DEUTSCHLANDS GEFÄHRLICHSTES DIY-MAGAZIN

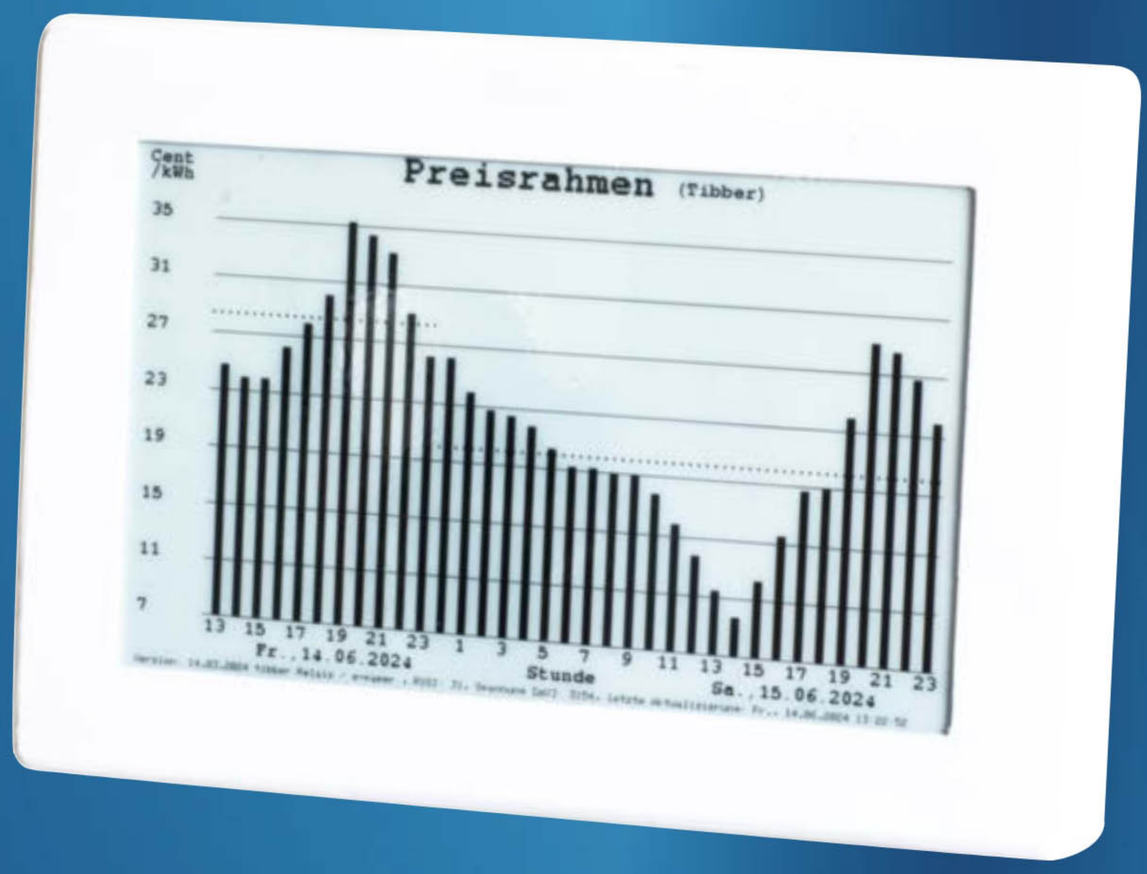
c't Make:

Make:

Endlich:
Taupunktlüfter ins Smarthome bringen

Anzeige für Strompreise

- Für dynamische Tarife
- Mit ePaper-Display und ESP8266
- Praxis mit Tibber

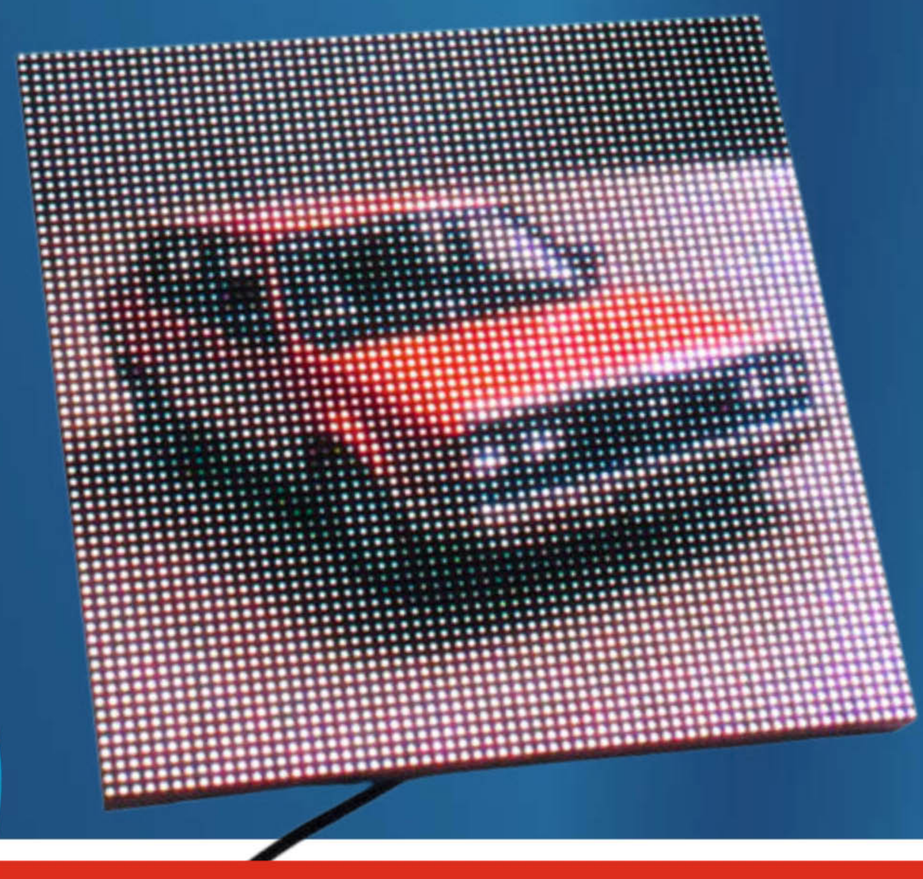


CNC-Sandmaltisch

- Zeichnet selbstständig Muster
- Detaillierte Nachbauanleitung
- Mit Holz, 3D-Druck und Elektronik

Animierte Pixeldisplays

- PNG-Grafiken anzeigen
- Eigene Bilder animieren
- Helligkeit mit LDR steuern



10 JAHRE Maker Faire HANNOVER

Workshops

- Rendern mit FreeCAD
- Möbel bauen mit System 32

Programmieren

- Spiel für BBC micro:bit
- Einstieg ins Debugging

4/24
26.7.2024
CH CHF 26.50
AT 14,90
Benelux 15,90
€ 13,50
4 192314 113500 04

Tüfteln, Knobeln, Basteln!

Mit unseren Büchern werden Sie zum Alles-Erfinder. Lassen Sie sich von spannenden Projekten für Maker und Tekkies inspirieren: Tomatenzucht mit dem Mikrocontroller, geniale Roboter-Autos und moderne Smart-Home-Technik. Garantierter Tüftelspaß – bis der Lötkolben raucht!

Aktion vom 9. August–2. September

Alle Bundles (Buch + E-Book) zum Vorteilspreis!

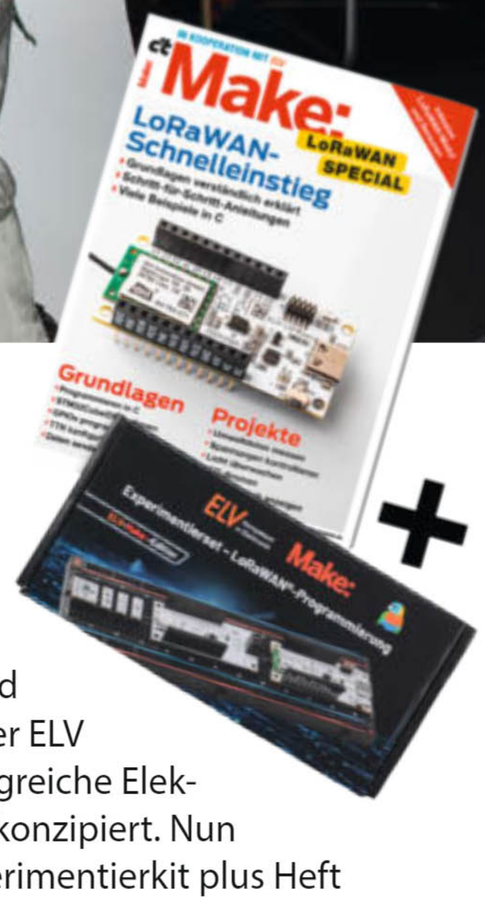
www.rheinwerk-verlag.de/maker

10 JAHRE Maker Faire HANNOVER

Wir sind auf der Maker Faire Hannover.
Besuchen Sie uns an Stand 94.

Rheinwerk

10 Jahre Make und Maker Faire Hannover



Seit der Erstausgabe im Oktober 2011 hat unser Magazin einige Änderungen durchgemacht. Hießen wir anfangs noch „c't Hardware Hacks", so strichen wir zur Ausgabe 1/14 das „Hardware" im Titel, um nur noch als „c't Hacks" zu firmieren und uns thematisch etwas breiter aufzustellen. Auf der Ausgabe 4/14 prangte dann erstmals groß „Make:" vorne drauf, sodass das Heft und die von uns veranstaltete Maker Faire Hannover nun namentlich besser zusammenpassten.

Das gefiel allerdings nicht allen Lesern. Einige warfen uns vor, jetzt einen unnötigen Anglizismus im Titel zu führen - als sei „Hacks" ein urdeutsches Wort. Andere sorgten sich um die Amerikanisierung unseres Heftes und dass unser US-Lizenzpartner künftig die Inhalte diktieren würde. Unkenrufer prophezeiten uns ein schnelles Ende, weil sich das Heft ja nun in ein Lifestyle-Magazin wandeln würde und diese neumodische Maker-Bewegung, mit ihren Arduinos, sowieso nicht ernstzunehmen sei.

Wie Sie sehen, es gibt uns immer noch und die Maker Faire Hannover ist zum 10. Mal in Hannover präsent (mehr dazu auf Seite 44). Wenn das kein Grund zum Feiern ist! Und auch die Maker-Bewegung hat sich nicht aufgelöst. Stattdessen findet man bundesweit hunderte von Makerspaces, in denen sich Interessierte organisieren und gegenseitig unterstützen. Making ist in Schulen und Universitäten angekommen. Das von selbsternannten „Profi-Bastlern" belächelte Arduino-Ökosystem mit seinen vielen praktischen Bibliotheken hat sich mittlerweile sogar in namhaften Unternehmen als Plattform für Rapid Prototyping etabliert.

Das Make-Magazin hat sich ebenfalls über all die Jahre weiterentwickelt und ist nicht mehr nur die Quelle der Inspiration und der verrückten (und bisweilen gefährlichen) Ideen, sondern auch eine Anlaufstelle, um Dinge von Grund auf zu lernen. Zahlreiche unserer Make Specials vertiefen das Wissen um Mikrocontroller, Programmiersprachen und Elektronik. Zusammen mit unserem Partner ELV haben wir in den letzten Jahren zwei erfolgreiche Elektronik-Experimentierkits mit Anleitungen konzipiert. Nun legen wir ein einzigartiges LoRaWAN-Experimentierkit plus Heft nach, mit dem Sie sofort eigene Sensorknoten programmieren können. Weitere Infos und einen Link zu unserem Shop finden Sie unter dem Link unten.

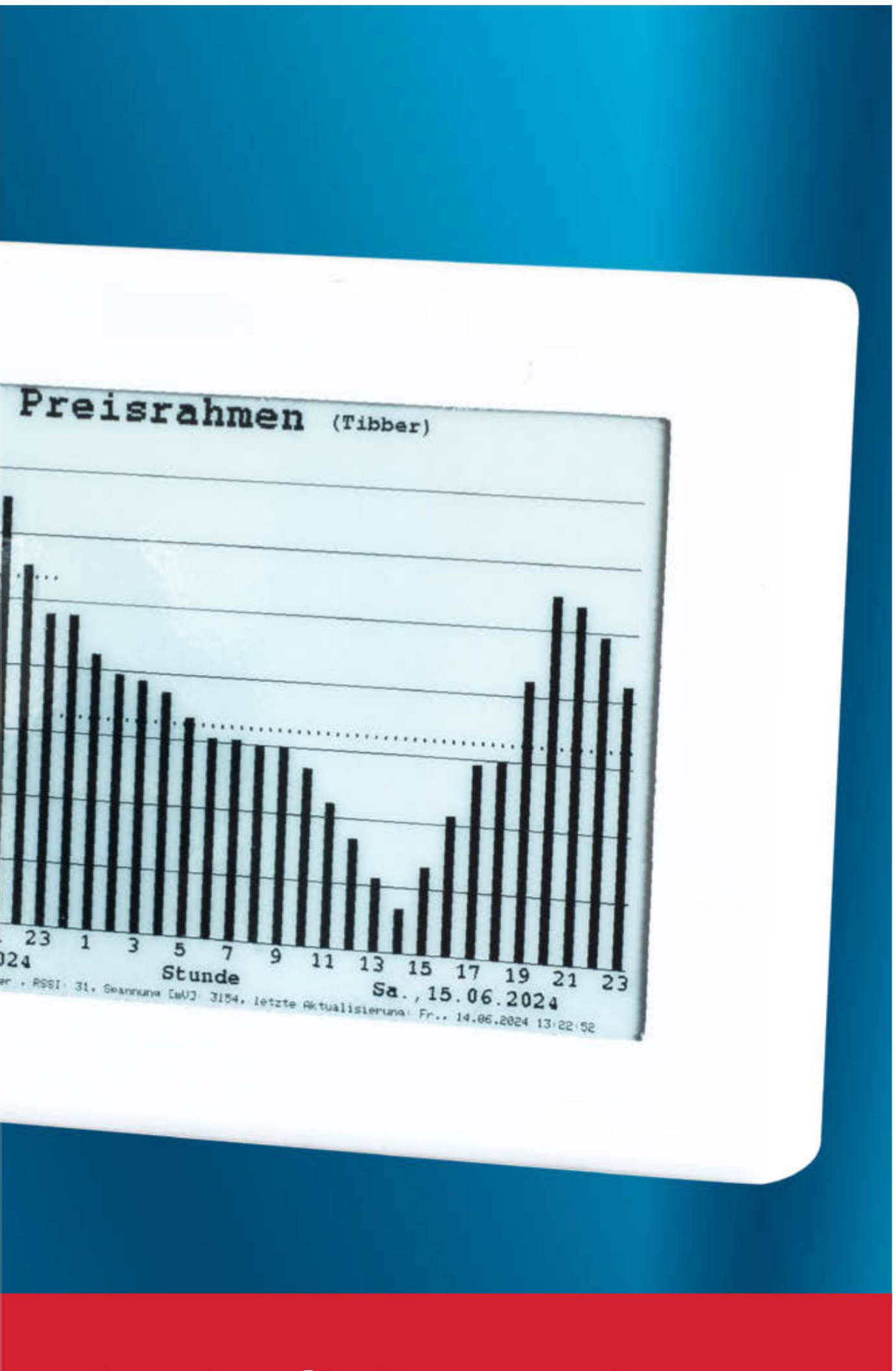
Doch damit nicht genug: In Kooperation mit dem Schweizer Hersteller Oxon planen wir, bis Ende des Jahres das Make Innovators Kit plus Playbook anzubieten, das auf einem ESP32 mit Display und Mini-Joystick basiert. Über ansteckbare Cartridges wie Breadboards, Sensoren und NeoPixel-Boards lassen sich Experimente in einer Web-IDE programmieren. Zusammen mit dem Umweltcampus der Hochschule Trier konzipieren wir zudem das Makey: Lab (siehe Foto S. 47), mit dem Schüler und Schülerinnen ab 14 Jahren leicht und verständlich in die digitale Erforschung ihrer Umwelt einsteigen können – dank eingebauter Sensoren und grafischer Programmierung spielend leicht. Das alles präsentieren wir den Besuchern auf unserem Stand 96 auf der Maker Faire. Wir sind sehr gespannt auf Ihr Feedback. Ein Grund mehr, am 17. oder 18. August nach Hannover zu kommen, oder? Wir sehen uns!

Happy Hacking!

Daniel Bachfeld

make-magazin.de/xqm9

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



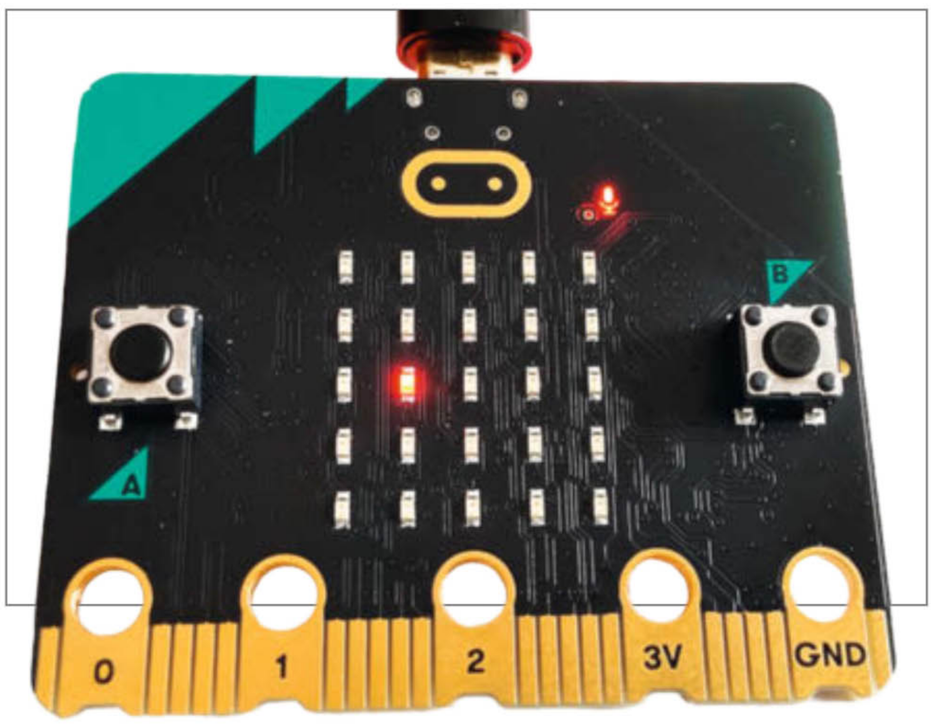
Anzeige für Strompreise

Mit dynamischen Stromtarifen lässt sich viel Geld sparen. Der Tibber-Preisrahmen zeigt die stündlichen Preise an, anhand derer man entscheiden kann, ob die richtige Zeit für den Waschmaschinengang gekommen ist. Der schicke batteriebetriebene Preisrahmen mit ePaper-Display hält dank Deep Sleep des Mikroprozessors lange mit einer Batterie durch. Weiterhin liefert der Artikel auch eine Einführung in dynamische Strompreise und wie man sie am besten ausnutzt.

Inhalt

Workshops

Haben Sie sich schon einmal gefragt, warum Möbelbeschläge herstellerübergreifend passen? Das System 32 ist ein ausgeklügeltes Prinzip zum Möbelbau. Mit diesem Wissen können dann auch zum System 32 kompatible Möbel in FreeCAD geplant und fotorealistisch gerendert werden. Im Weiteren geht es um die Entwicklung eines Spiels auf dem BBCMicro:bit und um die Fehlersuche mittels Debugger in ATtiny-Programmen.

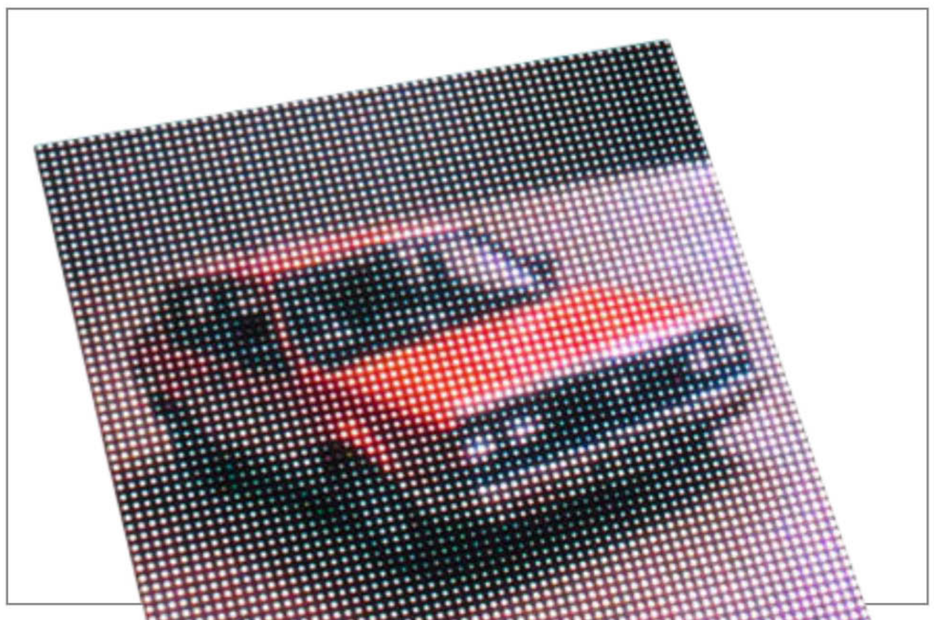


Taupunktlüfter ins Smarthome bringen

Unser Taupunkt-Lüftungssystem ist durch die dauernden Überschwemmungen immer noch hochaktuell. Um langfristig Räume von Feuchtigkeit zu befreien, wird das Lüftungssystem ins WLAN gebracht, der Datenaustausch wird per MQTT erledigt und das Lüftungssystem ins Smarthome integriert. Dies erlaubt eine zentralisierte Steuerung mehrerer Lüfter im Haus.

Animierte Pixeldisplays mit MicroPython

Ein Infotext oder eine Laufschrift auf einer LED-Matrix sind praktisch und machen optisch etwas her. Mit ein paar animierten Grafiken lässt sich das Ganze aber auch noch aufpeppen. Wir zeigen, wie Bilder in einfache Arrays umgewandelt, eingefärbt und animiert werden. Die Hardware und Software wird erweitert, um die Helligkeit der Matrix mittels eines Lichtsensors zu steuern.



Themen von der Titelseite sind rot gesetzt.

CNC-Sandmaltisch

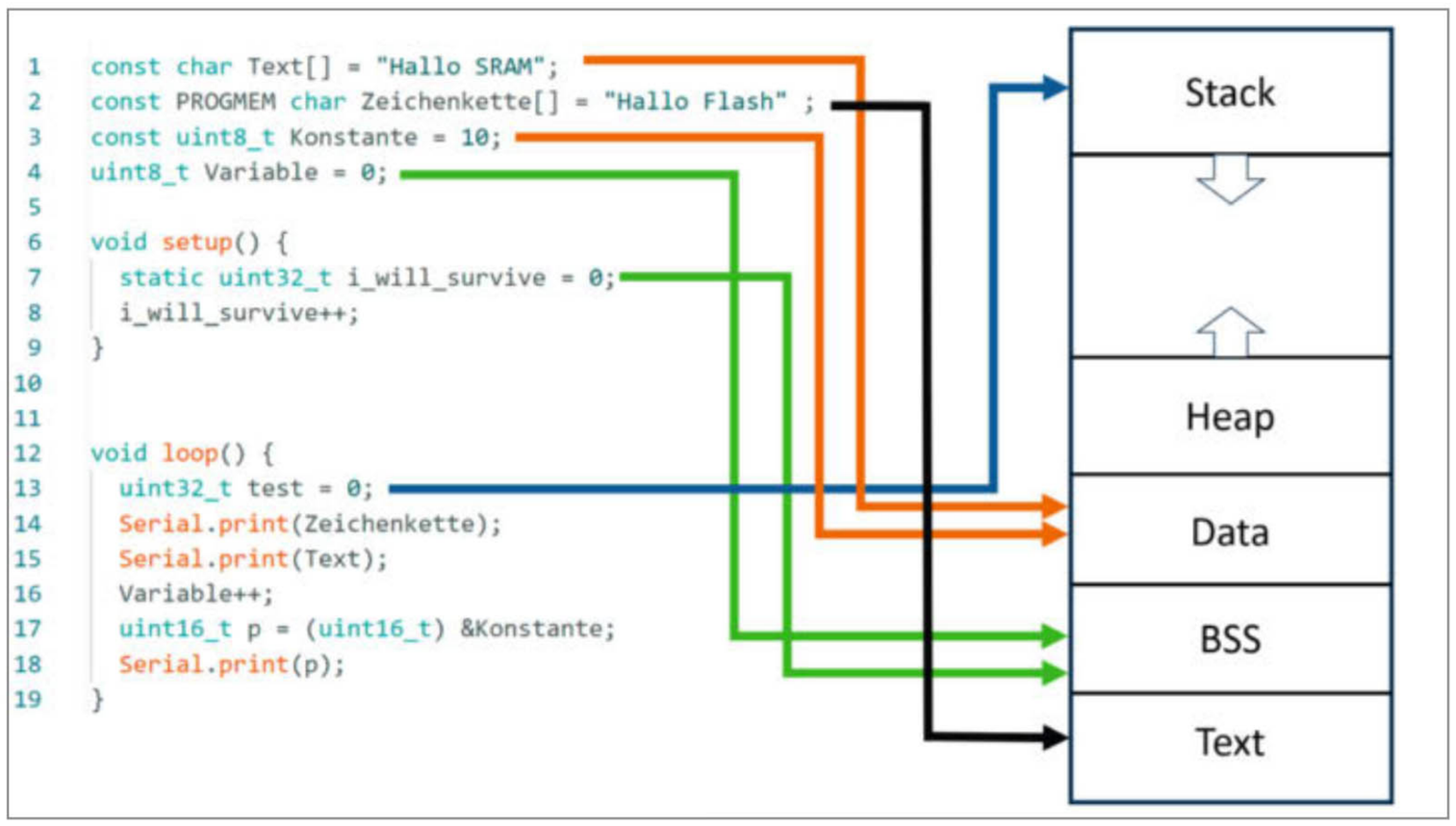
Nach unserem kurzen Artikel im Reparatur-Sonderheft forderten immer mehr Leser eine genaue Bauanleitung. Unser Autor hat keine Mühen gescheut und eine ausführliche Online-Schritt-für-Schritt-Anleitung, Stücklisten und CAD-Zeichnungen angefertigt. Somit kann sich jetzt jeder Maker mit etwas handwerklichem Geschick den Traum vom Sandmaltisch erfüllen.

Leserforum

Berichtigungen

Speicherverbrauch in Mikrocontrollern, Make 3/24, S. 92

Die statische Variable `i_will_survive` wird im letzten Bild fälschlicherweise dem Data-Segment zugeordnet. Sie liegt aber im BSS-Segment, folgend das richtige Bild:



Anregungen

Ich habe gerade beim Frühstück das neue Heft durchgeblättert und mich gefreut; es gibt wieder eine breite Palette von interessanten Projekten und Infos. Eventuell hätte ich noch ein paar Vorschläge für Artikel: Nach ca. 50 Jahren mehr oder weniger Abstinenz interessiert mich das Thema Synthesizer wieder. Dabei bin ich auf die Synthio Bibliothek unter CircuitPython gestoßen. Damit und mit einem Raspi Pico (ich bevorzuge europäische Produkte vor den chinesischen) wollte ich einmal weiter experimentieren. Vielleicht sind auch andere Leser interessiert?

Für die Darstellung der Daten meiner Solaranlage benutze ich Home Assistant. Da die WiFi-Übertragung Aussetzer hatte, benutze ich nun eine serielle Übertragung. Mit dem originalen seriellen Sensor in HA gab es Probleme, das führte dazu, dass ich den Quellcode studieren musste. Eine Zeile zur UTF-8-Decodierung machte Probleme, da sie keine Fehlerüberprüfung hatte. Bei „kaputten" Bytes hörte die Übertragung auf. Es genügte ein `try - except`, um das Problem zu lösen. Wie ich dann den neuen Quellcode einbaue, war ein nicht ganz triviales Problem. Hier hat mir interessanterweise ChatGPT geholfen, sodass ich weniger googeln musste. Liebe Grüße und weiter so!

Jean-Claude Feltes

Mehr Mechanik, bitte!

Titel, Make 3/24, S. 1

Nun lese ich Eure Zeitung schon einige Jahre. Leider fällt mir auf, dass Ihr Euch zu stark auf Elektronik eingeschossen habt, für die Jugend sicher nicht schlecht. Bin aber über 70 und gelernter Mechaniker. Deshalb würde ich doch lieber mehr mechanische Ideen bevorzugen! Ja, nun hatte ich mir eine CNC-Lasermaschine gebaut, lief auch mechanisch sehr gut! Programmierschwierigkeiten waren vorprogrammiert. Nun wollte ich dieses Optimieren-Testprogramm für LighBurn von 3 /4 2024 Seite 77 mal runterladen! Nach 2 Stunden hab ich es aufgeben müssen. Irgendwie klappte der Link nicht! Danke für Eure Einfälle, macht weiter so, beachtet aber auch, dass nicht jeder ein Elektroniker ist.

Udo

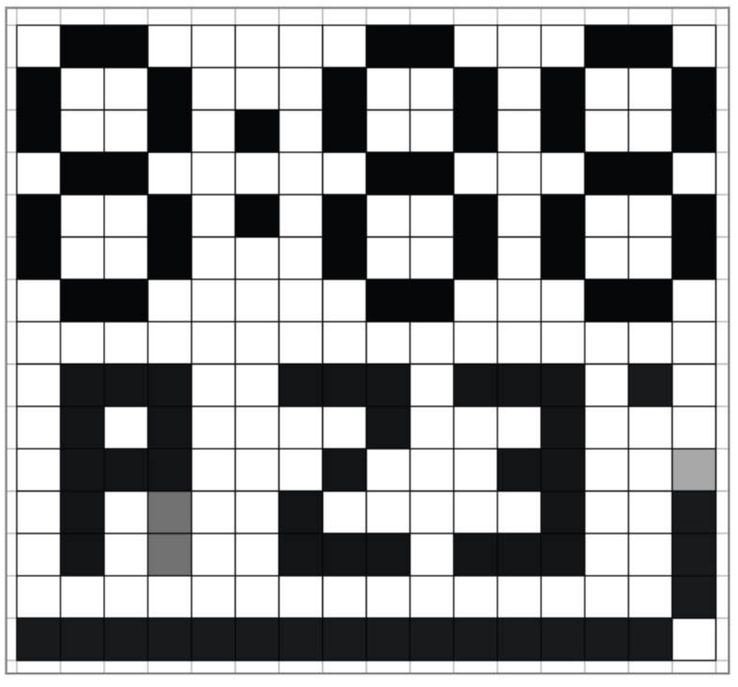
Danke für das Feedback! Gerne veröffentlichen wir auch Artikel rund um Mechanik, allein der Input aus der Community fehlt uns!

Verbesserung

Auch wenn ich vieles nicht umsetzen will oder kann, finde ich es doch immer wieder interessant und inspirierend. Selbst, wenn ich mich mit dem einen oder anderen Thema bereits sehr gut auskenne. Weiter so und nicht entmutigen lassen von irgendwelchen Moralaposteln, die sich besser mit Kreuzworträtsel-Heften befassen sollten – wobei man sich auch an einem Bleistift verletzen könnte. Meine Kommentare kommen erst jetzt, weil ich im Heft 3 über die Leserbriefe gestolpert bin und feststellte, dass ich das 2. noch gar nicht gelesen hatte.

Zum IKEA-Leuchtwürfel: Schon beim Lesen hatte ich eine Idee, wie es doch einzeilig gegangen wäre, sogar mit Zusatz-Features (z. B. Raumtemperatur etc.). Erklärung zum Bild: zweite Zeile links (AM / PM); rechts z. B. Raumtemperatur (Sekunden gingen natürlich auch); unten: 15 Pixel waagerecht für die Sekunden einer Viertelminute und 4 Pixel senkrecht für die Anzahl der Viertelminuten (wobei die 4. eigentlich entfallen kann). Inspiriert durch die „Berlin-Uhr"; Teiler-Punkte könnten noch im Sekundentakt blinken.

Dann noch ein Kommentar zum Tube-Amp: Für mich nichts Neues, da ich seit rund 40 Jahren mit Röhren bastel. Aber nichtsdestotrotz ein toller Artikel! Auch wenn er sich etwas befremdlich liest, wenn erst von einem liebevoll zusammengestellten Bausatz die Rede ist, und dann seitenweise Probleme behoben werden. Ich bin allerdings auch der Meinung, dass es hoch anzurechnen ist, wenn jemand völlig uneigennützig sich solch eine Mühe und Arbeit macht! Für alle, die weitere Schaltpläne studieren möchten und mehr über diese ungebrochen faszinierende Technologie lernen wollen, zwei Standardwerke: „Hören mit Röhren" von „Onkel FiHu" Fritz Hunold sowie



„Audio- und Gitarrenschaltungen“ von R. zur Linde. In Zweiterem finden sich auch Schaltpläne von bekannten Röhrenverstärkern, wie von der legendären Reußenzehn Gitarrenendstufe. Und wer es ganz nostalgisch mag und Daten sucht: Ab und an finden sich in Antiquariaten noch die kleinen Taschenbücher mit Röhren- und Transistor-Daten aus den 60ern (Valvo, Telefunken etc.). Ich habe da einige und ein Bekannter löst gerade seine Sammlung auf. Gibt es natürlich auch in „neuer“: „Röhren-Taschen-Tabelle“ RTT von Franzis leistet mir seit über 35 Jahren gute Dienste.

Andreas Basarke

Falsche Espressif-API

Die Erlenmeyer-Laserröhre, Make 3/24, S. 8

Vielen Dank für den interessanten Artikel. Habe sofort den Laserscanner bestellt. Leider kommt es beim Überprüfen/Laden der Software in der Arduino IDE 1.8.19 zu folgendem und weiteren Fehlern: `error: 'ledcAttachPin' was not declared in this scope`; was mache ich falsch?

Uwe Scheffer

Ich verwende in der Arduino IDE die Espessif API Version 2.0.17. Sie hingegen verwenden die Version 3.0.0 oder 3.0.1. Zwischen diesen beiden Versionen hat sich sehr viel geändert. So werden nun einzelne Befehle anders benannt, Funktionen wurden weggelassen oder ersetzt, Parameterlisten haben sich verändert usw. Ich habe versucht, den Erlenmeyer-Code für die neue API umzubauen, aber es ist mir auf die Schnelle leider nicht gelungen. In meinen Augen ist die Version 3 noch nicht ausgereift.

Bleibt noch Ihr Problem: Am einfachsten gehen Sie in den Board-Verwalter, dort zu esp32 und installieren dann die letzte Version der Espressif API vor 3.0.0 (das wäre die Version 2.0.17). Dies müsste dann zum Erfolg führen und Ihr Code kompiliert ohne Fehlermeldungen.

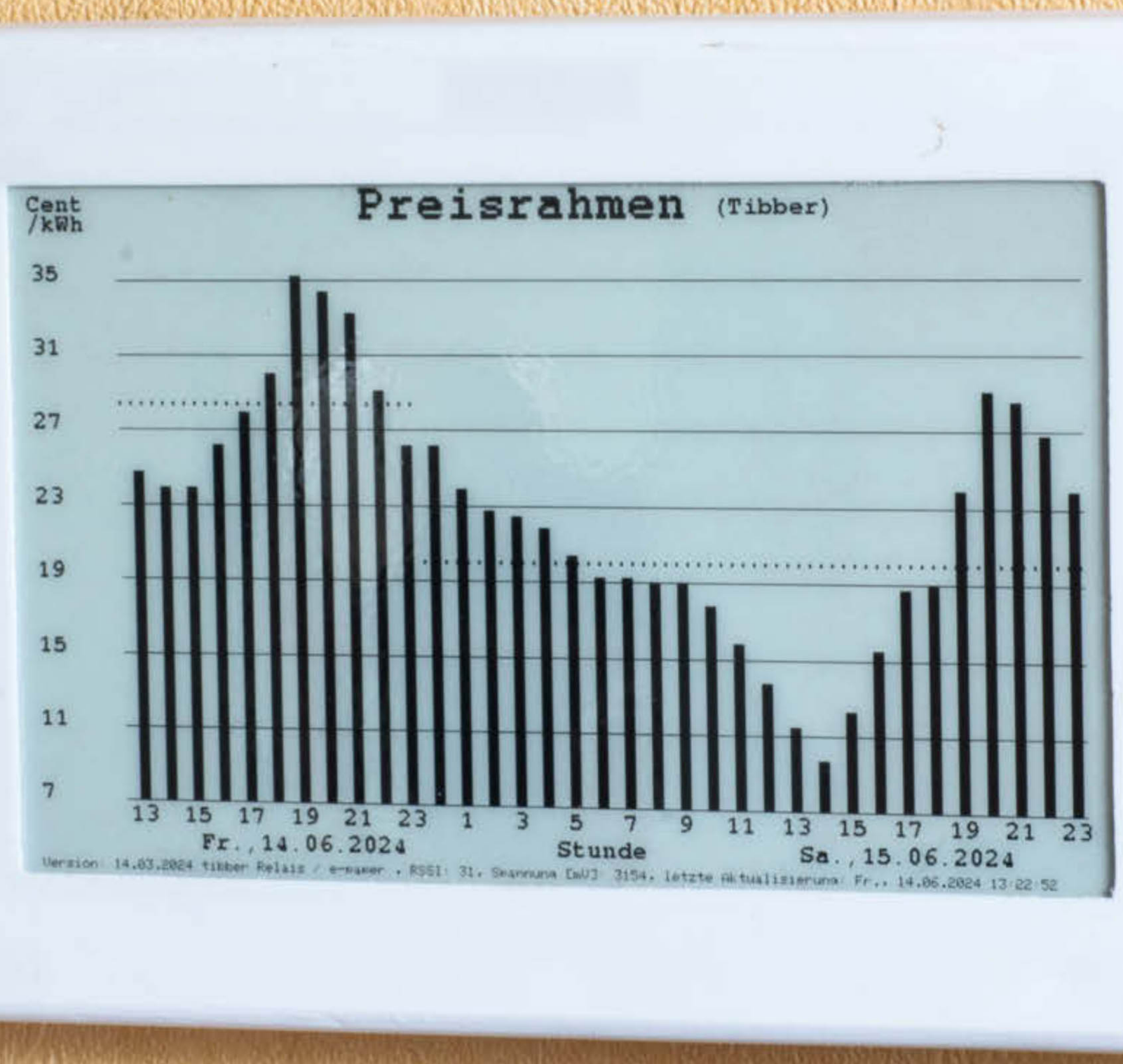
Warum Microchip Studio?

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

Danke für Ihren Artikel zu der neuen ATtiny-Familie. Was mir nicht einleuchtet, ist, warum Sie auf Microchip Studio für die Programmierung verweisen, da der Maker nur damit Programme schreiben könnte. Die ATtiny-Serien, einschließlich der neuen Familien, lassen sich z. B. auch mit der bei Makern sehr beliebten PlatformIO IDE über UPDI flashen und natürlich auch das Arduino Framework verwenden. Ich bin mir ziemlich sicher, dass das auch mit der Arduino IDE funktioniert. Beide IDEs sind so aufgebaut, dass sie weitestgehend auf die gleichen Frameworks und Libraries zugreifen können. Ein Beispiel für eine platformio.ini für ein Projekt mit ATtiny412 könnte z. B. so aussehen:

```
[env:ATtiny412_pyupdi_upload]
platform = atmelmegaavr
framework = arduino
board = ATtiny412
upload_speed = 115200
upload_port = COM11
upload_flags =
 -d attiny414
 -c $UPLOAD_PORT
 -b $UPLOAD_SPEED
upload_command = pyupdi
    $UPLOAD_FLAGS -f $SOURCE
```

Wolfgang Klenner



Strompreise im Auge behalten

Nicht nur an der Börse und an der Tankstelle schwanken die Preise. Das gilt zunehmend auch für den Strom aus der heimischen Steckdose, denn sogenannte dynamische Stromtarife, bei denen sich der Preis für den Strom stündlich ändert, werden immer beliebter. Wer darauf achtet, wann der Strom am günstigsten ist, kann so Geld sparen. Dabei hilft unser Preisrahmen, der die Preise übersichtlich darstellt.

von Uwe Rohne

Wer sich für einen dynamischen Strompreisvertrag entscheidet, erhält vom Energielieferanten keine festen kWh-Preise mehr, sondern sich stündlich oder sogar viertelstündlich wechselnde Preise. Je nach Anbieter und Tarif wird der aktuelle Strompreis unterschiedlich berechnet. Richtungsweisend ist der europäische Day-Ahead-Handel in Paris (epexspot.com). Dort werden gegen 13:00 Uhr die Preise für den nächsten Tag veröffentlicht.

Um die Stunden mit den niedrigsten Preisen optimal nutzen zu können, wurde unser Preisrahmen entwickelt. Natürlich kann man die tagesaktuellen Preise auch im Internet oder über verschiedene Apps abrufen, aber mit dem Preisrahmen geht es noch einfacher. Wenn unser anschauliches Projekt zum Beispiel im Wohnzimmer an der Wand hängt, können sich alle Mitbewohner über den aktuellen Strompreis informieren, ohne das Handy aus der Tasche ziehen zu müssen.

Voraussetzung für das hier vorgestellte Projekt ist ein Vertrag mit dem Ökostromanbieter Tibber, denn nur als Tibber-Kunde erhalten wir die benötigten Strompreisdaten per API (mehr dazu weiter unten). Es wäre aber wahrscheinlich nicht allzu schwierig, den Preisrahmen an andere Anbieter von dynamischen Stromtarifen anzupassen, da z.B. das österreichische Startup aWattar eine ähnliche API wie Tibber anbietet (siehe Link in der Kurzinfo). Die Make-Redaktion freut sich über Berichte von Personen, denen dies gelungen ist.

Täglich um kurz nach 13:00 Uhr holt sich der Preisrahmen die für den kommenden Tag neu festgelegten Bruttopreise (also inklusive Netzentgelte, Steuern usw.) und zeigt diese auf seinem ePaper-Display an. Da die Preise für den aktuellen Tag schon am Vortag festgelegt wurden (es heißt nicht ohne Grund Day-Ahead-Börse), müssen wir insgesamt auf dem Display die Preise von 36 Stunden darstellen. Wie auf den Bildern zu sehen ist, entspricht jeder Balken einer Stunde. Da ePaper-Anzeigen nur während der Aktualisierung Energie benötigen, reicht es aus, wenn wir den Rahmen mit einer CR123 Batterie betreiben. Die sollte deutlich länger als zwei Jahre halten.

Das Display hat ganz unten eine Statuszeile, dort wird die Software-Version, der WLAN-Empfangspegel, die Batteriespannung und der Zeitpunkt der letzten Aktualisierung angezeigt (Bild 1). Bei nicht vorhandenem WLAN oder zu geringer Batteriespannung wird im oberen Drittel des Displays eine Warnung angezeigt. Die beiden waagerechten Strichpunktlinien im Display zeigen den jeweiligen Mittelwert der beiden Tage von 0:00 bis 24:00 Uhr an. Denken Sie daran, dass auf dem Display bei dem links dargestellten Tag nur die Stunden ab 13:00 Uhr abgebildet werden und dadurch der angezeigte Mittelwert falsch erscheinen kann.

Kurzinfo

- **Schicker batteriebetriebener Rahmen, der die dynamischen Strompreise anzeigt**
- **Grundsatzinformationen über dynamische Strompreise**
- **Deep Sleep beliebig lange nutzen durch die Verwendung des ESP8266 Real-Time-Clock-Speichers**

Checkliste

Zeitaufwand:
ein Wochenende

Kosten:
90 Euro

Werkzeug

- **3D-Drucker** (nur für den Bilderrahmen)
- **Lötstation**
- **Software Arduino IDE**
- **USB-Seriell-Kabel-Adapter** (zum Programmieren)

Material

- **ESP8266-12 inkl. Adapterplatte**
- **Waveshare 7,5" ePaper-Display Hat Module V2 Kit 800x480**
- **2 Reset- und Programmiertasten**
- **3 Widerstände 10kΩ**
- **Gewinkelte Stiftleisten**
- **Batteriehalterung CR123**
- **Lochrasterplatine**
- **Jumperkabel zur Verbindung mit dem USB-Seriell-Wandler**

Mehr zum Thema

- Luca Zimmermann, ESP-Boards mit der Arduino IDE programmieren, Online-Artikel auf heise online
- Ulrich Schmerold, Günstiger IR-Lesekopf für Smart Meter mit Tasmota-Firmware, Online-Artikel auf heise online
- Uwe Rohne, Photovoltaik an der E-Auto-Wallbox – reloaded, Make 2/22, S. 52
- Clemens Gleich, Dynamische Strompreise nutzen: Ein Erfahrungsbericht mit Tibber, Online-Artikel auf heise+

Alles zum Artikel im Web unter
make-magazin.de/x1jx

Um an die Strompreisdaten zu gelangen, nutzen wir die vom skandinavischen Ökostromanbieter Tibber angebotene API-Schnittstelle (Application Programming Interface) zur Datenabfrage (siehe Link in der Kurzinfo). Die Tibber-API verwendet die Abfragesprache GraphQL. Die Antwort erfolgt als JSON-Objekt. Da wir immer nur eine konkrete Abfrage haben, braucht uns GraphQL nicht weiter zu interessieren, da die Abfrage statisch im C-Code steht (siehe dazu Listing 1 und die Zeile, die mit „String Payload“ beginnt).

Zur Authentifizierung muss zuvor ein „Tibber-Token“ einmalig unter Verwendung der Zugangskennung (in der Regel die eigene E-Mail-Adresse und das Passwort) generiert werden. Um gleich loslegen und testen zu können, ist im Sourcecode ein Demo-Token eingetragen. Dieser liefert nur Nettopreise aus der Vergangenheit ohne Netzentgelte.

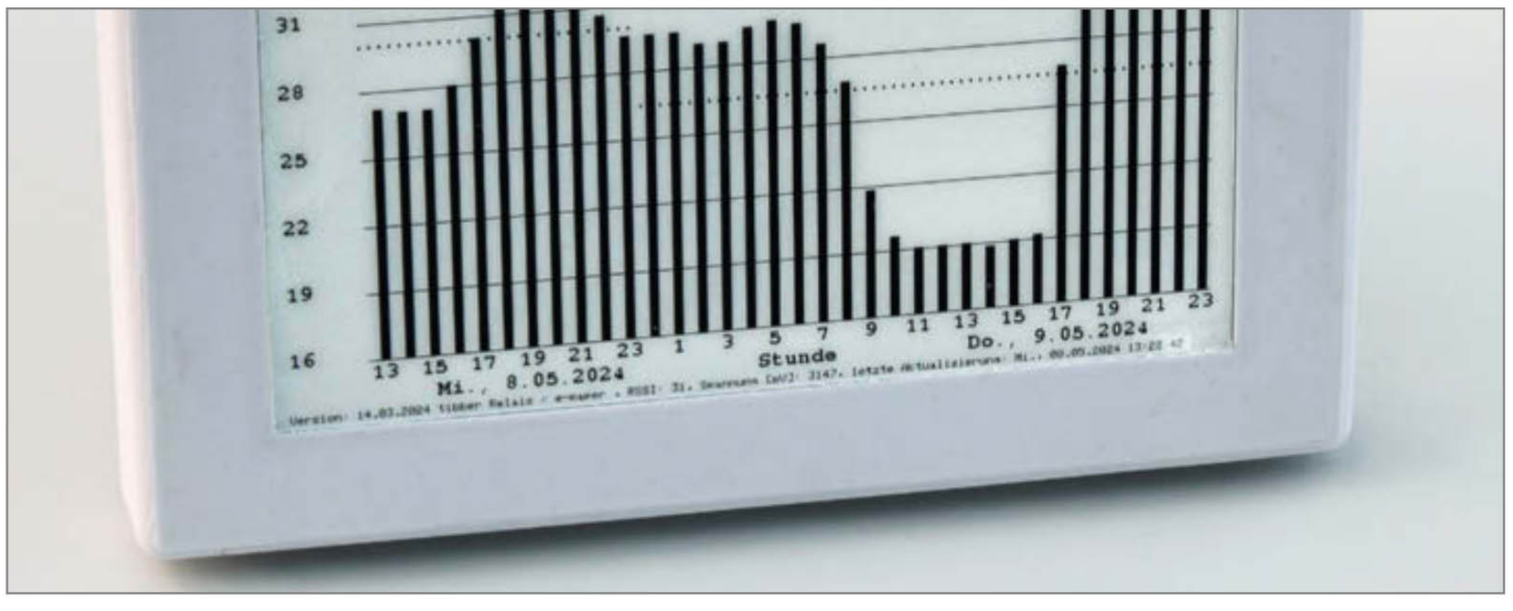
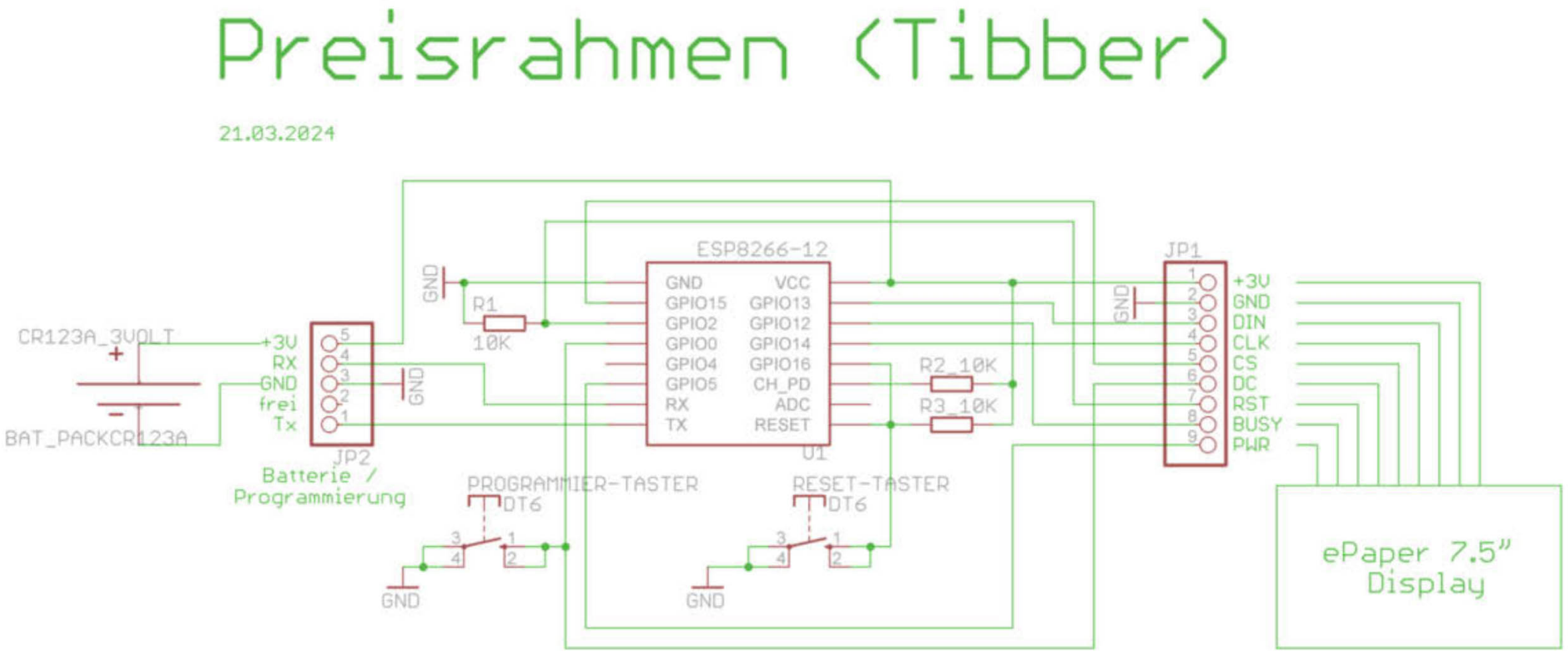


Bild 1: In der Statuszeile wird u.a. der Zeitpunkt der letzten Aktualisierung angezeigt.

Listing 1

```
void hole_tibber_preise() {
  WiFiClientSecure client; //HTTPS
  client.setInsecure(); //the magic line, use with caution
  HTTPClient https;
  strcpy(buf, "Bearer ");
  strcat(buf, token);
  https.begin(client, tibberApi);
  https.addHeader("Content-Type", "application/json");  // add necessary headers
  https.addHeader("Authorization",  buf);           // add necessary headers
String payload = "{\"query\": \"{viewer { homes { currentSubscription{ priceInfo{ today{ total  } tomorrow {
total  }}}}}}\"} ";
//char payload[100]; strcpy( payload, "{\"query\": \"{viewer { homes { currentSubscription{ priceInfo{ today{
total  } tomorrow { total  }}}}}}\"} " );
Serial.print("STRING PAYLOAD: "); Serial.println(payload);
int httpCode = https.POST(payload);
if (httpCode == HTTP_CODE_OK) {
  String response = https.getString();
 // strcpy(response, https.getString() );
  Serial.println(response);
  DeserializationError error = deserializeJson(doc, response);
  if (error) Serial.println("\n\n################################## Error parsing JSON response ################
##########\n");
  preise_aus_json();
} else {
  Serial.println("something went wrong");
  Serial.println(httpCode);
}
https.end();
  client.stop();  // Disconnect from the server
}
```

Bild 2: Der Schaltplan des Preisrahmens



Die Hardware

Die Hardware unseres Preisrahmens wird im Wesentlichen mit einem ESP8266-12 und einem 7,5" ePaper-Display realisiert. ESP8266-Prozessoren gibt es für weniger als 5 Euro und es ist wichtig, dass Sie für unser Projekt einen „nackten" kaufen, also keinen Wemos oder ähnliches. Der dort integrierte USB-Seriell-Wandler würde sonst die Batterie in sehr kurzer Zeit entladen. Außerdem benötigt man eine Adapterplatte, die manchmal zusammen mit dem Prozessor verkauft wird.

