DEUTSCHLANDS GEFÄHRLICHSTES DIY-MAGAZIN

c't Make:

Make:

KI: Katzenklo-Überwachung mit RasPi und ESP32-CAM

Schlauer schalten mit Shelly

- Welches Modell für welchen Zweck
- Vor- und Nachteile
- So konfiguriert man sie
- Praxisbeispiel mit Wärmepumpe

Projekte

- Touchscreen für ESP32
- Lampen für Fernbeziehungen
- Taupunkt-Lüfter mit ZigBee

Workshops

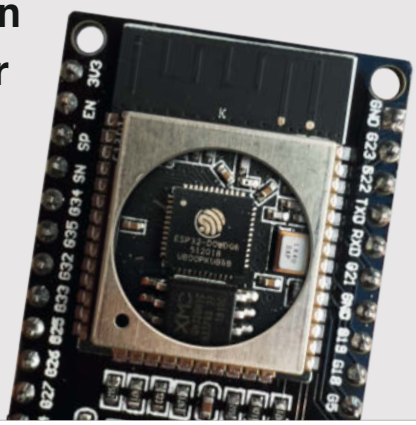
- GraviTrax mit ESP32 steuern
- Projekte auf GitHub verwalten
- Mehr Flash für Arduino Nano

Platinen selber machen

- KiCAD, Eagle und EasyEDA im Praxistest
- Vom Schaltplan bis zur Fertigung
- Für Ein- und Umsteiger

Durchblick im ESP32-Dschungel

- Welches Modell kann was
- SoCs, Module und DevKits verstehen
- Tipps zur Auswahl



2/25
4.4.2025
CH CHF 27.90
AT 15,90
Benelux 17,10
€ 14,50

4 192314 114507 02

www.make-magazin.de

Shutterstock, VZ_Art

Wider den Wissensfluch

Haben Sie schon mal vom Wissensfluch gehört? Dabei handelt es sich um eine kognitive Verzerrung, die auftritt, wenn eine Person mit Spezialwissen davon ausgeht, dass andere Personen ebenfalls über dieses Wissen verfügen. Dabei ist die Art des Spezialwissens unerheblich, es kann sich um technisches, naturwissenschaftliches, geisteswissenschaftliches oder wirtschaftliches Wissen handeln.

Viele Berufsgruppen leiden darunter: Lehrer, Professoren, Entwickler, Redakteure. Alle, die anderen etwas erklären wollen oder müssen. Ob man selbst verflucht ist, bemerkt man in der Regel, wenn man glaubt, irgendeine vermeintliche Banalität oder Selbstverständlichkeit zu erklären und man anschließend in die glasigen Augen seines Gegenübers schaut.

In der Make-Redaktion kämpfen wir beim Schreiben eines jeden Artikels mit dem Wissensfluch. Wir müssen uns immer wieder neu überlegen, für welches Niveau wir unsere Texte schreiben. Einsteiger, Umsteiger, Fortgeschrittener oder Profi? Welches Wissen können wir als gegeben voraussetzen? Welche Fragen stellen sich unsere Leser? Kollege Ákos hat sich diese Frage beim Schreiben der ESP32-Übersicht auf S. 16 häufig gestellt. Er hat als Übersetzer und Filter fungiert, um etwa Hardcore-Chipentwickler-Sprache auf Fortgeschrittenen-Niveau zu konvertieren.

Auch Sätze wie „Ein Treiber für einen virtuellen COM-Port ermöglicht die Kommunikation zwischen Mikrocontroller und Computer. Der Mikrocontroller verwendet dafür den CP210x-USB-zu-Seriell-Chip von Silicon Labs." stellen wir auf den Prüfstand. Überfordern sie den typischen Make-Leser oder fordern sie ihn eher auf, tiefer einzutauchen?

Arduino hat es vor 20 Jahren grandios geschafft (Happy Birthday übrigens!), aus dem hundertseitigen Datenblattsumpf der professionellen ATmega-Prozessor-Familie, proprietären Programmieradaptern und der Cross-Compiler-Toolchain ein leicht zu bedienendes Werkzeug für Anwender ohne spezielle Vorkenntnisse zu entwickeln.

Hernando Barragán, der Entwickler der Arduino-Software-Bibliothek Wiring, schreibt (siehe Link) über die Entstehung: „Ziel war es, Künstlern und Designern die Arbeit mit Elektronik zu erleichtern, indem komplizierte Details der Elektronik ausgeblendet werden, sodass sie sich auf ihre eigenen Zielvorstellungen konzentrieren können."

Um sich in die Bedürfnisse der Künstler und in deren Welt hineinzudenken, ist für einen Techniker einiges an Empathie nötig. Und weiter schreibt Barragán: „Die Syntax war das Ergebnis eines von mir gründlich durchgeführten Designprozesses, der Anwendertests mit Studenten, Beobachtung, Analyse, Anpassung und Iteration beinhaltete."

Was lernen wir daraus? Je mehr man sich bewusst mit seiner Zielgruppe beschäftigt, desto erfolgreicher ist man. Eigentlich eine Binsenweisheit. Ich würde mir wünschen, dass sich mehr Wissensvermittler mit dem Thema beschäftigen (müssen). Ich mag mir gar nicht vorstellen, wo wir heute in Schulen, Hochschulen und Unternehmen stehen würden, hätte es Arduino nicht gegeben.

Happy Teaching!

▶ **make-magazin.de/xty8**

Daniel Bachfeld

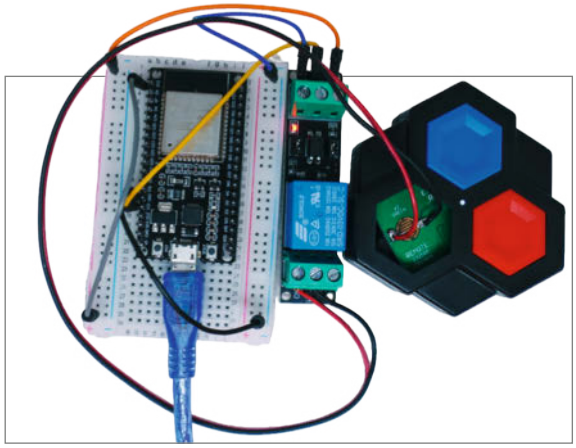
Schlauer schalten mit Shelly

Vor zehn Jahren brachte das bulgarische Unternehmen Allterco Robotics die ersten kleinen, robusten WLAN-fähigen Schaltrelais namens Shelly auf den Markt. Heute wird in über 100 Ländern eine sehr breite Palette von Unterputzrelais, Smart Plugs, Bewegungssensoren, Thermostatventilen und Smart Buttons verkauft. Wir erklären, wie dieses Smart-Home-Ökosystem zusammenhängt und wie es genutzt werden kann.

Inhalt

Projekte zum Verlieben

Manche Maker sind unsterblich in ihre Projekte verliebt. Damit die (wahre) Liebe zwischen Menschen nicht in Vergessenheit gerät, gibt es unsere LoversLamp - schicke Lampen, die über MQTT miteinander kommunizieren können. Und je mehr wir mit unserer automatisierten Kugelbahn spielen, desto intensiver blinkt und vibriert der herzförmige Herzschlaganzeiger.



KI für das Katzenklo

Mit RasPi, ESP-CAM und KI bauen wir ein System, mit dem festgestellt werden kann, welche Katze was auf der Toilette gemacht hat. Sehr praktisch, um bei kranken Katzen die richtige Kotprobe zu erwischen. Der Aufbau kann natürlich auch für andere kreative Anwendungen genutzt werden.

Platinen selber machen

Eine Platine am Computer zu entwerfen und fertigen zu lassen, ist gar nicht so schwierig und auch nicht mehr teuer. Aber welches Programm soll man lernen? Wir geben hier einen Überblick über drei sehr unterschiedliche E-CAD-Programme, die jeweils für eine ganze Klasse von Software stehen.

Themen von der Titelseite sind rot gesetzt.

ESP-Welt durchblicken

Die Chips von Espressif haben die Herzen vieler Maker erobert, aber wer ein Projekt mit dem ESP32-Mikrocontroller plant, sieht sich mit einer riesigen und unübersichtlichen Produktpalette konfrontiert. Offizielle Regeln für Form und Namensgebung gibt es nicht. Wir klären auf, was die Mitglieder der ESP32-Familie im Wesentlichen verbindet und was sie voneinander unterscheidet.

Leserforum

Projekte veröffentlichen, Freie Software und Programmieren

Make 1/25, S. 3

Als langjähriger Leser habe ich heute gleich mehrere Sachen auf dem Herzen. Zum Ersten: Ihr Editorial hat mich zum Nachdenken bewegt. Freie Software nutze ich häufig, Firefox, Thunderbird, Arduino und Co. gehören zu meinem Leben dazu. Über Bugs oder Ähnliches schimpft man, ohne zu realisieren, dass zum Teil freie Mitarbeiter in ihrer Freizeit daran arbeiten. Heute habe ich nach dem Download einer Freeware entsprechend einen Beitrag gespendet, das kommt jetzt vielleicht öfter vor.

Zum Zweiten: Das Thema Programmieren wird sowohl in der c't als auch in der Make gerne angesprochen. Dabei werden Plattformen wie die Arduino IDE häufig als Low Level und für Einsteiger dargestellt. Das mag zutreffen, aber ist es wirklich sinnvoll, eine kleine, feine IDE, mit der man gut zurechtkommt, durch eine leistungsfähigere, aber auch komplexere zu ersetzen?

Ich persönlich habe mir Visual Studio Code mit Arduino und auch PlattformIO angesehen. Um mal schnell was zu testen, falle ich immer wieder auf die Arduino IDE zurück. Mit Freude sogar. Die meisten Maker programmieren eher zu Hobbyzwecken. Ob man sich da einen professionellen Touch geben muss?

Und zuletzt: Ihr schreibt in der Make, dass gerne eigene Projekte zur Veröffentlichung gesucht werden. Dazu ein paar Fragen:

- Wie fertig muss ein Projekt sein, damit es veröffentlicht werden kann?
- Wie gut muss die Beschreibung formuliert sein, damit sie drucktauglich ist?
- Muss ein fertiges Projekt zur Veröffentlichung in der Make ein zweites Mal aufgebaut werden, um alle Bauschritte zu dokumentieren?
- Dürfen Libraries oder 3D-Druckvorlagen mitveröffentlicht werden, auch wenn ein Dritter diese entwickelt hat?

Ansonsten weiter so. Nicht jedes Projekt spricht mich an, aber alles in allem ist eure Arbeit okay.

HP Dietrich

Danke für das Lob und die kritischen Anmerkungen. Zum Thema Projekteinreichungen: Von der einfachen Idee bis hin zum fertigen Projekt freuen wir uns über jede Anregung und jeden Vorschlag. Vielleicht hängt das Projekt ja mit einem ähnlich gelagerten Beitrag aus einer Make zusammen? Dann schreibt den betreuenden Redakteur an, der am Kürzel am Ende des Beitrags zu erkennen ist. Die E-Mail-Adressen der Redakteure haben die Form xx@make-magazin.de oder xxx@make-magazin.de. Ansonsten schreibt zentral an die Make-Redaktion unter info@make-magazin.de. Wir setzen uns dann mit euch in Verbindung und begleiten die Entstehung des Beitrags bis zur Druckreife. Dazu gehört auch, dass wir rechtliche Dinge in Zusammenhang mit der Veröffentlichung abklären.

Platzsparende Koffer-Projekte

Make 1/25, S. 92 u. Make 7/24, S 74

In den letzten beiden Heften waren zwei schöne Koffer-Projekte (uLisp-Box und Admin-Koffer). Gerade wenn man etwas weniger Platz zu Hause hat und sein Heim noch mit der Familie teilt, muss man seine Hobbys, wenn möglich, in platzsparende Behältnisse packen. Die sollten aber auch schnellen Zugriff gewähren, denn die knappe Zeit soll ja nicht für Auspack- und Kabelsteckorgien draufgehen. Mein favorisiertes Format sind L-Boxxen. Da habe ich beispielsweise eine mobile Elektronikreparaturkiste oder eine für Fahrradreparaturen, die immer mit im Auto ist, wenn's mal weiter weg den Berg runtergeht. Meine neueste Box hat jetzt auch einen Bezug zu den beiden Koffer-Artikeln der letzten beiden Hefte. Die LAN-Party-Box geht in eine ähnliche Richtung, wenn auch für ein spielerisches Thema. Zum Inhalt: Mini-PC als Server, der kleine TP-Link für die Netzwerkverwaltung, ein 16-Port-Switch, 7"-Bildschirm, Tastatur / Touchpad … Details findet ihr unter dem Link https://ding-ding.de/wp/lan-party-box/. Alles drin, um direkt loszuspielen. Ich freue mich auf weitere tolle Hefte und eventuell noch den einen oder anderen Koffer.

Marco Ding

Variante vom Filamentschrank

Make 1/25, S. 32

Danke für den Artikel über den Filamentschrank. Ich habe ihn in einer schmaleren Variante nachgebaut, Infos siehe Kurzlink. Unter dem Kurzlink sind auch Fotos des Nachbaus des Erlenmeyerkolbens mit Galvoscanner enthalten. Ich habe das Regal mit Silikon und Montagekleber abgedichtet. Im Türrahmen sitzt eine Dichtung. Die Tür wird durch den Riegel in die Dichtung gedrückt. Der Einschalttaster im Entfeuchter wurde überbrückt (die ganze Elektronik ist rausgeflogen), sodass er über ein Relais eingeschaltet werden kann. Zusätzlich habe ich noch einen weiteren Lüfter verbaut, um den Luftdurchsatz zu erhöhen. Leider reicht die Entfeuchterleistung nicht aus, sodass ich zusätzlich noch zwei Behälter mit regenerierbarem Silicat ergänzt habe. Der ESP32 verfügt über eine MQTT-Schnittstelle zur Heimautomatisierung FHEM. Es werden die Messwerte der beiden Sensoren, das Erreichen der Grenzwerte und Einschaltbefehle für die Beleuchtung und den Entfeuchter übertragen. Sobald die Solaranlage genug Leistung erzeugt, wird der Entfeuchter eingeschaltet, und ein Bewegungsmelder schaltet die Beleuchtung ein beziehungsweise aus. Durch FHEM ist auch eine Schnittstelle zum Schalten über Alexa vorhanden. Ich freue mich auf weitere Projekte, die eine gute Kombi aus Mechanik, Konstruktion, 3D-Druck, Elektronik und Software benötigen. Grüße aus Herford!

Uwe Scheffer

Veraltete Installationsbefehle

Make 1/2021, S. 28

Die im Artikel „Nistkasten 2.0" aufgeführten Installationsbefehle sind leider veraltet. Da ich nicht so im Programmieren bewandert bin, wäre ich sehr froh, wenn ich bezüglich dieser Installationsbefehle Unterstützung bekommen könnte. Wäre es möglich, mit dem Autor dieses Artikels Kontakt aufzunehmen? Auch wenn der Artikel schon etwas veraltet ist, kann er mir vielleicht doch noch helfen.

Jeffrey Erxmeyer

Wenn pip install in der aktuellen OS-Version den Fehler „error: externally-managed-environment" ausgibt, kann man diesen durch die Ergänzung `--break-system-packages` *einzeln für jedes Package übergehen – oder mit dem Befehl* `sudo rm -rf /usr/lib/python3.11/EXTERNALLY-MANAGED` *dauerhaft ausschalten.*

Sie müssen bei letzterem aber schauen, ob Sie aktuell python3.11 oder eine andere Version haben.

iOS-Apps über App-Inventor

Make 1/25, S. 112

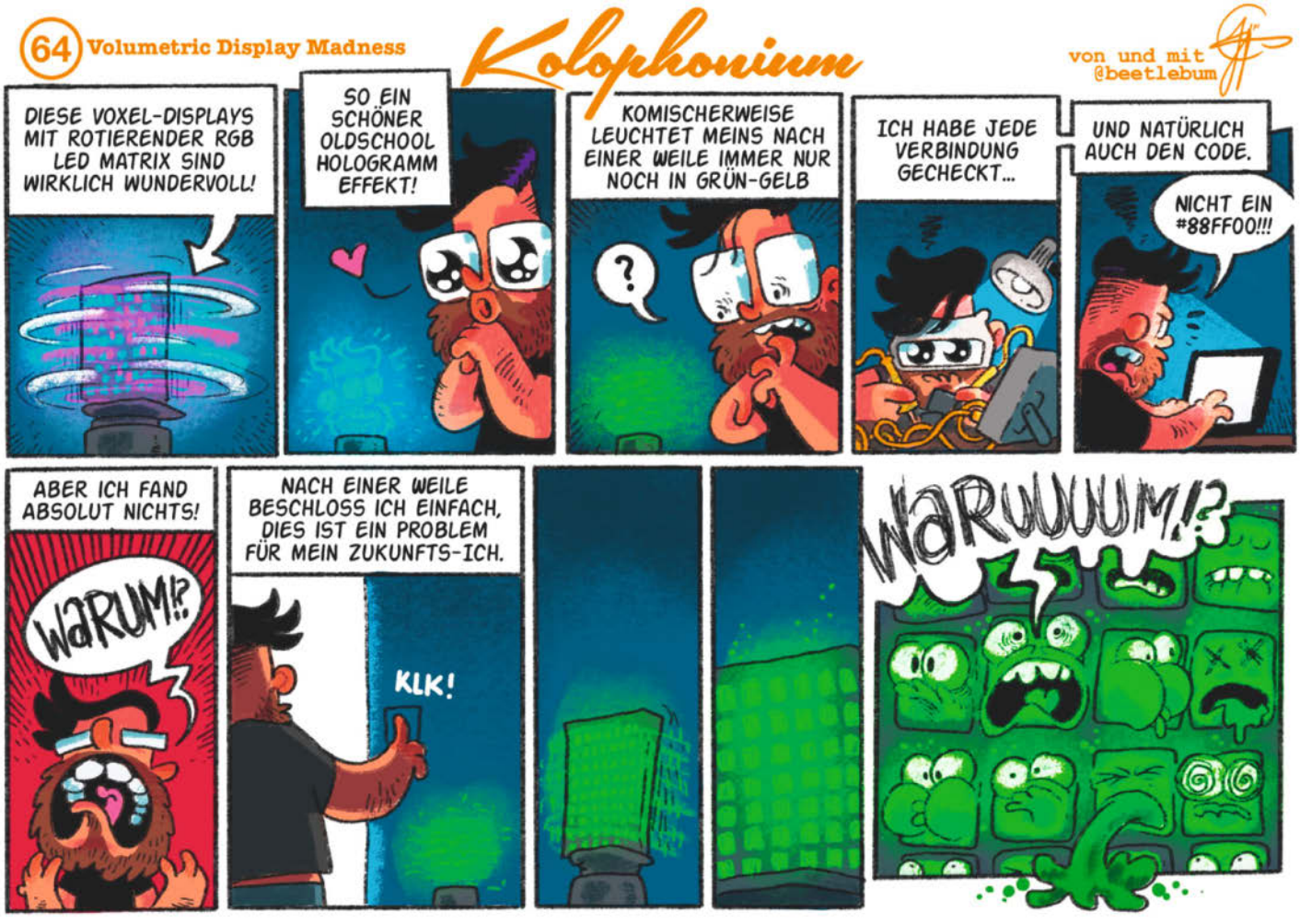
Im Beitrag „Arduino-Steuerung über App-Inventor-Apps“ wird leider nur am Rande auf die Möglichkeiten eingegangen, native iOS-Apps mit dem MIT App Inventor zu erstellen. Die Nutzung der MIT App Inventor-App aus dem App Store unter iOS ist analog der Beschreibung im Beitrag für Android möglich. Diese Brückenlösung ist sicher zu Testzwecken auch recht brauchbar, aber kein Ersatz für echte eigenständige iOS-Apps. Wie native iOS-Apps erstellt werden, darauf geht der Beitrag jedoch nicht weiter ein. Deshalb habe ich selbst etwas experimentiert.

Auch wenn das Erstellen einer eigenständigen .ipa-Datei (der nativen iOS-App-Datei) aktuell noch kompliziert ist, gibt es einen dokumentierten Weg dafür. Dazu muss die Mac-OSX-Version vom MIT App Inventor installiert werden. Neben einem Apple-Entwickleraccount sind einige zusätzliche Schritte auszuführen, die auf der MIT-Website beschrieben sind (siehe Kurzlink).

Nach dem Signieren der App besteht die Möglichkeit, diese auf den Geräten zu installieren, die mit dem Entwicklerprofil verknüpft sind. Es kann ein Ad-hoc-Build zu lokalen Testzwecken erstellt oder die App in den App Store hochgeladen werden. Wenn der Build im App Inventor abgeschlossen ist, erscheint ein QR-Code zum Scannen. Diesen kann man beispielsweise über die Kamera-App oder spezielle QR-Code-Apps anvisieren. Danach sollte eine Benachrichtigung zum Öffnen eines Links erscheinen. Nach dem Öffnen dieses Links startet die Installation der nativen App unter iOS.

Ulrich Kaiser

▶ **make-magazin.de/xg5u**



Shelly, der Alleskönner

Vor zehn Jahren brachte das bulgarische Unternehmen Allterco Robotics mit Sitz in Sofia die ersten kleinen, robusten WLAN-fähigen Schaltrelais namens Shelly auf den Markt. Heute wird in über 100 Ländern eine sehr breite Palette von Unterputzrelais, Smart Plugs, Smart Lamps, Bewegungssensoren, Thermostatventilen und Smart Buttons verkauft. In diesem Artikel erklären wir, wie dieses Smart-Home-Ökosystem zusammenhängt und wie es genutzt werden kann.

von Marcus Hansson

Kurzinfo

» Vor- und Nachteile von Shelly
» Die beliebtesten Shelly-Geräte kurz erklärt
» Anwendungsbeispiel: Überschüssigen PV-Strom mit Shelly zur eigenen Wärmepumpe leiten

Checkliste

Zeitaufwand:
8 Stunden

Kosten:
100 Euro

Material

» **Shelly 3EM** dreiphasiger WLAN-Energiezähler
» **Shelly Plus 1PM** Relaisschalter mit Leistungsmessung
» **2 Shelly Plus 1** Relaisschalter mit potentialfreien Kontakten

Mehr zum Thema

» Heinz Behling, Funktechniken fürs Smart Home, Make 5/24, S. 116
» Uwe Rohne, Vom Energiemesser zum Smart Meter, Make 1/25, S. 108
» Heinz Behling, Intelligentes Heim mit Home Assistant, Make 1/21, S. 100

Alles zum Artikel im Web unter
make-magazin.de/xb37

Kaum eine andere Smart-Home-Lösung scheint in den letzten Jahren so schnell gewachsen zu sein wie Shelly. Diesen Eindruck vermittelt der bulgarische Hersteller von smarten Steckdosen & Co. auch auf seiner eigenen Homepage. Dort zeigt ein Echtzeit-Dashboard, wie viele der weltweit mehr als zwanzig Millionen verkauften Shelly-Geräte gerade im Einsatz sind (Bild 1). Als ich dort hineinschaute, waren laut Dashboard in der letzten Minute über 45.000 Lichter und mehr als 900.000 Relais eingeschaltet. Pro Minute seien etwas mehr als 200 MWh gemessen worden. Und über einen „News-Ticker" werden die neu hinzugekommenen Shellys aufgelistet: Shelly Plus 1, Plus 2PM, BLU HT, Pro 4 DM, 1PM Mini Gen 3 und so weiter.

Kompatibilität steht im Vordergrund

Ein starkes Argument für Shelly, mit dem der Hersteller wirbt, ist, dass es ein sehr offenes und flexibles System ist, das eine hohe Kompatibilität bietet. Es wird großer Wert auf Benutzerfreundlichkeit gelegt und darauf, dass die Geräte sehr schnell und einfach einzurichten sind. Alle Shellys kommunizieren über WLAN und können ohne Vorkenntnisse schnell und einfach über die App eingerichtet werden. Geräte mit einem Plus im Namen bieten einen leistungsstärkeren ESP32, haben dafür aber nur 4 statt 8 MB Flash-Speicher. Shelly-Geräte sind mit den meisten Heimautomatisierungsplattformen, Protokollen und Sprachassistenten kompatibel. Seit 2022 unterstützt Shelly auch den Smart-Home-Standard Matter, der von Branchengrößen wie Google, Apple, Samsung und Amazon ins Leben gerufen wurde, um die Interoperabilität zwischen Smart-Home-Geräten verschiedener Hersteller zu ermöglichen.

Vierte Generation

Anfang des Jahres hat Shelly auch die vierte Generation seiner Geräte vorgestellt. Erstmals unterstützt Shelly nun auch ZigBee und Wi-Fi 6. Bluetooth Low Energy (BLE) ist jetzt standardmäßig integriert und ein Add-on für die Integration des LoRa-Netzwerkprotokolls ist laut einer Pressemitteilung geplant. Damit können Geräte Daten über eine Distanz von bis zu fünf Kilometern senden und empfangen.

Kein Cloudzwang

Das zweite starke Argument, mit dem Shelly gerne wirbt, ist der fehlende Cloudzwang. Zwar gibt es die App, mit der sich die Geräte in der Regel auch ohne eigene Smart-Home-Server sorgenfrei konfigurieren und steuern lassen, doch wer möchte, kann die Geräte auch mit anderen Smart-Home-Systemen wie Home Assistant nutzen. Dafür braucht man dann allerdings einen Smart-Home-Server, zum Beispiel einen Raspberry Pi. Wer im Shelly-System bleibt, braucht nur die App, um seine Geräte zu steuern.

Generell kann man sagen, dass die Shelly-Produkte von Haus aus für Eigeninitiativen von Bastlern und Entwicklern gut geeignet sind. Sie bieten eine REST-API, sodass die Steuerung

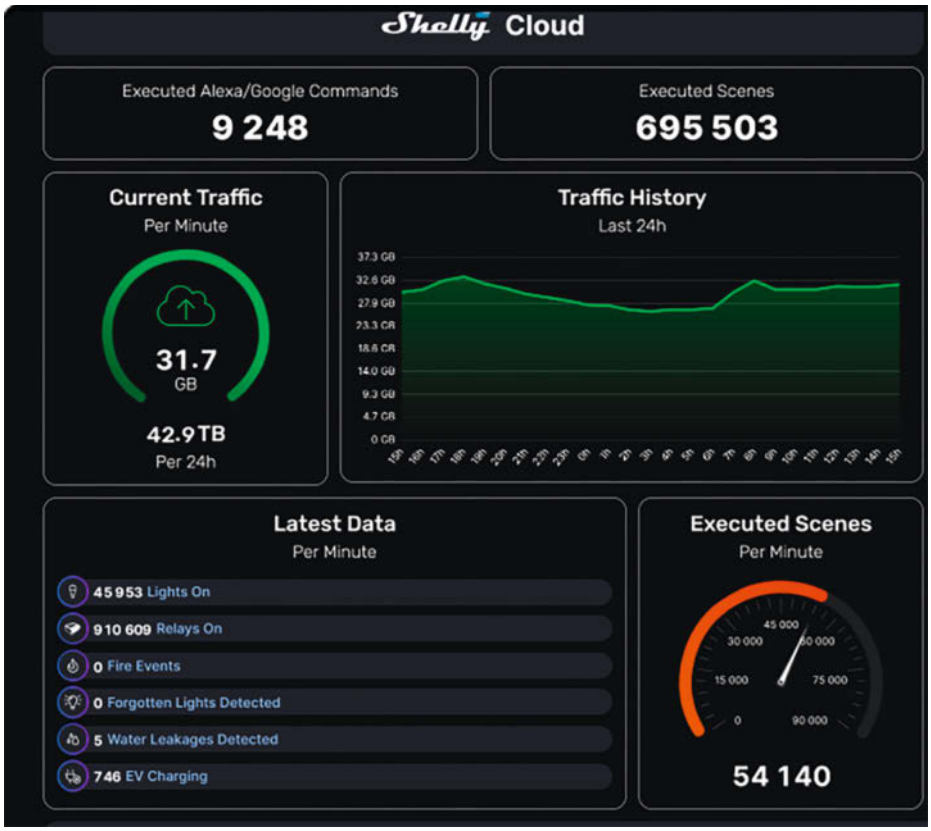


Bild 1: Wie verschiedene und weltweit angeschlossene Shelly-Geräte aktuell eingesetzt werden, zeigt ein Online-Dashboard auf der Shelly-Homepage.

und Abfrage von Shelly-Geräten über einfache HTTP-Anfragen möglich ist. Außerdem steht es den Nutzern frei, die Firmware einfach per Over-the-Air-Update auszutauschen, zum Beispiel mit der Open-Source-Alternative Tasmota. Damit ist man völlig unabhängig von der Cloud des Herstellers. Shellys können auch ohne Internet betrieben werden, indem Skripte direkt auf dem Gerät ausgeführt werden.

Sehr breite Produktpalette

Damit es nicht gleich nach Werbung klingt, hier ein Kritikpunkt: Das Shelly-Ökosystem ist fast schon *zu* umfangreich. Einerseits richtet sich die Marke an Endkunden, die möglichst wenig tüfteln wollen. Das sind Leute, die ganz simpel einen smarten Schalter in die Schuko-Steckdose stecken und ihn einfach über die Shelly-App steuern möchten, während sie eine Auswahl an Möglichkeiten in verständlicher Sprache präsentiert bekommen (Bild 2).

Auf der anderen Seite des Spektrums gibt es die Profi-Serie mit unzähligen Energiemessgeräten und Schaltern für die Hutschienenmontage. Damit lassen sich beispielsweise Restaurants, Lagerhallen, öffentliche Außenbeleuchtungsanlagen oder große Photovoltaikanlagen überwachen und steuern.

Dazwischen gibt es die mittlerweile ebenfalls sehr zahlreichen Unterputz-Schaltrelais, die wir auch hier auf dem Titelbild sehen. Diese können nicht nur Lampen oder Garagentore steuern, sondern auch LED-Streifen zum Leben erwecken, Sensoren auslesen oder Szenen aktivieren oder deaktivieren. Es gibt auch sogenannte Add-on-Module, die z. B. auf der Rückseite einer Shelly Plus 1PM (Bild 3) angebracht werden. Damit können verschiedene Sensoren direkt an die Shelly angeschlossen werden, um Temperatur, Feuchtigkeit, Licht usw. zu messen.

Dazu können wir auch die Shelly-X-Serie aufsetzen. Das sind kleine Entwicklerboards, die man für ein paar Euro bestellen kann, um schnell zu testen, wie man sie in eigene Produkte integrieren kann. Da verliert man schnell den Überblick und es bedarf einiger Recherche, um zu verstehen, was für die eigenen Zwecke am besten geeignet ist. Konkrete und aussagekräftige Informationen über den Unterschied zwischen „Plus“ und „normalen“ Shellys oder den Unterschied zwischen PM (Power Meter) und EM (Energy Meter) sind schwer zu finden. In dem Kasten „Die beliebtesten Shelly-Geräte“ stellen wir daher die wichtigsten Shellys und ihre grundlegenden Funktionen vor.

Abo für erweiterte Funktionalität

Ein weiterer negativer Punkt ist, dass nicht alle über die App angebotenen Funktionen kostenlos sind. Seit August 2023 bietet Shelly ein Premium-Abo für 35,99 Euro pro Jahr an, mit dem erweiterte Funktionen freigeschaltet werden können (Bild 4). Das sogenannte Shelly Smart Control Premium kann auch monatlich für 3,99 Euro gebucht werden. Nur mit Premium ist die Steuerung von Funktionen über Wetter-App-Daten möglich, die sogenannte wetterbasierte Szenenausführung. Auch die Datenvisualisierung ist ohne Premium-Abo sehr eingeschränkt: Basic zeigt Daten nur in Stundenauflösung an, Premium in Minutenintervallen. Zum Premiumangebot gehört auch ein monatlicher Energiebericht.

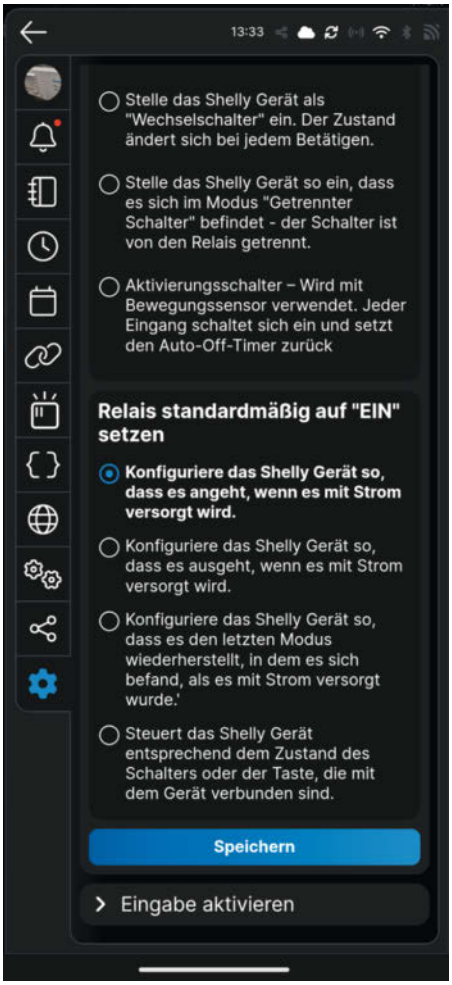


Bild 2: Bei der Gerätekonfiguration in der App werden die Optionen einfach und übersichtlich dargestellt.

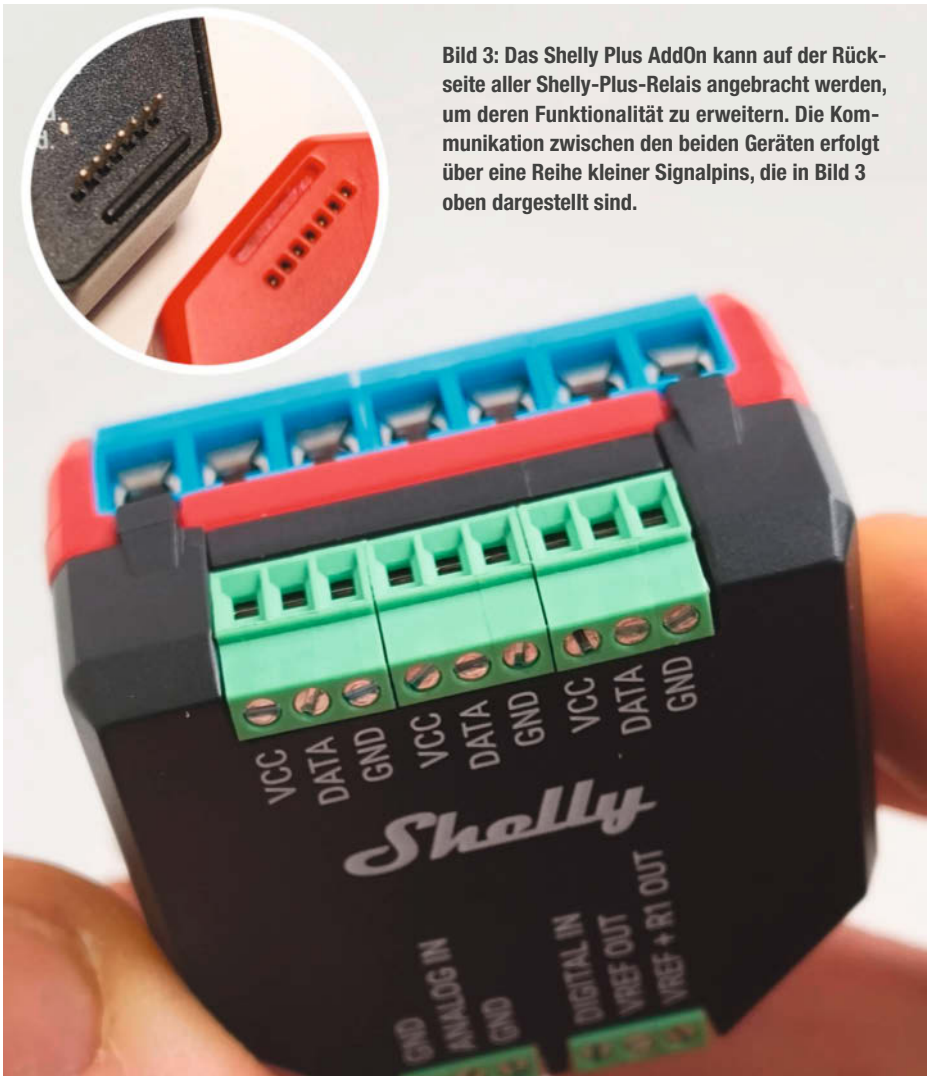


Bild 3: Das Shelly Plus AddOn kann auf der Rückseite aller Shelly-Plus-Relais angebracht werden, um deren Funktionalität zu erweitern. Die Kommunikation zwischen den beiden Geräten erfolgt über eine Reihe kleiner Signalpins, die in Bild 3 oben dargestellt sind.

Die beliebtesten Shelly-Geräte

Shelly 1 Gen 3

Shelly 1 Gen 3 ist die Standardwahl: Der 1-Kanal-Aktor schaltet einen Verbraucher ein und aus und versorgt ihn im Betrieb mit bis zu 3500 Watt (16 A). Die Hardware akzeptiert 240 Volt Wechselspannung oder 12 bis 60 Volt Gleichspannung. Wie alle Shelly-Aktoren ist auch dieser sehr kompakt. Mit Maßen von 37 mm × 42 mm × 16 mm verschwindet er selbst in kleineren Schaltdosen.

Shelly 1PM

Zum Schalten eines Verbrauchergerätes und zur Messung des Energieverbrauchs dient die Variante Shelly 1PM, deren Leistungsdaten weitgehend denen von Shelly 1 entsprechen. Wer zwei Ausgänge benötigt, kann auf das Modell Shelly 2PM zurückgreifen. Es schaltet und misst den Energieverbrauch von zwei Kanälen mit einem Leistungsbedarf von jeweils bis zu 2300 Watt (10 A). Damit lassen sich z. B. zwei Lampen über einen Aktor steuern oder ein Rollladen mit bidirektionalem Motor punktgenau heben und senken.

Shelly Plug S Gen3

Ein Plug-&-Play-Stecker für die vorhandene Steckdose mit Leistungsmessung und Matter-Kompatibilität. Durch die sehr kleine Bauform werden benachbarte Steckdosen nicht blockiert. Er dient zum Schalten von Beleuchtung oder anderen an die Steckdose angeschlossenen Geräten.

Shelly Plus AddOn

Das Shelly Plus AddOn ist eine Erweiterung für alle Shelly-Plus-Relais und kann auf deren Rückseite montiert werden, um die Funktionalität zu erweitern. Die Kommunikation zwischen den beiden Geräten erfolgt über eine Reihe kleiner Signalpins, die in Bild 3 oben dargestellt sind. Es bietet galvanisch getrennte Sensorschnittstellen zur Messung von Temperatur und Feuchte oder zum Anschluss verschiedener Sensoren.

Shelly Plus RGBW PM

Mit diesem Unterputzmodul können bis zu vier 12- oder 24-V-DC-LED-Streifen (farbig und weiß) per Handy oder Tablet gesteuert werden. Unterstützt werden die beiden gängigen Farbsysteme RGBW und RGB.

Shelly Plus i4 DC

Der Shelly Plus i4 DC wird mit 5 bis 24 V Gleichstrom betrieben. Das Gerät mit vier Ausgängen ist für komplexe Auslöseszenarien ausgelegt und kann verwendet werden, um andere Modelle zu steuern.

Was bedeutet „Plus"?

Viele Geräte gibt es auch mit einem „Plus" im Namen. Die Plus-Linie hat einen schnelleren ESP32-Chip und soll höhere Sicherheit bieten. Bei etwas älteren Shellys waren nur die Plus-Geräte mit Bluetooth ausgestattet. Viele, aber nicht alle Plus-Geräte erkennt man auch an der blauen Farbe der Schraubblöcke.

Was bedeutet „Mini"?

Einige Shellys gibt es auch mit der Zusatzbezeichnung „Mini". Die Mini-Geräte sind nur 29 × 34 × 16 mm groß – im Vergleich zu den „normalen" Shelly Plus 1, die 37 × 42 × 16 mm groß sind. Dafür können die Kleinen auch nur bis zu 2000 W Wechselstrom oder 150 W Gleichstrom schalten. Die „Normalen" schaffen 3840 W AC (16 A) oder 300 W DC (10 A).

Und dann auch noch Wave-Geräte!

Es gibt auch eine Reihe von Geräten mit dem Namenszusatz „Wave". Z-Wave ist ein drahtloses Kommunikationsprotokoll, das speziell für die Heimautomatisierung und Smart-Home-Anwendungen entwickelt wurde. Voraussetzung dafür ist ein Z-Wave-fähiger Smart Hub. Der Shelly Wave 1 entspricht von der Funktionalität her in etwa dem Shelly 1.

Für mich persönlich fühlt sich dieses Abo-Modell so an, als ob ich nach dem Kauf eines neuen Autos ein Abo abschließen muss, um die Sitzheizung zu aktivieren. Es wird oft mit Energieeinsparung für die Heimautomatisierung argumentiert. Wenn ich die Hardware erst kaufen und dann auch noch 3 bis 4 Euro im Monat zahlen muss, dann dauert es sehr lange, bis sich die Investition amortisiert hat. Nun ist es natürlich jedem selbst überlassen, ob er die Shelly-App nutzt oder nicht. Aber das ist ja der große Vorteil – wie schnell die Teile erkannt werden und wie einfach ich sie einbinden kann.

Alternativ kann man auch auf die Shelly-App und das Abo verzichten und zum Beispiel den Home Assistant nutzen. Dann braucht man aber wieder einen Raspberry Pi oder Ähnliches, was mit zusätzlichen Kosten verbunden ist.

Mit Shelly Wärmepumpe und PV steuern

Um Shelly weiter vorzustellen, haben wir uns ein Anwendungsbeispiel ausgedacht. Ich habe zu Hause eine Wärmepumpe und eine Photovoltaikanlage. Leider hatten die bisher keine Möglichkeit, miteinander zu kommunizieren. Mit einigen Shellys soll sich das jetzt ändern. Meine Wärmepumpe, ein 5-kW-Panasonic-Monoblock, bietet über eine Zusatzplatine SG-Ready-Funktionalität. Über die SG-Ready-Schnittstelle können Befehle an die Wärmepumpe gesendet werden: Steht etwa viel PV-Strom zur Verfügung, kann man diesen dann nutzen, um mehr wärmeres Wasser als im Normalbetrieb zu erzeugen.

Der erste Schritt ist die Messung des solaren Überschusses. Das machen wir mit dem Shelly

Bild 4: Wer das volle Potenzial von Shelly nutzen möchte, kommt um ein Premium-Abo nicht herum.

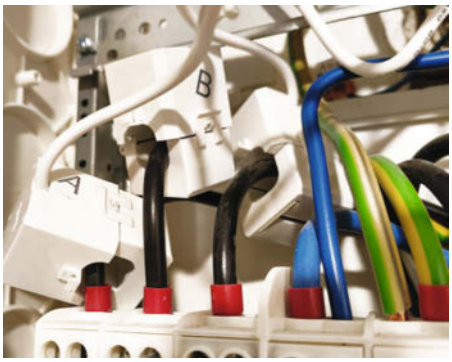


Bild 5: Die sogenannten Splitcore-Stromwandler sind aufklappbar, sodass sie leicht an die Kabel angeschlossen werden können.

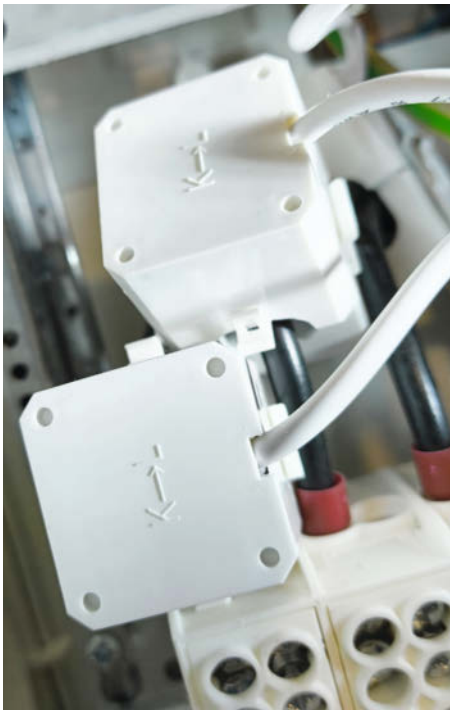


Bild 6: Der Pfeil auf dem Stromwandler muss in die richtige Richtung zeigen.

Bild 7: Startseite des EM3

3EM. Dabei handelt es sich um einen dreiphasigen Energiezähler mit Amperezangen (Stromwandler), der mit wenig Aufwand an die drei ankommenden Phasen im Verteilerschrank angeschlossen werden kann (Bild 5). Die Stromwandler werden geöffnet, um den Phasenleiter gelegt und wieder geschlossen.

Das Modul, an das die Kabel der Stromwandler angeschlossen werden, benötigt nur einen Modulplatz (18 mm) im Schaltschrank. Es sind keine drei zusätzlichen Sicherungen erforderlich, um das Modul zu installieren, aber es sollte abgesichert werden. Dafür können z. B. die drei Sicherungen des Herdes verwendet werden.

Es ist sehr wichtig, dass die Stromwandler in der richtigen Richtung montiert werden. Auf jeder der drei Klemmen ist ein Bild aufgedruckt, leider schwer zu erkennen, da es nur erhaben ist (Bild 6). Wofür K steht, ist mir nicht ganz klar, aber „kommend" ergibt als Eselsbrücke Sinn.

Die Kabel von den Stromwandlern sind auch nicht besonders lang. Bei mir hat es gereicht, aber ggf. muss eine Sicherung versetzt werden, damit das Modul nicht zu weit von den Ampereklemmen weg ist, was zusätzliche Arbeit bedeutet.

Die Installation hat insgesamt keine halbe Stunde gedauert. Nach dem Einschalten habe ich zunächst versucht, das Teil über Bluetooth zu konfigurieren, was aber nicht funktionierte. Die Bedienungsanleitung war wenig hilfreich und der Text mit gefühlten submillimeterkleinen Buchstaben schwer zu lesen. Dort steht nur: „Wenn Sie das Gerät mit der Shelly Smart Control App und unserem Cloud Service bedienen und steuern möchten, finden Sie eine Anleitung in der Bedienungsanleitung der mobilen App." Wo diese Anleitung zu finden ist, wird nicht erwähnt. Aber nach ein paar Minuten ungeduldigen Herumprobierens in der App wird der 3EM plötzlich über WLAN gefunden. Ich gebe mein WLAN-Passwort ein

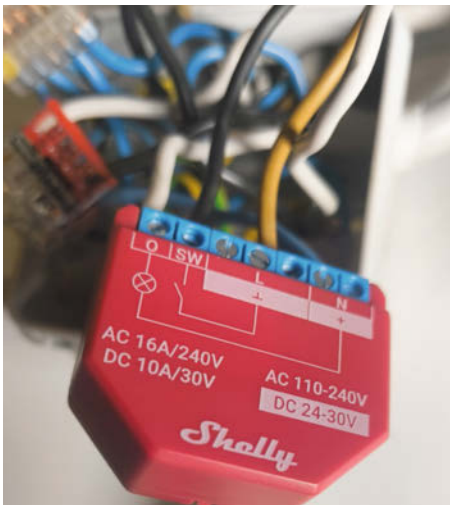


Bild 8: Shelly Plus 1PM kann unter anderem den Energieverbrauch überwachen.

und das war's. Der 3EM erscheint im Control Panel in der Shelly App auf meinem Handy (Bild 7) und zeigt unter anderem die aktuellen Verbrauchswerte an. Gut.

Überwachung des Energieverbrauchs der Wärmepumpe

Der nächste Schritt ist die Installation eines Shelly Plus 1PM, um den Verbrauch der Wärmepumpe überwachen zu können. Dies ist nicht unbedingt notwendig für das Funktionieren meines Anwendungsbeispiels; ich mache es nur, damit ich später live sehen kann, wie viel mehr Strom die Wärmepumpe verbraucht, wenn der SG-Ready-Ausgang eingeschaltet ist oder nicht.

Meine Wärmepumpe läuft einphasig, also ist auch hier die Installation unkompliziert und in wenigen Minuten in einer Verteilerdose im Keller erledigt, wo eine Leitung nur zur Wärmepumpe führt (Bild 8). Der Anschluss ist diesmal einfacher als bei der 3EM. Sobald der Strom nach der Installation wieder eingeschaltet wird, meldet sich die Shelly Plus 1PM über die App und möchte eingebunden werden. Diesmal muss ich nicht einmal das Passwort eingeben – alles läuft automatisch (Bild 9).

Was ist SG Ready

Nun wollen wir uns der SG-Ready-Funktion in der Wärmepumpe widmen. SG Ready ist ein Standard für Wärmepumpen in Deutschland, Österreich und der Schweiz, der es ermöglicht, überschüssigen Solarstrom für die Wärmepumpe zu nutzen (Bild 10). SG steht für Smart Grid und soll auch die Netzstabilität unterstützen, indem mehr Solarstrom lokal genutzt wird, statt ins Netz eingespeist zu werden (siehe auch Kasten „Ferndimmung der Wärmepumpe"). Der Standard wurde vom Bundesverband Wärmepumpe und 17 Wärmepumpenherstellern entwickelt und ist auf der Homepage des Bundesverbandes Wärmepumpe ausführlich beschrieben (Link in der Kurzinfo). Der Standard ist ab 2024 auch Voraussetzung für die staatliche Förderung einer Wärmepumpe.

SG Ready bietet vier Betriebszustände. Welcher davon aktiviert wird, wird durch zwei Bits gesteuert, d. h. Kontakte auf der Platine in der Wärmepumpe, die entweder 0 oder 1 sein können – je nachdem, ob 12 V angeschlossen ist oder nicht (Bild 11). Mit zwei Bits haben wir vier mögliche Werte, die in der Betriebsanleitung meiner Wärmepumpe wie in der Tabelle „SG-Ready-Steuerung" aufgeführt sind.

SG Ready einrichten

Um die Bits zu setzen, verwenden wir zwei Shelly Plus 1, die über einen potentialfreien Kontakt verfügen, d. h. einen Schalter, der von den anderen Schaltkreisen elektrisch

Bild 9: Neue Geräte werden in der Regel automatisch gefunden und können mit wenigen Klicks in die App eingefügt werden.

Bild 10: SG Ready vom Bundesverband Wärmepumpe

isoliert ist. Es können auch andere Shellys verwendet werden, z. B. Shelly Plus Uni, das für Niederspannung ausgelegt ist.

Die beiden Shelly Plus 1 werden über Wago-Klemmen an 230 V angeschlossen (Bild 12). Mit einem Multimeter wie im Bild kann getestet werden, ob sie auch richtig schalten.

Auch hier verlief die Installation zunächst ohne Probleme. Erst als ich die Stahlblechabdeckung der Wärmepumpe wieder angeschraubt hatte, bemerkte ich, dass die beiden Shelly Plus 1 nicht mehr online waren. Meine Wärmepumpe steht vor der Hauswand an einer Stelle, wo das WLAN-Netz zu schwach war. Das war sehr ärgerlich. Zuerst habe ich einen Fritz-Repeater in eine etwa vier Meter entfernte Steckdose gesteckt – das Signal war immer noch zu schwach. Erst nachdem ich eine Steckdose in der Wärmepumpe installiert und den Fritz-Repeater dort angeschlossen hatte, funktionierte es. Hier wäre eine von den ganz neuen Gen-4-Shellys mit ZigBee wünschenswert, vielleicht hätte das besser funktioniert.

Szene einrichten

Je nachdem, wie viel PV-Strom zur Verfügung steht, können wir auf einen der beiden Hoch-

Ferndimmung der Wärmepumpe

SG Ready kann aber auch genutzt werden, um dem Energieversorger die Möglichkeit zu geben, Wärmepumpen aus der Ferne zu drosseln, wenn beispielsweise das Stromnetz überlastet ist. Dies ist ab Anfang 2024 für neu installierte Wärmepumpen mit einer Leistung von mehr als 4,2 kW Pflicht. Wärmepumpen und Wallboxen dürfen dann für maximal zwei Stunden am Stück auf 4,2 kW gedrosselt werden.

Im Gegenzug für die Möglichkeit, die eigene Wärmepumpe ferngesteuert zu drosseln, erhält der Eigentümer eine Reduzierung der Netzentgelte. Dafür gibt es verschiedene Modelle. Wer sich näher informieren möchte, kann in einer beliebigen Suchmaschine „§ 14a EnWG“ oder „Einbindung steuerbarer Verbrauchseinrichtungen“ eingeben.

Die SG-Ready-Schnittstelle an sich bietet nur die Möglichkeit, die Wärmepumpe lokal anzusprechen und in verschiedene Modi zu versetzen. Wie die Fernkommunikation technisch erfolgen soll, scheint noch nicht ganz ausgereift zu sein. Laut Medienberichten haben die Netzbetreiber bisher nicht die technischen Möglichkeiten, eine Leistungsreduzierung einzelner Wärmepumpen durchzuführen.

SG-Ready-Steuerung

Betriebszustand		SG-Ready-Signal	
		VCC-Bit 1	VCC-Bit 2
1	Wärmepumpensperre: Wärmepumpe und E-Heizstab sind ausgeschaltet	1	0
2	Automatikbetrieb: Wärmepumpe läuft im Normalbetrieb	0	0
3	Verstärkter Betrieb: Leistungseinstellung 1 (in %) für Heizen und Brauchwasser	0	1
4	Maximalbetrieb: Leistungseinstellung 2 (in %) für Heizen und Brauchwasser)	1	1

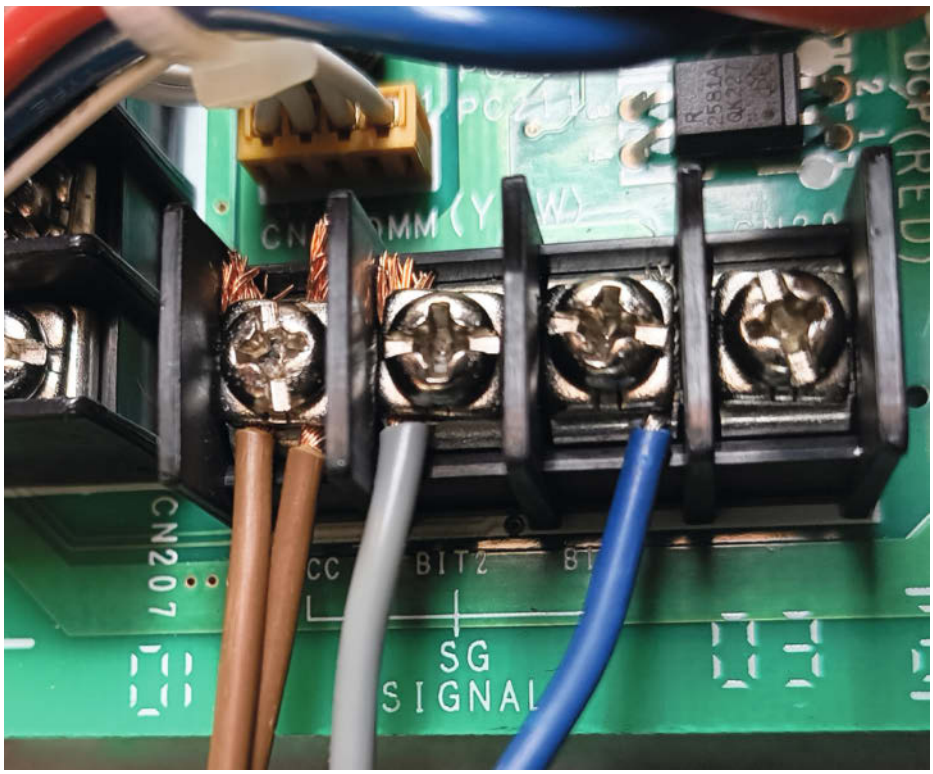


Bild 11: Die braunen Drähte ganz links sind VCC 12 V. Die Klemmen, an denen eine graue und eine blaue Leitung angeschlossen sind, sind die beiden Bits.

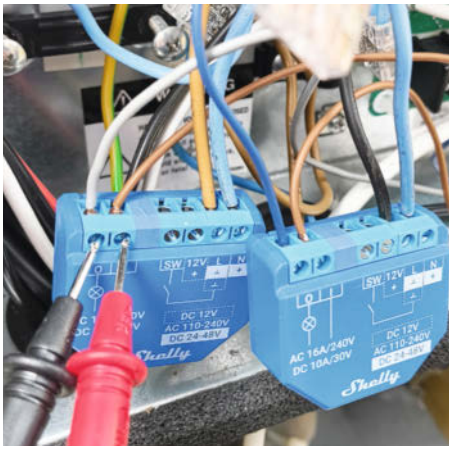


Bild 12: Zwei Shelly Plus 1 werden verwendet, um die beiden Bits in der Wärmepumpe zu schalten.

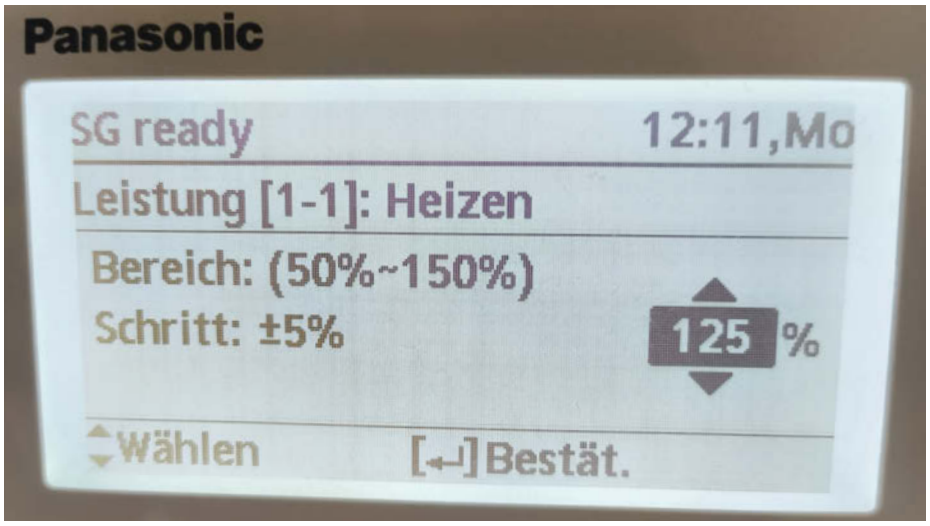


Bild 13: Wie viel mehr Wärme oder Kälte bei PV-Überschuss erzeugt werden soll, kann im Servicemenü der Wärmepumpe eingestellt werden.

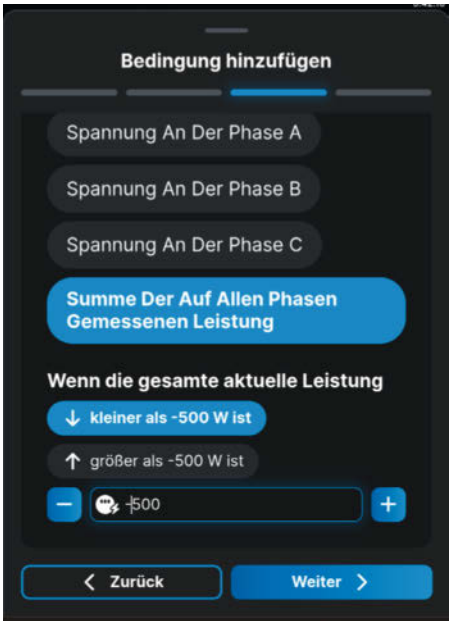


Bild 14: Ausgehend vom Shelly 3EM wird eine Szene erstellt, die ausgelöst wird, sobald 500 Watt ins Netz eingespeist werden.

leistungsmodi umschalten, die beide über das Bedienfeld der Wärmepumpe eingestellt werden können (Bild 13).

Jetzt sind wir endlich so weit, dass wir eine Szene einrichten können, um die Bits zu schalten, sobald ausreichend PV-Strom vorhanden ist. Der Einfachheit halber habe ich mich für den Modus 4 entschieden, bei dem beide Bits geschaltet sind und die Wärmepumpe Warmwasser mit 150 % des Standardwertes erzeugen soll.

Ausgehend vom 3EM habe ich den Stromverbrauch auf allen drei Phasen etwas umständlich auf „unter –500 W" eingestellt. Wenn also alle meine Verbraucher (Licht, PC, Lüftung usw.) gedeckt sind und ich noch 500 W Überschuss habe, sollen die beiden Shelly Plus 1 eingeschaltet werden (Bild 14). Dieser Zustand soll für 30 Minuten gelten, damit die Wärmepumpe nicht sofort wieder abgeschaltet wird, wenn z. B. ein paar Wolken aufziehen.

Die Einstellung in der App ist sehr intuitiv. Man braucht keine Programmierkenntnisse und die Einstellung – in meinem Beispiel sehr einfach – hat keine zwei Minuten gedauert. Das Endergebnis wird übersichtlich dargestellt (Bild 15). Wenn man mehr Kontrolle haben möchte, gibt es andere Möglichkeiten: Mit der Cloud-API können Parameter per HTTP-Request mit POST/GET übergeben werden oder mit Shelly Scripting können Skripte lokal auf Geräten ohne Cloud über das Webinterface des Shelly-Gerätes programmiert werden.

Fazit

Obwohl ich ein totaler Smart-Home-Neuling bin, war die Installation und Inbetriebnahme der Shellys im Großen und Ganzen sehr einfach, wobei ich bei Eingriffen in die Hausinstallation auch von einem Elektriker unterstützt wurde. Das größte Problem stellte der schlechte Empfang der in der Wärmepumpe installierten Shellys dar. Selbst beim 3EM in meinem Verteilerschrank erreicht mich hin und wieder die Meldung eines sehr schwachen WLAN-Signals.

Für einen ausführlichen Test der Umschaltung zwischen den vier verschiedenen SG-Ready-Modi fehlte mir leider die Zeit. Ich habe den Eindruck bekommen, dass die App vielleicht doch nicht genügend Möglichkeiten bietet, um die Szene genau nach meinen Vorstellungen einzustellen. Wenn das so der Fall wäre, müsste man wohl auf andere Lösungen ausweichen, wie z. B. das Senden von Skripten und Befehlen über HTTP, was wiederum für mich als Anfänger einen erheblichen Mehraufwand bedeuten würde.

Insgesamt wirkt das Shelly-Ökosystem freundlich und benutzerorientiert. Ein guter Ausgangspunkt für denjenigen, der mit Shelly anfangen möchte, ist die Shelly-Homepage, die auch einige kreative Beispiele zeigt, was man mit Shelly machen kann. Zum Beispiel, wie ein Wohnwagen mit den kleinen bunten Relais ausgestattet wurde, um etwa den Füllstand des Wassertanks und die Ladung der Batterien zu überwachen. Eine weitere gute Quelle ist das offizielle Onlineforum shelly-forum.com, das unter anderem viele sehr hilfreiche Schaltpläne bietet. —*mch*

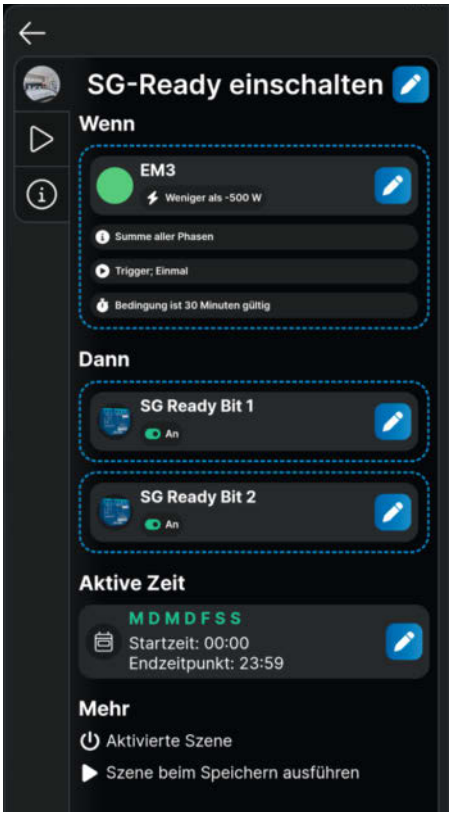


Bild 15: Hier wird die erstellte Szene in der App übersichtlich dargestellt.

S2N Storage Server Network

Die heise-Konferenz für Admins und IT-Verantwortliche

Call for Proposals läuft bis zum 29. April:
Wir freuen uns über Ihre Themeneinreichungen!

22. und 23. Oktober 2025, Regensburg

Platinsponsoren

INFINIDAT

THOMAS KRENN
IT's people business

Goldsponsoren

BACKUP EAGLE
an accompio company

DATACORE

FAST LTA

ITISO

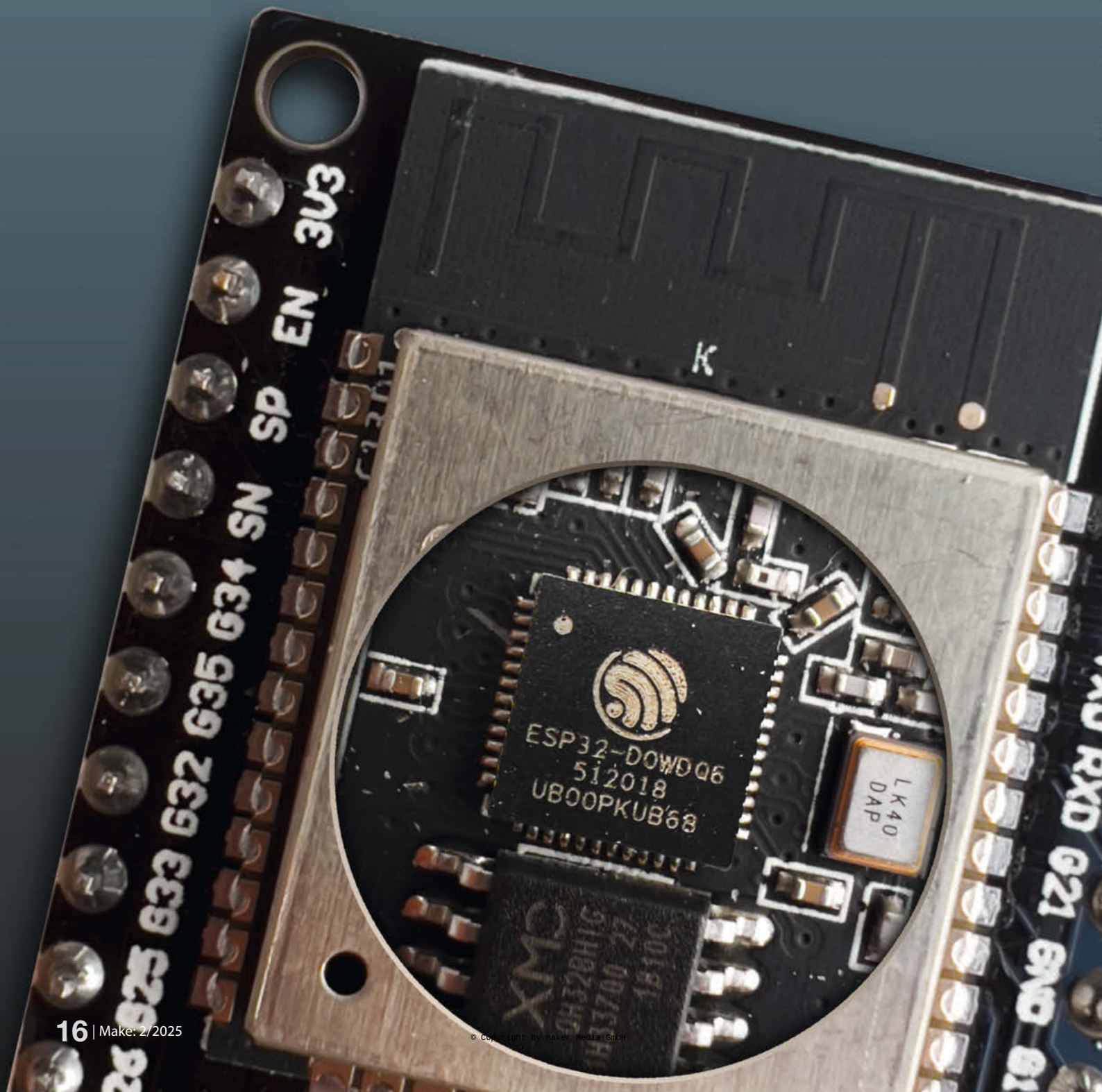
SEP
The Data Protection Company

s2n.heise.de

ESP32-Hardware-Kompass

Zwischen WROOM, NodeMCU und Co. kann man schnell den Überblick verlieren. Dieser Übersichtsartikel macht die ESP32-Familie verständlich und hilft bei der Auswahl der richtigen Hardware.

von Ákos Fodor



Wer ein Projekt mit dem ESP32-Mikrocontroller plant, wird bei seiner Recherche schnell feststellen, dass es eine riesige Auswahl an Produkten gibt, die sich gar nicht so leicht voneinander unterscheiden lassen. Das liegt einerseits daran, dass Espressif seit der Markteinführung des ersten ESP32 im Jahr 2016 fleißig neue Chips entwickelt hat, die sich in vielen Eigenschaften überschneiden. Aber es gibt auch keine offiziellen Regeln für die Form und Namensgebung. Begriffe wie ESP32-WROOM-32, NodeMCU und ESP32-S haben sich etabliert, aber allein zwei dieser Bezeichnungen stammen nicht von Espressif und sind mal Markenname, mal Handelsname. Das verwirrt nicht nur Einsteiger.

Damit ihr besser versteht, wie man die Hardware von Espressif auseinanderhalten kann, haben wir in diesem Artikel zusammengetragen, was die Mitglieder der ESP32-Familie im Wesentlichen vereint und voneinander unterscheidet. Dabei konzentrieren wir uns auf die aktuellen Modelle, die sich in der Serienfertigung befinden und frei erhältlich sind.

Kurzinfo

- » ESP32-Hardwareunterschiede verstehen
- » Schwerpunkte der Chipserien
- » Module besser erkennen und einordnen können

Mehr zum Thema

- » Daniel Bachfeld, Überblick: Speicherarten, Make 1/24, S. 104
- » Josef Müller, KI für den ESP32 – Teil 1, Make 6/21, S. 48
- » Daniel Bachfeld, RISC-V für Maker, Make 4/23, S. 94

Alles zum Artikel im Web unter *make-magazin.de/x8dy*

Einer für alle

Die Chips von Espressif haben die Herzen vieler Maker vor allem mit ihrer WLAN- und Bluetooth-Unterstützung erobert. Sie laufen zudem mit 32- statt 8-Bit wie beim ATmega und bringen in einem kompakten Format viele praktische Features, genügend Power und Speicher mit, um sie in einer großen Bandbreite von IoT-Projekten einzusetzen. Nicht umsonst beschreibt Espressif die ESP32-Modelle in seinen Datenblättern als „general purpose microcontroller", also als „Mikrocontroller für alles".

Ein paar Grundeigenschaften findet man tatsächlich bei den meisten Modellen. Dazu gehören gängige Peripherie-Schnittstellen wie I²C, SPI oder UART, genauso wie Analog-Digital-Wandler (ADC) und PWM-Signalgeneratoren für LEDs oder Motoren, die sich allesamt über zahlreiche GPIOs ansteuern lassen. Hinzu kommen Sicherheitsfeatures wie Secure Boot oder der Ultra-Low-Power-Coprozessor (ULP) – und viele mehr.

Für erste eigene Projekte reichen diese grundlegenden Funktionen oft schon aus und bis auf wenige Ausnahmen kann man dafür eigentlich fast jeden ESP32 nutzen. Möchte man aber ein Projekt umsetzen, das z. B. ZigBee oder Thread benötigt, oder mit KI, Touchpins, Digital-Analog-Wandlern, Audiosignalen und Kameras experimentieren, gibt es manche ESP32-Modelle, die sich besser eignen als andere. Schauen wir dafür zunächst einmal auf die Chips bzw. SoCs (System-on-a-Chip).

Varianten im Überfluss

Auch wenn prinzipiell jeder einen ESP32 mithilfe des Arduino-Frameworks oder des ESP-IDFs (Espressif IoT Development Framework) programmieren kann, werden die Chips in erster Linie nicht für Bastler, sondern für die Industrie entwickelt, die Prozesse und Ressourcen gnadenlos optimiert. D. h., alles, was zu viel ist, ist zu teuer – und fliegt raus. Daher gibt es ESP32-Varianten mit teilweise minimalen Unterschieden. Manche Chips haben integrierten Flash, andere verzichten auf WLAN oder variieren geringfügig in der Anzahl ihrer Peripherie-Schnittstellen.

Mithilfe von Espressifs „Product Selector" (siehe Link in der Kurzinfo und Bild 1) kann man bequem nach Modellen suchen, nach Eigenschaften filtern und mehrere miteinander vergleichen. Um zusätzlich etwas bei der Orientierung zu helfen, hat Espressif Modelle mit ähnlichen Eigenschaften in Serien gruppiert. In den letzten acht Jahren haben sich zum ursprünglichen ESP32 die S-, C- und H-Serie dazugesellt und eine P-Serie ist bereits angekündigt.

Der klassische ESP32

Diese Chipreihe ist der Alleskönner unter den ESP32-SoCs. Sie verfügt über Funktionen, die man auch in anderen, jüngeren Serien wiederfindet, bietet aber insgesamt die größte Bandbreite an Möglichkeiten. Als CPU nutzt der ESP32 den Xtensa LX6, einen 32-Bit-Dual-Core-Prozessor (RISC) mit einer Taktrate von bis zu 240 MHz und 520 KB SRAM. Der Chip unterstützt bis zu 8 MB internen Flashspeicher, ist aber auch in Modulen mit bis zu 16 MB sowie zusätzlichem externen Arbeitsspeicher (PSRAM, Pseudo-RAM) erhältlich.

Neben den üblichen Grundfunktionen verfügt diese Serie zusätzlich über I²S für Audio, zwei 8-Bit-DACs, zehn Touchpins, ein Kamera- und ein Ethernet-Interface sowie eine Single Precision FPU (Floating Point Unit) für Berechnungen mit Gleitkommazahlen. Mit 34 GPIOs liegt die ESP32-Serie im Vergleich mit den anderen etwa im Mittelfeld.

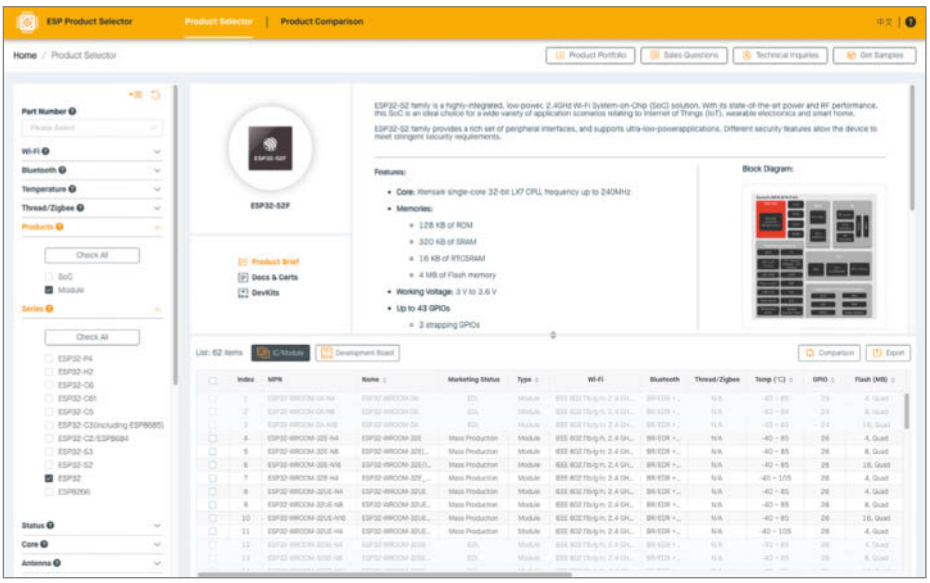


Bild 1: Mit dem „Product Selector" kann man gezielt nach ESP32-Produkten suchen und sie vergleichen.

ESP32-Chip Überblick

Serie	ESP32	S2	S3	C2	C3	C6	H2
Hauptprozessor (32-Bit)	Xtensa LX6 (RISC)	Xtensa LX7 (RISC)	Xtensa LX7	RISC-V	RISC-V	RISC-V	RISC-V
Kerne[1]	2	1	2	1	1	1	1
Takt	240 MHz	240 MHz	240 MHz	120 MHz	160 MHz	160 MHz	96 MHz
FPU	✓	–	✓	–	–	–	–
SRAM	520 KB	320 KB	512 KB	272 KB	400 KB	512 KB	320 KB
Drahtlose Kommunikation							
WLAN	802.11 b/g/n	802.11 b/g/n	802.11 b/g/n	802.11 b/g/n	802.11 b/g/n	802.11 b/g/n/ax	–
Frequenz	2,4 GHz	2,4 GHz	2,4 GHz	2,4 GHz	2,4 GHz	2,4 GHz	–
Übertragungsrate	150 Mbps	150 Mbps	150 Mbps	72,2 Mbps	150 Mbps	150 Mbps	–
Bluetooth Low Energy	4.2 (Classic + LE)	–	5.0	5.0	5.0	5.3	5.0
BLE Mesh	–	–	✓	–	✓	✓	✓
Long Range	–	–	✓	✓	✓	✓	✓
Matter via	WLAN	–	WLAN	WLAN	WLAN	WLAN, Thread	Thread
ZigBee + Thread	–	–	–	–	–	✓	✓
Peripherie							
GPIOs	34	43	45	14	22 oder 16[2]	30 oder 22[2]	19
Touchpins	10	14	14	–	–	–	–
UART	3	2	3	2	2	3	2
I²C	2	2	2	1	1	2	2
SPI	4	4	4	3	3	3	3
I²S	2	1	2	–	1	1	1
ADC	1 × 12-Bit, 18 Kanäle	2 × 13-Bit, 20 Kanäle	2 × 12-Bit, 20 Kanäle	1 × 12-Bit, 5 Kanäle	2 × 12-Bit, 6 Kanäle	1 × 12-Bit, 7 Kanäle	1 × 12-Bit, 5 Kanäle
DAC	2 × 8-Bit	2 × 8-Bit	–	–	–	–	–
PWM	1 × LED (16 Kanäle), 2 × Motor	1 × LED (8 Kanäle)	1 × LED (8 Kanäle), 2 × Motor	1 × LED (6 Kanäle)	1 × LED (6 Kanäle)	1 × LED (6 Kanäle), 1 × Motor	1 × LED (6 Kanäle), 1 × Motor
Camera-Interface	✓	✓	✓	–	–	–	–
LCD-Interface	✓	✓	✓	–	–	–	–

[1] Die in SOLO-Modulen verbauten Chips haben immer Single-Core-CPUs.
[2] Abhängig von der Chip-Konfiguration.

Die S-Serie

Im Jahr 2020 erschien die S2-Serie mit dem Xtensa LX7, einer Single-Core-CPU (RISC) mit 240 MHz, 320 KB SRAM sowie 4 bis 16 MB Flashspeicher. Auch beim S2 gibt es Module mit optionalem externen PSRAM bis 2 MB. Laut Datenblatt lassen sich theoretisch sogar 1 GB Flash und/oder PSRAM anbinden. Die zwei 8-Bit-DACs hat der Chip vom ESP32 geerbt, er kann vier zusätzliche Pins für Toucheingaben aufweisen und liefert mit 43 GPIOs insgesamt mehr Anschlussmöglichkeiten als die Vorgängergeneration.

Wer Kameraprojekte umsetzen will, wird sich über die viermal so schnelle Pixel Clock des S2 freuen (32 MHz statt 8 MHz), die Aufnahmen zügiger oder in einer höheren Auflösung verarbeiten kann. Mithilfe eines LCD-Interface lassen sich außerdem Displays direkt ansprechen. An einigen Stellen hat Espressif aber auch gespart: So hat der S2 unter anderem zwar WLAN, aber kein Bluetooth, keine FPU und man kann nur acht Pins für PWM-Signale nutzen – halb so viele wie beim (alten) ESP32.

Hier legt die S3-Serie nach. Bis auf Ethernet kann sie nämlich alles, was ihre Vorgängerserien konnten, und noch mehr. Dazu gehören Prozessorerweiterungen für AIoT-Projekte (IoT mit künstlicher Intelligenz), etwa für Vektorberechnungen, sowie eine etwas schnellere FPU. Damit eignet sich der S3 besonders für anspruchsvolle Projekte mit Audio- und Bilderkennung oder digitaler Signalverarbeitung (z. B. FFT). Espressif bietet für Gesichts- und Spracherkennung mit ESP-WHO und ESP-Skainet passende Entwicklungstools an, die jedoch die hausinterne Entwicklungsumgebung ESP-IDF erfordern.

Angetrieben wird der S3 von einer Dual-Core-Variante des LX7 mit bis zu 240 MHz.

Neue Chips im Anmarsch

Espressif kündigt neue Produkte oft schon lange vor dem Beginn der Serienproduktion an: Auf den C5 warten wir bereits seit Juni 2022, den P4 gab das Unternehmen Anfang 2023 bekannt, den C61 im Januar 2024 und den H4 drei Monate später.

Da sie allesamt noch nicht bei Händlern wie Reichelt und Co. erhältlich sind, haben wir sie in diesem Überblick nicht berücksichtigt. Im Moment erfordert es noch ein wenig Glück, um an ein Muster-DevKit mit einem der neuen Chips darauf zu gelangen.

Wie es der Zufall will, konnten wir für die aktuelle Ausgabe einen Blick auf den C5 werfen, den ersten ESP32 mit 5-GHz-WLAN. Auf S. 129 berichten wir über unsere ersten Eindrücke. Die Ankündigungen mit weiteren Details findet ihr über den Link in der Kurzinfo.

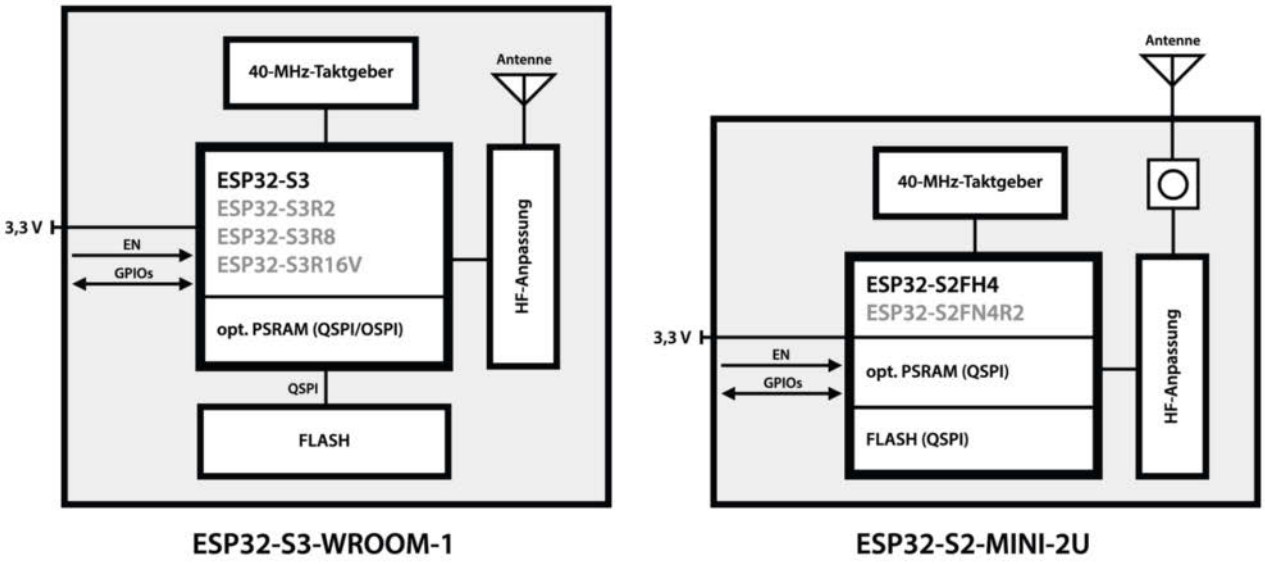


Bild 2: Schematischer Aufbau zweier ESP32-Module. Rechts eine U-Variante mit externer Antenne.

