudo apt-get update
sudo apt-get full-upgrade
```

Erstellen der Datenbank

```
CREATE USER "admin" WITH PASSWORD '{pwd}' WITH ALL PRIVILEGES
CREATE USER "user_iobroker" WITH PASSWORD '{pwd}'
CREATE DATABASE "iobroker"
GRANT ALL ON "iobroker" TO "user_iobroker"
```



ID	Raum	Funktion	Wert	Einstellun...
0_userdata				
admin				
alias				
discovery				
influxdb				
mqtt				
0				
info				
Count			4509.29	
Current			0	
DayEndCount			4505.7	
Hour			0.39	
Today			3.59	
Yesterday			7.56	

Dann mit dem Befehl `ifconfig` die IP-Adresse des Raspberry auslesen.

Auf dem PC im Browser folgende URL eingeben:

Raspi-IP-Adresse:8081

Damit wird ioBroker aufgerufen. Nun den Assistenten starten, die Lizenzbedingungen akzeptieren (die Statistiken können später noch deaktiviert werden) und dann das Admin-Passwort setzen. Zum Schutz kann auf der folgenden Seite *Authentifizierung für Admin* aktiviert werden. Das Erkennen (Discover) von lokalen Geräten wird hier nicht benötigt und mit *Schließen* beendet.

Im folgenden Fenster können weitere Einstellungen vorgenommen werden. Danach muss man sich als *admin* einloggen.

Bei der Installation von einem Image sollte im Adapter-Menü mit dem Pfeilkringel-Symbol nach Updates gesucht werden. Nach einer Installation mit `curl` ist dies nicht nötig, da `curl` immer die aktuellsten Pakete lädt.

Nun wird der MQTT-Adapter installiert. Dazu `mqtt` bei *Nach Name filtern* 5 eingeben und dann den Adapter *MQTT Broker/Client* suchen. Je nach Ansicht (Liste oder Icons) sehen Sie nun direkt den +-Button bei den Adaptern. In der Icon-Ansicht die drei Punkte anklicken, dann erscheint ein +-Button, mit dem der Adapter installiert und anschließend konfiguriert wird 6.

Wählen Sie *Verbindung/Server/Broker*. Dann speichern und Fenster schließen. Nun den Dienst starten (Play-Icon). Damit sind ioBroker und MQTT betriebsbereit.

Nun muss noch die Datenbank *influxdb* installiert werden, damit die empfangenen Gasverbrauchswerte auch dauerhaft gespei-

chert werden können. Dazu loggt man sich in den Raspberry ein (entweder direkt per Tastatur oder per SSH über LAN/WLAN).

Danach wird *influxdb* und der *influxdb-client* installiert:

```
sudo apt install influxdb
sudo apt install influxdb-client
```

Nun wird das Programm mit *influx* gestartet und eine Datenbank mit dem Namen *iobroker* und den Benutzern *admin* und *user_iobroker* angelegt. Dazu wird Zeile für Zeile der Inhalt des Kastens *Erstellen der Datenbank* eingegeben und jeweils mit Enter bestätigt.

Bitte {pwd} durch selbstgewählte Passwörter ersetzen! Danach das Programm mit *exit* verlassen.

Nachdem die Datenbank auf diese Weise erstellt wurde, müssen die Einstellungen für Influx angepasst werden. Dazu wird der nano-Editor aufgerufen:

```
sudo nano /etc/influxdb/influxdb.conf.
```

Im Konfigurationsbereich *http* folgende Einstellungen anpassen, dazu ggfs. Kommentarzeichen (#) löschen und die Werte ändern:

```
[http]
enabled = true
bind-address = ":8086"
auth-enabled = true
log-enabled = true
write-tracing = false
pprof-enabled = false
https-enabled = false
```

Dann *nano* mit *Strg-X* verlassen und die Änderungen speichern (Y=Yes, dann *RETURN*). Die Datenbank neu starten:

```
sudo service influxdb restart
```

Hier kommt Verstärkung



Das **Make-Sonderheft** bietet einen praxisorientierten Einstieg in Schaltungen mit Operationsverstärkern inkl. Experimentierset.

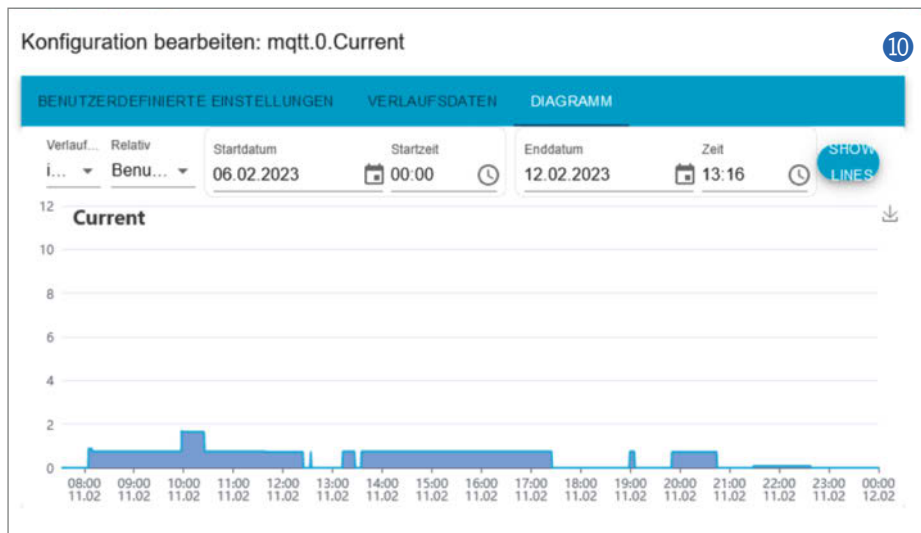
Will man Sensorsignale verarbeiten oder verstärken, Spannungen überwachen oder Audiosignale filtern: Mit geringem Aufwand und ohne komplizierte Berechnungen setzt man Operationsverstärker ein. Das Heft erklärt, wie alle Schaltungen funktionieren.

- ▶ Operationsverstärker verstehen
- ▶ Komparatoren und Schmitt-Trigger erklärt
- ▶ Spannungsversorgungen und virtuelle Masse
- ▶ Schaltungen selbst entwerfen und berechnen
- ▶ Viele praktische Anwendungen
- ▶ Inklusive Experimentierset Operationsverstärker

Heft + Experimentierset für nur 49,95 €

shop.heise.de/make-opv

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



Wieder per Browser vom PC folgt die Integration von influxdb in ioBroker. Dazu im Menü wieder *Adapter* anwählen 7.

Im Eingabefeld *nach Namen filtern* geben wir *influxdb* ein und installieren dann mit dem +-Icon (bei Icon Ansicht im drei Punkte Menü) den angezeigten Adapter *Dateien mit influxdb protokollieren*.

Nun wird das Modul *influxdb.0* direkt konfiguriert 8. Als Passwort das für den *user_iobroker* gewählte Passwort eingeben. Dann die Verbindung zum Server testen und bei OK die Einstellungen speichern und schließen.

Das war es dann auch schon mit der Installation von ioBroker und der Datenbank. Die per MQTT gesendeten Werte des Gaszählers werden nun empfangen und in der Datenbank 9 gespeichert, wie im Reiter *Objekte* zu sehen ist: Neu ankommende Werte leuchten kurz grün auf.

Ich habe in ioBroker die Werte *Current*, *DayEndCount* und den Zählerstand *Yesterday*

zum Speichern in der Datenbank aktiviert. Um diese oder andere Werte zu aktivieren, klicken Sie auf das Zahnrad rechts in der Zeile und wählen Sie die Checkbox *Aktivieren* im neuen Fenster an. Als Indikator ist das Zahnradsymbol danach blau und nicht mehr grau.

Die Standardeinstellungen werden beibehalten (nur Änderungen werden in der Datenbank gespeichert). Da sich der Wert für gestern (*Yesterday*) nur einmal am Tag ändert, ergibt sich in der Datenbank eine Tagesverbrauchshistorie. Diese hätte auch aus den einzelnen Zählerständen berechnet werden können. So ist es aber einfacher ohne komplizierte Datenbankabfragen. Die Speicherung des aktuellen Verbrauchs (*Current*) ergibt eine detaillierte Verbrauchshistorie 10 in m³/h über den Tag.

Die Einstellungen können natürlich den eigenen Wünschen angepasst werden, die Grafik wird live aktualisiert, sobald die Daten eintreffen. Über den Menüpunkt *VERLAUFS-*

DATEN können die Daten als Tabelle im .csv-Format heruntergeladen werden. Damit hat man schon mal einen Überblick, nur die Diagramme könnten etwas übersichtlicher und schöner sein.

Schönere Diagramme

Ansprechendere Visualisierungen können mit der internen Visualisierungskomponente *VIS* von ioBroker in Verbindung mit dem Adapter *Flot* erstellt werden. *VIS* für ioBroker erfordert einen Lizenzschlüssel, für den privaten Gebrauch ist die Nutzung jedoch kostenlos. Einfach bei iobroker.net (nicht .com!) mit der E-Mail-Adresse registrieren und unter *Cloud/Free/Pricing/Community Licenses iobroker.vis* für 0,00 EUR bestellen. Dann den gelieferten Schlüssel in die Instanz-Konfiguration für *vis.0* unter *Lizenz* kopieren. Das war's.

Wie in 11 zu sehen, habe ich meine Gasverbrauchswerte noch um die Temperatur und Luftfeuchtigkeit für außen und innen ergänzt. Das obere Diagramm zeigt den Verlauf des aktuellen Verbrauchs und der Innentemperatur. Das untere den Tagesverbrauch und die Außentemperatur. Die Diagramme sind übrigens interaktiv und sowohl Scrollen/Zoomen als auch die Anzeige der exakten Werte wird unterstützt.

Die Bedienung des Visualisierungseditors ist komplex, daher stelle ich die exportierte Visualisierung auf meiner GitHub-Seite zum Download bereit. Vor dem Import mit dem *VIS Editor* müssen allerdings die beiden Adapter *VIS* und *Flot* in ioBroker installiert sein. Dabei bitte genauso vorgehen wie bei der Installation von MQTT und Influx. Einfach bei Adapter im Feld *Nach Name filtern* für *VIS* erst *Visualisierung* suchen und für den Adapter *Flot* eingeben. Eine Konfiguration der beiden Adapter ist nicht notwendig.

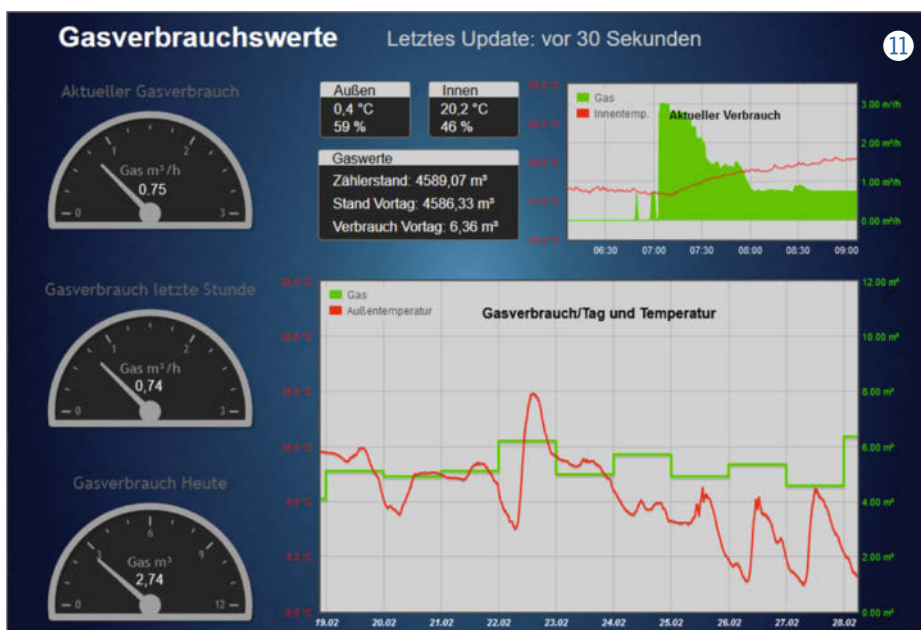
Nach dem Download der Datei *VIS-Project.zip* einfach in ioBroker den VIS-Editor starten und im Menü *Setup/Projekt-Export/Import/Import* auswählen. Den Projektnamen *Gasverbrauch* angeben und die heruntergeladene Datei *VIS-Project.zip* in das Fenster hineinziehen oder die Datei manuell auswählen. Das war es schon. Der Aufruf der Visualisierung im Browser erfolgt über die URL in dieser Form:

Raspi-IP-Adresse:8082/vis/index.html?Gasverbrauch#GasView_1

Es gibt zwei verschiedene Visualisierungen: Eine im Querformat (*GasView_l*) und eine im Portrait-Format (*GasView_p*). Der Browser sollte selbständig die passende Version, je nach Fensterbreite, anzeigen.

Übrigens funktionieren auch viele Wasserzähler mit magnetischen oder induktiven Sensoren. So kann man auch seinen Wasserverbrauch im Auge behalten. Viel Spaß beim Nachbauen!

—caw



Online-Artikel: Transistorklang, E-Paper-Kalender, Maker in der Ukraine

Unter HiFi-Gourmets ist der spezielle Röhrenklang von Verstärkern ein unerschöpfliches Thema – mit seinem nicht gerade alltäglichen Audio-Verstärker-Konzept zeigt Make-Autor Walter Orlov, dass es auch einen spezifischen und durchaus angenehmen **Transistorklang** geben kann. Sein Trick: Die Schaltung hat durch die reine Stromsteuerung einen sehr hohen Ausgangswiderstand, sodass Lautsprecher eher ein Eigenleben führen können und nur wenig bedämpft werden. Das wünscht man sich zum Beispiel für den typischen vollrunden 1960er-Musikbox-Klang in Verbindung mit offen (nur auf Schallwand) betriebenen, eher hart aufgehängten Lautsprechern mit großer Membran und hohem Wirkungsgrad. Hierfür reicht auch eine eher bescheidene Ausgangsleistung.

Der von ihm entworfene A-Klasse-Verstärker wird mit einem 19-Volt-Notebook-Netzteil



Röhren kann jeder – bei diesem Audio-Verstärker prägen die Transistoren den Klang.

betrieben und bezieht die Endtransistoren nicht in die Gegenkopplungsschleife ein. Auf diese Weise arbeiten sie frei, sodass ihre Stromkennlinien den Klang prägen – so entsteht der eingangs erwähnte, spezielle Transistorklang. Nichtsdestotrotz erlaubt die Parallelschaltung der Endtransistoren eine hohe Linearität des Verstärkers zu gewährleisten. Der Klirrfaktor bleibt bis zur Übersteuerungsgrenze unter 1 Prozent. Einen ersten kurzen Artikel dazu gibt es für Make- und heise+-Abonnenten (siehe unten auf dieser Seite) über die Kurz-URL online zu lesen, einen ausführlichen Bericht bringen wir dann in einem der kommenden Hefte.

Einen ausgedienten Kindle-E-Book Reader der vierten Generation fand Make-Autor Stefan Südfeld zu schade, um ihn zu entsorgen: Der ist stromsparend und sein **E-Ink-Display** ist groß und gut ablesbar. Zudem verfügt er über einen Webbrowser zur Darstellung von Webseiten (HTML mit CSS und JavaScript). Solche Geräte eignen sich somit gut für die Präsentation von Informationen, die sich nur selten ändern – also zum Beispiel zur Anzeige von Kalenderdaten. So wurde die Idee geboren, dem E-Book-Reader die neue Aufgabe zu übertragen, den **Familienkalender** bei Google unaufdringlich, aber stets sichtbar anzuzeigen.

Welche Lösungen Stefan Südfeld am Ende fand, die gewisse Einschränkungen des integrierten Browsers umgehen und dennoch ein Mindestmaß an Sicherheit garantieren, können Make- und heise+-Abonnenten ebenfalls online im Detail nachlesen.



Doch, das ist ein FPV-Copter. Und das sind Granaten aus dem 3D-Drucker. Das ist Maker-Alltag in der Ukraine.

Der seit über einem Jahr tobende Krieg bestimmt auch für die **Maker in der Ukraine** ihr Leben und hat ihre Projekte fundamental verändert: Mit 3D-Druckern werden jetzt ebenso Flugstabilisatoren für Sprengkörper wie Tourniquets für die Wundversorgung oder provisorische Ersatzteile für Autos gefertigt, so berichtet es Yuriy Vlasjuk, der zusammen mit seiner Frau Svitlana Bovkun vor dem Krieg die Maker Faires in der Ukraine organisiert hat. Consumer-Drohnen werden zu Kamikaze-FPV-Fluggeräten umfunktioniert und wer schweißen kann baut simple Holzöfchen zum Kochen, Panzersperren oder Lastenfahrräder als Behelfs-Transportmittel. Neben Projekten zur Verteidigung beteiligen sich die Maker im Land aber auch an Vorhaben, die bei der Reparatur von zerstörter Infrastruktur und beim Wiederaufbau des Landes helfen sollen – den ausführlichen Bericht gibt es für Abonnenten online über den Link zu lesen. —pek

► make-magazin.de/xcgd

Für Abonnenten: Mehr Make-Artikel online



Schon länger erscheinen Make-Artikel nicht nur gedruckt im Heft, sondern auch auf *heise online* – dort in aller Regel aber hinter der Paywall, die *heise+* heißt. Um hinter die zu kommen, brauchten Sie bis Ende 2022 ein spezielles *heise+*-Abo. Das ist seither anders und Sie können als Make-Abonnent automatisch alle *heise+*-Artikel lesen, die aus der Make-Redaktion stammen – ohne Aufpreis!

Dafür müssen Sie lediglich Ihre Abo-Nummer mit einem kostenlosen Benutzerkonto bei *heise online* verknüpfen. Wenn Sie ein digitales Abo oder Plus-Abo haben, sollte beides

bereits zusammengeführt sein, dann müssen Sie nichts weiter tun. Falls Sie unser Magazin als gedrucktes Heft abonniert haben, steht Ihre Abo-Nummer zum Beispiel auf jedem Adressaufkleber auf dem Heft. Unter dem Link unten finden Sie eine detaillierte Anleitung, die zeigt, wo genau die Nummer zu finden ist und wie Sie diese dann Ihrem Benutzerkonto bei *heise online* bekannt machen. Falls Sie ein solches Konto noch nicht haben, gibt es dort ebenfalls den Link, um ein neues anzulegen.

Wenn Sie sich anschließend bei *heise online* anmelden, können Sie alle Artikel hinter

der *heise+*-Paywall lesen, die aus der Make-Redaktion kommen. Es lohnt sich! Handelte es sich bei „unseren“ *heise+*-Beiträgen bis ins vergangene Jahr noch in der Regel um Artikel, die bereits im Heft veröffentlicht wurden, lesen Sie inzwischen als Abonnent zunehmend vorab Artikel online. Vor allem gibt es aber auch zusätzliche Artikel für Sie im Web zu lesen: Drei aktuelle stellen wir Ihnen oben auf dieser Seite vor. —pek

► make-magazin.de/xcgd

Agon Light – der schnellste 8-Bit- Computer der Welt

Wem der Raspberry Pi zu viel Respekt einflößt, für den bietet der 8-Bit-BASIC-Computer Agon Light eine einsteigerfreundliche Alternative. Wir haben ihn getestet.

von Maik Schmidt

FUJITSU
COMPUTERS
SIEMENS

SELECT / MENU / F1 / F2 / F3 / F4 / F5 / F6 / F7 / F8 / F9 / F10 / F11 / F12 / F13 / F14 / F15 / F16 / F17 / F18 / F19 / F20 / F21 / F22 / F23 / F24 / F25 / F26 / F27 / F28 / F29 / F30 / F31 / F32 / F33 / F34 / F35 / F36 / F37 / F38 / F39 / F40 / F41 / F42 / F43 / F44 / F45 / F46 / F47 / F48 / F49 / F50 / F51 / F52 / F53 / F54 / F55 / F56 / F57 / F58 / F59 / F60 / F61 / F62 / F63 / F64 / F65 / F66 / F67 / F68 / F69 / F70 / F71 / F72 / F73 / F74 / F75 / F76 / F77 / F78 / F79 / F80 / F81 / F82 / F83 / F84 / F85 / F86 / F87 / F88 / F89 / F90 / F91 / F92 / F93 / F94 / F95 / F96 / F97 / F98 / F99 / F100



Eine ganze Generation von IT-Experten wuchs mit Heim-Computern auf und hat an diesen Maschinen viel für die spätere Karriere gelernt. Moderne Rechner sind zwar deutlich leistungstärker, aber fürs Selbststudium sind sie oft zu kompliziert.

Zwar herrscht an einfachen Klein-Computern auch heute prinzipiell kein Mangel, aber kaum ein modernes Gerät hat dieselben Qualitäten wie ein klassischer Heim-Computer. Diese zeichnen sich durch eine sehr geringe Komplexität aus, sodass Nutzer die Möglichkeit haben, die Maschine mit wenig Spezialwissen in Gänze zu durchdringen und zu übernehmen.

Bernardo Kastrup war lange auf der Suche nach dem perfekten Computer zum Experimentieren und Studieren. Der sollte billig, komplett und dazu vollkommen offen und frei programmierbar sein. Weil er kein Gerät finden konnte, das seinen Vorstellungen entsprach, hat er kurzerhand sein eigenes entwickelt und es Agon Light getauft.

Herausgekommen ist ein nagelneuer 8-Bit-Computer, der einen VGA-Anschluss, einen Sound-Ausgang, einen SD-Karten-Slot und einen Tastatur-Anschluss hat. Darüber hinaus unterliegen die Hard- und Software des Projekts einer Open Source-Lizenz, so dass jeder – auch Unternehmen – den Computer bauen und sogar verkaufen darf. Beispielsweise hat die Firma Olimex die Chance sogleich ergriffen und mit dem AgonLight2

Kurzinfo

- » BASIC-Bastelcomputer für 50 Euro
- » Maßgeschneidertes, aber offenes Betriebssystem
- » GPIO-Pins für Erweiterungen

Mehr zum Thema

- » Maik Schmidt: So geht Heimcomputer im Jahr 2020, Make 5/20, S. 24
- » Video: Der Agon Light in Aktion



Alles zum Artikel
im Web unter
make-magazin.de/xz12



für 50 Euro eine leicht angepasste, aber völlig kompatible Variante ins Programm aufgenommen.

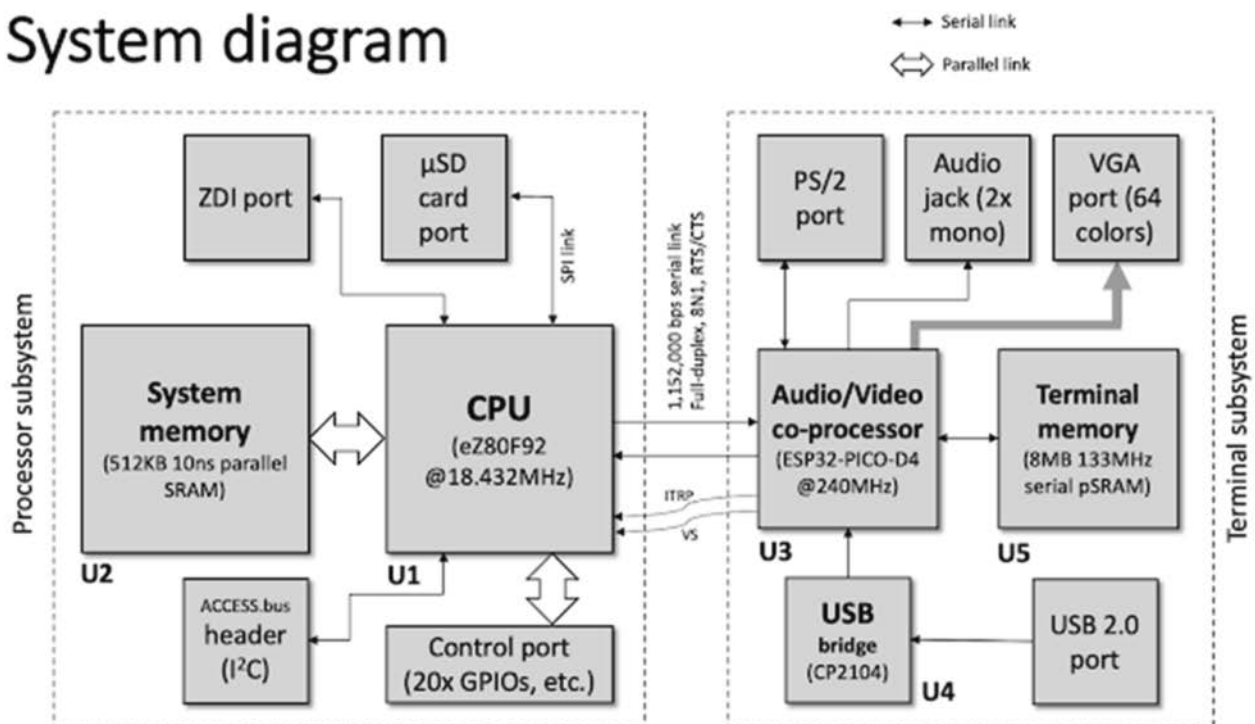
Die Hardware des Agon Light basiert nicht auf einem FPGA oder verwendet irgendeine Form von Emulation. Das ganze System ist übersichtlich und einfach aufgebaut und besteht aus modernen Bauteilen, die leicht erhältlich sind.

CPU & Speicher

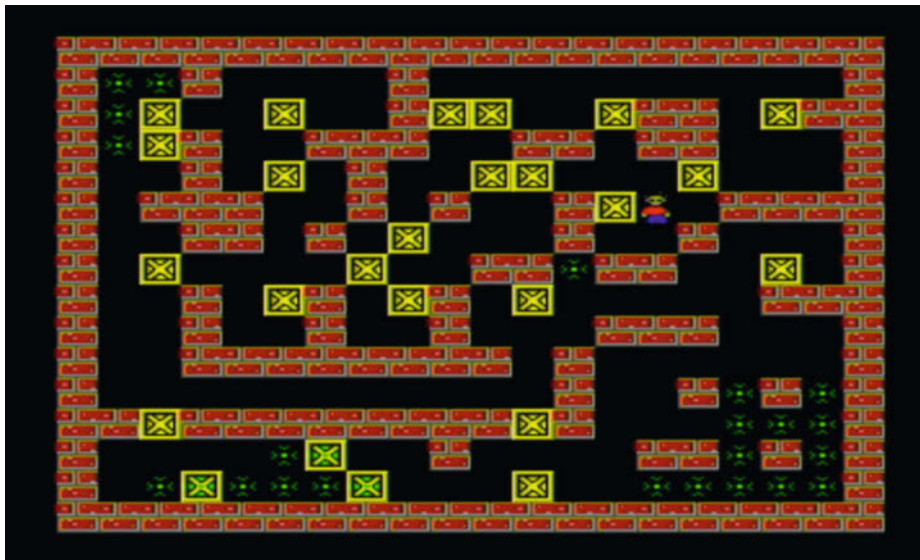
Der verwendete Microcontroller ist ein eZ80F92, also eine Neuauflage des altbekannten Z80. Der trieb viele populäre Heimcomputer an, zum Beispiel den ZX81, den ZX Spectrum oder die Schneider CPC-Reihe.

Der eZ80F92 ist eine echte 8-Bit-CPU und vollkommen abwärtskompatibel zum Original.

System diagram



Die Architektur des Agon Light ist zweigeteilt.



Spiele wie Sokoban funktionieren auf dem Agon Light prima.

nal. Allerdings bringt der Chip einige Verbesserungen mit, unter anderem eine 3-stufige Pipeline. Somit ist er nicht nur wegen seiner Taktfrequenz von 18,432 MHz deutlich schneller als sein Vorgänger. Auf der Webseite des Projekts zeigen unterschiedliche Benchmarks, dass der Agon Light tatsächlich der schnellste 8-Bit-Computer überhaupt ist. Verglichen wurde er dabei nicht nur mit klassischen 8-Bit-Rechnern, sondern auch mit Konkurrenten neueren Baujahrs.

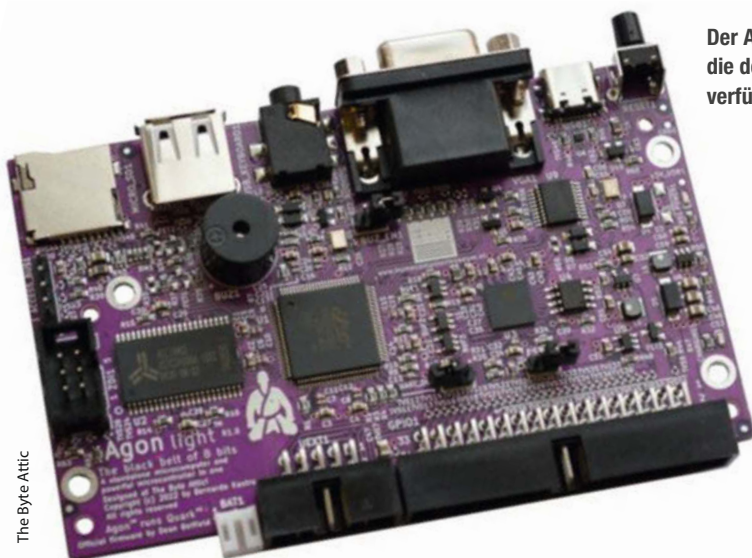
Ein weiterer Unterschied des eZ80F92 gegenüber dem klassischen Z80 ist der 24-Bit-Adress-Bus, der maximal 16 MB Speicher adressieren kann. Aus Gründen der Kompatibilität ist es möglich, ihn auf 16 Bit zu beschränken, sodass er nur noch auf 64 KB zugreifen kann. Der Agon Light nutzt diese Beschränkung nicht und verfügt über 512 KB SRAM.

Allzu große Datenmengen kann der Agon Light naturgemäß also nicht verarbeiten,

aber er hat einen Slot für Micro-SD-Karten mit einer Kapazität bis zu 32 GB. Er unterstützt das Dateisystem FAT. Dadurch ist es leicht, Daten mit PCs und Macs auszutauschen.

Visual Display Processor (VDP)

Eine gute CPU macht noch lange keinen guten Computer und so fehlen mindestens noch Ein- und Ausgabemöglichkeiten. Hier weiß der Agon Light ebenfalls zu glänzen, denn in seiner Brust schlagen gleich zwei Herzen. Neben dem eZ80F92 existiert auf dem Board nämlich ein ESP32-PICO-D4. In dem werkeln zwei 32-Bit-Kerne mit einer Taktfrequenz von 240 MHz. Der ESP32-PICO-D4 ist ein echtes Arbeitstier und er beherrscht sogar WLAN und Bluetooth. Von den meisten dieser Fähigkeiten macht der Agon Light aber keinen Gebrauch, sondern nutzt den ESP32 im Grunde als Grafik- und Soundkarte.



Der AgonLight2 ist die derzeit leichter verfügbare Variante.

Diese Multimedia-Einheit hat den Namen „Visual Display Processor“ (VDP) und ist mit der CPU seriell verbunden. Das heißt, wann immer die CPU Informationen per Audio oder per Video ausgeben muss, sendet sie ein Kommando über eine serielle Schnittstelle an den VDP. Das ist nicht die effizienteste Lösung, aber sie reicht vollkommen aus und ist leicht zu verstehen.

Der VDP verfügt über einen eigenen Audio-/Video-Speicher mit satten 8 MB PSRAM. Damit gibt der Agon Light problemlos Grafiken in Auflösungen von 320x200 bis 1024x768 Pixel in bis zu 64 Farben per VGA aus. Darüber hinaus bietet der VDP 256 Sprites und butterweiches Scrolling, sodass sich das Gerät hervorragend zur Programmierung von Videospielen eignet. Mit einem passenden Adapter funktioniert das alles auch an einem HDMI-Anschluss.

Ironischerweise ist der Co-Prozessor deutlich leistungstärker als die Haupt-CPU, aber das ist durchaus kein Widerspruch. Ziel des Agon Light-Projekts war es, einen möglichst einfachen und zugänglichen 8-Bit-Computer zu entwickeln, der über genügend Leistung verfügt, um anspruchsvolle und ansprechende Projekte umsetzen zu können.

An der Audio-Front sieht es nicht ganz so opulent aus, aber der Mini-Computer verfügt über eine 3,5mm-Klinkenbuchse zur Ausgabe eines Audiosignals und, falls dort nichts eingesteckt ist, leitet er den Sound auf einen eingebauten Buzzer um.

Ein Stereo-Signal unterstützt der VDP nicht, sondern sendet lediglich zweimal dasselbe Mono-Signal. Überhaupt ist die Unterstützung für Audioausgaben momentan noch recht rudimentär, aber das kann sich in Zukunft schnell ändern, denn die Firmware des VDP ist frei programmierbar.

Tastatur: PS/2

Der VDP kümmert sich nicht nur um die Ausgabe von Informationen, sondern auch um Eingaben, und zwar über eine Tastatur. Dazu setzen die Entwickler auf das PS/2-Protokoll und das Original-Board hat auch einen PS/2-Anschluss. Vor dem Hintergrund, das System möglichst einfach zu halten, ergibt diese Entscheidung durchaus Sinn, aber in der Praxis führt sie zu lästigen Problemen. Beispielsweise hat das AgonLight2-Board der Firma Olimex statt eines PS/2-Anschlusses einen USB-Anschluss. Der präsentiert sich nach außen aber gar nicht als USB-Anschluss, sondern als PS/2-Anschluss und so versagen die meisten USB-Tastaturen hier ihren Dienst. Nur Tastaturen, die das PS/2-Protokoll implementieren, funktionieren am Agon Light. Aber mittlerweile gibt es im Internet eine Liste mit kompatiblen Geräten.

Die Liste enthält auch compatible Adapter von PS/2 auf USB, denn auch die funktionieren nicht immer, weil manche davon zu intelligent

sind und vom PS/2- aufs USB-Protokoll übersetzen. Das ist in diesem Fall aber gerade nicht gewünscht.

Der Vollständigkeit halber sei der Reset-Knopf des Agon Light erwähnt, denn auch der ist ja in gewisser Weise ein Eingabegerät. Allerdings macht er nichts anderes, als den Rechner neu zu starten.

GPIO

Neustarts können insbesondere bei Elektronik-Experimenten oft notwendig sein und die machen auf dem Agon Light durchaus Spaß. Der führt nämlich 32 Pins nach außen, von denen 20 als GPIO-Pins fungieren. Daneben gibt es SPI, I²C, UART und Pins zur Spannungsversorgung sowohl für 3,3 V als auch für 5 V.

Der Agon Light selbst benötigt eine Stromquelle mit 5 V und mindestens 1,5 A über einen USB-A-Anschluss.

Intern läuft der Mini-Rechner mit 3,3 V und das ist auch die Spannung, mit der die Pins operieren. Es ist daher höchste Vorsicht beim Einsatz von 5 V-Bauteilen geboten, denn die müssen über einen Pegelwandler angeschlossen werden.

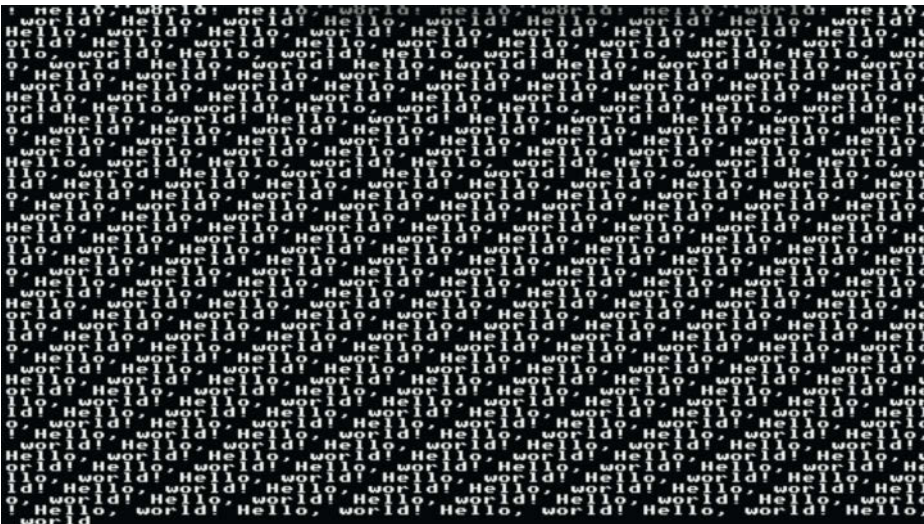
Der AgonLight2 von Olimex ist bezüglich der Anschlüsse und der Spannungsversorgung etwas üppiger ausgestattet als das Original. Zunächst einmal hat er einen USB-C-Anschluss und zusätzlich einen Anschluss für einen LiPo-Akku. Darüber hinaus hat er 34 Pins und die zwei zusätzlichen Pins (Vbat und Vin) können den Computer ebenfalls mit Strom versorgen. Schließlich hat das Olimex-Board einen UEXT-Anschluss, über den sich Erweiterungen leicht anschließen lassen.

Das Olimex-Board bietet auch noch ein paar weitere Detailverbesserungen, zum Beispiel eine Abdeckung für die GPIO-Pins. Es ist daher momentan die beste Wahl für alle, die sich für den Agon Light interessieren. Selbstverständlich spricht aber auch nichts gegen das Original, das versierte Bastler auch selbst bauen können.

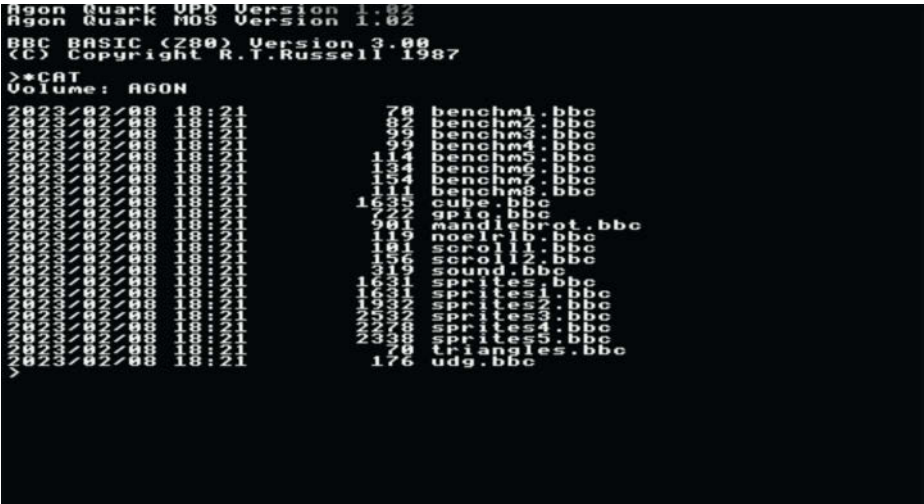
Software

Ohne gute Software nützt die beste Hardware nichts und diesbezüglich sieht es beim Agon Light richtig gut aus. Im Gegensatz zu den meisten Hobby-Projekten, deren Software oft eher mit der heißen Nadel gestrickt wird, kümmern sich in diesem Projekt engagierte Profis um den Code. Neue Boards enthalten in der Regel bereits eine aktuelle Version des Betriebssystems und die restlichen für den Betrieb notwendigen Dateien müssen lediglich auf eine SD-Karte kopiert werden. Diese liegen auf GitHub und bestehen im Wesentlichen aus einem BASIC-Interpreter und einigen Beispiel-Programmen.

Sobald das Gerät mit einer Tastatur, einem VGA-Monitor und einem Netzteil verbunden



Bildschirmausgaben sind deutlich flotter als auf alten Heimcomputern.



Auf der SD-Karte liegen ein paar lehrreiche Demo-Programme.

ist, geht es weiter wie in alten Zeiten, das heißt einschalten und loslegen. Nach circa vier Sekunden erscheint die BASIC-Eingabeaufforderung und beim Tippen der ersten Kommandos gibt es keinerlei Latenz – genau wie auf einem der guten alten Heimcomputer.

Erste Schritte

Erste Unterschiede werden jedoch schnell offenkundig, denn das klassische Test-Programm füllt den Bildschirm rasend schnell:

```
10 PRINT "Hello, world! ";
20 GOTO 10
```

Die BASIC-Variante für den Agon Light ist eine leicht angepasste Version des originalen BBC BASIC von 1981. Der Interpreter ist also alles andere als modern, aber er erfüllt seinen Zweck. Allerdings besteht er auf konsequenter Großschreibung, gibt nur rudimentäre Fehlermeldungen aus und erwartet, dass jede Zeile

eines Programms mit einer Zeilennummer beginnt. Davon abgesehen bietet BBC BASIC genug, um halbwegs anspruchsvolle Programme schreiben zu können.

Etwas lästig sind die primitiven Möglichkeiten zur Bearbeitung von Programmen, denn es gibt zurzeit keinen Bildschirm-Editor. Programmierer können nur einzelne Zeilen bearbeiten, indem sie die überarbeitete Zeile mit derselben Zeilennummer erneut eingeben. Die einzige Alternative ist das Edit-Kommando des Betriebssystems, das man mit einem vorangestellten Stern aufrufen kann.

TRIANGLES.BBC

10 MODE 0
20 GCOL 0,RND(255),RND(255),RND(255)
30 PLOT 85,RND(640),RND(480)
40 GOTO 20



Dreiecke zeichnet der Agon Light selbst in BASIC flott.

Um beispielsweise die Zeile 10 des „Hello, world!“-Programms zu editieren, muss man das Kommando `*edit 10` ausführen. Alle Kommandos, die mit einem Stern beginnen, sind keine BASIC-Anweisungen, sondern Betriebssystem-Kommandos. Bei denen spielt die Groß-/Kleinschreibung keine Rolle.

Am bequemsten ist es daher momentan, Programme mit einem BBC-BASIC-Editor auf dem PC zu editieren und über die SD-Karte auf den Agon Light zu bringen. Allerdings bieten neuere Versionen des Betriebssystems deutliche Verbesserungen bei der Eingabe von Kommandos und es gibt mittlerweile auch Bildschirm-Editoren, die aber noch nicht in das BASIC integriert sind. Ansonsten funktioniert alles wie gewohnt, das heißt, RUN startet das aktuelle Programm, NEW löscht es und LIST gibt den Code auf dem Bildschirm aus.

Programme laden

Die SD-Karte enthält ein paar Beispiel-Programme, die die Besonderheiten des BBC BASIC auf dem Agon Light demonstrieren. Das Kommando `*cat` gibt den Inhalt des aktuellen Verzeichnisses aus.

Zum Laden von Programmen dient die LOAD-Anweisung und die erwartet den Namen des zu ladenden Programms in Anführungszeichen. Die Anweisung `LOAD "triangles.bbc"` lädt also das Programm mit dem Namen TRIANGLES.BBC (siehe Kasten). Der RUN-Befehl startet das Programm und es zeichnet zufällige farbige Dreiecke in hoher Geschwindigkeit.

Das Listing TRIANGLES.BBC enthält nur vier Zeilen und ist denkbar einfach aufgebaut. Der MODE-Befehl setzt den zu verwendenden Grafikmodus und erwartet eine Zahl zwischen 0 und 3, welche die folgende Bedeutung hat:

- 0: 640 x 480 @ 60Hz
 - 1: 512 x 384 @ 60 Hz
 - 2: 320 x 200 @ 75 Hz
- (Nach MOS 1.02)

Der GCOL-Befehl setzt anschließend die aktuelle Zeichenfarbe auf einen zufälligen RGB-Wert und der PLOT-Befehl zeichnet ein farbig gefülltes Dreieck an einer zufälligen Bildschirm-Position.

Ein Druck auf die ESC-Taste bricht das aktuelle Programm ab. Dabei behält das System den Grafikmodus bei, der vom Programm eingestellt wurde und manchmal verschwinden auch Sprites nicht, die ein Programm angezeigt hat. Falls der CLS-Befehl, der den Bildschirm löscht, die Artefakte nicht beseitigt, sorgt der MODE-Befehl in jedem Fall für Abhilfe.

Es ist schon erstaunlich, was mit BASIC auf diesem Winzling möglich ist und so rotiert er beispielsweise problemlos einen 3D-Würfel auf dem Bildschirm. Das läuft nicht immer ganz ruckelfrei, ist aber dennoch bemerkenswert.

Das BBC BASIC auf dem Agon Light läuft also wirklich flott, aber wie bei den alten 8-Bit-Rechnern hilft bei manchen Problemen nur ein Griff zum Assembler. Praktischerweise bietet der BASIC-Interpreter einen Inline-Assembler, um Z80-Code direkt in BASIC-Programme einzubetten.

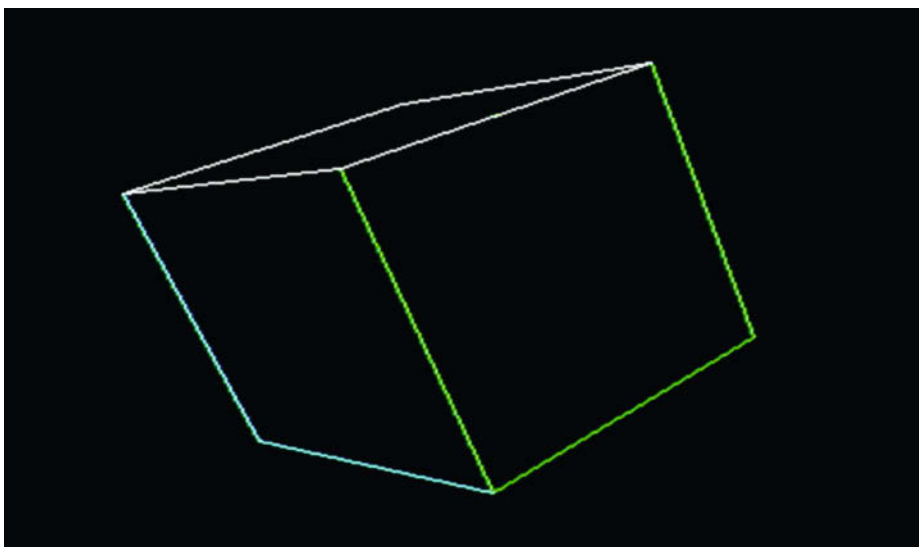
Blinkt es auch?

Für die hardwarenahe Programmierung ist zunächst kein Assembler nötig, denn die PUT- und GET-Befehle des BBC-BASIC erlauben das Schreiben und Lesen von Speicherinhalten.

Allerdings gilt es zuvor in Erfahrung zu bringen, welche Speicherbereiche denn relevant sind und diesbezüglich ist die Dokumentation auf den Seiten des Agon Light-Projekts noch sehr dürftig. Um beispielsweise eine LED zum Blinken zu bringen, ist gar nicht viel zu tun, aber vor dem Programmieren muss man das Datenblatt des eZ80F92 wälzen.

Dessen Kapitel „General-Purpose Input/Output“ beschreibt nämlich, wie die GPIO-Ports funktionieren. Wer die Port-IO eines Arduino Uno kennt, wird sich hier schnell heimisch fühlen, denn der eZ80F92 hat drei Ports mit den Namen B, C und D, die jeweils acht GPIO-Pins kontrollieren.

Die GPIO-Pins werden wiederum über vier Kontrollregister konfiguriert, die unter ande-



Der rotierende 3D-Würfel ist auch mit dabei.

BLINKT.BBC

```
10 PUT 159,254
20 PUT 158,1
30 D=INKEY 500
40 PUT 158,0
50 D=INKEY 500
60 GOTO 20
```

SCRAMBLE . BBC

```

10 REM Scramble
20 MODE 2
30 VDU 23,1,0
40 LET Y%=100
50 *FX 19
60 GCOL 0,5
70 DRAW 310,199,319,Y%
80 GCOL 0,7
90 PLOT &40,319,Y%
100 VDU 23,7,0,1,1
110 Y%=Y%+(RND(3)-2)
120 GOTO 50

```

rem festlegen, ob ein Pin gerade ein Eingabe- oder ein Ausgabe-Pin ist.

Das Beispiel-Programm in Listing BLINKT. BBC lässt eine LED blinken, die mit Pin 17 des Agon Light verbunden ist. Dieser Pin entspricht dem Pin GPIO_PC0 des eZ80F92, das heißt, er ist Pin 0 des Ports C. Das Data Direction-Register von Port C an der Speicheradresse 159 (0x9F) legt fest, welche Pins Eingabe- bzw. Ausgabe-Pins sind. Die erste Anweisung des Programms macht den Pin zum Ausgabe-Pin, indem es das unterste Bit auf 0 (254 = 11111110) setzt. Die übrigen sieben Pins werden zu Eingabe-Pins.



Butterweiches Scrolling mit prozedural generierten Grafiken sind kein Problem.

Das Data-Register von Port C liegt an Adresse 158 (0x9E) und auch hier kontrolliert das unterste Bit den Pin 0. Setzt man das Bit auf 1, wechselt der Pin zu HIGH. Andernfalls ist er LOW. Die Anweisung PUT 158,1 schaltet die LED also ein.

Weil BBC BASIC keinen SLEEP-Befehl kennt, verwendet das Programm den Befehl INKEY.

Der wartet auf einen Tastendruck und unterstützt als zweiten Parameter einen Timeout-Wert. Er wird an dieser Stelle also missbraucht, um eine halbe Sekunde zu warten und die GOTO-Anweisung am Programm-Ende sorgt für eine Endlosschleife.

Wer sich mit dem Z80 auskennt, kommt mit BASIC schnell ans Ziel und kann spätestens



Wir schreiben Zukunft.

2 Ausgaben MIT Technology Review
als Heft oder digital
inklusive Prämie nach Wahl

35%
Rabatt

mit-tr.de/testen



Der Sprite-Support vereinfacht die Programmierung von Spielen enorm.

mit Assembler so gut wie jede Hardware kontrollieren. Alle anderen müssen erst einmal das Datenblatt studieren. Ganz wie in alten Zeiten.

MOS-Z80?

Der Agon Light startet in der Standardkonfiguration direkt ins BBC BASIC, aber das ist nicht zwingend notwendig. Vor dem BASIC startet nämlich das eigens entwickelte Betriebssystem MOS, dessen Name vermutlich nicht zufällig an eine andere erfolgreiche 8-Bit-CPU erinnert. MOS liest beim Starten eine Datei mit dem Namen `autoexec.txt` und führt alle Anweisungen darin aus. Per Voreinstellung endet diese Datei mit den folgenden Anweisungen:

```
load bbcbasic.bin
run
```

Im Gegensatz zum `LOAD`-Befehl in BASIC erwartet MOS keine Anführungszeichen um den Dateinamen und die Groß-/Kleinschreibung spielt keine Rolle. MOS ist recht spartanisch, unterstützt aber die üblichen Befehle zur Verwaltung von Dateien und Verzeichnissen, wie zum Beispiel `cat`, `cd`, `copy`, `rename`, `mkdir` und `delete`. Die API des Betriebssystems ist auch nicht allzu umfangreich, aber sie bietet alles, um die Hardware des Systems komplett nutzen zu können.

Diese Einfachheit ist es, die den Charme des Agon Light ausmacht, denn es spricht überhaupt nichts dagegen, ein komplett eigenes Betriebssystem zu entwickeln oder einen Klassiker, wie zum Beispiel CP/M, auf das Gerät zu portieren. Letzteres ist tatsächlich schon passiert und so laufen auf dem Rechner unter anderem WordStar 4 und Turbo Pascal 3.

Wer kleiner anfangen möchte, kann zunächst auch das MOS modifizieren. Das ist

ziemlich leicht, denn es besteht im Wesentlichen aus C-Code, der an manchen Stellen mit ein wenig Assembler angereichert wurde. Das Projekt lässt sich mit der kostenlosen ZDS-II IDE von Zilog übersetzen und dann mit einem Flash-Programm auf dem Board installieren. In älteren Versionen war dazu noch ein spezielles Kabel notwendig, aber jetzt reicht es, das neue MOS auf eine SD-Karte zu kopieren und es anschließend zu installieren.

All das gilt auch für das BBC BASIC, denn das steht ebenfalls als Projekt für ZDS-II bereit.

Es spricht aber nichts dagegen, Programme komplett in Assembler zu schreiben und dazu gibt es auch bereits einen sehr guten Übersetzer. Darüber hinaus existiert ein Forth-Interpreter, der mit wenig Ressourcen auskommt.

Grafikkarten-Treiber

Die gesamte Programmierung bezog sich bisher auf den Z80-Kern des Agon Light, aber in dem Gerät steckt ja auch noch ein ESP32, der für die Multimedia-Abteilung namens VDP zuständig ist. Die ist recht leistungsfähig und sehr flexibel, wie Listing SCRAMBLE.BBC demonstriert.

Dieses kurze Programm generiert eine zufällige Landschaft, die ruckelfrei von rechts nach links scrollt. Das Ergebnis erinnert stark an das Videospiel Scramble und dank der Sprite-Unterstützung wäre es ein Leichtes, das Programm zu einem echten Spiel zu erweitern.

Der Code ist nicht nur knapp, sondern auch leicht zu verstehen. Wer früher auf einem Acorn Archimedes unterwegs war, wird auch gleich einige Bekannte erkennen.

Zeile 10 setzt den Grafikmodus und Zeile 20 schaltet den Text-Cursor ab. Dabei kommt der `VDU`-Befehl zum Einsatz und der sendet ein Kommando an den VDP. In diesem Fall ist es das Kommando 23, 1 und es erhält das Argu-

ment 0. Dank der guten Dokumentation wird schnell klar, was die einzelnen Kommandos bedeuten.

Die nächste Anweisung definiert eine Integer-Variable namens `Y`, die im weiteren Verlauf die aktuelle Höhe der zufällig berechneten Landschaft enthält. Spannender ist der folgende `FX`-Befehl, denn er wartet auf die vertikale Austastlücke des VGA-Signals. Das verhindert ein hässliches Flackern des Bildschirms.

Der Großteil der restlichen Zeilen dient dem Zeichnen der Landschaft, aber Zeile 100 bedarf einer genaueren Erklärung. Sie sorgt nämlich für das Scrolling und zwar mit einem weiteren Befehl an den VDP. Der Befehl heißt 23, 7 und er erwartet drei Parameter, nämlich den Scrollbereich, die Scrollrichtung und die Scrollgeschwindigkeit. Dabei kann der VDP nicht nur den ganzen Bildschirm, sondern auch nur Ausschnitte scrollen und zwar in alle vier Richtungen. Einfacher geht es kaum.

Die Software des VDP ist komplett offen und weil sie auf einem ESP32 läuft, steht sie als Arduino-Projekt zur Verfügung. Jeder, der sich mit dem ESP32 und der Arduino-IDE auskennt, kann sie also studieren, anpassen und erweitern.

Fazit

Der Agon Light ist ein rundum gelungenes Projekt, das die selbstgesteckten Ziele voll und ganz erfüllt. Bernardo Kastrup und den vielen freiwilligen Helfern ist es gelungen, quelloffene Hard- und Software zu entwickeln, die auch Einsteiger verstehen und beherrschen können.

Die geringe Komplexität grenzt den Agon Light von Produkten wie dem Raspberry Pi ab. Hinzu kommen der günstige Preis und die gute Verfügbarkeit, mit denen er gegenüber potenziellen Konkurrenten wie dem Commander X16 die Nase vorn hat.

Alte Hasen kommen ebenfalls voll auf ihre Kosten, denn viele kennen den Z80 noch aus ihrer Jugend und sind nach kurzer Einarbeitungszeit wieder voll auf der Höhe. Allerdings ist die Programmierung dank moderner Werkzeuge viel bequemer als früher.

Überhaupt ist so einiges besser, weil man das System ganz leicht auf vielfältige Weise anpassen kann. Beispielsweise kann man einfach mit Forth starten statt mit BASIC. Man kann auch ein eigenes BASIC bauen oder das mitgelieferte anpassen. Wer mag, kann sogar die Software für die Grafikkarte modifizieren.

Das Projekt ist zumindest so interessant, dass Hersteller wie Olimex eigene Boards produzieren, die unter anderem Mouser und ThePiHut zu günstigen Preisen und mit kurzen Lieferzeiten anbieten. Mittlerweile ist das Gerät daher weltweit problemlos zu bekommen. —das

MAKER FAIRE VIENNA AUSSTELLER 2023



Prusa Research was founded as a one-man startup in 2012 by Josef Prusa.

Now, there are more than 700+ people working in Prusa Research and we ship over 9000 printers worldwide directly from our HQ in Prague every month.

www.prusa3d.com



Infineon Technologies Austria steht für eine wirkungsvolle Kombination aus innovativer Forschung und qualitativ hochwertiger Produktion. Als eines der führenden Halbleiterunternehmen weltweit spielt der Infineon-Konzern eine Schlüsselrolle bei der Gestaltung einer besseren Zukunft.

www.infineon.com/cms/austria/de/ueber-infineon-austria/



Karriere mit Technik!

Mit über 8.000 Studierenden an sechs Standorten und fünf Kooperationsstandorten ist die FH Campus Wien die größte Fachhochschule Österreichs. Das Bachelor- und Masterangebot im Department Technik reicht von Elektronik und Informatik über Maschinenbau bis hin zu Krankenhaus-/Medizintechnik und erneuerbaren Energien.

www.fh-campuswien.ac.at



Seid ihr auch schon gespannt wohin die Reise im Lautsprecher Markt noch gehen wird? Wir auf jeden Fall und sind schon sehr gespannt auf eure Ideen und Anregungen und würden uns freuen wenn ihr uns mal auf unserer Seite besuchen würdet.

www.usound.com



Lust, als Maker mit der Top-Industrie gemeinsam zu tüfteln? Dann schau' mal bei Industry meets Makers vorbei! Da gibt es immer wieder neue spannende Challenges, bei denen Du mitmachen kannst. Das Spektrum reicht von IoT, Drohnen und Robotern, über 3D-Druck bis zur KI.

www.industrymeetmakers.com



Bei Soldered findest du alles, was das Herz eines „Makers“ begehrt - von Arduino-kompatiblen Platinen bis hin zu Sensoren und anderen Geräten. Unsere Open-Source-Produkte werden in Kroatien, Europa hergestellt. Nutze jetzt den Code „MAKER25“ und erhalte 25% Rabatt auf deine erste Bestellung.

www.soldered.com



Angewandte Forschung und Entwicklung, ständiger Austausch mit Wirtschaft und Industrie sowie Forschung in Hightech Labors mit modernstem Equipment - das ist Technik an der FH Wiener Neustadt! Das Innovation Lab ist ein öffentlich zugänglicher Makerspace. Auf 1.100 m² findest du eine vielfältig nutzbare Werkstatt, in der Ideen Wirklichkeit werden.

www.innolab.fhwn.ac.at



Nachhaltig Druck machen!

Mit unseren Filamenten aus recycelten Kunststoffen musst Du Dich nicht zwischen Nachhaltigkeit und Qualität entscheiden. Wir erweitern laufend unser Sortiment unterschiedlicher Materialien und Farben.

www.nobufil.com

BÜCHER / ZEITSCHRIFTEN



Der Verlag für kreative Köpfe!

Informatik und Elektronik können komplex, theoretisch und anstrengend sein. Es geht aber auch einfach, anschaulich und leicht nachvollziehbar – wenn man die Dinge in die eigenen Hände nimmt und zum »Maker« wird: Mit Büchern vom dpunkt.verlag.

www.dpunkt.de



Was Maker schon alles geschaffen haben!

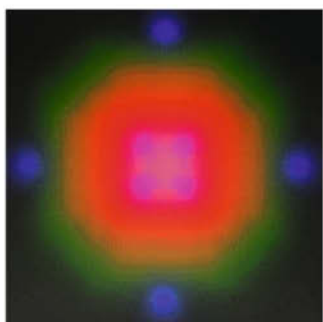
Die Antwort und viele Beispiele finden Leser in unseren Zeitschriften „Space – das Weltraum Magazin“, vom „Reißbrett“ ins Weltall und dem „Urknall“ vieler Computer- und Make-Enthusiasten – dem „Retro Gamer“.

www.emedia.de

Der LavaFrame

Wer hat nicht schon minutenlang auf eine Lava-Lampe geschaut und gedankenversunken den meditativen Eindruck genossen? Das geht auch als elektronische, selbst gebaute Variante mit abwechslungsreichen Effekten – und kostet dazu nur einen Bruchteil des Geldes vom Mathmos-Original.

von Daniel Springwald



Als meine Frau neulich ein paar Tage verreist war, nutzen mein Freund Thomas Smits und ich die Gelegenheit, um uns ein Wochenende lang im Werkkeller einzuquartieren und gemeinsam etwas zu bauen. Damit genug Vorlaufzeit zur Beschaffung benötigter Bauteile blieb, sammelten wir in den Wochen zuvor Ideen für entsprechende Projekte, um sie uns gegenseitig vorzustellen. Schnell zeigte sich, dass viele Vorschläge in eine ähnliche Richtung gingen und wir beide offenbar gern irgendetwas mit Licht und Bewegung entwickeln wollten. Am Ende machte die Idee „elektronische Lava-Lampe“ das Rennen – als markanten Titel für das fertige Gerät und das entsprechende GitHub-Projekt wählten wir: *LavaFrame*.

Der Aufbau

Das Herz des LavaFrame bilden ein ESP32 und 169 RGB-LEDs, die in einer 13×13-Matrix angeordnet werden. Die LEDs sind unter dem Produktnamen NeoPixel des Herstellers Adafruit bekannt geworden. Man findet im Internet mit diesem Suchbegriff inzwischen mehr alternative RGB-LEDs als originale NeoPixel. Für unser Projekt haben wir aufgrund von Verfügbarkeit und Preis eine Rolle eines No-Name-Herstellers verwendet.

Der LavaFrame ist in mehreren Schichten aufgebaut: Als Betrachter schaut man auf die vorderste, die aus transparentem Plastik besteht. Dahinter befindet sich eine Diffusionschicht, die das Licht der LEDs abdunkelt und zerstreut. Der Zwischenraum zur nächsten Schicht ist leer und dient zum Abstand zwischen Diffusionsmaterial und den RGB-LEDs. Je kleiner der Abstand gewählt wird, umso deutlicher kann man später die einzelnen „Pixel“ voneinander unterscheiden. Dahinter ist eine Holzplatte verbaut, auf welche die LED-Reihen aufgeklebt sind. Als Letztes folgt die Rückseite des Bilderrahmens, auf der die Elektronik befestigt ist.

Der Effekt

Die einzelnen RGB-LEDs sind durch die Diffusionsschicht nicht zu identifizieren, sondern ihr Licht verläuft optisch ineinander, wobei an den Übergängen Mischfarben entstehen können. Scharf erkennbare Formen oder gar (Lauf-)Schrift lassen sich damit nicht darstellen – am besten funktionieren runde, organische Formen oder Verläufe. Langsame und fließende Bewegungen eignen sich dabei gut, während schnelle und kontrastreiche Animationen eher wie Flackern erscheinen. Am besten wirkt der LavaFrame mit Animationen in der Art der namensgebenden Lava-Lampe: bewegten Farbverläufen oder Feuer- und Plasma-Simulationen. Damit könnte er gut in einer Lounge-Bar als Dekoration an der Wand hängen. Da der Effekt mit Worten und Fotos nicht so gut

Kurzinfo

- » Dekoratives und meditatives Lichtobjekt
- » Leicht mit selbst programmierten Effekten erweiterbar
- » Auch gut als Geschenk geeignet

Checkliste



Zeitaufwand:
ein Wochenende



Kosten:
unter 50 Euro



Löten:
einfache Lötarbeiten



Programmieren:
Code mit der Arduino-IDE auf ESP32 laden, für eigene Animationen programmieren in C/C++

Mehr zum Thema

- » Carsten Wartmann, Disco is back – Lichtshow mit WLED, Make 2/23, S. 8
- » Video: Der LavaFrame in Aktion



Alles zum Artikel
im Web unter
make-magazin.de/xay7



Material

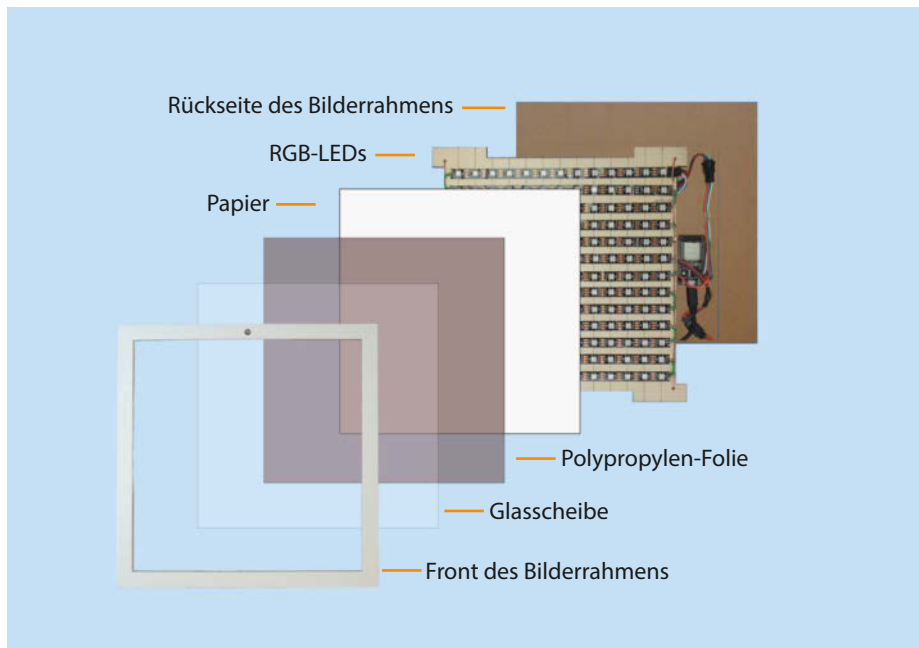
- » 169 WS2812B-LEDs als LED-Streifen mit 60 LEDs pro Meter
- » Widerstand 300 Ohm für RGB-LEDs
- » Elko 1000µF
- » ESP32-Board
- » Micro-USB-Buchse mit Verlängerungskabel zum Einbau
- » 5-Volt-USB-Netzteil für den ESP32
- » 5-Volt-Netzteil mit ausreichend Leistung zur Versorgung der 169 LEDs bei voller Leistung, typischerweise ab 35W/10A
- » 2 Taster zum Einbau in ein Gehäuse mit 8mm starker Holzwand
- » Lichtabhängiger Widerstand LDR 5516 5528 optional
- » Kabel 0,75 mm² Litze, je 2m in rot, blau, schwarz
- » Folie zur Abdunklung 25cm × 25cm, Handelsbezeichnung etwa „Polypropylen Dunkelgrau Antireflex Transluzent“
- » Papier normales weißes 80-Gramm-Papier als Diffusor, 25cm × 25cm
- » Ikea-Bilderrahmen SANNAHED, 25cm × 25cm
- » Sperrholz, MDF oder ähnliches Plattenmaterial von 4-8mm Stärke, 25cm × 25cm, als Träger zum Aufkleben der LEDs

Werkzeug

- » Lötkolben mit Zubehör
- » Übliches Elektronikwerkzeug zum Zuschneiden und Abisolieren von Kabeln
- » Heißkleber oder ähnlicher Klebstoff zum Befestigen der LED-Streifen
- » Bohrmaschine und Bohrer etwa für die Taster und Öffnungen zur Stromversorgung



RGB-LEDs auf dem Streifen auf der Rolle



Die einzelnen Schichten des LavaFrame

zu vermitteln ist, kann man sich am besten im Video einen Eindruck verschaffen (siehe Link in der Kurzinfo).

Augen auf beim Rahmenkauf

Der technische Aufbau des LavaFrame ist nahezu identisch mit der WordClock, die Thomas und ich ein paar Jahre zuvor gebaut hatten.

Auch diese besteht primär aus einem tiefen Ikea-Bilderrahmen und einer Matrix aus RGB-LEDs. Da ich vom Bau der WordClock noch einen Rahmen unbenutzt übrig hatte, hätte im schwedischen Möbelhaus eigentlich nur noch ein zweiter Rahmen für Thomas' LavaFrame gekauft werden müssen – so dachten wir. Was leider erst auffällt, wenn man zu Hause den alten neben den neuen Rahmen



RIBBA vs. SANNÄHED

Koordinaten-Mapping