Das 7,5" ePaper ist bei Amazon für 62 bis 80 Euro erhältlich. Bei der Bestellung sollten Sie unbedingt darauf achten, dass auch das SPI-Interface mit dabei ist. Sonst brauchen wir nur noch ein Batteriehalter, eine Lochrasterplatte, einige Widerstände, einen USB-Seriell-Wandler (FTDI-Adapter) für die einmalige Programmierung und einen handelsüblichen Bilderrahmen mit mindestens 18 mm Tiefe. Wer keinen passenden Bilderrahmen findet, kann sich auch einen drucken. Die STL-Dateien dafür sind online (siehe Kurzinfo) abrufbar. Die recht überschaubare Verdrahtung ist auch im Schaltplan (Bild 2) dargestellt. Wie das Ganze am Ende aussehen soll, ist in den Bildern 3, 4 und 5 dargestellt. Um die Elektronik zu schützen, habe ich handelsüblichen Tic-Tac-Dosen eine neue Lebensaufgabe gegeben (Bild 3).

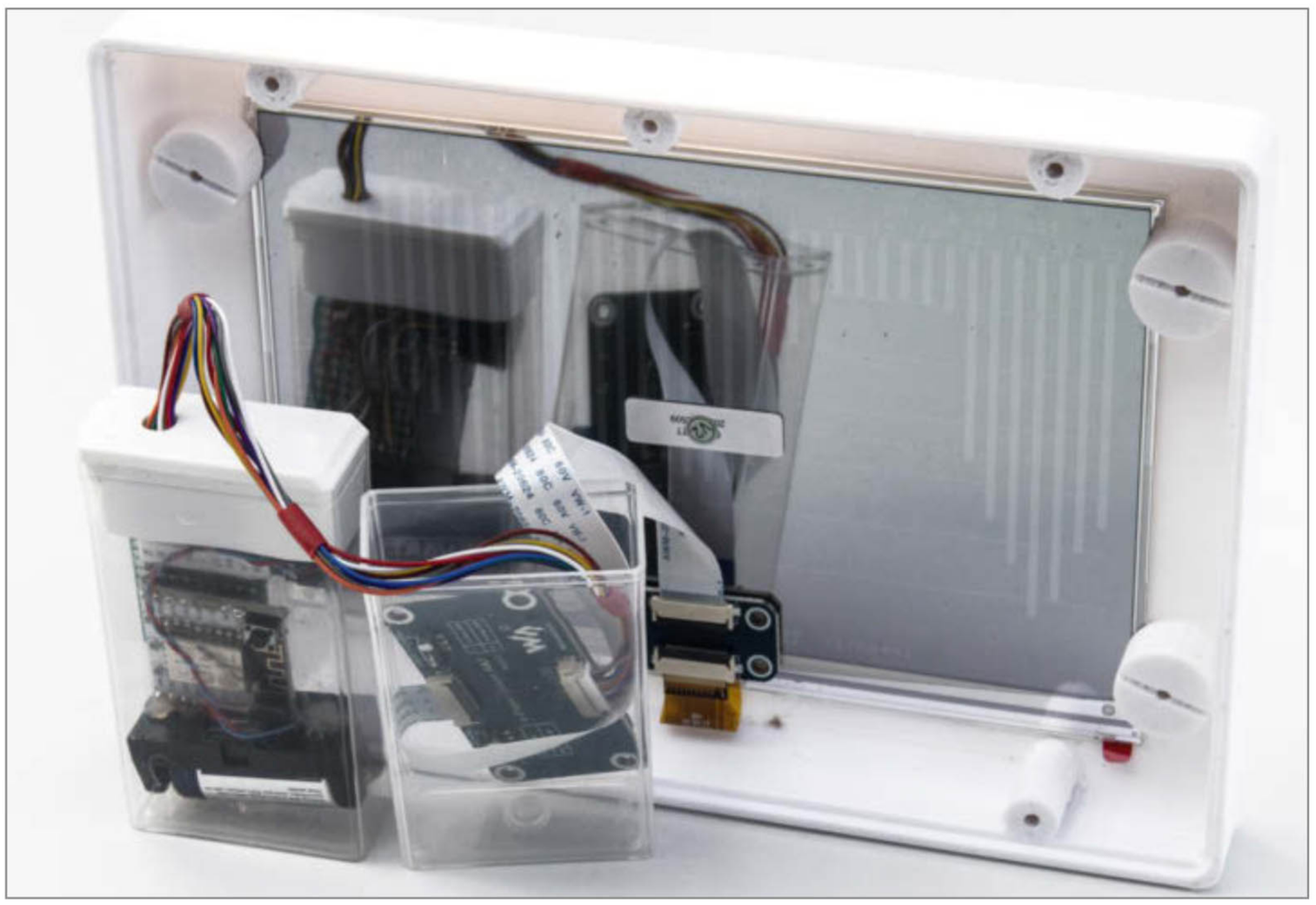


Bild 3: So sieht unser Preisrahmen von hinten aus. Zum Schutz der Hardware habe ich ein paar alte Tic-Tac-Dosen wiederverwendet. Ganz links befindet sich der ESP8266-12 zusammen mit der CR123-Batterie. In der rechten Tic-Tac-Dose ist das Interface-Board, das mit dem ePaper geliefert wird.

Die Software

Vor dem Übersetzen des Sourcecodes mit der Arduino IDE ist der Name des eigenen WLANs, das WLAN-Passwort und der eigens generierte Tibber-Token einzutragen. Ferner müssen alle verwendeten Bibliotheken installiert sein.

Um die Firmware auf den ESP zu flashen, brauchen wir einen USB-Seriell-Adapter und vier Jumper-Kabel (beidseitig weiblich). Achten Sie darauf, dass der Jumper auf dem Adapter auf 3,3 V gesteckt ist, um den ESP nicht zu grillen. Wichtig ist auch, dass die TX- und RX-Pins auf dem Adapter über Kreuz mit den entsprechenden Pins auf dem ESP verbunden werden. Also: TX mit RX und RX mit TX. Sonst brauchen wir nur noch GND und VCC beidseitig miteinander zu verbinden (Bild 6).

In der Arduino-IDE müssen Sie auch noch das ESP 8266-Board hinzufügen. Wie das geht, erklären wir in einem Online-Artikel (siehe Link in der Kurzinfo). Wenn Sie so weit sind, halten Sie zunächst den Programmier-Taster gedrückt und drücken dann den Reset-Taster. Damit wird der ESP in den Flashmodus versetzt und in der Arduino IDE kann nun unser Preisrahmen-Sketch kompiliert und auf dem ESP geflasht werden.

Stromsparen mit Deep Sleep

Um Strom zu sparen, wird der ESP nach Aufbau des WLAN, dem Holen der Internetzeit und der Tibber-Preise (inkl. Aufbereitung und Ausgabe) in einen Deep Sleep geschickt. Das Ganze

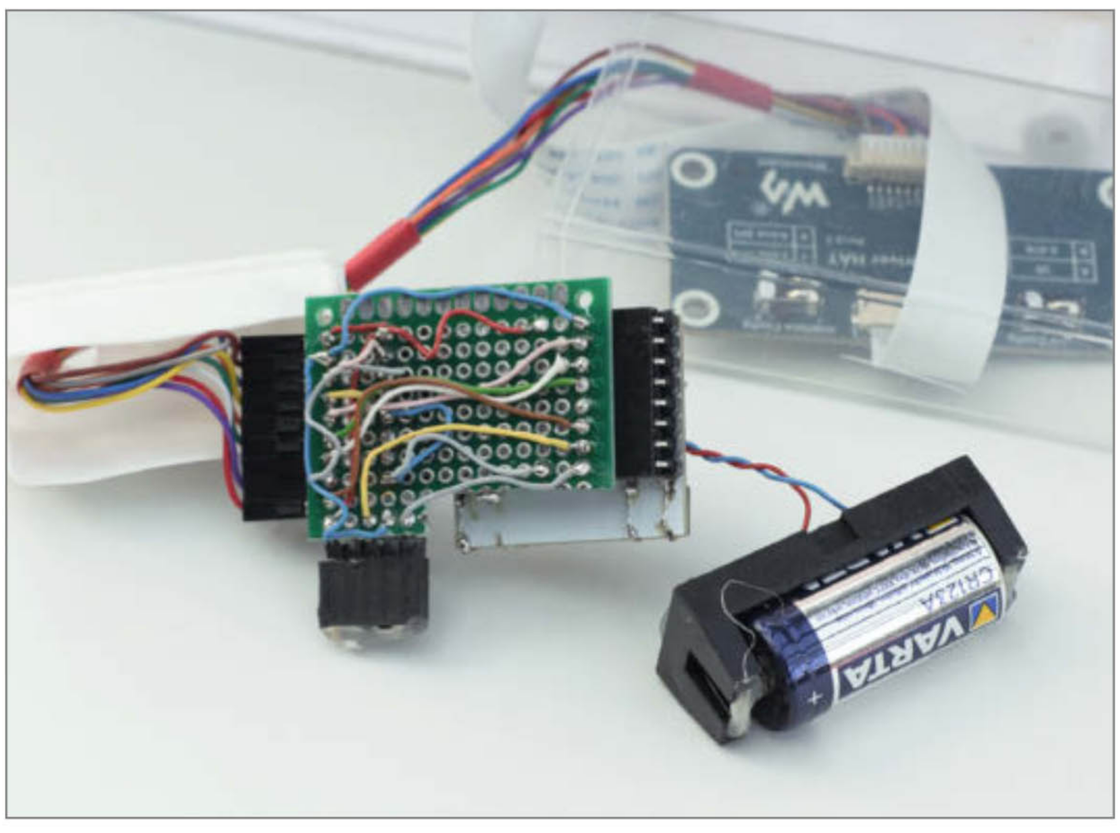


Bild 4: Die Lochrasterplatine von hinten, unten links der kombinierte Batterie- /Interface-Jumper. Wer das Ganze etwas schöner bauen möchte, findet online (siehe Kurzinfo) auch das Layout für eine einseitige Platine zum Selbermachen.

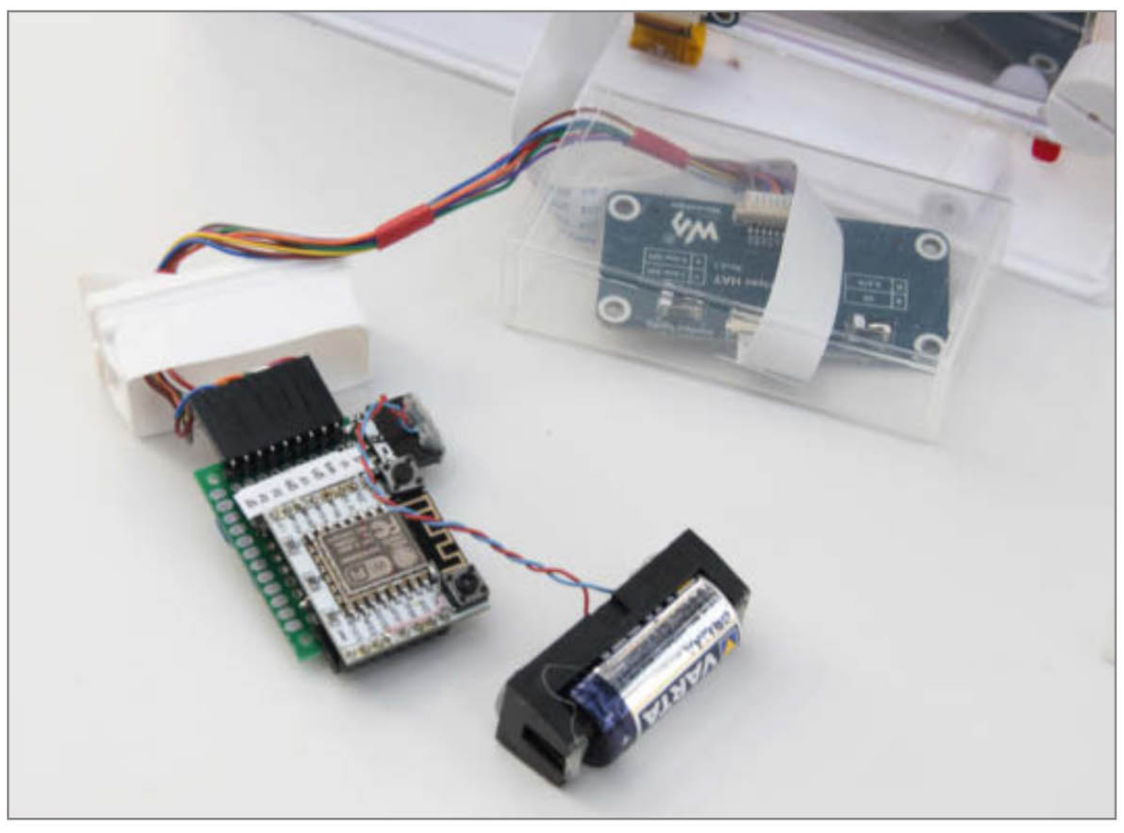


Bild 5: Der ESP wird auf die kleine weiße Adapterplatte gelötet, die von 2 mm Raster auf 1/10" Raster umsetzt. Die Platte hat auch zwei Pull-up-Widerstände in SMD und hier werden auch die Reset- und Programmiertasten angelötet.

Verwendete Bibliotheken

```
#include <GxEPD2_BW.h>                  ePaper Display
#include <Fonts/FreeMonoBold9pt7b.h>    Schriftfont für ePaper
#include <ESP8266WiFi.h>                „Standard ESP8266“
#include <ESP8266HTTPClient.h>          „Standard ESP8266“
#include <WiFiClientSecureBearSSL.h>    ermöglicht HTTPS-Zugriffe
#include <ArduinoJson.h>            JSON-Strings auflösen
#include <time.h>                       Internetzeit
```

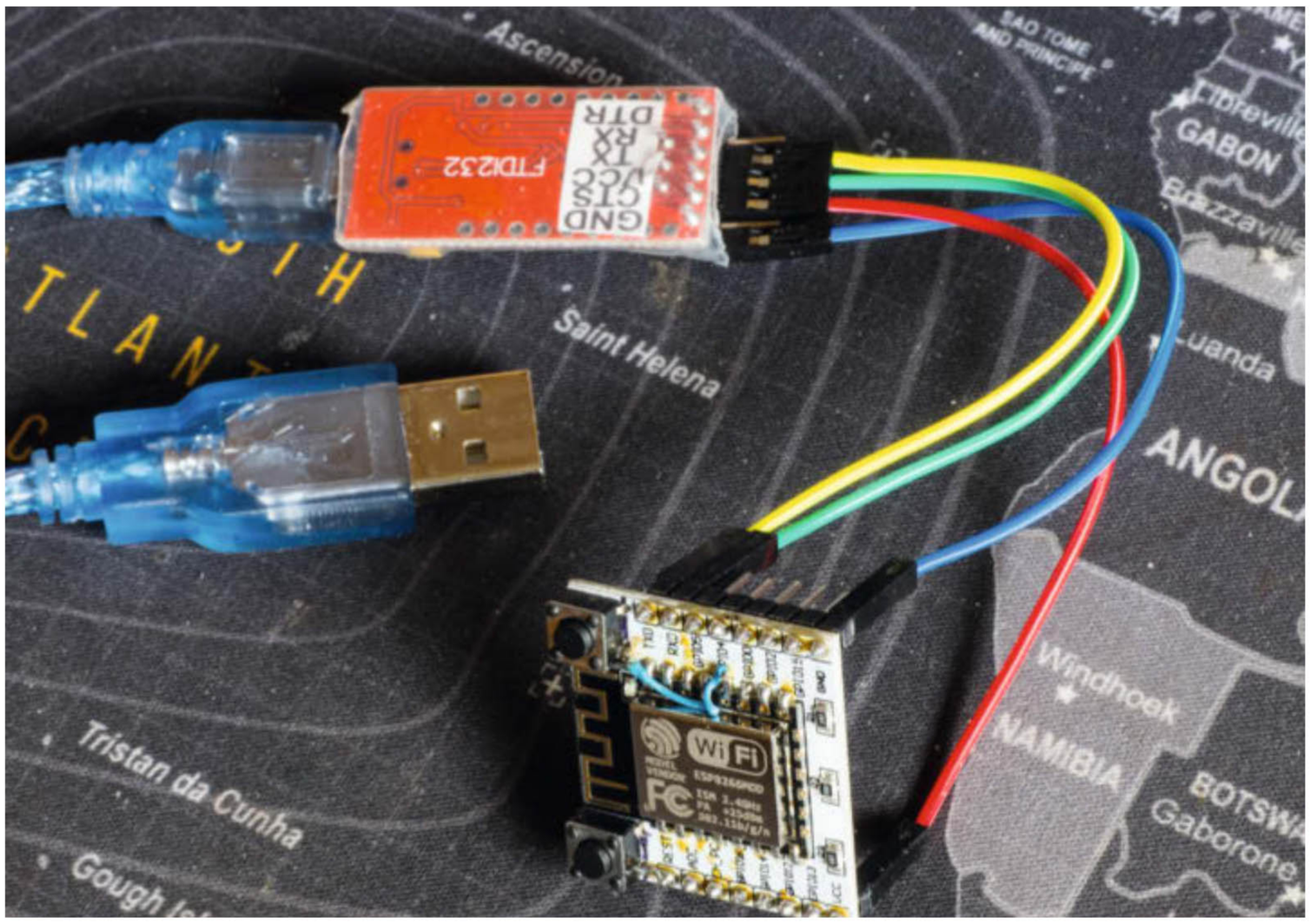


Bild 6: Beim Flashen sollen TX und RX über Kreuz miteinander verbunden werden.

Energiebilanz für den Preisrahmen

Aktionen pro Tag	Dauer [Sekunden]	Strom [mA]	Formel	mAh
1x WLAN+Display	10	100	10*100/3600	0,278
22x Aufwecken	0,003	30	22*0,003*30/3600	0,055
Deep Sleep	ca. 24*3600	0,02	86400*0,02/3600	0,720
SUMME mAh am Tag				1,053

dauert keine zehn Sekunden bei ca. 100 mA Stromaufnahme. Die maximal einstellbare Zeit des Schlafens beträgt 2^{32} Mikrosekunden, was einer guten Stunde entspricht. Das ist leider zu wenig, denn eigentlich ist nur alle 24 Stunden kurz nach 13 Uhr eine Aktion angesagt.

Da der ESP beim Beenden des Deep Sleep neu bootet und initialisiert, weiß er auch nicht, was vorher war. Die erste Überlegung wäre, den Status im EEPROM zu hinterlegen. Das sind allerdings im Kalenderjahr knapp 10.000 Schreibvorgänge, wobei maximal 30.000 erlaubt sind, bevor der EEPROM Schaden nimmt. Die Lösung liegt in den internen Registern der ESP-Uhr (welche ja auch das Aufwecken erledigt). In einem Teil dieser Register kann man Werte hinterlegen, die nur im Fall einer Stromunterbrechung verloren gehen würden, was bei unserer Anwendung nicht der Fall sein sollte. Der ESP rechnet also aus, wie lange er von der letzten Aktualisierung bis ca. 13:30 Uhr schlafen muss, wenn er insgesamt 22 Mal aufgeweckt wird. Wie das genau geht, sehen Sie in Listing 2.

Das Aufwecken zwischendurch findet ohne WLAN-Aktivierung statt: Counter aus dem Uhrenspeicher holen, um eins erhöhen und wieder schlafen. Das Ganze ist in etwa drei Millisekunden erledigt. Der Strombedarf beträgt ca. 30 mA, da das ePaper mittels Chip Select (PWR) auf inaktiv geschaltet ist. Unsere Energiebilanz sieht also wie in der Tabelle aus.

Eine CR123-Batterie hat eine Ladung von 1600 mAh. Jeden Tag verbraten wir davon 1,053 mAh. Somit ergibt sich eine rechnerische Batterielaufzeit von über vier Jahren: 1600 mAh / (1,053 mAh/Tag) = 1520 Tage.

Zusammenbau

Wer sich das Gehäuse drucken möchte, findet die STL-Druckdateien unter dem Link in der Kurzinfo. Das Display wird dabei mit vier gedruckten Zapfen gehalten. Das funktioniert ganz gut, weil diese Zapfen leicht oval sind.

Listing 2

```
strcpy(buf,ESP.getResetReason().c_str());
if(buf[0]=='D') { //D steht für DeepSleeep, weitere Faelle waeren z.B. 'P'ower on oder 'S'oftware watchdog . . .
  ESP.rtcUserMemoryRead(RTCMEMORYSTART, &bootcount, sizeof(bootcount)); //Counter aus dem ESP RTC
 (RealTimeClock) lesen
  if(bootcount<21) {
    bootcount++;
    ESP.rtcUserMemoryWrite(RTCMEMORYSTART, &bootcount, sizeof(bootcount)); //Serial.print("bootcount = ");
 Serial.println(bootcount);
    ESP.rtcUserMemoryRead(RTCMEMORYSTART+1, &deepsleeptime, sizeof(deepsleeptime));
    Serial.print(bootcount); Serial.print(" =bootcount, deepsleeptime= ");Serial.println(deepsleeptime);
    if(bootcount==21) {
      ESP.deepSleep(deepsleeptime, WAKE_RFCAL); delay(100); // nach dem Aufwachen WLAN ein
    }
    ESP.deepSleep(deepsleeptime, WAKE_RF_DISABLED); delay(100); //nach dem Aufwachen WLAN aus
  }
}
```

Es müssen also die vordere Bilderrahmenschale, die Rückwand und die vier Halterungen, die auf die Zapfen kommen, gedruckt werden (Bild 7). Das eigentliche System habe ich auf einer Lochrasterplatine gemäß Schaltplan zusammengelötet. Dabei habe ich, wie bereits im Abschnitt Hardware erwähnt, gewinkelte Stiftleisten am ESP verwendet, um die Einbauhöhe zu minimieren (Bild 8).

Wer sich diesen Schritt ersparen möchte, findet in der Artikelbeschreibung online auch das Layout für eine von mir entwickelte einseitige Platine zum Selbermachen. Diese ist allerdings ohne Gewähr, da ich keine Zeit hatte, sie zu testen.

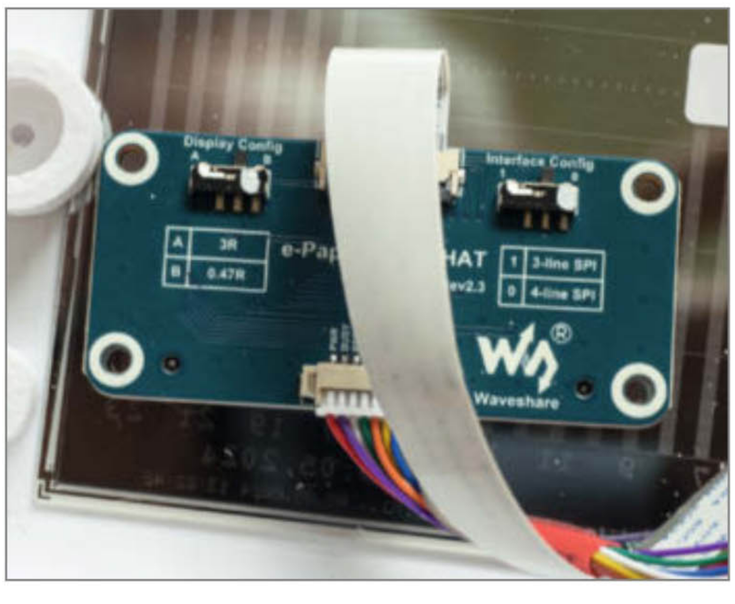


Bild 7: Ein ovaler Zapfen (10 x 11 mm) mit dazugehöriger ebenfalls leicht ovaler Buchse (10,5 x 11,5 mm Innen- und 20 mm Außendurchmesser) halten das Display auf dem gedruckten Rahmen. Auch gut zu erkennen ist das Interfacedisplay des ePaper mit den zwei Schalterstellungen (Display Config auf „B“ und Interface Config auf „0“).

Ausblick

Zusammen mit der Make-Redaktion plane ich, in einer der nächsten Ausgaben das Tibber-Relais vorzustellen, mit dem sich die Steuerung des Energieverbrauchs im Haushalt optimieren lässt. Das Tibber-Relais wird über eine einfache Weboberfläche bedienbar sein. Mit einem Slider wird der Nutzer einstellen können, bei wie vielen von den preiswertesten Stunden ein Ausgang eingeschaltet werden soll. Alternativ kann der Ausgang eingeschaltet werden, wenn der Strompreis unter den Tagesdurchschnitt fällt.

An der Ausgangsseite wollen wir sowohl mit Optokopplern als auch Relais die Steuerung einer Wärmepumpe ermöglichen. Auch das Schalten einer Shelly-Steckdose mit Befehlen per http wird möglich sein. Das Ganze wird auf einem Wemos D1 Mini mit USB-Netzteil laufen. *—mch*

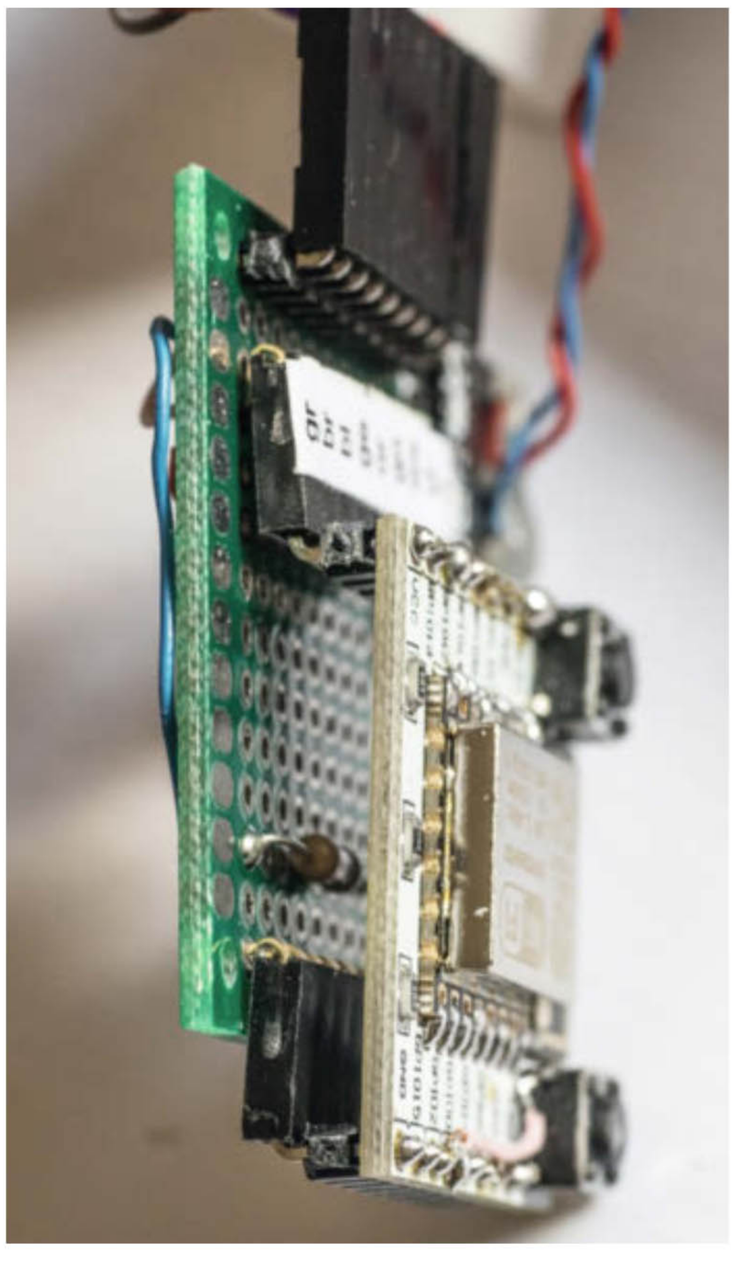
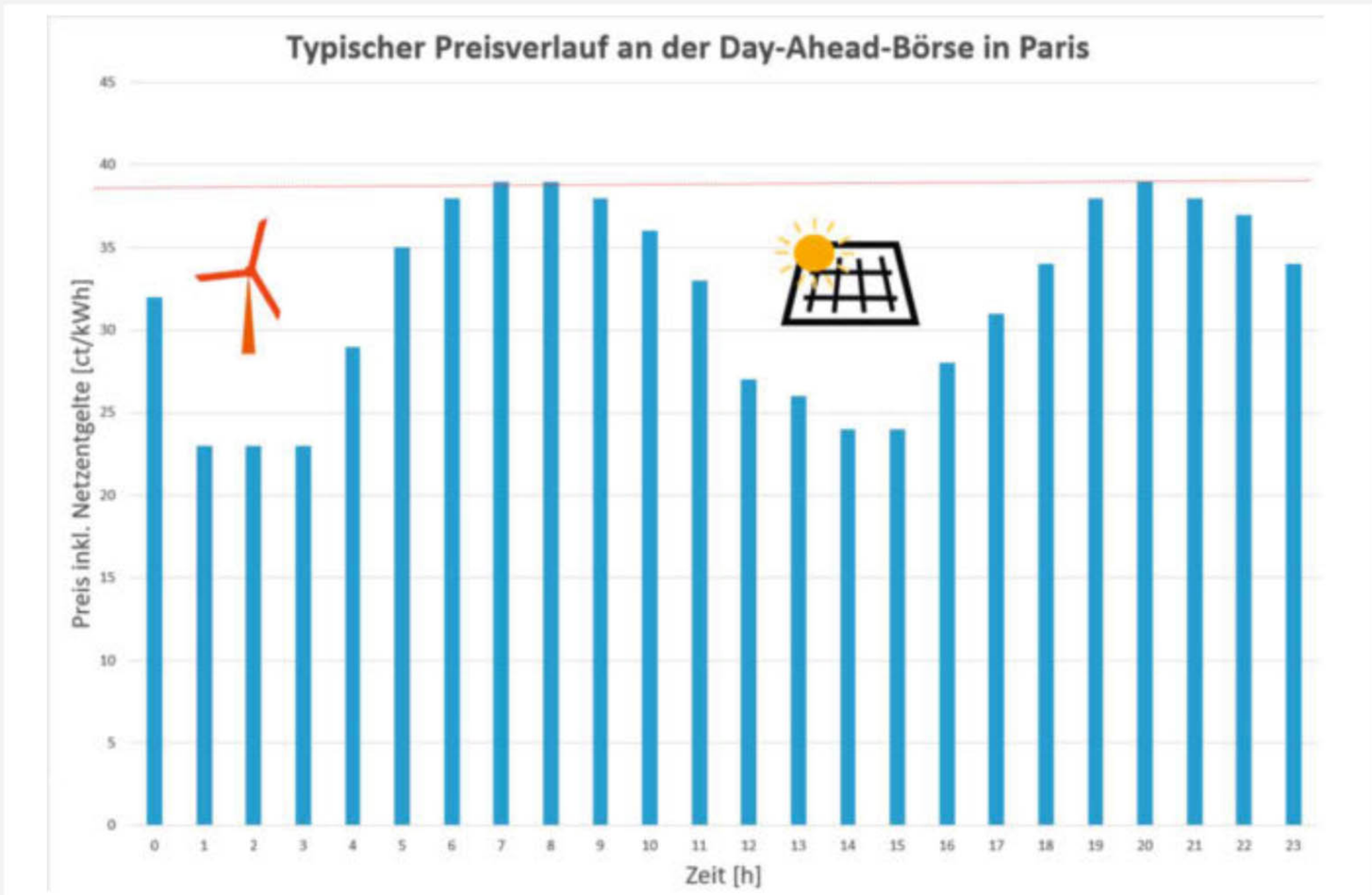


Bild 8: Mit gewinkelte Stiftleisten wird die Bauhöhe niedrig gehalten.

Dynamischen Stromtarifen gehört die Zukunft

Laut ihrer Befürworter bringen dynamische Stromtarife eine ganze Reihe von Vorteilen. Die sind gut für die Energiewende, können den Strompreis generell nach unten drücken und womöglich auch den Geldbeutel des einzelnen Verbrauchers entlasten.

Zurzeit gehen große Mengen an erneuerbarem Strom bei Überproduktion verloren, weil sie einfach nicht benutzt werden können oder die Übertragungskapazitäten fehlen. Das ist für alle Endverbraucher schlecht, denn die Energieerzeuger erhalten für den nicht eingespeisten Strom hohe Entschädigungszahlungen. Die Kosten dafür zahlen wir über die Netzentgelte, die im Strompreis inbegriffen sind. Nach Zahlen der Bundesnetzagentur gab es im Jahr 2022 ganze 5,8 Terawattstunden erneuerbaren Strom, der so quasi umsonst produziert wurde. Das entspricht mehr als einem Prozent des gesamten deutschen Jahresbedarfs an Strom oder dem Jahresbedarf von etwa zwei Millionen durchschnittlichen Haushalten.



Bei Wind und Sonne ergibt sich die im Bild dargestellte Preisdynamik, die es zu nutzen gilt.

Würde dieser überschüssige Strom genutzt, könnten die Kosten für die Endverbraucher sinken und gleichzeitig der Bedarf an fossiler Stromerzeugung reduziert werden. Im Grunde geht es also darum, den Stromkonsum in Zeiten zu verlagern, in denen es mehr grünen Strom gibt. Kein Wunder also, dass die Regierung die Einführung von Smart Metern vorantreibt und letztlich sogar durch neue Gesetze beschleunigt hat.

Bis 2032 sollen alle Zähler in Deutschland smart sein, und schon ab Anfang nächsten Jahres sind alle Stromanbieter dazu verpflichtet, ihren Kunden einen dynamischen Stromtarif anzubieten und sie darüber zu informieren. Wir Verbraucher können dabei alles so lassen wie es ist oder eben zu einem dynamischen Tarif wechseln.

Wie Anfangs im Artikel beschrieben, bedeutet ein dynamischer Stromvertrag, dass der Strompreis für die anstehenden 24 Stunden um ca. 13:00 Uhr an der Strombörse in Paris festgelegt wird. Starker Einfluss auf den Preis hat die Wind- und Sonnenscheinprognose (Angebot) und die Nachfrage. Dies führt in der Regel zu niedrigen Preisen in der Nacht und am Nachmittag. Morgens und abends dagegen, wenn in den meisten Haushalten viel Energie verbraucht wird, sind die Preise eher hoch.

Wann sich ein dynamischer Stromtarif lohnt

Wer die Vorteile von dynamischen Stromtarifen optimal nutzen möchte, muss sein eigenes Nutzungsverhalten anpassen. Größtes Einsparpotential haben Großverbraucher wie die Waschmaschine, die Wärmepumpe oder das Elektroauto.

Wer sein Elektroauto nachts (etwa zwischen 1 und 4 Uhr) lädt, kann Energiekosten im zweistelligen Prozentbereich einsparen. Als einfaches Rechenbeispiel lassen wir ein vollelektrisches Auto mit einer 60 kWh Batterie tagsüber bei einem Strompreis von 35 ct/kW voll aufladen. Das kostet genau 21 Euro. Nachts zieht ein Sturm auf, die Windmühlen drehen sich wie verrückt und produzieren Strom im Übermaß. An der Börse sinkt der Preis deshalb auf nur noch 20 Cent und wir hätten unser Elektroauto in diesem Fall für nur 12 Euro vollgetankt. Neun Euro wären eingespart. Wallboxen diverser Hersteller können mit dynamischen Stromverträgen umgehen und das Auto zur preiswerten Zeit laden. Beispiele dafür sind der go-eCharger, Fronius Wattpilot, Zaptec und OpenWB Wallboxen.

Unabhängig von der installierten Wallbox kann die Tibber-App mit fast allen gängigen eAutos kommunizieren, den SOC (State of Charge) ermitteln und das Auto zum günstigsten Zeitpunkt laden. An der Wallbox müssen dabei in der Regel keine Änderungen vorgenommen werden. Das Nutzungsverhalten von strompreisbewussten Menschen sähe dann etwa so aus:

- Zwischen 1-4 Uhr: Elektroauto laden, im Winter zusätzlich die Hausbatterie vollladen
- Zwischen 6-9 Uhr und ab 17 Uhr: „Strom“ sparsam „verbrauchen“
- Zwischen 13-16 Uhr: Waschmaschine, Trockner, Spülmaschine betreiben

Wer sich über die Möglichkeit freut, durch bewussten Stromkonsum Geld sparen zu können, sollte sich aber auch mit den damit verbundenen Risiken bekannt machen. Denn wenn das Angebot an Energie gering (Dauerflaute im Winter) und die Nachfrage hoch ist (abends wird gekocht), gehen die dynamischen Preise schnell durch die Decke. Auf Seiten wie energycharts.de kann man sich ein Bild davon machen, wie die Preise in der Vergangenheit schwankten und worauf man sich einlässt. Die Entwicklung insgesamt steht hier erst am Anfang.

Der schnelle Weg zum dynamischen Stromtarif

Um stundengenau abrechnen zu können, müssen dem Energieversorger natürlich die Verbräuche jeder Stunde mitgeteilt werden. Die noch in vielen Haushalten vorhandenen Ferraris-Zähler (schwarzer Kasten mit silbernem Drehrad) und auch die digitalen Zähler können dies nicht leisten. Der Zähler muss smart werden. Das geschieht, indem er eine Online-Anbindung zum Netzbetreiber erhält, der die Daten an den Energieversorger weitergibt. Die Übermittlung der Daten erfolgt über das Stromnetz, ein im smarten Zähler verbautes GSM-Modul oder über das Internet. Aktuell stehen viele Netzbetreiber, was den Einbau der Zähler betrifft, auf der Bremse, indem sie (sehr) hohe Einbaukosten verlangen oder gleich an Dienstleister verweisen. Ab 1.1.2025 wird die Zählergebühr bei normalen Haushalten bei 20 Euro/Jahr liegen, und der Einbau auf maximal 30 Euro gedeckelt sein. Zunächst werden Haushalte mit einem Jahresstromverbrauch von mehr als 6.000 kWh oder mit einer PV-Anlage größer als 7kW peak bevorzugt. Ab 2032 soll es dann nur noch smarte Zähler im Bestand geben.

Tibber Pulse haftet magnetisch am Zähler.

Keine Regel ohne Ausnahme

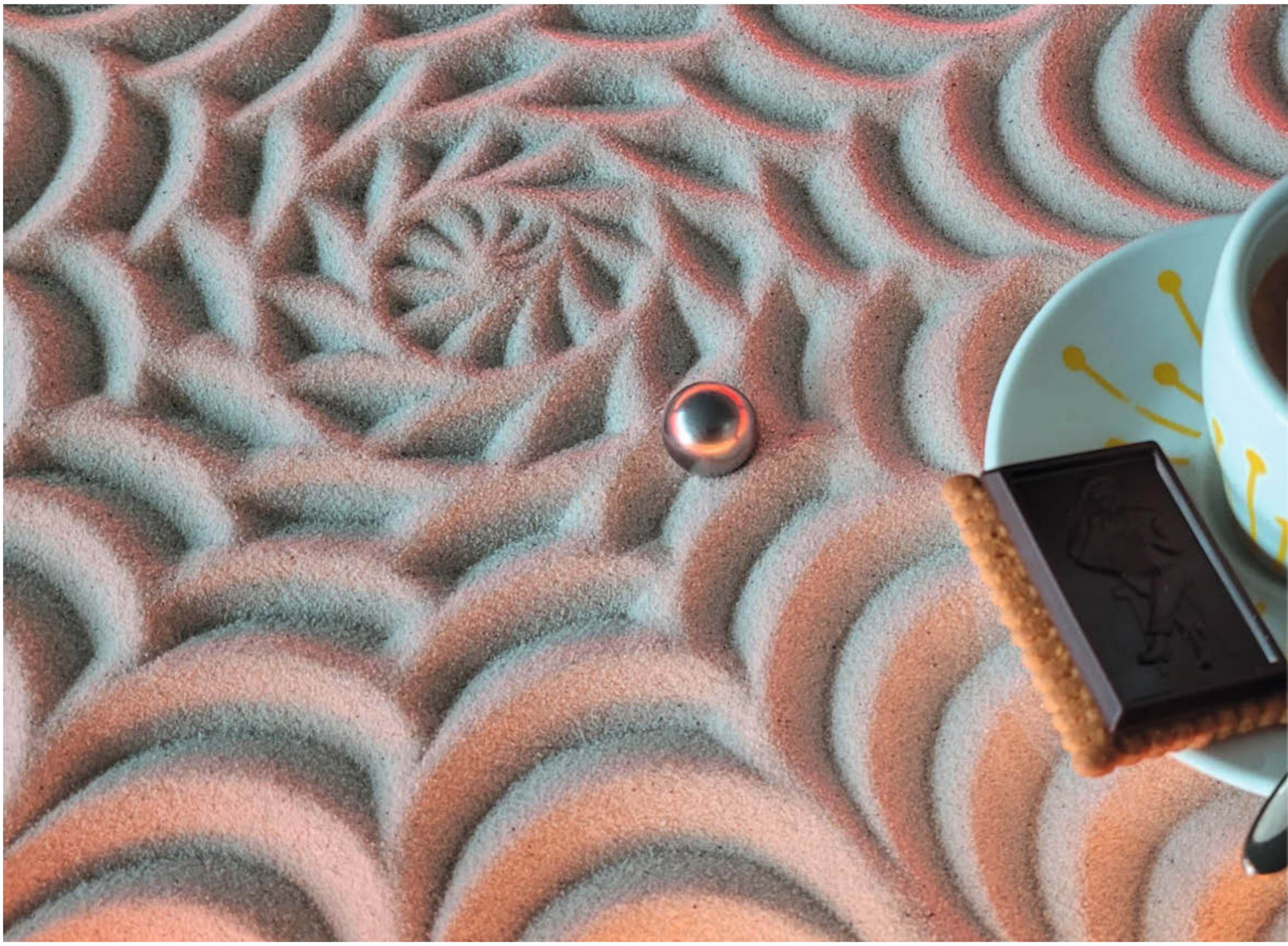
Wer nicht auf einen smarten Zähler warten will, findet beim Ökostromanbieter Tibber eine Alternative. Das Produkt Tibber Pulse ist ein infrarotgekoppeltes Zusatzgerät, das an einen „dummen" digitalen Zähler angeflanscht wird und diesen smart macht.

Ein Vorteil dieser Lösung ist, dass weder der Netzbetreiber noch ein Elektriker hier etwas an der Installation ändern muss. Der Tibber Pulse kostet um die 100 Euro, aber mit etwas Glück und Herumgesuche im Internet findet man aber auch günstigere Angebote. So können ADAC-Mitglieder beim Kauf die monatliche Tibber-Gebühr für ein halbes Jahr sparen und bei eBay werden für nur einen Euro Zugangscodes angeboten, die 50 Euro Rabatt auf den Tibber Pulse gewähren. Wer sich darauf einlässt, sollte allerdings sehr vorsichtig sein und möglichst sicherstellen, dass derselbe Code nicht an andere weiterverkauft wurde, was gegen die AGBs von Tibber verstoßen würde. Zuerst sollten Sie aber einen Blick auf Tibbers Whitelist werfen, um festzustellen, ob Ihr Zähler zusammen mit dem Tibber Pulse funktioniert (siehe Link in der Kurzinfo).

Zähler mit Taschenlampe freischalten

In der Regel müssen digitale Stromzähler (im Versorgerdeutsch „moderne Messeinrichtungen" genannt) erst freigeschaltet werden, bevor sie mehr anzeigen als den Energieverbrauch. Das hat datenschutzrechtliche Gründe, denn manche Zähler hängen an halböffentlichen Plätzen etwa im Keller eines Mietshauses. Um den Zähler freizuschalten, müssen Sie sich von Ihrem Netzbetreiber eine PIN geben lassen. Das ist in der Regel unkompliziert, wenn Sie die Zähler- und Vertragsnummer nennen. Haben Sie die PIN erhalten, müssen Sie diese über eine Taste am Zähler eingegeben. Alternativ, falls keine Taste vorhanden ist, kann die PIN durch Blinken mit einer Taschenlampe eingegeben werden. Wie das bei Ihrem Zählertyp genau geht, ist in einem Merkblatt des Netzbetreibers beschrieben. Viel Spaß beim Blinken!

Hier ist die Sende- und Empfangsdiode auf dem Stromzähler gut zu erkennen.



Der IKEA-Hack-Sandmaltisch

Nach dem Erfahrungsbericht über den Bau des Sandmaltisches wollten viele Leser mehr darüber wissen und der Ruf nach einer kompletten Bauanleitung wurde laut. Hier ist sie! Unser Autor hat keine Mühen gescheut und neben diesem Übersichtsartikel eine ausführliche Online-Schritt-für-Schritt-Anleitung, Stücklisten und CAD-Zeichnungen angefertigt.

von Benno Lottenbach

Kurzinfo

- Bau eines Sandmaltisches, Teil 1: Material und Zusammenbau
- Schritt-für-Schritt-Anleitung online
- Basierend auf einem IKEA-LACK-Tisch
- Allround-Projekt: Holzverarbeitung, 3D-Druck, Elektronik, Software

Checkliste

Zeitaufwand:
1–2 Wochenenden,
Druckzeit 50 Stunden

Kosten:
ca. 500 Euro

Material

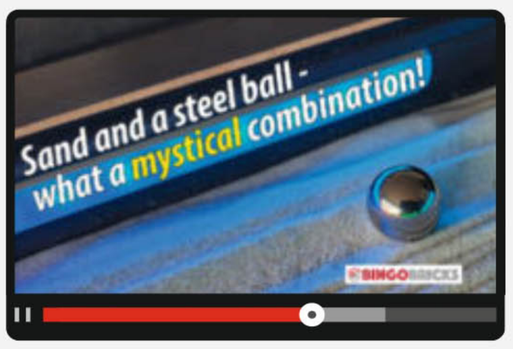
- **gemäß Materialliste im GitHub-Repository**

Werkzeug

- **3D-Drucker** mit min. 180 × 180 × 180 mm Bauraum
- **Multimeter**
- **Lötausrüstung**
- **Stichsäge**
- **Akkuschrauber, Bits und Bohrer**
- **übliches Maker-Werkzeug**

Mehr zum Thema

- Bernd Heisterkamp, Mobi-C, die mobile CNC-Fräse, Make 03/24, S. 26
- Dr. Ing. Armin Zink, IKEA-Matrix gehackt, Make 06/23, S. 64
- Benno Lottenbach, Sandmalerei im Couchtisch, Make 06/23, S. 72
- Video: Der Sandmaltisch in Aktion



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

In der Make 6/23 habe ich über die Entstehungsgeschichte meines Sandmaltisches basierend auf einem IKEA-LACK-Tisch berichtet. Die Idee war es, ihn auszuhöhlen, mit Sand zu befüllen und schließlich CNC-gesteuert eine Kugel im Sand rollen zu lassen. Die dadurch entstehenden Sandspuren formen sich mit jeder zusätzlichen Spur zu faszinierenden Bildern. Mit diesem Projekt ist es mir als höchste Auszeichnung gelungen, die ästhetischen Ansprüche meiner lieben Partnerin, der „Wächterin des Wohnzimmers“, zu erfüllen. Seitdem nimmt der Sandmaltisch einen zentralen Platz an prominenter Stelle in unserer Wohnung ein, wo er uns und unsere Gäste täglich mit seinen gemalten Kunstwerken entzückt.

Aufgrund vieler positiver Rückmeldungen habe ich mich dazu entschlossen, den Nachbau dieses Projekts für die Make-Leser zu erleichtern und eine entsprechend detaillierte und bebilderte Anleitung zu erstellen. Hierzu wurde die Konstruktion erst überarbeitet, dann nachgebaut und dabei dokumentiert. Die daraus entstandene PDF-Bauanleitung, inklusive Materialliste und Bilddateien für erste Sandbilder, findet ihr im GitHub-Repository (siehe Kurzlink) zu diesem Artikel. Auf den nachfolgenden Zeilen erläutere ich einige Grundüberlegungen und Projektdetails, die euch beim Nachbau unterstützen werden.

Die Idee eines Sandmaltisches ist grundsätzlich nicht neu. Im Zuge meiner Recherche bin ich auf einige Quellen gestoßen, die ähnliche Ideen umgesetzt haben. Dabei ist mir aber aufgefallen, dass ich durch meinen Start „auf der grünen Wiese“ einige Alleinstellungsmerkmale erreicht habe, die meinen Sandmaltisch (Abbildung 1) von ähnlichen Projekten unterscheiden: Ich nutze unter anderem einen IKEA-LACK-Tisch für das Projekt, statt selbst einen Holztisch dafür herzustellen. Das führt nicht nur zu weniger Materialkosten, auch die Anforderungen an das eigene handwerkliche Können fallen dadurch geringer aus. Man braucht keine Holzwerkstatt zu haben und auch kein Schreinermeister zu sein, um dieses Projekt realisieren zu können.

Ein weiteres, auf den ersten Blick nicht sichtbares Alleinstellungsmerkmal zeigt sich in der Verwendung der elektronischen Komponenten. Viele ähnliche Projekte verwenden einen Controller mit einer CNC-Firmware und einem zusätzlichen Raspberry Pi mit Octoprint, der das Abspielen der G-Code-Dateien übernimmt. Nebst den Kosten, die ein zusätzlicher Controller verursacht, sind bei diesem Setting eine erhöhte Komplexität und eine größere Fehleranfälligkeit gegenüber unserem Projekt zu erwarten. Unser Sandmaltisch hingegen kommt mit einem einzigen Controller aus. Sämtliche Hauptanforderungen wie eine webbasierte Bedienoberfläche, Integration ins lokale WLAN, G-Code-Kompatibilität und automatisches Abspielen von Bildern können damit dennoch erfüllt werden.

Stetiger Verbesserungsprozess

Der damals in der Make vorgestellte Prototyp bediente sich zu einem großen Teil der Kom-

Abbildung 1: Die Konstruktion unseres Sandmaltisches basiert auf dem populären LACK-Tisch von IKEA.

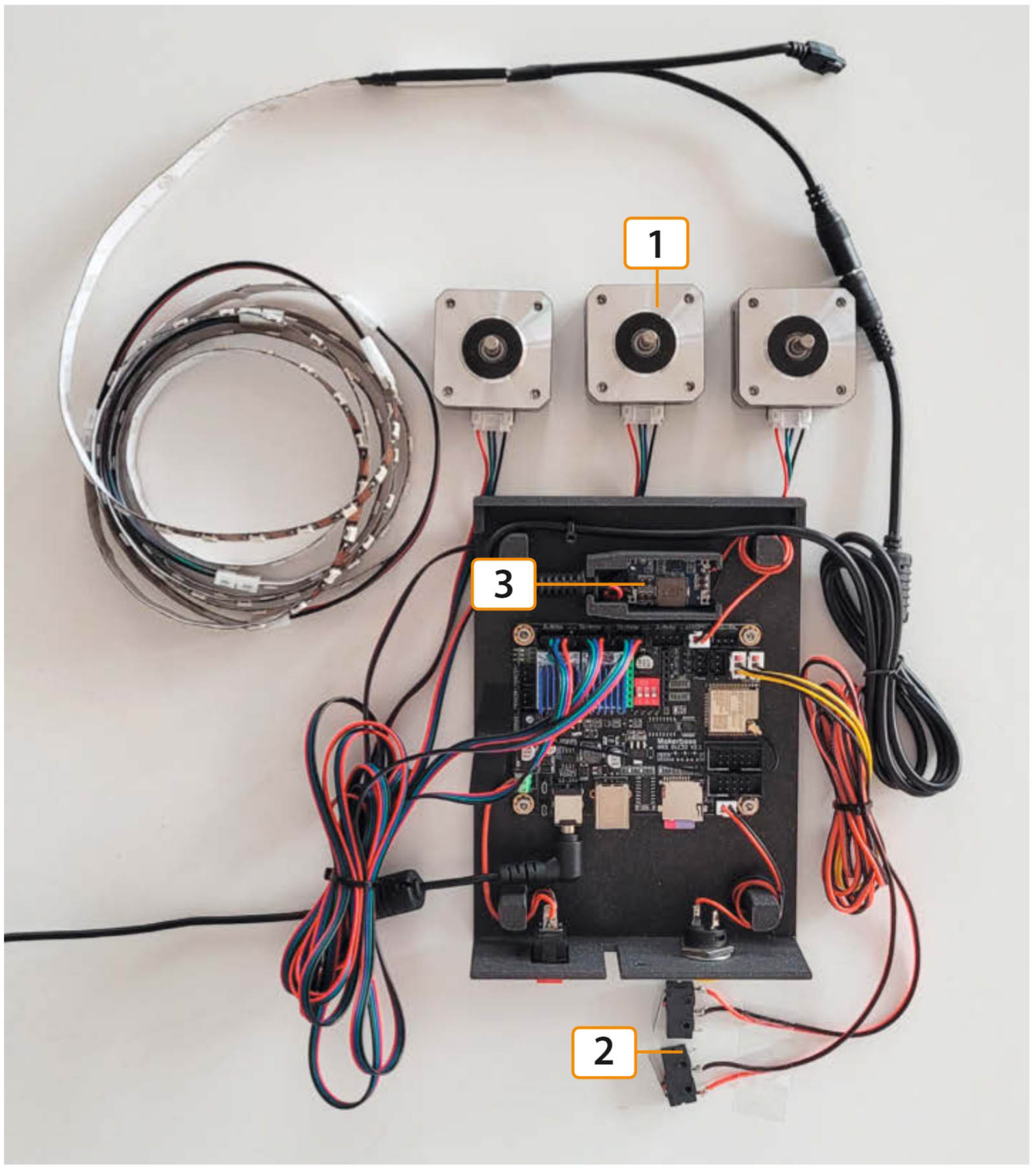


Abbildung 2: Die Elektronik des Sandmaltisches: Ein MKS DLC32-Controller steuert drei Stepper-Motoren (1). Zwei Limit-Schalter dienen als Endstopps (2). Die Versorgungsspannung für die LED-Streifen wird über einen Step-Down-Converter (3) direkt vom Board abgezapft.

ponenten aus meiner Kramkiste. Viele konstruktive Entscheidungen wurden spontan während der Entstehungsphase umgesetzt Inzwischen habe ich in vielen Dingen Klarheit geschaffen und das Projekt für den Nachbau stark vereinfacht. So wurden etwa die eingebauten Komponenten standardisiert, die Vielfalt an benötigten Schrauben und Muttern reduziert und eine Materialliste mit konkreten Angaben zu Bezugsquellen erstellt. Auch sind nun die Bohrlöcher möglichst symmetrisch platziert und haben wenig Durchmesservariationen.