Er nutzt 512 KB SRAM sowie 4 oder 8 MB Flash (im Modul sogar bis zu 32 MB) und bis zu 16 MB PSRAM. Extern lassen sich wie beim S2 beide noch erweitern und über QSPI (Quad SPI) oder mit dem schnelleren OSPI (Octal SPI, 8 parallele Datenleitungen) anschließen. Mit Bluetooth LE 5.0 kann sich der S3 mit dem Long-Range-Modus auch über weitere Distanzen hinweg mit anderen Geräten verbinden und mit 2 Mbps doppelt so schnell Daten mit ihnen austauschen. Zudem haben die S3-SoCs mit 45 GPIOs die meisten Anschlussmöglichkeiten (neben dem angekündigten ESP32-P4, der 55 mitbringen soll).

Die C-Serie

Mit diesen Chips hat Espressif im Jahr 2021 und 2022 die ersten Modelle mit Single-Core-RISC-V als Hauptprozessor veröffentlicht. Diese CPU-Architektur ist nicht nur lizenzfrei, sondern auch open source, was Espressif als Halbleiterentwickler ohne eigene Fabrik („fabless company") eine größere Auswahl lässt, bei wem sie zukünftig ihre Chips fertigen lassen.

Die C-Serie zeichnet sich durch einen geringeren Stromverbrauch im aktiven und ruhenden Modus (Deep-Sleep) aus und ihre Modelle lassen sich (bis auf den C2) sogar direkt über USB programmieren, ohne dass man den Umweg über einen USB-Seriell-Wandler gehen muss. Mit Chips, die BLE Mesh oder zum Teil auch WiFi 6, ZigBee und Thread unterstützen, eignet sich die C-Serie besonders für Projekte im Smarthome (z. B. für Matter).

Der C2 (ESP8684) ist mit 120 MHz das langsamste Modell dieser Reihe, besitzt 272 KB SRAM, internen Flash mit 2 oder 4 MB und keinen PSRAM. Auch sonst ist sein Funktionsumfang begrenzt: Mit 14 GPIOs hat er von allen Chips am wenigsten Anschlussmöglichkeiten, die Datenrate seines WLANs ist mit 72,2 Mbps nur halb so schnell wie das der anderen Modelle, und es fehlt ein I²S-Interface. Er lässt sich aber im Netzwerk gut dafür einsetzen, Sensordaten mit anderen Geräten auszutauschen.

Mit 160 MHz, 400 KB SRAM und (je nach Konfiguration) internem oder externem Flash (4 MB) in der Standardausführung ist der C3 eine etwas stärkere Alternative zum C2. Auch bietet er mit 22 GPIOs mehr Anschlussmöglichkeiten und kann über I²S auch Signale an ein angeschlossenes Soundboard schicken. Den Chip gibt es in einer abgespeckten Version als ESP8685 mit 2 MB Flashspeicher.

Besonders interessant ist der ESP32-C6 mit 160 MHz, 512 KB SRAM und 4 oder 8 MB Flash. Es handelt sich um den ersten Chip, den Espressif mit dem IEEE-Standard 802.15.4 ausgestattet hat, der Thread (1.3) und ZigBee (3.0) unterstützt. Hinzu kommt auch das erste Mal WiFi 6 (ax, 2,4 GHz). Damit lassen sich stromsparende IoT-Projekte umsetzen, bei denen die WLAN-Verbindung eines ESP32 zu einem (WiFi-6-kompatiblen) Access Point (AP) aufrechterhalten bleiben soll. Dafür nutzt der C6 die Target Wake Time (TWT), einen vereinbarten Zeitpunkt, zu dem er aus dem Light-Sleep aufwacht, um Daten mit dem AP auszutauschen. Bei kurzer Schlafdauer (wenigen Sekunden) ist TWT sogar sparsamer als eine Lösung mit Deep-Sleep. Außerdem ermöglicht WiFi 6 mit OFDMA (Orthogonal Frequency Division Multiple Access), kleine Datenmengen effizient zu versenden, was die Intervalle für eine Datenübertragung verkürzt und ebenfalls Strom spart.

Schließlich sei am Rande noch erwähnt, dass der C6 einen integrierten Zufallsgenerator besitzt, der echte 32-Bit-Zufallszahlen erzeugen kann. Dieser hat es auch in den ESP32-H2 geschafft.

Bild 3: Je nach erforderlicher Leistung gibt es ESP32-Module in verschiedenen Größen.

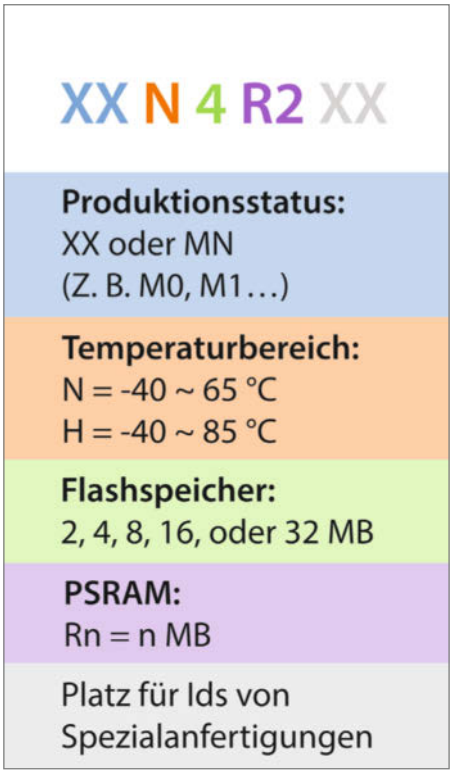


Bild 4: Die ID auf einem Espressif-Modul gibt Aufschluss über das Innenleben.

Die H-Serie

Der bisher einzige Vertreter der H-Serie in Serienproduktion ist der ESP32-H2. Er ist mit seinem RISC-V und 96 MHz der langsamste ESP32-Chip, allerdings verbraucht er durch die geringe Taktung auch entsprechend weniger Strom. Auf WLAN hat Espressif beim H2 verzichtet, der Chip unterstützt Matter aber über Thread und kann zudem mittels ZigBee und Bluetooth LE 5.3 kommunizieren. Damit eignet sich der H2 für leichte Aufgaben und Projekte, bei denen drahtloser Datenaustausch wichtig ist.

Bild 5: Welche Konfiguration verbirgt sich wohl hinter diesem No-Name-Modul?

In Module verpackt

ESP32-Chips werden im SMD-Format gefertigt. Die GPIO-Pins sind also entweder winzig oder befinden sich unterhalb des Chips, sodass man in der Regel eine Platine braucht, um die Pins nach außen zu führen. Außerdem nutzen die meisten ESP32-Modelle einen externen Taktgenerator und bei manchen muss oder kann man noch externen Flash oder PSRAM verbinden. Für die drahtlose Kommunikation benötigt der ESP32 auch eine geeignete Antenne, die man mit der entsprechenden Expertise so in die Hardware integrieren muss, dass sie später keine Störungen durch ungewollte Abstrahlungen erzeugt.

Damit man als Hersteller oder Maker gleich loslegen kann, nimmt Espressif uns diese Vorarbeit ab und bietet fertig konfigurierte und zertifizierte Module an (Bild 2). An diesen lassen sich die GPIO-Pins in der Regel gut erreichen und eine Metallabdeckung schirmt die Strahlung nach außen ab. Um sie zu programmieren, braucht man allerdings einen USB-Seriell-Wandler, z. B. in Form eines FTDI-Boards, wie wir es beim Tibber-Preisrahmen in der Make 4/24 gezeigt haben (dort lediglich mit einem ESP8266).

Espressif fertigt Module in den Varianten WROVER, WROOM, SOLO, MINI und PICO. Sie sind unterschiedlich dimensioniert (Bild 3), was bestimmt, wie viel Flash und PSRAM (im Chip oder auf dem Modul) verbaut sein kann:

- WROVER sind die größten Module, die auf einer Platine eine Fläche von etwa 18 × 31,4 mm belegen. Sie haben externen Flashspeicher sowie PSRAM auf dem Modul verlötet.
- WROOM-Modulen begegnet man als Maker am meisten. Diese kommen mit Flash auf dem Modul, aber überwiegend ohne PSRAM.
- SOLO-Module sind wie ein WROOM aufgebaut und zum Teil mit diesen Pin-kompatibel, nutzen jedoch nur eine Single-Core-CPU.
- MINI- und PICO-Module (15,4 × 20 mm und kleiner) haben nur begrenzt Platz für Flash und PSRAM im Chip.

Die Antenne der ESP32-Module ist entweder direkt in die Platine integriert oder es ist ein Anschluss für eine externe IPEX/U.FL-Antenne verbaut, was man in der Typenbezeichnung an einem angehängten I oder U erkennt (z. B. ESP32-S2-WROVER-I oder ESP32-H2-MINI-1U). Module ohne PCB-Antenne sind etwas kompakter in ihrer Höhe – außer der WROVER-I, weil er trotz des Antennenanschlusses eine PCB-Antenne besitzt, die aber nicht mit dem Schaltkreis verbunden ist.

Klopf, klopf

Welche Kombination von Komponenten sich genau unter der Metallabdeckung eines ESP32-Moduls befindet, lässt sich durch die reine Typenbezeichnung nicht herausfinden. Ein ESP32-WROOM-32E kann z. B. 4, 8 oder 16 MB externen Flashspeicher und optional 2 MB internen PSRAM haben. Damit sich das von außen erkennen lässt, graviert Espressif in die linke untere Ecke der Metallabdeckung eine ID (Bild 4). Die Ziffern XXN8 würden z. B. bedeuten, dass das Modul für niedrige Temperaturen ausgelegt ist und 8 MB Flashspeicher hat.

Jetzt gibt es aber nicht nur die Module von Espressif, sondern auch von zahlreichen Drittanbietern, die eigene Namen für ihre Konfigurationen verwenden (müssen), die aber ziemlich nah an den Originalen sind, z. B. ESP-WROOM-32 (ohne „32“ hinter dem „ESP“) oder ESP32-S. Sind diese Module dann nicht ordentlich/transparent dokumentiert oder gar keinem Hersteller zuzuordnen, kann man vor dem Kauf nur spekulieren, was das Modul eigentlich kann (Bild 5).

Hat man so ein Gerät bereits in seiner Elektronikkiste herumliegen, lassen sich die Interna mit dem esptool bestimmen, das man aus dem Internet herunterladen kann (siehe Link in der Kurzinfo). Sobald man das Programm entpackt

```
esptool.py v4.8.1
Found 1 serial ports
Serial port /dev/cu.usbserial-0001
Connecting.......
Detecting chip type... Unsupported detection protocol, switching and trying again...
Connecting....
Detecting chip type... ESP32
Chip is ESP32-D0WDQ6 (revision v1.0)
Features: WiFi, BT, Dual Core, Coding Scheme None
Crystal is 40MHz
MAC: 30:ae:a4:21:b1:a8
Uploading stub...
Running stub...
Stub running...
Manufacturer: c8
Device: 4016
Detected flash size: 4MB
Flash voltage set by a strapping pin to 3.3V
Hard resetting via RTS pin...
```

Bild 6: Das esptool offenbart den Modulinhalt, ohne dass man die Metallabdeckung entfernen muss.

und sein ESP32-Board per USB verbunden hat, gibt man in der Eingabeaufforderung oder im Terminal folgenden Befehl ein:

```
esptool flash_id
```

Daraufhin sucht esptool nach angeschlossenen ESP-Boards und gibt Informationen über den Chip (inkl. Revision), Features, den Taktgeber, die MAC-Adresse, ggf. den verfügbaren PSRAM sowie die Größe des Flashspeichers (Bild 6) preis. Nicht wundern: Auch wenn ein ESP32 mit bis zu 240 MHz laufen kann, treibt ihn ein 40-MHz-Kristall an.

Alternativ kann man auch die Metallabdeckung entfernen und die Bezeichnungen der Chips mit einer Lupe prüfen. Anschließend sollte man den ESP32 allerdings von Radios fernhalten.

Bei Seeed Studio, Adafruit und anderen namhaften Herstellern findet man neben den technischen Angaben in der Regel auch Schaltpläne zu den Produkten, auf denen sich die Konfiguration der ESP32-Module ebenfalls nachvollziehen lässt (Bild 7).

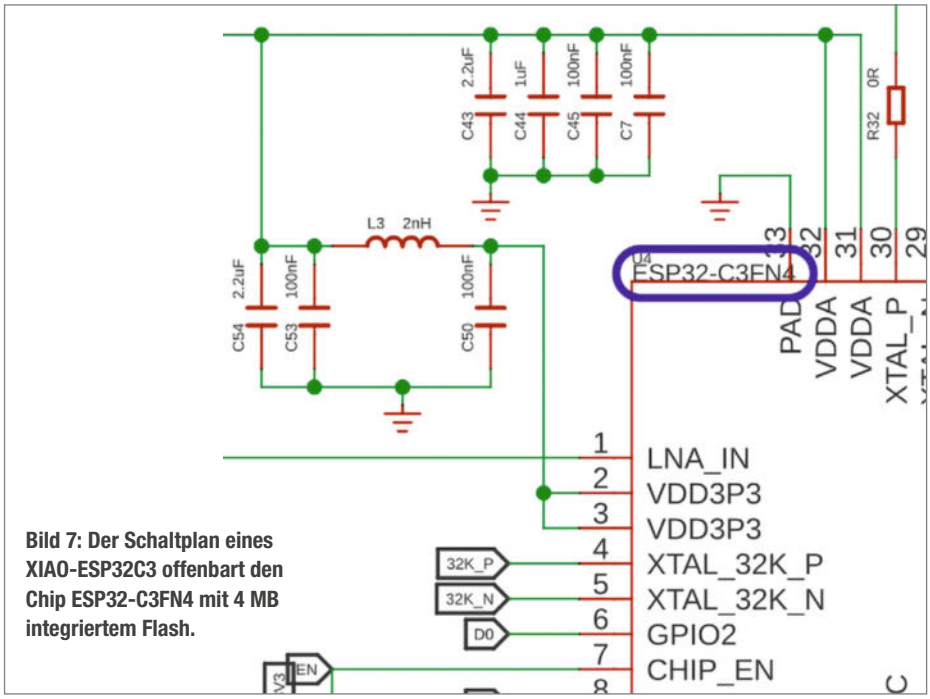


Bild 7: Der Schaltplan eines XIAO-ESP32C3 offenbart den Chip ESP32-C3FN4 mit 4 MB integriertem Flash.

Lebenszyklen

Espressif garantiert die Produktion seiner Chips für 12 Jahre (bei der ersten ESP32-Serie und dem ESP8266 sogar für 15 Jahre). Das bietet Planungssicherheit für Unternehmen und hat den positiven Nebeneffekt, dass auch Maker lange Zeit haben, mit den Produkten zu experimentieren. Nähert sich ein ESP32-Chip oder -Modul dem Ende seiner Produktion (EOL, End of Life), versieht Espressif die Komponente in den Datenblättern mit dem Hinweis „NRND" (Not Recommended for New Designs), nicht empfohlen für neue Designs. In der Regel passiert das bereits Jahre vorher oder wenn eine neue Revision auf den Markt kommt. Fünf ESP32-Chips sind z. B. derzeit auf NRND, deren Produktion aber erst 2031 eingestellt wird.

Weil Espressif seine Kunden behalten will, findet man immer auch einen Verweis auf neuere, verbesserte und günstigere Chipvarianten als Alternative.

Dies kann auch für fortgeschrittene Maker-Projekte relevant sein. Denn wenn man sich etwa ein ESP32-WROOM-32-Modul kauft, nutzt man den ersten ESP32-Chip von 2016. Mittlerweile empfiehlt Espressif das Nachfolgermodul ESP32-WROOM-32E oder -UE, dessen Chip im Jahr 2020 in der dritten Revision (ESP32-D0WD-V3) erschienen ist und u. a. Hardwareprobleme beim externen RAM-Zugriff und einen ungewollten Watchdog-Reset nach dem Deep-Sleep behoben hat (siehe Link in der Kurzinfo).

Entwicklerboards

Die meisten Maker starten ihre ESP32-Reise hier. Entwicklerboards sind schließlich die einfachste Art, mit einem ESP32 zu interagieren. Sie bestehen im Wesentlichen aus dem Modul, einem USB-Seriell-Wandler-Chip, der die Kommunikation mit dem Computer über USB ermöglicht, einem Spannungsregler, der die eingehenden 5 V in 3,3 V für den Mikrocontroller umwandelt, und zwei Tastern (BOOT und EN). Für das schnelle Prototyping sind zudem eine Reihe von GPIOs an zwei Pinheader hinausgeführt, mit denen man die Entwicklerboards schnell auf ein Breadboard stecken kann.

Espressif bietet eigene Referenzdesigns an (DevKits, Bild 8) und hat die Schaltpläne seiner Boards veröffentlicht. Das begünstigt Nachbauten und andere Designs wie die Winzlinge der XIAO-Reihe (Bild 9), die kaum größer als ein Modul sind. Zudem gibt es zahlreiche Ausführungen mit zusätzlichen Extras, die man sonst erst anschließen oder anlöten müsste, z. B. Displays, LoRaWAN-Module, NeoPixel, Kameras oder Akkuladeregler. Die Firma M5Stack (gehört mittlerweile zu einem Großteil Espressif) verpackt den ESP32 sogar in schicke Gehäuse wie den Cardputer oder M5Paper-E-Reader, die zum Experimentieren einladen (Bild 10). Und schließlich sitzt auch in der Oxocard Connect, die wir seit Kurzem mit einem Make-Special anbieten, ein ESP32.

Bild 9: Falls man ein kompaktes Board ohne viele GPIOs benötigt, geht es kaum kleiner als mit der XIAO-Reihe von Seeed Studio.

Zusammengefasst

Für generelle Projekte und den größten Experimentierspielraum ist der ursprüngliche ESP32 nach wie vor eine gute und günstige Wahl. Irgendwas mit WROOM-32 reicht für den Anfang völlig aus. Mit annähernd demselben Funktionsumfang, aber mehr Leistung für anspruchsvolle IoT- oder Kameraprojekte ist man mit dem S3 für die Zukunft gerüstet – und kann den S2 im Grunde überspringen. Im Bereich Konnektivität bietet die C-Serie bisher die spannendsten Optionen fürs Smarthome. Und wer einen Chip sucht, der nicht viel können muss, aber dafür wenig Strom verbraucht, ist mit dem H2 gut bedient. Zum generellen Thema Stromverbrauch bei ESP32-Modellen erzählen wir euch demnächst auch gern noch mehr. *—akf*

Bild 8: Espressif bietet SoCs, Module und DevKits an wie dieses ESP32-S2-DevKitM-1.

Bild 10: Dieser kleine Minicomputer mit Tastatur, Display, Mikrofon und weiteren Features läuft mit einem ESP32-S3 und lässt sich nach Belieben programmieren.

WIR TEILEN KEIN HALBWISSEN.
WIR SCHAFFEN FACHWISSEN.

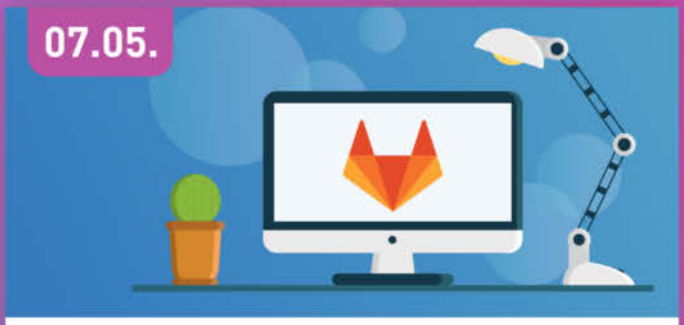
29.04.

WEBINAR

Sprach-KI produktiv einsetzen

c't-Redakteure geben einen Überblick über die gängigen Sprachmodelle. Sie erläutern Kosten, Ressourcenbedarf und Einsatzmöglichkeiten.

07.05.



WORKSHOP

Einführung Gitlab

Erfahren Sie, wie Sie GitLab einrichten, konfigurieren und anpassen. Außerdem lernen Sie, wie Sie eine eigene Instanz der Entwicklungsplattform betreiben.

08.05.

WORKSHOP

Von Excel zu Power BI

Der Workshop stellt praxisnah die wichtigsten Grundlagen des BI-Konzeptes und die verschiedenen Komponenten vor.

14. + 21.05.

WORKSHOP

CI/CD mit GitLab

Lernen Sie die Continuous-Integration-Funktionen (CI) der Entwicklungsplattform GitLab kennen und üben Sie, wie man damit Softwareprojekte baut.

03.06.

WEBINAR

KI-Schreibwerkzeuge im Praxiseinsatz

Wir zeigen Ihnen, wie Sie ein für Ihren Arbeitsalltag passendes Tool auswählen, gewinnbringend einsetzen und die Ergebnisse der KI kritisch prüfen.

04.06.

WEBINAR

Passkeys statt Passwörter

Wir erläutern was Passkeys sind, wie sie funktionieren und vor allem, wie man sie im Alltag nutzen kann.

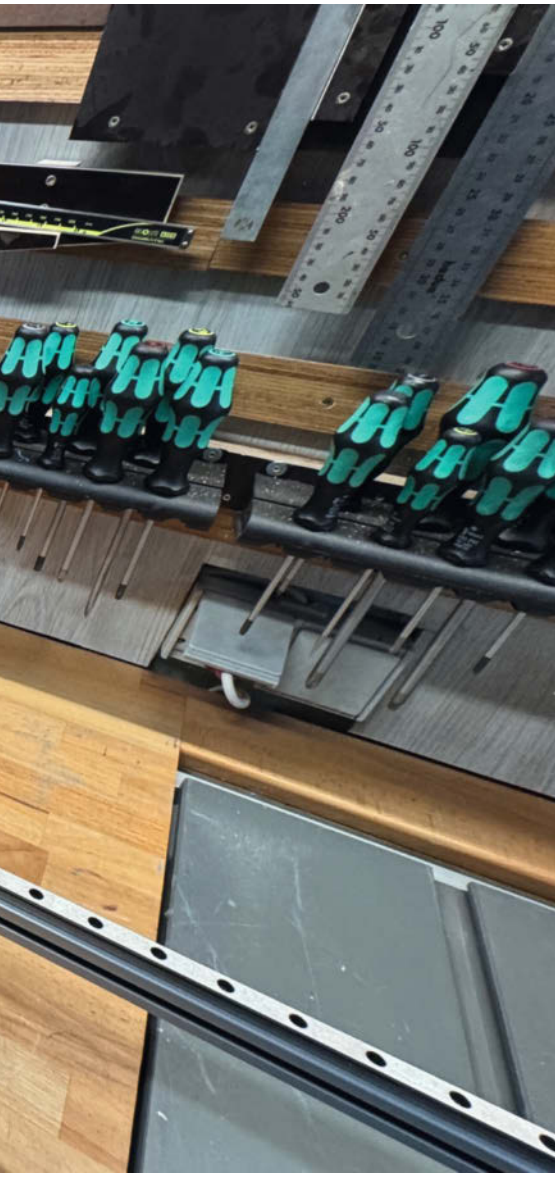
Sichern Sie sich Ihren Frühbucher-Rabatt:

heise.de/ct/Events

Touchscreen am ESP32

Unsere rasante Touchsteuerung sorgt für eine latenzfreie Abfrage der Displayberührungen und positioniert einen motorisierten Anschlag für eine Kappsäge. Die Software lässt sich aber leicht an ähnliche Positionierungsaufgaben anpassen.

von Frank Frohnert



Kurzinfo

- ESP32 als Modul in eigenen Projekten nutzen
- Einbindung des FT810-Grafik-Controllers für latenzfreie Darstellung
- Sichere Anbindung von Handrad, Sensoren und Stellmotor-Steuerung

Checkliste

Zeitaufwand:
20 Stunden

Kosten:
300 Euro

Material

- **ESP32-WROOM-32-UE-N4-Modul**
- **FT810-Grafik-Controller**
- **3,5-Zoll-IPS-LCD,** TL035QVH11-H1205A
- **TMC4210-Motion-Controller-IC**
- **Schrittmotor, Linearführung, Handrad**

Werkzeug

- **Lötausrüstung** (SMD-tauglich)
- **3D-Drucker** für Gehäuse und Anschlag

Mehr zum Thema

- Florian Schäffer, Intelligenter Grafik-Touchscreen, Make 5/18, S. 70

Alles zum Artikel im Web unter *make-magazin.de/xw11*

Der fertige Controller. Die Dateneingabe erfolgt wahlweise über das Touchdisplay oder das Handrad.

Die Umsetzung einer intuitiven Touchdisplay-Benutzeroberfläche mit dem ESP32 ist eine Herausforderung: Die üblichen Nextion-Displays für die Eingabe von Daten und deren visuelle Ausgabe sind bestenfalls Kompromisse, denn sie bringen erhebliche Nachteile mit sich. Der Editor ist umständlich zu bedienen und die Firmware für ESP32 und Nextion-Display muss separat aufgespielt werden. Für eine benutzerfreundliche Lösung, die auch technisch weniger versierte Anwender in Betrieb nehmen können, ist das keine Option. Besonders bei Software-Updates erfordert dieses Setup ein gewisses Maß an Fachwissen, um Fehler zu vermeiden. Hinzu kommen weitere Schwächen: Solche Systeme sind teuer, langsam und unflexibel.

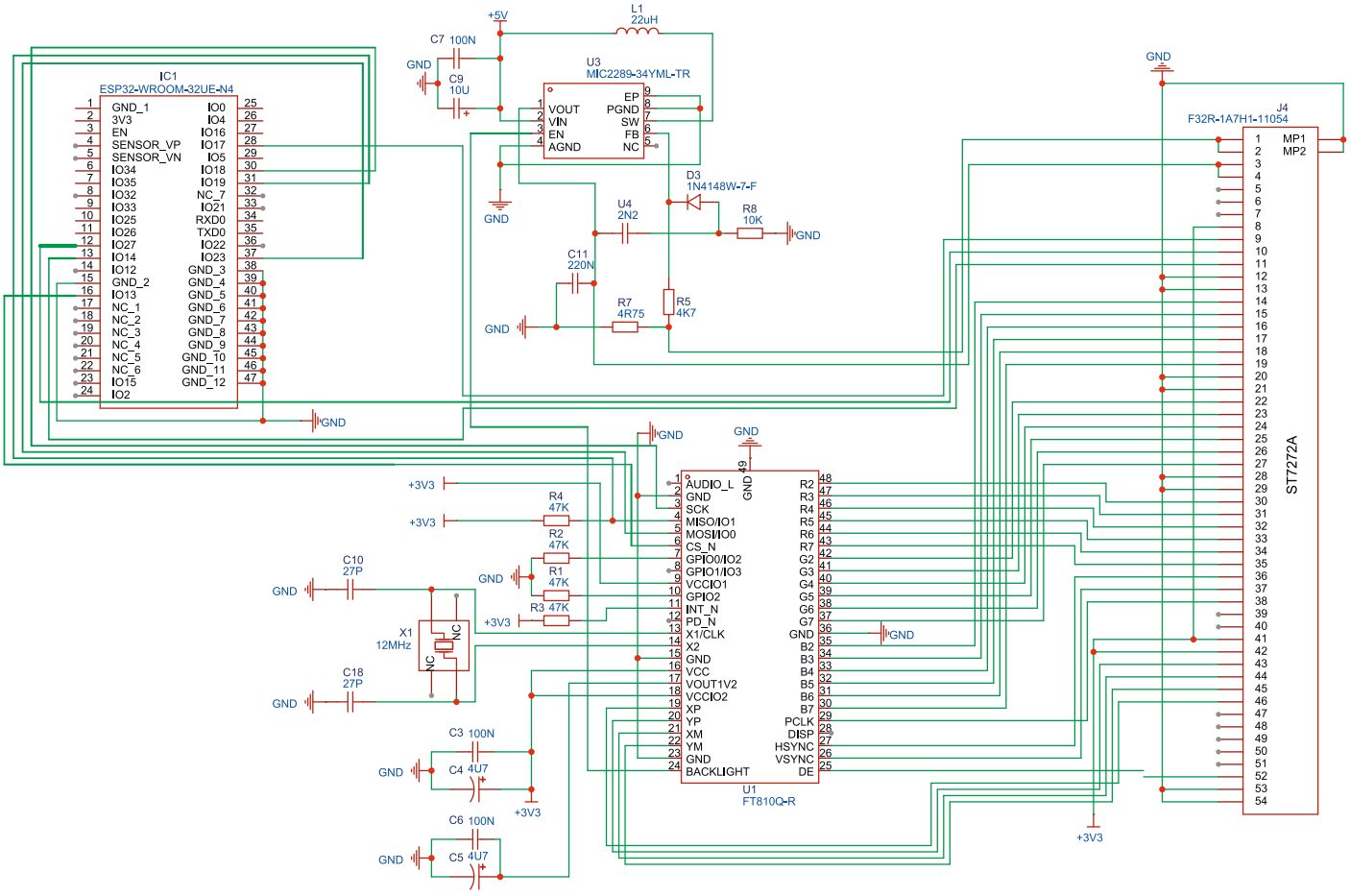
Für das in diesem Artikel sowie im Folgeartikel in Make 3/25 beschriebene Projekt habe ich daher auf Basis des FT810-Grafik-Controllers eine Alternative entwickelt, die ich euch näher vorstellen werde.

Es handelt sich dabei um einen motorisierten Anschlag für eine Kappsäge, der sich durch Drehung einer Spindel auf eine am Display einstellbare Werkstücklänge einstellt. Mit dem optionalen Handrad (A/B-Encoder) kann zusätzlich die Position des Anschlags verändert werden. Außerdem werden zwei induktive Näherungssensoren (NPN) abgefragt. Einer dient als Endschalter und Nullpunkt und ist so für die Positionsbestimmung des Anschlags zuständig. Ein zweiter, optionaler Sensor kann den Kappsägeanschlag ein Stück zur Seite fahren, um ein Verkanten des Werkstücks zwischen Anschlag und Sägeblatt zu verhindern. Ich habe ihn so an der Säge montiert, dass er den Kontakt verliert, wenn die Säge nach unten geklappt wird.

Die Steuerung lässt sich leicht für andere Positionierungsaufgaben anpassen, da in der Software alle relevanten Parameter eingestellt werden können.

Der ESP und seine Peripherie

Als Mikrocontroller kommt ein ESP32-Modul (ESP32-WROOM-32UE-N4) zum Einsatz. Dieses Bauteil hat statt der Platinenantenne für WLAN einen Antennenstecker und ist dadurch kleiner. Da wir kein WLAN benötigen, müssen wir keine Antenne anschließen. Zusätzlich benö-



Ausschnitt des Schaltplans, der ESP32 und FT810-Grafikchip sowie der Displayschnittstelle zeigt. Den vollständigen Schaltplan könnt ihr über den Link in der Kurzinfo herunterladen. Oben ist außerdem der Step-up-Wandler für die 34-V-Versorgung des Displays zu sehen.

tigen wir noch einen Boot/Reset-Taster, einen USB-Anschluss (mit dem der ESP32 über einen USB-Seriell-Wandler (FT230XS) beschrieben werden kann) und einen SD-Karten-Leser.

Der ESP32 kommuniziert über SPI mit Komponenten wie dem Display, der SD-Karte und dem Motioncontroller. SPI ist eine serielle Schnittstelle, die über vier Hauptleitungen arbeitet:

- SCLK (Serial Clock) gibt den Takt für die Übertragung vor.
- MOSI (Master Out, Slave In) sendet Daten vom ESP32 an die Peripheriegeräte.
- MISO (Master In, Slave Out) empfängt Daten von den Peripheriegeräten.
- CS (Chip Select) wählt gezielt das Gerät aus, mit dem der ESP32 gerade kommunizieren soll.

Da mehrere Geräte am SPI-Bus hängen, sorgt die CS-Leitung dafür, dass nur das jeweils aktive Gerät auf den Bus zugreift. Im Normalbetrieb wird nur mit dem Motioncontroller kommuniziert, während SD-Karte und Display lediglich bei der Initialisierung oder bei Updates angesprochen werden.

Stromversorgung der Schaltung

Die Schaltung stellt die Stromversorgung und Kommunikation des ESP32 mit den angeschlossenen Peripheriegeräten sicher: Der 24-V-Eingang wird zunächst über eine Schottky-Diode gegen Verpolung geschützt, eine Supressor-Diode kappt Spannungsspitzen über 30 V. Anschließend erzeugt ein Step-down-Wandler eine stabile 5-V-Versorgung, die wiederum über einen Linearregler auf 3,3 V reduziert wird – die Hauptbetriebsspannung der meisten Bauteile. Zusätzlich sorgt ein Step-up-Wandler für die 34-V-Versorgung der Display-Hintergrundbeleuchtung, wobei ein Widerstand die Strombegrenzung sicherstellt. Soll ein anderes Display zum Einsatz kommen, muss dieser Widerstand neu berechnet werden.

Motioncontroller und Displaysteuerung

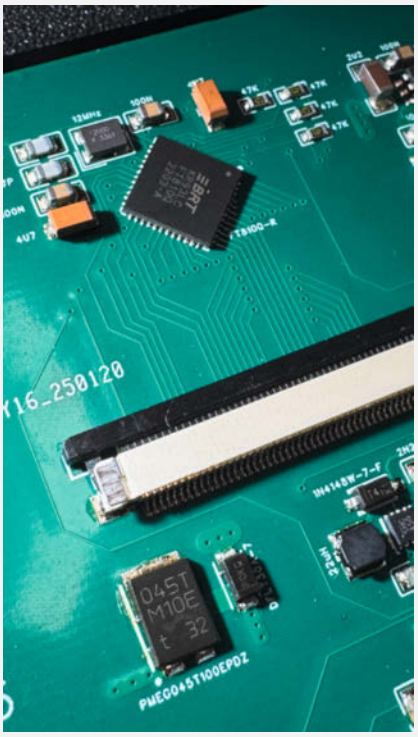
Statt auf dem ESP32 eine Softwarelösung zur Schrittmotorsteuerung zu programmieren, kommt der Motioncontroller-IC TMC4210 zum Einsatz. Er nimmt via SPI die absolute Zielposition vom ESP32 entgegen und berechnet selbstständig An- und Abfahrrampen, Beschleunigung und Geschwindigkeit. Die vom Motioncontroller ausgegebenen STEP- und DIR-Signale werden mit einem Schmitt-Trigger auf 5 V angehoben, damit sie von gängigen Schrittmotor-Endstufen verarbeitet werden können. Vom ESP32 kann jederzeit die aktuelle Position, Geschwindigkeit etc. abgefragt und auch in den laufenden Prozess eingegriffen werden.

Der TMC4210 hat eigene Eingänge für Referenzschalter. Diese habe ich jedoch nicht genutzt, da der ESP32 flexibler auf deren Signale reagieren kann.

Zur Dateneingabe verfügt die Steuerung über einen Touchscreen mit dem Grafik-Controller FT810, der ebenfalls per SPI angesteuert die gesamte Grafikerzeugung vornimmt, Toucheingaben auswertet und zusätzlich einen Audioausgang besitzt (den wir aber nicht nutzen werden). Der Chip beherrscht Grundformen wie Linien und Rechtecke

Die FT81x-Familie

Die einzelnen FT-81x-Chips sind grundsätzlich gleich. Sie unterscheiden sich nur in der Leistung, beim internen Speicher, zusätzlichen Funktionen und beim Touch-Typ (resistiv / kapazitiv). Ich nutze den kleinsten Chip der Familie für Displays mit kapazitivem Touch-Interface. Die Chips der FT-81x Serie sind eigentlich einen eigenen Artikel wert. Leider findet man in der DIY-Szene bisher kaum Projekte, die diese verwenden.



Der FT810-Grafikchip sowie die 54-Pin-Schnittstelle für das Display.

Start der Bildschirmausgabe

```
EVE_cmd_dl(CMD_DLSTART);
EVE_cmd_dl(DL_CLEAR_COLOR_RGB | 0x000000);
EVE_cmd_dl(DL_CLEAR | CLR_COL | CLR_STN | CLR_TAG);
EVE_cmd_dl(DL_TAG);
EVE_cmd_dl(DL_BEGIN);
EVE_color_rgb(0xffffff);
EVE_cmd_fgcolor(0x0020ff);
EVE_memWrite8(REG_PWM_DUTY, 255);
tracker = EVE_memRead8(REG_TOUCH_TAG);
```

Code für einen Button (ohne Funktion)

```
EVE_cmd_fgcolor(0x606060);
EVE_color_rgb(0xffffff);
EVE_cmd_button(85, 190, 70, 50, 27, EVE_OPT_FLAT, "POS+");
```

Bildschirmausgabe beenden

```
EVE_cmd_dl(DL_END);EVE_cmd_dl(DL_DISPLAY);  // mark the end
                                               of the display-list
EVE_cmd_dl(CMD_SWAP);    // make this list active
```

Button mit Funktion

```
EVE_cmd_dl(DL_TAG + 70);
  if (tracker == 70) {
    EVE_cmd_fgcolor(0xA0A0A0);
    EVE_color_rgb(0xffffff);
    upPressed = true;
  } else {
    EVE_cmd_fgcolor(0x606060);
    EVE_color_rgb(0xffffff);
    if (upPressed) {
      upPressed = false;
      setupPos++;
      if (setupPos > setupPositions) {
        setupPos = 0;
      }
    }
  }
  EVE_cmd_button(85, 190, 70, 50, 27, EVE_OPT_FLAT, "POS+");
```

(witzigerweise keinen Kreis), Alpha-Kanal, Antialiasing, Buttons, verschiedene Textgrößen, Linienstärken und vieles mehr.

Das Display besitzt einen ST7272A-Controller, der zum Start einmalig per SPI konfiguriert werden muss. Dabei werden Display-Reset, Helligkeit, Kontrast, Ausrichtung und RGB-Reihenfolge festgelegt. Der Bildschirm ist über ein 54-Pin-Flachbandkabel verbunden. Da die genaue Pinbelegung vom Displaytyp abhängig ist, muss sie bei Verwendung eines anderen Displays entsprechend angepasst werden.

Schutzmaßnahmen und Störungsunterdrückung

Die Eingänge für die Näherungssensoren (einer davon dient als Endschalter) und das Handrad (ein A/B-Inkremental-Encoder mit 100 Impulsen pro Umdrehung) sind mit einer Zenerdiode und darauffolgendem 2-kΩ-Widerstand auf GND gegen Überspannung geschützt. Außerdem befindet sich eine Ferritperle im Signalverlauf, die hochfrequente Störungen eliminiert. Der SPI-Bus von SD-Karte, Motioncontroller und LCD-Konfiguration ist zusätzlich mit je einem 100-Ω-Widerstand in Serie auf MOSI- und CLK-Leitungen bestückt. Die Widerstände befinden sich im Signalpfad hinter der SD-Karte und vor TMC4210 und LCD. Das verhindert, dass kapazitive Störungen vom TMC4210 und LCD die SD-Kommunikation beeinflussen.

Detaillierte Schaltpläne und Erklärungen sowie weiterführende Informationen zur Anpassung des Display-Beleuchtungswiderstandes und der Display-Pinbelegung sind über die URL in der Kurzinfo erreichbar.

Die Software

Bitte seht mir nach, dass die Software grauenhaft dokumentiert ist. Ich habe ursprünglich nicht mit einer Veröffentlichung gerechnet. Damit ihr die grundsätzliche Funktionsweise nachvollziehen könnt, gehe ich im Folgenden daher auf die wichtigsten Eckpunkte der Software ein.

Als externe Klassen werden die FT81x-Klasse von Rudolph Riedel und die TMC4210-Klasse von Tom Magnier (Tomag) eingebunden, wobei ich die FT81x-Klasse ein wenig modifizieren musste, damit sie von der Arduino IDE akzeptiert wird. Ich nutze die Arduino IDE

Die kleine, abgedunkelte Ecke bei den Buttons ADD und SET dient als Hinweis auf die Funktion beim Gedrückthalten der Schaltfläche.

2.3.4 und den ESP-Boardmanager 1.0.4. Diese Boardmanager-Version ist wichtig, da die Funktionsweise spezieller ESP-Befehle in höheren Versionen geändert wurde und der Code somit nicht funktionieren würde.

Die Software besteht aus folgenden Hauptbestandteilen:

- Grafik- und Eingabehandling (Touchscreen)
- Schrittmotorsteuerung
- Firmware-Update per SD-Karte
- Abfrage und Auswertung eines A/B-Drehencoders und der Näherungssensoren

ChatGPT

Den Code für die SPI-Display-Konfiguration (inklusive der Funktion `sendPanelCommand`) habe ich mir von ChatGPT (4o) erstellen lassen. Dazu habe ich das Datenblatt des Displays in ChatGPT geladen und nach der Konfiguration gefragt. Natürlich dauert es trotzdem noch ein bis zwei Stunden, bis alles läuft, aber die KI ist hier schon eine große Hilfe. Aber Vorsicht, ChatGPT fantasiert gerne, was bei Pinbelegungen von Elektronikbauteilen fatal enden kann. ChatGPT sagt niemals, dass es keine Ahnung hat. Stattdessen wird geschätzt und die Schätzung als Fakt dargestellt. Also lieber den digitalen Kumpel hinterfragen, dann kommt's meistens raus.

Grafik

Die Grafikerzeugung ist direkt in den Loop eingebunden. Somit ist es möglich, für jeden Loop-Durchlauf den Bildschirm aufzufrischen. Da man auch nur so an die Toucheingaben gelangt, die logischerweise zeitnah ausgewertet werden sollten, sollte man dies so oft wie möglich machen.

Der FT810 kann diverse Standard-UI-Elemente von Haus aus darstellen. Dazu gehören Buttons, Button-Arrays, Toggle Buttons, Slider, Gauges, Progressbars, Dials und natürlich Text. Alle Details finden sich im FT81x Series Programmers Guide (siehe Link in der Kurzinfo).

Die Bildschirmausgabe startet man mit den Befehlen, wie sie im gleichnamigen Listing-Kasten zu sehen sind. Mit ihnen löscht man den Bildschirm erst und stellt anschließend die Farben ein, mit denen man zeichnen möchte. Danach kann man grafische Objekte zu seiner Bedienoberfläche hinzufügen.

Das Listing „Code für einen Button (ohne Funktion)" erzeugt eine Schaltfläche an der Position X: 85 und Y: 190, 70 Pixel breit und 50 Pixel hoch, mit der Font-Nummer 27, ohne 3D-Effekt und mit der Beschriftung „POS+". Ihre Farbe ist dunkelgrau (RGB-Hex: 606060) und die Beschriftung weiß (RGB-Hex: FFFFFF). Danach beendet man die Bildschirmausgabe noch mit den Befehlen, wie im Listing „Bildschirmausgabe beenden" gezeigt.

Damit der Button auch funktioniert und im Programmcode darauf reagiert werden kann, muss er eine „Touch-ID" bekommen. Natürlich soll der Button auch seine Farbe ändern, wenn der Finger darauf ist. Also erweitern wir den obigen, grafischen Inhalt um diese Bedingung (siehe Listing „Button mit Funktion").

Der Befehl `EVE_cmd_dl(DL_TAG + 70);` bewirkt, dass alle Grafikelemente, die ab jetzt gezeichnet werden, die ID 70 bekommen. Abgefragt wird die ID über die Variable `tracker`, die im obigen Startcode der Displayliste zugewiesen wird: `tracker = EVE_memRead8(REG_TOUCH_TAG);`. Beendet wird der Block mit der ID 70 mit dem Setzen der ID 0 (oder einer anderen) mit dem Befehl `EVE_cmd_dl(DL_TAG + 0);`.

Werden Grafikinhalte nachträglich verschoben, werden die Touchbereiche vom FT810 automatisch auf die neue Position der Elemente angepasst. Im Button-Beispiel oben wird die Touchabfrage zusätzlich so gestaltet, dass eine Berührung erst beim Loslassen registriert wird. Das verhindert, dass eventuelle Zähler etc. losrasen, solange der Button gedrückt ist. Wenn also `tracker==70` (Button gedrückt) ist, ändert sich die Hintergrundfarbe des Buttons auf Hellgrau (A0A0A0) und die Hilfsvariable `upPressed` wird auf `true` gesetzt.

Wenn `tracker` nicht `70` ist, hat er die Farbe Dunkelgrau. Wenn aber zusätzlich die Variable `upPressed == true` ist, wird die gewünschte Buttonaktion ausgeführt (in diesem Fall wird die Setup-Position um 1 erhöht, mit Überlaufschutz) und `upPressed` wird auf `false` gesetzt, damit die Aktion nur einmal ausgeführt wird. So können nach und nach, anhand von Bedingungen, zusätzliche Elemente ein- bzw. ausgeblendet oder komplett neue Screens erstellt werden. Wichtig ist nur, dass sich der Code innerhalb des Bildschimstart- und Endcodes befindet.

Latenzfreie Grafik- und Touchabfrage

Bei einfachen Grafikausgaben mit dem FT810 habe ich Geschwindigkeiten von 130 fps erzielt. Das heißt, dass auch alle Toucheingaben und Grafikänderungen etc. mit dieser Geschwindigkeit bearbeitet werden. Das ist schon ziemlich beeindruckend. Die Kappsägensteuerung läuft bei ungefähr 60 Bildern pro Sekunde, was schnell genug ist, um eine gefühlte Latenzfreiheit zu vermitteln.

Das Aufwendigste ist aber nicht die Grafikprogrammierung selbst, sondern die Abstimmung auf das angeschlossene TFT-Display. Ich nutze in meinen Projekten in der Regel zwei Displaytypen: Ein 5-Zoll-IPS-TFT (800×480 Pixel) mit 40-poligem Anschlusskabel (z. B. bei meinem Routerlift, Link siehe Kurzinfo) und das oben beschriebene 3,5-Zoll-IPS-TFT-Display für den hier beschriebenen Kappsägenanschlag.

Während das 5-Zoll-Display die Daten direkt vom FT810 akzeptiert, was das Handling vereinfacht, muss man das 3,5-Zoll-Display zum Start einmalig per SPI konfigurieren. Das ist zwar nicht sehr aufwendig, macht man es aber nicht, bleibt das Display dunkel. Die benötigten Codezeilen für das 3,5-Zoll-Display zeigt das Listing „Display-Konfiguration".

Es gibt zusätzlich eine Konfiguration für den FT810-Chip, die jedes Display braucht. In der Klasse von Rudolph Riedel sind einige Konfigurationen schon voreingestellt, die man aber kaum gebrauchen kann. Ich habe mir daher einen Displaytyp fix gesetzt und verändere dessen Konfiguration je nach angeschlossenem Display.

Display-Konfiguration

```
sendPanelCommand(0x10, 0b00000001);  // Reset Display
delay(120);
sendPanelCommand(0x14, 0x40);        // Setze Helligkeit auf Standard
sendPanelCommand(0x11, 0x40);        // Standardwert Kontrast
sendPanelCommand(0x19, 0b11110000);  // VA-Modus/Ausrichtung
sendPanelCommand(0x1A, 0b11110110);  // RGB Interface Mode
```

Das Display findet man in der Datei EVE_config.h.

```
#if defined (EVE_FT810CB_HY50HD)
#define EVE_HSIZE (320L)
#define EVE_VSIZE (240L)
#define EVE_VSYNC0 (1L)
#define EVE_VSYNC1 (4L)
#define EVE_VOFFSET (4L)
#define EVE_VCYCLE (245L)
#define EVE_HSYNC0 (10L)
#define EVE_HSYNC1 (20L)
#define EVE_HOFFSET (64L)
#define EVE_HCYCLE (510L)
#define EVE_PCLK (2L)
#define EVE_PCLKPOL (0L)
#define EVE_SWIZZLE (2L)
#define EVE_CSPREAD (0L)
#define EVE_GEN 2
#endif
```

Diese Definitionen beinhalten Displaygröße, Daten für die vertikale und horizontale Synchronisation, Offsets, RGB-Reihenfolge, Pixeltakt und Polarität. Bei Verwendung eines anderen Displays können diese Daten hervorragend mit ChatGPT angepasst werden.

Taktsignal

```
#define TMC4210_CLK_FREQ 10000000L
#define TMC4210_CLK_PIN 16

ledcSetup(0, TMC4210_CLK_FREQ, 1);  //2 bits
ledcAttachPin(TMC4210_CLK_PIN, 0);
ledcWrite(0, 1);
```

Schrittmotorsteuerung

Nach Einbindung der TMC4210-Klasse von Tom Magnier braucht der TMC4210 ein externes Taktsignal mit bis zu 32 MHz. Je schneller der externe Takt ist, desto höhere Schrittraten kann der Chip ausgeben. Für dieses Projekt reicht ein 10-MHz-Takt. Dieser wird mit der ledc-Klasse für den ESP32 realisiert. Im Listing „Taktsignal" befinden sich die beiden `#define` nur zum Verständnis direkt über den Anweisungen für `ledc`.

Einmalig aufgerufen liegen jetzt dauerhaft 10 MHz am Pin 16 des ESP32 an. In der Startkonfiguration sind zwei Parameter sehr wichtig:

```
#define TMC4210_MAX_SPEED 20000

#define TMC4210_MAX_ACC 80000
```

Anhand dieser beiden Parameter wird eine Art Rampenteiler berechnet, dessen genaue Arbeitsweise mir noch nicht hundertprozentig klar ist. Jedoch läuft der Motor am saubersten, wenn `TMC4210_MAX_ACC` den vierfachen Wert von `TMC4210_MAX_SPEED` hat. Die maximale Geschwindigkeit und Beschleunigung können auf dem Display in den Einstellungen geändert werden, Änderungen sind sofort wirksam. Die Rampe wird einmalig beim Start der Software berechnet und gilt dann für alle Geschwindigkeiten und Beschleunigungen.

Drehencoder-Konfiguration

```
// Konfiguration des Pulse Counter
  pcnt_config_t pcnt_config;
  pcnt_config.pulse_gpio_num = ENCODER_PIN_A;  // Puls-GPIO (A-Signal)
  pcnt_config.ctrl_gpio_num = ENCODER_PIN_B;   // Steuer-GPIO (B-Signal)
  pcnt_config.channel = PCNT_CHANNEL_0;        // Kanal
  pcnt_config.unit = PCNT_UNIT;                // PCNT-Einheit
  pcnt_config.pos_mode = PCNT_COUNT_INC;       // Zählen bei
                                                  positiver Flanke
  pcnt_config.neg_mode = PCNT_COUNT_DEC;       // Zählen bei
                                                  negativer Flanke
  pcnt_config.lctrl_mode = PCNT_MODE_REVERSE;  // Umkehren bei
                                                  Steuer-GPIO LOW
  pcnt_config.hctrl_mode = PCNT_MODE_KEEP;     // Beibehalten bei
                                                  Steuer-GPIO HIGH
  pcnt_config.counter_h_lim = 32767;           // Obere Grenze
  pcnt_config.counter_l_lim = -32768;          // Untere Grenze

  // PCNT konfigurieren
  pcnt_unit_config(&pcnt_config);

  // Filter aktivieren (Entprellung)
  pcnt_set_filter_value(PCNT_UNIT, 50);  // Filterzeit in Taktzyklen
                                            (80 MHz Basis)
  pcnt_filter_enable(PCNT_UNIT);

  // Ereignisse und Interrupts aktivieren
  pcnt_event_enable(PCNT_UNIT, PCNT_EVT_H_LIM);  // Ereignis bei
                                                    oberer Grenze
  pcnt_event_enable(PCNT_UNIT, PCNT_EVT_L_LIM);  // Ereignis bei
                                                    unterer Grenze

  // Interrupt-Service-Routine registrieren
  pcnt_isr_register(pcnt_intr_handler, NULL, 0, NULL);
  pcnt_intr_enable(PCNT_UNIT);

  // Zähler starten
  pcnt_counter_pause(PCNT_UNIT);   // Pausieren, um initialen Wert
                                      zu setzen
  pcnt_counter_clear(PCNT_UNIT);   // Zähler auf 0 setzen
  pcnt_counter_resume(PCNT_UNIT);  // Zähler starten
```

Drehencoder-Auswertung im Loop

```
pcnt_get_counter_value(PCNT_UNIT, &encoderCount);

     // Ausgabe der Encoder-Position
     //Serial.print("Encoder-Position: ");
     //Serial.println(encoderCount);
     if (encoderCount >= 2 || encoderCount <= -2) {
       tft.addSollwert((encoderCount / 2) * 1);
       pcnt_counter_pause(PCNT_UNIT);   // Zähler pausieren
       pcnt_counter_clear(PCNT_UNIT);   // Zähler auf Null setzen
       pcnt_counter_resume(PCNT_UNIT);  // Zähler fortsetzen
     }
```

Im `loop` wird dauerhaft die aktuelle Position in Schritte umgerechnet und an den TMC-4210 gesendet. Des Weiteren wird im `loop` dauerhaft der Eingangspin des Endschalters überprüft. Wenn dieser auslöst, wird eine Ablaufkette gestartet, die den Motor stoppt und in Gegenrichtung fahren lässt, bis der Endschalter wieder frei ist.

Firmware-Update per SD-Karte

Da das LCD den Update-Prozess anzeigen soll, musste ich die Implementierung des Firmware-Updates in den `loop` setzen. Damit der Speicherkarten-Code nur einmalig aufgerufen wird, habe ich ihn mit einer boolschen Variablen abgesichert, die nach Ende eines Durchlaufes auf `true` gesetzt wird und somit einen weiteren Durchlauf verhindert.

Grundsätzlich schaut das Skript, ob eine BIN-Datei im Hauptverzeichnis der SD-Karte existiert. Wenn ja, wird versucht, diese Datei zu installieren. Bei Erfolg wird diese Datei auf die Endung .OLD umbenannt und der ESP32 wird neu gestartet. Da jetzt keine BIN-Datei mehr im Hauptverzeichnis vorhanden ist, läuft die Software normal weiter. Eine nicht eingesteckte SD-Karte verzögert den Programmstart etwas, weil versucht wird, eine Karte zu finden. Die Verzögerung ist aber nicht so groß, dass es stören würde.

A/B-Drehencoder

Neben der Eingabe einer Werkstücklänge auf dem Touchdisplay kann der Kappsägenanschlag auch per Handrad verstellt werden. Der Drehencoder (NPN-Signale, also -A und -B vom Encoder) wird mit dem Pulsecounter-Modul (PCNT) des ESP32 ausgelesen. Dieses Modul ist unabhängig vom Prozessor, belastet ihn also auch nicht. Es ist speziell für die Erfassung von A/B-Encodersignalen ausgelegt und kann durch die Triggerreihenfolge von A und B die Richtung und die Anzahl der Klicks ermitteln. So entstehen zwei Zählwerte pro Klick am Handrad, die je nach Drehrichtung entweder automatisch hoch- oder runterzählen. Der PCNT kann aber nur in einem Wertebereich von +/- 32768 zählen und bei zwei Signalen pro Klick halbiert sich dieser Wert noch einmal.

Ich erfasse im Loop mit jedem Durchlauf die Abweichung des Zählers zu null, rechne diesen Wert entsprechend in Millimeter um und addiere oder subtrahiere den Wert vom aktuellen Sollwert der Software. Dann wird der Zähler wieder auf null gesetzt. Im Programmcode gibt es zwar noch eine Interrupt-Routine für die Zählerauswertung, diese ist jedoch leer, weil sich herausgestellt hat, dass eine Auswertung im `loop` völlig ausreichend ist.

Es gibt diverse Bibliotheken für die Drehencoder-Auswertung, ich habe in diesem Fall aber wieder ChatGPT um Hilfe gebeten. Herausgekommen ist der Code im Listing „Drehencoder-Konfiguration“, der keine zusätzlichen Bibliotheken benötigt. Zum Auswerten verwende ich die Befehle aus dem Listing „Drehencoder-Auswertung im Loop“.

Schlusswort

Ein ESP32, der eigenständig einen LCD-Touchscreen betreiben kann, ist eine feine Sache und erweitert die Möglichkeiten solcher und ähnlicher Projekte enorm. Ich hoffe, ich konnte euch das Thema ein bisschen näherbringen.

In der nächsten Make wird es um den Aufbau der Hardware gehen, die aus der in diesem Artikel beschrieben Steuerung den motorisierten Kappsägenanschlag macht. Schließlich bringt die schönste und schnellste Steuerung nichts, wenn kein Motor daran angeschlossen ist, der den Anschlag bewegen kann. Außerdem wird es im Folgeartikel eine Kurzanleitung zur Steuerung und ihrer Konfigurationsparameter geben.

Wenn ihr diesen Artikel lest, wird das entsprechende Video zum Kappsägenanschlag mit aller Wahrscheinlichkeit bereits auf meinem YouTube-Kanal erschienen sein (Frohnix Bastelbude, Link über die Kurzinfo). Dort könnt ihr die Steuerung in Aktion sehen. *—jom*

iX @ heise academy

WORKSHOPS 2025

12. Mai

Jump Start in die Programmiersprache Rust

Sie lernen die Besonderheiten, effiziente Nutzung und Grundlagen in Speicherverwaltung und Fehlerbehandlung.

13. – 14. Mai

Effiziente IT-Sicherheit für kleine und mittlere Unternehmen

Cybersicherheit für Ihr Unternehmen: Lernen Sie die wichtigsten Best Practices für einen effektiven Schutz kennen.

15. Mai

CSRD und IT: Nachhaltigkeit messbar machen

Lernen Sie, CSRD-Anforderungen mit ESG-Reporting-Systemen zu erfüllen und Unternehmensdaten mit IT effektiv zu nutzen.

19. Mai

Mastering Azure: Administration & Konfiguration der Microsoft Cloud

Sie lernen die Komponenten der Microsoft Azure Cloud zu administrieren, zu konfigurieren und zu implementieren.

20. – 21. Mai

OWASP® Top 10 für Entwickler

Entdecken Sie Schwachstellen in Webanwendungen, verstehen und nutzen Sie Angriffstaktiken und Abwehrstrategien.

22. – 23. Mai

Strategisches IT-Management: Von der Vision zur Wertschöpfung

Diese Workshop präsentiert Einblicke in die Prinzipien des strategischen IT- und Technologie-Managements.

Weitere Infos unter **heise-academy.de/marken/ix**

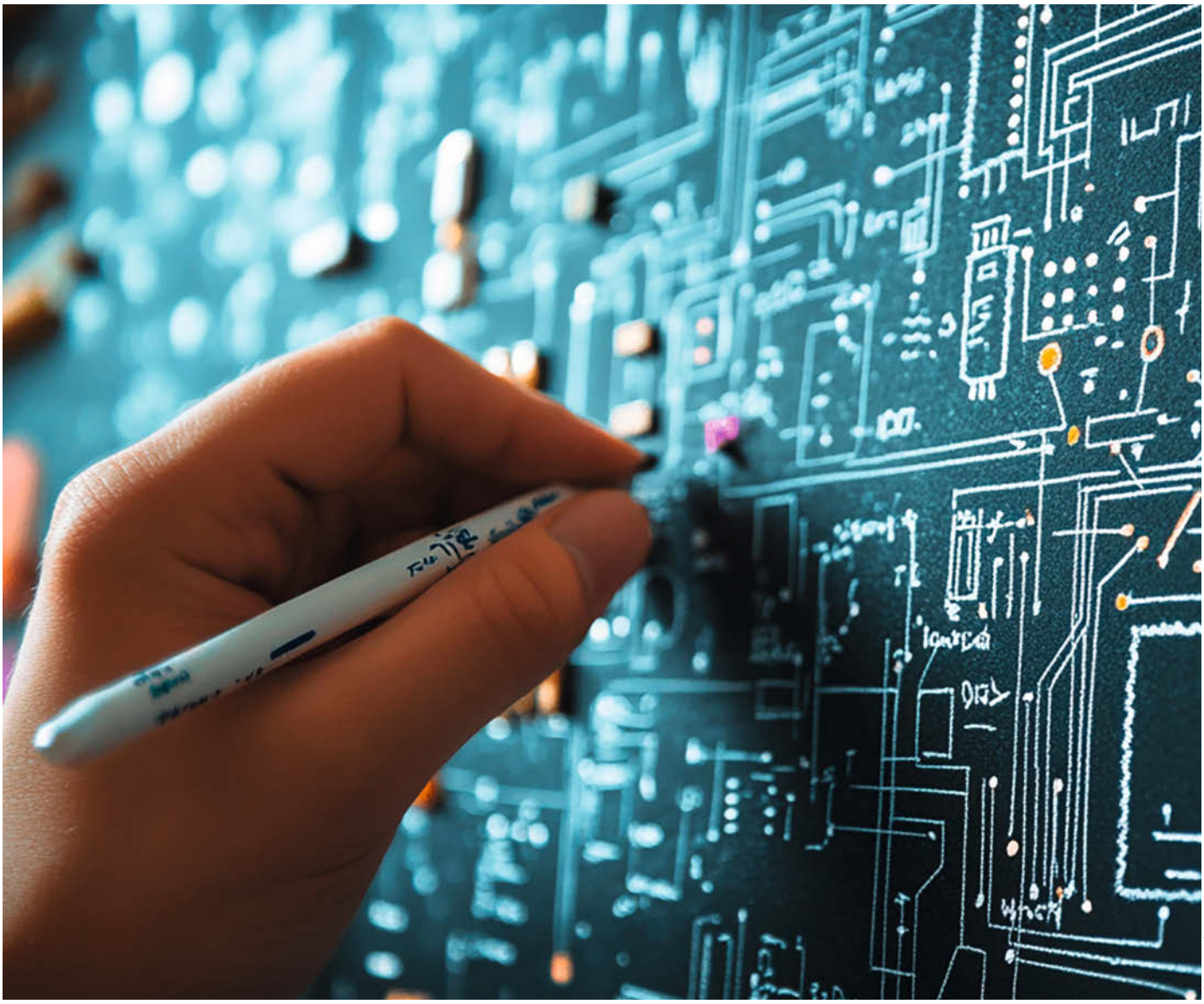


Bild: KI Midjourney

Gratis E-CADs

Eine Platine schafft eine stabile Basis für die Elektronik eines Projekts. Durch die Miniaturisierung können leider immer weniger Bauteile auf Breadboards und Lochraster aufgebaut werden. So wird eine Platine praktisch Pflicht. Mit unserer Übersicht findet ihr die richtige Software für das Erstellen von PCBs.

von Carsten Wartmann

Kurzinfo

» Drei „Klassen“ von E-CADs im Test
» Orientierungshilfe für Einsteiger
» Open Source, kostenlos oder browserbasiert

Mehr zum Thema

» Nikolai Radke, Schimpfolino: Vom Steckbrett zur Platine, Make 1/25, S. 16
» Dr. Stefan Recksiegel, SMD-Löten mit Mini-Heizplatte, Make 1/25, S. 78
» Carsten Wartmann, Platinen aus China – bestückt bestellen, Make 4/23, S. 108

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

Im Internet findet man praktisch zu allen elektronischen Problemen Schaltungen, die man in eigenen Projekten verwenden kann. Manchmal tut es eine fliegende Verdrahtung. Wird es komplexer, testet man die Schaltung auf einem Breadboard. Aber irgendwann soll dann das Projekt zuverlässig in einem Gehäuse seinen Dienst verrichten und eine Platine muss her. Es gibt welche, die wie Breadboards verschaltet sind und auf die man den Breadboard-Entwurf 1:1 übertragen und festlöten kann. Auch Punkt- und Streifenrasterplatinen sind beliebt, so richtig schön und mit professionellem Anstrich ist das aber oft nicht. Bleibt schließlich nur noch, eine Platine am Computer zu entwerfen und fertigen zu lassen.

Das ist gar nicht so schwierig und auch nicht mehr teuer (siehe „Mehr zum Thema“). Aber welches Programm soll man lernen? Wir geben hier einen Überblick über drei sehr unterschiedliche E-CAD- oder EDA-Programme (Electronic CAD oder Electronic Design Automation), die jeweils für eine ganze Klasse von Software stehen. Am Ende dieses Artikels wisst ihr, welche Programmklasse man ausprobieren sollte.

Einfache Platinen kann man selbst ätzen oder isolationsfräsen, inzwischen sind die Preise für professionell gefertigte Platinen aber so niedrig, dass sich das finanziell kaum noch lohnt. Europäische und deutsche Fertiger sind inzwischen auch im Preis den Firmen aus China ebenbürtig. Bei bestückten Platinen sieht es noch etwas anders aus, aber hey, wo bleibt da der Spaß beim Löten?

Wie sind diese Preise möglich? Hier kommt die E-CAD-Software ins Spiel: Sie erzeugt eine maschinenlesbare Datei, die haarklein alles beschreibt, was zum Herstellen nötig ist, sodass im Idealfall die Maschinen beim Fertiger alles allein erledigen können, ohne dass ein Mensch eingreifen muss. Die Aufträge von uns Makern werden in den „Lücken“ zwischen Industrieaufträgen mit Tausenden PCBs gepackt („Pooling“) und sorgen so sogar für eine effiziente Ausnutzung der Maschinen.

Schon lange frei: KiCad

KiCad EDA (so der offizielle Name) ist eine Open-Source-Software zum Zeichnen von Schaltplänen, Simulation von Schaltungen, Erstellen von Leiterplatten (PCB, Printed Circuit Board) und Vorbereitung der Fertigung. Sie besteht aus verschiedenen Einzelprogrammen, die miteinander verknüpft sind. KiCad ist für Windows, Linux, macOS und FreeBSD erhältlich. Tutorials gibt es im Internet zuhauf, in unterschiedlichster Qualität und Umfang.

KiCad ist eine mächtige Software, die ebenso in kommerziellen Produkten Verwendung findet. Die offene und auf professionelle Anwender abgestimmte Ausrichtung macht es Einsteigern nicht immer leicht, in dem Programm Fuß zu fassen: Daher haben die KiCad-Entwickler seit ein paar Jahren begonnen, die Entwicklung besser zu steuern. Der jährliche Release-Zyklus kommt der Kommunikation mit Anwendern und Entwicklern sehr zugute.

Nach dem Start von KiCad wird man von der Projektverwaltung (Bild 1) begrüßt. Das Projekt ist ein Ordner auf der Festplatte und sammelt alle zugehörigen Daten wie Schaltpläne, Board-Layouts oder Teilelisten. Man kann problemlos

Platinen entwerfen

Die Arbeitsweise ist für die drei der hier genannten E-CADs praktisch gleich. Die grundlegenden Arbeitsschritte bestehen aus:

» Projekt anlegen
» Schaltplan entwerfen / zeichnen
» elektrische Regeln (ERC) prüfen lassen
» PCB entwerfen und Verbindungen zeichnen (lassen)
» Designregeln (DRC) prüfen lassen
» Gerber-Dateien erzeugen und Platine fertigen lassen

Der hier unterschlagene Punkt „Footprints zuweisen“ wird unterschiedlich gehandhabt. Ein Footprint beschreibt, wie ein Teil kontaktiert wird und wie es zu montieren ist. In KiCad kümmert man sich um die tatsächlichen Bauformen nach dem Erstellen des Schaltplans, damit können dann etwa allen Widerständen mit einem Klick die Footprints zugewiesen werden. Eagle möchte dagegen, dass man die Footprints beim Setzen von Symbolen im Schaltplan sofort festlegt.

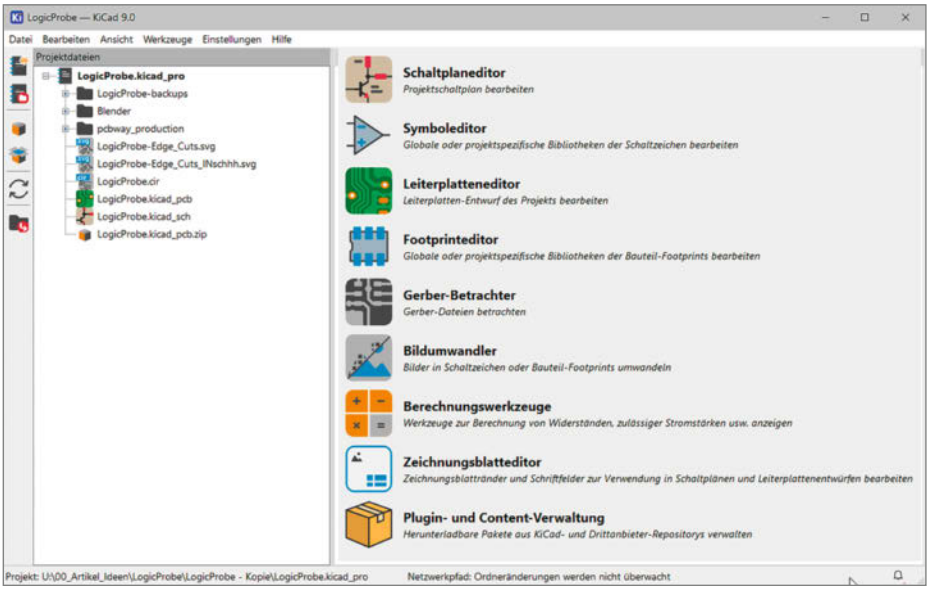


Bild 1: Die KiCad-Projektverwaltung.

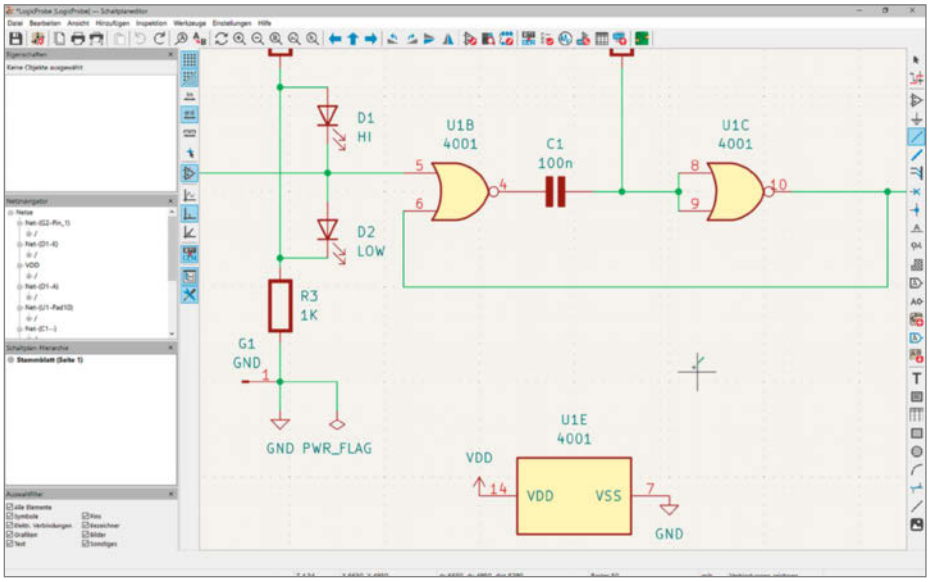


Bild 2: Der Schaltungseditor.

auch nur einen Schaltplan erstellen und auch außerhalb des Projektes bearbeiten. Allerdings birgt dies die Gefahr, dass die Änderungen nicht auf das Board übertragen werden, was zu Inkonsistenzen führen kann.

In dem Projektfenster gibt es auch noch andere Icons, etwa für den Symboleditor, Berechnungswerkzeuge mit Rechnern und Tabelle zur Planung von Schaltungen sowie die Plug-in- und Contentverwaltung. Übrigens sind nur die Icons anklickbar, auch wenn ich regelmäßig versuche, die Namen anzuklicken.

Die Fenster vom Schaltplaneditor und etwa dem Board-Layout können gleichzeitig offen sein, eine Selektion in einem Fenster wird auch im anderen durchgeführt – sehr praktisch auf großen Displays oder mehreren Monitoren.

Der Schaltplaneditor (Bild 2) ist klar aufgebaut, die wichtigsten Befehle ruft man mit einzelnen Tastendrücken auf, etwa mit „W" (für engl. wire, Draht), um eine Verbindung zu ziehen. Damit sind wir auch schon beim Platzieren von Schaltungskomponenten (Bild 3). Solch ein Symbol wählt man im „Symbol wählen"-Dialog (Taste A oder das OpAmp-Icon).

Wie auch bei anderen Programmen muss man sich erst einmal an die Namen der Komponenten und Rubriken gewöhnen, einen Widerstand findet man unter „R" oder „Resistor", da die Bibliotheken leider nur auf Englisch verfügbar sind. An dieser Stelle kann man bereits einen Footprint für das Bauteil auswählen oder dies später über „Footprints zuweisen" in einem Rutsch für mehrere Teile erledigen.

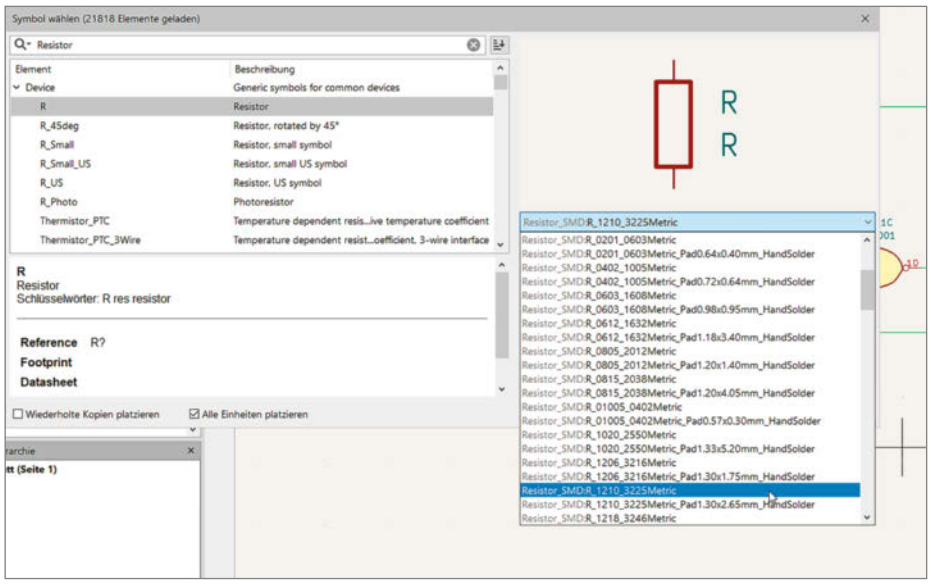


Bild 3: Auswahl von Symbol und Footprint.

Ein regelmäßiger Stolperstein für Einsteiger sind allerdings fehlende Power-Flags in einer Schaltung. Diese speziellen Symbole verwendet man in KiCad, um dem elektrischen Regelprüfer (ERC – Electrical Rules Checker) mitzuteilen, dass eine bestimmte Netzverbindung absichtlich mit einer Stromversorgung verbunden ist. Sie lösen Warnungen bezüglich unversorgter Stromnetze auf. Wie alle anderen Symbole findet man die Power-Flags im „Symbol Wählen"-Dialog (Taste A) in der Rubrik „Power".

Ist der ERC erfolgreich ohne Fehler gelaufen, geht es an das Designen des Boards. Man wechselt mit F8 zum Board-Editor (Bild 4). Das zuerst auftauchende Fenster informiert über eventuelle Fehler (nicht zugewiesene Footprints etwa). F8 ist später auch die Taste, mit der man Änderungen am Schaltplan auf das Board bringt.

Zuerst sind alle Bauteile einfach nur eng beieinander auf der Platinenfläche positioniert. Diese sollte man in einem nächsten Schritt manuell so platzieren, dass die „Luftlinien" (dünne farbige Linien, bisher nicht in Kupfer geroutet) sich möglichst wenig kreuzen. Natürlich sollten in der Regel auch Bauteile, die zusammengehören, nahe zusammen platziert werden (etwa LED und Vorwiderstand). Aber auch die Funktion kann darüber entscheiden, wo man ein Bauteil platziert, z. B. einen Anschluss oder eine LED so, dass man sie auch nach dem Einbau in ein Gehäuse sieht.

Jetzt geht es daran, tatsächlich Kupferbahnen zu ziehen. Standardmäßig hat KiCad keinen Autorouter, der alle Bahnen so zieht, dass alle Verbindungen möglichst ideal verlaufen. Für Einsteiger, die so gar nicht wissen, wo sie anfangen sollen, gibt es daher ein passendes Plug-in (s. a. unten) namens „Freerouting", das ein externes Programm zum Routen benutzt. Damit bekommt man schon mal ein Gefühl, wie Komponenten grundsätzlich gut angeordnet werden können.

Danach kann man Anpassungen vornehmen: Der interaktive „Push & Shove"-Router macht es leicht, Leiterbahnen zu verschieben. Dabei leitet und optimiert er, sodass man oft erstaunliche Wege findet. Komponenten kann man per „Drag" (D) ziehen (Bild 5), und der Router versucht, die Leiterbahnen zu erhalten.

Es gibt auch noch weitere Plug-ins und Erweiterungen für KiCad: Export-Tools für diverse Platinenfertiger wie Aisler, PCBWay oder JLCPCB, weiterhin Automatisierung mit Python und PDF-Export. Die Auswahl ist so bunt wie die Anwender von KiCad. Apropos, falls ihr den Dark-Mode von KiCad sucht: Auch den findet man in den Plug-ins.

Der Export der Platine als 3D-Modell ist in vielen Formaten möglich, für CAD ist sicher STEP am besten geeignet. Ich verwende Blender (Bild 6), um Gehäuse passend zu bauen,

mit dem pcb2blender-Plug-in sind aber auch fotorealistische Renderings in Blender möglich, etwa für Produktwerbung.

Jede EDA-Anwendung muss sich an den verfügbaren Bibliotheken für Symbole, Footprints und 3D-Modellen messen. KiCad ist hier vorn mit dabei, vor allem dank der riesigen User-Community.

Das Erstellen von Symbolen und Footprints ist mit den Werkzeugen in KiCad kein Hexenwerk, wenn man ein Datenblatt lesen kann und sich etwas mit Vektorgrafik auskennt. Inzwischen gibt es Dienste im Web (Ultra Librarian, SnapMagic (SnapEDA) etc., siehe Kurzlink), die automatisch neue Bibliotheken sammeln und für praktisch alle E-CAD-Systeme aufbereiten. Oft wird man dort schneller fündig als beim Hersteller. Aber auch hier steht dann die verwirrende Installation in mehreren Verzeichnissen und Libraries an.

Noch einfacher wird es in KiCad mit dem Plug-in „impartGui“, das Bauteile von den bekanntesten E-CAD-Sammeldiensten installiert. Dabei kann das Plug-in auch Verzeichnisse auf neu heruntergeladene Archive überwachen und die Datenbank aktuell halten.

Mit zahlreichen Entwicklungen und Neuerungen in den letzten Jahren ist KiCad für mich die Software der Wahl für Platinendesign geworden. Open Source verhindert, dass ein Hersteller mal eben den Support einstellt, und die Schnittstellen zu anderen freien Softwarepaketen, sei es Git oder Blender, sind hervorragend.

Der alte Herr: Eagle

Eagle wurde seit 1988 von der Firma CadSoft Computer GmbH entwickelt und mit einer kostenlosen (aber eingeschränkten) Version auch bei Makern und Bildungseinrichtungen beliebt. Autodesk hat Eagle 2016 gekauft und bietet seitdem eine Version, die auf zwei Schaltpläne beschränkt ist, sowie zwei Layer, max. 80 cm^2 große Boards, und ist nur für nicht kommerziellen Gebrauch zugelassen. Seit 2020 gab es keine Updates mehr von Autodesk. Eagle gibt es für Windows, macOS und Linux.

Die Installation ist simpel, auch wenn Eagle sich gerne direkt auf das Systemlaufwerk installieren möchte. Beim Start erscheint ein Login-Fenster für Autodesk, das man aber ignorieren kann.

Eagle präsentiert sich danach mit dem „Control Panel“ (Bild 7), über das man auf die Programmteile zugreift und das die vorhandenen Projekte übersichtlich auflistet. Aus diesem Fenster heraus kann man dann loslegen. Im Prinzip lassen sich auch die Einzelteile wie ein Schaltplan oder Board separat erstellen, aber der empfohlene Weg ist, ein Projekt anzulegen. Nur so ist sichergestellt, dass die Daten zwischen den Teilen des Projekts konsistent bleiben.

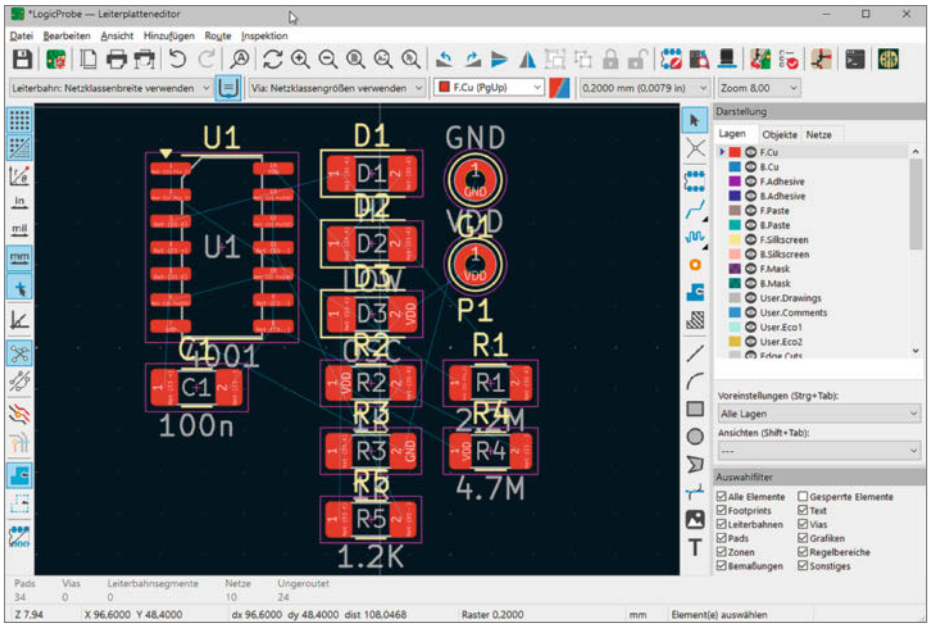


Bild 4: Im Board-Editor erscheinen die frisch aus dem Schaltplan importieren Bauteile.

Die goldene Regel für Eagle ist: Immer Schaltplan und Board-Fenster gleichzeitig offen behalten! Ein grüner Blitz rechts unten in der Ecke des Fensters zeigt, dass alles o. k. ist. Wird er magenta, passt etwas nicht. Wenn ein schwarz-gelbes Banner erscheint, wird es gefährlich! Man sollte auf keinen Fall weiterarbeiten, das fehlende Fenster wieder öffnen oder das Projekt, ohne vorher zu speichern, neu laden!

Die Einstellungen sind vielfältig, aber leider auch unübersichtlich: Von Farben über Raster bis zur Icon-Größe lässt sich viel konfigurieren. Die Icons vergrößere ich auf meinem System immer etwas. Eagle merkt sich die Anordnung der Fenster, was sehr praktisch ist.

Dann geht es los, natürlich mit einem Schaltplan, den man in einem Projekt über das Datei-Menü erstellt. Ob man erst alle Teile auf der Arbeitsfläche platziert und dann miteinander verbindet oder schon Teile der Schaltung miteinander verbindet und dann weitere Bauteile ergänzt, ist wie immer Geschmackssache.