```
// Macro to transform x/y coordinates into an index in the leds array
#define _tgt(x, y) (((y) % 2) ? ((y + 1) * LF_COLS - (x + 1)) : ((y) * LF_COLS + (x)))
```

Nachbau leicht gemacht

Die hier beschriebenen Prototypen des LavaFrame entstanden in bester Maker-Manier unter Einsatz von allem, was sich an Materialresten in der Werkstatt und Komponenten in der Bastelkiste fand. Wer über keinen so üppig bestückten Fundus verfügt, kann sich demnächst über den Link in der Kurzinfo viele zum Nachbau nötigen Teile als Paket bei Berrybase bestellen. Eine Schritt-für-Schritt-Anleitung für den Nachbau mit Hilfe dieses Teilesatzes wird es dann ebenfalls online geben.

legt: Der Hersteller hatte in der Zwischenzeit die Maße des Bilderrahmens verändert. Während der 2018er Rahmen (RIBBA) ein Maß von 23cm x 23cm hatte, weist die aktuelle Version (SANNÄHED) 25cm x 25cm auf. Gleichzeitig ist der neue Rahmen etwas tiefer, was für den LavaFrame aber von Vorteil ist, da dadurch ein größerer Abstand der LEDs zum Diffusionsmaterial möglich ist. Von der grundsätzlichen Annahme, dass man später nochmal einen Rahmen nachkaufen könnte, muss man sich aber wohl verabschieden: Die Passepartouts, Schnittmuster und LED-Abstände muss man bei jedem „Modellwechsel“ komplett neu anpassen. Bei fünf Euro pro Stück lohnt es sich dann wohl eher, ein Paar der Rahmen auf Vorrat zu kaufen, wenn es wahrscheinlich ist, dass man noch den einen oder anderen LavaFrame bauen und verschenken will (wozu er sich übrigens prima eignet).

Interne Verkabelung

Die verwendeten RGB-LEDs verfügen auf beiden Seiten über jeweils drei Anschlusspole für Masse, +5 Volt und den Datenkanal. Während die Pole für die Spannungsversorgung auf beiden Seiten direkt verbunden sind, ist der Datenpin an der einen Seite mit *in* und an der anderen mit *out* beschriftet. Der Data-in-Anschluss einer RGB-LED wird dabei mit dem Data out der vorherigen verbunden – was schon vermuten lässt, dass hier mehr Elektronik integriert ist, als man bei einem reinen Leuchtmittel erwarten würde. Tatsächlich verfügt jede RGB-LED über einen eigenen Mikrochip und verarbeitet die übermittelten Informationen, bevor diese an die nächste LED weitergegeben werden.

Wenn man die RGB-LEDs, so wie wir, auf einer Rolle kauft, sind alle bereits miteinander verbunden, sodass dort nicht mehr selbst gelötet werden muss. Zwischen den LEDs finden

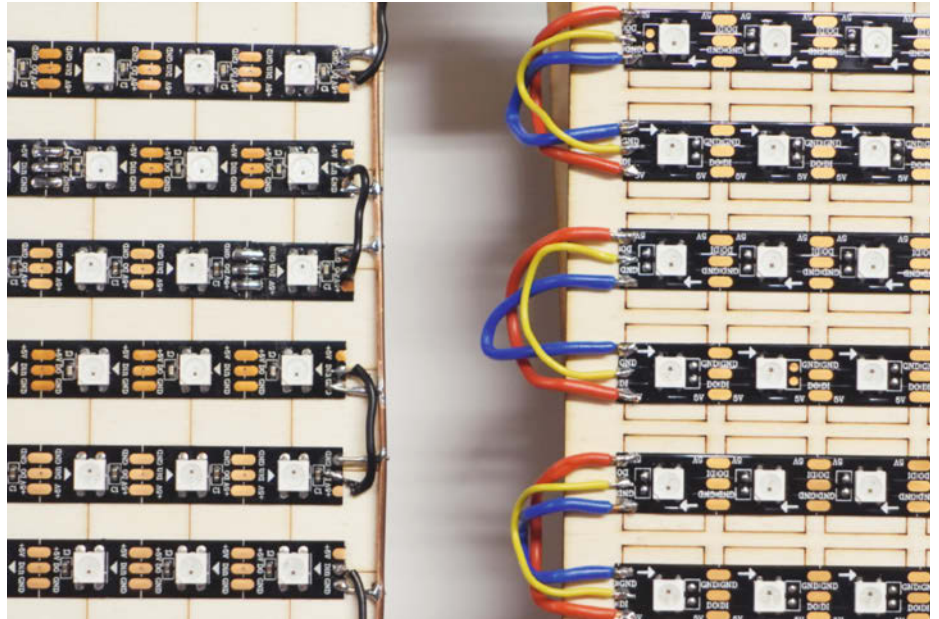
sich Lötunkte auf der Folienplatine, an denen man den Streifen mit einer Schere unterteilen kann. Für unser Projekt brauchten wir 13 solcher Stücke à 13 RGB-LEDs.

Für eine möglichst übersichtliche Verkabelung haben wir die LEDs nicht alle von links nach rechts gezählt, sondern jede zweite Reihe „rückwärts“. Dieser Umstand wird dann später im Arduino-Programmcode mit etwas Mathematik wieder herausgerechnet (siehe Kasten *Koordinaten-Mapping*). LF_COLS ist dabei die Anzahl der RGB-LEDs pro Zeile.

Ich hätte nun (wie schon vor einigen Jahren bei der WordClock) die LED-Reihen jeweils abwechselnd links und rechts mit drei Kabeln verbunden, wenn Thomas nicht noch eine Optimierung parat gehabt hätte: Die Leitungen für GND und +5 Volt sind an beiden Seiten einer Reihe identisch und müssen daher nicht von Reihe zu Reihe weitergeleitet werden. Stattdessen sind beim LavaFrame GND für alle Reihen auf einer Seite angelegt und +5 Volt für alle Reihen auf der gegenüberliegenden Seite. Als Leitungsdraht haben wir noch vorhandenes Elektro-Installationskabel verwendet, dessen Leiter wir dazu abisolierten.

Als Positionierhilfe haben wir in das Brett noch dünne Linien hinein gelasert. Wer keinen Lasercutter hat, kann sich ein quadratisches Hilfsgitter auch von Hand anzeichnen – die Gittergröße ist durch den Abstand der LEDs auf dem Streifen schon vorgegeben.

Um den LavaFrame wieder leicht auseinander nehmen zu können, haben wir alles mit Steckern verbunden, anstatt es fest zu verlöten. Das war insbesondere hilfreich, weil wir noch umfangreich mit dem Diffusionsmaterial experimentieren mussten und dazu



Links: Schlankere Verkabelung für GND (5-Volt-Verbindungen auf der anderen Seite, außerhalb des Bildes)
Rechts: Konventionelle Verkabelung von LED-Zeile zu LED-Zeile

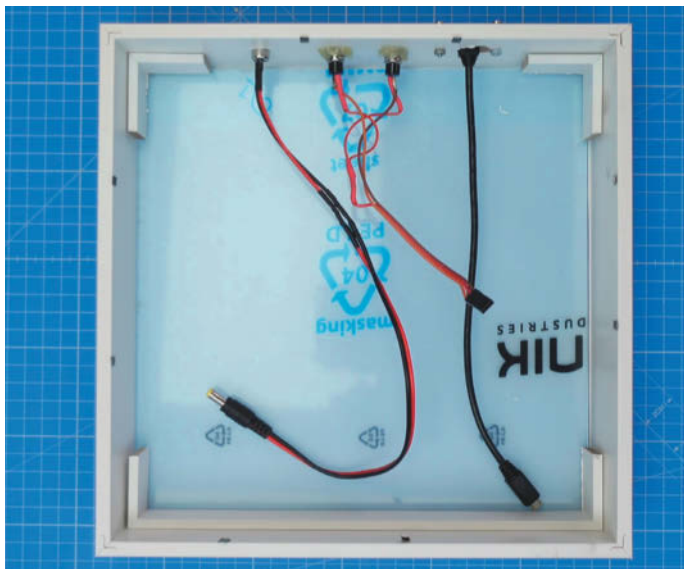
jedes Mal die untersten Schichten ausgetauscht wurden – doch dazu später mehr. Unabhängig davon ist es aber zumindest beim USB-Kabel optisch schöner, wenn dieses als Stecker nach außen geführt wird, statt das Kabel direkt lose in das Gehäuse hineinzuführen. Das Prinzip, alles mit Steckern zu verbinden, haben wir bei der gesamten Verkabelung konsequent verfolgt und sowohl USB-Kabel, Stromanschluss sowie das Kabel für Lichtsensor und Taster per Zwischenstecker vom Gehäuse getrennt. Wer sich diesen

Aufwand sparen will, kann aber auch alles fest verdrahten und anlöten.

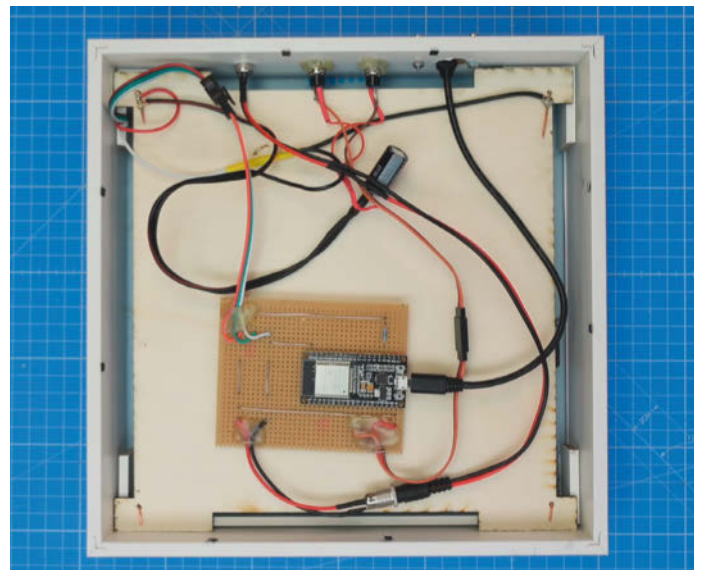
Die Platine

Grundsätzlich könnten die Kabel direkt an den Mikrocontroller angelötet werden. Erfahrungsgemäß brechen sie dann aber gern mal ab, wenn man den ESP32 öfter bewegt.

Zur Befestigung und Zugentlastung der Kabel haben wir daher noch zusätzlich eine Lochrasterplatine verwendet. Sie enthält



Um die Konstruktion beim Experimentieren mit dem Diffusionsmaterial besser zerlegen zu können, haben wir alle Verbindungen steckbar ausgeführt. Von links nach rechts: Spannungsversorgung für den LED-Streifen, die beiden Taster für die Steuerung, Spannungsversorgung für den ESP32



Die Montage des ESP32 auf der Lochrasterplatine sollte ebenfalls helfen, dass die Verbindungen zwischen Kabel und Pins viele Demontagen überstehen. Wer sich den Aufwand sparen will, kann die Kabel auch direkt anlöten.

außer einem 300-Ohm-Widerstand für den Datenpin der LEDs keine weiteren Bauteile. Die auf dem Bild sichtbaren Drahtbrücken enthalten somit auch keine zusätzliche Logik, sondern führen die Signale 1:1 vom Pin zum Kabel.

Die Pins des ESP32 können dabei grundsätzlich selbst gewählt werden und müssten nur entsprechend in der Datei `hardware.h` angepasst werden. Bei unserem ESP32-Board haben wir die Pins 15, 27, 26 und 34 verwendet und die Kabel so angeschlossen, wie es die Tabelle *Pinbelegungen* zeigt.

Stromversorgung

Wenn man den LavaFrame auf maximaler Helligkeit der RGB-LEDs betreiben möchte, sollte man auf jeden Fall die 5-Volt-Spannung für die LEDs getrennt von der Stromversorgung des ESP32 über ein eigenes Netzteil und Kabel zuführen. Wie viel Ampere und Watt das liefern muss, hängt vom Streifen ab, bei der Berechnung kann der im WLED-Artikel in der vorigen Ausgabe verlinkte Online-Rechner gute Dienste leisten (siehe *Mehr zum Thema* in der Kurzinfor). Der Kondensator parallel zur Spannungsver-

sorgung des LED-Streifens dient zu deren Glättung unter der Belastung von schnellen Lichtwechseln im Betrieb.

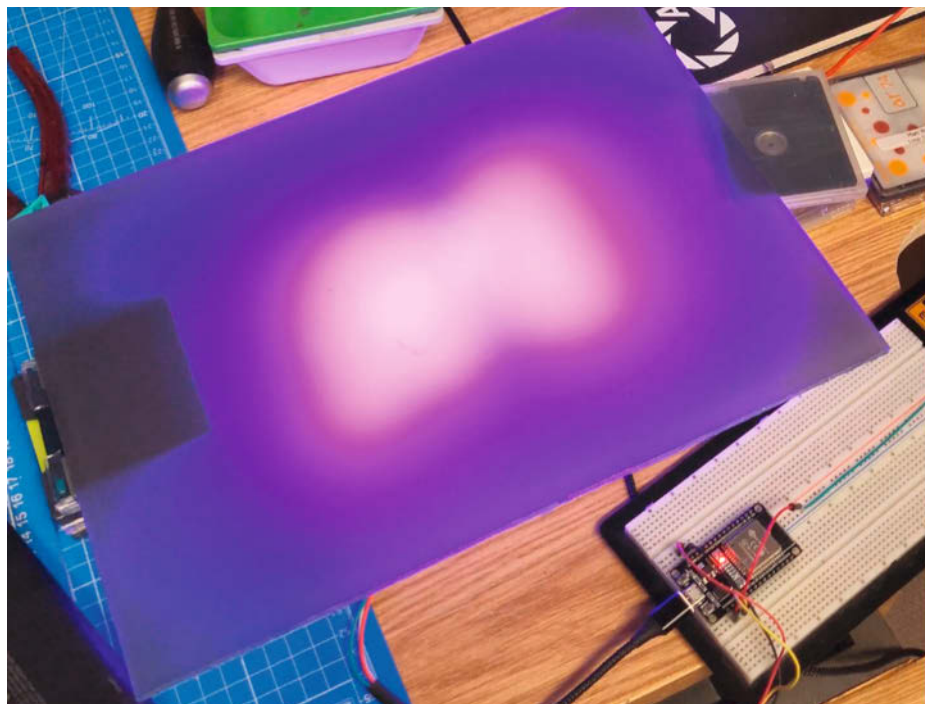
Bei per Software mindestens halbiert Maximalhelligkeit kann oft schon ein leistungsfähiges USB-Netzteil genügen, um sowohl den ESP als auch die LEDs zu versorgen – dann sollte man aber die Spannung für die LEDs nicht am ESP32 abgreifen, sondern direkt am USB-Kabel vorher. Andernfalls ist der Spannungswandler des ESP32 gern einmal überfordert und brennt durch.

Das Diffusionsmaterial

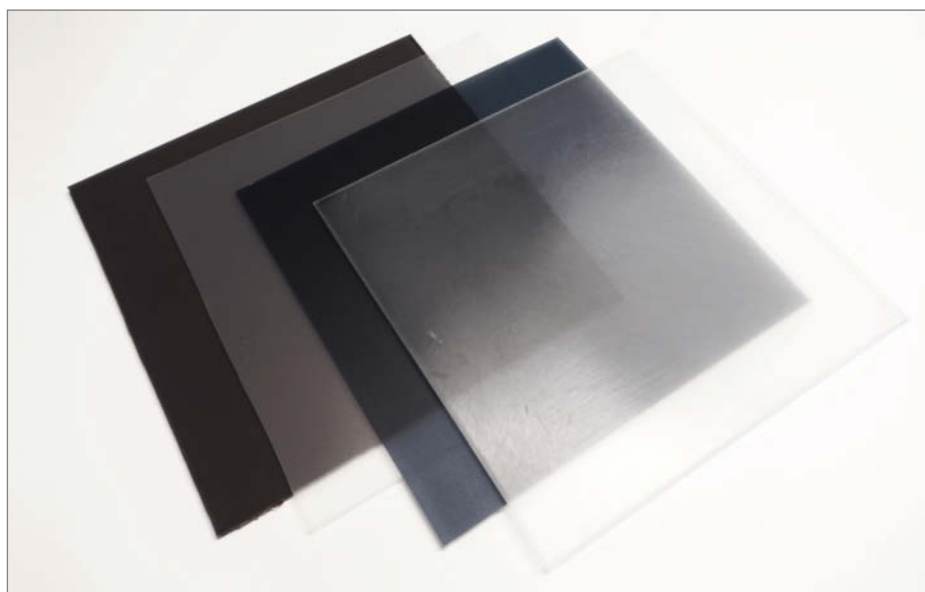
Der visuelle Effekt lebt von der Vermischung des Lichts der einzelnen RGB-LEDs untereinander. Die meisten Betrachter glauben zuerst nicht, dass der LavaFrame nur eine Auflösung von 13×13 Pixeln hat, weil das Licht und die Farben so organisch ineinander übergehen. Auch die Bewegungen wirken dadurch eher fließend und nicht ruckelnd, anders als zum Beispiel bei einer LED-Tafel, die Schrift anzeigt. Um diesen Effekt zu erzielen, müssen LED-Abstand und Diffusionsmaterial zueinander passen, sonst erkennt man die einzelnen Lichtpunkte dann doch durch die Scheibe hindurch.

Die Diffusionswirkung ist dabei von der Art des Materials abhängig. Wir haben uns daher mit verschiedenen Materialien beschäftigt, um den besten Effekt zu erzielen. Da wir bei der WordClock gute Erfahrung mit Pergamentpapier gemacht hatten, gingen unsere ersten Versuche ebenfalls in diese Richtung. Hier schienen die Pixel aber zu sehr durch – zumindest wenn man keine aufwändige Konstruktion mit mehreren Abstandsebenen verwendet. Als nächstes versuchten wir es mit dünnem, weißem Papier. Dieses hat zwar einen guten Diffusionseffekt, es fehlte dafür aber die Möglichkeit, dunkle Kontraste abzubilden. Beim Test verschiedener Papiere fiel uns auf, dass man teilweise die Faserstrukturen des Papiers sehr auffällig sehen konnte, dies aber bei glatt gestrichenem Papier am wenigsten auftrat.

Die besten Ergebnisse erzielten wir am Ende mit dünnem Papier in Kombination mit einer Polypropylen-Folie mit der Handelsbezeichnung „Dunkelgrau Antireflex Translucent“ zum Abdunkeln und für einen stärkeren Kontrast. Diese Folie wird zum Beispiel als Tiefziehfolie vertrieben. Als genauso span-



Beim Herausfinden des optimalen Abstands zwischen LEDs und Diffusor machten sich Stapel von MiniDisks in variabler Höhe nützlich. Alles andere stapelbare Material geht natürlich auch.



Materialversuche für die Diffusionsschicht

Pinbelegungen

GPIO-Pin	Anschluss
15	Data-in der LED-Streifen, dazwischen ein 300-Ohm-Widerstand
27	Taster 1
26	Taster 2
34	Lichtabhängiger Widerstand als Lichtsensor (optional)

nend wie die Auswahl des Diffusionsmaterials stellte sich die Ermittlung des Abstands zur LED-Ebene heraus: Hier gilt es etwas zu experimentieren, bis man das nach eigenem Geschmack beste Ergebnis erhält.

Die Software

Den verbauten ESP32 haben wir mit der Arduino IDE programmiert. Den Code gibt es zum Download über den Link in der Kurzinfo, ebenso grundlegende Anleitungen, wie man ESP-Boards in der Arduino IDE programmiert und gegebenenfalls erforderliche Bibliotheken installiert. Damit die einzelnen Animationen sauber getrennte Kontexte bekommen können, wurden C++-Klassen statt des sonst für Arduino typischen C-Quellcodes verwendet.

Wenn die eigene Hardware andere Hardware-Pins oder eine andere Anzahl als 13 RGB-LEDs pro Reihe verwendet, können die Hardwarespezifikationen im Programmcode angepasst werden. Dabei sollte das entstehende LED-Feld aber immer quadratisch sein, also in Höhe und Breite die gleiche Anzahl LEDs verwenden.

Die Software steuert die RGB-LEDs über ein eindimensionales Array, auf das man beim Programmieren neuer Animationen aber über entsprechende Methoden zugreift, sodass man mit den gewohnten X- und Y-Koordinaten arbeiten kann. Dabei wird für jede RGB-LED der gewünschte Farbwert mittels jeweils drei Bytes für die Farben Rot, Grün und Blau geschrieben. Die Farbwerte werden dabei als 8-Bit-Werte zwischen 0 und 255 angegeben. Die so angesteuerten RGB-LEDs leuchten dann entsprechend farbig, hell oder dunkel.

Wie das funktioniert, soll hier am Beispiel der Feuer-Animation vorgestellt werden, die recht organisch wirkt, aber dennoch nicht komplex programmiert ist: Die Basis bildet eine Palette aus 256 Farben, welche von hellblau (= Farbe 255) über weiß und rot nach dunkelbraun und am Ende zu schwarz (= Farbe 0) verläuft. Dieser Verlauf entspricht dem Lebenszyklus eines virtuellen Flammen-Partikels, das am unteren Rand hell leuchtend startet und dann nach oben steigt, um am oberen Rand zu schwarzem Rauch zu werden.

Bei jeder Aktualisierung der Animation wird in der untersten Zeile des virtuellen LED-Bildschirms an zufälligen Positionen die Farbe 255 (=hellblau) eingetragen. Eine Schleife durchläuft anschließend die LED-Reihen von oben nach unten und berechnet für jede LED die neue Paletten-Farbe nach folgender Formel:

$$NF = ((FLN + FLR + 2 \cdot FU) / 4) - 1$$

mit NF = neue Farbe, FLN = Farbe linkes Nachbarfeld, FLR = Farbe rechtes Nachbarfeld, FU = Farbe Feld darunter.

Die neue Farbe orientiert sich somit an den Feldern neben und unter ihr, wobei das untere

re Feld doppelt so großen Einfluss bekommt. Dadurch steigt das virtuelle Feuer-Partikel kontinuierlich nach oben, wobei die Geschwindigkeit durch das -1 am Ende noch etwas beschleunigt wird. Weil dieser Feuer-Algorithmus ziemlich echt wirkt, die Programmierung aber sowohl simpel als auch performant ist, war er bereits in den 1990er Jahren bei Programmierern von Grafik-Demos unter MS-DOS sehr beliebt.

Die Tastenbelegung

Die Belegung der beiden verbauten Taster zeigt die Tabelle.

Was mir vor dem Projekt gar nicht bekannt war und ich erst für die Programmierung des LavaFrame recherchiert habe: Man kann auf einem Arduino oder ESP32 Informationen speichern, die eine Strom-Abschaltung überdauern. Bis dahin war ich davon ausgegangen, dass man nur durch das Übertragen des Programms aus der IDE dauerhafte Veränderungen im Mikrocontroller hinterlassen kann.

Das Speichern nicht-flüchtiger Daten geht allerdings nur in begrenztem Umfang und die Gesamtanzahl der möglichen Schreibzugriffe bis zu einem möglichen Defekt ist limitiert (die Arduino-Dokumentation gibt hier 100.000 Schreib-/Löschzyklen an). Zum permanenten Loggen von Daten ist dieser Speicher daher eher nicht empfehlenswert – für das Speichern von Einstellungen wie der Rotation des LavaFrame aber optimal geeignet.

Um Daten permanent zu speichern, bindet man die Library EEPROM.h in sein Programm ein. Der dazu benötigte Programmcode ist übersichtlich, wie man am Beispiel der *Rotation* sehen kann.

Ein Ausblick

Wir haben den LavaFrame mit einem optionalen Helligkeitssensor ausgerüstet, der bei dunkler

Tastenbelegungen

Taste	Status	Aktion
1	Drücken	Nächste Animation
1	Halten	Modus zwischen <i>Slide-show</i> und <i>einzelne Animation</i> wechseln
2	Drücken	Animationen um 90° drehen
2	Halten	Helligkeit anpassen

Umgebung automatisch die Helligkeit der Animation herunterdimmt. Eine zusätzliche Erweiterungs-Idee für die Hardware wäre ein Lagesensor – für den Fall, dass man den LavaFrame öfter anders aufstellen möchte und sich die Rotation dann automatisch anpassen soll. Das könnte die beiden Taster sogar komplett überflüssig machen, sofern man darauf verzichten kann, Animationen manuell zu überspringen.

Eine zukünftige Version 2.0 der LavaFrame-Software könnte per WLAN gesteuert werden, sodass man den ESP32 für neue Animationen nicht immer neu per USB beschreiben muss. Die WLAN-Konfiguration und den dazu notwendigen Script-Interpreter einzubinden und die Live-Speicherung neuer Animationen zu konzipieren, hätte aber den Umfang unseres Wochenend-Hackathons deutlich überstiegen.

Nicht nur das Anschauen des LavaFrame macht Spaß, auch das Programmieren von Animationen ist spannend, weil man dort mit wenig Code optisch beeindruckende Effekte erzielen und das Ergebnis zudem direkt ausprobieren kann. Wir werden daher in Zukunft sicher noch die eine oder andere Animation hinzufügen. Das Ergänzen neuer Animationen ist dabei bewusst einfach gestaltet: Man muss nur eine neue Klasse für die neue Animation anlegen und diese in der Datei `LavaFrame.ino` eintragen. Wir freuen uns daher über Pull-Requests mit neuen Animationen in das GitHub-Projekt. —pek

Rotation

```
#include <EEPROM.h>
#define EEPROM_SIZE 512
#define EEPROM_ORIENTATION 0

...

// EEPROM-Nutzung initialisieren
EEPROM.begin(EEPROM_SIZE);

...

// Schreiben der geänderten Rotation
EEPROM.write(EEPROM_ORIENTATION, orientation);
EEPROM.commit();

...

// Lesen der geänderten Rotation
orientation = EEPROM.read(EEPROM_ORIENTATION);
```

Sauerteig aus dem IoT-Einmachglas

Sauerteig selber machen ist seit Corona in gewissen Kreisen in – aber mit Ultraschallsensor, ESP32 und Grafana-Visualisierung wird das Überwachen des Teig-Gehens gleich drei Klassen nerdiger. So geht's.

von Ivana Huckova (Übersetzung: Katja Heimann-Kiefer)



Gleich zu Anfang verrate ich Ihnen jetzt etwas: Ich bin eine von denen, die in der ersten Corona-Welle angefangen haben, ihr eigenes Brot zu backen. Ich habe sogar meinen Sauerteig selbst angesetzt. (Ein einfaches Rezept für den Sauerteig-Starter, in der Fachsprache auch Anstellgut genannt, gibt es über den Link in der Kurzinfor.) Den Anfang machen Sie mit Mehl und Wasser. Danach kommt es darauf an, den Teig auf der richtigen Temperatur zu halten und regelmäßig zu „füttern“. Nur wenn alle Bedingungen optimal sind, gärt der Teig richtig und geht schön auf. Wenn es Ihnen dann gelingt, damit zum optimalen Zeitpunkt zu backen, wird Ihre Mühe durch aromatisches, saftiges Brot belohnt.

Ich bin Softwareentwicklerin bei *Grafana Labs* und immer an der Welt um mich herum interessiert. Als ich dann erfahren habe, dass man ein Sauerteig-Überwachungssystem bauen kann, musste ich das einfach ausprobieren, denn es heißt, dass so ein System die Qualität des Teigs und damit auch den Geschmack des Brotes verbessert.

Sauerteig-Überwachungssysteme protokollieren die Temperatur, den Feuchtigkeitsgehalt und die Höhe des angesetzten und aufgehenden Sauerteigs. Mein System verwendet das Entwicklungsboard ESP32, das die Daten von den Sensoren am Deckel des Sauerteigglases ausliest ¹. Zur Messung der Teighöhe verwende ich einen Ultraschall-Entfernungssensor, für Temperatur und Feuchtigkeit einen DHT11-Sensor. Die ermittelten Daten werden an eine *Prometheus*-Datenbank gesendet, und *Grafana* erstellt daraus schicke Diagramme. In *Grafana* können Sie die gemessenen Werte über die Zeit verfolgen oder sogar einen Alarm einrichten, der Ihnen Bescheid gibt, wenn Sie den Sauerteig wieder füttern müssen.

Den Anstoß für mein eigenes System – mein allererstes IoT-Projekt! – gab mir das Sauerteig-Überwachungssystem *sourd.io* von Christine Sunu. Die Arbeit an meinem Projekt hat mir viel Freude gemacht, die ich weitergeben möchte, besonders an diejenigen, die ein erstes IoT-Projekt wagen möchten. Die Umsetzung ist wirklich einfach, und ich erkläre Ihnen Schritt für Schritt, was zu tun ist.

Software installieren

Installieren Sie zunächst die Arduino IDE auf Ihrem Computer. Ich habe seinerzeit mit der IDE-Version 1.8.10 gearbeitet, weil neuere Versionen unter *MacOS Monterey* zu Fehlern führten. Die Version 1.8.10 wurde in Arduino-Foren als Lösung empfohlen, und für mich hat sie auch funktioniert. Wer unter Windows unterwegs ist, kann für dieses Projekt aber bedenkenlos die aktuelle Version 2.1 nehmen. Auch auf dem Mac lohnt sich ein Versuch mit der aktuellen Version der IDE. Unsere Screen-

Kurzinfor

- » Mit elektronischer Überwachung zum perfekten Sauerteig
- » ESP32 misst Temperatur, Luftfeuchte und Teighöhe
- » IoT-Anbindung und Visualisierung per Grafana-Cloud

Checkliste



Zeitaufwand:
2 Stunden



Kosten:
ca. 20 Euro

Werkzeug

- » Bohrer, Blechschere für Durchbrüche im Deckel
- » Klebstoff, Klebeband z. B. Heißklebepistole, Sekundenkleber oder doppelseitiges Klebeband
- » Arduino IDE und Internetzugang

Material

- » ESP32-PICO-KIT oder anderes ESP32 Board
- » V4DHT11 Temperatur-/Feuchtigkeits-sensor-Modul
- » HC-SR04 Ultraschall-Entfernungssensor
- » HC-SR04-Halterungen, 2 ×
- » Dupont-Kabel Stecker/Buchse, 7 ×
- » Micro-USB-Kabel
- » USB-Ladegerät
- » Glas für den Sauerteig mittelgroß, mit einem Deckel, der sich durchbohren lässt

Mehr zum Thema

- » Guido Burger und Klaus-Uwe Gollmer, Flusspegel messen mit der Citizen Science Box, Make 1/23, S. 54
- » Carsten Wartmann, Internet-of-Things-Dienste für Maker, Make 3/21, S. 32
- » Heike Jurzik, Der IoT-Bienenstock, Make 5/19, S. 82

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



shots hier kommen von Version 2.1 mit dem *Light (Arduino)*-Theme, damit die Screens nicht so dunkel sind im Druck.

Falls Ihr Betriebssystem den seriellen USB-Anschluss nicht automatisch erkennt, müssen Sie wahrscheinlich noch den VCP-Treiber für

die *CP210x-USB-to-UART-Bridge* installieren (Download siehe Link in der Kurzinfor). Dieser Treiber sorgt dafür, dass Ihr Computer mit dem ESP32-Entwicklungsboard kommunizieren kann – ich zumindest musste diesen Treiber eigens installieren.





Prometheus-Datenbank einrichten

Der nächste, ganz wichtige Schritt: Sie müssen die Datenbank einrichten, in der Sie die Daten von den Sensoren speichern. Dieses Projekt verwendet *Grafana Cloud*, wozu auch eine kostenlose, gehostete *Prometheus* Datenbank gehört und zur Datenvisualisierung

natürlich *Grafana*. Registrieren Sie sich bei *Grafana Cloud* (Link siehe Kurzinfo) und legen Sie ein neues Benutzerkonto an. Wenn das Konto fertig eingerichtet ist und Sie sich dann einloggen, sehen Sie das Portal mit den gehosteten Instanzen von *Grafana* und *Prometheus* 2.

Klicken Sie nun im *Prometheus*-Bereich auf die Schaltfläche *Details*, scrollen Sie etwas runter und legen Sie beim Abschnitt *Password/API Key* per Klick auf *Generate now* einen neuen API-Schlüssel an. Vergeben Sie einen eindeutigen Namen. Als *Role* können Sie *Metrics Publisher* auswählen, *Admin* ist hier nicht nötig und würde auch Sicherheitsrisiken bergen. Speichern Sie diesen API-Schlüssel unbedingt in einer Textdatei, denn Sie brauchen ihn demnächst wieder, um Ihre Daten an die Datenbank zu schicken. Den Nutzernamen und die URL zum *Prometheus* Server sollten Sie auch speichern, diese sind aber leichter wieder von der Weboberfläche abzulesen (über *Send Metrics* schnell zu erreichen).

Sensoren anschließen

Der *DHT11* ist ein einfacher und kostengünstiger Sensor, der aus zwei Komponenten besteht: einem Feuchtigkeitssensor und einem Thermistor, also einem temperaturabhängigen Widerstand. Sie können damit die Daten der Umgebung erfassen, in der Ihr Sauerkeit lebt und wächst.

Wenn Sie ein *DHT11*-Sensormodul gekauft haben, sitzt der eigentliche Sensor auf einem

Breakout-Board mit üblicherweise drei Pins. Keine Sorge, falls es vier sind, dann haben Sie wohl nur den eigentlichen Sensor ohne Platine darunter gekauft. Das ist mir auch schon passiert und stellt kein Problem dar: Sie verwenden den dritten Pin einfach nicht 3. Allerdings kann hier ein 10kOhm Pull-Up-Widerstand nötig sein, der zwischen Datenpin und Vcc geschaltet wird.

Schließen Sie diese Pins so an das ESP32-Board an, wie in Abbildung 4 gezeigt:

- Vcc-Pin an den 3,3-V-Anschluss des Boards
- Daten-Pin an Pin 32 des Boards
- GND-Pin an den GND-Pin des Boards

Der Ultraschallsensor *HC-SR04* 5 misst Entfernungen zwischen 2 Zentimetern und 4 Metern auf rund 3 Millimeter genau. Mit diesem Sensor ermitteln Sie, wie weit der angesetzte Sauerkeit vom Deckel entfernt ist. Anhand dieser Messungen wird berechnet, wie der Teig aufgeht bzw. zusammenfällt. Der *HC-SR04*-Sensor hat vier Pins, die Sie wie folgt an das ESP32-Board anschließen:

- Vcc-Pin an den 5V-Anschluss des Boards
- Trig-Pin (Signal-Sound aussenden) an Pin 4 des Boards
- Echo-Pin (hier lauscht man auf das Echo) an Pin 5 des Boards
- GND-Pin an den GND-Pin des Boards

Nun müssen Sie ein Programm installieren und auf das Board aufspielen, welches die Daten von den Sensoren ausliest und an die *Prometheus*-Datenbank sendet. Schließen Sie das ESP32-Board (mit den angeschlossenen Sensoren) über ein Micro-USB-Kabel an den Computer an.

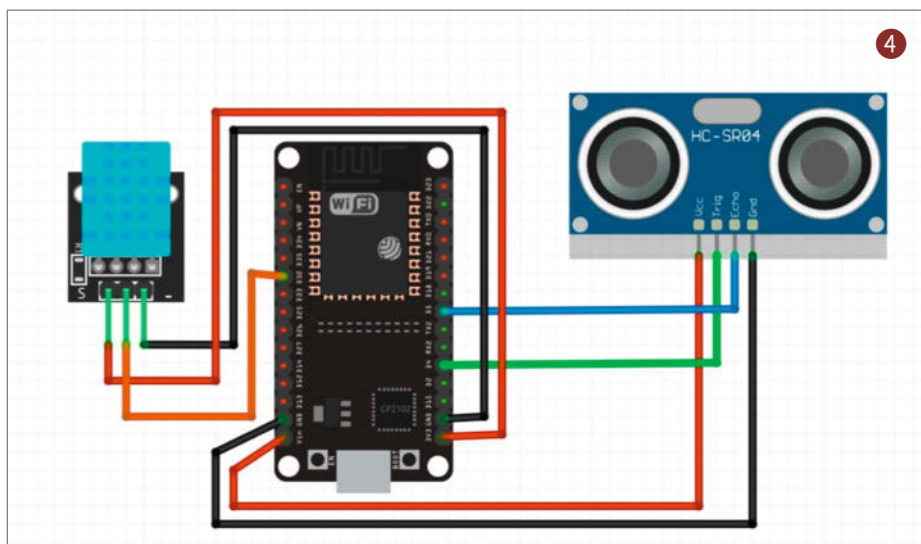
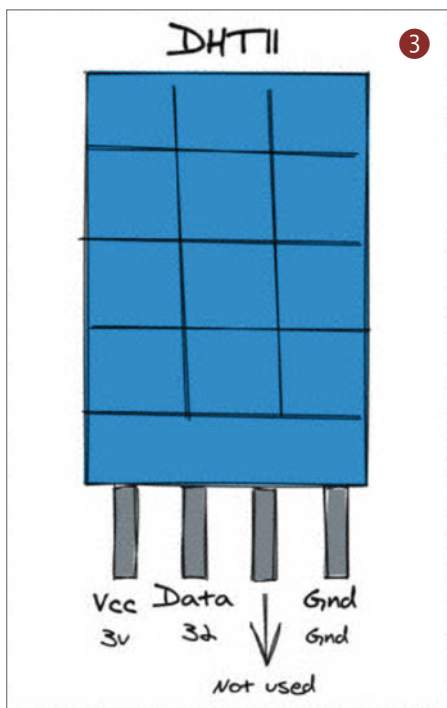
Arduino IDE einrichten

Im Folgenden verwenden wir die *Arduino IDE* 2.1. In 1.x Versionen heißen die Menüpunkte gleich, auch wenn die Dialoge unterschiedlich aussehen. Der bedeutendere Unterschied ist, dass 2.x automatisch die ESP-Boards unterstützt, bei 1.x aber in den Einstellungen der Support für ESP aktiviert werden muss. Dazu rufen Sie *Datei/Einstellungen* auf und geben Sie in das Feld *Zusätzliche Bordverwalter-URLS* die folgende URL ein:

https://dl.espressif.com/dl/package_esp32_index.json.

Allgemeine Anleitungen mit Screenshots und deutschen Menüs, die zeigen, wie man ESP-Boards in die *Arduino IDE* einfügt und Bibliotheken installiert, finden Sie über den Link in der Kurzinfo. In *Arduino 2.x* ist der ganze Prozess ähnlich, aber eher einfacher.

Wählen Sie nun aus der Werkzeugleiste 6 über das Ausklappenmenü den COM-Port für Ihr Board aus und wählen *ESP32 Pico Kit*, wenn Sie dasselbe Board wie ich benutzen, ansonsten den passenden Eintrag zu dem von Ihnen verwendeten Board. Oft ist es das *ESP32 Dev*



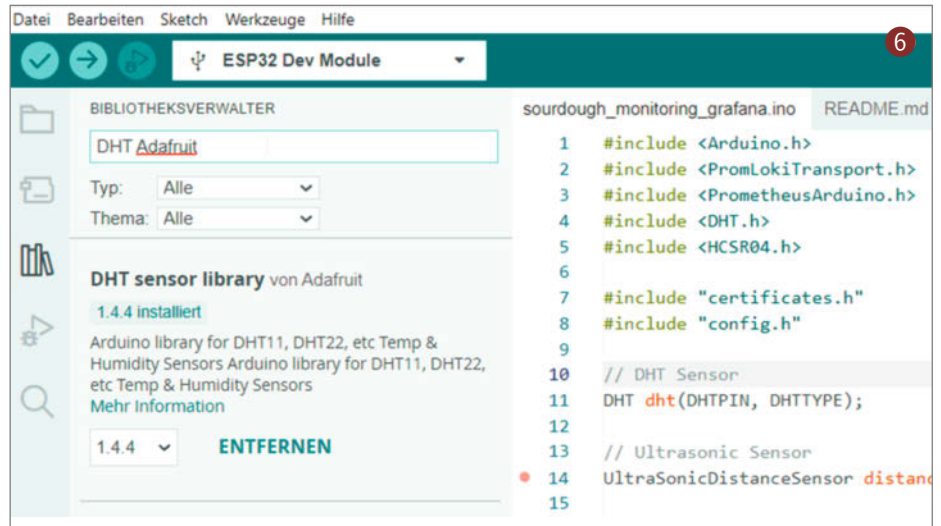
Module wie in unserem Screenshot. Dies teilt der Arduino IDE mit, welches Profil und welche Basisbibliotheken beim Kompilieren des Firmware-Images verwendet werden müssen und wie das Ganze auf das Board geflasht wird.

Rufen Sie *Sketch/Bibliotheken einbinden/Bibliotheken verwalten* auf oder benutzen Sie den Bibliotheksverwalter in der Seitenleiste **6** und installieren Sie die folgenden Bibliotheken:

- die *DHT sensor library* von *Adafruit* samt der davon abhängigen Bibliothek *Adafruit Unified Sensor*
- *HCSR04* von *Martin Sosic*
- *ArduinoBearSSL* von *Arduino* samt abhängiger Bibliotheken
- *ArduinoHttpClient* von *Arduino*
- *PrometheusArduino* von *Ed Welch*, inklusive der davon abhängigen Bibliotheken wie *PromLokiTransport* und *SnappyProto*

Das Programm anpassen

Laden Sie den Arduino-Sketch von *GitHub* herunter (siehe Link in der Kurzinfo). Dort finden Sie drei Dateien. Wichtig sind für uns sind diese beiden: *sourdough_monitoring_grafana.ino* mit dem Programm, und *config.h*, in der die Einstellungen vorgenommen werden. Wenn Sie auf die Erstere im Datei-Explorer (oder im Finder) doppelt klicken, öffnet sich die Arduino



Photovoltaik für Einsteiger

Grundlagen verstehen, Angebote beurteilen, selber bauen

Erweitern Sie Ihr Wissen über Photovoltaik! In unserem **Webinar** lernen Sie die **Grundlagen** zu Photovoltaik-Modulen, Wechselrichtern, Speicher und Auslegung von **Photovoltaik-Anlagen**. Entdecken Sie die verschiedenen Einsatzszenarien und erfahren Sie, wie Sie die **Wirtschaftlichkeit** Ihrer PV-Anlage berechnen können.

Informieren Sie sich jetzt und machen Sie den ersten Schritt in Richtung saubere Energie!

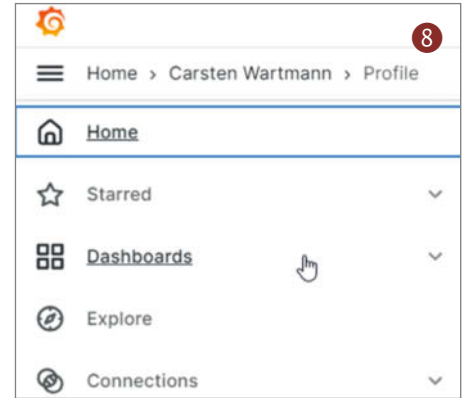
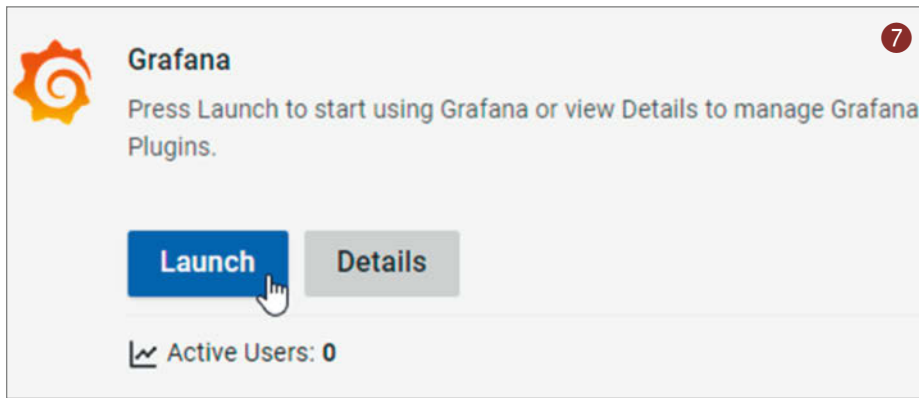
WEBINAR
AM 27.06.2023

ct
WEBINAR



Jetzt Tickets sichern:
webinare.heise.de/photovoltaik





IDE mit den drei Dateien im Ordner. Die dritte Datei *certificates.h* müssen wir nicht bearbeiten.

Wechseln Sie auf den Karteireiter mit der Datei *config.h* und fügen Sie die passenden, bzw. vorher notierten Werte hinzu:

- Name und Passwort Ihres WLAN, damit das ESP32-Board Internetzugang erhält

- GC_PROM_URL muss die URL der Prometheus Instanz enthalten

- Ihren zuvor notierten Nutzernamen für Prometheus (typischerweise eine Ziffernreihe) bei GC_PROM_USER

- Ihren API-Schlüssel bei GC_PROM_PASS

Im Hauptprogramm *sourdough_monitoring_grafana.ino* müssen Sie nichts anpassen, dürfen das aber gerne. Jedem Abschnitt sind Kommentare mit Erklärungen vorangestellt, die den Einstieg erleichtern.

Klicken Sie dann auf *Werkzeuge/Serieller Monitor* (oder das Lupen-Icon in der Werkzeugleiste), um die Debugging-Ausgabe des Boards zu sehen. Stellen Sie unten die Port-Geschwindigkeit auf *115200 baud* ein.

Und schließlich klicken Sie in der Werkzeugleiste der IDE auf die Schaltfläche mit dem Pfeil nach rechts: Damit kompilieren Sie das Programm und spielen es auf das ESP32-Board auf.

Wenn dieser Vorgang abgeschlossen ist, zeigt der serielle Monitor Folgendes an:

- Start und Aufbau der WLAN-Verbindung
- holen der Systemzeit aus dem Internet

- die eingehenden Messwerte und die Kommunikation mit der Datenbank

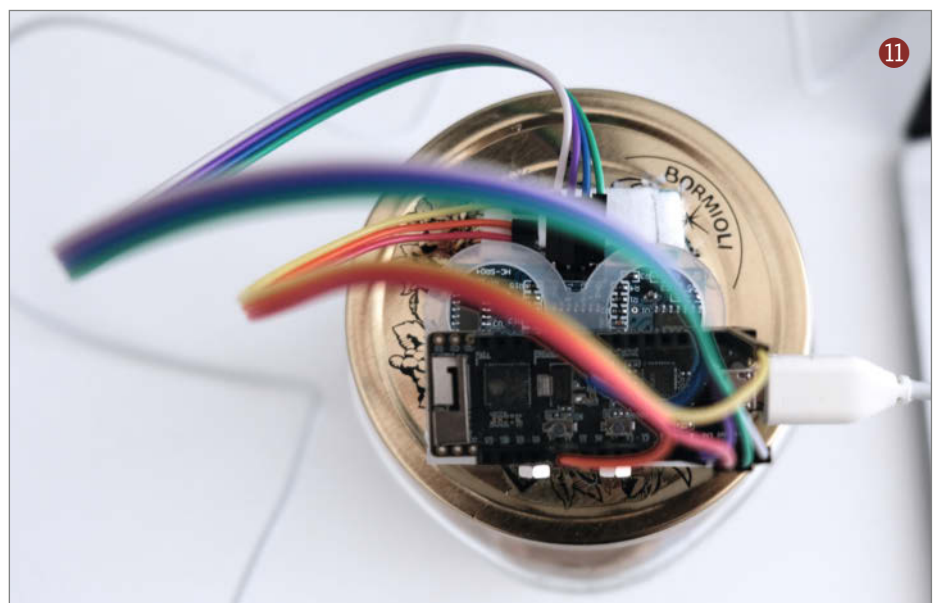
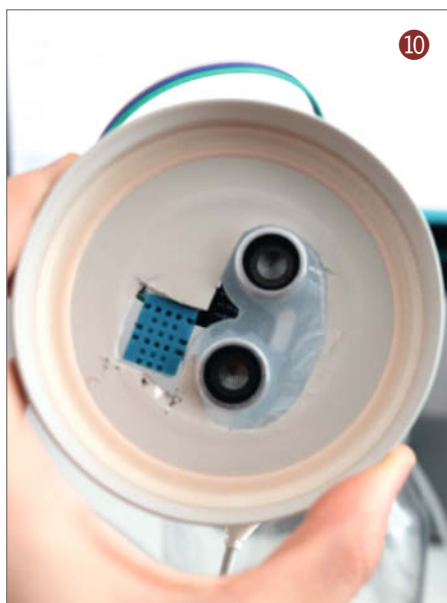
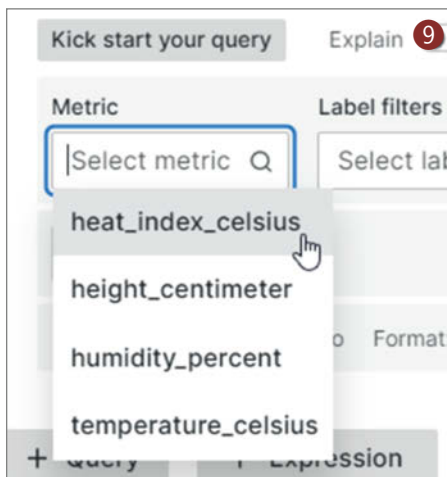
Wenn hier keine Fehlermeldungen erscheinen und Sie zuvor alles korrekt eingerichtet haben, sehen Sie die eingehenden Daten schon bald auch in der Prometheus-Datenbank.

Daten grafisch darstellen

Die Prometheus-Datenbank füllt sich nun mit Sauerteigdaten, die Sie mit Grafana grafisch darstellen können. Öffnen Sie Grafana, indem Sie auf der Seite von Grafana Cloud auf die Schaltfläche *Log In* (7) oder *Launch* klicken (falls Sie noch eingeloggt sind).

Wir verwenden wieder ein helles Design für unsere Screenshots. Diese Einstellung finden Sie in Ihrem Profil unter dem kleinen Avatar-Icon rechts oben im Fenster.

Navigieren Sie zuerst über das *Burger-Menü* (8) zum Dashboard-Abschnitt von Grafana. Dann mit dem blauen *New-Menü* rechts *New Dashboard* auswählen. In dem neuen Fenster können wir dann endlich mit *+ Add visualization* ein Diagramm erstellen.



Im *Edit panel*-Fenster kann man nun seine Datenbankabfragen erstellen. Die Datenquelle (*Data source*) sollte schon richtig auf Prometheus eingestellt sein. Da diese Datenbankabfragen recht komplex sein können, gibt es eine Art Assistenten, der dies wesentlich vereinfacht: Klicken Sie unter dem Punkt *Metrics* auf *Select metric* und in dem erscheinenden Menü **9** kann dann ganz bequem eine der Datenreihen aus der Datenbank gewählt werden. Danach starten Sie mit einem Klick auf den blauen Knopf *Run queries* die Datensammlung und ein Diagramm erscheint im Fenster.

Die Visualisierungen von Grafana sind sehr mächtig, daher empfehle ich für den Anfang erst einmal recht simple Diagramme zu erstellen. Ändern Sie aber am besten minimal den Panelnamen unter *Panel options/Title*. Wenn Sie in der Legende auf den kleinen farbigen Balken klicken, können Sie die Farbe des Graphen ändern. In der sehr langen Liste von Optionen rechts kann gesucht werden: mit *Standard* als Eingabe kommen Sie z. B. zu den *Standard options*, in denen Sie etwa die angezeigten Werte im Feld *Unit* auf *Temperature/°C* ändern können oder durch *Display name* den Bandwurm-Text in der Legende durch etwas Lesbares ersetzen.

Soll die Visualisierung dauerhaft im Dashboard verfügbar sein, müssen Sie diese mit dem *Save*-Knopf speichern! Ein Klick auf den blauen *Apply*-Knopf oben rechts übernimmt die Änderungen und kehrt zum Dashboard zurück. Hier können Sie nun weitere Diagramme (*Add/Visualization*) erstellen oder vorhandene über das Drei-Punkte-Menü in den Diagrammen weiter bearbeiten.

Das Sauerteig-Glas vorbereiten

So weit die Software, jetzt kommt der handwerkliche Teil. Ich zeige hier, wie ich die Elektronik in mein Glas eingebaut habe, aber sicher gibt es noch bessere Möglichkeiten. Sie müssen sich also nicht genau an meine Anleitung halten – machen Sie es so, wie es mit Ihrem Material am besten funktioniert.

Nehmen Sie den Deckel ab und schneiden Sie zwei Löcher hinein: das für den Ultraschallsensor in die Mitte und daneben ein zweites, kleineres Loch für den DHT11-Sensor **10**.

Stecken Sie den Ultraschallsensor in die Halterung für den HC-SR04 und kleben Sie die Halterung von oben auf den Deckel **10**. Kleben Sie den DHT11-Sensor mit doppelseitigem Klebeband so auf den Deckel, dass er in das Glas hinein zeigt. Setzen Sie das ESP32-Board oben in die HC-SR04-Halterung ein. Sie können das Board auch mit einer Heißklebepistole vorsichtig an der HC-SR04-Halterung festkleben. Achten Sie in dem Fall darauf, Kleber nur auf die Befestigungslaschen zu geben, nicht auf das Board selbst.



Das Ganze sollte nun ungefähr aussehen wie in **12**. Das fertige Glas können Sie in die Küche stellen (oder wo Sie Ihren Sauerteig sonst stehen haben) und an das USB-Ladegerät anschließen. Ich habe dem Glas noch eine Haube aus einer abgeschnittenen Plastikflasche übergestülpt, damit der Sauerteig durch die Löcher im Deckel nicht austrocknet.

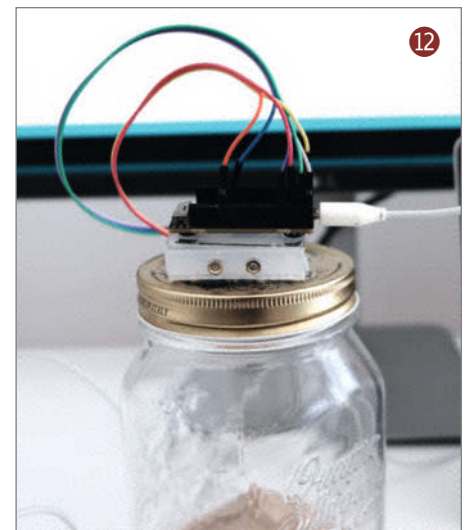
Die Diagramme ablesen

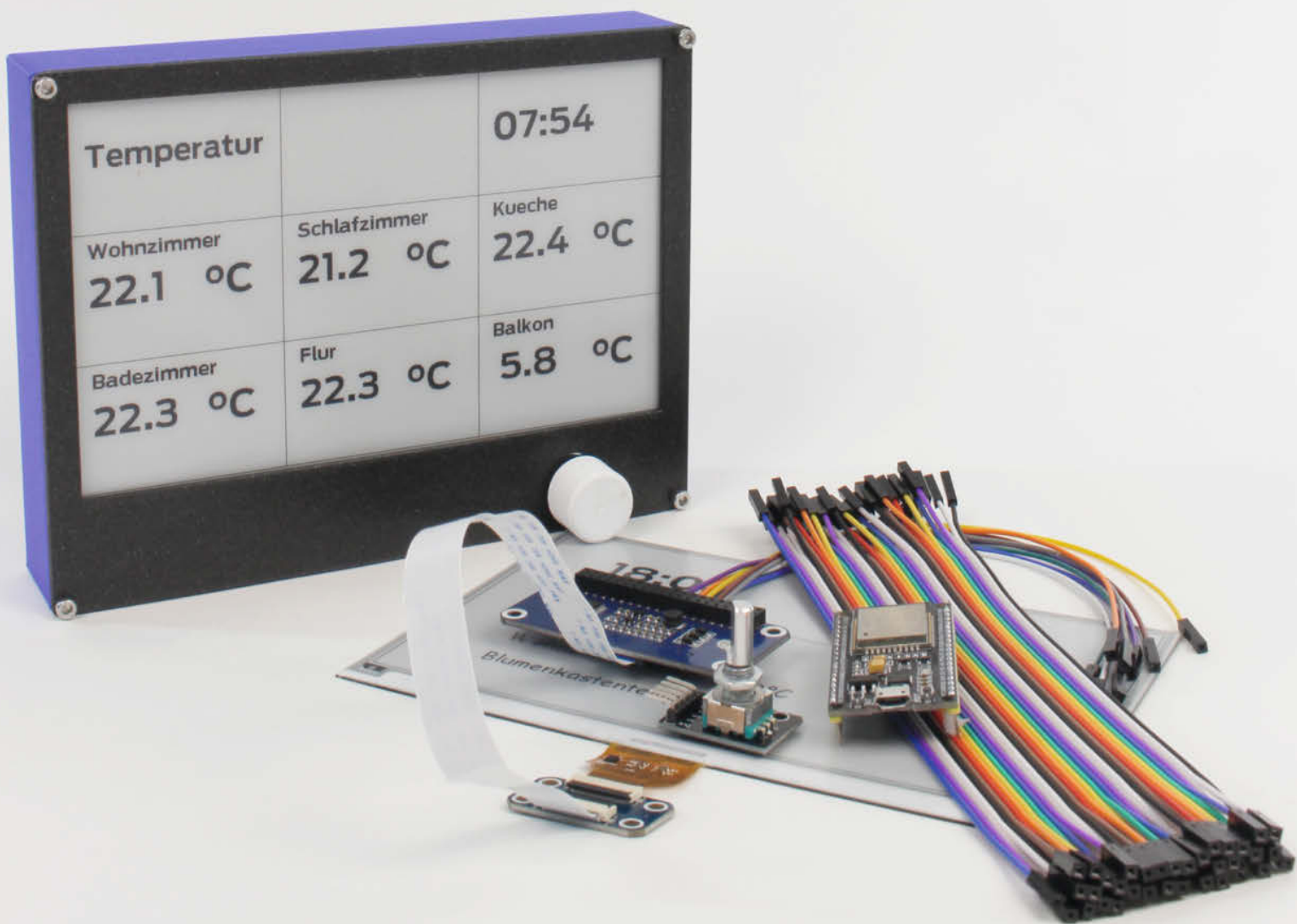
Grafana **13** liefert Ihnen Informationen über die Umgebung, in der Ihr Sauerteig geht, und über den Teig selbst. Die Diagramme zeigen die Temperatur und die Feuchtigkeit im Glas – beide Messwerte können anders sein als in dem Raum, in dem das Glas steht. Je wärmer es ist, desto öfter müssen Sie den Sauerteig füttern.

Ich nutze die Diagramme außerdem noch, um die Höhe des Sauerteigs auszuwerten: Wenn er seine Größe verdoppelt hat und dann wieder kleiner wird, ist es Zeit zum Füttern **14**. Ich habe sogar Benachrichtigungen (*Alerts & IRM*) in Grafana angelegt, damit ich erfahre, wenn es so weit ist.

Es würde mich sehr freuen, wenn Sie das, was ich hier beschrieben habe, ausprobieren und mir schreiben, wie es funktioniert hat. Und falls Sie noch Ideen für witzige und interessante IoT-Projekte haben, lassen Sie es mich unbedingt wissen – ein Tweet an @ivanahuckova reicht.

—caw





Mein Smarthome nimmt mir viele Aufgaben ab: Heizung und Licht werden automatisch gesteuert und berücksichtigen Tageszeit und Anwesenheit von Personen, die Luft im Bad wird bei Bedarf selbsttätig entfeuchtet oder ausgetauscht, selbst Temperaturen und Rauchentwicklung im 3D-Druckerschrank überwacht mein Home Assistant.

Trotzdem habe ich ab und zu das Gefühl, dass es zu warm oder zu kalt ist. Liegt das nun an den fünf Etagen Treppenstufen, die ich gerade hinter mich gebracht habe? Oder stimmt was mit der Heizungssteuerung nicht? Home Assistant zeigt alle Sensorwerte gerne auf seiner Web-Oberfläche oder in der Smartphone-App an. Doch wo steckt eigentlich das kleine Ding? Im Büro vergessen? Und jetzt extra den PC starten? Offenbar fehlt da noch eine ständig verfügbare Smarthome-Info-Zentrale, die nicht erst gestartet werden muss. Ein Display, das mir stets die wichtigsten Werte anzeigt, muss her. Allerdings möchte ich auch kein permanent leuchtendes Nachtlicht, dass mir im Sommer bei offenem Fenster alle in der Dunkelheit nach Blutquellen suchenden Kerbtierchen anlockt.

Ein E-Ink-Display (auch *e-Paper* genannt) erscheint mir da als die beste Wahl: Es leuchtet nicht, sondern zeigt wie ein Stück Papier Texte, Zahlen und Grafiken schwarz auf weiß an. Tagsüber wird es vom Tageslicht beleuchtet, zwischen Sonnenunter- und -aufgang von der Zimmerbeleuchtung, die sich ja ohnehin einschaltet, wenn ich ein Zimmer betrete.

Home Assistant und sein eingebauter Diener *ESPHome* machen es mir da leicht, denn die e-Paper-Displays des Herstellers *Waveshare* werden von Haus aus unterstützt. Diese Displays gibt es in vielen unterschiedlichen Größen von etwa 2,5cm bis über 33cm Diagonale. Bei der Auflösung kann man e-Paper von 128 Pixel × 80 Pixel bis zu 1872 Pixel × 1404 Pixel bekommen. Inzwischen gibt es die nicht nur in schwarzweiß, sondern auch in Farbe. Allerdings unterstützt ESPHome diese Displays noch nicht im Farbmodus. Meine Wahl: ein Schwarzweiß-7,5-Zoll-Display mit 800 Pixeln × 480 Pixeln komplett im Set mit der Ansteuer-Elektronik ¹. Damit kann man das Display direkt an einen Arduino oder ein ESP-Board anschließen.

Als steuernder Controller kam nur ein ESP32 infrage, da der alles für WLAN notwendige enthält und außerdem zahlreiche IO-Pins, sodass man auch noch weitere Funktionen einbauen kann. So sollte ein Rotary-Encoder mit eingebaut werden, mit dessen Hilfe ich dann den Display-Inhalt umschalten kann. Vielleicht kommt später ja noch mehr dazu, wer weiß ... Zur Stromversorgung des Ganzen sollte ein altes Handy-USB-Ladegerät dienen, von denen noch einige in der Bastelschublade liegen (wir stammen halt von Jägern und Sammlern ab). Alle Teile (bis auf das 3D-Druck-Gehäuse und das Netzteil) sind demnächst

Kurzinfo

- » E-Ink-Display-Ansteuerung mit ESP32
- » Textausgabe und Grafik auf dem Display mit ESPHome
- » Anzeige von Home-Assistant-Sensorwerten

Checkliste



Zeitaufwand:

1 Stunden (zzgl. 3D-Druckzeit)



Kosten:

80 Euro

Material

- » ESP32 NodeMCU
- » e-Paper-Display-Hat Waveshare 7,5 inch V2
- » Rotary-Encoder mit Breakout-Board
- » 5 Dupont-Jumperkabel female/female 20cm
- » 14 Schrauben M3 × 16
- » USB-Kabel USB-A auf Mikro-USB
- » USB-Netzteil 1000mA (altes Handy-Ladegerät)

Mehr zum Thema

- » Heinz Behling: Intelligentes Heim mit Home Assistant

Alles zum Artikel
im Web unter
make-magazin.de/xw5p



Werkzeug

- » Schraubendreher
- » 3D-Drucker

auch als Teilesatz erhältlich. Die Bezugsquelle erfahren Sie über den Kurzinfo-Link.

Alles zusammen sollte in einem selbst gedruckten Gehäuse Platz finden, das sich einem Bilderrahmen ähnlich an die Wand hängen oder auf den Tisch stellen lässt.

Rahmen-Gehäuse

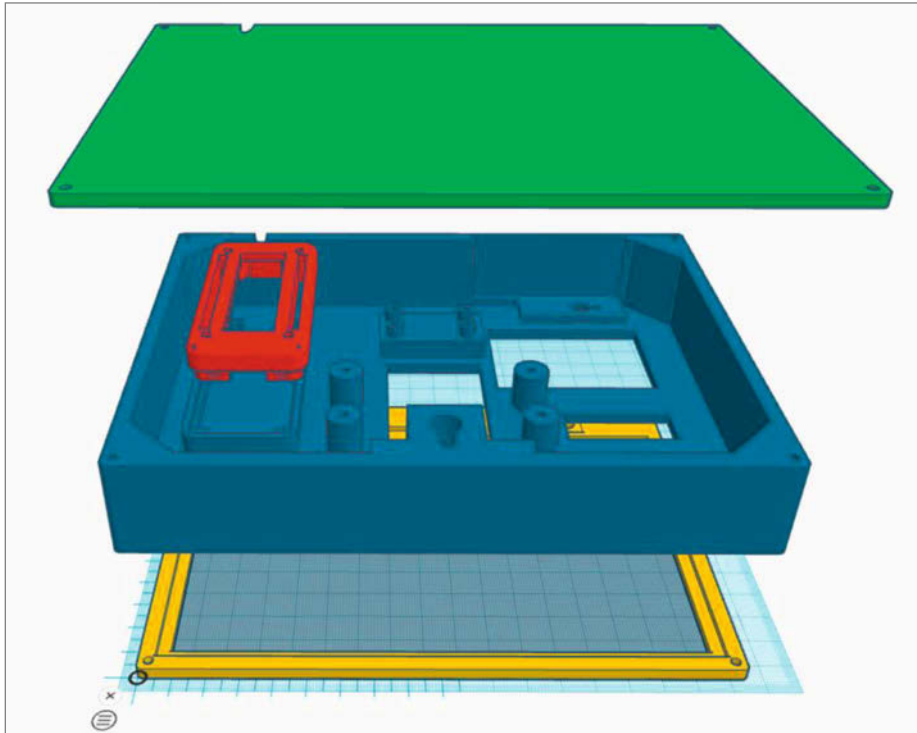
Das Gehäuse besteht aus fünf Teilen ²: Hauptbestandteil ist der große Rahmen (blau),

der das ESP32-Board, die Ansteuer-Elektronik des Displays sowie den Rotary-Encoder aufnimmt. Für die Befestigung des ESPs gibt es einen kleinen Halter (rot), der in ein entsprechendes Gegenstück des Rahmens eingeklickt wird. An der Vorderseite des Rahmens wird das Display mithilfe eines dünnen Frontteils (gelb) befestigt. Dieses Teil enthält eine Aussparung, in die das Display genau hineinpasst.

Außerdem gibt es noch einen Aufsteck-Knopf für den Encoder sowie eine Rückwand



¹ Das Display-Set enthält auch die Steuerelektronik und den Kabelsatz.

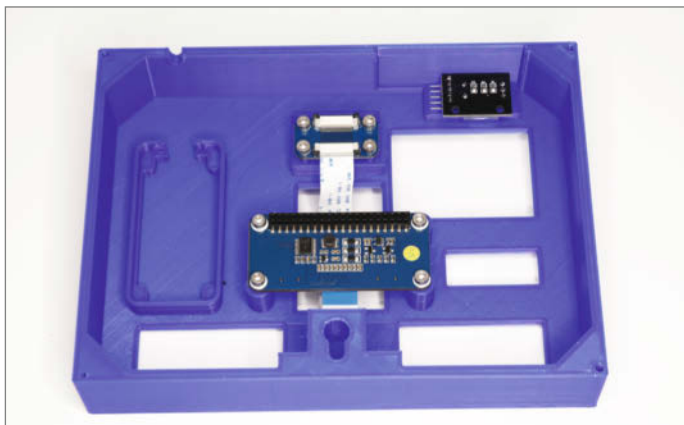


2 Die Teile des Display-Gehäuses: Dazu kommt noch ein Knopf für den Dreh-Encoder.

(grün) für den Fall, dass das Display auf den Tisch gestellt wird. Beim Aufhängen an die Wand ist sie nicht notwendig. Alle Teile sind aus PLA mit 15% Füllung gedruckt. Der große Rahmen braucht Stützstrukturen vom Druckbett aus. Zusammengeschraubt wird alles mithilfe von M3-Schrauben. Die entsprechenden Löcher sind in den Druckteilen vorhanden und so bemessen, dass sich die Schrauben selbst ein Gewinde schneiden.

Das Projekt beginnt man am besten mit dem Einbau des Displays ins Gehäuse. Lösen Sie das Folienkabel des Displays aus der Buchse der kleinen Platine. Dazu muss der braune Sicherungsriegel etwa 1mm aus der Buchse herausgezogen werden. Befestigen Sie die beiden Elektronik-Platinen des Displays mit je vier M3-Schrauben und schrauben Sie den Rotary-Encoder fest 3. Dessen Mutter sitzt etwas vertieft im Gehäuse. Dadurch kann später der Drehknopf an der Front bündig anliegen. Das Auge ist ja auch mit.

Legen Sie dann das Display auf die Vorderseite des Rahmens. Das Folienkabel muss durch die rechteckige Öffnung nach innen gesteckt werden. Bei dieser Gelegenheit: Haben Sie die transparente Schutzfolie vom



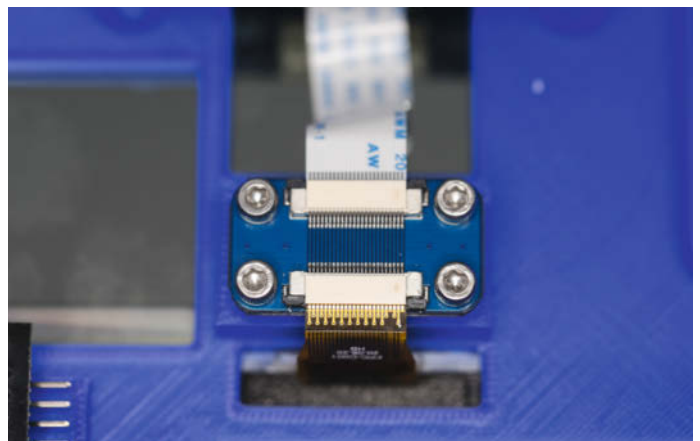
3 Die Display-Elektronik und der Dreh-Encoder im Rahmen



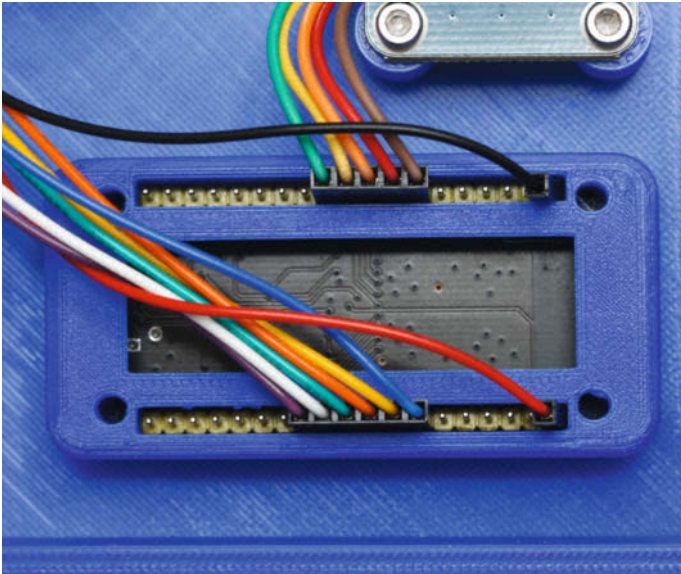
4 Achten Sie auf das Folienkabel!



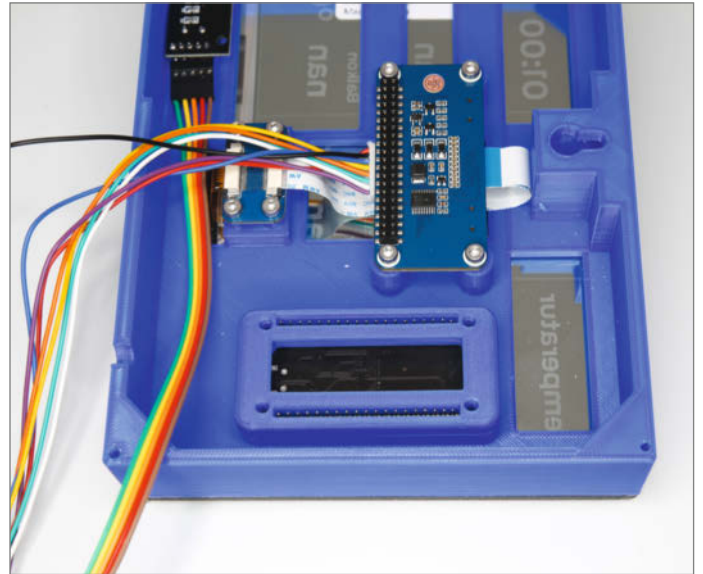
5 Nur wenn das Display richtig im Rahmen liegt, kann das Frontteil lückenlos angeschraubt werden.



6 Das Folienkabel beim Einstecken nicht zu sehr quälen, es ist mechanisch nicht sehr belastbar.



8 Zwischen dem roten beziehungsweise schwarzen Kabel und den jeweiligen restlichen Kabeln müssen jeweils fünf Pins frei bleiben.



7 Das ESP-Board sitzt mit den Pins nach oben in der Fassung. Die Stecker der Kabel passen in die Schlitzte.

Display abgezogen? Falls nicht, ist jetzt die letzte Gelegenheit 4.

Anschließend legen Sie den Frontrahmen auf das Display. Hier bitte darauf achten, dass das Display in der Aussparung des Rahmens liegt, andernfalls kann es später beim nun folgenden Zusammenschrauben (vier M3-Schrauben) Bruch geben 5.

Das Folienkabel des Displays muss nun noch in die entsprechende Buchse auf der kleinen, bereits im Gehäuse sitzenden Platine gesteckt und mit dem dunkelbraunen Riegel gesichert werden 6. Dabei geht es etwas eng zu, eine Pinzette kann da helfen.

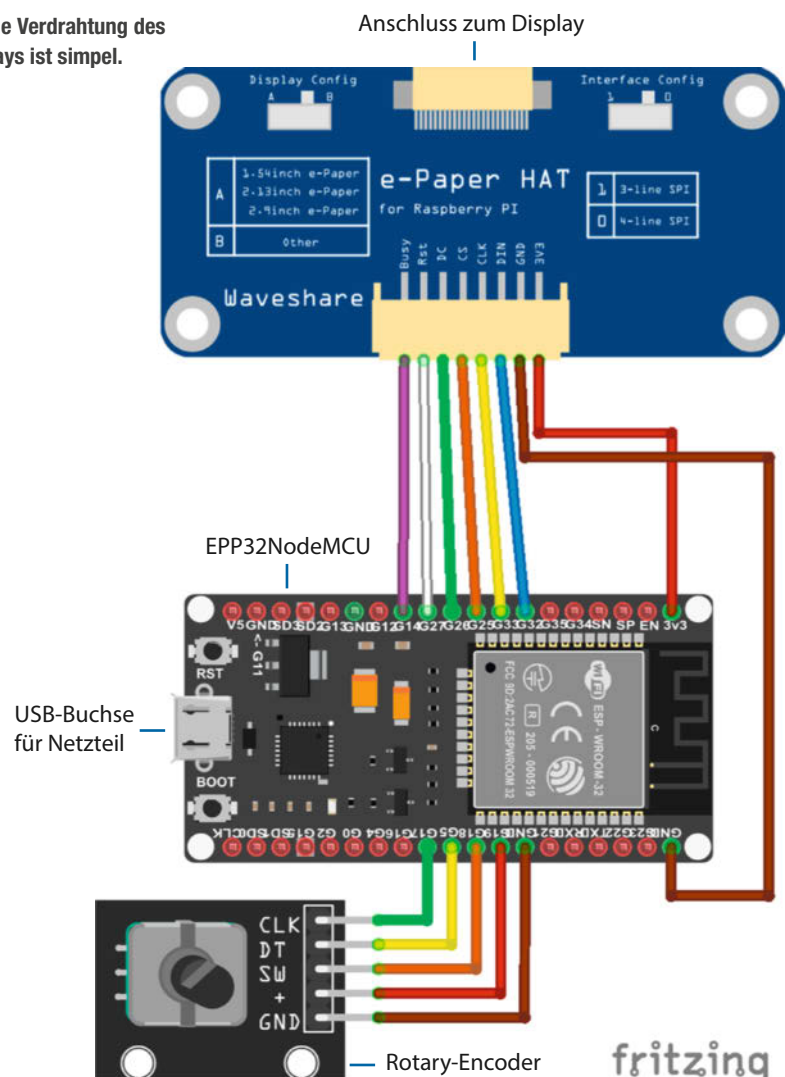
Jetzt geht es an die Verkabelung mit dem ESP32: Die erfolgt am einfachsten mit Jumperkabeln, die an beiden Seiten Buchsen haben (sogenannte female/female-Kabel). Ich bin dabei so vorgegangen: Zuerst habe ich das ESP32-Board in die Fassung im Inneren des Rahmens gelegt. Die USB-Buchse muss dabei zur Seite mit dem Dreh-Encoder zeigen, die Kontaktpins nach oben. Dann habe ich den Halterahmen aufgeklipst 7.

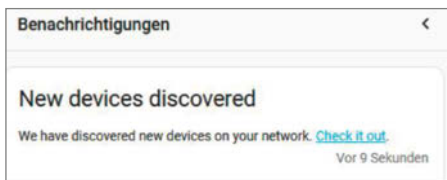
Bei der Schaltung wurden die Pins des ESP32 so ausgewählt, die mithilfe der Jumperkabel eine kreuzungsfreie Verbindung erfolgt. Dazu wurde ein kleiner Trick benutzt, um den Dreh-Encoder mit 3,3V zu versorgen. Darüber gibt es mehr Infos im Software-Teil. Zunächst stecken Sie einfach die Kabel wie in Bild 8 zu sehen auf.

Weitere Details können Sie dem Schaltplan 9 entnehmen.

Jetzt muss nur noch ein USB-Kabel in die Buchse des ESP-Boards gesteckt und mit dem Home-Assistant-Server verbunden werden. Dann kann es ans Überspielen der Firmware gehen.

9 Die Verdrahtung des Displays ist simpel.





10 Home Assistant hat etwas Neues entdeckt.

Software

Ich gehe hier davon aus, dass in Ihrem Smart-home bereits ein Home-Assistant-Server läuft. Er muss außerdem mit dem Add-on ESPHome ausgestattet sein. Falls nicht, holen Sie das unter *Einstellungen/Add-ons* und *Add-on Store* bitte nach. Es gibt drei Versionen von ESPHome. Die beta- und dev-Versionen sollten wegen eventueller Instabilitäten nicht benutzt werden.

Meine Software macht folgendes: Sie stellt zwei Bildschirmseiten zur Verfügung: Die erste zeigt die Temperaturen für Wohn-, Schlaf- und Badezimmer sowie Küche und Flur an. Außerdem steht noch die auf dem Balkon gemessene Außentemperatur zur Verfügung. Die zweite enthält jeweils die Luftfeuchtigkeit für dieselben Messstellen. Zwischen beiden wird mit dem Rotary-Encoder umgeschaltet: Jeder Klick damit schaltet jeweils eine Seite weiter. Die Software ist, wie in ESPHome üblich, in YAML geschrieben und so konstruiert, dass sie um weitere Seiten erweitert werden kann und der Encoder die dann auch berücksichtigt.

Über den Link in der Kurzinfor können Sie die Datei *epaper.yaml* herunterladen. Da die

aber mit den Sensor-Bezeichnungen meines Smarthomes arbeitet, kann sie Ihnen nur als Mustervorlage dienen. Sie müssen dort die Namen der Sensoren in Ihrem Heim eintragen. Die finden Sie auf der Home-Assistant-Webpage Ihres Servers unter *Einstellungen/Geräte & Dienste/Entitäten*.

Doch zunächst einmal müssen Sie das ESP32-Board des Displays dem Server bekannt machen: Wählen Sie dazu auf der Server-Webpage unter *ESPHome New Device*. Mit *Continue* geht es weiter zur Namensvergabe für das Board: Hier sind nur Kleinbuchstaben, Ziffern und der Unterstrich erlaubt. Mein Display habe ich *epaper* genannt. Mit *Next* gelangen Sie zur Board-Auswahl. Klicken Sie auf *Pick specific board* und wählen Sie *AZ-Delivery ESP-32 Dev Kit VC V4*. Weiter mit *Next* wird Ihnen der Verschlüsselungscode angezeigt. Klicken Sie auf *Install* und schon wird ein erstes Grundgerüst der Firmware auf dem ESP-Board installiert. Diese Firmware enthält bereits alles, um weitere Firmware-Updates übers WLAN überspielen zu können. Sie können daher das USB-Kabel aus der Buchse des Home-Assistant-Server herausziehen und in ein USB-Netzteil stecken.

Kurze Zeit später zeigt Ihnen der Home Assistant unter *Benachrichtigungen* an, dass er eine neue Komponente entdeckt hat 10.

Klicken Sie auf *Check it out* und dann auf *Konfigurieren*. Nach weiteren Klicks auf *Absenden* und *Fertig* ist das ESP32-Board in der Lage, mit dem Server zu kommunizieren.

Allerdings weiß der ESP32 noch nicht, was er eigentlich tun soll, denn dieser Teil der Firmware fehlt noch. Im ESPHome-Fenster können Sie die Firmware per Klick auf *edit* unter *epaper*

editieren. Sie endet bislang mit der Zeile *captive_portal*:

Darunter muss nun die Liste der Sensoren angelegt werden, deren Werte das Display anzeigen soll. Daher folgt als Erstes die Zeile *sensor*:. Für jeden Sensor ist darunter dann ein Eintrag nach dem Schema 11 notwendig. Der Begriff *flurtemp* ist der Name, unter dem der Sensor im weiteren Verlauf der Firmware angesprochen wird. Er ist frei wählbar. Der kompliziert aussehende Begriff *sensor.temperature_humidity_sensor_2dd9_temperature* ist der Entitätsname des Temperatursensors. Den müssen Sie durch die entsprechende Bezeichnung in Ihrem Smartphone ersetzen.

Legen Sie so für jeden Sensor jeweils einen vierzeiligen Eintrag an. Wichtig: Die Einrückungen zu Beginn der Zeilen sind sehr wichtig. Schauen Sie sich am besten die Musterdatei an, um den Programmtext entsprechend anzuordnen. Leider ist die Datei mit insgesamt fast 200 Zeilen zu lang, um sie hier abzdrukken.

Als letzter Sensor wird der Rotary-Encoder angelegt 12. Darin werden zunächst die PIN-Nummern definiert, an die der Encoder angeschlossen ist. Schließlich wird unter *on_clockwise* beziehungsweise *on_anticlockwise* festgelegt, was zu tun ist, wenn der Drehknopf im Uhrzeigersinn oder entgegengesetzt gedreht wird: Es wird jeweils die nächste beziehungsweise vorige Seite angezeigt und ein Display-Update gemacht, damit der neue Bildinhalt auch erscheint.

Das ESP32-Board hat nur einen Ausgangspin für die Versorgungsspannung von 3,3V. Aber sowohl das Display als auch der Dreh-Encoder brauchen diese Spannung. Um Lötarbeiten zu vermeiden, wird einfach der noch freie Pin 19 als Ausgang geschaltet. Somit liegen dort 3,3V an. Der Strombedarf des Dreh-Encoders ist sehr gering. Daher kann man ihn über diesen Ausgangspin versorgen. Deshalb führt im Schaltplan die +-Leitung des Encoders zu Pin 19. Der Programmcode 13 sorgt dafür, dass dort die richtige Spannung anliegt.

Für die Anzeige auf dem Display braucht man Zeichensätze, auch Fonts genannt. Hier kann man aus dem Vollen schöpfen, denn alle Fonts, die Sie auf Ihrem Computer haben, kön-

11 Mustereintrag für Sensoren