Einige andere Optimierungen zielen dabei eher auf die Konstruktion ab, damit das Möbelstück etwas stabiler wird: Die physische Belastbarkeit wurde durch den Einsatz einer auf den Tischbeinen aufliegenden Zwischenplatte erhöht. Schließlich soll das Ding ja nicht nur zum Angucken sein, sondern auch seine Funktion als Beistelltisch erfüllen. Einen stabilen, störungsfreien Betrieb begünstigt auch das neue Gehäuse für die elektronischen Komponenten, das jetzt hängend unter der Zwischenplatte montiert ist. Dort behindert es die sich bewegenden Achsen und Kabel nicht. Zusätzlich hat die Umplatzierung eine bessere Zugänglichkeit der Bedienelemente erwirkt und die Verkabelung der Komponenten stark vereinfacht.

Als Krönung des ganzen Optimierungsprozesses habe ich einen Ersatz für die Original-Firmware des eingesetzten Controllers gefunden: Der Sandmaltisch nutzt nun die Firmware FluidNC, die bestens dokumentiert ist, stabil läuft und als Open-Source-Software zur Verfügung steht.

Werkzeuge und Voraussetzungen

Einen Sandmaltisch zu bauen, erfordert einige Werkzeuge und Fertigkeiten. Das Projekt setzt voraus, dass man über einen 3D-Drucker mit einem minimalen Druckraum von 180 × 180 × 180 mm verfügt (z. B. Prusa Mini). Weiter sollte man zwingend Zugriff auf Holzbohrer, Stichsäge und Lötkolben haben und ein Multimeter in Griffweite haben. Für den Nachbau sind Vorkenntnisse hilfreich bezüglich 3D-Druck, Holzverarbeitung (Vollausschnitt aussägen), Metallverarbeitung (Aluminium-LED-Profile kürzen), Lötarbeiten (Verlängern von Kabeln), Elektronik (Referenzspannung von Motortreibern einstellen), der Präparation von LED Streifen und dem Aufspielen von Firmwares auf Mikrocontroller.

Ein weiterer wichtiger Punkt ist vor der Bestellung der Materialien ebenfalls unbedingt zu beachten: Nebst einer IKEA-Filiale und einem Baumarkt ist es von großem Vorteil, wenn auch ein fähiger Glaslieferant in der Region verfügbar ist. Dieser sollte in der Lage sein, eine 6-mm-Glasplatte aus Sicherheitsglas millimetergenau in der gewünschten Größe

zuschneiden und liefern zu können. Ebenfalls lohnt es sich, vorher zu prüfen, ob das in der Materialliste aufgeführte LED-Set von Paulmann verfügbar ist, denn die Anleitung stützt sich auf genau dieses spezifische Produkt. Wer andere LED-Streifen verbauen möchte, muss sich selbst einen Weg zum Ziel bahnen.

Materialbezug

Bevor wir mit dem Bau loslegen können, muss das Material beschafft werden. In der Materialliste (im GitHub-Repository des Artikels) ist jede Komponente inklusive aller Schrauben und sonstiger Kleinteile aufgeführt. Jedes Bauteil ist mit einem Link zu einer möglichen Bezugsquelle versehen. Da die Vielfalt der zu beschaffenden Materialien trotz aller Optimierungen immer noch recht groß ist, habe ich mich dazu entschlossen, hauptsächlich AliExpress als Bezugsquelle anzugeben. Dadurch können Maker aus dem gesamten deutschsprachigen Raum von einheitlichen, spezifischen Materialspezifikationen profitieren und dabei sehr viele Komponenten aus einer Quelle beziehen. Die sperrigen Baumaterialien (Hölzer, Glas) oder sehr spezifische Produkte wie d-c-fix-Klebefolie beschafft man sich aber am besten im regionalen Baumarkt.

Bei der Bestellung über AliExpress sollte man auf einige Dinge achten, um Enttäuschungen zu vermeiden: Erstens ist die korrekte Auswahl der Produktvariante anzugeben, wenn man die Artikel in den Warenkorb legt. Die Unterschiede der Varianten sind teilweise minimal und nicht auf den ersten Blick ersichtlich. Auch auf die Stückzahl ist achtzugeben. Braucht man zum Beispiel, wie für unser Projekt, zwei Kabel mit „JST XH"-Steckern, so sind nicht zwei Stück in den Warenkorb zu legen, sondern nur eines. Die Kabel werden nämlich in Einheiten zu 10 Stück verkauft, und die Anzahl gilt pro Verkaufseinheit. Der geringe Preis verleitet hier oft zur Annahme, dass der Preis pro Stück gilt.

Bezüglich der Versandvariante lohnt es sich vielfach, ein paar Euro mehr für „AliExpress Standard Shipping" zu bezahlen, statt der supergünstigen Versandmethoden für ein paar Cent. Man bekommt dafür die Ware einigermaßen zuverlässig innerhalb der versprochenen Frist von meist 14 Tagen, im Gegensatz von bis zu drei Monaten beim günstigen „Cainiao Super Economy"-Versand. Eine Lieferverfolgung, die einen großzügigen Toleranzbereich aufweist, ist bei „AliExpress Standard Shipping" meistens inbegriffen.

Je nach Staat, in den importiert wird, kommen noch Einfuhrumsatzsteuer, je nach Warenwert landesabhängig Zoll und eventuell Auslagen für den inländischen Transporteur dazu. Bitte beachtet das bei eurer Bestellung und informiert euch vorher, damit es keine unangenehmen Überraschungen gibt.

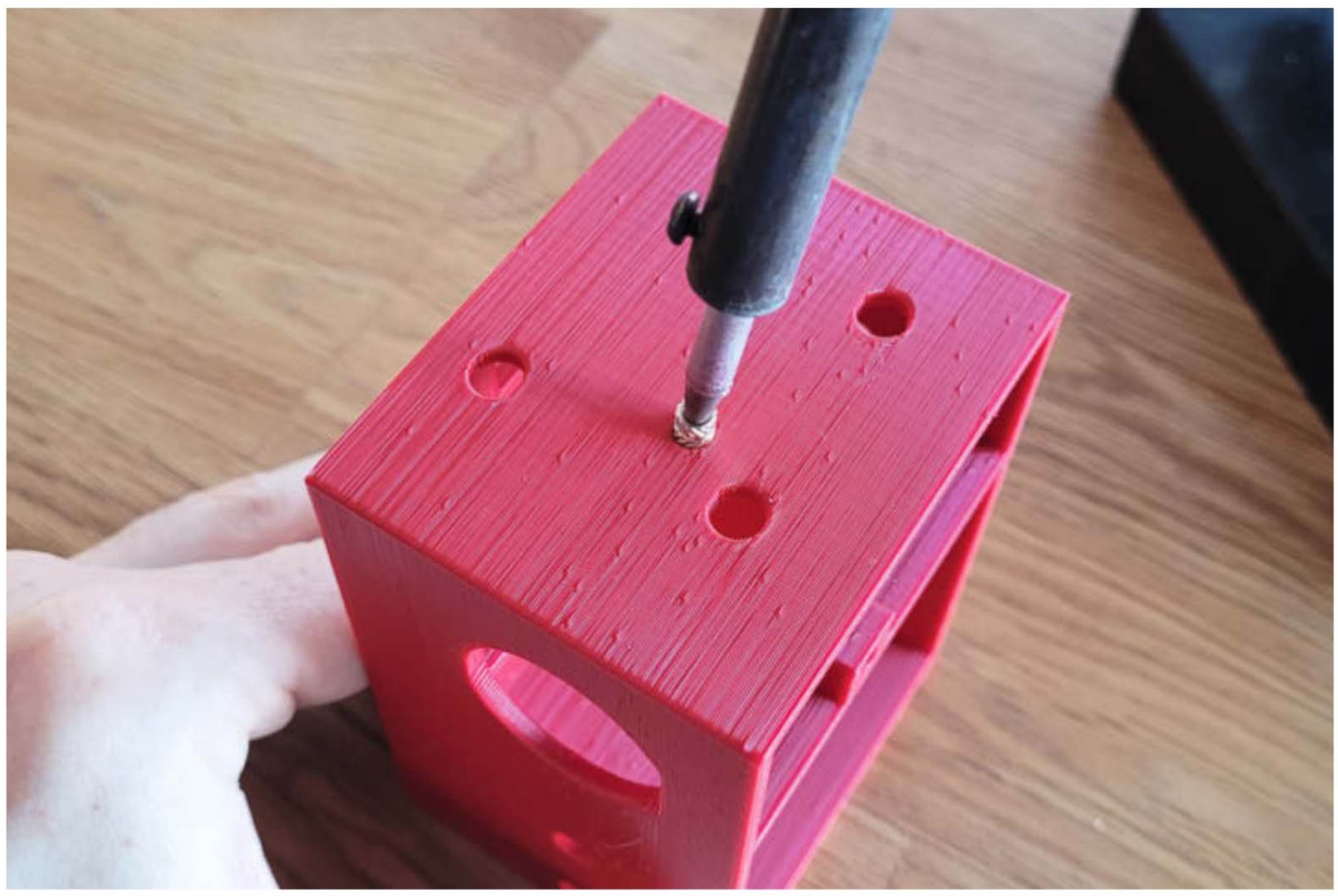


Abbildung 3: Gewindeeinsätze lassen sich leicht mit dem Lötkolben in die 3D-gedruckten Bauteile einschmelzen.

3D-Druck-Bauteile

Die Zeit, bis die bestellten Materialien eintreffen, sollte man mit dem 3D-Druck der Bauteile überbrücken. Sämtliche Bauteile können mit PLA oder PETG gedruckt werden. Für alle sichtbaren Bauteile empfehle ich schwarzes Filament. Bei den nicht sichtbaren Bauteilen ist die Farbe natürlich unbedeutend. Ich habe mich für Rot entschieden, damit diese auf den Fotos im Artikel und der Bauanleitung besser zu erkennen sind. Blau wählte ich für alle gedruckten Werkzeuge.

Die 3D-Daten der Bauteile stehen sowohl als STL- sowie als STEP-Dateien auf GitHub zur Verfügung. Mit den STEP-Dateien ist es sogar möglich, allfällige individuelle Anpassungen an den Bauteilen selbst vorzunehmen. Wer den PrusaSlicer für die Aufbereitung der Druckdateien verwendet, kann die ebenfalls bereitgestellten 3MF-Dateien nutzen, in denen die Modelle bereits richtig für den Druck ausgerichtet sind. Sämtliche Bauteile wurden so konstruiert, dass ein Druck ohne Stützen möglich ist, wobei ein Druck mit einer Layer-Höhe von 0,15 mm bei kritischen Überhängen zusätzlich hilft.

Die vier Eckteile der Konstruktion haben ein relativ hohes Gewicht zu tragen. Deshalb druckte ich diese Bauteile mit drei Perimetern statt der üblichen zwei. Insgesamt arbeitete mein Prusa MK4 für alle Bauteile rund 50 Stunden und er verbrauchte dabei etwa 1,2 kg Filament.

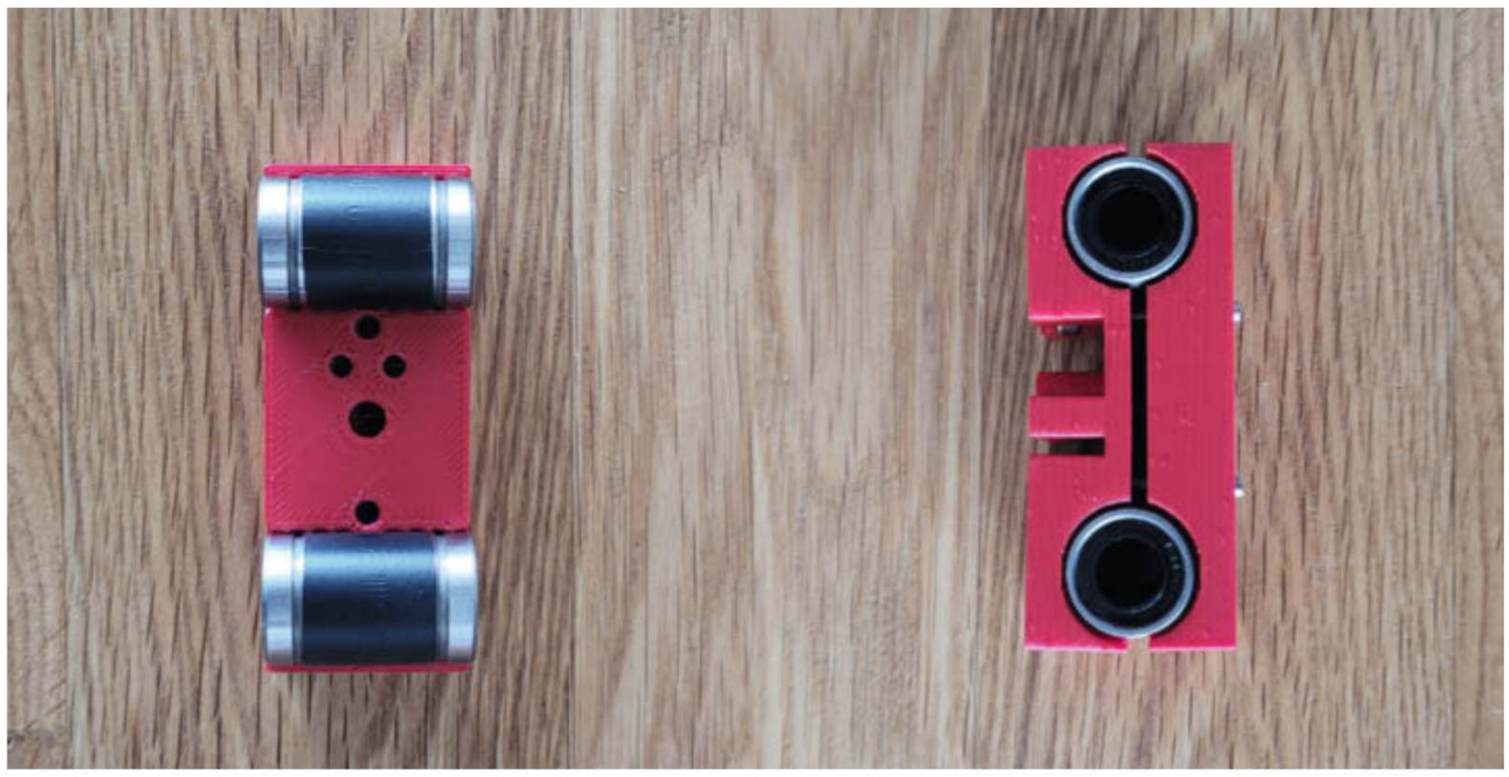


Abbildung 4: Die Linearlager werden für einen besseren Grip mit einer Schicht Isolierband umwickelt und anschließend auf dem X-Schlitten festgeklemmt.

Abbildung 5: Die Tischunterseite wird mit d-c-fix-Klebefolie beschichtet. Ein Spachtel hilft, allfällige Luftblasen nach außen zu streichen.

Abbildung 6: Mit der Bohrschablone aus dem 3D-Drucker lassen sich die Bohrlöcher präzise platzieren, ohne zu Winkel und Bleistift greifen zu müssen.

Abbildung 7: Mit dem Werkzeug aus dem 3D-Drucker lässt sich schnell, einfach und präzise eine Schnittlinie für die Stichsäge markieren.

Elektronik und Firmware

Herzstück der Elektronik ist das Mikrocontroller-Board MKS DLC32 von Makerbase. Es ist mit einem ESP32 bestückt und bietet volle WLAN-Unterstützung. Wer den in der Materialliste bereitgestellten Link verwendet, bekommt das Board mit drei TMC2208-Treibern. Diese sind besonders gut für unser Projekt geeignet, da sie die drei „NEMA 17"-Schrittmotoren im Pancake-Format praktisch geräuschlos antreiben können. Die auf dem Board befindlichen DIP-Schalter stellen wir alle nach unten auf OFF, womit die Motoren mit 1/8 Schrittweite arbeiten. Da ein Sandmaltisch im Gegensatz zu 3D-Druckern keine ultrapräzise Positionierung erfordert, reicht das für unsere Zwecke vollends.

Genauso verhält es sich auch mit dem Maximalstrom, der mithilfe der Stellschraube auf den TMC-Treibern einzustellen ist. Verwendet man die in der Materialliste empfohlenen Motoren im Zusammenhang mit TMC2208-Treibern, so kann die Referenzspannung auf niedrige 0,5 Volt eingestellt werden. Das schränkt das maximale Drehmoment der Motoren deutlich ein, was einen schonenden Betrieb ohne Gefahr auf Überhitzung der Motoren sicherstellt.

Die drei Motoren werden so an den Ports X-Motor, Y1-Motor und Y2-Motor des Boards angeschlossen, dass die roten Adern der Kabel jeweils auf der rechten Seite des Boards liegen (siehe Abbildung 2). Daher auch hier noch einmal der Hinweis: Werden die Stecker verpolt eingesteckt, drehen die Motoren in die falsche Richtung, was dazu führt, dass die bereitgestellten Beispieldateien nicht wie gedacht funktionieren. Ebenfalls sehr wichtig: Niemals Stepper-Treiber oder -Motoren anschließen, wenn der Controller unter Strom steht!

Die Limit-Schalter werden ebenso mit dreipoligen „JST XH"-Steckern versehen. Etwas verwirrend ist hier, dass die Anschlüsse für die Limit-Schalter auf dem Controller drei Pins aufweisen, ein Schalter aber klassischerweise nur zwei Anschlüsse miteinander verbindet. Da also ein Pin auf dem Board für unsere Zwecke überflüssig ist, schneiden wir die rote Ader des „JST XH"-Steckers einfach ab. Wir nutzen die beiden verbliebenen Adern und verbinden sie mit dem NO- und dem C-Anschluss des Limit-Schalters.

Die LED-Streifen schneiden wir in fünf Stücke nach der in der Anleitung angegebenen Länge und verbinden diese anschließend wieder mit den LED-Verbindungskabeln. Für die Energieversorgung der LED-Streifen nutzen wir den 12/24V-Anschluss des Boards. Da die in der Materialliste empfohlenen LED-Streifen von Paulmann aber 5 Volt benötigen, muss die Spannung über einen Step-Down-Converter auf 5 V abgesenkt werden. An diesem kommt ein Power-Jack-Kabel (auch Hohlstecker oder

Barrel-Jack genannt) zum Einsatz, an das die LED-Streifen angeschlossen werden können.

Beim Löten der Anschlüsse an den Step-Down-Converter ist unbedingt auf die korrekte Polung zu achten: Da der „JST XH"-Stecker nur in einer Ausrichtung in die Buchse gesteckt werden kann, kommt die rote Ader auf den Minuspol der Buchse am Board zu liegen. Entsprechend muss also die rote Ader mit dem Minuspol des Step-Down-Converters verbunden werden! Das ist unüblich, steht doch in der Regel Rot für den positiven Anschluss einer Komponente. Wer sich an den vertauschten Farbcodes der Verdrahtung stört, kann die beiden Kontakte im „JST XH"-Stecker vorsichtig tauschen.

Ein Ein-/Ausschalter darf natürlich auch nicht fehlen. Wir verwenden hierzu einen üblichen Kippschalter. Das für den Anschluss des Schalters am Board notwendige 2-Pin-Schraubterminal löten wir vorher auf das Board. Weiter verbinden wir den RESET-Anschluss des Boards mit einem geeigneten Drucktaster. Damit wird sich der in Betrieb befindliche Controller per Knopfdruck neu starten lassen.

Auf den Controller laden wir über die USB-Schnittstelle die Firmware FluidNC in der Version 3.7.8. Hinzu kommt eine im GitHub-Repository bereitgestellte, spezifisch für den Sandmaltisch parametrierte Konfigurationsdatei namens „config.yaml", die wir ebenfalls auf den Controller spielen. Ob alle elektronischen Komponenten korrekt funktionieren, lässt sich mit einem einfachen Test prüfen. Ergänzend zu den Anweisungen in der Anleitung habe ich hierzu einen kurzen YouTube-Clip auf meinem Kanal BINGOBRICKS erstellt, der den Testablauf anschaulich demonstriert.

Vorbereitung der Bauteile

In der Regel bin ich es bei meinen Projekten gewohnt, die Schrauben beim Verbinden von gedruckten Teilen direkt ins Filament zu drehen. Üblicherweise funktioniert das gut, solange man die Schrauben mit Gefühl festzieht und diese nicht allzu oft wieder gelöst werden müssen. Unser Sandmaltisch hingegen wird durchaus ein paar Jahre das Wohnzimmer schmücken. Daher wird er wohl das eine oder andere Mal für Wartungsarbeiten demontiert werden. Bei der Überarbeitung des Sandmaltisches entschloss ich mich deshalb, an den bei Wartungsarbeiten beanspruchten Bohrlöchern Gewindeeinsätze (Abbildung 3) für die Schraubverbindungen einzusetzen.

Diese Gewindeeinsätze lassen sich leicht mit dem Lötkolben in die Bauteile einschmelzen. Hierzu ist der Lötkolben ungefähr auf die Filament-Drucktemperatur einzustellen. Mit der heißen Spitze drückt man nun einfach die Gewindeeinsätze sanft in die Löcher. Selbstverständlich ist darauf zu achten, dass man

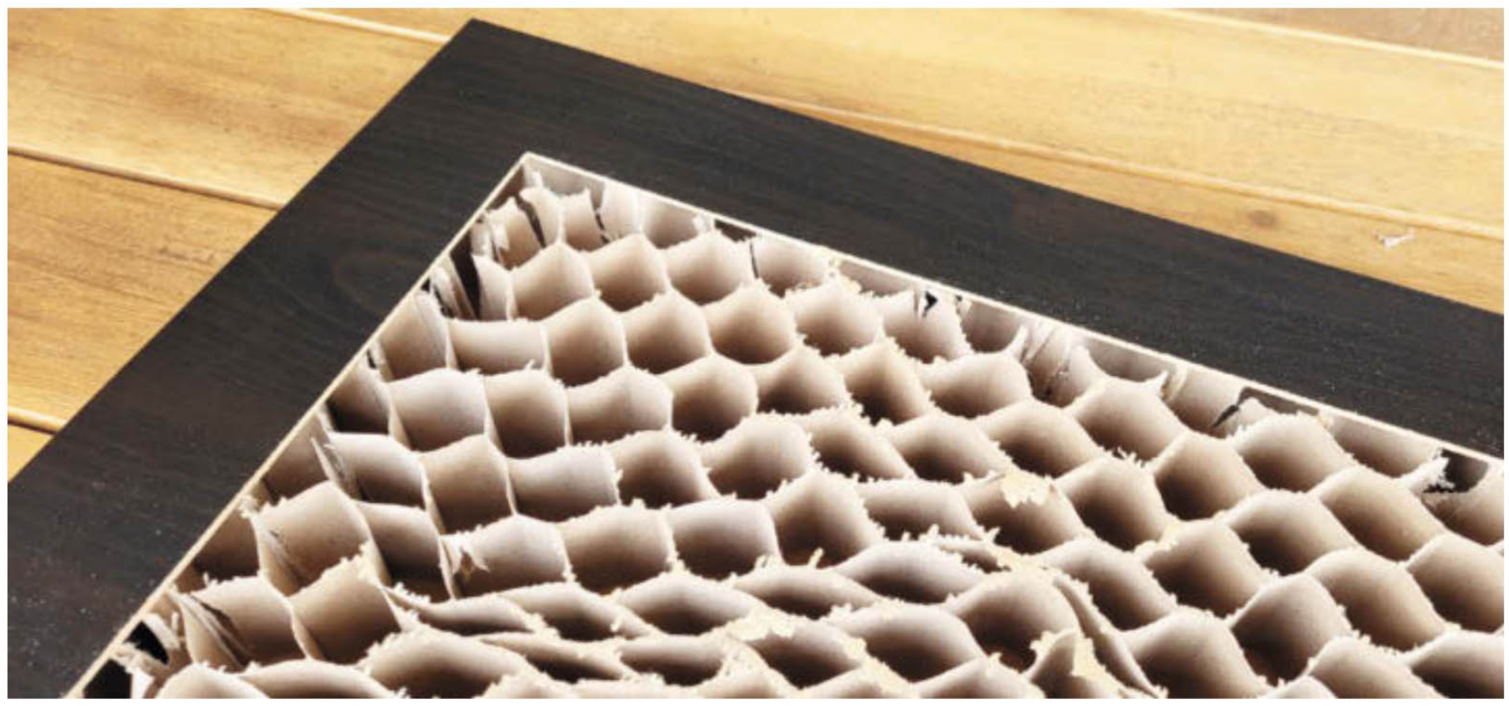


Abbildung 8: Die LACK-Tische sind innen mit einem Wabenmuster aus Pappe gefüllt.

Abbildung 9: Beschädigungen an der Schnittkante und die Reparatur mit d-c-fix-Klebefolie.

Abbildung 10: Die ausgeschnittene Tischplatte. Im Rahmen befinden sich die Kanthölzer. Den Boden habe ich zur Schalldämmung mit Moosgummi ausgelegt.

diese nicht zu weit oder schief eindrückt und damit die Konstruktion des Bauteils beschädigt. Das passiert leider schneller als erwartet.

Anschließend bestückt man die Bauteile mit den Stepper-Motoren, den passenden Antriebs- und Umlenkrollen sowie den Endschaltern. Alle Linearlager (Abbildung 4) werden mit ein paar Schichten Isolierband umwickelt, bevor sie in die entsprechenden Halterungen gelegt und festgeklemmt werden. Das weiche Isolierband erhöht die Griffigkeit der Quetschverbindung und hält so die Lager zuverlässig an Ort und Stelle.

Sägen, bohren und beschichten

Um den technischen Zwischenbau seitlich abzudecken, verwendete ich MDF-Platten in 3 mm Stärke. Da diese MDF-Platten sichtbar sind, sollten sie schwarz sein, ähnlich dem IKEA-Tisch. Nach langem Herumprobieren mit verschiedenen Beschichtungen habe ich mich schließlich dazu entschlossen, die Platten mit d-c-fix-Klebefolie (Abbildung 5) zu bedecken. Damit lassen sich nämlich auch die seitlichen Schnittkanten der MDF-Platten kaschieren. Auch die Zwischenplatte lässt sich mit der Klebefolie einseitig beschichten. Die rohe Fläche dieser Spanplatte wäre zwar nur aus der Froschperspektive sichtbar, insofern könnte man auch auf eine Beschichtung der Spanplatte verzichten. Aber da wir die Folie ohnehin auf der Materialliste stehen haben, nutzen wir sie, um diesen kleinen Makel zu beheben. Beim Anbringen der Klebefolie lohnt es sich übrigens, eine Person um Hilfe zu bitten.

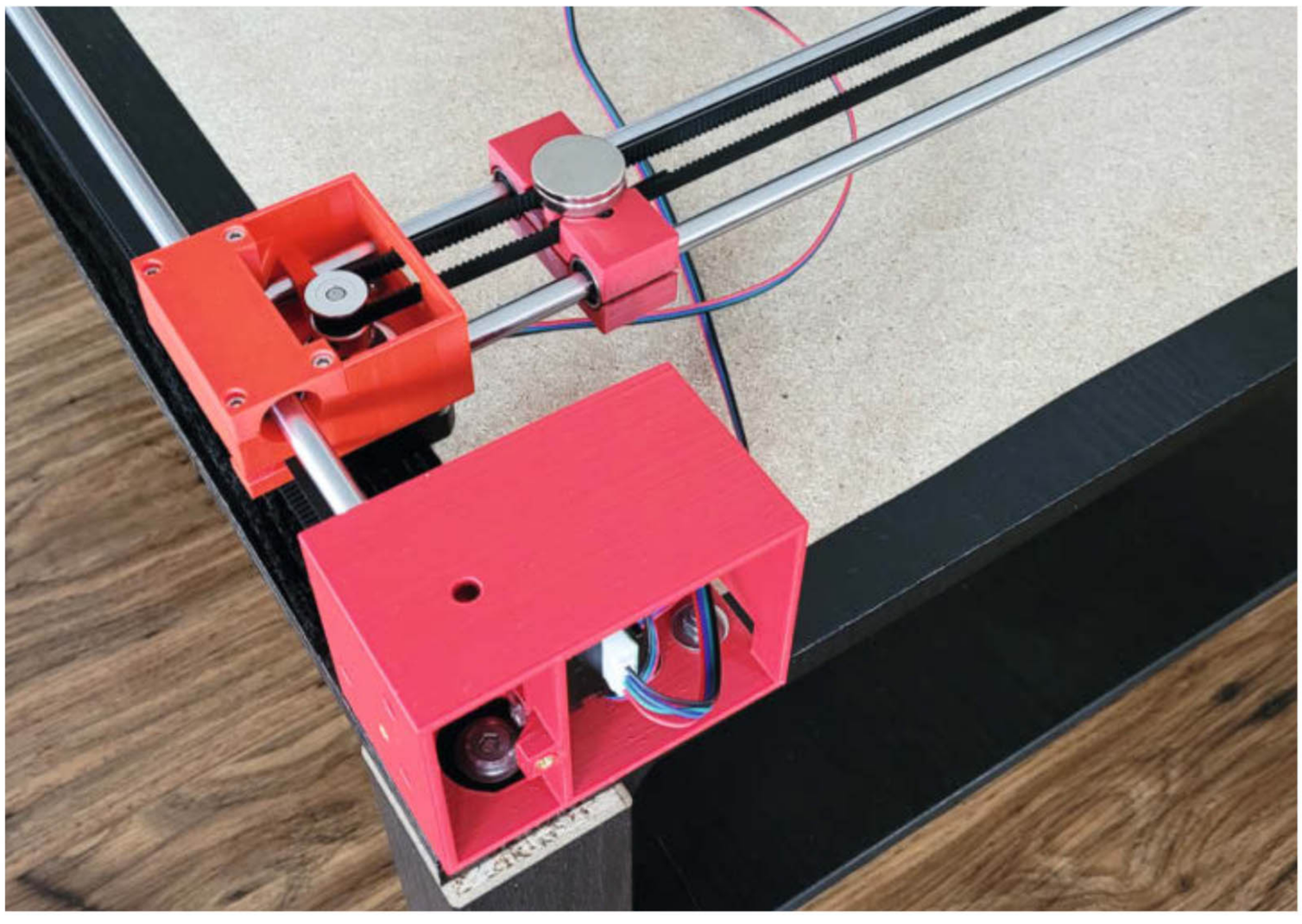


Abbildung 11: Die X- und Y-Achse im technischen Zwischenbau des Tisches

Sobald die Platten mit der Folie beschichtet sind, können die Löcher gebohrt werden. In diesen Situationen möchte ich meinen 3D-Drucker keinesfalls missen. Statt mühsam mit Winkel und Bleistift die Positionierung der Bohrlöcher abzumessen, erstelle ich lieber mit dem 3D-Drucker passende Werkzeuge, die diese Arbeit erleichtern. Abbildung 6 zeigt den Einsatz eines Werkzeugs aus dem 3D-Drucker in Form von Bohrschablonen: Diese werden einfach an den Kanten der Holzplatten ausgerichtet. Anschließend kann mit dem Bohrer in passendem Durchmesser durch die Aussparung der Schablone gebohrt werden.

Auch beim Entfernen der oberen Wand der Tischplatte leistet ein Werkzeug aus dem 3D-Drucker gute Dienste, wie in Abbildung 7 ersichtlich ist: Ein weißer Farbstift wird in der Ecke des Werkzeugs angesetzt, mit dem man nun über die gesamte Länge der Tischkante entlangfährt, und schon hat man eine Linie in exaktem, parallelem Abstand zur Tischkante gezogen. Dasselbe Werkzeug kann gleichzeitig auch genutzt werden, um tangential zur so gezeichneten Schnittlinie eine Bohrung für das Ansetzen der Stichsäge anzubringen.

Diese Bohrungen und Linien erleichtern das Anbringen eines Vollausschnitts in der Tischplatte. Die genaue und absolut rechtwinklige Ausführung des Vollausschnitts ist meines Erachtens die kniffligste Arbeit am Projekt. Die Herausforderung besteht darin, dass am Ende des Projekts der Ausschnitt mit einer eingelassenen Glasplatte bedeckt werden soll. Demzufolge sollte er millimetergenau und absolut rechtwinklig ausgeführt werden.

Für ungeübte Handwerker könnten sich hierzu ein paar Testschnitte lohnen: Einfach in der Mitte des Tisches einer geraden Linie entlang einige Probeschnitte mit der Stichsäge ausführen. Dieser Teil des Tisches wird ja anschließend sowieso ausgeschnitten, also kann man ruhig ein paar Testschnitte darauf machen. Und noch was: Dass die Stichsäge die untere Tischwand nicht auch aussägen darf, versteht sich wohl von selbst. Deshalb sollte man sie mit einem in der Länge passenden Sägeblatt bestücken.

Wird die obere Wand der Tischplatte vorsichtig entfernt, so kommt darunter ein Wabenmuster aus Pappe (Abbildung 8) zum Vorschein, das ebenfalls vollständig entfernt werden kann.

Leider ist mir die Ausführung des Vollausschnitts nicht einwandfrei gelungen, wie man in Abbildung 9 gut erkennen kann. Ich habe deshalb alle vier Schnittkanten des Vollausschnitts mit d-c-fix-Klebefolie abgeklebt, um die Beschädigungen zu kaschieren und damit den Tisch zu retten. Daraus hat sich ein ca. fünf Millimeter breiter, sichtbarer Klebefolienrand rund um den Ausschnitt ergeben. Zum Glück weist die verwendete Klebefolie der Variante „Blackwood“ eine leichte Faserstruktur auf. Diese sieht nicht nur schmuck

aus, sie macht auch die Schnittnähte fast unsichtbar. Schlussendlich bin ich mit der Reparatur sogar so zufrieden, dass ich diesen Arbeitsschritt inzwischen standardmäßig empfehlen kann.

Zur Verstärkung der Tischplatte und als Füllmaterial werden Kanthölzer (Abbildung 10) in den Tischrahmen geklebt. Daran lassen sich dann später ebenso die gedruckten LED-Profil-Halterungen anschrauben. Um Kratzgeräusche der rollenden Kugel auf dem Sand zu verhindern, wird eine dünne Schicht Moosgummi auf den Boden der Tischplatte gelegt. Die Übergänge an den Nähten der Moosgummimatten decken wir mit Gaffa-Tape ab.

Da der Sandmaltisch durch den technischen Zwischenbau um rund 8 cm höher als der originale IKEA-Tisch ist, können bei Nichtgefallen die Tischbeine gekürzt werden, um zurück auf die originale Tischhöhe zu gelangen. Hierzu sägen wir die Tischbeine auf eine neue Länge von 31 cm, wobei man bezüglich der Tischhöhe natürlich individuelle Vorlieben walten lassen kann. Die nun offenen Enden der Tischbeine schließen wir mit einer Abdeckung aus dem 3D-Drucker. Wer es nicht geschafft hat, alle Tischbeine in der gleichen Länge zurechtzusägen, kann sich erneut mit dem 3D-Drucker behelfen und entsprechende Tischbeinabdeckungen drucken, die den Längenunterschied wieder ausgleichen.

Von Einzelteilen zum Möbelstück

Nun können alle vorbereiteten Komponenten zusammengesetzt werden. Die LED-Profil-Halterungen werden an die im Tischrahmen angeklebten Kanthölzer geschraubt. Die zurechtgeschnittenen LED-Profile, mit bereits eingeklebten LED-Streifen, setzen wir in die Profil-Halterungen. Das Anschlusskabel des LED-Streifens führen wir durch eine eigens gebohrte Öffnung in den technischen Zwischenbau des Tisches.

Der Zusammenbau des Tischunterbaus ist einfach: Die Tischbeine werden an die mit Folie bezogene Zwischenplatte angeschraubt, mit Verstrebungen stabilisiert und mit der originalen IKEA-Ablageplatte bestückt. Das Gehäuse mit den elektronischen Komponenten wird an der Tischunterseite hängend festgeschraubt. Ein Klappmechanismus erlaubt hierbei später, die Steuerungsbox zu öffnen, die Motoren- und Sensorkabel anzuschließen oder ein USB-Kabel einzustecken. Alle für den Betrieb relevanten Kabel, insbesondere auch das Stromversorgungskabel, werden in den technischen Zwischenbau geführt. Das Stromkabel kann später diskret an einem Tischbein entlang zur Steckdose geführt werden. Ein USB-Kabel ist übrigens für den Betrieb nicht notwendig. Wenn man aber für Wartungszwecke einmal mit USB-Verbindung zum Controller arbeiten möchte, kann das Kabel durch eine Aussparung in der Blende direkt nach außen geführt werden.

Schließlich müssen wir uns noch um die Montage des technischen Zwischenbaus kümmern. An jeder Ecke des Tisches (Abbildung 11) werden hierzu die 3D-gedruckten Eckteile festgeschraubt, die die Y-Motoren mit den Umlenkrollen und einen Limit-Schalter enthalten. Die Eckteile fassen je zwei Linearstangen, die die Y-Achse bilden, auf denen wiederum die X-Achse läuft. Die Zahnriemen der Y-Achsen werden über die Umlenkrollen zum Schlitten geführt, wo sie mit den Aluminiumklemmen festgemacht werden können. Selbstverständlich sollten die Zahnriemen dabei etwas unter Spannung gebracht werden, damit später keine Schrittverluste auftreten.

Beim Zusammenbau der X-Achse ist es besonders wichtig, sich beim Einsetzen des

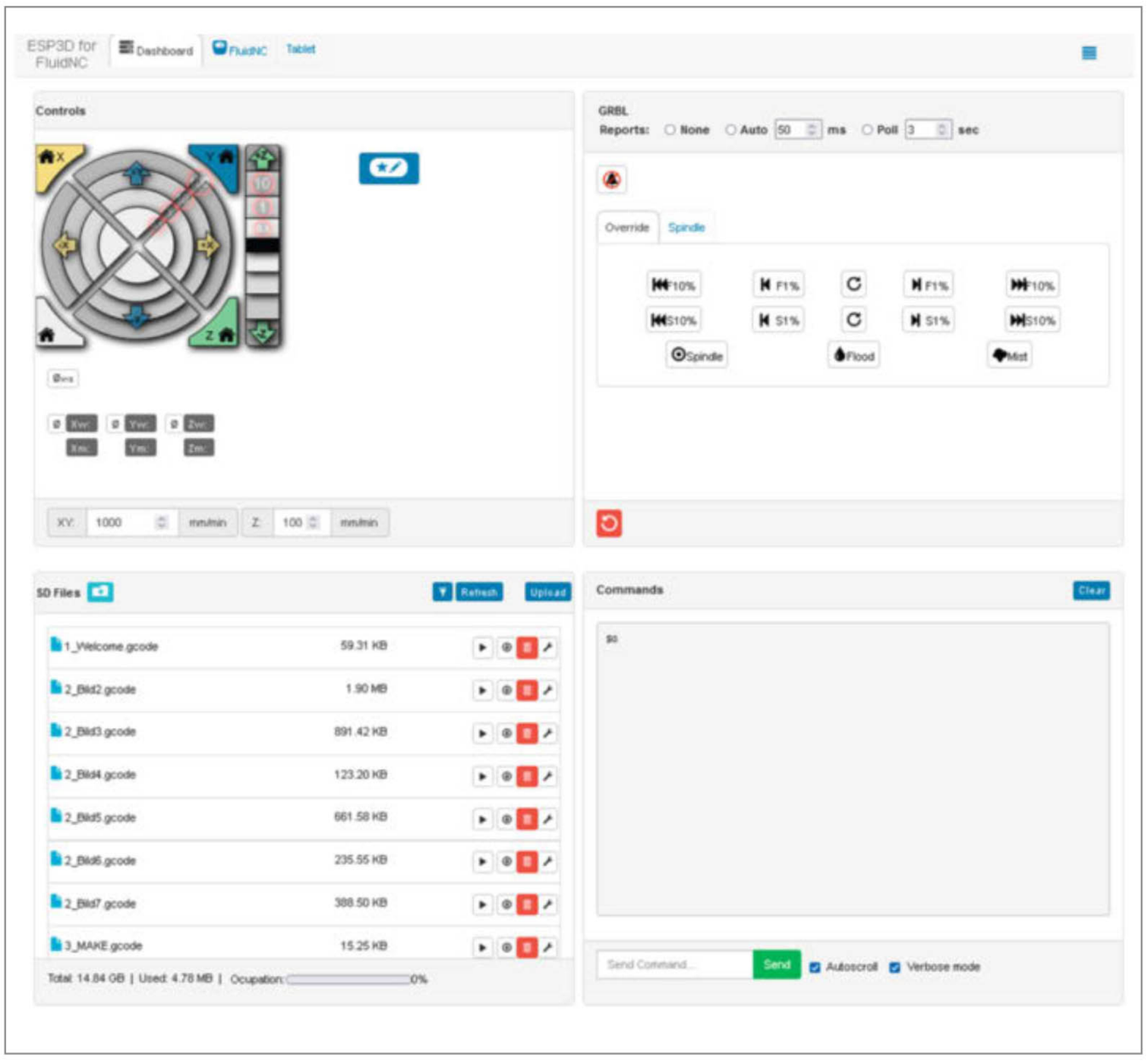


Abbildung 12: Die Benutzeroberfläche der Firmware FluidNC.

X-Schlittens und der Montage der gesamten X-Achse an die Anweisungen in der Bauanleitung zu halten. Wird der Schlitten oder die gesamte Achse nämlich falsch ausgerichtet montiert, hat das zur Folge, dass sich die Bewegungsrichtung des X-Schlittens invertiert. Im Prinzip ist dies kein großes Problem, ließe sich doch die Drehrichtung der Motoren in der Konfigurationsdatei der Firmware wieder korrigieren. Da ich aber allen Makern das Herumwerkeln an der Konfigurationsdatei ersparen möchte – man kann hier viel kaputt machen – ist es eine Notwendigkeit, sich genau an den konzipierten Aufbau zu halten.

Für das Fixieren des X-Zahnriemens mittels Aluminiumklemmen hat der Bauraum leider nicht mehr ausgereicht. Deswegen wird der X-Zahnriemen einfach um eine Schraube gewickelt, die Schraube eingeschraubt und der Zahnriemen mithilfe einer Nut zusammengequetscht. Um die notwendige Spannung des Zahnriemens hinzubekommen, kann der X-Motor leicht gelöst, nach außen versetzt und wieder fixiert werden. Als krönender Abschluss darf nun noch das Herzstück, ein Neodym-Scheibenmagnet mit Loch, auf dem X-Schlitten platziert und festgeschraubt werden.

Ist der technische Zwischenbau vollständig montiert und sind alle Kabel korrekt verbunden, muss man nur noch die Tischplatte obenauf setzen, diese festschrauben und die Seiten mit den vorbereiteten MDF-Wänden abdecken. Im Prinzip steht damit unser Tisch. Es fehlen noch Sand, eine Kugel und die Glasplatte als Abdeckung.

Für diese Abdeckung aus Glas verwende ich 6 mm starkes Einscheiben-Sicherheitsglas (ESG). Dieses wird vom Baumarkt oder einem Glashändler maßgenau zugeschnitten. Bei der Bestellung sollte man die Kanten säumen lassen (grob geschliffen, um Verletzungsgefahr zu reduzieren). Da die Kante verdeckt ist, reicht das. Von rodierten oder polierten Kanten rate ich ab. Dies würde die Kanten zu stark brechen, wodurch der Spalt zwischen Glas und Tisch deutlich größer ausfallen würde.

Da Glas grundsätzlich eher teuer ist und auch die Lieferzeiten meistens lang sind, lohnt es sich, als Test zuerst eine Attrappe aus günstigem MDF-Holz gleicher Größe zu bestellen. Damit lässt sich prüfen, ob sich die Platte schön in den Tischausschnitt einfügen wird. Ist die Glasplatte einmal eingesetzt, bekommt man sie übrigens händisch kaum noch raus. Am besten fügt man bei der Materialbestellung auch gleich einen Saugnapf hinzu, mit dem sich das Glas jederzeit aus der Versenkung heben lässt. Bevor die richtige Glasplatte eingesetzt wird, gilt es aber natürlich noch, den Tisch mit feinem Sand zu befüllen und eine ferromagnetische Kugel von etwa 20 mm Durchmesser in den Sand zu setzen. Dann kann endlich der Spaß losgehen.

Das erste Bild entsteht

Sobald der Controller mit Energie versorgt wird, meldet er sich als Access Point „FluidNC“. Nachdem wir uns mit diesem WLAN verbunden haben, öffnen wir mit einem Webbrowser die Adresse http://fluidnc.local. Daraufhin erscheint die grafische Bedienoberfläche der Firmware (Abbildung 12).

Als Erstes laden wir danach ein paar vorbereitete Sandbilder auf den Controller. Hierzu sind die zur Verfügung gestellten G-Code-Beispieldateien mittels der Schaltfläche Upload auf die SD-Speicherkarte des Controllers zu laden. Dabei sollte man schön langsam vorgehen und dem Controller zwischen den Dateien Zeit lassen. Ich habe festgestellt, dass sich der Controller aufhängt, wenn man eine Datei hochlädt, bevor der letzte Hochladeprozess vollständig beendet wurde.

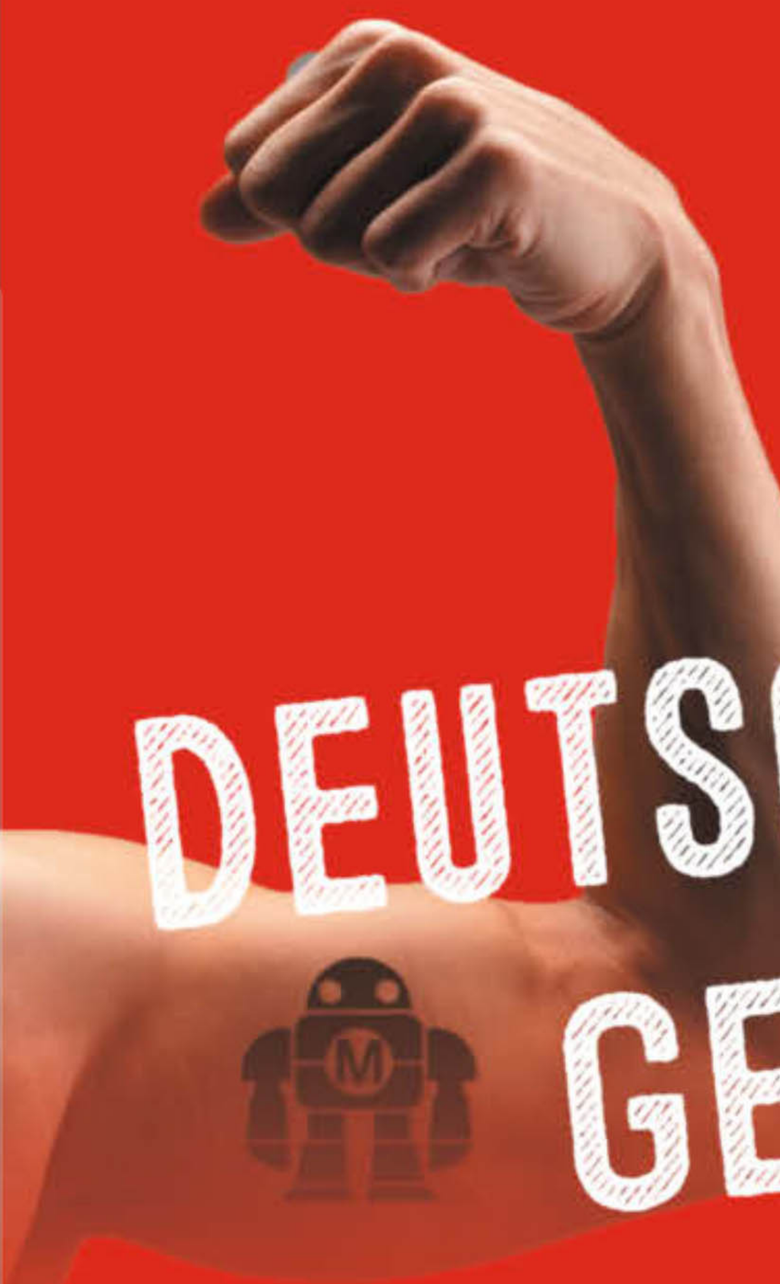
Anschließend kann die G-Code-Datei durch Klick auf das Play-Symbol abgespielt werden. Der Sandmaltisch führt nun zuerst ein Homing aus. Das heißt, der Schlitten mit dem Magnet geht auf die Suche nach den eingebauten Endschaltern. Sobald diese einen Kontakt melden, haben sie ihre Nullposition erreicht.

Von nun an kann der Sandmaltisch die in den G-Code-Dateien angegebenen X/Y-Koordinaten abfahren, wobei die im Sand liegende Kugel eine entsprechende Spur hinterlässt. Spur an Spur führt dies am Ende zu einem faszinierenden Sandbild. Ich persönlich finde dabei nicht nur das fertige Bild beeindruckend, ebenso zieht mich der Entstehungsprozess des Bildes in den Bann. Der Anblick einer langsam dahinrollenden Kugel wirkt entspannend und inspirierend zugleich.

Eigene Sandbilder

Sind die ersten Sandbilder einmal erzeugt, kommt schnell der Wunsch nach weiteren und eigenen Sandbildern auf. In einer nächsten Make berichte ich deshalb davon, wie man solche Sandbilder selbst mithilfe von entsprechenden Softwaretools erzeugen kann. Eine kurze Einführung in den zugrundeliegenden G-Code wird euch dabei helfen, das technische Prinzip der Fahrwege des Magneten besser zu verstehen und rudimentäre Änderungen daran vorzunehmen. Schließlich, und das ist das Sahnehäubchen dieses Projekts, zeige ich euch auf, wie sich einzelne Sandbilder miteinander verketten und automatisch abspielen lassen. Damit werdet ihr und die Mitbewohner eurer guten Stube täglich mit einem neuen Sandbild überrascht. In der Zwischenzeit dürft ihr schon mal mit dem Bau dieses vielseitigen Projekts beginnen. *—caw*

Make:



DEUTSCHLANDS GEFÄHRLICHSTES ABO-ANGEBOT*

*VON DEUTSCHLANDS GEFÄHRLICHSTEM DIY-MAGAZIN (LAUT LESERN)

2× Make testen mit über 30 % Rabatt

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

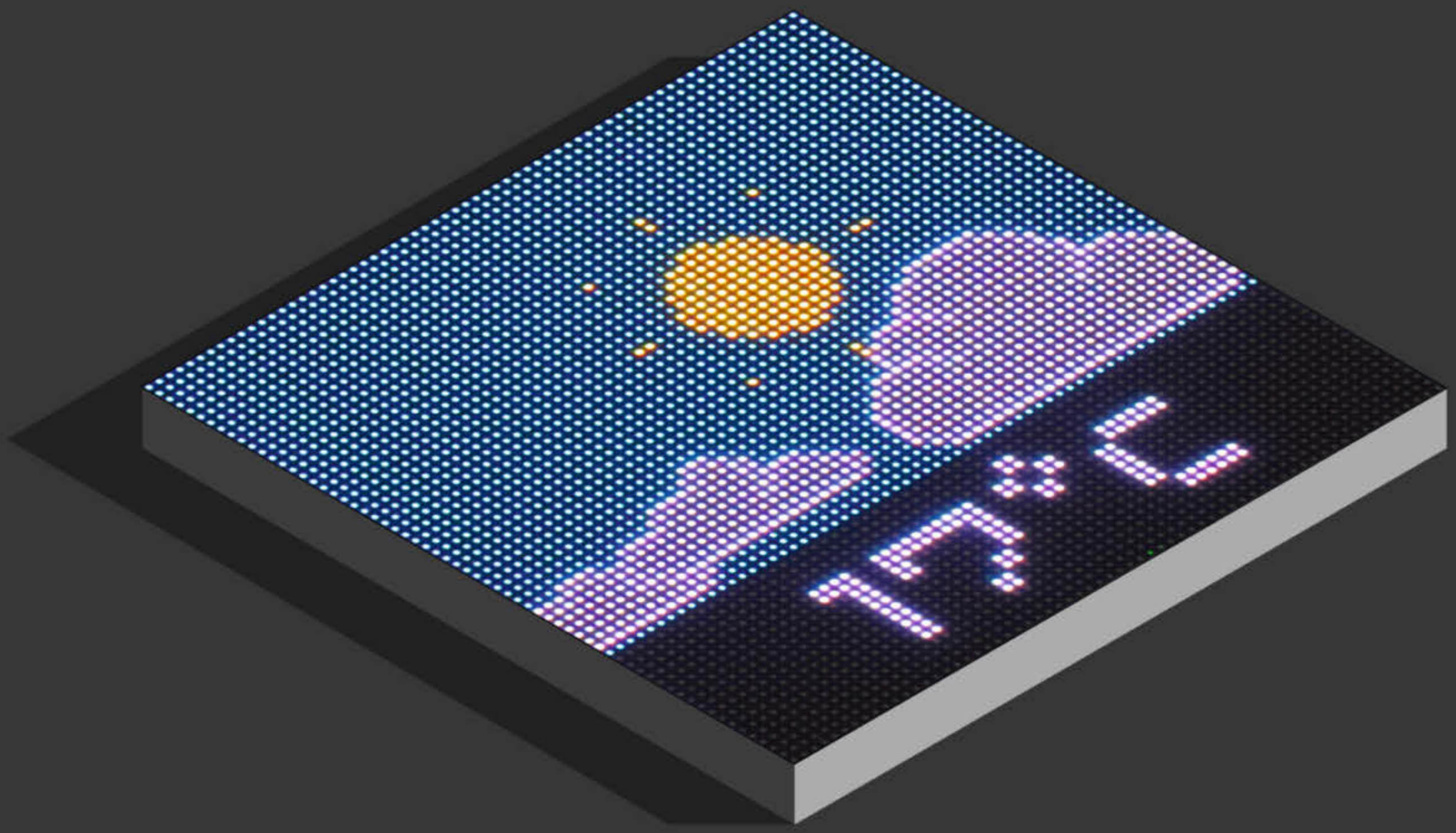
Warum eigentlich gefährlich?
Laut Lesern sind wir das "gefährlichste DIY-Magazin" Deutschlands. Das ist aber natürlich nur Spaß! Sie können unser Magazin ganz unbesorgt lesen und 2 Ausgaben als Heft + digital testen, zusätzlich erhalten Sie ein Geschenk Ihrer Wahl – klingt doch eigentlich ganz ungefährlich.