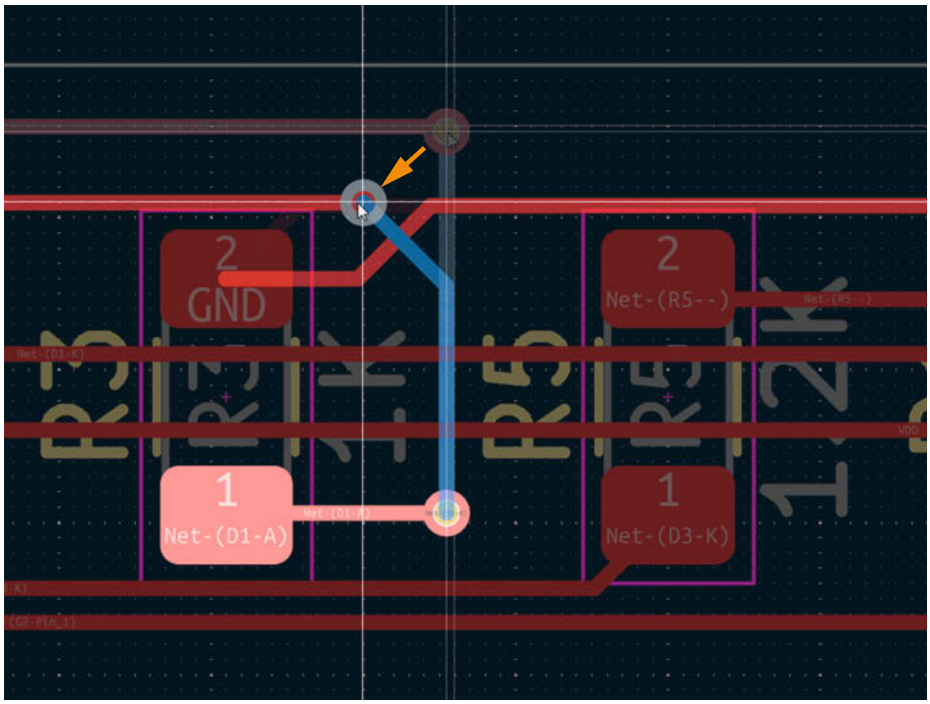


Bild 5: Optimierungen der Leiterbahnen sind schnell durchgeführt. Ein Via (Layerwechsel) wurde entlang des Pfeils geschoben und automatisch die Leiterbahn von R3 angepasst.

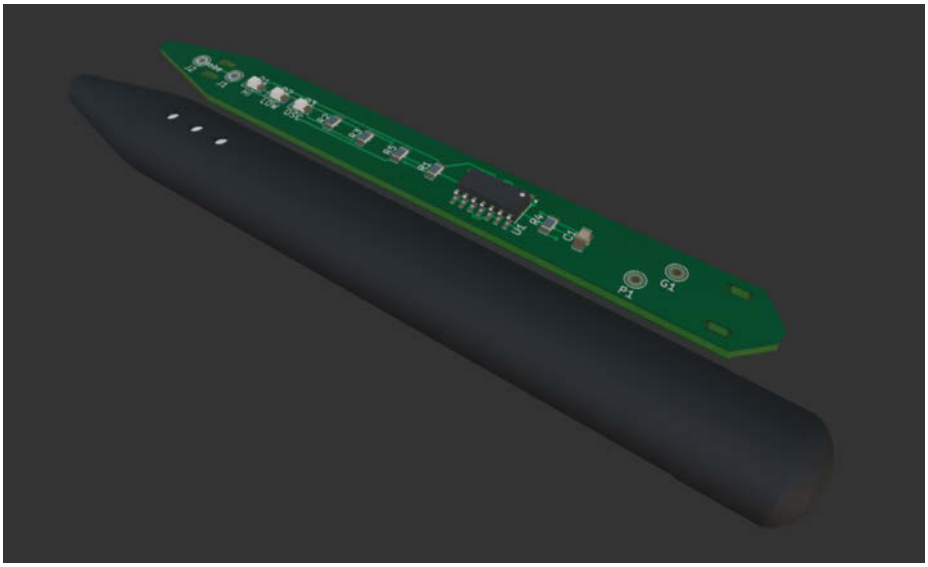


Bild 6: Das Modell der Platine kann in Blender zum Rendern oder der Konstruktion eines Gehäuses benutzt werden.

Mit einem Klick auf das Icon „Bauteil Hinzufügen“ erscheint der ADD-Dialog (Bild 9). Hier kann man nach Teilenamen und Funktionen (etwa „NAND“) suchen und das Teil so aus der großen Liste eingrenzen. In der Bibliothek sind die Teile auch schon mit den verfügbaren Footprints, also den Anschlüssen und Maßen, hinterlegt. In Eagle muss man sich also bereits beim Platzieren im Schaltplan für eine Bauform (THT (Durchsteckmontage) oder SMD (Oberflächenmontage)) entscheiden. Das ist eine etwas andere Arbeitsweise als bei KiCad, wo man sich in dieser Phase bisher nicht darum kümmern muss (es aber kann, wenn man dies bevorzugt). Die Suche ist allerdings sehr scharf. Einen Tippfehler verzeiht sie nicht, und je nach Suchanfrage bekommt man Hunderte oder gar kein Teil zu sehen. Hier muss man sich erst etwas in die Benennung der Symbole einfuchsen, mehrere Suchbegriffe (getrennt durch Leerzeichen) und die Filter bzw. Platzhalter (Wildcards, etwa *,?) und Attribute benutzen.

Fügt man wie im Beispiel (Bild 9) einen IC (integrierten Schaltkreis) mit mehreren identischen Funktionsblöcken hinzu, dann kann man die einzelnen Funktionsblöcke nacheinander in den Schaltplan einfügen.

Für den Einsteiger kommt unweigerlich die Frage auf, wie man diesen denn mit Strom versorgt. Dies wird in vielen Tutorials nicht erwähnt. Nach etwas Suchen fand ich den Befehl „Invoke“, den man im Kontextmenü eines Symbols auf dem Schaltplan findet. Leider muss man oft genau auf das kleine Mittelkreuz im Symbol klicken (rechte Maustaste), was mich anfangs massiv verwirrt hat.

Geholfen hat mir als „Oldtimer“ oft die Kommandozeile in Eagle, mit der man Befehle aufrufen kann. Da kam ein lang vergessenes Autocad-Wissen wieder hervor. Profis konnten so allerdings schon immer Skripte für Routinearbeiten in Eagle schreiben. Die lange Zeit, die Eagle am Markt ist, hilft, da praktisch alle Antworten auf Einsteigerfragen nur eine Suchanfrage im Internet entfernt sind.

Der nächste Stolperstein taucht dann beim Verbinden der Bauteile auf. Man sollte auf keinen Fall die „Line“-Funktion verwenden, wenn man nicht genau weiß, was man tut. Früher war diese Funktion prominent in den Icons vorhanden und führte zu viel Frust, weil Schaltungen nicht so funktionierten, wie gedacht. Wir wollen unsere Bauteile mit „Net“ (Tastenkürzel Alt + N) verbinden. Apropos Tasten: Alle Tastenkürzel werden per Modifier (Strg / Alt / Shift) aufgerufen.

Jetzt sollte man, sofern nicht bereits nach dem Platzieren der Bauteile erledigt, die Werte der Bauteile (Widerstand, Kapazität etc.) eintragen. Dann prüft man seinen Schaltplan mit einem ERC (Electrical Rules Check). Ist alles o. k., kann man ein Board hinzufügen, das dann gleich mit den Bauteilen belegt wird.

Nun geht es an den spaßigen Teil im Board-Editor (Bild 11)! Ja, tatsächlich hat das Entflechten der Verbindungen und das geschickte Platzieren auf dem Board etwas Meditatives wie Sudoku oder ähnliche Beschäftigungen. Der Autorouter kann gut eingesetzt werden, um problematische Stellen zu finden und dort die Ausrichtung und Lage der Bauteile zu ändern. Dabei folgen Verbindungen und Bauteile der Maus.

Danach folgt ein DRC-Check (Design Rule Check), in dem geprüft wird, ob die für die Fertigung festgelegten Grenzen beachtet wurden. Nicht wundern: Der DRC meldet sich nur, wenn Fehler vorliegen. Das kann verwirren, weil scheinbar gar nichts passiert, wenn man „Prüfen“ anwählt. Jetzt muss man noch die Größe und Umrisse des Boards festlegen. Hier sind auch ungewöhnliche Formen mittels DXF-Import möglich. Danach kann man schon mal wagen, die Fertigungsdateien (Gerber, gezippt) anzulegen und zu prüfen.

Eagle hat leider keinen Viewer, in dem man in 2D oder 3D die Fertigungsdateien ansehen könnte. Man muss also einen externen Dienst oder Viewer benutzen, um die Platine zu prüfen. Möchte man die Platinen ohnehin fertigen lassen, bieten natürlich auch die meisten Firmen solch einen Service an. Der Export der Platine in 3D (etwa als STEP-Datei) existiert in Eagle, dazu muss man sich aber für Fusion 360 anmelden, an das diese Aufgabe delegiert wird.

Eagle ist und bleibt beliebt. Es hat über die Jahre eine treue Fangemeinde generiert und

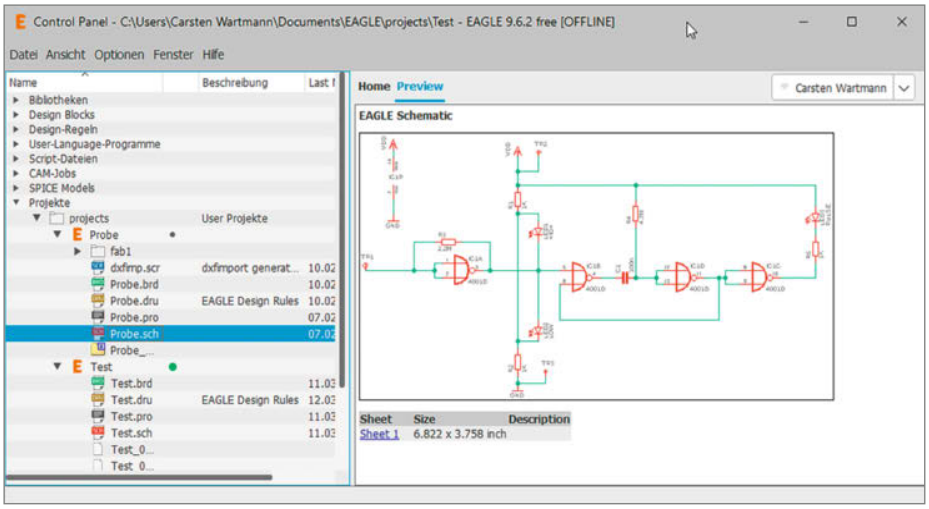


Bild 7: Das „Control Panel“ zeigt die einzelnen Dateien und Vorschauen.

Bild 8: Achtung! Hier droht Ungemach.

auch behalten. Der Kauf von Eagle durch Autodesk ist natürlich schon etwas bedenklich und Autodesk wird auch nicht müde, in Eagle für sein Fusion zu werben. So ist für mich die Zukunft von Eagle etwas zu ungewiss, um hier voll einzusteigen.

Ich persönlich würde Eagle nur noch empfehlen, wenn man damit in früherer Zeit schon viel gemacht und die Arbeitsweise verinnerlicht hat.

China at its best: EasyEDA

EasyEDA (Pro) gehört zur gleichen Unternehmensgruppe wie LCSC, einem großen chinesischen Bauteile-Distributor, und JLCPCB, einem der größten Leiterplattenfertiger in China.

Das kostenlose EasyEDA war lange Zeit der beste Tipp, wenn man in das PCB-Design einsteigen wollte. Komplett ohne Installation (es gibt für beide Varianten allerdings auch Windows- / macOS- / Linux-Apps), aber dennoch für die Zukunft gerüstet. Auch meine ersten Schritte zum PCB-Design fanden mit EasyEDA statt. Inzwischen gibt es parallel ein „EasyEDA Pro", das praktisch eine Weiterentwicklung ist, aber für Einzelpersonen auch „Free Forever" (Zitat von EasyEDA-Website) sein soll. Ein Account ist für beide Versionen nötig.

EasyEDA war bisher immer auf eine Internetverbindung angewiesen, wenn man die Bauteilbibliothek und Teamwork nutzen wollte. Wahlweise gab es noch die Möglichkeit, die Projekte lokal auf dem Rechner statt in der Cloud zu speichern. Durch die Cloud-Funktionen hat sich allerdings eine recht große Community gebildet, die Designs tauscht und wo man in fremden Designs stöbern und diese je nach Lizenz auch direkt weiter nutzen kann.

EasyEDA Pro zielt etwas höher und kann nun ebenfalls komplett offline benutzt werden, was besonders Firmen zu schätzen wissen, die ihre eigenen Bauteilebibliotheken nutzen und ihre Designs schützen wollen. Die Software ist nun praktisch auf dem Stand von EasyEDA, bietet aber neue Funktionen, die selbst ein ambitionierter Hobbyist kaum nutzen wird und die in den Bereich „Digital Fabrication" gehen. Online gibt es eine seitenlange Liste, die die Fähigkeiten der Versionen vergleicht (siehe Link in der Kurzinfo). Ich persönlich habe bisher keinen Grund gefunden, einen Wechsel auf die Pro-Version durchzuführen.

Die Bedienoberfläche von EasyEDA ist hell und klar, die Aufteilung einstellbar (ein Dark-Mode ist mit Extensions zu bekommen). Im Vollbildmodus sieht es wie ein Desktop-Programm aus und bedient sich, wie ich finde, sogar flüssiger als die lokal laufende Pro-Version. Das mag aber mit einer anderen Grafikkarte besser sein.

Man kann EasyEDA auch im Browser auf zwei Monitoren benutzen, die Fenster sind synchronisiert. Wenn man im Schaltplan etwas anwählt, wird es auch im PCB-Editor ausgewählt und Änderungen vererben sich zwischen Schaltplan und Board. Auf einem Monitor helfen die Reiter oben im Hauptfenster, zwischen den einzelnen Aufgabenbereichen wie Schaltplan, Board oder 3D-Viewer umzuschalten. Mit zwei oder einem sehr breiten Monitor kann man durch einen Rechtsklick auf einen Reiter diesen aber auch in einem neuen (Browser-)Fenster öffnen. Die Illusion, eine echte Anwendung zu bedienen, verpufft, wenn man aus EasyEDA heraus etwa den LCSC-Electronics-Knopf anklickt, um die Lieferbarkeit eines Teils zu checken, und sich diese Website dann in dem Fullscreen-Fenster öffnet. Dann muss man mit F11 umschalten, um wieder zu seinem EasyEDA zurückzukommen. Und Dialogfenster kann man natürlich auch nicht aus dem Browserrahmen herausschieben.

Auch EasyEDA kann einem den komplexen Prozess des Board-Designs nicht komplett abnehmen, aber gerade als Einsteiger findet man sich irgendwie immer durch. Die Funktionen sind moderner implementiert, Klicks und Menüs tun, was man erwarten würde. Man muss am Anfang nur die vielen Optionen und Informationen, die in Tabs und Fenstern auf einen einprasseln, ignorieren und erst einmal nur die Basistools verwenden.

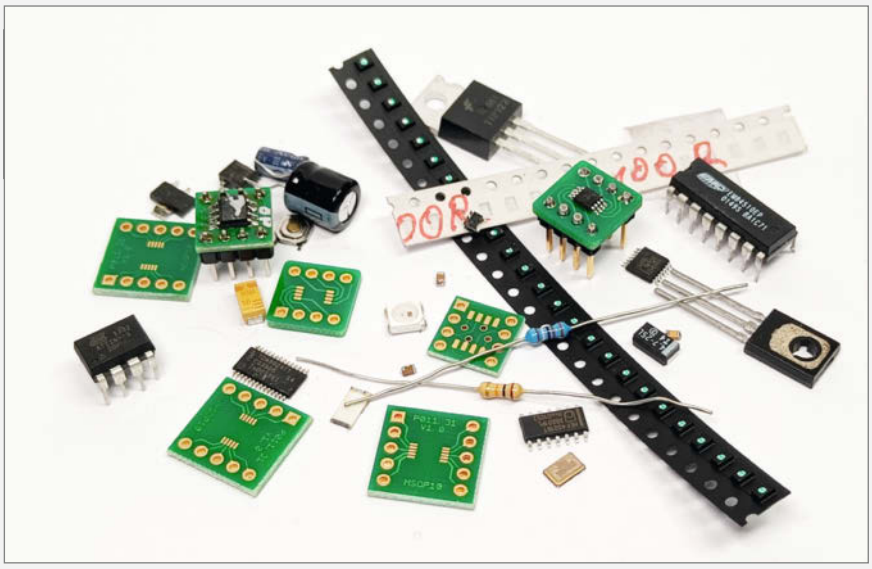
Die wichtigsten Werkzeuge im Schaltplaneditor (Bild 13) sind auf einem Tastendruck verfügbar: „W" zieht z. B. eine Verbindung und die Selektion von Einzelteilen erfolgt mit einem Klick. Möchte man mehrere Elemente auswählen, zieht man einen Rahmen mit gedrückter linker Maustaste und schon kann man das Ganze verschieben, duplizieren oder Ähnliches. Das ist einfach und intuitiv. Ein Symbol hinzuzufügen, ist auf Umschalt + F gelegt. Hier muss man sich durch teils lange Listen scrollen, da dort jede Variante und Footprint existiert. Immerhin hilft die blitzschnelle Vorschau von Symbol und Footprint bei der Wahl.

THT oder SMD?

Bedrahtete Bauteile (THT, Through-Hole-Technic) kommen meinen schlechter werdenden Augen zugute, allerdings finde ich das Löten von nicht zu kleinen SMD-Bauteilen (minimal 805) inzwischen angenehmer als THT löten und es gelingt mir zunehmend schneller: keine Beinchen biegen, keine Drahtenden abschneiden, die Platinen sind leichter zu reinigen und die ebene Unterseite und niedrigen Teile sparen viele Millimeter Bauhöhe.

Da meine Schaltungen selten perfekt sind, kommen andererseits oft Breadboards oder Rasterplatinen zum Einsatz, und diese nehmen nun mal nur THT-Teile auf. Und ja: Messen an einer THT-Schaltung ist viel einfacher und sicherer, da die Bauteile-Beinchen leicht erreichbar sind.

Neuere Chips und Sensoren gibt es aber teilweise gar nicht mehr in der THT-Bauform. Mikrocontroller verwenden wir meist als komplette Boards und auch für Sensoren findet man viele Adapter, die mit Pin-Headern als THT eingesetzt werden können. Auch für einzelne Chips gibt es Adapterplatinen, die einen SMD-Chip auf das DIP-Maß bringen.



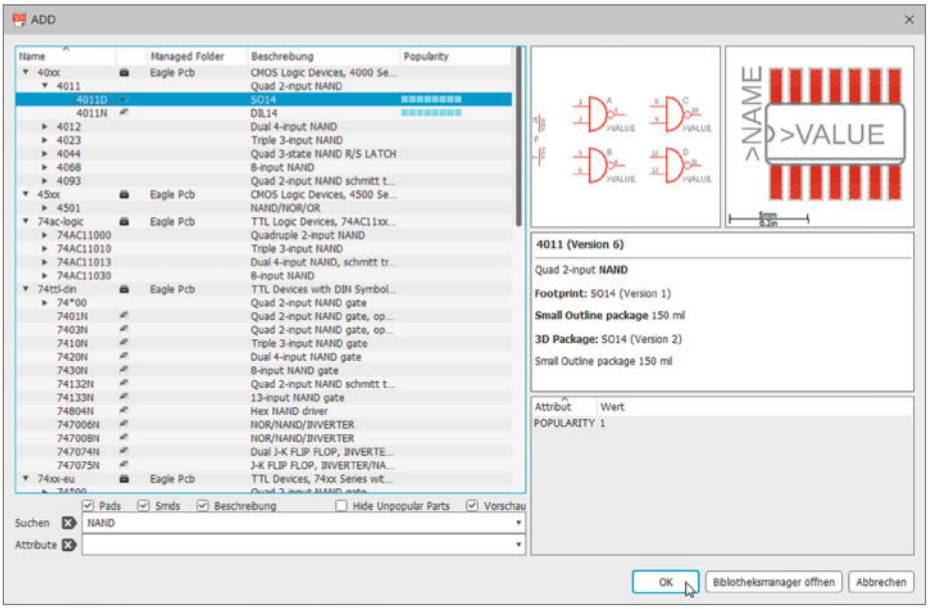


Bild 9: Auswahl von Symbol und Footprint: In Eagle gibt es für jede Variante einen Eintrag.

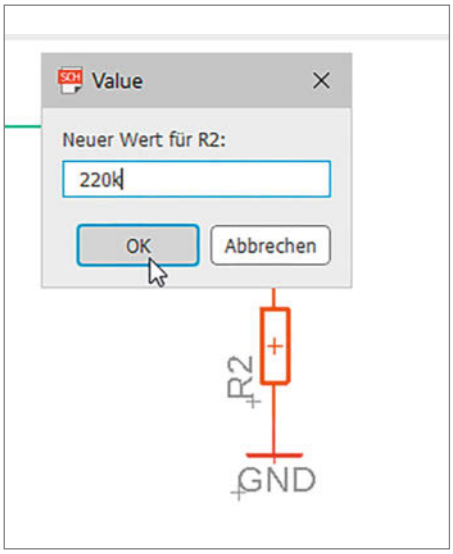


Bild 10: Eine gute Strategie ist es, mehrfach vorkommende Werte einzutragen und dann das Symbol zu kopieren.

Die Dokumentation ist teils schwierig zu finden und zu lesen. Oft landet man in einem Forum, um dann von dort zu allgemeinen Tutorials zu gelangen. Dann wird es manchmal unklar, ob es nun um die STD- oder Pro-Variante geht. In der offiziellen Dokumentation sollte man das ausführliche Tutorial zuerst mindestens mal durchspielen, um den (auch grafisch gezeigten) Produktionsfluss kennenzulernen. Der Star der Dokumentation ist aber hier auch YouTube: Natürlich gibt es Inhalte unterschiedlicher Qualität, aber Video ist bei solch einem komplexen Programm sicher die beste Methode, um zu lernen.

Der Board-Editor ist auch gut zu bedienen. Es gibt einen Autorouter in der Cloud, der für meine kleinen Designs immer gut und schnell funktioniert. Alternativ kann man auch lokal einen Autorouter installieren oder Autorouter-Daten ex- und importieren.

Eine integrierte 3D-Ansicht (Bild 15) hilft besonders, wenn eine Platine mit Nicht-Standard-Maßen oder -Form erzeugt werden soll. Auch Ausschnitte und Bohrungen sind so leicht zu- und einzuordnen.

Die Anbindung zum Platinenfertiger JLCPCB ist natürlich perfekt. Inzwischen ist es auch nicht mehr so teuer, Platinen direkt bestücken zu lassen. Dabei sollte man nur darauf achten, dass möglichst nur Standardteile von LCSC benutzt werden, jedes „Extended"- oder THT-Teil wird teurer. Alternativ kann man auch direkt Teile anhand einer Stückliste mitbestellen, spart sich einmal Porto und kann sich sicher sein, dass die Teile genau zu den Footprints passen.

Mit Plug-ins (hier Erweiterungen genannt) oder über normale Gerber-Dateien kann man aber auch problemlos bei anderen Fertigern bestellen. Auch hier merkt man kaum einen Unterschied zu einer lokal installierten Anwendung. So kann man etwa einen SVG-Import

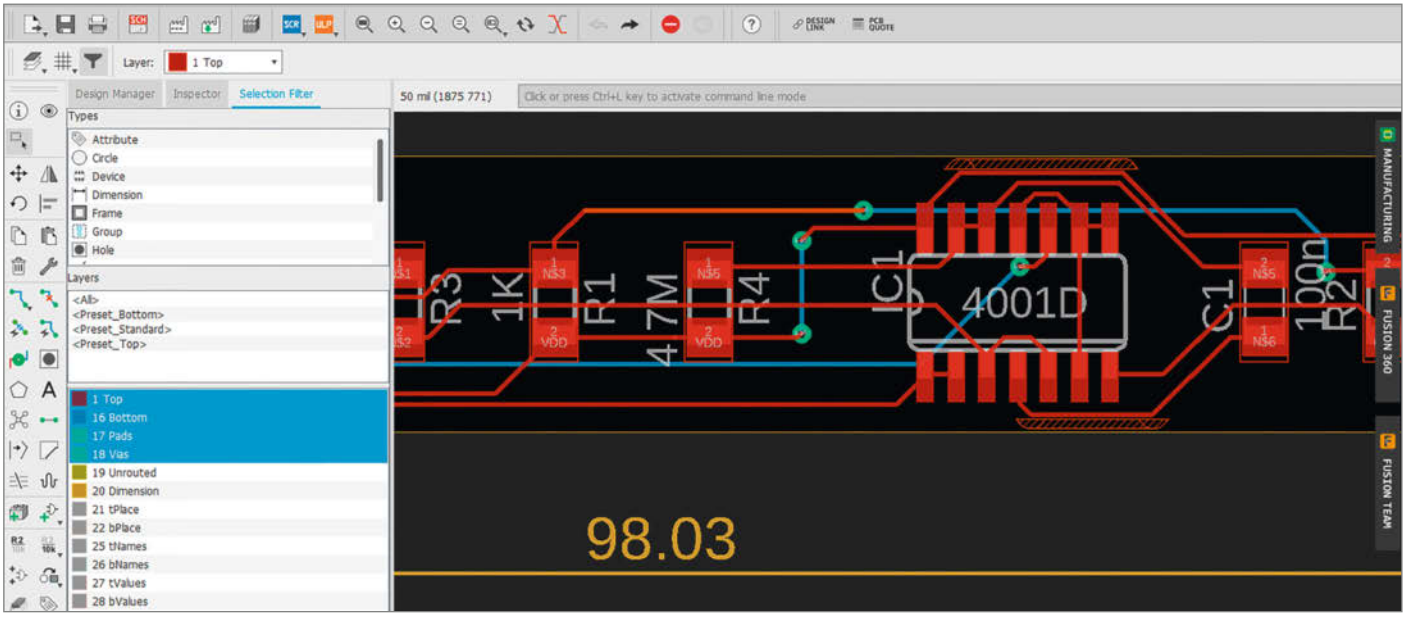


Bild 11: Der Board-Editor von Eagle.

Die drei E-CAD-Klassen im Vergleich

	KiCad EDA	Autodesk Eagle	EasyEDA STD
URL	kicad.org	autodesk.com/products/eagle	easyeda.com
Lizenz	Open Source (GPLv3)	Kostenlos (beschnittene Version)	Kostenlos (Premium mit Abo)
Plattformen	Windows, macOS, Linux	Windows, macOS, Linux	Webbasiert (und Apps für Windows, macOS, Linux)
Im- / Export	Gerber, DXF, STEP, IDF, SVG, SPICE	Gerber, DXF, STEP, IDF, Spice	Gerber, DXF, SVG, Eagle, Altium Designer
3D-Board View	Ja, sehr gut	Nur mit Autodesk Account/Fusion	Ja, mittelgut
Gerber View	Integriert	Nur mit Autodesk Account/Fusion	Integriert
Autorouter	Mit Plug-in	Ja	Cloud oder lokal
Bibliotheken	> 15.000 Bauteile + Community-Bibliotheken	> 10.000 Bauteile + Erweiterungen	> 200.000 Bauteile (direkter Zugriff auf LCSC-Katalog)
Boardgröße	Unbegrenzt	80 cm^2	Unbegrenzt
Anzahl Layer	Unbegrenzt	2 Signallagen	6 Layer
Cloud-Speicher	Nein (lokale Dateien, Git)	Ja, mit Autodesk-Account	Unbegrenzt
Team-Kollaboration	Mit Git/GitHub möglich	–	Ja, Community/Share/Co-Edit
Simulation	Einfache SPICE-Simulation	Einfache SPICE-Simulation	Erweiterte Schaltkreissimulation
Direkte Fertigung	Ja (Plug-ins von PCB-Fertigern)	Ja, mit ausgewählten Partnern	Ja, direkte Integration mit JLCPCB und Plug-ins
Update-Häufigkeit	Regelmäßig (jährlich, nightly)	Seit 2022 kein Update	Häufig
Lernkurve	Mittelsteil (die vielen Funktionen können Einsteiger überfordern)	Mittelsteil (Workflow teils inkonsistent, altertümlich)	Flach
Community-Support	Sehr aktiv, umfangreich	Groß, etabliert	Wachsend, mit Fokus auf asiatischen Markt

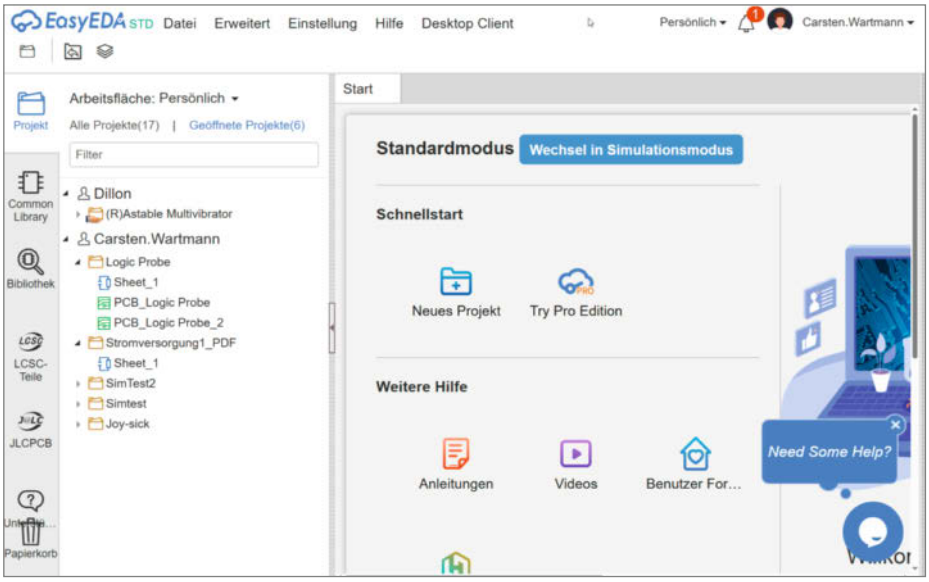


Bild 12: Die EasyEDA-Projektverwaltung mit Links zu Hilfe, Community und Werbepartnern.

nachrüsten, der dann erlaubt, Board-Umrisse, Grafiken und Ähnliches zu importieren.

Ich kann EasyEDA für den Einstieg und gelegentlichen Gebrauch sehr empfehlen. Man muss nichts installieren, keine Updates einspielen und die Projekte sind auf jedem Rechner mit Internet und Browser verfügbar. Die Software ist einfach zu bedienen, es gibt eine große Community und viele Tutorials. Will man es lokal und sicher, sollte man natürlich nicht zu einer ausländischen Cloud-Anwendung greifen oder in China Platinen bestellen.

Die Außenseiter

Soweit zum Mainstream, aber vielleicht sucht ihr ja etwas ganz anderes? Auch lieber Open Source oder seid ihr kommerzielle Programme gewohnt? Daher hier mal ein paar interessante Alternativen:

LibrePCB hat sich den Slogan „Create electronics the easy way" gegeben und vor Kurzem die Version 1.2.0 erreicht. Ziel ist es, dem Einsteiger eine logische und leicht erlernbare Oberfläche zu bieten, aber auch dem Fortgeschrittenen ein schnelles Arbeiten zu ermöglichen. Der Wechsel zwischen Schaltplanentwurf, Bauteilanpassung, Leiterplattenansicht und Bestellvorgang soll schnell und reibungslos erfolgen. Seit Version 1.2.0 kann LibrePCB KiCad-Bibliotheken importieren. Vielleicht der Open-Source-Einsteiger-Tipp für alle, denen KiCad zu mächtig ist. Fertige Installer gibt es für Windows, macOS, Linux und BSD.

VeroRoute ist etwas anders. An sich für die hohe Schule der Lochstreifenplatinen entwickelt, ist es längst diesem alleinigen Anwendungszweck entwachsen und designt nun auch herkömmliche PCBs. Es ist Open Source, fertige Installer für Windows, Linux und kurioserweise auch Android stehen zur Verfügung. Daneben gibt es die C++/QT6-Quellen. VeroRoute hat einen interaktiven Autorouter, Leiterplatten können allerdings nur zwei Lagen haben, eine Stückliste (BOM) wird vom Programm erstellt. Was es nicht hat, ist ein Schaltplaneditor: Hier muss man eine Netlist aus TinyCAD, gEDA oder KiCad verwenden.

Die kostenlosen Versionen von großen kommerziellen EDAs, gern auch „Maker

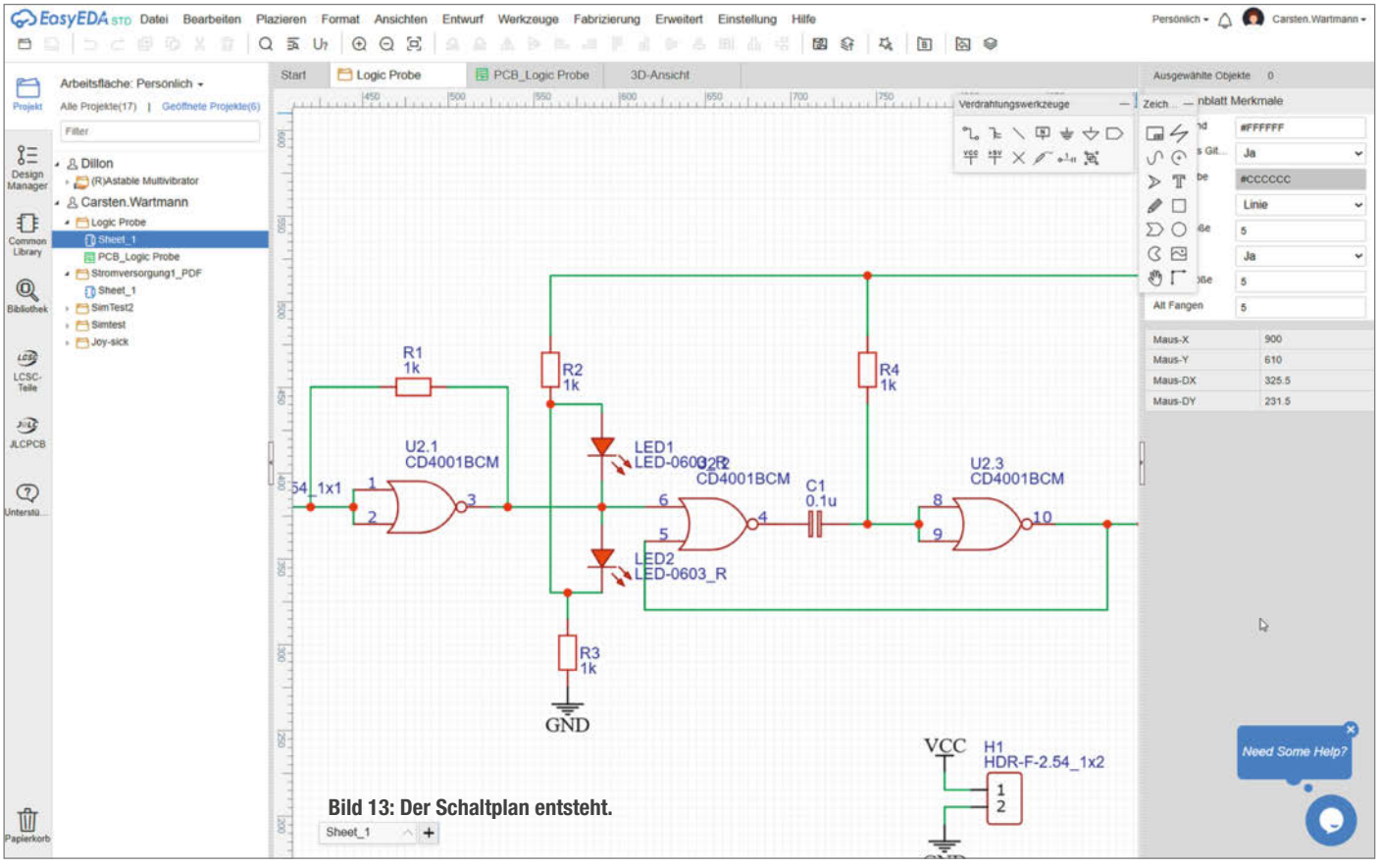
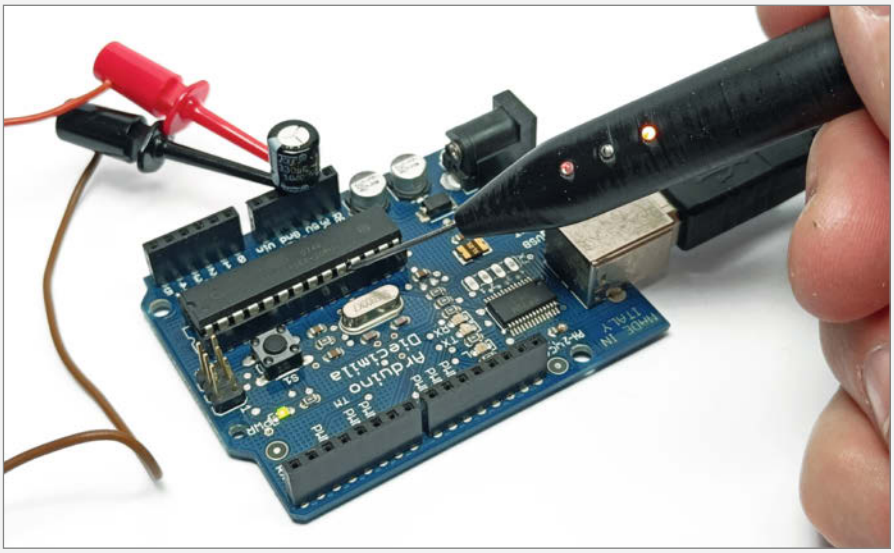


Bild 13: Der Schaltplan entsteht.

Logic Probe

Eine der Platinen, die ich zum Test verwendet habe, entstand aus der Not heraus: Im Digitalteil meines alten Hameg HM205-2 tritt ein Fehler auf. Nach dem Kauf vor ein paar Jahren fand ich den damals defekten Chip mit einem USB-Oszilloskop. Das war aber sehr zeitraubend. Für die Fehlersuche in (TTL-)Digitalschaltungen ist eine sogenannte Logic Probe eine feine Sache. Sie zeigt mit drei LEDs an, ob ein Signal im High-, Low- oder Floating-Zustand ist. Zusätzlich werden periodische Signale oder kurze Bursts durch ein Blinken einer LED angezeigt. Leider sind durch moderne Logic-Analyser diese Geräte rar geworden und gute Probes, etwa von HP (Hewlett Packard), sehr teuer. Also selbst bauen!



Damit bot es sich an, das Erstellen einer Platine anhand dieser Logic Probe durchzuspielen. Es dauerte nicht zu lange, den Schaltplan zu zeichnen. Im PCB-Editor muss man eine Form für das Ausfräsen der Platine zeichnen oder importieren. Die Platine habe ich dann als 3D-Objekt exportiert und für den 3D-Druck eines Gehäuses verwendet.

Editionen“ genannt, sind sicher ein Teil Werbung. Wem die Arbeitsweise gefällt und wer schon viel Zeit in eine Software gesteckt hat, kauft eventuell die kommerzielle Version. Hier eine kleine Auswahl der Programme.

Circuitmaker baut auf der Altium-Technologie auf, ist kostenlos und zielt deutlich auf Maker ab, die vielleicht Altium auf der Arbeit haben. Die Community-Funktionen sind ausgefeilt. Es gibt einen Account- und Registrierungszwang.

DesignSpark PCB (RS Components) „Maker Edition“ erfordert eine Registrierung mit E-Mail. Das Ganze kommt aber zusätzlich mit DesignSparc Mechanical und dem DesignSpark Circuit Simulator und wird so zum kompletten Produktdesigner.

DipTrace sieht ebenfalls interessant aus. Es gibt viele gestaffelte Kaufoptionen von Trial bis Full und kein Abomodell. Spezielle Non-Profit- und Education-Versionen mit mehr Fähigkeiten und günstiger als die normalen Varianten werden auch angeboten.

Und dann gibt es noch **Fritzing**: Benutzt wird es meistens, um Breadboards zu planen und die beliebten einfachen Steckanleitungen für Projekte zu erstellen. Aber das Programm hat tatsächlich einen Teil, mit dem man Platinen erstellen kann. Funktioniert die Schaltung auf dem Breadboard, so kann man direkt daraus eine zweilagige Platine erstellen und sogar mit einem Autorouter arbeiten. Dabei stehen als Platinenform Arduino-Uno-Shields oder freie Formen zur Verfügung. Am Ende wird eine normale Gerber-Datei exportiert. —*caw*

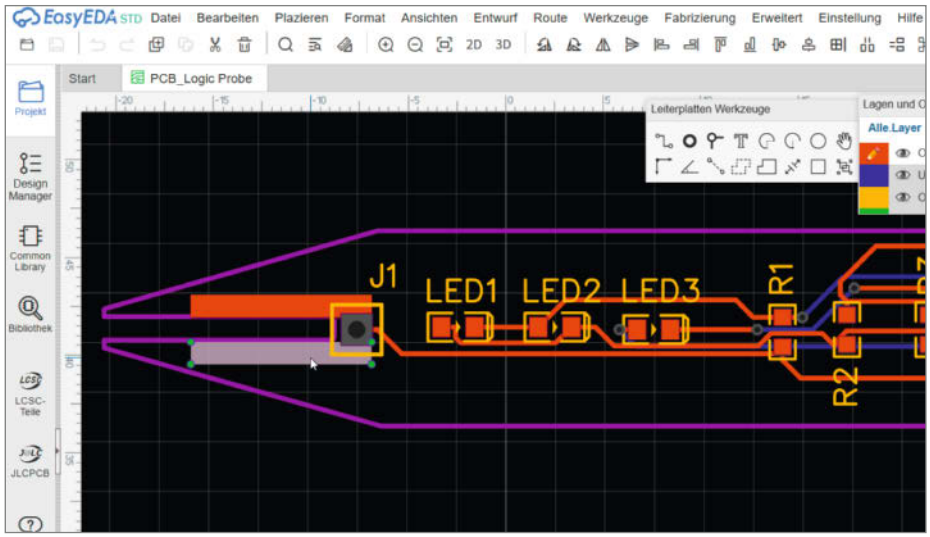


Bild 14: Der Board-Editor bedient sich wie ein modernes Grafikprogramm.

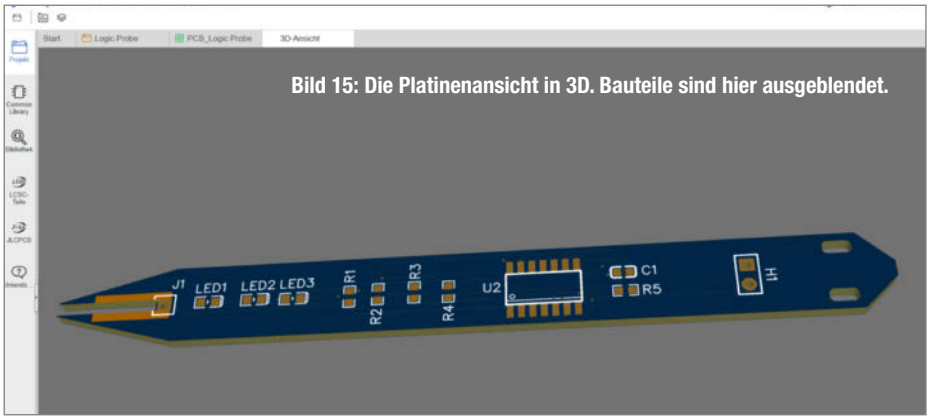


Bild 15: Die Platinenansicht in 3D. Bauteile sind hier ausgeblendet.

Eierlege-Automat Deluxe

Passend zu Ostern bringen wir euch einen bunten Eierlege-Revolver, der auf Knopfdruck ein Ei legen kann. Der Automat kann genauso gut mit Süßigkeiten wie mit Hühnereiern geladen werden.

von Marcus Hansson

Kein Kind glaubt mehr wirklich an den Osterhasen. Um Kinder heute zu begeistern, braucht es mehr, zum Beispiel einen lebensgroßen, beweglichen, mechanischen Strauß, der Eier legen kann. Dieses Ungetüm haben wir in der Make 5/24 vorgestellt (Link in der Kurzinfo). In diesem Artikel konzentrieren wir uns auf einen einzigen Teil des Vogels, nämlich den Eierrevolver. Wir zeigen, wie er funktioniert und wie man ihn nachbauen kann.

Insgesamt passen sechs 10 cm lange Eier in das Magazin. Auf Knopfdruck wird eines davon „gelegt", indem sich das innere Magazin dreht und die Kammer, in der sich das Ei befindet, eine nach unten gerichtete Öffnung erreicht. Ein solcher Eierspender ist nicht nur auf Festivals und als Körperteil eines Vogelstraußes nützlich, sondern kann auch in der heimischen Küche als Eieraufbewahrungslösung oder als Highlight auf dem Kindergeburtstag eingesetzt werden.

Kurzinfo

- Bauanleitung für einen Eierrevolver
- Für Kindergeburtstage oder zum Frühstück
- Antrieb durch Elektromotor

Checkliste

Zeitaufwand:
Ein Wochenende

Kosten:
180 Euro

Mehr zum Thema

- Heinz Behling, Lasern mit Lightburn, Make 1/24, S. 24
- Marcus Hansson, Laufmaschinen konstruieren, Make 7/24, S. 58
- Heinz Behling, Schneiden und gravieren, Make 4/18, S. 126

Werkzeug

- **Lasercutter**
- **Lötkolben**
- **Maulschlüssel**
- **Bohrmaschine**
- **Metallbohrer** 6–10 mm
- **Cuttermesser**
- **Metallsäge**

Material

- **2 Acrylplatten** 5 mm stark, 36×36 cm
- **2 Acrylplatten** 5 mm stark, 32×32 cm
- **1 Acrylplatte** 5 mm stark, 30×30 cm
- **Klarsichtfolie** aus Polycarbonat 1,2 mm, 12×120 cm
- **6 M6-Gewindestangen** 135 mm
- **6 M6-Gewindestangen** 105 mm
- **30 M6-Muttern** selbstsichernd
- **12 50-mm-M6-Schrauben** mit Flachkopf
- **Servo oder anderer Motor**
- **Acrylglasrohr** 80/76 mm (außen/innen), 600 mm lang
- **Rollenhebel**
- **2 Flanschlager** KFL000
- **10-mm-Achse** 150 mm lang
- **2 300-mm-Flachstahl** 5×20 mm
- **3 400-mm-Vierkantstahlprofile** 40×40 mm oder stabile Holzplatte als Standfuß
- **Acrylkleber**
- **6 Acryleier** 100 mm lang

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

Erster Versuch

Die erste Version des Eierrevolvers habe ich in wenigen Stunden aus Spanplatten, dünnem Stahlblech und einem braunen, schäbigen Plastikrohr zusammengebaut (Bild 1). Obwohl die Konstruktion in vielerlei Hinsicht sehr mangelhaft war und viele Schwachstellen aufwies, hat sie doch bewiesen, dass die Grundidee funktioniert. Sie sieht ein kreisrundes Gehäuse vor, das über eine Achse in der Mitte einen rotierenden Kern wie in einem Revolver aufnimmt.

Der rotierende Kern besteht aus sechs Röhren, die sternförmig um die zentrale Achse angeordnet sind. In die Röhren werden die Eier gelegt. Durch die äußere Umhüllung können die Eier nicht unkontrolliert aus ihren Kammern fallen. Außerdem ist an einer Seite des Gehäuses ein kleiner Motor befestigt. Der Motor dreht ein Gummirädchen, das durch eine Öffnung in der Wand an das innere, drehbare Magazin anstößt. Das Gummirädchen dreht das Magazin.

Kein Uterus

Nach dem ersten Versuch mit dieser Konstruktion aus Spanplatten und Blech begann die Arbeit an einer verbesserten Version des Eierrevolvers. Das Projekt habe ich oft als „Uterus" bezeichnet, bis ich erfuhr, dass Vögel keinen Uterus haben, sondern ihre Eier über den Legedarm und die Kloake ablegen. Aus diesem Grund bleiben wir bei der Bezeichnung Eierrevolver.

Tinkercad schnell beherrschen

Als Hauptmaterial habe ich mich für 5-mm-Acrylglas entschieden. Es lässt sich sehr gut mit dem Lasercutter bearbeiten und sieht besonders in Verbindung mit der LED-Beleuchtung cool aus. Alle für den Bau benötigten Acrylteile sind in Bild 2 dargestellt. Gezeichnet habe ich sie mit der einsteigerfreundlichen CAD-Software Tinkercad. Obwohl ich vorher kaum Erfahrung damit hatte, konnte ich mir die für meine Zwecke notwendigen Schritte schnell aneignen.

Obwohl Tinkercad nur wenige Werkzeuge besitzt, lassen sich dennoch auch komplexere Modelle mit ihnen erstellen. Dies ist möglich, indem man Grundformen wie Zylinder, Rechtecke, Kegel usw. zusammensetzt, um die gewünschte Form zu erhalten. Alle Formen können als „Solid" addiert oder als „Hole" subtrahiert werden. Das Ergebnis ist eine intuitive, kreative, puzzleartige Arbeitsweise, die wirklich Spaß macht.

Einmal im Kreis

Für den Revolver musste ich unter anderem mehrere Löcher in einem Bauteil kreisförmig anordnen. Das kann man schnell machen, indem man eine Bohrung dupliziert und verschiebt, sodass sie in einer gewissen Entfer-

Bild 1: Die erste notdürftig zusammengebastelte Version des Eierrevolvers.

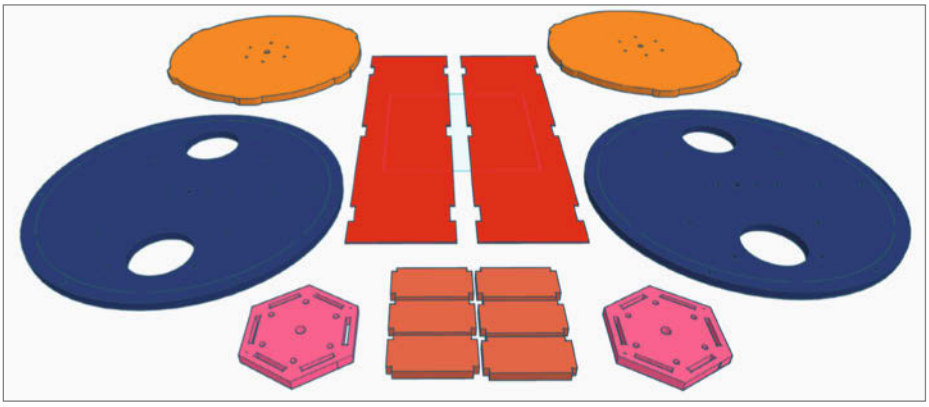


Bild 2: Die beiden roten Platten wurden aus 1,2-mm-Klarsichtfolie geschnitten; alle anderen Teile aus 5-mm-Acrylglas.

nung nebeneinanderliegen. Dann markiert man beide Bohrungen und fügt sie zu einem Objekt zusammen (Bild 3). Dadurch erhalten sie einen gemeinsamen Mittelpunkt, um den sie sich drehen lassen. Dupliziert man dieses neue Objekt und dreht es um 60 Grad um diesen Punkt, so erhält man vier Löcher, die alle kreisförmig angeordnet sind (Bild 4). Fügt man noch mal zwei weitere Löcher hinzu, die um weitere 60 Grad gedreht werden, hat man schließlich insgesamt sechs Löcher gleichmäßig in einem Kreis verteilt (Bild 5).

Eine weitere kleine Herausforderung war es, wirklich runde Kreise zu zeichnen, da ein Kreis in Tinkercad bei genauerer Betrachtung aus 64 Seiten besteht. Für kleine Kreise wie ein Bohrloch ist das gut genug, denn dann sieht es für das Auge perfekt rund aus. Aber für die Seiten des Eierrevolvers mit seinen 355 mm Durchmesser ist das Endergebnis nicht wirklich befriedigend.

Glücklicherweise gibt es einen Trick, um diese Einschränkung zu umgehen. Die Lösung besteht darin, den Kreis zu duplizieren und die Kopie um einige Grad um ihren Mittelpunkt zu drehen, sodass zwei Kreise mit jeweils 64 Seiten übereinanderliegen. Anschließend muss man nur die beiden Kreise markieren und zu einem Objekt zusammenfügen. Dann hat man auf einmal 128 Seiten. Diesen Schritt kann man mehrmals wiederholen, bis der Kreis fast perfekt rund erscheint.

Nach dem Modellieren habe ich alle Teile in dem Format SVG gespeichert und über Lightburn in unseren Lasercutter eingespeist. Zu diesem Projekt gibt es insgesamt fünf SVG-Dateien, die über den Link in der Kurzinfo heruntergeladen werden können.

Zusammenbau des Kerns

Nachdem wir alle Teile gelasert haben, können wir mit dem Zusammenbau beginnen. Als Erstes nehmen wir eine der beiden großen Platten mit den kreisförmig angeordneten Nocken entlang des Umfangs. Die 105 mm langen Gewindestangen bekommen an einem Ende eine selbstsichernde Mutter und werden dann in die sechs kreisförmig angeordneten Löcher in der Mitte der Platte geschoben. Dann wird eine der beiden sechseckigen Platten aufgesetzt, mit den Gewindestangen durch die sechs Löcher. Als Nächstes steckt man die sechs rechteckigen Platten mit den winzigen Aussparungen in den Ecken in die Schlitze der sechseckigen Platte (Bild 6). Darauf kommt eine weitere sechseckige Platte und die zweite große runde Platte.

Um das Ganze zusammenzuhalten, setzen wir Muttern auf die Enden der Gewindestangen. Für den nächsten Schritt brauchen wir sechs 10 cm lange Stücke von dem Plexiglasrohr. Diese habe ich mit meiner Kreissäge zugeschnitten (Bild 7).

Die Rohrstücke werden dann zwischen die beiden äußeren großen Platten geschoben, bis sie die Platten im Kern erreichen (Bild 8). Damit alle sechs Rohre bis zum Kern durchgehen, müssen sie auf der Außenseite leicht angeschrägt werden. Das macht man am besten mit einem Dremel. Anschließend werden die Rohre mit Acryl- oder Epoxidkleber befestigt. Nun ist der innere Kern fertig.

Das äußere Gehäuse

Für den nächsten Schritt nehmen wir eine der beiden größeren Acrylscheiben (ohne Nocken), stecken die sechs 135 mm langen Gewindestangen in die Löcher am Rand und befestigen sie auf beiden Seiten der Scheibe mit selbstsichernden M6-Muttern (Bild 9).

Mir war wichtig, dass die Achse stabil gelagert ist, damit die Drehung möglichst leicht geht. Daher habe ich eine 10-mm-Stahlstange und zwei Flanschlager verwendet, die man mit je zwei M6-Schrauben an den Außenseiten der großen Acrylscheiben befestigen muss. Die beiden 5×20-mm-Flachstahlteile (unten in Bild 11 und oben in Bild 12 zu sehen) werden nun ebenfalls an den Außenseiten mit je drei Schrauben befestigt. Diese dienen später als Halterung für die gesamte Konstruktion. Achtet darauf, dass die verwendeten Schrauben

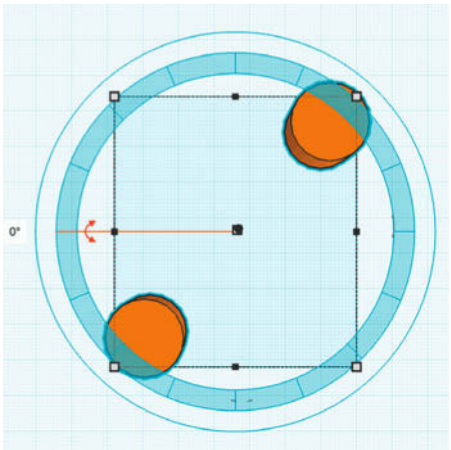


Bild 3: Die beiden Löcher werden markiert und zu einem Objekt zusammengefügt, …

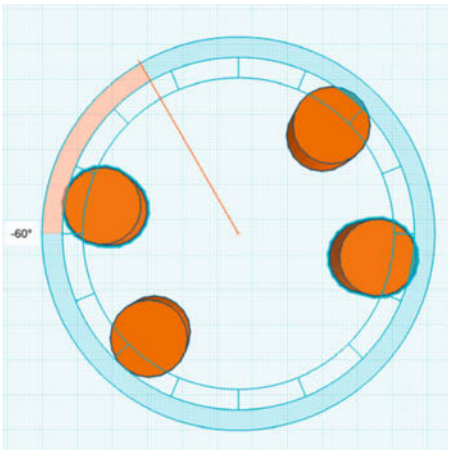


Bild 4: … dann dupliziert und um 60 Grad gedreht, damit vier Löcher entstehen, …

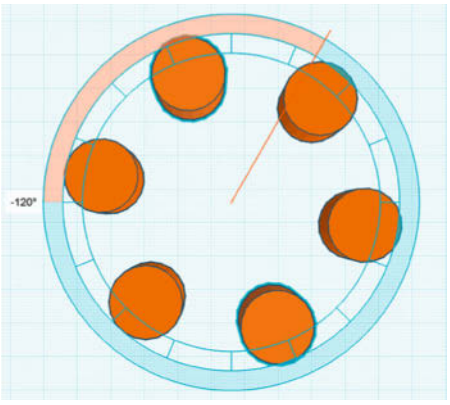


Bild 5: … und abschließend werden die beiden ersten Löcher ein zweites Mal dupliziert und um 120 Grad gedreht, damit wir sechs Löcher bekommen.

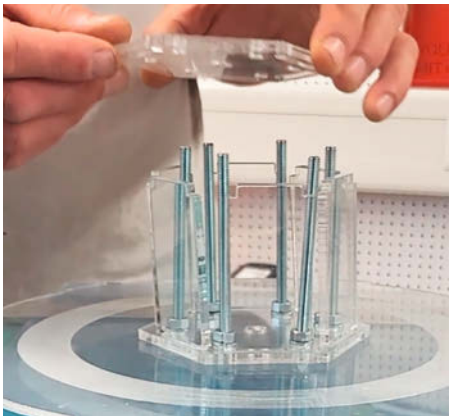


Bild 6: So wird der Kern zusammengebaut.

Bild 7: Aus diesem Rohr entsteht die Kammer, in die später die Eier gelegt werden.

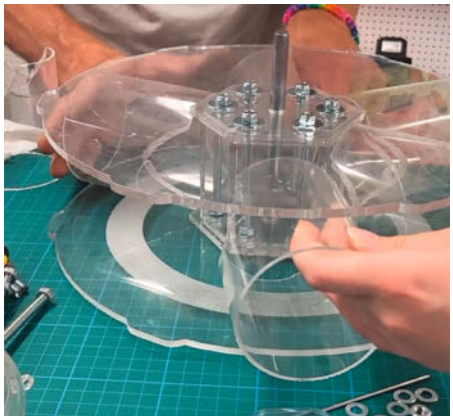


Bild 8: In diesen Rohren stecken später die Eier.

von innen nach außen montiert werden, damit sie später nicht die Drehung des inneren Kerns behindern.

Nachdem die Achse in eine der großen Acrylscheiben gesteckt und auf einer Seite mit dem Gewindestift des Flanschlagers fixiert wurde, kann auch der Innenkern eingehängt werden. Damit der Kern genau in der Mitte bleibt, müssen auf beiden Seiten zwischen Kern und Außengehäuse Nylonscheiben auf die Achse gelegt werden. Auch die beiden Abdeckungen aus 1,2-mm-Klarsichtfolie werden nun in die schmalen Spuren am Rand eingeschoben. Dabei müssen oben und unten Öffnungen frei bleiben, damit man die Eier oben einlegen kann und sie unten herausfallen können.

Zuletzt setzt man die zweite große Scheibe auf (Bild 10). Hier ist es sehr hilfreich, mindestens zwei weitere Hände zu haben, um die Gewindestangen, die Achse und die dünnen Platten aus Klarsichtfolie alle richtig in die vorgesehenen Löcher einzufädeln. Bevor alles endgültig verschraubt wird, sollte man prüfen, ob sich der Innenkern frei drehen kann. Gegebenenfalls müssen die Gewindestangen etwas gekürzt oder mit einem Dremel abgeschliffen werden, da im Inneren des Eierrevolvers nicht viel Platz ist.

Das Licht

Zusammen mit unserem Autor Stefan Recksiegel habe ich auf der Maker Faire Hannover zwei WS2812-LED-Streifen in den Eierrevolver eingebaut (Bild 11). Das war eine sehr fummelige Arbeit und erforderte auch, ihn noch einmal zu öffnen, um die LED-Streifen auf die Innenseiten der 1,2-mm-Abdeckungen zu kleben. Die Beleuchtung wird von einem ESP D1 Mini gesteuert. Die Pinbelegung ist in dem Sketch angegeben. Dieser erzeugt ein angenehmes, langsam pulsierendes Regenbogenmuster. Nur wenn der Knopf gedrückt wird, um ein Ei zu legen, wird kurz ein anderes blinkendes Farbmuster gezeigt.

Der Antrieb

Als Letztes benötigen wir noch einen Motor und eine einfache Schaltung für den Revolver. Da ich keinen passenden Motor zur Hand hatte, habe ich einfach ein billiges 5-V-Servo geöffnet und die Steuerplatine abgelötet, sodass nur noch Motor und Getriebe übrig blieben. Dann ließ ich einen Lego-Gummireifen von meinen Kindern mitgehen und befestigte ihn mit einem Servoarm an der Motorachse. Den Motor habe ich auf ein mühsam ausgesägtes Stück Holz montiert, wie in Bild 12 zu sehen ist. Ein 3D-gedrucktes Teil aus transparentem Filament wäre schöner gewesen.

Auf dem Bild sind auch zwei Schrauben auf jeder Seite des Gummireifens zu sehen. Mit den Muttern an diesen beiden Schrauben lässt sich der Druck des Reifens auf den inneren drehbaren Kern einstellen. Ist der Druck zu hoch, wird der Motor überlastet und der innere Kern dreht sich nur noch schwer. Bei zu wenig

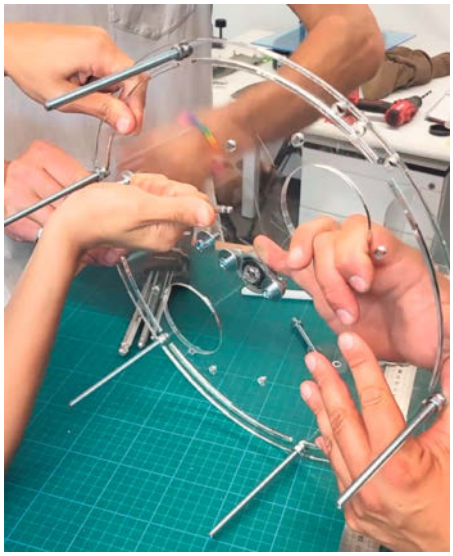


Bild 9: Für das äußere Gehäuse benötigt man die sechs längeren Gewindestangen.

Bild 10: Mit vier Händen lässt sich das Gehäuse leichter zusammenbauen.

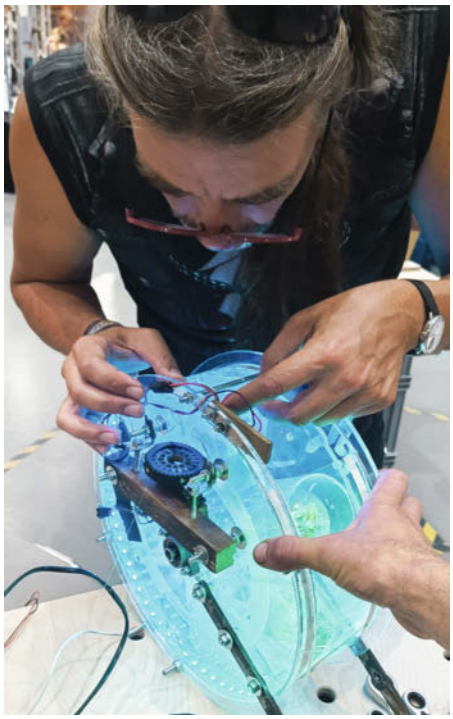


Bild 11: Wir haben die selbstklebenden WS2812-LEDs an den Innenseiten der Abdeckungen angebracht.

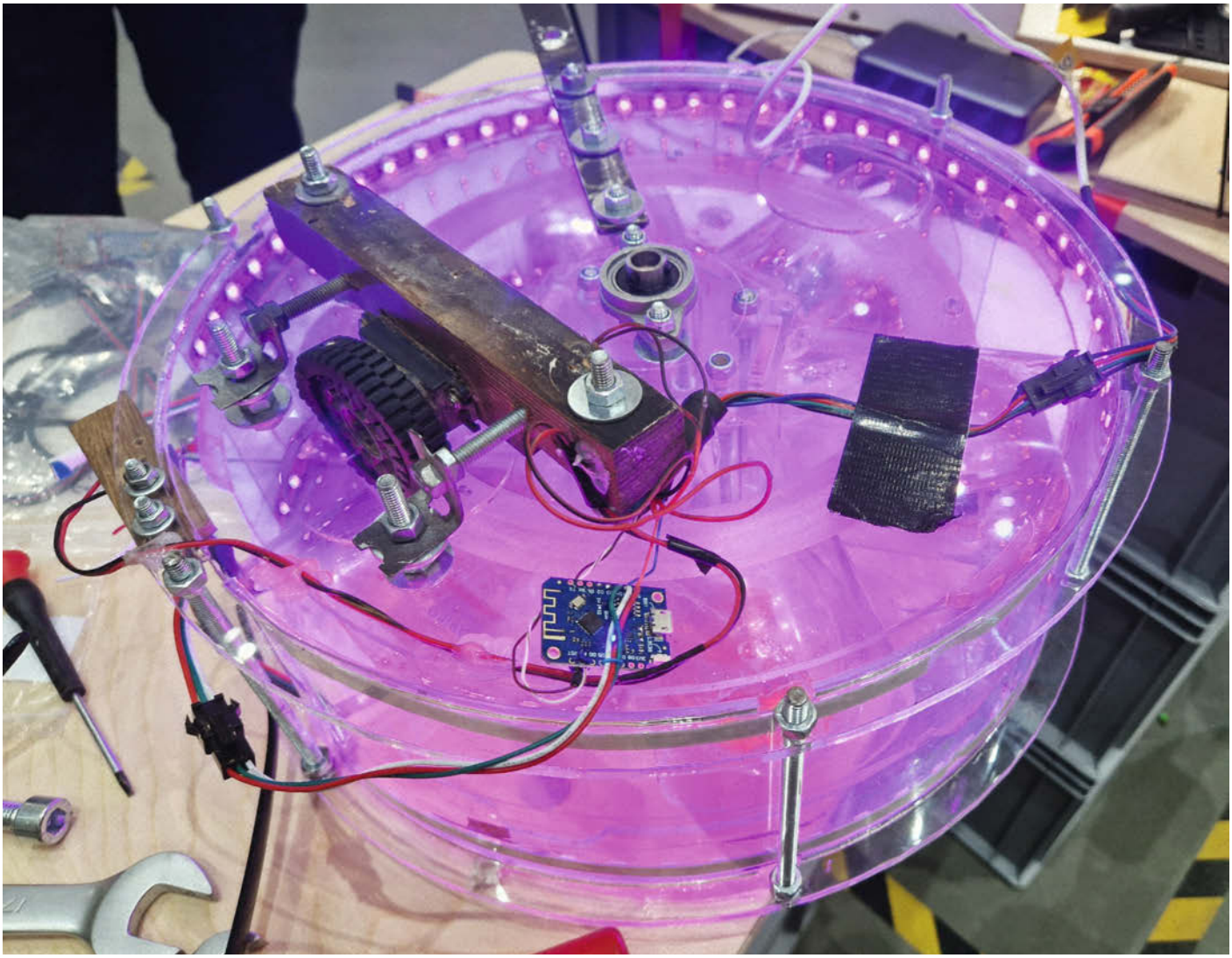


Bild 12: Nicht so schön, aber praktisch: Der Holzklotz dient dazu, den Motor zu halten. Mit den Muttern an den beiden Schrauben auf der jeweiligen Seite des Gummirades wird der Druck auf das innere bewegliche Teil eingestellt.

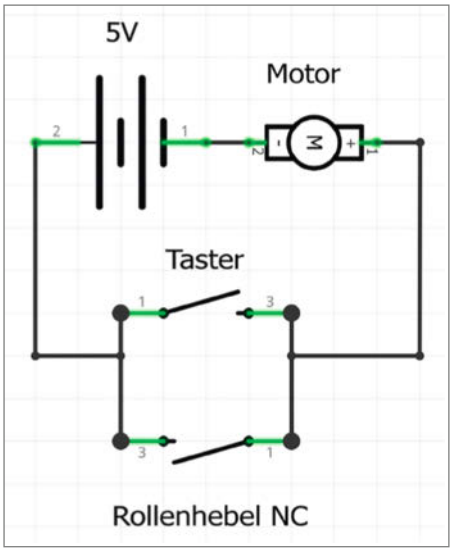


Bild 13: Entweder per Knopfdruck oder durch Absenken des Rollenhebels wird der Motor mit Strom versorgt und dreht sich weiter.

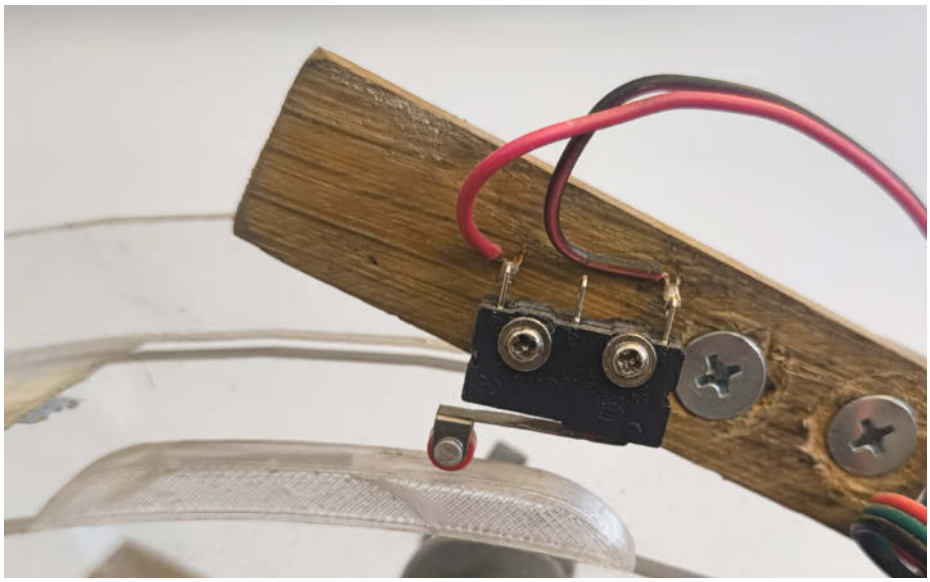


Bild 14: Nachdem ein Ei gelegt worden ist, wird der Rollenhebel durch den Nocken nach oben gedrückt und der Motor bleibt stehen.

Druck rutscht der Gummireifen durch und der innere Kern dreht sich überhaupt nicht. Wenn der Druck richtig eingestellt ist, funktioniert die Konstruktion erstaunlich gut.

Nicht alle Eier auf einmal legen

Die Idee ist, dass ein Ei gelegt wird, sobald man einmal kurz auf einen Knopf gedrückt hat. Dass sich der innere Kern nach dem Eierlegen nicht weiterdreht, regelt ein kleiner Schaltmechanismus, der wie eine Art Treppenhausschaltung funktioniert (Bild 13).

Das Ganze funktioniert so: Am äußeren Gehäuse ist ein kleiner Rollenhebel so angebracht, dass er direkt über einer der Acrylplatten des inneren Kerns sitzt (Bild 14). Am Rand dieser Acrylplatte befinden sich sechs Nocken. Einer davon ist in Bild 14 direkt unter dem Rollenhebel zu sehen. Wenn der Rollenhebel eingedrückt ist, erhält der Motor keinen Strom und bleibt stehen.

Wenn man nun aber den Taster betätigt, bekommt der Motor Strom – und zwar so lange, wie er gedrückt gehalten wird, und der innere Zylinder beginnt sich zu drehen. Dann bewegt sich der Nocken weiter und der Hebel geht runter. Dadurch bekommt der Motor auch darüber Strom und kann sich weiterdrehen, ohne dass man den Taster gedrückt halten muss. Wenn der nächste Nocken kommt und den Hebel nach oben drückt, wird der Strom wieder abgeschaltet und der Eierrevolver bleibt stehen. Es ist wichtig, den kleinen Hebel so zu positionieren, dass er in dem Moment gedrückt wird, in dem ein Ei gerade nach unten gefallen ist.

Kleine Justierungen

Manchmal hat der innere Zylinder so viel Schwung, dass er sich kurz weiterdreht, auch wenn der Motor nicht mehr läuft. Dies kann dazu führen, dass der Rollenhebel nicht mehr eingedrückt wird und der Motor wieder anläuft. Mit anderen Worten: Die Nocken sind zu kurz, um die Stromzufuhr zum Motor lange genug zu unterbrechen, damit alles zum Stillstand kommt.

Dafür gibt es zwei Lösungen: Entweder kann der Motor mit einem Spannungsregler so eingestellt werden, dass er sich langsamer dreht, oder die Nocken werden verlängert, sodass der Rollenhebel länger eingedrückt wird. Längere Nocken haben wir mit einem schnell hergestellten 3D-Druckteil erreicht, das hinter die vorhandenen Nocken geklebt wurde. Die STL-Datei dazu ist ebenfalls unter dem Link in der Kurzinfo zu finden.

Fazit

Die Konstruktion ist überraschend stabil und zuverlässig. Die in Bild 15 zu sehenden Federn

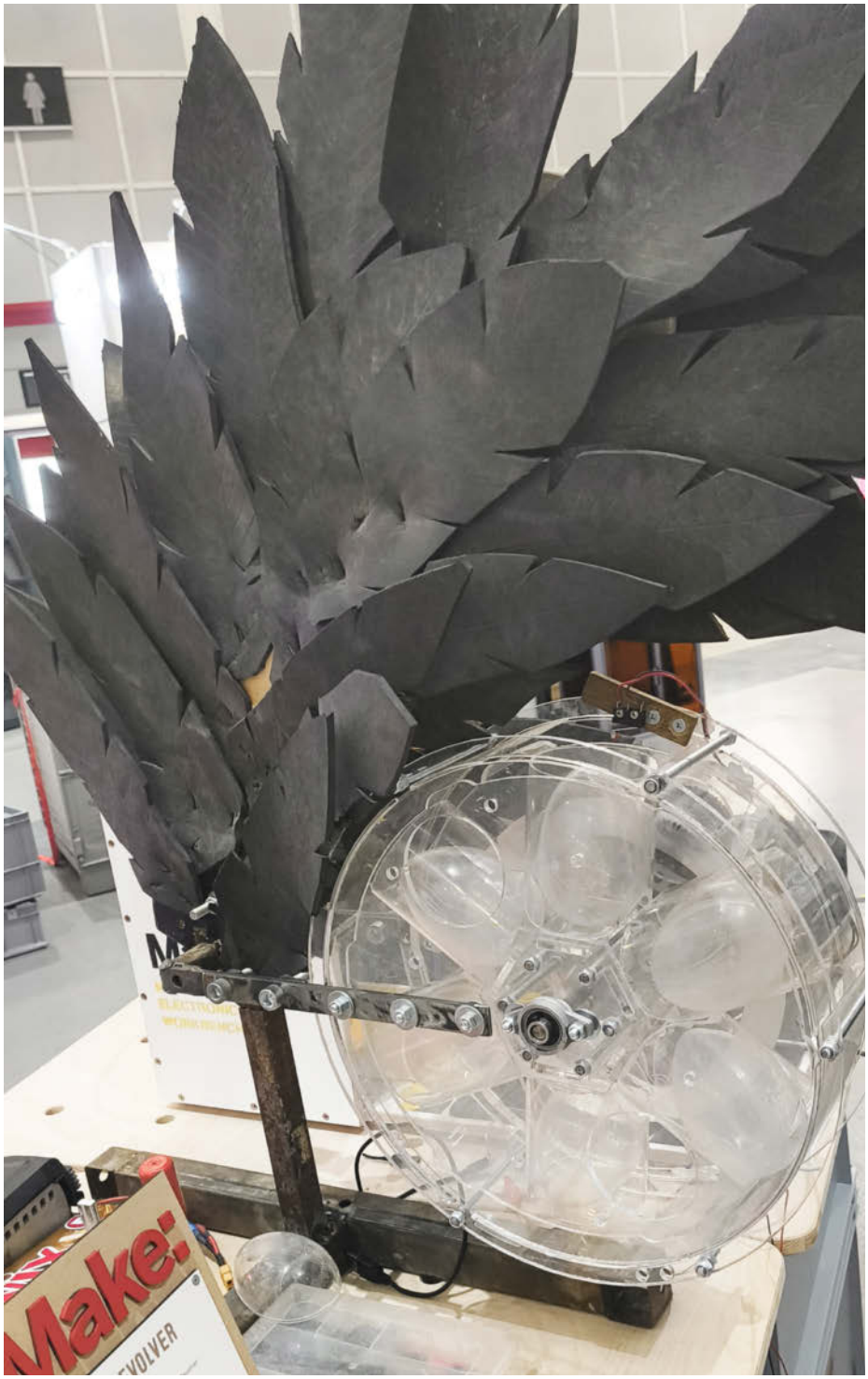


Bild 15: Damit der Eierrevolver nicht umfällt, muss der Standfuß unter dem Zylinder hervorstehen.

habe ich aus Moosgummi hergestellt und einfach auf ein gebogenes Holzstück getackert. Der Ständer, ebenfalls auf Bild 15 zu erkennen, kann individuell gestaltet werden. Zu beachten ist, dass der Eierrevolver recht schwer ist. Der Standfuß sollte dem Gewicht entsprechend ausgelegt sein. Ich habe meinen Fuß aus Metall geschweißt, aber wer sich das ersparen möchte, kann ihn auch aus Holz fertigen.

Damit der Eierrevolver befüllt werden kann, ohne dass man die untere Öffnung mit einer Hand abdecken muss, lohnt es sich, einen ca. 40 cm langen und 10 cm breiten Streifen der Klarsichtfolie zwischen die beiden fest verbauten Abdeckfolien und die Gewindestangen im unteren Bereich zu schieben. Dann fallen die Eier nicht mehr unten heraus, sondern werden im Karussell weiterbefördert. Viel Spaß beim Nachbauen! *—mch*

So gelingt Making in der Schule

Probleme lösen, Ideen entwickeln und Prototypen bauen: An der Wilhelm-August-Lay-Schule im baden-württembergischen Bötzingen sind Maker-Skills fester Bestandteil des Informatikunterrichts. Seit sechs Jahren beteiligt sich die Schule am Projekt „Make Your School", dessen Höhepunkt die dreitägigen Hackdays sind. Lehrer Matthias Keldermann berichtet, wie die Maker-Kultur die Schule positiv verändert hat.

von Matthias Keldermann

Vor sechs Jahren hat unsere Schule eine Chance genutzt, die viel bewegt hat. Wir durften eine der ersten Hackdays des Projekts „Make Your School – Eure Ideenwerkstatt“ durchführen. An diesen Projekttagen tüfteln Schüler an kreativen Lösungen für Herausforderungen, die sie im Schulalltag erleben.

Die Begeisterung und das Engagement, das wir bei den ersten Hackdays erlebt haben, waren überwältigend. Aus einer einmaligen Gelegenheit wurde schnell ein fester Bestandteil unseres Schuljahreskalenders. Aber wir wollten mehr: Warum das kreative Potenzial der Maker-Bewegung nur auf Projekttage beschränken? Deshalb haben wir die Prinzipien der Maker Education in unseren Informatikunterricht integriert. Dort stehen den Schülern ganzjährig moderne Werkzeuge wie 3D-Drucker, Lasercutter, Stickmaschinen sowie eine Reihe von Mikrocontrollern und Zubehör zur Verfügung. Im Informatikunterricht können die Teilnehmer der Hackdays dann ihre Prototypen weiterentwickeln. So verbindet sich die Dynamik der Projekttage mit der Kontinuität des Unterrichts – aus einmaligen Ideen werden nachhaltige Lernprozesse.

Eine besondere Motivation für die Schüler ist die Möglichkeit, sich mit ihren Prototypen für das Maker Festival in Berlin zu qualifizieren, das wie die Hackdays von der gemeinnützigen Organisation „Wissenschaft im Dialog“ organisiert wird. Bereits dreimal wurden unsere Teams nach Berlin eingeladen, um ihre Hacks vor Publikum zu präsentieren und von einer Jury bewerten zu lassen.

Kurzinfo

» Wie eine Schule Making in den Schulalltag integriert hat
» Ablauf der dreitägigen Hackdays erklärt
» Wie Mentoren die Teamarbeit unterstützen

Mehr zum Thema

» Elke Schick, So realisieren Sie Maker Education in der Schule, Make 3/19, S. 64
» Ákos Fodor, Von Arduino zur PlatformIO IDE, Make 1/25, S. 8
» Marco Düvelmeyer und Ákos Fodor, Roboter fürs Klassenzimmer, Make 7/23, S. 66

Alles zum Artikel im Web unter
make-magazin.de/xb7a

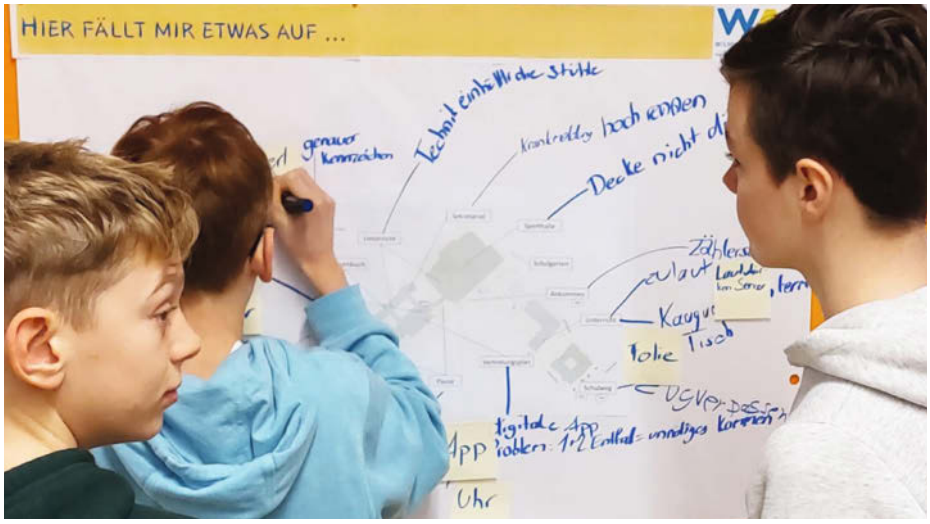


Bild 1: Kreativphase und Ideenfindung am Inspirationsplakat. Die Schüler notieren am Grundriss der Schule, wo es Probleme gibt.

Von der Idee zum Prototyp

Es ist Donnerstagmorgen. Vor den Technikräumen herrscht reges Treiben: Acht Schülerinnen und 29 Schüler der 8. Klasse stehen bereit, um drei intensive Tage lang in die Welt des Makings einzutauchen. Endlich sind sie bei den Hackdays dabei und dürfen ihre Ideen umsetzen – mittels Lasercuttern, 3D-Druckern und Elektronik.