```
- platform: homeassistant
  id: flurtemp
  entity_id: sensor.temperature_humidity_sensor_2dd9_temperature
  internal: true
```

12 Sensor-Eintrag für den Rotary-Encoder

```
- platform: rotary_encoder
  name: "Rotary Encoder"
  pin_a: 17
  pin_b: 5
  min_value: 1
  max_value: 2
  publish_initial_value: True
  id: rotary

  on_clockwise:
    - display.page.show_next: my_display
    - component.update: my_display
  on_anticlockwise:
    - display.page.show_previous: my_display
    - component.update: my_display
```

13 Der Trick mit der Stromversorgung

```
switch:
  - platform: gpio
    pin:
      number: 19
      inverted: true
    id: e_voltage
```

14 Fonts

```
font:
- file: 'fonts/Antenna-Bold.ttf'
  id: font1
  size: 36
```

nen vom Display benutzt werden. Sie müssen nur die Font-Datei in ein Verzeichnis auf dem Home-Assistant-Server kopieren, das im Konfigurationsordner liegt. Ich habe es passenderweise *fonts* genannt. In der Firmware müssen Sie die Fonts wie in 14 angegeben einsetzen. Sie können beliebig viele Fonts verwenden, müssen ihnen aber jeweils eine eigene id geben. Sie können auch immer denselben Font verwenden, aber in verschiedenen Größen, ganz nach Belieben.

Unter *display*: folgt die Festlegung der Pins für den Bildschirm und das Update-Intervall. E-Paper-Displays sollten aus technischen Gründen regelmäßig refresht werden, solange die Betriebsspannung anliegt, sonst können Sie laut Hersteller Schaden nehmen. Das erledigt im Minutenabstand der Befehl `update_interval: 60s`.

Schließlich folgt noch die Definition der Bildschirmseiten. Sie erfolgt nach dem Schema 15. Die Seiten müssen durchgehend nummeriert sein. Die eigentlichen Ausgabebefehle erfolgen im Lambda-Format, das ist C-Code innerhalb des YAML-Quellcodes. Die Musterdatei zeigt, wie man damit Texte und formatierte Zahlen positioniert und ausgibt. Außerdem lassen sich so auch Linien zeichnen, was für eine Aufteilung des Bildes in insgesamt neun Felder genutzt wird. Wie bereits erwähnt gibt so mein Display Temperaturen 16 und Feuchtigkeitswerte 17 aus.

Haben Sie alle Sensoren, Fonts usw. eingetragen? Dann können Sie die Firmware per Klick auf *Install* und *Wirelessly* kompilieren

15 Das Seiten-Schema

```
id: my_display
pages:
- id: page1
  lambda: |-
    int breite = 799;
    int hoehe = 479;
    it.line(0, 0, breite, 0);
    it.line(breite, 0, breite, hoehe);
    it.line(breite, hoehe, 0, hoehe);
    it.line(0, hoehe, 0, 0);

    it.line(2*(breite/3),hoehe,2*(breite/3),0);
    it.line(breite/3,hoehe,breite/3,0);
    it.line(0,hoehe/3,breite,hoehe/3);
    it.line(0,2*(hoehe/3),breite,2*(hoehe/3));

    it.printf(20,50, id(font1), "Temperatur");

    it.strftime(552, 100, id(font3),TextAlign::BOTTOM_LEFT,
      "%H:%M", id(hass_time).now());

    it.printf(20,(hoehe/3)+10, id(font2), "Wohnzimmer");
    it.printf(20,(hoehe/3)+50, id(font3), "%4.1f",
      id(wohntemp).state);
    it.printf(165,(hoehe/3)+50, id(font3), "°C");

    .
    .
    .

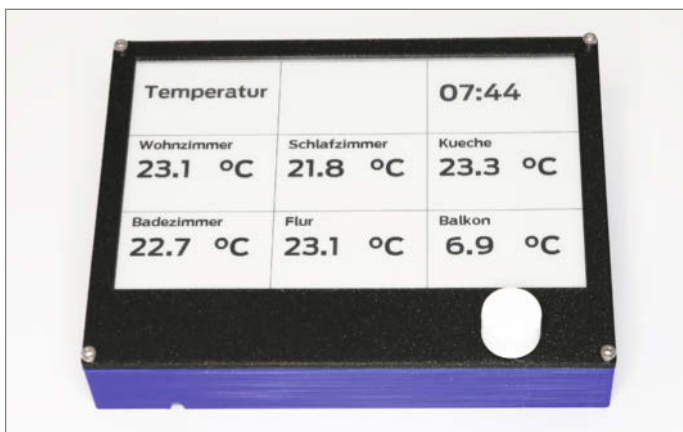
- id: page2
  lambda: |-
    int breite = 799;
    int hoehe = 479;
    it.line(0, 0, breite, 0);
    it.line(breite, 0, breite, hoehe);
    it.line(breite, hoehe, 0, hoehe);
    it.line(0, hoehe, 0, 0);