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

LED-Matrizes mit MicroPython steuern, Teil 2

Ein Infotext oder eine Laufschrift auf einer LED-Matrix sind nicht nur praktisch, sondern machen auch optisch was her. Mit ein paar animierten Grafiken lässt sich das Ganze aber auch noch ordentlich aufpeppen.

von Ákos Fodor



Im letzten Artikel zum Pimoroni Interstate 75 habe ich gezeigt, wie man HUB75-Matrizes mit MicroPython ansteuert und einfache geometrische Formen sowie eine Laufschrift erstellt. Mit dem kleinen Mikrocontroller-Board sind aber auch grafisch aufwendigere Projekte möglich. Ob für Spiele oder praktische Anwendungen wie eine dynamische Wetteranzeige – in diesem Artikel erkläre ich zwei Methoden, mit denen man Bilder auf der LED-Matrix darstellen kann. Begleitend dazu gibt es alle Programmcodes und Bilder im GitHub-Repository des Projekts zum Download, das ihr über den Link in der Kurzinfo erreicht.

Kurzinfo

- **PNG-Grafiken mit dem Interstate 75 auf einer HUB75-Matrix ausgeben**
- **Bilder in einfache Arrays umwandeln, einfärben und animieren**
- **Helligkeit der Matrix über RGB-Werte mit einem Sensor steuern**

Checkliste

Zeitaufwand:
ab 2 Stunden

Kosten:
5 bis 10 Euro (zzgl. Kosten aus dem ersten Artikel)

Material

- **HUB75-Matrix, Interstate 75 und Netzteil** aus dem ersten Artikel
- **LDR-Sensor**
- **Widerstand,** 10 kΩ
- **Kabel**

Werkzeug

- **Lötkolben und -zinn**
- **Schraubendreher**
- **Seitenschneider**

Software

- **Bildbearbeitungsprogramm**
- **Thonny**

Mehr zum Thema

- Ákos Fodor, LED-Matrizes mit MicroPython steuern, Make 3/24, S. 106
- Ákos Fodor, Adafruit GFX Library, Teil 2: Bitmaps erstellen und animieren, Make 3/23, S. 94
- Daniel Bachfeld, Pixelart mit Pi, Make 5/21, S. 56
- Heinz Behling, Wetter- und Windmühle, Make 5/22, S. 64

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

Einstieg mit PNGs

Eine einsteigerfreundliche und schnelle Methode, um Bilder auf der LED-Matrix auszugeben (Bild 1), bietet die Bibliothek PNGdec, die in die Interstate75-Bibliothek integriert ist. Mit ihr lassen sich PNGs komplett oder ausschnittsweise darstellen. Man kann also eine Datei pro Bild verwenden oder mehrere Motive in einer Datei speichern und immer nur einen Teilbereich davon anzeigen. Hat das PNG-Bild transparente Bereiche, stellt die Bibliothek diese zudem auch auf der LED-Matrix transparent dar, sodass die dahinter liegende Grafik sichtbar ist und nicht etwa mit Schwarz überdeckt wird.

Um eine PNG-Datei (z. B. car.png aus dem GitHub-Repository des Projekts) auf dem Panel auszugeben, kopiert man sich zunächst die Grafik auf das Mikrocontroller-Board. Dazu verbindet man den Interstate 75 mit dem PC, öffnet Thonny und drückt einmal auf die Stopp-Taste in der Symbolleiste, um mögliche laufende Prozesse zu unterbrechen. Danach sucht man links oben im Thonny-Dateifenster auf seinem PC nach dem PNG-Bild, klickt mit der rechten Maustaste darauf und wählt „Upload to /“ aus. Dieser Befehl kopiert das Bild in das Hauptverzeichnis des Interstate 75. Sollte das Dateifenster nicht sichtbar sein, lässt es sich über das Menü „View/Files“ einblenden.

Als Nächstes erstellt man sich ein neues Dokument in Thonny und importiert die Interstate75-Bibliothek und die notwendigen Befehle, mit denen man die LED-Matrix initialisiert (siehe Listing „png.py“). Die PNGdec-Bibliothek bindet man mit folgendem Befehl in sein Programm ein:

```
from pngdec import PNG
```

Danach lässt sich die PNG-Datei mit nur vier Zeilen Code öffnen, dekodieren und auf der LED-Matrix ausgeben:

```
png = PNG(buffer)
png.open_file("car.png")
png.decode(0, 0)
matrix.update(buffer)
```

Die beiden Nullen im `decode()`-Befehl stehen für die X- und Y-Position der linken oberen Ecke des Bildes. Überdies gibt es für `decode()` noch weitere Parameter:

- Möchte man sein Bild skalieren, kann man in der Klammer den Befehl `scale = (x, y)` ergänzen und gibt für die X- und Y-Skalierung jeweils einen positiven Integer ein.
- Mit `rotate = n` lässt sich ein Bild auch in 90-Grad-Schritten drehen.
- Und wenn man, wie eingangs beschrieben, nur einen Teilbereich der PNG zeigen will, kann man diesen mit `source = (x, y, b, h)` festlegen, wobei `b` und `h` die Breite und Höhe bestimmen.

Wenn man mehrere dieser Parameter verwendet, muss man allerdings die Reihenfolge (source, scale, rotate) beachten, mit denen die

Bild 1: Mit PNGdec kann man auch kleine Fotos auf einer LED-Matrix anzeigen.

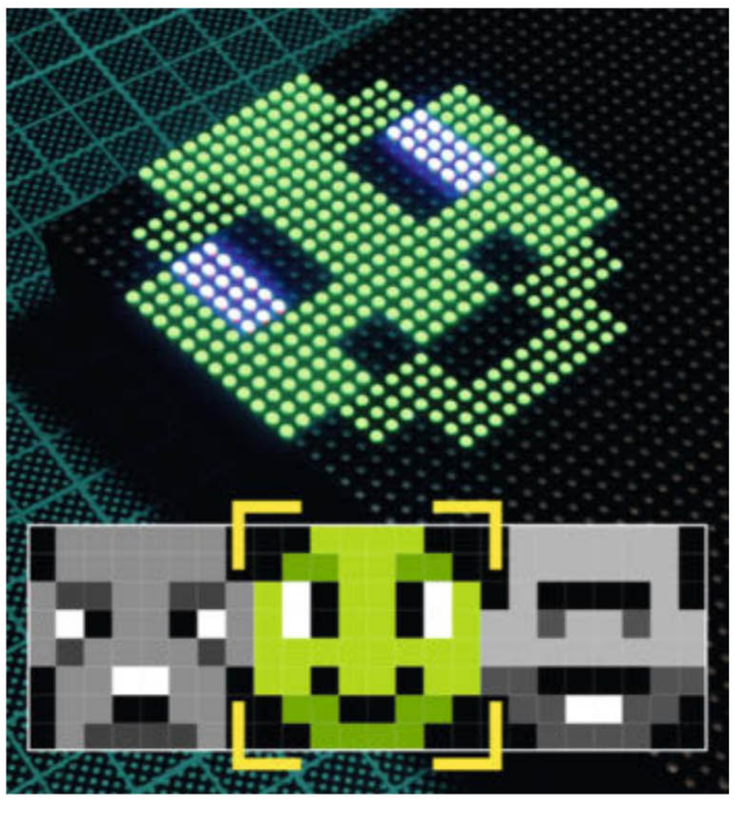


Bild 2: Mit dem Parameter source kann man auch nur einen Bereich einer PNG ausgeben.

png.py

```
from pngdec import PNG
from interstate75 import (
     Interstate75,
     DISPLAY_INTERSTATE75_64X64)

matrix = Interstate75(display =
 DISPLAY INTERSTATE75_64X64)

buffer = matrix.display
width = matrix.width
height = matrix.height

png = PNG(buffer)
png.open_file("car.png")
png.decode(0, 0)
matrix.update(buffer)
```

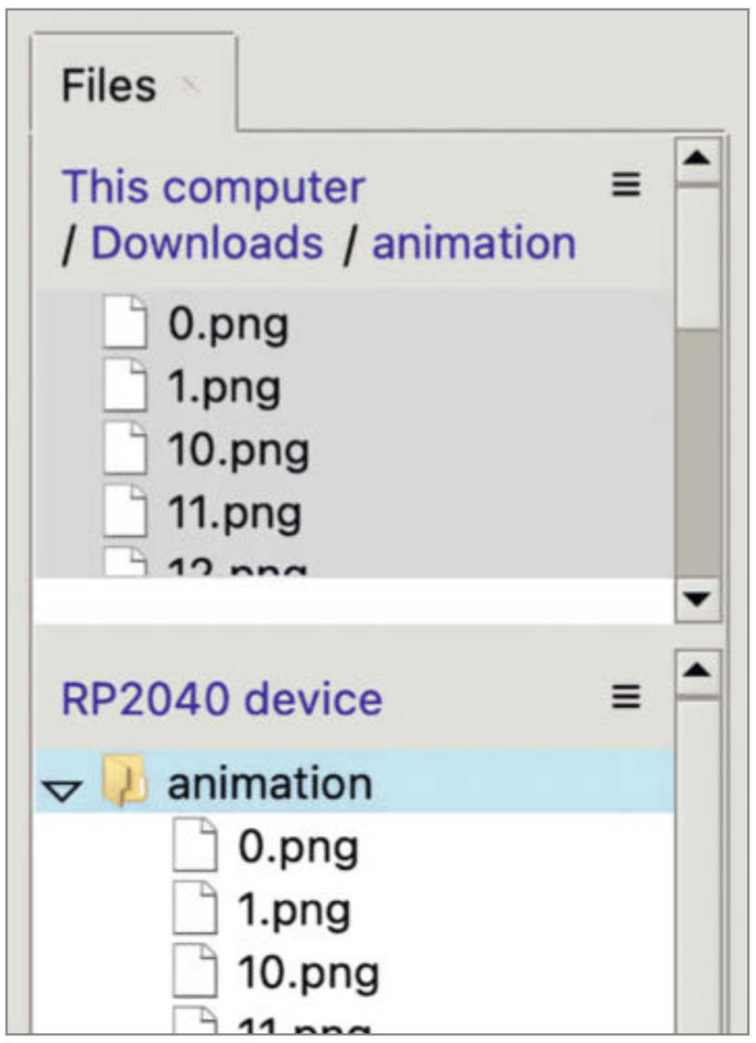


Bild 3: In Thonny kann man über das Dateifenster Bilder auf den Interstate 75 kopieren.

png_animation.py

```
from pngdec import PNG
from interstate75 import (
     Interstate75,
     DISPLAY_INTERSTATE75_64X64)

matrix = Interstate75(display =
         DISPLAY_INTERSTATE75_64X64)

buffer = matrix.display
width = matrix.width
height = matrix.height

png = PNG(buffer)

for image in range(36):
    png.open_file("/animation/"
                  + str(image)
                  + ".png")
    png.decode(0, 0)
    image += 1
    matrix.update(buffer)
```

Bibliothek diese abarbeitet. Das kann dann etwa so aussehen:

```
png.decode(0,0,source = (8,0,8,8),
           scale = (3, 3))
```

Wie Bild 2 zeigt, lässt sich mit diesem Befehl ein 8 × 8 Pixel großer Ausschnitt aus einer längeren PNG-Datei (faces.png) auswählen und links oben auf der LED-Matrix mit `scale` in dreifacher Größe ausgeben.

PNGs animieren

Um etwas mehr Leben auf die LED-Matrix zu bringen, kann man PNGs auch animieren. Dafür bietet sich entweder der `source`-Parameter im `decode()`-Befehl an, mit dem man für jeden Animationsschritt jeweils den Ausschnitt versetzt, oder man benutzt einzelne Bilder, die man in einer `for`-Schleife nacheinander lädt und anzeigt.

Für die zweite Variante benennt man die einzelnen PNGs, die man für die Animation benötigt, am besten von `0` bis `n` (0.png, 1.png usw.) und erstellt auf dem Interstate 75 in Thonny einen Ordner namens „animation“. Dafür klickt man links unten in Thonny im Dateifenster, das den Inhalt des Mikrocontrollers zeigt, mit der rechten Maustaste und wählt im Pop-up-Menü „New directory...“ aus. Danach öffnet man den Ordner mit einem Doppelklick und kopiert alle PNG-Dateien mithilfe des Befehls „Upload to /animation“ vom PC in den Ordner „animation“ (Bild 3).

Die einzelnen Bilder lassen sich dann, wie im Listing „png_animation.py“, in einer `for`-Schleife nacheinander aufrufen, indem man sich im Befehl `open_file()` den benötigten Dateinamen aus Strings zusammensetzt. Der Wert `36` kennzeichnet im Listing das letzte Bild der Animation (36.png).

Arrays als Alternative

Möchte man, statt einfach nur PNGs abzubilden, Grafiken nutzen, die sich samt ihrer Farbe und Helligkeit im laufenden Betrieb per Code anpassen lassen, bietet sich eine alternative Methode zur eben beschriebenen an: Man übersetzt die Bilddateien in einfache Arrays. Diese enthalten dann für jedes Pixel eine Farbnummer – etwa 1 für Orange und 2 für Blau (Bild 4). Diese verknüpft man dann mit Farbwerten, die sich als Variablen im Programm jederzeit ändern lassen. Auf diese Art kann man Symbole oder Sprites schnell in unterschiedlichen Farben darstellen.

Um dieses Prinzip an einem praktischen Beispiel zu erklären, habe ich mir eine Wetteranzeige überlegt, die für leicht bewölktes Wetter passende Bilder sowie die aktuelle Temperatur (derzeit per Zufall generiert) darunter anzeigen soll. Nachstehend erkläre ich, wie man die benötigten Grafiken in Arrays konvertiert und das Programm schrittweise aufbaut. Dabei gehe ich mit ein paar Listings auf die wichtigsten Details ein. Aufgrund der zunehmenden Komplexität lässt sich aber nicht jedes Mal der vollständige Code abdrucken.

Daher empfehle ich, die Python-Programme und die dazugehörigen Grafiken aus dem GitHub-Repository herunterzuladen und parallel in Thonny zu öffnen, um die Zusammenhänge besser zu verstehen. Falls ihr selbst Grafiken erstellen wollt, reicht das Paint-Programm in Windows dafür vollkommen aus.

Bilder in Arrays umwandeln

Auch wenn sich Arrays für Bilder mit wenigen Pixeln noch recht mühelos von Hand im Code anfertigen lassen, ist es auf Dauer komfortabler, sie in einem Grafikprogramm zu erstellen und mit einem Werkzeug in Arrays umzuwandeln. Daher habe ich für die Wetteranzeige zunächst eine Wolke mit einer Sonne in vier Graustufen (mit 100, 65, 45 und 0 Prozent Helligkeit) gemalt und als JPG-Datei exportiert. Das Bild soll nachher mit Gelb, zwei Grauwerten und Schwarz ausgegeben werden (Bild 5). Wieso ich dafür diese spezifischen Helligkeitswerte gewählt habe, hängt mit der Konvertierungsmethode zusammen, die ich im Folgenden erkläre.

Um das Bild in ein geeignetes Array umzuwandeln, nutze ich die Python-Bibliothek Pillow und habe ein kleines Programm erstellt (image_to_array.py), das diese Aufgabe übernimmt. Pillow kann Bilder in den gängigsten Dateiformaten wie PNG, JPG oder BMP öffnen und verarbeiten. Damit man die Bibliothek auch direkt in Thonny nutzen kann, muss man zunächst die Entwicklungsumgebung (rechts unten im Programmfenster) bei der Board-Auswahl auf „Local Python 3“ umstellen (Bild 7). Danach klickt man in der Menüleiste auf „Tools/Manage Packages“, sucht im Paket-Manager nach „pillow“ und installiert die Bibliothek. Anschließend öffnet man die Datei image_to_array.py.

In den ersten paar Zeilen des Programms muss man ein paar Informationen zu dem Bild angeben, das man konvertieren möchte, wie die Größe und den Pfad (siehe Listing „image_to_array.py“). Wenn man die Grafik in denselben Ordner legt, in der sich das Programm befindet, trägt man lediglich den Dateinamen bei `image_path` ein.

Sobald man den Code ausführt, wandelt Pillow das Bild mit dem Befehl `convert('L')` in Graustufen um. Mithilfe einer Formel (siehe Kasten „Farben und Graustufen“) werden dabei die möglichen 16,78 Millionen RGB-Farben auf maximal 256 Helligkeitswerte reduziert. Danach segmentiert das Programm diese Werte zeilenweise in Schwarz, Dunkelgrau, Hellgrau und Weiß und weist den Pixeln die

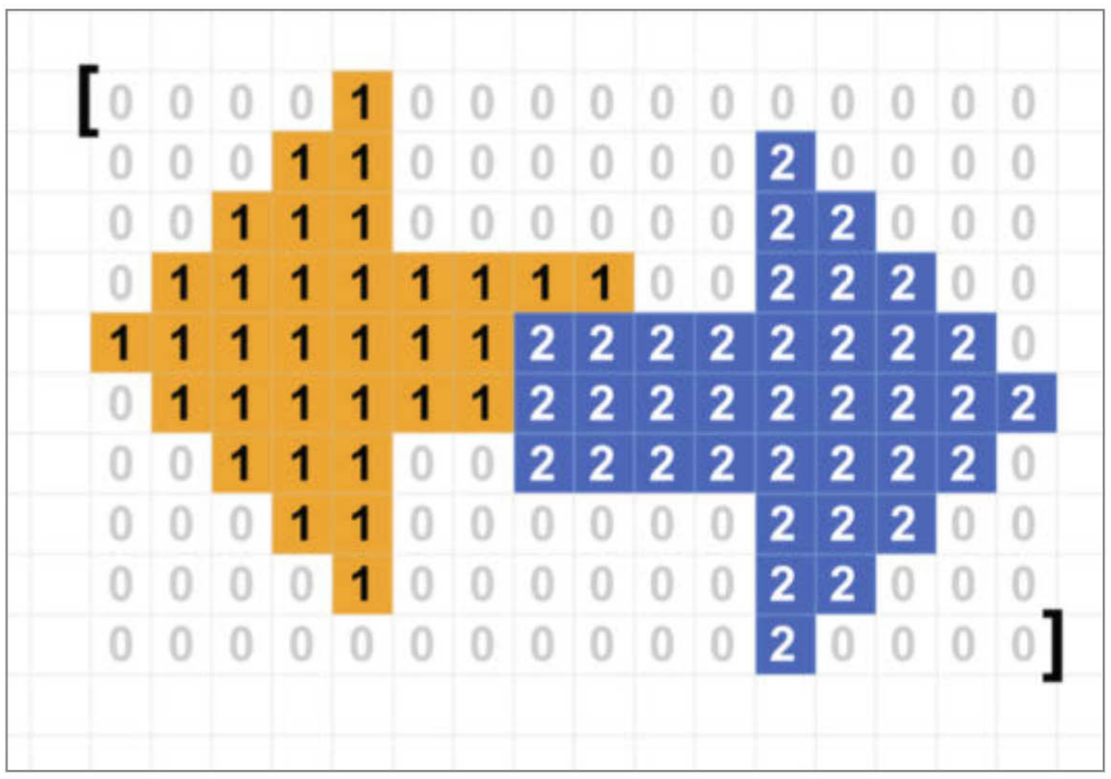


Bild 4: Dieses Bild lässt sich mit wenigen Farben gut in vereinfachte Arrays übersetzen.

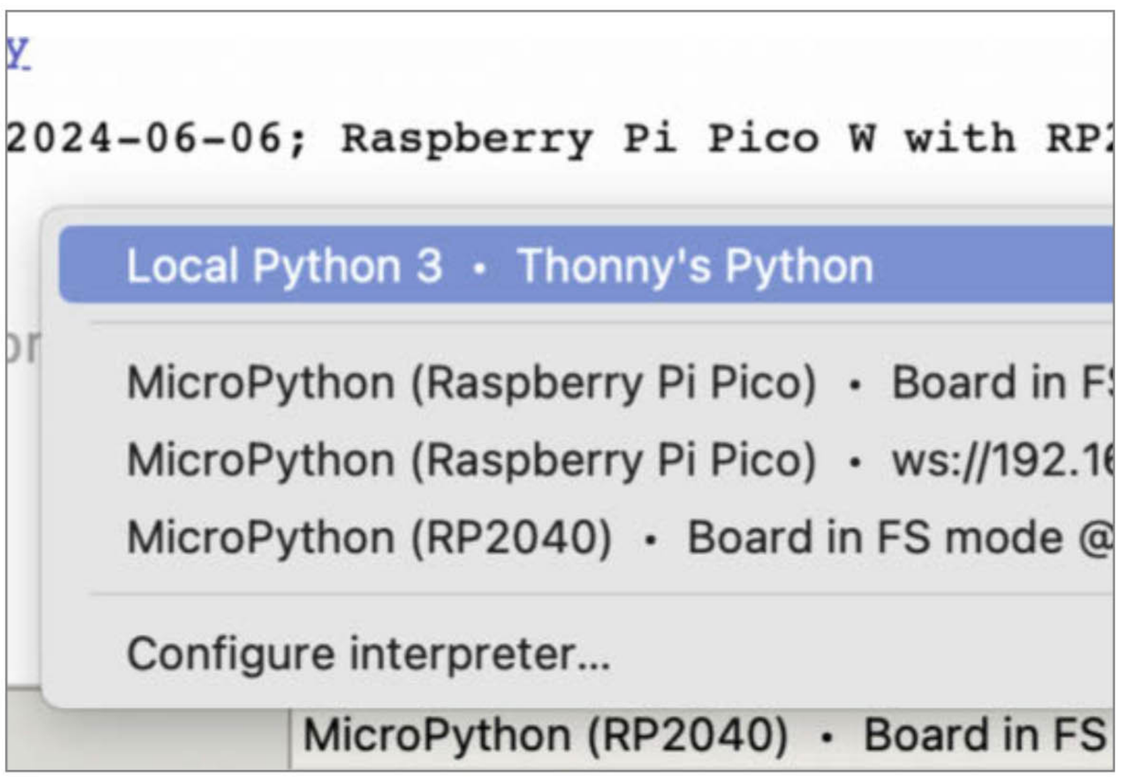


Bild 7: Rechts unten kann man in Thonny zu Python wechseln.

Bild 5: Sonne, Wolken und Hintergrund bestehen insgesamt aus vier Graustufen.

entsprechenden Farbnummern `0`, `1`, `2` und `3` zu. Anschließend gibt ein `print()`-Befehl das Ergebnis im Shell unterhalb des Code-Fenstersals Array aus.

Dass das erzeugte Array wie in dem Beispiel in Bild 4 dabei zeilenweise umbricht, ist zwar nicht notwendig, um die Grafik auf der LED-Matrix abzubilden, aber so kann man später im Code noch einigermaßen erkennen, was das Array darstellen soll, und schnell kleine Änderungen vornehmen, ohne noch einmal ins Grafikprogramm wechseln zu müssen. Einzig die eckigen Klammern muss man noch ergänzen, sowie das letzte Komma löschen , sobald man das Array in sein Programm einfügt. Am besten kopiert man sich dafür zunächst das Array aus dem Shell in eine Textdatei als Zwischenablage.

Array auf Matrix anzeigen

Um die Grafik auf der LED-Matrix auszugeben, habe ich das Wetterprogramm wetter_01.py erstellt und das eben erzeugte Array dort als icon_array eingefügt (Bild 8). Wie in dem Listing „wetter_01.py“ zu sehen, befinden sich

Farben und Graustufen

Wenn man die RGB-Farbwerte eines Bildes auf 256 Graustufen reduziert, kann es passieren, dass selbst eindeutig voneinander unterscheidbare Farben denselben oder zumindest einen sehr ähnlichen Grauwert ergeben. Wie nah sie beieinander liegen, hängt davon ab, wie die Werte der einzelnen RGB-Kanäle miteinander verrechnet werden. Während mein Bildbearbeitungsprogramm, wie in Bild 6 zu sehen, einen Mittelwert aus Rot, Grün und Blau erzeugt hat und alle Farben denselben Grauwert erhalten, nutzt die Bibliothek Pillow zum Konvertieren in Graustufen eine Formel, die die Farben gewichtet:

```
R*299/1000 + G*587/1000 + B*114/1000
```

Daraus ergeben sich für das gelbe Feld links oben mit `RGB(190, 162, 0)` der Grauwert `133` und für das blaue mit `RGB(0, 181, 190)` der Wert `144` – das orangefarbene Feld erhält aber mit `RGB(190, 95, 0)` aufgrund der Gewichtung als Ergebnis `113`. Wenn man ein Bild nun in vier Graustufen zerteilt wie in image_to_array.py und die Grenzen bei `64`, `128`, `192` und `255` zieht, erhalten das gelbe und das blaue Feld den Wert `2`, weil ihre Helligkeit zwischen `128` und `192` liegt. Das orangefarbene Feld fällt mit `113` aber unter die `128` und wird daher mit einer `1` gekennzeichnet.

Die Gewichtung in Pillow, aber auch schon die Reduktion von RGB auf Graustufen erschwert es, im Grafikprogramm abzuschätzen,

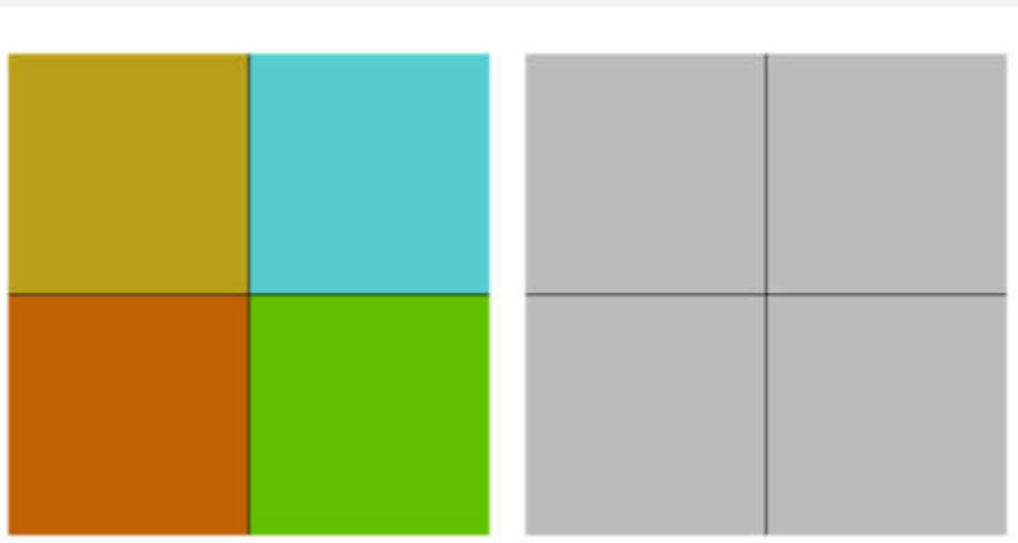


Bild 6: Farben können beim Konvertieren in Graustufen identische Werte erhalten.

welchen Grauwert eine Farbe später erhalten wird. Daher ist es sinnvoll, die Grafiken gleich in Graustufen zu erstellen und Werte zu verwenden, die – für das hier gezeigte Beispiel – eindeutig unter 25, 50, 75 oder 100 Prozent liegen, was im Code den Werten `64`, `128`, `192` und `255` entspricht.

image_to_array.py

```python
from PIL import Image

width = 36
height = 36
image_path = 'bild.jpg'
image = Image.open(image_path)

image = image.convert('L')
array = []

for y in range(height):
  row = []
  for x in range(width):
    pixel = image.getpixel((x,y))
    if pixel < 64:
      value = 0
    elif pixel < 128:
      value = 1
    elif pixel < 192:
      value = 2
    else:
      value = 3
    row.append(value)
  array.append(row)

for row in array:
  print(','.join(map(str, row))+',')
```

darunter ein paar Eckdaten zum Bild wie die Größe, die Position sowie die beiden Werte `icon_array_x` und `icon_array_y`, auf die ich gleich noch eingehe. Außerdem sind vier Farben für die Farbnummern definiert. Aus Platzgründen ist das Array hier gekürzt dargestellt.

Damit aus dem Array wieder eine Grafik wird, setzt eine `for`-Schleife die im Array enthaltenen Werte erneut zusammen und weist ihnen eine Farbe zu. Dadurch erhält z. B. die Sonne, die im Array aus lauter `2`-en besteht, mit `yellow` ein sattes Gelb. Eine weitere `if`-Bedingung gleicht danach die Bilddimensionen (`icon_width` und `icon_height`) mit dem aktuell gesetzten Pixel (`icon_array_x` und `icon_array_y`) ab, damit das Bild mit den korrekten Zeilenumbrüchen auf der LED-Matrix erscheint. Hat `icon_array_x` den Wert von `icon_width` überschritten, taucht das folgende Pixel in der nächsten Zeile auf und `icon_array_x` wird wieder auf `0` gesetzt. Das geht so lange, bis alle Array-Werte bzw. Pixel abgearbeitet sind.

Zum Schluss wechselt das Programm den Pinsel noch einmal auf `white`, setzt einen zufälligen Temperaturwert als Text mittig unter die Grafik und gibt alles, was bisher lediglich im Puffer stattgefunden hat, mit `matrix.update(buffer)` auf der LED-Matrix aus (Bild 9).

Um den Code auf der LED-Matrix auszuprobieren, verbindet man den Interstate 75 mit dem PC, wählt rechts unten im Thonny-Fenster mit „MicroPython (RP2040) - Board in FS mode" sein Board aus und führt das Programm aus.

Wolken, die fliegen

Als Nächstes lassen wir ein paar Wolken am Himmel entlangziehen. Dafür muss man die Grafik mit der Sonne und der Wolke zunächst in einzelne Bilder und Dateien zerteilen und den Teil der verdeckten Sonne voll ausmalen. Da ich mehr als nur eine vereinzelte Wolke am Himmel herumfliegen lassen wollte, habe ich zudem noch zwei etwas kleinere Varianten ergänzt (Bild 10) – ebenfalls in Graustufen.

Mit den hinzugekommenen Grafiken muss man auch ein paar mehr Arrays erstellen, also Thonny erst mal wieder auf Python umstellen, die Bilder konvertieren und die Arrays Stück für Stück in sein Wetterprogramm (wetter_02.py) übertragen. Nachdem das erledigt ist, gibt man für jedes einzelne Element noch die Größe und die Position an, wo die Bilder zu Beginn sitzen sollen.

Ein paar Dinge wie die Farbdefinitionen und die Textausgabe können bleiben wie bisher. Da nun aber mehrere Arrays gleichzeitig auf der Matrix umherschwirren, musste ich das Ganze etwas koordinieren und zeitlich abstimmen. Daher sind zwei Funktionen hinzugekommen. Die eine lautet

```
show_array(icon_array,
           icon_pos_x,
           icon_poy_y,
           icon_width,
           icon_height)
```

und arbeitet wie im vorherigen Beispiel ein Array ab, aber dieses Mal das Array, das man als Parameter `icon_array` in der Klammer samt Position und Größe angegeben hat. Zudem gibt es die Funktion `clouds_move()`, die steuert, wie schnell sich welche Wolke (von rechts nach links) bewegt und dafür sorgt, dass sie wieder am rechten Bildschirmrand auftaucht, sobald sie den sichtbaren Bereich verlassen hat.

Das Listing „wetter_02.py (Wolken)" zeigt exemplarisch die Funktion für eine Wolke. Das Timing der Bewegungen mit `time.ticks_ms()` gelöst, sodass sich die einzelnen Abläufe nicht in die Quere kommen. Die Dauerschleife `while True:` sorgt schließlich dafür, dass sich alles dauerhaft bewegt, in den Puffer gezeichnet und auf der LED-Matrix ausgegeben wird.

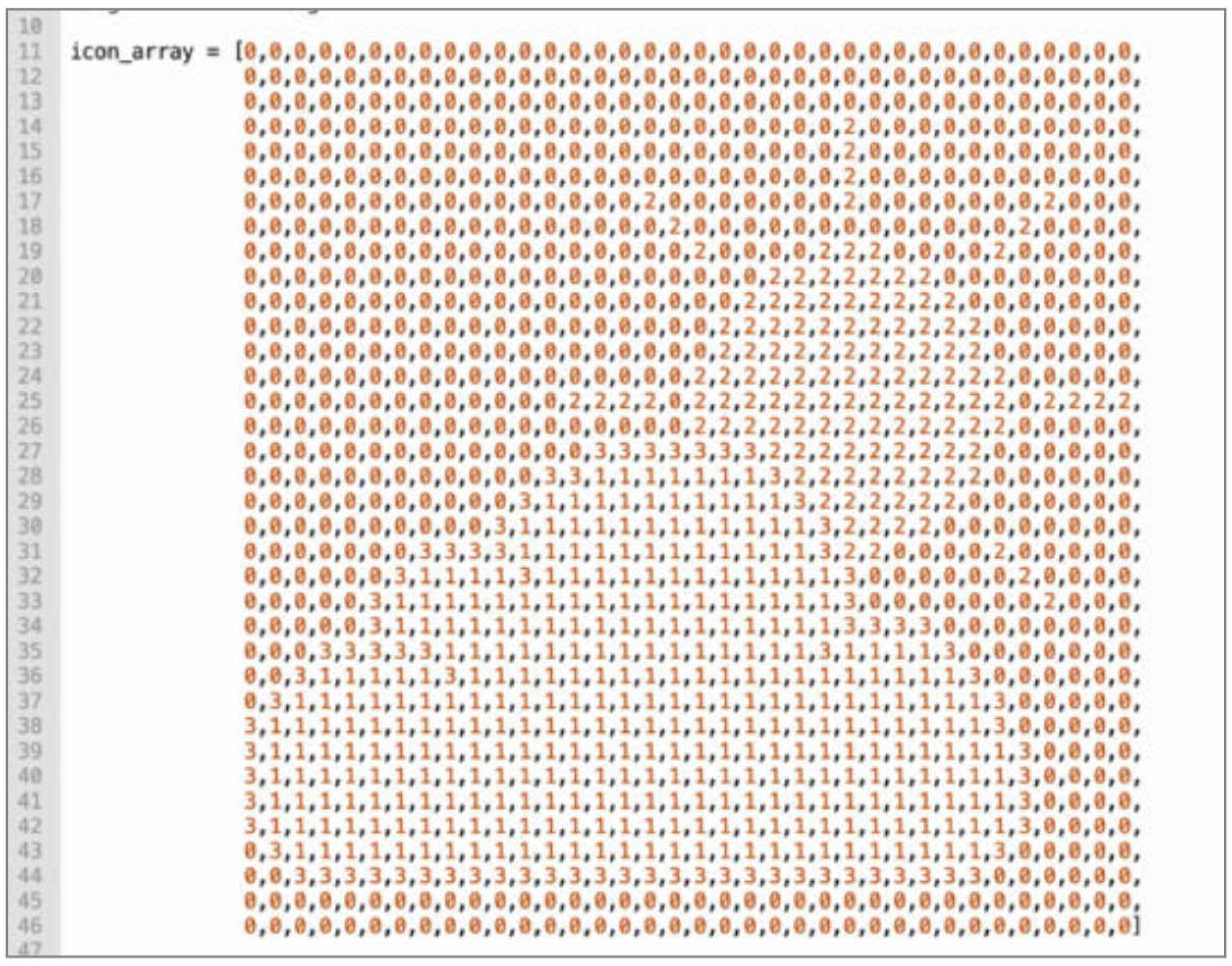


Bild 8: Das fertige Array in eckigen Klammern und ohne das Komma am Ende.

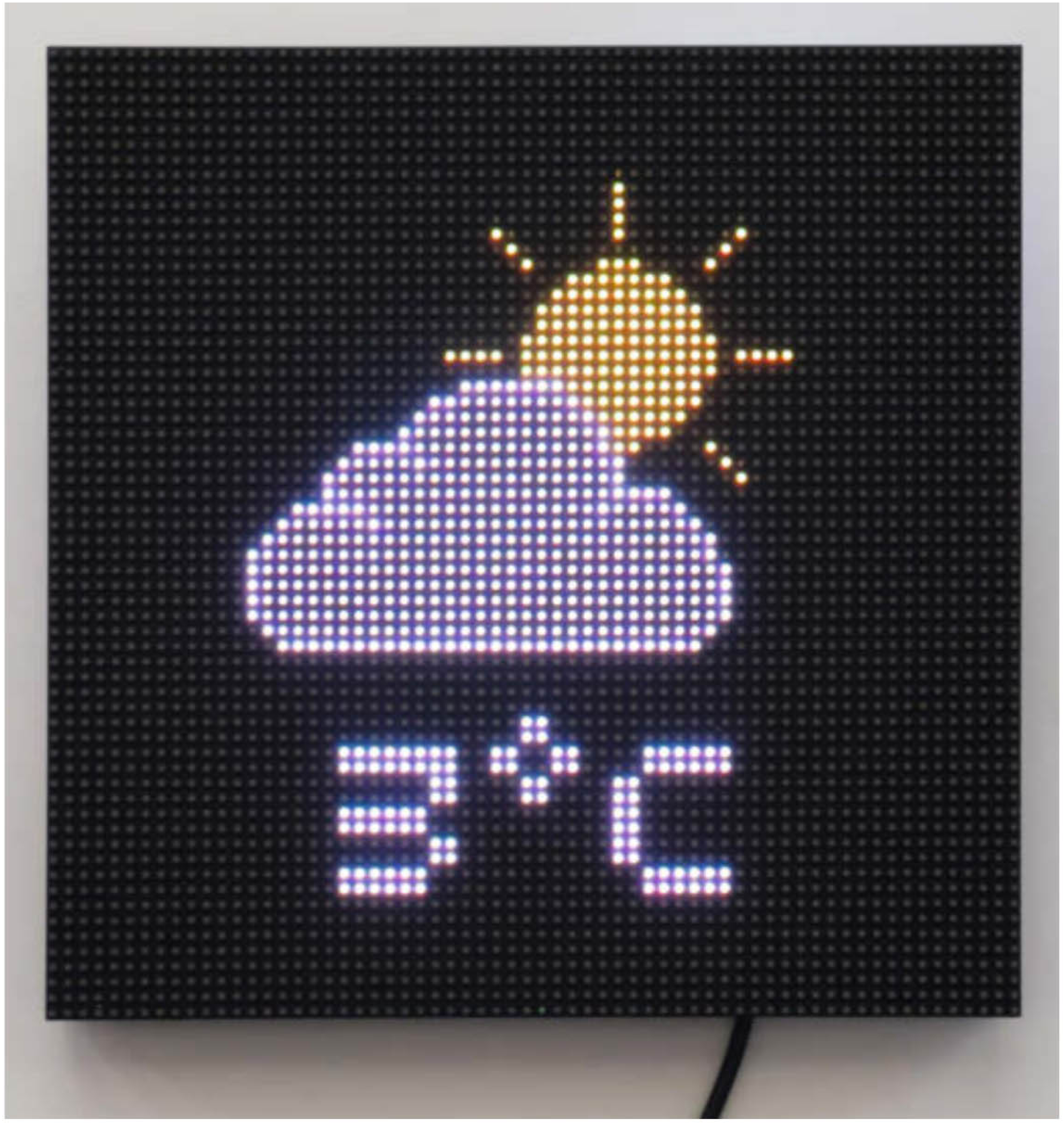


Bild 9: Mit den richtigen Farben sieht der erste Versuch schon ganz gut aus.

wetter_01.py (gekürzt)

```
import random
from interstate75 import (
     Interstate75,
     DISPLAY_INTERSTATE75_64X64)

matrix = Interstate75(display =
         DISPLAY_INTERSTATE75_64X64)
buffer = matrix.display
width = matrix.width
height = matrix.height

icon_array = [0,0,0 ... 0,0,0]

icon_width = 36
icon_height = 36
icon_array_x = 0
icon_array_y = 0
icon_pos_x = 13
icon_pos_y = 6

black=buffer.create_pen(0,0,0)
white=buffer.create_pen(200,200,200)
yellow=buffer.create_pen(201,168,0)
grey=buffer.create_pen(128,128,128)

temp = random.randrange(-20,40)
temp_width = buffer.measure_text(
                  str(temp)+"°C")
text_pos_x = int(width/2 -
                  (temp_width-2)/2)

for pixel in range(len(icon_array)):
    if icon_array[pixel] == 0:
        buffer.set_pen(black)
    elif icon_array[pixel] == 1:
        buffer.set_pen(grey)
    elif icon_array[pixel] == 2:
        buffer.set_pen(yellow)
    else:
        buffer.set_pen(white)

    buffer.pixel(icon_array_x +
                 icon_pos_x,
                 icon_array_y+
                 icon_pos_y)

    if icon_array_x < (icon_width-1):
        icon_array_x += 1
    else:
        icon_array_x = 0
        icon_array_y += 1
```

Bild 10: Hier zwar auf einem Bild, aber eigentlich ist jede Grafik eine eigene Datei.

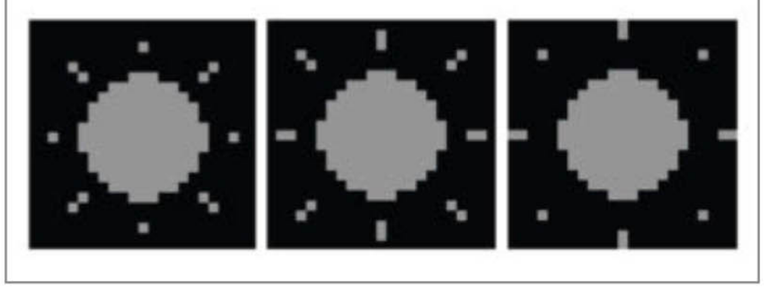


Bild 11: Eine Animation kann auch mit drei Bildern funktionieren.

wetter_02.py (Wolken)

```
def clouds_move():
  global cloud_old_ticks
  global cloud_pos_x
  global cloud_pos_y
  cloud_waiting_time = 200

  if ((time.ticks_ms()
      - cloud_old_ticks) >=
      cloud_waiting_time):

    cloud_old_ticks=time.ticks_ms()
    cloud_pos_x -= 1

  if cloud_pos_x < 0 - cloud_width:
    cloud_pos_x = 65
    cloud_pos_y = random.randint(25,
                                 29)
```

Da ich die Grafiken nur auf den oberen Bereich der LED-Matrix beschränken wollte und tief umherfliegende Wolken nicht aus Versehen in den Text ragen sollten, habe ich die Clipping-Funktion verwendet. Diese kann man sich wie eine rechteckige Schablone vorstellen, in der man malen kann, ohne die übrigen Bereiche auf der Matrix zu beeinflussen. Ist man damit fertig, nimmt man sie einfach wieder weg. So habe ich zuerst ein Clipping mit

```
buffer.set_clip(0, 0, 64, 46)
```

gesetzt (64 Pixel breit und 46 hoch), danach die hellblaue Farbe ausgewählt und den Himmel mit `clear()` gefüllt. Darauf folgen die Sonne und dann die Wolken, sodass diese immer vor der Sonne vorbeihuschen. Sobald das Clipping mit `remove_clip()` wieder entfernt ist, kann man den Text unter die Grafik setzen.

Animierte Sonnenstrahlen

Auch die Array-Grafiken lassen sich animieren, sodass man der Sonne bereits mit ein paar wenigen Einzelbildern (Frames) etwas Leben einhauchen kann. Diese muss man natürlich auch erst mal erstellen. Bild 11 zeigt die Bilder, mit denen ich die Sonnenstrahlen in wetter_03.py animiert habe.

Nachdem man die Grafiken für die Animation in Arrays umgewandelt und in seinen Code eingefügt hat, benennt man sie zur Übersicht am besten mit `sun_frame_0`, `sun_frame_1` usw. Danach erstellt man ein weiteres Array (z. B. `sun_frames`) und fügt die

wetter_03.py (Animation)

```
def sun_animation():
  global sun_frames
  global sun_cur
  global sun_old_ticks
  sun_waiting_time = 500

  if ((time.ticks_ms() -
       sun_old_ticks) >=
       sun_waiting_time):

    sun_old_ticks = time.ticks_ms()

    if sun_cur < len(sun_frames)-1:

      sun_cur += 1
    else:
      sun_cur = 0

...

while True:
  show_array(sun_frames[sun_cur],
             sun_pos_x, sun_pos_y,
             sun_width, sun_height)
```

wetter_04.py (Helligkeit)

```
def set_brightness():
  global br_old_ticks
  global white
  global yellow
  global grey
  global sky
  br_waiting_time = 2000

  if ((time.ticks_ms()
       -br_old_ticks) >=
        br_waiting_time):

    br_old_ticks = time.ticks_ms()
    read_ldr = ldr.read_u16()
    brightness = read_ldr*0.000015

    white = buffer.create_pen(
                int(200*brightness),
                int(200*brightness),
                int(200*brightness))
    yellow = buffer.create_pen(
                int(201*brightness),
                int(168*brightness),
                0)
    grey = buffer.create_pen(
                int(128*brightness),
                int(128*brightness),
                int(128*brightness))
    sky = buffer.create_pen(
                int(16*brightness),
                int(63*brightness),
                int(84*brightness))
```

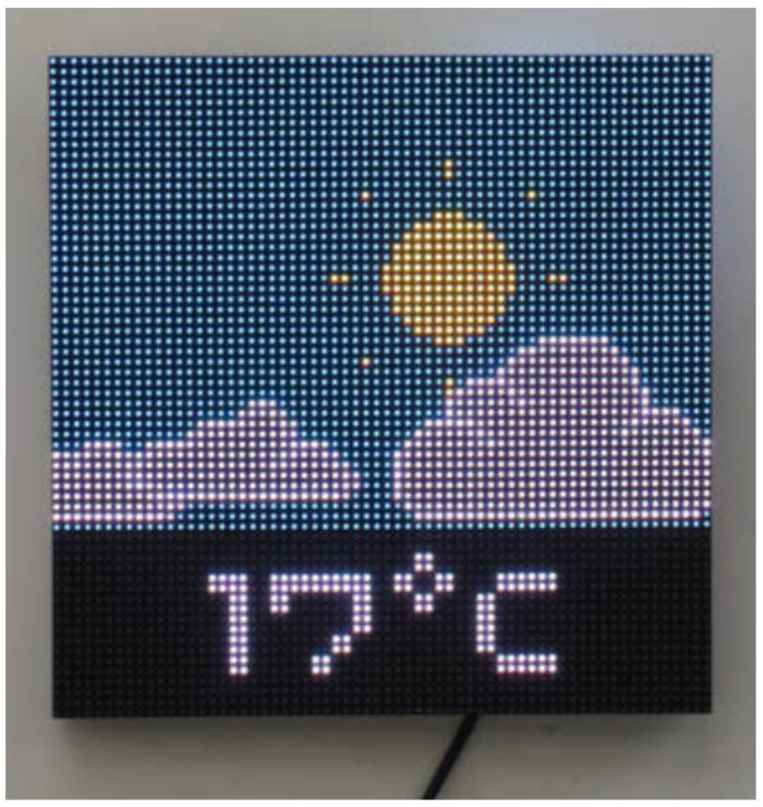


Bild 12: Die Wolken fliegen von rechts nach links und die Sonne strahlt.

anderen dort ein. Wie man im Listing „wetter_03.py (Animation)“ sehen kann, ruft die Funktion `sun_animation()` die Einzelbilder aus dem Array `sun_frames` alle 0,5 Sekunden nacheinander auf. Die Variable `sun_cur` speichert dabei den aktuellen Frame und wird bei jedem Durchlauf so lange hochgezählt, bis alle Frames in `sun_frames` abgespielt sind. Danach beginnt die Animation von vorn. Im `while True:` schreibt der Befehl `show_array()` dann den jeweils aktuellen Sonnen-Frame in den Puffer (Bild 12).

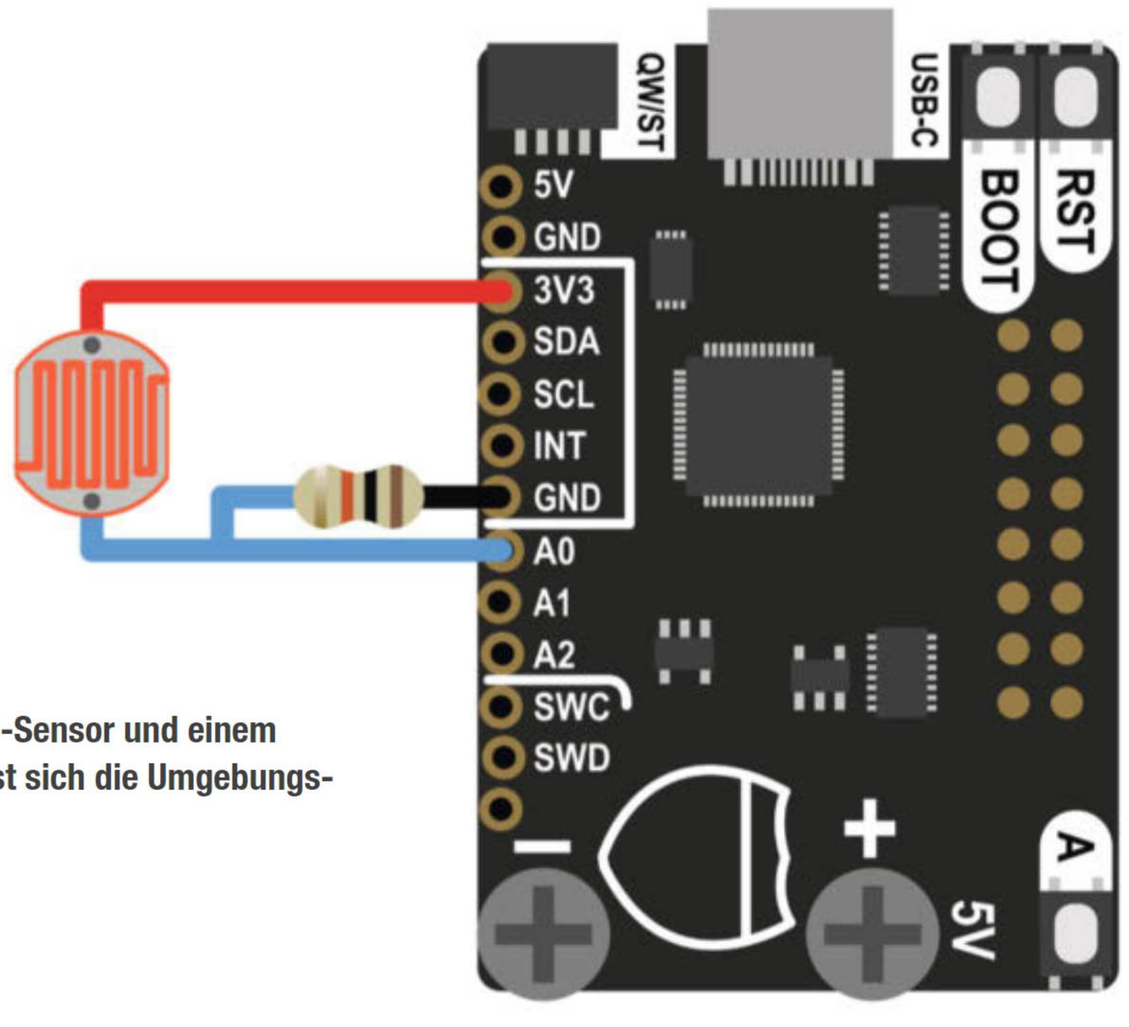


Bild 13: Mit einem LDR-Sensor und einem 10-kΩ-Widerstand lässt sich die Umgebungshelligkeit messen.