Zunächst erhalten alle eine kurze Einführung in die Konzepte Making und Hacken. Dabei wird betont, dass es nicht darum geht, ein fertiges Produkt zu entwickeln, sondern einen Prototyp, der als Lösung für ein bestimmtes Problem im Schulalltag dienen kann.

Teamarbeit mit Mentor

Ein besonderes Merkmal der Hackdays ist die Unterstützung durch Mentoren von Hochschulen, die den Schülern zur Seite stehen. Leider konnte unsere Schule diese externe Unterstützung nicht mehr in Anspruch nehmen. Deshalb haben wir ein neues Mentorenmodell entwickelt, für das wir vier Auszubildende aus einem benachbarten Unternehmen gewinnen konnten. Sie brachten wertvolle Impulse aus der Berufspraxis ein.

Zusätzlich unterstützten uns Schülerinnen und Schüler, die bereits an den Hackdays teilgenommen hatten und ihre Kenntnisse im Wahlfach Informatik vertiefen konnten. So wurde jede Gruppe während der Hackdays von einer festen Bezugsperson betreut, was die Projekte noch mehr zu ihren eigenen machte und die Zusammenarbeit spürbar persönlicher gestaltete.

Mentoren unterstützen die Schüler.

Bild 2: Präsentation der Prototypen am dritten Tag in der Schulaula. Die Besucher wählen ihren Lieblingshack für den Publikumspreis.

Probleme im Schulalltag identifizieren

Deshalb steht als Nächstes die Ideenfindung auf dem Programm. In Kleingruppen suchen die Schüler nach Problemen und Hindernissen im Schulalltag und notieren diese auf einem Plakat mit den Umrissen ihrer Schule (Bild 1).

Danach versammeln sich alle wieder und den Schülern wird kurz gezeigt, welche Materialien zur Verfügung stehen: Sensoren und Aktuatoren der Seeed-Grove-Serie und der Arduino-Plattform.

Als Hilfestellung nutzen die Schüler die Website makeyourschool.de und die Unterstützung der Mentoren (siehe Kasten „Teamarbeit mit Mentor"). Planungshilfen und Zwischenpräsentation stellen sicher, dass am dritten Tag der Prototyp auf einer kleinen Ausstellung präsentiert werden kann.

Lösungsvorschläge ausdenken

Den Schülern wird nun eine halbe Stunde Zeit gegeben, um auf Klebezetteln Lösungsvorschläge für die beschriebenen Probleme zu schreiben. Anschließend findet eine Kurzbesprechung im Plenum statt, in der das weitere Vorgehen diskutiert wird. Die Teilnehmer sollen die Klebezettel nun sortieren und ähnliche Lösungsideen zusammenfassen. Das kann später dabei helfen, einen Hack auszuwählen.

Anschließend erhält jeder Schüler drei Klebepunkte und soll diese auf seinen Lieblingshack kleben. Die Ideen mit den meisten Punkten werden ausgewählt und nach einer kurzen Pause ordnen sich die Schüler den verschiedenen Ideen zu. Es werden Gruppen von zwei bis vier Schülern gebildet, die sich in den Technikräumen verteilen und dort ihre Arbeitsplätze einrichten. Jeder Gruppe stehen ein Laptop und ein Arduino mit dem Grove-Starter-Kit zur Verfügung.

Die Schüler beginnen nun gemeinsam zu überlegen, wie sie ihre Idee in einen Prototyp umsetzen können. Die ersten Sensoren und Aktoren werden mithilfe der Mentoren in Betrieb genommen und getestet. Gehäuse und Zubehör werden in Tinkercad gezeichnet und anschließend mit Lasercutter und 3D-Drucker hergestellt, erste Teilprozesse erprobt und optimiert.

Am Ende des ersten Tages präsentiert jede Gruppe den aktuellen Stand ihres Prototyps und berichtet über Schwierigkeiten und geplante Arbeitsschritte.

Tag zwei und drei

Der zweite Tag steht ganz im Zeichen des Makings in den Teams. Die Arbeitsabläufe sind bereits verinnerlicht und es herrscht eine wuselige Arbeitsatmosphäre. Für Abwechslung sorgen zwei Impulsvorträge zum Thema Digitalität. Lokale Unternehmen und Betriebe haben hier die Möglichkeit, sich vorzustellen und zu erklären, in welcher Form Digitalität in ihrem Arbeitsumfeld umgesetzt wird. Wir wollen, dass die Schülerinnen und Schüler erkennen, dass das, was sie bei den Hackdays machen, schon jetzt von großer Bedeutung ist.

Der dritte Tag ist ein Samstag, damit auch die Eltern teilnehmen können. Jetzt müssen die Teams einen kleinen Präsentationstisch und ein Plakat gestalten und ihren Hack in der Aula in einer Marktplatzsituation vorstellen (Bild 2). Alle Gäste haben nun die Möglichkeit, sich bei den Gruppen über ihre Arbeit zu informieren und sich für einen Lieblingshack zu entscheiden, der dann mit dem Publikumspreis ausgezeichnet wird.

Making in der Ausbildung

An unserer Schule legen wir großen Wert darauf, die Impulse der Hackdays nachhaltig in das Schulleben zu integrieren. Ich erlebe die Schülerinnen und Schüler in diesem Format als sehr selbstwirksam und freue mich, wenn sie sich kreativ mit selbst gewählten Themen auseinandersetzen. Sie erhalten Einblicke in die textbasierte Programmierung von Mikrocontrollern und setzen mithilfe von Sensoren und Aktoren ihre Lösungsvorschläge in greifbare Prototypen um. Die gemeinsame Arbeit in kleinen Gruppen fördert nicht nur die Teamfähigkeit, sondern auch das Vermögen, kleine und große Herausforderungen zu meistern. Die Vielfalt der Projekte sorgt dafür, dass die Teams oft eigenständig Problemlösungen entwickeln müssen.

Ein wichtiger Aspekt des Formats ist, dass die Projekte nicht mit Schulnoten bewertet werden. Stattdessen liegt der Schwerpunkt auf der Präsentation der Ergebnisse am dritten Tag vor einem Gastpublikum, das wertschätzendes Feedback gibt.

Besser lernen mit der Pomodoro-Lampe

Während die Hackdays 2023 haben sich die drei Schüler Julian Wirth, Karim Maamoun-Peulier und Lars Heitzler entschieden, eine Tischlampe zu entwickeln, die das Lernen mit der sogenannten Pomodoro-Studientechnik vereinfachen soll. Bei dieser Lernmethode, die nach einem Küchentimer in Tomatenform benannt ist, wird das Lernen in Arbeits- und Pausenphasen unterteilt. Nach 20 Minuten konzentrierter Arbeit folgt eine fünfminütige Pause. Die Tischlampe soll in der Arbeitsphase weißes Licht zeigen und in der Pause auf buntes Discolicht umschalten.

Die erste Version wurde mit einem weißen, kegelförmigen Lampenschirm ausgestattet,

den die Schüler in Tinkercad entworfen und auf einem 3D-Drucker hergestellt haben (Bild 3). Der Lampenarm mit drei Gelenken wurde zunächst aus Pappe mit einem Lasercutter ausgeschnitten und anschließend verschraubt. Für die Beleuchtung sorgt ein NeoPixel-Ring. Im Fuß der Lampe befindet sich ein Arduino UNO mit Seeed-Grove-Baseshield, der die Steuerung übernimmt. Bedient wird die Lampe über einen Grove-Button.

Die Programmierung der Lampe basiert auf einfachen Beispielsketches aus der Arduino IDE. Wird der Button gedrückt, leuchtet die Lampe zunächst für eine bestimmte Zeit weiß, was die Lernphase signalisiert. Nach Ablauf dieser Zeit wird ein Regenbogen-Farbeffekt mithilfe der Neopixel-Bibliothek gestartet. Gerade das Programmieren bereitete den Schülern Schwierigkeiten, so Julian Wirth: „Das war das Schwierigste an dem Projekt, weil es sehr viele Fehler gab, die wir beheben mussten."

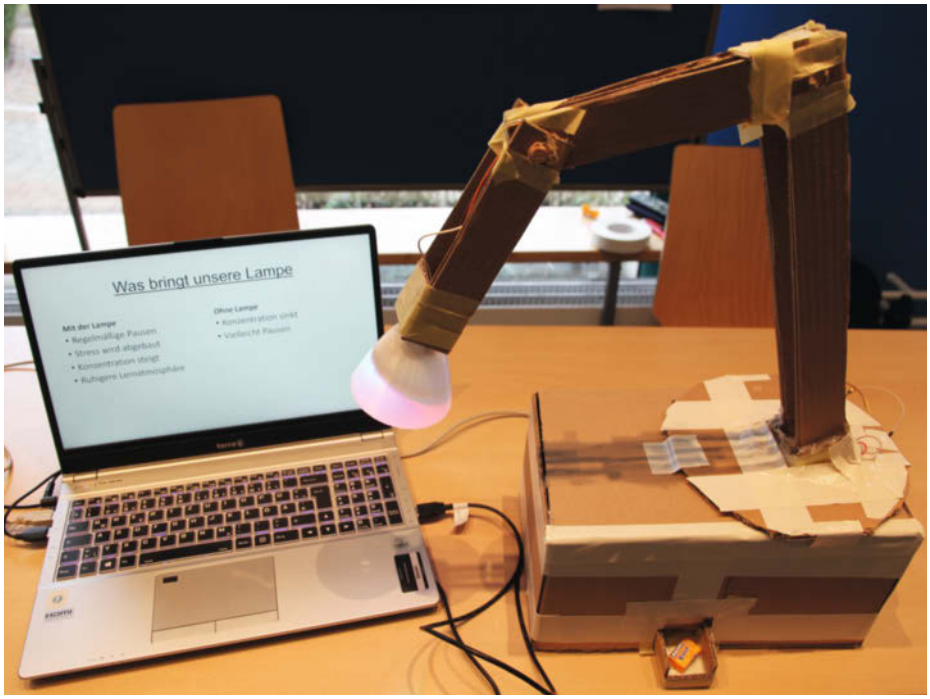


Bild 3: Die erste Version der Pomodoro-Lampe mit viel Klebeband und Heißkleber.

Von der Werkbank nach Berlin

Während der Arbeit an der Pomodoro-Lampe durchliefen die Schüler den gesamten Prozess des Prototypenbaus und entwickelten drei verschiedene Versionen mit mehreren Zwischenschritten. Die erste Version, die während der Hackdays entstand, war aus Pappe. Die zweite Version war wesentlich stabiler: Lampenschirm sowie Gehäuse wurden aus Kraftplex gefertigt, die Gelenkarme zusätzlich verschraubt.

Mit dieser optimierten Version bewarben sich die Schüler beim Maker Festival in Berlin. Nach der erfreulichen Zusage, dass ihr Team teilnehmen darf, arbeiteten sie eigenständig an weiteren Verbesserungen. Sie entwickelten und testeten eine Mechanik, die in der Pausenphase ein Kaubonbon als Belohnung ausgibt.

Kurz vor ihrer Abreise passten sie das Gehäuse der Lampe noch einmal an, um die neuen Funktionen zu integrieren. In Berlin präsentierten sie schließlich eine voll funktionsfähige Pomodoro-Lampe (Bild 4). Während der Veranstaltung interviewte eine Jury das Team und ließ sich den gesamten Entwicklungsprozess erklären. Am Ende des Tages wurden die Schüler mit dem Designpreis ausgezeichnet.

„Das war sehr überraschend, denn wir hatten nicht damit gerechnet, etwas zu gewinnen, da die anderen Teams auch sehr tolle Sachen vorgestellt haben", sagt Lars Heitzler.

Mit dem Preisgeld haben die drei nun die Aufgabe bekommen, ihren Hack noch zu optimieren. Ihr Ziel ist es, die Lampe so weit zu verbessern, dass sie benutzerfreundlicher und robuster wird, damit sie sich ideal für den Einsatz im Unterricht eignet. *—mch*

Bild 4: Lars Heitzler, Julian Wirth, Karim Maamoun-Peulier und Matthias Keldermann auf dem Maker Festival in Berlin.

Git für Maker, Teil 2

In der Make-Ausgabe 1/25 haben wir erklärt, was genau Git ist, wie man es einrichtet und wie man die verschiedenen Funktionen des Programms für die lokale Versionsverwaltung nutzt. In dem zweiten Teil wird auf diesem Wissen aufgebaut, um weitere Funktionen von Git nutzen zu können: Projekte nicht nur lokal zu verwalten, sondern sie online auf der Plattform GitHub zu teilen und dort zu verwalten.

von Daniel Schwabe

Die Vorgänge werden beispielhaft an GitHub gezeigt, weil es sich dabei um die größte Git-Plattform handelt. Ähnliche Services funktionieren grundlegend gleich. Befehle und Konzepte lassen sich auch auf andere Anbieter anwenden.

Was kann GitHub denn?

Lädt man ein mit Git verwaltetes Projekt auf GitHub hoch, überträgt sich auch der gesamte Projektverlauf, wie er lokal abgespeichert wurde. Aber nicht nur der aktuelle Stand des Projekts ist online verfügbar, sondern auch alle früheren Versionen, die durch Commits versioniert wurden. Dadurch können andere Nutzer sämtliche Änderungen nachvollziehen und gezielt auf frühere Versionen zurückgreifen. Auch Branches werden online abgebildet.

Neben der reinen Bereitstellung bietet GitHub auch die Möglichkeit zur Zusammenarbeit. Andere Entwickler können nicht nur den Code einsehen, sondern auch aktiv mitwirken. Sie können z. B. eigene Commits vorlegen, die der Besitzer des Repositorys überprüft und entweder akzeptiert oder ablehnt. So bleibt die Kontrolle über das Projekt in der Hand des Besitzers, während gleichzeitig mehrere Personen gemeinsam daran arbeiten können.

Bei GitHub registrieren

Um auf GitHub arbeiten zu können, braucht man als Erstes einen Account auf der Plattform. Dafür klickt man auf github.com oben rechts auf die Schaltfläche „Sign up". Dort trägt man dann seine E-Mail-Adresse und ein Passwort ein. Was in dem Feld „Username" eingetippt wird, verwendet GitHub später als öffentlichen Anzeigenamen, der von anderen Usern einsehbar ist. Ob man dort ein Alias oder den Klarnamen verwenden möchte, muss man selbst entscheiden.

Um alle Funktionen (Beitragen von Code, Ändern der Kontoeinstellungen etc.) von GitHub nutzen zu können, muss man für sein Konto die sogenannte Zwei-Faktor-Authentifizierung aktivieren. Was das ist und wie man sie nutzt, steht im Kasten „Zwei-Faktor-Authentifizierung".

Ein Onlinerepository anlegen

Im ersten Teil dieser Artikelreihe haben wir ein lokales Repository auf dem Computer ange-

Kurzinfo

- Onlinerepository bei GitHub anlegen
- Commits hoch- und herunterladen
- In öffentlichen Repositorys mitarbeiten

Mehr zum Thema

- Daniel Schwabe, Git für Maker, Make 1/25, S. 54
- Florian Schäffer, AVR-Programme debuggen, Teil 1, Make 4/24, S. 104

Alles zum Artikel im Web unter
make-magazin.de/x96w

Zwei-Faktor-Authentifizierung

Unter einer Zwei-Faktor-Authentifizierung (meistens mit 2FA abgekürzt) versteht man, dass man sich bei der Anmeldung z. B. auf einer Website mit dem Passwort und einer zweiten Verifizierungsmethode, einem zweiten Faktor, anmelden muss.

Das kann ein immer wechselnder Code sein, den man bei jedem Anmeldeversuch per E-Mail oder SMS zugeschickt bekommt oder der von einer speziellen App generiert wird. Aber auch spezielle USB-Geräte können ein zweiter Faktor sein. Diese Geräte nennen sich „Passkeys". Nutzt man so ein Gerät, wird auf der Website, bei der man sich mit diesem Passkey anmelden möchte, eine Datei gespeichert, mit der das USB-Gerät zweifelsfrei identifiziert werden kann.

Um 2FA auf GitHub einzurichten, klickt man nach der Anmeldung auf der Startseite oben rechts auf das Profilbild des Accounts und dann auf „Settings". Im sich neu öffnenden Fenster befinden sich die entsprechenden Optionen unter „Password and authentication". Hier kann man das Passwort ändern, einen Passkey oder 2FA über eine App oder SMS einrichten.

Wer einen Passkey besitzt, kennt sich schon mit dessen Einrichtung aus. Deshalb wird hier exemplarisch die Möglichkeit über eine Authenticator-App genutzt. Dafür klickt man unten auf der Seite auf den großen grünen „Enable two-factor authentication"-Button. Im neuen Fenster wird prominent ein QR-Code angezeigt.

Dieser QR-Code beinhaltet das sogenannte Geheimnis für diesen Account. Dabei handelt es sich um eine einmalige Zeichenfolge, mit der dann in Abhängigkeit von der Uhrzeit alle 30 Sekunden neue Sicherheitscodes (die dann der zweite Faktor sind) erzeugt werden. Dieses Geheimnis darf man niemals weitergeben.

Um mit diesem QR-Code Passwörter für den zweiten Faktor zu generieren, braucht man eine spezielle App wie den Google Authenticator oder den Microsoft Authenticator. Wer schon für eine andere Website so eine App nutzt, kann dieses Geheimnis dort auch einspeichern. Hier ist einmal der Vorgang im Google Authenticator erklärt.

Nach dem Öffnen der App tippt man auf das bunte Plus-Zeichen unten rechts in der Ecke. Dort wählt man dann „QR-Code scannen" aus und richtet die Handykamera auf den QR-Code auf dem Bildschirm. Und das war es auch schon. Jetzt hat man einen Eintrag zu diesem GitHub-Account, der sich alle 30 Sekunden aktualisiert und einen Code generiert.

Um die 2FA auf der Website fertig einzurichten, muss man diesen unterhalb des QR-Codes eintragen und bestätigen. Hat man das getan, bekommt man 16 Sicherheitscodes angezeigt. Das sind feststehende Einmalpasswörter, die immer gelten und nicht von der Zeit abhängig sind. Sie sind hilfreich, wenn man keinen Zugriff mehr auf den zweiten Faktor hat (weil einem z. B. das Handy kaputtgegangen ist). Diese Codes muss man sicher irgendwo aufbewahren.

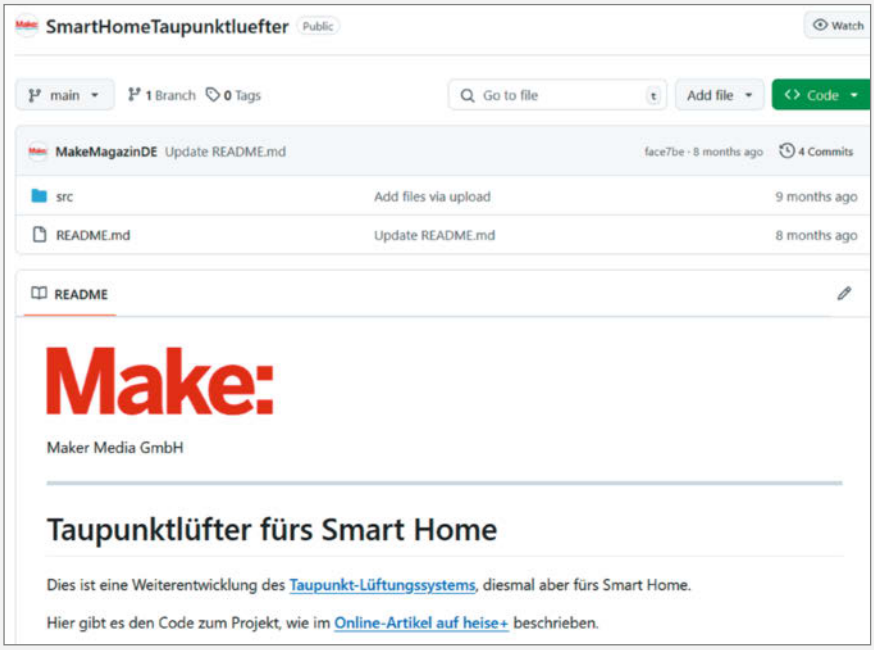
Wenn man sich jetzt bei GitHub anmeldet, gibt man zuerst die E-Mail und das Passwort ein und danach das aktuelle Einmalpasswort aus dem Authenticator.

Readme

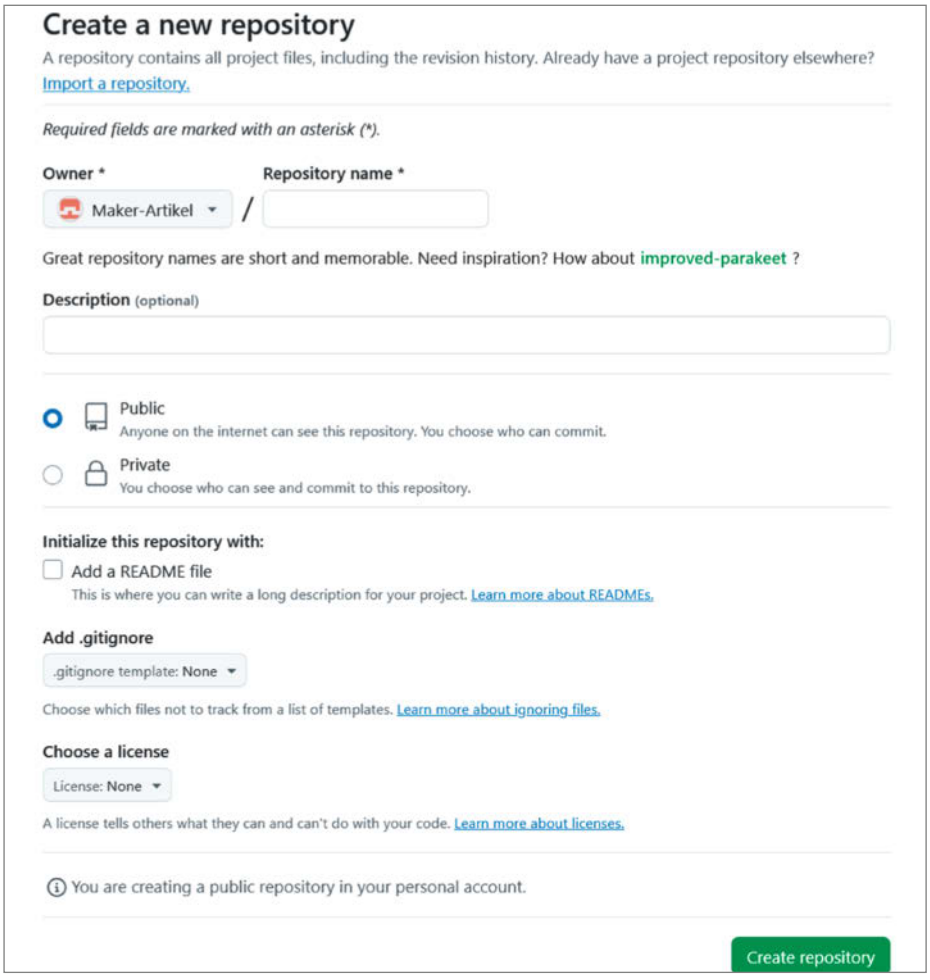
Aktiviert man bei der Erstellung eines Repositorys die Option „Add a README file“, wird das Repository nicht leer erzeugt, sondern mit einer bereits bestehenden Datei – der README.md.

Dabei handelt es sich um eine Textdatei im Markdown-Format, in der Informationen über das Projekt in Text- und Bildform oder Links bereitgestellt werden können. Der Inhalt der Readme wird auf der Hauptseite des Repositorys direkt angezeigt.

Damit der Inhalt dieser Datei automatisch angezeigt wird, muss sie im obersten Verzeichnis des Repositorys liegen. Wenn man keine README.md bei der Erstellung des Repositorys angelegt hat, kann man das nachträglich machen.



Eine Readme kann nicht nur Text, sondern auch Bilder und Links beinhalten.



Die Optionen beim Erstellen eines Repositorys sind übersichtlich. Vor allem Name und Sichtbarkeit („Public“/„Private“) sind wichtig.

legt. Um nicht nur lokal zu arbeiten, sondern den Inhalt des lokalen Repositorys online bereitzustellen, muss man auch auf GitHub ein Repository anlegen.

Dafür klickt man nach der Anmeldung oben rechts auf das große Plus-Symbol und dann auf „New Repository“.

In der überschaubaren Maske für die Erstellung trägt man als Erstes einen Repository-Namen ein. Dieser darf keine Leerzeichen beinhalten. Trägt man einen Namen mit Leerzeichen (oder Umlauten) ein, werden automatisch Bindestriche eingefügt. Als Beispiel nutzen wir hier mal das Projekt „Taupunktluefter“ aus Make 4/24.

Darunter ist ein Feld für eine Beschreibung des Projektes. Darin sollte stehen, worum es in dem Projekt geht, z. B.: „Taupunktlüftersystem, um Räume zu trocknen, basierend auf Wemos D1 Mini. Integration ins Smarthome über MQTT“.

Als Letztes legt man fest, ob das Repository „Public“ oder „Private“ sein soll. Ist ein Projekt „Public“, kann es von jedem im Internet gesehen und heruntergeladen werden. Beitragen können nur Leute, die man als Besitzer explizit festgelegt hat. Ist das Projekt „Private“, können nur zugelassene Accounts (einschließlich man selbst) das Projekt sehen und herunterladen. Diese Einstellung kann nachträglich geändert werden. Deshalb wird für dieses Beispiel-Repository erst mal „Private“ ausgewählt.

Die drei Optionen „Add a README file“, „Add .gitignore“ und „Choose license“ werden in den jeweiligen Kästen „Readme“, „Gitignore“

Gitignore

Die Datei .gitignore in einem Repository gibt an, welche Dateien und Ordner von Git ignoriert werden sollen. Das ist z. B. nützlich für Log-Dateien, temporäre Dateien oder Builds.

Sind eine explizite Datei, ein Ordner oder generell Dateien mit einer bestimmten Endung in der .gitignore eingetragen, werden sie einem auch nicht mehr für Commits vorgeschlagen oder mit dem „git status"-Befehl als verändert oder nicht verwaltet angezeigt (siehe Teil 1 dieser Artikelreihe). In der Praxis wird eine .gitignore-Datei verwendet, um besonders große Dateien zu ignorieren. Auch sicherheitsrelevante Dateien (z. B. Konfigurationsdateien, die Zugangsdaten für Server etc. beinhalten) werden über die .gitignore von der Git-Verwaltung ausgeschlossen, damit diese nicht einsehbar im Internet landen.

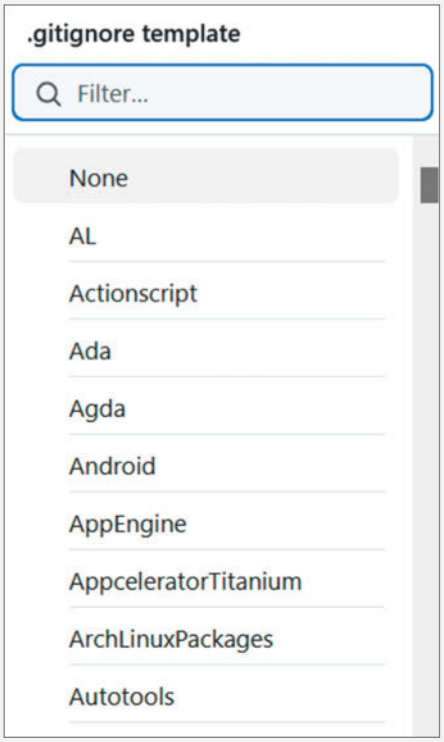
Eine .gitignore, die beispielsweise .temp-Dateien, Log-Dateien und Bilder ignoriert, würde so aussehen:

```
*.temp
Log-*.txt
*.jpg
*.png
```

Das * bedeutet „Hier kann alles stehen". *.temp stellt also ein, dass alle Dateien, die mit der Dateiendung temp gespeichert werden, ignoriert werden.

Bei der Erstellung eines Onlinerepositorys kann man über ein Drop-down-Menü aus einer Reihe vorgefertigter .gitignore-Dateien für verschiedene Programmiersprachen und Programmiertools auswählen.

Pro Branch eines Repositorys kann es nur eine .gitignore geben. Diese kann sich dann aber in unterschiedlichen Branches unterscheiden.



GitHub stellt verschiedene .gitignore-Templates bereit. Für jedes Projekt gibt es ein passendes.

und „Lizenz" erklärt. Mit einem Klick auf „Create repository" schließt man den Vorgang ab und wird auf die Seite des neuen, komplett leeren Repositorys geleitet.

Lokales Repository mit Onlinerepository verbinden

Mit den Informationen aus dem letzten Artikel liegt in diesem Szenario schon ein lokales Repository vor. Es gibt mit Git verwaltete Dateien und auch schon einige Commits – allerdings nur lokal. Um dieses Projekt jetzt 1:1 (wie auf dem lokalen Computer) mit GitHub zu verbinden, geht man wie folgt vor.

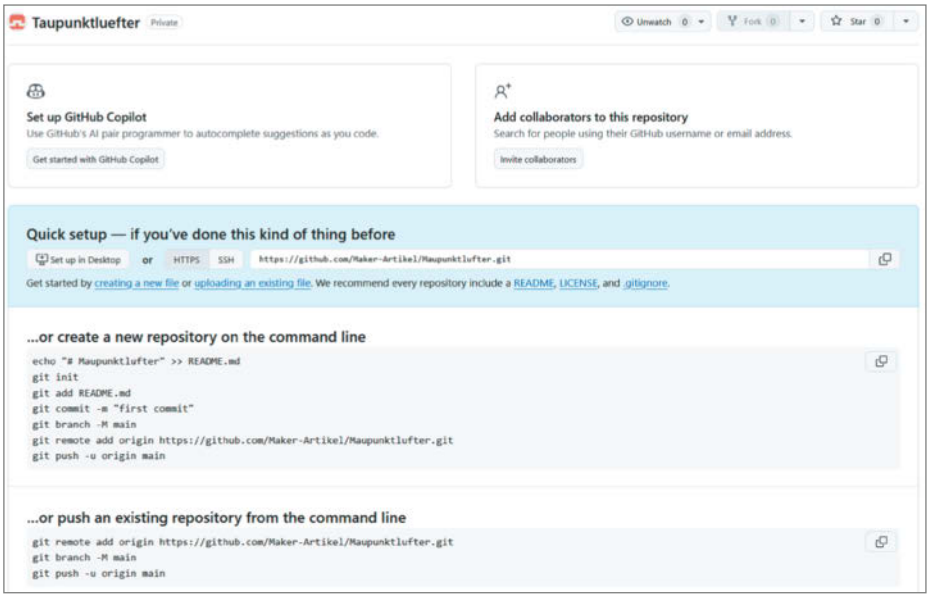
Nachdem man auf GitHub ein komplett leeres Repository angelegt hat – ohne Readme, Lizenz oder .gitignore –, landet man auf einer „Next Steps"-Seite.

Diese bietet drei Möglichkeiten, das Repository jetzt lokal auf den eigenen Computer zu synchronisieren; im vorliegenden Fall mit einem lokal existierenden Projekt ist die letzte Option „push an existing repository from the command line" die richtige, um weiterzumachen.

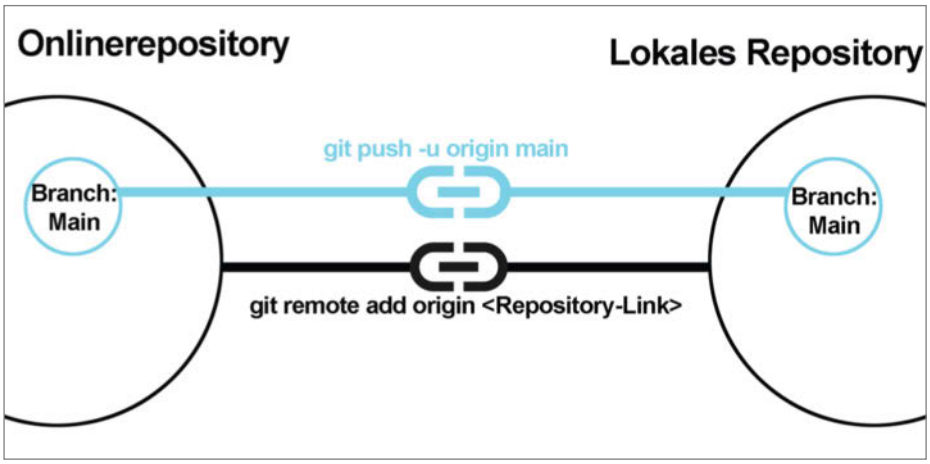
Dort sind drei git-Befehle aufgelistet, mit denen man die beiden Repositories verbindet. Diese Befehle müssen jetzt auf dem Computer in die Kommandozeile eingegeben werden, während man sich im Ordner befindet, in dem das lokale Repository gespeichert ist. Wie man das Terminal öffnet, ist dem ersten Teil dieser Artikelreihe zu entnehmen.

Der erste Befehl `git remote add origin <Repository-Link>` verbindet das lokale Repository mit dem online angelegten. Danach benennt `git branch -M main` den aktuellen Branch des lokalen Repositorys in „main" um. Das ist wichtig, weil viele Workflows auf GitHub davon ausgehen, dass der Haupt-Branch eines Repositorys „main" heißt. Selbst wenn man im Repository keinen neuen Branch erstellt hat, arbeitet man im main-Branch. Denn die Commits werden behandelt, als ob sie einem Branch zugeordnet sind. Eben dem main-Branch.

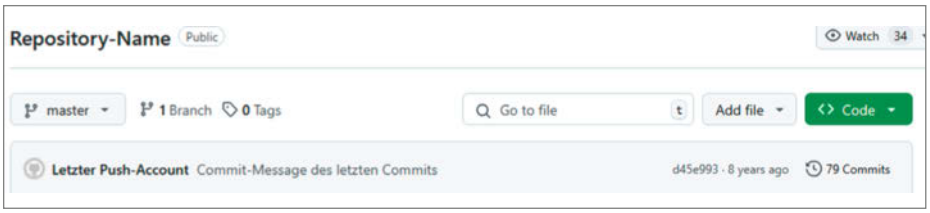
Als Letztes wird mit `git push -u origin main` festgelegt, dass der lokale main-Branch mit dem main-Branch von Origin verknüpft ist, den man mit dem ersten Befehl beim Einrichten des GitHub-Onlinerepositorys erstellt hat. Anschließend lädt man alles aus dem lo-



GitHub zeigt bei einem neuen Repository genau an, wie man damit jetzt weiterarbeiten kann.



Mit den Befehlen werden erst die Repositories und dann die main-Branches verbunden.



Ein Onlinerepository mit Namen, dem letzten Beitragenden und einer Anzahl an Commits.

Wenn man das erste Mal über einen Git-Befehl mit GitHub interagiert, muss man sich anmelden.

kalen Repository (Dateien und alle Commits aus der Vergangenheit) auf GitHub hoch.

Ist es das erste Mal, dass man mit GitHub auf diese Art und Weise interagiert, muss man sich erst anmelden, um den aktuellen PC als vertrauenswürdig auszuweisen und über die Kommandozeile Git-Interaktionen mit GitHub ausführen zu können.

Es öffnet sich ein „GitHub Sign in"-Fenster. Dort klickt man auf „Sign in with your browser" und meldet sich wie gewohnt auf GitHub an (über die Anmeldemethode mit 2FA). Danach bestätigt man, dass man den Computer authentifizieren will. Jetzt kann man mit dem Computer problemlos mit GitHub interagieren.

Danach reicht der kurze Befehl `git push` ohne weitere Attribute, um Commits hochzuladen.

Upload und Download

Wenn man ein lokales Repository hat, das mit einem online geführten Repository auf GitHub verbunden ist, kann man lokale Änderungen hochladen und Änderungen aus dem Onlinerepository herunterladen, z. B. wenn man mit jemandem zusammenarbeitet. Siehe dazu Kapitel „Zusammenarbeiten".

Um lokale Änderungen hochzuladen, erstellt man zuerst einen Commit. Danach lädt man diesen Commit mit all seinen Änderungen mit dem Befehl `git push` hoch. Die neuen Änderungen sind sofort online sichtbar.

Ist online etwas verändert worden, lädt der Befehl `git pull` die neuen Änderungen herunter.

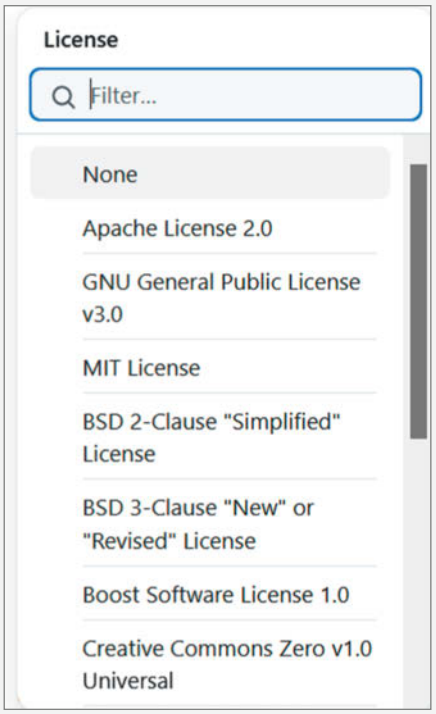
Diese Befehle funktionieren zwar ähnlich zu einfachem Hoch- und Herunterladen von Dateien, allerdings wird jeweils 1:1 der aktuelle Projektstand übertragen. Änderungen werden in die lokalen Dateien eingepflegt, neu erstellte Ordner werden ebenfalls angelegt. Es können Dateien verschoben oder gelöscht werden. Man synchronisiert mit diesen Befehlen Veränderungen im Projekt als Ganzes.

Lizenz

Wenn ein Repository öffentlich ist, kann jeder den darin enthaltenen Code herunterladen. Privat kann er dann damit machen, was er will. In der License-Datei wird allerdings festgelegt, was man damit in der Öffentlichkeit anstellen darf: ob der Code z. B. in anderen Projekten verwendet werden darf, ob dann der Name des Erstellers genannt werden muss (oder nicht) oder ob der Code gar nicht zur Weiterverwendung freigegeben ist.

Auch diese Datei kann man beim Erstellen des Onlinerepositorys aus einer Dropdown-Liste auswählen. Welche Lizenz was erlaubt, muss man individuell nachlesen.

Für Maker, die ihren Code der Community zur Verfügung stellen wollen, sind folgende Lizenzen das Mittel der Wahl: Die GNU General Public License, bei der Code zwar frei genutzt werden darf, das daraus resultierende Projekt allerdings auch unter GNU General Public License Open Source sein muss. Oder Apache 2.0 – auch hier darf Code frei verwendet werden, allerdings kann auf diesen verwendeten Code dann kein Patentanspruch von Dritten erhoben werden.



Mit einer Lizenz legt man den Umfang fest, in dem der eigene Code von Dritten in der Öffentlichkeit verwendet werden darf.

Probleme bei Pull

Beim Ausführen des Befehls `git pull`, um Änderungen in ein lokales Projekt zu übernehmen, kann es passieren, dass noch ungespeicherte lokale Änderungen existieren. Da diese durch den Pull überschrieben würden, entsteht ein sogenannter Merge-Konflikt.

Diesen kann man lösen, indem man entweder einen lokalen Commit erstellt, den man aber nicht mit `git push` hochlädt, sondern dann direkt mit `git pull` die Online-Änderungen herunterlädt, oder indem man sie zwischenspeichert.

Mit dem Befehl `git stash` werden die Änderungen vorübergehend zwischengespeichert. Danach kann man wieder mit `git pull` Änderungen aus dem Internet laden und dann mit `git stash pop` die zwischengespeicherten Änderungen wiederherstellen.

Die Option, einen lokalen Commit anzulegen, ist die eleganteste Lösung. Dabei bleiben Änderungen auf jeden Fall bestehen und man kann jederzeit darauf zugreifen, um sie wieder ins Projekt einzubauen oder anderweitig zu verwenden.

Zusammenarbeiten

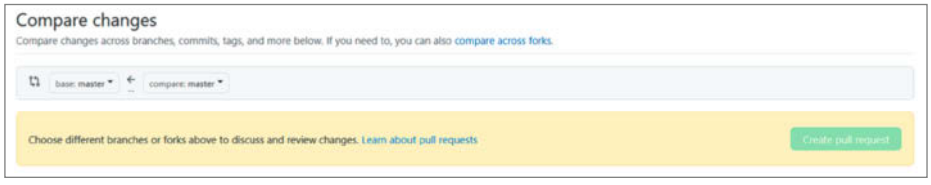
GitHub ist vor allem eine Plattform, die es Entwicklern ermöglicht, gemeinsam an Code zu arbeiten, Änderungen nachzuverfolgen und Feedback auszutauschen.

Bei der Zusammenarbeit auf GitHub gibt es zwei Szenarien:

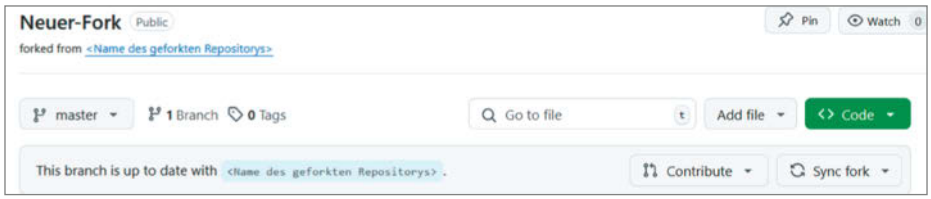
1. Man arbeitet als fest umrissenes Team an einem Projekt und alle können darauf zugreifen und Änderungen vornehmen.
2. Jemand Projektfremdes möchte eine Änderung in das Projekt einbringen und legt seine Anpassungen vor, damit diese geprüft und eventuell in das Projekt aufgenommen werden.

Im ersten Fall fügt man die Mitarbeiter explizit zu einem Repository hinzu, um ihnen bestimmte Rechte zu geben. Dafür öffnet man ein Repository auf GitHub, klickt über dem Repository-Namen ganz rechts auf „Settings“ und dort dann auf „Collaborators/Add

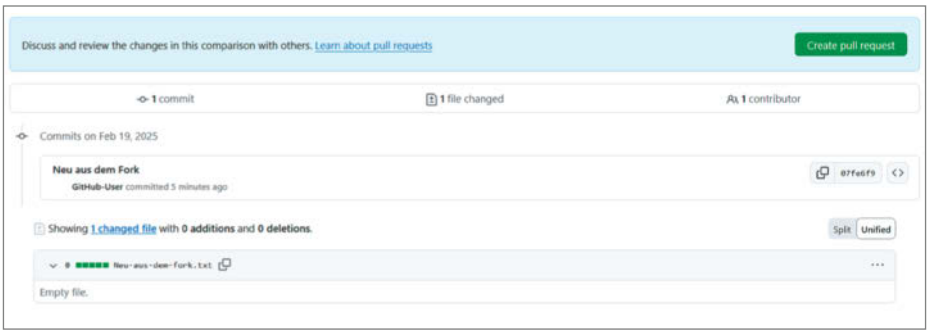
Ein Fork legt eine Kopie eines Repositorys an. In diese können Änderungen gepusht werden.



Bei einem Pull Request kann man Branches und Forks auswählen, aus denen neuer Code in ein Projekt einfließen soll.



Das Repository eines Forks hat zusätzlich die Vermerke, von welchem originalen Repository es kopiert wurde und wie viele Änderungen es gab.



Die Informationen über den Pull Request enthalten eine Liste der geänderten Dateien.

People“. Danach öffnet sich ein Eingabefeld, in dem man Mitarbeiter über ihren GitHub-Usernamen oder ihre E-Mail hinzufügen kann.

Hat man Mitarbeiter eingeladen, bekommen diese eine E-Mail, in der sie ihre Teilnahme am Projekt bestätigen. Jetzt können sie ihre Änderungen am Projekt ganz normal mit `git push` hochladen. Zugriffe auf Repository-Optionen (wie beispielsweise eine Namensänderung oder Sichtbarkeitseinstellungen) hat der Account dann nicht.

Im zweiten Szenario möchte jemand, der nicht berechtigt ist, in das Repository zu pushen, etwas zu dem Projekt beitragen. Deshalb muss diese Person zuerst einen Fork (eine Abzweigung/Kopie) des Repositorys anlegen. Das geschieht, indem man auf der Hauptseite eines Repositorys ganz rechts neben dem Namen des Repositorys auf die Schaltfläche „Fork“ und dann auf „Create a new fork“ klickt.

Danach öffnet sich ein Fenster, in dem man den Namen des neuen Forks ändern kann. Unten rechts klickt man zum Bestätigen auf den Button „Create Fork“. Nachdem der Kopiervorgang abgeschlossen ist, erhält man ein neues Repository, in dem alle Commits des kopierten Projektes enthalten sind. Unter dem Namen des Repositorys und über dem Inhalt wird darauf hingewiesen, dass es sich um einen Fork handelt und wie sich dieser vom Original unterscheidet.

Dieses Repository kann man jetzt auf den Computer klonen, Änderungen vornehmen und diese dann in das eigene Repository pushen.

Hat man die Änderungen vorgenommen und in diesen Fork gepusht, geht man zurück auf das Originalrepository und klickt in der Menüleiste unter dem Namen auf „Pull requests“ und dort rechts auf den grünen Button „New Pull Request“. Da hier die Änderungen aus einem anderen Repository kommen, muss man noch auf „compare across forks“ unter der „Compare changes“-Überschrift klicken.

Jetzt wählt man auf der rechten Seite den Fork aus; es werden direkt die Unterschiede angezeigt.

Nach einem Klick auf „Create pull request“ kann man in einem Eingabefeld noch Informationen zu den Änderungen im Projekt geben und dann noch einmal mit „Create pull request“ bestätigen.

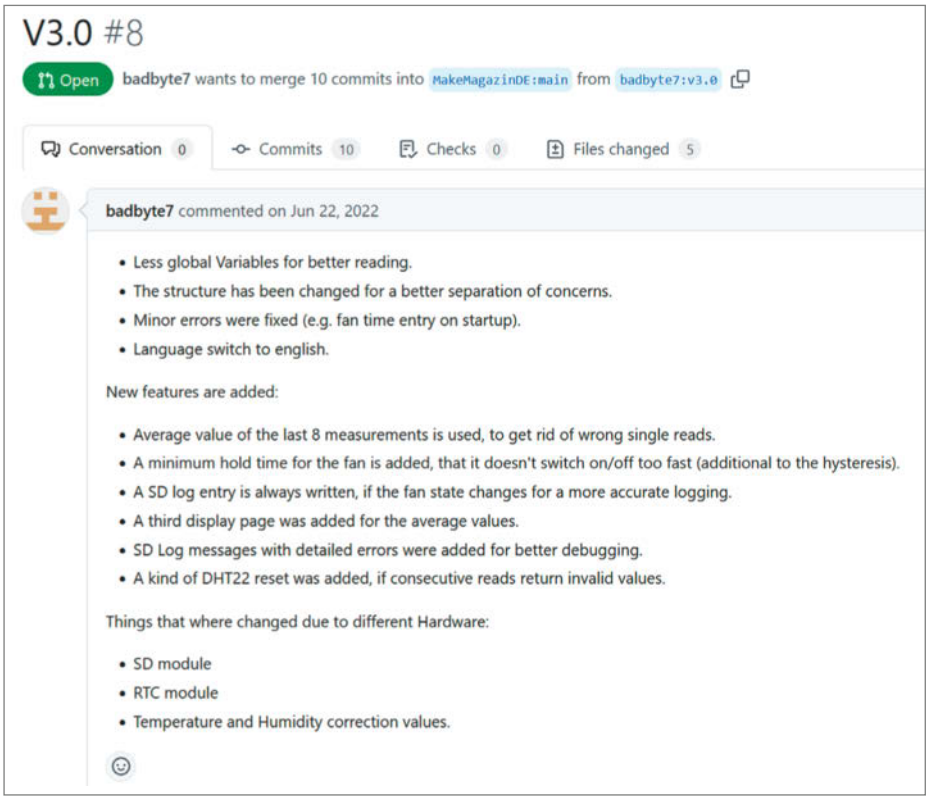
Jetzt ist der Besitzer des Repositorys an der Reihe, in das man mergen möchte. Der Besitzer des Repositorys bekommt eine Nachricht, dass neue Änderungen am Code vorgeschlagen wurden. Diese muss er jetzt überprüfen.

Dafür geht man als ursprünglicher Ersteller des Projekts auf GitHub bei dem Repository wieder auf die „Pull requests“-Seite, auf der alle Vorschläge aufgelistet sind. Klickt man auf einen, bekommt man eine neue Übersicht.

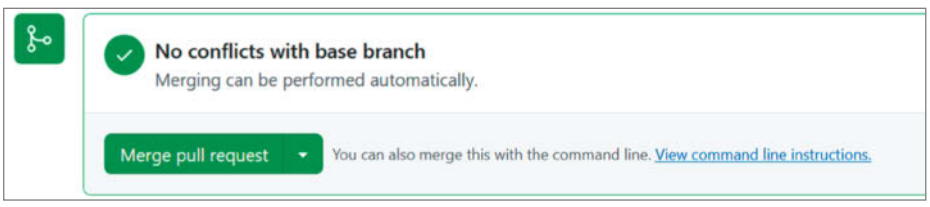
Im „Commit“-Tab werden auch die vorgenommenen Änderungen genau angezeigt. Wenn man in dieser Übersicht direkt erkennt, dass der vorgeschlagene Code einen Mehrwert hat, kann man diesen im Browser mit seinem mergen.

Über die Verlinkungen unter der Pull-Request-Überschrift kommt man auch zum Repository des Forks. Das ist wichtig, weil man dort das ganze Projekt herunterladen kann. Dann kann man überprüfen, ob die Änderungen funktionieren, sich das Projekt noch kompilieren lässt etc. Ist alles okay und man möchte die Änderungen in das Hauptprojekt übernehmen, klickt man im Pull Request auf „Merge pull request“.

Das erzeugt einen neuen Commit und kann dadurch auch wieder rückgängig gemacht



Die Beschreibung eines Pull Requests sollte so detailliert wie möglich sein. Hier ein Beispiel.



Einen Pull Request kann man direkt im Browser in das Projekt einfügen.

Klonen

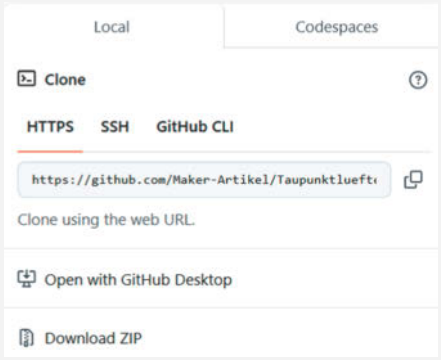
Um ein Repository lokal auf den Computer zu bekommen, muss man es klonen, also das Projekt inklusive aller Commits, Branches etc. und der Verbindung zum Onlinerepository herunterladen. Das passiert über den Konsolenbefehl `git clone <Onlinerepository-referenz>`. Die Referenz bekommt man auf der Repository-Seite über den grünen Button, auf dem „<> Code“ steht.

Dieser Button öffnet ein Menü, das uns drei Optionen zur Verfügung stellt: Klonen, mit der grafischen Oberfläche „GitHub Desktop“ öffnen oder Download als ZIP.

Für diesen Artikel wird wieder die Konsole genutzt. Für die Verwendung des grafischen GitHub-Clients gibt es online eine Anleitung. Diese ist in der Kurzinfo verlinkt.

Ein Repository lässt sich über drei Wege klonen: über HTTPS, SSH oder über GitHub CLI (Command Line Interface, Kommandozeile). Klonen über HTTPS und SSH ist bei allen Git-Services identisch nutzbar und basiert auf Git-Basisbefehlen. GitHub CLI ist ein spezielles Programm, das nur für GitHub funktioniert. Da es nicht standardisiert ist, liegt der Fokus in diesem Artikel auf HTTPS und SSH (siehe Kasten „SSH“).

Um über HTTPS zu klonen, kopiert man die URL, die unter HTTPS angezeigt wird, und fügt sie im Terminal hinter `git clone` ein. Wenn man jetzt mit Enter bestätigt, wird im aktuellen Ordner, in dem das Terminal geöffnet ist, ein neuer Ordner mit dem Namen des Repositorys angelegt und in diesen dann die Inhalte des Projektes heruntergeladen. Hat man aber schon einen Extraordner angelegt, kann man mit `git clone <Onlinerepository-referenz> .` (ein Leerzeichen und ein Punkt nach dem Befehl) direkt in den aktuellen Ordner speichern.



Die verschiedenen Arten, Code herunterzuladen.

Hat man ein volles Repository geklont, hat man jetzt alle Dateien, Commits und Branches, die online bereitgestellt wurden, lokal auf dem Computer. Das Projekt ist weiterhin mit der Online-Version verbunden. Wenn dort jetzt etwas Neues hochgeladen wird, kann man mit dem Befehl `git pull` diese Änderungen herunterladen.

Ein öffentliches Repository kann man immer klonen.

werden, sollten sich doch Probleme mit dem neuen Code ergeben.

Solche Pull Requests können auch gemacht werden, um Branches in den Haupt-Branch einzufügen. Das funktioniert genauso wie für Forks. Generell ist dieses Werkzeug dafür gedacht, dass parallel entwickelter Code in ein Hauptprojekt eingefügt wird, aber vorher noch vom Besitzer des Repositorys oder anderen Projektmitgliedern untersucht werden kann. Diese Überprüfung ist bei großen Open-Source-Projekten sehr wichtig. Dort gibt es immer wieder Versuche, schädlichen Code einzufügen. Aber sie bietet auch eine gute Möglichkeit, sich von anderen Programmierern den einen oder anderen Trick oder eine Herangehensweise abzuschauen.

Für kleine Pull Requests, in denen wirklich nur einzelne Codezeilen angepasst werden, ist das wirklich ein sehr einfacher Vorgang, bei dem man oft nicht einmal den Browser verlassen muss. Für große Änderungen sollte man sich aber immer das ganze geänderte Projekt anschauen und auf Funktionalität prüfen. Da kommt man um einen Download des Forks nicht herum.

Pull Requests werden über den Browser verwaltet, nicht über die Kommandozeile wie andere Git-Operationen.

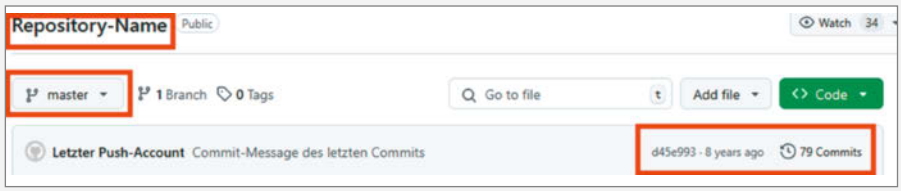
Mit diesem Wissen kann man mit Git nicht nur lokal seine Projektversionen verwalten, sondern auch auf ein GitHub-Repository hochladen. Dort stehen einem jetzt alle Möglichkeiten der Kollaboration offen. Andere können nun mit diesem Code ihre eigenen Projekte verwirklichen oder eigene Verbesserungen einbringen. Entweder als Teammitglied oder als dritte Partei, mit frischem Blick von außen.

Online gibt es einen Zusatzartikel, der die hier gezeigten Befehle anhand der grafischen GitHub-Desktop-Oberfläche zeigt. Er ist in der Kurzinfo verlinkt. *—das*

Das Onlinerepository

Bei einem Repository auf GitHub steht oben links immer der vergebene Name. Darunter befindet sich ein Drop-down-Menü, mit dem man durch die verschiedenen vorhandenen Branches schalten kann.

Darunter befindet sich dann das Dateifenster, in dem alle Dateien im Projekt angezeigt werden. Dieses Fenster verfügt über eine Kopfzeile, in der links der Name des Accounts steht, der als Letztes etwas zum Projekt gepusht hat. Direkt daneben steht der Text der letzten Commit-Message. Ganz rechts findet man die Commit-ID.



Von links oben nach rechts unten: Repository-Name, ausgewählter Branch, ID des aktuellen Commits und Commitanzahl.

Wenn man auf diese klickt, wird einem angezeigt, was im Repository mit diesem Commit verändert wurde. Und daneben ist ein Zähler aller Commits. Klickt man dorthin, bekommt man eine Liste aller Beiträge und kann sich anschauen, was diese Commits hinzugefügt und geändert haben.

SSH

SSH (Secure Shell) bei GitHub ist eine Möglichkeit, sicher mit Git-Repositories zu arbeiten, ohne jedes Mal Benutzernamen und Passwort eingeben zu müssen. Das Konzept dahinter basiert auf zwei Dateien, dem sogenannten Schlüsselpaar: einem privaten und einem öffentlichen Schlüssel. Der private Schlüssel bleibt sicher auf dem eigenen Computer gespeichert, der öffentliche Schlüssel wird bei GitHub hinterlegt. Wenn man sich per SSH verbindet, überprüft der Server, ob der öffentliche Schlüssel zum privaten Schlüssel passt. Dazu sendet der Server eine verschlüsselte Nachricht, die nur der passende private Schlüssel entschlüsseln kann. Gelingt das, weiß der Server, dass der richtige Benutzer zugreifen möchte, und gewährt den Zugang. Das Ganze passiert im Hintergrund, sodass man keine Passwörter mehr eingeben muss, sondern direkt Zugriff erhält.

Da der private Schlüssel niemals übertragen wird und die Authentifizierung ohne Passworteingabe (was sonst auch übertragen werden würde) erfolgt, schützt SSH vor Phishingangriffen und Brute-Force-Versuchen, weil der private Schlüssel nie den eigenen PC verlässt und dadurch nicht abgefangen werden kann.

Klonen mit SSH

Um ein Repository über SSH zu klonen, braucht man als Erstes das Schlüsselpaar. Das erzeugt man im Terminal mit dem Befehl `ssh-keygen -t ed25519 -C <E-Mail>`.

Danach wird man gefragt, ob man den Speicherort der Schlüssel ändern möchte. Das lässt man auf der Standardeinstellung und bestätigt einfach mit Enter. Dann wird man nach einer Passphrase für die Schlüssel gefragt. Hier kann man entweder ein Passwort setzen oder auch einfach mit Enter bestätigen, um keins zu setzen.

Danach werden die Schlüssel generiert und unter C:\Users\<Nutzername>\.ssh\ gespeichert. Dort befinden sich jetzt zwei Dateien mit dem Namen id_ed25519. Eine PUB-Datei und eine ohne Endung. Die ohne Endung ist der sogenannte Private Key. Der bleibt immer auf dem Rechner. Die PUB-Datei ist der Public Key. Diese Datei bzw. ihren Inhalt hinterlegt man jetzt auf GitHub. Dafür öffnet man die PUB-Datei mit dem Windows-Editor und kopiert den Inhalt.

Auf GitHub klickt man oben rechts auf den Account-Avatar und dann auf „Settings/SSH and GPG keys“. Auf dieser Optionsseite gibt es oben rechts den Button „New SSH key“. Auf den klickt man und kann dem Schlüssel danach einen Namen geben. Das sollte der Name des Computers sein, mit dem man diesen Schlüssel nutzt. Im Feld „key“ fügt man jetzt den kopierten Schlüsselinhalt ein. Als Letztes klickt man auf „Add SSH key“.

Jetzt kann man beim Klonen auf „SSH“ klicken, den dortigen SSH-Link kopieren und für den git-clone-Befehl nutzen. Anmelden mit Passwort fällt jetzt weg.

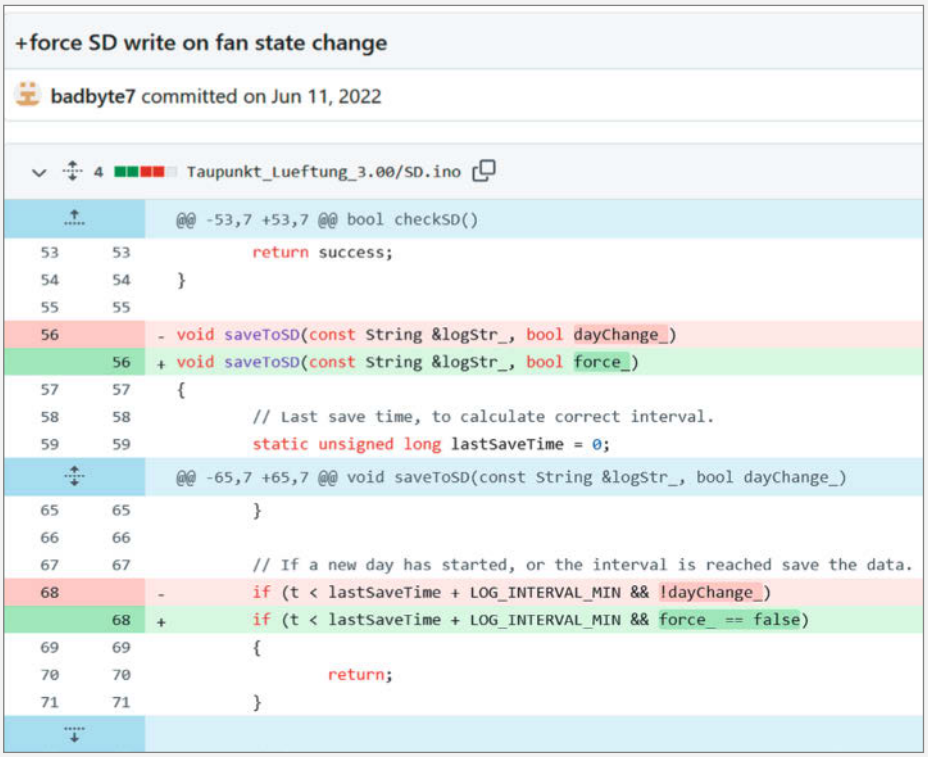
Vergleich von Codeversionen

Im Text wird immer wieder davon gesprochen, dass man sich auf GitHub die Unterschiede im Code zwischen Commits ansehen kann. Das sieht wie folgt aus:

Wenn eine Datei verändert wird, werden die Stellen mit neuen Inhalten farblich codiert. Rote Codezeilen sind Stellen, in denen etwas Altes verändert wurde. Die Zeile wird gelöscht und ersetzt. Der neue Inhalt wird grün angezeigt.

Diese Ansicht ist nützlich, um bereits bestehende Commits in einem Repository zu vergleichen und herauszufinden, wo genau Änderungen stattgefunden haben. Zum Beispiel könnte seit dem letzten Commit bei der Ausführung des Codes ein Bug auftreten, der vorher nicht da war. Mit dieser Anzeige sieht man sofort, welche Stellen verändert wurden und wo der Bug verortet sein kann.

Auch bei Pull Requests ist diese Ansicht wichtig. Dadurch kann man direkt alle Stellen überprüfen, die verändert wurden – und einen Code sowohl auf Programmierfehler als auch auf eventuell eingefügten Schadcode kontrollieren.



Vergleich aus einem Pull Request. Die geänderten Zeilen sind hervorgehoben.

Die Maker Faire lebt vom Antrieb und der Begeisterung eines jeden einzelnen Teilnehmers. Im interdisziplinären Zusammenarbeiten und dem Wissensaustausch liegen die Chancen für Fortschritt und Neues.

Gleichermaßen fasziniert die Maker Faire Kinder und Jugendliche für die digitale Welt, IT, KI und Technik. Das kreative und neutrale Umfeld vereinfacht ein ungezwungenes Recruiting von Talenten.

Unterstützt die Maker Faire als Partner oder Sponsor und erfahrt Wertschätzung für euer Unternehmen.

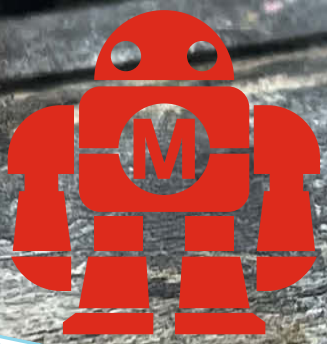
Jetzt für einen Stand anmelden!

Nähere Informationen und Preise unter:

www.maker-faire.de/partner **partner@maker-faire.de** **Tel.: +49 (0)511 5352 844**

Angenommen, du wärst ein elektronisches Bauteil oder Gerät – welches wäre es und warum?

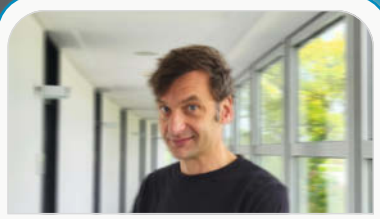
Unser Team



Daniel Bachfeld

Chefredakteur, dab@make-magazin.de

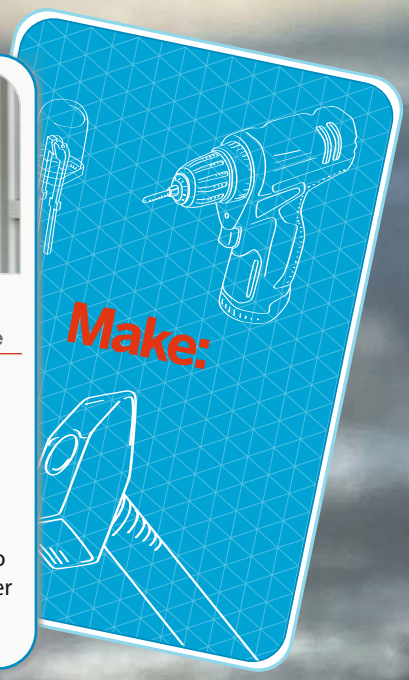
Ich wäre ein Standmixer. Als technischer Eklektiker vermische ich gerne verschiedene Technik-Disziplinen in Projekten. Ich teste auch gerne SBCs und µCs auf ihre Leistung und sehe Parallelen zur YouTube-Serie „Will it Blend?“, in der nicht nur Nahrungsmittel, sondern auch elektronische Geräte wie Camcorder und iPhones gegen einen Mixer antreten.



Marcus Hansson

Redakteur, mch@make-magazin.de

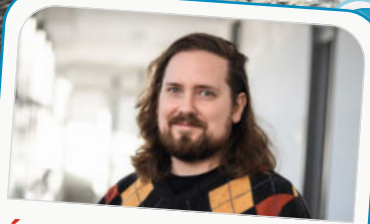
Ein Mikroprozessor, denn das menschliche Gehirn arbeitet bekanntlich mit elektrischer Signalübertragung. Mein Kopf ist also nichts anderes als ein Haufen wild verdrahteter Transistoren, die zusammen einen ATtiny85 bilden. „So small but so powerful“, wie es in der Beschreibung von Fritzing heißt.



Carsten Wartmann

Redakteur, caw@make-magazin.de

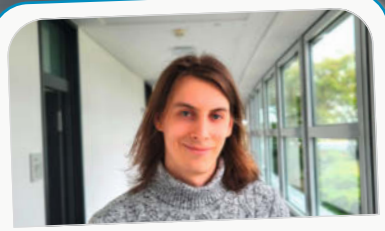
Ich wäre ein Euro-Rack-Modul. Ein Rauschgenerator. White Noise, Pink Noise, Sample & Hold. Alles etwas chaotisch, aber sehr zuverlässig. Gefiltert klinge ich plötzlich interessant und bin brauchbar. Ich bringe Leben in perfekte, sterile Klanglandschaften.



Ákos Fodor

Redakteur, akf@make-magazin.de

Bei den vielen Entscheidungen, die ich als Redakteur treffen muss, wäre ich sicher ein passables Logikgatter: entweder ein strenges AND, das alles auf Richtigkeit überprüft, oder ein wählerisches XOR – und als OR durchaus auch mal zu Kompromissen bereit. Nur ein stures NOT wäre ich schon aus Prinzip nicht.



Daniel Schwabe

Technical Writer,
das@make-magazin.de

Eindeutig ein NeoPixel. Hübsch leuchten and chill. Ich persönlich freue mich immer, wenn Projekte blinken, das passt also perfekt. Falls ich Teil einer größeren Matrix wäre, könnte man über mich auch Informationen weitergeben. Das passt zu meinem Beruf. Oder ich werde Teil einer 80er-Sci-Fi-Maschine.

Dunia Selman

Social-Media-Managerin,
dus@make-magazin.de

Wenn ich ein elektronisches Bauteil wäre, wäre ich vermutlich ein Thermistor. Ich bin ziemlich kälteempfindlich. Aber wie ein Thermistor, der auf äußere Einflüsse reagiert, stelle ich mich auch gerne quer, wenn es nötig ist. Dazu brauche ich aber keine Temperaturschwankungen.

Nicole Wesche

Mediengestalterin, niwe@heise.de

Ich sehe mich als ein Digital/Analog-Wandler. Als Mutter ist es meine Aufgabe, eine Brücke für meine Kinder zu sein, ihre Wünsche und Bedürfnisse aufzunehmen, zu deuten und schließlich so zu übersetzen und zu verstärken, dass sie in der „analogen Welt" richtig verstanden werden.

Daniel Rohlfing

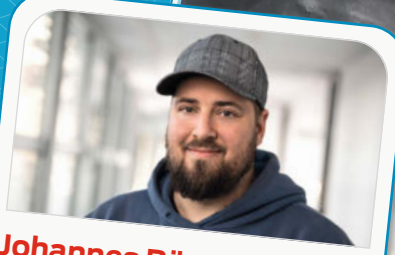
Leiter Events und Sales,
dnr@maker-faire.de

Eine Siebträgermaschine: blendend angezogen und schön anzuschauen. Stets heiß, für viele unverzichtbar, spende ich bei Wertschätzung und der richtigen Pflege langfristig Genuss und schnelle Energie.

Kristina Fischer

Projektleitung Maker Faire,
krfi@maker-faire.de

Ich wäre ein Satellit. Zum einen kann ich so die ganze Welt (und noch mehr) sehen. Ohne mich würde zudem nichts funktionieren, GPS, Kommunikation, Wettervorhersagen, Umweltbeobachtungen – ich bekomme einfach alles mit. Entspannte Schwerelosigkeit mit Blick auf die Weltwunder der Natur – oder sogar auf Area 51?



Johannes Börnsen

Redakteur und YouTube-Host,
jom@make-magazin.de

Laut meinem Sohn bin ich ein 3D-Drucker. Seit er versteht, dass dieses Gerät Wünsche erfüllen kann, sprudeln die Ideen! Ich wäre aber lieber das Blitzdings aus „Men in Black". Dann könnte ich unseren Chefredakteur vergessen lassen, dass ich vor lauter Druckaufträgen diesen Text zu spät abgegeben habe.

Rebecca Poweleit

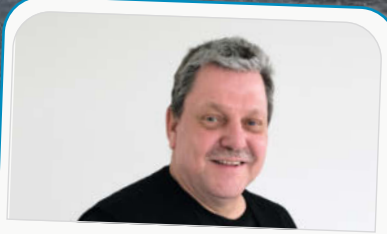
Werkstudentin Social Media,
repo@make-magazin.de

Ich wäre ein Akku. Denn meistens bin ich voller Energie, aber ab und zu brauche ich auch mal eine Pause, um meine Social Battery wieder aufzuladen. Danach habe ich aber sofort wieder Lust, coole Sachen zu unternehmen.

Alexander Panknin

Redakteur, pan@make-magazin.de

Ich wäre ein Noise-Generator – für die einen nur zufälliges Rauschen, für die anderen die Quelle neuer kreativer Möglichkeiten. Manchmal braucht es ein bisschen Chaos, um wirklich innovative Ideen zu erzeugen! Achtung: Kann in hohen Dosen zu unerwarteten Ergebnissen oder leichter Verstörung führen.



Ulrich Schmitz

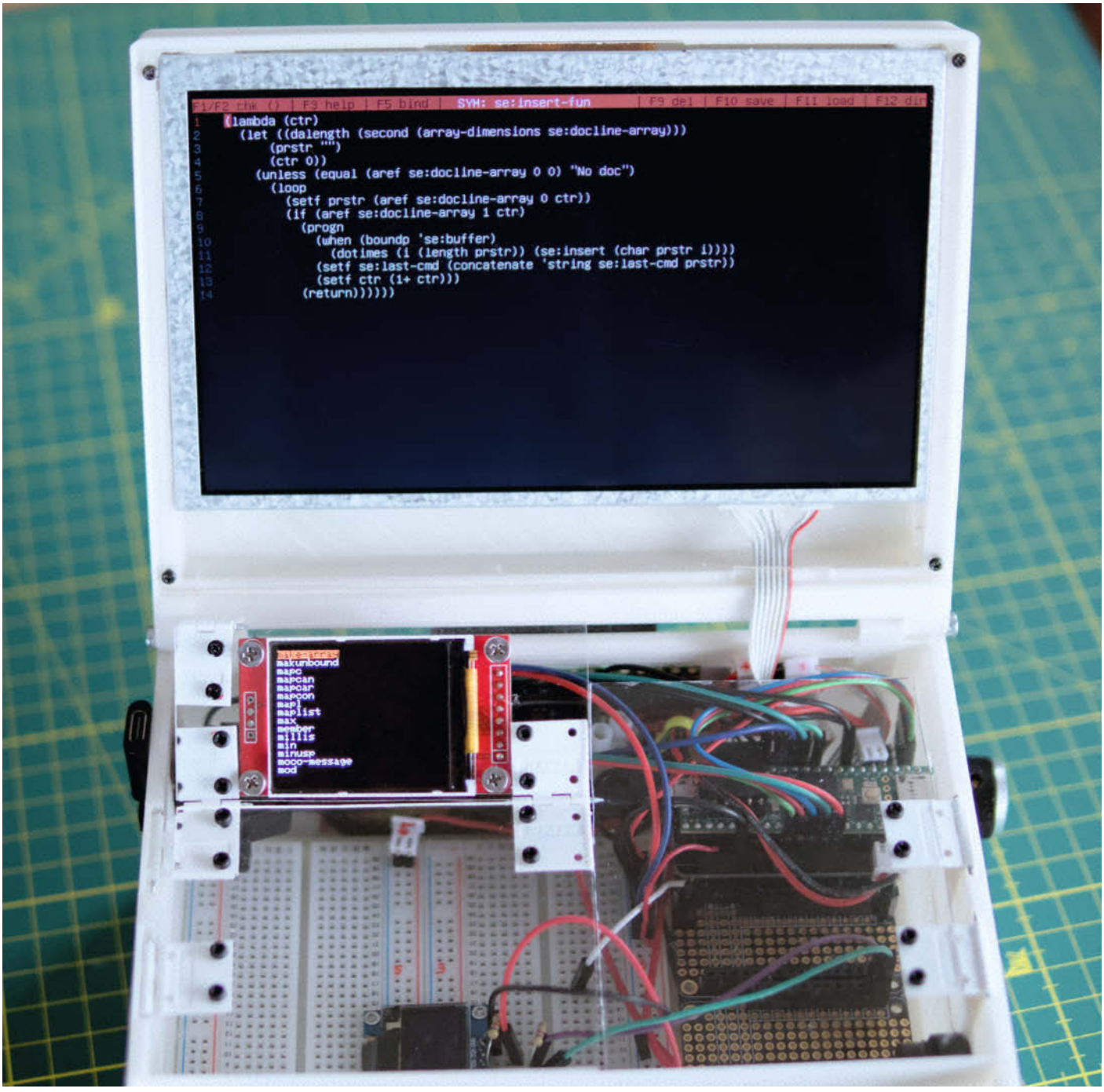
Redakteur, usz@make-magazin.de

Ein Kondensator, denn er verkörpert viele Eigenschaften, die mir sympathisch sind: Er ist fähig, schnell Energie aufzunehmen, zu speichern und bei Bedarf abzugeben. Gleichzeitig zuverlässig, sehr traditionell und beständig. Und wenn alles etwas zu viel wird, kann es auch schon mal passieren, dass er explodiert.

Lisp-Box: Minidisplay als Programmierhilfe

Ein Drehencoder und ein zusätzlicher Bildschirm machen die Lisp-Box aus der Make 7/24 jetzt noch praktischer. Während das Display hilfreiche Informationen zu allen verfügbaren Programmierbefehlen liefert, lässt es sich mit dem Encoder komfortabel wie ein Rolodex bedienen.

von Hartmut Grawe



Zwar benutze ich uLisp auf meiner Lisp-Box inzwischen fast täglich, aber die oft etwas eigentümlichen, historisch gewachsenen Symbolnamen der Programmiersprache kann ich mir zum Teil immer noch nicht merken. Gleichzeitig ist es unpraktisch, jedes Mal auf einem anderen Gerät in der Online-Dokumentation nachschauen zu müssen. Deshalb habe ich die internen Hilfstexte von uLisp mithilfe eines zusätzlichen Displays nutzbar gemacht, die sich dadurch wie ein Lexikon in der REPL (Read-Eval-Print-Loop) oder im Editor verwenden lassen. Die Steuerung erfolgt über einen Drehgeber und die Tastatur. Wie man die Lisp-Box mit diesem Minilexikon erweitert, zeige ich in diesem Artikel.

Mehr ist besser

Ich hatte mir in den Kopf gesetzt, das Lexikon zwecks gesteigerter Coolness auf einem 1,8 Zoll großen Zusatzdisplay (ST7735 mit SPI) auszugeben. Wieso nur ein Display nutzen, wenn man auch zwei haben kann? Dafür musste ich zunächst Platz finden, ohne den Experimentierraum einzuschränken, und montierte den Bildschirm schließlich unter der linken transparenten Seitenklappe. Dort ist er geschützt und befindet sich in unmittelbarer Nähe zum Hauptmonitor, verdeckt diesen aber nicht und lässt sich mit einer Stütze in einem gut ablesbaren Winkel aufstellen. Die Anschlusslitzen sind dort verstaut, wo sie idealerweise sein sollten: im Kabelschacht hinter den Breadboards.

Der Einbau des Displays ist kein Hexenwerk: Ich habe die 1 mm dicke PETG-Klappe gegen eine 2-mm-Plexiglasscheibe ausgetauscht, damit sie das Bildschirmmodul besser hält. Direkt unterhalb der Displaykante habe ich das Plexiglas geteilt und mit zwei weiteren Modellbauscharnieren befestigt. Ein drittes Scharnier ist die Stütze, die wie bei der Motorhaube eines Autos funktioniert und im Kabelschacht verschwindet, sobald man die Box schließt (Bild 1).

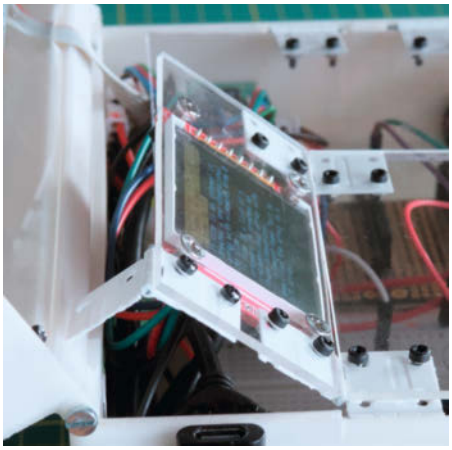


Bild 1: Das Display ist geschützt und lässt sich mit einer Stütze temporär aufstellen.

Kurzinfo

- Lisp-Box um nützliches Hilfsdisplay erweitern
- Drehgeber mit uLisp verwenden und entprellen
- Code-Vorlagen per Knopfdruck in den Editor oder die REPL einfügen

Checkliste

Zeitaufwand:
ein Nachmittag

Kosten:
10 bis 15 Euro

Material

- **1,8-Zoll-TFT-Display** mit 160×128 Pixeln, ST7735, SPI
- **KY-040-Encoder** mit Befestigungsgewinde, schmale Platine, vormontierte Widerstände
- **2-mm-Plexiglasscheibe** etwa 85×40 mm
- **JST-Buchse** 8-polig
- **Kavan Ruderscharniere** 34×16 mm
- **M2-Schrauben und Muttern**

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

Werkzeug

- **Lötkolben**
- **Cuttermesser**
- **Schraubendreher**
- **Crimpzange** optional

Mehr zum Thema

- Hartmut Grawe, Die Lisp-Box, Make 7/24, S. 74
- Andreas Voß, Volumio mit Drehgebern erweitern, Make 3/24, S. 118
- Video: Lisp-Box-Erweiterung in Aktion

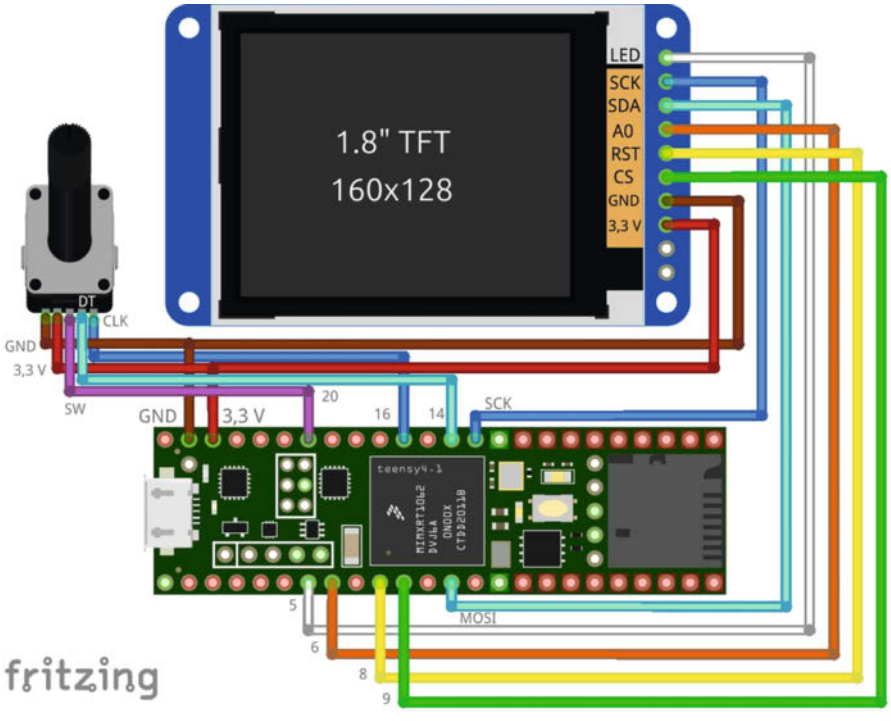


Bild 2: So verbindet man den Zusatzbildschirm mit dem Teensy in der Lisp-Box.

Die Verbindungen zwischen Display und Teensy müsst ihr so vornehmen, wie in Bild 2 zu sehen. Achtet dabei darauf, dass die Pins bei eurem Bildschirm möglicherweise anders beschriftet oder in einer anderen Reihenfolge angeordnet sein können. Ein SPI-Anschluss (Serial Peripheral Interface) am Displaymodul ist in jedem Fall Pflicht, da I²C (Inter-Integrated Circuit) für dieses Vorhaben zu langsam reagieren würde. Um Fehler bei der schnellen Übertragung zu vermeiden, habe ich ca. 15 cm kurze Kabel verwendet. Softwareseitig wird der Zweitbildschirm von den Grafikroutinen von uLisp angesprochen, die in der Firmware bereits aktiviert sind.

Eindeutige Signale

Die mechanischen Kontakte des Encoders erzeugen beim Drehen ein kleines Feuerwerk uneindeutiger Signalflanken. Diese mit Hardware zu entprellen, wie in der Make 3/24 von Andreas Voß beschrieben (siehe Link in der Kurzinfo), wäre elegant, hätte in meinem Fall aber ein gewisses Platzproblem verursacht. Daher habe ich mich für eine Softwarelösung entschieden, wie in Listing „se:wait-encoder" zu sehen.

Diese Funktion befindet sich in der Datei LispboxLibrary.h und lässt sich auch im Lisp-Box-Editor betrachten, wenn man in der REPL

```
(se:sedit 'se:wait-encoder)
```

eingibt. Sie löscht zunächst den Tastaturpuffer, setzt die lokalen Variablen für die drei Eingabemöglichkeiten (Encoder drücken, rotieren oder Tastatureingabe) auf null oder false und wartet in der darauffolgenden Endlosschleife, bis eine der `when`-Bedingungen erfüllt ist. Dann gibt sie die erfasste Eingabe als Liste zurück.