    .
    .
    .
```

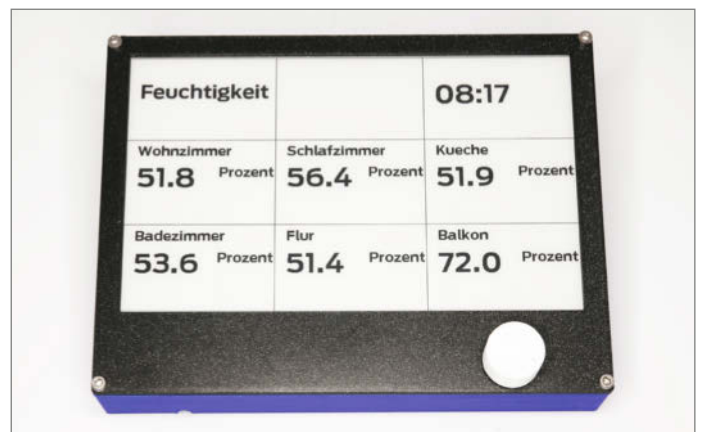
und aufs Display übertragen. Falls es dabei zu Fehlermeldungen kommen sollte, sind fast immer diese (vermaledeiten) Einrückungen Schuld: Die müssen genau eingehalten werden, also alle exakt so untereinander stehen wie in der Musterdatei.

Wenn Ihr Fluchen über diese Art der Quellcode-Formatierung erstmal nachgelassen hat,

werden Sie feststellen, dass Sie noch eine Menge mehr mit dem Display anstellen können. In der ESPHome-Online-Dokumentation erfahren Sie zum Beispiel, wie man auch Bilder anzeigen lassen kann. Falls Sie Ihre Messwertanzeige zum Beispiel noch mit Icons ausstatten, schicken Sie uns doch ein Bild davon. Viel Spaß dabei. —hgb



16 Seite 1 zeigt die Temperaturen ...



17 ... und Seite 2 die Luftfeuchtigkeit.

THE 100: Der schnellste 3D-gedruckte Drucker der Welt

Der THE 100 ist ein Open-Source-Projekt und druckt das Testmodell Benchy in sechs Minuten, das ist Platz 26 in der Weltrangliste. Dabei ist der Drucker-Rahmen praktisch komplett aus PLA gedruckt.

von Matthäus Szturc



Als 3D-Druck-Fan braucht man Geduld. Selbst kleine Modelle benötigen oft mehrere Stunden Druckzeit und bei etwas größeren Modellen kann es auch mal mehrere Tage dauern, bis ein Druck fertig ist.

Um meine und eure Geduld nicht allzu sehr zu strapazieren, habe ich einen 3D-Drucker entwickelt, der den Fokus auf schnelles Drucken legt, ohne dabei die Qualität zu vernachlässigen. Auch der Bauraum ist mit $165 \times 165 \times 150\text{mm}$ groß genug für praktische Aufgaben. Das Projekt, das ich unter dem Entwicklungsnamen *THE 100* gestartet habe und dessen Ziel es war, 100k, also 100.000mm/s^2 Beschleunigung zu erreichen, wurde am 15.03. veröffentlicht.

Mit Geschwindigkeiten von 600mm/s , Beschleunigungen von bis zu 100.000mm/s^2 und einem Volumenstrom von $43\text{mm}^3/\text{s}$ gehört der Drucker zu den schnellsten 3D-Druckern, die an der *Speedboat Challenge* teilgenommen haben. Mit einer Zeit von 5:40min erreichte der Drucker Platz 26 in der Weltrangliste.

Das Besondere an diesem Drucker ist, dass er ohne Aluminiumprofile auskommt, stattdessen wird der komplette Rahmen selbst gedruckt und besteht aus PLA. Dieser Ansatz ist neu und ermöglicht es, die Motoren in den Rahmen zu integrieren, die Riemenwege zu verkürzen und damit die Belastung der Konstruktion zu reduzieren. Weitere positive Nebeneffekte sind geringere Vibrationen, ein höheres Drehmoment und damit höhere Geschwindigkeiten.

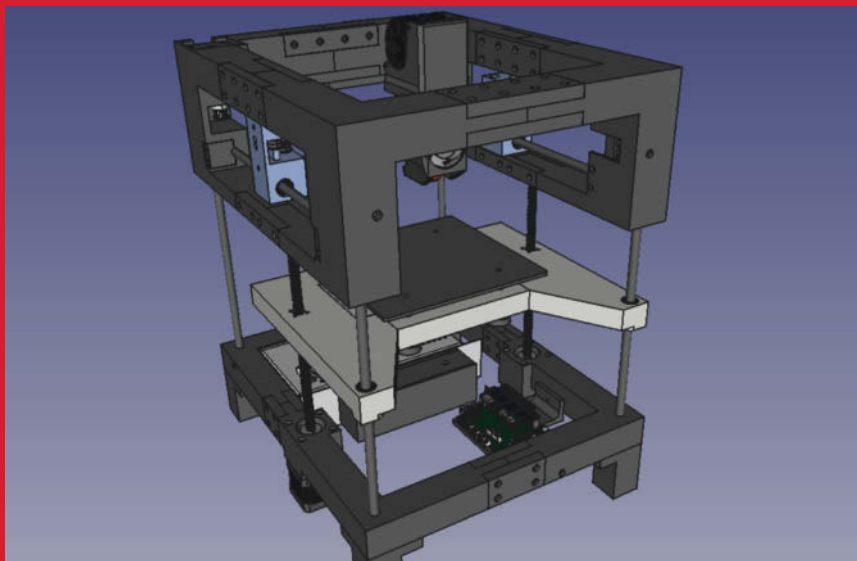
Das Projekt ist vollständig Open Source und kann daher von jedermann nachgebaut und modifiziert werden. Alles, was dazu benötigt wird, also CAD-Dateien, 3D-Druckdateien, Stücklisten, Firmware und Slicer-Profile, ist frei verfügbar. Eine Videoreihe, die vom Team rund um den *THE 100* produziert wurde, erklärt Schritt für Schritt den Aufbau des Druckers. Damit wird auch enthusiastischen Einsteigern das Bauen von 3D-Druckern ermöglicht.

Mit Materialkosten von unter 350 Euro bietet der Drucker einen kostengünstigen Einstieg in die Welt des *Speed-Printing*. Bis zum Erscheinen dieses Artikels haben sich bereits mehr als 400 Menschen entschieden, einen *THE 100* zu bauen. Im *Discord*-Chat tauschen sich viele talentierte Konstrukteure über Modifikationen, Verbesserungen, neue Teile und ihre Erfahrungen aus.

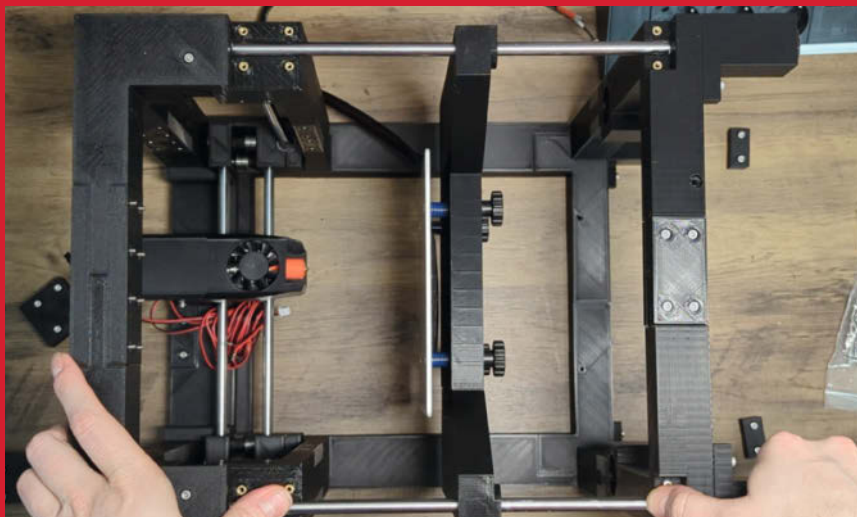
Bisher ist mein *THE 100* noch der schnellste 3D-Drucker der Welt, der auf einem gedruckten Rahmen basiert. Aber das wird sich ändern. Was als Ein-Mann-Projekt begann, ist heute eine Community von über 2000 Menschen, die täglich wächst und im Sinne des Open-Source-Gedankens ihre Veränderungen mit der Community teilt. Es wird nicht mehr lange dauern, bis einer von ihnen es schafft, meinen *Speed-Benchy-Rekord* zu brechen.

—caw

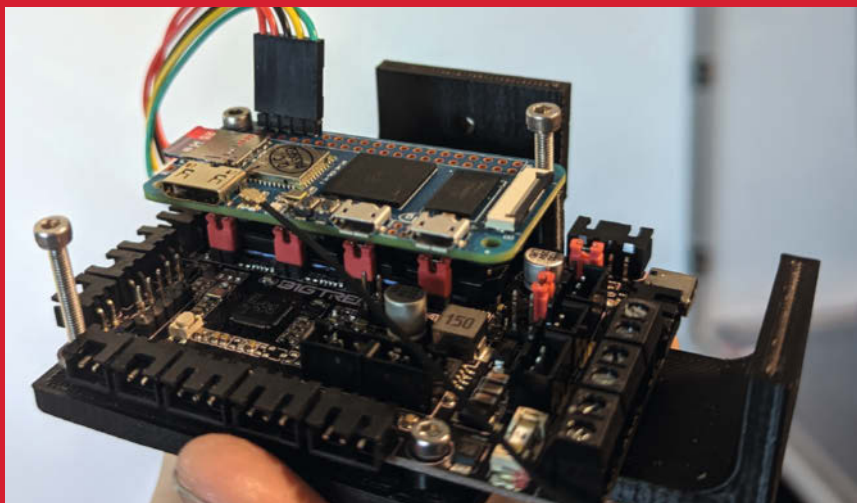
► github.com/MSzturc/the100



Die Konstruktionspläne des Druckers sind als STEP-Dateien frei verfügbar. Dadurch können Änderungen am Druckerdesign einfach durchgeführt werden.



Das Druckbett wird über zwei Gewindespindeln angetrieben und über zwei Wellen stabilisiert.

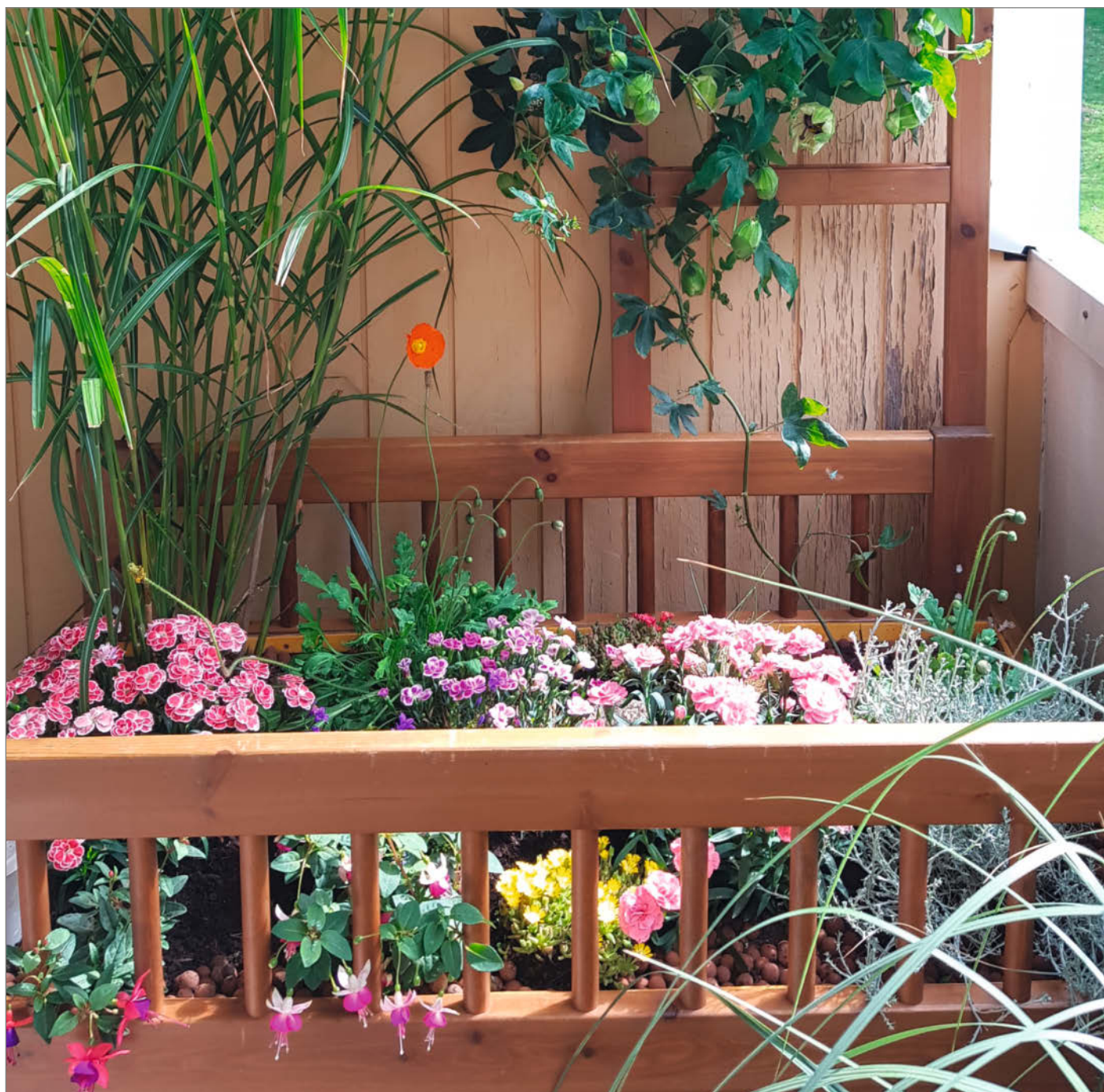


Das Gehirn des Druckers: Über UART sendet ein Banana Pi in Echtzeit Druckbefehle an die darunter liegende Controllerplatine.

Ikea-Hack: Hochbett zu Hochbeet

Wie ein Etagenbett aus Massivholz nach 15 Jahren eine völlig neue Aufgabe bekommt:
Mit viel Ideenreichtum und kreativer Verwendung von vorhandenem Material wurde ein funktionales und ansehnliches Hochbeet für den Balkon geschaffen.

von Roman Zajonz



Es war einmal ein Etagenbett von Ikea, das unseren beiden Söhnen 15 Jahre lang hervorragend gedient, dabei aber auch gelitten hat – mal als Muckibude, mal als Oase der Ruhe. Es wurde zusammengeschoben und wieder auseinandergenommen und hat doch all die Jahre einwandfrei funktioniert. Jetzt ist es Zeit für einen Wechsel, ein neues Bett muss her. Wohin mit dem alten? Ein Ende in der Müllverbrennung hatte das Bett auf keinen Fall verdient, aber für das völlig intakte Bett interessierte sich scheinbar niemand.

So entstand die Idee, diesem wunderbaren Möbel ein zweites Leben zu schenken. Zunächst mussten einige Teile nach Plan zugeschnitten werden. Von den vorhandenen Bettteilen wurden die Eckwinkel und Teile des Lattenrostes nach einer Behandlung mit Holzgel gegen Feuchtigkeit angepasst. Alles wurde mit vorhandenen Schrauben zusammengebaut.

Aus Resten wurde eine Grundplatte mit Bohrungen für den Wasserablauf hergestellt. Auf die Grundplatte wird ein Edelstahlnetz gelegt, damit das Holz atmen kann und eventuelles Kondenswasser oder Wasserreste durch die Löcher in der Grundplatte abfließen können.

Als Wärmedämmung und Schutz für die nächste Schicht wurden noch vorhandene Reste einer Laminat-Trittschalldämmung verwendet. Darüber wurde eine dicke Folie als Wassersperre verlegt. Alte Dias und Diahalter wurden als unverrottbare Drainageschicht verwendet. Die Dias wurden natürlich vorher digitalisiert.

Mit den montierten Rädern bleibt das Beet-Bett mobil, um die gefüllte Tonne schweren Beete zur Reinigung zu bewegen. In der Zwischenzeit ist es schon bepflanzt, aber zwischen der Drainage und der Erde sind noch Gartenfolien eingelegt worden, sodass die Drainage frei von Erde und Wurzeln bleibt.

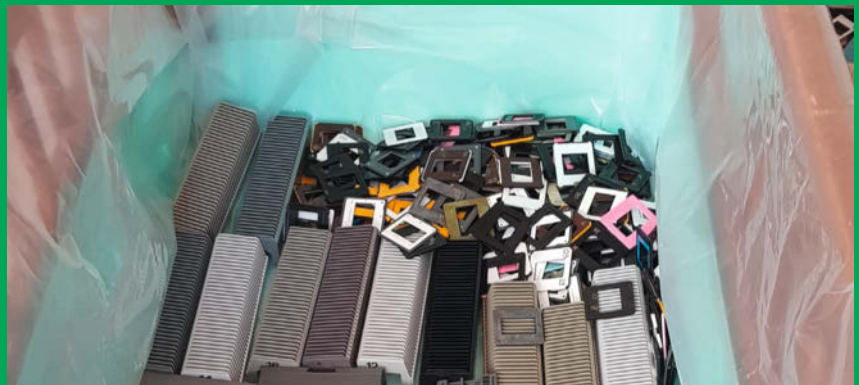
Bis zur Wassersperre (Folie) wurde ein Rohr PG25 verlegt, um den Wasserstand in der Drainage zu kontrollieren. Mit einer kleinen Handpumpe kann das Wasser wieder auf die Beetoberfläche gepumpt werden, sodass den Pflanzen kein Wasser verloren geht. Die Ränder der Folie wurden mit gegen Feuchtigkeit behandelten Leisten an das Beet geschraubt und rundum mit Blähton aufgefüllt, damit keine direkte Verbindung zwischen Erde und Holzleisten besteht.

Die vielen Reste wie die Leiter für das obere Bett und auch andere Konstruktionsteile konnte ich als Kletterhilfe verwenden. Da die Konstruktion kein Fertigprodukt aus dem Baumarkt ist, kann man nach Lust und Laune experimentieren, z. B. mit LED-Beleuchtung oder automatischer Bewässerung: Im Sommer bei 30 Grad Celsius verbraucht das Hoch-Beet-Bett nämlich rund fünf Liter Wasser pro Tag! Mehr Details und Bilder finden Sie im Link. —caw

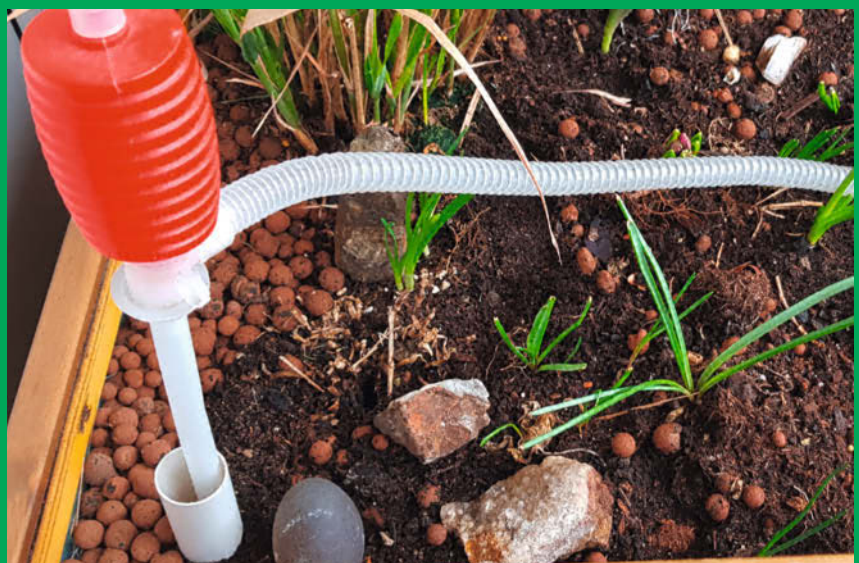
► github.com/MakeMagazinDE/Beetbett



Die Einzelteile werden zurechtgeschnitten und wieder zusammengesetzt, sodass der Rahmen des Hochbeetes genau auf den Balkon passt.



Wärmeisolation aus Trittschalldämmung, Sperrschicht aus dicker Folie und Drainage aus alten Diarahmen und -haltern



Eine einfache Handpumpe verteilt das Wasser, das sich am Boden des Hochbeetes sammelt, wieder auf der Oberfläche.



Der X-Wing zum Anbeißen – ein Geburtstagstraum für Star-Wars-Fans

Über ein Raumschiff zum Geburtstag freut sich doch jeder Star-Wars-Fan. Wie man eines baut, das man obendrein noch vernaschen kann, zeigen wir euch in diesem Artikel.

von Thomas Gamisch

Mit *Star Wars* hat *Lucas Film* ein Fantasy-Universum erschaffen, das Generationen verbindet. Selbst viele Erstklässler kennen inzwischen sämtliche Charaktere, Raumschiffstypen, Waffen sowie den Plot der Filme so gut, dass sie in einer *Star Wars*-Quiz-Show die meisten Erwachsenen problemlos in die Tasche stecken würden – und dies ohne je einen der Originalstreifen (FSK12) selbst gesehen zu haben, versteht sich!

Klar, dass an Kindergeburtstagen *Star Wars* schwer angesagt ist. Wer den Bonsai-Sternenkriegern eine coole Überraschung bereiten will, kann sich mal an einem essbaren *X-Wing-Fighter* versuchen. Die Baumaterialien dazu gibt es in jedem gut sortierten Supermarkt, als Raumschiffswerft dient die häusliche Küche. Für die richtigen Proportionen habe ich mir ein passendes Blueprint des *Ur-X-Wing T-65* aus dem Internet geladen.

Waffeln als Baumaterial

Das X-Wing-Modell besteht aus zusammengeklebten Vanillecreme-Waffeln, die wir zuerst zurechtschneiden müssen. Dafür eignet sich ein Messer mit Sägezahn-Schneide oder Wellenschliff. Wenn man beim Bearbeiten sanft und ruhig vorgeht, bricht auch so schnell nichts ab.

Für jeden der vier Flügel benötigen wir zwei Waffeln, wobei wir jeweils eine Waffel an der hinteren Seite anschrägen. Die Laserkanonen bestehen aus einer halben Crispy-Roll eines bekannten Milchstraßen-Herstellers, in die ein Zahnstocher eingeschoben wird. Ein Stück Weintraube bildet die Parabolantenne an der Mündung des Kanonenlaufs.

Den Rumpf modellieren wir in Vollmaterialbauweise aus insgesamt 6 Waffelschichten ❶. Vollmaterial ist stabiler und von der Konstruktion her einfacher zu verarbeiten. Auch lassen sich kleine Fehler später leicht



❶ Das Waffel-Vollmaterial besteht aus 6 Schichten.

mit einer Raspel korrigieren. Allerdings erhöht sich auch das Gewicht des Modells. Damit der Rumpf stabil wird, müssen die Stoßfugen der Waffeln von Schicht zu Schicht versetzt angeordnet werden wie beim Mauerbau.

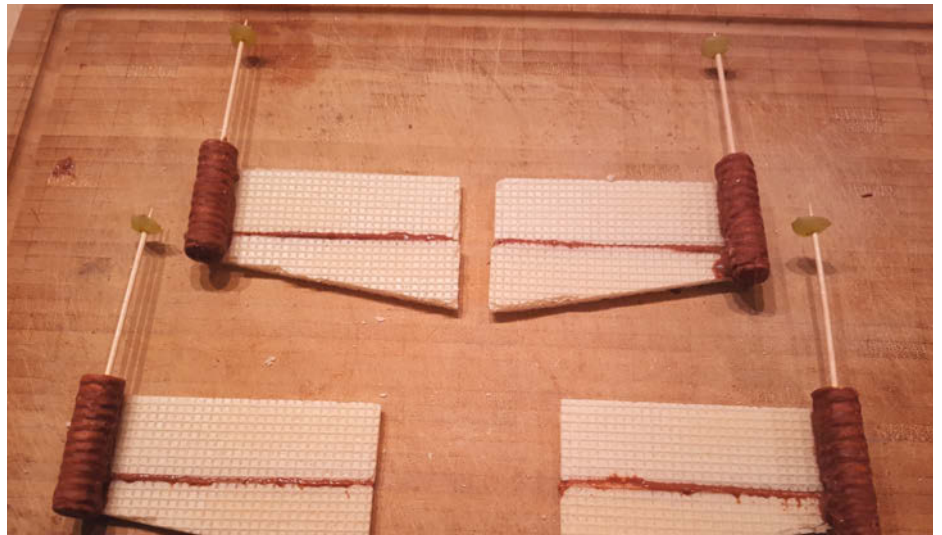
Schokolade als Kleber

Die Waffelschichten verkleben wir mit Kuvertüre, wie sie zum Überziehen von Kuchen verwendet wird. Dazu zerteilen wir sie zunächst in handliche Stücke und schmelzen sie in einem heißen Wasserbad. Danach tragen wir die flüssige Masse schichtweise mit einem Lebensmittel-Pinsel auf die Waffel-Stücke auf und lassen die Schokolade nach jeder Schicht abkühlen. Besonders schnell geht das in einem Gefrierfach.

Die Waffeln der Flügel verkleben wir auf Stoß und setzen die Laserkanonen oben auf die Flügelenen **2**, während der angeschrägte Flügerteil nach hinten zeigt. Nachdem die Rumpfschichten verklebt und erkaltet sind, bringen wir den Rumpf mit Messer und Raspel auf Endmaß und kleben danach noch die Triebwerke aus Waffelrollen zusammen.

Teile vorab lackieren

Beginnend mit den Flügeln überziehen wir das Modell nun auf einem Backpapier mit weißer Kuvertüre **3**. Dafür tauchen wir zuerst die Zahnstocher der Laserkanonen mit den Parabolantennen direkt in die Kuvertüre und lassen sie anschließend abtropfen, damit sich ein gleichmäßiger Überzug bildet. Nach dem Erkalten wiederholen wir diesen Prozess so oft, bis die gewünschte Schichtdicke erreicht ist. Danach ist der Rumpf dran. Eine heiße Messerklinge hilft dabei, den Überzug zu glätten und die Details zu modellieren. So können wir auch eventuelle Fehlstellen noch mit Kuvertüre verspachteln.



2 Jeder Flügel besteht aus zwei Waffeln. An ihren Enden befinden sich die Laser.

Den X-Wing zusammensetzen

Schließlich setzen wir den Rumpf mit den Flügeln und Triebwerken zusammen. Dazu fixieren wir die Flügel in angewinkelter Kampfstellung mit geeigneten Gegenständen, zum Beispiel zwei Gläsern **4**. Da diese Verbindung später stark belastet wird, tragen wir an der Klebnaht entsprechend viel Material auf und lassen es gut abkühlen.

Danach drehen wir das Modell auf den Rücken, stützen den Rumpf ab, damit das erste Flügelpaar nicht abbricht und bringen das zweite Flügelpaar an, das wir mit einem Apfel und Schaschlikspießen fixieren **5**.

Damit der X-Wing später über dem Kuchen schwebt, bohren wir danach drei Löcher in den Rumpf und stecken Schaschlikspieße hinein – für mehr Stabilität könnte man aber auch Esstäbchen nehmen. Zum Abschluss verzieren wir das Cockpit noch mit Lebensmittelfarbe und fertig ist der X-Wing.

Eine ausführlichere Fassung dieses Artikels mit mehr Bildern und Details findet ihr auch auf [heise+.](https://heise+.de/make-magazin.de/xa1r) Folgt dazu einfach dem Link. —akf

► make-magazin.de/xa1r



3 Vorlackierte Flügel auf Backpapier



4 Flügel und Rumpf sollte man mit reichlich Kuvertüre verkleben.



5 Das zweite Flügelpaar wird mit Äpfeln und Schaschlikspießen fixiert.

KI für die Modelleisenbahn, Teil 1

Ein Raspberry Pi überwacht mit einer Kamera und einem neuronalen Netz, ob Objekte am richtigen Platz sind. Prinzipiell kann man das für die Kontrolle von Parkplätzen oder den Inhalt von Lagerregalen benutzen. Hier demonstrieren wir eine Gleisüberwachung, die meldet, wenn Loks auf falschen Schienen fahren oder überfällig sind. Wie man dafür ein neuronales Netz modelliert und trainiert, zeigt unser Artikel.

von Detlef Heinze



Modelleisenbahner haben es nicht leicht. Viele Züge fahren gleichzeitig auf der Anlage. Die Züge rattern über die Gleise, und es kann viel schiefgehen. Ein Zug entgleist plötzlich, der andere fährt auf dem falschen Gleis. Wiederum ein anderer Zug wurde vergessen und steht zu lange am Bahnhof. Und das alles passiert auch auf schwer einsehbaren Gleisen.

Zwei oder vier Augen reichen manchmal nicht aus, auf größeren Anlagen das Geschehen sicher im Blick zu haben. Da wäre es doch schön, einen zusätzlichen maschinellen Helfer zu haben, der einen wichtigen Gleisabschnitt im Auge behält. Stimmt etwas nicht, informiert er uns automatisch.

Dieser Artikel stellt einen solchen maschinellen Helfer vor und beschreibt, wie man ihn für die eigene Anlage anpasst. Der maschinelle Helfer heißt „Smart Model Railway Cam“ und wird mit SMRC abgekürzt.

Unser YouTube-Video (siehe Link) zeigt die SMRC in Aktion. Der erste Artikel in diesem Heft beschreibt das Konzept des SMRC, wie man Bilder von Loks auf den Gleisen sammelt, kennzeichnet und für das Training eines neuronalen Netzes aufbereitet. Im zweiten Teil im kommenden Heft schauen wir uns an, wie man das Netz mit Google Colab trainiert, auf den Pi portiert und mit SMRC die Gleise überwacht.

SMRC

Die SMRC-Anwendung besteht aus einem einzigen Fenster (Bild 1). Es zeigt alle Informationen für die Überwachung an. Damit Zügelemente überwacht werden können, muss die SMRC sie erkennen, auf dem Kamerabild lo-

Kurzinfo

- » Überwachung von Gleisabschnitten verstehen und anwenden
- » Trainingsbilder erstellen, auswählen und Bildbereiche annotieren
- » Training des KI-Modells auf Google Colab

Checkliste

**Zeitaufwand:**
ein Wochenende

**Kosten:**
200 Euro

Mehr zum Thema

- » Josef Müller, Neuronale Netze zur Bilderkennung mit Python trainieren, Make 6/21, S. 48
- » Josef Müller, KI für den ESP32: Neuronales Netz trainieren für Strom-/Wasser-/Gaszähler-Daten, Make 1/22, S. 74
- » Video: Smart Model Railway Cam in Aktion

Material

- » Raspberry Pi 4 (ab 1GByte RAM)
- » PiCam V2
- » Schwenkbares Kameragehäuse (z.B. TEK-RPI CSW)
- » Raspberry-Pi-4-Gehäuse mit Lüfter (z.B. RPI CASE ALU2FAN)
- » MicroSD-Speicherkarte (mind. 16 GByte)
- » Netzteil für Pi
- » Coral USB Accelerator
- » Einkaufsliste siehe Link



**Alles zum Artikel im Web unter**
make-magazin.de/xa37



kalisieren, markieren und die Position im Bild ermitteln. Im Unterschied zu bisherigen Projekten in der Make, bei denen im Bild nur ein großes Objekt zu erkennen ist, werden hier

mehrere trainierte Objekte im Bild erkannt und lokalisiert, dazu gleich mehr. Auf der linken Seite kennzeichnet die SMRC eine Dampflok und eine Diesellok je-





Bild 2: Mit Googles Coral USB Accelerator lassen sich Objekte schneller erkennen.

weils durch ein Rechteck. Im Rechteck steht der Klassenname („:Dampflok“ oder „:Diesellok“). Die Prozentzahl gibt an, mit welcher Wahrscheinlichkeit die Erkennung stattgefunden hat.

Die SMRC übernimmt die Zuordnung von erkannten Objekten zu den Bildbereichen, hier den Gleisen. Damit können wir z.B. feststellen, welche Lokomotive sich gerade auf welchem Gleis befindet. Auf der rechten Seite in Bild 1 zeigt die Tabelle „Belegung“ an, auf welchem Gleis die Dampflok und die Diesellok halten.

Die Bereiche (Gleise) macht die Checkbox „Bereiche anzeigen“ sichtbar. Bereiche sind konfigurierbare Rechtecke. Damit kann man die SMRC an unterschiedliche Streckenführun-

gen anpassen. Die Gleise sollten aber gerade verlaufen. Zur Konfiguration der Bereiche später mehr.

Darüber hinaus stellt die SMRC den Zustand von Lokomotiven fest. Wir unterscheiden je Lok und je Gleis (Bereich) die drei Zustände: „hält“, „fährt“ und „abwesend“. Sie werden ebenfalls in der Tabelle „Belegung“ angezeigt.

Pro Gleis (Bereich) darf nur jeweils eine Lokomotive einer Klasse im Bild sichtbar sein. Nur so kann die SMRC den Zustand eindeutig ermitteln. Zwei gleiche Loks dürfen somit im Bild nicht auf dem gleichen Gleis (Bereich) zur gleichen Zeit fahren oder halten. Mehrere unterschiedliche Loks aber schon. Jede Klasse

steht für ein Objekt, das erkannt werden soll, z.B. eine spezielle Lok oder ein spezieller Waggon. Eine andere Klasse könnte Waggonen oder andere Fahrzeuge enthalten. Für Klassen mit einem Namen respektive ID ohne führenden „:“, z.B. Waggonen, ermittelt die SMRC keine Zustände. Sie zeigt aber ihre Gleisbelegung an.

Mit den beschriebenen Funktionen lassen sich Lokomotiven automatisch mit Hilfe von konfigurierbaren Regeln überwachen. Eine Regel bezieht sich auf die Lokomotive, ihre Gleiszuordnung und ihren Zustand. Beispiele sind:

- Die :Dampflok darf nicht auf Gleis 3 halten
- Die :Dampflok darf auf Gleis 1 maximal 15s abwesend sein
- Die :Diesellok darf auf Gleis 3 maximal 25s halten

Mit der zweiten Regel können wir feststellen, ob die Dampflok auf ihrem Oval, das sie normalerweise z.B. in 12s fährt, entgleist ist. In Bild 1 zeigt die SMRC rechts in der Tabelle Regelverletzungen an, welche der konfigurierten Regeln aktuell verletzt sind. Auf Gleis 3 hält die „:Diesellok“ länger als 25s.

Installation, Betrieb und Konfiguration der SMRC und das Erstellen von Regeln beschreiben wir detaillierter im zweiten Artikel dieser Strecke. In diesem Teil konzentrieren wir uns zunächst auf das Sammeln von Bildern und dem Trainieren des KI-Modells.

KI-Modell

Wir modellieren und trainieren ein neuronales Netz zum Erkennen einer kleinen Dampflok und einer roten Diesellok. Hat die eigene Modellbahnanlage andere Loks, so muss das Modell neu trainiert werden. Dieser Vorgang besteht aus einer Reihe von Schritten. Die Artikelserie „KI für den ESP32“ in der Make hat einen solchen Vorgang für die Klassifikation von Bildern detailliert beschrieben (siehe Kurzinfo-Kasten).

Da wir in diesem Projekt Objekterkennung statt Ziffernerkennung einsetzen, ist der Vorgang ähnlich, im Detail aber verschieden. Die Hauptschritte sind:

1. Sammeln und Auswahl von Trainingsdaten
2. Objekt(e) in jedem Bild annotieren
3. Training vorbereiten
4. Training durchführen und bewerten
5. KI-Modell übernehmen und testen

Das Projekt kann auch wachsen: Starten Sie klein mit zwei bis drei Lokomotiven und führen Sie die fünf Schritte aus. Später können Sie den Vorgang mit mehr Objekten wiederholen.

Vorbereitungen

Unser Projekt läuft auf einem Raspberry Pi mit PiCam V2 und einem externen KI-Beschleuniger „Google Coral“ (Bild 2). Eine vorkonfigurierte Einkaufsliste bei unserem Partner Berrybase finden Sie unter den Links. Die Software-Unterstützung für letzteren ist zwar prinzipiell im Pi OS



Bild 3: Lego-Technics-Komponenten halten unseren Hardware-Aufbau zusammen.

vorhanden, allerdings benötigt unser Projekt die nicht ganz aktuelle, auf Buster basierende Version. Die auf dem aktuellen Bullseye basierende Version funktioniert leider nicht.

Eine detaillierte Beschreibung aller Installationsschritte für den Raspberry Pi befindet sich in einer Readme-Datei im GitHub-Repository zum Projekt im Ordner „10_SMRC-Application“. Den Link zum GitHub-Repository finden Sie wie üblich oben in der Kurzinfor. Sie können das gesamte Projekt auf den Pi herunterladen und entpacken. Neben der Beschreibung zur Installation von Software befinden sich im Repository die Skripte für SMRC, zum Anfertigen der Bilder mit der PiCam, zum Annotieren der Bilder (dazu gleich mehr) sowie eine Sammlung von Beispiel-Bildern, falls Sie gerade nicht die Zeit haben, selbst Bilder zu machen und zu annotieren.

Machen Sie sich zu Beginn mit der Ordnerstruktur des heruntergeladenen GitHub-Repositories vertraut. Ein Baumdiagramm der Verzeichnisstruktur sehen Sie in Bild 4. Im Ordner „10_SMRC_Application“ sind alle Skripte für SMRC enthalten, darauf kommen wir erst im nächsten Artikel zu sprechen. Der Ordner „20_Example_Training“ enthält sowohl unsortierte als auch handverlesene Beispielbilder von Loks, die Sie für erste eigene Eindrücke geeigneter und nicht geeigneter Bilder anschauen können. Der Ordner „30_Your_Training“ dient für eigene Aufnahmen und Trainings und hat die gleiche Struktur wie „20_Example_Training“, es fehlen nur die Bilder und Daten dazu.

Trainingsdaten

Loks und Waggons stehen und bewegen sich auf der Anlage. Bewegte Aufnahmen sind daher immer etwas verwischt. Das müssen wir bei den Aufnahmen berücksichtigen. Um das neuronale Netz optimal zu trainieren, benötigen wir eine ausgewogene Mischung von Bildern. Die Tabelle gibt an, wie viele Bilder pro Objekt, in welchem Zustand und mit welcher Beleuchtung zu guten Ergebnissen in unserem Projekt geführt haben.

Art der Trainingsdaten		
Objekt-verhalten	Tageslicht	Tages- mit Kunstlicht
Langsame Fahrt	30	30
Schnelle Fahrt	30	30
Fahrt mit Anhalten	30	30
Stehend	30	30

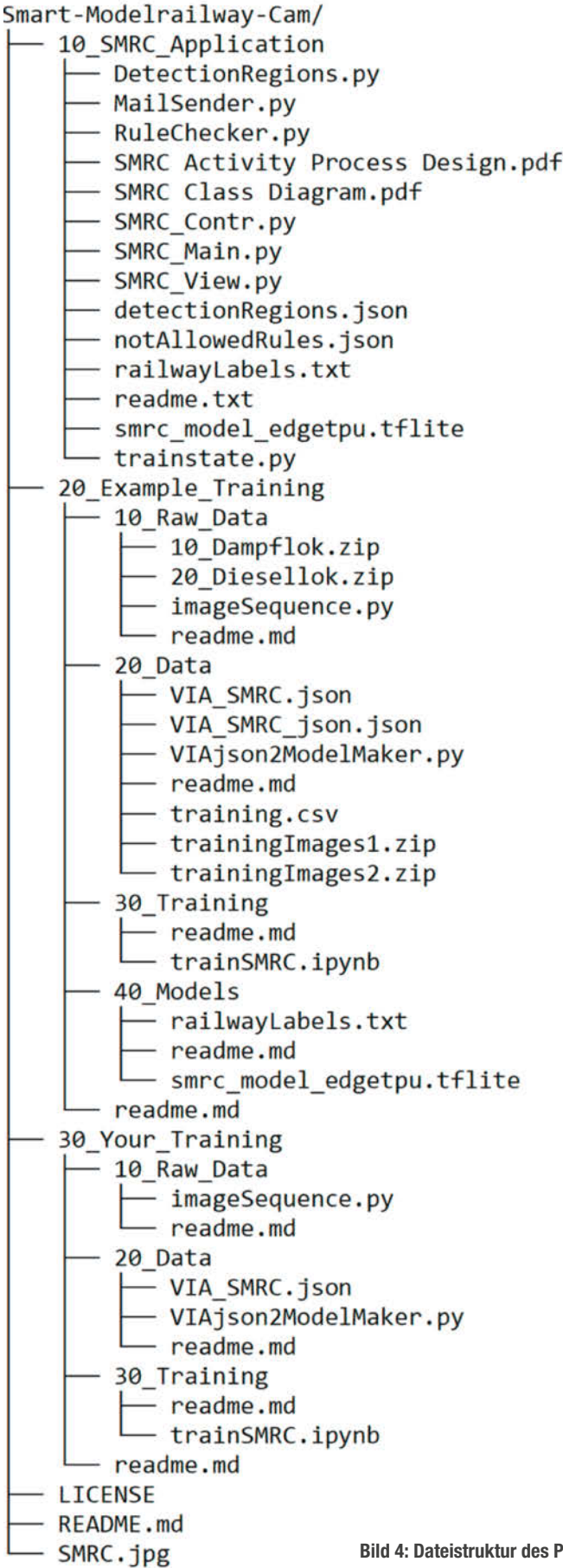


Bild 4: Dateistruktur des Projekt-Repositories

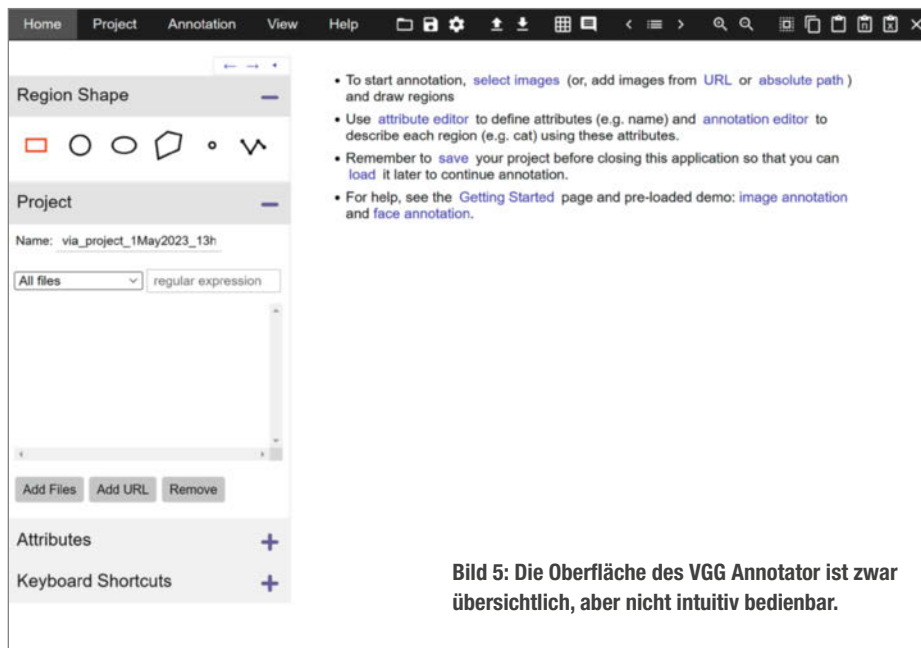


Bild 5: Die Oberfläche des VGG Annotator ist zwar übersichtlich, aber nicht intuitiv bedienbar.

Beispielbilder

Im Unterordner „20_Example_Training/10_Raw_Data“ liegen die Dateien „10_Dampflok.zip“ und „20_Diesellok.zip“, die jeweils 240 noch nicht nach gut und schlecht beurteilte Bilder einer Diesellok und einer Dampflok in unterschiedlichen Positionen zeigen. Entpacken Sie die Dateien in das aktuelle Verzeichnis „20_Example_Training/10_Raw_Data“. Auf dem Pi funktioniert das mit dem Befehl `unzip -j 10_Dampflok.zip` und `unzip -j 10_Diesellok.zip`. Zur Orientierung: Der vollständige Pfad etwa zu den Dampflok-Bildern lautet nun „20_Example_Training/10_Raw_Data/image07Mar22104200065126.jpg“ und so weiter.

Sammeln und Auswahl eigener Bildern

Wollen Sie lieber mit eigenen Bildern im Projekt loslegen, ist es wichtig, auf deren Qualität zu achten, damit die Objekterkennung zuverlässig funktioniert. Wir benötigen pro Objekt mindestens 240 Bilder in der Auflösung 320 x 320 Pixel. Um beste Ergebnisse zu erzielen, legen Sie vor den Aufnahmen den Gleisabschnitt und den Bildausschnitt (Kameraabstand) fest, den Sie später überwachen wollen.

Ein kleinerer Abstand führt zu besseren Ergebnissen, da die Loks und Waggonen dann nicht zu klein auf den Aufnahmen sind. Sorgen Sie bei den Aufnahmen für identische Lichtbedingungen, wie sie normalerweise auf der

Anlage herrschen. Heller ist dabei besser. Mehrere Objekte, die das Modell erkennen soll, können gleichzeitig auf einem Bild sein, müssen aber vollständig sichtbar sein. So kann man die Anzahl der Aufnahmen reduzieren. Wichtig ist auch, dass von jedem Objekt gleich viele Aufnahmen vorhanden sind. Dann sind die Trainingsdaten ausbalanciert, weitere Tipps dazu finden Sie im Kasten „Trainingsdaten“.

Der Raspberry Pi kann mit seiner Kamera die Bilder mit unserem Python-Skript „image-Sequence.py“ aufnehmen. Es liegt im heruntergeladenen GitHub-Repository im Ordner „30_Your_Training/10_Raw_Data“. Starten Sie das Programm an diesem Ort. Ohne Parameter nimmt es 14 Bilder innerhalb von 2 Sekunden in 320 x 320 Pixel auf. Das passt gut für eine Zugdurchfahrt, die Sie dann mehrfach wiederholen.

Anschließend wählen Sie geeignete Bilder gemäß der Mengen in der Tabelle aus, auf denen die Objekte vollständig zu sehen sind und kopieren sie in den Ordner „30_Your_Training/20_Data“. Das sind die Trainingsdaten. Dieser erste Schritt dauert je nach Übung ca. 4-5 Stunden für 2 Objekte mit je 240 Bildern. Alternativ liegen in „20_Example_Training/20_Data“ für Sie fertig vorsortierte Beispielbilder von Loks als Zip-Archive „trainingImagesxy.zip“. Sie können diese ebenfalls mit `unzip -j` entpacken.

Wir haben nun 240 Bilder von Dieselloks und 240 von Dampflok, entweder im Ordner „30_Your_Training/20_Data“ (selbst gemachte und sortierte Bilder) oder im Ordner „20_Example_Training/20_Data“ (unsere mitgelieferten Bilder). Nun beginnt die nächste Fleißarbeit.

Daten klassifizieren

Einer der wichtigsten Schritte beim Trainieren von neuronalen Netzen ist die richtige Beschreibung der Daten. In unserem Beispiel müssen wir zu jedem Bild eine Beschreibung mitliefern, was darin zu sehen ist. Diesen Vorgang nennt man Annotieren. Die konkrete Beschreibung, welches Objekt im Bild zu sehen ist, nennt man Labeling. In der Praxis werden Annotation und Labeling oft synonym benutzt.

Im Unterschied zur klassischen Objekterkennung, bei dem nur ein Objekt im Bild ist (Bild zeigt Katze oder Hund) enthalten unsere Bilder eine komplette Szenerie, in der mehrere Objekte zu sehen sind. Die große Herausforderung ist, in unseren Bildern mehrere Objekte mit einem Label zu versehen. Wie sagt man dem neuronalen Netz, dass das Objekt unten links eine Diesellok ist und das rechts oben eine Elektro-Lok?

Glücklicherweise gibt es freie Tools, die einem dabei helfen. Das kostenlose Online-Tool „VGG Image Annotator (VIA)“ hilft uns

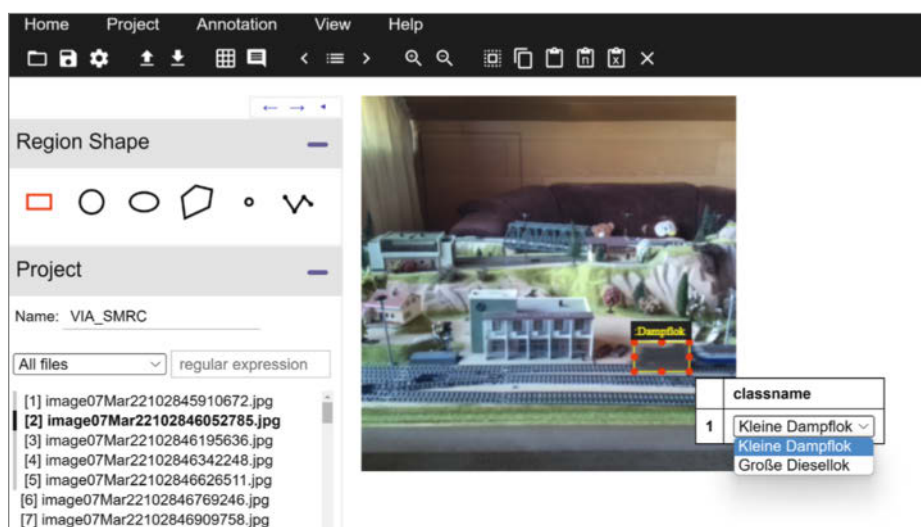


Bild 6: In den Bildern sind die Objekte mit Rechtecken markiert und der classname wird angezeigt. Links ist die Liste aller geladenen Bilder.

dabei. Mit VIA zeichnet man mit der Maus einen Kasten um ein Objekt und ordnet ihm eine zuvor definierte ID aus einer Liste zu, in unserem Beispiel die Art der Lok.

VIA läuft auf unserem Raspberry Pi mit Pi OS Buster leider nicht fehlerfrei, weshalb wir die nächsten Arbeitsschritte auf einen PC verlagern. Sofern Sie eigene Bilder für die Annotierung verwenden, transferieren Sie diese am besten per USB-Stick. Wer einen Fileserver zu Hause hat, nutzt natürlich diesen Weg. In unserer Anleitung auf Github beschreiben wir auch einen Weg per VNC.

Wollen Sie sich die von uns annotierten Bilder anschauen, benötigen Sie auf jeden Fall die Dateien „VIA_SMRC.json“ und das Python-Skript „VIAJson2ModelMaker.py“ aus dem Ordner „20_Example_Training/20_Data“.

Der Download-Link für VIA ist unter dem Link in der Kurzinfo zu finden. VIA kommt als ZIP-Archiv auf die Platte (etwa 400 kB), die Sie auf dem PC entpacken. Ein Doppelklick auf die Datei „via.html“ startet VIA im Browser. Achtung! In unserem Test funktionierte das Laden und Speichern von Projekten unter Windows nicht mit Firefox, sondern nur mit Chrome und Edge.

Ansätze

Wollen Sie nur mit unseren Bildern experimentieren, laden Sie die Datei „VIA_SMRC.json“. Achtung! Liegen die Trainingsbilder und die JSON-Datei nicht im gleichen Verzeichnis, gibt es eine Fehlermeldung. Tragen Sie in den „Project Settings“ den absoluten Pfad zu den Bildern ein und speichern die Einstellungen (runterscrollen und „Save“ klicken). Wichtig ist, dass der Pfad mit einem \ abschließt (Backslash), sonst findet VIA die Bilder nicht. Der Pfad müsste auf „...\\20_Example_Training\\20_Data“ enden.

Wenn Sie links (Bild 6) auf einen Dateinamen klicken, zeigt VGG das dazugehörige Bild an und welches Objekt es enthält. Die Objekte sind per ID gelabelt. Hier beginnen die IDs mit einem Doppelpunkt; „:Dampflok“ und „:Diesellok“. Sie gehören zum Attribut „classname“. Die vorbelegten Attribute und IDs sehen Sie unter der Liste der Dateinamen.

Wollen Sie eigene Lok-Bilder annotieren? Die zu annotierenden Bilder fügt man links über „Add Files“ hinzu. Am besten markiert man alle Bilder im sich öffnenden Dialog und bestätigt mit „OK“. Die Liste füllt sich nun mit den Namen aller Bilder. Anschließend klickt man unter der Liste auf das Pluszeichen neben der Überschrift „Attributes“ und gibt „classname“ als Attributname ein und klickt auf das kleine Pluszeichen daneben. Der Name darf nicht geändert werden, da auf ihn später in den Skripten Bezug genommen wird (Bild 7).

Innerhalb dieser Klasse fügen wir nun IDs über die Option „Add new option id“ hinzu: „:Dampflok“ und „:Diesellok“. Um die Arbeit zu erleichtern, öffnet sich nach dem Markieren der Objekte in den Bildern eine Option, um die ID zuzuordnen. Unter anderem stehen „Checkbox“ und „Dropdown“ zur Auswahl. Wir wählen im Beispiel „Dropdown“. Nun kann man in den eigenen Bildern die Objekte labeln, indem man mit gedrückter linker Maustaste einen Kasten zeichnet. Beim Loslassen der Maustaste öffnet sich das Zuordnungsmenü, in dem man die Klasse anklickt.

Sind Sie bei Bild 437 kurz vor dem Einschlafen, können Sie das Projekt inklusive aller Annotationen als JSON-Datei unter „Projekt/Save“ speichern und später wieder laden. Sind dann alle 480 Bilder bearbeitet, exportiert man unter dem Menüpunkt „Annotation“ (schwarze Leiste oben) und dem Punkt „An-

The screenshot shows a web interface titled 'Attributes'. It has two tabs: 'Region Attributes' and 'File Attributes'. Below the tabs is a form with 'attribute name' and 'classname' fields. The 'classname' field is set to 'classname'. Below this is a table with columns 'Name', 'Desc.', and 'Type'. The table has one row with 'classname' in the 'Name' column, 'Klassenname von :Lok oder Wag' in the 'Desc.' column, and 'dropdown' in the 'Type' column. Below the table is a section titled 'id' with a table containing two rows: ':Dampflok' with description 'Kleine Dampflok' and ':Diesellok' with description 'Große Diesellok'. The ':Diesellok' row has a selected radio button in the 'def.' column. At the bottom is a button 'Add new option id'.

Bild 7: Um bei Bildern mit mehreren Objektarten leichter die Label zuordnen zu können, definiert man sie vor. Markiert man dann ein Objekt grafisch, öffnet sich gleich daneben ein Auswahlménü mit den vordefinierten Labels.

notation - Export Annotation (as json)“ die Datei „VIA_SMRC_json.json“ in den Ordner der Trainingsbilder. Für unsere mitgelieferten Bilder existiert diese Datei bereits.

Spiele Sie in VIA ruhig mit eigenen Markierungen, IDs, Checkboxes und Dropdown-Menüs. Vielleicht entwickeln Sie bis zum zweiten Teil des Artikels und dem Training unseres neuronalen Netzes Ideen, was Sie noch alles auf diese Weise annotieren und überwachen könnten. —dab

ct SOLARSTROM-GUIDE
Kleine Photovoltaik-Anlagen planen und aufbauen

Das eigene Balkonkraftwerk
Wechselrichter, Module und Befestigungen organisieren
Technik, Anmeldungen und Regularien durchschauen

Ertrag und Verbrauch im Blick
Mit und ohne Smart Meter: Leistung beobachten,
erfassen und auswerten

Photovoltaik für alle
Was brauche ich, wer baut es, welche Produkte?
Reportage: 4,5 Kilowatt-Anlage im Eigenbau

So kann jeder Stromkosten senken
Balkon, Fassade, Dach und Gartenhaus: Kosten sparen
mit Sonnenenergie · Schritt für Schritt zur Mini-PV

Heft + PDF mit 26 % Rabatt

Jetzt gibt's eine aufs Dach!

- So kann jeder Stromkosten senken
- Das eigene Balkonkraftwerk
- Ertrag und Verbrauch im Blick
- Photovoltaik für alle
- Mikrowechselrichter kaufen und einsetzen
- Auch als Angebots-Paket Heft + PDF + Buch „Photovoltaik - Grundlagen, Planung, Betrieb“ erhältlich!

Heft für 19,90 € • PDF für 16,90 €
Bundle Heft + PDF 26,90 €

shop.heise.de/ct-solarstromguide23

Stylus-Timer

Echte Vinyl-Freunde legen großen Wert auf eine Plattenspieler-Nadel in einwandfreiem Zustand, die ihre Schätze pfleglich behandelt – deshalb soll sie auch nach einer gewissen Betriebszeit gewechselt werden. Unser Nadel-Timer misst die echten Betriebsstunden und mahnt rechtzeitig einen Austausch an.

von Carsten Meyer



Die Schallplatte erlebt ein echtes Revival: Anfang 2023 wurden weltweit wieder mehr Vinyl-Scheiben verkauft als Audio-CDs; deren Anteil an verkauften physikalischen Tonträgern sinkt unaufhaltsam, während die schwarzen Scheiben schon 2021 ein sattes Plus von 50 Prozent gegenüber dem Vorjahr verzeichneten. Auch wenn ein erheblicher Teil der Pressungen in den Händen solventer HiFi-Freunde mittleren Alters landet, hat mittlerweile die jüngere Generation das Kulturgut Schallplatte wiederentdeckt.

Mit der Liebe zur Schallplatte gehen auch einige Pflichten einher, was vor allem die behutsame Behandlung der Tonträger betrifft: Die nehmen einen allzu sorglosen Umgang nämlich deutlich eher übel als die vergleichsweise pflegeleichte Audio-CD. Wer auf eine knisterfreie und über viele Jahre gleichbleibend gute Wiedergabe Wert legt, behandelt seine Schätze mit der gebotenen Sauberkeit und Vorsicht.

Spanabhebendes Abspielen

Dazu gehört auch ein regelmäßiger Austausch der Nadel (neudeutsch: *Stylus*) im Tonabnehmer-System. Selbst ein Diamant verschleißt nach einigen hundert Stunden Spieldauer, und dann ist ein schonendes Abspielen nicht mehr sichergestellt – vom verminderten Hörgenuss einmal ganz abgesehen. Aufmerksame Vinyl-Freunde notieren sich deshalb die Spieldauer ihrer Nadel und tauschen sie nach Hersteller-Vorgaben aus.

Die Lebensdauer ist vor allem vom Schliff der Nadel abhängig: Bei elliptischen Nadeln kann man von 300 bis 500 Stunden ausgehen, Nadeln mit Shibata-Schliff halten mit 800 Stunden etwas länger. Deutlich robuster sind sphärisch geschliffene, sie erreichen bis zu 2000 Stunden – allerdings auf Kosten der Wiedergabetreue besonders von hohen Frequenzen. Im Zweifel hält man sich an die Empfehlung des Herstellers, die man getrost etwas aufrunden darf.

Wer nicht jede abgespielte LP-Seite mitzählen will, erhält im Handel sogenannte Stylus-Timer, die allerdings manuell zu starten und anzuhalten sind – keine besonders elegante Lösung. Unser Bauvorschlag dagegen startet automatisch mit Einsetzen des Audio-Signals und zählt minutengenau die wirkliche Laufzeit.

Play-Detektor

Möglich macht dies ein Audio-Detektor, der in die Cinch-Leitung vom Plattenspieler zum Verstärker eingeschleift wird und das Audio-Signal verlustfrei abgreift. Ein simpler Arduino reagiert darauf und zählt die Spielstunden, die er dann auf einer Siebensegment-Anzeige oder auch einem LC-Display anzeigt.

Kurzinfo

- » Betriebsstunden-Zähler für Plattenspieler-Nadeln
- » Echte Betriebszeit-Erkennung mit Audio-Detektor
- » Anzeige auf LED, LCD oder ePaper-Display

Checkliste



Zeitaufwand:
5 Stunden



Kosten:
20 Euro



Löten:
Bestücken einer (Lochraster-)Platine

Material

- » Arduino Uno
- » Multi Function Shield mit Siebensegment-Anzeige
- » Detektor-Platine nach Stückliste (siehe Link)
- » Metallgehäuse Hammond Manufacturing 1550WQ
- » 4 Cinch-Buchsen für Chassismontage
- » 2 Schrauben M3 x 16 mit Muttern
- » 2 Abstandshülsen Metall, M3/5mm

Mehr zum Thema

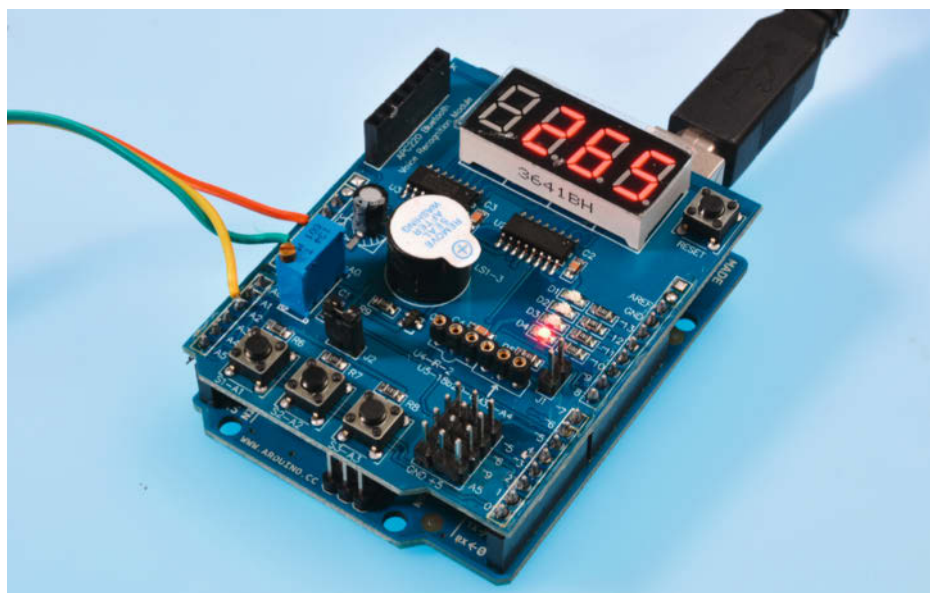
- » Make Special Operationsverstärker

Alles zum Artikel
im Web unter
make-magazin.de/x6d7

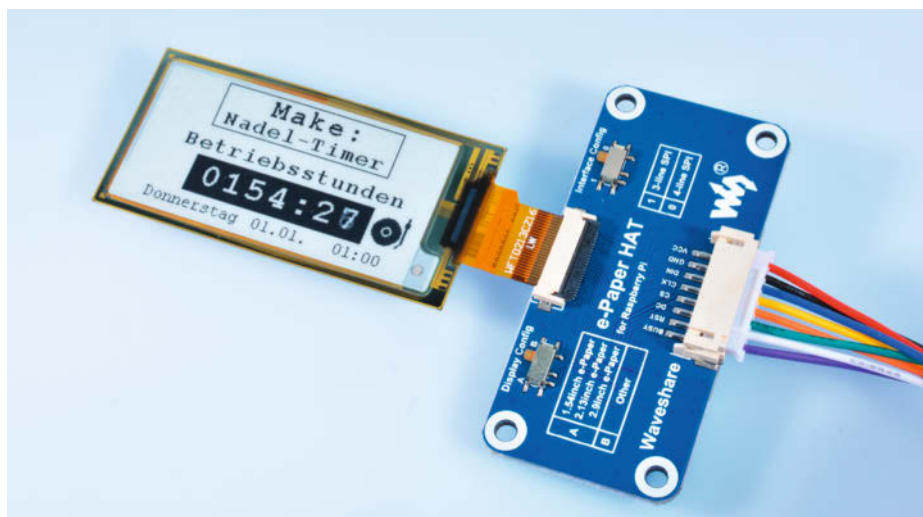


Der Audio-Detektor hat einige Vorgaben zu erfüllen: Zunächst muss er das mit nur wenigen Millivolt recht schwache Signal vom Magnetsystem kräftig verstärken und gleichrichten, damit der auswertende Arduino einen brauchbaren Logik-Pegel erhält. Ebenso wichtig ist ein verlustfreies Arbeiten: Der Detektor darf die Signalquelle weder kapazitiv noch resistiv nennenswert belasten, er muss also sehr hochohmig angekoppelt werden.

Zum Glück gibt es mit den CMOS-Operationsverstärkern Bauteile, die diese Forderung bestens erfüllen. Allerdings darf man nicht den kapazitiven „Belag“ der nötigen Anschlusskabel vernachlässigen; jeder Meter zusätzliches Cinch-Kabel würde die Höhenwiedergabe nachteilig beeinflussen. Unser Stylus-Timer ist deshalb zweigeteilt: Der Detektor ist separat aufgebaut und wird möglichst nahe an den Phono-Eingangsbuchsen des Verstärkers platziert, zum Anschluss reichen dann wenige



Für die Anzeige der Betriebsstunden eignet sich zum Beispiel das leicht erhältliche und preiswerte Multi Function Shield. Ein Druck auf den Taster S3 „spult“ die Betriebsstunden vor, mit S1 wird der Zähler auf 0 zurückgesetzt.



Auf einem ePaper-Display ließen sich neben der Betriebsstunden-Anzeige noch weitere Informationen unterbringen, etwa das Plattenspieler-Icon bei aktivem Audio-Signal oder das Datum des letzten Stylus-Tauschs.

Zentimeter Kabel. Die Verbindung zum eigentlichen Timer erfolgt über ein dreipoliges Kabel, das neben der Stromversorgung für den Detektor-Vorverstärker nur das Logik-Signal für den Arduino führt.

Wenn Sie einen separaten Phono-Vorverstärker verwenden (heutige Verstärker und Receiver sind nur noch selten mit einem Phono-Eingang ausgestattet), lässt sich der Detektor oft im gleichen Gehäuse unterbringen. Man koppelt ihn dann besser nicht am Eingang, sondern am Ausgang des Vorverstärkers an; damit ist eine auch noch so geringe

Beeinflussung des Eingangssignals völlig ausgeschlossen. Im Schaltbild sind dann allerdings die beiden Widerstände R2 und R5 an den nun sehr viel höheren Pegel anzupassen (siehe Schaltbild).

Die Widerstände R3 und R4 verhindern übrigens, dass bei abgeschalteter Versorgungsspannung und hohem Eingangspegel (also bei Anschluss an einen Phono-Vorverstärker-Ausgang) ein merklicher Strom in die Eingangs-Schutzdioden des OpAmps fließt; das würde Verzerrungen verursachen.

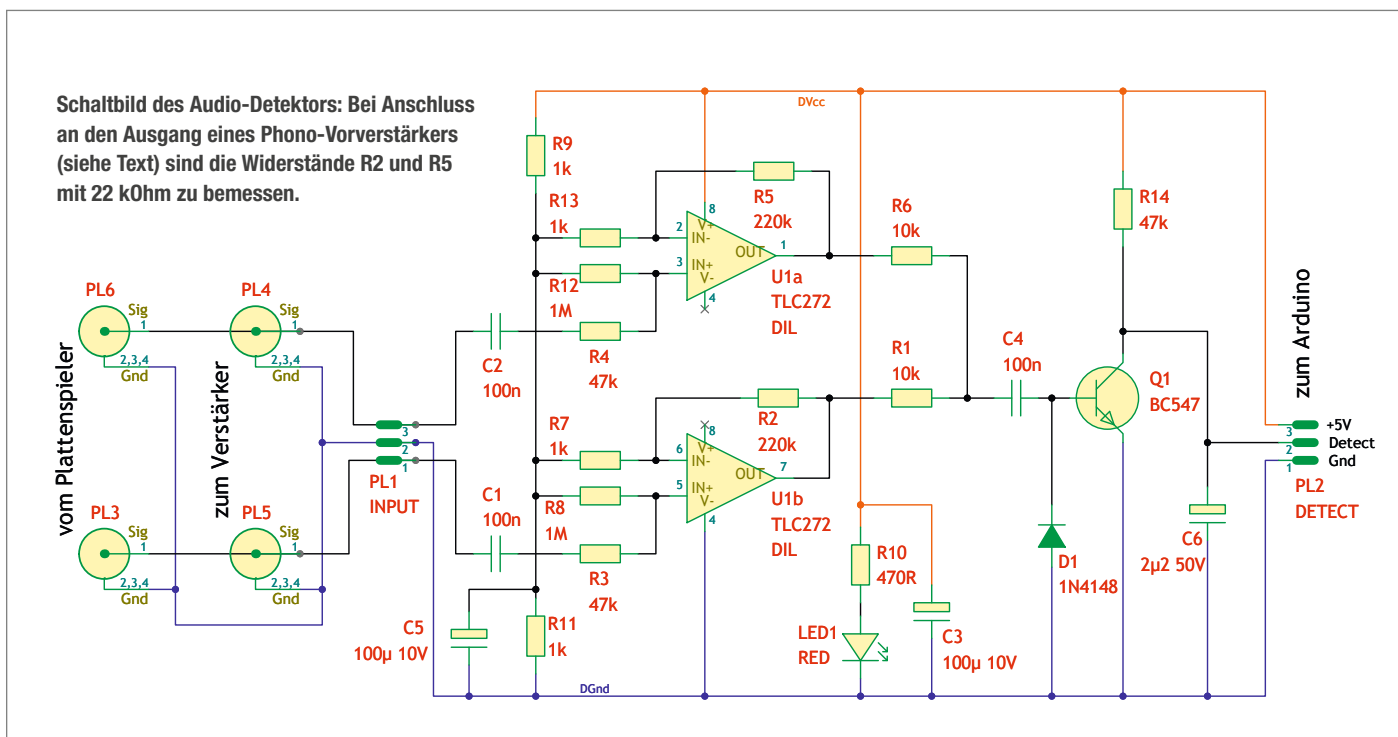
Für den Detektor haben wir eine kleine Platine entworfen, die Schaltung lässt sich aber ebenso gut auf einer Lochraster-Platine aufbauen. In jedem Fall sollte sie in ein kleines Metallgehäuse (z.B. von Teko oder Hammond Manufacturing) eingebaut werden, das mit der Schaltungsmasse verbunden ist. Ohne dessen abschirmende Wirkung würde das Audio-Signal von Netzbrumm und eingestreuten Störungen überlagert.

Letztere können übrigens auch von billigen Steckernetzteilen verursacht werden, die als Masse-Ausgleichsstrom über die Abschirmung in den Verstärker gelangen. Manchmal hilft es, das Steckernetzteil „anders herum“ in die Schuko-Dose zu stecken. In der Regel sind die älteren stabilisierten Netzteile mit Transformator störungsärmer als moderne Schaltnetzteile, sie verbrauchen aber auch mehr Energie im Leerlauf.

Preiswert anzeigen

Wir verwenden für unser Muster das preiswerte *Multi Function Shield* für den Arduino Uno, das neben einem vierstelligen Display auch gleich drei Taster eingebaut hat. Aus Gründen des Stromverbrauchs wäre (mit Anpassung der Firmware) auch ein unbeleuchtetes, einzeiliges LCD und eine Master/Slave-Steckdosenleiste möglich, die sich automatisch mit dem Receiver einschaltet.

Der Ausgang des Detektors führt zum Arduino-Pin A2, die Betriebsspannung erhält er direkt aus den Stiftleisten-Pins „5V“ und „Gnd“. Bei Verwendung des *Multi Function Shields* ist der A2-Eingang mit dem Taster S2-A2 belegt, was aber nicht weiter stört. Ein Druck auf diesen



Taster simuliert dann ein anliegendes Audio-Signal, was für Debug-Zwecke vielleicht ganz hilfreich ist.

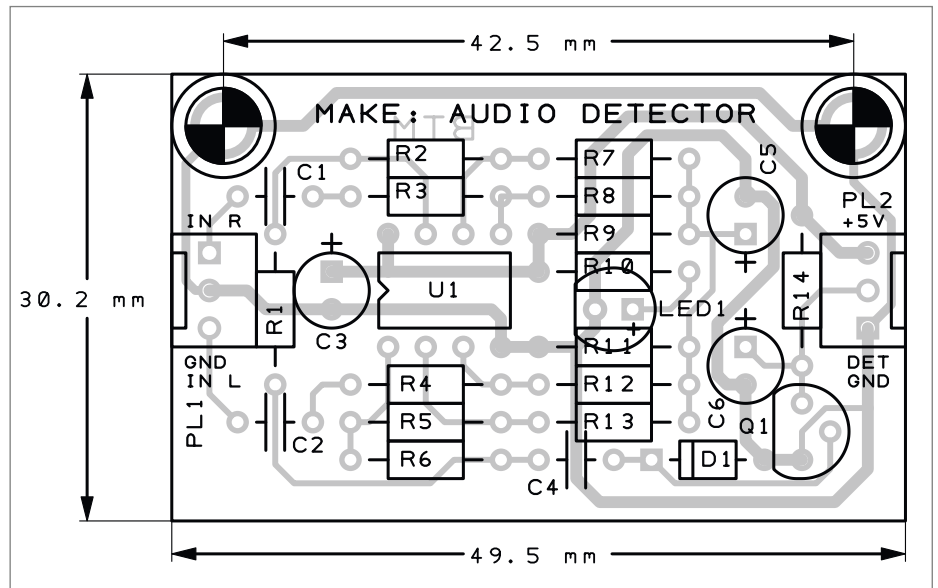
Timer-Firmware

Der zum Projekt gehörende Arduino-Sketch weist eine recht begrenzte Erfindungshöhe auf: Wenn er am Eingang A2 einen Low-Pegel feststellt, startet ein nachtriggerbarer Timer, der einige Sekunden lang den aktuellen Spieldauer-Zählerstand aktualisiert und die Kontroll-LED D1 einschaltet; stellt der Detektor weiterhin ein Audio-Signal fest, wird die Zeit wie bei einem Treppenlicht-Zeitschalter verlängert. Das verhindert, dass kurze Pausen (etwa zwischen den Titeln einer LP) im Audio-Signal den Zählvorgang stoppen.

Eine auftauchende Pause (entsprechend einem High-Pegel am Ausgang des Detektors) von wenigstens 30 Sekunden sorgt allerdings dafür, dass der (minutengenaue) Zählerstand im EEPROM des Arduino gesichert wird, sodass ein Stromausfall oder ein Abschalten des Geräts nicht zu einem Reset des Zählerstandes führt. Nur ein Druck auf den Taster S1 am Eingang A1 führt einen Reset des Spieldauer-Zählers auf 0 aus.

Mit einem Druck auf den Taster S3-A3 kann man die Betriebsstunden „vorspulen“, etwa wenn ein bereits gebrauchter Stylus zum Einsatz kommt oder man versehentlich den Reset-Taster gedrückt hat. Hier sorgt eine Autorepeat-Funktion dafür, dass sich auch einige hundert Stunden leicht einstellen lassen. Das Display zeigt den Zählerstand auf Stunden gerundet an, es wird regelmäßig aktualisiert, so lange ein Audio-Signal anliegt.

Eine Alternative zum Text-LCD oder der LED-Anzeige wäre ein ePaper-Display wie im



Die Fertigungsdaten für die kleine Detektor-Platine finden Sie unter dem Link im Info-Kasten.

Bild oben: Das behält auch nach Abschalten der Betriebsspannung seinen Display-Inhalt bei. Nachteil des Grafik-Displays: Das RAM des Arduino Uno ist mit nur 2 KByte zu knapp, das abgebildete kleine Display benötigt aber schon 2756 Bytes für den Frame-Buffer (212 × 104 Pixel, 8 Pixel pro Byte).

Wir haben den Arduino-Sketch deshalb auch für den deutlich leistungsfähigeren ESP8266 angepasst und ebenfalls online gestellt (siehe Link im Info-Kasten). Dabei kommt der Vorteil des ePaper-Displays zum Tragen: Auch wenn zum Beispiel die gesamte HiFi-Anlage über eine Schalter-Steckdosenleiste abgeschaltet wird, bleibt der Display-Inhalt erhalten. Ein Nachteil des ESP8266 sei hier aber nicht verschwiegen:

Da er kein EEPROM aufweist, muss die verstrichene Zeit im Flash-Speicher gesichert werden, der weit weniger Schreibvorgänge mitmacht als das AVR-EEPROM.

Begrenztes Verstärken

Die Schaltung besteht aus einem Doppel-OpAmp vom Typ TLC272, der mit einer niedrigen, unsymmetrischen Betriebsspannung auskommt. Er verstärkt das Eingangssignal von beiden Kanälen etwa 200-fach, die Wechselspannung am Ausgang richten eine Diode und ein Transistor gleich. Kondensator C6 am Ausgang glättet den entstehenden Gleichspannungspegel. Der Ausgang liegt in Ruhe

ICH WARTE NICHT AUF UPDATES. ICH PROGRAMMIERE SIE.



40%
Rabatt!



ct

c't MINIABO PLUS AUF EINEN BLICK:

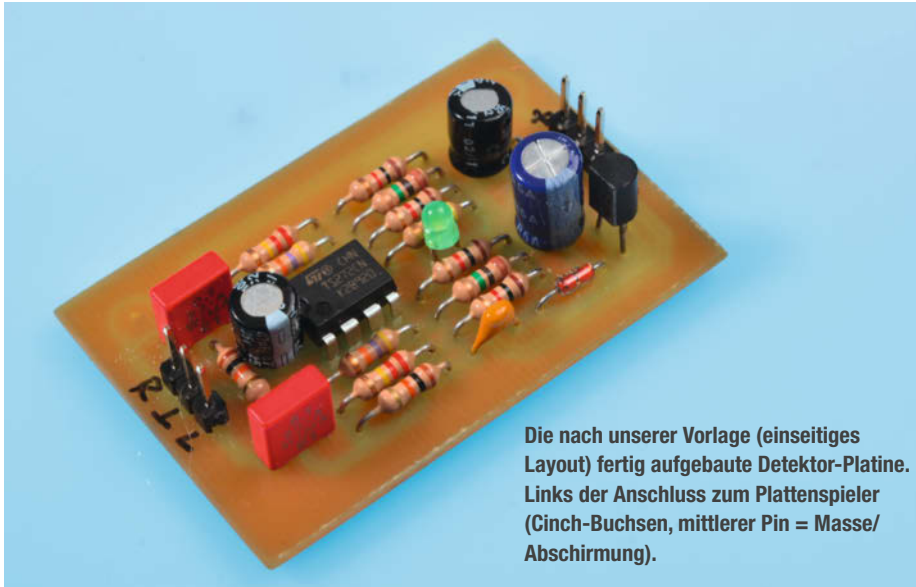
- 6 Ausgaben als Heft, digital in der App, im Browser und als PDF
- Inklusive Geschenk nach Wahl

- Zugriff auf das Artikel-Archiv
- Im Abo weniger zahlen und mehr lesen

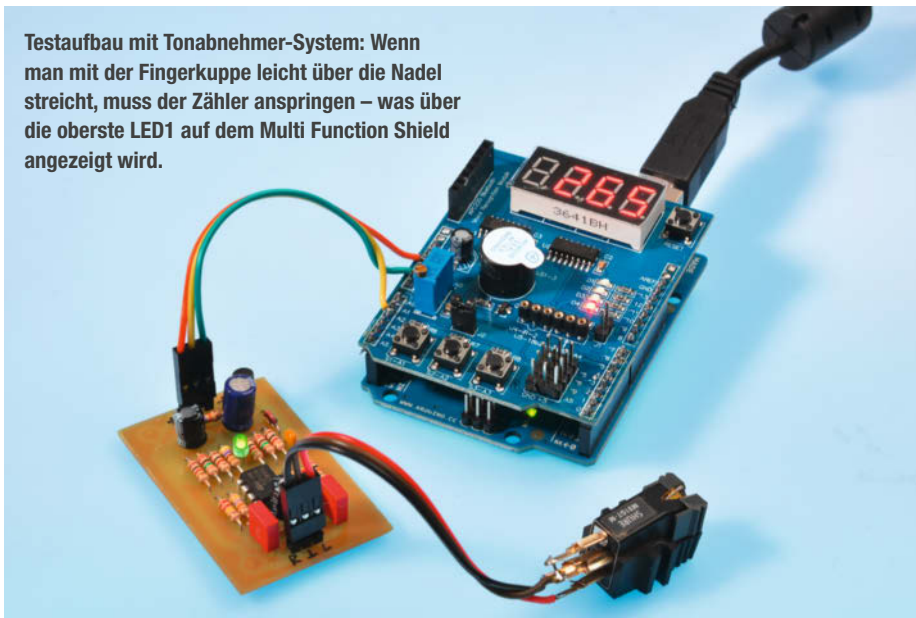
Jetzt bestellen:

ct.de/angebotplus





Die nach unserer Vorlage (einseitiges Layout) fertig aufgebaute Detektor-Platine. Links der Anschluss zum Plattenspieler (Cinch-Buchsen, mittlerer Pin = Masse/ Abschirmung).



Testaufbau mit Tonabnehmer-System: Wenn man mit der Fingerkuppe leicht über die Nadel streicht, muss der Zähler anspringen – was über die oberste LED1 auf dem Multi Function Shield angezeigt wird.



Der Detektor passt perfekt in das kleine Druckguss-Metallgehäuse 1550WQ von Hammond Manufacturing. Die Masse-Verbindung erfolgt über die zwei Befestigungsbohrungen.

auf *High*-Pegel und wechselt bei anliegendem Audio-Signal auf *Low*.

Durch die hohe Verstärkung gerät der Verstärker bei hohen Pegeln in die Begrenzung, was hier aber durchaus zulässig ist. Eine Schneidkennlinien-Entzerrer-Schaltung, wie sie sonst bei Phono-Vorverstärkern nötig ist, können wir uns ebenfalls sparen – es kommt ja nur grob auf das Vorhandensein eines Eingangspegels an.

Zusammen mit der 200-fachen Verstärkung und unter Berücksichtigung der Schaltspannung des Transistors (etwa 0,6V) ergibt sich eine Schaltschwelle von etwa 12mV_{ss} ($4,2\text{mV}_{eff}$) am Eingang. Für Moving-Coil-Systeme, die nur Ausgangsspannungen im Bereich weniger hundert Mikrovolt liefern, eignet sich die Schaltung also nicht.

Die beiden Eingänge des Detektors sind sehr hochohmig ausgelegt, sie werden das Signal des Tonabnehmers also nicht beeinflussen. Trotzdem sollte man, wie oben erwähnt, auf möglichst kurze Anschlussleitungen zwischen Detektor-Kästchen und Phono-Eingang des Verstärkers achten. Pingelige HiFi-Adepten werden hier kapazitätsarmes RG58-Koaxkabel verwenden. Die Kabellänge ist dagegen unkritisch, wenn der Detektor an den Ausgang einer Phono-Vorstufe angeschlossen wird.

Beachten Sie, dass der Detektor durch die enorme Verstärkung zum Schwingen neigt, wenn kein Plattenspieler-System angeschlossen ist. Für einen Test können Sie die Eingänge gegen Masse (mittlerer Pin) kurzschließen, ansonsten würde sofort die Zählung der Betriebsstunden beginnen. Die Kabellänge vom Detektor zum Arduino-Shield (3poliges Kabel) ist völlig unkritisch, sie darf durchaus einige Meter betragen.

Zweitverwertung

Der Audio-Detektor kann noch ganz andere Aufgaben erfüllen – mit einem angeschlossenen Mikrofon zum Beispiel als Geräusch-Detektor oder Klatsch-Schalter. Ebenso gut könnte er automatisch das Mitschneiden der LP auf Kassette (noch mehr retro!) oder MP3-Recorder starten oder ein Signal zum Einschalten einer Aktivbox liefern; der deutlich höhere Pegel verlangt dann aber eine Anpassung der Widerstände R2 und R5 auf 22kOhm oder weniger. Im Mono-Betrieb (Aktivbox) werden beide Eingänge miteinander verbunden.

Dazu ist noch nicht einmal ein Arduino nötig: Pausen ließen sich ebenso gut durch Vergrößern des Kondensators C6 und des Widerstands R14 überbrücken. Der dann recht hochohmige Ausgang benötigt aber zwingend eine weitere Treiberstufe, etwa einen simplen PNP-Transistor (Emitter als Ausgang, Basis an *Detect*, Kollektoranschluss an Masse).

—cm

Maker Faire®

Das Format für
Innovation & Macherkultur

Die nächsten Events



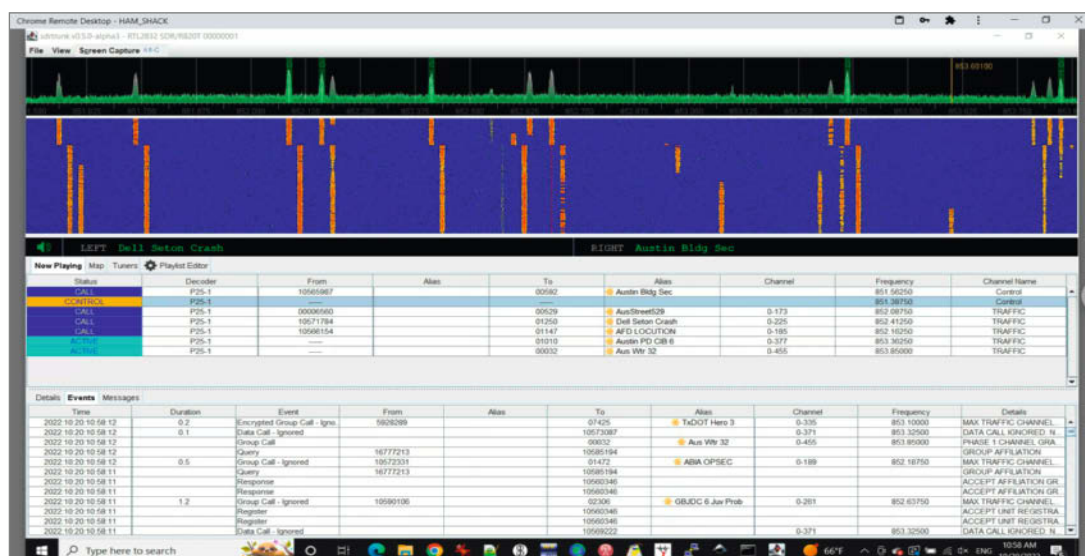
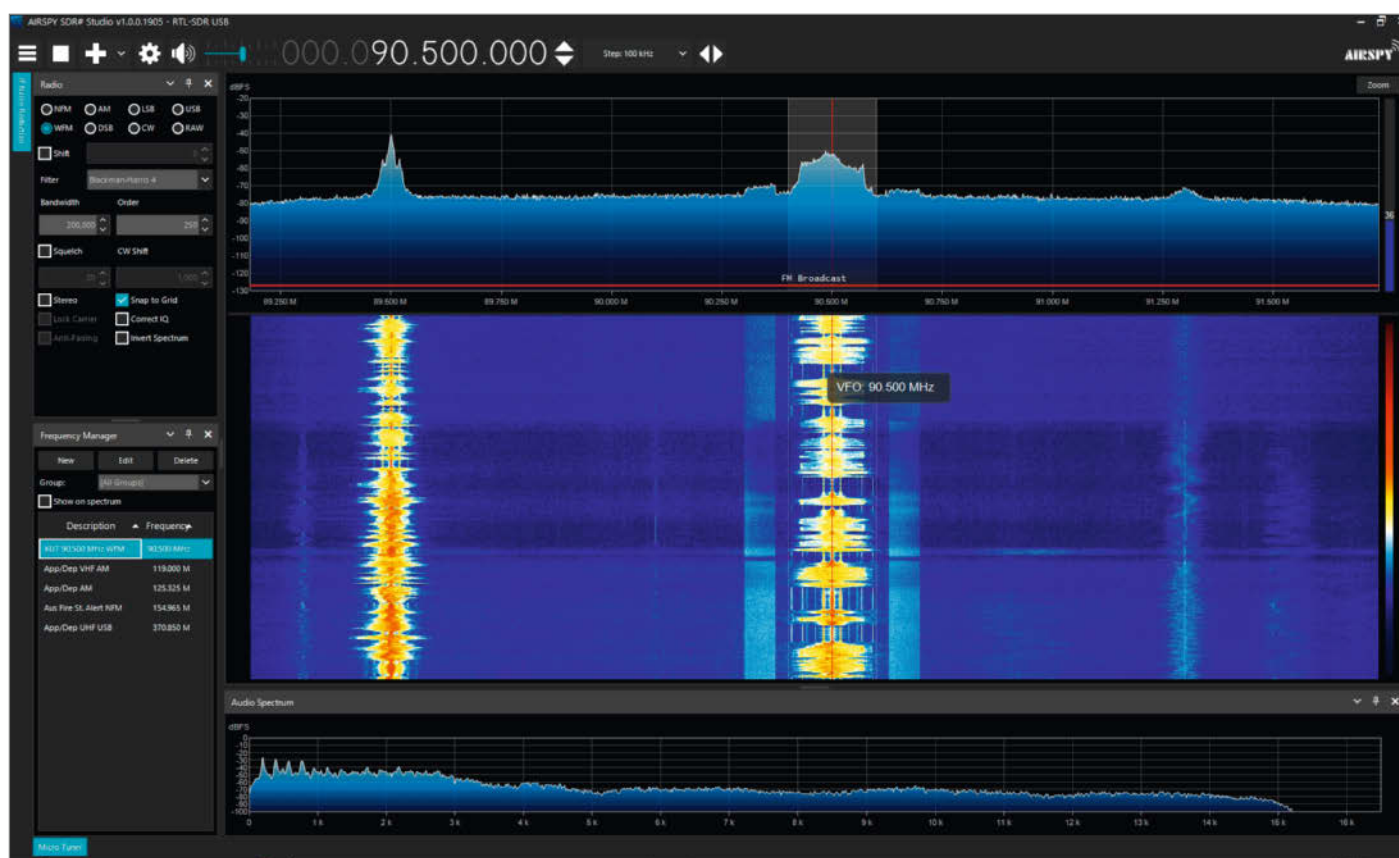
... weitere folgen.

maker-faire.de

Software Defined Radio (SDR) – ein Überblick

Die Zeiten von teuren Funkgeräten voller Elektronik und Schwingquarzen für jeden Bandbereich sind vorbei: Bei Software Defined Radio macht etwas günstige Hardware und ausgefeilte Software das gesamte Funkspektrum zugänglich – Mausklick genügt.

von Tim Deagan, KJ8U (Übersetzung: Katja Heimann-Kiefer)



Software statt teurer Funktechnik: Radioprogramme anhören mit SDR# (SDRSharp)

Die Software SDRTrunk, eingestellt auf den Funkverkehr von Rettungsdienst und Polizei in Travis County, Texas

Es ist noch gar nicht so lange her, da standen Fans von Funk und Radio beim Gerätekauf vor schwierigen Entscheidungen, wenn es darum ging, welches Gerät am besten zu ihren Interessen passte: Multiband-Geräte waren auf ausgewählte Frequenzen abgestimmt und benötigten in der Regel für jedes Frequenzband einen eigenen Quarz. Einfache Geräte konnten dann Mittel- Kurz- und Ultrakurzwelle und in den USA die Mitteilungen der Wetterbehörde NOAA empfangen. Und für alles, was darüber hinaus ging, musste man deutlich mehr Geld in die Hand nehmen.

Heute jedoch kann man mit einem kleinen USB-Dongle, der um die 30 Euro kostet, weit mehr empfangen, etwa Mittel-, Kurz- und Ultrakurzwelle, Amateurfunk („ham radio“), VHF, UHF, SHF, Seefunk, Flugfunk, Rettungsdienste, Satelliten, Wetterdaten, HD-Radio, Bündelfunk, P25, GPS, APRS, Radar, HAARP, CW, TV, GSM, Pager, Babyfone, DMR (digitaler Mobilfunk), D-STAR, Phasenumtastung/PSK, Funkfern schreiben, SSTV und außerdem fast alles andere, das mit Frequenzen zwischen 500 kHz und 1,75 GHz übertragen wird. Teurere Modelle können mit noch größeren Frequenzbereichen umgehen und teils sogar senden.

Dieser Artikel stellt Ihnen die zukunftsweisende Technik vor.

Von analog zu digital

Früher war das Senden und Empfangen von Funkwellen eine Sache der Analogtechnik. Für die Kernaufgaben eines Funkgeräts – Frequenzabstimmung, Erkennung, Schwingungserzeugung, Filterung, (De-)Modulation und Verstärkung – waren Quarze, Kondensatoren, Spulen, Röhren, Transistoren und weitere elektronische Komponenten vonnöten. Diese Komponenten kann man zu enorm komplizierten Schaltkreisen zusammenbauen, mit denen sich das Radiospektrum nutzen lässt. Der Analogfunk hat jedoch Grenzen, vor allem hinsichtlich des Frequenzbereichs, in dem die Geräte operieren können. Mit Ausnahme der alleraufwändigsten Modelle sind analoge Funkgeräte nur für einen äußerst begrenzten Frequenzbereich geeignet, oder sie besitzen mehrere Module, um verschiedene Frequenzbänder nutzen zu können.

In den 1970er Jahren begannen Forschungsgruppen des US-Verteidigungsministeriums, mit verschiedenen Verfahren zur softwaregestützten Signalverarbeitung zu experimentieren, die Hardwarekomponenten ersetzen sollten. Schnell erzielte man Fortschritte bei den mathematischen Grundlagen und der Codierung für die digitale Signalverarbeitung (Digital Signal Processing, DSP). Software eröffnete mit ihrer ungeheuren Leistungsfähigkeit und Flexibilität äußerst attraktive Perspektiven, um viele der hardwarebedingten Einschränkungen zu umgehen. In den

Kurzinfo

- » Das ganze Funkspektrum belauschen – mit Hardware ab 30 Euro und kostenloser Software
- » Radio hören, Wetterdaten empfangen, Flug- und Schiffsverkehr sowie die ISS verfolgen
- » Gratis in SDR reinschnuppern dank weltweit verteilter WebSDR-Server

Mehr zum Thema

- » Friederike Maier, Daniel Bachfeld: Radio in Software-Hand, c't Hacks 2/14, S. 132, kostenlos als PDF verfügbar (siehe Link)
- » Friederike Maier: HackRF One, c't Hacks 4/14, S. 80
- » Friederike Maier: Wettersatellitenbilder mit dem RTL-SDR empfangen, c't Hacks 4/14, S. 98
- » Friederike Maier: Wettersatelliten mit Windows jagen, Make 1/15, S. 86



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

1990er Jahren erwachte auch jenseits der regierungseigenen Labore das Interesse an Forschung und Entwicklung von „Software-Radio“, wie Software Defined Radio (SDR) damals noch hieß. Wirtschaftsunternehmen holten rasant auf, und 1997 tauchte die Technik in den ersten High-End-Autoradios auf.

Die Einführung von DVB-T, der europäischen Norm für digitales Fernsehen (Digital Video Broadcasting – Terrestrial), im Jahr 1997 gab den Startschuss für Unternehmen wie Realtek, in die Herstellung kostengünstiger Chips wie dem RTL2320U einzusteigen. Diese dekodierte die DVB-T-Signale im Bereich von 174 MHz bis 786 MHz und waren in Form kleiner USB-Sticks erhältlich. Anfang der 2010er Jahre entdeckten Eric Fry und Antti Palosaari, dass sich der RTL2320U auch als SDR-Empfänger nutzen ließ, und im Rahmen des Osmocom-Projekts veröffentlichte Steve Markgraf mit seinem Team die Software RTL-SDR (siehe auch c't Hacks 4/14, S. 98, Download der Software über den Link in der Kurzinfo), mit der man unter Linux, Windows und Mac-OS Funksignale in einem beeindruckenden Frequenzbereich von 500 kHz bis 2,2 GHz empfangen konnte, je nach verwendeter Chipvariante.

Wie funktioniert SDR?

Herzstück eines SDR-Empfängers sind zwei Analog-Digital-Wandler (ADC), die in einem Demodulator-Chip wie dem RTL2320U steckt. Sie wandeln das analoge elektrische Signal, das die Funkantenne auffängt, in Einsen und Nullen um, vergleichbar der Digitalisierung von Tönen, die man mit dem Mikrofon aufnimmt. Ein digitaler Tuner-Chip wie der R820T übernimmt in einem ausgewählten Frequenzbereich die Signale von der Antenne, verrin-



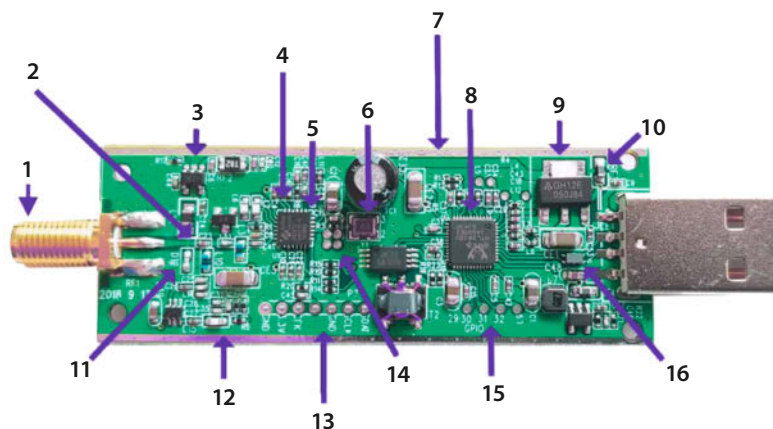
Einfache USB-Dongles
für SDR gibt es schon
für rund 30 Euro.

gert die Signale durch ein besonderes Verfahren auf eine niedrigere Frequenz und leitet sie an den Demodulator. Wenn das Signal dann in digitaler Form vorliegt, beginnt die Datenverarbeitung durch die DSP-Software, die filtert, dekodiert und die Daten in den gewünschten Output umwandelt.

So weit die Theorie – in der real existierenden Praxis ist man allerdings zu Kompromissen gezwungen. Beispielsweise können Analogsignale sehr schwach sein, weshalb sie oft verstärkt werden müssen. Verstärker wiederum werfen ganz eigene Probleme hinsichtlich Filterung, Rauschen, Dynamikumfang und anderen Aspekten auf. Die Bandpass-Filter, mit denen man oft versucht, diesen Problemen zu begegnen, schränken jedoch den Bereich der empfangenen Frequenzen ein. Viele SDR-Module haben mehrere analoge Eingangskanäle mit jeweils optimierter Verstärkung und Filterung, um über einen möglichst großen Frequenzbereich gute Ergebnisse erzielen zu können. Ähnliches gilt auch für die Antennen,

RTL-SDR BLOG V3

Was steckt alles in dem aktuellen Dongle RTL-SDR Blog V3?



1. SMA-Buchse
2. Verbessertes Frontend für höheren Signal-Rausch-Abstand im Mikrowellenbereich
3. Softwaregesteuertes 4,5-V-Einspeiseelement
4. Layout mit besserer Wärmeableitung für höhere VCO-Stabilität
5. Digitaltuner-Chip RAFAEL R820T2
6. Temperaturgesteuerter Quarz-Oszillator (TCXO) mit 1 ppm
7. Komplett neues, rauschärmeres Platinenlayout
8. REALTEK-Demodulatorchip RTL2832U
9. Besserer LDO, weniger Rauschen, geringere Betriebsspannung
10. Ferritperle auf der 5-V-Leitung
11. Zusätzlicher ESD-Schutz
12. Schaltung für Direktabtastung ermöglicht HF-Empfang (mit Tiefpass)
13. Anschlüsse für Erweiterungen
14. Jumper zur Auswahl des Zeitgebers
15. GPIO-Erweiterungsanschlüsse
16. USB-HF-Drossel entfernt USB-Rauschen

Ran an die Funkwellen!

Hier ist eine Liste mit vielen tollen Möglichkeiten, die Ihnen mit SDR offenstehen. Links dazu finden Sie unter RTL-SDR.com.

Achtung: Eventuell sind nicht alle Vorschläge in Ihrem Land legal, handeln Sie also verantwortungsbewusst.

- » Polizeifunk abhören
- » Funkverkehr von Rettungsdienst/Krankenwagen/Feuerwehr abhören
- » Funkverkehr der Flugverkehrskontrolle abhören
- » Flugzeugpositionen mittels ADS-B-Dekodierung verfolgen
- » ACARS-Nachrichten des Flugverkehrs dekodieren
- » Bündelfunk-Gespräche scannen
- » Unverschlüsselte digitale Sprachübertragung wie bei P25/DMR/D-STAR dekodieren
- » Schiffspositionen mittels AIS-Dekodierung verfolgen
- » Signale von POCSAG-/FLEX-Pagern dekodieren
- » Nach kabellosen Telefonen und Babyfonen suchen
- » Daten von öffentlichen Wetterballons empfangen
- » Die Position eines privaten Stratosphärenballons verfolgen, um die mitgeführten Geräte einzusammeln
- » Signale von Funksensoren für Temperatur und Leistung empfangen
- » UKW-Amateurfunk mithören
- » APRS-Pakete des Amateurfunks dekodieren
- » Analog-TV ansehen
- » GSM-Signale abhören
- » RTL-SDR auf einem Android-Gerät als portablen Funkscanner nutzen
- » GPS-Signale empfangen und dekodieren
- » RTL-SDR zur Analyse des Funkspektrums nutzen
- » Bilder von Wettersatelliten empfangen

- » Satellitensignale abhören
- » Radioastronomie
- » Meteorscatter beobachten
- » UKW-Radio hören und RDS-Daten dekodieren
- » DAB-Radio hören
- » HD-Radio hören und dekodieren (NRSC5)
- » RTL-SDR als Panadapter für herkömmliche Radiogeräte nutzen
- » Taxifunk-Signale dekodieren
- » RTL-SDR als hochwertige Entropiequelle für Zufallszahlengeneratoren
- » RTL-SDR als Rausch-Indikator
- » Reverse-Engineering unbekannter Protokolle
- » Signalquellen triangulieren
- » Nach HF-Rauschquellen suchen
- » HF-Filter charakterisieren und Antennen-SWV messen
- » EGC-Nachrichten des Inmarsat-C-Systems dekodieren
- » Signale der ISS auffangen

Und mit einem Upconverter oder dem Dongle RTL-SDR V3 zum Empfang von HF-Signalen können Sie außerdem:

- » Amateurfunk in SSB mit LSB-/USB-Modulation anhören
- » Digitalen Amateurfunk wie CW/PSK/Fernschreiben/SSTV dekodieren
- » HF-Wetterfaxe empfangen
- » DRM (Digital Radio Mondiale) über Kurzwelle empfangen
- » Kurzwellensender aus vielen Ländern hören
- » Nach Radarsignalen wie Überhorizontradar (OTH) und HAARP-Signalen suchen

die üblicherweise für einen Frequenzbereich optimiert sind.

Es ist die DSP-Software, der die beeindruckendsten Fortschritte bei der Signalübertragung zu verdanken sind, seit Bell Laboratories 1962 das erste digitale Übertragungssystem T1 in Betrieb nahm. Hat man erst einmal die physikalischen Einschränkungen analoger Schaltkreise hinter sich gelassen, kann man alle Signale mit Änderungen, die im Zeitverlauf gemessen werden – seien es etwa Audio- oder Videosignale, seismologische oder medizinische Signale – mit einem Maß an Präzision und Flexibilität bearbeiten, das für die analoge Datenverarbeitung unerreichbar bleiben wird.

Erste Schritte mit SDR

Für die ersten Experimente mit SDR sollten als Mindestausrüstung eine Antenne, ein SDR-Peripheriegerät und ein Computer vorhanden sein. Eine sehr beliebte Plattform für Einsteiger ist RTL-SDR als USB-Dongle, der in der Regel um die 30 Euro kostet. Auf dem Markt gibt es aber auch zahlreiche besser ausgestattete SDRs in ganz unterschiedlichen Preissegmenten, die auch senden können, beispielsweise einen umfassenderen Frequenzbereich anbieten, oder aufwändigere Schaltungen enthalten. Aber der schlichte RTL-SDR-Dongle eignet sich für den Einstieg ganz hervorragend.

Zwar sind viele SDR in der Lage, Signale am unteren Ende des Frequenzspektrums – je nach Modell unterschiedlich, aber meist unterhalb von 300 MHz – mittels Direktabtastung oder Direktkonvertierung zu dekodieren, dies aber oft auf Kosten der Klarheit. Viele greifen daher zu einem günstigen Upconverter: ein Gerät, das Signale mit Frequenzen unterhalb des SDR-Empfangsbereichs – vielfach sogar bis 0 Hz – in den Normalbereich des SDR-Empfängers konvertiert.

Besser ausgestattete SDR-Geräte bieten Nutzern, deren Ansprüche über gelegentliches Lauschen hinausgehen, zusätzliche Funktionen. In der Preisklasse um 200 Euro hat Airspy einen Receiver mit erweiterter DSP-Filterung (ein probates Verfahren, um Rauscheinflüsse zu reduzieren) im Angebot. Für gut 300 Euro gibt es von Great Scott Gadgets den HackRF (siehe auch c't Hacks 14/04, S. 80): Dieser wurde als Experimentier- und Prototyp-Plattform konzipiert. Er bespielt den Bereich zwischen 1 MHz und 6 GHz und kann auch senden.

Mit mehr als einem SDR-Gerät macht SDR so richtig Spaß. Ich selbst habe zwei RTL-SDR für einen P25-Bündelfunk-Dekoder im Einsatz, der den Funkverkehr von Rettungsdiensten und Polizei mithört, und nutze einen HackRF zum Experimentieren und als Panorama-Adapter (auch Panadapter genannt) für meine Kurzwellengeräte, um in Form einer sogenannten Wasserfallanzeige das Frequenzspek-



Einfache Teleskopantenne



Discone-Antenne

trum in der Umgebung der Empfangsfrequenz anzuzeigen.

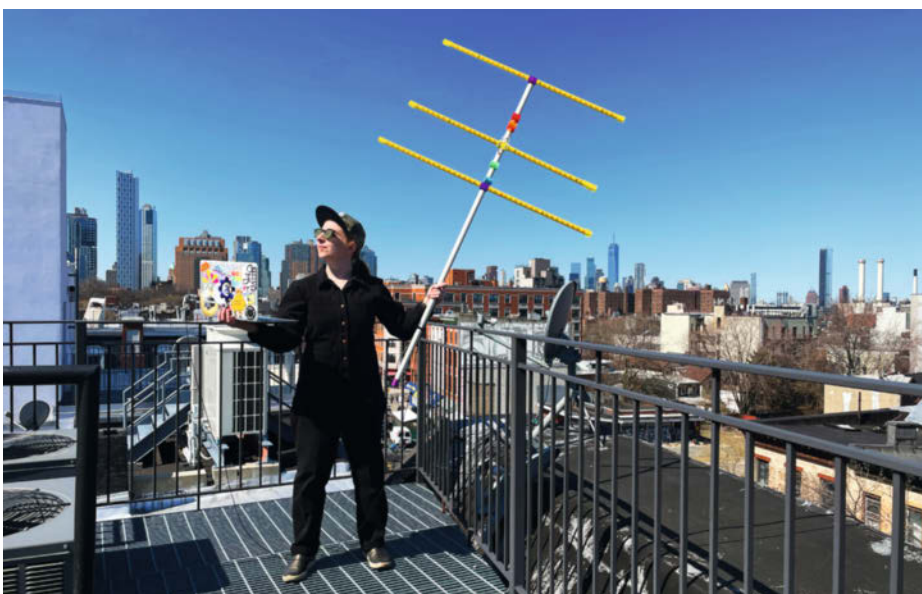
Antennenfragen

Wie so meist in der Funkerei hat die Empfangsqualität oft mehr mit der Antenne als mit dem Gerät zu tun. Zwar sind alle Antennen auf bestimmte Frequenzbereiche abgestimmt, aber Empfangsantennen sind da toleranter als Sendeantennen.

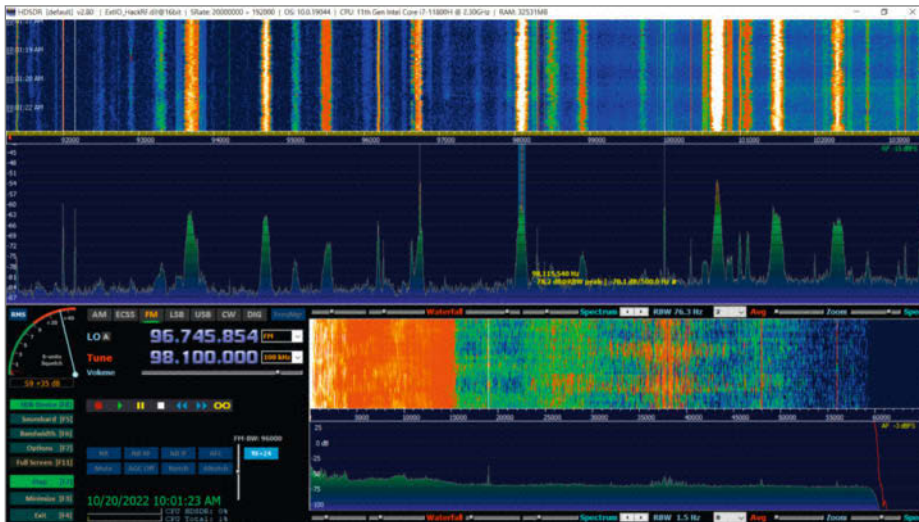


Blitzgefahr

Installieren Sie für jede Außenantenne eine Blitzschutzanlage mit einer guten Erdung. Diese Anlage leitet den Blitz und seine Energie ab, bevor sie ins Haus eindringt und Ihren Computer in eine Leuchtinstallation verwandelt.



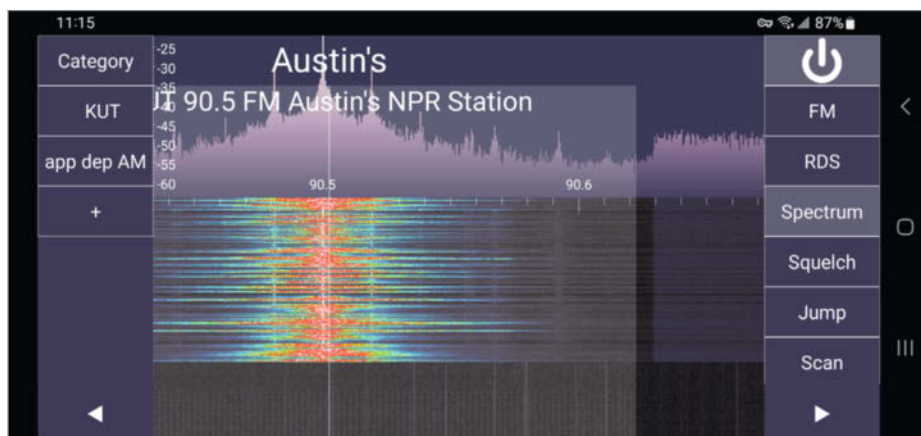
Eigenbau-Richtantenne: Die Anleitung für Becky Sterns Yagi-Antenne gibt es unter dem Link in der Kurzinfo.



Screenshot der Software HDSDR auf einem PC

Eine einfache Teleskopantenne, wie man sie von Kofferradios kennt, kann beim direkten Anschluss an den RTL-SDR eine überraschend große Bandbreite zwischen 60 MHz und 700 MHz abdecken. Bessere Modelle wie Discone-Antennen, die in erhöhter Position im Freien angebracht werden, können über

einen sehr großen Frequenzbereich zwischen 25 MHz und 1300 MHz eine verblüffende Empfangsqualität erzielen. Richtantennen (auch selbstgebaute, siehe Link in der Kurzinformatio) können einiges leisten, um Rauschen herauszufiltern und den Empfang zu verbessern. Will man das Potenzial einer Antenne voll



Software SDR Touch auf einem Android-Gerät



Karte von Online-SDR-Servern in aller Welt

ausreizen, kommt es auf das optimale Zusammenspiel von Höhe, Hindernissen, Speiseleitungen und weiteren Faktoren an – aber das ist ein unerschöpfliches Thema.

SDR-Software

Wie so oft bei moderner Technik ist die Hardware die Generalistin und die Software die Spezialistin. SDR-Software übernimmt das Breitbandsignal vom SDR-Gerät, zeigt es an, filtert es und dekodiert es in nutzbare Signale. Digitalfunk hat mittlerweile derart vielfältige Einsatzzwecke, dass ein Großteil all der interessanten Inhalte ohne Software gar nicht nutzbar wäre: Ob es nun darum geht, ADS-B-Positionssignale von Flugzeugen aufzulösen, Bündelfunksystemen mit wechselnden Frequenzen zu folgen, digitale UKW-Sendungen zu dekodieren, das Frequenzspektrum zu analysieren oder herauszufinden, welche Satelliten gerade das Haus überfliegen – eine SDR-Software vervielfacht die Einsatzmöglichkeiten der SDR-Hardware aufs Eindrucksvollste.

Die meisten SDR-Neulinge scrollen auf der Suche nach Hörbarem vermutlich in der Wasserfallanzeige herum – wer mehr als das möchte, findet Rat in Online-Tutorials für Spezialinteressen wie Flugbewegungen oder Satellitentelemetrie.

Heutzutage findet man kaum noch Laptops oder Desktop-Computer, Raspberry Pis, Tablets oder Smartphones, auf denen sich nicht irgendeine Art von SDR-Software ausführen lässt. Das umfangreichste Angebot gibt es für Windows, aber auch für Linux, Android, macOS und iOS finden sich leistungsfähige Tools. SDRangel und CubicSDR sind zwei der wenigen Apps, die plattformübergreifend (Windows, Mac, Linux) laufen (Download siehe Link in der Kurzinformatio).

SDR – los geht's

Sie haben sich also mit einem RTL-SDR-USB-Dongle und einer einfachen Teleskopantenne ausgestattet. Jetzt steht Ihnen die gesamte Welt der Funksignale offen – allerdings liegen viele dieser Signale in digitaler Form vor, viele sind Bündelfunk (wechseln also ständig die Frequenz), viele erfordern eine gute Außenantenne in ausreichender Höhe und viele Signale kommen nur sehr unregelmäßig, so dass man lange warten muss, bis wieder etwas zu hören ist. Was tun also, um überhaupt etwas zu hören und zu wissen, dass die neue Installation funktioniert?

Wie generell bei Peripheriegeräten sollten Sie auch hier als erstes den passenden Treiber für Gerät und Betriebssystem besorgen. Eine sehr gute – englische – Anleitung zur Einrichtung von Airspy's Anwendung SDR# (sprich: „SDR sharp“) unter Windows und Linux finden Sie über den Kurzinformatio-Link. Mac-Nutzer auf der

Suche nach Anleitung für CubicSDR werden über unseren Link bei Adafruit fündig.

Aber bevor Sie nun anfangen, Koaxkabel zu verlegen, einen Blitzschutz zu installieren oder Antennenmasten zu errichten: Viel einfacher ist es, zunächst mit der einfachen Tischantenne die überall präsenten UKW-Radiostationen zu empfangen. Sie senden mit Breitband-FM (WFM) zwischen 88 und 108 MHz, sind praktisch an jedem Ort zu empfangen und so stark, dass für den Empfang auch die einfachste Antennenkonfiguration reicht.

Stellen Sie den Tuner-Modus Ihres SDR auf WFM und scrollen Sie dann in der Spektrumanzeige im Frequenzbereich ab 88 MHz aufwärts umher. Auf der Wasserfallanzeige der Frequenzen erkennen Sie aktive UKW-Sender, die Ihre Antenne empfangen kann, an den hellen Streifen. Klicken Sie in diesen Streifen herum, um das Signal zu demodulieren, und dann können Sie es als Ton hören!

Natürlich kann man UKW-Sender auch auf viele andere (und einfachere) Arten empfangen, aber der Weg über SDR kann Ihnen ganz neue Hörwelten eröffnen. Wenn Sie später einmal mit Antennen arbeiten, die mehr können, gehen Sie genauso vor, um Kurzwellen, Amateurfunk und andere Frequenzbänder des Funkspektrums zu erkunden.

SDR online nutzen

Aber auch wenn das Kaufen und Konfigurieren von Hard- und Software nicht Ihr Ding ist, müssen Sie nicht auf SDR verzichten, denn Sie können es auch über ein Smartphone, Tablet oder einen Rechner nutzen, vorausgesetzt, das Gerät hat Internetzugang. Pieter-Tjerk de Boer (PA3FWM) hat eine Software für WebSDR-Server geschrieben, mit der viele Anbieter weltweit den webgestützten Zugriff auf ihre SDR und Antennen ermöglichen – kostenlos. Auf WebSDR.org finden Sie eine Liste von Websites mit SDR-Servern. Klicken Sie auf eine davon und stellen Sie den SDR-Empfänger auf das gewünschte Frequenzband ein. Sie sehen dann die Wasserfalldarstellung dieses Frequenzbands, können den Tuner-Modus wählen (AM für Mittelwelle, FM für UKW, SW für Kurzwelle usw.), Filter und weitere Einstellungen – genau wie an Ihrem Gerät bzw. SDR zu Hause.

Die meisten Server vertragen zwischen 30 und 80 Nutzer gleichzeitig und sind auf bestimmte Frequenzbänder beschränkt. Die begrenzte Bandbreite hat mit den angeschlossenen Antennen zu tun. Die meisten WebSDR-Server sind auf die HF-Bänder für Amateurfunk und Kurzwelle (30 MHz–300 MHz) ausgelegt, aber immer mehr weiten ihr Angebot auch auf UHF, VHF und andere Frequenzbereiche aus.

WebSDR-Server sind selbst dann nützlich, wenn man eigene Funk- oder SDR-Installationen hat: Mit Online-SDR an anderen Stand-

orten lässt sich der eigene Empfang vor Ort mit dem Empfang an anderen Orten oder mit anderen Antennenspezifikationen vergleichen. Und viele Amateurfunken nutzen WebSDR-Server in anderen Landesteilen auch, um ihre eigenen Aussendungen anzuhören und zu prüfen, wie sie „rüberkommen“.

Fazit

Software-Defined Radio (SDR) setzt sich rasant als die vorherrschende Funk- und

Radiotechnik des 21. Jahrhunderts durch. Software statt Schaltkreise zu verwenden, eröffnet eine solch ungeheure Flexibilität, dass davon auszugehen ist, dass SDR zukünftig noch in verschiedensten Anwendungen und Geräte integriert wird. Angesichts der niedrigen Einstiegskosten machen Funkinteressierte auf keinen Fall etwas verkehrt, wenn sie diese spannende Technologie einmal ausprobieren. Dank WebSDR-Servern weltweit braucht man beim ersten Schritt noch nicht einmal Hardware. —pek

The screenshot shows the WebSDR interface in a web browser. The top part displays a frequency spectrum with a waterfall view. Below this, there are controls for frequency (14261.370 kHz), mode (USB), and bandwidth (2.80 kHz). The interface includes a 'Memories' section, a 'Bandwidth' section with various filter options (CW-wide, LSB-wide, etc.), and a 'PassBand Tuning (PBT)' section. On the right side, there are settings for volume, noise reduction, audio buffering, and gain control. The bottom part of the interface shows a 'Logbook' section with a table of stations.

WebSDR-Server KFS, Half Moon Bay, Kalifornien

A WebSDR is a Software Defined Radio receiver connected to the internet, allowing many listeners to listen and tune it simultaneously. SDR technology makes it possible that all listeners tune independently, and thus listen to different signals. This is in contrast to the many classical receivers that are already available via the internet.

More background information is available [here](#). Questions and comments can be sent to [SDR@KFS](#), the author of the WebSDR software and maintainer of this site, but please check the [frequently asked questions](#) first.

WebSDR servers can register themselves automatically on this site, leading to the below list of currently active WebSDR servers.

Currently there are 111 servers active, with 2132 users and 707 MHz of radio spectrum.

Location and URL	Frequency range	Antenna
WebSDR at the University of Toronto, Toronto, NL http://webdr1.kfsdr.com P2XSDR, 111 users	0.000 - 29.999 MHz	Mini Whip
BATC & AMSAT UK QO-100 (E-Flat-2) Geostationary Satellite WebSDR http://webdr1.kfsdr.com QO-100, 98 users	10489.500 - 10490.500 MHz	1.5m Dish - GPS locked Orange LNB - Jumper Receiver
WebSDR - Paradise, SP - Brazil http://webdr1.kfsdr.com OG7W, 47 users	1.800 - 1.900 MHz 1.800 - 1.900 MHz 6.000 - 7.000 MHz 13.800 - 14.000 MHz 20.000 - 21.000 MHz 26.000 - 27.000 MHz 40.000 - 41.000 MHz 0.000 - 2.000 MHz	GR7V, 2x2m Beam 10m 1.5 vertical Beam 10m Beam 10m
Northern Utah WebSDR, Cannonville, UT, U.S.A. - WebSDR #1 (Yellow) - Landing Page http://webdr1.kfsdr.com DZ1J, 80 users	0.000 - 0.010 MHz 0.000 - 0.010 MHz 0.000 - 0.010 MHz 0.000 - 0.010 MHz 0.000 - 0.010 MHz 0.000 - 0.010 MHz 0.000 - 0.010 MHz	Omni-directional E-Flat whip at 6.1 Meters TC3 150 Ohm Log Periodic
WebSDR Maastricht Netherlands, near low noise level and full 100MHz/100MHz/100MHz band coverage, CW arguments included http://webdr1.kfsdr.com K01W, 47 users	1.800 - 1.900 MHz 2.700 - 2.800 MHz 6.000 - 7.000 MHz 13.800 - 14.000 MHz 20.000 - 21.000 MHz 26.000 - 27.000 MHz 40.000 - 41.000 MHz	Low noise active receiving loop and RSP1a Low noise active receiving loop and RSP1a Low noise active receiving loop and RSP1a Low noise active receiving loop and RSP1a Low noise active receiving loop and RSP1a Low noise active receiving loop and RSP1a Low noise active receiving loop and RSP1a
WebSDR QO-100 (E-Flat-2) Geostationary SAT 20MHz (10489.500-10490.500 MHz) Transponder http://webdr1.kfsdr.com P2XSDR, 47 users	10489.500 - 10490.500 MHz	1m Dish Antenna - PLL LNB & SDR Interface fully GPSDO Locked
KFS WebSDR on the Pacific coast south of San Francisco http://webdr1.kfsdr.com CME/ty, 40 users	3.100 - 4.000 MHz 6.000 - 7.000 MHz 9.000 - 10.000 MHz 13.800 - 14.000 MHz 18.000 - 19.000 MHz 20.000 - 21.000 MHz 26.000 - 27.000 MHz	'Omni' (TC3 150)

Screenshots einer Seite mit WebSDR-Servern

Pybricks: Python steuert Lego-Roboter

Die alternative Firmware bringt MicroPython auf Robotik-Gehirne von Lego. Im Gegensatz zur bisherigen Herstellerlösung entstehen so autonome Roboter, die nicht auf ein Tablet angewiesen sind. Die Programmierung erfolgt in einer modernen IDE im Browser über Bluetooth LE.

von Carsten Wartmann



Mit *Powered Up* brachte Lego eine leichter zugängliche und günstigere Alternative zum Robotersystem *EV3-Mindstorms* auf den Markt. Mit Legos *Boost* folgte ein Set, das für rund 130 Euro drei Servomotoren, einen Farb- und Distanzensensor sowie viele Teile pro Geldeinheit bot. Diese Serie gipfelte im Set *Roboter Erfinder (Inventor)*, welches den Namen *Mindstorms* bekam (wie auch schon die *NXT*-Serie) und die *EV3*-Serie ablösen sollte. Ende 2022 war aber schon wieder Schluss. In diesem Artikel werde ich zeigen, dass das System aufgrund der Hardware sehr interessant ist, vor allem wenn man mit der Open Source Firmware *Pybricks* plötzlich autonome Roboter programmieren kann. Die Verfügbarkeit der *Powered-Up*-Komponenten ist allerdings sehr schlecht geworden: Auch hier gebe ich Tipps, was man noch kaufen kann und was man besser auf dem Gebrauchtmärkte findet.

Viele Maker kennen oder besitzen Lego. Neben dem offensichtlichen Spaß beim Bauen bieten Robotik-Sets noch viel Lehrreiches. So schnell kann man sonst kaum Konzepte der Robotik ausprobieren: Vom Linienverfolger, Laufroboter oder Balancierroboter bis hin zu ganzen Fabrikmodellen ist vieles möglich und alle Teile können später wiederverwendet werden.

Bei *Boost* und *Powered Up* war bislang nur eine grafische Programmier-App für Tablets verfügbar, für *EV3* und *SPIKE* gab es auch MicroPython, aber das war hauptsächlich für Ausbildungszwecke gedacht. Der große Nachteil der App-Lösung ist, dass die Roboter nur funktionieren, wenn auch das Tablet und die App laufen.

Die Open Source (MIT-Lizenz) Firmware *Pybricks* vereint nun alle *Powered-Up*-Systeme unter einer MicroPython *API* (Application Programming Interface, Programmierschnittstelle). Sogar *EV3* wird unterstützt. Mit der Version 3.x ist der Funktionsumfang der Hardware (Motoren, Sensoren, Zubehör) fast zu 100 Prozent abgedeckt und das ganze System läuft sehr gut und ca. 100 × schneller als über die App. Für *EV3* benötigt man noch eine Visual-Studio-Code-*EV3*-MicroPython-Erweiterung, aber das *Powered-Up*-System wird komplett in einer browserbasierten IDE programmiert, debuggt und auf die Hubs hochgeladen. Im Gegensatz zu vielen anderen Browser-IDEs bleibt dabei der Komfort nicht auf der Strecke: Autovervollständigung, Parameterlisten, integrierte Dokumentation und weitere Komfortfunktionen sind hier enthalten. Die Community ist sehr hilfreich und der *Issues* Tracker auf Github sehr aktiv.

Der Code-Editor läuft in Chrome oder Edge auf Systemen, die Bluetooth LE (Low Energy) unterstützen, oder mit einem externen USB-zu-BLE-Dongle. Dies deckt *Windows*, *macOS*, *Chrome OS* und *Android* ab. Unter *Linux* muss

Kurzinfo

- » **Lego Powered Up ist eine günstige Roboter-Serie von Lego**
- » **Pybricks bringt Python auf Steuer-Hubs ohne Tablet und Cloud Zwang**
- » **Programme und Daten per Bluetooth LE für autonome Roboter**

Checkliste



Zeitaufwand:
ab 1 Stunde



Kosten:
ab etwa 100 Euro

Material

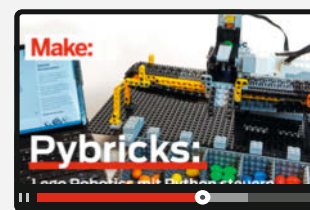
- » **Lego Powered Up Hub** Technic Hub oder Move Hub
- » **Sensoren, Motore** aus dem *Powerd-Up*-Programm
- » **Lego oder andere Klemmbausteine** Technic- und normale Bausteine

Werkzeug

- » **PC oder Tablet** Windows, macOS, Android, Chrome OS
- » **Browser** Chrome oder Edge

Mehr zum Thema

- » Achim Bertram, *Tipps&Tricks*, Lego als Endmaße im Maschinenbau, *Make* 2/22, S. 94
- » Detlef Heinze, *Pfadplanung für Roboter*, *Make* 1/21, S. 78
- » Alexander Ehle, *LEGO-Grusellabor mit Licht und Sound*, *Make* 2/20, S. 54
- » Video zum Artikel: Installation, erste Schritte mit *Pybricks*, *Sortierer in Aktion*

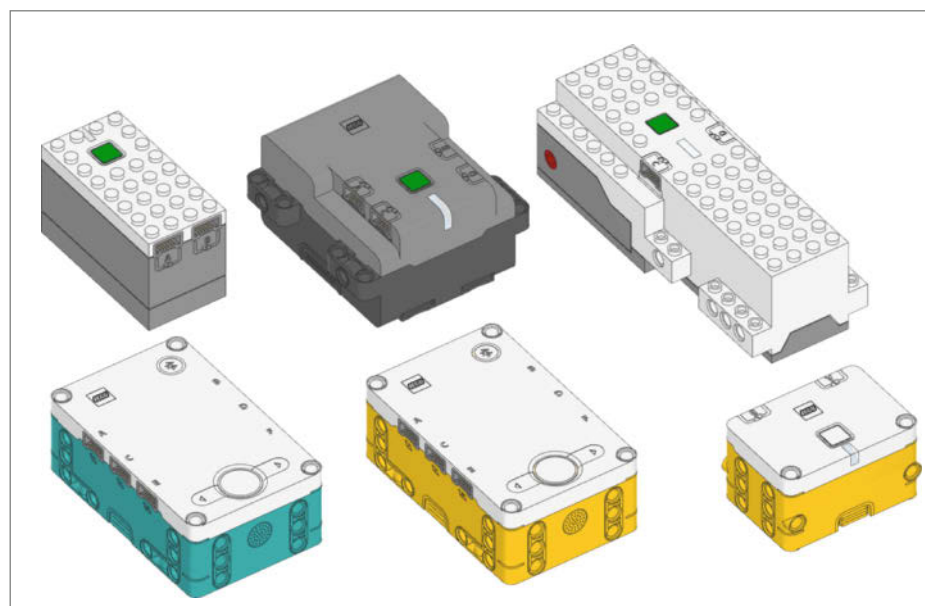


Alles zum Artikel
im Web unter
make-magazin.de/xdrd

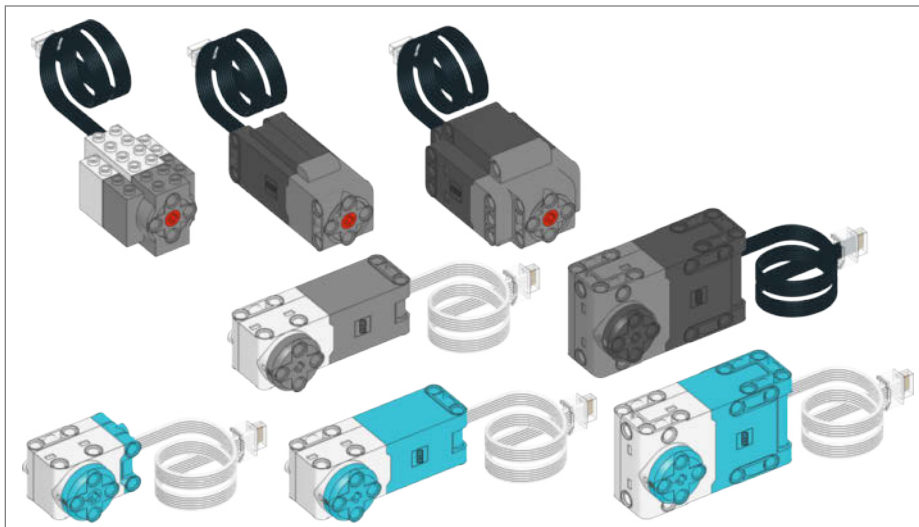


je nach Distribution eine Entwicklerversion von Chrome mit Bluetooth-Unterstützung installiert werden. *iOS* wird aufgrund der fehlenden BT-Unterstützung in Chrome derzeit nicht unterstützt. Sobald der Code-Editor

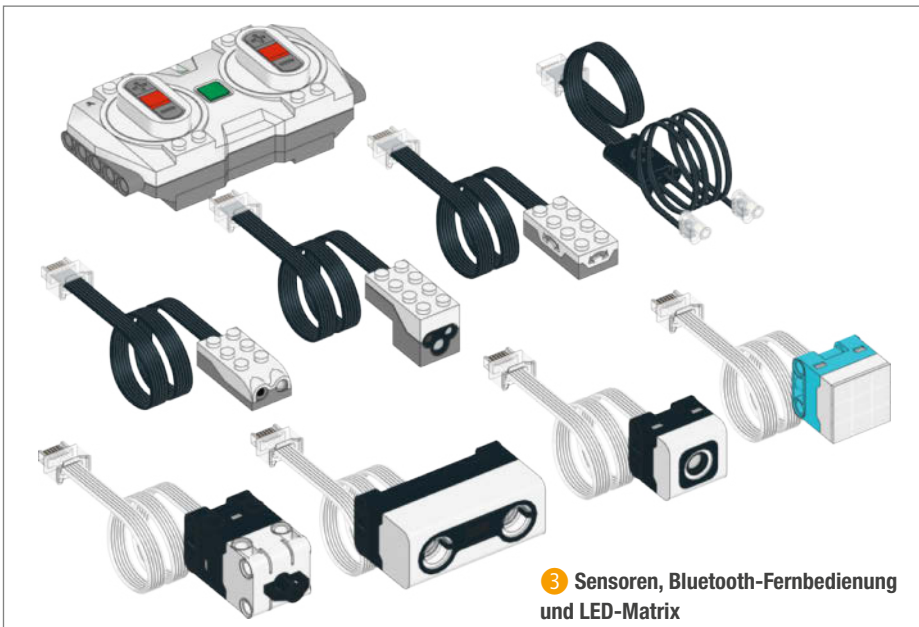
vollständig geladen ist, wird keine Internet-Verbindung mehr benötigt. Für *EV3* wird ein PC mit VS Code und SD-Kartenleser benötigt, für Firmware-Updates der großen *Powered Up Bricks* noch ein USB-Port.



1 Die Hubs der *Powered-Up*-Serie



2 Servomotoren für Powered Up



3 Sensoren, Bluetooth-Fernbedienung und LED-Matrix

Die Hubs

Die Hardware der Hubs lässt sich wie folgt kategorisieren:

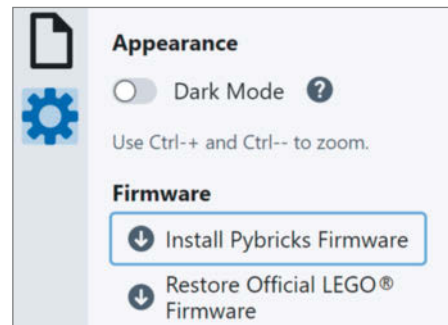
- **City Hub**, **Technic Hub** und **Boost Move Hub** (1 obere Reihe) funktionieren mit allem Powered Up-Zubehör. Firmware und Programme werden per BLE übertragen.
- Der **SPIKE Prime Hub** und der baugleiche **Mindstorms Inventor** (1 untere Reihe) funktionieren mit allem Powered-Up-Zubehör, Firmware per USB, Programme per BLE.
- EV3, das alte Mindstorms: MicroPython/Pybricks wird hier mit einer *Visual-Studio-Code*-EV3-MicroPython-Erweiterung programmiert, Lego-Mindstorms-Zubehör, Firmware per Micro-SD

Bei allen Systemen kann die Original-Firmware ganz einfach über die normalen Up-

date-Funktionen von Lego wiederhergestellt werden.

Mit der Einstellung des neuen Mindstorms-Systems auf Basis von *Powered Up* ist die Verfügbarkeit leider nicht mehr so gut. Im offiziellen Lego Shop werden im Mai 2023 noch Roboter-Erfinder-Sets (51515) für 360 Euro ausverkauft. Die Inventor Hubs sind nur noch als Sets für den Schulunterricht erhältlich (dann *SPIKE* genannt, ab etwa 400 Euro). Hier bietet es sich an, auf den Gebrauchtmärkte auszuweichen, wo man Hubs auch einzeln findet.

- **Boost Move Hub** (88006): Neu ca. 80 Euro. Ist leistungsmäßig der schwächste. Bietet Neigungssensoren (IMU), zwei Ein/Ausgänge, hat aber zusätzlich noch zwei Servomotoren fest eingebaut, 6 × AAA Batterien/NiMh-Akkus, die aber schnell leer werden.



4 Der Knopf für die Installation der Firmware

- **City Hub** (88009): hat nur zwei Anschlüsse, keine IMU, schmal, hauptsächlich für die Lego-Eisenbahn, neu 50 Euro, 6 × AAA
- **Technic Powered Up Hub** (88012), vier Aus-/Eingänge, IMU, Taste, RGB-LED, 6 × AA(!) Batterien/NiMh, Neupreis etwa 40 Euro
- **Technic Hub** (klein, 45609), IMU, nur zwei Ports, neu mit 240 Euro sehr teuer, Akku per USB zu laden
- **Technic Hub** (groß, 88016), 6 Ein/Ausgänge, Lautsprecher, 6 RGB-LEDs, IMU, Akku ist wechselbar, nicht mehr verfügbar außer im Set-Ausverkauf (s.o.)
- **Lego Mindstorms** (EV3, 31313): Dieses Set gibt es weiterhin zu kaufen, es ist mit fast 700 Euro recht teuer, kommt dafür aber mit vielen Sensoren und Aktoren und hat den wohl leistungsfähigsten Prozessor (ARM9, Linux). Somit sind der *Move Hub* und der *Technic Powered Up Hub* eine gute Wahl, wenn man mit den Einschränkungen leben kann: Beim *Move Hub* muss man ohne Fließkommazahlen auskommen, von der Rechenleistung ist er der langsamste, bietet keinen Zufallszahlen-Generator, die Fehlerausgaben per Python sind sehr rudimentär (keine Zeilennummern der Fehler), die Batterien sind sehr schnell leer (man kann aber prima einen externen 2S-Lipo-Akkupack dranhängen). Die beiden leistungsfähigen eingebauten Servomotoren sind aber sehr praktisch, um fahrende Roboter zu bauen und relativieren den Preis. Wenn man auf dem Gebrauchtmärkte Ausschau hält, kauft am besten ein komplettes Boost-Set: Damit erhält man dann eine Menge Technic-Teile, den *Color-Distance-Sensor* und einen externen Servomotor.

Der *Technic Powered Up Hub* bietet das volle Programm unter Pybricks mit vier Ein/Ausgängen und läuft mit AA-Akkus. Zum Preis muss man allerdings noch die fast 40 Euro teuren Motoren und die benötigten Sensoren dazurechnen.

Powered-Up-Zubehör

- Normale Gleichstrom-Motoren (DC): Steuerbar sind Laufrichtung, Geschwindigkeit/Kraft und Bremse. Servomotoren (2) mit Encoder können als Motoren, Encoder oder

elektrische Federelemente verwendet werden. Sie halten/regeln ihre absolute Position auch über mehr als 360°. Abfrage von Stromverbrauch, Blockierung, Position, Geschwindigkeit, Kraft. Pybrick-Unterstützung von PID (außer beim Move Hub).

- Sensoren ③: *Tilt Sensor* (IMU, Neigungssensor), Infrarotsensor (Entfernung, Reflexion, Zähler für Objekte), *Color Distance Sensor* (Farbe, Entfernung, Reflexion, kann IR-Befehle zu *Power-Functions*-Empfängern senden, RGB-LED), *Color Sensor* (wie Color Distance Sensor ohne IR), *Ultrasonic Sensor* (Ultraschall-Entfernungsmesser, RGB-LED), *Force Sensor* (misst Kräfte bis 10N und Sensorauslenkung)
- Weiterhin gibt es noch eine Bluetooth-LE-Fernsteuerung (vier Richtungsknöpfe, zwei Feuerknöpfe, eine RGB-LED), 2-fach-Lichter und eine 3×3-RGB-LED-Matrix.

Bei den Sensoren ist man als Maker natürlich etwas verwöhnt, es fehlen Temperatur-, Feuchte- oder Luftdrucksensoren. Dafür sind die Servomotoren sehr vielseitig: Gleichstrommotoren, Schrittmotoren und (Modellbau-) Servos sind für den Maker leicht verfügbar. Ein Servomotor, der nicht nur jeden Winkel einstellen, sondern auch kontinuierlich drehen kann, der immer seine Position kennt und sie bei Bedarf hält, der seine Stromaufnahme abfragen und ein Blockieren (Stall) erkennen kann, der aber auch als Eingabegerät per Encoder dienen kann, ist nicht so leicht zu bekommen.

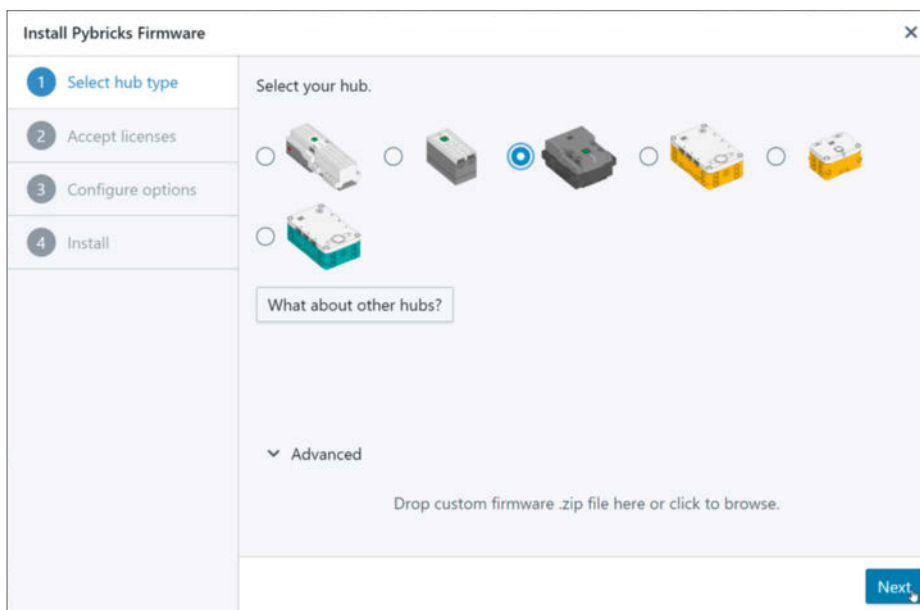
Installation

Die Installation der Pybricks Firmware kann über das Zahnradsymbol ④ (Einstellungen) erfolgen oder wird beim Versuch, einen Hub mit originaler Lego-Firmware zu koppeln, aufgerufen. Dadurch wird ein interaktiver Prozess ⑤ gestartet, der die notwendigen Abfragen wie Hub-Typ, Lizenzvereinbarung und den gewählten Hub-Namen abfragt.

Beim Updatevorgang über Bluetooth ist es am kritischsten, die Taste lange genug gedrückt zu halten und dabei noch eine Hand für die Maus frei zu haben: Am besten startet man das Demovideo und folgt dann der Liste, die die entsprechenden Punkte durch Fettschrift anzeigt. Dann den Knopf nicht loslassen, im Bluetooth-Dialog den Hub auswählen und auf Koppeln klicken. Nach wenigen Augenblicken kommt der erlösende Dialog, man könne den Knopf nun loslassen, ein paar Minuten später ist die Firmware auf dem Board. Den ganzen Vorgang zeigen wir auch noch einmal im Video zum Artikel.

Das erste Pybricks-Programm

Mit der Pybricks-Firmware haben wir nun einen MicroPython-Interpreter auf dem Hub. Ein Hello-World-Programm könnte einfach



⑤ In ein paar einfachen Schritten wird man durch die Installation geleitet.

aus einem `print("Hello Pybricks World!")` bestehen, aber das wäre langweilig. Zumindest sollte eine spezielle Hardware des Bricks angesteuert werden. Eine solche Hardware ist die eingebaute RGB-LED, die alle Hubs haben: Im folgenden Listing blinkt diese LED mit 1Hz für 10 Sekunden, dann wird das Programm verlassen. Je nach Hub muss hier bei `from pybricks.hubs import` der Hubtyp angepasst werden, im Beispiel (siehe auch Video) wird ein *TechnicHub* verwendet, für den *Boost Hub* wäre *MoveHub* zu importieren und auch in Zeile 5 (`hub = ...`) zu verwenden.

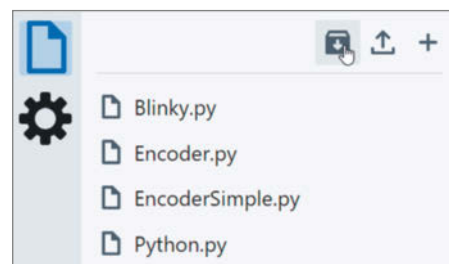
```
from pybricks.hubs import TechnicHub
from pybricks.parameters import Color
from pybricks.tools import wait
```