Helligkeit regeln

Die Interstate75-Bibliothek bietet keinen direkten Befehl, um die Helligkeit der LED-Matrix zu steuern, und auch der Befehl `set_backlight()` in Pico Graphics hat keine Auswirkungen auf HUB75-Panels. Mit einem kleinen Workaround lässt sich die Matrix dennoch dimmen. Dazu multipliziert man die RGB-Werte der einzelnen Farben mit einer Variable, die die Helligkeit steuert.

Diese Variable heißt in wetter_04.py `brightness` und ihr Float-Wert kann zwischen `0` und `1` liegen (z. B. auch bei `0.003`). Wenn ich einen Rotwert von `128` mit dieser Variable multipliziere und `brightness` den Wert `0.5` hat, erhalte ich als Ergebnis `64` – und damit ein halb so hell leuchtendes Rot. Überträgt man dieses Prinzip auf alle Farben in der Wetteranzeige, lassen sie sich wie im Listing „wetter_04.py (Helligkeit)“ definieren. Da die RGB-Werte aus ganzen Zahlen bestehen müssen, werden sie mit `int()` in diese umgewandelt.

Jetzt fehlt nur noch eine Möglichkeit, den Wert von `brightness` zu beeinflussen. Dafür kann man einen Sensor an einen der Analog-Pins des Interstate 75 anschließen. Diese sind mit A0, A1 und A2 gekennzeichnet und stehen für die GPIO-Pins 26, 27 und 28 am RP2040. Verbindet man einen LDR-Sensor (**L**ight **D**ependant **R**esistor oder Fotowiderstand) wie in Bild 13 mithilfe eines 10-kΩ-Widerstands mit A0, kann man ihn mit

```
ldr = ADC(Pin(26))
```

in sein Programm einfügen und dessen Messwerte in der Dauerschleife mit

```
read_ldr = ldr.read_u16()
```

abfragen. Das Ergebnis wird in 16 Bit ausgegeben, also mit Werten von `0` bis `65535`. Wenn man die maximale Helligkeit (`brightness = 1`) durch den höchsten Messwert 65536 teilt, erhält man 0,000015. Multipliziert man dieses Ergebnis mit `read_ldr`, bewegen sich die Werte von `brightness` immer zwischen `0` und `1`. Und da sich die Helligkeit der LEDs nicht so oft an die Umgebung anpassen muss, ändere ich diese nur alle zwei Sekunden (man könnte den Wert auch noch höher setzen). Mit `global` erlaube ich der Funktion, die Farbwerte global anzupassen, sonst springen diese immer wieder auf ihren Ursprungswert zurück, sobald die Funktion abgearbeitet ist. Anstatt eines LDR kann man auch ein Poti anschließen, aber dass sich die Helligkeit automatisch dimmt, ist schon praktisch.

Bei einem Interstate 75 W sind die analogen Pins nicht extra gekennzeichnet, aber da Pimoroni anstatt eines RP2040 einen kompletten Raspberry Pi Pico aufgelötet hat, kann man sich an dem Pinout des Pico orientieren oder Bild 14 nutzen.

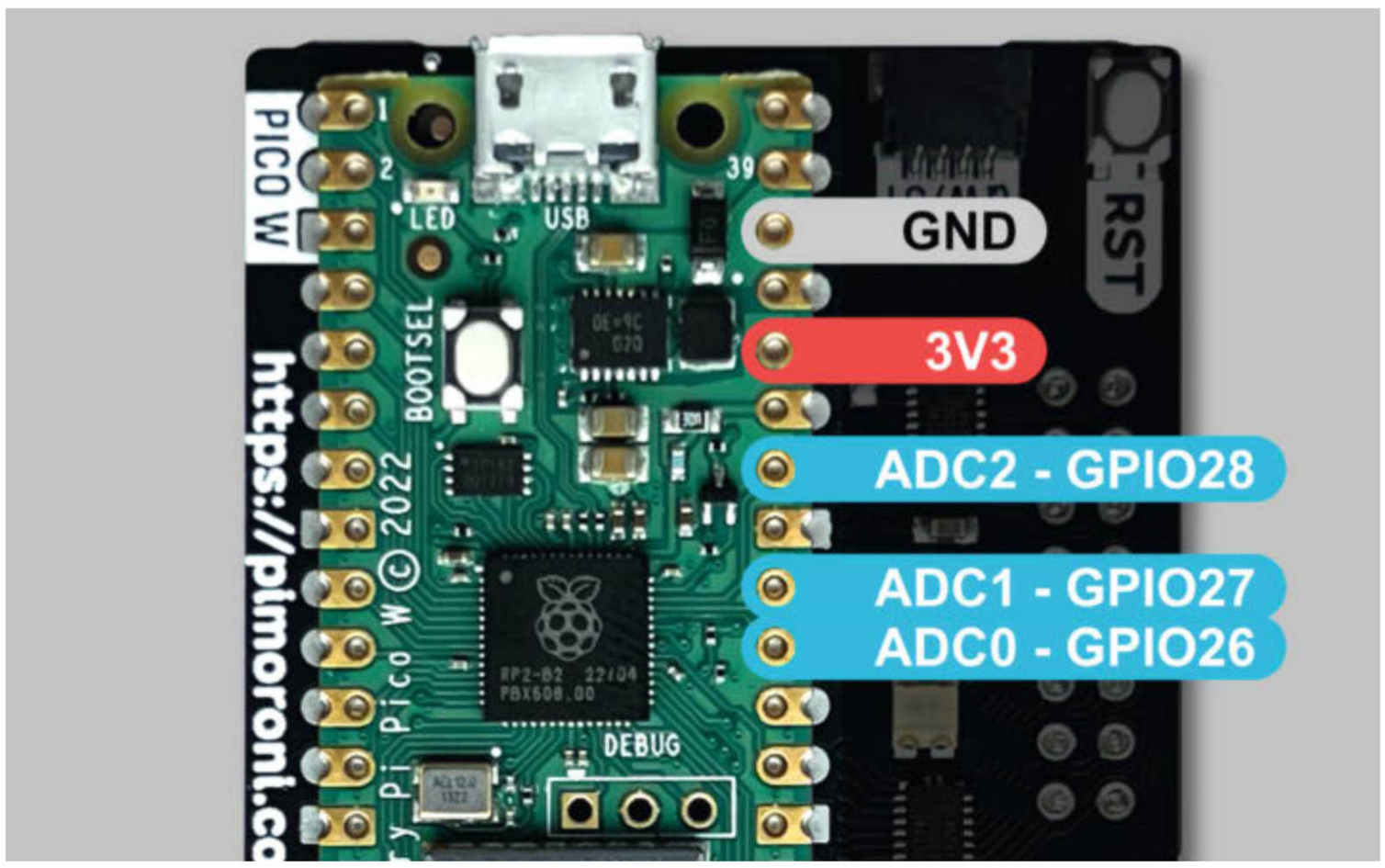


Bild 14: Am Interstate 75 W sind die Pins nicht extra beschriftet, wie bei dem Interstate 75.

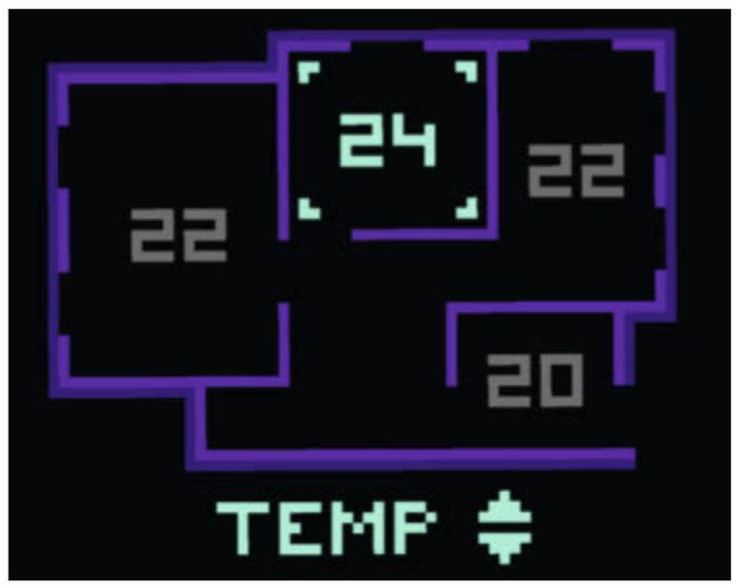
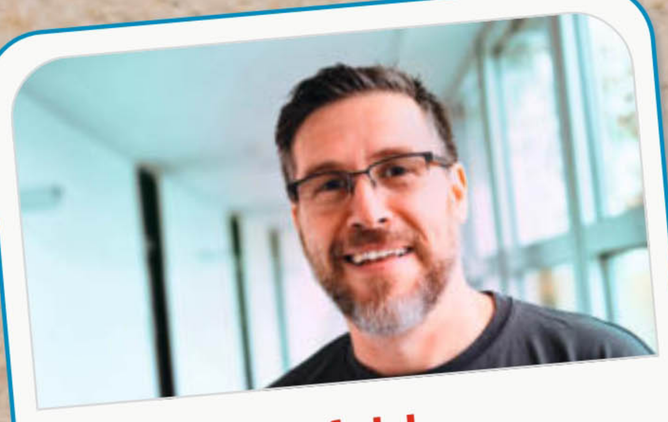


Bild 15: Wie wäre eine kleine Steuerung fürs Smart Home mithilfe eines Joysticks?

Weitere Möglichkeiten

Für eine vollständige Wetteranzeige fehlen jetzt noch ein paar echte Wetterdaten, die sich – einen Interstate 75 W vorausgesetzt – aus dem Internet ziehen lassen, sowie ein paar weitere Grafiken. Man könnte die Geschwindigkeit der Wolken in Relation zur tatsächlichen Windgeschwindigkeit setzen, den Himmel je nach Tageszeit farblich anpassen und die Sonne oder einen Mond auf- und untergehen lassen.

Mit einem Joystick ließen sich zudem auch tolle Spiele bauen oder eine kleine Steuerung fürs Smart Home (Bild 15). Falls man mit mehr Farben arbeiten möchte, könnte man die Array-Konvertierung auch feiner abstufen. Schickt uns gern eine Nachricht, wenn ihr mit dem hier Gelernten etwas umgesetzt habt. Bis dahin wünsche ich viel Spaß beim Ausprobieren. —*akf*



Daniel Bachfeld

Chefredakteur, dab@make-magazin.de

Ich nehme einen Klappspaten mit. Damit buddle ich mich dann unterirdisch zu anderen Inseln und den Kollegen weiter. Da sich der Spaten auch prima als Beil einsetzen lässt, kann ich damit Hütten bauen und Feuerholz machen und auf selbigem erlegtes Wild grillen. Für Hinweise, wie man Bier mit Spaten braut, wäre ich dankbar!

Marcus Hansson

Redakteur, mch@make-magazin.de

Zuerst dachte ich an ein Feuerzeug, Machete oder 3D-Drucker. Dann fiel mir ein, dass das letztendlich alles dasselbe ist, denn in der Antwort steckt die Vermutung, dass ich überleben und zurück in die Zivilisation möchte. Ich will aber nicht dazu beitragen eine unberührte Insel zu zerstören. Wäre ich gezwungen mich zu entscheiden, würde ich dem Rat eines guten Freundes folgen und ein Destilliergerät mitnehmen. Damit kann man aus Pflanzen Alkohol herstellen.

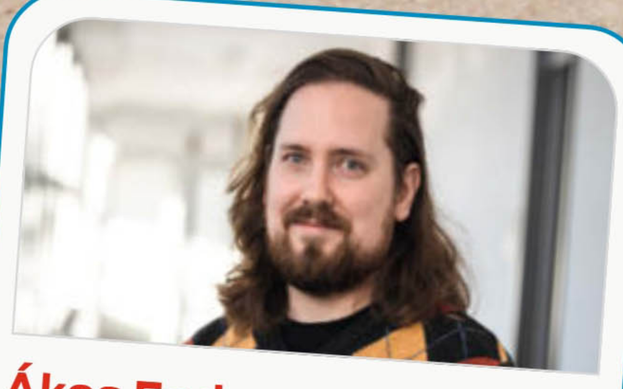
Wenn Du auf eine einsame Insel nur ein Werkzeug mitnehmen dürftest, welches wäre das? Und warum?

Unser Team

Carsten Wartmann

Redakteur, caw@make-magazin.de

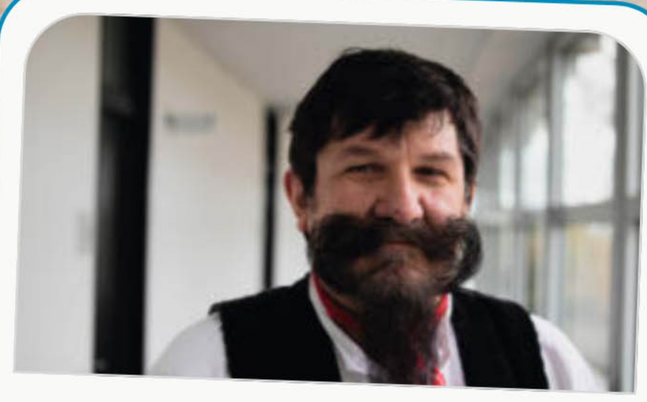
Ich würde einen Star-Trek-Replikator mitnehmen. Ok, Carsten, realistisch bleiben: einen 3D-Drucker. Mit Akku. Ach, und einen Computer für CAD. Dann doch lieber ein Multitool? Jetzt weiß ich es: einen Notfallsender. Lieber wieder schnell zu Hause sein und in sicherer Umgebung weiter maken.



Ákos Fodor

Redakteur, akf@make-magazin.de

Ich nehme auf jeden Fall meine japanische Säge mit und nutze die Gelegenheit, mich endlich mal ausgiebig mit Holz zu beschäftigen. Neben einer gemütlichen Hütte baue ich mir dann auch gleich ein Floß, um meine Kollegen auf den anderen Inseln zu besuchen und gemeinsam eine Grillparty zu feiern.



Heinz Behling

Redakteur, hgb@make-magazin.de

Ein richtig großes Schweizer Taschenmesser mit sämtlichen Extras. Seit der Besichtigung einer Manufaktur in der Nähe Zürichs, die die berühmten Taschenmesser herstellt, weiß ich, dass die kleinen Dinger mit nahezu allen zum Überleben notwendigen Werkzeugen ausgestattet werden können (entsprechender Geldbeutel vorausgesetzt). Daher wähle ich solch ein Schweizer Taschenmesser.

Johannes Börnsen

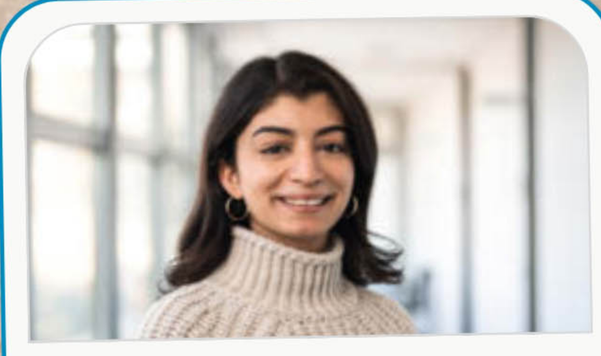
Redakteur und YouTube-Host,
jom@make-magazin.de

Ich würde eine große, scharfe Machete mitnehmen. Sie dient zum Schneiden von Nahrung, Bauen von Unterkünften und Erkunden des Urwaldes. Ihr vielseitiger Nutzen macht sie unentbehrlich auf einer einsamen Insel. Und wenn die Kokosnuss mir doof kommt, werde ich ihr zeigen, wer das letzte Wort hat! Mjam!

Daniel Schwabe

Technical Writer,
das@make-magazin.de

Ich würde ganz klassisch eine Axt mit auf eine Insel mitnehmen. Schneiden, hacken und mit der abgeflachten Seite sogar hämmern. Mit ein wenig Weitsicht lassen sich sogar eventuelle Schäden am Stiel on-the-Island reparieren. Ich denke, dass die Axt das perfekte Robinson-Crusoe-Werkzeug ist.



Dunia Selman

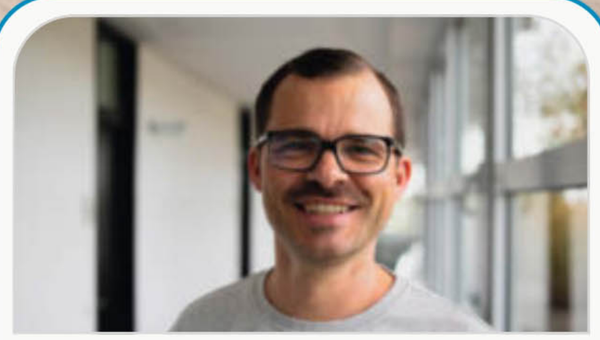
Social-Media-Managerin,
dus@make-magazin.de

Nach einer Phase des intensiven Studiums von Survival-Serien kann ich sehr selbstbewusst sagen: einen Feuerstahl. Mein größter Gegner als Frostbeule wird die Kälte sein und er ist nicht so begrenzt wie ein Feuerzeug.

Nicole Wesche

Mediengestalterin,
niwe@heise.de

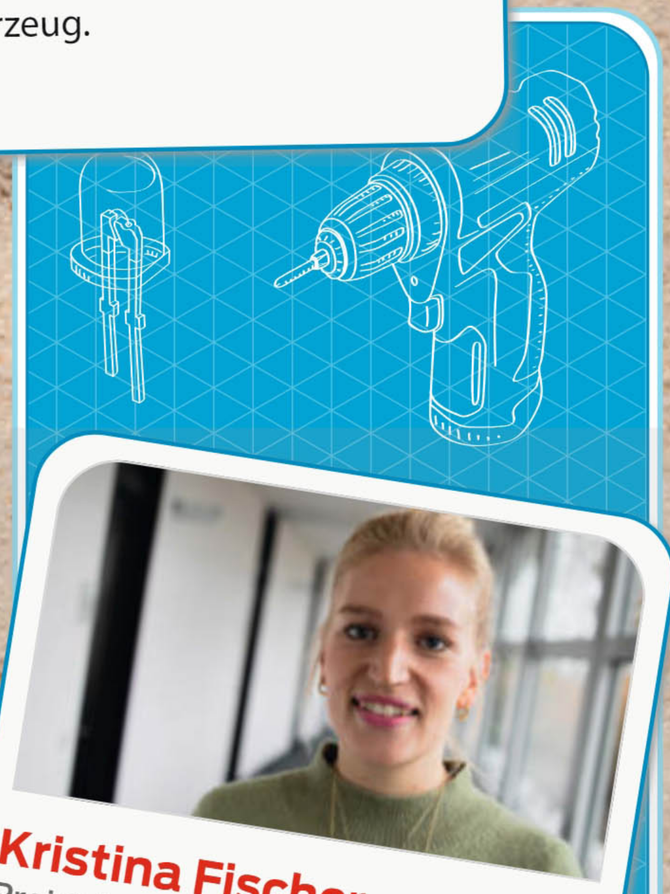
Da ich eh nicht fachgerecht mit Werkzeugen umgehen kann, nehme ich nur Block und Bleistift mit. Damit halte ich die Erlebnisse des restlichen Teams mit ihren mitgebrachten Werkzeugen fest. Wenn alles gut läuft und wir gerettet werden, kann ich das dann vielleicht erfolgreich in Buchform umsetzen!



Daniel Rohlfing

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

Ein dickes Schweizer Taschenmesser. So habe ich gleich mehrere sinnvolle Werkzeuge in einem kompakten Format dabei. Der kanadische Astronaut Chris Hadfield sagte einmal: „Verlassen Sie den Planeten nie ohne!“ Wenn ein Schweizer Taschenmesser im Weltall hilft, dann sicherlich auch auf einer Insel.



Kristina Fischer

Projektleitung Maker Faire,
krfi@maker-faire.de

Aloha! Ich würde einen Multischleifer mitnehmen, inklusive einer breiten Auswahl an Schleifpapier. Im besten Fall finde ich Treibholzbretter von alten Holzschiffen und schleife mir daraus ein Surfbrett, mit dem ich jeden Morgen zum Sonnenaufgang surfen gehen kann. Muss nur noch der Swell stimmen.

Louis Behrens

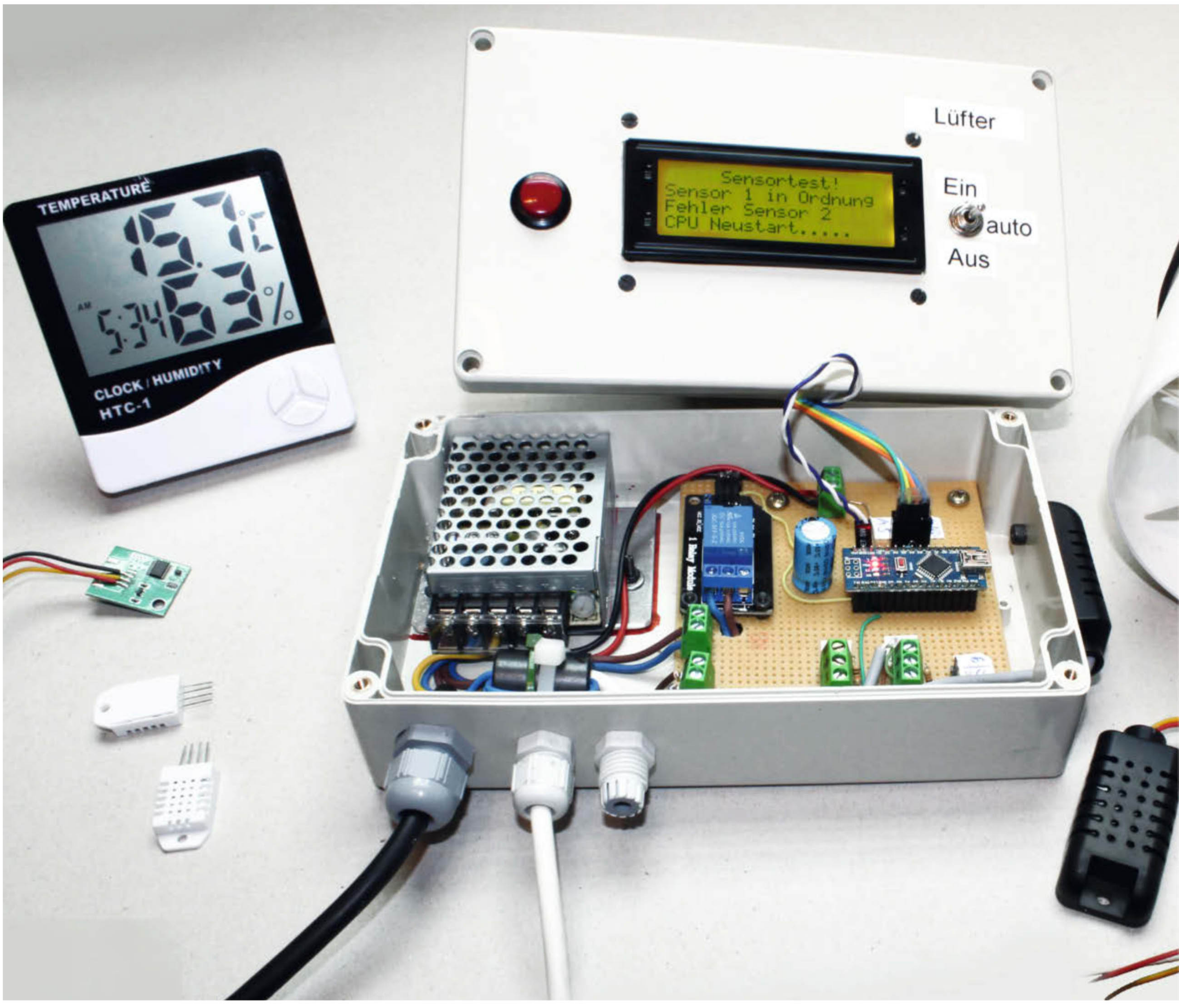
Werkstudent Social Media,
loub@make-magazin.de

Ich würde ein Multifunktions-Taschenmesser mitnehmen, weil es viele Funktionen wie eine Säge, ein Messer oder Feuerstahl beinhalten kann. Außerdem ist es handlich.

Nele Krautberger

Werkstudentin Maker Faire,
nekr@maker-faire.de

Wenn ich auf einer einsamen Insel stranden würde, würde ich einen kleinen Akkuschrauber von Bosch auf jeden Fall dabeihaben wollen.



Taupunkt-Lüftungs-system im Smart Home

Nach dem Hochwasser bleiben oft die Kellerwände feucht. Da hilft das selbst gebaute Taupunkt-Lüftungssystem. Um alle Kellerräume von der Hochwasserfeuchtigkeit zu befreien, habe ich das Lüftungssystem ins WLAN integriert, den Datenaustausch per MQTT ermöglicht und so mehrere im Kellergeschoss verteilte Ventilatoren zentral steuerbar gemacht.

von Jan Moritz Behnken

Kurzinfo

- Datenerfassung mit ESP8266 und MQTT-Broker
- Individuelle Ventilatorsteuerung mit jeweils eigenem Mikrocontroller
- Grafische Darstellung der Feuchtigkeit mittels Telegram-Bot

Checkliste

Zeitaufwand:
2 bis 3 Stunden (je nach Anzahl der Ventilatoren)

Kosten:
ca. 40 Euro pro belüftetem Raum

Material

- **Wemos D1 Mini** (ein Board für die Messungen zzgl. je eines pro Ventilator)
- **USB-Netzteil mit Kabel, passend zum Wemos-Board**
- **2 Sensoren DHT22**
- **Lochrasterplatine**
- **2 Micro-USB-Breakout-Boards** (z. B. Berrybase, Art.-Nr. MUSB-BO)
- **2 USB-A-Breakout-Boards** (z. B. Berrybase Art.-Nr. BOB-12700)
- **2 USB-Kabel** (USB-A auf Micro-USB, Länge je nach Abstand zwischen Sensor und Wemos-Board)
- **2 Buchsenleisten** (4-polig, RM 2,54 mm)
- **2 Stiftleisten** (4-polig, RM 2,54 mm)
- **Ventilator 230 V** (Einbauversion, je nach räumlicher Gegebenheit)
- **Schaltsteckdose**
- **MQTT-Server** (Raspberry Pi 3 mit Netzteil, Speicherkarte und Netzwerkanschluss oder Smart-Home-Server)

Werkzeug

- **Lötausrüstung**
- **Bohrmaschine**
- **Stichsäge**
- **Schraubendreher**

Mehr zum Thema

- Ulrich Schmerold: Das Taupunkt-Lüftungssystem, Make 1/22, S. 22
- Peer Stelldinger: Eigenbau-Wärmetauscher für das Taupunkt-Lüftungssystem, Make 4/22, S.20
- Ulrich Schmerold: Logging-Funktion für das Taupunkt-Lüftungssystem, Make 2/22, S. 82

Alles zum Artikel im Web unter
make-magazin.de/x8he

Der Anlass für meine Überlegungen und Recherchen zu Fragen einer Taupunkt-Lüftung waren die Folgen des Hochwassers zum Ende des vergangenen Jahres. In der Region, in der meine Eltern wohnen, stieg infolgedessen der Grundwasserspiegel erheblich an und das Wasser fand auch einen Weg in die Kellerräume. Die jüngsten Fluten in Süddeutschland zeigen, dass das vermutlich kein Einzelereignis gewesen sein wird.

Nachdem nach langen Wochen endlich kein Wasser mehr durch Wände und Bodenplatte sickerte, blieb das durchfeuchtete Mauerwerk zurück. Mein Ziel war es jetzt, die Feuchtigkeit in den Kellerwänden so schnell wie möglich zu reduzieren und die relativ hohe Luftfeuchtigkeit in den Kellerräumen nach draußen zu befördern.

Dabei habe ich mich an das von Ulrich Schmerold vorgestellte Taupunkt-Lüftungssystem (siehe Kurzinfo) erinnert: Über zwei Feuchtigkeitssensoren und einen Arduino steuert er einen kleinen Ventilator, der immer zur richtigen Zeit feuchte Luft aus dem Keller befördert, während trockenere von draußen über eine weitere Öffnung nachströmen kann.

Weil wir es bei uns mit einem großen Keller zu tun haben, entschieden mein Vater und ich uns für den Einsatz mehrerer Lüfter. Da eine direkte zentrale Steuerung durch nur ein Mikrocontroller-Board erheblichen Verkabelungsaufwand bedeutet hätte, haben wir das System deshalb so angepasst, dass wir es in ein Smart-Home-System integrieren konnten und es sich auch von außerhalb überwachen lässt. Genauer gesagt erfolgt der Datenaustausch über einen MQTT-Broker, der entweder als Stand-alone-Version auf einem Raspberry Pi eingerichtet wird oder Bestandteil eines (bereits vorhandenen) Smart-Home-Servers ist. Von diesem Broker holen dann wiederum die zu jedem Ventilator gehörenden Controller Daten und schalten ihren Lüfter entsprechend ein und aus.

Die Hardware

Das Kernstück der Messwerterfassung ist hierbei ein Wemos-D1-Mini-Board mit zwei DHT22-Sensoren (siehe Bild 1), das deren Messwerte an den MQTT-Broker übersendet. Mittels Node-RED auf einem Raspberry Pi werden dann das Daten-Logging und die Steuerung der Lüfter realisiert. Und um das System schlussendlich noch im Auge behalten zu können,

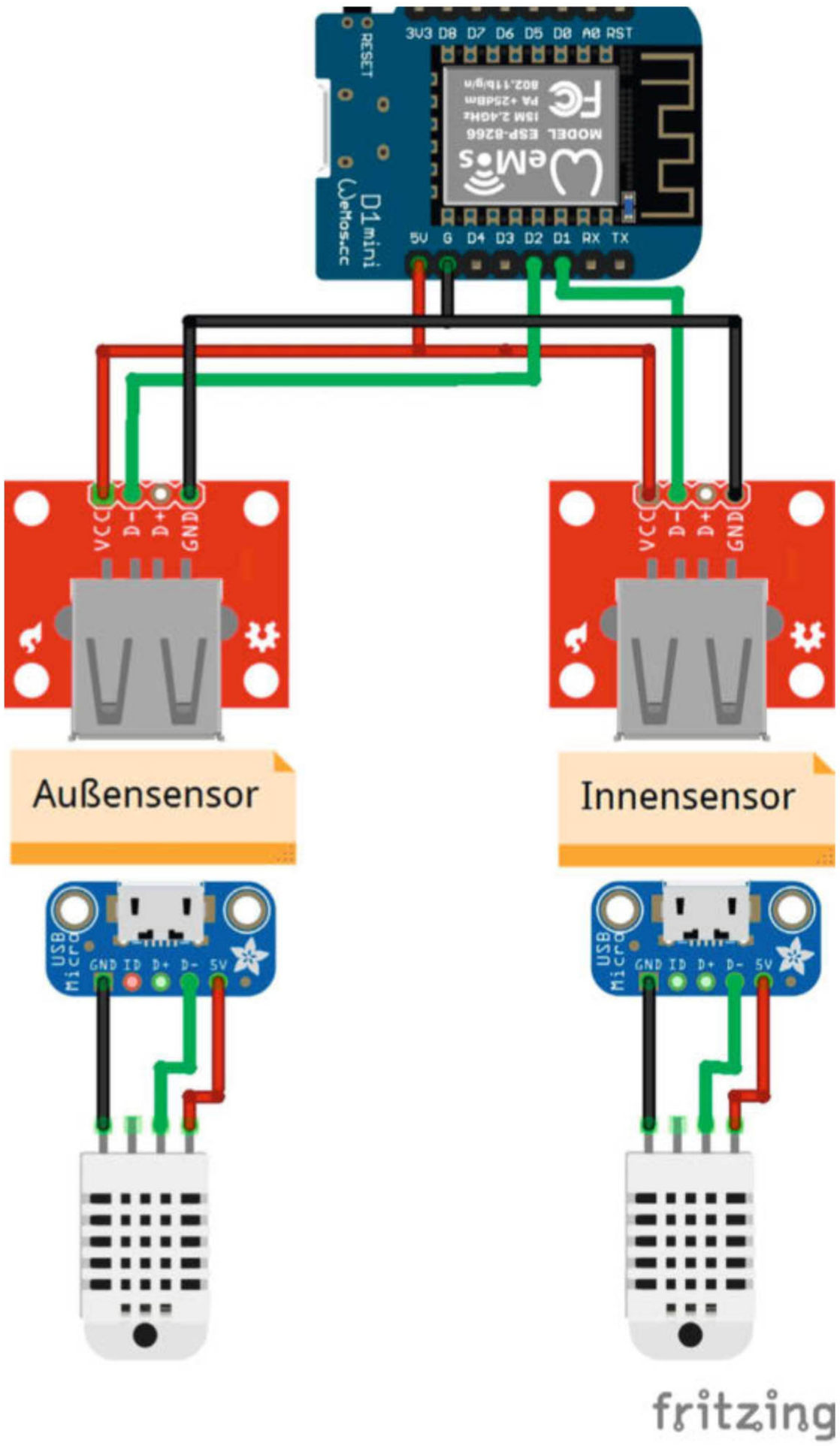


Bild 1: Der Schaltplan der Messzentrale ist sehr übersichtlich.

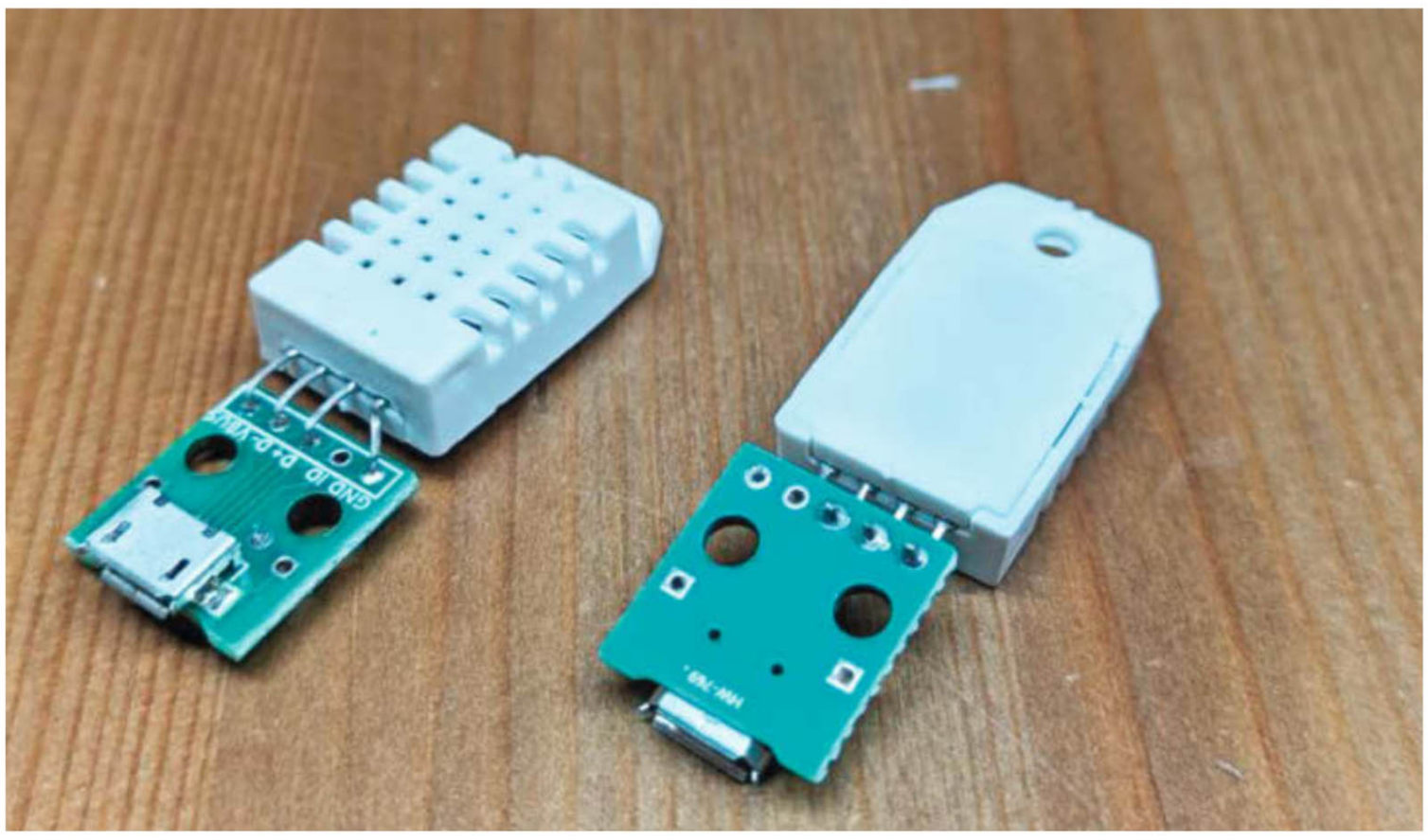


Bild 2: Die DHT22-Sensoren wurden an Micro-USB-Buchsen gelötet, um fertige USB-Kabel zur Verbindung verwenden zu können.

schickt es uns täglich Auswertungen des vergangenen Tages über einen Telegram-Bot zu.

Die DHT22-Sensoren liefern mir verlässliche Werte über die relative Luftfeuchtigkeit in Prozent und über die aktuelle Temperatur. Aus beiden Werten gemeinsam lässt sich dann näherungsweise die absolute Luftfeuchtigkeit in g/m^3 errechnen.

Sobald einer der Sensoren drinnen im Keller und der andere draußen vor einem Kellerfenster angebracht wird, lässt sich aus ihren Messergebnissen ermitteln, ob die Kellerluft oder die Umgebungsluft mehr Wasser enthält. Anhand dieses Vergleichs wird dann entschieden, ob der im jeweiligen Raum eingebaute Badezimmerlüfter angestellt wird und so den Austausch der zu feuchten Kellerluft mit der trockeneren Umgebungsluft startet.

Für die Kabelverbindung habe ich mir USB-Buchsen an die Sensoren gelötet (Bild 2). Verbunden sind sie dann über ein zwei Meter langes USB-Datenkabel mit dem Wemos-Board. Somit kann ich schnell einen der Sensoren wechseln, sollte mal ein Fehler auftreten. Auch lässt sich einfach ein längeres Datenkabel einstöpseln, wenn der Sensor doch noch weiter vom Fenster entfernt platziert werden soll.

Das Wemos-Board steckt auf einer Lochrasterplatine, die außer dem Board-Steckplatz auch noch Kontaktstifte für die beiden USB-Buchsen enthält. Die Verdrahtung erfolgte gemäß Schaltplan durch angelöteten Draht an der Unterseite (Bild 3).

Alles zusammen sitzt dann geordnet auf der Lochrasterplatine (Bild 4).

Diese Zentrale kam dann in ein Kunststoffgehäuse, das das Eindringen von Feuchtigkeit und Kurzschlüsse verhindern soll (Bild 5). Die Stromversorgung erfolgt über ein USB-Netzteil.

Auch die Sensoren kamen in solche Kunststoffgehäuse. Die sind aber nicht luftdicht, sodass eine Messung der Luftfeuchtigkeit weiterhin möglich war. Angebracht habe ich den Außensensor vor einem Kellerfenster der Nordseite des Hauses. Dies dürfte die kälteste und feuchteste Seite sein und damit den verlässlichsten Wert für die Außenfeuchtigkeit liefern. Der Innensensor liegt dann so weit weg wie möglich vom Fenster, sodass ich den Einfluss von draußen minimiere und einen repräsentativen Wert für den Innenraum messen kann.

Über einen WLAN-Repeater nahe des Treppenhauses habe ich dem D1 Mini den Zugang zu unserem Heimnetzwerk ermöglicht. Über das MQTT-Protokoll sendet der D1 Mini minütlich seine Messwerte zu dem MQTT-Broker (Mosquitto) auf einem Raspberry Pi 3B im Arbeitszimmer (Bild 6).

Die Software – Datenmessung

Den Code für die Messzentrale (DHT22_Sensoren.ino) und für die Ventilatoren-Boards

(MQTT_Relais.ino) gibt es im GitHub-Repository des Projekts zum Download (siehe Link in der Kurzinfo). Softwaretechnisch werden im Messprogramm zuerst die Bibliotheken eingebunden:

- `<DHT.h>` – um die DHT22-Sensoren auszulesen
- `<PubSubClient.h>` – um Daten mit einem MQTT-Broker auszutauschen
- `<ESP8266WiFi.h>` – um eine Verbindung zum WLAN aufzubauen
- `<ArduinoJson.h>` – um Unterstützung für JSON-Objekte zu erhalten
- `<math.h>` – um die benötigten Formeln berechnen zu können

Anschließend werden die festen Werte definiert. Dazu gehören sowohl die Pins für die Sensoren, die Zugangsdaten zum WLAN als auch die IP-Adresse vom MQTT-Broker (hier müsst ihr natürlich die bei euch passenden Angaben eintragen). Ebenso finden sich hier die Zugangsdaten zum Broker.

Auch habe ich die vermessenen Korrekturwerte für meine beiden Sensoren hier festgelegt. Damit nämlich beide Sensoren in gleicher Umgebung möglichst ähnliche Werte liefern, habe ich sie mehrere Stunden direkt nebeneinander betrieben, die Werte protokolliert und anschließend den Mittelwert für jeden der beiden Sensoren berechnet. Die Differenz der Mittelwerte nutze ich und kalibriere beide Sensoren. Während der eine um die halbe Differenz nach oben korrigiert wird, korrigiere ich den anderen um denselben Betrag nach unten. Somit verringere ich den systematischen Fehler und beide Sensoren sollten nun einen möglichst kleinen Unterschied bei ihren Messungen ausgeben.

Daraufhin werden die Verbindungen zum WLAN, dem Broker und den Sensoren vorbereitet und in der `setup`-Funktion gestartet. In der Funktion `absoluteHumidity` führe ich die Berechnungen für den absoluten Feuchtigkeitswert (g/m^3) in der Luft durch. Eine nähere Beschreibung der Formel gibt es im ursprünglichen Artikel zum Taupunkt-Lüfter (s. Kurzinfo).

In der `loop`-Funktion führe ich fünf Messungen hintereinander durch und bilde daraus den Mittelwert für den finalen Vergleich. Sollte der Messwert zu sehr schwanken, wird er hierdurch etwas geglättet. Zu guter Letzt erzeuge ich ein JSON-Objekt mit den gemessenen und den berechneten Werten. Dieses wird im Anschluss an den Broker gesendet und steht dort dann in einem eigenen Topic anderen Programmen zur Weiterverarbeitung zur Verfügung.

Damit sich bei langer Betriebsdauer keine Fehlerquellen einschleichen, lasse ich den D1 Mini nach 30 Sekunden neu starten und alles Obige erneut ausführen.

Eine mir bis dahin unbekannte Fehlfunktion tauchte bei der ersten Inbetriebnahme

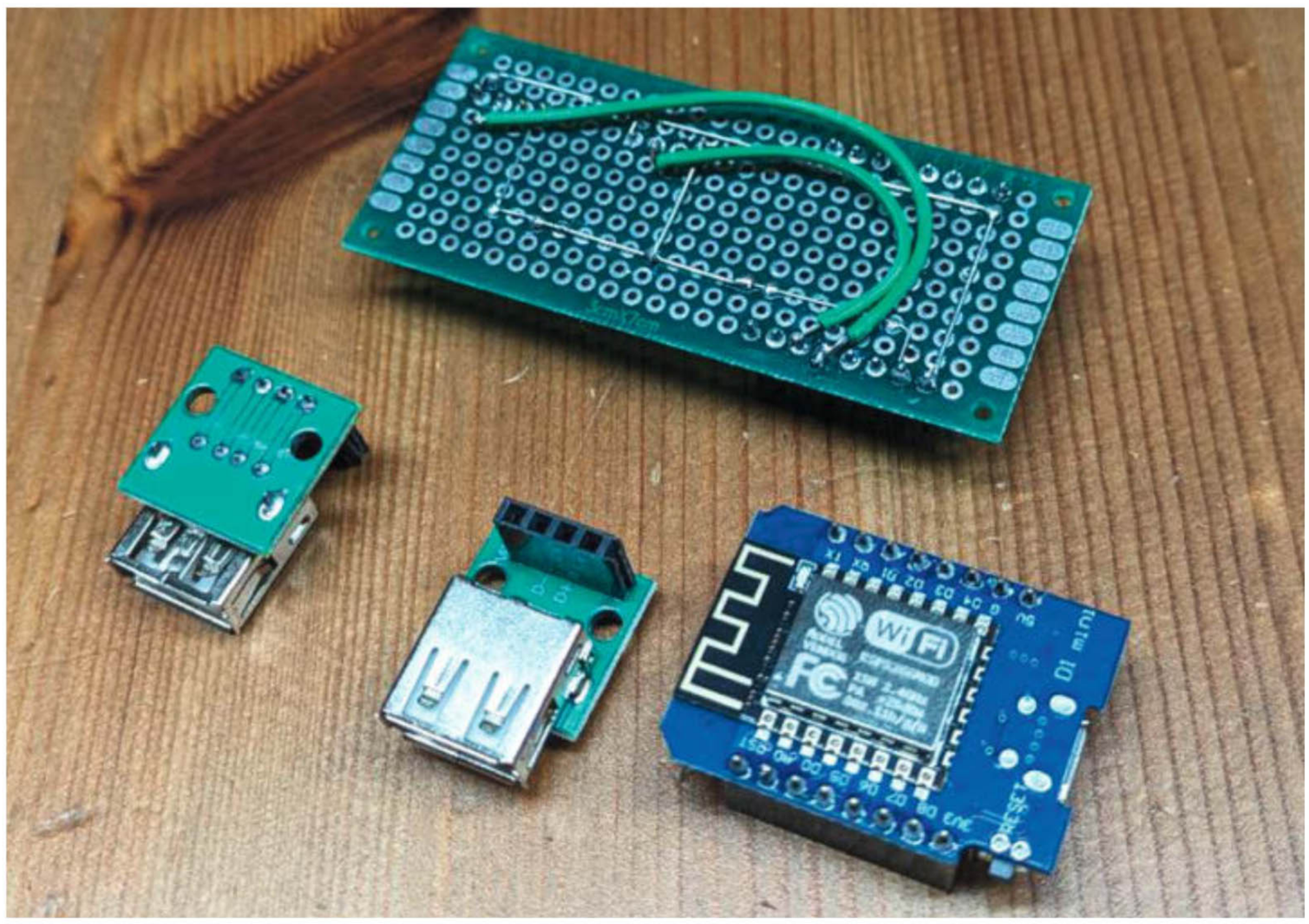


Bild 3: Die Lochrasterplatine ist an der Unterseite verdrahtet.

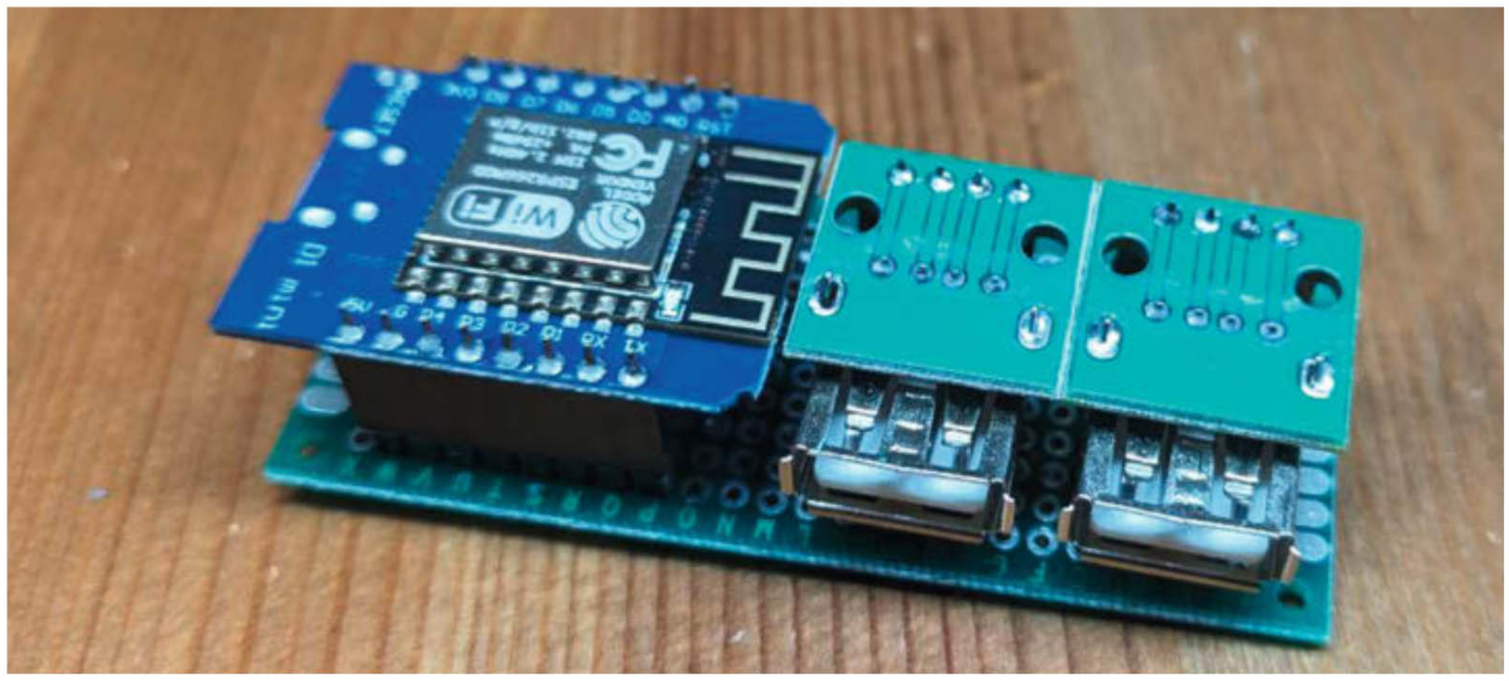


Bild 4: Sauber und platzsparend sitzt alles auf der kleinen Hauptplatine.

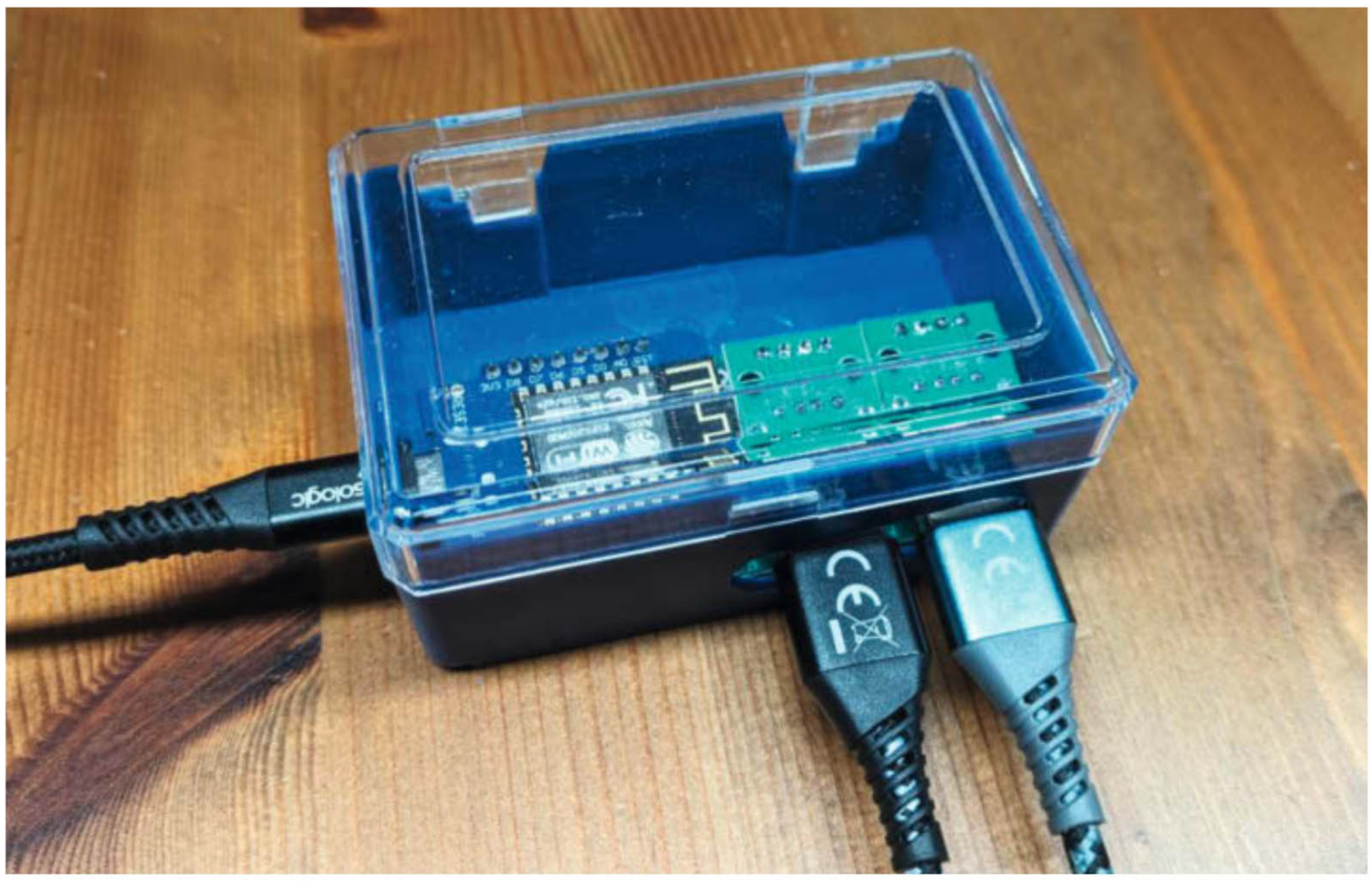


Bild 5: Ins Kunststoffgehäuse müssen Öffnungen für die Sensorstecker und das USB-Kabel zum Netzteil geschnitten werden.