Die beiden `unless`-Blöcke innerhalb der Schleife fragen ab, ob sich der Zustand des `SW`- oder `CLK`-Pins geändert hat. Bei einem Tastendruck gibt es eine kleine Verzögerung von 2 Millisekunden vor dem Speichern des Pin-Zustands, was den Taster ausreichend entprellt. Sobald man den Encoder dreht, legt `(setf se:clk-state (digitalread se:CLK))` zunächst den Zustand des Drehgebers in einer globalen Variable ab und die nächsten Zeilen ermitteln die Drehrichtung.

Die lokale Variable `rot-event` erfasst dabei die Drehung und zählt die Drehereignisse. Das ist notwendig, da der KY-040 auch zwischen den spürbaren Rastungen einen Schritt signalisiert. Wie beim Rolodex wollen wir aber, dass nur ein Einrasten den nächsten Eintrag ansteuert. Also ignoriere ich den Zwischenschritt, indem ich jeweils zwei Drehereignisse abwarte.

Ein Ding zum Drehen

Das uralte mechanische Rollkarteisystem Rolodex funktioniert unerreicht intuitiv und praktisch: Um einen Karteieintrag zu finden, dreht man an einem seitlichen Rädchen und blättert so durchs Alphabet (Bild 3). Exakt diese Idee wollte ich für das Lexikon umsetzen und habe der Lisp-Box daher einen Drehencoder (KY-040) spendiert und diesen softwareseitig entprellt (siehe Kasten „Eindeutige Signale"). Es gibt ihn mit vorgelöteten Pull-up-Widerständen auf kleinen Breakout-Boards, die kaum breiter sind als der eigentliche Encoder.

shutterstock.com/Brett Hondow

Bild 3: Alt, aber gut. Mit einem Dreh findet man schnell, was man sucht.

se:wait-encoder (LispboxLibrary.h)

```
(defun se:wait-encoder ()
  (keyboard-flush)
  (let ((sw-event nil)
        (rot-event 0)
        (lkd nil))
        (loop
          (when (or (eq sw-event t) (= rot-event 2) lkd) (return))

          (unless (eq (digitalread se:SW) se:sw-staste)
            (delay 2)
            (setf se:sw-state (digitalread se:SW))
            (setf sw-event t)
          )

          (unless (eq (digitalread se:CLK) se:clk-state)
            (delay 1)
            (setf se:clk-state (digitalread se:CLK))
            (if (eq (digitalread se:CLK) (digitalread se:DT))
              (setf se:direction "CCW")
              (setf se:direction "CW")
            )
            (setf rot-event (1+ rot-event))
          )

          (setf lkd (keyboard-get-key t))
        )
        (list sw-event rot-event lkd)
  )
)
```

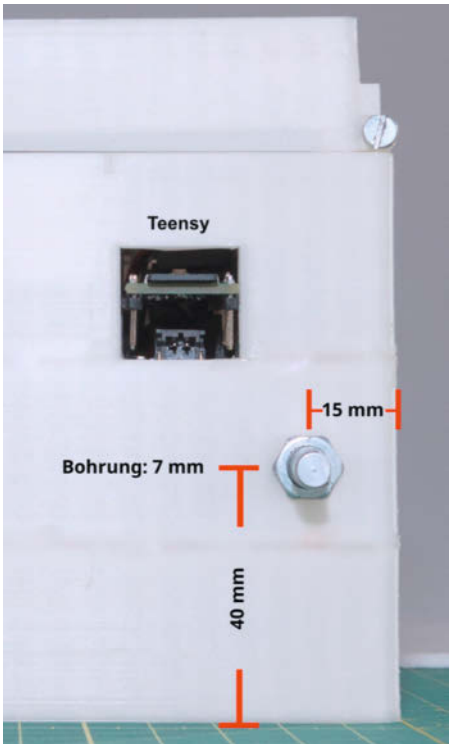


Bild 4: Wenn man das Loch für den Drehencoder hier bohrt, …

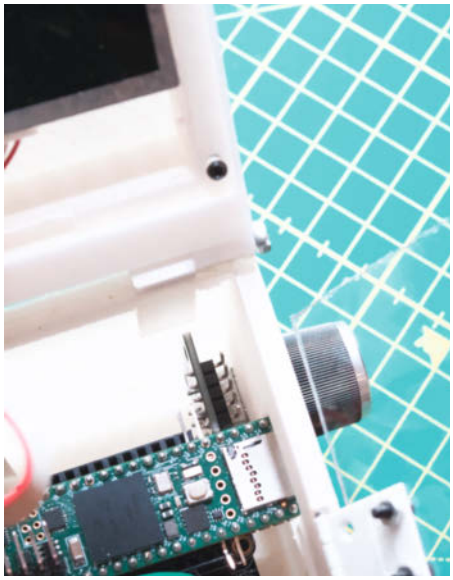


Bild 5: … passt er neben den Teensy.

Wie es der Zufall will, ist das Eingabegerät sogar klein genug, dass man es direkt hinter dem Teensy im Bauraum der Lisp-Box verstauen kann.

Um den Encoder zu montieren, bohrt ihr an der Position, die Bild 4 zeigt, vorsichtig ein Loch mit 7 mm Durchmesser in das Gehäuse, steckt die Welle des Encoders samt Montagegewinde hindurch und fixiert ihn mit der mitgelieferten Mutter. Die Anschlusspins der Platine sollten nach oben zeigen, damit sie gut zugänglich bleiben (Bild 5). Auf die Welle des Encoders gehört dann noch ein passender Drehknopf. Danach verbindet ihr das Modul mit 3,3 Volt und Masse (GND) sowie CLK (Clock) mit Pin 16 des Teensy, DT (Data) mit Pin 14 und SW (Switch) mit Pin 20.

Helferlein im Editor

Sobald ihr die aktuelle Firmware aus dem GitHub-Repository des Projekts heruntergeladen und auf den Teensy eurer Lisp-Box geflasht habt, könnt ihr die neue Hilfsfunktion direkt ausprobieren. Öffnet dazu den Editor in der REPL mit `(se:sedit)` und drückt danach F3 auf der Tastatur. Daraufhin meldet das Hilfsdisplay: „Press key or encoder!“ Drückt ihr nun eine alphanumerische Taste, erscheint eine scrollbare Liste aller uLisp-Symbole, die mit dem gewählten Zeichen beginnen (Bild 6). Gibt es keine, wird „No entry“ angezeigt. Habt ihr statt auf die Tastatur den Encoder gedrückt oder gedreht, seht ihr die Liste aller verfügbaren Symbole, also aller Funktionen und globalen Variablen.

Dreht nun den Encoder, bis der Eintrag hervorgehoben ist, den ihr euch ansehen wollt, und drückt ihn danach erneut. Schon zeigt das Display den Hilfstext zu dem Symbol an, sofern vorhanden (Bild 7). Die zugehörigen Informa-

Kein Haken, kein Stottern

Je schneller man den Encoder dreht, desto häufiger schleichen sich falsche Messwerte ein, d. h. angebliche Drehungen in die andere Richtung. Dem Problem begegne ich mit der Funktion `se:eval-encoder` (siehe gleichnamiges Listing). Wenn ich am Encoder drehe, braucht ein Richtungswechsel meiner Handbewegung einige Millisekunden. Daher habe ich in die Auswertung eine zeitliche Trägheit eingebaut: Wird ein Richtungswechsel nach zu kurzer Zeit gemeldet, handelt es sich sehr wahrscheinlich um eine falsch erfasste Bewegung, und ich passe die gemeldete Richtung einfach der vorherigen an.

Möglich macht das die uLisp-Funktion `for-millis`. Mit ihr kann man eine Kette von Ereignissen auslösen und festlegen, wie lange die Ausführung in Millisekunden dauern soll, oder – wenn man keine Zeitspanne angibt – herausfinden, wie viele ms das Abarbeiten einer Funktion gedauert hat. Dies mache ich mit `se:wait-encoder` und speichere die Zahl zunächst ab.

Hat `se:wait-encoder` innerhalb von `for-millis` eine Drehung erfasst, sehe ich jedes Mal nach, wie viel Zeit für diesen Ablauf von zwei Encoder-Meldungen vergangen ist. Sind es weniger als 10 ms und mir wurde gleichzeitig eine Änderung der Drehrichtung zurückgemeldet, sage ich frech: So schnell ist meine Hand nicht! Also wird die angeblich neue Drehrichtung `se:direction` der vorherigen angepasst und in `se:previous-direction` gespeichert.

Dieser kleine Trick filtert Falschmeldungen zuverlässig aus, und das Zusammenspiel von Drehgeber und Hilfslexikon fühlt sich in der Summe genauso weich an, wie es soll.

se:eval-encoder (LispboxLibrary.h)

```
(defun se:eval-encoder (cstart cmax rot-fun sw-fun &optional key-fun)

  (let ((sw-event nil)
        (rot-event 0)
        (key-event nil)
        (wait-time 0)
        (result nil)
        (ctr cstart))

        (loop
          (setf wait-time (for-millis () (setf result (se:wait-encoder))))
          (setf sw-event (first result))
          (setf rot-event (second result))
          (setf key-event (third result))

          (when key-event
            (when key-fun (key-fun key-event))
            (keyboard-flush)
            (return key-event)
          )
          (when (and sw-event se:sw-state)
                (sw-fun ctr)
                (return nil)
          )
          (when (= rot-event 2)
                (when (and (< wait-time 10) (not (equal se:direction
                         se:previous-direction)))
                  (setf se:direction se:previous-direction)
                )
                (if (equal se:direction "CCW")
                  (setf ctr (min cmax (1+ ctr)))
                  (setf ctr (max 0 (1- ctr)))
                )
                (rot-fun nil ctr)
                (setf rot-event 0)
                (setf se:previous-direction se:direction)
          )
        )
        key-event
  )
)
```

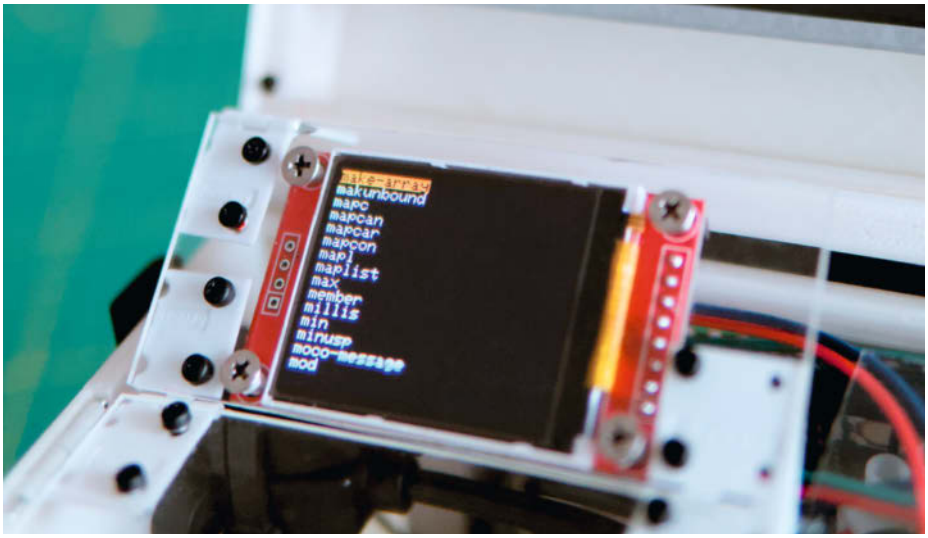


Bild 6: Wenn man auf der Tastatur „m“ drückt, nachdem man das Helferlein aktiviert hat, erscheinen alle Symbole mit diesem Anfangsbuchstaben in einer Liste.

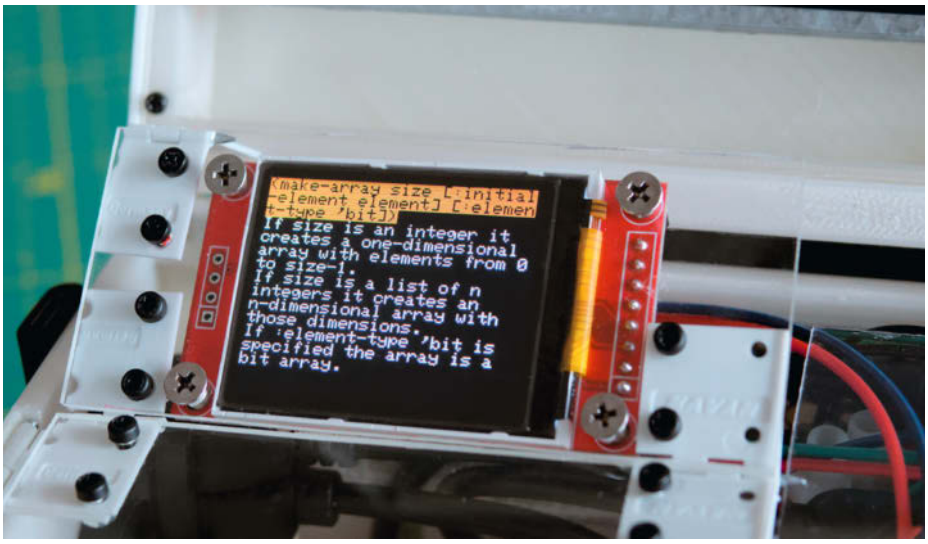


Bild 7: Ein weiterer Druck auf den Encoder öffnet den Hilfstext.

Effiziente Hilfe

Das Helferlein holt sich die Liste der verfügbaren Symbole aus der eingebauten uLisp-Funktion `apropos-list` und arbeitet sie so auf, dass nur Einträge mit dem gewählten Anfangsbuchstaben übrig bleiben. Das erledigt der Befehl `mapcan` innerhalb der Funktion `se:ref` (in LispboxLibrary.h). Wenn man ihm eine Funktion übergibt, wird diese auf jedes Element einer Liste angewendet. Hier ist das die anonyme Funktion `(lambda (x) [...]`, die den Anfangsbuchstaben jedes String-Elements in einer Liste prüft. Dann hängt `mapcan` die jeweilige Rückgabe der Funktion an eine neue Liste an und liefert diese am Ende als Ergebnis. Den ausführlichen Hilfstext zu einem Symbol bzw. Lisp-Namen liefert die eingebaute Funktion `documentation`.

In beiden Fällen formatiert das Helferlein die Strings passend zum TFT-Display und wirft jeweils eine fertig formatierte Zeile als neuen String in ein Array. Das beschleunigt die Ausgabe erheblich, denn ein Element aus einem Array zu entnehmen, dauert bei klassischer Umsetzung immer gleich lange, egal wo sich das Element im Array befindet.
Bei einer Liste hingegen ist die Zugriffszeit umso größer, je weiter hinten sich das Element in der Liste befindet, da Lisp hier zunächst alle vorherigen durchlaufen muss.

tionen sind zu einem Großteil in dem Kern von uLisp als Strings gespeichert (ulisp-lispbox.ino, ab Zeile 5804), aber auch in LispboxExtensions.ino und teilweise in der Datei LispboxLibrary.h (Bild 8) zu finden. Wie die Lisp-Box nach den Begriffen sucht und sie sortiert, erklärt der Kasten „Effiziente Hilfe“.

Zu den meisten Symbolen gibt es auch einen Funktionsprototyp, also einen leeren Dummy. Wenn ihr aus dem Hilfstext heraus auf den Encoder drückt, wird dieser Prototyp in den Editor geschrieben, sodass man ihn anpassen kann, und das Lexikon kehrt zur Liste zurück. Mit F3 könnt ihr das Helferlein wieder beenden.

Unterstützung auch in der REPL

Mit F3 gelangt ihr auch in der REPL in die Hilfsfunktion. Allerdings lassen sich die Funktionsprototypen hier nicht direkt ausgeben. Der Grund: Ein laufendes Programm kann nicht in die nächste (an das Ende des Programms anschließende) Eingabezeile der REPL schreiben. Stattdessen merkt sich das Helferlein alle Funktionsprototypen, wenn ihr den Encoder

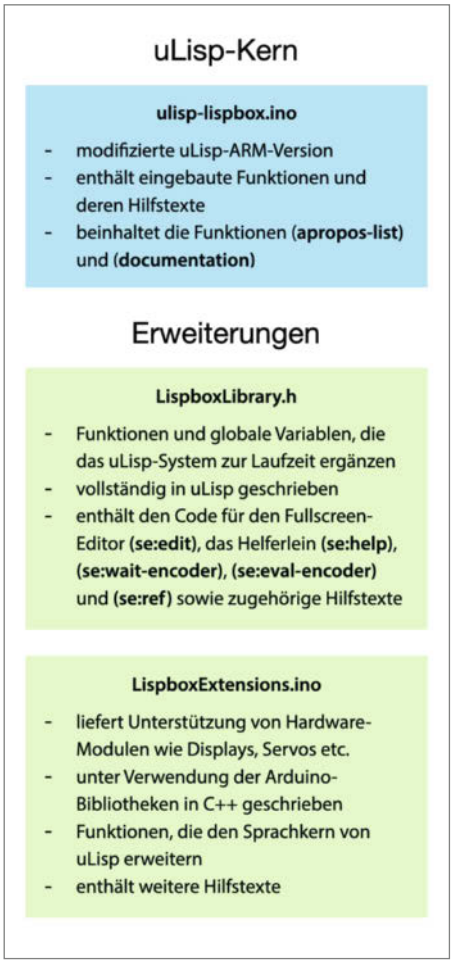


Bild 8: Die Hilfstexte und Funktionen der Lisp-Box sind auf mehrere Dateien verteilt.

drückt, und gibt sie beim Beenden des Lexikons als Rückmeldung aus, sodass man sie als Vorlagen hat.

Lisp-Box weitergedacht

Das hier vorgestellte Projekt hat mir Lust gemacht, einen zweiten eigenständigen Lisp-Computer mit etwas anderer Ausrichtung zu entwerfen: Das Lisp-Deck ist eine flache Handheld-Variante der Lisp-Box mit kleinerem Experimentierraum, die einen 5-Zoll-Monitor mit 800×480 Pixeln Auflösung nutzt (Bild 9). Das Helferlein mit dem Drehencoder ist genauso an Bord wie ein Funkmodul. Andere Bestandteile lassen sich einsetzen.

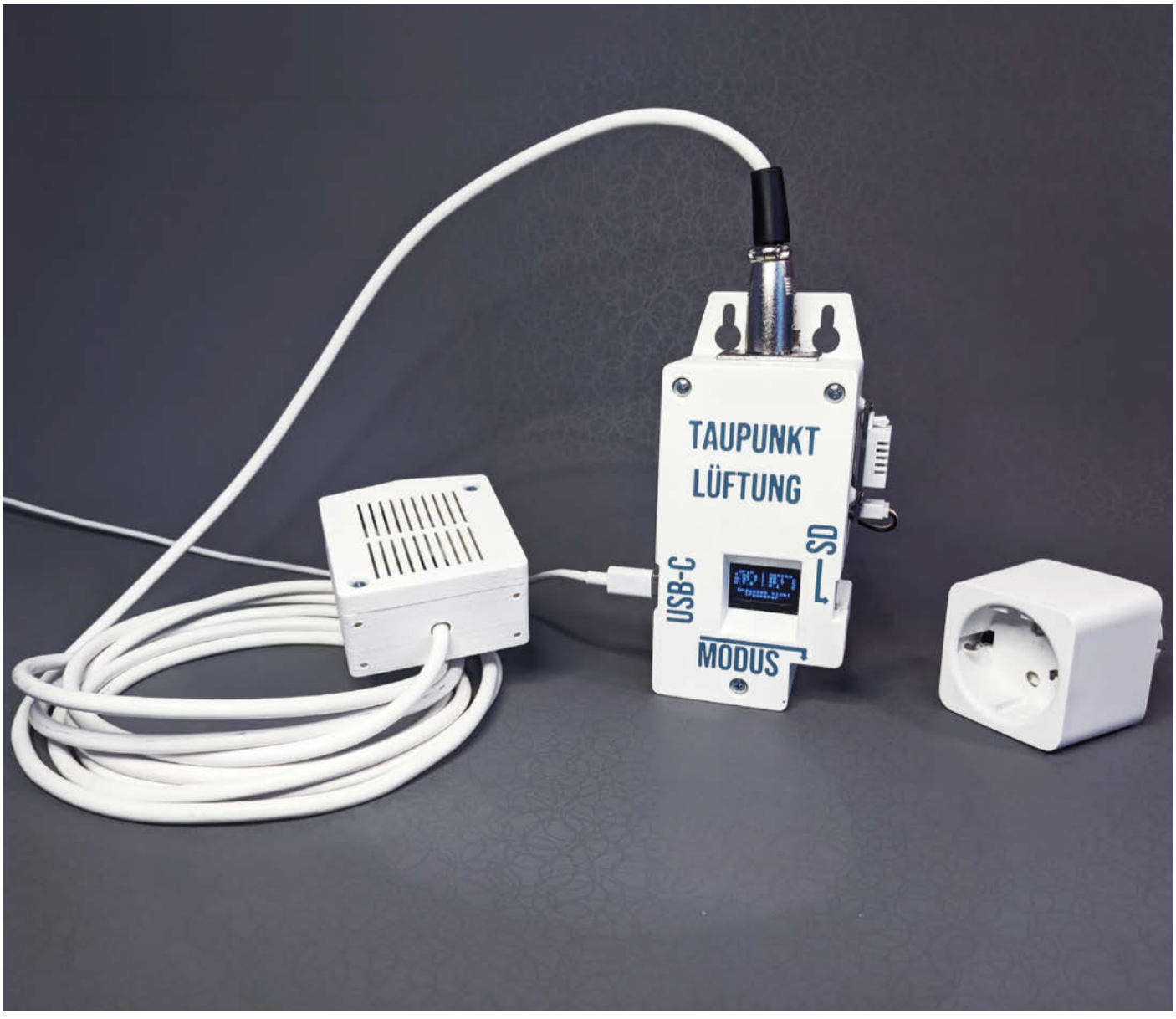
Ihr findet die 3D-Druckdaten ebenfalls im GitHub-Repository der Lisp-Box. Die Firmware ist identisch und das Lisp-Deck daher 100 Prozent kompatibel mit der Lisp-Box – aber für das Zurücklehnen im Sessel und zum Mitnehmen besser geeignet. Und wer weiß, wofür man das Zusatzdisplay noch nutzen könnte. *—akf*

Bild 9: Kompakter als die Lisp-Box, aber mit denselben Funktionen bestückt.

Taupunkt-Lüftung mit ZigBee-Funksteckdose

Der neue ESP32-C6 hat eine ZigBee-Antenne und lässt sich dank Erweiterungsboard mit einem Display und Steckanschlüssen ausrüsten. So kann man einen Taupunkt-Lüfter ohne 230-V-Verkabelung oder komplexe Lötarbeiten umsetzen.

von Johannes Kreuzer



Der Taupunkt-Lüfter aus der Make 1/22 hat in unserem heimischen Keller seinen festen Platz, damit immer dann gelüftet wird, wenn es draußen trockener als drinnen ist. Die bei uns verbaute, klassische Variante mit einem 230-V-Netzteil und Relais neben einem Arduino Nano auf einer Lochrasterplatine führte jedoch beim Anlaufen des Gebläses in der Vergangenheit zu Aussetzern in der Datenaufzeichnung, da quasi keine Filterung vorhanden ist.

Für einen Nachbarn habe ich daher eine robustere Version ohne 230-V-Verkabelung bauen wollen. Da in dessen Keller kein WLAN zur Verfügung steht und die Installation von Home Assistant den Rahmen gesprengt hätte, kam die Nachricht über den relativ neuen ESP32-C6 gerade recht, der über eine ZigBee-Antenne verfügt und damit beide Probleme auf einmal löst. Bild 1 zeigt eine Übersicht der neuen Taupunkt-Lüftung ohne selbst verkabelte Netzspannungsrelais oder WLAN.

Die Sensoren, die Steuerung und das Datenlogging werden komplett vom 230-V-Netz und der Ansteuerung des Lüfters getrennt: Zum Schalten des Lüfters wird eine ZigBee-Steckdose verwendet, wie sie beispielsweise von Ikea verkauft wird. Die drahtlose ZigBee-Verbindung sorgt für einen potenzial-getrennten und damit störungsfreien Betrieb. Die Steuerung selbst wird ausschließlich von einem USB-C-Handy-Netzteil versorgt.

Kurzinfo

- **Ansteuerung einer ZigBee-Funksteckdose mit XIAO ESP32-C6**
- **Das Erweiterungsboard bringt alles fürs Datenlogging mit**
- **Keine Verdrahtung von Netzspannung nötig**

Checkliste

Zeitaufwand:
4 bis 6 Stunden

Kosten:
80 Euro

Material

- **XIAO ESP32-C6 und Expansion Board von Seeed Studio**
- **2 Temperatur- und Feuchtigkeitssensoren DHT22**
- **ZigBee-Funksteckdose**
- **XLR-Kabel und -Buchse**
- **microSD-Karte, CR1220-Batterie**
- **Handynetzteil**

Werkzeug

- **3D-Drucker** (für das Gehäuse)
- **Einfacher Lötkolben**

Mehr zum Thema

- Ulrich Schmerold, Das Taupunkt-Lüftungssystem, Make 1/22, S. 22
- Ulrich Schmerold, Logging-Funktion für das Taupunkt-Lüftungssystem, Make 2/22, S. 82
- Ákos Fodor, Von Arduino zur PlatformIO IDE, Make 1/25, S. 8a

Alles zum Artikel im Web unter *make-magazin.de/x3z2*

Steuerungselektronik

Das Herz der Steuerung ist der ESP32-C6, der hier im Modul XIAO-ESP32-C6 von Seeed Studio verwendet wird. Dieses lässt sich einfach auf das XIAO Expansion Board aufstecken und bildet damit eine ideale Plattform für das Projekt: Auf der Erweiterungsplatine sind zwei Grove-Anschlüsse für die beiden DHT22-Sensoren zu finden. Einer davon wird direkt an das Steuerungsgerät angeschlossen. Den zweiten – und hierfür ist die einzige Lötarbeit erforderlich – führt man über ein drei Meter langes XLR-Kabel nach draußen.

Das Expansion Board bringt außerdem noch ein kleines, aber zweckdienliches Display und einen Taster für die Modusauswahl mit. Auf der Unterseite befinden sich außerdem ein microSD-Kartenslot und eine RTC (Real Time Clock). Mithilfe einer Knopfzelle läuft die Uhr auf dem Erweiterungsboard auch ohne USB-Netzteil weiter und stellt somit einen robusten Datenlogger dar. Das Expansion Board bietet zudem noch eine Lademöglichkeit für einen Akku sowie einen Buzzer, aber beide bleiben bei diesem Projekt ungenutzt.

Arduino jetzt mit ZigBee-Support

Weil sich die ZigBee-Bibliothek für den Arduino noch in der Entwicklung befindet und ich mehrfach die Version wechseln musste, habe ich mein Projekt schließlich mit PlatformIO entwickelt. Die einfache Versionsverwaltung von Bibliotheken über die platformio.ini-Datei ist hier Gold wert! In Make 1/25 haben wir einen Einstiegsartikel zu PlatformIO veröffentlicht, den Link finden Sie in der Kurzinfo.

Die Unterstützung von ZigBee als Arduino-Library ist recht neu und so habe ich beim Schreiben dieses Projekts auch ein wenig

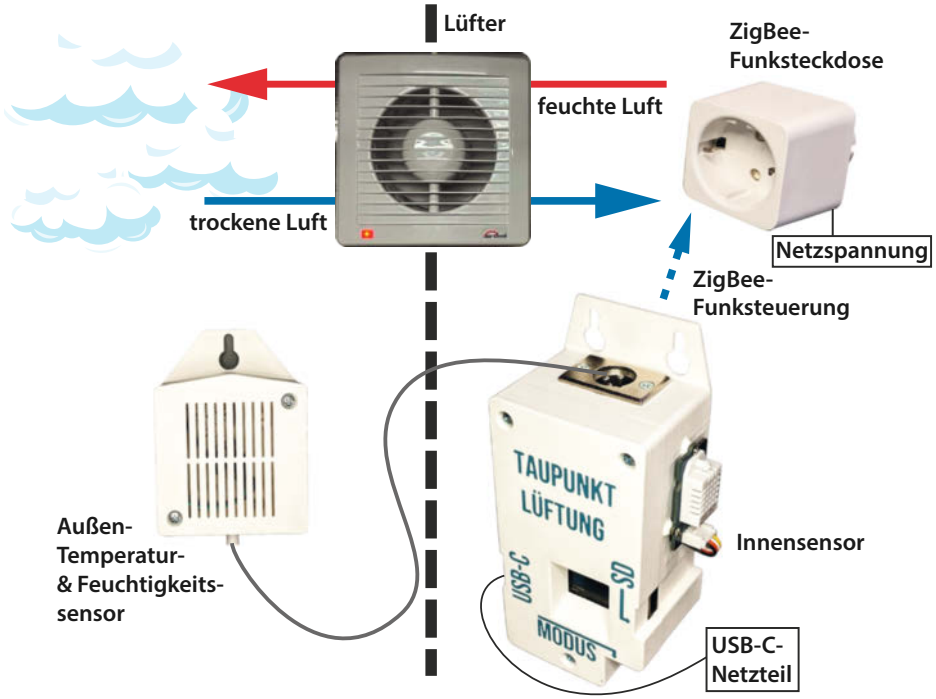


Bild 1: Oben ist der Lüfter an der ZigBee-Steckdose dargestellt, unten die Steuerung mit einem Außensensor und einem Handynetzteil zur Stromversorgung.

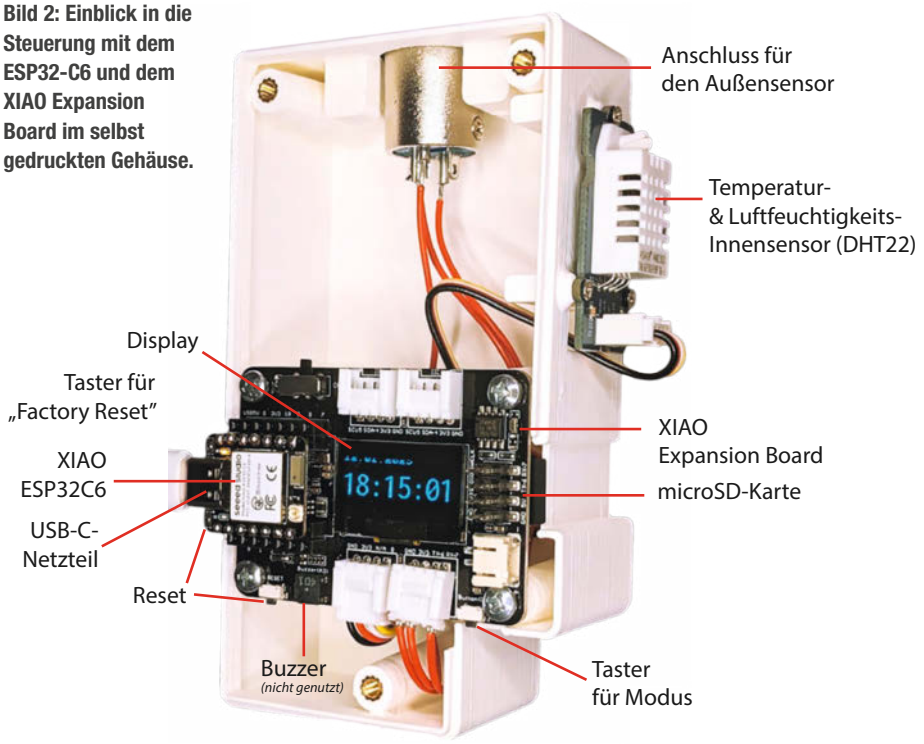


Bild 2: Einblick in die Steuerung mit dem ESP32-C6 und dem XIAO Expansion Board im selbst gedruckten Gehäuse.

DAU-Testing (dümmster anzunehmender User) der Library betrieben und an der Lösung von ein paar Bug-Reports mitgewirkt.

Inzwischen ist die Bibliothek aber relativ stabil und für ein simples An- und Ausschalten einer Steckdose absolut ausreichend. Aktuell wird zwar der Parallelbetrieb von ZigBee und WLAN noch nicht unterstützt, aber das steht ja beim Nachbarn ohnehin nicht zur Verfügung. Der Link zum GitHub-Repository arduino-esp32 sowie zu diesem vollständigen Projekt ist über den Link in der Kurzinfo zu finden.

Die Steuerung meines Taupunkt-Lüfters besteht aus den üblichen und in der Make 1/22 bereits erklärten Abläufen, die ich daher hier nur kurz erläutere: Die beiden Temperatur- und Feuchtigkeitssensoren werden ausgelesen und über jeweils acht Werte gemittelt. Dann berechnet das Programm die Taupunkte der Innen- und Außenluft und vergleicht diese. Wenn der Taupunkt draußen um 5 °C geringer ist als drinnen (und Mindesttemperaturen eingehalten sind), ist Lüften sinnvoll. Befindet sich die Steuerung im Automatikmodus, wird der Lüfter nun für 15 Minuten aktiviert. Anschließend schaltet sich das Gebläse zur Schonung des Motors für 15 Minuten aus. Falls Lüften dann immer noch sinnvoll ist, startet der Zyklus von vorne.

Den Aufruf der ZigBee-Library habe ich in eine Helfer-Klasse ausgelagert: Nach dem Einschalten der Steuerung ist für 180 Sekunden die Kopplung neuer Steckdosen möglich. Prinzipiell können mit einer Steuerung so auch mehrere ZigBee-Steckdosen und damit weitere Lüfter verbunden werden. Technisch verhält sich die Steckdose außerdem ähnlich wie eine Lampe und man könnte auch eine gewöhnliche ZigBee-Lampe koppeln, die als Statusanzeige dient. Das sind aber beides Ideen für zukünftige Erweiterungen, sie sind noch nicht getestet.

Wenn sich der Sollzustand der Lüftungssteuerung ändert, wird dies an alle gekoppelten ZigBee-Endpunkte, also Steckdosen, gesendet. Um sicherzugehen, dass das Funksignal auch ankommt, wiederholt meine Helfer-Klasse den ZigBee-Befehl alle 20 Sekunden.

Auf dem ESP32-C6-Board befindet sich ein Taster, der im normalen Betrieb aufgrund des darüberliegenden 3D-gedruckten Gehäuses nicht zugänglich ist. Öffnet man das Gehäuse und hält den Taster gedrückt, vergisst die ZigBee-Library alle bisher gekoppelten Geräte und startet den Controller anschließend neu. Der Rest des Programms bleibt bei diesem „Factory-Reset" unangetastet. Mein Nachbar wird diese Taste jedoch nicht drücken müssen, weil er die Steuerung mit einer bereits gekoppelten Steckdose überreicht bekommt.

Die auf dem ESP32 verbaute ZigBee-Antenne hat in meiner Wohnung fünf Meter durch einen Flur überbrückt. Aktuell begrenzt also eher der kabelgebundene Außensensor die Reichweite des Systems. Vielleicht möchte jemand die Steuerung um einen kabellosen ZigBee-Außensensor erweitern?

Anzeige und Datenlogging

Die Messwerte und das Ergebnis des Taupunkt-Vergleiches werden auf dem Display angezeigt (Bild 3). Außerdem gibt es noch weitere Informationen aus: den durch einen Taster gewählten Modus (Aus, Automatik und An), die Uhrzeit sowie den Status von ZigBee und der microSD-Karte. Da der übliche Weg, die aktuelle Uhrzeit via WLAN aus dem Internet zu beziehen, nicht zur Verfügung steht, wird die Echtzeituhr (RTC) mit dem Datum und der Uhrzeit des Kompilierens des Programms abgeglichen.

Beim Start des Arduino überprüft das Programm, ob die Echtzeituhr bereits eine dem Format nach gültige Zeit gespeichert hat. Gleichzeitig wird die im Code hinterlegte Uhrzeit des Kompilierens ausgelesen. Falls beide Zeiten gültig sind, wird geprüft, welche neuer ist: Ist die Kompilationszeit aktueller, wird die RTC damit aktualisiert. Ist die RTC-Zeit aktueller oder gleich, bleibt sie unverändert.

Schließlich wird die Zeit in die interne, stromabhängige Systemuhr des ESP32 übernommen, sodass der Mikrocontroller eine korrekte Zeitbasis hat. Danach läuft die RTC des Erweiterungsboards dank ihrer Batterie eigenständig weiter – auch wenn der Mikro-

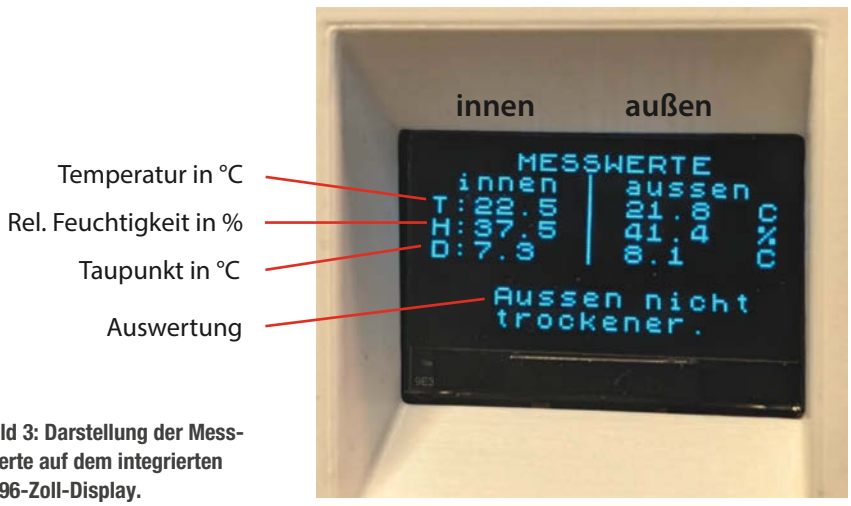


Bild 3: Darstellung der Messwerte auf dem integrierten 0,96-Zoll-Display.

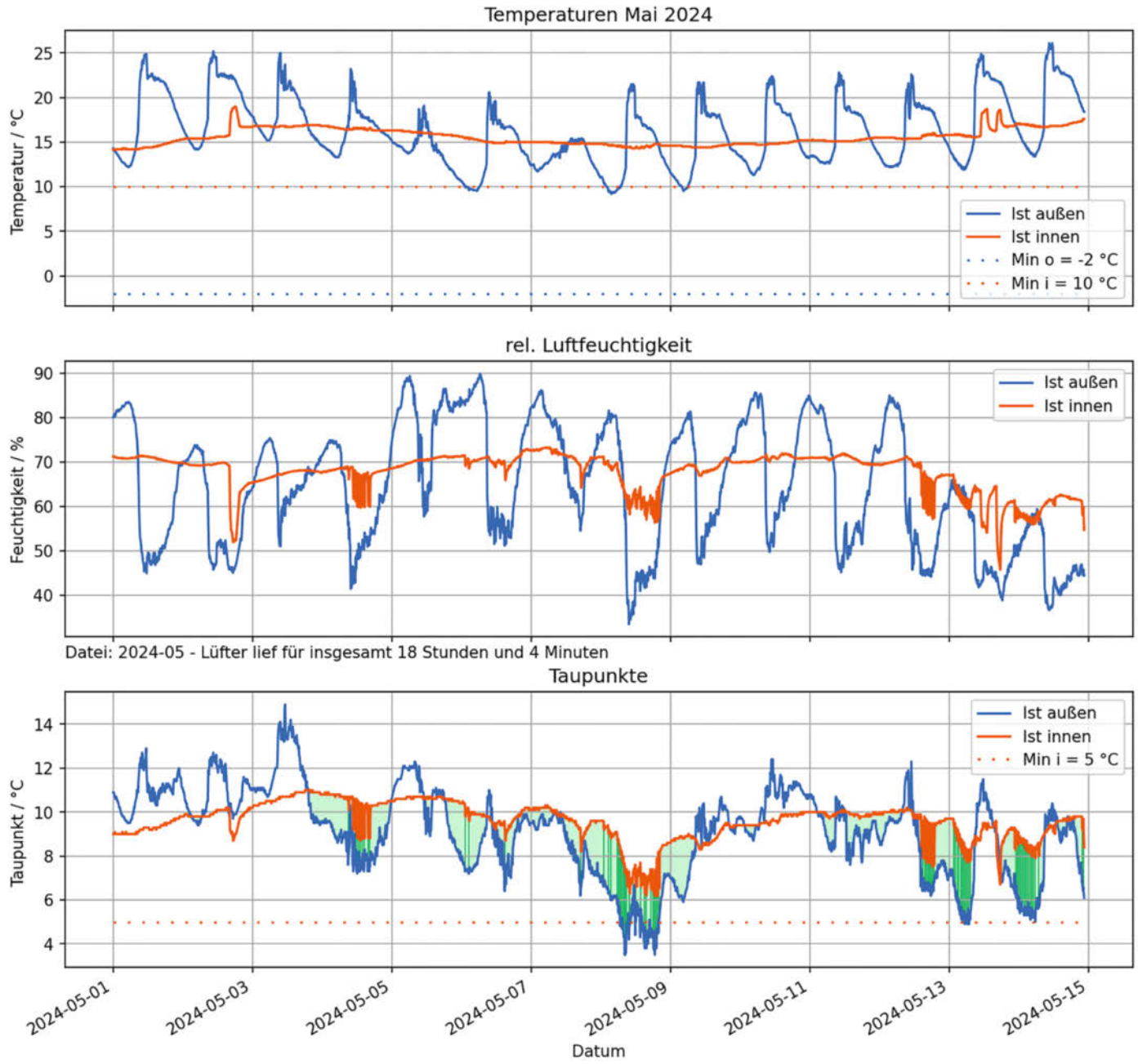


Bild 4: Visualisierung der Messdaten für Temperatur, relative Feuchtigkeit und Taupunkt. Außensensor in Blau, Innensensor in Rot.

controller neu gestartet wird. Falls die RTC aber irgendwann keinen Strom mehr bekommt (z. B. wenn die Knopfzellen-Batterie leer ist), verliert sie die gespeicherte Zeit und muss neu gesetzt werden.

Wenn eine microSD-Karte eingelegt ist, werden alle sechs Minuten die aktuellen Messwerte und der Status aufgezeichnet. Es wird eine CSV-Datei pro Monat geschrieben. Wenn der Nachbar die SD-Karte vorbeibringt, können wir die Auswertung gemeinsam bei einem Kaffee besprechen.

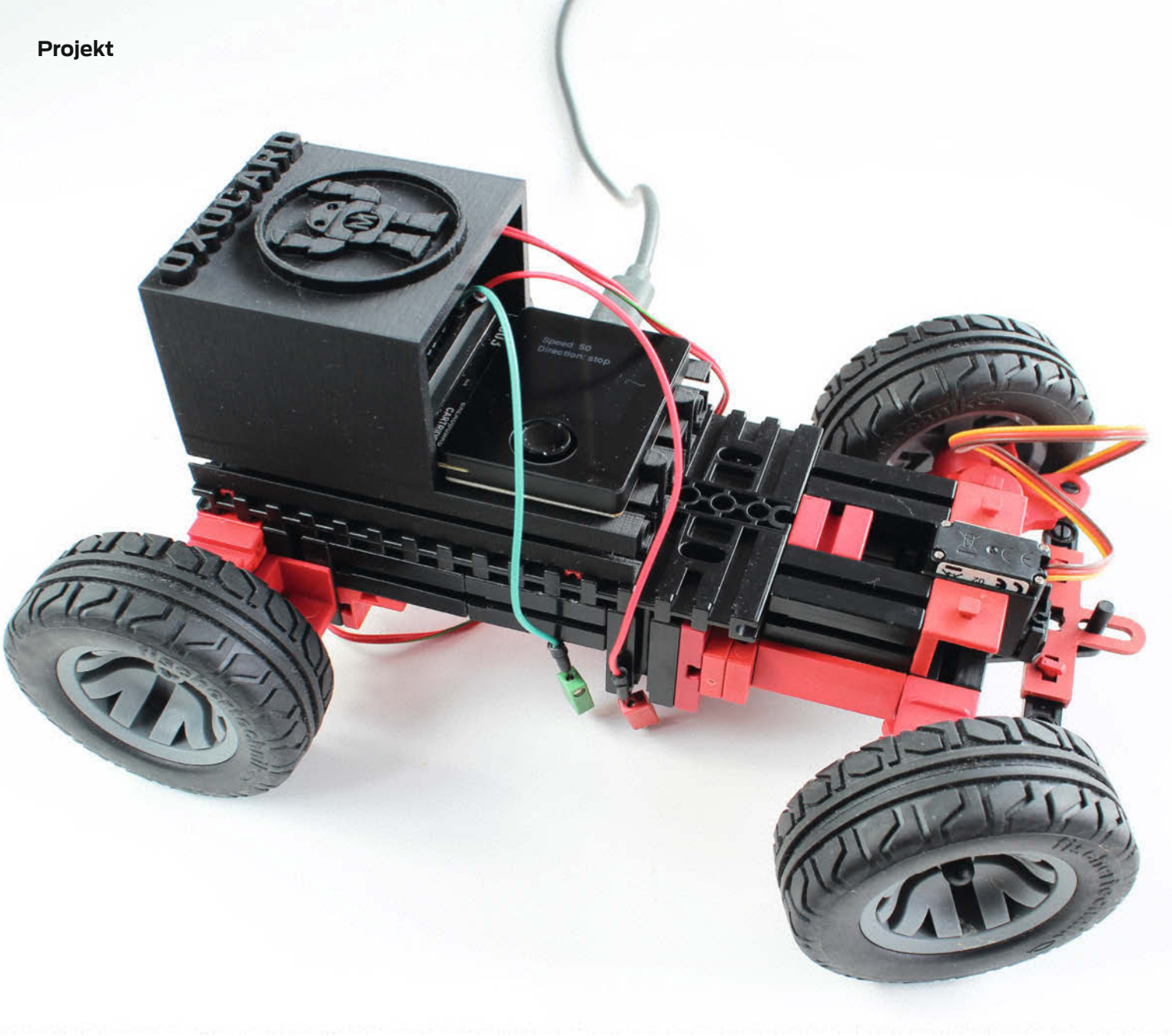
Visualisierung mit Python

Die Datei wird mithilfe eines selbst geschriebenen Python-Programms visualisiert, das im verlinkten Repository zur Verfügung steht. Die Auswertung ist in Bild 4 mit Daten von Mai 2024 dargestellt, da das Lüften da häufiger sinnvoll ist, als im kalten Januar.

In der ersten Zeile ist die Lufttemperatur dargestellt. Die hohen Mittagstemperaturen an den Mai-Tagen (blaue Linie) unterscheiden sich deutlich von den gleichmäßigen Kellertemperaturen (rote Linie). Die zweite Zeile zeigt die relative Luftfeuchtigkeit und in der dritten Zeile ist der Taupunkt dargestellt. Sehr praktisch ist es, die Fläche zwischen den Taupunkt-Kurven hellgrün zu färben, wenn der Taupunkt außen geringer ist als drinnen. Wenn die Steuerung tatsächlich den Lüfter startet, wird die Fläche zwischen den Kurven dunkelgrün eingefärbt.

Gehäuse für die Steuerung und den Außensensor

Das Gehäuse für den Sensor habe ich mit meinem 3D-Drucker gefertigt. Durch zwei manuelle Filamentwechsel während des Drucks wurde die Beschriftung auch ohne automatischen Materialwechsler realisiert. Mithilfe von M3-Einschmelz-Gewinden kann das Gehäuse einfach geöffnet und geschlossen werden. Ösen zur Aufhängung machen die Montage an der Wand einfach. Die Druckdateien stehen bei Printables zur Verfügung (siehe Link in der Kurzinfo). Das Gehäuse bietet sich auch für andere ZigBee-Projekte als Ausgangspunkt an, bei denen das gleiche Modul und Expansion Board eingesetzt wird. *—jom*



Fischertechnik mit der Oxocard steuern

Fischertechnik hat mit den „Maker-Kits“ eine Produktlinie eingeführt, deren offenes System darauf baut, Basismodelle durch individuelle Ergänzungen zu bereichern. Über die Oxocard lässt sich mit wenig Aufwand eine leistungsstarke Steuerung für die Maker-Kits und andere Fischertechnik-Modelle entwickeln.

von Dirk Fox

Anders als bei Standard-Baukästen von Fischertechnik gehört es hier zum Konzept, das Modell durch Fremdsysteme funktionell zu ergänzen.

2024 hat Fischertechnik drei „Maker Kits“ herausgebracht (Bild 1), die mit einem Mikrocontroller und Sensoren nach Wahl erweitert und gesteuert werden können: ein Fahrzeugchassis mit Antriebsmotor und Lenkservo („Car“), eines mit Mecanum-Rädern und vier Antriebsmotoren („Omniwheels“) und eines mit vier „Beinen“ aus je zwei Servos („Bionic“). Was dieser Basistechnik fehlt, ist eine Steuerung, beispielsweise durch einen programmierbaren Mikrocontroller. Genau hier kommt die Oxocard ins Spiel.

Mit der Oxocard alleine lassen sich solche Maker-Modelle nicht ansteuern, da die Motoren von Fischertechnik (Abb. 3) eine 9-V-Stromversorgung benötigen und unter Last bis zu 1 A Strom ziehen – das übersteigt die Leistungsgrenzen der Oxocard deutlich.

Für die Oxocard gibt es bisher noch keine Cartridge mit Motortreibern, die wie das Motor Shield für den Arduino die Versorgung der 9-V-Motoren über eine externe Spannungsquelle übernehmen und sie über H-Brücken ansteuern. Doch es gibt eine Lösung: Die Motortreiber kann man auf dem Breadboard der Oxocard Connect mit H-Brücken

Kurzinfo

» Auswertung der Signale des Fischertechnik-Encoder-Motors
» Erweiterung der Oxocard Connect um Motortreiber
» 9-V-Motoren mit Motortreibern steuern

Checkliste

Zeitaufwand:
2 Stunden

Kosten:
165 Euro (inkl. Oxocard Connect Innovators Kit)

Material

» **Oxocard Connect Innovators Kit (Make Edition)**
» **USB-Powerbank**
» **Breakout-Board (TB6612 oder DRV8866)**
» **Fischertechnik-Maker-Kit „Car“**
» **Fischertechnik-Akkuset**
» **Filament**

Werkzeug

» **Lötkolben**
» **Litze**
» **Drahtbrücken**
» **3D-Drucker**

Mehr zum Thema

» Roman Radtke, Universeller Schubvektorantrieb, Make Magazin 1/19, S. 62
» Ákos Fodor und andere, Bausätze für Maker, Make Magazin 7/23, S. 22
» Daniel Bachfeld, Einführung in Oxocard Connect: Experimentierset für Maker, Make Magazin 5/24, S. 38

Alles zum Artikel im Web unter *make-magazin.de/xbw3*

Oxocard Connect auf einen Blick

Die vom Berner Unternehmen Oxon entwickelte Oxocard Connect ist ein ESP32 in einem ansprechenden, knapp 5 × 4 cm großen Gehäuse mit einem 240 × 240-Pixel-Display, einem Joystick, 2 MByte PSRAM, 8 MByte Flash, WLAN und Bluetooth. Die Firmware beherrscht die Protokolle HTTP(S), MQTT, I2C und SPI. Die Programmierung der Oxocard erfolgt in Blockly oder NanoPy, einer an Micropython angelehnten und einfach zu erlernenden Skriptingsprache, die vom Hersteller Oxon entwickelt wurde. Die NanoPy-Entwicklungsumgebung wird im Browser gestartet und steht in Deutsch, Englisch und Französisch zur Verfügung. Sie umfasst Beispielprogramme, ein Tutorial und einen komfortablen Debugger. Eine USB-Verbindung der Oxocard mit dem Computer ist für die Programmierung nicht erforderlich: Der Mikrocontroller kann via WLAN oder USB mit dem Computer verbunden werden.

Über die Breadboard Cartridge (Bild 2) kann man die Oxocard Connect mit zahlreichen handelsüblichen Sensoren verbinden: entweder über die fünf digitalen und zwei analogen IO-Ports oder über den I2C- oder SPI-Bus. Damit lässt sich die Oxocard leicht mit Breakout-Boards um beispielsweise Kompass-, Beschleunigungs-, Neigungs- oder GPS-Sensoren erweitern. Die Oxocard arbeitet mit 3,3-V-Logik; damit kommen die meisten I2C- und SPI-Sensor-Boards zurecht.

Einfache Schaltungen ohne leistungsstarke Aktoren können daher direkt auf dem kleinen integrierten Breadboard aufgebaut

Bild 2: Die Oxocard Connect (rechts) mit angesteckter Breadboard Cartridge (links).

und ohne externe Spannungsquelle über die Oxocard mit 3,3 V (VDD) oder 5 V versorgt werden. Die 5-V-Versorgung der Oxocard übernimmt eine USB-(C-)Verbindung beispielsweise zu einer Powerbank. Die Make-Edition der Oxocard Connect kann im heise-Shop im Bundle mit einer Breadboard Cartridge sowie unterschiedlichen Sensoren, LEDs, einem Mikroservo, Kabeln und Widerständen als „Oxocard Connect Innovators Kit“ für knapp 90 € erworben werden.

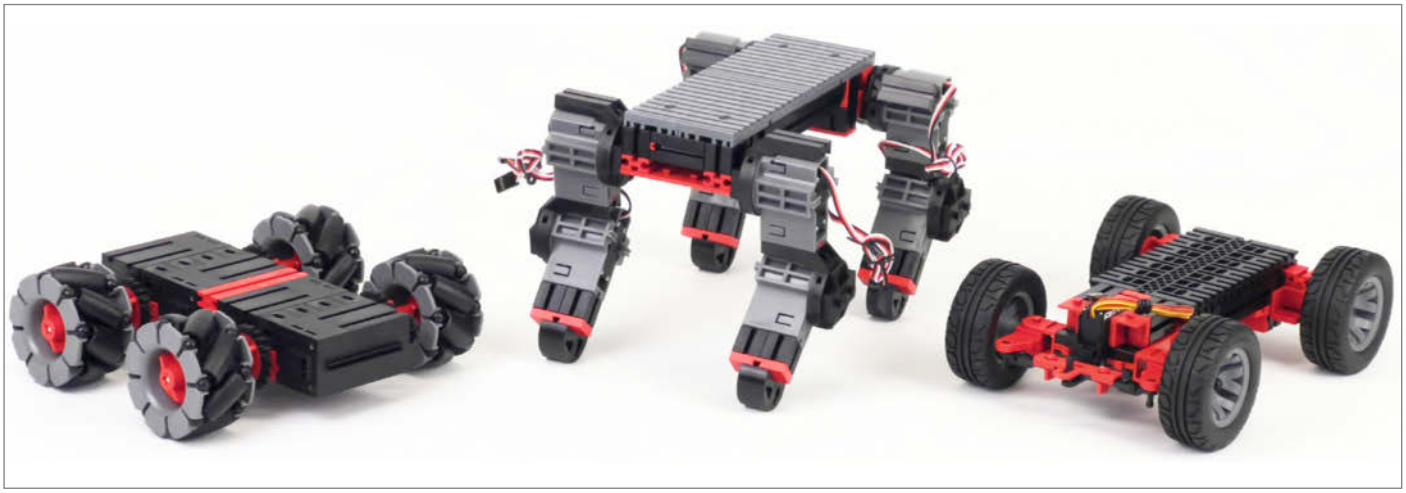


Bild 1: Zur Fischertechnik „Maker Kit“-Reihe gehören aktuell die Modelle „Omniwheels“, „Bionic“ und „Car“ (v. l. n. r.).

aufbauen. Bevor man allerdings die Schaltung selbst entwirft: Es gibt sehr günstige fertige Breakout-Boards, die bereits ICs für die Motorsteuerung enthalten und sich auf das Breadboard aufstecken lassen.

Motortreiber für die Oxocard

Für die 9-V-Fischertechnik-Motoren mit einer Leistungsaufnahme von bis zu 1 A bieten sich als Motortreiber beispielsweise der DRV8833 von Texas Instruments (bis 10,8 V Eingangsspannung und einer Leistungsaufnahme von maximal 1,5 A je Motor; Abb. 4a) oder der TB6612FNG von Toshiba (bis 13,5 V und 1,2 A je Motor; Abb. 4) an. Für beide ICs gibt es von verschiedenen Anbietern wie Adafruit, Pololu oder Sparkfun fertige Breakout-Boards. Damit wird die Erweiterung der Oxocard Connect um Motortreiber zum Kinderspiel: Breakout-Board bestellen, Stiftleisten anlöten und auf die Breadboard Cartridge aufstecken – (fast) fertig.

Mit diesen Breakout-Boards können über die Anschlüsse AOUT und BOUT (DRV8833) bzw. MOT.A und MOT.B (TB6612) je zwei Gleichstrommotoren von Fischertechnik (oder ein Schrittmotor) gesteuert werden. Die Bewegungsrichtung der Motoren wird jeweils über die Pins AIN1/AIN2 und BIN1/BIN2 festgelegt: An je einem der Pins wird 0, am anderen 1 angelegt.

Bei der Geschwindigkeitssteuerung unterscheiden sich die beiden ICs: Beim DRV8833 wird statt einer logischen 1 das PWM-Signal an jeweils einen der Eingangspins übertragen. Der TB6612 verfügt über einen weiteren Anschluss je Motor (PWMA bzw. PWMB), über den das PWM-Signal eingespeist wird. Vorteil des DRV8833: Ein Motor benötigt nur zwei GPIO-Ports für die Ansteuerung, beim TB6612 sind es drei.

Im Folgenden stellen wir zunächst die Ansteuerung von Fischertechnik-Motoren mit dem Breakout-Board von Adafruit für den TB6612 vor; sie funktioniert mit den TB6612 Breakout-Boards anderer Hersteller genauso.

Beschaltung des Adafruit-Breakout-Boards

Abbildung 5 zeigt die Beschaltung des Adafruit-TB6612-Boards auf der Breadboard Cartridge für die Ansteuerung eines Fischertechnik-9-V-Motors. Als externe 9-V-Stromversorgung für die Motoren verwenden wir einen Fischertechnik-Akku, der mit 1800 kWh einen längeren Spielspaß verspricht. Das Anschlusskabel für den Akku wird an einem Ende mit Fischertechnik-Steckern versehen, am anderen an die Board-Pins VMotor angeschlossen (grüne Anschlussklemme in Abb. 6; rotes Kabel an Plus, grünes Kabel an Minus).

Bevor das Kabel mit der Stromversorgung verbunden wird, sind die folgenden Ausgänge der Cartridge mit den Pins des Breakout-Boards zu verbinden:

- Die Board-Logik (VCC, GND) wird über die Cartridge-Pins 5 V und GND versorgt. **Achtung**: Der Anschluss „VM“ ist tabu – da liegen die 9 V des Fischertechnik-Akkus an; mit dem Breadboard der Oxocard dürfen sie auf keinen Fall verbunden werden.
- Der Fischertechnik-Motor wird mit den MOT.A-Ausgangspins verbunden. Die bei-

Bild 3: Die klassischen Fischertechnik-9-V-Motoren (v. l. n. r.): XS (bis 0,2 A), Mini (bis 0,4 A), XM (bis 0,9 A), Encoder (bis 0,5 A).

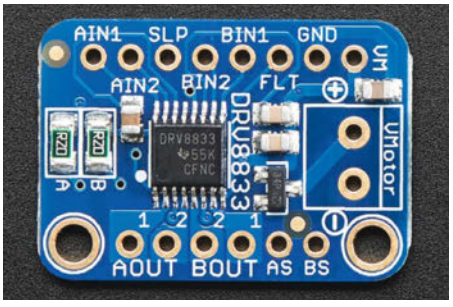


Bild 4: : Zum Ansteuern der Motoren benötigt die Oxocard eine sogenannte H-Brücke, hier der DRV8833.

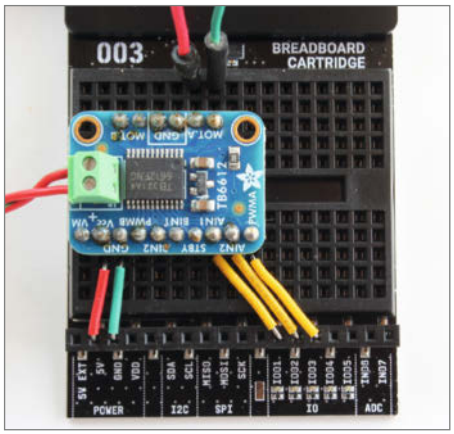


Bild 5: Die Beschaltung des Adafruit-Breakout-Boards TB6612 auf der Breadboard Cartridge ist unkompliziert und übersichtlich.

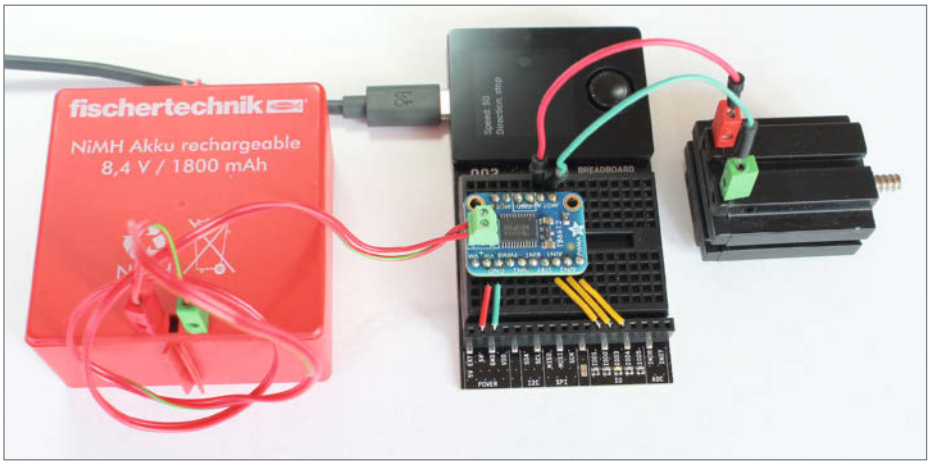


Bild 6: Ansteuerung eines Fischertechnik-Motors über die Oxocard mit einem TB6612 und Fischertechnik-Akku zur Stromversorgung.

den Kabel werden an einem Ende ebenfalls mit Fischertechnik-Steckern versehen.
- Die Eingangspins AIN1 und AIN2 des Boards werden mit den digitalen GPIO-Ports IO01 und IO02 der Cartridge verbunden.
- Das PWM-Signal (Board-Anschluss PWMA) wird über den GPIO-Port IO03 eingespeist.

Jetzt lassen sich der Fischertechnik-Akku und die Stromversorgung der Oxocard (Powerbank mit USB-C-Kabel) verbinden. Der gesamte (Versuchs-)Aufbau mit einem Fischertechnik-Mini-Motor ist in Abb. 6 zu sehen; er funktioniert mit allen anderen Fischertechnik-Motoren genauso.

Skriptgesteuerte Motoren

Nun fehlt noch das Skript zur Steuerung des Motors. Wer schon einmal einen Arduino programmiert hat, dem gelingt das in NanoPy schnell: Zunächst müssen die IO-Ports initialisiert (initGPIO) und die PWM-Frequenz eingestellt werden (`setPWMFrequency`).

Die Drehrichtung des Motors (links/rechts) und die Geschwindigkeit (höher/niedriger) soll über den eingebauten Joystick der Oxocard einstellbar sein. Wird der Joystick gedrückt, soll der Motor stoppen. Die gewählte Motorgeschwindigkeit und die Drehrichtung sollen alle 250 ms auf dem Display ausgegeben werden. Das machen wir mit der Ereignisprozedur `onClick()`.

Das Ergebnis zeigt das Listing. Die Anfangsgeschwindigkeit des Motors (`speed`) wird auf 50 gesetzt, die Schrittweite (`STEP`) beim Drücken des Joysticks auf 10. Die Funktion `map()`, die einige Leser sicher vom Arduino kennen, bildet den Geschwindigkeitswert (0-100) auf den Wertebereich des PWM-Signals an IN07 (0-4096) ab. Die Ereignisfunktionen steuern die Ausgabe auf dem Display (`onDraw()`) und die Abfrage des Joysticks (`onClick()`).

Das Prinzip der Motor-Ansteuerung sollte damit deutlich werden. Das Skript bietet Raum für eigene Experimente: So kann beispielsweise der Wert der Geschwindigkeitsänderung bei der Betätigung des Joysticks (`STEP`) angepasst werden (im Beispiel: 10 %) oder der `MIN_DUTY_CYCLE` auf den kleinsten Wert gesetzt werden, ab dem der Motor sich in Drehung versetzt.

Zwar bietet das Adafruit-Breakout-Board die Möglichkeit zum Anschluss eines zweiten

DC-Motor mit Oxocard und DRV8833

```
const AIN1 = C_PIN_01       # AIN1 Pin connected to IO01
const AIN2 = C_PIN_02       # AIN2 Pin connected to IO02
const MIN_DUTY_CYCLE = 0
const MAX_DUTY_CYCLE = 4095

initGPIO(AIN1, OUTPUT)
initGPIO(AIN2, OUTPUT)
setPWMFrequency(1000)       # default: 500 Hz

speed = 50                  # Geschwindigkeit: 0..100 (Startwert: 50)
direction = "stop"          # Bewegungsrichtung: "left", "right", "stop"
const STEP = 10

background(0,0,0)
textFont(FONT_ROBOTO_24)

def onDraw():
    dutyCycle = map(speed, 0, 100, MIN_DUTY_CYCLE, MAX_DUTY_CYCLE)
    if isEqual(direction, "right"):
        writePWM(AIN1, 0)   # OFF
        writePWM(AIN2, dutyCycle)
    elif isEqual(direction, "left"):
        writePWM(AIN2, 0)   # OFF
        writePWM(AIN1, dutyCycle)
    elif isEqual(direction, "stop"):
        writePWM(AIN1, 0)   # OFF
        writePWM(AIN2, 0)   # OFF
    clear()                 # output speed and direction on display
    drawText(10, 10, "Speed: " + speed)
    drawText(10, 40, "Direction: " + direction)
    update()
    delay(50)

def onClick():
    b = getButton()         # direction and speed control with joystick
    if b == 2:              # up: faster
        speed = min(speed + STEP, 100)
    elif b == 4:            # down: slower
        speed = max(speed - STEP, 0)
    elif b == 3:            # right: turn motor clockwise
        direction = "right"
    elif b == 5:            # left: turn motor anticlockwise
        direction = "left"
    elif b == 1:            # middle: turn motor off
        direction = "stop"
```

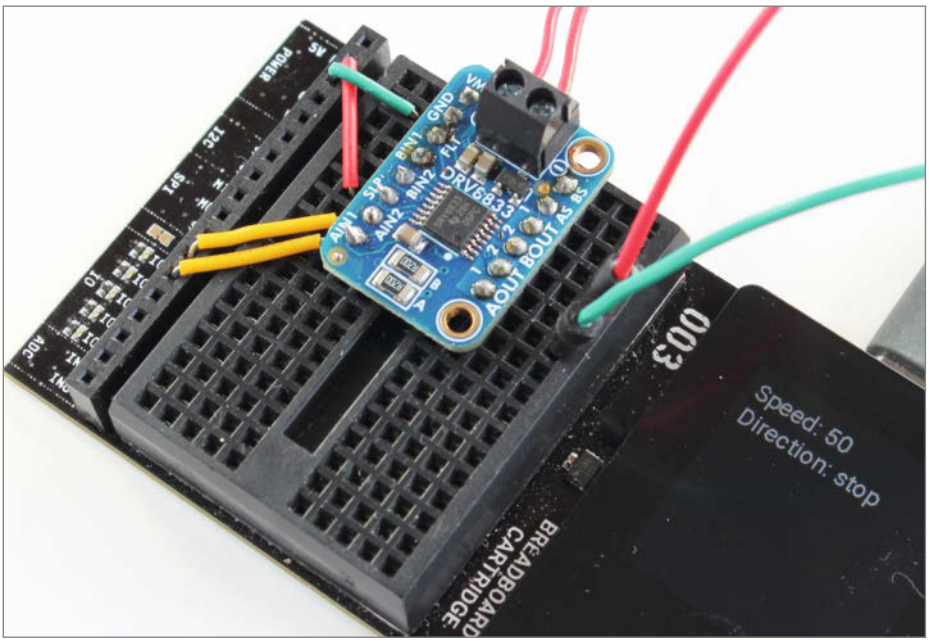


Bild 7: Die Beschaltung des Adafruit-Breakout-Boards DRV8833 auf der Breadboard Cartridge auf einen Blick.

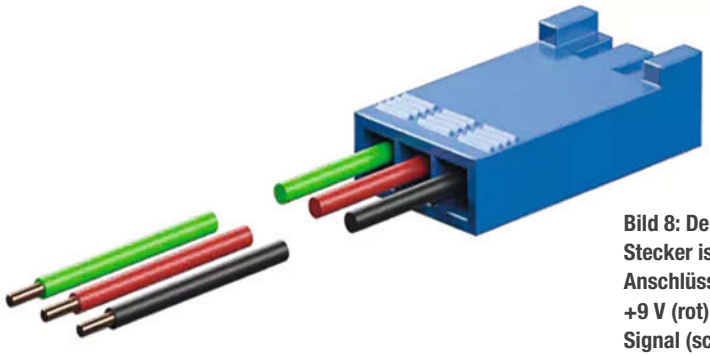


Bild 8: Der verpolungssichere Stecker ist mit den Encoder-Anschlüssen für GND (grün), +9 V (rot) und dem Encoder-Signal (schwarz) belegt.

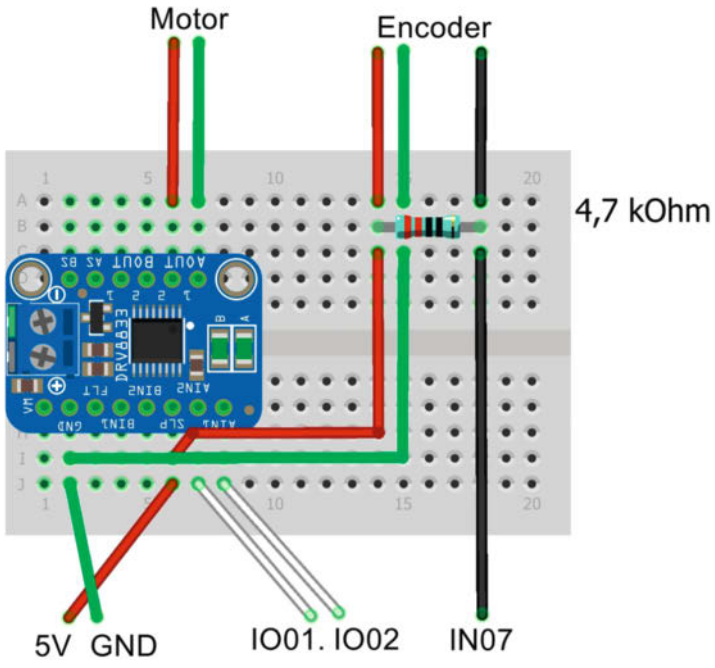


Bild 9: Der Anschluss des Encoders am DRV8833 auf dem Breadboard sowie die Anschlüsse an Motor, Spannung (5V) und GND auf einen Blick.

Motors (MOT.B-Anschlüsse), doch sind nur noch zwei GPIO-Ports (IO04 und IO05) dafür frei. Darüber können wir den zweiten Motor nach Verbindung der beiden GPIO-Ports mit den Pins BIN1 und BIN2 ein- und ausschalten sowie die Drehrichtung ändern. Für den PWMB-Pin fehlt uns jedoch ein sechster Output-Port. Bestenfalls können wir noch den PWM-Ausgang an Port IO03 mit beiden PWM-Pins (PWMA und PWMB) verbinden und darüber an beiden Motoren dieselbe Geschwindigkeit einstellen. Die entsprechende Erweiterung des NanoPy-Skripts ist nicht schwer.

Verwendet man den DRV8833 statt des TB6612, genügen uns insgesamt vier GPIO-Ports, um zwei Motoren mit einem PWM-Signal zu steuern. Für die Ansteuerung der Fischertechnik-Motoren sind nur kleine Änderungen in der Beschaltung und am NanoPy-Skript erforderlich. Der Anschluss des Fischertechnik-Akkukabels an der Anschlussklemme erfolgt wie beim TB6612.

Bevor das Breakout-Board und die Oxocard mit einer Stromversorgung verbunden werden, nehmen wir die folgende Beschaltung der Cartridge vor:

- Zur Versorgung der Board-Logik wird der Cartridge-Pin GND mit dem GND-Pin des Breakout-Boards und der 5-V-Anschluss der Cartridge mit dem Pin SLP verbunden. Auch hier ist der Anschluss „VM“ tabu – da liegen die 9 V des Fischertechnik-Akkus an; mit dem Breadboard der Oxocard dürfen sie auf keinen Fall verbunden werden.
- Der Fischertechnik-Motor wird mit den AOUT-Ausgangspins verbunden. Auch hier werden die beiden Kabel an einem Ende mit Fischertechnik-Steckern versehen.
- Die Eingangspins AIN1 und AIN2 des Boards werden mit den digitalen GPIO-Ports IO01 und IO02 der Cartridge verbunden.

Bild 10 zeigt die durch den Wegfall des separaten PWM-Pins im Vergleich mit dem TB6612 einfachere Beschaltung des DRV8833 Breakout-Boards von Adafruit.

Das Basis-Skript müssen wir nun noch so umstellen, dass das PWM-Signal abhängig von der Drehrichtung des Motors an den GPIO-Pin angelegt wird, an dem wir beim TB6612 einfach das HIGH-Signal angelegt haben.

Ansteuerung des Encoder-Motors

Der Motor von Fischertechnik ist mit einem Magnet-Encoder ausgestattet, der nach Werksangaben je Umdrehung der Motorachse 190/3 (also etwa 63,3) Impulse liefert. Anhand der Anzahl der Impulse kann man den Motor um einen definierten Winkel drehen und bezogen auf die Räder um eine definierte Strecke fahren lassen.

In der „Fischertechnik-Welt“ werden Encoder und Motor beide mit 9 V versorgt; die Im-

pulse liefert der Encoder in 9-V-Logik. Die Belegung der Kontakte zeigt Abb. 8.

In der Oxocard-Welt und auf dem Breadboard versorgen wir den Encoder mit 5 V, was ohne Probleme funktioniert. Wenn wir die Impulse des Encoders mit der 3,3-V-Logik des Boards verarbeiten wollen, müssen wir allerdings zwischen der 5-V-Stromversorgung und dem Encoder-Signal einen 4,7-kΩ-Pullup-Widerstand schalten, der die Stromstärke auf zirka 1 mA begrenzt. Das schützt die 5-V-tolerante Logik. Alternativ könnte man auch einen Spannungsteiler einbauen, der die Spannung auf 3,3 V bringt. Den Encoder-Ausgang verbinden wir dann mit dem analogen Eingangs-Pin IN07 der Oxocard (Abb. 9).

Beim Adafruit-Breakout-Board für den TB6612 erfolgt der Anschluss des Encoders in gleicher Weise. Für die Auswertung der Encoder-Impulse in NanoPy hat Oxon drei zusätzliche Funktionen implementiert:

```
initPulseCounter()
resetPulseCounter()
readPulseCounter()
```

Mit diesen Funktionen ist es möglich, die Oxocard Impulse zählen zu lassen. Dazu müssen

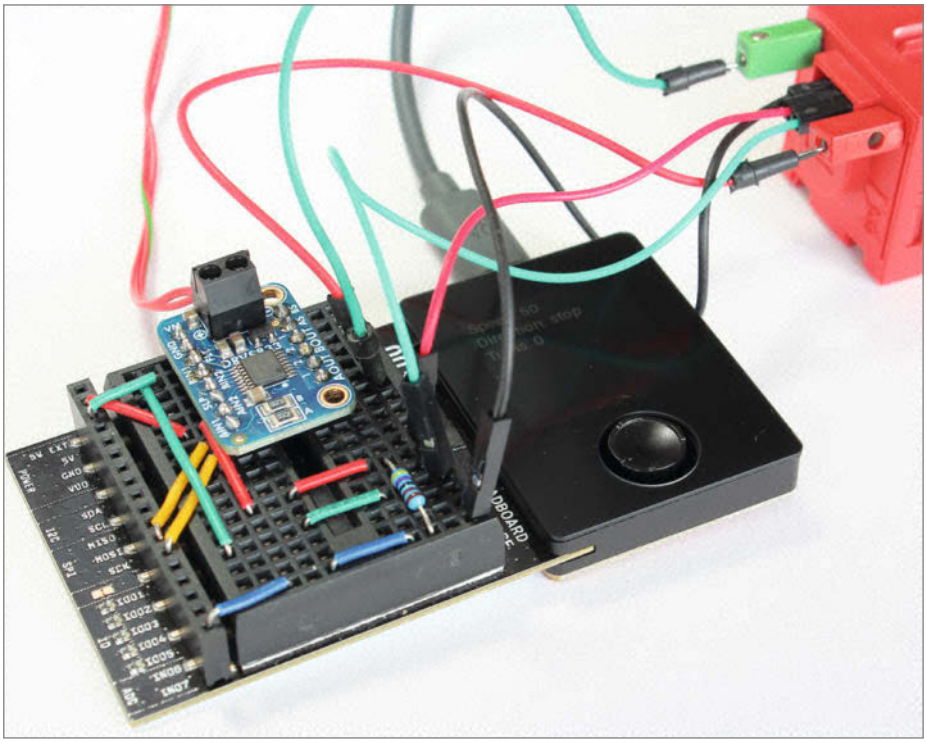


Bild 10: Der komplette Schaltungsaufbau mit DRV8833 und Fischertechnik-Encoder-Motor.

wir zunächst den Analog-Pin IN07, an dem die Impulse anliegen, als Impulszähler initialisieren (`initPulseCounter()`). Beim Start des Zählvorgangs wird der Impulszähler auf null zurückgesetzt (`resetPulseCounter()`) und kann dann jederzeit im Programmverlauf ausgelesen werden (`readPulseCounter()`).

Die Nutzung dieser drei Funktionen in NanoPy kann man sich im „Beispielprogramm 17" des Innovators Kits direkt in der Entwicklungsumgebung am Beispiel eines Taster anschauen. Im NanoPy-Code bestimmen wir anhand der Impulse die Anzahl der Motorumdrehungen seit der letzten Betätigung des Joysticks und lassen das Ergebnis auf dem Display anzeigen. Damit es bei maximaler Motorgeschwindigkeit nicht zu „Sprüngen" in der Anzeige kommt, müssen wir das Aktualisierungsintervall auf 50 ms verringern.

Das entsprechend erweiterte Skript für den DRV8833 zeigt das Listing. Die Anpassung des Skripts an ein Breakout-Board für den TB6612 sollte auch Einsteigern keine Schwierigkeiten bereiten (Listings im Downloadbereich), genauso wie die Erweiterung für die Steuerung eines zweiten Motors. Den zweiten Encoder können wir am Analog-Pin IN06 auswerten.

Fischertechnik-Oxocard-Case

Damit die Oxocard auch einen passenden Platz findet, kann man das im Aufmacherbild zu sehende Case verwenden. Es bietet Raum für die Oxocard mit angestecktem Breadboard und lässt sich unauffällig in Modelle integrieren, wobei das Display frei sichtbar bleibt. Das Case verfügt sowohl über drei zur Fischertechnik-Welt passende Halteschienen als auch über sechs Klemmbaustein-kompatible Befestigungspunkte. Im hinteren Bereich ist Platz für Kabeldurchführungen sowie für fünf Standard-LEDs (5 mm). Das Case und die Erweiterungsmöglichkeiten sind Bestandteil zukünftiger Modellbeschreibungen, die auf der im Beitrag beschriebenen Antriebssteuerung aufbauen. Die STL-Datei zum 3D-Druck ist im Downloadbereich abrufbar.

So geht es weiter

Mit der Erweiterung der Oxocard Connect und dem Breakout-Board für den Motortreiber-IC TB6612 oder den DRV8866 können wir nun bis zu zwei Motoren aus NanoPy ansteuern. Auch die Impulse eines Fischertechnik-Encoder-Motors können wir auswerten und darüber die Umdrehung der Motorachse in rund 5,7°-Schritten steuern.

Im nächsten Teil zeigen wir, wie man mit ein paar ergänzenden Sensoren sowie der Oxocard als steuerndem Element ein teilautonomes Fahrzeug bauen kann. *—usz*

Encoder-Motor mit Oxocard und DRV8833

```
const AIN1 = C_PIN_01       # AIN1 Pin connected to IO01
const AIN2 = C_PIN_02       # AIN2 Pin connected to IO02
const ENCODER = C_PIN_07    # Motor encoder on IN07
const MIN_DUTY_CYCLE = 0
const MAX_DUTY_CYCLE = 4095

initGPIO(AIN1, OUTPUT)
initGPIO(AIN2, OUTPUT)
initPulseCounter(ENCODER)
setPWMFrequency(1000)       # default: 500 Hz
setPrecision(2)

speed = 50                  # Geschwindigkeit: 0..100 (Startwert: 50)
direction = "stop"          # Bewegungsrichtung: "left", "right", "stop"
const STEP = 10

background(0,0,0)
textFont(FONT_ROBOTO_24)

def onDraw():
    dutyCycle = map(speed, 0, 100, MIN_DUTY_CYCLE, MAX_DUTY_CYCLE)
    if isEqual(direction, "right"):
        writePWM(AIN1, 0)   # OFF
        writePWM(AIN2, dutyCycle)
    elif isEqual(direction, "left"):
        writePWM(AIN2, 0)   # OFF
        writePWM(AIN1, dutyCycle)
    elif isEqual(direction, "stop"):
        writePWM(AIN1, 0)   # OFF
        writePWM(AIN2, 0)   # OFF
    impulses = readPulseCounter(ENCODER)
    clear()                 # output speed and direction on display
    drawText(10, 10, "Speed: " + speed)
    drawText(10, 40, "Direction: " + direction)
    drawText(10, 70, "Turns: " + impulses/63.33)
    update()
    delay(50)

def onClick():
    b = getButton()         # direction and speed control with joystick
    if b == 2:              # up: faster
        speed = min(speed + STEP, 100)
    elif b == 4:            # down: slower
        speed = max(speed - STEP, 0)
    elif b == 3:            # right: turn motor clockwise
        direction = "right"
        resetPulseCounter(ENCODER)
    elif b == 5:            # left: turn motor anticlockwise
        direction = "left"
        resetPulseCounter(ENCODER)
    elif b == 1:            # middle: turn motor off
        direction = "stop"
        resetPulseCounter(ENCODER)
```

Know-how: H-Brücken

Eine H-Brücke ist eine Schaltung zur Steuerung von Gleichstrommotoren, die es ermöglicht, die Drehrichtung und Geschwindigkeit zu regeln. Sie besteht aus vier Schaltern, meist Transistoren, die in einer „H"-förmigen Anordnung verbunden sind, wobei der Motor die Querlinie des „H" bildet. Durch gezieltes Schließen bestimmter Schalterpaare kann der Stromfluss durch den Motor gesteuert werden. Schließt man den oberen linken und unteren rechten Schalter, fließt der Strom in eine Richtung, und der Motor dreht sich vorwärts. Schließt man stattdessen den oberen rechten und unteren linken Schalter, kehrt sich der Stromfluss um, und der Motor dreht sich rückwärts. Öffnet man alle Schalter, bleibt der Motor stehen. Die Geschwindigkeit wird oft durch Pulsweitenmodulation (PWM) geregelt, bei der die Spannung schnell ein- und ausgeschaltet wird, um die Leistung zu variieren. H-Brücken sind in Robotik, Modellbau, Automobilindustrie und Automation weit verbreitet.