```
hub = TechnicHub()
hub.light.blink(Color.GREEN, [500, 500])

wait(10000)
```

Die Klassen und Methoden der einzelnen Komponenten sind im API-Panel rechts sehr gut aufbereitet und mit Beispielen illustriert, sodass man zusammen mit der Autovervollständigung und den Popups in der IDE nur in seltenen Fällen auf externe Dokumentation zurückgreifen muss. Eine weitere sehr interessante Hardware, die in den meisten Hubs vorhanden ist, ist der Lagesensor (IMU), mit dem man die Lage des Bricks als Winkel um zwei Achsen erhält. Ein einfaches Beispiel *Reading the tilt value* findet sich in der API-Dokumentation der Hubs.

Um das Python-Skript auf den Hub zu bekommen, benutzt man den Play-Button oder *F5*, gestoppt wird das Programm mit dem Stop-Button oder *F6*. Fehler bei der Ausführung werden im Konsolen-Tab gemel-



⑥ Mit diesem Knopf sollte man regelmäßig die Skripte sichern.

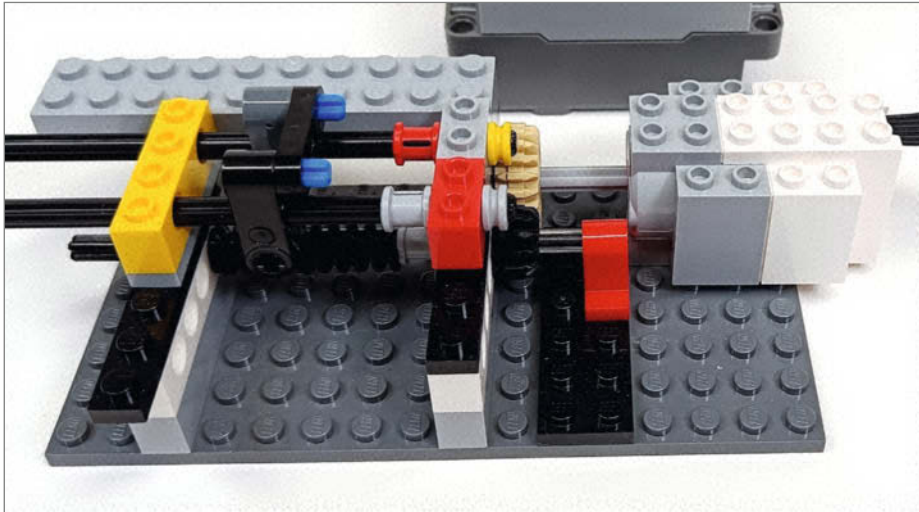
det, beim *Move Hub* leider ohne Zeilennummern.

Die Programme werden in einer internen Datenbank im Browser gespeichert. Löscht man die Browserdaten, so sind auch die Programme weg, daher am besten mit dem Backup-Button ⑥ ein ZIP erstellen und auf der Festplatte speichern. Darin werden die einzelnen Python-Skripte gespeichert und können so per Copy&Paste wieder in die IDE geladen werden.

Soll ein Programm dauerhaft auf dem Brick verbleiben, muss man es einmal auf dem Hub ausführen und dann den Brick mit dem Taster ausschalten (lange drücken, bis die LED erlischt). Nun kann dieses Programm auch verwendet werden, wenn der Hub nicht per Bluetooth gekoppelt ist. Dazu den Hub per Button einschalten – dann genügt ein kurzer Druck auf den Button und das Programm läuft.

Servomotoren

Wie bereits kurz erwähnt, stehen für das Powered-Up-System interessante Motoren zur Verfügung. Für reine Antriebe gibt es einen



7 Schnecken-Linearantrieb-Testmodell



8 Detail des X-Achsen-Schlittens

Weitere Bauideen

Viele meiner Ideen habe ich natürlich auch auf den einschlägigen Lego-Seiten und in der Projektliste von Pybricks (siehe Links) gefunden, aber das heißt nicht, dass man es nicht besser machen kann.

- » Zahlencodeschloss wie bei einem Safe: ein Motor als Encoder, ein weiterer Motor öffnet den Safe.
- » Man kann das Drehmoment der Servomotoren in Milli-Newton abfragen – mit einem Wägearm bekannter Länge sollte man eine brauchbare Waage bauen können.
- » Mit den Neigungssensoren in den Hubs sollte es möglich sein, einen Winkelmesser

oder eine Wasserwaage zu bauen. Ein Zeiger an einem Servomotor könnte den Winkel anzeigen.

- » Kann man ein kardanisches Stativ bauen?
- » Schildkröten-Zeichenroboter: Ein Roboter, der seinen Weg aufzeichnet und den Stift heben/senken kann.
- » Ein einfacher Linienfolger ist mit einem Move Hub schnell zusammengesteckt, aber die Programmierung ist wegen der fehlenden Fließkomma-Unterstützung aufwendiger. Auf den anderen Hubs ist es viel einfacher und mit den PID-Reglern kann man hier einen schnellen und genauen Linienfolger bauen.

normalen Gleichstrommotor und eine Antriebseinheit für Lego-Eisenbahnen. Beide können über den Hub in Richtung und Geschwindigkeit gesteuert (PWM, Power) und auch elektrisch gebremst werden. Die übrigen acht Motoren verfügen jeweils über Encoder, die die Position eines Motors erfassen und regeln können, sie sind also praktisch Servomotoren, die es in verschiedenen Größen und Ausführungen gibt.

Diese Servomotoren halten eine gewünschte Position durch Regelung und können auch für die Eingabe durch Drehung der Achse verwendet werden. Mit dem Skript in dem folgenden Listing wird ein solcher Servomotor zu einem Eingabeknopf, die Position der Achse wird in ganzzahligen Grad angezeigt. Dies wird auch im Video zum Artikel gezeigt.