Bild 6: Raspberry Pi 3B mit Stromversorgung und LAN-Kabel

auf: Unregelmäßig stürzte der Mikroprozessor ab und sendete nur noch fehlerhafte Daten. Nach einigen Untersuchungen fand ich den Zusammenhang zwischen dem Lichtschalter, der zugehörigen Leuchtstoffröhre und den Systemabstürzen. Immer wenn ein größerer Strom über den Stromkreis startete oder stoppte, ging dies nicht spurlos am Mikrocontroller vorbei. Schlussendlich konnte ich die Abstürze umgehen, nachdem das System an einen anderen Stromkreis im Keller angeschlossen wurde.

Die Schaltzentrale

Empfangen werden die gesendeten JSON-Datenpakete von einem handelsüblichen Raspberry Pi 3B, welcher die weitere Verarbeitung und Speicherung übernimmt. Auch die Logik der Ventilatorsteuerung und die täglichen Telegram-Nachrichten habe ich hier implementiert.

Bevor ich ins Detail gehe, folgen wir dem Weg der Daten einmal für den genaueren Überblick: Der MQTT-Broker Mosquitto empfängt die Messdaten vom D1 Mini und stellt sie in einem Topic anderen Anwendungen im Netzwerk zur Verfügung. Mit dem Programm Node-RED nehme ich die Daten aus dem Topic entgegen (Bild 7) und verarbeite sie weiter. Zuerst wird hier die Außen- mit der Innenwassermenge verglichen und die boolsche Entscheidung für die Lüftersteuerung getroffen.

Damit ich auch im Nachhinein alles analysieren und nachvollziehen kann, übergibt Node-RED das um die Entscheidung erweiterte Datenpaket an eine Datenbank (InfluxDB), welche die langfristige Speicherung übernimmt. Parallel dazu leitet Node-RED die Entscheidung übers Lüften als Boolean wieder zurück zu Mosquitto in ein eigenes Topic. Somit können andere Geräte im Netzwerk jederzeit den aktuellen Lüfter-Soll-Zustand erfragen.

Damit ich die Programme verlässlich laufen lassen kann, habe ich mich für Docker und dessen Container als Grundlage entschieden. Um die benötigten Container einfacher steuern zu können, greife ich auf Docker-Compose zurück. Und um die einzelnen Container und deren Verbindungen untereinander nicht selbst definieren zu müssen, habe ich auf meinem Raspberry Pi eine Kopie des mpous/ming-Projekts von GitHub laufen.

Abseits dieser Logik habe ich noch ein Python-Skript für einen Telegram-Bot geschrieben (Telegram_Keller_Daily.py), das ebenfalls auf dem Raspberry Pi ausgeführt wird. Mehr dazu am Ende dieses Artikels.

In Node-RED lassen sich einfach Schnittstellen zwischen Programmen und Komponenten realisieren. Der angelegte Flow liest sich von links nach rechts (Bild 8).

Der erste Block ist die Verbindung zum MQTT-Mosquitto-Docker-Container. Sobald der Broker ein neues Datenpaket erhält, wird es an diesen Block weitergeleitet, da Node-RED

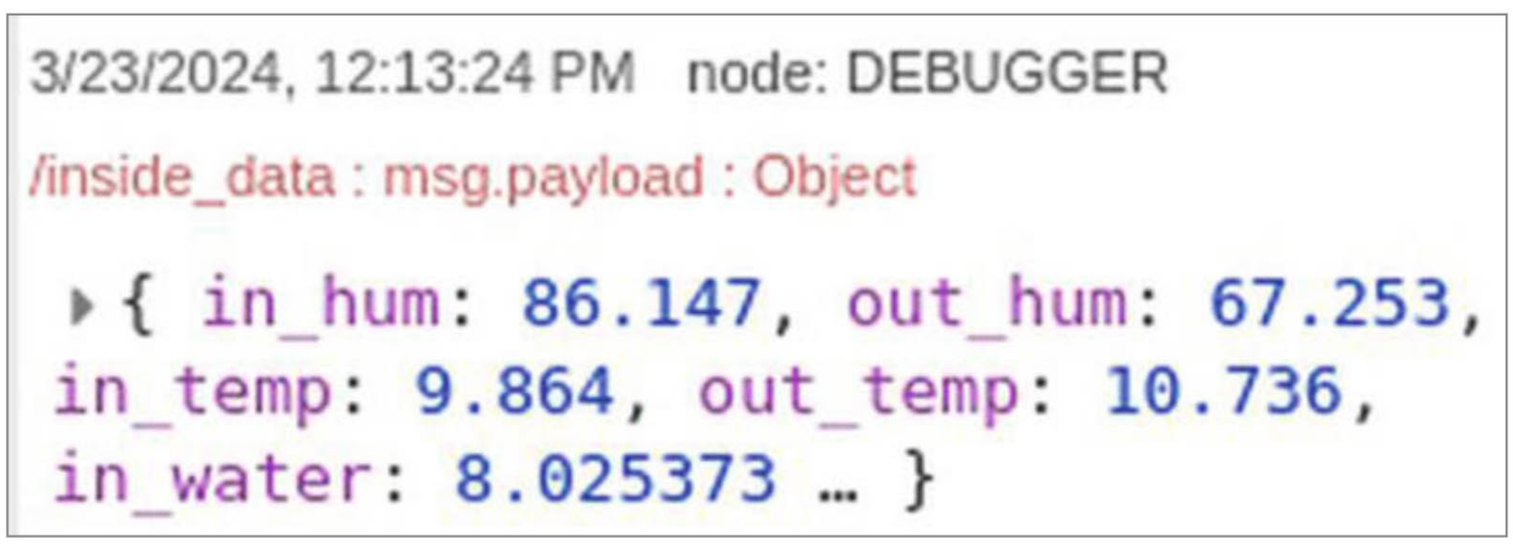


Bild 7: Die Messwerte in einem JSON-Objekt verpackt

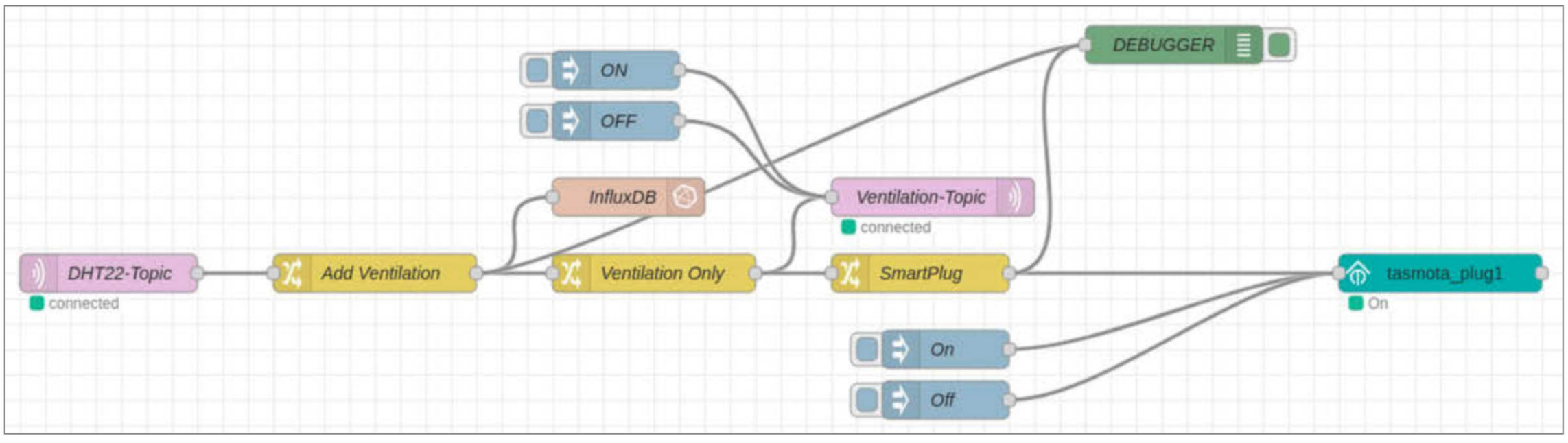


Bild 8: Node-RED-Flow für die Kellerbelüftung

hier als Subscriber des Topics fungiert. Im Block `Add Ventilation` prüfe ich, ob der Wassergehalt im Keller größer ist als draußen (Bild 9).

Einen Sicherheitspuffer von 0,5 g/m³ beziehe ich hier mit ein. Das Vergleichsergebnis wird als Boolean an das JSON-Datenpaket gehängt und an zwei separate Blöcke gereicht.

Einerseits wird das Paket an einen InfluxDB-Connector geleitet, welcher alle enthaltenen Werte einliest und mit einem Zeitstempel versehen in die Datenbank für die Archivierung abspeichert. Eine Kopie des Datenpakets geht dann direkt den Weg zur Steuerung der Lüfter. Weil ich zwei unterschiedliche Lüftersteuerungen benötige, splittet sich der Pfad nochmals zu einem Ventilation-Topic und einem TasmotaPlug.

Das Topic ist erneut in Mosquitto angesiedelt und wird schlussendlich von einem zweiten D1 Mini abgefragt, welcher dann den Ventilator über ein Relais steuert.

Neben den angesprochenen Blöcken finden sich noch zwei weitere (`Ventilation Only` und `SmartPlug`), in denen ich einfach nur den Typ der Daten anpasse, sodass der SmartPlug und der D1 Mini gut damit umgehen können. Zusätzlich ist ein Debugger sichtbar, der mir die Datenpakete nochmals in Node-RED

```
format expression
1   payload.in_water>payload.out_water + 0.5 ? 1 : 0
```

Bild 9: Diese Zeile checkt, ob der Wassergehalt im Keller höher ist als außerhalb.

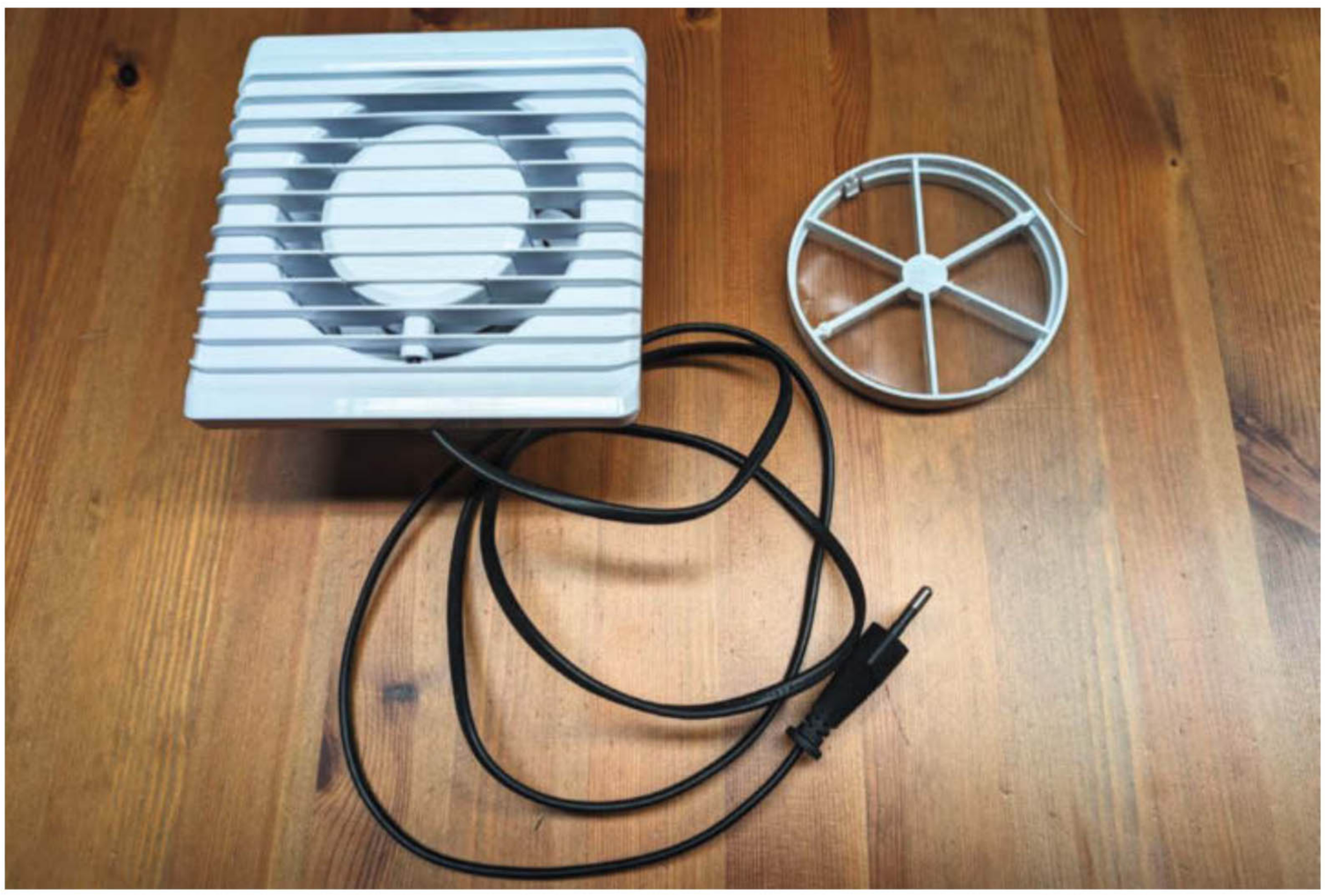


Bild 10: Der Ventilator mit Stromkabel und Rückstauklappe

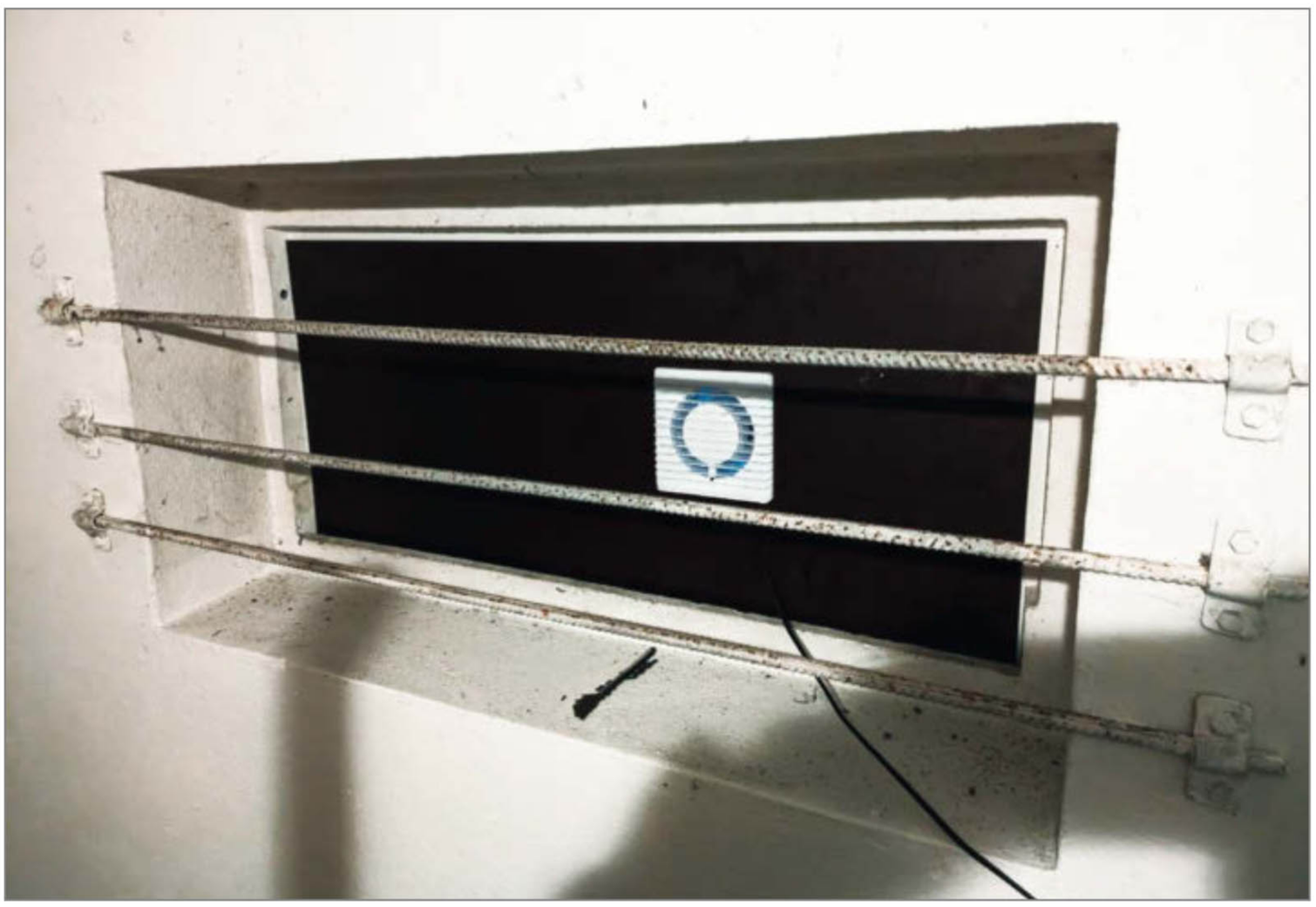


Bild 11: Die Glasscheibe im Fenster musste einer Siebdruckplatte weichen.

anzeigt. Auch vier Blöcke für manuelle Testeingaben „On/Off“ habe ich integriert.

Die Ventilator-Hardware

Belüftet wird mit einem günstigen Badezimmerventilator mit Rückstauklappe (Bild 10). Diesem haben wir ein reguläres Netzstromkabel mit Stecker spendiert und ihn in eine Siebdruckplatte eingesetzt.

Diese Platte sitzt anstatt in einer Glasscheibe im Kellerfensterloch (Bild 11). Weil wir hoffen, dass es ein vorübergehendes Projekt ist, haben wir das Tageslicht gegen den gesparten Arbeitsaufwand getauscht und keine dauerhafte Lösung angestrebt.

Die Steuerung des Ventilators läuft über eine handelsübliche Relaisplatine, die von einem zweiten Wemos D1 Mini gesteuert wird (Bild 12). Der Mikrocontroller wählt sich selbst ins WLAN ein und nimmt Kontakt zum MQTT-Broker auf. Dort meldet er sich als Subscriber für das Topic der Ventilatorsteuerung in Node-RED und reagiert damit direkt auf die Messungen.

Damit der D1 Mini auch betrieben werden kann, habe ich eine winzige 230-V-zu-5-V-Netzteilplatine eingebaut. Die zweigt vom Ventilatorkabel Strom ab und stellt dem Mikrocontroller und dem Relais stabile 5 V für den Betrieb bereit. Alles landet in einer Aufputzdose (Bild 13), muss dort aber kurzschlusssicher befestigt werden.

Erst im Anschluss an die Fertigstellung dieser Schaltung kam mir die Idee, einen handelsüblichen Smart Plug zu verwenden. Bei Action bin ich über eine günstige Version (Bild 14) gestolpert und auf den ersten Blick schien er auch Tasmota-fähig zu sein. Bei genauerer Prüfung nach dem Kauf stellte ich jedoch fest, dass ich die dritte Generation erworben hatte, bei der das System nicht mehr so einfach überspielt werden konnte.

Glücklicherweise kann man das integrierte WLAN-Modul mit der Steuerung durch eine ESP-02S-Platine ersetzen, welche schlussendlich wieder mit Tasmota geflasht werden kann. Gesagt, getan – und somit ist auch der zweite Ventilator im WLAN und über Node-RED steuerbar. Sollten wir in Zukunft noch weitere Ventilatoren integrieren, werde ich erneut den Weg über den Smart Plug wählen. Einmal gefunden, ist der Weg deutlich schneller und weniger fehleranfällig.

Die Überwachung – Telegram-Bot

Damit jeder aus meiner Familie sich einen schnellen Überblick über die aktuelle Situation im Keller verschaffen kann, ist ein Überblick über die gesammelten Daten und damit auch über die erreichten Erfolge von Anfang an eingeplant gewesen. Weil nicht jeder von uns permanent vor Ort sein kann und auch der Aufwand der Datenanalyse am PC für alle möglichst gering gehalten werden sollte, habe ich die täglichen Auswertungen über einen Telegram-Bot realisiert.

Der Bot läuft ebenfalls auf dem Raspberry Pi. Das Python-Programm Telegram_Keller_Daily.py lädt sich morgens die Daten der vergangenen 24 Stunden aus der InfluxDB-Datenbank herunter und visualisiert diese in ein paar Graphen (Bild 15). Schlussendlich postet der

Bild 12: Pro Ventilator ist ein Wemos-Board (vorn), ein Netzteilmodul (mitte) und eine Relaiskarte (hinten) erforderlich.

Bild 13: Die Teile passen in eine handelsübliche Aufputzdose, müssen darin aber kurzschlusssicher befestigt werden.

Bild 14: Alternativ zur Wemos-Relaiskarten-Kombination kann auch eine Schaltsteckdose verwendet werden, falls sie sich noch mit Tasmota flashen lässt.

Bot diese noch im Familienchat, sodass alle jederzeit nachschauen können, wie es gerade um die Feuchtigkeit im Keller steht.

Fazit

Für uns hat das Projekt die unsichtbare Feuchtigkeit im Keller sichtbarer gemacht und wir können nun gezielter dagegen vorgehen. Während die Hardware aus simplen Bausteinen besteht und auch die Software nicht zu kompliziert ist, so ermöglichen sie gemeinsam einen Einblick in ein äußerst komplexes System aus Feuchtigkeit, Temperatur und Belüftung (Bild 16).

In unserer Langzeitauswertung lässt sich erkennen, dass die Werte draußen unseren Keller stark beeinflussen und die Keller-Kurven den Draußen-Kurven folgen. Auch wenn der gleitende Durchschnitt draußen noch trockener ist als drinnen, so sorgen die feuchten Gemäuer noch dafür, dass der Keller feucht bleibt.

Als Erfolg verbuchen wir, dass die Differenz der absoluten Feuchtigkeit zwischen drinnen und draußen immer kleiner geworden ist. Durch das zeitlich gesteuerte Lüften konnten wir also schon eine gute Portion der Feuchtigkeit nach draußen pusten. Abzuwarten bleibt, wie es sich mit den steigenden Temperaturen verhält, wenn die Luft draußen generell mehr Feuchtigkeit aufnehmen kann und es dadurch immer seltener die Gelegenheit geben wird, korrekt zu lüften.

Insgesamt bin ich äußerst zufrieden mit dem Projekt und auch mit den Ergebnissen. Langfristig überlegen wir, das System eventuell mit weiteren Ventilatoren zu erweitern oder gar ein automatisch klappbares Fenster zu integrieren. Mein weiteres Ziel ist es, die Kellerfeuchtigkeit weniger von der draußen herrschenden Luftfeuchtigkeit abhängig zu machen. Auf jeden Fall werde ich die Daten weiterhin fleißig im Auge behalten und wünsche euch viel Erfolg bei ähnlichen Projekten. *—hgb*

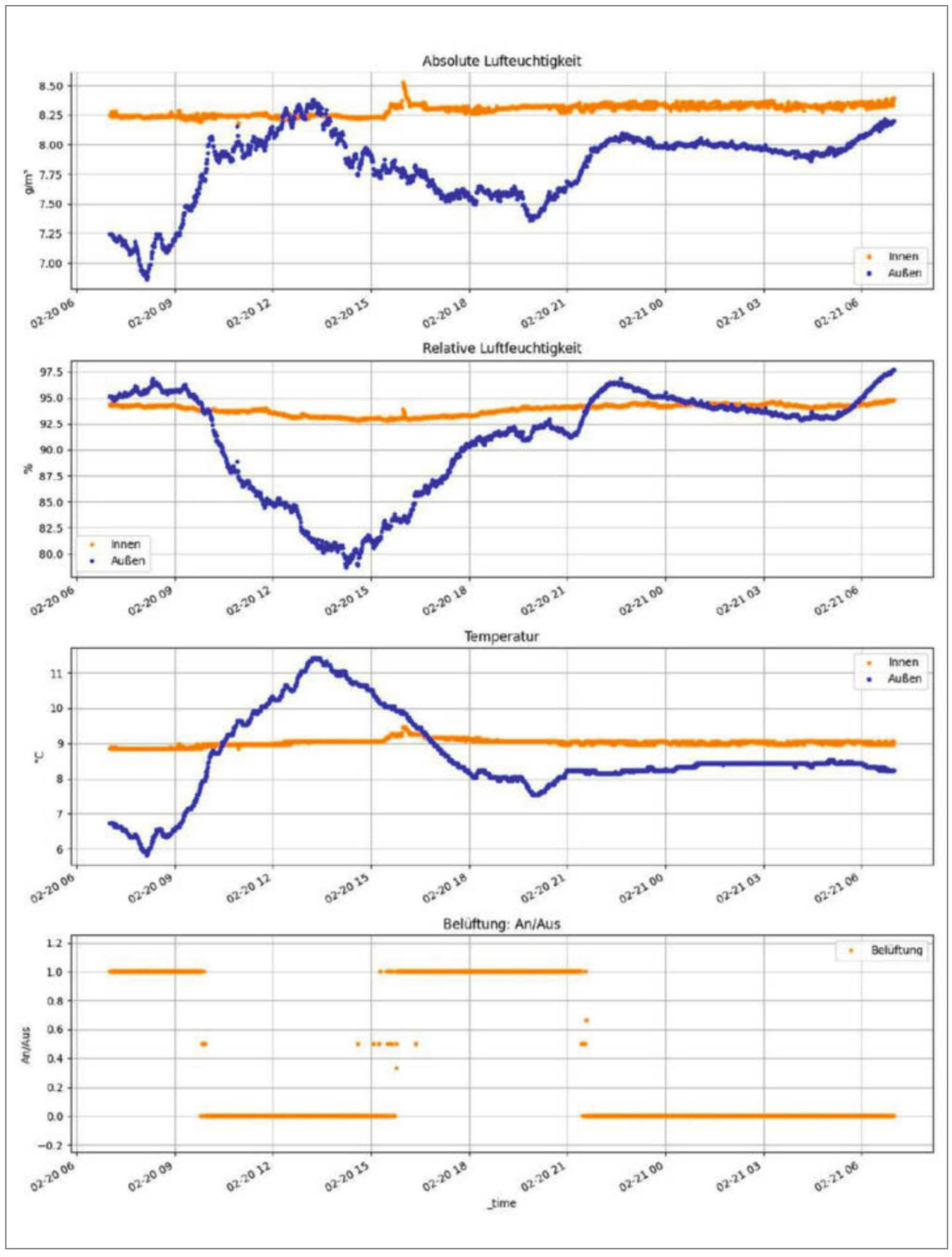


Bild 15: So sieht die tägliche Auswertung mittels Python und Telegram-Bot aus.

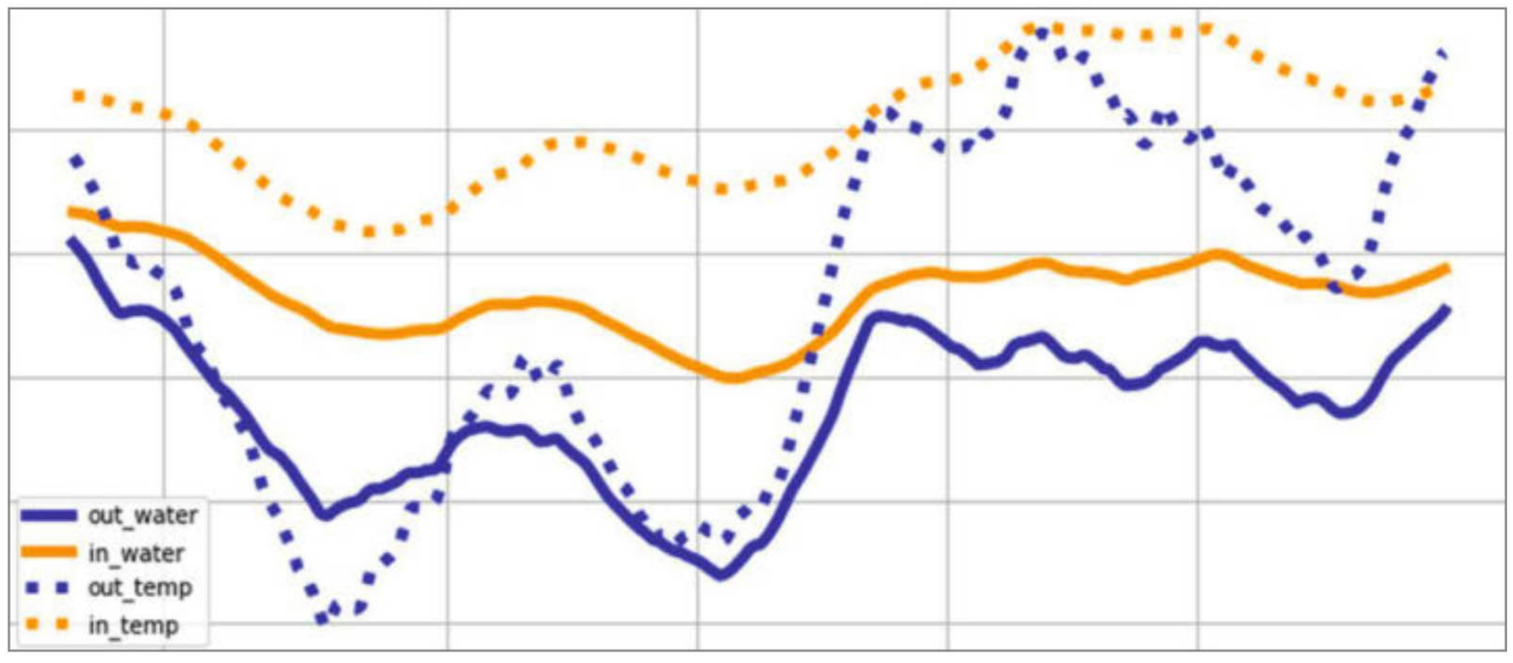


Bild 16: Diese Grafik zeigt die Feuchtigkeiten und Temperaturen innen und außen.

Alle Wege führen nach Hannover

Alles steht bereit für die 10. Maker Faire im Hannover Congress Centrum. Wir werfen schon mal einen Blick auf die Aussteller, die Hallen und das Rahmenprogramm.

von Daniel Schwabe

Alles ist bereit für das Maker-Highlight am 17. und 18. August 2024. Zur Jubiläums-Maker-Faire stehen rund 250 Aussteller in den Startlöchern, um ihre Projekte vorzustellen. Dieses Jahr haben sich besonders viele Hochschulen in Hannover angemeldet.

Was macht man auf einer Maker Faire?

Die Maker Faire ist die Anlaufstelle für Inspiration und die neuesten Informationen im Bereich Maker-Technik. Maker aus aller Welt bringen ihre Projekte mit und freuen sich auf den Austausch mit anderen Tüftlern. Denn von wem kann man sich bessere Tipps und Ideen holen als von anderen Makern? Maker-Networking heißt das Zauberwort! Viele Projekte sind auch interaktiv und lassen sich direkt vor Ort testen und in einer Vorführung in der Halle oder im Außenbereich bewundern.

Die anwesenden Firmen fahren ebenfalls die großen Geschütze auf und bringen High-Tech für den Maker-Bereich mit auf die Messe. Auf der Maker Faire bleibt man dahingehend immer topaktuell.

Als Maker-in-the-making ist die Maker Faire eine gute Anlaufstelle, um reinzuschnuppern in alle Aspekte, die der DIY-Bereich so bietet und vielleicht mit dem einen oder anderen neuen Hobby nach der Messe wieder nach Hause zu gehen.

Das alles gilt natürlich sowohl für die großen als auch für die kleinen Maker. Kindgerechte Projekte werden ebenfalls angeboten und viele Mitmachaktionen lassen die Zeit auf der Messe wie im Flug vergehen.

Die Redaktion auf der Maker Faire

Die Mitglieder der Make-Redaktion wuseln auf der Maker Faire herum. Auf 100 Quadratmetern an Stand 96 in der Eilenriedehalle stehen wir allen Lesern (und solchen, die es nach der Messe werden wollen) zur Verfügung. Egal ob Feedback zum Heft, Fragen zu einzelnen Artikeln oder Hilfegesuchen bei kniffligen Maker-

Vollgepackt mit guten Ideen: Bei einer Maker Faire ist jeder Winkel mit tollen Projekten belegt.

Problemen. Wer eine Idee für einen Artikel hat und gerne Make-Autor werden möchte, kann sich eine der unlimitierten Make-Visitenkarten abholen, um seine tolle Idee an die Redaktion zu schicken.

Der Make-Stand wurde dieses Jahr von der Redaktion komplett überarbeitet. Mehr Tische für mehr Projekte und entspannte Unterhaltungen. Natürlich alles im Self-Made-Make-Stil. Auf der Maker Faire Ruhr konnte sich das neue Konzept bereits im Kleinen bewähren und wird auf der Heimspiel-Maker-Faire sein volles Potenzial entfalten.

Natürlich kann der Make-Stand nicht nur personell und optisch punkten, sondern auch mit tollen Projekten aus dem Heft aufwarten. Welche Wunderwerke wir mit auf die Messe nehmen, wollen wir natürlich noch nicht verraten. Aber dass blinkende LEDs, surrende Motoren und formvollendet geschliffenes Holz dabei sein werden, versteht sich von selbst.

Für das (beaufsichtigte) Hands-On-Make-Erlebnis also unbedingt zu Stand 96 in der Eilenriederhalle kommen.

Wer auf der Messe seine ersten Schritte beim Maken probieren will, bekommt dafür am Make-Stand die Möglichkeit. In einem Löt-Workshop an Makey-Platinen können Löt-Neulinge unter Aufsicht fleißig üben. Wer sich lieber etwas vorlöten lassen möchte, für den bietet Make-Redakteur a. D. Carsten Meyer eine Darbietung seiner Lötkunst.

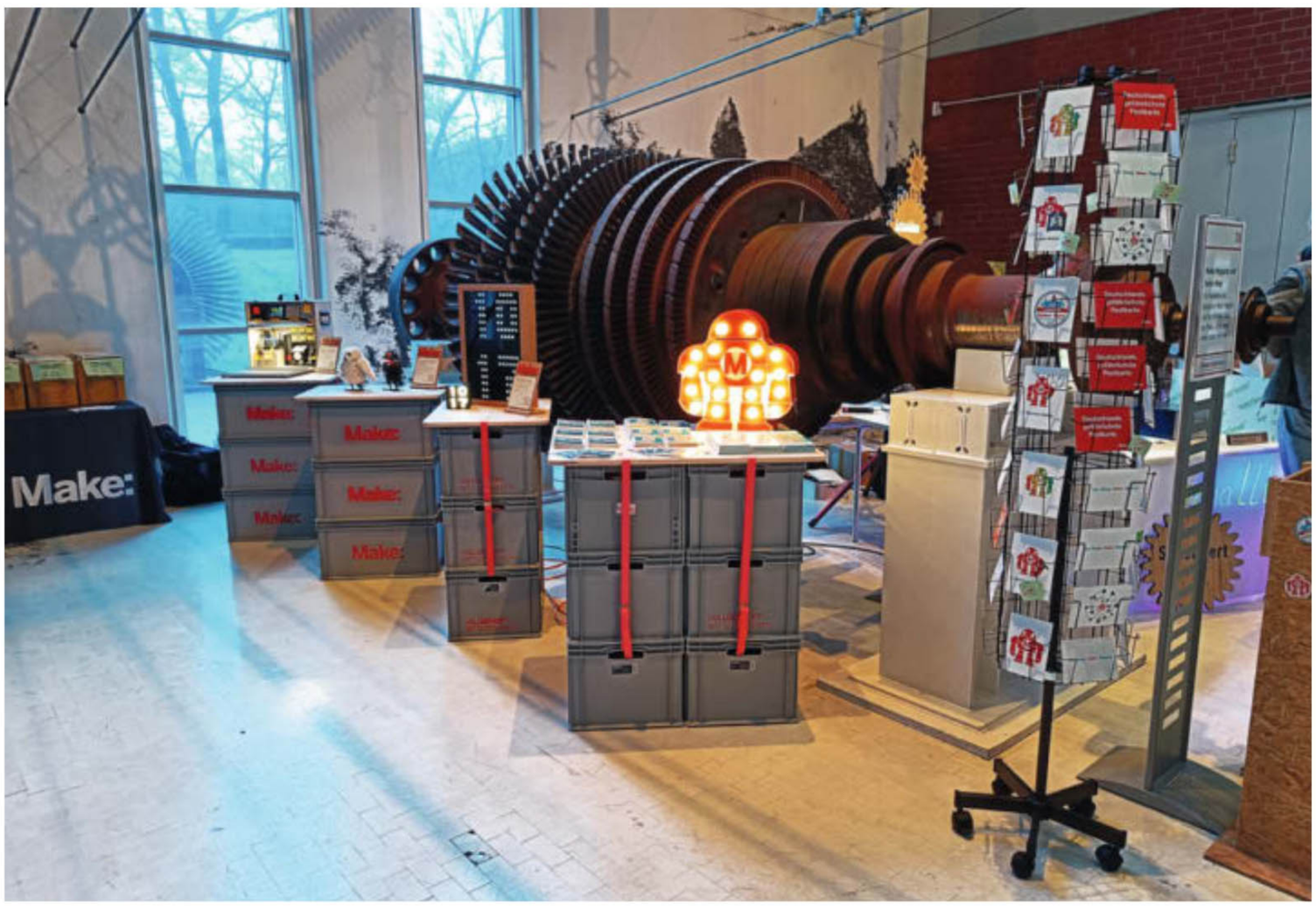
Aber nicht nur Make-Urgesteine sind zugegen. Der neue Kollege Marcus Hansson ist erstmals dabei und trumpft mit einem 2 Meter großen, mechanischen Vogel Strauß auf, der mittels eines 1800-W-Motors über die Messe läuft. Dieser war bereits als Make-Vertretung beim NoWhere-Festival zu bestaunen und wird jetzt in Hannover – natürlich artgerecht – im Außenbereich vorgeführt.

Von der Make für Maker

Dieses Jahr hat die Make etwas ganz Neues zu bieten: Make-Hardware. Maker wissen, was Maker brauchen. Deshalb hat sich die Make mit der IoT-Werkstatt des Umwelt-Campus Trier zusammengetan und eine Lernplattform für die Maker von Morgen entwickelt, mit der an echter Maker-Hardware programmiert und über verschiedene Sensoren die Umwelt erforscht werden kann. Alles natürlich im stylischen Makey-Design.

Mit der Firma Oxon bringt die Make-Redaktion eine schicke Makey-Version der Oxocard Connect auf den Markt und bündelt diese mit einem Elektronik-Kit und einem passenden Heft, neudeutsch Playbook.

Wer diese beiden Maker-Leckerbissen als Erster ausprobieren möchte, muss nur an den Stand der Make kommen. Nach Release werden diese Produkte über den heise Online-Shop verkauft.



Nur echt mit dem Make-Logo: Beim Werkstatt-Look fühlt man sich als Maker wohl.

Bald gibt es ein neues Bild! Schon 2023 war die Redaktion gespannt auf die Maker Faire 2024.

Vom NoWhere-Festival direkt auf die Maker Faire.

Und wer noch eine Hand für noch mehr Hardware frei hat, kann sich druckfrisch das neue Special-Heft zum Thema LoRaWan besorgen. Zusammen mit dem Partner ELV, der an Stand 36 in der gleichen Halle wie die Make ist, hat die Redaktion ein Experimentierkit nebst Programmieranleitung für den Funkstandard herausgebracht.

Abseits der Make

Nicht nur die Make vertritt den heise Verlag vor Ort. Auch der heise Shop ist wieder auf der Messe dabei und bietet vom Mikrocontroller über Makey-Figuren bis zum Make-Heftarchiv alles an, was das Maker-Herz begehrt. Und selbst die c't konnte der Verlockung der Maker Faire nicht widerstehen und hat dieses Jahr zum ersten Mal einen eigenen Messestand. Die c't ist - wie der Make-Stand - in der Eilenriedehalle, allerdings auf Platz 93, der Shop ist an Stand 95 zu finden.

Spannende Shows und Interaktion

Neben den Ständen mit ihren Projekten gibt es auch einige Shows, in denen meisterhafte Maker-Projekte und wissenschaftliche Wunder vorgeführt werden.

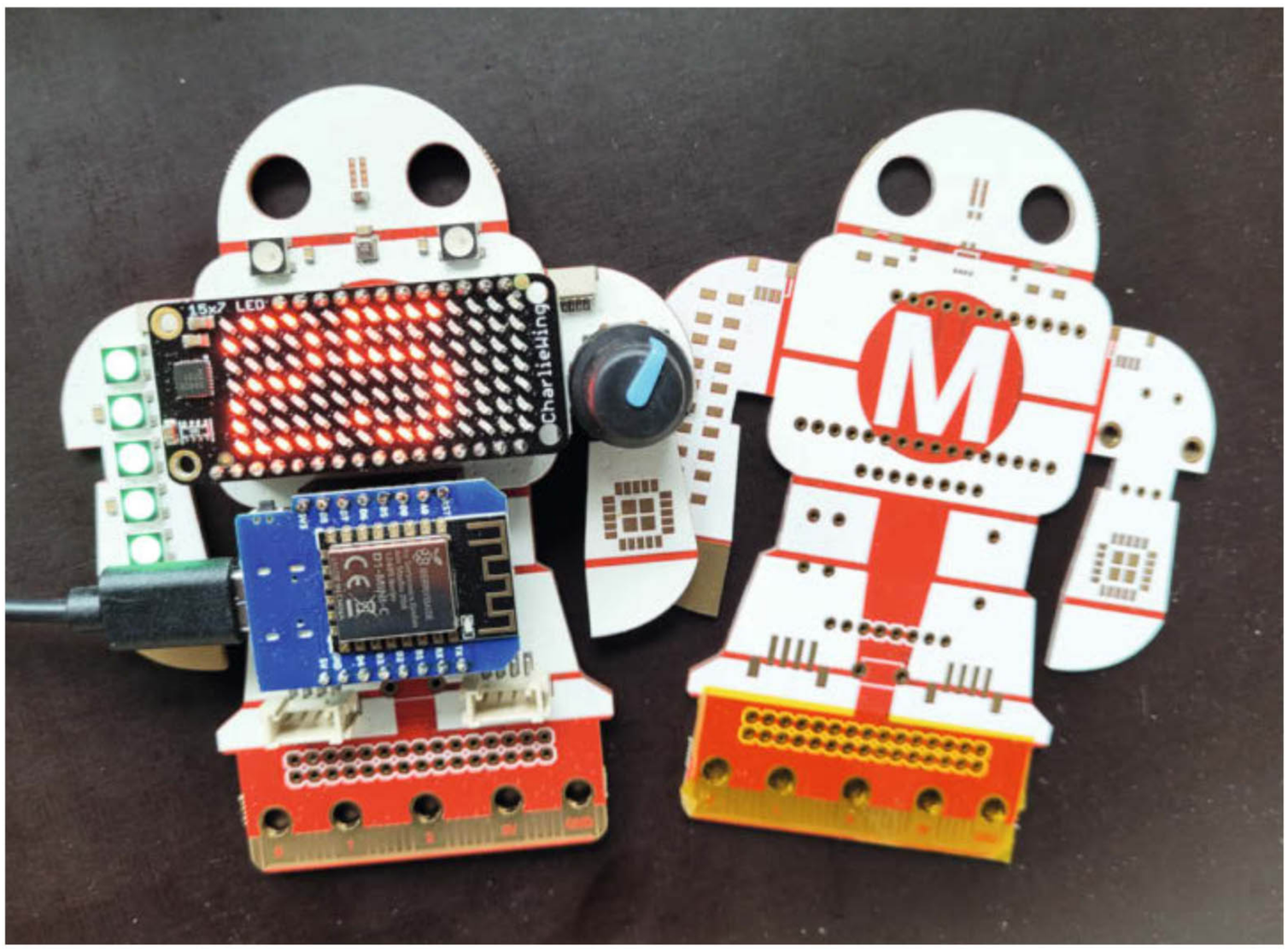
Beispielsweise führt Matthias Vijverman bei der Show Watch?! seine rasselnden, dampfenden und klappernden Erfindungen vor. Bei diesem Erlebnis vergeht die Zeit wie im Flug.

Auch ein Garant für Spaß wird das Karussell der Fundgegenstände sein. Auf aus Schrott gebauten Sitzen können die jüngsten Maker gemütlich im Kreis fahren und dabei den Blick über die Messe schweifen lassen. Wer jetzt glaubt, dass das ein guter Ort ist, um die Kleinen kurz abzusetzen und zu verschnaufen, der irrt: Das Karrusell der Fundgegenstände läuft nur mit der Muskelkraft von Makern.

Ein Maker-Faire-Urgestein – der R2 Builders Club – ist wieder auf der Messe vertreten. Egal ob der hellen oder der dunklen Seite der Macht verpflichtet – alle Droiden müssen zuerst einmal gebaut werden. Und wie das geht, kann man vor Ort erfahren, vorausgesetzt, man entlockt es den Mitgliedern des R2 Builders Clubs.

Dieses Jahr gibt es natürlich auch wieder die nervenaufreibenden Roboterkämpfe im Stil von Robot Wars und BattleBots. Dei diesen Kämpfen treten selbstgebaute Mini-Kampfmaschinen gegeneinander an, um am Ende als Last-Bot-Standing aus dem Turnier hervorzugehen. Im Gegensatz zum Vorjahr finden diese Kämpfe nicht mehr in den Hallen, sondern im Außenbereich in einer abgesperrten Arena statt.

Im Außenbereich befinden sich selbstverständlich auch diverse Möglichkeiten, um die Kraftreserven an einem Essensstand wieder



Damit kann man in das Making einsteigen: Der Makey:Lab.

aufzuladen. Doch das ist natürlich nicht alles. Während man sich im Freien einen Snack holt, kann man dort weitere Aussteller besuchen, die die Freiheit vor den Hallen voll auskosten.

Wie komme ich da hin?

Das ist die Frage aller Fragen. Die Maker Faire Hannover findet am 17. und 18. August im Hannover Congress Centrum statt. Tickets gibt es auf der Maker-Faire-Website. Ein reguläres Ticket für Maker ab 16 Jahre kostet 19,80 €. Für Schüler, Studenten, Rentner und Schwerbehinderte gibt es ein ermäßigtes Ticket für 15,80 €. Für Maker-Familien gibt es ein Gruppenticket für 30 €. Wer abends noch schnell einen Maker-Fix braucht, der kann Samstag von 17 - 18 Uhr und Sonntag von 15 - 17 Uhr für 10 € über die Messe wandeln. Karten können natürlich auch an der Tageskasse gekauft werden. Achtung: Vor Ort ist keine Barzahlung möglich. —*das*

Das Komplettpaket für Einsteiger. Hardware und Erklärung.

Frei rollende Droiden auf der Messe.

Nicht nur die Technik muss hart arbeiten. Bei Watch?! muss auch Maker Matthias Vijverman alles geben.

Das KI-Orakel

Analoges Kartenlegen war gestern. Heute sieht man digital in die Zukunft, mit Raspi, Beamer und ChatGPT. Für den stilechten Look sorgt eine Glaskugel, in der die Video-Vorhersagen wie durch Zauberhand erscheinen.

von Dirk Wahl

Wenn der Wahrsager des Vertrauens gerade mal keine Zeit hat, kann man sich selbst einfach einen Ersatz bauen, der garantiert genauso professionelle Vorhersagen trifft. Dafür braucht man gar nicht viel: einen Raspberry Pi, künstliche Intelligenz und einen Mini-Beamer, der die Zukunft in Videoform von innen auf eine Glaskugel projiziert. Die Interaktion funktioniert per Sprache mithilfe eines USB-Mikrofons, und ein LED-Ring signalisiert, ob das Orakel gerade empfangsbereit ist. Wie man die einzelnen Komponenten hard- und softwareseitig verbindet und konfiguriert, erkläre ich in diesem Artikel Schritt für Schritt. Die Python-Programme zum Projekt kann man im GitHub-Repository des Projekts herunterladen, das in der Kurzinfo verlinkt ist.

Kurzinfo

- **Ein eigenes Orakel mit Raspberry Pi bauen**
- **Sprachbefehle mit OpenAI transkribieren, übersetzen und kürzen**
- **Videos mit einem Mini-Beamer in eine Glaskugel projizieren**

Checkliste

Zeitaufwand:
ca. 3 Stunden

Kosten:
ca. 150 Euro (plus evtl. Kosten für OpenAI)

Material

- **Raspberry Pi 4 B**
- **MicroSD-Karte** mit 32 GB
- **Mini-Projektor YG300 + USB-Stromkabel**
- **HDMI-Kabel mit Microstecker** für den RPi4
- **USB-Powerbank**
- **USB-Mikrofon**
- **Hohle Glaskugel,** offen und getrübt, ca. 20 cm Durchmesser
- **Lüftungsrohr** aus dem Baumarkt, Durchmesser wie Glaskugelöffnung
- **Fresnel-Linse** 100 × 100 mm, 50 mm Brennweite
- **5-V-LED-Ring mit 24 LEDs,** passend zur Öffnung der Glaskugel, WS2812B

Werkzeug

- **Laubsäge**
- **Kleber**
- **Kreditkarte** für OpenAI-Guthaben

Mehr zum Thema

- Dirk Wahl, KI-Sprachassistent mit eigenem sprechenden Avatar, Make 7/23, S. 80
- Dirk Wahl, Staumeldung durch KI-Bilder, Make 4/23, S. 76
- Dirk Wahl, KI auf der Schreibmaschine, Make 2/23, S. 28
- YouTube-Video, KI-Orakel in Aktion

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

Das Funktionsprinzip

Den Kern des Orakels bildet ein Python-Programm, das auf einkommende Befehle wartet und den gesamten Ablauf steuert. Um das Orakel aufzuwecken, nennt man zunächst ein Wake Word, z. B. „Orakel“, und kann danach seine Frage stellen. Diese zeichnet der Raspi dann als Audiodatei auf und sendet sie zum Transkribieren an OpenAI. Anschließend kondensiert ChatGPT den „abgetippten“ Text mit einem Befehl auf zwei Begriffe (siehe Listing „ChatGPT-Befehl aus Glaskugel.py“). Mit diesen sucht das Python-Programm dann auf der Plattform Pexels nach einem passenden Video und gibt es letztlich in der Glaskugel aus (Bild 1).

Wake Word erstellen

Ein Wake Word für sprachgesteuerte Projekte einzurichten ist sinnvoll, damit das Mikrofon nicht ungewollt alles Gehörte an OpenAI schickt. Man kennt das Prinzip von gängigen Sprachassistenten im Smart Home, die erst reagieren, wenn man sie beim Namen nennt. Für eigene Projekte kann man ein Wake Word etwa beim Anbieter Picovoice erzeugen (siehe Link in der Kurzinfo). Dabei wird aus einer Audiodatei ein KI-Modell trainiert, das man danach für eigene Anwendungen verwenden kann, z. B. mit Python.