Erste Maker Faire 2025 für die Redaktion

Die Maker Faire Ruhr in Dortmund ist gerade vorbei! Am 29. und 30. März 2025 war die DASA Arbeitswelt Ausstellung wieder ein Paradies für Tüftler, Technikbegeisterte und kreative Bastler.

von Daniel Schwabe

Die Redaktion packt aus – wortwörtlich!

Während diese Zeilen geschrieben werden, ist die Maker Faire Ruhr noch drei Wochen entfernt. Aber wenn die nach Inspiration dürstende Leserschaft die neue Make-Ausgabe in den Händen hält, ist das Event leider schon Vergangenheit. Gerade ist die Make-Redaktion mit ganzer Kraft dabei, die Projekte für den Messestand zu sammeln oder zu bauen. Dieses Jahr will die Redaktion mit den kontroversesten Ausstellungsstücken der Messe auftrumpfen. Extra für die Maker in Dortmund hat sich der fast mannshohe Herrscher der Schimpfolinos, bekannt aus Make 5/24, aus seiner Höhle begeben, um die Besucher des Make-Standes höchstpersönlich zu beleidigen. Wir sind gespannt, ob ein „du schmieriger Schimmelpickel" oder der „lästige Rappelgeiger" zu einem Eklat auf der Messe führen wird.

Falls die überdimensionierte Schimpf-Schleuder nicht ausreicht, hat die Redaktion noch ein Ass im Ärmel: der mit Richtungsblinker modifizierte Fahrradhelm aus Make 2/25. Ein spannendes Projekt, das in Foren und Kommentarspalten zu hitzigen Diskussionen ob seiner Legalität im Straßenverkehr geführt hat.

Eventuell wird es aber auch ganz ruhig. Die animatronische Posteule wird sich vieler Streicheleinheiten erwehren müssen, und die portable Elektrowerkstatt sowie deren neuester Ableger werden dem ein oder anderen Maker-Projekt das Leben retten.

Große Augen werden auf jeden Fall alle Besucher machen, wenn es um den Makey:Lab geht – die Hardware von Make für Maker, an der die Make-Redaktion hinter den Kulissen fleißig bastelt. Erste Prototypen konnten bereits letztes Jahr die Maker in Hannover beeindrucken, jetzt sind die Ruhrgebietler dran.

Was die Redaktion aber zu 100 % weiß: Eine Maker Faire ist immer ein Erlebnis, auf dem die besten Fragen gestellt, die besten Gespräche geführt und kiloweise Inspirationen gesammelt werden.

Nach Heilbronn ist vor Dortmund

Die Maker Faire Heilbronn hat bereits am 8. Februar stattgefunden – und war ein voller Erfolg! Die Experimenta war rappelvoll. An 24 Ständen zeigten kreative Köpfe unglaubliche Projekte für rund 700 Gäste, und es wurde ausgiebig gelötet, programmiert und diskutiert. Und wie wir alle wissen, sind die Ruhrpott-Maker genauso einfallsreich und leidenschaftlich!

Noch mehr Maker Faires in 2025

Falls ihr es nicht nach Dortmund geschafft habt oder einfach nicht genug Maker-Faire-Feeling bekommen könnt, gibt es 2025 noch weitere Gelegenheiten, euer Technik-Herz höherschlagen zu lassen. Hier eine kleine Roadmap für euer Maker-Jahr:

Dirk Baierlipp

Auch in Heilbronn wurde ganz genau hingeschaut. Maker sind überall auf der Suche nach Inspiration.

- Maker Faire Wuppertal – 09. Mai 2025 in der Stadtbibliothek Wuppertal
- Maker Faire Solothurn – 28.–29. Juni 2025 in der Enter Technikwelt Solothurn
- Maker Faire Hannover – 23.–24. August 2025 im Hannover Congress Centrum (DAS Maker Faire Highlight des Jahres!)
  Der Call for Maker ist bis zum 1. Juni 2025 auf der Webseite der Maker Faire geöffnet. Dort können sich Maker ohne Verkauf, Hochschulen, Schulen und weitere Bildungsinstitutionen kostenlos anmelden.
- Maker Faire Salzburg – 08. November 2025 in der TriBühne Lehen, Salzburg

Die Saison ist also vollgepackt mit Gelegenheiten, um sich inspirieren zu lassen, eigene Projekte zu zeigen oder einfach nur mit Gleichgesinnten zu fachsimpeln. *—das*

Eierlege-Game

Das schnelle Drücken eines Tasters füllt eine LED-Leiste. Umgesetzt mit einem ESP32, macht die Make-Redaktion mit diesem bekannten Ausdauerspiel eines ihrer künstlerischen Projekte auf der Maker Faire Ruhr interaktiv.

von Daniel Schwabe

Für den mechanischen Strauß aus Make 7/24 existiert ein Eierlege-Add-on, dessen Aufbau wir auf Seite 42 beschreiben. Dabei handelt es sich um eine Revolvertrommel, die, von einem Motor angetrieben, Eier legen kann. Der Strauß selbst ist zu groß für unseren Messestand, deshalb hat die Redaktion nur den Eierlege-Revolver mit auf die Maker Faire in Dortmund genommen.

Um dieses Ausstellungsstück interaktiver zu gestalten, hat die Redaktion ein kleines elektronisches Spiel gebaut. Dabei kann man ein Ei gewinnen, indem man in Höchstgeschwindigkeit einen Button mehrfach drückt, um eine LED-Leiste zu füllen.

Das klingt leichter, als es ist. Natürlich leert sich die Leiste auch langsam wieder und man muss ordentlich Durchhaltevermögen beweisen, um die 300 LEDs alle zum Leuchten zu bringen.

Ist die Leiste voll, schaltet ein Relais den Eierlege-Revolver und belohnt den Gewinner mit einem Ei. Bei den Eiern handelt es sich weder um hart- noch weichgekochte Eier, sondern um kleine Plastikbehälter, die mit köstlich-süßen Schweinereien gefüllt sind.

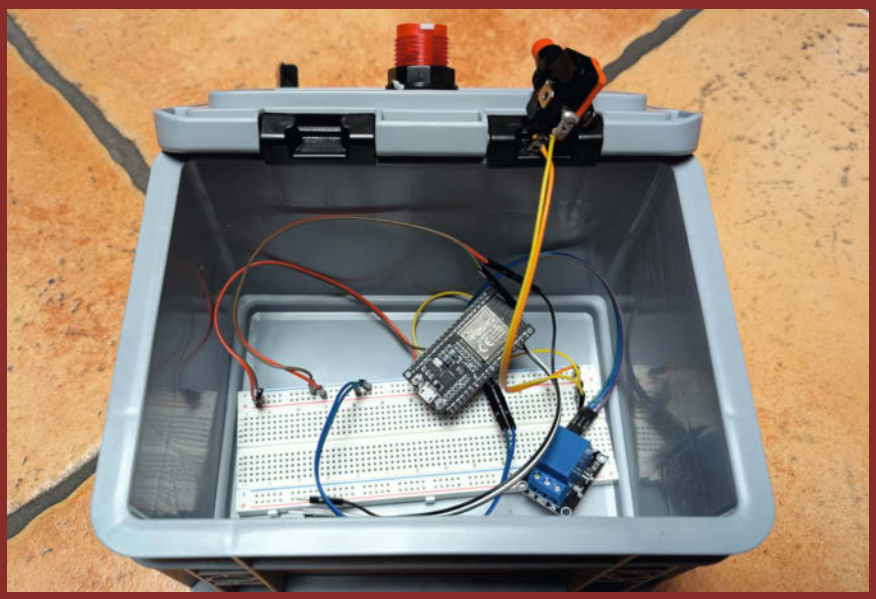
Das Herzstück des Spiels ist ein ESP32. An ihm ist ein Button für die Eingabe angeschlossen. Dabei handelt es sich um einen hochwertigen Arcade-Automaten-Schalter. Dieser erzeugt ein tolles haptisches Drückgefühl und ein sehr befriedigendes Klicken bei jedem Druck.

Für die LED-Leiste ist ein WS2812-Streifen um ein Plastikrohr mit 27 mm Durchmesser gewickelt. Durch das Verhältnis Rohrumfang zu LED-Abstand entsteht auch direkt ein cooles Muster. Am Fuß des LED-Stabs des Rohres (definiert durch die Seite, an der sich die LED-Anschlusskabel befinden) klebt man einen starken Magneten mit Heißkleber ein. Der LED-Streifen kann direkt mit 5 V über den ESP32 versorgt werden, und als Schalter für den Revolver fungiert ein Relais. Die Steuereinheit habe ich in eine kleine Eurobox eingebaut, die optisch zum aktuellen Standdesign der Redaktion auf der Maker Faire passt.

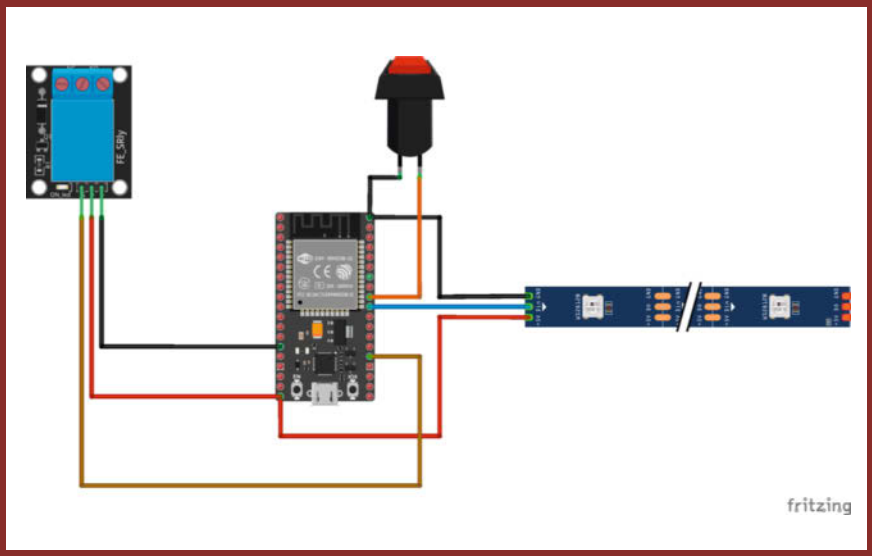
Hat man das Gehäuse so modifiziert, wird nun die Technik in die Dose gepackt. Sobald man das Spiel mit einem 5-V-Netzteil in Betrieb nimmt, kann man loslegen und mit wildem Button-Mashen die LED-Leiste füllen. Auf der Maker Faire schaltet dann das Relais und der Strauß legt ein Ei.

Wer das Spielchen nachbauen will, findet den Code und eine genaue Materialliste unter dem Link. Man kann statt des Eier-Revolvers auch andere Mechanismen anschließen oder z. B. über einen Lautsprecher eine Fanfare ertönen lassen, sobald das Ziel erreicht ist. *—das*

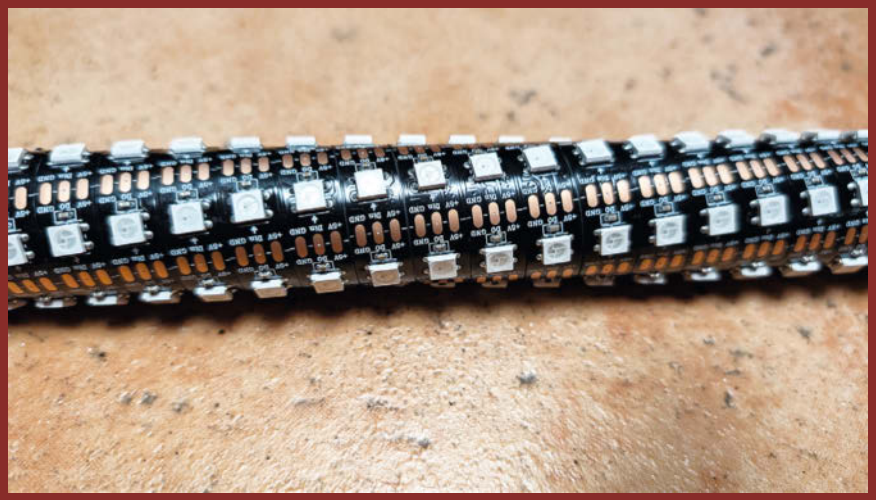
▶ make-magazin.de/xz6j



Ein kleiner ESP32 steuert das LED-Spiel.



Die Verkabelung der Technik ist einfach. Man braucht nur ein paar Jumperkabel.



Der Versatz der LEDs bei einem Rohr von 27 mm macht optisch schon was her.

Pixelwolken-Display

Mit LED-Panels und einem Raspberry Pi kann man sich einen dreidimensionalen Bildschirm bauen, der sogar in der Lage ist, flüssige Animationen abzuspielen. Wie das geht, zeigt dieses Projekt aus Neuseeland.

von Ákos Fodor

Foto: James Brown

Der Maker James Brown, auch bekannt als AncientJames, hat zwei volumetrische Displays entwickelt: Vortex und Rotovox. Diese Geräte sind in der Lage, Bilder und 3D-Objekte holografisch abzubilden und sogar Spiele wie Doom, GTA 3 oder Lander räumlich darzustellen.

Für die Anzeige nutzen die Displays eine sich drehende Vorrichtung, an der zwei HUB75-LED-Panels mit jeweils 128 × 64 LEDs befestigt sind. Das Vortex-Display hat die Panels Rücken an Rücken angeordnet, was eine hellere Ausgabe und eine höhere Framerate ermöglicht, als es beim Rotovox möglich ist. Dafür kann Letzterer mehr Pixel darstellen, weil seine um 90 Grad gekippten Panels zu einer Fläche von 128 x 128 LEDs verbunden sind. Ein 12-V-Gleichstrommotor (YM2776) beschleunigt die Displays auf mehrere Hundert Umdrehungen pro Minute. Mit dem richtigen Timing entstehen so leuchtende Punktwolken aus Voxeln mit einer Framerate von bis zu 30 Bildern pro Sekunde.

Um Verletzungen durch die drehenden Panels zu vermeiden und sie optisch ein wenig auszublenden, verwendet Brown eine Abdeckung aus einer leicht getönten Kunststoffkugel von etwa 30 oder 40 cm Durchmesser (je nach Variante). Der Rest des Gehäuses kommt aus dem 3D-Drucker.

Die Steuerung der Displays übernimmt ein Raspberry Pi 4, der die anzuzeigenden Inhalte / 3D-Modelle als Voxel aufbereitet und an die LED-Panels weitergibt. Um zu den unterschiedlichen Drehwinkeln die passenden LEDs aufleuchten zu lassen, nutzt das System eine Fotodiode, mit deren Hilfe der Raspi erkennt, wann sich die Panels einmal um die eigene Achse gedreht haben.

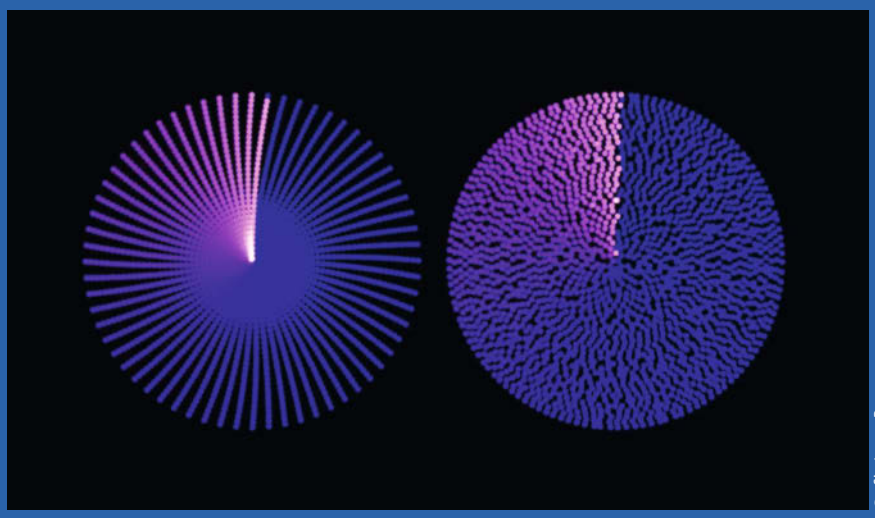
Was zunächst einfach klingt, hat Brown während der Entwicklung vor einige Herausforderungen gestellt. Beim Rotovox musste er etwa eine Lösung dafür finden, wie man die Helligkeit der Punktwolke gleichmäßig verteilt. Mit dem regulären, linearen Multiplexing staute sich die Helligkeit nämlich im Bereich der Drehachse und ließ die Darstellung zum Rand hin dunkler erscheinen. Dieses Problem konnte der Maker mit einer in Python erzeugten Look-up-Table (LUT) beheben, die die zwei Scanlines des Panels (Scan 1:32) so aktualisiert, dass eine homogene Darstellung entsteht.

Den aktuellen Stand des Projekts kann man auf YouTube bestaunen und wer es nachbauen möchte, findet auf GitHub die benötigten 3D-Druckdaten sowie die Software für den Raspberry Pi. Auch ein paar Demos sind mit dabei, die sich mit Bluetooth-Controllern steuern lassen, und man kann auch erst mal mit dem Simulator Virtex auf einem normalen Bildschirm experimentieren. Weitere Details findet ihr unter dem folgenden Link. —*akf*

▶ make-magazin.de/xhay

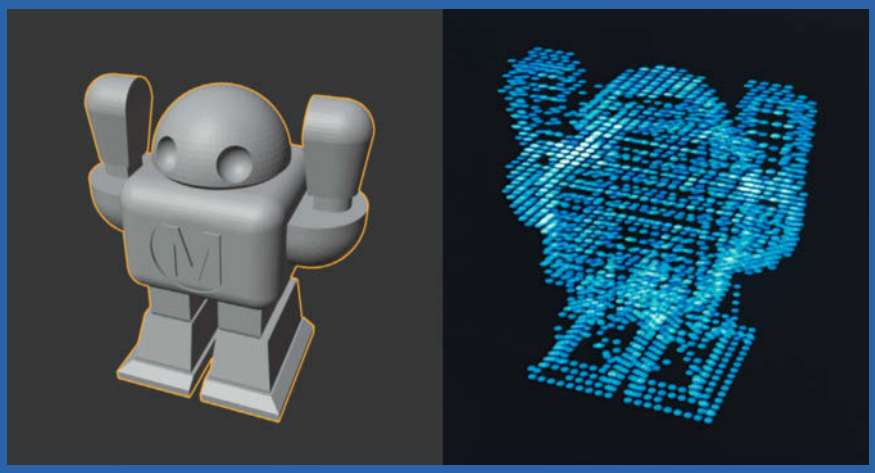
Foto: James Brown

Die beiden Displays im ausgeschalteten Zustand: der Vortex mit Panels nach vorne und hinten (links) und der Rotovox mit der größeren Anzeigefläche, die in eine Richtung zeigt (rechts).



Grafik: James Brown

Die Rotovox-Scanlines von oben betrachtet: Jeder Punkt ist eine LED-Spalte. Das reguläre Multiplexing (links) erzeugt Lücken. Mit der LUT ist die Ausleuchtung homogen (rechts).



Das Makey-Model als OBJ-Datei in Blender (links) lässt sich mit dem Simulator und dem Viewer auch ohne 3D-Display auf dem Raspi darstellen (rechts).

Wie ein Drehzahlregler funktioniert

Um die Drehzahl von Bohrmaschinen, Bandschleifern und unzähligen anderen elektrischen Maschinen einzustellen, drehen wir an einem kleinen Rädchen, ohne darüber nachzudenken. Hinter diesem Rädchen verbirgt sich ein – sowohl mechanisch als auch elektronisch – ausgeklügeltes Teil. Wir gehen auf Erkundungsreise ins Innere einer raffinierten Alltagsschaltung.

von Marcus Hansson

Letztes Jahr wollte mein alter, treuer Rührer der Marke ATIKA, den ich 2017 für 54,90 Euro gekauft hatte, nicht mehr mitspielen. Immer öfter bekam ich beim Anrühren von Beton oder Putz leichte Stromschläge, wenn ich auf den Schalter drückte. Und irgendwann ging es gar nicht mehr: Sobald ich das Gerät über die Steckdose einschalten wollte, flog die Sicherung raus.

Als ich das Gerät zur Reparatur zerlegte, stellte ich schnell fest, dass der Schalter (links in Bild 1) einer gründlichen Reinigung bedurfte. Dazu tauchte ich die Teile in ein wohltuendes Zitronensäurebad, um Kalkreste zu lösen. Dann fiel mein Blick auf den Drehzahlregler (unten in Bild 1), ebenfalls ein trauriger Anblick. Das kleine Rädchen, mit dem die Leistung zwischen 1 und 6 eingestellt werden sollte, war mit Gips und Beton quasi eingegossen. Die Schrift an der Seite war kaum lesbar. Aber mit etwas Mühe konnte ich sie entziffern: DR2-8/1FE. Im Internet schnell gefunden für 19,99 Zloty, umgerechnet 4,66 Euro. Sofort ein neues bestellt, ist ja kein Geld. Aber was für eine Technik sich in dem zuckerwürfelgroßen Gehäuse verbirgt, wollte ich trotzdem wissen. Dafür eignete sich das alte Teil sehr gut für eine Obduktion. Also öffnete ich es, um zu verstehen, wie es funktioniert.

Kurzinfo

» Die Funktionsweise eines Drehzahlreglers gründlich erklärt
» Eine drehende Bewegung in eine lineare umwandeln
» Simulierung der Schaltung in Falstad

Material

» **Drehzahlregler** DR 2-8/1FE

Werkzeug

» **Multimeter** mit Kapazitätsmessung
» **Schraubenzieher**

Mehr zum Thema

» Carsten Wartmann, Schaltungen simulieren, Make 3/24, S. 84
» Florian Schäffer, Ins rechte Licht gesetzt, Make 2/17, S. 32
» Carsten Meyer, Transistor-Tricks, Make 1/23, S. 92

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

Das Labyrinth

Unmittelbar nachdem ich die äußere Hülle entfernt hatte, fiel mir eine kleine Platine in die Hand und ich hörte irgendwo auf dem Boden eine kleine Metallkugel herumspringen. Die Kugel, keine 2 mm im Durchmesser, konnte sich mithilfe einer winzigen Feder, die ich ebenfalls vom Boden aufheben musste, so weit entfernen. Dann kam der Rest: eine weitere kleine Feder, ein kleines weißes Plastikteil mit einem schleppkontaktähnlichen Blech auf der einen Seite und einem Stift auf der anderen, und das Rädchen, an dem man dreht, um die Geschwindigkeit einzustellen. Die Bilder 2 und 4 zeigen das Gehäuse von beiden Seiten, Bild 3 zeigt das Rädchen und das Plastikteilchen.

Das Rädchen passt logischerweise in das Gehäuse in Bild 2. Von der Seite betrachtet zeigt es ein sehr interessantes, labyrinthartiges Muster. Eine wichtige Rolle spielt dabei die etwas kräftigere, geschwungene Rille, die von der Mitte der Scheibe bis zum Rand verläuft. In ihr steckt nämlich der Stift des kleinen weißen Plastikteils. Dieser schaut durch den Gehäusespalt in Bild 2 und greift in die Rille des Rädchens.

Das weiße Teil ist ein Schieber, der in den Schlitz im Gehäuse von Bild 4 passt. Wenn nun das Rädchen gedreht wird, bewegt sich der weiße Schieber in Längsrichtung in diesem Schlitz. Wir haben also eine Drehbewegung in eine lineare Bewegung umgewandelt.

Die Platine

Das wird verständlich, wenn man sich die kleine Platine (Bild 5) anschaut. Dort sind zwei längliche dunkle Flächen, über die die Metallbleche des weißen Kunststoffteils gleiten. Das ist ein verstellbarer Widerstand. Wenn man das Rädchen dreht, gleiten die kleinen Metallzungen über die Kohlebahnen, um den Widerstand stufenlos einzustellen. Mit dem Messgerät habe ich festgestellt, dass der Widerstand bis zu 534 kOhm beträgt.

Aber ein Widerstand allein macht noch keinen Drehzahlregler. Genau genommen han-

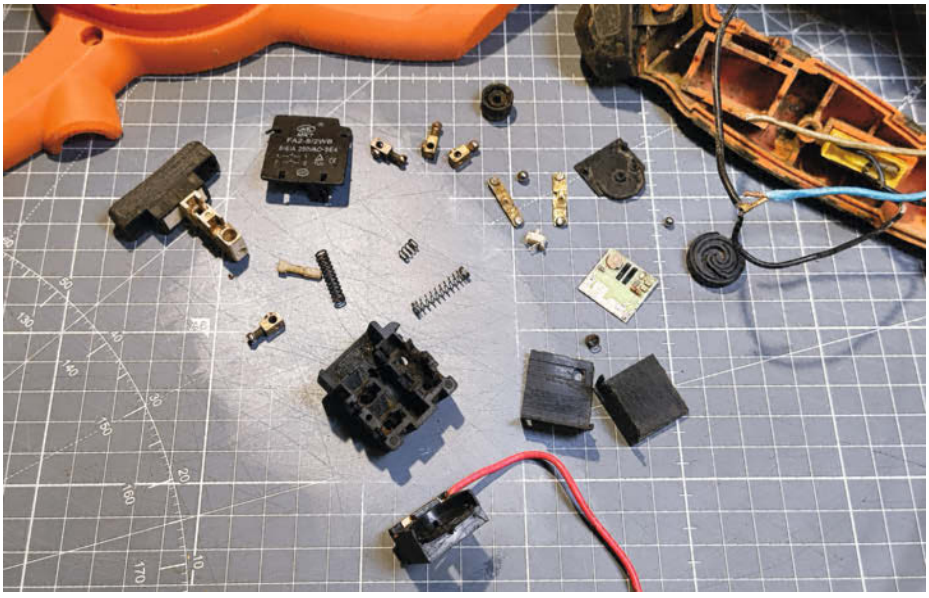


Bild 1: Außer Schalter und Drehzahlregler gibt es noch oben rechts den gelben Entstörkondensator.

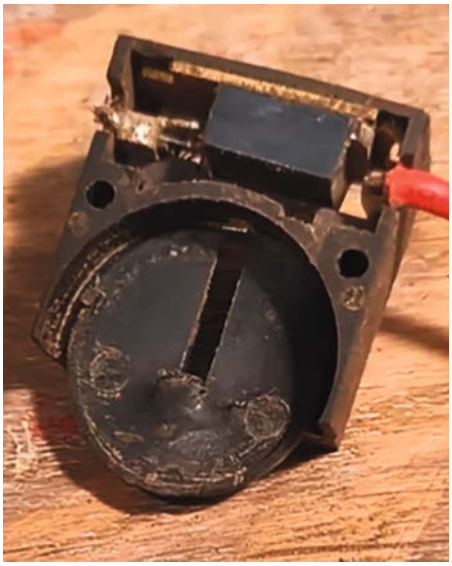


Bild 2: Hier passt das Rädchen hinein. Das Labyrinthmuster zeigt nach unten.

Bild 3: Das Rädchen und der Schieber.

delt es sich auch gar nicht um einen Drehzahlregler, sondern nur um einen Drehzahlsteller, da der Schaltung eine Regelung fehlt.

In meinem Eifer, die Funktionsweise im Detail zu erforschen, habe ich die gesamte Schaltung einmal im Online-Tool Falstad nachgezeichnet (Bild 6). Anders als zum Beispiel bei Fritzing kann man bei Falstad die gezeichneten Schaltungen auch simulieren, um zu überprüfen, ob alles so funktioniert, wie es soll. Dazu musste ich aber erst einmal alle Bauteile richtig identifizieren und vermessen.

Auf der Platine sind einige SMD-Bauteile aufgelötet: ein Diac (das kleine rot-schwarze Röhrchen), ein paar Widerstände und ganz oben im Bild zwei Kondensatoren (silberfarben und hellbraun). Das kleine weiße Bauteil oben links hielt ich lange für einen weiteren Kondensator, bis ich die Platine auf die Heizplatte legte und das Teil entlötete. Es stellte sich heraus, dass es ein Widerstand war, aber mit der Beschriftung nach unten! Wert: 10 kOhm. Alle drei hier vorhandenen Widerstände sind mit einer dreistelligen Zahl beschriftet. Die Zahlen auf den SMD-Widerständen sind so zu interpretieren: Die dritte Ziffer gibt an, wie viele Nullen auf die ersten beiden Ziffern folgen sollen. 123 steht also für 12.000 Ohm und 753 können wir als 75.000 Ohm entziffern. 103 ergibt 10.000 Ohm. Ich habe beide Kondensatoren mit 47,27 nF gemessen. 47 nF ist auch ein üblicher Wert für Kondensatoren.

Auf der Platine befindet sich noch ein regelbarer Widerstand, diesmal kreisförmig (unten links auf der Platine zu sehen). Dieser kann von außen mit einem Schraubendreher eingestellt werden. Auf dem Titelbild ist der Schraubenschlitz dafür zu sehen. Unter einer kleinen Kunststoffkuppel befindet sich eine runde Metallplatte mit zwei im Kreis angeordneten Schleifkontakten (Bild 7). Die beiden Schleifkontakte verbinden die beiden darunter liegenden Kohlebahnen. Die Messung ergibt hier einen Wert von bis zu 680 kOhm.

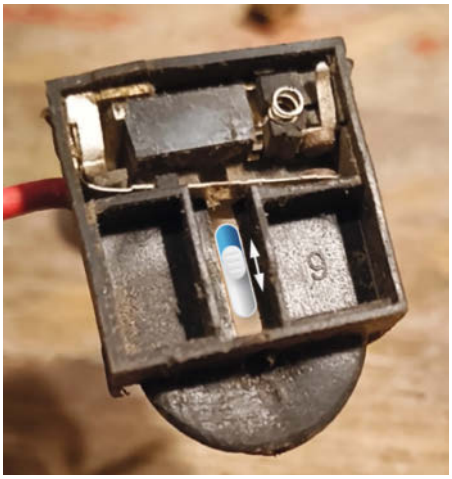


Bild 3: Der kleine weiße Schieber passt in die mittlere Spalte. Darüber liegt die Platine mit den Bauteilen nach unten.

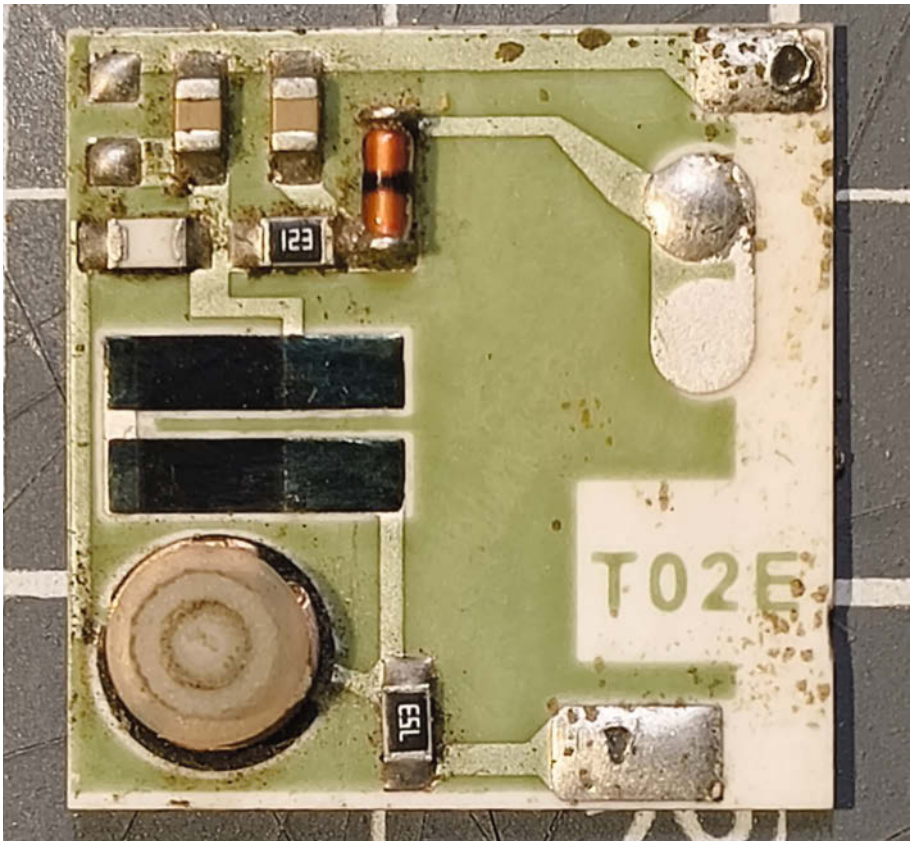


Bild 5: Die Platine.

Triac

Zuletzt entdeckte ich im Gehäuse noch ein kleines elektronisches Bauteil mit drei Beinen – einen Triac. Der Triac ist im oberen Teil der Bilder 2 und 4 zu sehen. Triac ist die englische Abkürzung für triode for alternating current und er hat die Eigenschaft, Strom in beide Richtungen zu leiten, wenn er „gezündet" wird. Dies geschieht über sein drittes Bein, das Gate.

Um was für einen Triac es sich hier handelt, konnte ich leider nicht herausfinden, ohne das Gehäuse irreparabel zu zerstören. Als ich das getan hatte, stellte sich heraus, dass es sich um einen Triac mit der Bezeichnung BTA12-600B handelte. 12 steht für 12 Ampere und 600 für 600 Volt. Nach verschiedenen Datenblättern aus dem Netz bedeutet B einen Triggerstrom von 50 mA, also den Strom, der zum Zünden des Triacs benötigt wird.

Phasenanschnittsteuerung

Mit all diesen Zahlen und Fakten bewaffnet habe ich die Schaltung wie bereits erwähnt auf dem Online-Schaltungssimulator Falstad nachgebaut (Bild 6). Die Schaltung im Bild ist übrigens auch über den Link in der Kurzinfo verfügbar, für alle, die damit herumspielen wollen. Da unter den Bauteilen kein Wechselstrommotor vorhanden war, habe ich stattdessen eine Glühbirne verwendet.

Zusammen bilden diese Bauteile eine Phasenanschnittsteuerung, die sehr häufig zur stufenlosen Steuerung von Wechselstrommotoren und auch zum Dimmen von Lampen

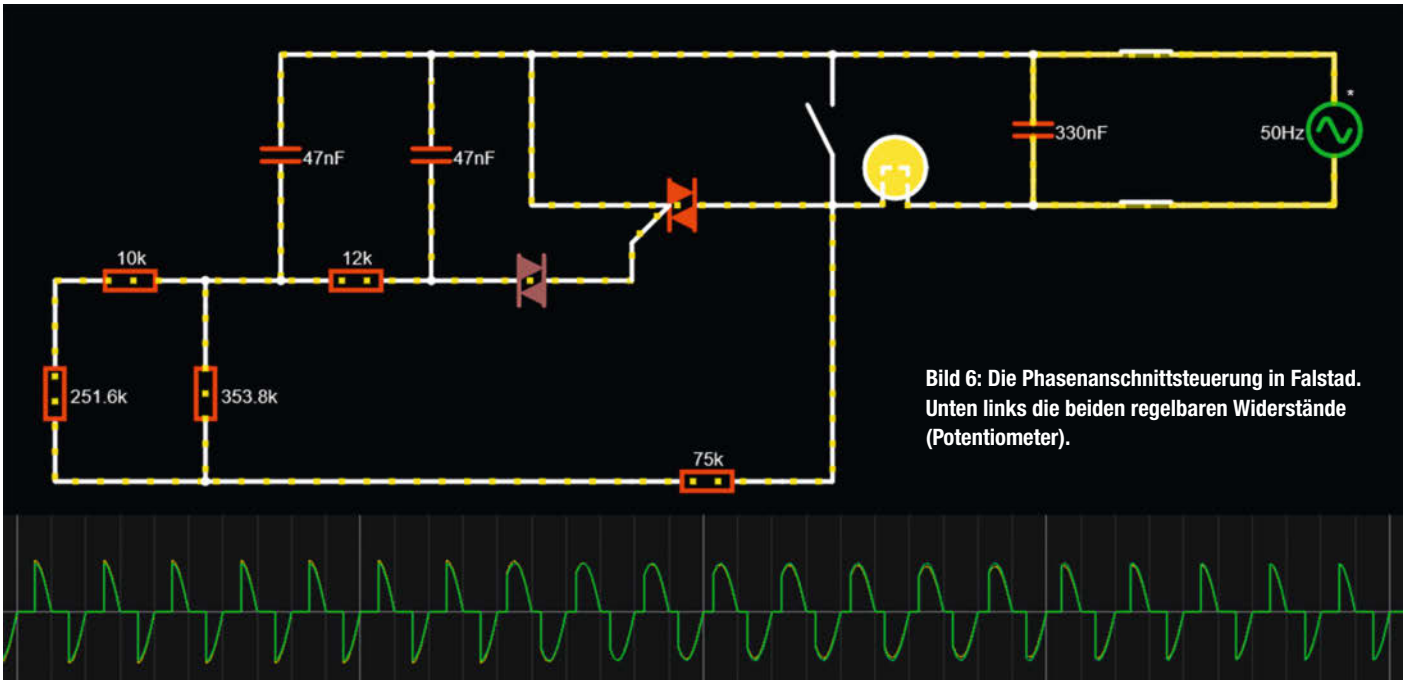


Bild 6: Die Phasenanschnittsteuerung in Falstad. Unten links die beiden regelbaren Widerstände (Potentiometer).

verwendet wird. Dabei wird, wie der Name schon sagt, die Phase angeschnitten (Bild 8).

Man kann sich das wie einen sehr schnellen Lichtschalter vorstellen: Unser Wechselstromnetz hat eine Frequenz von 50 Hertz. Das bedeutet, dass der Wechselstrom 50 mal pro Sekunde in einer Sinuskurve auf- und abschwingt. Dabei durchläuft jede Sinuskurve zweimal den sogenannten Nulldurchgang – bei 0 und 180 Grad in Bild 8. Bei einer Netzfrequenz von 50 Hertz haben wir also 100 Nulldurchgänge pro Sekunde.

Nun kommt der Triac ins Spiel. Einmal gezündet, braucht er einen gewissen Haltestrom, um weiterzuschalten. Aber jedes Mal, wenn der Wechselstrom durch den Nulldurchgang geht, sind Strom und Spannung für einen kurzen Moment gleich null. Deshalb schaltet der Triac 100 Mal pro Sekunde ab und es wird kein Strom weitergeleitet. Erst wenn wieder genügend Strom durch das Gate fließt, wird der Triac gezündet und kann wieder leiten. In dieser kurzen Pause, bevor er wieder einschaltet und wieder Strom durchlässt, wird unser Motor also nicht angetrieben. Je länger die Pause, desto weniger Strom wird dem Motor zugeführt.

Einstellung der Wartezeit

Mit einem einstellbaren Widerstand (Potentiometer) und einem Kondensator kann diese Pause oder Zündverzögerung eingestellt werden. Je höher der eingestellte Widerstand, desto länger dauert es, bis der Kondensator in der Schaltung aufgeladen ist. Wenn die Spannung am Kondensator hoch genug ist, zündet der Triac.

Wenn wir nun an dem kleinen Plastikrädchen drehen, verändert sich der Widerstand an den beiden länglichen Kohlebahnen und damit die Verzögerung, bis der Triac wieder zündet und den Strom zum Motor freigibt. Natürlich merken wir von diesem ständigen Ein- und Ausschalten nichts, denn mit 100 Mal pro Sekunde geht das sehr schnell.

Wozu der Diac

Der Triac allein ist nicht ausreichend. Da er sehr empfindlich auf kleine Spannungsänderungen reagiert, wird zusätzlich ein Diac benötigt, um die Zuverlässigkeit und Genauigkeit der Regelung zu verbessern. Auch Diac ist eine englische Abkürzung: diode for alternating current. Es handelt sich also um eine Diode, die im Gegensatz zu normalen Dioden den Strom in beide Richtungen leitet.

Es sollte kurz erwähnt werden, dass es auch noch die Phasenabschnittsteuerung gibt. Dabei wird der hintere Teil der sinusförmigen Wechselspannung abgeschnitten, bevor er den Nulldurchgang erreicht. Das funktioniert aber anders und hat mit unserem Drehzahlregler nichts zu tun.

Bild 7: Kreisförmiger regelbarer Widerstand, mit dem der Drehzahlregler nach der Fertigstellung kalibriert wird.

Fazit

Noch eine Funktion verbirgt sich im Drehzahlregler. Wenn vollständig aufgedreht wird, schiebt das kleine weiße Plastikteilchen einen Hebel (in Bild 4 zu sehen), und damit fließt der Strom ungehindert an der ganzen Phasenanschnittschaltung vorbei – so, wie ich es mir eigentlich all die Jahre angewöhnt habe, als der Drehzahlregler blockiert war. *—mch*

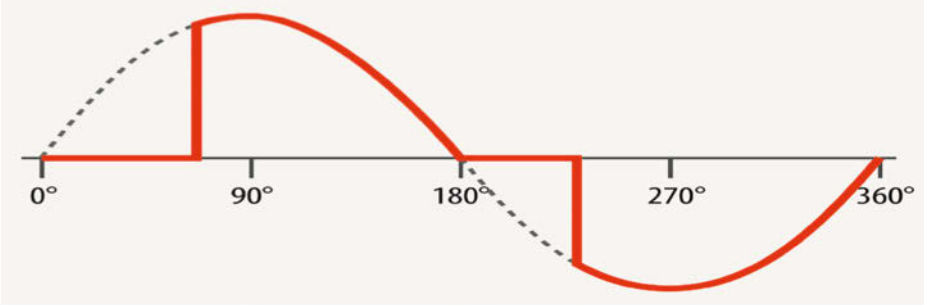


Bild 8: Nach dem Nulldurchgang kann der Triac nicht mehr schalten und die Phase wird angeschnitten.

Herzschlaganzeige über Bluetooth

Herzklopfen zum Nachbauen: Ein ESP32 empfängt den Puls per BLE von einem beliebigen Brustgurt und lässt LEDs im Takt des eigenen Herzens aufleuchten. Als Extra sorgt ein Vibrationsmotor für spürbare Herzschläge – Technik zum Fühlen und Staunen!

von Kai-Uwe Mrkor

In diesem Artikel geht es darum, an einem spannenden Projekt die Möglichkeiten eines inzwischen allgegenwärtigen Funkstandards zu erkunden und einen ebenso praktischen wie faszinierenden Gegenstand zu entwickeln, der von Herzen kommt.

Was aber ist Bluetooth Low Energy (BLE)? Es hat zwar dieselben Wurzeln wie Bluetooth, unterscheidet sich jedoch erheblich in Anwendung und Energieverbrauch. Während Bluetooth für kontinuierliche Verbindungen wie bei drahtlosen Lautsprechern oder Headsets mit hohen Datenraten (bis zu 3 Mbit/s) optimiert ist, liegt der Fokus bei BLE auf minimalem Stromverbrauch und schnellem Verbindungsaufbau.

BLE wurde speziell für Geräte wie Fitnesstracker oder Smartwatches entwickelt, die nur sporadisch kleine Datenmengen senden. Diese Geräte befinden sich die meiste Zeit im Ruhezustand und werden nur zur Datenübertragung aktiviert. Mit einer Datenrate von ca. 1 Mbit/s und Verbindungen bei Bedarf ist BLE ideal für IoT-Anwendungen, bei denen lange Akkulaufzeiten entscheidend sind.

Das Ganze hat leider einen Haken: Die Programmierung einer BLE-Verbindung unterscheidet sich deutlich von der klassischen Bluetooth-Programmierung. Ein erster Blick auf eines der beim ESP32 beiliegenden Demoprogramme wirft daher oft mehr Fragen auf, als der Quelltext beantwortet.

Kurzinfo

- Das Wichtigste zum Thema EKG und Herzfrequenz kurz erklärt
- Bluetooth Low Energy zur Datenübertragung nutzen
- Nach Vorbild eines Fitnesstrackers die Herzfrequenz messen

Checkliste

Zeitaufwand:
2 bis 3 Stunden

Kosten:
zirka 60 Euro

Material

- **Adafruit Feather ESP32 V2**
- **Schiebeschalter** (Ein- und Ausschalter)
- **LiPo-Akku**
- **Adafruit-LED-Jewel** (oder LED-Streifen)
- **Transistor** (2N2222 oder BC547)
- **Mikro-Vibrationsmotor** (3-5 V)
- **Filament** (PLA oder PLA+)

Werkzeug

- **3D-Drucker**
- **Lötkolben**

Mehr zum Thema

- Ramon Hofer Kraner, Bluetooth-LE-Sensoren in die Cloud bringen, Make 6/2024 S. 44
- ESP-Boards mit der Arduino-IDE programmieren

Alles zum Artikel im Web unter
make-magazin.de/x7vv

Einfaches Grundprinzip

Dabei ist es gar nicht so kompliziert. Man muss lediglich das Grundprinzip für die kontinuierliche Abfrage eines Wertes über Bluetooth Low Energy verstehen. Es basiert auf der Kommunikation zwischen einem Client (zum Beispiel unserem ESP32) und einem Server (beispielsweise einem Sensor), die über sogenannte Merkmale (Characteristics) innerhalb eines GATT-Profils (Generic Attribute Profile) erfolgt. Es gibt dazu zwei völlig gegensätzliche Ansätze:

Der erste Ansatz ist das klassische Polling, bei dem der Client den Server, also in unserem Fall den Brustgurt zur Herzfrequenzmessung, in regelmäßigen Abständen auffordert, den aktuellen Wert eines bestimmten Merkmals zu senden. Dies geschieht durch wiederholte Leseanfragen (Read Requests), auf die der Server mit dem entsprechenden Wert antwortet. Dieser Ansatz ist zwar einfach zu implementieren, aber nicht sehr effizient, da die Daten auch dann angefordert werden, wenn sich der Wert nicht geändert hat.

Der zweite, wesentlich energieeffizientere Ansatz verwendet Benachrichtigungen (Notifications). Hierbei abonniert der Client das gewünschte Merkmal, indem er eine Schreibanfrage sendet. Sobald dies geschehen ist, informiert der Server den Client automatisch, wenn sich der Wert ändert, ohne dass der Client weitere Anfragen stellen muss. Dadurch wird die Kommunikation erheblich reduziert und die Geräte können energiesparender arbeiten, da sie nur dann aktiv werden, wenn tatsächlich neue Daten vorliegen.

Für die kontinuierliche Abfrage eines Wertes wird in den meisten Fällen der Benachrichtigungsmodus bevorzugt, da er die Stärken von BLE – also kurze und energieeffiziente Übertragungen – optimal ausnutzt.

Vorabeiten und Einstellungssachen

Es ist für das Projekt nur wenig Vorarbeit in der Arduino-IDE nötig. Sollte bisher noch kein ESP32 zum Einsatz gekommen sein, muss dieser noch zugefügt werden (genaue Vorgehensweise siehe Anleitung in der Linkliste). Will man das LED-Juwel verwenden, ist noch Adafruit Neopixel by Adafruit als weitere Bibliothek zu installieren.

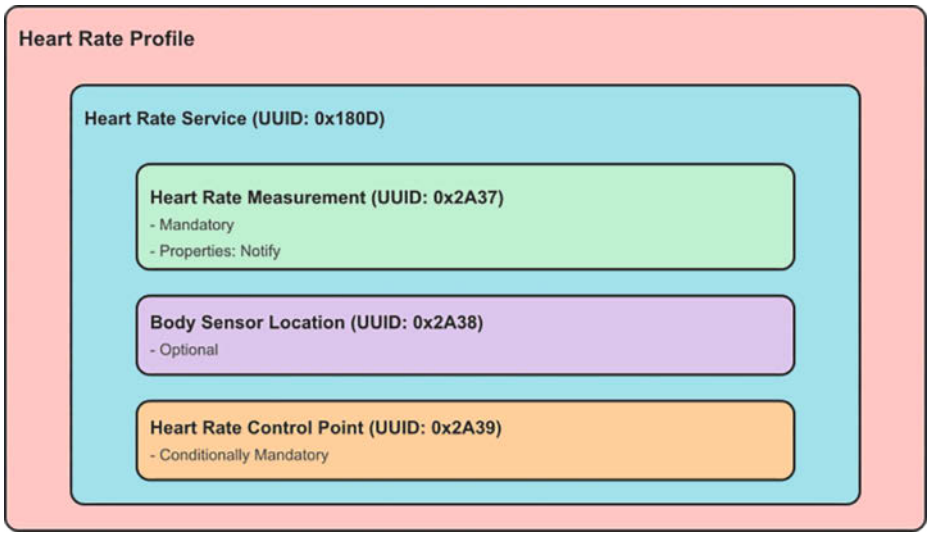


Bild 1: Bluetooth Low Energy (BLE) stellt vorgefertigte Profile bereit.

Alles Weitere ist schon vorbereitet, wofür das bereits erwähnte GATT-Profil sorgt. Es beschreibt, wie Daten zwischen BLE-Geräten organisiert und ausgetauscht werden. Auch regelt es die Kommunikation zwischen einem Client wie beispielsweise einem Smartphone und einem Server wie dem Sensor in unserem Projekt. Der Server stellt die Daten zur Verfügung, während der Client die Daten abruft oder sendet.

Daten werden im GATT als Attribute dargestellt, die jeweils eine eindeutige Kennung, die sogenannte UUID (Universally Unique Identifiers), besitzen. Diese Attribute sind in einer hierarchischen Struktur organisiert: Services dienen als Container für thematisch zusammengehörige Daten, während Characteristics einzelne Datenpunkte innerhalb eines Services repräsentieren.

Ein Beispiel dafür ist die Messung der Herzfrequenz durch einen Fitnesstracker (Bild 1). In unserem Fall ist es der Heart Rate Service mit der UUID 0x180D. Innerhalb dieses Services finden sich Characteristics wie Current Heart Rate mit der UUID 0x2A37, der den aktuellen Herzfrequenzwert in Schlägen pro Minute überträgt.

Die beiden vorab genannten UUIDs sind in unserem Code fest definiert. Sie können in der Spezifikation für Bluetooth Low Energy gefunden werden, die von der Bluetooth Special Interest Group (SIG) veröffentlicht wurde. Solch standardisierte UUIDs ermöglichen es verschiedenen BLE-Geräten und -Anwendungen, miteinander zu kommunizieren, ohne dass proprietäre Protokolle erforderlich sind. Entwickler können so auf eine breite Palette von standardisierten BLE-Diensten zugreifen, ohne eigene UUIDs erstellen zu müssen.

Listing 1

```
void setup() {
  ...
  // BLE initialisieren
  BLEDevice::init("");
  BLEScan* pBLEScan = BLEDevice::getScan();
  pBLEScan->setAdvertisedDeviceCallbacks(
            new MyAdvertisedDeviceCallbacks());

  // Intervall 1 Sekunde (1000 ms)
  pBLEScan->setInterval(1000);

  // Fenster 300 ms
  pBLEScan->setWindow(300);
  pBLEScan->setActiveScan(true);

  // Kurzzeitiges Scannen (5 Sekunden)
  pBLEScan->start(5, false);

  ...
}
```

Listing 2

```
// Klasse für das Scannen nach BLE-Geräten
class MyAdvertisedDeviceCallbacks: public BLEAdvertisedDeviceCallbacks {
  void onResult(BLEAdvertisedDevice advertisedDevice) {
    Serial.print("Gefundenes Gerät: ");
    Serial.println(advertisedDevice.toString().c_str());

    if (advertisedDevice.haveServiceUUID() &&
          advertisedDevice.isAdvertisingService(serviceUUID)) {
       myDevice = new BLEAdvertisedDevice(advertisedDevice);
       doConnect = true;
       deviceFound = true;
    }
  }
};
```

Listing 3

```
// Callback-Funktion für eingehende Benachrichtigungen
void notifyCallback(
  BLERemoteCharacteristic* pBLERemoteCharacteristic,
  uint8_t* pData,
  size_t length,
  bool isNotify) {

  if (pRemoteCharacteristic != nullptr && length > 1) {
    uint8_t heartRate = (uint8_t)pData[1];  // Herzfrequenz extrahieren
    Serial.print("Herzfrequenz: ");
    Serial.println(heartRate);

    // Berechne das Intervall zwischen zwei Herzschlägen
    // (in Millisekunden)
    heartInterval = 60000 / heartRate;
  } else {
    Serial.println("Ungültige Herzfrequenzdaten empfangen.");
  }
}
```

Suchauftrag

Im Setup-Bereich unseres Programms werden die LEDs und der Motor eingerichtet. Gleich danach erfolgt die Initialisierung des BLE-Parts (Listing 1). Zum Einsatz kommt dazu eine Instanz der Klasse BLEScan. Sie ermöglicht es, nach BLE-Geräten zu scannen, deren Daten zu sammeln und bestimmte Informationen über die Geräte zu erhalten (beispielsweise deren Bezeichner, Signalstärke, verfügbare Dienste und andere Dinge).

Dazu wird in unserem Quelltext eine eigene Callback-Klasse übergeben. Solch eine Klasse ermöglicht es, darin definierte Funktionen als Reaktion auf bestimmte Ereignisse aufzurufen. Callbacks sind besonders nützlich in der ereignisgesteuerten und asynchronen Programmierung, da sie es ermöglichen, Funktionen nur bei Bedarf auszuführen. Dies spart Ressourcen und sorgt für eine effizientere Codeausführung, besonders bei Prozessen wie der Kommunikation mit externen Geräten oder der Verarbeitung von Eingaben. Übergeben wird unser Callback über die Methode `setAdvertisedDeviceCallbacks` unserer Instanz der Klasse `BLEScan` (Listing 1). Die dort angelegte Klasse vom Typ `MyAdvertisedDeviceCallbacks` (Listing 2) wird nach dem Start des Scans für jedes gefundene BLE-Device aufgerufen. Die darin enthaltene Methode `onResult()` prüft bei jedem Aufruf, ob das aktuell übergebene Gerät den von uns benötigten Herzfrequenzdienst anbietet. Wenn das der Fall ist, wird die Verbindung mit diesem Gerät vorbereitet. Dazu wird ein Device `myDevice` angelegt und die Flags `doConnect` und `deviceFound` auf `true` gesetzt.

Listing 4

```
...
// Wenn verbunden, Herzfrequenzdaten auswerten
if (connected && pRemoteCharacteristic != nullptr) {
   if (millis() - lastHeartbeatTime >= heartInterval) {
      startHeartbeatAnimation();
      Serial.print(millis()-letztePeriode);
      Serial.println(" ms");
      letztePeriode = millis();
      lastHeartbeatTime += heartInterval;
    }
  }
// Animation aktualisieren
updateHeartbeatAnimation();
...
```

Die Callback-Funktion `onResult()` im Listing 2 dient zur Überprüfung, ob ein gefundenes BLE-Gerät die Herzfrequenz liefert.

Anbandeln

In der `loop()` erfolgt bei jedem Durchlauf eine Abfrage auf `doConnect == true`. In diesem Fall wird die Funktion `connectToServer()` gestartet. Als Erstes wird dort ein Client vom Typ `BLEClient` erzeugt und für diesen werden weitere Callbacks festgelegt. Dieses Mal für Änderungen des Verbindungsstatus, also ob die Verbindung zu einem Gerät hergestellt oder eine bestehende Verbindung unterbrochen wurde. Im nächsten Schritt wird der gewünschte Service auf unsere vorab definierte `serviceUUID` (Herzfrequenzdienst) festgelegt. Danach kommt noch die gewünschte Charakteristik `charUIID` und schon ist das Abonnement der Herzfrequenzdaten abgeschlossen.

Gestartet wird der Empfang über

```
pRemoteCharacteristic->registerForNotify(notifyCallback);
```

Hier ist nun unser letzter Callback zu finden (Listing 3). Der hat es aber auch in sich, denn die ab jetzt zyklisch eingehenden Herzfrequenzdaten werden in genau dieser Methode verarbeitet. Die empfangenen Daten werden dort dekodiert und die Herzfrequenz daraus extrahiert. Anschließend wird das Intervall zwischen zwei Herzschlägen berechnet und in der globalen Variable `heartInterval` abgelegt.

Das Auslesen der Herzfrequenz ist hier übrigens bewusst nachlässig programmiert. Denn bei Frequenzen, welche die Kapazität eines Bytes übersteigen, wird nämlich standardmäßig auf 16-Bit-Werte umgeschaltet. Da aber Werte über 255 bpm (Schläge pro Minute) nicht zu erwarten sind, wurde hier darauf verzichtet. Selbst Frischverliebte sollten solche Herzfrequenzen nicht erreichen. Ganz im Gegenteil, Betroffene sollten unbedingt einen Arzt aufsuchen, da es sich mit hoher Wahrscheinlichkeit um eine ernsthafte Störung handelt.

Herzflackern

Der in unserem Callback aktualisierte Inhalt von `heartInterval` wird in `loop()` ausgewertet (Listing 4).

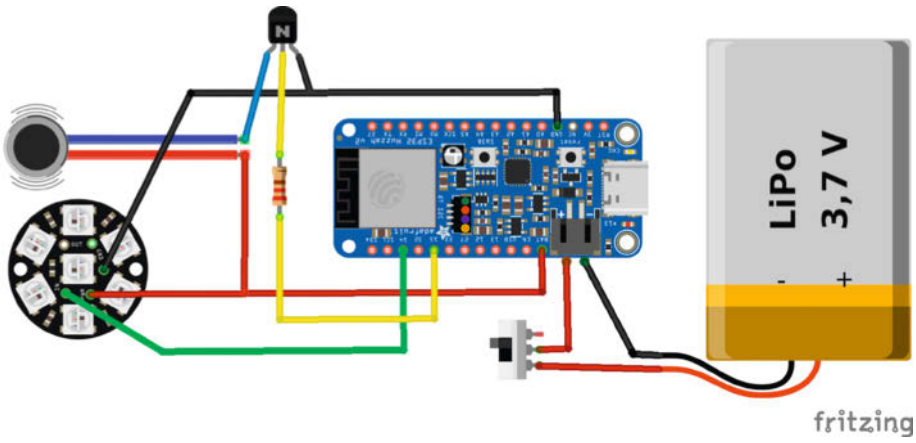


Bild 2: Der Hardware- und Schaltungsaufwand ist gering und muss nachher in das kleine Gehäuse aus dem 3D-Drucker passen.

Bild 3: Das Gehäuse besteht aus zwei tragenden Schalen und einem Haltestift.

Problemvermeidung

Sobald die Verbindung zum Herzfrequenz-Sensor unterbrochen wird, startet unser Code automatisch einen neuen Scan, um das Gerät wieder zu finden. Wird innerhalb einer bestimmten Zeit kein Gerät gefunden, beginnt der Scan erneut. Gemanagt wird dies über die Auswertung der schon angesprochenen globalen Variable `connected` in unserer `loop()`. Ist die Verbindung wieder hergestellt, läuft die Herzfrequenzüberwachung inklusive Animation normal weiter.

Dabei zeigt das LED-Juwel jederzeit, in welchem Zustand sich das Programm befindet und ob etwas nicht funktioniert. Beim Start leuchtet der äußere Ring, bis die Verbindung steht. Im laufenden Betrieb pulsieren alle LEDs im Takt der übertragenen Herzfrequenz. Wenn die Verbindung abbricht oder gar nicht erst zustande kommt, leuchtet nur noch die innere LED. Und wenn das noch nicht reicht, liefert uns die serielle Schnittstelle weitere Informationen.

Das dabei verwendete Vorgehen ist durchaus als klassisch zu bezeichnen. Bei jedem Durchlauf unserer Hauptschleife wird erst einmal geprüft, ob überhaupt ein Gerät angeschlossen ist. Ist dies der Fall, wird nun geprüft, ob die Zeit seit der letzten Ausgabe des Herzschlags bereits größer als `heartInterval` ist. Ist dies der Fall, wird eine neue Ausgabe des Herzschlags gestartet und anschließend `lastHeartbeatTime` mit der aktuellen Zeit überschrieben.

Die Funktion `updateHeartbeatAnimation()` ist innerhalb der `loop()` für die visuelle und haptische Darstellung des Herzschlags verantwortlich. Es wird eine einfache Animation verwendet, bei der die LEDs des verwendeten LED-Jewels langsam in Rottönen aufleuchten und sich danach wieder verdunkeln. Der Motor vibriert dabei leicht, um auch eine haptische Rückmeldung zu geben.

Die Immersion ist jedoch nicht perfekt, da eine minimale Phasenverschiebung zwischen Herzschlag und Ausgabe auftreten kann. Der Grund dafür ist, dass das `BLE-GATT-Profil` keine Funktion vorsieht, die bei jedem Herzschlag ein Ereignis auslöst. Ein solcher „Heartbeat-Trigger" wäre zwar schön, wird aber eigentlich nicht benötigt. Wenn man es nicht weiß, merkt man es auch nicht. Die Frequenz stimmt ja.

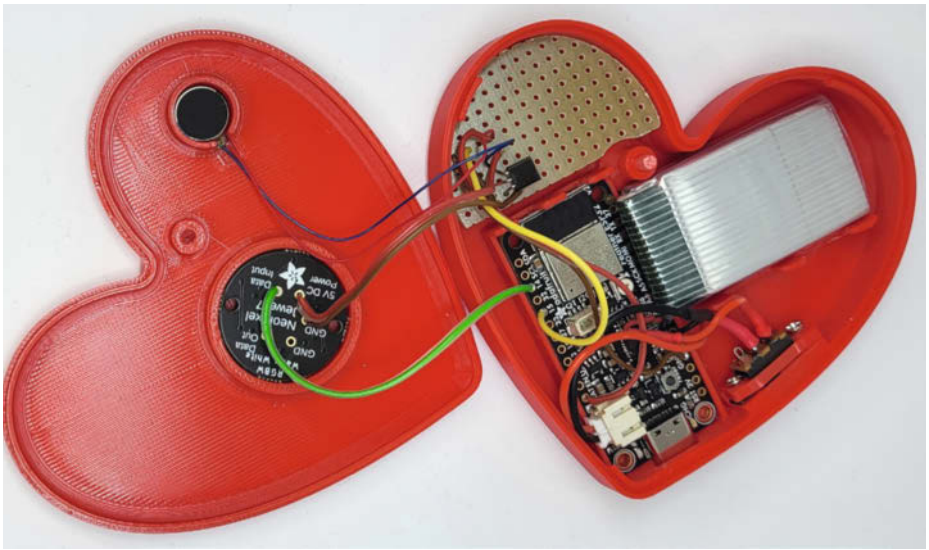


Bild 4: Das Gehäuse ist passgenau auf die einzelnen Bauteile abgestimmt.

Harte Ware

Keine Software ohne die Hardware, auf der sie laufen kann. Viele moderne Mikrocontroller haben die entsprechende Peripherie bereits fest im IC integriert. Die folgenden Ausführungen sind daher nur als Designvorschlag zu verstehen. Hier kann natürlich jeder seiner Kreativität freien Lauf lassen – sei es bei der Elektronik oder beim anschließenden Gehäusedesign.

Ich verwende einen Adafruit Feather ESP32 V2 (Bild 2). Der trägt den ESP32 schon im Namen und kann daher auch mit BLE umgehen. Zusätzlich hat er einen Akku-Anschluss inklusive Ladeschaltung schon dabei. Das minimiert unseren ohnehin schon sehr übersichtlichen Schaltungsaufwand noch weiter. Zum Einsatz kommen lediglich noch ein Schiebeschalter zum Ein- und Ausschalten des Gerätes, ein Transistor 2N2222 oder BC547 mit einem 1 KΩ großen Vorwiderstand in Basisschaltung zur Ansteuerung des Motors, dessen maximaler Strom von 70 mA für eine direkte Versorgung über einen Portpin einfach zu hoch ist, und ein LED-Jewel, sozusagen ein LED-Strip auf einer kleinen Platine.

Das Jewel kann auch bedenkenlos durch einen solchen LED-Streifen ersetzt werden. Wichtig ist nur, und das gilt sowohl für den Jewel als auch für den Streifen, dass sie mit einer Spannungsversorgung aus dem Akku auskommen. Denn der liefert natürlich keine 5 Volt. Das lässt sich aber vor dem finalen Zusammenbau schnell überprüfen.

Zum perfekten Prototypen gehört natürlich auch ein passendes formschönes Gehäuse aus dem 3D-Drucker (Bild 3) in stilisierter Herzform.

Vom EKG zur Herzfrequenz

Das Herz ist der zentrale Motor des menschlichen Kreislaufs. Es arbeitet unermüdlich und pumpt in rhythmischen Kontraktionen das Blut durch unseren Körper. Hinter diesen rhythmischen Bewegungen steckt eine Abfolge von elektrischen Impulsen, die in spezialisierten Herzmuskelzellen entstehen und sich durch das Herz ausbreiten. Die dabei auftretenden Spannungsschwankungen lassen sich mit einem EKG (Elektrokardiogramm) an der Körperoberfläche messen. Die einfachste Variante dieser Messmethode ist ein einkanaliger EKG-Sensor. Er zeichnet die elektrische Aktivität des Herzens mit zwei Elektroden in einer einzigen Ableitung auf.

Die von einem EKG gemessenen Signale sind sehr klein und liegen im Bereich von wenigen Mikrovolt bis zu einigen Millivolt. Sie werden daher in der Regel mit einem rauscharmen Operationsverstärker verstärkt und gefiltert. Übliche Filterkombinationen sind Hoch- und Tiefpassfilter, die tiefe und sehr hohe Frequenzanteile unterdrücken, um das eigentliche EKG-Signal sauber zu isolieren. Nach der Digitalisierung in einem Analog-Digital-Wandler können die Signale zur weiteren Verarbeitung an einen Mikrocontroller oder Computer weitergeleitet werden. Dort werden sie als die uns bekannten Kurven dargestellt.

Bild 5: Die einzelnen Abschnitte eines Herzschlags im EKG

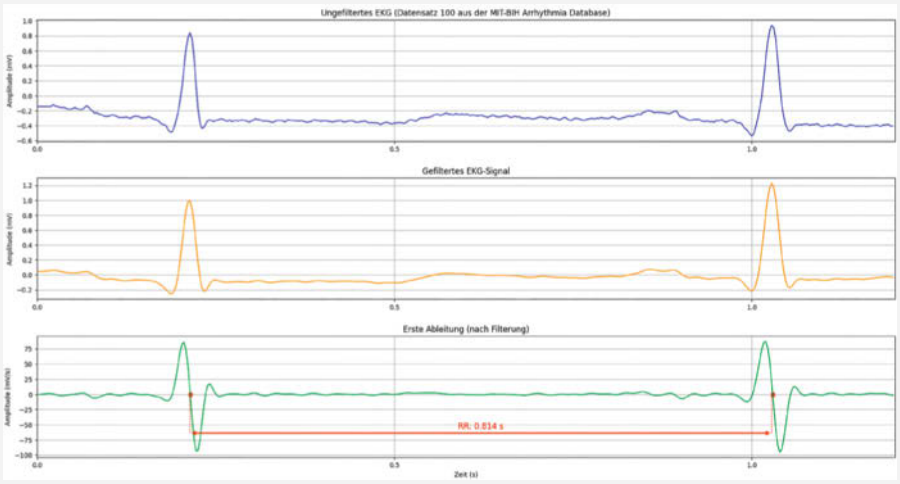


Bild 6: Die einzelnen Abschnitte eines Herzschlags.

Schon früh begann man, die Darstellung des Herzschlags in Teilabschnitte zu zerlegen. Dadurch können die verschiedenen elektrischen Ereignisse im Herzen – vom Sinusknoten bis zur Erregung der Herzkammern – klar unterschieden und analysiert werden. Dazu unterteilt man den Verlauf eines Herzschlages in die P-Welle, den QRS-Komplex und die T-Welle (Bild 5).

Ein einkanaliger EKG-Sensor eignet sich besonders für schnelle Untersuchungen und für den Einsatz in Wearables wie Smartwatches oder mobilen Messgeräten. Er ist einfach anzulegen und ermöglicht eine grundlegende Überwachung der Herzaktivität. Allerdings bietet er nur einen eingeschränkten Blick auf das Herz, da nur eine Ableitung betrachtet wird.

Für eine genauere Diagnose, beispielsweise zur Lokalisierung von Infarkt-Arealen oder zur differenzierten Analyse von Herzrhythmusstörungen, werden daher meist mehrkanalige EKG-Systeme eingesetzt. Die Herzfrequenz kann direkt aus einem einkanaligen EKG bestimmt werden. Dazu wird der zeitliche Abstand zwischen den R-Zacken (RR-Intervall) gemessen, wobei ein Algorithmus diese R-Zacken automatisch erkennt. Vorher wird aber noch das gemessene Signal mit einem Bandpass gefiltert und danach eine Steigungsanalyse durchgeführt (Bild 6).

Das EKG-Ergebnis der Steigungsanalyse (untere Kurve in Bild 6) zeigt sehr schön die ansteigenden und abfallenden Flanken der R-Zacke. Dies ist nicht verwunderlich, da es sich mathematisch um nichts anderes als die erste Ableitung der gefilterten Amplitude nach der Zeit und damit um die Differenz zwischen aufeinanderfolgenden Werten handelt. Üblicherweise sucht man nun nach dem Peak. Dieser ist besonders leicht zu finden, da er zwischen den beiden Extrempunkten der Ableitung der R-Zacke liegt und selbst eine Steigung von null aufweist. Durch den Abstand zwischen zwei solcher R-Zacken kann die Herzfrequenz nach der Formel Herzfrequenz (Schläge pro Minute) = 60 / (RR-Intervall in Sekunden) berechnet und somit direkt in Schlägen pro Minute ermittelt werden. In unserem Beispiel ergibt sich somit aus den dort ermittelten RR-Intervall von 0,814 s eine Herzfrequenz von 73,7 Schlägen pro Minute.

Als besonderes Schmankerl und als Test für den verwendeten 3D-Drucker liegt den Druckdateien auch eine Vorlage für die benötigte Schraube (M3×14) bei. So hat alles die gleiche Farbe. Alternativ lässt sich auch eine handelsübliche Schraube verwenden, die sich dann aber farblich nicht so harmonisch einfügt. Im Gehäuse lässt sich alles gut verstauen (Bild 4). Wegen der geringen Komplexität der Schaltung reicht für die Verdrahtung aller Bauelement eine kleine Lochrasterplatine. Platz ist mehr als genug vorhanden.

Winken und Lächeln

Dieses Projekt zeigt, wie BLE zur Herzfrequenzmessung und zur Erstellung eines interaktiven Geräts mit LEDs und Motoren verwendet werden kann. Es bietet eine interessante Möglichkeit, BLE-basierte Gesundheitsüberwachungsgeräte mit visuellem und haptischem Feedback zu kombinieren. Daneben kann solch ein kleines Tool auch ein wundervolles Geschenk für den oder die Partner(in) sein. Es kommt im wahrsten Sinne des Wortes von Herzen und ist eine gute Möglichkeit, die eigene Herzfrequenz mal so richtig in Schwung zu bringen. —*usz*

Mehr Flash für den Nano

Der Arduino Nano nutzt denselben Mikrocontroller wie der Arduino UNO. Dennoch steht beiden Boards unterschiedlich viel Speicher zur Verfügung. Woran das liegt und wie man das behebt, erklären wir hier.

von Birger Töpelmann und Ákos Fodor

Ein wesentliches Merkmal von Arduino-Boards sind ihre Bootloader, durch die sich ihr Flash ohne zusätzliche Programmier-Hardware beschreiben lässt. Mittlerweile gibt es verschiedene Bootloader mit unterschiedlichen Eigenschaften, was mitunter zu Problemen führt.

Bei einem Arduino Nano (mit AVR-Chip) lässt sich von außen nicht erkennen, welchen Bootloader er verwendet. Je nach Alter und Quelle des Boards kann das noch ATmegaBOOT oder schon Optiboot sein, der seit 2010 verfügbar ist. Mit welcher Variante der Chip geflasht wurde, merkt man spätestens, sobald man mit der Arduino IDE einen Sketch hochlädt. Erscheint trotz richtig eingestelltem Port und USB-Datenkabel ein Verbindungsfehler, ist es ATmegaBOOT, da die Arduino IDE standardmäßig Optiboot erwartet. Im Menü „Werkzeuge/Prozessor" lässt sich dann auf den alten Bootloader umstellen oder man nutzt die Gelegenheit, um gleich zu Optiboot zu wechseln.

Das lohnt sich in jedem Fall, denn der neuere Bootloader ist mit 512 Bytes um einiges schlanker als sein Vorgänger (2 KB), kann Sketches schneller hochladen und löst einen Fehler, bei dem sich der Chip in einer Reset-Schleife aufhängen kann. Mit einem In-System-Programmer (ISP) kann man den Nano schnell mithilfe der Arduino IDE auf Optiboot aktualisieren.

Wo ist der Haken?

Wer einen Blick in die technischen Daten des originalen Arduino Nano wirft, wird mit Erstaunen feststellen, dass der Speicherbedarf des Bootloaders nach wie vor mit 2 KB angegeben ist – und das, obwohl Arduino werkseitig den kompakteren Optiboot auf neue Geräte flasht. Schaut man sich zum Vergleich die Daten des Arduino UNO Rev 3 SMD an, der denselben ATmega328-Chip verwendet, nutzt dieser nur 512 Bytes im Flash-Speicher. Wie kann das sein?

Kurzinfo

» **Blockierten Flash-Speicher freigeben**
» **Neuen Bootloader auf einen Arduino Nano flashen**
» **Arduino Nano als UNO nutzen**

Checkliste

Zeitaufwand:
30 Minuten

Material

» **Arduino Nano**
» **Arduino UNO** oder ISP-Programmer
» **Jumper-Kabel (male-female)**
» **USB-Kabel für Boards**

Mehr zum Thema

» Daniel Bachfeld, Speicherverbrauch in Mikrocontrollern, Make 3/24, S. 92
» Daniel Bachfeld, Memory Maps verstehen, Make 2/24, S. 122
» Daniel Bachfeld, Überblick Speicherarten, Make 1/24, S. 104

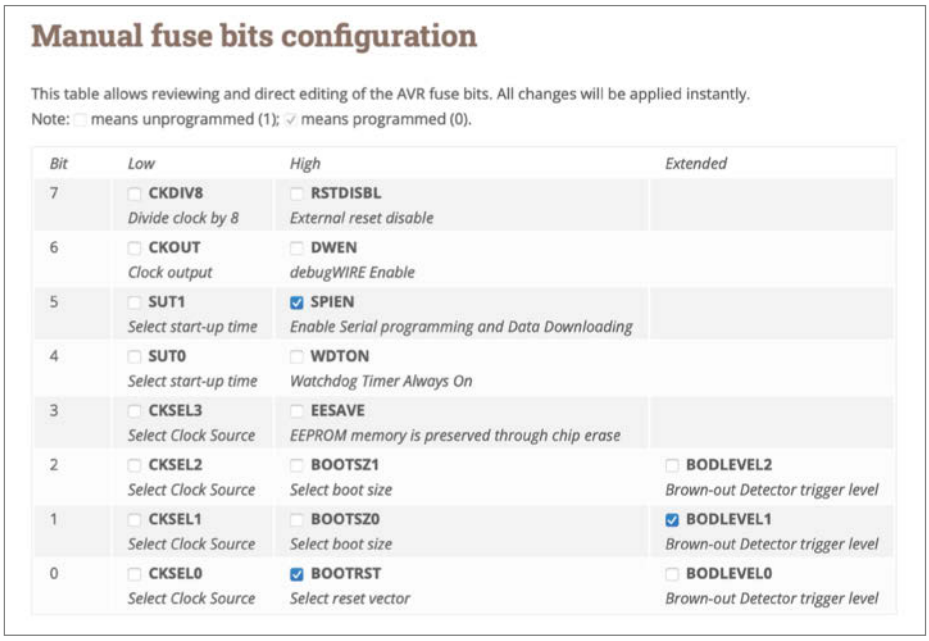
Alles zum Artikel im Web unter *make-magazin.de/xtn4*

Die Antwort ist recht einfach und ärgerlich zugleich: Arduino setzt in der Fertigung des Nano die Fuse-Bits wie beim alten Bootloader, sodass pauschal 2 KB reserviert werden. Dies ließe sich nicht ändern, ohne die Funktion des Boards zu beeinträchtigen, wie man auf Git-

```
uno.name=Arduino Uno
...
uno.upload.tool=avrdude
uno.upload.tool.default=avrdude
uno.upload.tool.network=arduino_ota
uno.upload.protocol=arduino
uno.upload.maximum_size=32256
uno.upload.maximum_data_size=2048
uno.upload.speed=115200

uno.bootloader.tool=avrdude
uno.bootloader.tool.default=avrdude
uno.bootloader.low_fuses=0xFF
uno.bootloader.high_fuses=0xDE
uno.bootloader.extended_fuses=0xFD
uno.bootloader.unlock_bits=0x3F
uno.bootloader.lock_bits=0x0F
uno.bootloader.file=optiboot/optiboot_atmega328.hex

uno.build.mcu=atmega328p
uno.build.f_cpu=16000000L
uno.build.board=AVR_UNO
uno.build.core=arduino
uno.build.variant=standard
```

```
nano.name=Arduino Nano
...
nano.upload.tool=avrdude
nano.upload.tool.default=avrdude
nano.upload.tool.network=arduino_ota
nano.upload.protocol=arduino

nano.bootloader.tool=avrdude
nano.bootloader.tool.default=avrdude
nano.bootloader.unlock_bits=0x3F
nano.bootloader.lock_bits=0x0F

nano.build.f_cpu=16000000L
nano.build.board=AVR_NANO
nano.build.core=arduino
nano.build.variant=eightanaloginputs

## Arduino Nano w/ ATmega328P
## ~~~~~~~~~~~~~~~~~~~~~~~~~
nano.menu.cpu.atmega328=ATmega328P

nano.menu.cpu.atmega328.upload.maximum_size=30720
nano.menu.cpu.atmega328.upload.maximum_data_size=2048
nano.menu.cpu.atmega328.upload.speed=115200

nano.menu.cpu.atmega328.bootloader.low_fuses=0xFF
nano.menu.cpu.atmega328.bootloader.high_fuses=0xDA
nano.menu.cpu.atmega328.bootloader.extended_fuses=0xFD
nano.menu.cpu.atmega328.bootloader.file=optiboot/optiboot_atmega328.hex

nano.menu.cpu.atmega328.build.mcu=atmega328p
```

Bild 1: Die Einstellungen der beiden Boards in der boards.txt unterscheiden sich nur kaum.



Manual fuse bits configuration

This table allows reviewing and direct editing of the AVR fuse bits. All changes will be applied instantly.
Note: ☐ means unprogrammed (1); ☑ means programmed (0).

Bit	Low	High	Extended
7	☐ CKDIV8 Divide clock by 8	☐ RSTDISBL External reset disable	
6	☐ CKOUT Clock output	☐ DWEN debugWIRE Enable	
5	☐ SUT1 Select start-up time	☑ SPIEN Enable Serial programming and Data Downloading	
4	☐ SUT0 Select start-up time	☐ WDTON Watchdog Timer Always On	
3	☐ CKSEL3 Select Clock Source	☐ EESAVE EEPROM memory is preserved through chip erase	
2	☐ CKSEL2 Select Clock Source	☐ BOOTSZ1 Select boot size	☐ BODLEVEL2 Brown-out Detector trigger level
1	☐ CKSEL1 Select Clock Source	☐ BOOTSZ0 Select boot size	☑ BODLEVEL1 Brown-out Detector trigger level
0	☐ CKSEL0 Select Clock Source	☑ BOOTRST Select reset vector	☐ BODLEVEL0 Brown-out Detector trigger level

Bild 2: Der „AVR Fuse Calculator“ schlüsselt die Aufgabe der Fuse-Bits auf.

Bootloader-Fuse-Bits

BOOTSZ0	BOOTSZ1	reservierter Flash-Speicher
1	1	512 Bytes
0	1	1024 Bytes
1	0	2048 Bytes
0	0	4096 Bytes

programmiert = 0, unprogrammiert = 1

Fuse-Bits kurz und knapp

Ein Fuse-Bit in einem Mikrocontroller ist ein spezielles Konfigurationsbit, das grundlegende Eigenschaften und Verhaltensweisen des Mikrocontrollers festlegt. Diese Eigenschaften können Dinge wie die Taktquelle (z. B. interner oder externer Oszillator), die Startverzögerung nach dem Einschalten, die Aktivierung des Watchdog-Timers oder den Schutz eines Speicherbereichs umfassen – etwa denjenigen, in dem sich der Bootloader befindet.

Wichtig: Gerade weil Fuse-Bits solch wichtige Einstellungen vornehmen, sollte man sie nur ändern, wenn man genau weiß, welche Auswirkungen das zur Folge hat. Ansonsten läuft man Gefahr, seinen Mikrocontroller unbrauchbar zu machen. Mit der Methode, die wir im Artikel verwenden, kann das aber nicht passieren.

Hub erklärt (siehe Link in Kurzinfo). Dadurch verschenkt Arduino jedoch 1,5 KB – bei insgesamt 32 KB Flash-Speicher sind das immerhin etwa fünf Prozent, die anderweitig nutzbar wären. Versucht man eine auf Knirsch programmierte Firmware von einem UNO auf den kompakteren Nano aufzuspielen, heißt es im Zweifel also: Der Speicherplatz reicht nicht aus.

Fuse-Bits falsch gesetzt

Der Bootloader des ATmega328 befindet sich im hintersten Bereich des Flash-Speichers und sein Startbereich bzw. der für ihn reservierte Platz lässt sich mithilfe zweier Fuse-Bits definieren: BOOTSZ0 und BOOTSZ1. Je nachdem, wie sie eingestellt sind (0 oder 1), ergeben sich vier mögliche Größen für den Speicherbereich des Bootloaders, wie in der Tabelle „Bootloader-Fuse-Bits“ dargestellt.

Die Bits für diese und andere Einstellungen setzt die Arduino IDE mit dem Programm Avrdude beim Flashen des Bootloaders und macht das gruppenweise in Form von Hex-Codes. Diese findet man in der boards.txt-Datei im Arduino-Programmverzeichnis. In dem Dokument sucht man bei dem jeweiligen Mikrocontroller nach diesen drei Einträgen:

– …bootloader.low_fuses
– …bootloader.high_fuses
– …bootloader.extended_fuses

BOOTSZ0 und BOOTSZ1 werden über die Gruppe der High-Fuses gesteuert. Vergleicht man die Werte dieser Einstellung zwischen Arduino UNO und Nano, lässt sich sofort ein Unterschied feststellen: Beim UNO steht bei den High-Fuses DE und beim Nano DA (Bild 1).

Um zu verstehen, wie sich diese Hex-Werte auf die Fuse-Bits auswirken, kann man den

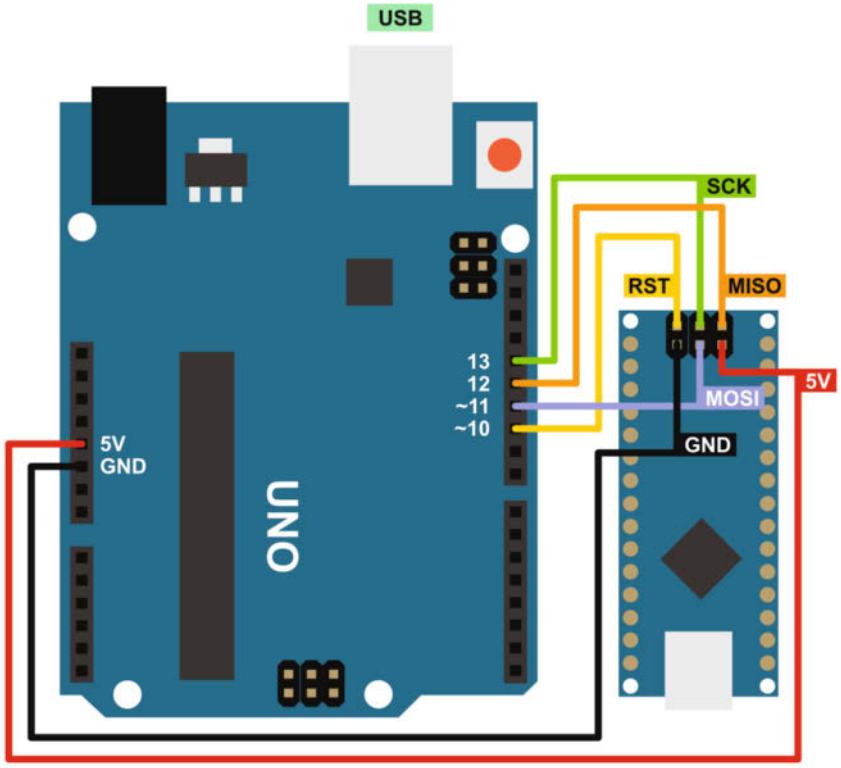


Bild 3: Wir programmieren den Arduino Nano mit einem UNO mithilfe der ISP-Schnittstelle. Den Strom erhält das kleine Board vom UNO.