```
from pybricks.hubs import TechnicHub
from pybricks.pupdevices import Motor
from pybricks.parameters import Port
from pybricks.tools import wait
```

```
hub = TechnicHub()
motor= Motor(port=Port.A)
```

```
while 1:
    print(motor.angle())
    wait(100)
```

Durch die Stalling-Funktion (Blockierererkennung) kann z. B. das automatische Homing ohne Endschalter, wie es moderne 3D-Drucker bieten, erprobt werden. Aber auch eine Kollisionserkennung ohne weitere Sensoren für Roboter ist damit realisierbar.

Dazu gibt es direkt in der Motorklasse die Funktion `run_until_stalled()`, die man mit dem Parameter `duty_limit` auf die maximale Kraft einstellen sollte, um keine Teile zu beschädigen (oder das Modell zu zerlegen) und trotzdem eine sichere Erkennung zu erhalten.

Im Beispiel (siehe Video) wird dies verwendet, um die beiden Grenzen für einen Schneckenantrieb 7 zu bestimmen und dann den Servomotor zufällig innerhalb dieser Grenzen zu bewegen. Weitere Funktionen aus der API erlauben es, den Motor auf einen bestimmten Winkel anzufahren, ihn eine bestimmte Zeit oder eine bestimmte Anzahl von Grad laufen zu lassen. Bei allen Funktionen können Parameter wie Geschwindigkeit, Drehmoment und ob der Motor blockiert ist, abgefragt werden, was ausgeklügelte Steuerungen, aber auch elektronische Aufhängungen (aktive Radaufhängung) ermöglicht.

Darüber hinaus bietet Pybricks auf allen Hubs außer dem Move Hub auch Funktionen, um die Motoren per PID-Regelung (Proportional Integral Derivative) zu steuern. Dies ermöglicht selbstbalancierende Roboter mithilfe von Positionssensoren in den HUBs oder autonome Steuerungen.

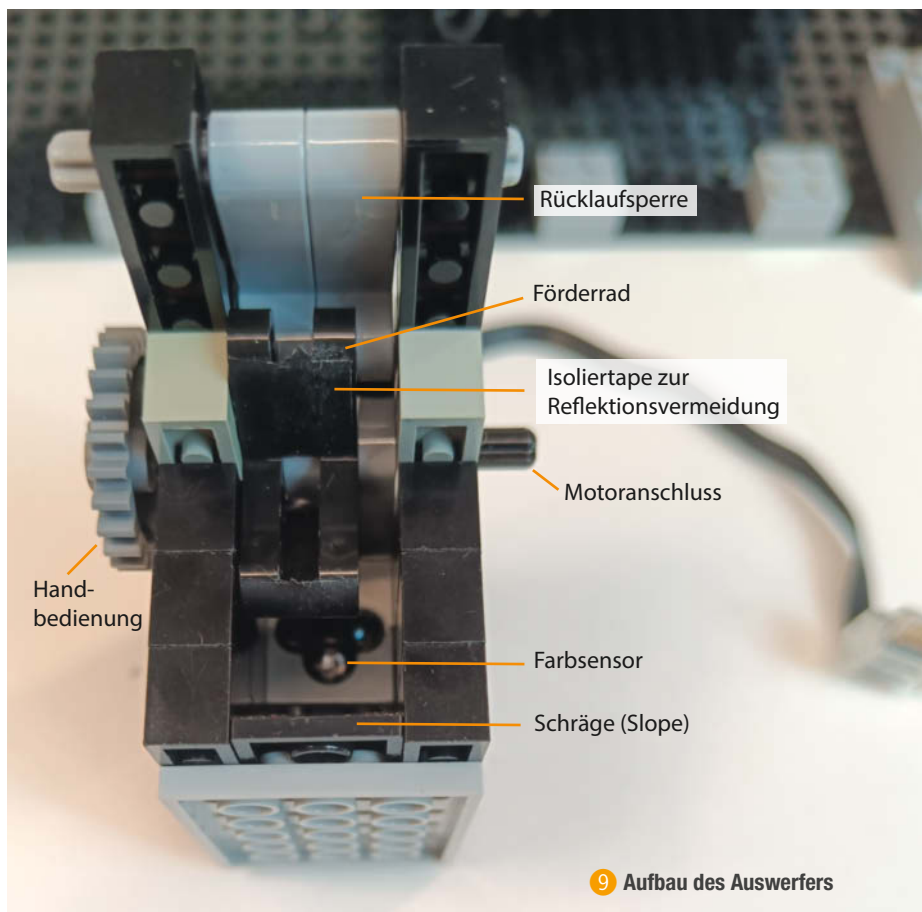
M&Ms-Sortierer

Ja, ich weiß, die schmecken alle unabhängig von der Farbe gleich. Aber wenn ich bastle, dabei nasche und einen Farbsensor in der Hand habe, kommen mir halt so komische Einfälle. Zugleich lernt man durch die natürliche Variation in Farbe und Form einiges über Systeme, die echte Objekte und nicht nur gleichförmige Testobjekte verarbeiten sollen. Und so ganz nebenbei auch einiges über Technik und Programmierung.

Der Sortierer wurde in mehreren (vielen) Anläufen bis zum heutigen Zustand gebaut. Als Transportband dient eine Raupenkette, der Auswerfer rutscht auf Lego Technic-Beams ⁸ wie auf Schienen, kurze Winkelarme greifen unter die Schienen und kleine Rollen sorgen für Leichtgängigkeit und geringes Spiel. Angetrieben wird das Band durch einen Motor des MoveHub, beim Start fährt der Auswerfer an die kleinen Winkelarme und nullt sich so (s. o. Stall-/Blockadeerkennung). Beim Start der Maschine fährt das Auswerferrad rückwärts gegen eine Sperrklinke, um zu nullen. Das Transportband arbeitet ohne Probleme und ist auch recht schnell und genau. Details zu den Konstruktionen finden Sie im GitHub zum Artikel (siehe Link in der Kurzinfo).

Der Auswerfer und die Farberkennung ⁹ sind viel komplizierter. Mit Lego kann man praktisch nur auf einem Raster bauen (mit Tricks auch auf einem halben Raster), aber die M&Ms halten sich nicht an ein Raster und zwei M&Ms werden immer einen Weg finden, sich im Magazin oder im Auswerfer zu verklemmen.

Die M&Ms fallen also über eine Schräge aus dem Magazin auf einen Arm eines Auswerferrades mit vier Kammern. Bei jeder 90°-Drehung wird ein M&M vor den Farbsensor befördert und das vorherige ausgestoßen. Wird die Farbe erkannt, fährt das Band an die entsprechende Stelle und das M&M vor dem Sensor wird in das richtige Fach befördert. Das



ausführlich kommentierte Programm mit gerade einmal 112 Zeilen findet sich in unserem GitHub (Link in Kurzinfo).

Die zu erkennenden Farben werden dem Sensor im Programm bekannt gemacht. Dazu werden mit einem kleinen Hilfsprogramm die einzelnen Farbwerte im HSV-System (Hue Saturation Value, Farbwert, Sättigung, Helligkeitswert) der M&Ms gemittelt. Dadurch kann der Farbsensor die Farben zuverlässiger erkennen. Leider hat der Sensor eine Rot-Schwä-

che, er kann nicht immer korrekt zwischen Rot und Orange unterscheiden. Auch die braunen M&Ms fielen aus dem Spiel, hier erkennt der Sensor kaum etwas oder nur `Color.NONE`. Eventuell kann man durch eigene Verarbeitung der HSV-Farbwerte (KI?) noch etwas verbessern, aber das ist ein anderes Projekt. Ich werde auf jeden Fall weiter mit Pybricks experimentieren und mein Sohn möchte einen fahrbaren Raupenschlepper mit der BT-Fernsteuerung bauen. —caw

TECHNIKUNTERRICHT MACHT ENDLICH SPAß!



Make:Education

Mit **Make Education** erhalten Sie jeden Monat kostenlose Bauberichte und Schritt-für-Schritt-Anleitungen für einen praxisorientierten Unterricht:



Für alle weiterführenden Schulen



Fächerübergreifend



Digital zum Downloaden



Monatlicher Newsletter

© Copyright by Maker Media GmbH.

Jetzt kostenlos downloaden: make-magazin.de/education

Tipps & Tricks

Helferlein: Code-Snippets im PICAXE-Editor

Um einerseits Tipparbeit zu sparen und andererseits die korrekte Befehlssyntax zu verwenden, gibt es im PICAXE-Editor ein nützliches Feature. An der aktuellen Cursorposition im Quellcodebereich können sogenannte *Code Snippets* eingefügt werden. Hierzu öffnet man per Rechtsklick mit der Maus das Kontextmenü und wählt dort *Insert/Code Snippets* aus. Zur Auswahl stehen verschiedene *do-loop* und *for-next*-Schleifentypen, I²C-Initialisierungen, sowie *if-then* und *select-case*-Konstrukte. Per Doppelklick auf den jeweiligen Menüpunkt kann das passende Template ausgewählt werden. Dadurch erspart man sich zum Teil auch einen Blick in die Hilfe zur jeweiligen Befehlssyntax. Abhängig vom eingefügten Code Snippet sind die erforderlichen Parameter wie beispielsweise Variablennamen oder Startwerte farblich markiert und können dann direkt individuell per Tastatureingabe angepasst werden. Mit der TAB-Taste kann das nächste Element ausgewählt werden.

Im Beispiel haben wir eine inkrementierende *for-next*-Schleife gewählt. Dadurch wird automatisch das passende Gerüst dieser Schleife an der aktuellen Cursorposition eingefügt, das wir nun individuell parametrieren können. Nach der Anpassung des Defaultnamens *b0* für die Zählvariable auf den gewünschten Variablennamen gelangen wir mit der TAB-Taste zum Startwert der Schleife, danach weiter zum Endwert, gefolgt von der Schrittweite. Abschließend müssen wir

```
1 for b0 = 1 to 10 step 1
2
3 next b0
```

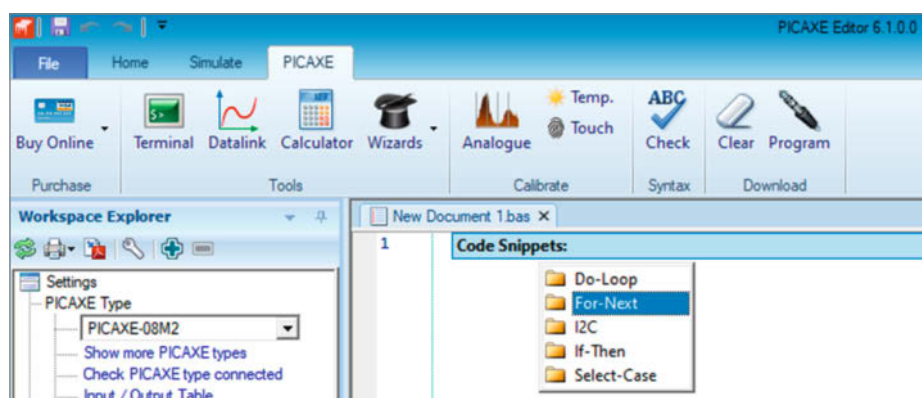
... können die farblich markierten Parameter mit der TAB-Taste nacheinander ausgewählt und individuell angepasst werden.

Ein Tipp für den PICAXE-Editor spart viel Tipparbeit und vermeidet Fehler. Handfester wird es bei dem Ikea-Hack für den Helmer Unterschrank, der die Schubladen zugänglicher macht.

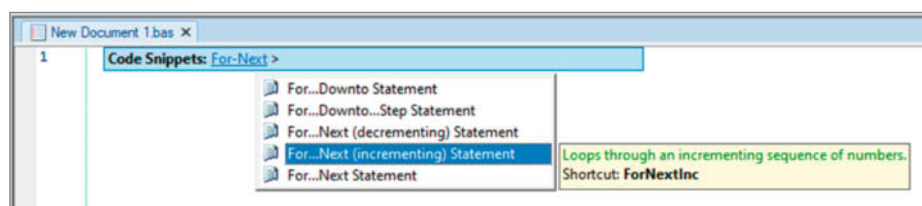
von Michael Gaus, Miguel Köhnlein und Carsten Wartmann

entweder nochmals den entsprechenden Variablennamen nach der *next*-Anweisung eingeben, da dieser leider nicht schon automatisch beim ersten Schritt mit angepasst wird, oder wir löschen diesen optionalen Ausdruck einfach, sodass nur *next* stehen bleibt, was ebenfalls eine korrekte Befehlssyntax ist.

Mit etwas Handarbeit ist es auch möglich, eigene Code Snippets anzulegen und anschließend bequem per Mausklick im Quellcode einzufügen. Wir konnten zwar keine offizielle Dokumentation hierzu entdecken, sind aber durch Ausprobieren zum Ziel gekommen. Im PICAXE-Editor klickt man bei *File/Help* auf den Link



Per Mausklick können Code-Vorlagen wie Schleifen eingefügt werden.



Nach Auswahl der inkrementierenden *for-next*-Schleife ...

ProgramData > Revolution Education > PICAXE Editor > Settings > Code Snippets A >			
Name	Änderungsdatum	Typ	
Do-Loop	18.08.2020 19:54	Dateiordner	
For-Next	18.08.2020 19:54	Dateiordner	
I2C	18.08.2020 19:54	Dateiordner	
If-Then	18.08.2020 19:54	Dateiordner	
Select-Case	18.08.2020 19:54	Dateiordner	

Die Snippets befinden sich im Unterverzeichnis namens Code Snippets A.

```

1  #rem
2  /**
3   * @brief Short description.
4   * @param b0 Input parameter
5   * @return b1 Result
6   */
7  #endrem
8
9  subname:
10
11  return
12

```

Das Snippet für eine Subroutine mit Funktionsbeschreibung in Aktion

```

1  hi2cout address, (b0)
2

```

Im eingefügten I²C-Write-Snippet sind Adresse und Daten konfigurierbar

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <CodeSnippets xmlns="http://schemas.microsoft.com/VisualStudio/2005/CodeSnippet">
3    <CodeSnippet Format="1.0.0">
4      <Header>
5        <Title>I2C write</Title>
6        <Author>Make</Author>
7        <Description>Write data to I2C bus.</Description>
8        <Shortcut>hi2cout</Shortcut>
9      </Header>
10     <Snippet>
11       <Imports>
12       </Imports>
13       <Declarations>
14         <Literal>
15           <ID>address</ID>
16           <Type>Variable or Number</Type>
17           <ToolTip>Replace with the byte or word address.</ToolTip>
18           <Default>address</Default>
19         </Literal>
20         <Literal>
21           <ID>data</ID>
22           <Type>Variable or Number</Type>
23           <ToolTip>Replace with the data byte(s) to be written.</ToolTip>
24           <Default>b0</Default>
25         </Literal>
26       </Declarations>
27       <Code Language="PICAXE" Kind="method body"><![CDATA[hi2cout $address$, ($data$)
28     ]]></Code>
29     </Snippet>
30   </CodeSnippet>
31 </CodeSnippets>
32

```

So sieht die XML-Datei unseres i2c_write.snippet aus.



// heise devSec()

Die Konferenz für sichere
Software- und Webentwicklung

11.–13. September 2023 // Karlsruhe

Sichere Software beginnt vor der ersten Zeile Code

Security ist fester Bestandteil der Softwareentwicklung – vom **Entwurf** über den **Entwicklungsprozess** bis zum **Deployment**. Die **heise devSec** hilft Ihnen dabei mit Vorträgen zu den wichtigsten Themen wie Software Supply Chain, Kryptografie und der Auswirkung von KI auf die Sicherheit.

Aus dem Programm:

- // Das ABC sicherer Webanwendungen
- // Software Supply Chain Security mit dem SLSA
- // Multifaktor-Authentifizierung in der Praxis
- // Skalierung von Sicherheit in Kubernetes
- // Erweiterung des Secure Development Lifecycle um Privacy by Design
- // Wie man mit Mathematik eine Bank übernehmen kann

**JETZT
FRÜHBUCHER-TICKETS
SICHERN!**

www.heise-devsec.de

Veranstalter

 heise Security

 heise Developer

 dpunkt.verlag

Gold-Sponsoren

© Copyright by Maker Media GmbH.
 openText Cybersecurity

 WIBU
SYSTEMS

Bronze-Sponsor

 andrena
OBJECTS

PICAXE Editor Common Settings Folder. Dadurch öffnet sich im Dateixplorer das *Settings*-Verzeichnis. Dort findet man im Unterverzeichnis *Code Snippets A* weitere Unterverzeichnisse mit den bekannten Code-Snippets.

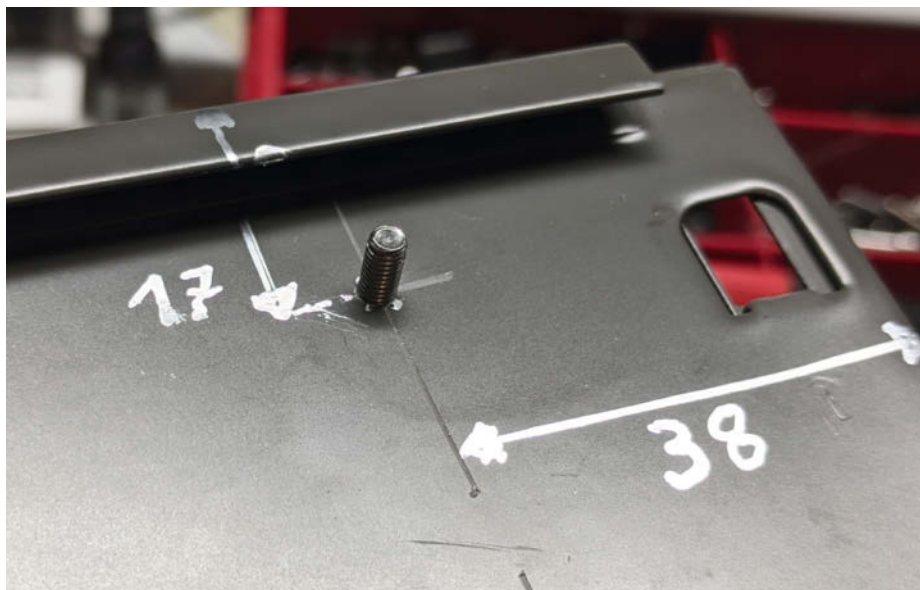
Hier legt man ein neues Unterverzeichnis an, z. B. *Make*. Anschließend kopiert man ein bestehendes Snippet in dieses Verzeichnis und benennt es um. Bei den Snippet-Dateien handelt es sich um Textdateien im *XML*-Format mit der Dateiendung *.snippet*. Wir haben die Datei *ForNext.snippet* kopiert und in *i2c_write.snippet* umbenannt, denn wir möchten als einfaches Beispiel das Grundgerüst zum Schreiben von Daten per I²C-Bus erstellen.

Mit einem Texteditor lässt sich nun diese Datei bearbeiten. Zunächst ändert man im Bereich *<Header>* die vier Texte bei *<Title>*, *<Author>*, *<Description>* und *<Shortcut>* entsprechend ab. Anschließend definiert man im Bereich *<Declarations>* die gewünschten Parameter, die später bei der Verwendung des Snippets mit der *TAB*-Taste ausgewählt und individuell angepasst werden können. Jeder Parameter ist in einem Bereich *<Literal>* eingebettet und benötigt eine eindeutige Kennung, die bei *<ID>* definiert wird. Zusätzlich wird noch der Typ, ein Tooltip sowie der Defaultwert angegeben. Im Beispiel benötigen wir zwei IDs und nennen diese *address* und *data*. Abschließend wird im Bereich nach dem Ausdruck *<CDATA[* der einzufügende Quellcode eingetragen. Zeilenumbrüche sind ebenfalls möglich. Hier können auch die bei *<ID>* angegebenen später konfigurierbaren Parameter aus den *<Literal>* Bereichen eingetragen werden, wobei die ID-Bezeichnung jeweils mit einem Dollarzeichen \$ am Anfang und Ende umschlossen werden muss. Die Syntax für die ID *address* lautet also *\$address\$*. Der Eintrag des Quellcodes wird mit *]]>* abgeschlossen.

Nach dem Neustart des PICAXE-Editors sollte das neu erstellte Snippet nun in der Auswahlliste in der Rubrik *Make* unter dem Namen *i2c_write* verfügbar sein. Das bietet den Vorteil, dass man sich den zugrunde liegenden Befehl *hi2cout* nicht mehr merken muss. Nach dem Einfügen können mit der *TAB*-Taste die beiden konfigurierbaren Parameter für die Adresse und die Daten ausgewählt werden, um diese individuell anzupassen.

In einem etwas umfangreicheren Beispiel haben wir noch ein Snippet für eine Subroutine (siehe Kurzlink zum Artikel) erstellt, bei dem zusätzlich noch ein Kommentarbereich im *Doxygen*-Format zur Kurzdokumentation des Unterprogramms vorangestellt wird. Die zugehörige XML-Datei lautet *subroutine.snippet*. Damit kann man bequem per Mausklick das entsprechende Grundgerüst für eine Funktionsbeschreibung einfügen.

Miguel Köhnlein und Michael Gaus



Position der Schraube in Millimeter an der Schubladenseite

Sicht in den letzten Schubladenwinkel: Ikea-Helmer-Hack

Der Schubladenschrank *Helmer* von Ikea ist ein preiswerter Blechschrank für Büro und Werkstatt, wenn es mal kein Pressholz sein soll. Nur die recht einfache Konstruktion der Schubladenauszüge hat mich gestört. Der Auszug wird durch Metalllaschen gebremst. Leider so früh, dass man bei normalem Blickwinkel nicht ganz nach hinten in die Schublade schauen kann. Wenn man sich verrenkt, sieht man auch nicht mehr, da eine normale Deckenbeleuchtung kaum Licht ins Dunkel bringt.

Mein Hack besteht darin, pro Schublade zweimal zu bohren, zwei Gewinde zu schneiden und bei eingeschobener Schublade eine Maschinenschraube M3×20 einzudrehen, was mit einer Taschenlampe und einem guten Sechskantschraubendreher kein Problem ist. Die Laschen an den Laden müssen natürlich

ggf. wieder zurückgebogen werden. Wo die Schraube sitzen muss, habe ich auf dem Bild eingezeichnet. Mal sehen, wie lange die Gewinde in dem dünnen Blech bestehen, bisher hält alles. Wenn hier mal das Gewinde ausleierte, muss ich wohl eine flache Kontermutter zwischen Schrankwand und Lade fummeln, was mit ein wenig Kleber, der die Mutter hält, gut machbar sein sollte.

Mit dem Hack bekommt man etwa 8cm mehr Auszug und sowohl Licht als auch Sicht reichen nun bis in die hinteren Ecken der Schubladen, sodass gesuchtes Werkzeug sich nicht mehr verstecken kann.

Carsten Wartmann

Alles zum Artikel im Web unter make-magazin.de/xy4t

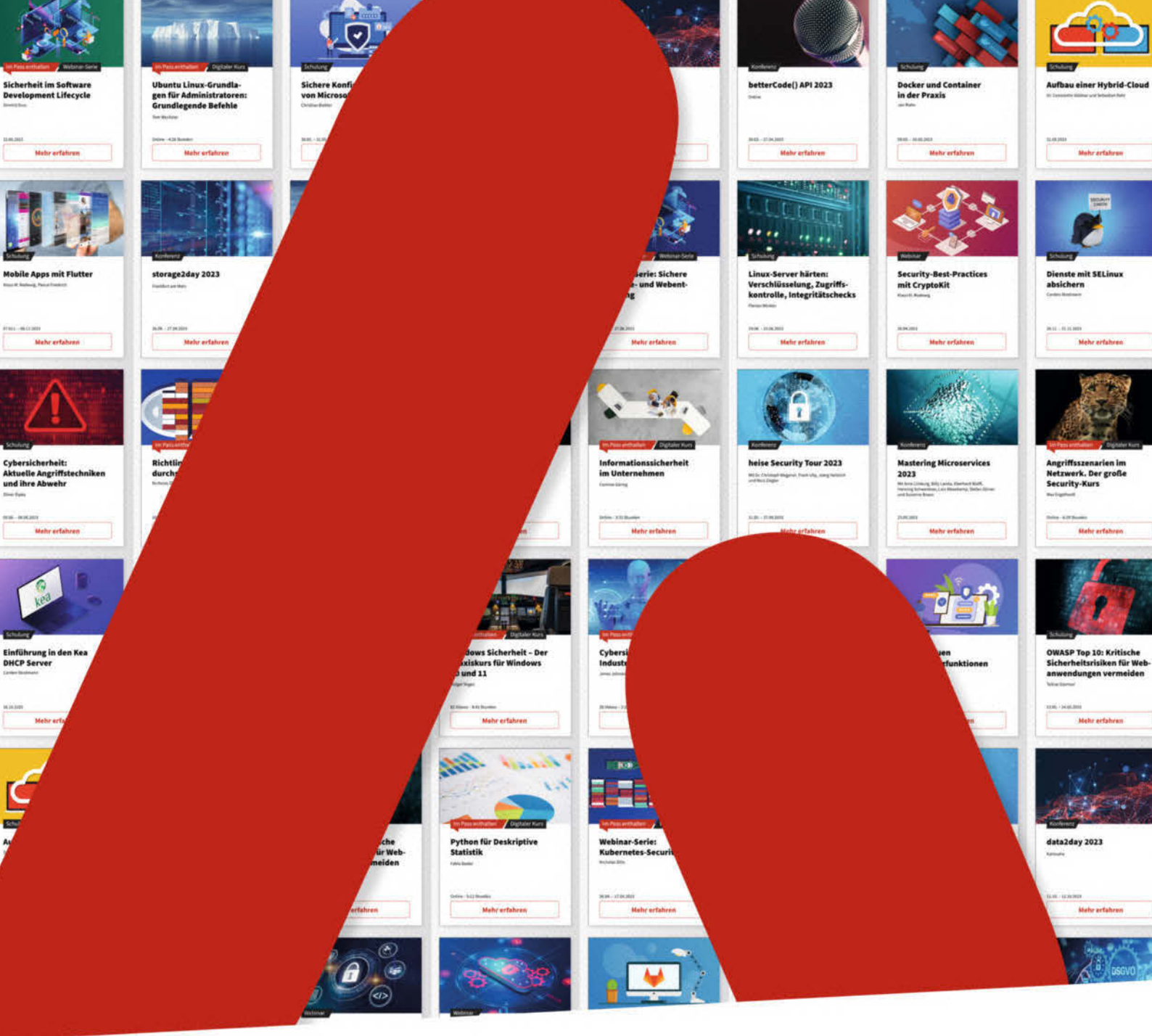


Machen Sie mit!

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, Upcycling ...).

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



Wissenslücken? Nicht mit uns!

Wir helfen Ihnen dabei, die IT-Themen zu lernen, die heute – und morgen – wichtig sind.

Die Zukunft des Lernens ist digital:

Die heise Academy bietet Ihren IT-Teams die Weiterbildungslösungen an, die Sie benötigen. Lassen Sie Ihre Fachkräfte nach Bedarf und direkt am Arbeitsplatz lernen.

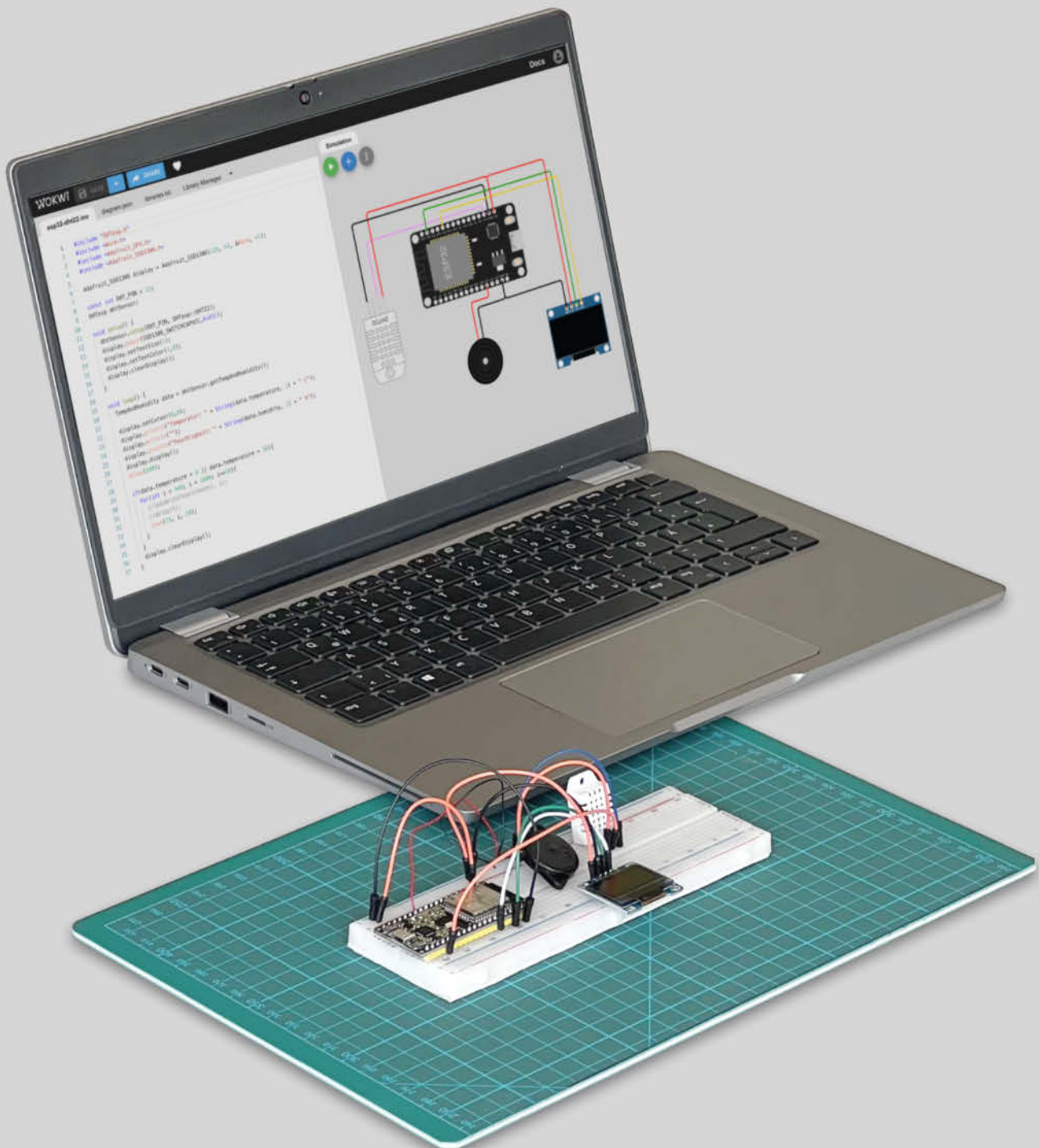
Intensivieren Sie diese Lernerfahrung mit relevanten, topaktuellen Schulungen und Webinaren. Sichern Sie sich das IT-Wissen, das Ihr Unternehmen heute – und morgen – braucht: bei **heise Academy, dem Zuhause Ihrer professionellen IT-Weiterbildung.**



Mikrocontroller-Projekte simulieren mit Wokwi

Ob zu Hause oder unterwegs: Mit dem Simulator Wokwi lassen sich Projekte mit ESP32, Arduino und Raspberry Pi Pico schnell im Web-Browser erstellen und testen. Wir zeigen euch, wie das geht.

von Ákos Fodor



Projektideen finden einen manchmal in den ungünstigsten Situationen: unterwegs, unter der Dusche oder wenn man es sich gerade auf dem Sofa gemütlich gemacht hat. Manchmal wartet man aber auch einfach nur auf die bestellte Hardware, um sein Projekt endlich in die Tat umsetzen zu können. Wer vorab schon seine Ideen festhalten und testen will, kann dafür einen Simulator verwenden wie das von Uri Shaked ins Leben gerufene *Wokwi*.

Damit ihr wisst, was ihr in etwa von dem Simulator erwarten könnt, gibt euch dieser Artikel einen Einblick in die verfügbaren Werkzeuge. Außerdem zeige ich euch anhand eines Beispiel-Projekts, wie schnell man mit Wokwi erste Prototypen erstellen kann.

Simulieren im Browser

Wokwi ist eine Web-Anwendung, mit der man Mikrocontroller-Projekte in einem Browser planen und simulieren kann, ohne dafür eine zusätzliche Software oder App installieren zu müssen oder durch eine Fehlbeschaltung echte Hardware zu gefährden. Die Berechnungen des Simulators finden, wie bei Web-Apps üblich, auf einem Server statt, sodass der Browser lediglich als Client genutzt wird. Dadurch funktioniert Wokwi auf einer Vielzahl internetfähiger Endgeräte, z.B. auch in Klassenzimmern, die nicht über die neueste Hardware verfügen oder unterwegs auf einem Smartphone ❶ oder Tablet. Dafür muss man sich nicht einmal bei Wokwi registrieren. Wer aber Projekte auf dem Wokwi-Server speichern oder mit anderen teilen möchte, kann sich einen Wokwi-Account erstellen, der wie auch die meisten anderen Funktionen kostenfrei ist.

Eine Handvoll Mikrocontroller

Jedes Wokwi-Projekt beginnt zunächst mit der Wahl eines Mikrocontrollers ❷. Zur Verfügung stehen der *Arduino Uno*, *Nano* und *Mega* sowie diverse *ESP32*-Varianten, der *Raspberry Pi Pico* und sein WLAN-fähiger Nachfolger *Pico W*. Für alle, die es gern noch kleiner haben, hält Wokwi außerdem den *ATtiny85* und den darauf basierenden *Franzininho* bereit. Sofern die virtuelle Hardware es unterstützt, kann man die Kleinstcomputer mit der *Arduino*-Variante von *C++*, *MicroPython*, *CircuitPython* oder *Rust* programmieren.

Außerdem gesellen sich zu den Mikrocontrollern weitere 65 simulierte Elektronik-Komponenten ❸, unter anderem *LEDs*, *Displays*, *Knöpfe* und *Regler*, *Motoren*, *Sensoren*, *Ausgabegeräte* oder auch *Breadboards*, die man allerdings im Simulator nicht wirklich braucht. Ähnlich wie in der Schaltungs-Software *Fritzing* lassen sich die Komponenten nämlich in einem grafischen Editor anordnen und

Kurzinfo

- » Einstieg in den kostenfreien Online-Simulator Wokwi
- » Features, Besonderheiten und Grenzen kennenlernen
- » Einen Temperatur-Alarm mit virtueller Hardware simulieren

Checkliste



Zeitaufwand:
2 Stunden



Kosten:
0 Euro (optional 7 Euro pro Monat für eine Wokwi-Club-Mitgliedschaft)

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



Mehr zum Thema

- » Ákos Fodor, Adafruit GFX Library: Malen mit Zahlen, *Make* 2/23, S. 64
- » Florian Schäffer, Programmierumgebungen: Die richtige IDE für Maker, *Make* 2/19, S. 66
- » Florian Schäffer, Web-IDEs: Programmieren im Browser, *Make* 2/19, S. 80



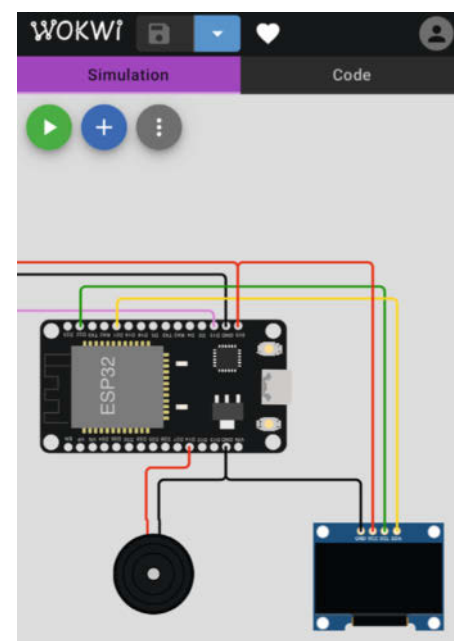
mit virtuellen Kabeln verbinden. Wer sich bei einem bestimmten Teil nicht sicher ist, wie man es verwendet oder z.B. die *Pin-outs* nachschlagen will, gelangt über eine Hilfe-Funktion ❹ schnell auf den zugehörigen Artikel im (englischsprachigen) Handbuch. Neben grundsätzlichen Informationen und Hinweisen findet man dort in der Regel auch ein paar Links zu Beispiel-Projekten. Bei weitergehenden Fragen zu Wokwi hilft außerdem der Community Chat über Discord weiter.

Damit alle Komponenten miteinander funktionieren, benötigt auch der Simulator die passenden *Bibliotheken* (*Arduino*), *Module* (*MicroPython*) oder *Crates* (*Rust*). Wer mit *Arduino*-Code arbeitet, kann bequem auf Wokwis *Library Manager* zurückgreifen, über den sich so ziemlich jede Bibliothek finden lässt, sofern sie in der *Arduino IDE* enthalten ist. Programmiert man seinen Mikrocontroller mit einer der *Python*-Varianten oder *Rust*, kann man die benötigten Dateien einfach in sein Projekt hochladen.

Erweiterte Funktionen

Da Wokwi sowieso über einen Server läuft, liegt es auf der Hand, auch die WLAN-Funktionen des *ESP32* und *Pico W* zu simulieren. Tatsächlich können sich beide Mikrocontroller mit einem virtuellen WLAN oder Webservern verbinden, um beispielsweise Wetterdaten abzurufen oder über *MQTT* (*Message Queuing Telemetry Transport*) mit anderen Geräten zu kommunizieren. Aber auch andersherum

erlaubt es Wokwi, den simulierten *ESP32* als Webserver zu betreiben. In der kostenfreien Version läuft die Kommunikation über *öffentliche Gateways*. Man sollte daher keine sensiblen Daten übertragen. Wer etwas mehr Privatsphäre wünscht und Projekte testen möchte, die auch auf das heimische WLAN zugreifen sollen, kann als *Wokwi-Club*-Mitglied über ein



❶ Selbst auf dem Smartphone funktioniert Wokwi. Allerdings ist die Bedienung mit Fingern schwieriger als mit Maus und Tastatur.

privates Gateway eine direkte Verbindung zum Simulator herstellen.

Darüber hinaus bietet Wokwi einige Funktionen an, die sich noch in der Beta-Phase befinden: Dazu gehört etwa die *Chips API*, mit der man sich eigene Chips für seine Simulationen programmieren kann, sowie diverse *Debugging*-Features, die sich je nach Mikrocontroller unterscheiden. Wer *Visual Studio Code* nutzt, kann Wokwi außerdem auch dort als Erweiterung einbinden.

Mehr mit dem Wokwi-Club

Für 7 Euro im Monat kann man dem *Wokwi-Club* beitreten und damit nicht nur das Projekt finanziell unterstützen, sondern auch ein paar weitere Funktionen freischalten.

Zu diesen gehört die Möglichkeit, eigene oder nicht in Wokwi verfügbare Arduino-Bibliotheken für seine Projekte hochzuladen. Außerdem kann man auf den Datenspeicher des virtuellen SD-Karten-Moduls zugreifen

und dort z.B. eigene Bilder oder Sounds ablegen. Wie im vorherigen Absatz erwähnt, ist auch das private Gateway Teil der Mitgliedschaft. Außerdem kann man entscheiden, ob man seine erstellten Projekte öffentlich (public) oder nicht öffentlich (unlisted) auf dem Server ablegen möchte.

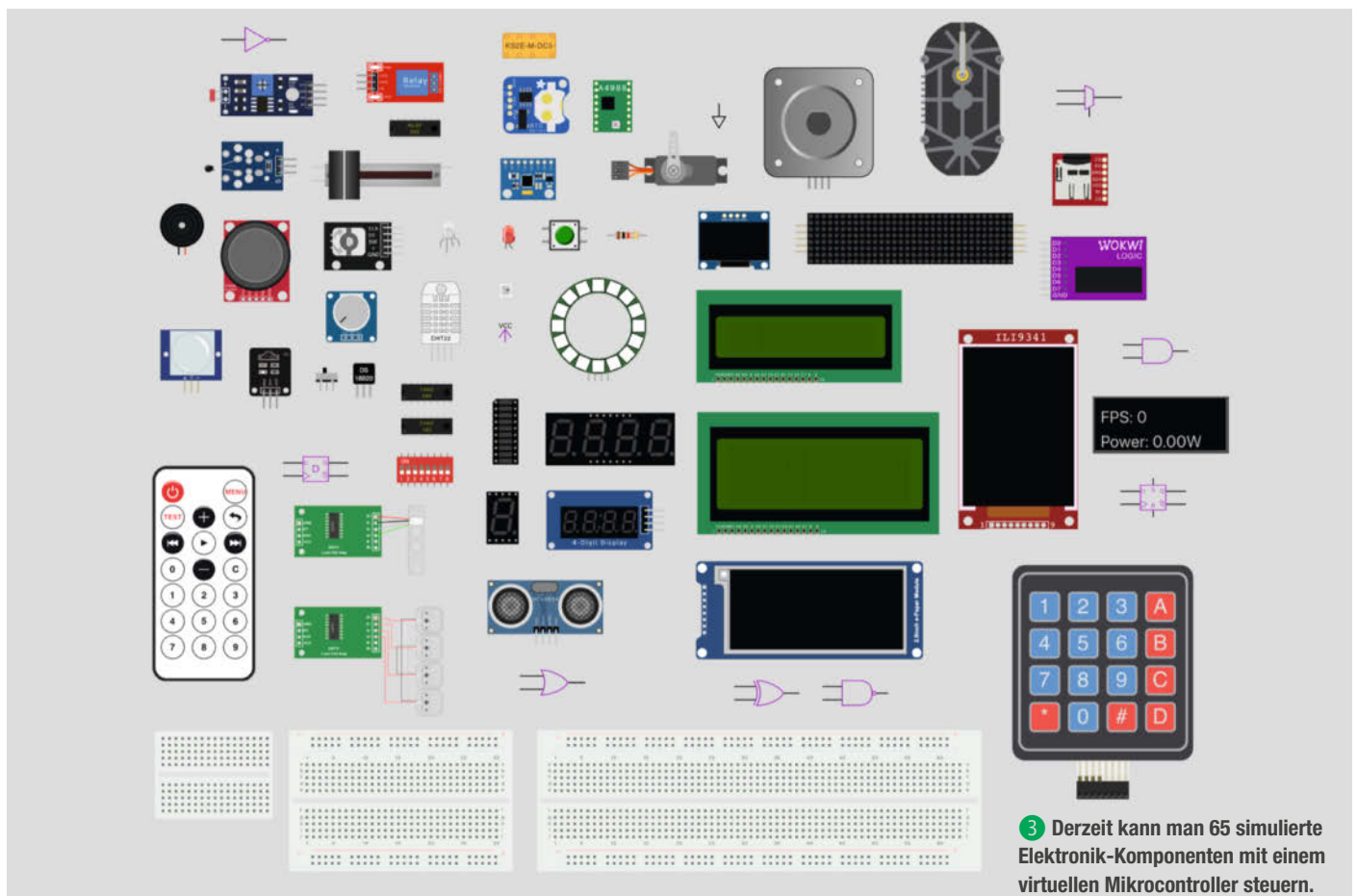
Da Wokwi stetig weiterentwickelt wird, können Wokwi-Club-Mitglieder außerdem mitentscheiden, welche Features als Nächstes entwickelt werden sollen. Dafür erhält man jeden Monat 14 Vote-Powers, die man auf der Roadmap (siehe Link in der Kurzinfor) auf diverse Verbesserungsvorschläge verteilen kann. Aus diesem System sind bereits ein Debugger und die Anbindung an Visual Studio Code entstanden.

Kein Simulator ohne Grenzen

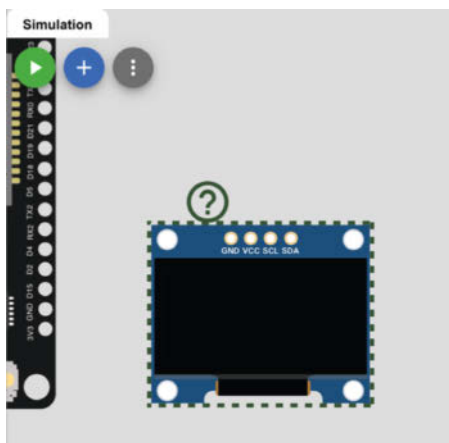
Wokwi wurde für digitale Schaltungen entwickelt und spielt seine Stärken vor allem dann aus, wenn es um die Programmierung und Simulation der Mikrocontroller geht. Denn auch wenn sich variable Widerstände an einen Analog-Pin anschließen lassen, um etwa einen Servo mithilfe eines Poti zu steuern, ist das Programm nicht dafür gedacht, analoge Schaltungen ordnungsgemäß zu simulieren. Dementsprechend löst sich in Wokwi eine LED z.B. nicht in *Magic Smoke* auf, wenn man sie ohne



2 Jedes Wokwi-Projekt basiert auf dem gewählten Mikrocontroller.



3 Derzeit kann man 65 simulierte Elektronik-Komponenten mit einem virtuellen Mikrocontroller steuern.



4 Wer mehr über ein Bauteil erfahren möchte, gelangt mit einem Klick auf das Fragezeichen zu der passenden Stelle im Handbuch.

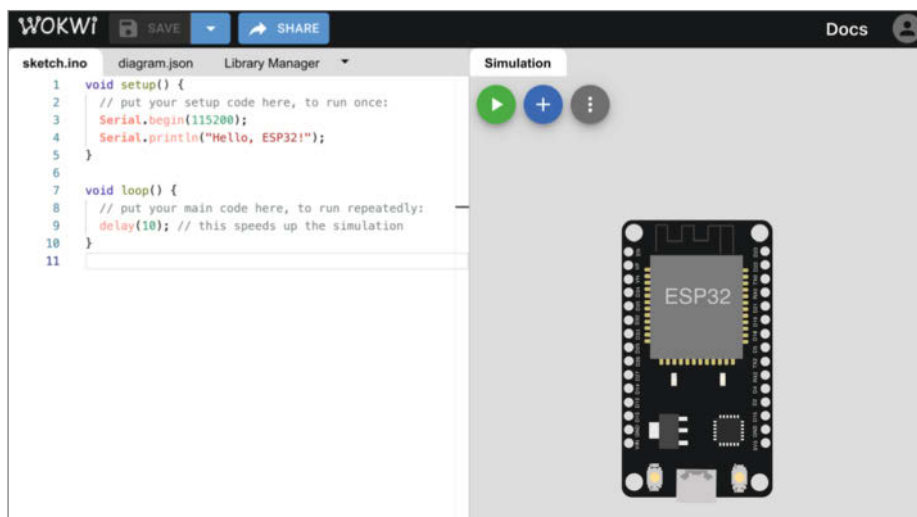
Vorwiderstand an eine Stromquelle anschließt und auch die Ports der Mikrocontroller kann man virtuell nicht überlasten. Wer nicht auf Transistoren, Kondensatoren oder andere analoge Komponenten verzichten mag, findet solche Funktionen beispielsweise bei Tinkercad, dessen Simulations-Funktion wir uns für einen zukünftigen Artikel auch noch anschauen werden.

Ein schneller Testlauf

Wie schnell man in Wokwi ans Ziel kommen kann, zeige ich euch im Folgenden mit einem kleinen Projekt auf Basis eines ESP32, an den ich einen DHT22-Sensor anschließen werde, um die Umgebungstemperatur und -feuchtigkeit zu messen. Zur Ausgabe der Messwerte verwende ich außerdem ein SSD1306-OLED-Display und falls die Temperatur einen vordefinierten Bereich über- oder unterschreitet, wird ein kleiner Buzzer einen Alarmton abspielen.

Um ein neues Projekt zu erstellen, öffne ich als Erstes die Website von Wokwi und scrolle an den vielen Beispiel-Projekten vorbei, bis der Punkt *Start from Scratch* auftaucht. Dort klicke ich auf den ESP32 und finde mich kurz darauf auf Wokwis aufgeräumter Benutzeroberfläche wieder 5, deren Arbeitsbereich in zwei Hälften unterteilt ist: Links befindet sich der *Code-Editor* und rechts der grafische *Diagramm-Editor*, in dem man den Mikrocontroller mit den anderen Bauteilen verdrahtet. Alle Änderungen im Schaltplan werden zeitgleich in der Datei *diagram.json* festgehalten, sodass man Bauteil-Attribute auch nachträglich in Textform ändern kann, etwa die Länge eines Schiebereglers. Zusätzlich erscheint während einer Simulation der serielle Monitor unterhalb des Diagramm-Editors.

Ich füge als Nächstes den Sensor, das Display und den Buzzer im Diagramm-Editor hinzu, indem ich auf das Plus-Symbol im blauen Kreis klicke 6.



5 Die Benutzeroberfläche von Wokwi teilt sich in zwei Hälften: links der Code-Editor, rechts der Diagramm-Editor.

Tastenkürzel sind der Schlüssel

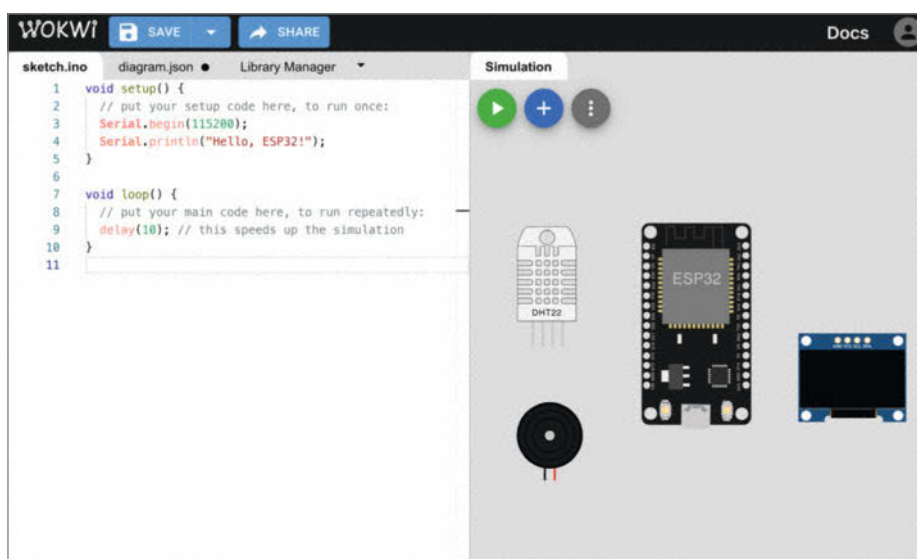
Während das Verschieben der Teile kein Problem ist, stoße ich beim Versuch, den Mikrocontroller und den Buzzer zu drehen, auf Hindernisse, denn dafür gibt es weder eine Schaltfläche noch ein Menü und auch ein Rechtsklick auf die Bauteile hilft mir nicht weiter.

Wokwi verwendet für manche Funktionen Tastenkürzel. Das macht die Benutzeroberfläche auf der einen Seite schön übersichtlich, andererseits muss man die verwendeten Tastenkürzel kennen. Um Komponenten zu drehen, verwendet man etwa die Taste *R* und mit *D* kann man ein einzelnes oder mehrere Objekte duplizieren. Zum Glück gibt es von diesen Befehlen nicht so viele und in der Regel folgen sie gängigen Konventionen. Dennoch hat es mich anfangs verwirrt, dass nur der

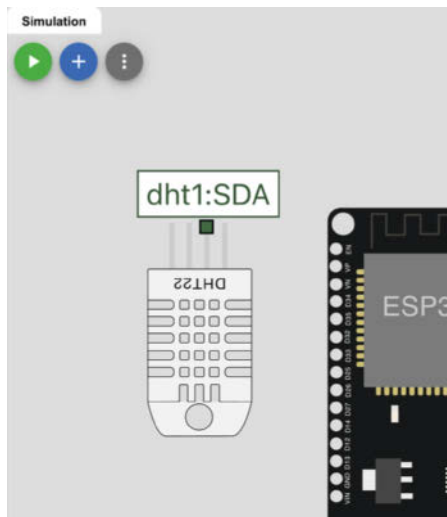
Diagramm-Editor über einen gesonderten Menüpunkt verfügt, über den man etwas rückgängig machen kann, der Code-Editor aber nicht. Dafür verwendet Wokwi nämlich den Widerruf-Befehl des Browsers. Auch hier ist jedoch die einfachste Variante, das Kürzel *Strg+Z* zu verwenden und vorher in den Bereich zu klicken, in dem man etwas rückgängig machen möchte. Wer sich alle Tastaturbefehle ansehen will, kann im Wokwi-Fenster rechts oben auf *Docs* klicken und findet die Auflistung in der Sektion *Guides* bei *Diagram Editor* und *Keyboard Shortcuts*.

Virtuell verbunden

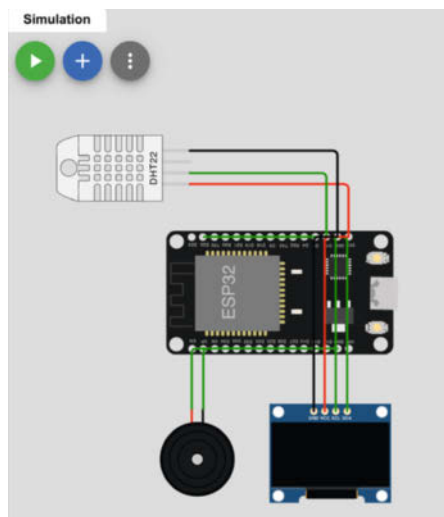
Um den Mikrocontroller mit den übrigen Komponenten zu verbinden, klickt man sich einfach von Pin zu Pin. Diese sind zwar nicht immer beschriftet, aber wenn man



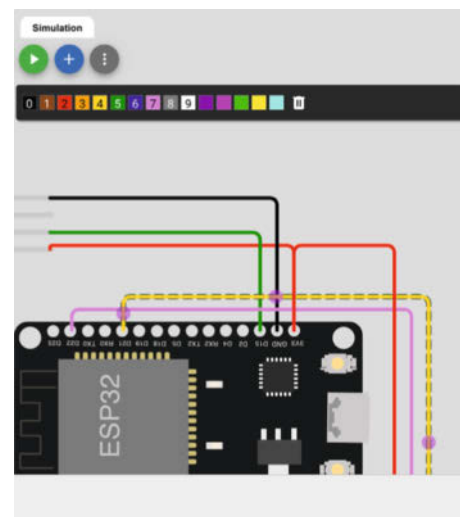
6 Komponenten kann man über die blaue Schaltfläche mit dem Plus-Symbol hinzufügen.



7 Die Pin-Bezeichnung erscheint als Pop-up, wenn man mit der Maus über einen Pin fährt.



8 Wenn man die Pins miteinander verbindet, kann man die Kabel zunächst kaum voneinander unterscheiden.



9 Nach dem Ausrichten und mit ein paar unterschiedlichen Farben sieht der Aufbau schon wesentlich ordentlicher aus.

mit der Maus über einen Pin fährt, erscheint auch bei unbeschrifteten Komponenten ein Pop-up, das uns die jeweilige Funktion anzeigt 7.

Ich verbinde also den *Data*-Pin des DHT-22, der in Wokwi mit SDA bezeichnet ist, mit dem Pin *D15* des ESP32, *VCC* mit *3V3* und *GND* mit *GND*. Als Nächstes schließe ich das OLED-Display an: *SCL* geht zu *D22*, *SDA* zu *D21*, *VCC* zu *3V3* und *GND* zu *GND*. Zum Schluss kommt

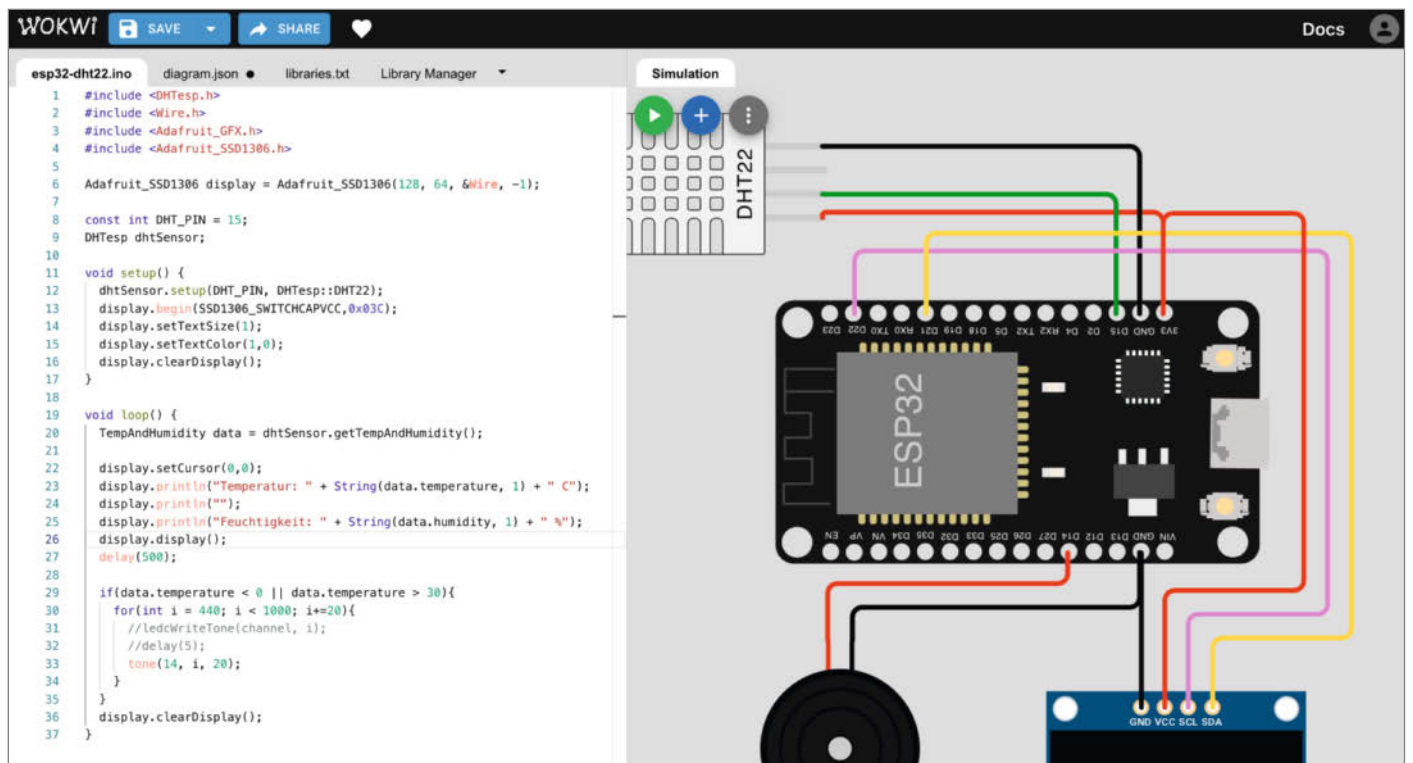
das rote Kabel des Buzzers an *D14* und die andere Verbindung zu *GND* 8.

Nach diesem Schritt richte ich die Bauteile erneut aus und ordne die Kabel, indem ich sie einzeln anklicke und mithilfe der kreisförmigen Ankerpunkte verschiebe. Zur Unterstützung aktiviere ich mit der Taste *G* das Gitter, an dem sich die Kabel beim Ziehen magnetisch ausrichten. Um die Kabel besser unterscheiden zu können, passe ich schließlich noch ihre Farbe

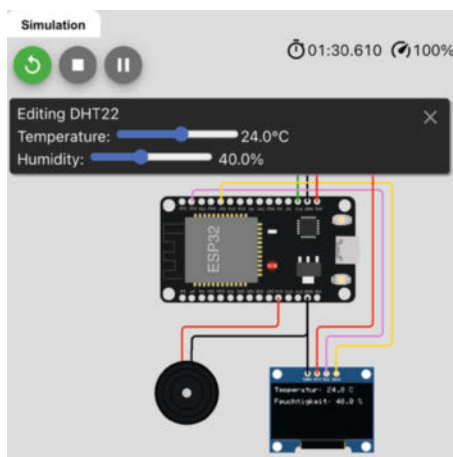
an, indem ich sie anklicke und aus den verfügbaren Farbfeldern eines auswähle 9. Habe ich etwas falsch verbunden, lösche ich Kabel einfach mit einem Doppelklick.

Bibliotheken mit einem Klick

Damit der Mikrocontroller weiß, wie er die angeschlossenen Komponenten ansteuern muss, klicke ich auf den *Library Manager*, dort



10 Das fertiggestellte Projekt ist jetzt bereit für den Test: links der Sketch, rechts der Aufbau.



11 Während die Simulation läuft, kann man den DHT22-Sensor anklicken und mit zwei Schieberegler die Temperatur und Luftfeuchtigkeit einstellen.

auf das *Plus*-Symbol in der rechten oberen Ecke und suche nach den benötigten Bibliotheken: *DHT sensor library for ESPx*, *Adafruit SSD1306* und *Adafruit GFX Library*. Wokwi erstellt daraufhin die Datei *libraries.txt* und fügt sie im Code-Editor als Reiter hinzu. In dieser werden alle verwendeten Bibliotheken aufgelistet. Wenn man eine frühere Version einer Bibliothek benötigt, kann man mithilfe des @-Symbols die Versionsnummer ergänzen, z.B. indem man *Adafruit SSD1306@2.5.4* schreibt.

Programmablauf

Jetzt ist es an der Zeit, dem Mikrocontroller zu sagen, wie er mit den angeschlossenen Komponenten umgehen soll. Dazu klicke ich im Code-Editor auf den Reiter *sketch.ino* und füge als Erstes die Bibliotheken im globalen Bereich des Sketches hinzu und initialisiere das Display sowie den DHT22-Sensor. Damit dessen Messwerte auf dem Display als Text ausgegeben werden, stelle ich danach die Schrift-Skalierung und -farbe im `void setup()` ein.

Mit `TempAndHumidity` liest der ESP32 daraufhin im `void loop()` zweimal pro Sekunde die Temperatur und Feuchtigkeit der Umgebung und gibt diese über den Befehl `print()` auf dem Display aus. Damit der Text nicht aus dem Bild läuft, setzt `setCursor()` den Startpunkt für die Ausgabe auf beiden Achsen jedes Mal auf 0. Als Letztes programmiere ich mithilfe der Funktion `tone()` einen Alarm, der vom Buzzer abgespielt wird, sollte die Temperatur unter 0 fallen bzw. über 30 steigen 10.

Interaktive Simulation

Mit einem Klick auf den *Play*-Button erwacht die simulierte Schaltung schließlich zum Leben und zeigt mir virtuelle Sensordaten auf einem virtuellen Display. Um die Umgebungs-Temperatur und -Feuchtigkeit zu verändern, klicke ich

Temperatur-Alarm

```
#include <DHTesp.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

Adafruit_SSD1306 display = Adafruit_SSD1306(128, 64, &Wire, -1);

const int DHT_PIN = 15;
DHTesp dhtSensor;

void setup() {
  dhtSensor.setup(DHT_PIN, DHTesp::DHT22);
  display.begin(SSD1306_SWITCHCAPVCC, 0x03C);
  display.setTextSize(1);
  display.setTextColor(1,0);
  display.clearDisplay();
}

void loop() {
  TempAndHumidity data = dhtSensor.getTempAndHumidity();

  display.setCursor(0,0);
  display.println("Temperatur: " + String(data.temperature, 1) + " C");
  display.println("");
  display.println("Feuchtigkeit: " + String(data.humidity, 1) + " %");
  display.display();
  delay(500);

  if(data.temperature < 0 || data.temperature > 30){
    for(int i = 440; i < 1000; i+=20){
      tone(14, i, 20);
    }
  }
  display.clearDisplay();
}
```

auf den DHT22, woraufhin zwei *Schieberegler* erscheinen 11. Das ist praktisch, denn so kann ich direkt sehen, ob meine Programmierung richtig funktioniert. Auch andere Komponenten lassen sich mit der Maus oder der Tastatur interaktiv steuern. Man kann z.B. Wägezellen drücken, Bewegungsmelder auslösen oder die Nähe eines Objekts vor einem Ultraschall-Sensor simulieren. Verwendet ihr Taster in einem Projekt, könnt ihr diese sogar mit einzelnen Tasten auf der Tastatur verknüpfen.

Sobald der Buzzer einen Ton abspielt, erscheint im seriellen Monitor unterhalb des Diagramm-Editors die Fehlermeldung, dass **LEDC (Led control)** nicht initialisiert wurde. Normalerweise erzeugt der ESP32 darüber PWM-Signale, etwa für das Ansteuern eines Buzzers. Da im Sketch aber stattdessen die Funktion `tone()` verwendet wird, könnt ihr diese Fehlermeldung ignorieren.

Teilen und herunterladen

Wenn man sein virtuelles Projekt mit anderen teilen möchte, kann man das direkt aus Wokwi heraus tun – sogar ohne Account. Dazu klickt man über dem Code-Editor auf die Schaltfläche *Share*, woraufhin sich ein Fenster mit dem zugehörigen Link öffnet. Schickt man diesen an eine andere Person, kann auch sie das Projekt (als Kopie) ohne weitere Barrieren

betrachten und bearbeiten. In einer Zeit, in der man sich für jeden Dienst registrieren muss, ist das erfrischend nutzerfreundlich. Projekte, die innerhalb von 6 Monaten nicht aufgerufen wurden und mit keinem Wokwi-Account verknüpft sind, werden nach Ablauf der Frist allerdings gelöscht. Wer seine Projekte also länger behalten und von überall auf sie zugreifen können will, sollte sich einen kostenfreien Wokwi-Account erstellen.