Nachdem man sich bei Picovoice registriert hat, kann man im Abschnitt „Start Building“ durch einen Klick auf die Schaltfläche „Porcupine Wake Word“ mithilfe eines ausgedachten Begriffs ein Wake Word erstellen (Bild 2). Als Plattform wählt man Raspberry Pi aus und kann dann eine ZIP-Datei herunter-

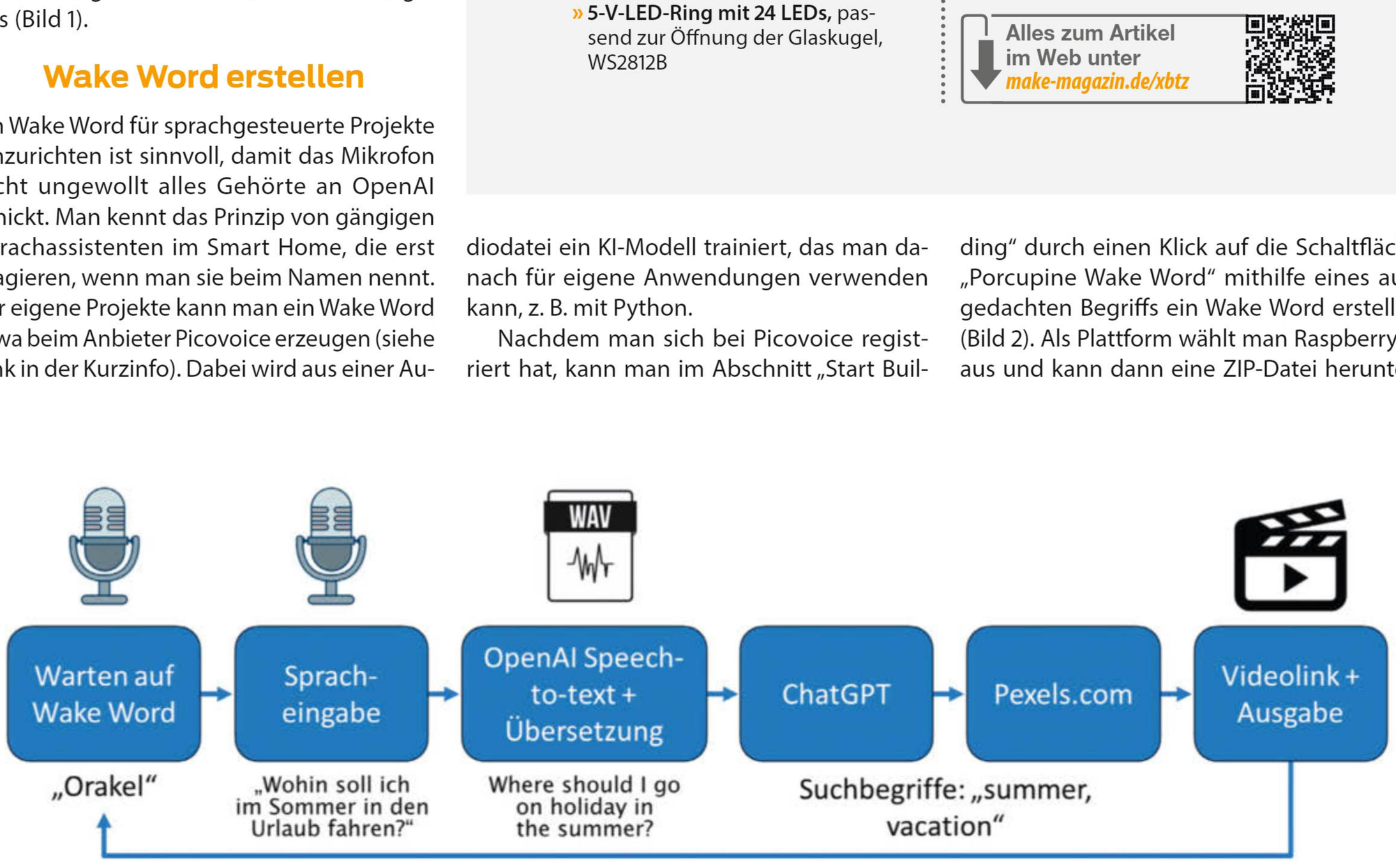


Bild 1: So funktioniert der Ablauf des KI-Orakels.

ChatGPT-Befehl aus Glaskugel.py

```
"You are an AI agent given the sole task of summarizing an audio transcript to
keywords that can be used for a search field. The transcript can either be of
poor or good quality. The transcript generated from the audio file is given below.
If the transcript is of poor quality or some words have been poorly transcribed,
make sure to guess what the word is supposed to be and return all the important
information from the transcript in exactly 2 english nouns. If you do not
understand anything at all, just generate 2 english nouns. IMPORTANT: No
explanation needed, only the two nouns and nothing else."
```

laden, die die benötigte PPN-Datei (das trainierte Modell) enthält. Da ich das deutsche Wake Word „Orakel" verwende, musste ich zusätzlich noch die Datei porcupine_params_de.pv für die Spracherkennung herunterladen (siehe Link in der Kurzinfo). Außerdem benötigt man noch den API-Key von der Picovoice-Startseite.

Bei OpenAI anmelden

Um mit dem Speech-to-Text-Modul von OpenAI und ChatGPT zu kommunizieren – sei es über die Weboberfläche oder per API –, ist eine Registrierung notwendig. Dazu klickt man ganz unten auf der OpenAI-Website auf API-Login. Nachdem man sich einen Account erstellt hat, landet man auf dem Dashboard. Dort erzeugt man in der Kategorie „API Keys" mit einem Klick auf „Create new secret key" einen API-Schlüssel für die Kommunikation zwischen Raspberry Pi und OpenAI (Bild 3). Falls noch nicht geschehen, muss man sich dafür zusätzlich mit seiner Mobilfunknummer registrieren. Den generierten API-Schlüssel muss man später in dem Python-Code einsetzen (dazu gleich mehr).

Normalerweise erhält man als neuer OpenAI-Nutzer ein wenig Guthaben geschenkt, sobald man den Account mit seiner Telefonnummer verknüpft hat, um auch die kostenpflichtigen KI-Dienste auszuprobieren. Ist dieses verbraucht oder hat man keine Credits erhalten (das kann OpenAI nämlich jederzeit ändern), muss man mindestens einen Betrag von 5 US-Dollar als Guthaben aufladen. Dazu klickt man im Dashboard rechts oben auf sein Profilkürzel, wählt in dem Menü „Your Profile" aus und anschließend in der linken Menüleiste „Billing". Zum Aufladen der Credits benötigt man eine Kreditkarte.

Wie viel man von seinem Budget für API-Aufrufe verbraucht hat, ist über das Dashboard bei „Usage" einsehbar. Zwei bis drei API-Aufrufe mit dem Orakel – also für Speech-to-Text und ChatGPT – kosten etwa 1 Cent.

Videoplattform Pexels

Eigentlich wollte ich die Videos in der Glaskugel durch eine Video-KI wie Sora oder Pika erstellen lassen, aber entweder fehlt noch eine API-Anbindung oder die Videoerstellung dauert zu lange. Daher habe ich mich für fertige Videos der Berliner Plattform Pexels entschieden (siehe Link in der Kurzinfo). Diese kann man lizenzfrei nutzen und zudem mithilfe der REST-API-Schnittstelle Videolinks generieren, die sich in ein Programm einbinden lassen.

Nachdem man sich bei Pexels registriert hat, gibt es im Profilmenü den Punkt „Bild- und Video-API". Um einen API-Schlüssel zu bekommen, muss man vorher noch ein paar Angaben zum Projekt eingeben (Bild 4).

Die Pexels-API stellt mit den von Chat-GPT erstellten Suchwörtern Videos in unterschiedlicher Qualität zur Verfügung – wenn auch ohne Audiospur. Ich nutze für die Glaskugel jeweils die SD-Version, um möglichst schnell eine ruckelfreie Darstellung zu bekommen.

Raspberry Pi vorbereiten

Jetzt geht es auf den Raspberry Pi: Zuerst einmal spielt man mit dem Pi Imager (siehe Link in Kurzinfo) das Raspberry Pi OS (64-Bit) auf eine SD-Karte. Unter „Einstellungen bearbei-

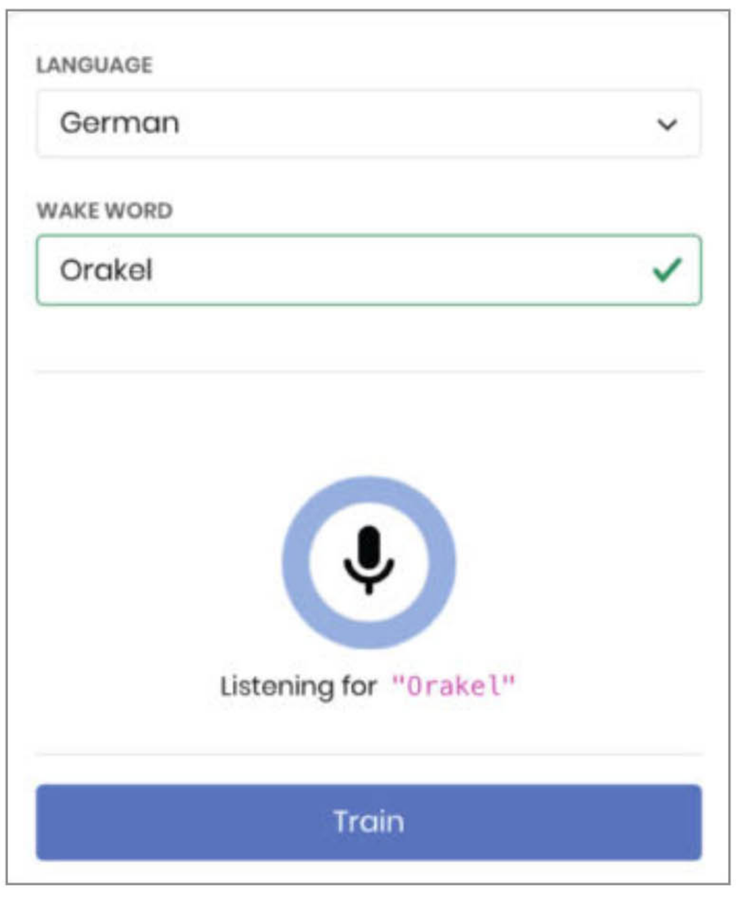


Bild 2: In Picovoice trainiert man das Wake Word.

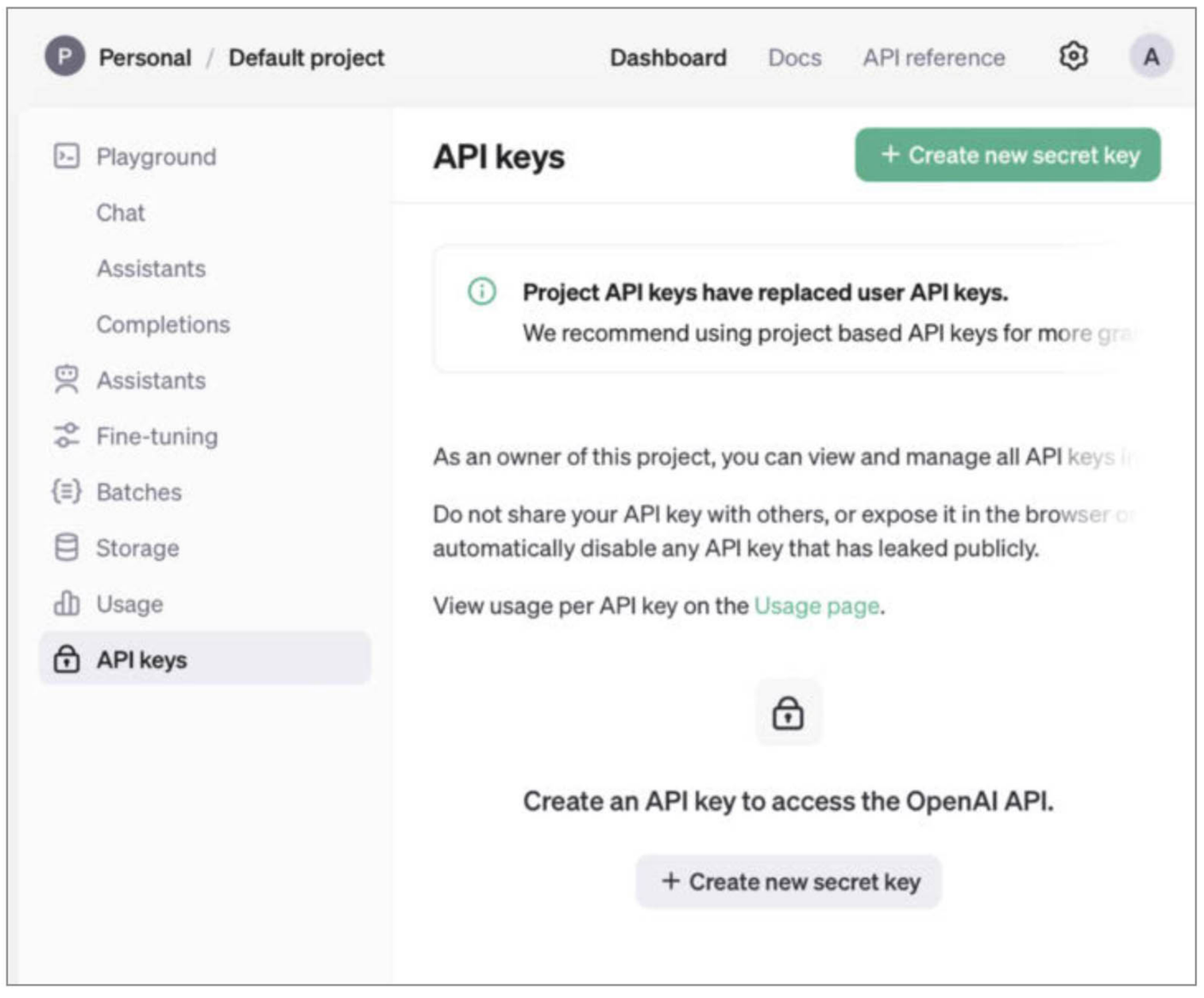


Bild 3: Der API-Key von OpenAI gilt für das Transkribieren und ChatGPT.

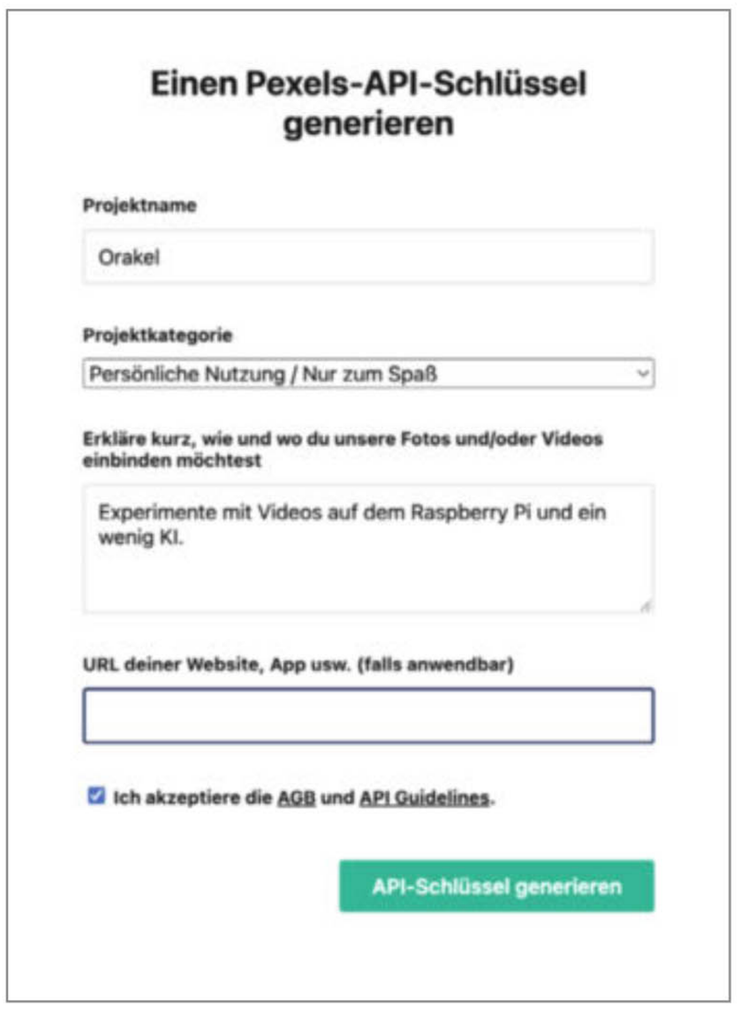


Bild 4: Pexels möchte wissen, wofür man die API verwendet.

ten“ im Pi Imager kann man auch gleich die WLAN-Zugangsdaten eingeben. Nach dem Flashen wirft man die Karte aus, steckt sie in den Raspberry Pi und verbindet diesen mit Monitor, Tastatur, Maus und Netzteil.

Nach der initialen Konfiguration des Pi aktualisiert man alle Programme, indem man ein Terminalfenster öffnet und nacheinander die folgenden Befehle ausführt:

```
sudo apt update
sudo apt full-upgrade
sudo reboot
```

Als Nächstes ist es sinnvoll, VNC (Virtual Network Computing) zu aktivieren. Damit kann man das Orakel später vom PC aus fernsteuern und die auf dem PC gespeicherten Dateien auf den Raspberry Pi übertragen. Dafür muss man (in der aktuellen OS-Version) das System zunächst auf X11 umstellen: Dazu gibt man im Terminal den Befehl

```
sudo raspi-config
```

ein, navigiert zu„Advanced Options/Wayland/X11“ und startet den Raspi nach dem Aktivieren neu. Anschließend öffnet man noch einmal `raspi-config` und aktiviert den VNC-Server über „Interface Options/VNC“. Im Tray erscheint daraufhin neben dem Bluetooth-Symbol ein VNC-Icon. Auf seinen PC lädt man sich danach noch den VNC Viewer von RealVNC herunter und installiert ihn (siehe Link in der Kurzinfo).

Sobald das erledigt ist, muss man auf dem Raspberry Pi im Terminal erst einmal etliche Bibliotheken installieren (siehe gleichnamiges Listing). 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.

Danach lädt man sich aus dem GitHub-Repository des Projekts die Python-Dateien auf den PC herunter und fügt zunächst die API-Keys in das Hauptprogramm Glaskugel.py ein (die Stellen sind in der Datei gekennzeichnet). Falls man im Raspberry Pi Imager den Benutzernamen `pi` angepasst hat (z. B. in `orakel`), muss man ebenfalls noch die Pfade im Python-Code anpassen, etwa `/home/pi/Desktop/` in `/home/orakel/Desktop/`. Danach kopiert man Glaskugel.py, die vier LED-Programme, die PPN-Datei mit dem Wake Word sowie den Porcupine-Parameter für Deutsch auf den Raspberry-Pi-Desktop, indem man auf dem PC zunächst den VNC Viewer öffnet und mit der IP-Adresse des Raspberry eine Verbindung herstellt. Die IP-Adresse des Raspi findet man einem Klick auf das VNC-Icon im Tray des Pi heraus.

Falls man den Benutzernamen und das Passwort des Raspberry Pi im Imager nicht angepasst hat, gibt man als Benutzer `pi` und als Passwort `raspberry` für die Verbindung ein.

Bibliotheken installieren

```
pip install python-vlc
pip install pvrecorder
pip install pvporcupine
pip install picovoice
pip install python-dotenv
pip install openai
pip install sounddevice
sudo apt-get install libportaudio2
sudo pip install rpi_ws281x adafruit-circuitpython-neopixel
```

Bild 5: Mit dem VNC Viewer kann man auch Dateien auf den Raspberry Pi übertragen.

Sobald man den Desktop von Raspberry Pi OS sieht, fährt man mit der Maus an den oberen Bildschirmrand des VNC Viewer und es erscheint ein verstecktes Menü. Über das Icon „Dateien übertragen“ (zwei Pfeile in einem Dateisymbol) kopiert man die Dateien vom PC auf den Desktop des Pi (Bild 5).

Damit die Glaskugel später im Wartemodus nicht alle Icons auf dem Desktop anzeigt, habe ich alle Dateien (bis auf Glaskugel.py) in einen Unterordner namens „Glaskugel“ verschoben und den Bildschirmhintergrund sowie die Taskleiste auf Schwarz eingestellt (Bild 6).

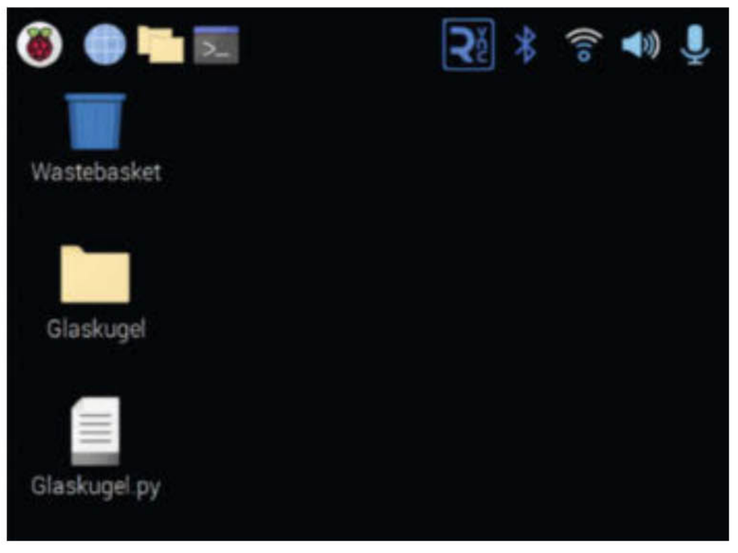


Bild 6: Den Hintergrund und die Taskleiste in Schwarz.

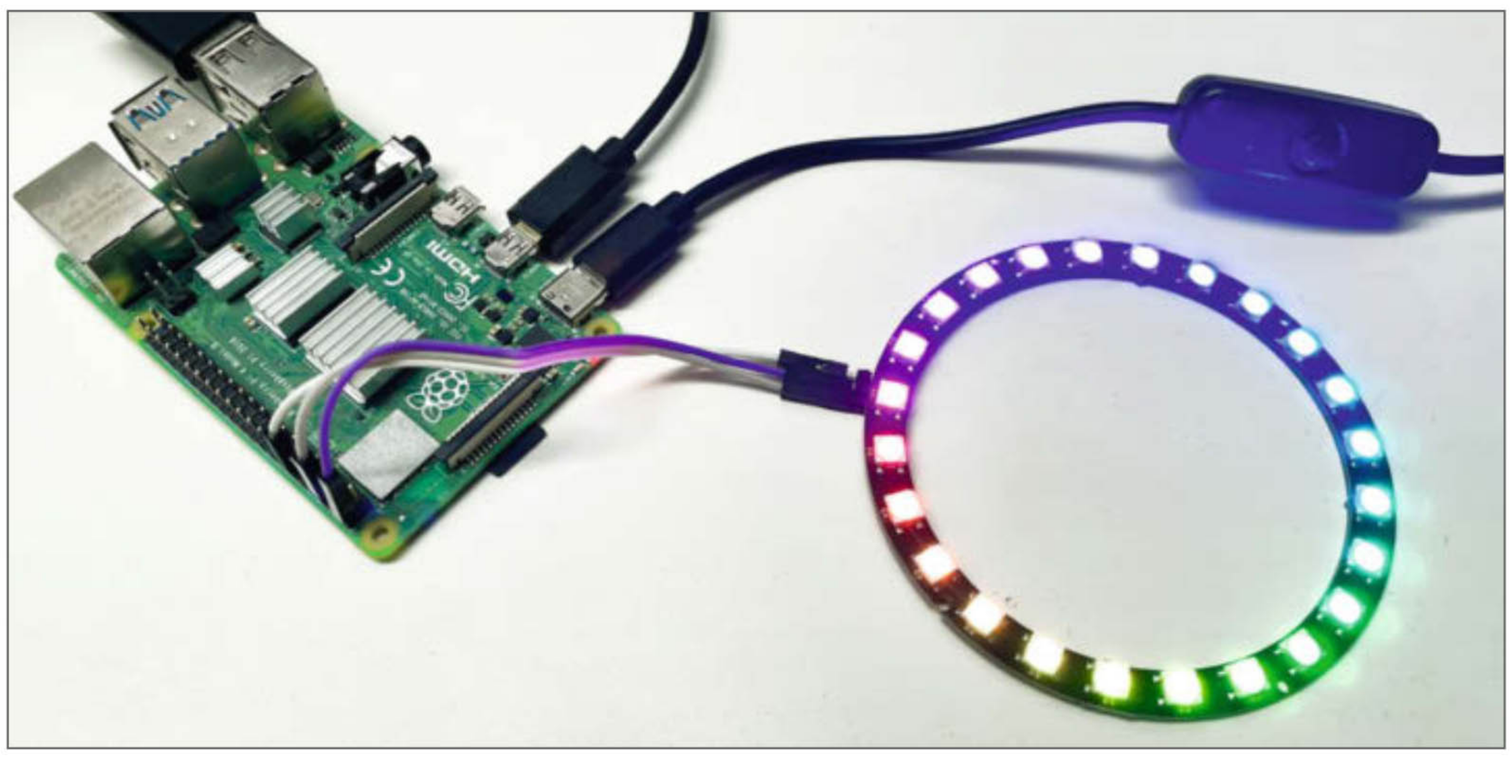


Bild 7: Die Funktion des LED-Rings kann man mit separaten Python-Programmen testen.

LED-Ring anschließen

Der WS2812B-LED-Ring ist einfach zu installieren: Man verbindet den 5-V-Pin des Rings mit dem 5-V-GPIO des Pi, GND mit GND des Pi und das Signal DI (Digital In) des Rings mit dem GPIO 18 des Pi.

Die vier Programme LED_Idle.py, LED_Orakel_Start.py, LED_WiFi.py und LED_Off.py muss man mit `sudo` starten. Deshalb sind sie aus dem Hauptprogramm Glaskugel.py ausgegliedert und werden später von dort aus mit `sudo` aufgerufen. Wenn man den Ring (Bild 7) ausprobieren will, muss man die Programme einzeln über das Terminal starten, z. B. mit

```
sudo python3 LED_Idle.py
```

Zu beachten ist, dass keine zwei LED-Programme parallel laufen dürfen, sonst kommt der Ring durcheinander. Mit

```
ps aux | grep LED
```

sieht man, ob das eventuell der Fall ist; falls ja, schießt man den Prozess mit

```
sudo pkill -f LED_Idle.py
```

ab (hier: LED_Idle.py).

Glaskugel testen

Jetzt schließt man noch das USB-Mikrofon und den Projektor über HDMI an und startet und das Python-Programm Glaskugel.py mit Thonny für einen ersten Test. Nach einer Rotphase während des WLAN-Verbindungsaufbaus zeigt der LED-Ring einen Regenbogen an. Dieser signalisiert, dass das Orakel auf das Wake Word wartet. Nachdem man es ausgesprochen hat, meldet sich das Orakel mit drei kurzen LED-Blitzen, schaltet die LEDs auf Rot und wartet auf die Frage. Hat man sie gestellt, erscheint das Ergebnis wenige Sekunden später als Video über den Projektor. Sobald das Video fertig gespielt hat, kann man mit dem Wake Word weitere Fragen stellen. Enthält eine Frage das Wort „Orakel“, wird das Programm beendet und der Raspberry Pi automatisch heruntergefahren.

Autostart 1

```
chmod +x /home/pi/Desktop/Glaskugel.py
sudo nano /etc/xdg/autostart/Glaskugel.desktop
```

Orakel beim Booten starten

Soll nun das Hauptprogramm direkt nach dem Booten automatisch starten, sind noch die Eingaben aus dem Listing „Autostart 1“ im Terminal notwendig. Mit dem zweiten Befehl öffnet man die Datei Glaskugel.desktop im Editor nano. In diese trägt man die Zeilen aus dem Listing „Autostart 2“ ein, speichert die Eingabe mit STRG+O und beendet den Editor mit STRG+X.

Technischer Aufbau

Da der Mini-Projektor das Bild nach oben in die Glaskugel projizieren soll, habe ich ihn im ersten Schritt aufgeschraubt und gekürzt. Die beiden Fresnel-Linsen sowie das transparente LC-Display sind im Gehäuse geblieben (Bild 8). Zwischen Spiegel und Linse habe ich das Gehäuse durchgesägt und den Spiegel in einem 45-Grad-Winkel direkt vor der Projektionslinse befestigt, sodass er nach oben zeigt (Bild 9). Darüber wurde eine zusätzliche Fresnel-Linse befestigt, auf der später die Glaskugel steht.

In Ermangelung eines 3D-Druckers habe ich das Ganze mit LEGO-Steinen gebaut. Bei den Abständen zwischen den Linsen, dem Spiegel und der Kugel musste ich etwas experimentieren, um das Bild so scharf wie möglich

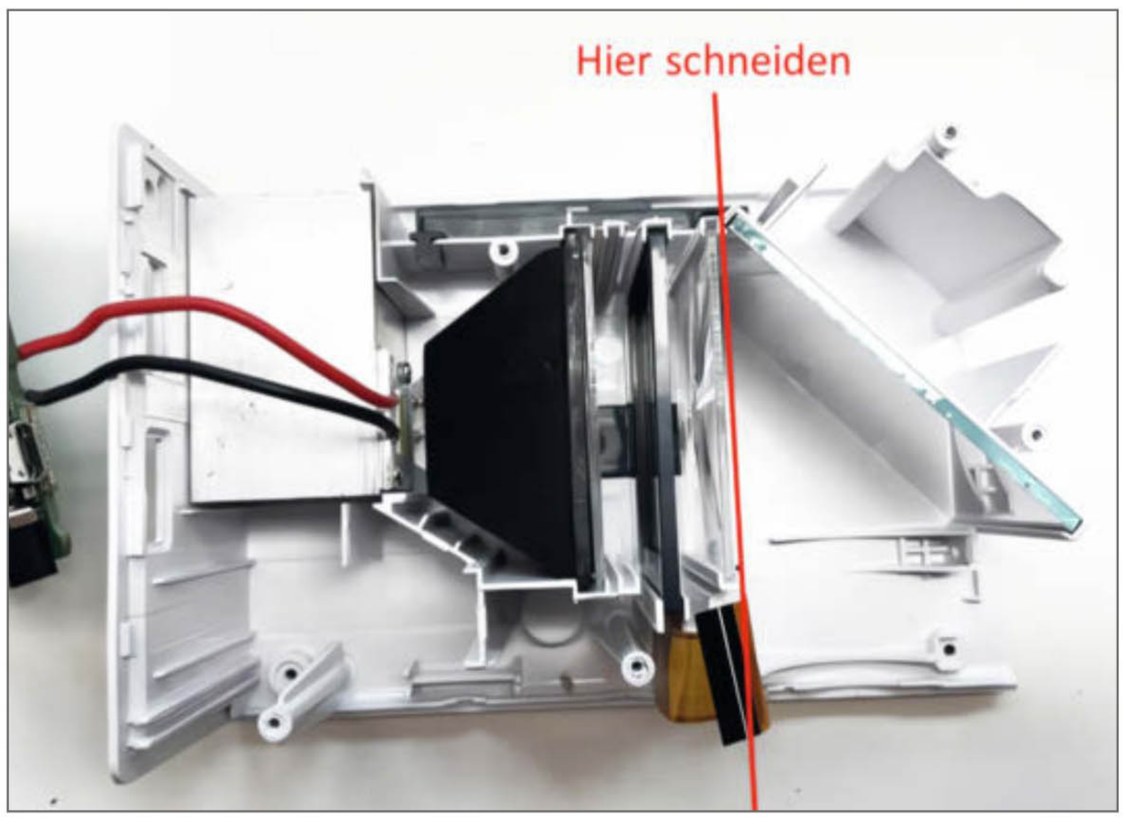


Bild 8: Das Kunststoffgehäuse des Beamers lässt sich gut mit einer Laubsäge zerteilen.

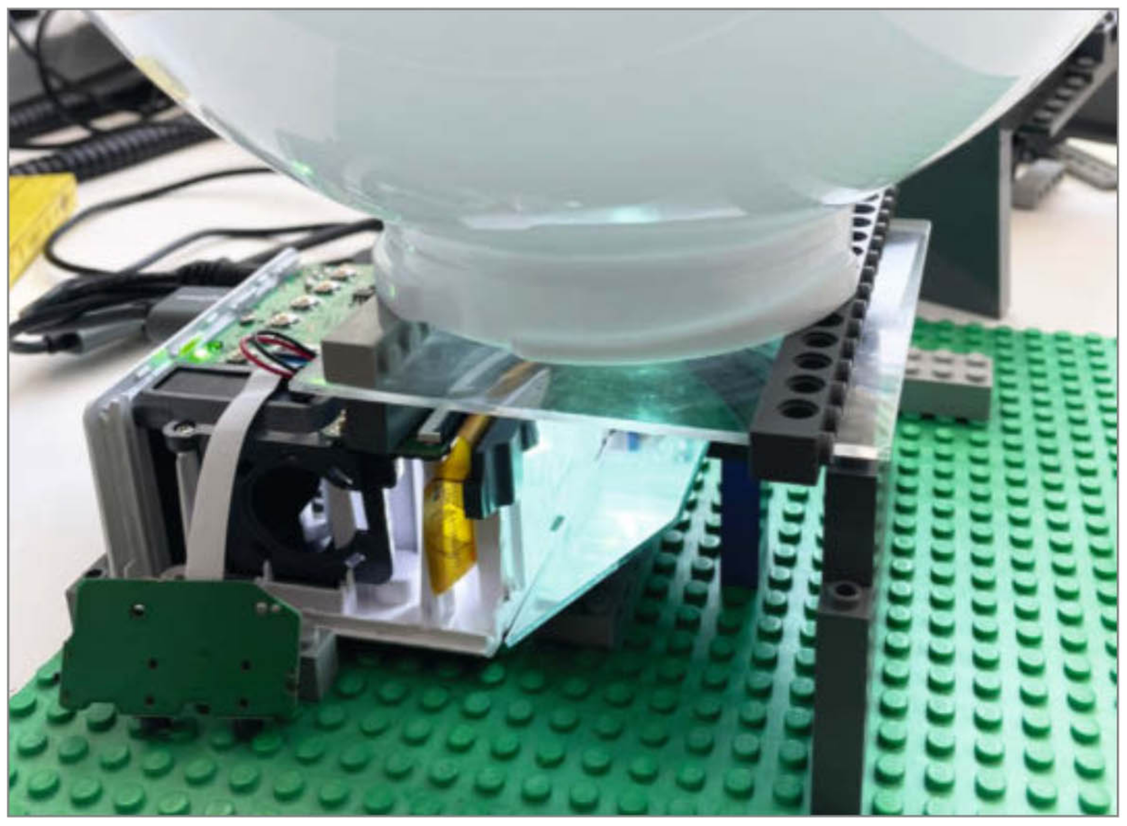


Bild 9: Der gedrehte Spiegel lenkt das Bild des Beamers jetzt nach oben.

Bild 10: Jetzt noch schnell den LED-Ring in das Lüftungsrohr kleben.

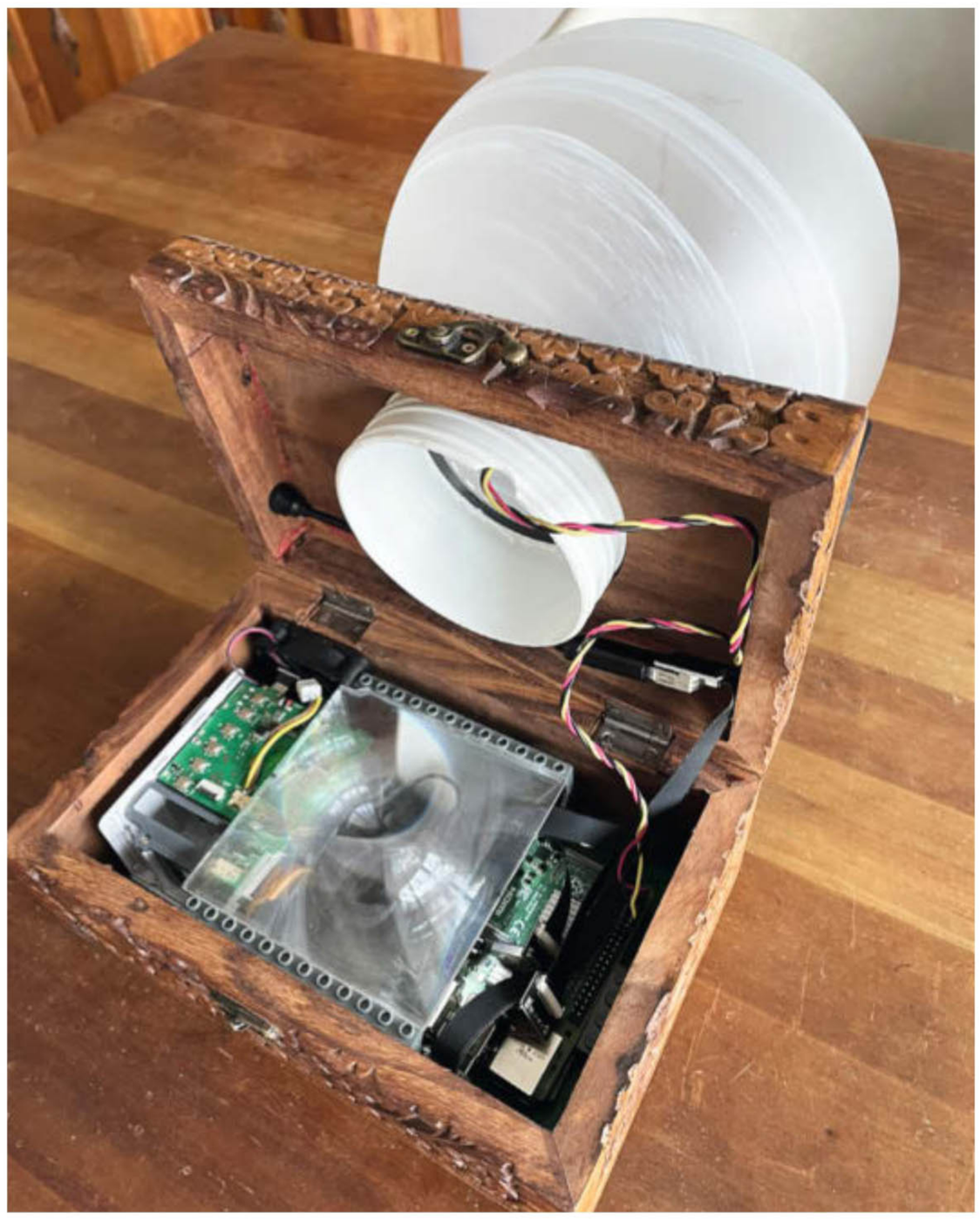


Bild 11: Stilecht für ein Orakel habe ich die Technik in einer verzierten Holzkiste verstaut.

zu bekommen. Es bleibt durch die Projektion auf die runde Oberfläche größtenteils unscharf, was aus meiner Sicht aber zu der Darstellung einer Glaskugel passt. Schaut euch das in der Kurzinfo verlinkte YouTube-Video an, ob euch die Qualität genügt.

Als Abstandhalter zwischen Fresnel-Linse und Glaskugel habe ich ein weißes Lüftungsrohr aus dem Baumarkt verwendet und auf die richtige Länge geschnitten, nachdem ich mit der Auflösung zufrieden war. Dann musste ich nur noch den LED-Ring einkleben (Bild 10).

Autostart 2

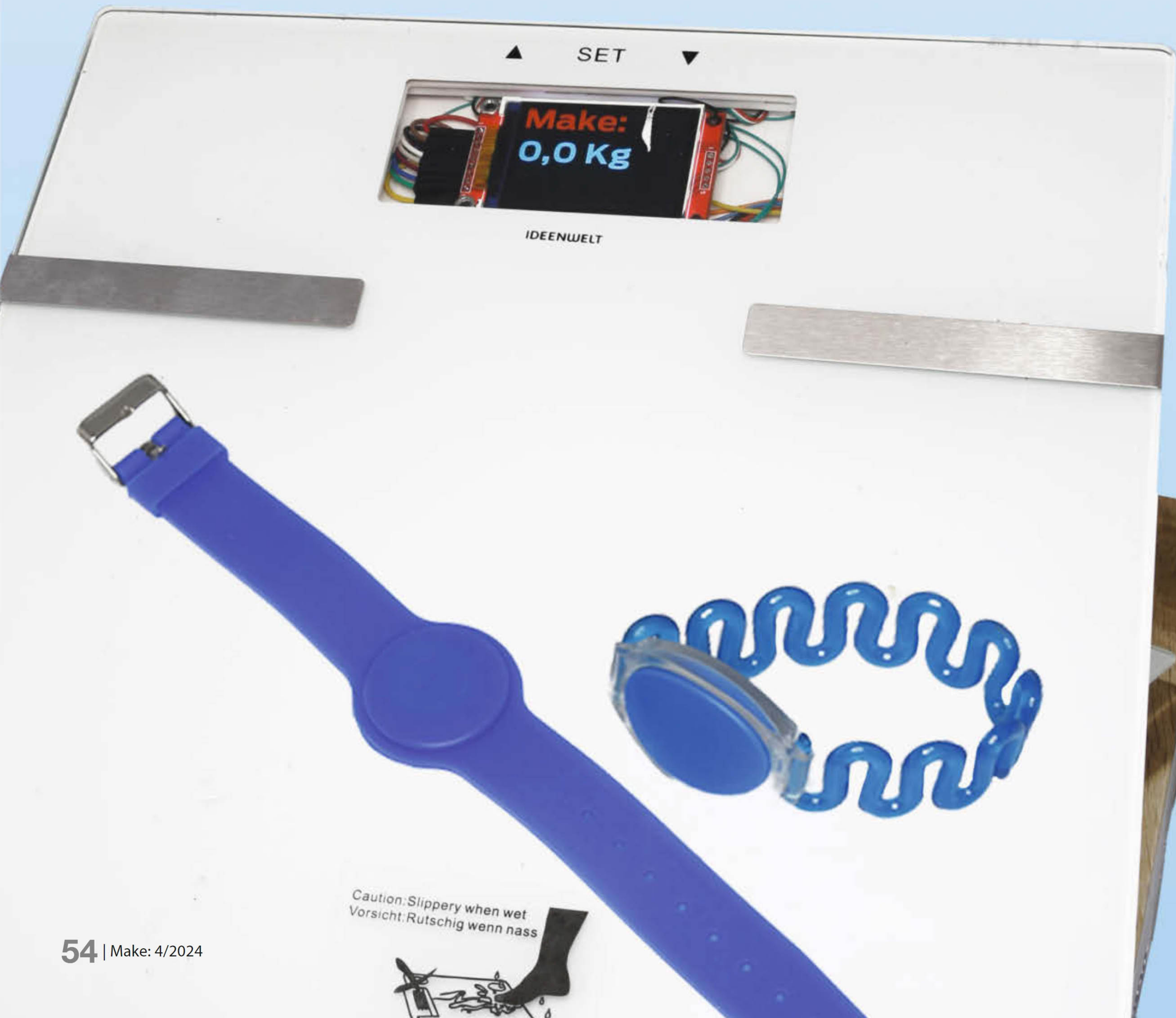
```
[Desktop Entry]
Name=Glaskugel
Exec=/usr/bin/python3 /home/pi/Desktop/Glaskugel.py
```

In meinem Gehäuse – einer kleinen Holzbox – geht es sehr eng zu (Bild 11). Deshalb sind die Komponenten mit Flachkabeln und Winkelsteckern verbunden. Das Mikrofon ist oben seitlich im Gehäusedeckel versteckt. Da der Originallüfter des Mini-Projektors recht laut war, habe ich diesen durch einen leiseren Lüfter ersetzt. Nun noch eine Powerbank anschließen, den Raum etwas abdunkeln, und das Orakel kann nach der Zukunft befragt werden. —*akf*

Die diskrete Smarthome-Waage

Weiter geht's mit dem Umbau der Rossmann-Waage. Jetzt lernt sie, die Gewichte ihrer Kunden getrennt zu erfassen und anzuzeigen. Ein Funk-Chip für jeden Benutzer macht das möglich. Sie und ich lernen bei diesem in Entwicklung befindlichen Projekt eine Menge über MQTT, RFID und Schalter in Home Assistant.

von Heinz Behling



Wiegen und das Gewicht anzeigen kann die gehackte Drogeriemarkt-Waage ja bereits seit dem ersten Teil dieses Projekts. Jetzt kommt die angekündigte Erweiterung, um die Gewichtsdaten mehrerer Personen getrennt zu erfassen mit dem späteren Ziel, sie ausschließlich dem oder der jeweils Betroffenen anzuzeigen und die Entwicklung im Laufe der Zeit grafisch darzustellen.

Für die Erkennung der einzelnen Personen gibt es mehrere Methoden. Da wäre zum Beispiel eine Kamera mit Gesichtserkennung. Doch bei dem Gedanken, dass im Badezimmer eine Kamera arbeitet, bekommen viele zu Recht Unbehagen. Deshalb habe ich diese Methode verworfen.

Die nächste Methode mit Usernamen und Passwort erfordert entweder ein Touchdisplay oder eine Tastatur an der Waage. Das erschien mir ungeeignet, denn die Bedienung mit feuchten Fingern (falls man gerade aus der Dusche kommt) könnte da schwierig werden.

Als dritte Methode kam ich auf RFID-Chips: Jeder Waagenbenutzer erhält seinen eigenen Chip, zum Beispiel in Form eines wasserdichten Armbandes (Bild 1) oder eines Folienchips, den man in die Badelatschen einkleben kann. In der Waage sitzt ein entsprechender RFID-Leser, der die Chip-ID ausliest. Da jeder RFID-Chip eine individuelle Nummer hat, kann man damit sicher die Personen identifizieren. Ich habe mich für den Leser RDM6300 entschieden, der nur etwa 6 Euro kostet und für den es in ESPHome bereits entsprechende Software-Module gibt.

Kurzinfo

- **Personenidentifikation mit RFID-Chips/-armband**
- **Datenübertragung mittels MQTT**
- **Schaltbare Stromversorgung des Chip-Lesers**

Checkliste

Zeitaufwand:
3 Stunden

Kosten:
15 - 20 Euro (zzgl. der Kosten aus Teil 1)

Material (zusätzlich zum Material aus Teil 1)

- **RFID-Leser RDM 6300**
- **pro Person je ein RFID-Chip 125 KHz oder**
- **pro Person je ein RFID-Armband 125 KHz**
- **Schaltlitze**

Werkzeug

- **Lötausrüstung**
- **Heißklebepistole**
- **3D-Drucker (nur bei externer Antenne notwendig)**

Mehr zum Thema

- Heinz Behling: Der Rossmann-Waagen-Hack Teil 1, Make 7/23, S. 70
- Stefan Draeger: Zugangskontrolle mit RFID, Make 3/22, S. 46
- Heinz Behling: Controllerboards fürs Smarthome, Make 4/23, S. 12

Alles zum Artikel im Web unter
make-magazin.de/xq2v

RFID-Leser einbauen

Vor dem Einbau des Kartenlesers müssen eventuell bereits in die Platine eingelötete Kontaktpins entfernt werden, denn damit wäre die Platine zu hoch für den Einbau in das enge Waagengehäuse.

Für die Montage der Antennenspule habe ich zwei Varianten. Die erste kommt ohne zusätzliches Gehäuse für die Spule aus, denn sie wird in die Waage links neben das Display-Fenster eingebaut. Vorteil: Man braucht keinen 3D-Drucker für das Antennengehäuse und kein Kabel von der Waage zur Antenne. Nachteil: Die Handhabung später ist etwas mühsamer, da man den RFID-Chip auf die Waage legen muss, während man sich wiegt.

Daher gibt es auch die zweite Variante: Hier sitzt die Antennenspule in einem kleinen Gehäuse, das über ein etwa 1,5 m langes Kabel mit der Waage verbunden ist. Vorteil: Man kann zum Beispiel sein RFID-Armband am Arm lassen, an die Spule halten und gleichzeitig auf die Waage steigen. Aber Nachteile gibt es auch: Waage und Antenne sind fest mit einem Kabel verbunden. Einzelheiten zur dieser Version mit externer Antenne finden Sie im Textkasten „Variante 2“.

Im Prinzip könnte man den Chip-Leser auch mit einem eigenen Mikrocontroller ausstatten und dann die Kommunikation über WLAN vornehmen. Vielleicht nehme ich diese Änderung auch noch später einmal in Angriff. Aber wie bereits im ersten Teil dieses Projekts erwähnt, ist dies ein noch in Entwicklung befindliches Projekt. Sie können sich aber gerne daran beteiligen. Dann könnte demnächst ein Artikel mit Ihrem Namen als Autor hier stehen …

Zunächst zu Variante 1: Die Antennenspule wird am nicht mehr benötigten Batteriefach befestigt. Dazu müssen nach Abklipsen des Kunststoffunterteils der Waage die Batteriekontakte entfernt werden und mittels Dremel oder einem scharfen Messer zur Spule passende Nuten in den Kunststoff geschnitten werden. Die Spule muss hinterher so sitzen, dass sie mit der Oberkante des Plastikgehäuses auf einer Ebene liegt (Bild 2).

Die Spule wird sparsam mit Heißkleber befestigt. Bitte darauf achten, dass auch der Kleber nicht über die Gehäusekanten hinausragt. Das würde den späteren Zusammenbau erschweren. Danach verbindet man die Leser-Platine mit dem Mikrocontroller-Board und der Antennenspule entsprechend dem Schaltplan (Bild 3).

Die Kontaktbelegung der Leser-Platine zeigt Bild 4. Der Leser ist zwar für 5V-Versorgungsspannung ausgelegt. Die stehen auf dem Mikrocontroller aber nicht zur Verfügung. Er arbeitet aber auch problemlos mit 3,3 V, die an einem Pin des Controllerboards bereitstehen. Jedoch hängt da bereits das Display dran. In meinen Versuchen führte der zusätzliche Anschluss des RFID-Lesers dazu, dass die Spannung dort auf etwa 3,0 V zusammenbrach. Der Spannungsregler auf dem Board kann offenbar nicht gleichzeitig Prozessor, Display und Leser versorgen. Daher habe ich den Spannungseingang des Lesers direkt mit dem Pluspol des Akkus verbunden. Dort liegen immerhin mindestens 3,7 V an. Damit der Stromspeicher aber nicht durch permanente Stromentnahme leer gesaugt wird, habe ich den Masseanschluss des Lesers über Port 5 des Controllerboards

Bild 1: In solchen Armbändern sitzen die RFID-Chips wassergeschützt.

geschaltet. So zieht er nur dann Strom, wenn auch die Waage eingeschaltet ist.

Auch die Leser-Platine wird mit Heißkleber in der Waage befestigt (Bild 5), ebenso die Kabel. Dabei darauf achten, dass die Kabel später nicht vom Gehäuse eingeklemmt werden oder durchs Displayfenster laufen.

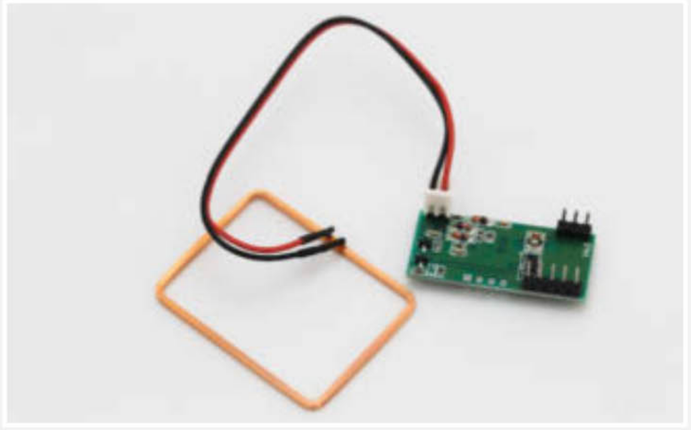
Die Waage kann nun wieder zusammengebaut werden. Allerdings muss am Wägezellenhalter der Glasoberfläche links vom Displayfenster die Rastzunge herausgebrochen werden, denn sie würde mit der Antennenspule kollidieren. Keine Angst, die Waage hält auch ohne sie zusammen.

Details zum RFID-Leser

Auf dem Markt tummeln sich eine ganze Reihe von RFID-Chip-Lesern. Der wichtigste Unterschied ist die von den darauf befindlichen Sendern benutzte Frequenz. Zur Erläuterung: Die RFID-Leser senden über ihre Antennen Funkwellen aus, die von der Chip-Antenne empfangen werden. Diese Empfangsenergie versorgt den Chip mit Strom, sodass er wiederum seine Daten an den Leser zurücksenden kann.

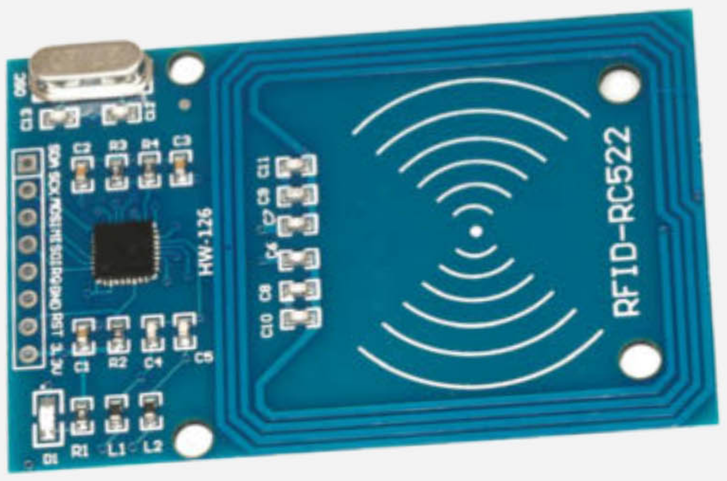
Das erfordert, dass Leser und Chip mit derselben Frequenz arbeiten. Weit verbreitet sind 13,5 MHz und 125 kHz. Beim Kauf zusätzlicher Chips (zum Beispiel als Armbänder, Klebeetiketten oder Schlüsselanhänger) müssen die zum Leser passen.

In diesem Projekt wird ein 125-kHz-Leser benutzt. Der Grund liegt in der meist separaten Antenne in Form einer Drahtspule, die sich in der Waage einfacher befestigen lässt. Da er die Daten per serieller Schnittstelle an den Mikrocontroller überträgt, sind zur Verdrahtung lediglich drei Adern (GND, 5V, TX) notwendig.



Diesen RFID-Leser (Typ RDM6300) verwende ich in der Waage.

Leser im 13-MHz-Bereich haben die Antenne meist auf der Platine in Form einer Leiterbahnschleife aufgedruckt. Diese Platinen sind recht groß und daher für die Montage in der Waage weniger geeignet. Oft erfolgt die Datenübertragung per SPI-Schnittstelle, was den Verdrahtungsaufwand deutlich erhöht.



Ein 13-MHz-Chip-Leser ist meist zu groß.

Test-Firmware flashen

Nun wird es Zeit für einen Test. Dazu muss die bisherige Firmware erweitert werden. Sie erinnern sich? Sie finden die Firmware in ESPHome (Bild 6).