„AVR Fuse Calculator“ zu Hilfe nehmen (siehe Link in der Kurzinfo). Mit ihm wählt man zunächst den ATmega328P als Mikrocontroller aus und gibt ganz unten auf der Seite bei „Current Settings“ die Hex-Codes aus der boards.txt ein. In dem darüberliegenden Bereich „Manual fuse bits configuration“ kann man anhand der Häkchen ablesen, welche Fuse-Bits programmiert sind (Bild 2). Aber Achtung, die Logik ist invertiert: Ein programmiertes Bit hat eine 0 statt einer 1. Mit der High-Fuse-Einstellung (DE) des UNO stehen beide Fuse-Bits auf 1, was 512 Bytes ergibt. Beim Nano hingegen (mit dem Hex-Code DA) ist BOOTSZ0 auf 1 und BOOTSZ1 auf 0 eingestellt. Daraus ergibt sich eine reservierte Größe für den Bootloader von 2048 Bytes.

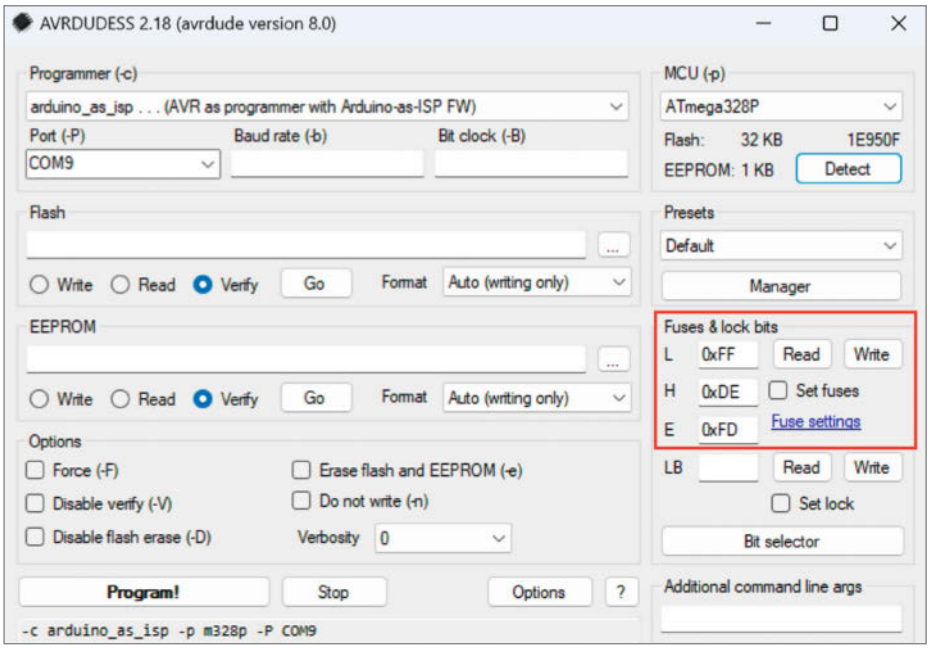


Bild 4: In Avrdudess können wir überprüfen, ob das Setzen der Fuse-Bits funktioniert hat.

Nano wird zum UNO

Um mehr Speicher auf dem Nano freizugeben, könnte man die Werte des UNO in der boards.txt in den Bereich des Nano übertragen und dann den Bootloader flashen. Allerdings gehen diese Änderungen jedes Mal verloren, sobald man die Arduino IDE aktualisiert. Die Mühe können wir uns sparen, denn es gibt einen viel einfacheren Weg: Wir tun einfach so, als würden wir den Bootloader auf einen UNO übertragen. Da die Pins auf beiden Boards mit denselben ATmega328-Ports verbunden sind, macht es technisch keinen Unterschied und der IDE ist es auch ganz gleich, auf welchem Board sich der Chip befindet. Die Funktion der beiden Analog-Digital-Wandler (ADC6 und ADC7), die man zusätzlich auf dem Nano findet, wird nicht beeinflusst.

Zum Flashen des Bootloaders verwenden wir einen Arduino UNO. Diesen muss man wie in Bild 3 mit den ISP-Pins des Nano verbinden. Danach schließt man den UNO per USB am PC an, wählt ihn in der Arduino IDE als Board aus und programmiert ihn mithilfe des Beispiel-Sketches (unter „Datei/Beispiele/11.Arduino-ISP“) als ISP. Anschließend stellt man im Menü „Werkzeuge/Programmer“ noch „Arduino as ISP“ ein und klickt schließlich auf „Werkzeuge/Bootloader brennen“. Dadurch, dass wir noch den UNO als Board ausgewählt hatten, ist der Nano jetzt zu einem UNO geworden.

Ob die Fuses richtig gesetzt sind, lässt sich komfortabel mit dem Programm Avrdudess (alternativ mit Burn-O-Mat) überprüfen (siehe Links in der Kurzinfo), das die Funktionen von dem Kommandozeilenprogramm Avrdude in eine grafische Oberfläche verpackt. Dort wählt man als Programmer „arduino_as_isp“ aus, stellt den COM-Port ein, wählt als Mikrocontroller (MCU) den ATmega328P aus und klickt schließlich bei „Fuses & Lock Bits“ auf „Read“ (Bild 4). Schon erscheinen die drei Hex-Werte, von denen der High-Fuse bei DE stehen sollte.

Problem gelöst

Ab jetzt kann man den Nano in der Arduino IDE als UNO programmieren. Dadurch entfällt auch das lästige Auswählen des richtigen Bootloaders im Menü.

Am besten klebt ihr euch auf das modifizierte Nano-Board einen Aufkleber mit der Aufschrift „UNO“, damit ihr später nicht durcheinanderkommt. Aber keine Sorge: Es ist überhaupt nicht schlimm, falls ihr das Board doch mal versehentlich als Nano mit einem Sketch bespielt. Die Arduino IDE meckert in diesem Fall nur, wenn das Programm 30720 Bytes überschreitet. *—akf*

heise academy

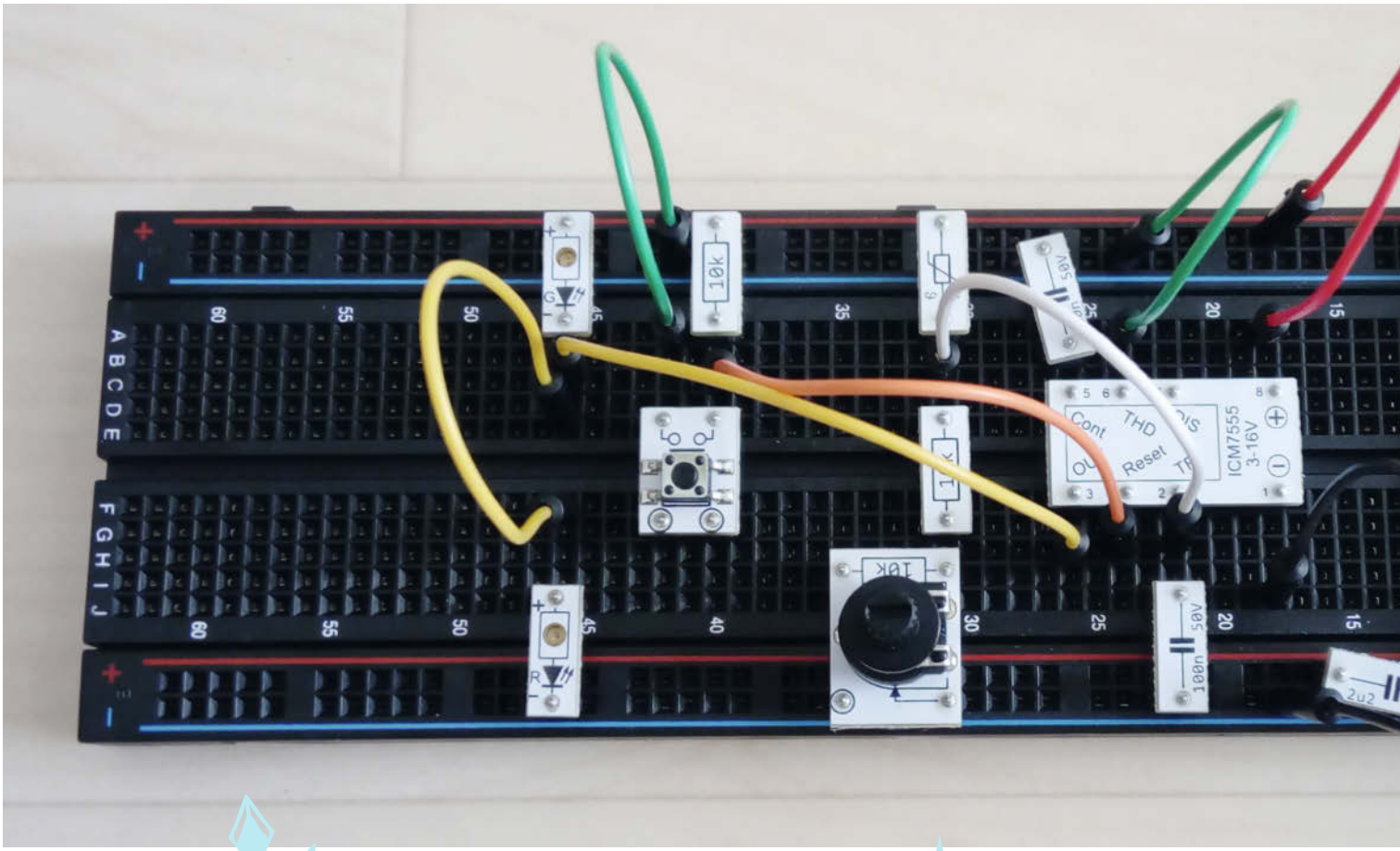
Gemeinsam weiterkommen

Ihr Partner für professionelle IT-Weiterbildung

Die heise academy bietet praxisrelevante Weiterbildung, die Sie voranbringt. Lernen Sie von führenden Experten, erweitern Sie gezielt Ihre IT-Skills und wenden Sie Ihr Wissen direkt an. Bauen Sie heute das IT-Know-how auf, das morgen den Unterschied macht.

- Individuelle Lösungen für Einzelpersonen und Teams & Organisationen
- Flexibles Lernen mit On-Demand-Kursen und Live-Events
- Aktuelle IT-Themen in bewährter heise-Qualität
- Direkter Austausch mit erfahrenen IT-Experten

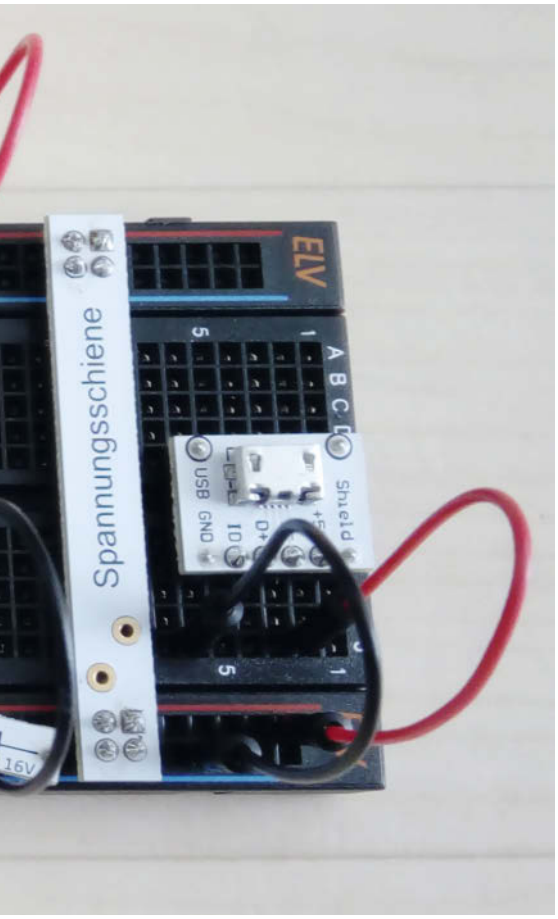
> Jetzt IT-Wissen vertiefen unter heise-academy.de



Frostwarner

Eine praktische Schaltung mit Nutz- und Lerneffekt ist eine Frosterkennung mit einem Temperatursensor und dem Timer-IC ICM7555. Eine LED zeigt eine Temperaturunterschreitung so lange an, bis die Schaltung zurückgesetzt wird. Der Aufbau lässt sich einfach mit dem Make-Experimentierset, auf einem herkömmlichen Breadboard oder auf Platinen umsetzen.

von Michael Gaus und Miguel Gashi



Kurzinfo

» Verwendung des Timer-ICs ICM7555 (verbesserter NE555)
» Temperaturschwelle per Potenziometer veränderbar
» Bei Temperaturunterschreitung leuchtet die LED so lange, bis ein Reset erfolgt.

Checkliste

Zeitaufwand:
1 Stunde

Kosten:
44,95 Euro (Heft mit Experimentierkit)

Mehr zum Thema

» Thomas Garaio, Weihnachtsstern mit Oxocard, Make 7/24, S. 50
» Carsten Wartmann, Make-Breadboard++ Experimentierset zum Selbstbau, Make 2/23, S. 82
» Michael Gaus und Miguel Köhnlein, Make-Experimentierset: Servo-Tester selber bauen, Make 5/22, S. 98

Material

» **ICM7555** oder IC NE555
» **LEDs** rot und grün
» **Vorwiderstände für LEDs** 470 Ω (beim Experimentierkit im LED-PAD integriert)
» **Widerstände** 2 × 10 kΩ
» **Poti** 10k
» **Kondensatoren** 2 × 100 nF
» **Elektrolytkondensator** 2,2 µF
» **Taster**
» **NTC 10k** (bei 25 °C) temperaturabhängiger Widerstand (negativer Temperaturkoeffizient)
» **Breadboard**
» **5-V-Spannungsversorgung** (etwa USB-Netzteil)
» **DuPont-Jumperkabel** Stecker / Stecker

Alles zum Artikel im Web unter
make-magazin.de/xvv7

Wenn im Frühling die Gartenzeit beginnt, dann stehen viele Hobbygärtner vor der Frage, ob es nachts aufgrund von Frost vielleicht doch noch zu kalt für die empfindlichen Pflanzen sein könnte. Diese sehr einfach aufzubauende Schaltung kann hier nützliche Dienste leisten.

Die Schaltung wird einfach einige Nächte lang draußen im Garten platziert, um überprüfen zu können, ob es nachts immer noch Frost gibt. Wenn morgens die rote statt der grünen LED leuchtet, dann ist es noch zu kalt für die empfindlichen Pflanzen. Im Prinzip handelt es sich um eine Erkennung von Temperaturunterschreitungen mit einem 1-Bit-Speicher.

Schaltungskonzept

Bei dieser Schaltung wird nur einer der beiden integrierten Komparatoren des ICM7555 (im Schaltplan, Bild 1, durch den kompatiblen NE555 dargestellt) verwendet, und zwar derjenige, der den SET-Eingang des internen Flipflops steuert. Der negative Eingang dieses Komparators ist an Pin 2 (Trigger, TRI) verfügbar, der hier an einen Spannungsteiler, bestehend aus einem temperaturabhängigen Widerstand (NTC 10 kΩ) und einem Pull-down-Widerstand (R1 + Potenziometer R2), angeschlossen ist. Der Kondensator C1 glättet das Signal leicht, sodass sich Widerstandsänderungen am Potenziometer weicher auf den Komparatoreingang auswirken. Abhängig von der Temperatur ändert sich der Widerstandswert des NTC. Er beträgt 10 kΩ bei einer Temperatur von 25 °C und steigt bei niedrigerer Temperatur an. Die Kennlinie ist nicht linear. Der von uns verwendete Typ hat laut Datenblatt bei 5 °C einen Widerstandswert von ca. 25,3 kΩ, bei 0 °C sind es ca. 32,5 kΩ.

Der positive Eingang des integrierten Komparators liegt auf einem Spannungspegel von ungefähr 1/3 der Versorgungsspannung (V_{cc}). Wenn die Spannung an Pin 2 unter diese Schwelle sinkt, dann schaltet der Ausgang des Komparators auf einen High-Pegel, wodurch der integrierte Flipflop im ICM7555 durch den

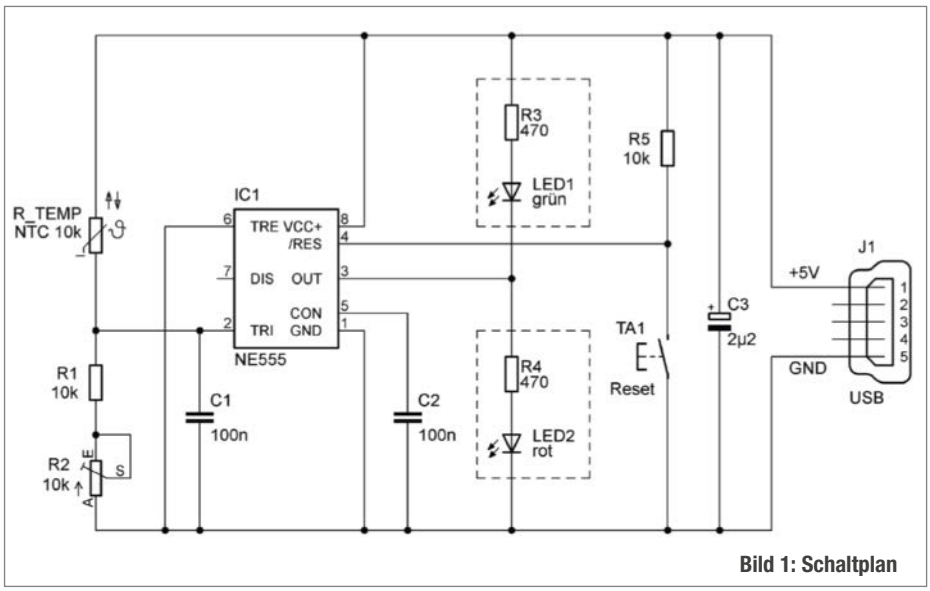


Bild 1: Schaltplan

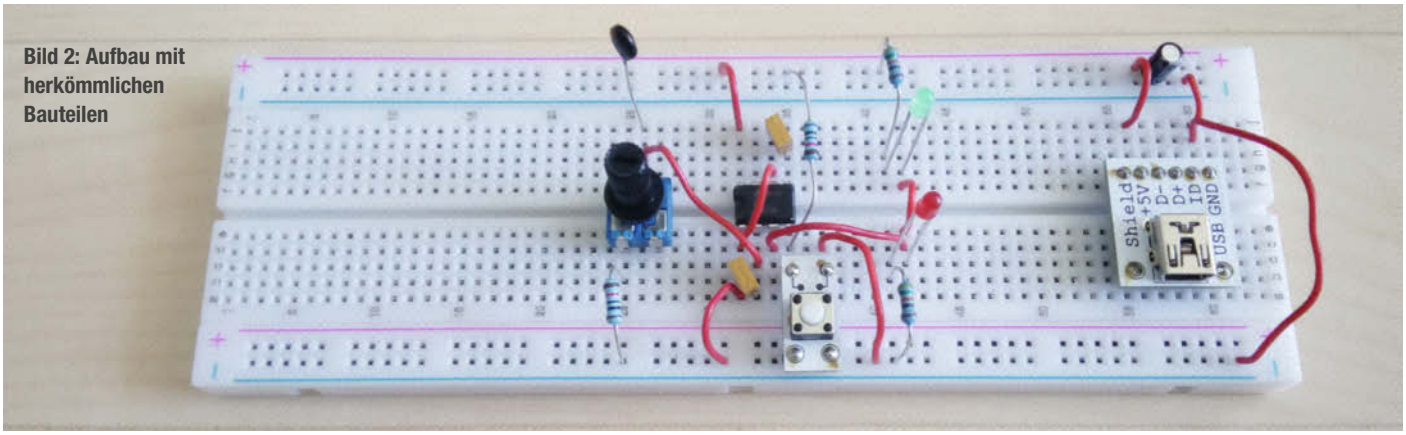


Bild 2: Aufbau mit herkömmlichen Bauteilen

internen SET-Eingang gesetzt wird. Somit wechselt der Ausgang an Pin 3 von Low- auf High-Pegel, sodass die grüne LED erlischt und die rote LED leuchtet.

Der zweite Komparator im ICM7555 wird hier nicht benutzt. Der entsprechende Eingang an Pin 6 (Threshold) ist mit GND verbunden, sodass der Komparatorausgang auf Low-Pegel liegt und somit den internen Reset-Pin des integrierten Flipflops nicht aktivieren kann. Die einzige Möglichkeit, den Flipflop-Ausgang wieder zurückzusetzen, besteht darin, am externen Reset-Pin (/RES, Pin 4) einen Low-Pegel anzulegen. Dies kann über den Taster TA1 erfolgen. Wenn also die Temperatur unterhalb einer bestimmten Schwelle abgesunken ist, wird dies so lange gespeichert, bis der Reset-Taster betätigt wird.

Kurz zusammengefasst: Sobald die Spannung an Pin 2 unterhalb 1/3 V_{CC} abgesunken ist, wird die grüne LED ausgehen und die rote LED so lange leuchten, bis der Reset-Taster betätigt wird. Die Spannung an Pin 2 entspricht einer bestimmten Temperatur des NTC, die Triggerschwelle kann über das Potenziometer R2 eingestellt werden.

Temperaturschwelle einstellen

Vor der Benutzung muss mit dem Potenziometer die gewünschte Triggerschwelle für die Temperatur eingestellt werden. Der ungefähre Widerstandswert des NTC kann dem Datenblatt entnommen werden. In einer Tabelle sind die entsprechenden Werte in 5-°C-Schritten aufgeführt. Zwischenwerte können bei Bedarf über eine Formel berechnet werden.

Wenn eine Triggerschwelle von 0 °C gewünscht ist, dann beträgt der NTC-Widerstandswert ungefähr 32,5 kΩ. Um einen Spannungspegel von 1/3 V_{CC} an Pin 2 zu erreichen, muss der Pull-down den halben Wert haben, in diesem Fall dann 16,25 kΩ. Da der Festwiderstand R1 bereits 10 kΩ hat, müsste also das Potenziometer auf ungefähr 6,25 kΩ eingestellt werden.

Möchte man stattdessen eine Triggerschwelle von 5 °C haben, ergibt sich folgende Rechnung:

- Der NTC-Widerstandswert bei 5 °C beträgt 25,339 kΩ
- Pull-down = halber NTC-Widerstandswert = 25,339 kΩ / 2 = 12,6695 kΩ
- Poti-Widerstand = Pull-down – 10 kΩ = 12,6695 kΩ – 10 kΩ = 2,6695 kΩ

Im ausgeschalteten Zustand der Schaltung kann dann das Potenziometer mithilfe eines Multimeters eingestellt werden, um diesen Widerstandswert zu erzielen. Wenn eine Feineinstellung vorgenommen werden soll, muss der NTC auf die gewünschte Trigger-Temperatur gebracht, akklimatisiert und dann das Potenziometer so eingestellt werden, bis die rote LED gerade von aus nach ein umschaltet. —*caw*

Make: Online

Beliebt auf heise+

Trenntrafo mit zusätzlichem Gleichspannungsausgang selbst bauen

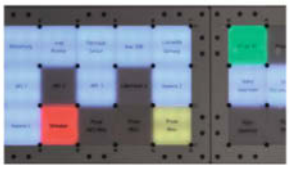
Zum festen Inventar jeder Elektro-Werkstatt gehört ein Trenntrafo, der bei Reparatur und Test von Geräten für Sicherheit sorgt. Wir bauen das Gerät selbst.

22.12.2022 10 | Make Magazin

Bastelprojekt: Taupunkt-Lüftung mit ZigBee-Funksteckdose

Der ESP32-C6 hat eine ZigBee-Antenne und lässt sich mit Display und Steckanschlüssen ausrüsten. So setzt man einen Taupunkt-Lüfter ohne 230-V-Verkabelung um.

05.03.2025 31 | Make Magazin



Bastelprojekt: Ein modulares Kontrollpanel wie aus einem Agentenfilm bauen

Gebaut wurde ein Panel, das mit hinterleuchteten Statusfeldern die Zustände und Fehler von 3D-Druckern, Sensoren und weiteren Geräten in der Werkstatt anzeigt.

03.02.2025 44 | Make Magazin

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

8 clevere 3D-Generatoren

In diesem Video zeigt Johannes, wie man mit acht innovativen Online-Generatoren schnell und ohne Installation komplexe Daten für 3D-Drucker, CNC-Fräsen oder Laserschneider erstellen kann. Interessant sind der Kugellagergenerator und der Zahnradgenerator von Makerworld, wo sich die Modelle für verschiedene Anwendungen anpassen lassen. Außerdem gibt es Generatoren, die das Erstellen von Schrauben, Staubsaugeranschlüssen und Puzzleteilen erheblich erleichtern. Diese praktischen Tools sparen nicht nur Zeit beim Modellieren, sondern sind auch sehr nützlich für die Erstellung von Prototypen oder als Platzhalter in größeren Projekten. —*dus*

▶ www.youtube.com/@MakeMagazinDE

Weitere aktuelle Videos:

Bleib informiert:

www.make-magazin.de

@makemagazinde

Instagram: @makemagazinde

Facebook: @makemagazin.de

X (Twitter): @MakeMagazinDE

TikTok: @makemagazinde

Bluesky: @makemagazin.de

WhatsApp Channel: Make Magazin Deutschland

GitHub: MakeMagazinDE

Threads: @makemagazinde

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

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

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

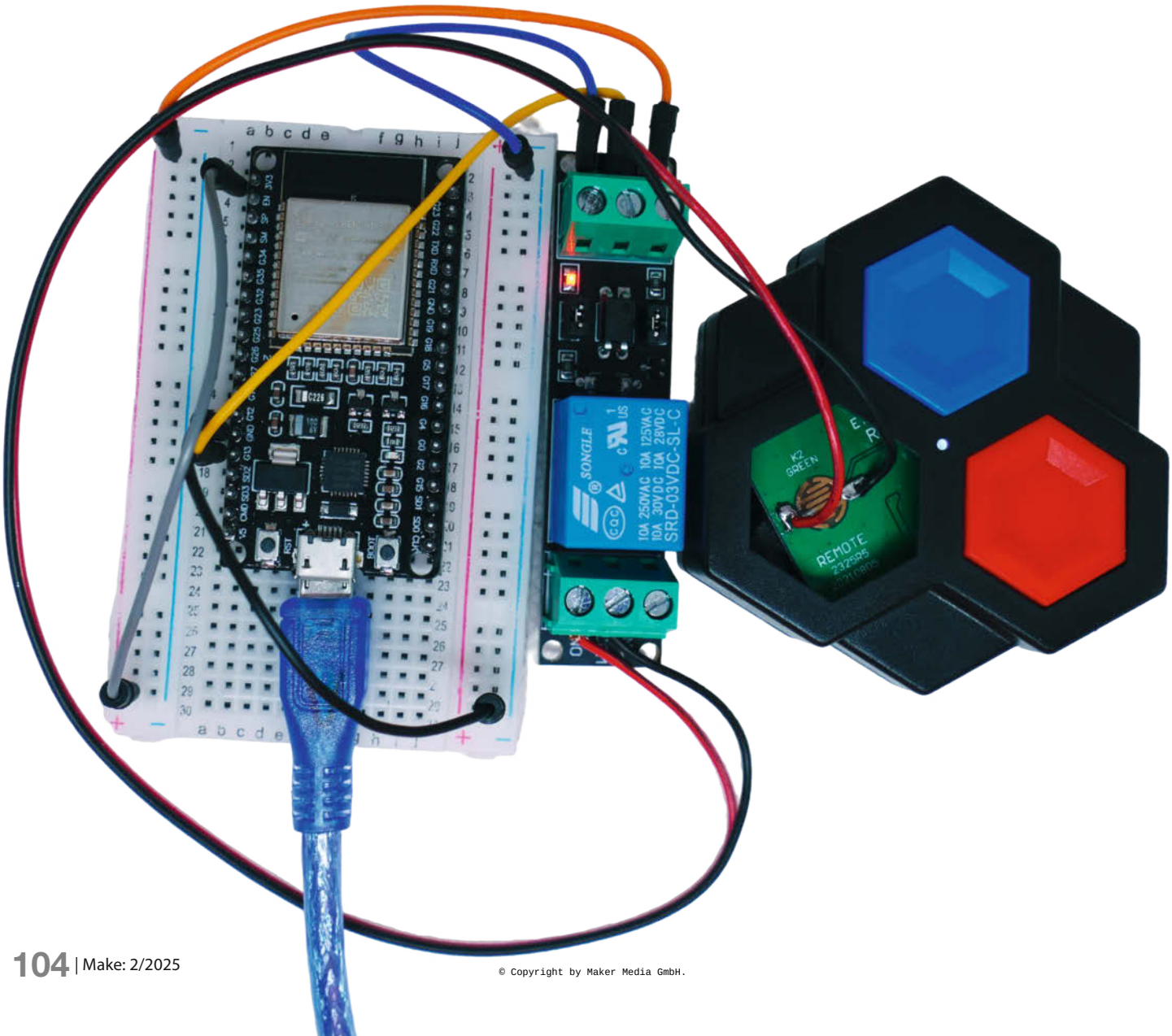
Nichts mehr verpassen: Abonniere unseren Newsletter! Weitere Infos unter:
www.maker-faire.de

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

GraviTrax mit ESP32 steuern

Seit 2017 bietet Ravensburger die GraviTrax-Kugelbahn-Baukästen an. Die „Power"-Serie sorgt seit 2022 für technische Finesse. Der GraviTrax Power Element+ Controller erlaubt es, die Power-Elemente per Knopfdruck zu steuern. Durch einen kleinen Hack lassen sich die Taster durch ESP32-gesteuerte Relais ersetzen und damit programmgesteuerte Kugelbahnen realisieren.

von Ulrich Schmitz



Das GraviTrax-System ist ein modulares, interaktives Kugelbahn-Bausystem, das Kinder und Erwachsene dazu einlädt, kreative Bahnen mit verschiedenen Bauteilen, Erweiterungen und physikalischen Effekten zu konstruieren. Die Bewegung der Kugeln kann durch Steuerungselemente wie Weichen oder Magnete aktiv beeinflusst werden.

Kurzinfo

- Vorstellung des GraviTrax-Systems
- Umbau des GraviTrax Element+ Controllers
- Ansteuerung eines Relais über ESP32
- Programmierung einer Steuerung

Checkliste

Zeitaufwand:
2 Stunden

Kosten:
20 Euro (zzgl. GraviTrax-Baukasten)

Material

- ESP-32-Wroom-32 Dev Kit C V4
- USB-Powerbank
- USB-Kabel
- Drahtbrücken
- GraviTrax-Baukasten mit Element+ Controller / Empfängerbaustein
- Breadboard oder Platine zum Aufbau
- 1-Kanal-Optokoppler-Modul mit 3-V-Relais-Power-Switch

Werkzeug

- Lötkolben
- Seitenschneider

Mehr zum Thema

- Dr. Harald Bögeholz, Programmier die Kugel, c't 23/2023 S. 78
- Ulrich Schmitz, Arduino über App-Inventor-Apps steuern, Make 1/2025 S. 112

Alles zum Artikel im Web unter *make-magazin.de/xfd9*

Das GraviTrax-System

Die Bauelemente der unterschiedlichen GraviTrax-Linien (Standard, Vertical, Junior usw.) sind untereinander kompatibel und lassen sich beliebig austauschen. Dabei sind die Power-Steine zum Betrieb der Kugelbahnen nicht zwingend erforderlich, erhöhen aber die Gestaltungs- und Steuerungsmöglichkeiten erheblich. Es gibt Elemente, die eine vorbeikommende Kugel erkennen und daraufhin ein Signal an andere Bausteine senden. Die Empfängerbausteine setzen diese Signale je nach Bauart in unterschiedliche Aktionen um.

Über den Kanal-Button wählt man am Empfängerbaustein vor, ob man Rot, Grün oder Blau als Kanal nutzen möchte. Die Kanäle sind alle gleichberechtigt. Am Empfänger wechselt die Farbe beim Drücken des Buttons von Weiß nach Grün, Rot, Blau und dann in den Auszustand. Die maximale Zahl von drei Kanälen scheint dabei willkürlich gesetzt zu sein. Durch die Farben kennzeichnet man gleichzeitig, welche Sendermodule welche Empfängerstationen bedienen. Dabei ist es auch möglich, mehrere Empfängerstationen auf eine Kanalfarbe zu setzen, die unterschiedliche Aktionen ausführen können. In der GraviTrax-Welt gibt es aktuell drei Steine, um Signale zu empfangen und dadurch ausgelöst unterschiedliche Aktionen durchzuführen.

Funktionen der Signalempfänger

Switch: Mit dem Switch-Empfängermodul lassen sich Weichen kontrollieren. Wenn ein Signal auf dem eingestellten Kanal empfangen wird, schaltet die Weiche um und leitet die Kugel auf einen anderen Weg.

Lever: Bausteine wie der Lever verfügen zusätzlich über einen autonomen Modus. In diesem Modus (weiße LED) startet der Lever automatisch, wenn eine Kugel ihn erreicht, und katapultiert sie nach oben. Danach stellt er sich selbst zurück. Im Empfänger-Modus (Rot, Grün oder Blau) wartet der Lever auf ein Signal auf dem eingestellten Kanal, um sich zurückzustellen. Das bedeutet, er katapultiert die Kugel erst, wenn ein Signal empfangen wird.

Starter: Der den Anfang einer Bahn markierende Starter-Baustein funktioniert ebenfalls als Empfänger. Wenn ein Signal auf dem eingestellten Kanal empfangen wird, gibt der Starter eine Kugel aus dem Magazin frei.

Signale senden

Als Senderelement bietet GraviTrax zum einen eine App-Steuerung an, wobei die App die Signale über das GraviTrax Power Element Connect per Bluetooth überträgt. Ravensburger unterstützt darüber hinaus auch eine Python-Bibliothek sowie eine GUI, um den Controller über Python etwa von einem PC aus zu steuern.

Eine weitere Möglichkeit bietet der Element+ Controller, der über einen roten, grünen und blauen Taster jeweils ein Signal an die entsprechenden Kanäle schickt. Hier setzt unser Projekt an: Durch Simulation der Taster-Drücke über ein 3-V-Relais und Anschluss der Relais an einen ESP32 lassen sich die Kanäle beliebig steuern.

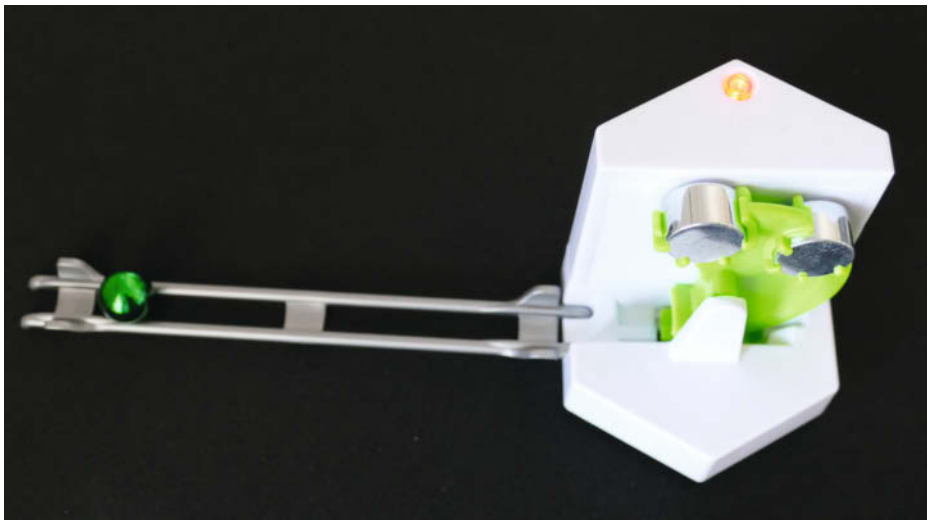


Bild 1: Der Lever-Baustein ist aktuell im autonomen Modus, was an der gelblich-weißen LED des Kanalwahl-Buttons zu erkennen ist.

Bild 2: Die bunten Buttons des Power Element+ Controllers senden ein Signal zu den drei Kanälen der entsprechenden Farbe.

GPIO-Pins zur Relaissteuerung

Das ESP32-DevKit (siehe Infokasten) hat 34 GPIO-Pins, aber nicht alle sind für die Relaissteuerung geeignet. Die nachfolgenden Pins bieten sich zur Steuerung an: GPIO 2, 4, 5, 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23, 25, 26, 27, 32, 33. Diese Pins sind in der Regel frei verfügbar und haben keine speziellen Einschränkungen. Dabei ist GPIO 2 auf vielen ESP32-Entwicklungsboards mit der internen LED verbunden, was zu Testzwecken sinnvoll sein kann. Die Relais ersetzen dabei die Taster-platten des GraviTrax Element+ Controllers und lassen sich über den ESP32 steuern. Im Beispiel haben wir GPIO 13 verwendet, um damit den grünen Kanal des Element+ Controllers zu triggern.

Umbau des Element+ Controllers

Der Controller-Baustein ist durch vier Triangel-Schrauben gesichert. Nach Entfernen der Schrauben lässt sich das Gehäuse öffnen. Ein leichter Druck genügt, um die Platine aus der Halterung der seitlichen Klemmnasen zu heben. Achtung! Auch wenn man es schafft, den Controller beschädigungsfrei zu demontieren, erlischt der Gewährleistungsanspruch durch das Öffnen.

Die beiden dünnen Verbindungsdrähte, an denen die 1,5-V-Batterie-Stromversorgung der Platine hängt, neigen schon bei geringstem Zug zum Reißen. Falls das passiert, zeigt Bild 4, wo die Stromversorgung an der Platine anzulöten ist.

Die Standard-Taster bestehen aus Gummipuffern, in denen eine Metallplatte eingebettet ist. Auf Druck schließt diese die Kontaktfläche auf der Platine.

An den Druckkontakten lötet man die Verbindungskabel am Rand der Kontaktfläche an. Bei den seitlich der Kontaktfläche liegenden kleinen Ausbuchtungen ist die Gefahr am geringsten, die eigentliche Kontaktfläche durch Lötzinn kurzzuschließen. Im Idealfall schafft man es, die Lötpunkte an den Druckkontakten so zu platzieren, dass die Buttons weiterhin

Bild 3: Die Demontage und der Zusammenbau des Controllers sind auch für Laien einfach durchführbar.

Kontakt herstellen können und so optional zur programmierbaren Steuerung funktionieren.

Die Kabel kann man entweder durch eine zu bohrende Öffnung oder, wenn man die manuelle Steuerung nicht braucht, durch eine der Button-Öffnungen nach außen führen. Damit bleibt das Gehäuse unbeschädigt. Vor dem Anschluss an das Relais kann man die beiden Kabelenden als Funktionstest verbinden. Im Controller sollte die zentrale Kontroll-LED in der Mitte der Taster aufleuchten, wie im Titelbild zu sehen. Auch beim manuellen Auslösen durch die Taster leuchtet die kleine LED zur Kontrolle auf.

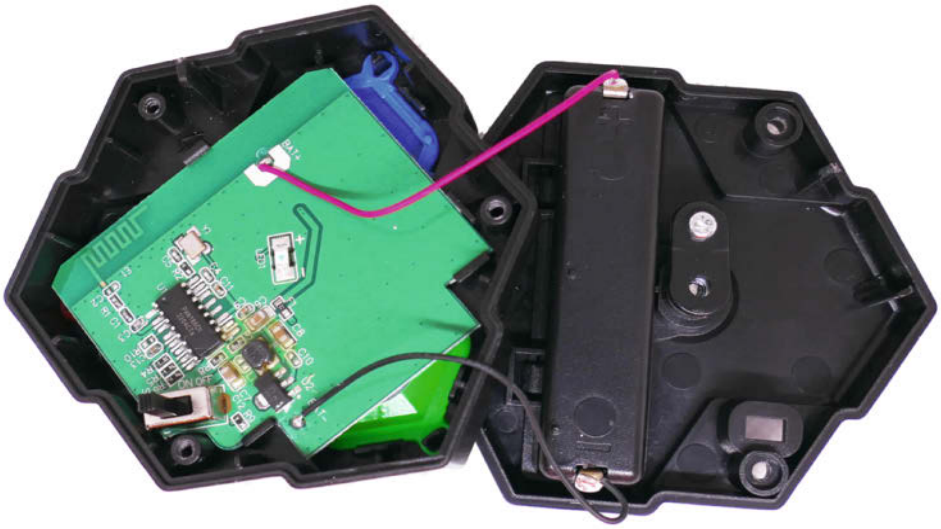


Bild 4: Die zur 1,5-V-Batterie führenden Kabel reißen schnell ab, lassen sich aber ebenso schnell wieder anlöten.

Bild 5: Bei Entfernung der Taster sollte man sich die Farbreihenfolge merken.

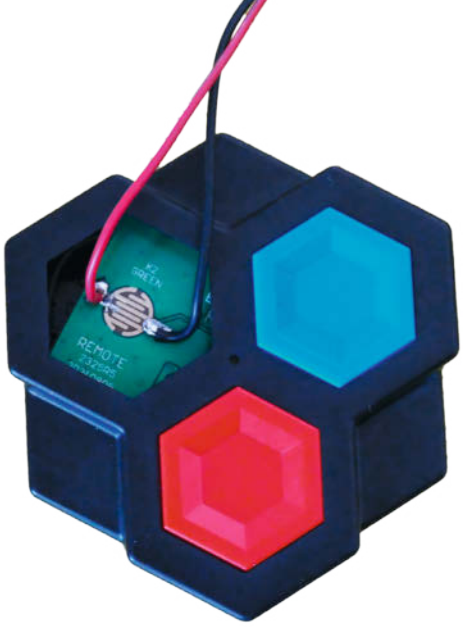


Bild 6: Mit ruhiger Hand und dem richtigen Werkzeug bekommt man vermutlich noch schönere Lötstellen als die hier abgebildeten. Wichtig ist, dass die Kontaktflächen frei bleiben.

Besteht Kontakt, sind die Enden entsprechend Bild 7 an das 3-V-Relais anzuschließen. Bei dem hier verwendeten Optokoppler-Modul mit 3-V-Relais-Power-Switch sind das die Kontakte NO (Normally Open) und COM. Ein Optokoppler bietet gegenüber einem einfachen Relais den Vorteil, eine elektrische Trennung zwischen dem Steuerkreis des ESP32 und dem geschalteten Kreis zu gewährleisten. Dies schützt den ESP32 vor Spannungsspitzen, Überspannungen oder Rückkopplungen, die in der geschalteten Schaltung auftreten könnten. Vom ESP32-Board gehen die Anschlüsse zu VCC und GND.

VCC wird an 3,3 V des ESP32 angeschlossen, GND wird an GND des ESP32 angeschlossen, und IN oder SIG (Steuerpin) wird an GPIO 13 des ESP32 angeschlossen. Der 3,3-V-Ausgang des ESP32 kann nur begrenzten Strom liefern. Normalerweise liegt die verfügbare Stromstärke modellabhängig zwischen 300 und 600 mA. Wenn das Relais-Modul mehr Strom benötigt und nicht einwandfrei schaltet, sollte man eine externe 3,3-V-Stromquelle verwenden. Zur Stromversorgung des ESP32 wurde eine USB-Powerbank genutzt.

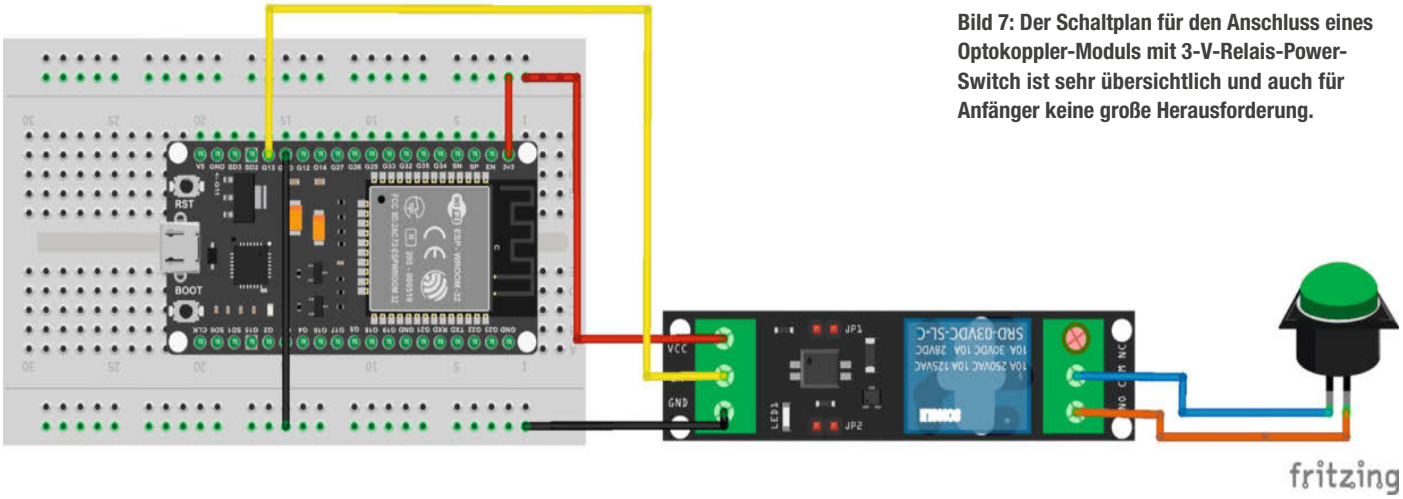


Bild 7: Der Schaltplan für den Anschluss eines Optokoppler-Moduls mit 3-V-Relais-Power-Switch ist sehr übersichtlich und auch für Anfänger keine große Herausforderung.

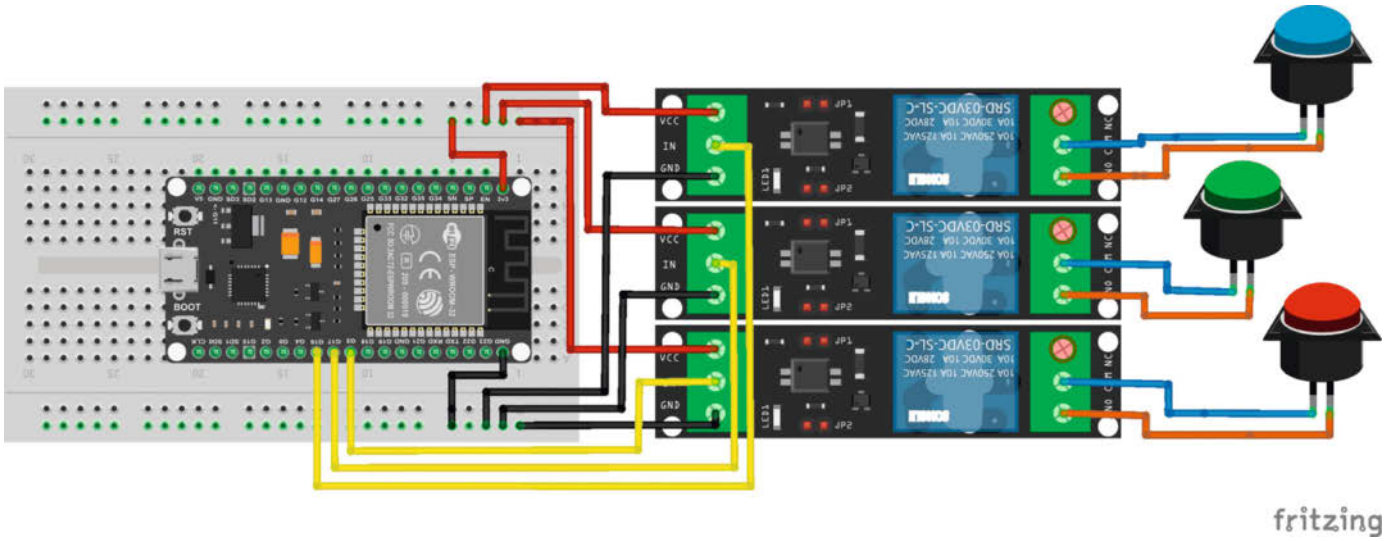


Bild 8: Der Schaltplan für den Anschluss von drei Relais zur Steuerung aller Kanäle ist nur unwesentlich aufwendiger als die Version für ein Relais.

ESP32-Relais-Steuerung

```
// Definieren des Relais-Pins
const int RELAY_PIN = 13;

// Variablen für den Relais-Status
bool relayState = false;

// Zeitintervall für das Umschalten (in Millisekunden)
const unsigned long INTERVAL = 5000;  // 5 Sekunden
unsigned long previousMillis = 0;

void setup() {
  // Serielle Kommunikation starten
  Serial.begin(9600);

  // Relais-Pin als Ausgang konfigurieren
  pinMode(RELAY_PIN, OUTPUT);

  // Relais initial ausschalten
  digitalWrite(RELAY_PIN, LOW);
  relayState = false;

  Serial.println("ESP32 Relais-Steuerung gestartet");
}

void loop() {
  // Aktuelle Zeit erfassen
  unsigned long currentMillis = millis();

  // Prüfen, ob das Intervall abgelaufen ist
  if (currentMillis - previousMillis >= INTERVAL) {
    // Relais-Zustand umschalten
    relayState = !relayState;

    // Relais schalten
    digitalWrite(RELAY_PIN, relayState ? HIGH : LOW);

    // Status ausgeben
    Serial.print("Relais Status: ");
    Serial.println(relayState ? "AN" : "AUS");

    // Zeitstempel aktualisieren
    previousMillis = currentMillis;
  }
}
```

Programm zur Ansteuerung

Der Sketch für den ESP32 steuert das 3,3-V-Relais an GPIO 13 und schaltet es alle 5 Sekunden ein und aus. Der Code definiert den Relais-Pin und den Status in Variablen und nutzt `millis()` für non-blocking Timing. Am Lever bewirkt das Einschalten, dass eine am Lever liegende Kugel auf eine höhere Ebene geworfen wird. Nur im Automatikmodus erkennt der Lever, ob eine Kugel zum Weitertransport bereitliegt oder nicht. Bei der Ansteuerung über einen der drei Kanäle muss das Timing entsprechend selbst angepasst werden, damit der Lever nur bei einer Kugel in Wartestellung aktiviert wird.

`setup()` startet die serielle Kommunikation, konfiguriert den Pin als Ausgang und schaltet das Relais initial aus. In `loop()` prüft der Code das Zeitintervall, schaltet den Relais-Zustand um, setzt den Pin entsprechend und gibt den Status seriell aus. Die Status-Ausgaben erfolgen über den seriellen Monitor der Arduino-IDE.

Wer alle 3 Kanäle über Relais steuern möchte, muss die Schaltung nur entsprechend erweitern. Wie das zu schalten ist, zeigt Bild 8. Das entsprechend erweiterte Sketch-Skript dazu ist im Downloadbereich zum Beitrag abrufbar.

Erweiterungsmöglichkeiten

Die Steuerung der Relais könnte auch etwas eleganter über eine App erfolgen. Wie sich so etwas ohne viel Aufwand machen lässt, zeigte der Beitrag „Arduino über App-Inventor-Apps steuern“ (siehe Infokasten). Die in dem Beitrag verwendete LED-Steuerung über ESP32 lässt sich mit geringfügigen Änderungen auch zur Kontrolle und Steuerung von Relais verwenden. *—usz*

Eine Lampe für Verliebte

In unserem Projekt „LoversLamp“ synchronisieren sich mehrere ESP-gesteuerte Lampen übers Internet per MQTT miteinander, um einer lieben, aber entfernten Person zu zeigen, dass man an sie denkt.

von Daniel Schwabe

Mittlerweile ist es klar: Die Make-Redaktion besteht aus unverbesserlichen Romantikern. Mit dem Flirt-Amulett in Ausgabe 7/25, Seite 34, und mit dem Herzschrittnehmer in dieser Ausgabe auf Seite 90 ist der Fall eindeutig: Wir lieben nicht nur Technik, sondern auch unsere Mitmenschen.

Für all die Leute, die über unsere Flirt-Amulette ihre bessere Hälfte gefunden haben, aber noch nicht zusammengezogen sind oder in einer Liebes- oder Freundesbeziehung über große Entfernung leben, haben wir die DIY-LoversLamp entwickelt

Mit dieser Lampe könnt ihr jederzeit sehen, ob eure Liebsten an euch denken. Wenn eine Person ihre Lampe einschaltet, geht die andere synchron an – egal, wo auf der Welt sie sich befindet.

Die Idee hinter der LoversLamp

Die LoversLamp ist eine smarte Lampe, die mit anderen identischen Lampen über das Internet verbunden ist. Wenn eine Person ihre Lampe einschaltet, wird diese Information an

Kurzinfo

» **Synchronisierung über MQTT**
» **Steuerung über Touchfunktion des ESP32**
» **Verbindung zum Internet über TLS/SSL**

Checkliste

Zeitaufwand:
3 Stunden

Kosten:
60 Euro

Material

» **ESP32**
» **LED-Streifen** Typ WS2812b
» **Plastikrohr** Durchmesser 27 mm
» **Lampenfuß**
» **Lampenschirm**
» **Kabel**

Werkzeug

» **Dremel**
» **Rohrzange**

Mehr zum Thema

» Dr. Stefan Recksiegel, Flirt-Amulette mit ESP-NOW, Make 7/24, S. 34
» Thomas Garaio, Weihnachtsstern mit Oxocard, Make 7/24, S. 50

Alles zum Artikel im Web unter
make-magazin.de/xyf6

MQTT

MQTT ist ein besonders leichtgewichtiges Kommunikationsprotokoll, das speziell für den Datenaustausch zwischen Geräten mit geringer Leistung und instabilen Netzwerken bzw. Netzwerken mit geringer Bandbreite entwickelt wurde. Es basiert auf dem Prinzip des Publish-Subscribe-Systems. Geräte können Nachrichten zu einer bestimmten Topic (Überschrift) auf einem Server veröffentlichen und auch bestimmte Topics abonnieren, um direkt die neuesten Nachrichten zu diesem Thema angezeigt zu bekommen. Durch seinen geringen Ressourcenverbrauch eignet sich MQTT ideal für Anwendungen im Internet der Dinge (IoT), bei denen Sensoren, Maschinen oder smarte Geräte miteinander kommunizieren müssen.

Je nachdem, wie wichtig sie sind, kann man gesendete MQTT-Nachrichten in drei Kategorien einteilen:

QoS 0 bis 1. Bei Level 0 wird eine Nachricht einmal gesendet und es findet keine Überprüfung statt, ob sie angekommen ist.

Stufe 1 überprüft den Eingang einer Nachricht beim Empfänger und sorgt dafür, dass sie bei Bedarf noch einmal gesendet wird, falls der Empfänger nicht antwortet.

Und Stufe 2 stellt sicher, dass die Nachricht genau einmal ankommt, indem es einen Handshake zwischen Sender und Empfänger gibt. Im Gegensatz zu Level 1 kann hier eine Nachricht nicht zufällig mehrfach gesendet werden.

Das Publish-Subscribe-Prinzip, kurz Pub/Sub, ist ein Kommunikationsmodell, bei dem Sender und Empfänger nicht direkt miteinander verbunden sind, sondern über eine zentrale Instanz, den Broker (hier HiveMQ), Nachrichten austauschen. Dabei gibt es zwei Hauptakteure: den Publisher, der Nachrichten veröffentlicht, und den Subscriber, der diese Nachrichten empfängt. Ein Publisher sendet seine Nachrichten an ein bestimmtes Thema, die sogenannte Topic, während ein Subscriber sich für ein oder mehrere Topics anmelden kann, um entsprechende Nachrichten zu erhalten. Der Broker übernimmt die Vermittlung, indem er eingehende Nachrichten an alle Subscriber verteilt, die die entsprechende Topic abonniert haben.

Ein wesentlicher Vorteil dieses Modells ist die vollständige Entkopplung von Publishern und Subscribern. Der Publisher muss nicht wissen, wer die Nachrichten empfängt, und der Subscriber muss nicht wissen, von wem sie gesendet werden.

Durch diese Trennung kann man problemlos weitere Geräte in ein Netzwerk einbauen, das über MQTT kommuniziert.

Im Falle der Lampe funktioniert das Ganze wie folgt:

Die Logik der Lampe läuft in einem Loop: Erst wird überprüft, ob lokal geschaltet wurde, danach, ob online ein neuer Status hinterlegt wurde. Die Lampe überprüft „Wurde ich lokal ein-/ausgeschaltet?". Ist das der Fall, schaltet die Lampe das Licht um und sendet an den MQTTBroker (HiveMQ) die passende Nachricht „ON" oder „OFF"). Danach überprüft die Lampe „Wie ist der letzte Status auf HiveMQ?". Wenn sich der vom lokalen Status unterscheidet, schaltet die Lampe auf den aktuell online gestellten um. Die Überprüfung geschieht auch, wenn die Lampe gerade erst einen neuen Status gesendet hat. Aber da dieser natürlich mit dem lokalen Status übereinstimmt, schaltet die Lampe nicht um.

Keine der Lampen kommuniziert selbst mit einer anderen oder verwaltet den Status des Lampen-Netzwerks. Dadurch können beliebig viele LoversLamps miteinander verbunden werden. Diese können dann natürlich auch verschiedene Formen annehmen.

Wie wähle ich einen öffentlichen MQTT-Server aus?

Bei der Auswahl eines öffentlichen MQTT-Brokers gibt es einige wichtige Kriterien, die beachtet werden sollten. Ein MQTT-Server spielt eine zentrale Rolle in der Kommunikation bei IoT-Projekten. Bei der LoversLamp werden zwar keine Staatsgeheimnisse übertragen, aber man sollte direkt mit der Best Practice anfangen und immer den Sicherheitsaspekt im Hinterkopf haben.

1. Sicherheit und Verschlüsselung

Ein öffentlicher MQTT-Server muss die TLS-Verschlüsselung (Transport Layer Security) unterstützen, damit die Datenübertragung zwischen einer Lampe und dem Server sicher bleibt. Ohne TLS könnten Angreifer die Nachrichten abfangen, manipulieren oder sogar unerwünschte Befehle an verbundene Geräte senden. Ein guter Broker bietet standardmäßig eine verschlüsselte Verbindung über Port 8883 an und stellt ein gültiges Root-Zertifikat zur Verfügung, mit dem sich der ESP32 sicher mit dem Server verbinden kann. Wie genau TLS funktioniert und was es mit diesem Root-Zertifikat auf sich hat, steht im Kasten TLS.

2. Datenschutz und Serverstandort

Da IoT-Geräte Daten aus dem heimischen Smart Home übertragen, muss man darauf achten, dass der MQTT-Server in einem datenschutzkonformen Land gehostet wird. Server innerhalb der EU unterliegen der DSGVO (Datenschutz-Grundverordnung), was sie zu einer besseren Wahl macht als Server in Ländern wie den USA oder China. Der hier verwendete Dienst HiveMQ ist ein Anbieter aus Deutschland und somit eine gute Wahl.

3. Verfügbarkeit und Stabilität

Ein guter MQTT-Broker sollte hohe Verfügbarkeit haben, damit die damit verbundenen Geräte jederzeit zuverlässig funktionieren. Ein instabiler Server, der häufig ausfällt oder lange Latenzzeiten hat, würde zu Verzögerungen oder sogar Kommunikationsabbrüchen zwischen Geräten führen. Bevor man sich für einen Anbieter entscheidet, sollte man daher prüfen, ob es Erfahrungsberichte über die Verfügbarkeit gibt oder ob der Anbieter selbst eine Statusseite mit Live-Informationen zur Erreichbarkeit bereitstellt.

4. Kosten und Nutzungslimits

Viele öffentliche MQTT-Server bieten eine kostenlose Nutzung an, haben aber Einschränkungen in Bezug auf Nachrichtenfrequenz, Verbindungsanzahl oder Datenmenge. Vor der Wahl eines Brokers sollte überprüft werden, ob die kostenlosen Limits für das eigene Projekt ausreichen. HiveMQ bietet beispielsweise in seiner kostenlosen Cloud-Variante bis zu 100 gleichzeitige Geräteverbindungen und maximal 10.000 Nachrichten pro Monat, was für die LoversLamp völlig ausreichend ist.

5. MQTT-Protokollversion

Die meisten modernen MQTT-Server unterstützen MQTT 3.1.1 oder MQTT 5.0. Während MQTT 3.1.1 für die meisten Anwendungen ausreicht, bietet MQTT 5.0 einige erweiterte Funktionen wie verbesserte Sicherheitsmechanismen und eine effizientere Datenübertragung.

alle verknüpften Lampen gesendet, die dann ebenfalls angehen.

Technisch basiert jede Lampe auf einem ESP32-Mikrocontroller, der als Steuerungseinheit für die jeweilige Lampe dient. Die Kommunikation erfolgt über das MQTT-Protokoll, das speziell für die Signalübertragung in IoT-Netzwerken entwickelt wurde.

Warum ein öffentlicher MQTT-Server?

Damit die LoversLamp über das Internet funktioniert, benötigen wir einen zentralen Vermittler, einen sogenannten MQTT-Broker. Dieser empfängt die Nachrichten der Lampen und verteilt sie an alle verbundenen Geräte.

Eine Möglichkeit wäre, einen eigenen MQTT-Server zu betreiben. Das würde jedoch bedeuten, dass man eine eigene Infrastruktur bereitstellen und Ports in der heimischen Firewall öffnen müsste – ein potenzielles Sicherheitsrisiko. Um dies zu vermeiden, nutzen wir einen öffentlichen MQTT-Broker. Nach welchen Kriterien man einen öffentlichen Server auswählen sollte, steht im Kasten „Wie wähle ich einen öffentlichen MQTT-Server aus?“.

In diesem Artikel wird HiveMQ verwendet. HiveMQ ist ein deutscher MQTT-Dienst, der eine kostenlose Cloud-Variante anbietet. Es ist eine zuverlässige und datenschutzkonforme Lösung, die perfekt für dieses Projekt geeignet ist. Mit HiveMQ kann man sicherstellen, dass die LoversLamp verlässlich und sicher funktioniert.

Anmeldung bei HiveMQ

Um HiveMQ nutzen zu können, muss man dort einen Account anlegen. Die Webseite dafür ist www.hivemq.cloud. Nach der Anmeldung muss man seinen „Nutzungsplan“ auswählen. Der kostenlose ist perfekt für so ein kleines Projekt geeignet. Man bekommt damit 100 Verbindungen, 10 GB Datenaustausch pro Monat und, ganz, ganz wichtig, MQTT über TLS, also mit Verschlüsselung.

Hat man sich angemeldet, bekommt man einen „Cluster Overview“ angezeigt. Hier findet man sowohl die personalisierten Server-URLs als auch die Ports, um MQTT zu verwenden. Beides muss später im Sourcecode nachgetragen werden.

Bevor man das allerdings macht, legt man noch Zugangsdaten für die LoversLamp-Clients an.

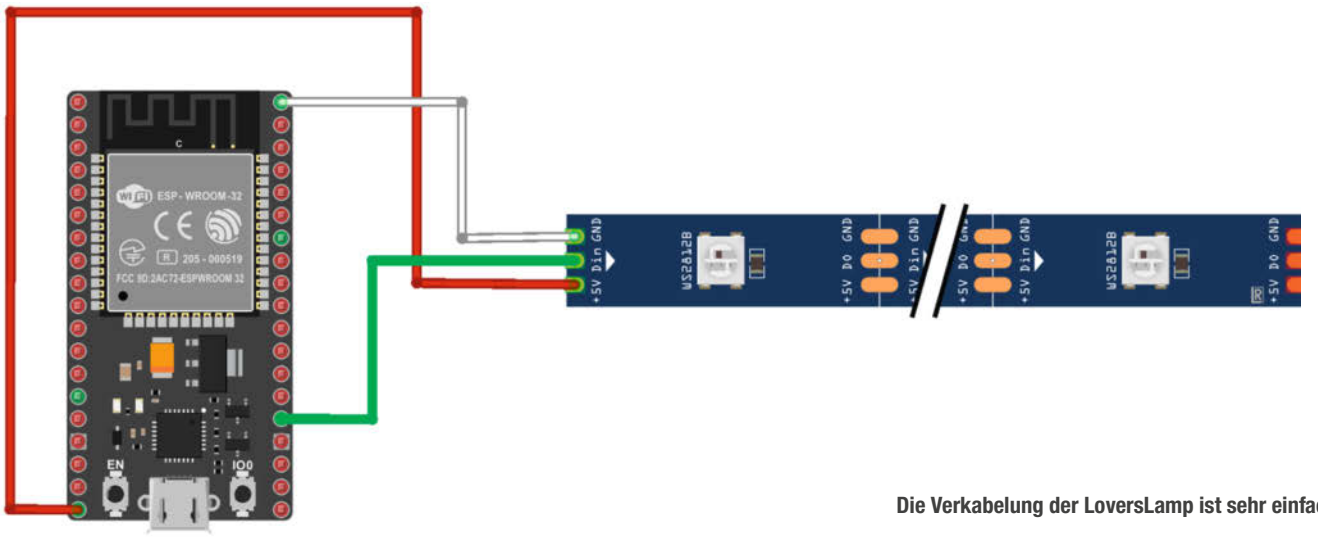
Dafür scrollt man auf der Übersichtsseite nach unten und öffnet das „Access Management“. Dort klickt man auf den großen gelben „Add new Credentials“-Button und wählt einen Usernamen, ein Passwort und setzt die Permissions auf „Publish and Subscribe“. Als Letztes dann auf den gelben „Save“-Button klicken.

Sicherheit durch TLS

Da die LoversLamp über das Internet kommuniziert, müssen wir sicherstellen, dass die gesendeten Daten vor Angriffen geschützt sind. Ohne Verschlüsselung könnten Dritte die MQTT-Nachrichten abfangen und möglicherweise sogar manipulieren. Hier kommt TLS (Transport Layer Security) ins Spiel, ein Standardprotokoll zur Verschlüsselung von Internetkommunikation.

TLS stellt sicher, dass die Datenübertragung zwischen dem ESP32 und dem MQTT-Broker verschlüsselt und sicher ist. Verbindet man sich mit einem Server mittels TLS, erfolgt als Erstes eine bestimmte Abfolge von Aktionen. Diese Aktionen bilden den sogenannten TLS-Handshake:

1. Client Hello: Der ESP32 schickt eine Nachricht an den Server, in der er angibt, welche Verschlüsselungsmethoden er unterstützt. Außerdem wird eine zufällige Zahl generiert, die später für die Verschlüsselung genutzt wird.
2. Server Hello: Der MQTT-Server antwortet mit seiner eigenen Liste unterstützter Verschlüsselungsmethoden und sendet sein SSL-Zertifikat, das von einer vertrauenswürdigen Zertifizierungsstelle (CA) signiert wurde.
3. Zertifikatsprüfung: Der ESP32 überprüft, ob das Zertifikat des Servers gültig ist, indem



Die Verkabelung der LoversLamp ist sehr einfach.

er es mit einer gespeicherten Liste vertrauenswürdiger Zertifikate vergleicht. Falls das Zertifikat nicht stimmt, wird die Verbindung abgelehnt.

4. Schlüsselaustausch: Der ESP32 und der Server erzeugen nun mit mathematischen Verfahren einen gemeinsamen geheimen Schlüssel aus dem Public Key des Servers. Dieser Schlüssel wird für die spätere Datenverschlüsselung genutzt.
5. Datenverschlüsselung beginnt: Ab diesem Punkt werden alle Daten verschlüsselt übertragen, sodass niemand die Nachrichten mitlesen oder verändern kann.

Ohne TLS könnten Angreifer in öffentlichen oder unsicheren Netzwerken die Datenpakete der LoversLamp mitlesen und sogar manipulieren. Ein Hacker könnte beispielsweise falsche MQTT-Nachrichten senden und die Lampen nach Belieben ein- oder ausschalten. Mit TLS ist die gesamte Kommunikation verschlüsselt und

Wichtige Stellen im Code

```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <PubSubClient.h>
#include <Adafruit_NeoPixel.h>


// Anzahl der LEDs und Pin definieren
#define LED_PIN 2       // GPIO-Pin für Datenleitung
#define NUM_LEDS 300      // Anzahl der LEDs im Streifen


Adafruit_NeoPixel strip(NUM_LEDS, LED_PIN, NEO_GRB +
NEO_KHZ800);


// WLAN-Zugangsdaten
const char* ssid = "xxx";
const char* password = "xxx";


// MQTT HiveMQ Broker (Public Broker von HiveMQ oder
eigener Server)
const char* mqtt_server = "xxx";  // Ändere auf den
eigenen HiveMQ-Broker.
const int mqtt_port = 8883;  // TLS-Port für MQTT


// MQTT-Zugangsdaten
const char* mqtt_user = "esp32";
const char* mqtt_password = "xxx";

// MQTT-Topic
const char* mqtt_topic = "esp32/test";


// TLS-Zertifikat (Root CA für HiveMQ oder eigenen
Server)
const char* root_ca = R"EOF(
-----BEGIN CERTIFICATE-----
xxx
-----END CERTIFICATE-----


)EOF";


// WiFi- und MQTT-Clients erstellen
WiFiClientSecure espClient;
PubSubClient client(espClient);
```

In den Zeilen 13 und 14 wird die WLAN-Verbindung eingerichtet mit dem Netzwerknamen und dem Passwort. Beides muss vor dem Flashen auf den ESP geändert werden.

In den Zeilen 17 bis 25 sind die Daten für MQTT hinterlegt; sowohl die URL für den Server und dessen Port als auch die Login-Daten für den ESP-Client und die Topic, zu der Daten gesendet und empfangen werden sollen.

Zeile 28 bis 32: Im richtigen Code befindet sich hier das Root-Zertifikat, das bei der TLS-Verbindung als Referenz verwendet wird. Das muss das Zertifikat von HiveMQ sein. Den Link dazu findet man in der Kurzinfo.

controlWLED

```
void controlWLED() {
    if (ledState) {
        for (int i = 0; i < NUM_LEDS; i++) {
            strip.setPixelColor(i, strip.Color(255, 0, 0)); // Rot setzen
            }
            strip.show();           // Farben anzeigen
    } else {
        for (int i = 0; i < NUM_LEDS; i++) {
            strip.setPixelColor(i, strip.Color(0, 0, 0));
          }
          strip.show();
    }
}
```

In Zeile 4 dieses Codes kann man die Farbe der LEDs im eingeschalteten Zustand ändern. Aktuell sind sie auf Rot eingestellt. Über ein Webtool kann man sich die **R**(ed)**G**(reen)**B**(lue)-Werte für verschiedene Farben erzeugen lassen. Der Link dazu befindet sich in der Kurzinfo.

abgesichert, sodass nur der ESP32 und der HiveMQ-Server die Nachrichten entschlüsseln können.

Wie richten wir TLS für die LoversLamp ein?

Um TLS mit dem ESP32 zu nutzen, benötigen wir das Root-Zertifikat des MQTT-Servers. Dieses Zertifikat dient dazu, den Server zu identifizieren und sicherzustellen, dass wir uns mit dem richtigen Broker verbinden. Daneben steckt in dem Zertifikat der öffentliche Schlüssel, mit dem später der Schlüsselaustausch für TLS geschützt wird.

HiveMQ stellt ein öffentliches Root-Zertifikat zur Verfügung, das wir in unseren Code einbinden müssen (Link dazu in der Kurzinfo). Der ESP32 nutzt dann WiFiClientSecure, um eine verschlüsselte Verbindung zum MQTT-Server herzustellen. Wieso WiFiClientSecure benötigt wird, steht im Kasten „Was ist WifiClientSecure?“.

Hardware

Die Basis des Projektes bildet ein ESP32. Dieser steuert als Leuchtmittel einen WS2812-LED-Streifen. Um die Lampe zu bedienen, wird kein Taster verwendet, sondern ein Touchpin des ESP32 – genauer gesagt Pin G0. Details dazu stehen im Kasten „Touch Me – Wie funktioniert der Touchsensor“.

Verkabelt werden die LEDs wie in der Grafik zu sehen. Versorgt wird der Streifen direkt mit 5 Volt über den ESP. Der „+5V“-Anschluss des LED-Streifens kommt an einen 5-Volt-Pin des ESPs und der GND-Anschluss wird mit dem GND des ESP verbunden. Der Datenanschluss „Din“ kommt an Pin G2.

Wie funktioniert der Touchsensor?

Das Sensorkonzept, das hier verwendet wird, nennt man kapazitiven Touch. Das ist die gleiche Technologie, die auch bei Smartphones für den Bildschirm verwendet wird.

Der ESP32 behandelt einen seiner Pins wie einen Kondensator und misst Schwankungen in der Kapazität (die Fähigkeit, elektrische Ladung zu speichern) dieses Kondensators. Berührt man den Pin, ändert sich dieser Kapazitätswert. Diese Änderung erkennt der ESP32 und kann daraufhin eine Aktion ausführen.

Wir bauen die Lampen

Die oben beschriebene Technik kann jetzt auf verschiedene Art und Weise in eine Lampe umgebaut werden. Zum Beispiel kann man die LED-Kette direkt an die Wand kleben oder an ein Möbelstück.

Für diesen Artikel wird die LoversLamp als Tischlampe umgesetzt. Diese Lampe besteht aus einem Holzfuß, einem Lampenschirm und einem Plastikrohr.

Als Erstes wird der Holzfuß vorbereitet. Dafür wird eine funktionierende Lampe auseinandergenommen. Diese Spenderlampe (Typ HITO-Tischlampe, verlinkt in der Kurzinfo) besteht aus dem Holzblock, der zum Fuß für die neue Lampe wird, einer eingelassenen 27er-Lampenfassung und einem Stromkabel. Die Lampenfassung wird zuerst ausgebaut. Dafür wird an der Unterseite des Holzblocks das weiche, rutschfeste Material vorsichtig abgezogen. Darunter befindet sich eine Vertiefung, in der das Stromkabel in die verbaute Lampenfassung übergeht. Die Lampenfassung ist mit einer Mutter festgeschraubt.

Jetzt schraubt man die Zugentlastung des Kabels (Kreuzschraube) ab, schneidet das Kabel nahe an der Mutter ab und zieht es aus dem Sockel heraus. Dann dreht man die Mutter mit einer Rohrzange von der Hohlachse herunter.

Jetzt kann man die 27er-Lampenfassung oben aus dem Sockel herausziehen. Damit ist der Sockel von aller alten Technik befreit.

Unter dem Gummi verbirgt sich eine kleine Einbuchtung.

Der ESP32 ist zwar schon ein kleines Board, aber trotzdem noch zu groß für das Sockelloch.

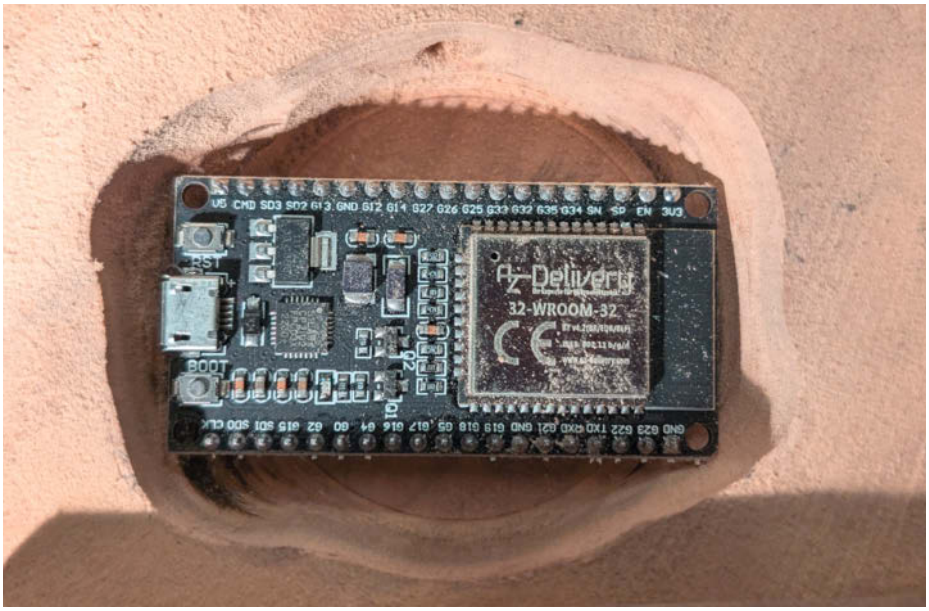
Was ist WifiClient-Secure?

WiFiClientSecure ist eine Erweiterung von WiFiClient, die TLS/SSL-Verschlüsselung unterstützt. Das bedeutet, dass die Verbindung zu einem Server über HTTPS (Port 443) erfolgen kann. Dadurch werden die Daten verschlüsselt übertragen, was wichtig ist, wenn sicherheitskritische Informationen, z. B. Passwörter oder API-Schlüssel, über das Internet gesendet werden.

WiFiClient ist für die Nutzung in einem LAN-Netzwerk geeignet. Diese Library kann keine verschlüsselten Serververbindungen aufbauen. Die Funktionen für TLS/SSL, die WiFiClientSecure bietet, fehlen. Wenn man aber über das Internet Daten versendet, sind die Sicherheitsfeatures von WiFiClientSecure wichtig, um die übertragenen Daten zu schützen.

Im Fall der LoversLamp überprüft eine Methode von WiFiClientSecure das Zertifikat des MQTT-Brokers, indem es das Serverzertifikat mit einem im Programm eingefügten vergleicht. Dadurch wird sichergestellt, dass das Programm mit dem richtigen Server verbunden ist. Dann kann die Übertragung der Daten, also der Zugangsdaten des Clients und der gesendeten Nachrichten, verschlüsselt geschehen.

Wie man in der Arduino IDE Bibliotheken installiert, wird in einem Artikel aus der Kurzinfo erklärt.



Mit einem spanenden Verfahren wird der Sockel ESP-kompatibel gemacht.

Die kleinen Klips an der Seite muss man vorsichtig aufhebeln. Sie dürfen nicht abbrechen.

Die Fassung brauchen wir noch. Sie muss aber angepasst werden.

In die nach unten zeigende Öffnung (in der man gerade die Mutter gelöst hat) kommt später der ESP32. Der ist allerdings etwas zu groß. Deshalb muss die Öffnung mit einem Dremel und einem 80er-Schleifpapier-Aufsatz vergrößert werden.

Wer das Projekt mit einem Wemos-D1-Mini nutzt, hat direkt viel Beinfreiheit im Sockel.

Sobald der ESP32 genug Platz im Sockel hat, kann man mit der Technik loslegen.

Das erwähnte Plastikrohr wird zum Hals der Lampe, um den man auch den LED-Streifen wickelt. Man beginnt damit, den LED-Streifen an einer Seite des Rohrs (im Weiteren „oben" genannt) eng herumzuwickeln.

Der LED-Streifen ist mit doppelseitigem Klebeband versehen. Damit der Streifen am Rohr hält, muss man nur den blauen Schutzfilm abziehen. Man umwickelt das Rohr bis zur Hälfte und schneidet überschüssige LEDs einfach mit einer Schere ab.

Am oberen Ende des Rohres hängen jetzt die Kabel, die für die Strom- und Datenverbindung als LEDs dienen. An diese Kabel lötet man jeweils 30 cm Klingeldraht an und schiebt die drei langen Kabel durch das Rohr. Zusätzlich führt man noch ein viertes Kabel in das Rohr. Dieses ist später für den Touchpin gedacht.

Nun legt man das Rohr erst mal beiseite und schnappt sich die Lampenfassung, die man vorhin aus dem Sockel ausgebaut hat. Diese Fassung hebelt man an den Seiten mit einem Schraubenzieher auf.

Dadurch erhält man zwei Teile: zum einen den Teil mit dem Gewinde und zum anderen den Teil mit der Metallstange, mit der die ganze Fassung am Sockel befestigt war.

An der Seite mit dem Gewinde muss man jetzt ein Loch durchbohren. Da kommen nämlich später wieder Kabel durch.

Jetzt fädelt man alle Teile wie bei einer Kette auf: die an den LED-Streifen angelöteten

Die Fassung passt genau auf das Rohr.

Kabel durch das Rohr, dann durch die durchbohrte Fassung (das Gewinde muss zum Rohr zeigen), dann durch den zweiten Teil der Fassung durch das Metallrohr mit dem Gewinde und als Letztes durch den Holzsockel.

Sind alle Kabel richtig durchgefädelt, drückt man die zweigeteilte Lampenfassung wieder zusammen und schraubt sie wieder in den Holzsockel.

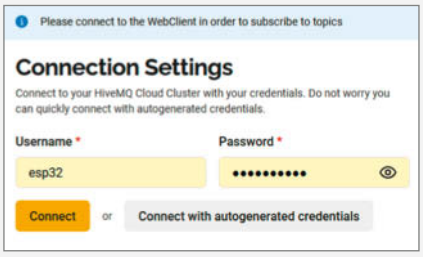
Ist das geschehen, drückt man das Rohr mit den LEDs in die Lampenfassung. Hierbei muss man ein bisschen sanfte Gewalt anwenden. Hält das Rohr nicht, kann man auch etwas Heißkleber zu Hilfe nehmen.

Auf der Unterseite des Sockels verkabelt man den ESP32, legt ein Stromkabel durch das Loch, durch welches das Stromkabel der alten Lampe ging, und klebt das rutschfeste Material wieder fest.

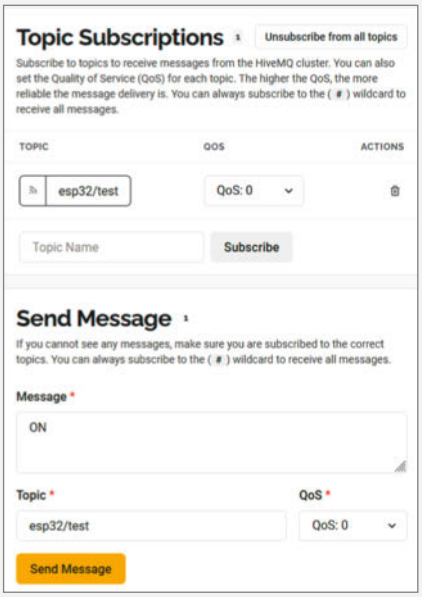
Jetzt ist der Lampenschirm dran. Für den gibt es in der Kurzinfo verlinkt eine Halterung

Toller Test-Tipp

Wenn man testen möchte, ob die Lampe richtig funktioniert und wirklich bei einem geänderten Zustand (der über MQTT empfangen wird) umschaltet, kann man das über HiveMQs Webclient probieren. In der HiveMQ-Übersicht klickt man dafür oben in der Leiste auf „Web Client“. Dort meldet man sich dann zuerst mit den oben angelegten Zugangsdaten an und kann dann Nachrichten unter einer Topic veröffentlichen. Die Topic im Code dieses Projektes ist „esp32/test“.