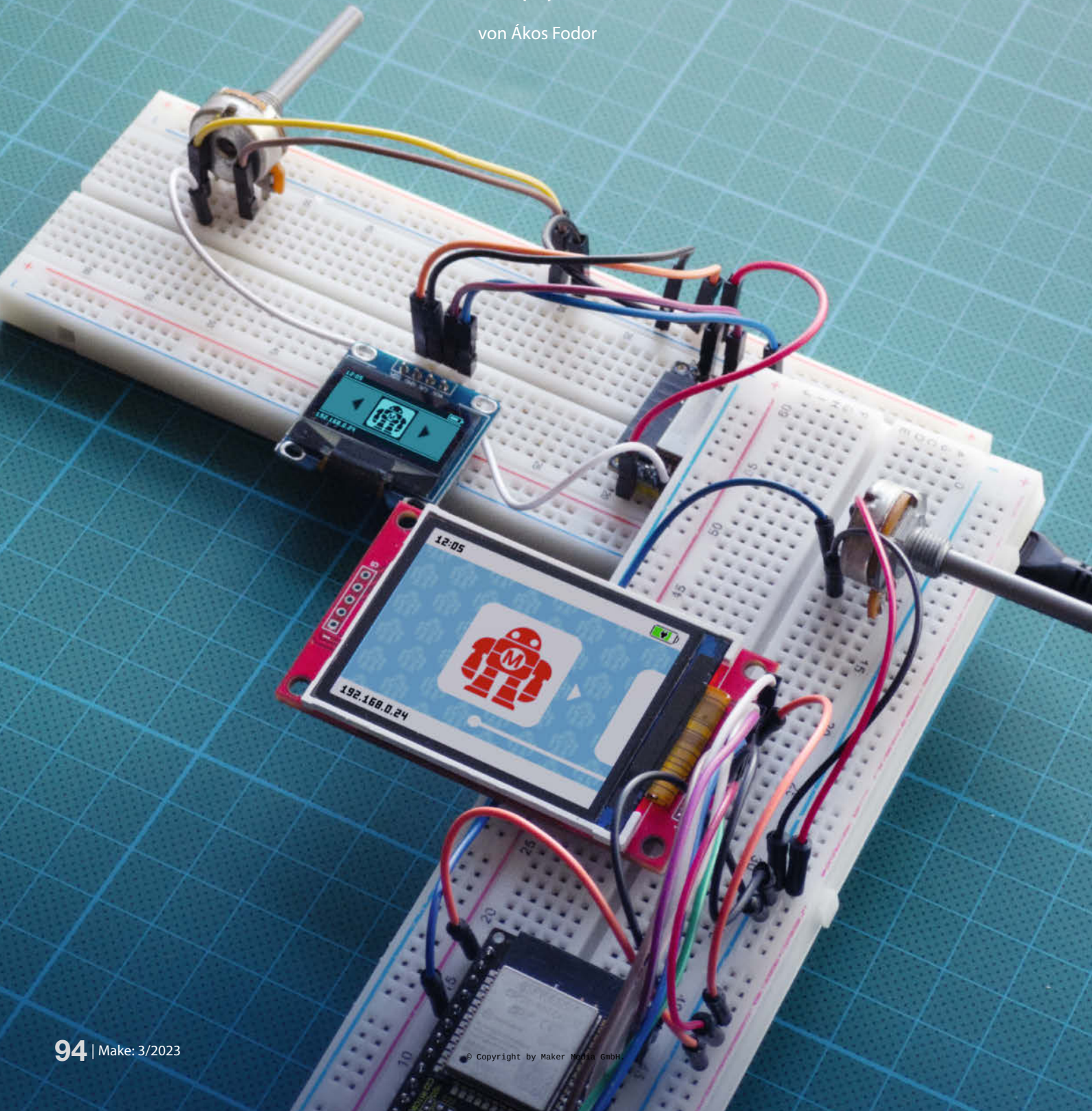
Da ich meinen Aufbau erfolgreich simuliert habe, fehlt eigentlich nur noch, den Sketch auf der echten Hardware zu testen. Dazu lade ich mein Projekt als *Zip*-Datei herunter. Dieses Paket enthält den Programmcode, die *diagramm.json*, die *libraries.txt* sowie eine Textdatei namens *wokwi-project.txt*, in der sich ein Link zu dem Projekt befindet. Den heruntergeladenen Sketch öffne ich schließlich in der Arduino IDE, lade gegebenenfalls noch fehlende Bibliotheken herunter und übertrage ihn danach auf meinen ESP32.

Zu meiner Freude verhält sich der Aufbau auf dem Breadboard genauso wie im Simulator. Hätte ich aber z.B. LEDs anstatt des Buzzers (hier mit 1,5mA) verwendet, um sie bei einer Temperaturüberschreitung blinken zu lassen, hätte ich womöglich meinen Mikrocontroller beschädigt, ehe ich Vorwiderstand sagen kann. Im echten Aufbau sollte man also vorsichtshalber zweimal hinschauen. —akf

Adafruit GFX Library, Teil 2: Bitmaps erstellen und animieren

Wo Kreise und Dreiecke an ihre Grenzen stoßen, kommen die Bitmap-Funktionen der Adafruit GFX Library ins Spiel. Mit ihnen lassen sich selbst komplexeste Pixel-Grafiken auf ein Display zaubern.

von Ákos Fodor



Mit einfachen geometrischen Formen und Text lässt sich auf einem Display schon eine Menge gestalten, wie im vorigen Heft gezeigt. Aber es gibt auch Grafiken, die man lieber nicht zeilen- und pixelweise im Code erstellen sollte, sondern doch eher in einem Bildbearbeitungs-Programm, z.B. Symbole, Logos oder QR-Codes. Das geht nämlich nicht nur schneller, sondern ist einfacher und übersichtlicher, um nicht zu sagen, sinnvoller. Dass man dafür lediglich ein paar Grundfunktionen kennen und kein großer Grafik-Experte sein muss, zeige ich in diesem Artikel. Mit Icons als Beispielgrafiken führe ich euch von der Programmierung über praktische Beispiele mit Photopea bis hin zum Icon, das auf dem Bildschirm tanzt. Die begleitenden Sketches und Beispieldateien findet ihr, wie gewohnt, im GitHub-Repository des Projekts (siehe Link in der Kurzinfor).

Malen mit vielen Zahlen

Bitmaps sind Grafiken, die aus Arrays bestehen, in denen die Beschaffenheit der einzelnen Pixel festgelegt ist. Die *Adafruit GFX Library* kann diese monochrom (1 Bit) und in RGB (16 Bit pro Pixel) darstellen. Allerdings werden die Bitmaps nicht als Bilddateien von einem Datenträger gelesen, sondern direkt als *Byte-Arrays* in den Programmcode integriert. Beim Ausführen des Programms werden die darin enthaltenen Informationen pixel- und zeilenweise wieder zusammengesetzt.

Für monochrome Bitmaps verwendet die Bibliothek Arrays mit den kleinstmöglichen Integer: `uint8_t` bzw. `unsigned char`. Mit ihren 8 Bit wirken sie auf den ersten Blick etwas überdimensioniert, da sie Werte im Bereich von 0 bis 255 speichern können (die Null mitgezählt) und wir eigentlich nur zwischen Schwarz und Weiß bzw. 0 und 1 wechseln müssen. Betrachten wir einen 8-Bit-Integer jedoch nicht dezimal, sondern binär, sehen wir 8 Bits nebeneinander, die jeweils auf 0 oder 1 stehen können. Um Ressourcen zu sparen, fasst die *Adafruit GFX Library* daher 8 horizontale Pixel in einem Byte zusammen **1**. Das Array für ein Bitmap mit 8 × 8 Pixeln besteht also nur aus 8 Integern und nicht 64. Daher ist es sinnvoll, Grafiken zu verwenden, deren Breite sich durch 8 teilen lässt. Denn bei einem Bitmap, das 9 Pixel breit ist, benötigt das Array insgesamt zwei 8-Bit-Integer, um das neunte Pixel darzustellen. Dadurch werden aber auch 7 Bits verschwendet, weil sie ungenutzt sind.

Ein 8-Bit-Array muss im Sketch innerhalb des globalen Bereiches deklariert werden, wo auch die übrigen Variablen stehen. Das sieht in etwa so aus:

```
const uint8_t name [] PROGMEM = {
    0x00, ... };
```

Der Modifikator `PROGMEM` stellt sicher, dass das Array als Konstante im Flash-Speicher (oder

Kurzinfor

- » Komplexe Grafiken mit der *Adafruit GFX Library* darstellen
- » Pixel-Grafiken in *Photopea* bearbeiten
- » Bitmaps in *Byte-Arrays* umwandeln und animieren

Checkliste



Zeitaufwand:
2 Stunden



Kosten:
0 Euro

Material

» ESP32 und Displays aus dem ersten Artikel (Make 2/23)

Mehr zum Thema

- » Ákos Fodor, *Adafruit GFX Library*: Malen mit Zahlen, Make 2/23, S. 64
- » Florian Sommer, *Digitaler Bilderrahmen* mit ePaper, Make 2/22, S. 12
- » Matthias Helneder, *TFT-Displays* im Stil von Nixie-Röhren, Make 5/20, S. 46

Alles zum Artikel
im Web unter
make-magazin.de/x1vh



auch Program Memory, daher das Kürzel) bleibt und nicht beim Programmstart in den Arbeitsspeicher geladen wird. Diese Besonderheit kommt ursprünglich daher, dass die Bibliothek für Arduino-Boards entwickelt wurde, deren Arbeitsspeicher um ein vielfaches kleiner ist als ihr Flash-Speicher.

Bitmaps im *RGB565*-Format, also mit 16-Bit-Farbtiefe, werden von der *Adafruit GFX Library* mit Arrays aus 16-Bit-Integern dargestellt: `uint16_t` oder `unsigned short`. Im Vergleich zu den 8-Bit-Arrays erhält hier jedes Pixel einen eigenen Integer, dessen Wert zudem zwischen 0 und 65535 liegen kann (die Null mitgezählt). Das verbraucht dementsprechend mehr Platz, hat aber auch den Vorteil, dass man nicht auf die Breite seiner Bitmaps achten muss, weil die Arrays nicht zeilen-, sondern pixelweise abgearbeitet werden. Ein 16-Bit-Array wird im Sketch folgendermaßen deklariert:

```
const uint16_t name [] PROGMEM = {
    0xa476, ... };
```

Bitmaps anzeigen

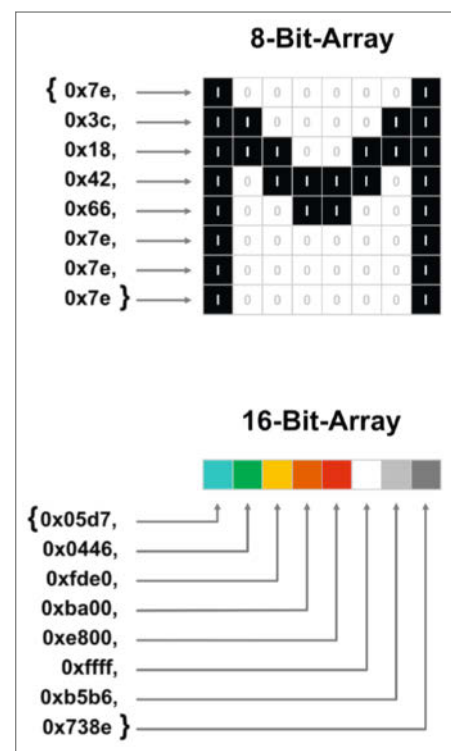
Eingebettete *Byte-Arrays* kann die *Adafruit GFX Library*, je nach Farbtiefe, mit zwei verschiedenen Befehlen darstellen. Monochrome Bitmaps werden mit

```
drawBitmap(x,y,n,w,h,c);
```

auf dem Display angezeigt. Denkt daran, dass ihr das Display-Objekt vor den Befehl setzen müsst, also `display.drawBitmap()`. Gebt in der Klammer als Parameter zunächst die X- und Y-Koordinaten der linken oberen Ecke der Grafik an, danach den Namen des *Byte-Arrays* (`n`), dessen Breite (`w`), Höhe (`h`) und die darzustellende Farbe (`c`) – bei einem monochromen OLED-Display z.B. die 1 für Weiß und bei dem

ILI9341-Farbdisplay eine beliebige Farbe aus dem *RGB565*-System (siehe Tabelle im ersten Artikel), z.B. `ILI9341_RED`.

Eine weitere Besonderheit monochromer Bitmaps ist ihre Transparenz. Wie auch beim Darstellen von Text könnt ihr den Hintergrund von 1-Bit-Grafiken auf undurchsichtig stellen, indem ihr eine Hintergrundfarbe definiert. Fügt dafür in der Klammer einen weiteren Wert als Hintergrundfarbe ein:



1 8-Bit-Arrays fassen 8 Pixel in einem Integer zusammen. In 16-Bit-Arrays erhalten hingegen alle Pixel ein eigenen Integer für ihren Farbwert.

Speicherbedarf von Bitmaps

Die Farbtiefe und Größe eines Bitmaps beeinflussen den gestalterischen Spielraum und die Hardware, die man verwenden sollte, denn jeder Pixelwert in einem Array verbraucht Speicherplatz. Im Vorfeld grob zu überschlagen, wieviel Platz man in etwa benötigen wird, hilft in der Planungsphase.

Um den Speicherbedarf eines Bitmaps mit 32×32 Pixeln zu ermitteln, multipliziert man die Breite, die Höhe und die Farbtiefe (in Bits) miteinander. Zum Schluss teilt man das Ergebnis durch 8, um es in Bytes umzurechnen. Da die Breite mit 32 Pixeln durch 8 teilbar ist, sieht die Gleichung für ein monochromes Bitmap wie folgt aus:

$$(32 \cdot 32 \cdot 1)/8 = 128 \text{ Bytes}$$

Hätte die Grafik eine Breite von 35 Pixeln, müsste man mit dem nächst höheren

Vielfachen von 8 rechnen, nämlich 40. Verwendet man ein Bitmap mit denselben Dimensionen, aber mit 16-Bit-Farbtiefe, berechnet man:

$$(32 \cdot 32 \cdot 16)/8 = 2048 \text{ Bytes}$$

Das RGB-Bitmap ist also 16-mal so groß wie das monochrome. Wenn man viele Grafiken für sein Projekt plant oder über Animationen nachdenkt, kann der Speicherbedarf schnell in die Höhe gehen. Mit 4MB bietet der ESP32 erst einmal genug Raum, um sich grafisch etwas auszutoben, ohne gleich optimieren zu müssen. Ein Arduino Uno Rev3 hat im Vergleich jedoch nur 32kB Flash-Speicher. Das entspricht dem Speicherbedarf eines Bitmaps mit 64×64 Pixeln bei 16-Bit-Farbtiefe. Also eignet sich dieses Board eher für Projekte mit monochromen Displays.

```
drawBitmap(x,y,n,w,h,c,bgr);
```

Diese Option ist praktisch, wenn man Grafiken austauschen will. Denn normalerweise müsste man die zu ersetzende Grafik mit einem gefüllten Rechteck überzeichnen, um sie zu löschen. Von dem entfallenden Extraschritt profitieren besonders Animationen mit ihren vielen Einzelbildern.

Bitmaps mit 16 Bit benötigen keine Farbparameter, da diese im Array für jedes Pixel festgelegt sind. Hier lautet der Befehl:

```
drawRGBBitmap(x,y,n,w,h);
```

Diese Bitmaps sind standardmäßig nicht transparent, da in RGB565 kein Alpha-Kanal enthalten ist. Daher benötigt man eine Maske, die bestimmt, was aus dem Bitmap weggeschnitten werden soll – als würde man eine Schablone darüber legen. Auch hier wird der `draw()`-Befehl um einen weiteren Parameter ergänzt:

```
drawRGBBitmap(x,y,n,m,w,h);
```

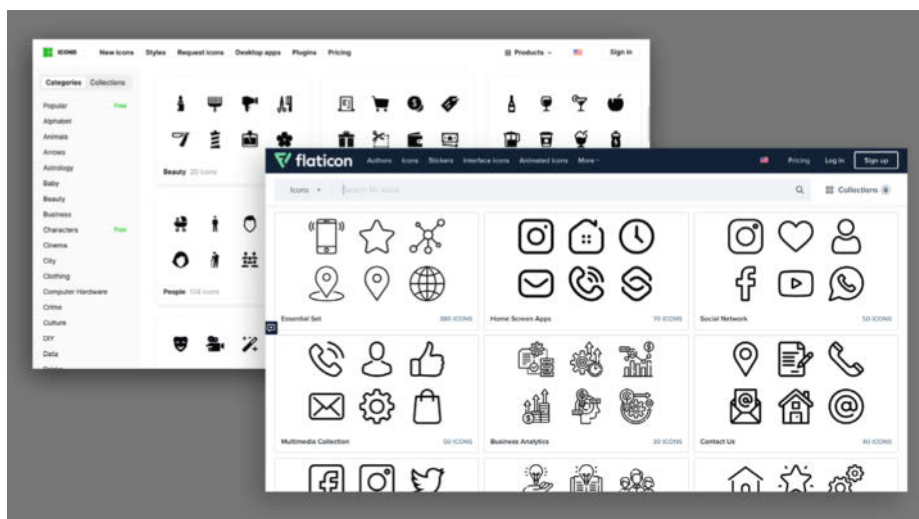
Als Maske (m) erwartet die Bibliothek ein monochromes 8-Bit-Array, in dem die transparenten Bereiche mit schwarzen Pixeln gekennzeichnet sein müssen. Wie das in der Praxis funktioniert, zeige ich euch in einem der folgenden Abschnitte.

Grafiken finden oder erfinden

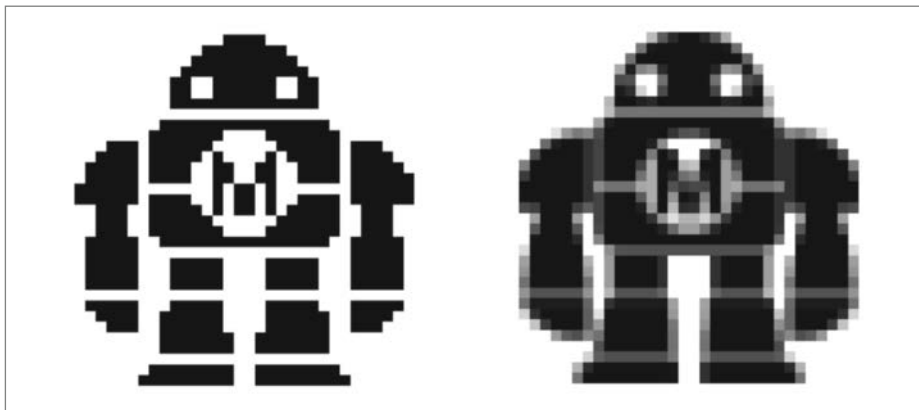
Soviel zur Theorie. Jetzt ist es an der Zeit, das Ganze einmal auszuprobieren. Dafür benötigen wir ein paar passende Grafiken. Als Übungsobjekte eignen sich Icons sehr gut, weil sie kompakt und schnell erstellt, bzw. zahlreich im Internet verfügbar sind, z.B. auf Plattformen wie icons8.com oder flaticon.com **2**. Wie sie von dort auf das SSD1306- und ILI9341-Display kommen, zeige ich euch jetzt. Natürlich könnt ihr mit den Werkzeugen, die wir verwenden werden, auch eure eigenen Entwürfe umsetzen.

1-Bit-Icons erstellen

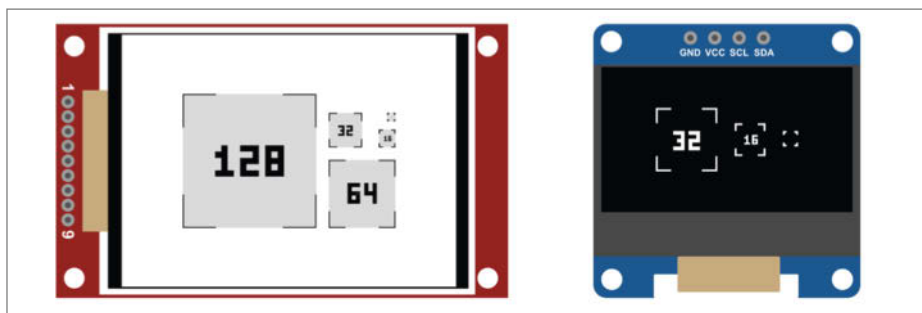
Wer Vorlagen für monochrome Bitmaps sucht, sollte auf zwei Dinge achten: dass sie pixelgenau gezeichnet sind **3** und bereits die Größe haben, die man auf dem Display abbilden will **4**. Denn das Konvertieren von Vektor-Icons in Pixelgrafiken sowie das Skalieren derselben erzeugt in der Regel eine Kantenglättung (Anti-Aliasing), die sich in halbttransparenten Pixeln äußert, die monochrome Bitmaps nicht abbilden können. Besonders bei Rundungen und Schrägen schleichen sich solche weichen Kanten gerne ein. Da man online allerdings selten pixelgenaue Icons oder solche in der richtigen Größe findet, muss man bei den meisten ein wenig nacharbeiten. Dafür bietet sich das browserbasierte *Photopea* an. Mit dem kostenlosen Bildbearbeitungs-Programm kann man Gra-



2 Auf Icon-Plattformen findet man eine Menge Vorlagen.



3 Das pixelgenaue Bild (links) ist viel schärfer als die skalierte Version (rechts)

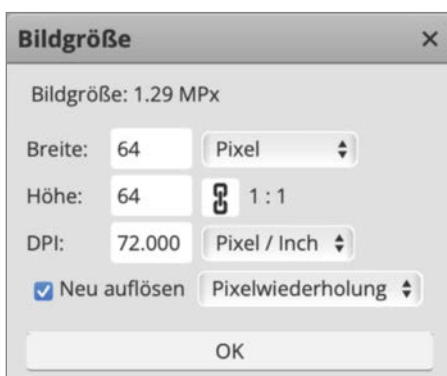


4 Die optimale Icon-Größe variiert je nach Bildschirmauflösung.

fiken nämlich im Nu entglätten und in eine monochrome Version verwandeln.

Ladet euch dazu ein schwarz-weißes Icon eurer Wahl im PNG-Format herunter, dessen Breite nach Möglichkeit durch 8 teilbar sein sollte. Öffnet danach photopea.com und erstellt ein neues Projekt, bei dem ihr die Breite und Höhe auf die Maße des Icons einstellt 5. Drückt auf der Tastatur *Strg+0*, um in das Dokument zu zoomen (*CMD+0* auf dem Mac) und zieht danach die heruntergeladene Icon-Datei auf die leere weiße Arbeitsfläche. Sollte die Grafik ungenutzte Pixel an den Rändern haben, beschneidet das Icon zunächst mit dem *Freistellungswerkzeug* 6, das ihr in der *Werkzeugleiste* am linken Rand findet. Zieht dazu den Rahmen so klein, dass er genau mit der Grafik abschließt und bestätigt mit der *Eingabetaste*. Skaliert danach das Bild auf die gewünschte Pixelanzahl, indem ihr im Menü *Bild* auf *Bildgröße* klickt 7. Achtet darauf, unter *Neu auflösen* den Punkt *Pixelwiederholung* auszuwählen, weil Photopea dadurch beim Skalieren auf Anti-Aliasing verzichtet. Wählt anschließend im Menü *Ebene* den Punkt *Neue Einstellungsebene* und dort *Schwellenwert* aus. Diese Funktion wandelt das geglättete Icon in eine monochrome Grafik um. Ihr könnt mithilfe eines Reglers das Verhältnis von schwarzen zu weißen Pixeln steuern.

Manchmal funktioniert das ziemlich gut. Hin und wieder muss man jedoch manuell mit dem *Bleistift* nachhelfen, z.B. bei sehr kleinen



7 Mit der Option *Pixelwiederholung* werden Grafiken ohne Anti-Aliasing skaliert.

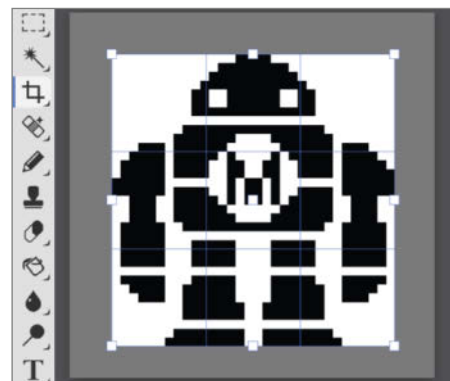
Icons oder runden Elementen. Bewegt die Maus in der *Werkzeugleiste* links auf das *Pinsel-Symbol* und haltet die *Maustaste* gedrückt. Daraufhin erscheint ein Menü, in dem ihr den *Bleistift* auswählen könnt 8. Dieser ist ideal, um pixelgenau zu malen, denn die *Bleistiftspitze* hat im Vergleich zum Pinsel eine harte Kante, d.h. sie erzeugt kein Anti-Aliasing.

Ganz unten in der *Werkzeugleiste* befinden sich zwei sich überlappende Farbfelder, die anzeigen, welche Farbe als Vorder- bzw. Hintergrundfarbe ausgewählt ist. Mit der *x-Taste* auf der Tastatur könnt ihr zwischen beiden hin und her wechseln. Ihr müsst also beim Malen nichts radieren, sondern übermalt verpatzte Stellen einfach mit der anderen Farbe. Oberhalb der *Werkzeugleiste* wird das aktuelle Werkzeug angezeigt (jetzt der *Bleistift*). Mit einem Klick auf das Symbol rechts daneben könnt ihr die Größe der *Bleistiftspitze* einstellen.

Wählt anschließend in dem *Ebenen-Fenster* rechts unten die Ebene eures importierten Icons aus und passt die Grafik nach euren Vorstellungen an. Sobald ihr fertig seid, könnt ihr die Grafik über das Menü *Datei* als PNG-Datei



5 Am besten erstellt man seine Grafiken im gewünschten Endformat.

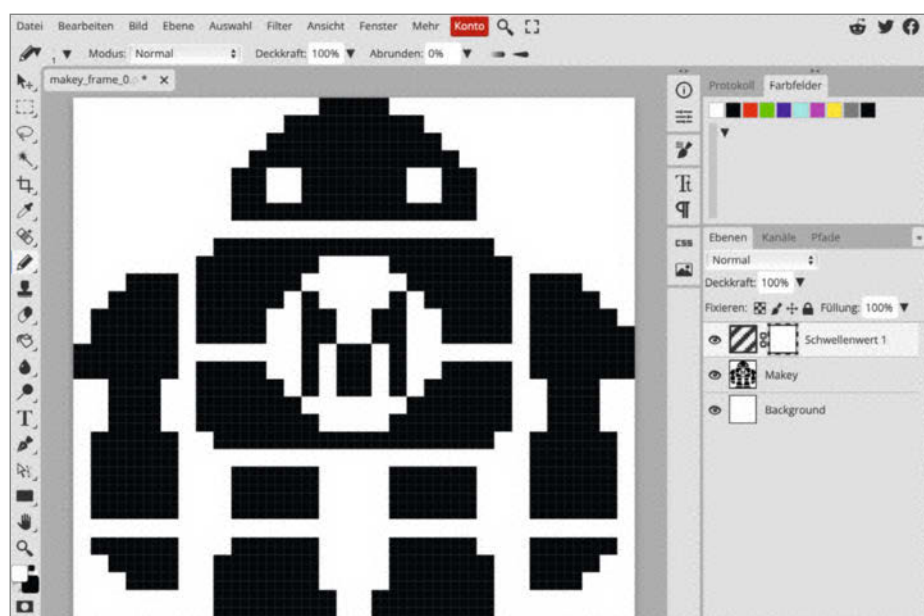


6 Um keine Pixel zu verschenken, sollte man Ränder abschneiden.

exportieren und seid bereit für den nächsten Schritt.

Monochrome Byte-Arrays

Auch zum Konvertieren einfarbiger Bitmaps gibt es ein kostenfreies Web-Tool: *image2cpp* (Link in der Kurzinfor). Ladet dort im ersten



8 Der Arbeitsbereich von Photopea: Links die *Werkzeugleiste* mit ausgewähltem *Bleistift*, rechts unten das *Ebenenfenster* und oben das Menü.

4. Output

Code output format

Arduino code, single bitmap

Adds some extra Arduino code around the output for easy copy-paste. generates a single byte array.

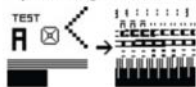
Identifier/Prefix:

epd_bitmap_

Draw mode:

Horizontal - 1 bit per pixel

If your image looks all messed up on your display, like the image below, try using a different mode.



Generate code

Copy Output

Download as binary file (.bin)

```
const unsigned char NaN [] PROGMEM = {
  // *Makey_mono_complete, 40x45px
  0x00, 0x78, 0x00, 0x00, 0x00, 0xfe, 0x00, 0x00, 0x00, 0x01, 0x9f, 0x00, 0x00, 0x00, 0x00,
  0x0f, 0x00, 0x00, 0x00, 0x0e, 0x07, 0x0e, 0x00, 0x00, 0x3f, 0x87, 0x3f, 0x00, 0x00, 0x7f, 0xc7,
  0x7c, 0x00, 0x00, 0x41, 0xe6, 0x70, 0x00, 0x80, 0x00, 0x76, 0xe0, 0x00, 0x80, 0x00, 0x30, 0x00,
  0x12, 0x24, 0x0f, 0x15, 0xf0, 0x08, 0x88, 0x3f, 0xc4, 0xf8, 0x03, 0xe0, 0x79, 0xdc, 0x7c, 0x17,
  0x14, 0x70, 0x19, 0x0c, 0x07, 0xf0, 0xc0, 0x03, 0xc6, 0x6f, 0xfb, 0x80, 0x39, 0xc6, 0x07, 0xf0,
  0x00, 0x31, 0xe0, 0x17, 0xf4, 0x00, 0x00, 0xe0, 0x03, 0xe0, 0x00, 0x70, 0xc0, 0x08, 0x88, 0x00,
  0xe0, 0xc0, 0x12, 0x24, 0x00, 0xc0, 0x80, 0x00, 0x80, 0x00, 0x00, 0x00, 0x80, 0x01, 0xc0,
  0x00, 0x00, 0x00, 0x03, 0x9e, 0x00, 0x00, 0x00, 0x03, 0x3f, 0x00, 0x00, 0x00, 0x03, 0x7f, 0x82,
  0x00, 0x00, 0x00, 0x5e, 0x87, 0x00, 0x00, 0x7f, 0x7f, 0x8f, 0x00, 0x00, 0x40, 0x00, 0x3c, 0x00
```

9 Mit den richtigen Ausgabe-Einstellungen erhält man in `image2cpp` ein passendes Byte-Array für Arduino-Sketches.

Abschnitt eure monochrome Grafik hoch. Bei *Image Settings* könnt ihr fast alle Punkte überspringen, außer ihr verwendet ein SSD1306-Display. Dann ist es sinnvoll, unter *Invert image color* die Farben des Bildes umzukehren, damit der Bildschirm das Icon weiß darstellt und nicht den Hintergrund. Weiter unten im dritten Abschnitt zeigt euch ein kleines Vorschau-Bild, was ihr in etwa erwarten könnt. Jetzt müsst ihr nur noch das Ausgabeformat (*code output format*) einstellen, das die Form des Arrays bestimmt. Wählt dazu im ersten Drop-Down-Menü den Punkt *Arduino Code - Single Bitmap* aus und als Zeichensmethode (*draw method*) die Option *Horizontal - 1 bit per pixel*. Mit einem Klick auf *Generate* erzeugt die Website anschließend das passende Byte-Array für die Adafruit GFX Library 9.

Öffnet jetzt den Sketch *bitmapSSD1306.ino* oder *bitmapILI9341.ino*, je nachdem, welchen Bildschirm ihr benutzt und kopiert das erstellte Byte-Array in den globalen Bereich. Benennt anschließend das Array von *NaN* in einen Namen eurer Wahl um, z.B. *IconTest*. Im `void start()` habe ich bereits einen `drawBitmap()`-Befehl eingefügt. Ändert in der Klammer den Platzhalter *name*, ergänzt die Breite und Höhe und passt die Koordinaten sowie die Farbe an. Ladet danach den Sketch auf euren ESP32 und freut euch über euer erstes Bitmap.

16-Bit-Icons erstellen

Möchtet ihr 16-Bit-Grafiken auf einem Farbdisplay verwenden, habt ihr wesentlich mehr Freiheiten in der Gestaltung als bei mono-

chromen Bitmaps. Ihr müsst nämlich weder beachten, ob die Grafiken pixelgenau sind, noch ob ihre Breite durch 8 teilbar ist. Also eignen sich die meisten Grafiken schon beim Herunterladen für eine 16-Bit-Darstellung und ihr könnt prinzipiell direkt zum Konvertieren in ein Byte-Array übergehen. Es gibt allerdings einen Sonderfall, für den noch eine Extrarunde gedreht werden muss: nämlich dann, wenn ihr den Hintergrund eines RGB-Bitmaps maskieren bzw. transparent haben wollt. Dafür benötigt ihr ein separates monochromes Byte-Array.

Erstellt in diesem Fall ein neues Projekt in Photopea, wie in dem vorherigen Abschnitt beschrieben und passt die Breite auf die Icon-Größe bzw. das nächstgrößere Vielfache von 8 an – bei einem Icon mit 42 Pixeln Breite z.B. auf 48. Zieht anschließend das heruntergeladene RGB-Icon auf die leere Arbeitsfläche und fügt eine neue Ebene hinzu. Malt auf dieser mithilfe des Bleistifts alles mit Weiß aus, was sichtbar bleiben soll und den transparenten Bereich schwarz 10. Ist die Maske fertig, exportiert sie anschließend als PNG und blendet danach die Masken-Ebene aus, indem ihr in dem Ebenen-Fenster auf das kleine Auge links neben der Ebenen-Bezeichnung klickt. Dadurch wird die darunter liegende Ebene bzw. das farbige Icon sichtbar. Exportiert die Grafik erneut als PNG, denn dadurch ist sichergestellt, dass beide Byte-Arrays später exakt übereinander liegen werden. Jetzt seid ihr bereit, das 16-Bit-Icon in ein Byte-Array zu konvertieren.

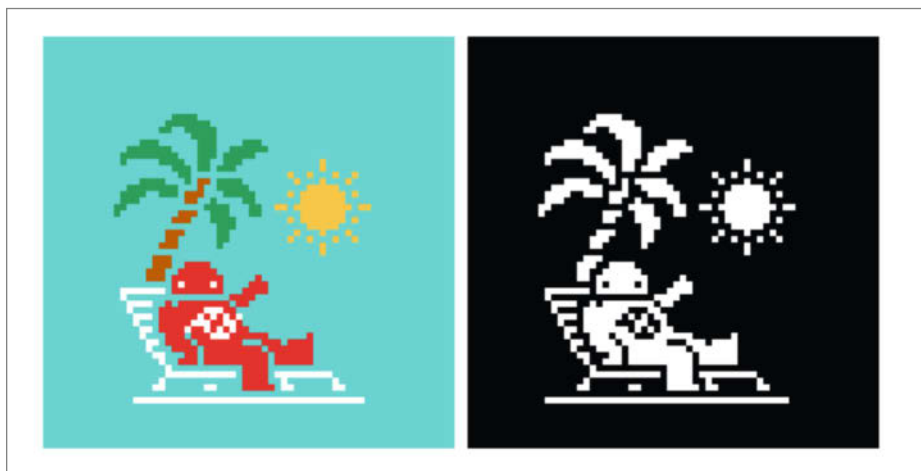
Byte-Arrays mit vielen Farben

16-Bit-Grafiken lassen sich genauso einfach umwandeln wie monochrome. Ihr müsst diesmal aber vor dem Hochladen der Grafik das Ausgabeformat in *Horizontal - 2 bytes per pixel (565)* ändern. Klickt auf *Generate* und kopiert das erstellte Byte-Array. Öffnet in der Arduino IDE anschließend den Sketch *bitmapILI9341.ino* und fügt das Byte-Array in den globalen Bereich ein. Ändert den Namen *NaN*, fügt ihn im Befehl `drawRGBBitmap()` ein, der im `void start()` steht, und ergänzt anschließend noch die Breite und Höhe des Arrays.

Wollt ihr eine Maske verwenden, wandelt diese mit *image2cpp*, in ein 8-Bit-Array um, wie im Abschnitt *Monochrome Byte-Arrays* oben beschrieben und kopiert es ebenfalls in den globalen Bereich eures Sketches. Fügt nach dem Umbenennen den Namen eures Masken-Arrays hinter den Array-Namen des Bitmaps in den `draw()`-Befehl ein. Dieser könnte dann beispielsweise so aussehen:

```
drawRGBBitmap(10,10,icon,mask,32,32);
```

Ladet den Sketch danach auf den Mikrocontroller.



10 Mit einer monochromen Maske lässt sich der Hintergrund ausschneiden.

Animationen erstellen

Auf den Icon-Plattformen tummeln sich auch viele animierte Varianten im GIF-Format. Ähnlich wie bei den statischen Icons sind diese in der Regel ebenfalls nicht pixelgenau. Für Photopea sind GIFs aber kein Problem. Es zerteilt die Animationen sogar in Einzelbilder, sodass ihr auch ein paar löschen könnt, falls die Animation zu lang sein sollte. Im Folgenden zeige ich euch die Schritte, die ihr zum Erstellen monochromer Animationen benötigt.

Ladet euch zunächst ein animiertes schwarz-weißes Icon herunter. Ihr werdet dabei feststellen, dass sich die Größe nicht wie bei PNG-Dateien auswählen lässt und FlatIcon z.B. gerne Animationen mit 640×640 Pixeln herunterlädt. Die sind dann zwar gestochen scharf, aber selbst für den ESP32 etwas zu groß (ca. 820 kB pro Einzelbild), sodass wir als Erstes skalieren müssen.

Öffnet als Nächstes Photopea und öffnet damit das heruntergeladene GIF. Beschneidet die Grafik mit dem *Freistellungswerkzeug* und geht anschließend in das Menü *Bild* und wählt dort *Bildgröße* aus. Ändert die Breite auf die gewünschte Ausgabegröße auf ein Vielfaches von 8. Im vorherigen Abschnitt habt ihr bereits gelernt, wie man die Schwellenwert-Anpassung über eine Einstellungsebene vornimmt. Dieses Mal müsst ihr den Effekt auf alle Animations-Frames anwenden, bevor ihr sie exportiert. Wählt dazu alle Ebenen (bis auf den Ordner) im Ebenenfenster aus, klickt im Menü *Bild* auf *Anpassungen* und dann auf *Schwellenwert*. Verstellt den Regler, bis das Ergebnis euch zusagt und bestätigt mit *OK*. Überprüft danach die einzelnen Frames, indem ihr sie im Ebenenfenster ein- und ausblendet. Es kann durchaus passieren, dass derselbe Schwellenwert nicht bei allen Frames zum gewünschten Ergebnis führt. In diesem Fall müsst ihr mit dem Bleistift manuell nachhelfen.

Seid ihr mit dem Ergebnis zufrieden, entfernt als Nächstes das Animationskürzel *_a_* vor den einzelnen Ebenenbezeichnungen **11**, damit Photopea sie beim Exportieren nicht mehr als Animation zusammensetzt. Wählt dazu erneut alle Ebenen aus und klickt im Menü *Ebene* unter *Animation* auf *Unmake Frames*. Daraufhin verschwinden die Kürzel. Macht danach einen Doppelklick auf den Ebenen-Ordner und ergänzt vor der Bezeichnung das Kürzel *_e_* mit einem anschließenden Leerzeichen **12**. Mithilfe dieses Kurzbefehls weiß Photopea, dass es alle im Ordner befindlichen Ebenen auf einmal exportieren darf – das spart ordentlich Zeit. Wählt zum Exportieren im Menü *Datei* den Punkt *Ebenen exportieren*, klickt in dem erscheinenden Fenster auf den Reiter *Combinations* und exportiert die Ebenen als PNG. Daraufhin erhaltet ihr eine ZIP-Datei, die alle Einzelbilder beinhaltet.

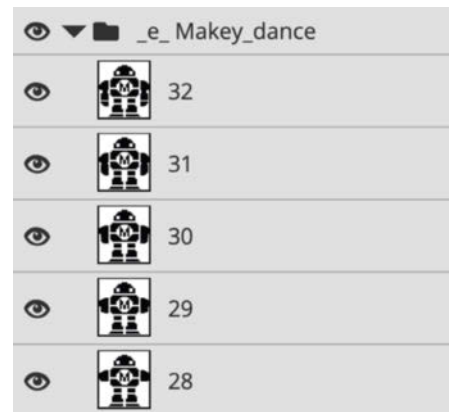


11 Um die Animation in Einzelbilder exportieren zu können, muss das Prefix *_a_* entfernt werden. Dabei hilft die Menü-Funktion *Unmake Frames*.

Mit diesen Bildern könnt ihr anschließend *image2cpp* füttern. Invertiert die Farben unter *Image Settings* und stellt als Ausgabe-Format *Arduino code* ein. Diese Option erstellt für jede Grafik ein eigenes Array, das nach dem Dateinamen benannt wird. Löscht den *Identifier/Prefix*, außer ihr benötigt einen Prefix vor den Array-Namen und klickt anschließend auf *Generate*. Daraufhin erhaltet ihr Byte-Arrays für jeden einzelnen Frame sowie ein weiteres Array, das alle Byte-Arrays zusammenfasst und die Animation erleichtert. Prüft allerdings, ob die Reihenfolge der Frames stimmt, denn diese kommt gerne mal durcheinander. Den berechneten Speicherplatzverbrauch könnt ihr ignorieren, da das Tool pauschal je Array-Zeile 16 Bits berechnet und daher etwas ungenau ist. Öffnet anschließend den Sketch *aniSSD1306.ino* oder *aniILI9341.ino* und kopiert die Arrays in den globalen Bereich.

Byte-Arrays animieren

Sobald der ESP32 eingeschaltet wird, zählt er in Millisekunden die vergangene Zeit mit. Diesen Wert kann man mithilfe der Funktion *millis()* abfragen und dafür nutzen, das Timing von Animationen zu steuern. Dafür vergleicht man die vergangene Zeit (*previousTime*) mit der aktuellen (*currentTime*) und legt fest, wie groß der Abstand sein darf, bis der nächste Frame abgespielt wird – die *Frames per Second* (fps). Mithilfe der Variablen *currentFrame* wird außerdem mitgezählt, welches Byte-Array gerade angezeigt wird. Jedesmal, wenn der Abstand zwischen *currentTime* und *previousTime* größer als fps ist, wird *currentFrame* um 1 hochgezählt und das nächste Bild angezeigt, bis der letzte



12 Damit man nicht jedes Bild einzeln exportieren muss, setzt man von Hand das Prefix *_e_* vor den Namen des Ebenen-Ordners.

Frame erreicht ist und den *currentFrame* auf 0 setzt, sodass die Animation von vorn beginnt. Ersetzt in der Funktion *void animation()* im Befehl *drawBitmap()* den Platzhalter *makey* mit dem Namen eures Animations-Arrays und passt die maximale Anzahl (hier 27) der Einzelbilder an. Überträgt anschließend den Sketch auf den ESP32. Sobald der ESP32 neu startet, habt ihr es geschafft und ihr seht eine Animation, die in Endlosschleife läuft. Damit habt ihr erfolgreich gelernt, wie ihr einfache und komplexe Grafiken für die Adafruit GFX Library erstellen, bearbeiten und auf einem kleinen Display anzeigen könnt.

Geschickt kombiniert

Verwendet ihr ein Farbdisplay, könnt ihr mehrfarbige Kompositionen auch mithilfe einzelner kleiner 1-Bit-Grafiken zusammenstellen und ihnen unterschiedliche Farben zuweisen. Der Makey auf der Liege verbraucht mit dieser Technik bei 40×45 Pixeln nur 277 Bytes – kaum mehr als die 1-Bit-Version des gesamten Motivs (225 Bytes), aber viel weniger als die vergleichbare 16-Bit-Grafik mit 3600 Bytes. —akf

Animation mit millis()

```
uint8_t fps = 30;
unsigned long previousTime = 0;
uint8_t currentFrame = 0;

void animation(){
    unsigned long currentTime = millis();

    if(currentTime - previousTime >= 1000/fps){
        if(currentFrame <= 27){
            display.drawBitmap(10,10,
                               makey[currentFrame],
                               24,30,ILI9341_WHITE,ILI9341_BLACK);
            currentFrame += 1;
            currentTime = previousTime;
        } else {
            currentFrame = 0;
        }
    }
}
```



Bild 1: Das Kontroll-siegel hatte der erste Techniker selbst schon gebrochen und nicht wieder erneuert ...

Glasfaseranschluss

Mit dem Umstieg vom Kupfer auf Lichtwellenleiter kommt neue Technik wie Spleißboxen und Medienkonverter ins Haus. Wir haben in die kleinen Kästchen gleich mal reingeschaut.

von Daniel Bachfeld

Eigentlich wollte ich in dieser Rubrik über ein anderes Gerät schreiben. Weil der Glasfaseranschluss nach der Inbetriebnahme bereits nach einem Tag ausgefallen war, ich schon seit Stunden auf den Techniker warten musste und meine Neugier mich drängte, wagte ich einen Blick in die Kästchen. Im Prinzip ahnte ich, was drin steckte, da ich Mitte der 90er ähnliche Anschlüsse für Firmenkunden projektierte. Damals™ hießen Switches noch Multiport-Bridges und Übertragungstechniken wie ATM waren der neue Star am Netzwerkthimmel.

Das linke Kästchen in Bild 1 ist der sogenannte Hauptübergabepunkt, kurz HÜP, in dem die ins Haus verlegten LWL-Fasern (das orangene Verlegekabel) mit Patchkabeln verbunden werden. Die Fasern sind als Mono Mode bzw. Single Mode ausgeführt. Vereinfacht gesagt breitet sich in dem nur 9µm dünnen

Glaskern nur ein einziger Lichtstrahl (Mode) aus, was sehr hohe Datenraten erlaubt.

An zwei der Fasern werden sogenannte Pigtails gespleißt. Pigtails sind vorkonfektionierte Faserstücke, hier mit einem genormten LP/APC-Stecker an einem Ende. Beim Spleißen werden das offene Ende des Pigtails und das Ende der ankommenden Faser mit einem Spleißgerät per Lichtbogen regelrecht miteinander verschmolzen. Anschließend kommt noch ein Spleißschutz um den Spleiß und das Ganze wird im Spleißkamm befestigt (siehe Bild 2, Mitte rechts). Technisch gesehen ist der HÜP also eine Spleißbox.

Aus Redundanzgründen spleißt der Techniker meist mehrere Fasern, obwohl im Betrieb nur eine erforderlich ist. Der LP/APC-Stecker wird dann in eine Kupplung gesteckt. In die Kupplung wird von der anderen Seite (im Bild 2 von unten) ein LWL-Patchkabel (LP/APC-



Bild 3: Die Basisplatte des GF-TA dient im Prinzip nur der Aufnahme eines Patchkabels.

Stecker) angeschlossen, das zum Glasfaser-Teilnehmeranschluss (GF-TA) führt. In Mehrfamilienhäusern würde vom HÜP zum GF-TA dedizierte Glasfaser im Haus verlegt werden.

An den GF-TA ist üblicherweise ein Glasfasermodem (NT) angeschlossen, das der Umsetzung der optischen Signale auf elektrische Signale dient, konkret Gigabit-Ethernet. Nebenbei wickelt das Modem noch diverse Protokolle mit der Gegenstelle des Providers ab.

In meiner Konfiguration sind GF-TA und NT in einem Gerät der Marke Genexis FiberTwist (2110B) vereint. Das FiberTwist besteht aus einer an der Wand verschraubten Grundplatte (Bild 3), auf der das vom GF-TA kommende Patchkabel mit Reserve aufgewickelt ist und mit seinem Stecker abermals in einer Kupplung endet. Der Deckel (Bild 4, 5 und 6) des Medienwandlers enthält einen integrierten LWL-Stecker, der beim Aufdrehen auf die Unterseite in der Kupplung einrastet und somit die Verbindung schließt. Im Deckel steckt das Glasfasermodem. Das Glasfasermodem zeigt per LED an, ob eine Verbindung besteht (Bild 1, rechte LED, bei mir leider noch aus). —dab

► make-magazin.de/xw5b



Bild 4: Der Deckel des FiberTwist enthält das Glasfasermodem.

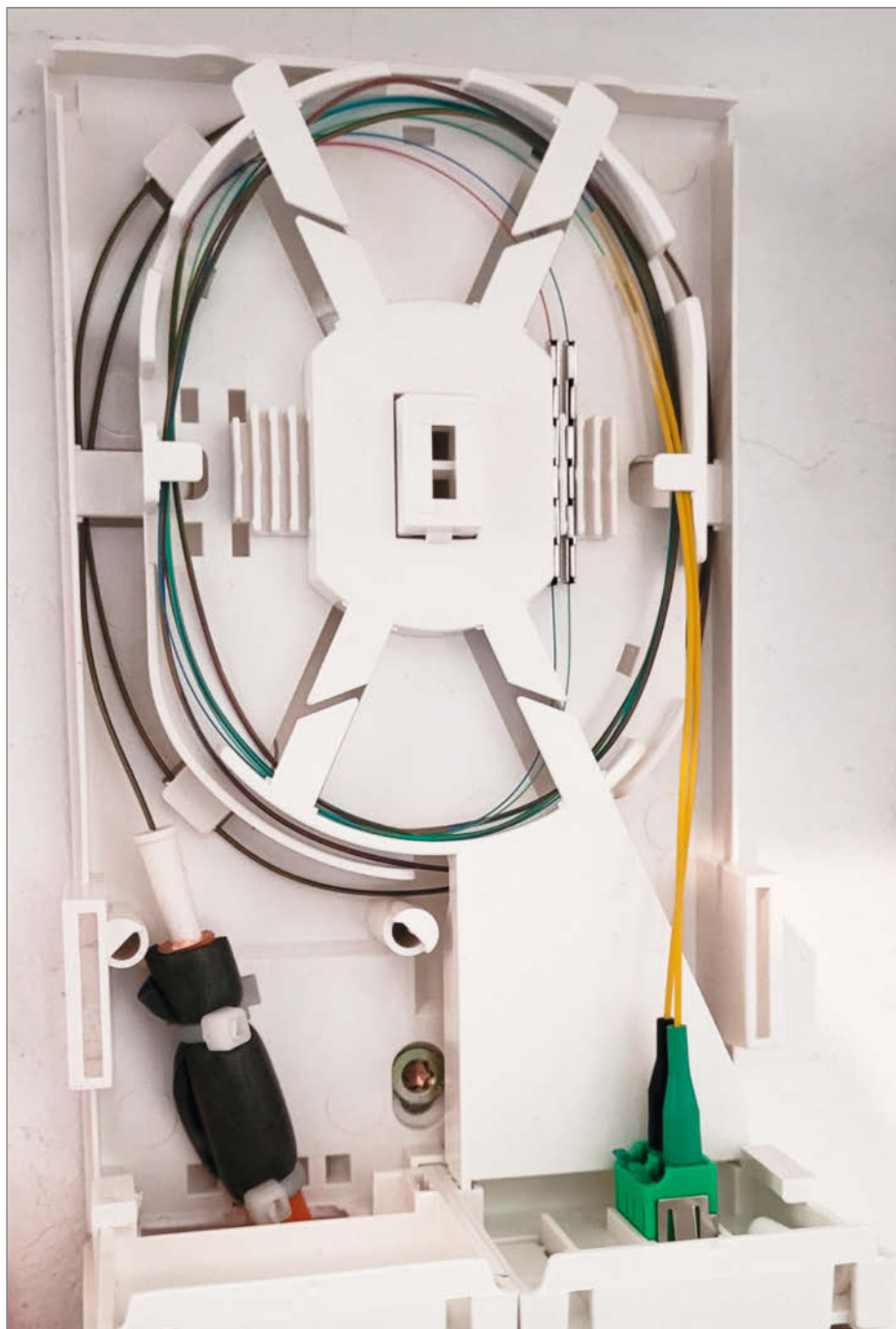


Bild 2: Im HÜP werden die ankommenden Fasern (link) mit Reserve aufgewickelt und mit den gelben Pigtails (rechts) verbunden. Die Stecker der Pigtails (grün) stecken in einer Kupplung.

Bild 5: Den Deckel des FiberTwist muss man von der Grundplatte abdrehen. Beim Zudrehen rastet der integrierte Stecker des Deckels (Bild 6) in die Kupplung der Grundplatte ein.

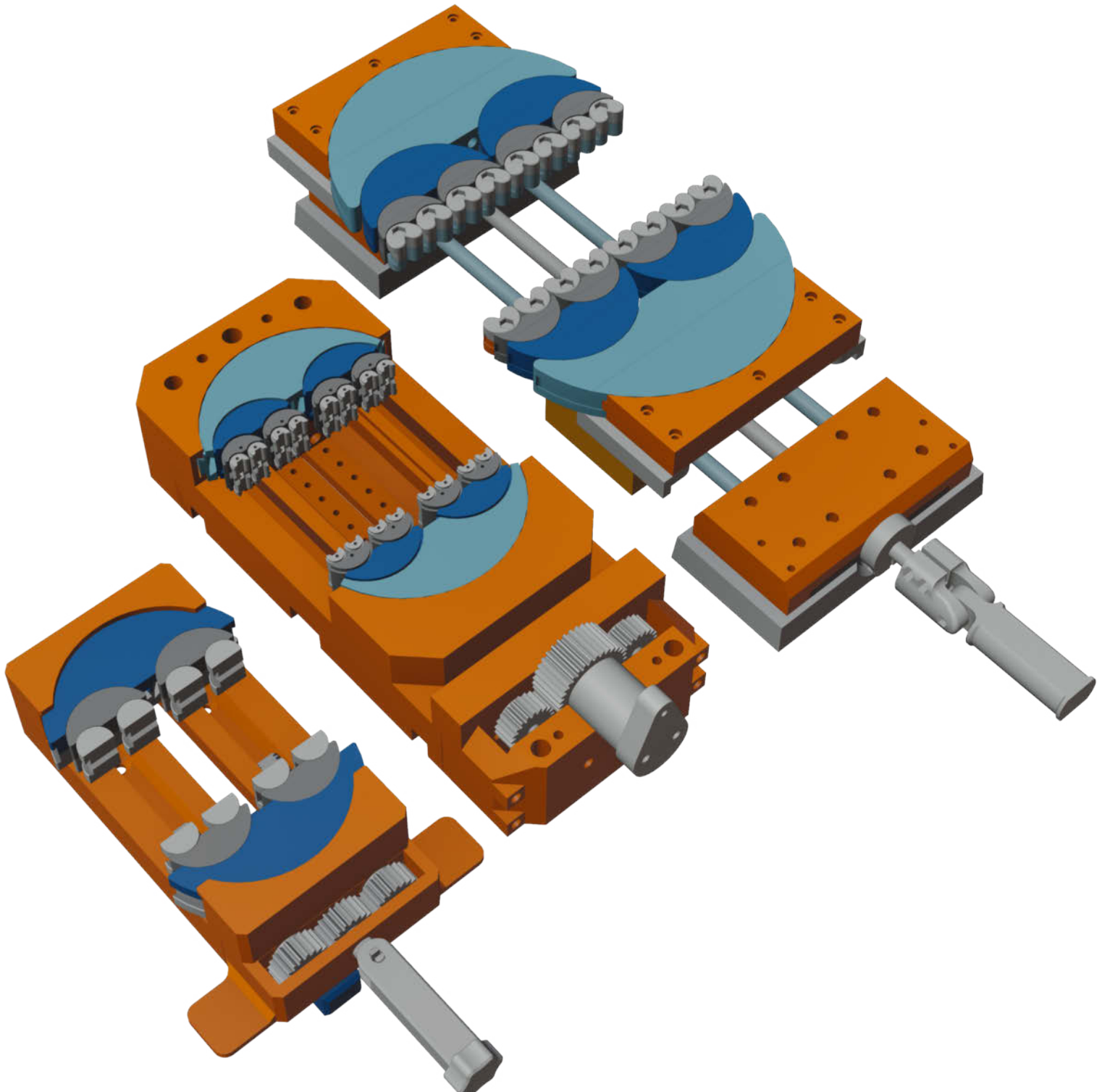


Bild 6: Der integrierte Stecker des Deckels in Großaufnahme. Im Unterschied zu den Steckern der anderen Kabel handelt es sich hier um einen SC/APC-Stecker.

Fraktaler Schraubstock

Unregelmäßig geformte Werkstücke kann man nur schlecht einspannen – außer mit diesem anschmiegsamen Schraubstock nach dem Fraktal-Prinzip, dessen Konzept schon über hundert Jahre alt ist.

von Christopher Borge (Übersetzung: Karin Hirmer)



Bei einem Abstecher zu YouTube fand sich im Juni 2021 unter den Vorschlägen gerne das Video *Rare Antique Fractal Vise [Restoration]* des Users *Hand Tool Rescue* (siehe Link in der Kurzinfor). Der Beitrag zeigt einen einzigartigen Schraubstock, den, obwohl er bereits 1913 patentiert wurde, die meisten von uns noch nie gesehen hatten – und auch nichts Vergleichbares. Das Video ging schnell viral und hat bis heute 20 Millionen Klicks generiert.

Der so genannte fraktale Schraubstock, der als „Vorrichtung zur Herstellung eines engen Kontakts mit Körpern beliebiger Form, und deren Bearbeitung oder Halterung“ beschrieben wird, besteht aus 30 halbkreisförmigen Backenteilen, die mit den Hauptbacken eines Schraubstocks verbunden sind.

Die kleinste Backengruppe ist in der nächstgrößeren eingebettet, und so weiter. Dafür werden sorgfältig gearbeitete Schwalbenschwänze verbaut, die es den Backen ermöglichen, sich in praktisch jede beliebige Anordnung zu drehen, wenn sie um ein Objekt geschlossen werden.

Die User fanden es äußerst fesselnd, dem Mechanismus zuzuschauen und die Anfragen, ob und wo so ein Werkzeug erhältlich ist, häuften sich schnell in den Kommentarspalten.

Dort warfen viele Kommentierende die Idee in den Raum, dass eine 3D-Druck-Version eine Möglichkeit sein könnte, dieses großartige Werkzeug für die breite Masse verfügbar zu machen.

Ein faszinierender Mechanismus

Ein erster Design-Entwurf wurde noch am Veröffentlichungstag auf Thingiverse gepostet, und zwar von Evan DeLosh aus Michigan (Benutzername *Clerick*), dessen *Antique Fractal Vise Replica* sein 3D-Modell der fraktalen Backen zeigt (siehe Link in der Kurzinfor).

Der Beitrag fand einige Beachtung, da er aber nicht zeigte, wie die Backen tatsächlich

Kurzinfor

- » Geniales Prinzip mit moderner Technik neu umgesetzt
- » Drei Varianten zum Nachdrucken im Vergleich: komplett aus dem 3D-Drucker oder mit Metall-Zusatzteilen, feste oder variable Größe

Mehr zum Thema

- » Carsten Meyer, Spannmittel für die CNC-Fräse, *Make* 1/16, S. 102, kostenlos im Volltext online
- » Video: Demonstration der Version 2 und Aufbautipps (englisch)



Alles zum Artikel
im Web unter
make-magazin.de/xnx7



gedruckt wurden und ob sie funktionierten, verlor er schnell wieder an Aufmerksamkeit.

Zusammen mit meiner Erfahrung im Entwerfen von Modellen für den 3D-Druck brachte mich das Video auf die Idee, meine eigene Version des gesamten fraktalen Schraubstocks zu entwerfen. Es kostete mich fünf Tage, bis die erste funktionstüchtige Version entworfen und gedruckt war. Am 30. Juni stellte ich das Video, das meine Erfahrungen bis dorthin zeigt, auf der Plattform zur Verfügung (siehe Link).

Mein ursprünglicher Entwurf war ein sehr einfacher; der kleinste, vierte Satz Backen fehlte. Ein erstes Experiment eben, um zu sehen, wie gut sich der Mechanismus auf den 3D-Druck übertragen lässt. Mein Entwurf des Schraubstocks war allerdings nur der erste und in den folgenden Wochen stellten verschiedene Designer die ihren vor.

Schraubstock-Variationen

Inzwischen gibt es allein auf Thingiverse über ein Dutzend verschiedene fraktale Schraubstock-Designs; zu viele, um sie hier alle zu vergleichen. Es folgen drei der beliebtesten Entwürfe, die sich besonders stark voneinander unterscheiden: der Schraubstock von Michael Laws von *Teaching Tech*, der vollständig gedruckte Schraubstock von James Chiang alias *ToMaTo Lab* und mein eigener V2-Entwurf (alle Links siehe Kurz-URL in der Kurzinfor).

In der Tabelle wird klar, wie unterschiedlich die drei Entwürfe selbst auf der grundlegendsten Ebene sind.

Unterschiedliche Design-Ziele

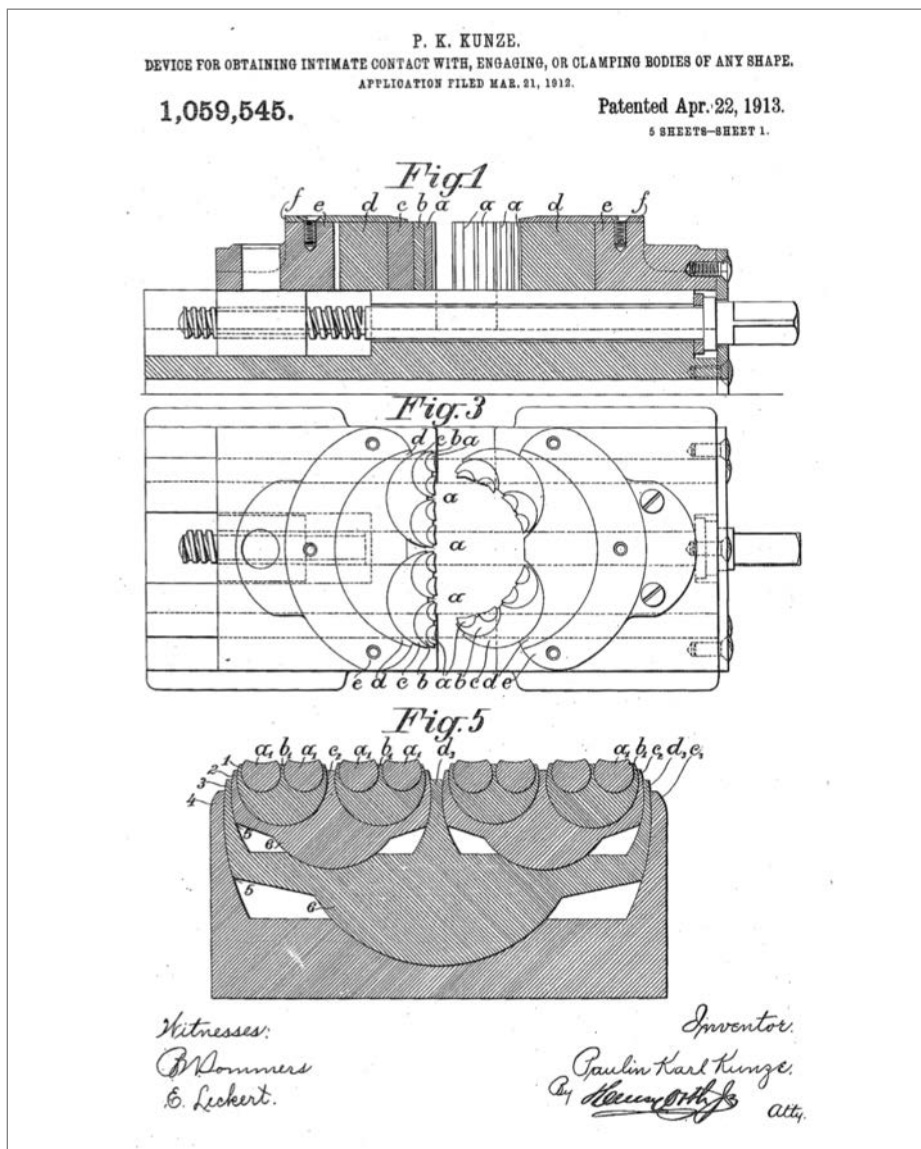
Neben der Nachbildung des ursprünglichen Mechanismus verfolgt jeder Entwurf einige



Das Video, mit dem alles begann ...

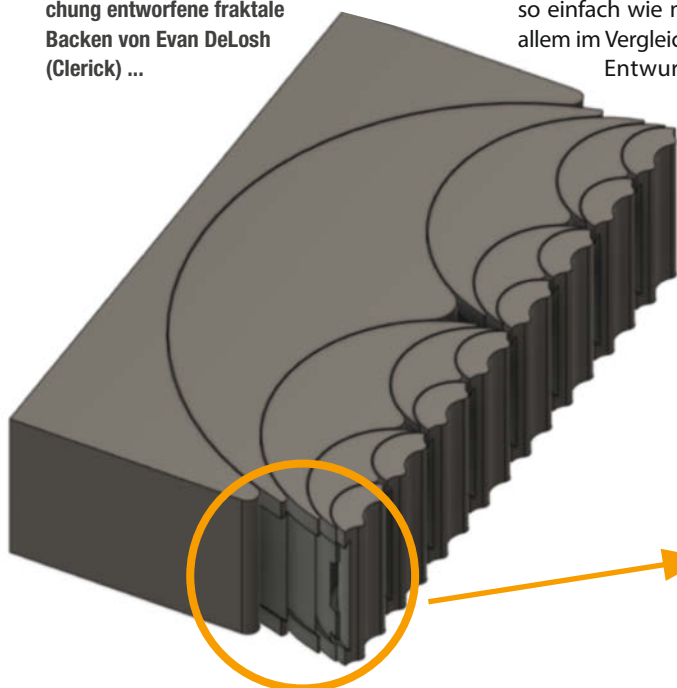


... veröffentlicht am 24. Juni 2021 von Hand Tool Rescue

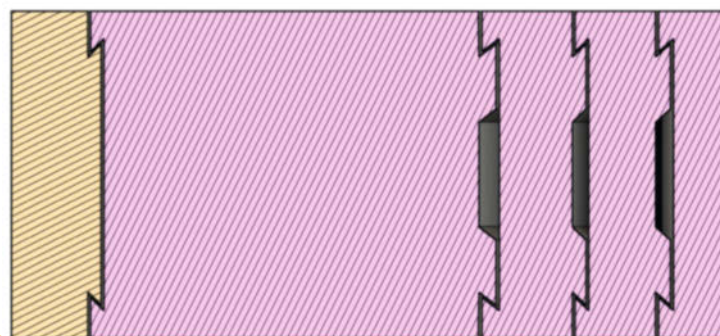


Auszug aus der Patentschrift von 1913

Am Tag der Veröffentlichung entworfene fraktale Backen von Evan DeLosh (Clerick) ...



zusätzliche Ziele. Michael Laws Entwurf sollte so einfach wie möglich zu drucken sein, vor allem im Vergleich zu meinem ursprünglichen Entwurf – eine sehr verständliche Vorgabe. Meine beiden Entwürfe sind einteilige Drucke mit Stützmaterial



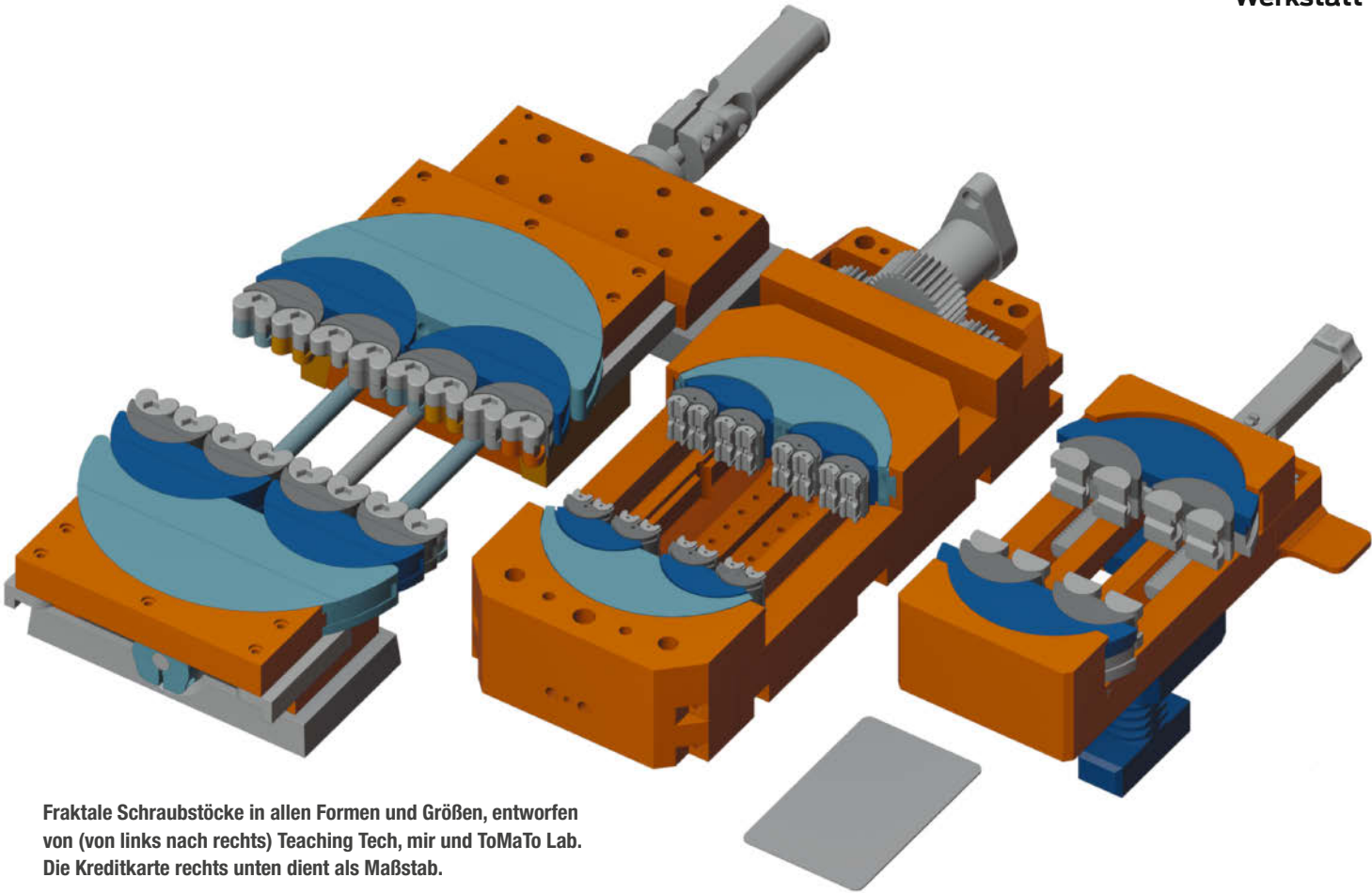
... mit Schwalbenschwänzen für die freie Rotation (hier der Querschnitt)

für die Schwalbenschwänze an den Backen. Das funktioniert zwar, erfordert aber einen sehr gut kalibrierten 3D-Drucker – auf meinem alten i3-B-Klon etwa wäre das nicht möglich. Michael Laws Lösung besteht darin, die Backen in zwei Hälften zu teilen und sie später zusammenzuschrauben. So können die Schwalbenschwänze ohne Stützmaterial gedruckt werden.

Diese Lösung hat auch den Vorteil, dass sich die Backen durch die Art der Befestigung nicht so weit drehen können, um am Ende herauszufallen. Da er die beiden Hälften zusammenfügt, muss er die Backen nicht aufschieben, im Gegensatz zu meinem Entwurf. Dadurch können die Backen mit einer Lippe versehen werden, die sie festhält. (Beim Originalwerkzeug aus Metall wurde dafür eine kleine Madenschraube verwendet, aber in dem Maßstab, in dem hier gedruckt wird, ist das zu kleinteilig). Der Entwurf von ToMaTo Lab greift ebenfalls auf diese geteilten Backen zurück, um das Drucken zu erleichtern.

Ein weiteres Ziel von Michael Laws war es, so wenig Kunststoff wie möglich zu verwenden. Anstelle eines großen, 3D-gedruckten Bettes, auf dem die Backen befestigt werden, sieht sein Entwurf vier Module vor, die an Metallstäben befestigt werden, wie sie oft in 3D-Druckern für Linearführungen benutzt werden. Man kann den Schraubstock so breit wie gewünscht anlegen und ist nur durch die Länge der verfügbaren Stäbe eingeschränkt. Im Vergleich dazu arbeiten die Entwürfe von ToMaTo und mein eigener mit einer festen Länge. Es sind jedoch die Dateien für alle drei Entwürfe als Open Source verfügbar und könnten leicht angepasst werden, um die Länge abzuändern.

Aufgrund dieser Konstruktionsentscheidungen ist von den drei Entwürfen Michael Laws Schraubstock der mit den wenigsten gedruckten Teilen. ToMaTo Labs ist ganz das Gegenteil: James Chiangs Entwurf ist dem meinen sehr ähnlich – er verwendet zwei Gewindespindeln, die durch Zahnräder über einen gemeinsamen Schraubgriff bewegt werden – aber hier war das Hauptziel, komplett in 3D zu drucken. Durch einige clevere technische Ansätze konnte er das umsetzen,



Fraktale Schraubstöcke in allen Formen und Größen, entworfen von (von links nach rechts) Teaching Tech, mir und ToMaTo Lab. Die Kreditkarte rechts unten dient als Maßstab.

z. B. indem er die Gewindespindeln im Wesentlichen rechteckig gestaltet hat. Dadurch können sie gleichzeitig für den Druck flach liegen und im montierten Schraubstock als Schrauben fungieren. Der Rest seines Entwurfs wird durch gut platzierte Klammern und Reibung zusammengehalten.

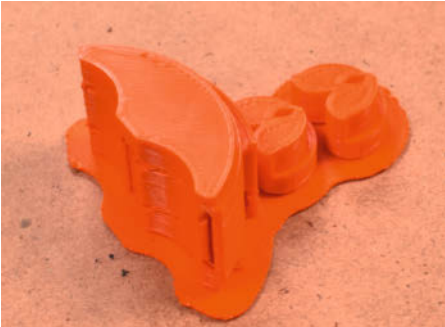
Mein Entwurf im Detail

Was meinen Entwurf betrifft, so war mein Hauptziel bei V2, ihn so funktional wie möglich zu gestalten. Zu diesem Zweck habe ich einige zusätzliche Funktionen eingebaut, um ihn vielseitig einsetzbar zu machen: Befestigungspunkte für Zubehör wie Lupen, eine *Dritte Hand* und Licht, Aussparungen für die Verlegung von Kabeln für eine Hintergrundbeleuchtung und ein Schienensystem zur Befestigung von Stützplatten, wenn ein Objekt mehr Halt von unten benötigt.

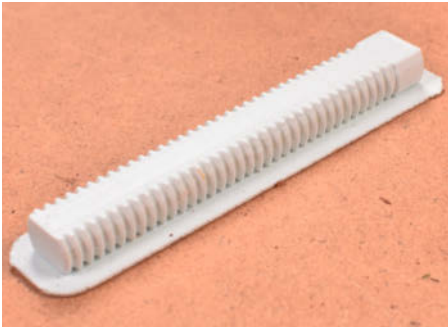
Ich habe außerdem versucht, meinen Entwurf so stabil wie möglich zu gestalten. Dazu habe ich nach einer bei mir bewährten Methode Gewindestangen durch die gesamte Länge des Drucks geführt, die durch Festziehen den Druck komprimieren und den Kunststoff steifer machen. Insgesamt kleinere, dickere Backen helfen gegen den Durchhang, den ich bei dünneren Backen noch als problematisch empfand.

Ich habe auch eine zweite Variante des kleinsten Backensatzes entworfen, bei der die

	Borge V2	ToMaTo Lab	Teaching Tech
Maximale Spannweite	60 mm	50 mm	unendlich
Montage-Optionen	T-Mutter, Festschrauben auf Arbeitsplatte	Tischklemme, Festschrauben auf Arbeitsplatte	TPU-Füße
Gedruckte Teile	41	39	78
Nicht-gedruckte Teile	60	0	200
Druckzeit (ca.)	3–4 Tage	1,5–2 Tage	2–3 Tage
Benötigtes Material (ca.)	650 Gramm	275 Gramm	500 Gramm
Anzahl der Backensätze	4	3	4
Open Source	Ja	Ja	Ja
Anzahl der Gewindespindeln	2	2	1



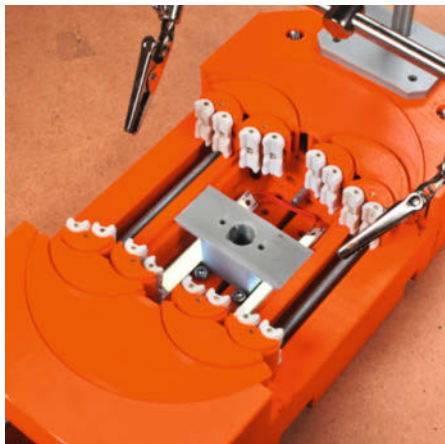
Die Teaching-Tech-Backen benötigen kein Stützmaterial und lassen sich daher leichter drucken.



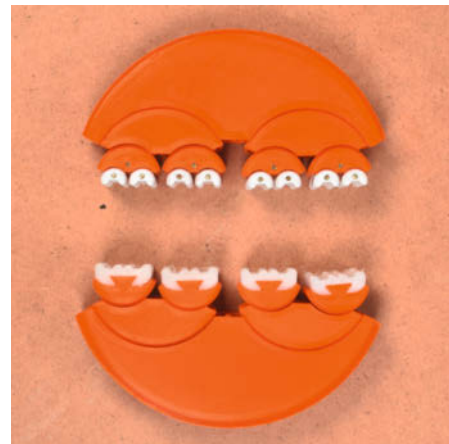
Durch die angeschnittenen Abflachungen oben und unten erleichtert ToMaTo Lab das Drucken dieser Schrauben erheblich.



Mein Entwurf mit angebrachtem Add-on, hier eine „Dritte Hand“



Eine Hintergrundbeleuchtung und Stützblöcke, die im „Graben“ meines Schraubstocks montiert sind



Optionale flexible TPU-Backen (unten) im Vergleich zu den normalen starren Backen (oben)

Backen durch ein gummiartiges TPU-Teil ersetzt werden, das viel einfacher herzustellen, aber weniger anpassungsfähig ist. Insgesamt habe ich mich für eine kleine, aber klobige Form entschieden, um das Ganze so stabil und einsetzbar wie möglich zu machen.

Die vielen verschiedenen Entwürfe, ihre einzigartigen Eigenschaften und ihre superschnelle Umsetzung zeigen, was an der 3D-Drucktechnologie und der 3D-Druck-Community so besonders und so gut ist. Alle Designer haben andere dazu ermutigt, die eigenen Entwürfe weiterzuentwickeln und zu verbessern, um für alle auf das beste Ergebnis zu kommen.

Diese Open-Source-Philosophie hat es dem 3D-Druck ermöglicht, sich so schnell dahin zu entwickeln, wo er heute steht. Da all diese Entwürfe Open Source sind, ist es ein Leichtes, eine Funktion, die einem Maker an einem fremden Entwurf gefällt, in den eigenen zu integrieren.

Fantasievolle Einsatzmöglichkeiten

Fraktale Schraubstöcke lassen sich gut als Lösung auf der Suche nach einem Problem beschreiben. Es gibt viele Verwendungszwecke, aber solange kein konkreter Bedarf da ist, kann es eine Herausforderung sein, einen konkreten Zweck zu finden. Zum Glück finden sich einige Ideen, wie der fraktale Schraubstock zum Einsatz kommen könnte, in den Kommentaren unter meinen Videos:

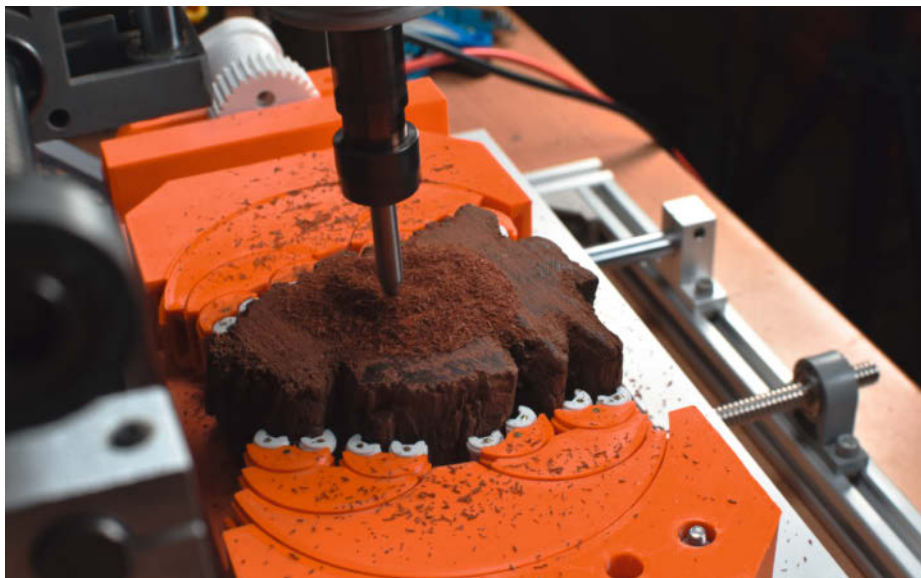
- Halten von Messerteilen/Messergriffteilen
- Bemalen und Verkleben von Modellteilen
- Gravieren von seltsam geformten Steinen
- Gravieren kleiner Gegenstände, insbesondere Münzen und Zippo-Feuerzeuge

Ein Einsatzbereich wurde auch bei Menschen mit Behinderungen genannt; denn manche können nur schwer bestimmte Dinge halten – eine Selbstverständlichkeit für die meisten Menschen.

In meinem Video zur Version 2 (siehe Link) wurde der Schraubstock als Einspannvorrichtung für eine CNC-Fräse eingesetzt. Das war als Test gut geeignet, da die Maschine nicht besonders vorsichtig mit dem Schraubstock umgeht – sollte die Konstruktion versagen, würde die CNC-Maschine nicht aufhören, Druck auf sie auszuüben.

Für meine Demonstration habe ich ein sehr seltsam geformtes Stück Holz verwendet, das sich sonst nur schwer einspannen ließe, ohne Zugriff auf einen Großteil der Oberseite zu verlieren. Da es von den Seiten her eingespannt wurde, konnte ich das Stück noch maschinell flächig glätten. Bei diesem Test ging es um ein einfaches Tic-Tac-Toe-Muster und der Schraubstock hielt ordentlich stand.

Fraktale Schraubstöcke sind ein faszinierendes Beispiel dafür, wie die Maker-Community zusammenarbeitet, um eine neue (oder ist es eine alte?) Technologie für alle verfügbar zu machen. Bei Interesse an einem der hier erwähnten Designs finden sich die 3D-Dateien und Bauanleitungen bei den jeweiligen Usern auf Thingiverse. Eine Suche nach „Fractal Vise“ wird diese Entwürfe und viele mehr auswerfen. Wer meinen Entwurf nachbauen will, findet viele nützliche Hinweise in einem zweiten Video (siehe Link in der Kurzinfo). —pek



Mein fraktaler Schraubstock, montiert auf einer herkömmlichen CNC-Maschine Typ 3018



Ergebnis: Der Schraubstock hielt stand!



DIGITAL DESIGN & UX NEXT

Produktentwicklung, Technologiepotenziale
und Gestaltung zusammendenken

Konferenz im Kraftwerk · München · 14. und 15. Juni 2023

Für Usability- & UX-Profis, Digital Designer, Requirement Engineers
und Product Owner.

Aus dem Programm:

- Sind Digital Designer die neuen Product Owner?
- Vom UX-Neuling zum Experten: Wie UX ein integrierter Teil der Produktentwicklung wurde
- UX in der Robotik – Der Arbeitsalltag zwischen Mensch und Maschine
- UX trifft KI – Sind Digital Designer mit ihren Ideen eigentlich noch zu retten?

Workshops am
13. und 15. Juni

www.dd-ux.de | Jetzt Tickets sichern!

Veranstalter



MAIBORNWOLFF



dpunkt.verlag

Kooperationspartner



GERMAN UPA

© Copyright by Maker Media GmbH
und User Experience Professionals



VERBAND
DEUTSCHER
INDUSTRIE
DESIGNER





Hightech-Messgerät für große Flächen

Große, unregelmäßige Formen lassen sich mit Meterstab, Maßband und Lasermessgerät nur unzureichend messen. Abhilfe will das Messgerät Moasure One schaffen, dass die eigene Bewegung im Raum erfasst und so Längen, Höhen, Flächen und Winkel messen kann. Wir haben das Gerät getestet und auf Genauigkeit überprüft.

von Johannes Börnsen

Wer einen Rasenmähroboter installieren, die Fassade streichen oder einen Zaun bauen möchte, hat immer dasselbe Problem: Diese Flächen und Strecken lassen sich mit Hausmitteln schlecht messen. Das liegt einerseits an der Größe der zu messenden Fläche, vor allem aber auch an deren Komplexität: Die wenigsten Gärten halten sich an gerade Linien und rechte Winkel. Hier will 3D Technologies Limited mit ihrem Moasure One (zusammengesetzt aus move und measure) Abhilfe schaffen. Das Messgerät mit fest verbautem Akku ist etwa handtellergroß und beherbergt Sensoren, die die Bewegung des Messgeräts im dreidimensionalen Raum registrieren. Zum Einsatz kommen dabei laut Herstellerwebseite „Hochleistungssensoren der Raketentechnologien“. Gemeint sind damit Beschleunigungssensoren, Gyroskope und Magnetometer, ähnlich den Sensoren, die in Fitnessarmbändern die gelaufenen Kilometer erfassen.

Ein Display hat das Gerät nicht, stattdessen wird es über Bluetooth mit dem Smartphone und der hauseigenen App gekoppelt. Dort lassen sich Entfernungen, Winkel, Flächen, Kurven, Steigungen, Höhen, Durchmesser und

Volumen messen, anschauen, bearbeiten und in CAD-taugliche Formate wie DWG und EPS exportieren.

Der Messablauf

Nach dem Start einer neuen Messung in der App legt man das Gerät Messpunkt für Messpunkt entlang der zu messenden Strecke ab. Der Abstand dieser Messpunkte kann dabei, je nach zu messender Form, wenige Zentimeter bis einige Meter betragen. Am Messpunkt angekommen muss das Gerät für einige Sekunden ruhig verweilen, bevor es zum nächsten Messpunkt weiterbewegt werden kann. Die Aufzeichnung der Strecke zwischen den Messpunkten kann dabei wahlweise als gerade Linie erfolgen oder jede Bewegung zwischen den Messpunkten beinhalten. Auf diese Weise lassen sich auch mehrere hundert Meter lange Umrisse an die App übermitteln. Bei jedem Messpunkt werden alle drei Dimensionen aufgezeichnet. Misst man eine Rasenfläche, erhält man also nicht nur eine zweidimensionale Kontur, sondern ein dreidimensionales Höhenprofil – zumindest für die Rasenkante.

Das Höhenprofil der Fläche innerhalb der Kontur wird nicht erfasst. Möchte man auch Beete, Fenster oder ähnliche Formen innerhalb einer Kontur vermessen, muss man diese mit unsichtbaren Messlinien von der Außenkontur aus einbinden. Ein Video vom Messablauf finden Sie über den Artikellink.



Kein Display, keine Taste. Zum Einschalten tippt man doppelt auf das Gehäuse. Die hervorstechende Ecke des Messgeräts ist der Bezugspunkt für die Messung.

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



Den Dreh raus haben

Der Messvorgang benötigt etwas Übung: Dreht oder bewegt man das Gerät zu schnell oder zu langsam, weist ein Warnhinweis in der App sowie eine LED am Messgerät auf die nun ungenaue Messung hin. Auch das Anstoßen an einen Baum oder Stein, zu weiter Abstand und zu lange Wartezeiten zwischen den Messpunkten beeinträchtigen das Ergebnis. Die Messung lässt sich zwar dennoch fortsetzen, für eine genaue Messung muss man jedoch von vorne beginnen. Im Test ist uns die Vermessung einer 1200qm großen Rasenfläche erst im vierten Durchlauf ohne Fehler gelungen, was jeweils einen Zeitaufwand von etwa 20 Minuten erforderte. Mit der Zeit entwickelt man jedoch ein Gefühl für die richtige Bewegung und senkt so die Fehlerquote. Apropos Fehlerquote: Während der Messung sollte man keine Anrufe annehmen oder auf eingehende Nachrichten tippen: Das hat uns im Test eine fast fertige Messung abgebrochen.

Um sich nicht an jedem Messpunkt bücken zu müssen sowie um an unzugängliche Messpunkte zu gelangen, gibt es einen auf den Namen Moasure Stick getauften, etwa 1,20m langen Teleskopstab, an dessen unterem Ende sich das Messgerät einschieben lässt. Das Smartphone wird am oberen Ende befestigt, den Status der aktuellen Messung hat man stets im Blick. So ausgerüstet benötigt man für 100 Meter Messstrecke, abhängig von deren Komplexität, etwa zwei bis vier Minuten.

Auf die Länge kommt es an

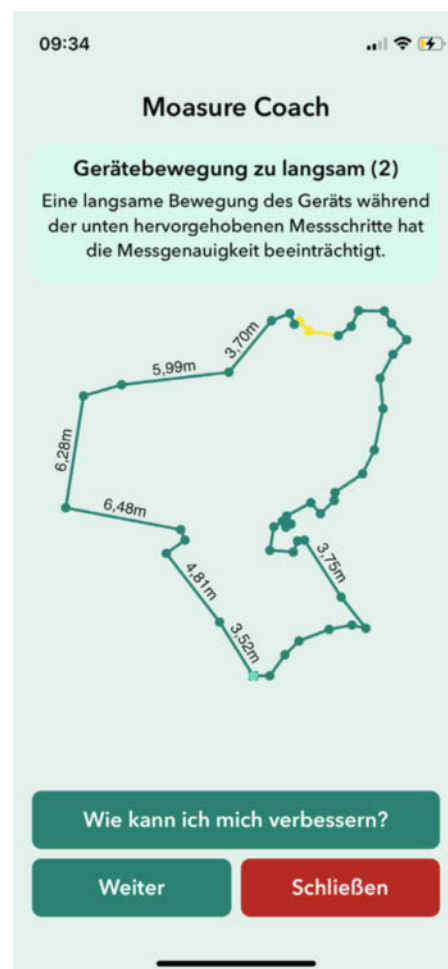
3D Technologies Limited gibt für Moasure One eine Genauigkeit von +/- 0,5% bei Längenmaßen und durchschnittlich 2% bei Flächenberechnungen an. Eine solche Prozentangabe findet man auch je Messvorgang in der App. Nach einigen Übungsrunden entspricht diese Angabe der vom Hersteller versprochenen Genauigkeit. Dieser Wert dürfte allerdings nur anzeigen, wie gut man die Wunschbewegungsgeschwindigkeit des Messgeräts eingehalten hat und nicht, wie verlässlich einzelne Maße tatsächlich der Realität entsprechen – schließlich kennt das Messgerät ja die tatsächlichen Maße als Vergleich nicht. Um dies zu überprüfen haben wir einen 25 Meter langen, U-förmigen Messpfad abgesteckt und zehn identische Messungen vorgenommen, die laut App ohne Fehler durchgeführt wurden.

Anschließend haben wir den Abstand der beiden Enden des U in der echten Welt mit den Messergebnissen verglichen. Im Mittelwert sind die beiden Enden laut Messgerät 2,42 Meter voneinander entfernt, was der Realität mit 2,50 Meter nahe kommt. Allerdings ist der Abstand laut Messgerät im kleinsten Fall nur 1,69 Meter, im größten 3,15 Meter, was deutlich über der vom Hersteller angegebene

nen Toleranz liegt. Da die beiden Schenkel des U bei allen Messungen recht ähnlich aussehen, wird diese Abweichung am Wendepunkt entstehen, was sich entsprechend in den Abstand der Endpunkte auswirkt. Ganz anders sieht es aus, wenn man den Umfang des U anschaut. Im Mittelwert liegt dieser bei 53,45 Meter, der größte Umfang beträgt 54,32 Meter, der kleinste 52,25 Meter. Das entspricht einer Abweichung von etwa +/- einem Meter bzw. etwa 2%. Allerdings wurden in unserer Vergleichsmessung auch lediglich die beiden Endpunkte des U an definierten Stellen gemessen, die Umfangabweichung überrascht also nicht und liegt an den pro Durchlauf etwas unterschiedlich gesetzten Messpunkten.

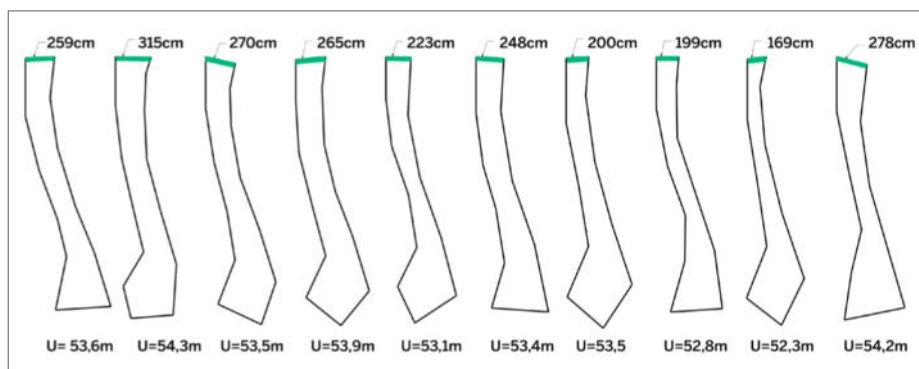
Auf der Webseite empfiehlt der Hersteller Moasure One für die „Ermittlung des Platzbedarfs, der benötigten Menge an Material, schnelle Kostenaufstellung oder Angebotserstellung, Begutachtung, insbesondere Erstellung von 3D-Darstellungen und unmittelbare Weiterbearbeitung in CAD“ und weist darauf hin, dass Moasure One nicht geeignet ist für Anwendungen, bei denen exakte Millimeter erforderlich sind und dass es nicht die Anforderungen von Dachdeckern oder Innenarchitekten erfüllt. Das bestätigen unsere Messungen: Hat man das nötige Gefühl für das fehlerfreie Führen des Messgeräts entwickelt, lassen sich in wenigen Minuten große, komplexe Formen vermessen. Die dabei erreichbare Genauigkeit ist für die eingangs genannten Anwendungen gut geeignet und mit herkömmlichen Messmethoden schwer zu erreichen. Oder anders gesagt: Um einen Zaun zu kaufen ist Moasure One gut geeignet, zuschneiden sollte man ihn anhand der damit erstellten Messungen aber nicht.

Einen Vorteil hat der klassische Meterstab aber trotzdem: Er ist unschlagbar günstig, während Moasure One für den nur gelegentlichen Einsatz zu teuer ist: Für das Messgerät samt Schutztasche muss man rund 505 Euro



Im gelb markierten Bereich oben hat sich das Gerät zu langsam bewegt. Das reduziert die Messgenauigkeit. Auch zu schnelles Drehen tadelt die App.

und für den Moasure Stick noch einmal weitere 90 Euro einplanen, die App ist kostenlos. Bei regelmäßigem Einsatz ist der Zeitgewinn für Bauleiter, Architekten und Garten- und Landschaftsbauer jedoch groß. —jom



Das schwarz dargestellte U wurde zehn mal mit Moasure One vermessen. Die grün markierte obere Strecke ist in der echten Welt 250cm lang. Diesen grünen Abstand der Schenkel haben wir als Testmaß genommen. Vergleicht man stattdessen den Umfang der gemessenen Strecken, liegen die Ergebnisse sehr dicht beieinander. Die Abweichung liegt auch an den nicht immer gleich platzierten Messpunkten.

Ausprobiert
— von Make: —

BB Q20

Die wahrscheinlich kleinste USB-Tastatur der Welt



Das *BB Q20 Keyboard* ist eine Mini-Tastatur für USB und I²C inklusive eines Trackpads, das auch als USB-HID-Maus arbeitet. Der Entwickler Arturo aus Schweden entwickelt Open Hardware und ist Gründer der Firma *Solderparty*. Das BB Q20 ist seine neueste Kreation für alle, die es etwas kleiner mögen oder brauchen.

Manche Projekte benötigen einfach eine Hardwaretastatur, manchmal sind virtuelle Tastaturen auf Bildschirmen einfach nicht geeignet oder der Benutzer mag nicht darauf tippen. Die Hardwaretastaturen der BlackBerry-Geräte gehören sicherlich zu den besseren kleinen Tastaturen – so ist es nicht verwunderlich, dass sie schon oft für ähnliche Projekte verwendet wurden und ein solches Keyboard steckt auch in diesem Produkt beziehungsweise Projekt.

Das BB Q20 Keyboard kann man fertig kaufen oder dank Open-Source-Lizenz anhand der Quellen selbst herstellen. Zum Lieferumfang beim fertigen Produkt gehören eine QWERTY-BlackBerry-BQ10-Tastatur und ein kleines Tastaturgehäuse aus Acrylglas. Kontakt zum Computer oder Mikroprozessor nimmt die Tastatur über USB-C-Buchse oder I²C auf; Stemma-, PMOD-Anschlüsse und natürlich Löt pads sind vorhanden. Zusätzlich verfügt das BB Q20 über vier Zusatztasten, ein Trackpad und ein Mikrofon. Das Trackpad kann als USB-HID-Maus oder mit der ALT-Taste als Pfeiltasten fungieren. Gedrückt erzeugt es einen Mausklick. —caw

Hersteller	Solderparty
URL	solder.party/docs/bbq20kbd
Preis	30 €

Grove UART WizFi360

Wireless-Modul mit ARM Cortex M3

Ende 2018 erschien mit dem W600 des chinesischen Herstellers Winner Micro ein Konkurrent zu den Espressif-Prozessoren, wie sie in den bekannten ESP-Modulen verwendet werden. Der W600 sorgte für Aufsehen, weil er zu vergleichbaren Preisen mit einem ARM Cortex-M3-Kern und integriertem Flash ausgestattet ist. Er wird im neuen WizFi360-Modul verwendet, das die gleichen Fähigkeiten wie der im IoT-Bereich beliebte ESP8266 aufweist und pinkompatibel zu den ESP12E/F-Modulen ist und beim Hersteller rund 3,60 Euro kostet.

Wie beim ESP8266 ist auf jedem WizFi360-Modul eine AT-Befehlssatz-Firmware vorinstalliert, die mit einfachen Textbefehlen gesteuert werden kann. WizFi360 unterstützt die IEEE802.11 b/g/n-Standards und die Modi SoftAP (Access Point), Station (Router-Verbindung) und SoftAP+Station. Mit dem Grove UART WizFi360 bietet Seeedstudio nun auch



Seeedstudio

gleich ein Grove-Modul für das auf seriellen Verbindungen basierende eigene Bausteinsystem an. Pegelwandler sind integriert, es ist daher mit 5V-Logik kompatibel. —cm

Hersteller	Seeedstudio
URL	make-magazin.com/xvjh
Preis	12,90 US-\$

Original Prusa MK4

FDM-3D-Drucker

Der Nachfolger des bewährten Prusa MK3S+ soll schneller und präziser arbeiten und leichter zu warten sein – dafür hat der Hersteller nach eigenen Angaben mehr als 90 Prozent der Bauteile überarbeitet. Die wichtigste Neuerung bildet das hausintern produzierte 32-Bit-Mainboard *xBuddy* auf STM32-Basis. Darauf läuft eine von *Klipper* inspirierte Firmware, die *Input Shaper* und *Pressure Advance* verwendet (siehe dazu auch Seite 10). Kombiniert mit präziseren Stepper-Motoren druckt der MK4 dadurch schneller als seine Vorgänger, während die beim Druck entstehenden Resonanzen ausgeglichen werden und der Materialfluss optimiert wird.

Den Direct-Extruder hat Prusa gegen eine kleinere Variante seines *Nextruder* getauscht, der bereits mit dem Prusa XL (Make 6/21, S. 124) vorgestellt wurde. Dessen Komponenten lassen sich samt Hotend mit wenigen Handgriffen austauschen, da ihre Kabel nicht mehr direkt bis zum Mainboard verlaufen, sondern auf ein kleines Breakout-Board am *Nextruder* gesteckt sind (siehe Video unter dem Kurzlink in der Tabelle). Von diesem verläuft lediglich ein einzelnes Kabel zum Mainboard des Druckers. Eine weitere Neuerung bietet die im *Nextruder* integrierte Wägezelle, die für das Homing der Z-Achse verwendet wird. Dadurch entfällt das manuelle Nachjustieren des Z-Offset, ganz gleich, welche Druckoberfläche man verwendet.



Prusa Research

Optisch hat der MK4 ein Farbdisplay als Upgrade erhalten, das G-Code-Miniaturansichten, mehr Text und bei Fehlern sogar QR-Codes anzeigen kann, die zu den Hilfeseiten des Prusa-Supports führen. Der SD-Kartenleser ist einem USB-Steckplatz gewichen, wobei sich Druck-Daten auch über das Netzwerk an den 3D-Drucker schicken lassen.

Wie bei Prusa üblich, gibt es den MK4 sowohl fertig montiert als auch als Bausatz; bei Redaktionsschluss war die Lieferzeit für das Komplettgerät deutlich kürzer. Ab Juni soll es drei Upgrade-Kits für Besitzer eines MK3S und MK3S+ geben, um diese auf die Versionen MK3.5, MK3.9 oder MK4 aufrüsten zu können. Wie sich die drei Varianten unterscheiden, lässt sich unter der Kurz-URL in der Tabelle nachlesen. —akf

Hersteller	Prusa Research
URL	make-magazin.com/xvjh
Preis	Fertiggerät: 1199 €, Bausatz: 889 €

Creality K1 und K1 Max

Schnelle FDM-3D-Drucker

Mit bis zu 600mm/s und damit bis zu 12-mal schneller als übliche 3D-Drucker soll der neue K1 von Creality drucken. Möglich machen soll dies unter anderem die stabile CoreXY-Mechanik, Multipoint-Leveling mithilfe von Dehnungsmessstreifen im Druckbett und die neue Firmware, deren Funktionen stark an Klipper erinnern (siehe auch Seite 10).

Weitere Verbesserungen sind das neue Hotend, dessen Heizleistung laut Hersteller einen Filament-Durchfluss von bis zu 32mm³ pro Sekunde zulassen soll. In nur 40 Sekunden soll es von kalt auf 200°C aufheizen. Die Mechanik kann mit bis zu 20.000mm/s² Beschleunigung in nur 0,03s auf die maximale Verfahrgeschwindigkeit von 600mm/s beschleunigen. Beindruckende Videos auf der Herstellerseite demonstrieren dieses Tempo im Vergleich zu üblichen Druckgeschwindigkeiten von 50mm/s (siehe Link in der Tabelle): Das bekannte Boot *Benchy* wird in nur 13 Minuten fertiggestellt. Gedruckt wird in einem komp-

plett geschlossenen Gehäuse, die maximale Modellgröße beträgt 25cm in der Höhe und 22cm Grundfläche im Quadrat.

Den Creality K1 in der Standardversion kann man bereits im Online-Shop des Herstellers bestellen, er soll ab Ende Mai direkt von China aus per Luftfracht ohne zusätzliche Versandkosten ausgeliefert werden. Die teurere Version K1 Max bietet nicht nur eine mit 30cm in allen drei Dimensionen deutlich größeren Bauraum, sie soll zusätzlich über ein LIDAR im Druckkopf die erste Schicht überwachen und Alarm schlagen, sollte es Unregelmäßigkeiten geben. Darüber hinaus behält eine Kamera den Druck im Auge, vergleicht das Werkstück mit der Vorlage und registriert Spaghetti-Bildung, Fremdkörper oder lose Teile im Druckraum, ganz ähnlich wie der *AnkerMake M5* (S. 8). Der K1 Max ist noch nicht vorbestellbar, man kann beim Hersteller aber seine Mail-Adresse hinterlassen und wird benachrichtigt, sobald der Drucker verfügbar ist. —hgb



Hersteller	Creality
URL	make-magazin/xvjh
Preis	K1: 649 €, K1 Max: 1049 €

Balkon-Photovoltaik-Anlagen

Solarstrom selber erzeugen für Hausbesitzer:innen und Mieter:innen

Balkon- oder Kleinsolaranlagen mit bis zu 600W liegen im Trend; auch in der Make sind bereits mehrere Artikel zu diesem Thema erschienen. Dennoch bleiben für Mieter und Hausbesitzer viele Fragen offen: Ist das legal? Wie und wann muss ich die Anlage anmelden? Lohnt sich das? Kann ich das selbst machen? Welche Ausrichtung ist richtig? Diese und viele weitere Fragen klärt das kompakte 96-seitige Buch (Print und PDF) *Balkon-Photovoltaik-Anlagen* der Autoren Rolf Behringer und Sebastian Müller, erschienen im Ökobuch Verlag.

Die Autoren sind in der Balkonsolarbranche keine Unbekannten: Beide sind in verschiedenen Verbänden und als Autoren aktiv an der lebhaften Entwicklung von solaren Kleinanlagen beteiligt. Das merkt man dem Buch an. Aus der Praxis heraus wird er-

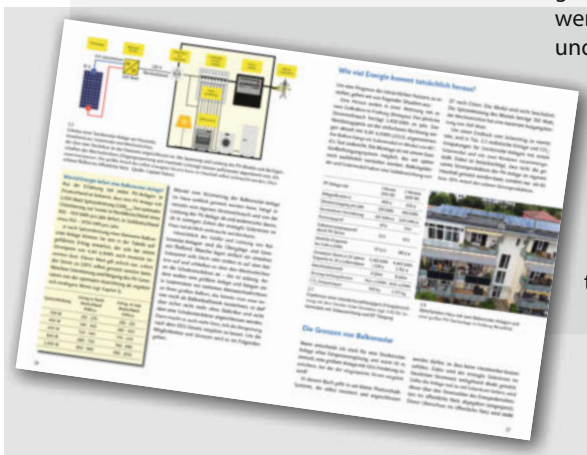
klärt, wie man schnell zu einem Balkonkraftwerk, Solartisch oder Gartensolar für den eigenen Bedarf gelangt. In den Kapiteln 2 und 3 wird eine Art Kochrezept vorgestellt, wie man schnell, sicher, legal und im Einvernehmen mit dem Vermieter zum eigenen Solarstrom kommt. Die restlichen Kapitel bleiben dann für Fragen, Probleme und Infos über die (E-)Technik oder die Elektronik hinter Solaranlagen.

In den weiteren Kapiteln finden Sie zusätzliche Informationen, um eine Anlage für Ihre Situation optimal auszulegen, alle Komponenten selbst zu planen und auch an ungewöhnlichen Standorten sicher zu montieren. Es folgen Tipps zur Messung der erzeugten Energie, zur optimalen Nutzung des Solarstroms und zur Speicherung. Ein ganzes Kapitel widmet sich der Wiederverwendung von gebrauchten Solarmodulen und damit einer sehr kostengünstigen Lösung (siehe auch Make 1/22, S. 38). Dabei geht es um die Beschaffung und Auswahl der Module, das benötigte Werkzeug, die Auswahl der Module nach Leistung und deren Verschaltung untereinander und mit dem Wechselrichter. Aber auch die Entsorgung defekter Module wird berücksichtigt.

Das Buch ist verständlich geschrieben, die vielen Grafiken und Fotos sind gut gemacht, zeigen die Praxis und nicht nur die heile Welt der Hersteller und Stockfotos. Die Quellen sind ausführlich mit Links belegt und die verschiedenen Solarrechner und andere externe Informationen sind gut verlinkt. Wo es sinnvoll ist, werden QR-Codes verwendet, um z.B. Musterbriefe herunterzuladen. —caw



Autoren	Rolf Behringer, Sebastian Müller
Verlag	Ökobuch Verlag
Umfang	96 Seiten
ISBN	978-3-947021-35-2
Preis	17,95 €
Website	oekobuch.de/buecher/balkon-photovoltaik-anlagen/



NFC-Powered e-Paper

7.5"-E-Ink-Display mit drahtloser Stromversorgung



Waveshare

E-Ink-Displays (auch E-Paper genannt) brauchen nur beim Wechseln des Bildinhalts Strom. Daher halten die Akku-Ladungen zum Beispiel in eBook-Readern meist viele Wochen oder gar Monate. Dieses 7,5-Zoll-Display von Waveshare kommt sogar ohne Akku oder Buchse zur Stromversorgung aus.

Trotzdem braucht es natürlich ein Quäntchen Energie, wenn man es neu beschreiben möchte. Sowohl die Datenübertragung als auch die Stromversorgung erfolgen bei diesem Display aber drahtlos über Nahfeld-Kommunikation (**N**earfield communication, **NFC**). Diese Technik wird bislang meist für Diebstahlschutz oder zum Bezahlen per Smartphone eingesetzt. Viele Smartphones sind aber in der Lage, auch eine gewisse elektrische Leistung an NFC-Geräte zu übertragen.

Um das Display mit einem Smartphone programmieren zu können, muss diese Leistung groß genug sein. So weist der Hersteller *Waveshare* darauf hin, dass sich keine Samsung-Smartphones dafür eignen. Für geeignete Geräte stehen entsprechende Android- und iOS-Apps zur Verfügung. Mit denen lassen sich Text und Grafiken anlegen und übertragen. Als Alternative zum Phone gibt es das Display aber auch im Set mit dem NFC-Programmiergerät ST25R3911B.

Das Display sitzt in einem dünnen weißen Kunststoff-Gehäuse (195,84mm x 137,68mm x 13,00mm). Die Auflösung beträgt 800 x 480 Pixel bei 170° Blickwinkel. Es ist bislang nur in schwarzweiß erhältlich. —hgb

Hersteller	Waveshare
URL	make-magazin/xvjh
Preis	Stand-alone: 75 US-\$, mit Programmiergerät: 104 US-\$

Arduino UNO R4

32-Bit-Version des klassischen Bastelboards

Der *Arduino UNO R4* soll ein Meilenstein werden und 32 Bit im mal geliebten und von anderen gehassten UNO-Formfaktor bieten, genauso leicht zugänglich sein wie seine Vorgänger, nur schneller und mit WLAN. Zwar hat Arduino bereits einige 32-Bit-Boards im Portfolio. Diese sind aber bei weitem nicht so populär, vor allem in der Ausbildung werden immer noch gerne die UNO-Boards verwendet. Das haben in der Vergangenheit bereits andere Firmen erkannt und Boards im UNO-Format und mit besseren Prozessoren herausgebracht. Jetzt zieht Arduino nach.

Den neuen UNO soll es in zwei Varianten geben, mit und ohne WLAN. Beide werden von einem Renesas-ARM-RA4M1-32-Bit-Cortex-M4-Prozessor angetrieben, der mit 5V arbeitet und damit praktisch alle Shields des UNO nutzen können soll. Das Board wird nun über USB-C statt USB-B angeschlossen, die externe Versorgungsspannung kann bis zu 24V betragen. Weitere Verbesserungen sind ein 12-Bit-ADC (Analog to Digital), SPI und CAN-Bus.

Auch wenn andere Hersteller wie *Adafruit* und *Sparkfun* ihre eigenen leistungsfähigen Boards produzieren, die ebenfalls ein komplettes Ökosystem mitbringen und natürlich alle mit der Arduino-Softwarewelt kompatibel

sind und trotz des Beharrens von Arduino auf Buchsenleisten, die es nicht erlauben, aus einer Lochrasterplatine einfach eine Erweiterungsplatine zu machen, haben viele Maker, Bildungseinrichtungen und Maker Spaces immer noch ein Herz für den UNO: Ein angenehmer Formfaktor, die gut zugänglichen Pins, 5V-Logikpegel, viele verfügbare Erweiterungen beschenken den robusten Boards in der Regel ein langes Leben.

Der *Arduino UNO R4* soll Ende Mai 2023 erscheinen, über Preise und Verfügbarkeit ist leider noch nichts bekannt – und auf Fotos des neuen Boards sind stets wichtige Bereiche abgedeckt. Man darf gespannt sein. —caw



Hersteller	Arduino.cc
URL	store.arduino.cc/pages/unor4
Preis	noch nicht bekannt

Alpakka

Open-Source-Gamecontroller

Der Gamecontroller von *Input Labs* sieht aus wie ein herkömmliches Gamepad, bietet aber dank moderner Gyrosensoren, analoger und digitaler Steuerknüppel sowie Buttons die Kombination aus herkömmlicher Controller- und Maussteuerung in einem couchtauglichen Gerät für das Spielen an Konsole und PC in einer Einheit.

Für den Betrieb ist dank des eingebauten Raspberry-Pico-Mikrocontrollers kein Treiber notwendig: Die komplette Palette der möglichen Konfigurationen kann in Profilen auf dem Controller statt auf dem PC oder der Konsole gespeichert werden. So hat man auch auf fremden Rechnern immer seine eigene Konfiguration dabei. Der *Alpakka* emuliert auf Hardware-Ebene Tastatur, Maus und XInput-Gamepad und ist damit sofort unter Windows, Linux und den gängigen Game-Launchern einsatzbereit. Die Firmware steht unter der GPL-Lizenz.

Die gesamte Hardware wird unter der CC-Hardware-Lizenz (Creative Commons) entwickelt. Dank der verfügbaren Quellen für das 3D-druckbare Gehäuse, für die Knöpfe und die Platine kann das Projekt selbst nachgebaut werden – oder auch nach Belieben und individuel-

len Bedürfnissen angepasst.

Auf den sehr ausführlichen Seiten von *Input Labs* kann man sich die Hard- und Software anschauen (siehe im Detail an-Link in der Tabelle). Vor allem die Videos von Aufbau und Konfiguration sowie die Game-Demos mit sichtbarem Controller lassen erahnen, wohin die Reise geht: Auf der Agenda von *Input Labs* stehen noch die Punkte „Profilpflege und mehr Anwendungsprofile“, „Wireless“ und „Vibration“. Derzeit gibt es noch keine kompletten Bausätze, geschweige denn einen fertigen Controller zu kaufen: Derzeit sind noch Maker mit Löterfahrung und 3D-Drucker gefragt. —caw



Input Labs

Hersteller	Input Labs
URL	make-magazin/xvjh
Lizenz	Open Source

Informatikmagie

Kreativer Einstieg in die faszinierende Welt des Programmierens

Das Buch dient als Lernbegleitung für die Oxocards, die wir hier auf der gleichen Seite vorstellen. Es geht einen anderen Weg als viele herkömmlichen Bücher rund ums Programmieren für Einsteiger. Der Autor erklärt anhand von gut gewählten Praxisbeispielen, wie die Python-ähnlichen Skripte für die Oxocards funktionieren und was passiert, wenn man bestimmte Parameter respektive Konstanten ändert.

Das exerziert er schön und leicht verständlich an 13 gut gewählten Beispielen durch, die sich alle rund ums Display und teilweise auch um die verbauten Sensoren drehen. So werden drei Uhren durchprobiert, Bälle, Würmer und Feuer animiert, Schneeflocken und Bäume gezeichnet, Neigungen gemessen und Spannungen angezeigt. Erst nach diesen Kapiteln erläutert Thomas Garaio die Grundelemente der Sprache, etwa welche Arten von Variablen es gibt, wie sie im Speicher abgelegt sind, wie Schleifen und Bedingungen funktionieren und was Klassen und Listen sind.

Der Vorteil dieses Ansatzes: Erst wird man begeistert und inspiriert und dann folgt die trockene Theorie. Man wünscht sich, mehr Autoren würden diesen Weg wählen.

Ein wenig Kritik gibt es dennoch: Der browserbasierte Editor erscheint in einem dunklen Schema, was dazu führt, dass manche Abbildungen im Buch nur schwer zu erkennen sind. Bei der Erklärung von Fließkommazahlen und der Exponentialschreibweise sind im Druck leider die Exponenten verrutscht: aus 2^1 wurde 21 und



aus 10^1 wurde 101. Da steht man als Neuling auf dem Schlauch. Dennoch ist das Buch eine Empfehlung. —dab

Autor	Thomas Garaio
Verlag	hep Verlag (Schweiz)
Umfang	176 Seiten
ISBN	978-3-0355-2230-3
Preis	40 €



Oxocard mini

Lernplattform mit Display und Sensoren

Die Oxocard kommt auf einer kreditkarten-großen Platine daher und richtet sich an Programmierneinsteiger ab 14 Jahren. Sie basiert auf einem ESP32 und ist ausgestattet mit einem Display, einem Lautsprecher und einem kreuzförmig angeordneten Eingabepad. Insgesamt gibt es drei Versionen, die sich leicht unterscheiden: Art-work, Galaxy und Science. Während Artwork und Galaxy nur unterschiedlich dekoriert, aber sonst baugleich sind und einen Beschleunigungssensor enthalten, ist Science zusätzlich mit diversen Umweltsensoren ausgestattet.

Obwohl die Oxocard über kein Gehäuse verfügt, sieht sie ansprechend aus, da die Leiterplatte mit künstlerischen Bildern bedruckt ist. Aufgrund einer fehlenden Batterie muss sie

jedoch immer über USB mit einer Stromquelle verbunden sein.

Außer zwei vorinstallierten Spielen und einigen Demo-Skripten, die die technischen Möglichkeiten des Gerätes vorführen, hat die Oxocard anfangs noch nicht viel zu bieten. Das ändert sich allerdings, wenn man das kleine Gerät mit dem Internet verbun-

den hat und mit dem Online-Code-Editor koppelt (Link siehe Kurz-URL, alternativ kann auch eine Verbindung per USB-Kabel hergestellt werden). Mittels einer Python-ähnlichen Programmiersprache lassen sich hier selber Programme entwickeln. Es handelt sich um eine vereinfachte Version von Python.

In der Online-IDE hat man Zugriff auf den Code von allen Spielen und Demo-Skripten. So kann man sich

durch die Beispiele arbeiten und mit Änderungen spielerisch herausfinden, was welcher Befehl macht und wie man bestehenden Code anpassen kann. Man arbeitet hier in einer sehr „realistischen“ Entwicklungsumgebung, von der der Sprung in eine „richtige“ IDE nicht schwerfallen sollte.

Ich konnte bislang kein Python. Trotzdem konnte ich innerhalb eines Tages eine einfache Version von Snake auf einer Oxocard programmieren. Dank der Dokumentation fühlte ich mich immer angenehm an die Hand genommen. Wäre ich heute 14 und hätte auf diese Weise die Möglichkeit, direkt in eine richtige Programmiersprache einzusteigen, wäre ich Feuer und Flamme für die Oxocards. —das

► make-magazin.de/xvjh

Hersteller	Oxon
URL	https://www.oxocard.ch
Preis	59 CHF (Artwork und Galaxy), 89 CHF (Science)



Ausprobiert
— von Make: —

IMPRESSUM

Redaktion

Make: Magazin
Postfach 61 04 07, 30604 Hannover
Karl-Wiechert-Allee 10, 30625 Hannover
Telefon: 05 11/53 52-300
Telefax: 05 11/53 52-417
Internet: www.make-magazin.de

Leserbriefe und Fragen zum Heft: info@make-magazin.de

Die E-Mail-Adressen der Redakteure haben die Form xx@make-magazin.de oder xxx@make-magazin.de. Setzen Sie statt „xx“ oder „xxx“ bitte das Redakteurs-Kürzel ein. Die Kürzel finden Sie am Ende der Artikel und hier im Impressum.

Chefredakteur: Daniel Bachfeld (dab)
(verantwortlich für den Textteil)

Stellv. Chefredakteur: Peter König (pek)

Redaktion: Heinz Behling (hgb), Johannes Börsnes (jom), Akos Fodor (akf), Carsten Meyer (cm), Daniel Schwabe (das), Carsten Wartmann (caw)

Mitarbeiter dieser Ausgabe: Beetlebum (Comic), Christopher Borge, Tim Deagan, Thomas Gamisch, Michael Gaus, Katja Heilmann-Kiefer (Übersetzungen), Detlef Heinze, Karin Hirmer (Übersetzungen), Ivana Huckova, Andreas Koritnik, Miguel Köhnlein, Daniel Springwald, Matthäus Szturc, Leonard Ure, Roman Zajonz

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, Pascal Wissner

Hergestellt und produziert mit Xpublisher:
www.xpublisher.com

Verlag

Maker Media GmbH
Postfach 61 04 07, 30604 Hannover
Karl-Wiechert-Allee 10, 30625 Hannover
Telefon: 05 11/53 52-0
Telefax: 05 11/53 52-129
Internet: www.make-magazin.de

Herausgeber: Christian Heise, Ansgar Heise

Geschäftsführung: Ansgar Heise, Beate Gerold

Anzeigenleitung: 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@dermedienvvertrieb.de
Internet: dermedienvvertrieb.de

Einzelpreis: 13,50 €; Österreich 14,90 €; Schweiz 26,50 CHF;
Benelux 15,90 €

Abonnement-Preise: Das Jahresabo (7 Ausgaben) kostet inkl. Versandkosten: Inland 80,50 €; Österreich 88,90 €;
Schweiz 123,90 CHF; Europa: 95,20 €; restl. Ausland 100,80 €

Das Make-Plus-Abonnement (inkl. Zugriff auf die App, Heise Magazine sowie das Make-Artikel-Archiv) kostet pro Jahr 6,30 € Aufpreis.

Make: Nächste Ausgabe erscheint
am 27. Juli 2023

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

Eine Haftung für die Richtigkeit der Veröffentlichungen kann trotz sorgfältiger Prüfung durch die Redaktion vom Herausgeber nicht übernommen werden. Kein Teil dieser Publikation darf ohne ausdrückliche schriftliche Genehmigung des Verlags in irgendeiner Form reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden. Alle beschriebenen Projekte sind ausschließlich für den privaten, nicht kommerziellen Gebrauch. Maker Media GmbH behält sich alle Nutzungsrechte vor, sofern keine andere Lizenz für Software und Hardware explizit genannt ist.

Für unverlangt eingesandte Manuskripte kann keine Haftung übernommen werden. Mit Übergabe der Manuskripte und Bilder an die Redaktion erteilt der Verfasser dem Verlag das Exklusivrecht zur Veröffentlichung. Honorierte Arbeiten gehen in das Verfügungsrecht des Verlages über. Sämtliche Veröffentlichungen in Make erfolgen ohne Berücksichtigung eines eventuellen Patentschutzes.

Warennamen werden ohne Gewährleistung einer freien Verwendung benutzt.

Published and distributed by Maker Media GmbH under license from Make Community LLC, United States of America. The 'Make:' trademark is owned by Make Community LLC. Content originally partly published in Make: Magazine and/or on www.makezine.com, ©Make Community LLC 2023 and published under license from Make Community LLC. All rights reserved.

Printed in Germany. Alle Rechte vorbehalten.
Gedruckt auf Recyclingpapier.

© Copyright 2023 by Maker Media GmbH

ISSN 2364-2548

Nachgefragt

Wir haben den Test eines 8-Bit-BASIC-Computers im Heft. Was war eure erste Berührung mit Computern oder Konsolen?



Detlef Heinze
Bremen, überwacht seine Modelleisenbahn auf S. 60 per KI

Mein erster Kontakt war der Sharp MZ-80K (8-Bit Z80 Prozessor, 48kB RAM, ca. 1981). Man kann ihn in BASIC, Pascal und Assembler programmieren. Daten und Programme speichert er auf Audio-Kassette.



Linda Wittkugel
Hannover, gestaltet im Marketing nerdige Social-Media-Kampagnen für Make

Ich habe meinem Bruder gerne dabei zugehört, wenn er Mario Kart auf der N64 gespielt hat. In seltenen Ausnahmen durfte ich auch mal selber an den Controller und habe so die Nintendo-Welt entdeckt.



Daniel Schwabe
Hannover, unterstützt seit Mai die Make als Technical Writer

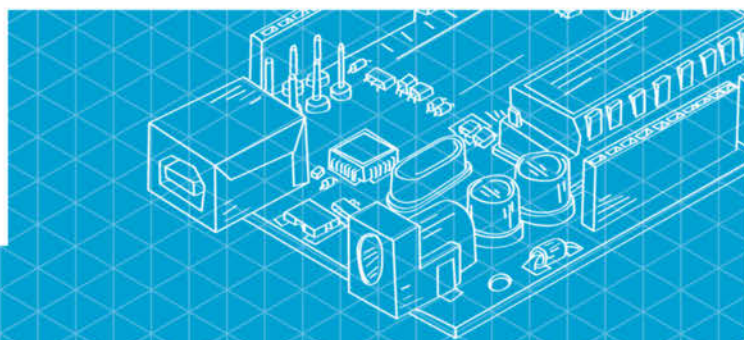
Mein erster Kontakt mit Computern war der ausrangierte Arbeitslaptop meines Vaters. Auf dem lange überholten Windows 98 entdeckte ich meine Liebe zu Point-and-Click-Adventures, die bis heute anhält.



Thomas Gamisch
Holzkirchen, hat für den 4. Mai eine besondere Anleitung auf S. 58 geschrieben

Das war ein selbst zusammengelöteter SC/MP-8-Bit-Computer aus dem Christiani Mikroprozessorlabor: 512 Byte(!) RAM, Siebensegmentanzeige, hexadezimale Tastatur, Maschinensprache, Thermodrucker.

Make:



DAS KANNST DU AUCH!



GRATIS!



2× Make testen und über 7 € 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 19,40 € statt 27 €

Jetzt bestellen:
make-magazin.de/miniabo



* Für die Laufzeit des Angebotes.



i 16:10-16-Zoll-Display und maximal großer 99-Wh-Akku. Kompatibel mit der TUXEDO Aquaris.

i GeForce RTX 4090, Core i9-13900 HX und eine mechanische Tastatur mit Cherry-MX-Schaltern.

Interstellare Spitzenleistung im Kompaktformat! **TUXEDO Stellaris 16 - Gen5**



CPU-Leistung



Grafikleistung



Mobilität



Akkulaufzeit



100%
Linux

5

Jahre
Garantie



Lifetime
Support



Gefertigt in
Deutschland



Deutscher
Datenschutz



Support
vor Ort

TUXEDO

 tuxedocomputers.com

© Copyright by Maker Media GmbH.