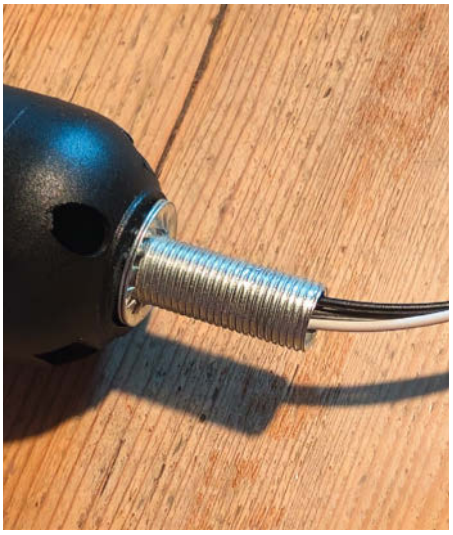


Unter „Connection Settings“ meldet man den Webclient mit den Anmeldedaten an, die man festgelegt hat.

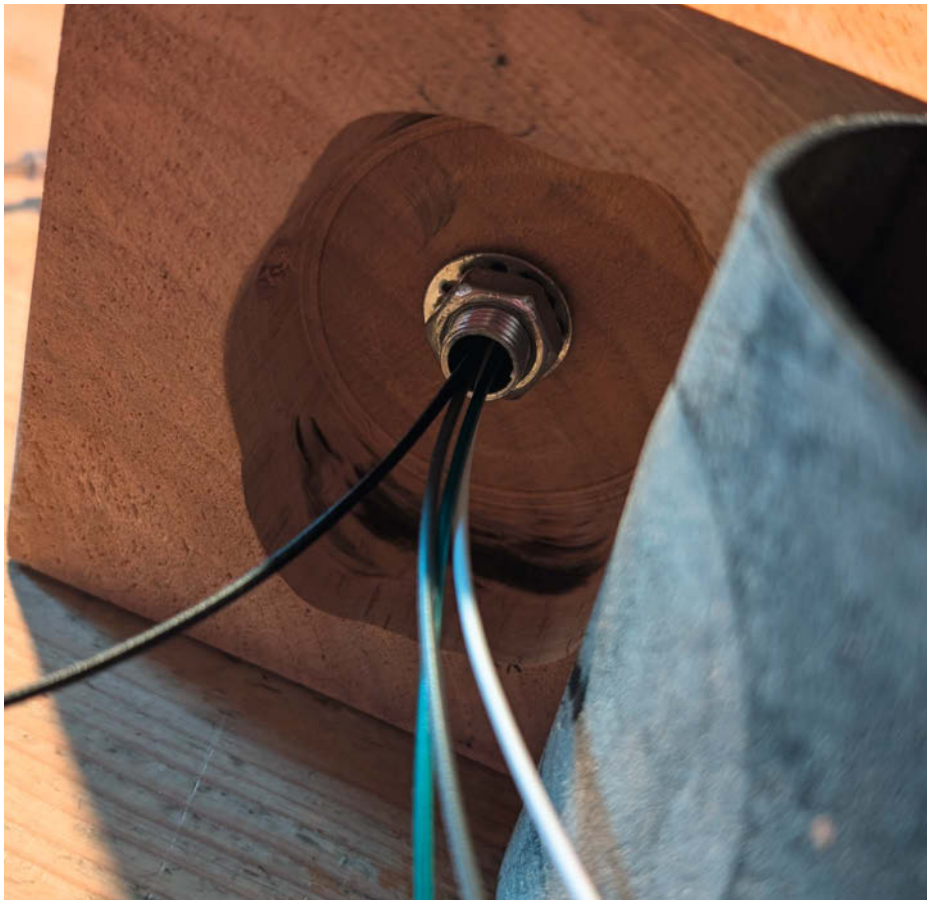


Danach kann man unter einer bestimmten Topic Nachrichten senden.

Die Halterung ist im Endeffekt ein flacher Teller.



Das kleine Kabelrohr an der Lampenfassung ist groß genug für alle Kabel.

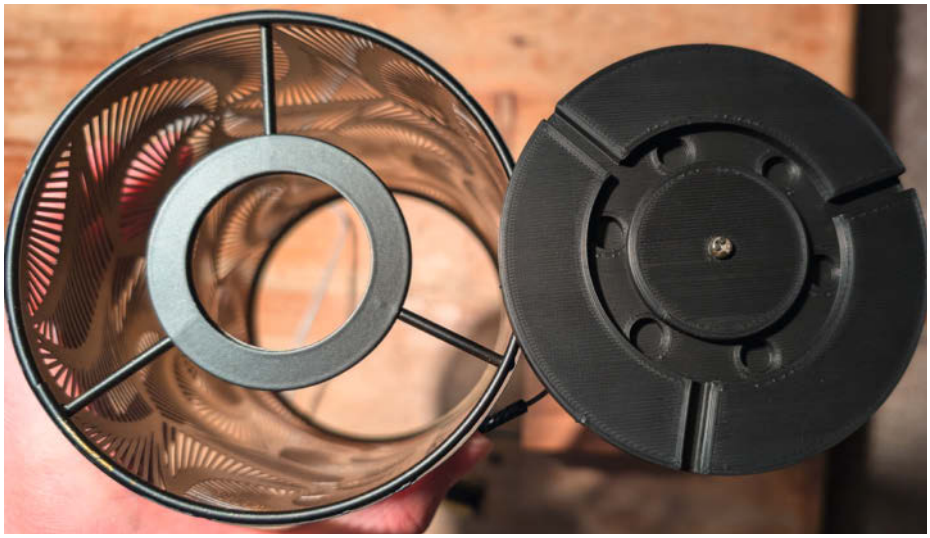


Und so enden alle Kabel in der erweiterten ESP-Kuhle.

zum 3D-Drucken. Diese Halterung passt genau zu einem bestimmten Lampenschirm, der ebenfalls in der Kurzinfo verlinkt ist, und muss aus leitfähigem PLA gedruckt werden, damit man später die Oberseite der Lampe berühren kann, um sie anzuschalten.

In die Halterung klebt man zuerst ein paar kleine Magnete ein (siehe Bild). Sie halten später den Lampenschirm fest. Dann schraubt man auf der Unterseite der Halterung das lose Kabel für den Touchpin des ESPs mit einer Schraube so ein, dass es Kontakt mit dem Plastik der Halterung hat. Nun setzt man die Halterung auf das Rohr an der Oberseite auf und legt den Lampenschirm auf die Magnete.

Jetzt ist die LoversLamp fertig. Wer noch niemanden für eine zweite Lampe hat, wird mit seinen Maker-Fähigkeiten bestimmt bald jemanden finden! *—das*



Die Halterung sitzt ganz oben auf dem Rohr.

KI-Katzentoilette

Künstliche Intelligenz (KI) zielt darauf ab, gefährliche oder monotone Arbeiten zu erleichtern. Das kann auch die korrekte Probenahme von Katzenkot der richtigen Katze sein. Dieses Projekt demonstriert den Einsatz von kostengünstigen ESP32-CAMs und KI-gesteuerter Objekterkennung zur Überwachung der Klonutzung. Bei Benutzung durch die kranke Katze wird eine Benachrichtigung via Home-Assistant gesendet. Das entwickelte System lässt sich zudem leicht für andere Anwendungen modifizieren.

von Florian Klug-Göri

Wer Haustiere hat, der weiß, dass sich Diagnose und Behandlung der diversen Wehwehchen nicht grundsätzlich von denen seiner Besitzer (oder Untergebenen?) unterscheiden. Allerdings kann eine Katze nicht sagen, wo es weh tut. Ein wichtiger diagnostischer Baustein, um eine Krankheit zu ermitteln, sind daher Stuhlproben. Deshalb stand ich als Verantwortlicher für die Katzenklo-Entleerung bei vier felinen Mitbewohnern immer wieder vor dem Problem: „Welches Gacksi gehört zu welchem Katzi?"

Anfänglich verbaute ich Bewegungsmelder in den Kisterln (österreichisch für Katzenklo), sodass ich immer verständigt wurde, sobald eine Katze ihr Geschäft verrichtete und ich zeitnah die Kotprobe entnehmen konnte. Dieses System hatte aber mehrere Nachteile. Zum einen merkt man so erst, wie oft vier Katzen im Laufe eines Tages das Katzenklo benutzen, meistens aber nur Pipi machen. Aber es kam auch zu vielen Fehlalarmen, da sich eine unserer vierbeinigen Freundinnen offenbar für die Kontrolle der Toiletten verantwortlich fühlt und mehrmals täglich allen Örtchen einen kurzen Kontrollbesuch abstattet. Zusammenfassend kann man sagen, die umsonst zurückgelegten Meter bei den Menschen sammelten sich an.

Zur Lösung dieses Problems entwickelte ich den „Sophisticated Helper Identifying Cat Entities" – kurz SHICE. Die prinzipielle Funktionsweise ist schnell erklärt: In den Katzenklos montierte Kameras erfassen Fotos der Katzen, diese werden an einen Server geschickt, welcher die Bilder mittels künstlicher Intelligenz analysiert, die Katze identifiziert und abschließend eine Benachrichtigung auslöst.

Kurzinfo

- **Objekterkennung mit Azure Custom Vision**
- **ESP32-Kamera mit PIR-Sensor auslösen**
- **Lokale KI auf Raspberry Pi und Anbindung an Home Assistant**

Checkliste

Zeitaufwand:
ein Wochenende

Kosten:
ca. 20 Euro pro Kamera

Material

- **Raspberry Pi** als Server
- **ESP32-CAM-Modul**
- **HC-SR501** PIR-Sensor
- **High-Power-LED**
- **Widerstand** 10 Ω
- **N-Channel MOSFET** IRLZ44N
- **Lochrasterplatine**
- **Drähte**
- **Steckverbinder**
- **USB-Netzteil**
- **Schutzscheibe** transparentes Acryl, Polystyrol o. Ä., 1,8 mm

Werkzeug

- **Maker-Werkzeug** Schraubendreher, Säge, Cutter, Zangen …
- **3D-Drucker** für Kamera-Gehäuse
- **Lötkolben** und Lötutensilien

Mehr zum Thema

- Ákos Fodor, Von Arduino zur PlatformIO IDE, Make 1/25, S. 8
- Martin Siegmann, Mobil 3D-konstruieren, Make 6/24, S. 82
- Daniel Bachfeld, Docker für Raspberry Pi, Make 1/21, S. 92

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

Machine Learning

Da dieses Projekt meinen ersten Kontakt mit maschinellem Lernen darstellt, war ich anfänglich ziemlich überfordert von der Flut an neuen Konzepten und Fachbegriffen. Nach etwas Recherche stieß ich auf Microsofts Custom Vision (siehe Kasten „Azure und Custom Vision"), das mir einen einfachen Einstieg in die Welt der KI-gestützten Objekterkennung ermöglichte.

Dafür ist erst einmal ein kostenloser Microsoft-Account Voraussetzung. Nachdem man sich diesen zugelegt hat, loggt man sich bei Custom Vision ein und wird darauf aufmerksam gemacht, dass für die Nutzung des Dienstes auch ein Azure-Konto benötigt wird. Leider muss man dafür eine langwierige Registrierung inklusive Hinterlegung einer Kreditkarte durchlaufen (keine Sorge, es bleibt kostenlos). Hat man diese Hürde aber erst einmal genommen, kann man auf der Website von Custom Vision damit beginnen, ein neues Projekt anzulegen. Nach dem Klick auf „New Project" vergibt man einen Namen für das Projekt (Bild 1) und muss eine neue Custom-Vision-Ressource erstellen und auswählen.

Dabei kann eine beliebige Ressourcen-Gruppe (Bild 2), die man an dieser Stelle ebenfalls anlegen kann, verwendet werden. Als „Kind" (engl. für Art/Sorte) wählen wir beim Erstellen der Ressource „Custom Vision Training". Zu beachten ist, dass als „Pricing Tier" „F0" ausgewählt wird, welches die Gratis-Variante darstellt. Als Location habe ich mich für „Germany West Central" entschieden, womit

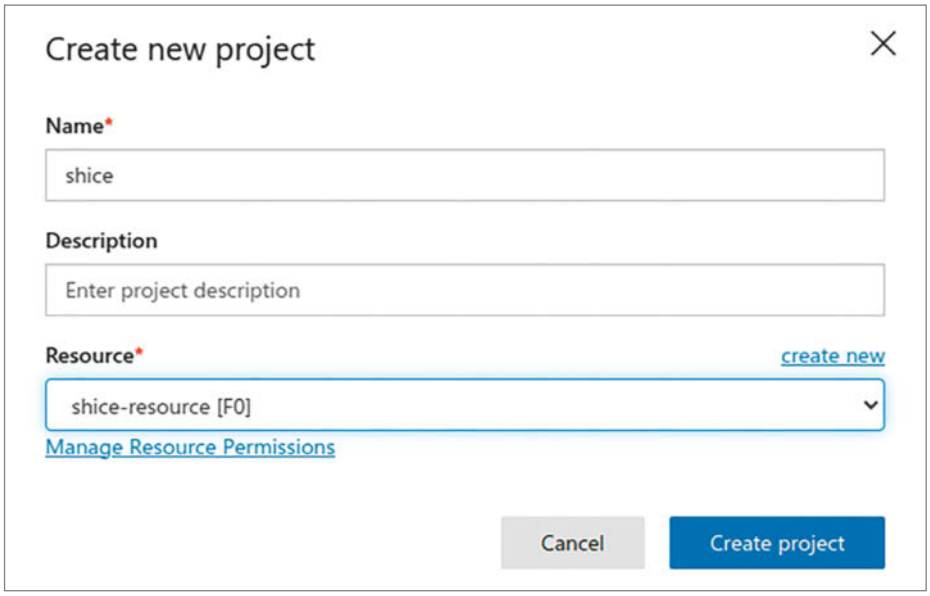


Bild 1: Ein neues Projekt in Custom Vision anlegen.

Azure und Custom Vision

Unter dem Namen „Azure“ fasst Microsoft sein Angebot an Cloud-Dienstleistungen zusammen. Sie bestehen aus mehr als 200 einzelnen Produkten. „Azure AI Custom Vision“ ist ein in dieser Cloud laufender Bilderkennungsdienst, den man mit eigenen Trainingsdaten füttern und mit dem damit erstellten KI-Modell Bilder nach Motiven klassifizieren kann. Dabei können entweder Bilder als Ganzes klassifiziert werden (z. B. Bild mit Katze, Vogel, Hund …) oder einzelne Objekte in einem Bild erkannt und deren Koordinaten zurückgeliefert werden. Für unsere Zwecke reicht die Klassifizierung aus, da uns nicht interessiert, wo genau eine Katze im Bild zu finden ist, sondern nur, ob überhaupt eine im Bild ist, und wenn ja, um welche genau es sich handelt.

Unsere Anforderungen erfüllt das kostenlose Preismodell völlig, bei dem man auf zwei Projekte mit jeweils 5.000 Trainingsbildern und einer Trainingsstunde pro Monat limitiert ist. Auch die Restriktion auf zwei Anfragen pro Sekunde sollte verschmerzbar sein. Dennoch muss man zur Registrierung leider eine Kreditkarte hinterlegen, die allerdings nicht belastet wird.

Mithilfe von Azure Ressourcengruppen können diverse Ressourcen (virtuelle Maschinen, Speicher, Dienste …), die die Azure Cloud anbietet, gebündelt verwaltet werden, um Verwaltung, Zugriffskontrolle und Abrechnung zu erleichtern – etwas, das vor allem für Organisationen relevant ist.

die Daten in Europa gespeichert werden sollten und unter hiesigen Datenschutz fallen.

Nach einem Klick auf „Create project“ passierte dann allerdings erst mal nichts. Zumindest war das bei mir so und ich nehme an, es handelte sich um einen Bug in der Oberfläche. Erst bei einem erneuten Klick auf „New Project“ erhielt ich ein Dialogfenster wie in Bild 3.

Immerhin ist nun die vorhin angelegte Ressource bereits vorausgewählt, den Namen des Projekts muss man allerdings erneut eingeben. Als Projekttyp wählen wir „Classification“, für „Classification Types“ „Multiclass (Single tag per image)“. Damit wird jedem Bild die wahrscheinlichste Kategorie zugewiesen. Als Domain stellen wir „General (compact) [S1]“ ein. „General“ steht in diesem Zusammenhang für allgemeine Erkennungsaufgaben, „compact“ für ein ressourcenschonendes Modell und „S1“ für ein Modell, dessen Antworten ohne Nachbearbeitung Verwendung finden können. Bei den „Export Capabilities“ genügen uns die „Basic Platforms“. Nach einem Klick auf „Create Project“ wird man automatisch auf die Projektseite weitergeleitet.

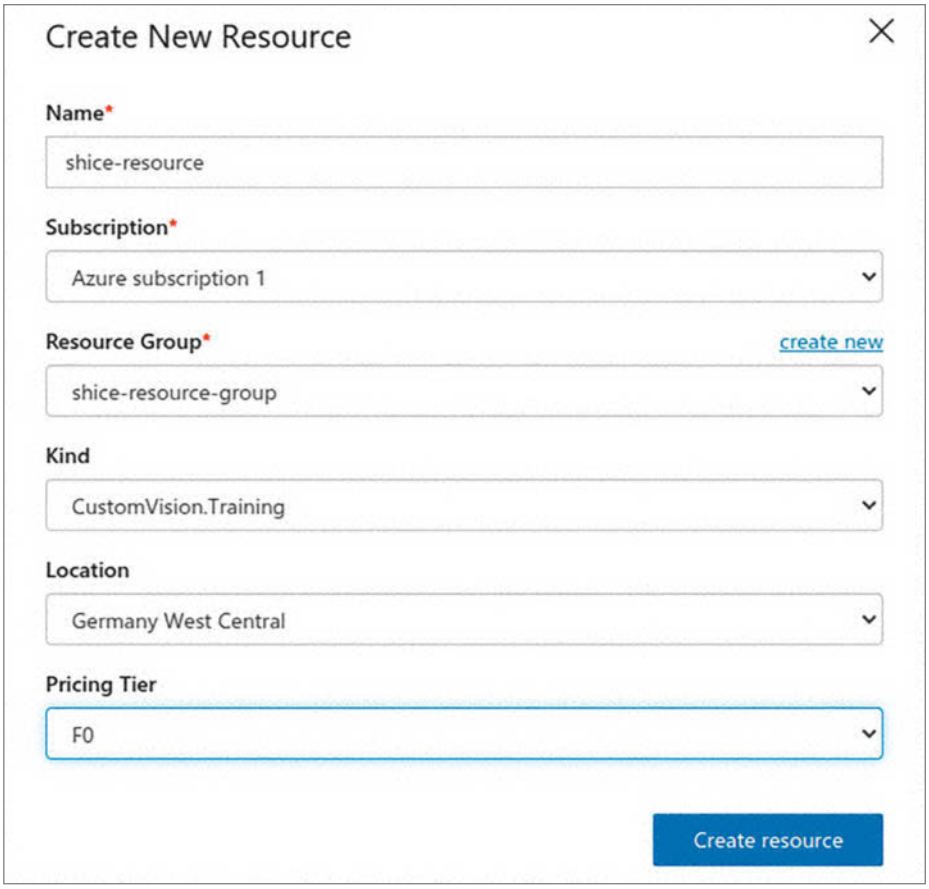


Bild 2: Jede Ressource muss einer Ressourcengruppe zugewiesen werden.

KI mit Bildern trainieren

Jetzt geht es weiter mit dem Upload der Trainingsdaten, indem man den blauen Button „Add images“ wählt. Für das erste Training wählte ich jeweils ca. 30 bereits vorhandene, mit dem Smartphone aufgenommene Bilder einer jeder unserer Katzen. Dabei sollte sich auf den Fotos die Katze gut von der Umgebung abheben, sich aber möglichst in Aufnahmewinkel, Beleuchtung, Pose usw. unterscheiden.

Für jede Katze wurden die Bilder getrennt hochgeladen und dabei der Name der Katze als „Tag“ (das eindeutige Kennzeichen) vergeben (Bild 4). Der bereits vorhandene Negative-Tag kann dafür verwendet werden, um Custom Vision beizubringen, auf welchen Bildern keine Katze zu sehen ist. Dies wird allerdings erst später fürs weitere Training benötigt (siehe gleichnamiger Abschnitt weiter unten).

Ist dieser Prozess abgeschlossen, sieht man in der linken Leiste die neu angelegten Tags nebst der Anzahl der Bilder (Bild 5). Nun ist die Zeit gekommen, das Training des Modells anzustoßen. Dies geschieht durch einen Klick auf den grünen Button „Train“ in der Menüleiste. Im folgenden Dialog wählt man „Quick Training“ aus und klickt dann auf „Train“. Nun ist es an der Zeit, sich eine Kaffeepause zu gönnen. Das Training erfolgt auf Microsofts Azure-Cloud und kann je nach Anzahl der Bilder schon einige Minuten in Anspruch nehmen.

Nach Abschluss des Trainings erhält man eine Übersicht, wie erfolgreich das Training war. Links oben kann man mit dem Schieberegler einen Schwellwert für eine Wahrscheinlichkeit einstellen, die zum Überprüfen des Trainingserfolgs anhand dreier Prozentzahlen „Precision“, „Recall“ und „AP“ genutzt wird. Dabei legt die eingestellte Wahrscheinlichkeit den Grenzwert fest, ab dem eine Erkennung als erfolgreich eingestuft wird.

- Precision: Prozentsatz an korrekten Klassifizierungen (Bsp.: Es wurden 100 „Tags“ mit Apfel vergeben, 99 waren tatsächlich Äpfel, die Präzision beträgt 99 Prozent.)
- Recall: Trefferquote über alle Klassifizierungen (Bsp.: Es gab 100 Bilder mit Äpfeln, davon wurden 80 als Äpfel identifiziert, eine Trefferquote von 80 Prozent.)

– (m)AP: statistisch die mittlere durchschnittliche Genauigkeit, Maßzahl für die Leistungsfähigkeit

Eine tiefgehende Betrachtung dieser Metriken findet man auf Microsofts Seiten zu „Characteristics and Limitations of Custom Vision“ (siehe Kurzinfo-Link).

Wenn man mit dem Ergebnis zufrieden ist, kann man das Modell zur lokalen Verwendung exportieren. Dafür wählt man im „Choose your platform“-Dialog, der sich nach einem Klick auf Export öffnet, „TensorFlow Android“, welches ungeachtet des Namens auch für nicht Android Umgebungen geeignet ist, stellt danach das Dropdown auf „TensorFlow Lite“ und verwendet den Export-Button. Nach kurzer Bedenkzeit bietet Custom Vision schließlich einen Button an, der den Download eines ZIP-Archivs anstößt. Darin enthalten ist das trainierte Modell, welches wir später der Server-Komponente hinzufügen werden.

Benachrichtigungen

Nach erfolgreicher Erkennung bedarf es natürlich auch einer Benachrichtigung, damit man vom Ereignis auch was mitbekommt. Diese erfolgt mittels eines „Webhooks“, sodass eigentlich alles, was einen HTTP-Request entgegennehmen kann, als Empfänger der Benachrichtigung infrage kommt. Ich habe mich naheliegenderweise dazu entschieden, dies meine bereits vorhandene Home-Assistant-Installation (HA) erledigen zu lassen. Damit die Endpunkte für die Benachrichtigungen zur späteren Verwendung im Arduino- (Kamera-Knoten) und Python-Code (Server) bereits bekannt sind, empfiehlt es sich deshalb, mit dem HA-Teil zu beginnen.

Das Endergebnis soll so aussehen: Wann immer eine Katzenerkennung registriert wurde, sollen unsere Google/Nest-Smartspeaker uns mitteilen, in welchem Kisterl welche Katze gerade ihr Geschäft verrichtet. Ich will außerdem direkt in HA die Benachrichtigung für jede einzelne Katze ein-/ausschalten können. Zusätzlich sollen Wiederholungen verhindert werden, sodass bei mehrmaliger Auslösung durch dieselbe Katze im selben Klo nur eine Durchsage erfolgt.

Um dies umzusetzen, habe ich zuerst ein paar Helfer in HA angelegt (Bild 6). Für jeden „Tag“ einen „input_boolean“-Helfer (Boolesche Eingabe), insgesamt also fünf Stück. Des Weiteren bedarf es jeweils eines Helferleins vom Typ „input_text“ (Texteingabe) und „input_number“ (Zahlenwert-Eingabe) sowie eines „timers“. Dieser erhält eine Laufzeit von einer Minute.

Insgesamt habe ich also sechs Helfer eingerichtet, für jede unserer vier Katzen einen „shice_tagX_aktiv“-Helfer (X = Katzennummer) sowie einen für den „Negative-Tag“, wenn keine Katze im Bild zu sehen ist. Die restlichen drei Helfer werden später in den Automatisierungen zum Abspielen der Benachrichtigungen verwendet.

Damit man die Benachrichtigungen für jede Katze im HA ein- und ausschalten kann, muss man im Frontend noch die entsprechenden Umschalter hinzufügen. Dafür klickt man rechts oben auf das Bleistiftsymbol (Dashboard bearbeiten) und anschließend unten auf „+ Karte hinzufügen“. Ich wählte eine „Entitäten“-Karte (Bild 7), da ich damit alle Toggles (Schalter) übersichtlich auf einer Karte versammeln kann. Nachdem man die entsprechenden, im vorigen Schritt angelegten „input_boolean“-Entitäten der Karte hinzugefügt hat, wird sie durch einen Klick auf „Speichern“ dem Dashboard hinzugefügt.

Danach wird eine Automatisierung angelegt, die als Trigger das Ablaufen des vorhin erstellten Timers erhält. Die zwei Aktionen dieser Automatisierung setzen die zwei Helfer `input_number.shice_letztes_kisterl` und `input_text.shice_letzter_tag` auf ihre Default-Werte zurück – nämlich `0` und die leere Zeichenkette `""`. Dies ist nötig, da diese zwei Helfer dazu dienen, mehrmaliges Abspielen derselben Benachrichtigung kurz hintereinander zu verhindern. Die riesengroßen Dialoge von HA und die Beschreibung der Aktionen finden Sie zur Referenz komplett im Onlineteil (siehe Link in der Kurzinfo).

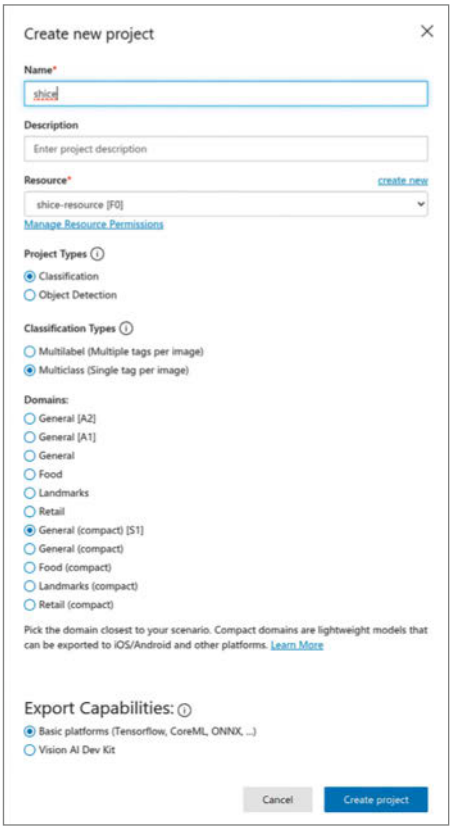


Bild 3: Der Dialog zum Konfigurieren des Projekts.

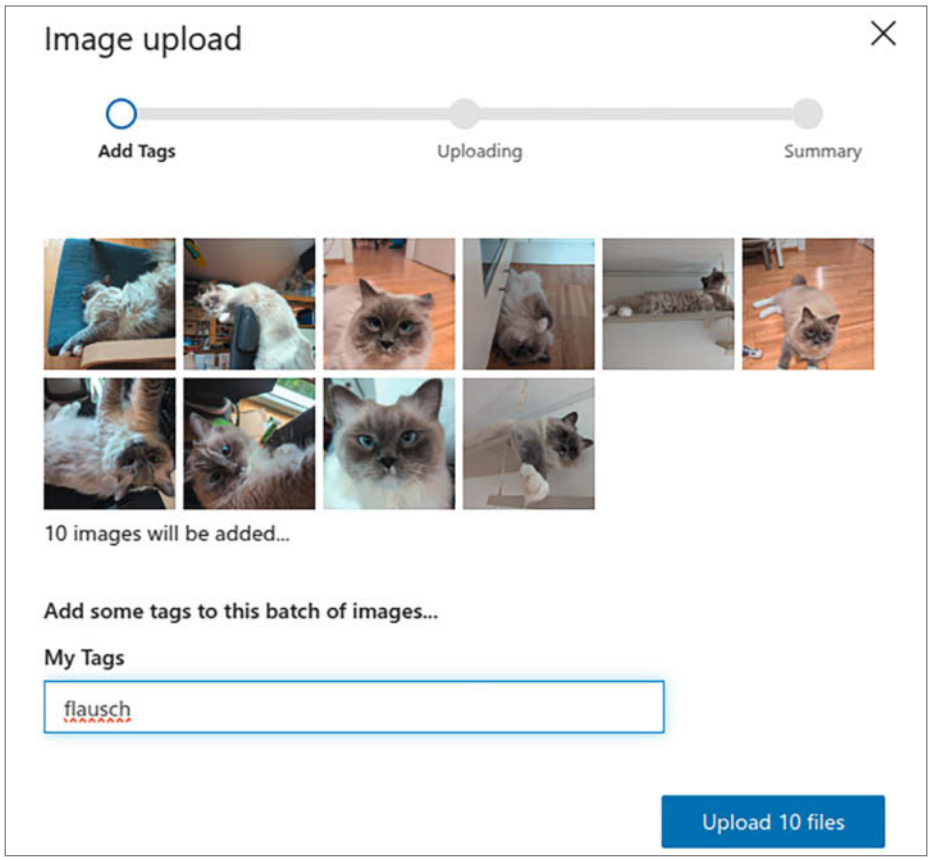


Bild 4: Bilder-Upload in Custom Vision.

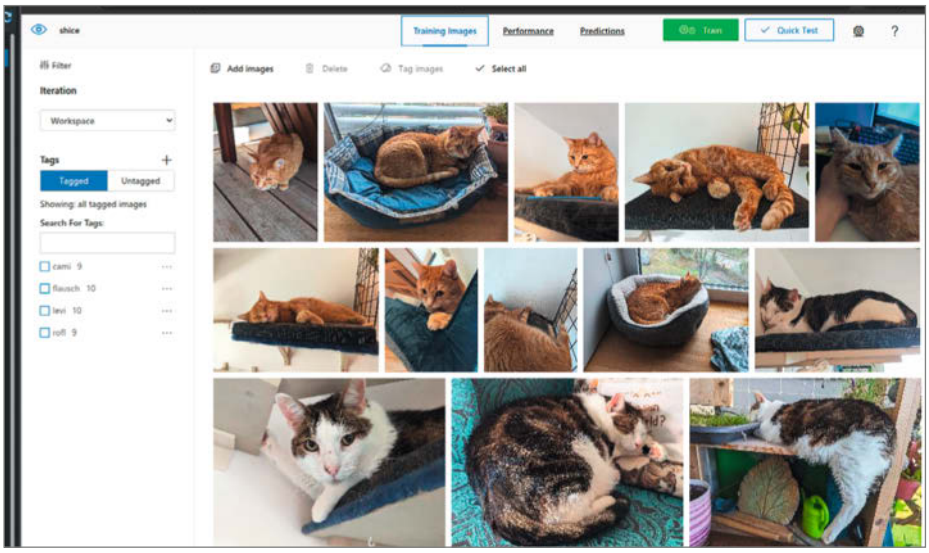


Bild 5: Nach dem Upload der ersten Trainingsbilder kann das Training mit den Daten angestoßen werden.

Diese Automatisierungen kann man nun von der Kommandozeile aus beispielsweise mithilfe des Tools „cURL" testen:

```
curl -k -X POST https://HA_ADDRESS:\
HA_PORT/api/webhook/WEBHOOK_ID_\
KISTERL
```

Wobei der Parameter `-k` cURL Zertifikatsfehler ignorieren lässt und `-X POST` eine „HTTP POST"-Anfrage veranlasst (GET-Anfragen werden erst seit HA 2023.5 unterstützt). Funktioniert die Automation, wird der Helfer `input_number.shice_letztes_kisterl` auf den in der Automatisierung hinterlegten Wert gesetzt und nach einer Minute wieder zurückgesetzt. Die Webhook-IDs notieren wir uns, diese benötigen wir später noch.

Die letzte und komplexeste Automation wird ausgelöst, sobald die Katzenerkennung abgeschlossen ist, und ist dafür zuständig, die entsprechende Durchsage auf den Smartspeakern anzustoßen. Als Trigger dient ebenfalls wieder ein HA-Webhook, welcher nach erfolgreicher Erkennung aufgerufen wird. Diesem wird der Name/Tag der Katze als POST-Parameter übergeben, damit jener in der Automatisierung als `trigger.data.label` zur Verfügung steht. Um Wiederholungen zu verhindern und die Benachrichtigung für jede Katze deaktivierbar zu machen, sind ein paar Bedingungen notwendig. Aus Platzgründen erkläre ich den Code ausführlich in der Online-Ergänzung zum Artikel (siehe Kurzlink). Am Ende steht eine Sprachausgabe der Benachrichtigung per Text-To-Speech auf Google/Nest-Smartspeakern.

Lokaler KI-Server

Als Server verwende ich meinen Raspberry Pi 4 mit Raspberry Pi OS, der bereits als Home Server seine Dienste tut. Grundsätzlich sollte sich jegliche Hardware, auf der Python 3 und „TensorFlow lite" laufen, dafür eignen.

Die Server-Komponente besteht im Wesentlichen aus zwei Python-Skripten: Das Skript `manage.py` ist eine Flask-Anwendung (siehe Kasten Flask), die einerseits das Webinterface und andererseits den Endpoint zum Upload der Bilder für die Kameras bereitstellt. Die hochgeladenen Bilder werden an `predict.py` weitergereicht, das die Objekterkennung mittels „Tensorflow Lite"-Framework – welches übrigens im September 2024 von Google in LiteRT umbenannt wurde – und entsprechend trainiertem KI-Modell umsetzt und abschließend eine Benachrichtigung auslöst (siehe auch „Mehr zum Thema"). Das Listing „Installation von SHICE" zeigt exemplarisch die Installation auf einem aktuellen Raspberry Pi OS.

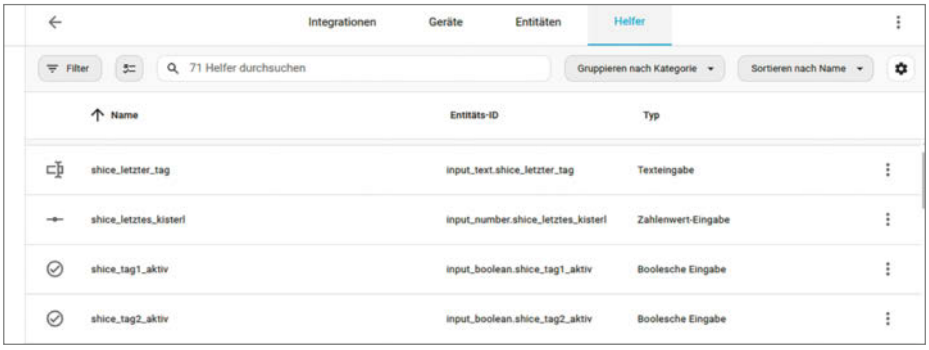


Bild 6: „Home Assistant Helfer" (gekürzt).

Nach dem Klonen des GitHub-Repositorys und dem Wechseln in das neue Verzeichnis wird in Zeile 3 eine virtuelle Python-Umgebung eingerichtet und in Zeile 4 (hier für Bash, für andere Shells bitte anpassen!) aktiviert. Zeile 5 installiert die benötigten Abhängigkeiten und die letzte erstellt die Datei `secret.py`, in der der aufzurufende Webhook eingetragen wird. Das soll jene URL sein, die wir vorhin aus der „Katze-erkannt"-Automatisierung gewinnen konnten. Sie ist im `echo`-Befehl anzupassen, kann alternativ aber jederzeit auch nachträglich in der Datei geändert werden.

An dieser Stelle kommt das aus Custom Vision exportierte ZIP-Archiv ins Spiel. Es extrahiert zu einem Ordner mit fünf Dateien. Den Ordner muss man in `tensorflow` umbenennen und am Server in das SHICE-Verzeichnis kopieren.

Damit sollte die Einrichtung der Erkennung abgeschlossen sein und man kann sie, nachdem man den Server mit `python manage.py` gestartet hat, einem Test unterziehen. Der Befehl

```
curl -F "imageFile=@test.jpg" \
SERVER_IP:5000/upload
```

lädt die Datei `test.jpg` auf den Server hoch. Anschließend kann man das Ergebnis unter `http://SERVER_IP:5000/list_tagged` begutachten.

Um den SHICE-Server beim Booten automatisch zu starten, habe ich eine Systemd-Init-Datei `/etc/systemd/system/shice.service` mit folgendem Inhalt angelegt:

```
[Unit]
Description=SHICE service

[Service]
Type=Simple
Restart=on-failure
RestartSec=3
User=USER
WorkingDirectory=/ARBEITSVERZEICHNIS
ExecStart=/ARBEITSVERZEICHNIS/venv/\
          bin/python3 manage.py

[Install]
WantedBy=multi-user.target
```

Die Platzhalter `USER` und `ARBEITSVERZEICHNIS` sind den eigenen Gegebenheiten anzupassen.

Mit folgenden Befehlen wird der Systemd-Service dem OS bekannt gemacht:

```
sudo chmod 0644 \
/etc/systemd/system/shice.service
sudo chown root:root \
/etc/systemd/system/shice.service
sudo systemctl daemon-reload
systemctl enable shice.service
```

Nach einem Neustart kann man mit dem Befehl `sudo systemctl status shice.service` überprüfen, ob der Server erfolgreich gestartet wurde.

Für diejenigen, die Docker bevorzugen und vielleicht bereits auf dem Raspi installiert haben, habe ich die sehr rudimentären Dateien `Dockerfile` und `docker-compose.yaml` ins Repository (siehe Kurzlink) gepackt. Damit ist es möglich, durch ein einfaches `docker compose up --build shice` im selben Verzeichnis einen Docker-Container zu bauen und zu starten, in dem die Software läuft. Standardmäßig wird auch hier der Port 5000 genutzt. Das kann allerdings ohne viel Aufwand in der `docker-compose.yaml` angepasst werden. Die dort ebenfalls zu findende Option

```
restart: always
```

sorgt dafür, dass der Docker-Container beim Booten des Systems ebenfalls gestartet wird. Mehr zum Umgang mit Docker findet man im Artikel „Docker für Raspberry Pi“ (Link in der Kurzinfo).

Kamera-Hardware

Als Kameras boten sich ESP32-CAM-Boards an, die ich noch in der Grabbelkiste hatte. Die PIR-Bewegungsmelder (Passive InfraRed) vom Typ HC-SR501, die auf die Bewegung von warmen Objekten reagieren, verrichteten bereits ihren Dienst bei dem Vorgängerprojekt und wurden wiederverwendet. Der Zusammenbau nach dem Schaltplan (Bild 8) auf einer Lochrasterplatine sollte den geübten Maker vor keine unlösbaren Aufgaben stellen und schnell erledigt sein. Für einen funktionierenden Kamera-Node reicht es, das ESP32-CAM-Board mit dem HC-SR501 zu verbinden und für Strom zu sorgen. Was in meinem Fall je ein altes USB-Netzteil übernimmt.

Bei unzureichender Beleuchtung, besonders nachts, kann optional eine Lichtquelle angeschlossen werden. Im Schaltplan ist diese mit der LED D1, dem Vorwiderstand R1 zur Strombegrenzung sowie dem N-Channel MOSFET Q1 zum Schalten realisiert. Natürlich ist es genauso gut möglich, statt der LED-Schaltung z. B. ein Relaisboard mit dem Mikrocontroller Pin anzusteuern.

Es hat sich als vorteilhaft herausgestellt, eine LED außen am staubdichten Gehäuse anzubringen, da eine interne LED – wie sie auf

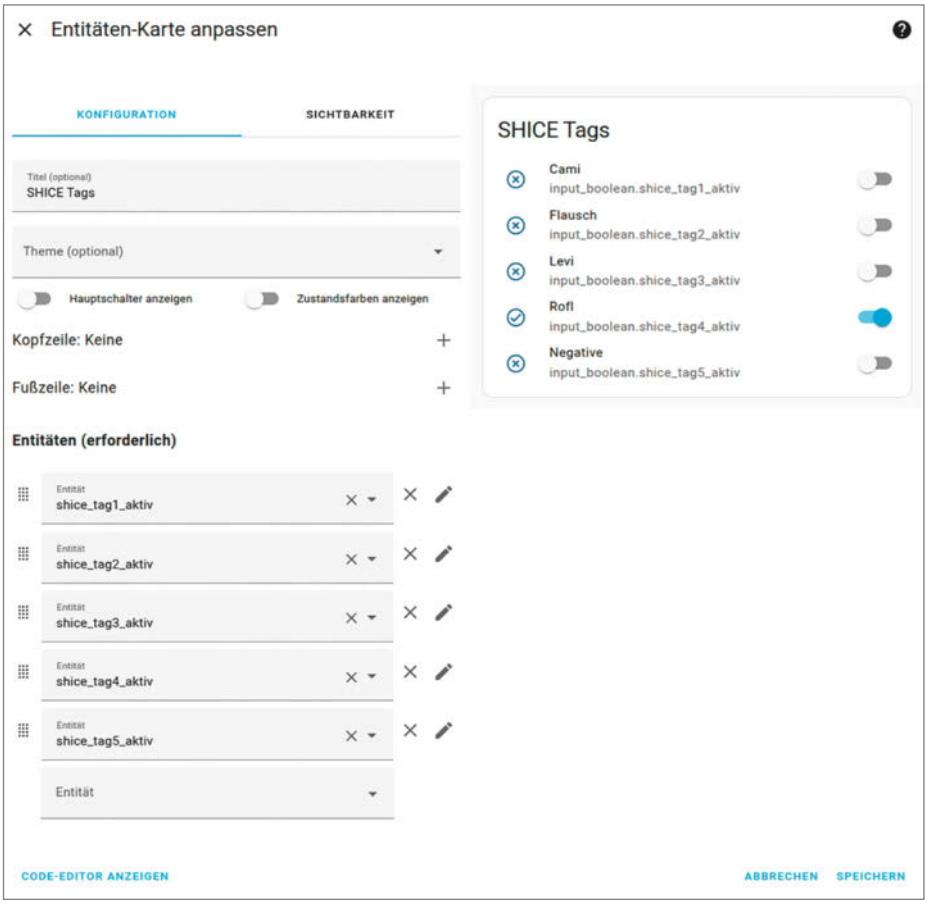


Bild 7: Entitäten-Karte zum Ein-/Ausschalten der einzelnen Benachrichtigungen.

den Kameramodulen vorhanden ist – zu Reflexionen und Überstrahlungen führte.

Für den HC-SR501 hat es sich aus Platzgründen angeboten, eine rechteckige Aussparung in die Lochrasterplatine zu sägen, die Sensorabdeckung von hinten durchzustecken und die Sensorplatine abschließend mit Heißkleber an der Rückseite der Lochrasterplatine zu fixieren (Bild 8).

Um die Kameras vor Staub zu schützen, sind für entsprechende Scheiben (75 × 40 × 1.8 mm) Nuten am Gehäuse vorgesehen (Bild 9).

Ein dafür passendes Material für die Scheiben zu finden, stellte sich allerdings als komplizierter als gedacht heraus (siehe Kasten „IR Transmission“). Die Anforderungen klingen erst mal nicht so anspruchsvoll: transparent im sichtbaren (für die Kameraaufnahmen) und im infraroten Bereich (für die Bewegungsmelder) sowie mit normalem Werkzeug bearbeitbar. Glas und Acrylglas fallen damit meines Wissens nach schon weg, da beide Infrarotlicht nicht durchlassen. Also versuchte ich mein Glück mit Polystyrolplatten. Leider arbeiteten die Bewegungsmelder damit unzuverlässig. Schlussendlich setzte sich der Pragmatismus durch und ich schnitt in die Scheiben an den Stellen, hinter denen sich die PIR-Sensoren

Flask

Flask ist ein in Python geschriebenes leichtgewichtiges Webframework, welches die Implementierung von Webservices erheblich erleichtert, indem es die dafür benötigten grundlegenden Funktionen (Routing, Request/Response Handling, Templating, etc.) bereitstellt.

Für eine minimale Webanwendung reichen folgende Zeilen:

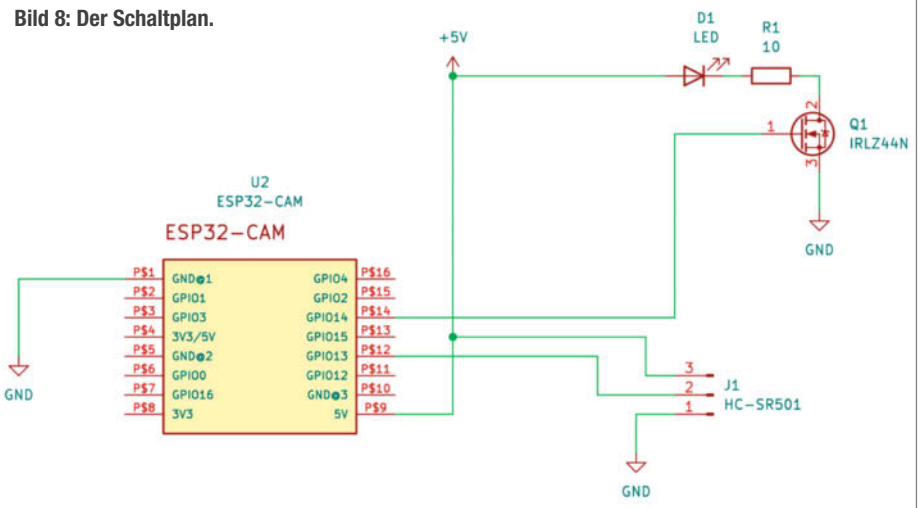
```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def home():
    return "Hello, World!"
if __name__ == '__main__':
    app.run(debug=True)
```

Damit erstellt man einen Endpunkt, der unter dem Root-Verzeichnis (/) erreichbar ist und „Hello, World!“ zurückliefert.

Installation von SHICE

```
git clone https://github.com/flurl/shice.git
cd shice/
python -m venv venv
source venv/bin/activate
pip install -r requirements.txt
echo "POST_PREDICTION_HOOK = \
"https://HA_ADDRESS:HA_PORT/api/webhook/WEBHOOK_ID_KATZE_ERKANNT\"" > secret.py
```

Bild 8: Der Schaltplan.



ESP32-CAM

ESP32-CAM-Boards gibt es zuhauf aus unterschiedlichsten Quellen zu erwerben. Leider ist aber ESP32-CAM nicht gleich ESP32-CAM. Abgesehen von Qualitätsunterschieden an sich und unterschiedlicher Bildqualität kann vor allem ein nicht vorhandener oder angeschlossener PSRAM (externer Arbeitsspeicher) ein Spielverderber sein. Der Mangel solch eines Speichers verhindert, dass man Bilder mit hoher Auflösung verarbeiten kann. Glücklicherweise ist das in diesem Projekt nicht von Belang, da die Bilderkennung mit Bildern in 800 × 600 Pixeln ausgezeichnet funktioniert. Ich habe damit alten ESP32-CAMs ohne PSRAM neues Leben einhauchen können.

Ein weiteres Problem, das ich immer wieder mit ESP32-Modulen habe, ist eine mangelnde WLAN-Reichweite. Bei den auf den ESP32-CAM-Boards verwendeten ESP32-S Modulen kann eine externe Antenne für Abhilfe sorgen. Allerdings ist es nicht damit getan, eine Antenne mit U.FL-IPX-Stecker mit dem Board zu verbinden, sondern man muss auch einen 0-Ohm-

Widerstand umlöten oder, wenn man das winzige SMD-Bauteil sofort nach dem Auslöten verliert, mit einem Lötzinnpatzen ersetzen.

Auf GitHub findet man eine Diskussion von Usern des „AI on the Edge Device"-Projekts, in der Erfahrungsberichte und Bezugsquellen geteilt werden (siehe Kurzlink).

befinden, Löcher und klebte sie mit transparentem Paketband ab, was offenbar genügend Infrarotstrahlung durchlässt, damit die Sensoren anschlagen (Bild 10). Einen Vergleich der Materialien, per Wärmekamera aufgenommen, finden Sie im Online-Ergänzungsteil.

An der Rückseite des Gehäuses befindet sich eine Durchführung für die Stromversorgung, für die optionale Beleuchtung sowie für eine externe Antenne. Um ein Eindringen von Staub zu verhindern, setzte ich eine Kabeltülle ins Loch ein (Bild 11).

Die 3D-Druckvorlagen stehen auf Printables zum Herunterladen bereit (siehe Link in der Kurzinfo). Dort findet man auch einen Link zum Onshape-Dokument, falls man das Design adaptieren möchte. Das kann nötig sein, wenn man die Breite der Nuten, die die Frontscheibe aufnehmen, anpassen muss (siehe Kasten „Onshape-CAD"). Um das Gehäuse möglichst staubdicht zu bekommen, habe ich mich nämlich möglichst genau an der Dicke meiner Scheiben orientiert.

ESP32-CAM-Firmware

Die Firmware für die Kameraknoten wurde von mir mit PlatformIO (kurz PIO) unter VSCode entwickelt (für eine Einführung siehe „Mehr zum Thema"). Nach dem Klonen des Projekts von GitHub kann man in VSCode mittels File/Open Folder… den Ordner `arduino` im soeben geklonten Verzeichnis öffnen. Nun muss die Datei platformio.ini durch einen Klick links im Datei-Explorer geöffnet und den eigenen Bedürfnissen entsprechend angepasst werden.

In der `[common]`-Sektion ist der Name des WLAN-Netzes sowie dessen Passwort einzutragen (`YOURSSID` und `YOURWIFIPW`). Außerdem ist der Netzwerkname bzw. die IP-Adresse des HA-Servers beim Build-Flag `SERVER_NAME` zu ändern. `SERVER_PATH` und `SERVER_PORT` bedürfen keiner Anpassung, sofern man nicht von der Standardkonfiguration abgewichen ist.

In der globalen `[env]`-Sektion sollten keine Adaptierungen notwendig sein, wenn man die ESP32-CAM-Boards verwendet. Für jedes Kisterl ist ein dezidierter `[env:NAME]`-Abschnitt mit den spezifischen Parametern erforderlich. In der Datei befindet sich eine auskommentierte Beispielkonfiguration.

Nach dem ersten erfolgreichen Flashen kann man zukünftig die OTA-Update-Funktion (Over-The-Air) nutzen und neue Firmware drahtlos aufspielen. Dazu muss die IP-Adresse des Knotens als `upload_port` angegeben werden und für `upload_protocol` der Wert `espota` gesetzt sein.

Die Datei platformio.ini wird ausführlich im ergänzenden Onlineteil gezeigt und erklärt.

Um den Upload zu starten, muss man in der Statusleiste von VSCode die gewünschte Umgebung (Environment, `env:`) aus der platformio.ini auswählen (Bild 12, rote Markierung). Anschließend kann man durch einen Klick auf den PIO-Button (kleiner Alien-Kopf) links in der „Activity Bar" die PIO-Tasks aufrufen. Ein Klick auf „Upload" – zu finden unter „General" – baut das Projekt und lädt es anschließend auf den ESP hoch.

Nach dem erfolgreichen Aufspielen der Firmware sollte sich der ESP32 mit dem heimischen WLAN verbinden und im Serial-Monitor (PIO Alien/Project Tasks/General/Monitor) mit `ESP32-CAM IP Address: ...` die erhaltene IP-Adresse ausgeben. Die korrekte Funktion der Kamera kann überprüft werden, indem man im Browser zuerst die Adresse `http://ESP32_IP/capture` aufruft. Die Meldung „Taking Photo" bestätigt, dass der ESP das Kommando entgegengenommen hat. Nach einigen Sekunden kann man unter `http://ESP32_IP/latest_image` das aufgenommene Bild begutachten.

Montage

Sind diese Tests abgeschlossen, kann es an die Montage der Kameras an den gewünschten Örtchen gehen. Ich habe sie mithilfe eines 3D-gedruckten Kugelgelenks an der Decke der Toiletten befestigt (Bild 13). Damit sind alle Arbeiten erledigt und SHICE ist einsatzbereit.

Weiteres Training

Ich war erstaunt, wie gut die Erkennungsleistung bereits nach der ersten Iteration war. Dennoch sah ich bei der Trefferquote noch Luft nach oben. Nachdem ich genügend Bilder gesammelt hatte, verfeinerte ich das Modell.

Dafür muss man zuerst die Bilder sortieren. Für diese Aufgabe stellt das Webinterface (Bild 14) den Button „Prepare training data" (6) zur Verfügung. Daraufhin werden alle Bilder in das Verzeichnis „training/TAG" verschoben, das nach dem erkannten „Tag" (1) benannt ist. Zuvor sollte man allerdings überprüfen, ob die Bilder auch korrekt klassifiziert wurden. Ein falsch zugeordnetes Bild kann durch einen Klick auf den entsprechenden Tag-Button rechts neben dem Score (5) manuell richtig einsortiert werden. Alternativ kann man auch einzelne Bilder fürs Training vorbereiten, indem man auf das Häkchen (2) klickt, das erscheint, wenn man mit der Maus über ein Bild fährt; mittels Klick auf den Papierkorb (3) löscht man das Bild. An der gleichen Stelle erscheint auch eine Checkbox (4), mit deren Hilfe man mehrere Bilder auswählen und anschließend mit den Buttons oben in der Seite entsprechende Sammelaktionen (7) und (8) ausführen kann.

Die so sortierten Bilder fügt man nun wiederum in Custom Vision hinzu und startet ein weiteres Training. Generell sollte sich die Anzahl der Trainingsbilder für die einzelnen Tags

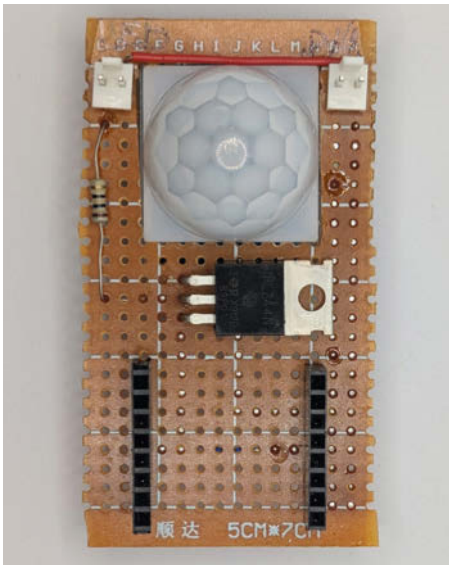


Bild 8: Platine mit PIR-Sensor.

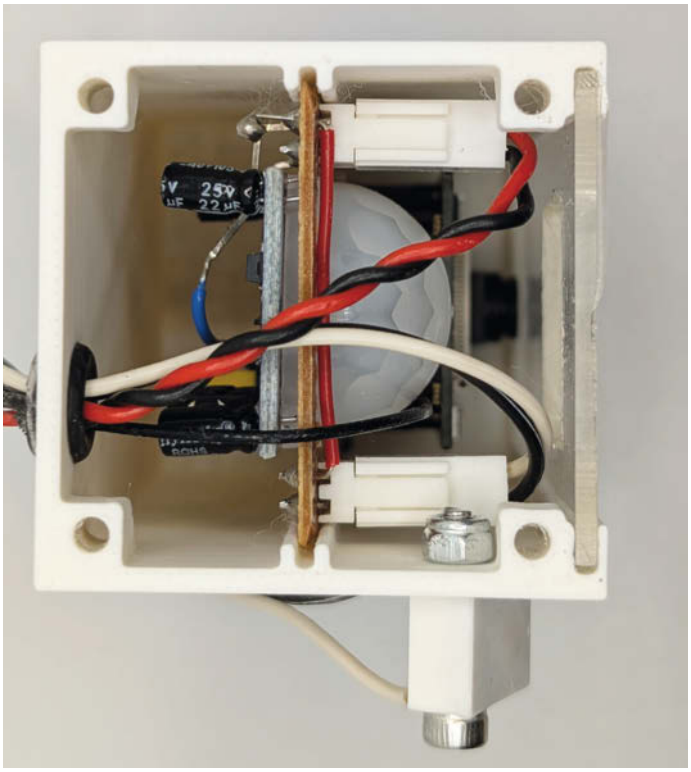


Bild 9: Die Breite der Nuten für die Frontscheibe kann im CAD-Programm angepasst werden.

Bild 10: ESP32-CAM-Node.

Bild 11: Rückseite eines ESP32-CAM-Nodes.

in etwa die Waage halten. Custom Vision weist auch darauf hin, wenn es ein Ungleichgewicht erkennt. Ein Klick auf „Add images“ öffnet den Dateidialog und man kann analog, wie beim initialen Training beschrieben, neue Bilder hochladen und taggen.

Hier lässt sich jetzt auch der vordefinierte `Negative`-Tag verwenden, um Fehlalarme bei Bildern, auf denen keine der Katzen zu sehen ist, zu kennzeichnen, was zu einer weiteren Verbesserung der Klassifizierung führt.

Das Training stößt man wieder durch einen Klick auf den grünen „Train“-Button an. Nach erfolgreichem Training erscheint erneut die Perfomance-Oberfläche und man bekommt am linken Rand eine Liste mit den bisherigen Iterationen angezeigt. Dort wählt man die neueste aus und kann die Erkennungsleistung des verfeinerten Modells überprüfen und es wie bereits beschrieben exportieren. Um es zu verwenden, reicht es aus, einfach die Dateien im tensorflow-Verzeichnis mit denen aus dem ZIP-Archiv zu ersetzen (und gegebenenfalls das Docker-Image neu zu erstellen).

Fehlalarme

Ein Problem bestand noch, nämlich dass beim Reinigen der Klos natürlich auch der Bewegungssensor auslöste und so ein Haufen sinnloser Bilder anfielen. Die Fehlalarme lassen sich zwar minimieren, indem man diese Bilder, wiederum mit dem `Negative`-Tag versehen, den Trainingsdaten hinzufügt, aber lieber war mir eine Lösung, bei der die Bilder gar nicht erst erstellt und gesammelt werden. Deshalb habe ich der Firmware der ESP32-CAM-Nodes die Endpoints `http://ESP32_IP/pir_sensor`

Onshape-CAD

Onshape ist eine CAD-Software zur 3D-Modellierung, die im Browser oder einer iOS/Android-App läuft und zeitaufwendige Berechnungen in die Cloud auslagert (siehe auch „Mehr zum Thema“). Ein weiterer Vorteil von Onshape ist das einfache Teilen von Dokumenten. Einen Link zu übermitteln, ist dafür ausreichend. Um ein Dokument modifizieren zu können, muss der Empfänger allerdings über einen Onshape-Account verfügen. Wie dieser angelegt wird und wie man dann mit den Daten weiterarbeitet, erklären wir ausführlicher im Onlineteil (siehe Kurzlink und „Mehr zum Thema“).

Bild 12: Die VSCode-Fußleiste.

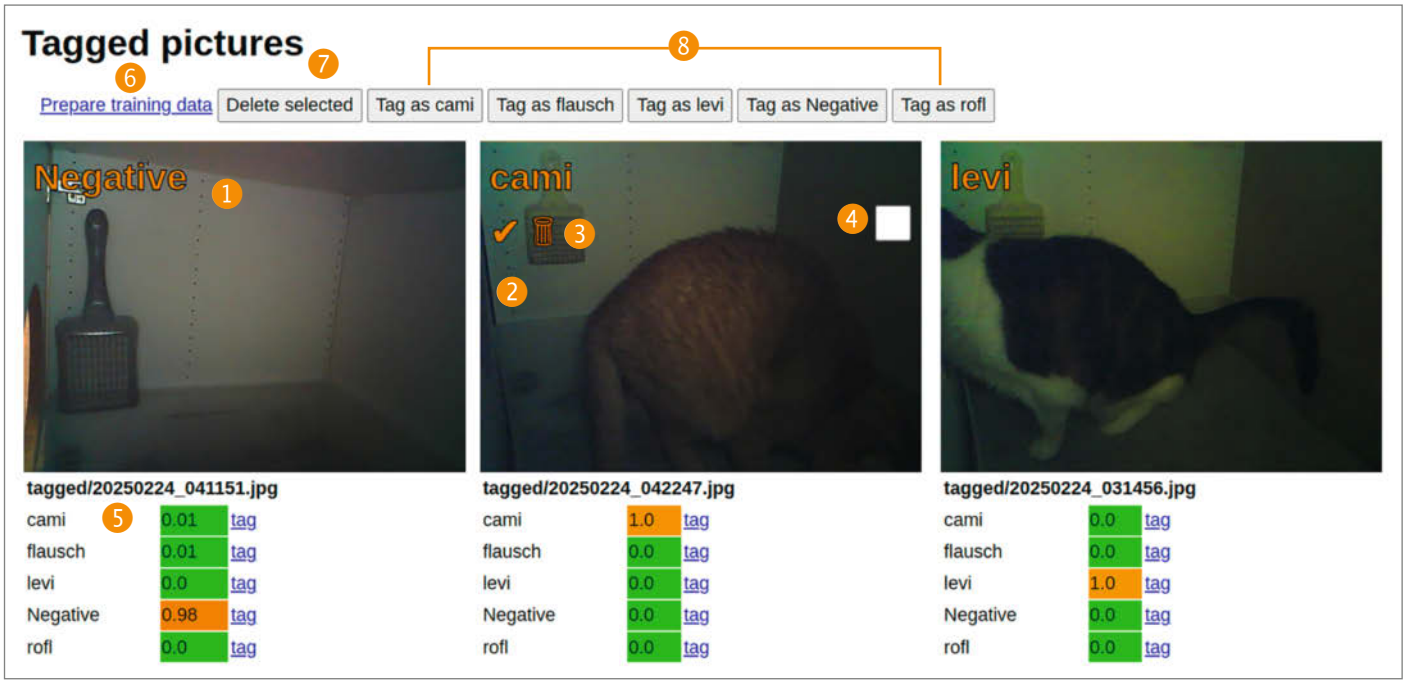


Bild 14: Custom Vision Webinterface.

Bild 13: Montierte Kamera.

`?state=(on|off)` hinzugefügt (`on` oder `off` entsprechend eintragen), mit dem sich der Bewegungsmelder (de-)aktivieren lässt. Bei der Gelegenheit baute ich auch gleich noch `http://ESP32_IP/light?state=(on|off)` ein, was das Ein- bzw. Ausschalten der LEDs ermöglicht. Zusätzlich legte ich in der HA-Konfigurationsdatei `switches.yaml` für jeden Ort zwei RESTful-Schalter an. Die ausführliche Konfiguration inklusive Listings der Konfigurationsdateien finden Sie im Onlineteil.

Somit kann ich nun, da ich meine HA-Installation mit Google-Assistant verbunden habe, mit dem Sprachbefehl „Wartungsmodus aktivieren" das Skript ausführen und für zehn Minuten die Bewegungserkennung aus- und das Licht einschalten, um die Kisterlreinigung unter verbesserten Lichtbedingungen durchzuführen.

Erfahrungen

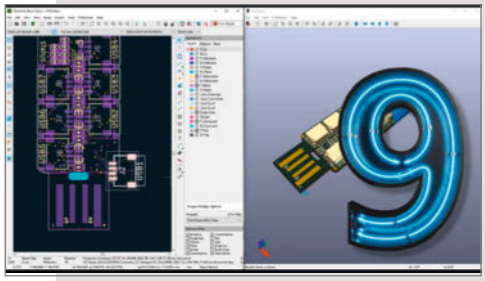
SHICE verrichtet seit mehreren Monaten zuverlässig seinen Dienst und hat sich bewährt. Einzig zwei der LEDs habe ich bereits tauschen müssen. Entweder rechnen ich und diverse Vorwiderstandsrechner im Web falsch und der Widerstand R1 ist nicht richtig dimensioniert oder es dürfen Zweifel an der Korrektheit des „Datenblatts" der LEDs aufkommen, das aus einem AliExpress-Produktlisting besteht. Möglicherweise ist das ein Punkt, bei dem es sich rentiert, in Markenprodukte aus bewährten Bezugsquellen zu investieren.

Der Tausch der LEDs hatte allerdings einen durchaus erwähnenswerten, positiven Nebeneffekt: Die Bilder einer der drei Kameras hatte stets einen grünen Farbstich. Im Web finden sich dazu ein paar Informationen (Suchbegriff „esp32cam green tint"), aber keiner der dort beschriebenen Workarounds schaffte wirklich Abhilfe. Da die Erkennung dennoch klappte, beschäftigte ich mich vorerst nicht weiter damit. Bei den ausgefallenen LEDs handelte es sich um kalt-weiße. Ich ersetzte sie durch warm-weiße gleicher Spezifikation und siehe da, das Problem mit dem Grünstich hat sich erledigt.

Vermutlich ist die Überwachung von Katzenklos für viele nicht gerade ein Killer-Anwendungsszenario, aber die Beschäftigung mit KI-gestützter Bilderkennung kann sich durchaus auch anderweitig rentieren. Mir persönlich sind dadurch einige Ideen gekommen, wie sich diese Technologie in meinem Brotberuf sinnvoll einsetzen lassen könnte. Insbesondere denke ich dabei an die Warenhaltung und an Inventurbelange. Aber erst mal werde ich versuchen, damit einen automatischen Balkontüröffner und -schließer für die Stubentiger zu realisieren – man muss ja schließlich Prioritäten setzen. —*caw*

KiCad EDA V9.0

Das beliebte E-CAD in neuer Version



KiCad ist eine Open-Source-Software, die PCB-Designern und Hobbyisten eine kostenlose und dennoch leistungsstarke Alternative zu kommerziellen EDA-Tools bietet. Sie enthält Funktionen für die Erstellung von Schaltplänen, das Layout und die 3D-Visualisierung von Leiterplatten. KiCad Version 9 bringt bedeutende Upgrades, wie ein neues Zone-Manager-Tool, eine verbesserte Netzinspektion und schematische DNP (Do Not Populate) auf Blattebene.

Am 18. Februar 2025 erschien die Version 9 von KiCad, die bereits in den letzten Monaten als Release Candidate (RC) von den Entwicklern und Benutzern getestet werden konnte. Wie gewohnt gibt es Pakete für Windows, macOS, Linux und auch Docker-Container. Und natürlich kann man es auch selbst kompilieren. Inzwischen ist bereits ein RC V9.1 erschienen.

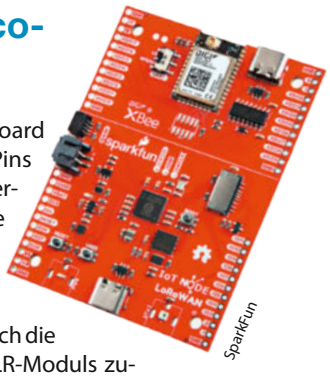
Beim Durchblättern der langen Liste der Verbesserungen, die in diese Version eingeflossen sind, fällt eines auf: Es handelt sich um eine Liste von Upgrades und Optimierungen einer stabilen Software und nicht mehr um wesentliche neue Funktionen. KiCad ist erwachsen.

Übrigens findet vom 11. bis 13. September wieder die KiCon-Europe in Bochum statt. Hier wird es auch um die Professionalisierung und weitere Entwicklung von KiCad gehen, aber es wird auch wieder Workshops für Einsteiger und Profis geben. —*caw*

Hersteller	KiCad EDA
URL	kicad.org
Preis	kostenlos

SparkFun IoT Node for LoRaWAN

LoRaWAN-Board mit Raspberry-Pico-RP2350A-Chip



Dieses Open-Source-Board (83 × 59 mm) basiert auf dem Raspi-Chip RP2350A und ermöglicht LoRaWAN-Projekte mit kleinem Fußabdruck. So sollen sich Maker-Projekte leicht ins Smart Home integrieren lassen.

LoRaWAN ist ein Funkstandard, der über mehrere Kilometer hinweg Daten übertragen kann. Auch wenn man solche Reichweiten im eigenen Heim selten benötigt, ist LoRaWAN aufgrund seiner Energieeffizienz mittlerweile ein gängiger Smart-Home-Standard.

Der Mikrocontroller integriert je zwei kleine ARM (Cortex-M33, 150 MHz)- und RISC-V-Kerne (Hazard3). Gepaart ist der Chip mit 16 MByte Flash-Speicher und 8 MByte RAM. Zusätzlich verfügt das Board über einen microSD-Karten-Slot.

Für die Kommunikation in einem LoRaWAN-Netzwerk ist ein XBee-LR-Modul von Digi integriert, das Frequenzen von 902 bis 928 MHz sowie 868 bis 870 MHz unterstützt. Sowohl der RP2350A als auch das XBee-LR-Modul können jeweils über USB-C angesprochen werden.

Auf dem Board sind 15 GPIO-Pins des RP2350A herausgeführt, die sich auf UART, I²C und SPI verstehen. Zusätzlich dazu sind auch die Pins des XBee-LR-Moduls zugänglich. Neben diesen ist auf dem Board auch eine JST-Buchse für einen Lithium-Polymer-Akku vorhanden, der sich direkt über einen auf dem Board befindlichen Ladeschaltkreis laden lässt.

Standardmäßig ist die Firmware zur Verwendung mit Digis X-ON-System vorgesehen. Der Chip lässt sich jedoch problemlos flashen und dann auch mit der Arduino IDE benutzen. —*das*

Hersteller	SparkFun
URL	make-magazin.de/xwdj
Preis	49 US-$

FreeCAD 1.0.0

Stabile Version mit wesentlichen Verbesserungen

Nach 22 Jahren ist es jetzt endlich so weit: FreeCAD hat Ende 2024 die Version 1.0.0 erreicht. Damit bringt das quelloffene CAD-Programm wichtige Verbesserungen mit und wird auch für den professionellen Einsatz attraktiver. Die Version 1.0.0 ist zwei Meilensteinen zu verdanken: der Integration der Assembly Workbench und der Lösung eines topologischen Benennungsproblems, das FreeCAD schon lange geplagt hatte.

Diese und weitere Neuerungen stammen aus dem Ondsel-Projekt, einem Fork von FreeCAD, dessen kommerzielle Ausrichtung in den vergangenen Jahren nicht genügend Akzeptanz fand. Die Entwicklung von Ondsel ES (Engineering Suite) wurde etwa zeitgleich mit dem Release der neuen FreeCAD-Version eingestellt.

Bereits das moderne Logo lässt vermuten, dass sich FreeCAD nicht nur technisch, sondern auch optisch verändert hat. So begrüßt den Nutzer beim ersten Start ein Willkommensbildschirm, in dem man die Sprache, Einheiten, den Navigationsstil und das Theme der grafischen Oberfläche einstellen kann. Neu ist auch der verschlankte Startbereich. Dafür kann man nun beim Erstellen eines neuen Projekts gleich den Typ auswählen, z. B., ob es sich um ein parametrisches Design oder ein Architekturmodell handelt.

In älteren FreeCAD-Versionen konnte man die Abmessungen einer Geometrie erst nach dem Erstellen mithilfe von Einschränkungen (oder Beschränkungen, engl. Constraints) festlegen. In Version 1.0.0 ist es jetzt möglich, die Dimensionen eines Objekts direkt beim Hinzufügen einzustellen.

Neu sind u. a. die experimentelle „Allow Compound"-Funktion und die integrierte Assembly Workbench. Weiteres lest ihr in unserem kostenlosen Online-Artikel (heise.de/s/gOgO3). —*Matthias Mett*

Hersteller	FreeCAD
URL	freecad.org
Preis	kostenlos

Chat GPT – Schlag die KI

Die will doch nur spielen

So jung die Anwesenheit von künstlicher Intelligenz (KI) im Alltag ist, so scheint sie mittlerweile in aller Munde und in jedem Wohnzimmer zu Hause zu sein. Mit „Chat GPT – Schlag die KI" geht der frechverlag noch einen Schritt weiter. Die KI wird zum Mitspieler.

Grundlage bilden 200 Fragen auf 100 Karten. Eine davon wird von einem Spieler laut vorgelesen, z. B. „Erfinde einen witzigen Slogan für einen Actionurlaub für Rentner!". Alle Mitspieler notieren ihre Ideen (Papier und Stifte sind nicht enthalten).

Wenn alle fertig sind, wird dieselbe Frage der KI gestellt. Alle Antworten, auch die der KI, werden laut vorgelesen und in der Runde entschieden, welcher Vorschlag der beste ist. Auf diese Weise wird so lange gespielt, wie der Spaß anhält. Man kann es aber auch auf ein paar Runden begrenzen – dadurch variiert die Spielzeit. Gedacht ist „Chat GPT – Schlag die KI" für einen bis acht Spieler ab 13 Jahren.

Praxistest: Mit vier realen Spielern zwischen 9 und 57 Jahren und der KI erhält man eine Menge Abwechslung und Spielspaß. Manche Fragen werden von der KI sehr komplex beantwortet, insbesondere, wenn es um die Erstellung von Gedichten und Regeln geht. Einige Fragen wie „Erfinde einen Namen für einen Arthouse-Film in Äthiopien" können die Spielenden (bis auf die KI) möglicherweise überfordern. Diese lassen sich dezent beiseitelegen.

Für eine spontane, lustige und kurzweilige Spielzeit ist „Chat GPT – Schlag die KI" zu empfehlen. Beim angegebenen Mindestalter ist noch Luft nach unten, wenn man die Individualität der Kinder beachtet, denn an Kreativität mangelt es ihnen nicht! Durch den Ausblick, mit einer KI zu spielen, lassen sich sogar pubertierende Jugendliche aus ihrer Höhle locken.

Wer nicht genug bekommt oder Abwechslung sucht, der kann noch auf die Variante „Chat GPT – Schlag die KI – Krass kombiniert!" zurückgreifen.

Jetzt seid ihr am Zug! Wie lässt sich KI in andere Spiele integrieren? Fallen euch selbst verrückte Fragen ein? Oder gar ein komplett neues Spiel mit der KI?

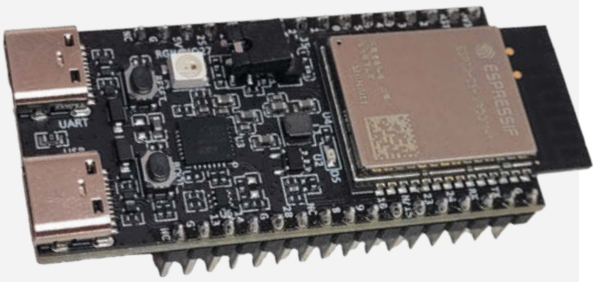
—Andrea Pfundt-Wartmann

Hinweis: Das Spiel wurde vom frechverlag zur Verfügung gestellt.

Hersteller	frechverlag GmbH
ISBN	4007742184834
Preis	10,99 €

ESP32-C5

RISC-V und 5-GHz-WLAN für Maker



Mit dem ESP32-C5 von Espressif können Maker ihre IoT-Projekte nun endlich ins 5-GHz-Band mit WiFi-6-Unterstützung hieven. Wie schon die anderen Modelle der C-Serie unterstützt der C5 Bluetooth LE, ZigBee, Thread und Matter und hat einen RISC-V-Kern, der mit 240 MHz getaktet ist. Der neue SoC wird vom Hersteller bislang allerdings nur spärlich – und auf Sample-DevKits verlötet – an ausgesuchte Maker verteilt. Um ihn zu testen, muss man die aktuelle Version des ESP-IDF 5.4 installieren, was wir als Plug-in in VSCode unter Linux erledigt haben. Darin wird der C5 als „Preview" unterstützt. In der Praxis funktionierte das Konfigurieren und Übersetzen von Programmen für den C5 jedoch reibungslos, sodass wir die WLAN-Funktion und die Leistungsfähigkeit des SoC testen konnten.

Für den Funkbenchmark haben wir über das Tool iperf die Verbindung zwischen dem ESP32-C5 und einem Notebook mit Intels AX201 (WiFi 6) ausgemessen. Schon im 2,4-GHz-Band erreicht der C5 mit gut 30 MBit/s mehr als den doppelten Durchsatz im Vergleich zu einem normalen ESP32. Auf sehr kurzen Entfernungen und mit dem C5 als Access-Point gelangen sogar bis zu 50 MBit/s. Im 5-GHz-Band drehte der C5 dann noch weiter auf: Über ein paar Meter Entfernung maßen wir 70 MBit/s. Das WROOM-Modul hat jeweils eine PCB-Antenne für 2,4 und 5 GHz.

Beim Coremark erreichte die CPU 719 Punkte, womit sie – abgesehen vom kommenden Modell P4 – alle anderen Kerne jedweden ESP-Modells in den Schatten stellte. Auch beim Whetstone-Benchmark lag sie klar vorn. Beim FFT-Benchmark mit 1024 Samples fiel sie mangels Gleitkommaeinheit (FPU) weit hinter einem normalen ESP32 und dem S3 ab. Eine genauere Übersicht aller Benchmarks sind in unserem Online-Artikel auf heise+ zu finden (siehe Link).

Eine „stable" Unterstützung ist für das ESP-IDF 5.5 angekündigt, das laut Roadmap Mitte dieses Jahres erscheinen soll. Vermutlich gibt es dann auch eine Unterstützung im Arduino-Core für die Arduino IDE. Wann Espressif den C5 final ins Rennen schickt, ist unklar. Einen offiziellen Preis gibt es auch noch nicht, aber bei JLCPCB war das C5-WROOM-1-Modul immerhin schon mit einem Preis von knapp 9 US-Dollar gelistet. *—Stefan Recksiegel*

Hersteller	Espressif
URL	make-magazin.de/xwd
Preis	9 US-$

IMPRESSUM

Make: Nächste Ausgabe erscheint am 30. Mai 2025

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)

Redaktion: Johannes Börnsen (jom), Ákos Fodor (akf), Marcus Hansson (mch), Daniel Schwabe (das), Dunia Selman (dus, Social Media), Ulrich Schmitz (usz), Carsten Wartmann (caw)

Mitarbeiter dieser Ausgabe: Beetlebum, Dirk Fox, Frank Frohnert, Miguel Gashi, Michael Gaus, Hartmut Grawe, Matthias Keldermann, Florian Klug-Göri, Johannes Kreuzer, Kai-Uwe Mrkor, Dr. Stefan Recksiegel, Andrea Pfundt-Wartmann

Assistenz: Susanne Cölle (suc), Martin Triadan (mat)

Layout und Satz: Steffi Martens, Lisa Reich, Nicole Wesche, Heise Medienwerk GmbH & Co. KG

Korrektorat: Anne-Marie Berndt, Dörte Bluhm, Lara Bögner, Marei Stade, Christiane Tümmler, Heise Medienwerk GmbH & Co. KG

Titel: Nicole Wesche

Fotografie und Titelbild: Andreas Wodrich

Digitale Produktion: Melanie Becker, Thomas Kaltschmidt, Pascal Wissner

Hergestellt und produziert mit Xpublisher:
www.xpublisher.com

Verlag

Maker Media GmbH
Postfach 61 04 07, 30604 Hannover
Karl-Wiechert-Allee 10, 30625 Hannover
Telefon: 05 11/53 52-0
Telefax: 05 11/53 52-129
Internet: www.make-magazin.de

Herausgeber: Christian Heise, Ansgar Heise

Geschäftsführung: Ansgar Heise, Beate Gerold

Anzeigenleitung: Daniel Rohlfing (-844)
(verantwortlich für den Anzeigenteil),
mediadaten.heise.de/produkte/print/das-magazin-fuer-innovation

Leiter Vertrieb und Marketing: André Lux (-299)

Service Sonderdrucke: Julia Conrades (-156)

Druck: Dierichs Druck + Media GmbH & Co. KG, Frankfurter Str. 168, 34121 Kassel

Vertrieb Einzelverkauf:
DMV DER MEDIENVERTRIEB GmbH & Co. KG
Meßberg 1
20086 Hamburg
Telefon: +49 (0)40 3019 1800
Telefax: +49 (0)40 3019 1815
E-Mail: info@dermedienvertrieb.de
Internet: dermedienvertrieb.de

Einzelpreis: 14,50 €; Österreich 15,90 €; Schweiz 27.50 CHF; Benelux 17,10 €

Abonnement-Preise: Das Jahresabo (7 Ausgaben) kostet inkl. Versandkosten: Inland 87,50 €; Österreich 95,90 €; Schweiz 130,90 CHF; Europa 102,20 €; restl. Ausland 107,80 €

Das Make-Plus-Abonnement (inkl. Zugriff auf die App, Heise Magazine sowie das Make-Artikel-Archiv) kostet pro Jahr 6,30 € Aufpreis.

Abo-Service:

Bestellungen, Adressänderungen, Lieferprobleme usw.:
Maker Media GmbH
Leserservice
Postfach 110 242
69071 Heidelberg
E-Mail: leserservice@maker-media.de
Telefon: 0511/592 99 077

ISSN 2364-2548

Nachgefragt




Wir zeigen im Heft, wie man den Herzschlag visualisiert. Welche weitere Körperfunktion würdest du gerne anzeigen wollen?

Dirk Fox
Karlsruhe, steuert auf S. 74 Fischertechnik mit der Oxocard.

Den Blutzuckerwert visualisieren bisher nur sehr teure Sensoren. Großartig wäre es, den Kalorienverbrauch angezeigt zu bekommen. Und gelegentlich wäre es hilfreich, seinen genauen Blutalkoholwert zu kennen.

Florian Klug-Göri
Graz, überwacht auf S. 50 seine Katzen per KI.

Seitdem mich meine Ärztin als Wohlstandskrankheits-Patient tituliert, hege ich besonderes Interesse an meinem Blutzuckerspiegel und Blutdruck. Da wäre eine Echtzeitvisualisierung sehr hilfreich für mich.

Frank Frohnert
Brunsbüttel, steuert auf S.82 den Anschlag seiner Kappsäge.

Definitiv den Blutdruck. Für alle Außenstehenden gut sichtbar, von „freundlichem Grün" bis „tiefrot"!

Matthias Keldermann
Grundelfingen, erklärt ab S. 36, wie Making in der Schule gelingt.

Ich hätte gern einen Sensor, der meinen Koffeinspiegel misst – von „Tiefschlaf" über „Kaffeedurst" bis „Espressogott". Optional mit Warnung: „Maximale Dosis erreicht, bitte entkoffeinieren!"

Inserentenverzeichnis

Ein Teil dieser Ausgabe enthält Beilagen der DIMABAY GmbH, München.

Hochleistungs-Silent-PC mit Gehäusedämmung

TUXEDO PRO Desktop-Serie

Das geräuschlose PC-Erlebnis – die TUXEDO PRO Desktop-Serie ist in drei Größen erhältlich und mit einem geräuschgedämmten Aluminium-Stahl-Gehäuse ausgestattet. Dieses sorgt für Stabilität und flüsterleisen Betrieb, während die integrierte CPU-Wasserkühlung für niedrige Temperaturen sorgt. Stelle dir deinen TUXEDO Pro genau nach deinen Anforderungen zusammen – egal ob für Gaming, kreative Projekte oder professionelle Anwendungen. Deine Bedürfnisse stehen im Fokus. Außerdem bleibt Dank der großen Anschluss-Vielfalt dein Setup flexibel und zukunftssicher.

Linux kompatibel

Bis zu 5 Jahre Garantie

Sofort einsatzbereit

TUXEDO

tuxe.do/make0225p

Gefertigt in Deutschland

Deutscher Datenschutz

Deutscher Tech Support