Mit einem Klick auf „Edit“ gelangen Sie in den YAML-Quellcode. Den müssen Sie nun an zwei Stellen ergänzen. Zunächst die Zeilen zur Aktivierung der seriellen Schnittstelle und der Software-Bibliothek für den RDM6300 (Bild 7).

Fügen Sie sie vor dem Abschnitt `sensors:` ein.

Damit der RFID-Leser aber auch mit Strom versorgt wird, müssen wir dafür sorgen, dass Pin 5 auf Ground schaltet (darüber bekommt er ja seine Masseleitung). Dies geschieht mit einer Ergänzung im Abschnitt `switch:`, in dem wir ja schon die Stromversorgung des Wägezellen-Messverstärkers HX711 schalten (untere vier Zeilen in Bild 8).

Anschließend klicken Sie auf „Install“ und „Wirelessly“. Die Firmware wird kompiliert. Vergessen Sie nicht, die Waage einzuschalten (beide Metallstreifen auf der Glasoberfläche berühren oder den mittleren Schalter an der Rückseite drücken), damit sie die neue Firmware auch empfangen kann. Das Kompilieren kann einige Minuten dauern (Bild 9).

Nach dem erfolgreichen Kompilieren und Übertragen der Firmware startet die LOG-Ausgabe der Waage. Halten Sie nun einen Ihrer RFID-Chips auf die Glasfläche links neben dem Displayfenster. Im Log erscheint daraufhin ein Eintrag mit der Nummer des Chips (Bild 10).

Im Beispiel ist es 12113815. Notieren Sie sich die Nummern aller Chips, die Sie später mit der Waage verwenden möchten.

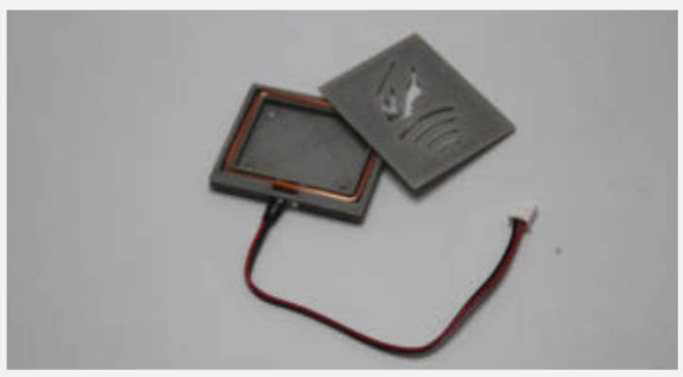
Falls dieser Test fehlschlägt, liegt ein Fehler in der Verdrahtung vor. Kontrollieren Sie alle Anschlüsse. Vergessen Sie dabei auch die Anschlüsse der Antennenspule nicht.

Variante 2

Für diesen Fall brauchen wir ein zweiadriges Verbindungskabel (Zwillingslitze), das wir vor dem Einbau der Leserplatine an deren Antennenanschluss löten. Das andere Ende wird mit der Antennenspule verbunden. Wenn Sie das über eine Steckverbindung machen, können Sie später die Waage auch bewegen (zum Beispiel beim Putzen). Übrigens: Trotz der relativ hohen Frequenz von 125 kHz (das entspricht dem Langwellenradio) werden keine höheren Ansprüche an das Verbindungskabel gestellt. Selbst einfache Schaltlitze, Klingeldraht oder Ähnliches reichen elektrisch gesehen völlig aus. Sie können also rein nach optischen Gesichtspunkten wählen.

Ein eigenes Gehäuse habe ich nicht konstruiert, denn auf Thingiverse gibt es einige geeignete. Ich habe mich für das von Alex_195 entschieden. Den passenden Link erhalten Sie mittels Kurzinfo-Link. Das Gehäuse kann dann samt Antenne mit doppelseitigem Klebeband an der Wand etwa in Brusthöhe über der Waage befestigt werden.

Falls Sie das Kabel an die Spule löten, achten Sie darauf, es vorher durch die Gehäuseöffnung zu führen. Und sichern Sie das Kabel sowohl im Antennengehäuse als auch in der Waage mit etwas Heißkleber gegen Zug, damit Sie es nicht versehentlich abreißen.



Das 3D-Druck-Gehäuse ist nur wenige Millimeter hoch.

Nach der Fehlerkorrektur wiederholen Sie den Test.

MQTT-Broker installieren

Über MQTT wird künftig der gesamte Datenverkehr zwischen der Waage und dem Smart Home laufen. Falls Sie nicht wissen, was MQTT bedeutet, finden Sie im Kasten „MQTT, was ist das?“ eine kleine Einführung zum Thema.

Sollte der Broker in Ihrem Home Assistant bereits installiert sein, können Sie dieses Kapitel natürlich überspringen. Alle anderen müssen den Broker Mosquitto als Home-Assistant-Add-On erst noch installieren.

Beginnen wir mit dem Anlegen eines neuen Benutzers in Home Assistant, der ausschließlich für MQTT benutzt wird. Das geschieht auf der Web-Oberfläche im Menü „Einstellungen“ unter „Personen/Benutzer“. Nach einem Klick auf „+ Benutzer hinzufügen“ muss sein Anzeigename eingegeben werden, zum Beispiel „mqtt-user“. Anschließend klicken Sie ins Passwort-Feld. Der Anzeigename wird dadurch auch ins Feld Benutzername kopiert.

Geben Sie nun ein möglichst sicheres Passwort ein und bestätigen Sie es auch im darunterliegenden Feld. Damit niemand mit dieser Namen-/Passwort-Kombination von außerhalb des lokalen Netzwerks auf den Broker zugreifen kann, schalten wir noch „Nur lokalen Zugriff“ ein (Bild 11).

Danach geht es mit „Benutzerkonto anlegen“ weiter. Geschafft, der neue Benutzer erscheint in der Liste (Bild 12).

Zur Installation klicken Sie auf „Einstellungen/Add-ons“. Ein Klick auf „Add-on Store“ zeigt die verfügbaren Erweiterungen an. Unser Wunschkandidat ist der Mosquitto Broker (Bild 13), auf den wir ebenfalls klicken.

Im nächsten Fenster startet der Klick auf „Installieren“ alles Weitere. Das Kopieren des Add ons und die Installation dauern einen kurzen Moment. Dann erscheint das Fenster des neuen Add ons. Die beiden Schalter „Beim Booten starten“ und „Watchdog“ sollten eingeschaltet sein, damit der Broker auch wirklich immer zur Verfügung steht. Die automatischen Updates hingegen lassen Sie besser ausgeschaltet, damit der Broker sich nicht unerwartet zwischendurch mal für zehn Minuten zum Update verabschieden kann (Bild 14).

Mit einem Klick auf „Starten“ legt der Mosquitto auch schon los, was man an der angezeigten CPU-Auslastung sieht (Bild 15).

Jetzt ist ein Neustart fällig, und zwar ein kompletter. Dazu klicken Sie auf „Einstellungen/System/Hardware“ und auf das Schaltersymbol oben rechts. Unter den erweiterten Optionen wählen Sie dann „System neu starten“ und bestätigen die Sicherheitsfrage. Nun haben Sie einige Minuten Zeit für ein Tässchen Kaffee oder Ähnliches, bis Ihr Home Assistant komplett gestartet ist.

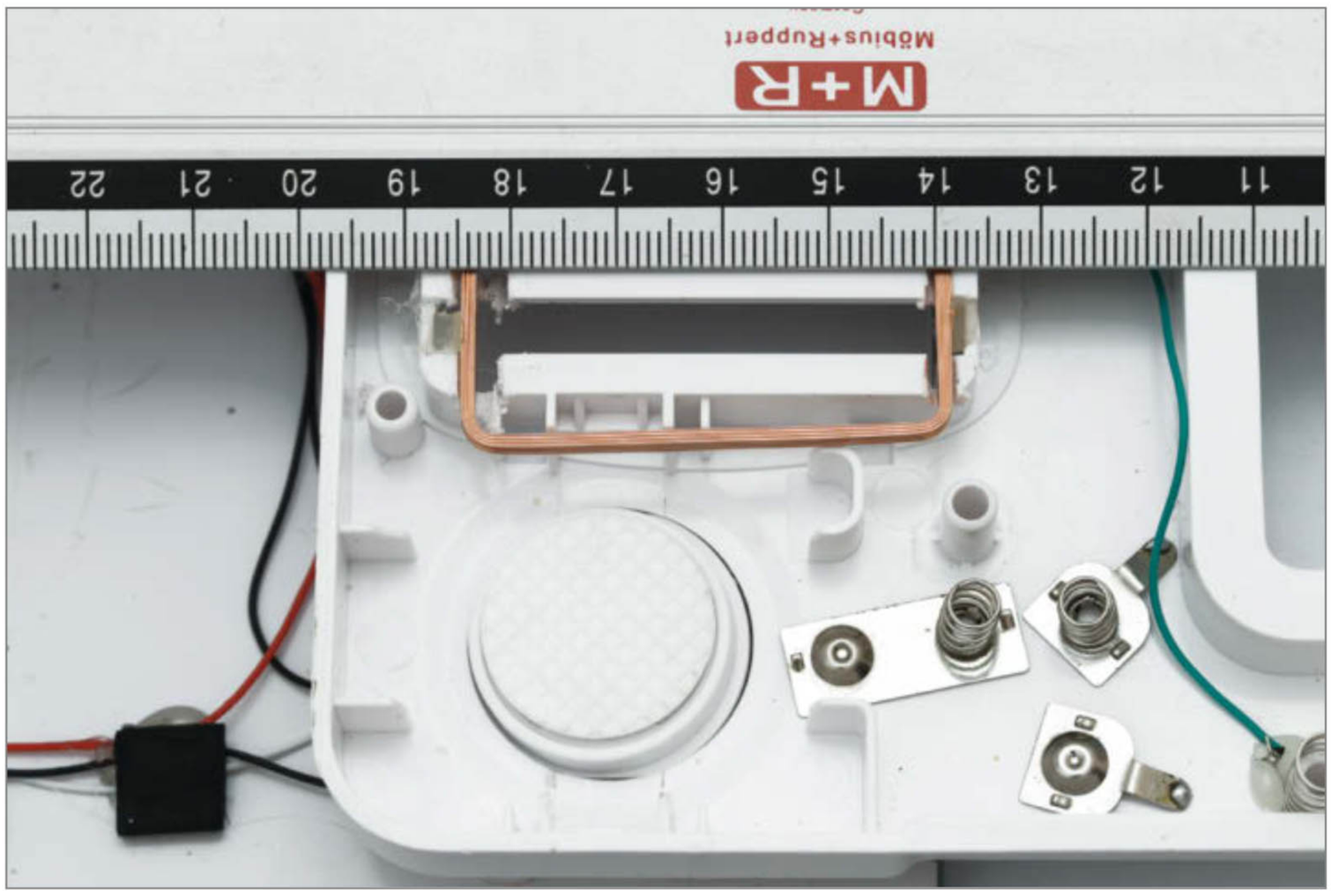


Bild 2: Mit einem Lineal sollte kontrolliert werden, dass die Oberkanten der Spule und des Gehäuses auf gleicher Höhe liegen.

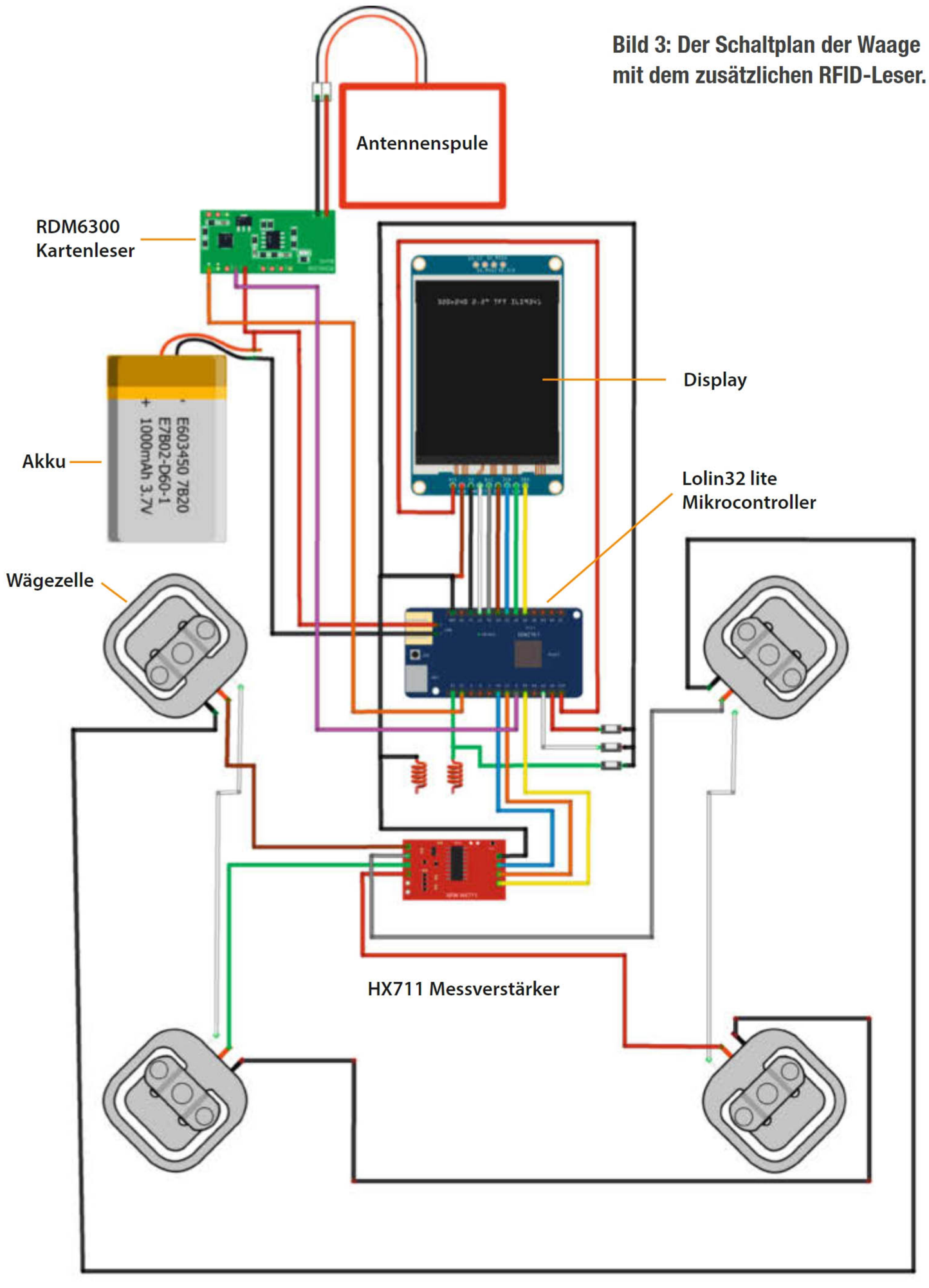


Bild 3: Der Schaltplan der Waage mit dem zusätzlichen RFID-Leser.

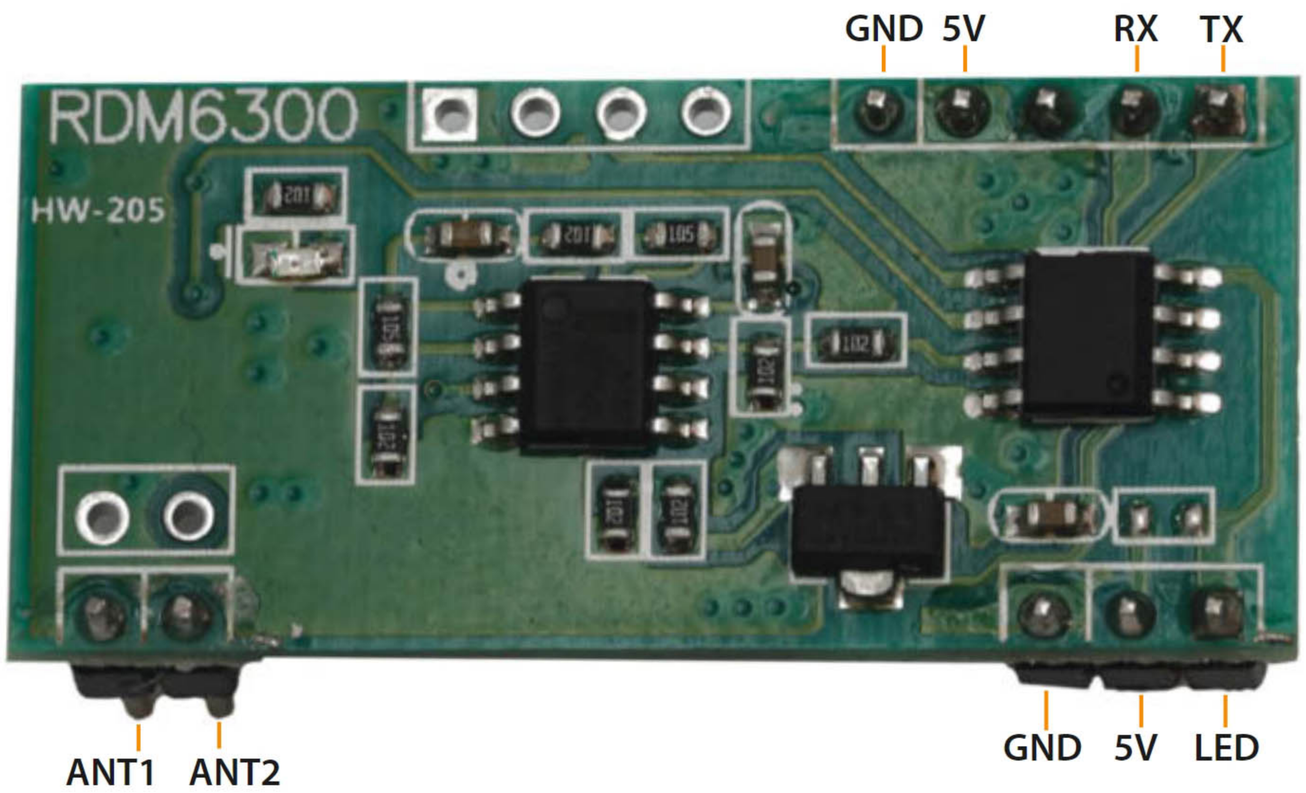


Bild 4: Außer den beiden Antennenanschlüssen werden nur noch GND, 5V und TX benutzt.

Nach dem Neustart liegt eine Benachrichtigung vor, was Sie an der kleinen orangen Marke in der linken Menüleiste merken. Schauen Sie sich die Benachrichtigung an: Eine neue Integration verlangt Ihre Aufmerksamkeit (Bild 16).

Über einen Klick auf „Check it out“ geht es weiter. Ein Klick auf „Absenden“ beendet die Konfiguration. Damit ist der Broker schmal in die Liste der zugelassenen Home-Assistant-Integrationen aufgenommen. Was noch fehlt: Wir müssen ihm noch den zuvor angelegten Benutzer vorstellen. Dazu klicken wir unter den Integrationen auf „MQTT“ (Bild 17).

Mit Klicks auf „Konfigurieren“ und „MQTT neu konfigurieren“ gelangen wir in das Fenster, in dem wir den Benutzernamen und das Passwort eingeben können (Bild 18). Ersetzen Sie hier den vorgegebenen Benutzernamen homeassistant durch den zuvor angelegten Benutzer, ebenso verfahren Sie mit dem Passwort.

Den Broker kann man hier auch gleich einmal testen. Geben Sie ins Feld „Topic zum Abonnieren“ einen Topic-Namen wie „/Test / Nachricht“ ein und klicken Sie auf „Anfangen zuzuhören“. Denselben Topic-Namen geben Sie auch unter „Ein Paket veröffentlichen“ ein. Zusätzlich schreiben Sie dort als Payload einen kleinen Text wie „Es geht!“. Sobald Sie das veröffentlichen, wird unter „Auf Topic hören“ Ihre Nachricht angezeigt (Bild 19). Klicken Sie danach auf „Weiter“ und schließlich auf „Absenden“ und „Fertig“.

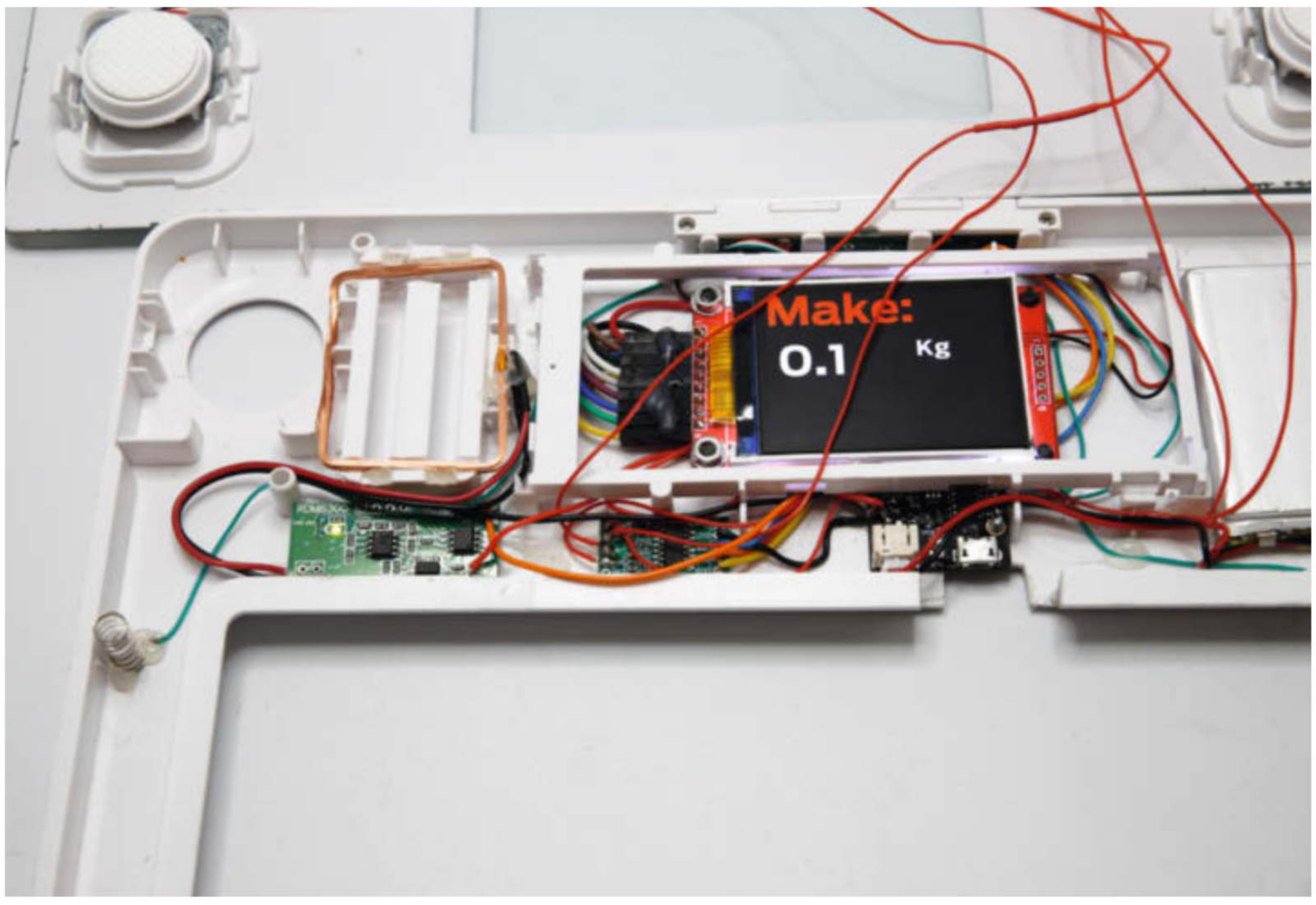


Bild 5: Die Antennenspule links neben dem Display, darunter ist die Leser-Platine.

MQTT-Firmware für die Waage

Nachdem der Broker nun einsatzbereit installiert ist, muss die Firmware der Waage so angepasst werden, dass sowohl Gewichtsdaten als auch die Nummer des RFID-Chips und damit die Identität der gewogenen Person an ihn übermittelt werden.

Als Erstes muss Kontakt zum Broker aufgenommen werden, wozu wir die Angaben für dessen IP-Adresse (in diesem Fall ist sie mit der Adresse des Home-Assistant-Servers identisch), den Benutzernamen und das Passwort brauchen. Die Zeilen habe ich direkt hinter dem `api:`-Block gesetzt (siehe Listing „MQTT-Kontakt“, der Quelltext der Firmware steht auf GitHub bereit).

Dann brauchen wir noch die Befehle zur Datenübermittlung selbst. Dazu müssen wir für jede Person, die mit einem eigenen RFID-Chip die Waage künftig benutzen soll, einen eigenen Code-Block einfügen (siehe Listing „MQTT-Datenübertragung“). In diesem Beispiel

```
uart:
  rx_pin: 15
  baud_rate: 9600

rdm6300:
  on_tag:
    then:
      - homeassistant.tag_scanned: !lambda 'return to_string(x);'
```

Bild 7: Diese Zeilen aktivieren die Schnittstelle und den Datentransport zu Home Assistant.

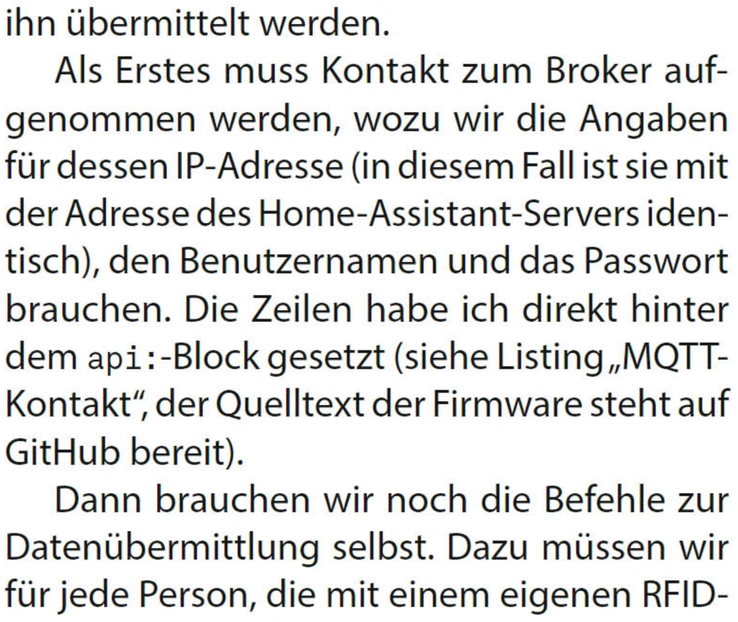


Bild 6: Der Firmware-Eintrag aus Teil 1 findet sich im Home Assistant unter ESPHome.

```
switch:
  - platform: gpio
    pin: 18
    inverted: true
    name: "HX711 Switch"

  - platform: gpio
    pin: 5
    inverted: false
    name: "RDM6300Power"
```

Bild 8: Die letzten vier Zeilen sind für die Stromversorgung des RDM6300 nötig.

geht es um Klaus und Claudia, die jeweils einen Chip mit der Nummer 12113815 (Klaus) und 11520666 (Claudia) haben. Wie Sie die Nummern ermitteln, haben Sie ja bereits im Abschnitt über die Test-Firmware erfahren.

Was machen diese Programmzeilen? Da sie im Abschnitt `binary_sensor:` stehen, richten sie jeweils binäre Sensoren ein, die sich wie Schalter verhalten, also gedrückt sein können oder nicht. Gedrückt heißt in diesem Fall, dass der Wert von `uuid:` mit dem übereinstimmt, was von der `- platform: rdm6300` geliefert wird, der Leser also den Chip mit der angegebenen Seriennummer erkannt hat.

Ist das der Fall (`on_press:`), dann werden die Anweisungen der Programmzeilen im `then:`-Block ausgeführt. Als Erstes wird ein json-Objekt mit dem Topic `/rdm6300` an den MQTT-Broker übertragen (`- mqtt.publish_json:`). Als Nutzinhalt (`payload:`) enthält es zwei Werte: den Namen der Person und deren Gewicht.

Allerdings würde es dieses Objekt immer wieder übertragen, solange der Chip am Leser anliegt. Um diese Datenflut zu verhindern, schalten wir nach der Übertragung die Spannung des RFID-Lesers mit `switch.toggle` ab.

Für jeden Waagenbenutzer müssen wir einen solchen Programmblock anlegen. Die Blöcke unterscheiden sich jeweils nur durch den Personennamen und die Chipnummer.

Schließlich brauchen wir noch eine kleine Änderung, damit wir den Schalter für die RDM6300-Spannung auch ansprechen können. Der dazugehörende `switch` braucht eine `id:`, die in diesem Fall `rdm6300power` heißt.

Übertragen Sie die geänderte Firmware auf die Waage.

Install waage1.yaml

```
INFO ESPHome 2024.6.6
INFO Reading configuration /config/esphome/waage1.yaml...
WARNING GPIO15 is a strapping PIN and should only be used for I/O with care.
Attaching external pullup/down resistors to strapping pins can cause unexpected failures.
See https://esphome.io/guides/faq.html#why-am-i-getting-a-warning-about-strapping-pins
WARNING GPIO5 is a strapping PIN and should only be used for I/O with care.
Attaching external pullup/down resistors to strapping pins can cause unexpected failures.
See https://esphome.io/guides/faq.html#why-am-i-getting-a-warning-about-strapping-pins
INFO Generating C++ source...
INFO Compiling app...
Processing waage1 (board: lolin32_lite; framework: arduino; platform: platformio/espressif32@5.4.0)
--------------------------------------------------------------------------------
HARDWARE: ESP32 240MHz, 320KB RAM, 4MB Flash
 - toolchain-xtensa-esp32 @ 8.4.0+2021r2-patch5
Dependency Graph
|-- AsyncTCP-esphome @ 2.1.3
|-- WiFi @ 2.0.0
|-- FS @ 2.0.0
|-- Update @ 2.0.0
|-- ESPAsyncWebServer-esphome @ 3.2.2
|-- DNSServer @ 2.0.0
|-- ESPmDNS @ 2.0.0
|-- noise-c @ 0.1.4
|-- SPI @ 2.0.0
Compiling .pioenvs/waage1/src/main.cpp.o
Linking .pioenvs/waage1/firmware.elf
```

Bild 9: Sollte es beim Kompiliervorgang zu einem Error kommen, hat sich die Waage wieder abgeschaltet. Dann hilft ein erneutes Einschalten und ein Klick auf Retry.

```
[10:12:22][D][esp32_touch:298]: Touch Pad 'ESP32 Touch Pad GPIO27' (T4): 467
[10:12:22][D][rdm6300:060]: Found new tag with ID 12113815
[10:12:22][D][rdm6300:060]: Found new tag with ID 12113815
[10:12:22][D][hx711:031]: 'HX711 Value': Got value -212726
[10:12:22][D][rdm6300:060]: Found new tag with ID 12113815
[10:12:22][D][rdm6300:060]: Found new tag with ID 12113815
[10:12:22][D][rdm6300:060]: Found new tag with ID 12113815
[10:12:22][D][esp32_touch:298]: Touch Pad 'ESP32 Touch Pad GPIO27' (T4): 467
[10:12:22][D][rdm6300:060]: Found new tag with ID 12113815
[10:12:22][D][rdm6300:060]: Found new tag with ID 12113815
[10:12:22][D][rdm6300:060]: Found new tag with ID 12113815
[10:12:22][D][rdm6300:060]: Found new tag with ID 12113815
```

Bild 10: Wenn die Waage Chipnummern überträgt, ist der Einbau gelungen.

Bedienungsanleitung

Jetzt kann die Waage benutzt werden. Das funktioniert recht einfach. Stellen Sie sich barfuß auf die Waage. Dadurch wird sie eingeschaltet.

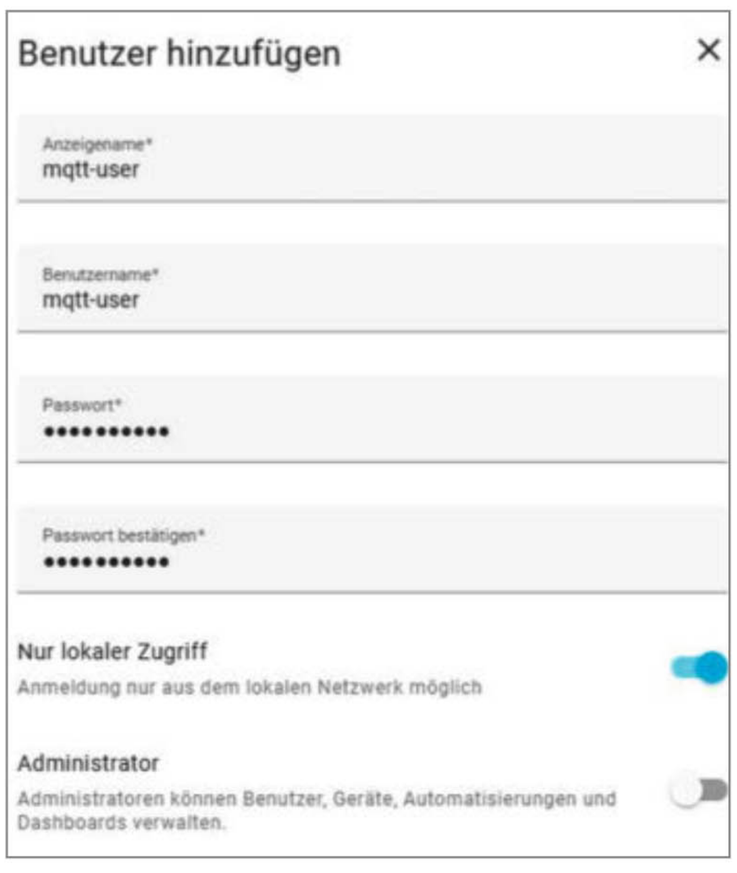


Bild 11: Die Eingabemaske für den neuen Benutzer

Anzeigename	Benutzername	Gruppe	Aktiv	System	Lokal
Heinz	heinz	Besitzer	✓		
Home Assistant Cloud	–	Administratoren	✓	✓	
Home Assistant Content	–	Nur-Lesen Benutzer	✓	✓	
mqtt-user	mqtt-user	Benutzer	✓		✓
Supervisor	–	Administratoren	✓	✓	

Bild 12: Erfolg: Der neue Benutzer erscheint in der Liste.

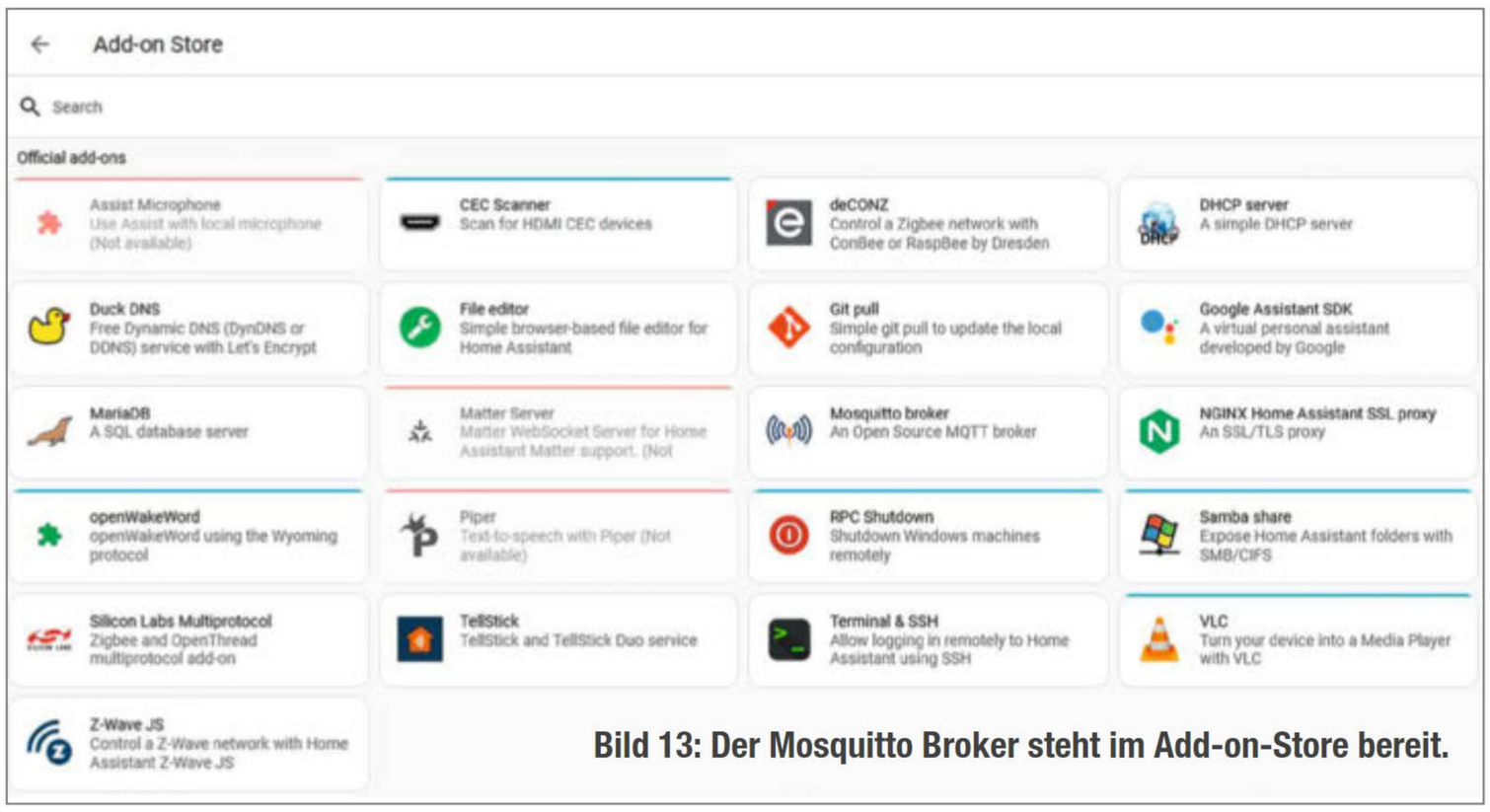


Bild 13: Der Mosquitto Broker steht im Add-on-Store bereit.

Sobald Ihr Gewicht stabil angezeigt wird, halten Sie den Chip an die Antenne des RFID-Lesers. Bei der Version mit der externen Antenne genügt es, ihn vor die hoffentlich in angenehmer Höhe angebrachte Spule zu halten. Bei der einfachen Version mit eingebauter Antenne kommen Sie leider ums Bücken nicht herum, um den Chip auf die Waage links des Displayfensters zu legen. Es sei denn, sie haben den Chip in die Badelatschen geklebt.

Die Waage überträgt nun Ihren Namen und Ihr Gewicht einmal an den Broker. Um einen erneuten Wägevorgang zu starten, müssen Sie warten, bis sich die Waage automatisch abschaltet und dann von vorn beginnen.

Den Datentransport können Sie einfach kontrollieren. Auf der Web-Oberfläche von

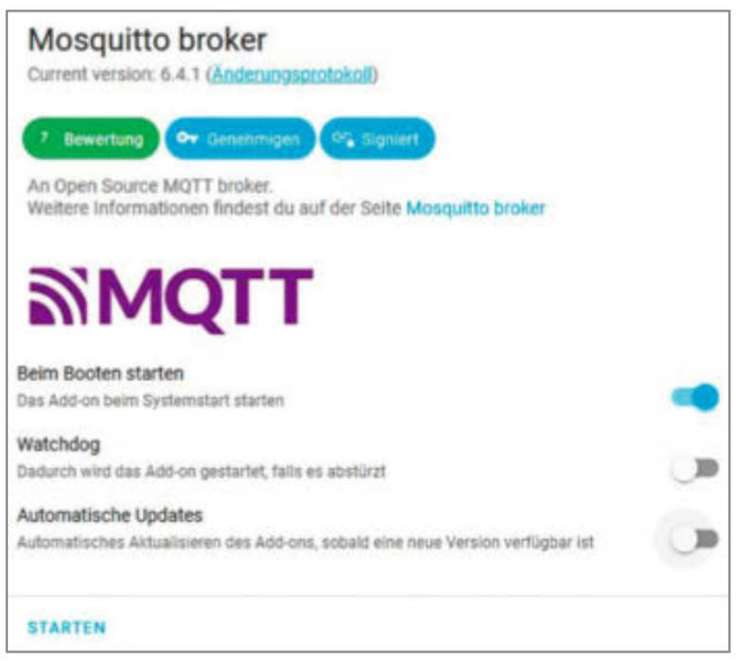


Bild 14: Bevor Sie den Broker starten, kontrollieren Sie die Schalterstellungen.

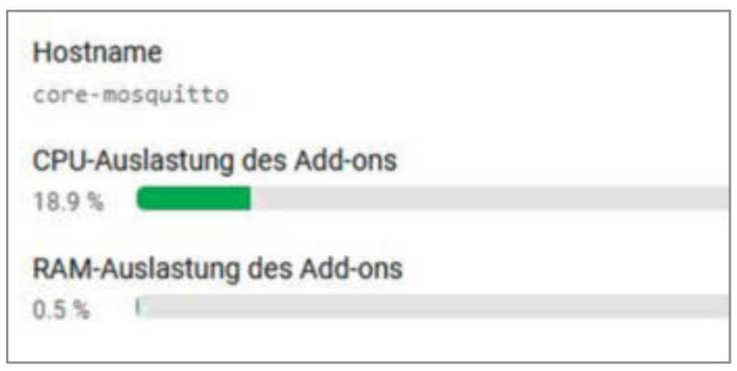


Bild 15: Der Broker läuft.

Bild 16: Obwohl eigentlich ein Add on, wird Mosquitto jetzt als Integration gemeldet.

Bild 17: Mosquitto heißt jetzt MQTT.

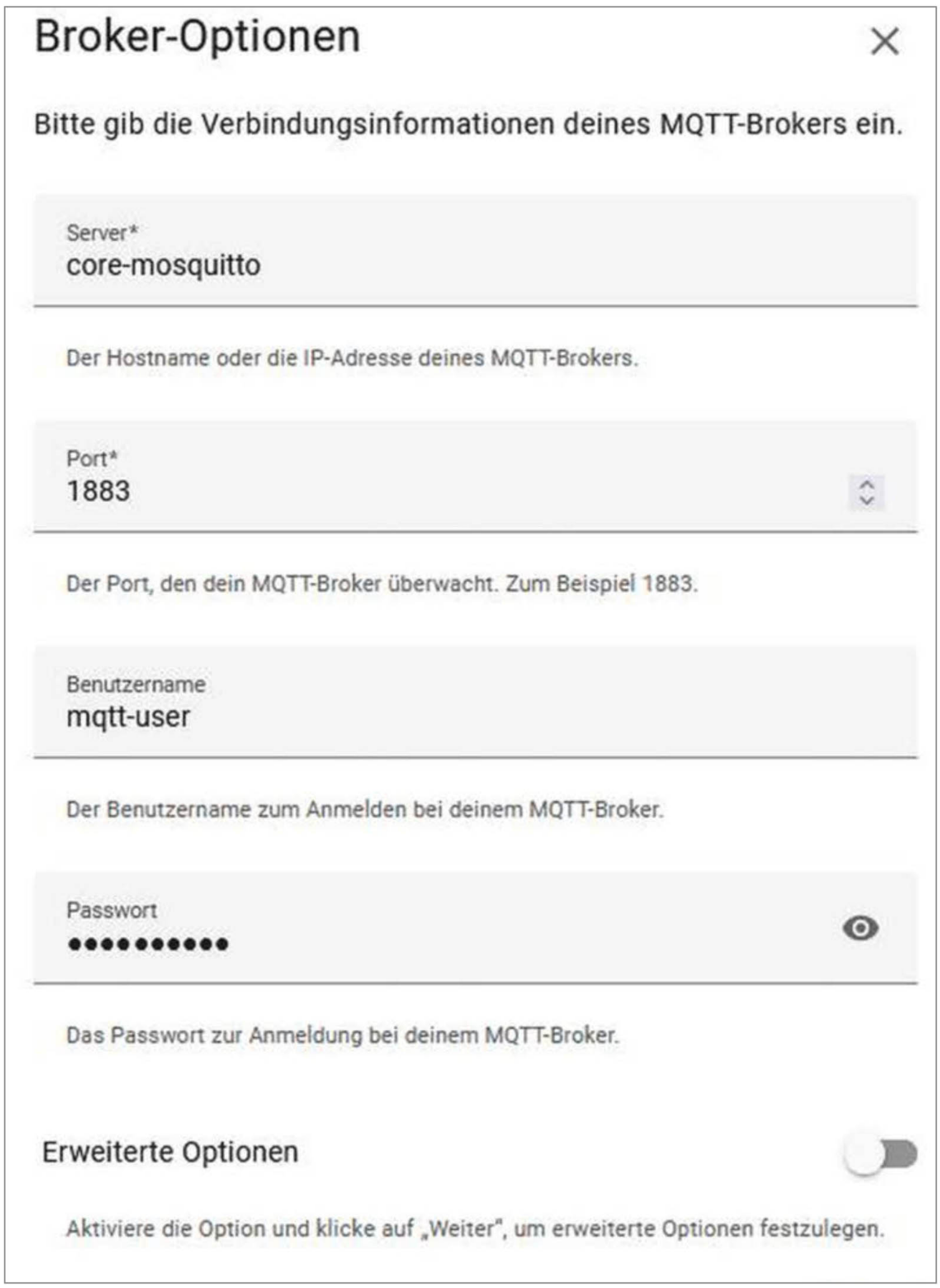


Bild 18: Erst mit diesen Eingaben ist dem Broker auch der neue Benutzer bekannt.

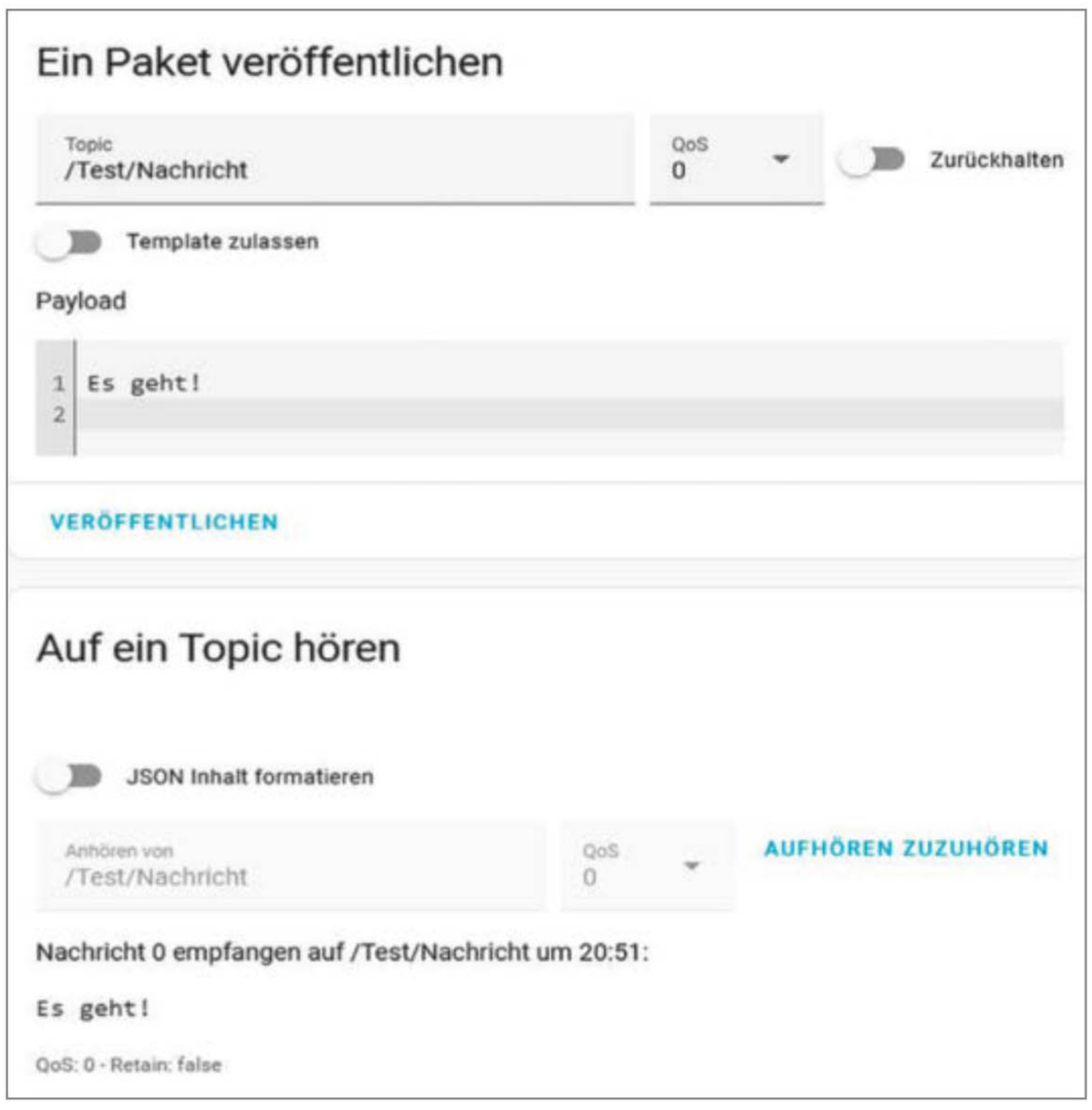


Bild 19: Der Broker besteht den ersten Test.

Bild 20: Gewicht und Namen wurden übertragen.

Home Assistant gehen Sie unter Einstellungen/Geräte & Dienste/MQTT auf „Konfigurieren“. Unter „Auf ein Topic hören“ geben Sie „/rdm6300“ ein und klicken auf „Anfangen zuzuhören“. Starten Sie dann einen Wägevorgang und Sie sehen, was der Broker empfängt (Bild 20).

Weiter geht's

Jetzt haben wir die Hardware-Grundlagen zusammen, damit Ihnen die Waage beim Erreichen Ihres persönlichen Wunschgewichts helfen kann. In einem weiterführenden Onlineartikel beschreibe ich, wie Sie die an den Broker übertragenen Daten in einer Datenbank speichern, Ihr Wunschgewicht einstellen und sich die Annäherung an dieses Ziel als Grafik anzeigen lassen können. Damit sollten es Ihnen leichter fallen, zielgenau abzunehmen oder manchmal auch zuzunehmen. *—hgb*

MQTT, was ist das?

MQTT (**M**essage **Q**ueing **T**elemetry **T**ransport) ist ein offenes Protokoll zur Übermittlung von relativ kleinen Datenmengen zwischen unterschiedlichen Geräten. Es ist besonders für Netzwerkverbindungen geeignet, in denen die Datenübertragung unzuverlässig ist, weil beispielsweise der Empfänger zum Zeitpunkt des Sendens der Daten (noch) nicht verfügbar ist.

Um sicherzustellen, dass die Daten ankommen, werden sie nicht direkt zum Empfänger gesendet, sondern der Datensender veröffentlicht die Nachricht (publish) zunächst auf einem Nachrichtenserver, dem Broker. Jede Nachricht hat dabei einen Namen (Topic), mit dem der Dateninhalt (payload) gesendet wird oder unter dem er abgerufen werden kann. Empfänger, die auf diese Nachricht zugreifen möchten, müssen diese Topics (subscribe) abonnieren und erhalten dann die Daten zugesendet.

Sowohl Datensender als auch Empfänger stellen für den Broker Clients dar. Daher kann ein Sender auch Daten von anderen Clients empfangen. Der Zugang zum MQTT-Broker erfolgt in der Regel über seine IP-Adresse, an der sich Clients mit Benutzernamen und Passwort anmelden müssen. Dabei muss der Broker nicht unbedingt Bestandteil eines lokalen Netzwerkes sein, sondern kann auch außerhalb im Internet angeordnet werden.

In der Nachricht können außerdem ein oder mehrere Flags gesetzt werden, die festlegen, wie der Broker mit den Daten umgehen soll. Das Flag `retain` beispielsweise bestimmt, dass die Daten auf dem Broker dauerhaft erhalten bleiben. Mit weiteren Flags kann der Nachrichtensender sicherstellen, dass die Daten auch wirklich entgegengenommen wurden. Kann deshalb, weil es verschiedene Stufen (**QoS, Q**uality **o**f **S**ervice) gibt. Im einfachsten Fall (QoS Level 0) sendet der Broker die Daten einmal (fire and forget), ohne eine Bestätigung für den Empfang anzufordern. Hierbei ist nicht sichergestellt, ob der Empfänger die Daten auch wirklich komplett erhalten hat. Bei QoS Level 1 (at least once) hingegen wird die Nachricht so oft gesendet, bis der Empfang quittiert wird. Hierbei kann es auch zu mehrfachem Empfang kommen, was in einigen Fällen (Umschaltprozesse o. ä.) Probleme bereiten kann. Dagegen hilft QoS Level 2, bei dem sichergestellt wird, dass jede Nachricht genau einmal beim jeweiligen Empfänger ankommt.

Die Topics bilden eine hierarchische Ordnung, das heißt, zu einem übergeordneten Topic (zum Beispiel „Waage“) sind Unter-Topics in mehreren Ebenen erlaubt (etwa „Waage/Klaus/Gewicht/Datum“). Als Nachrichten kommen nicht nur Daten, sondern beispielsweise auch Befehle infrage, die etwa an ein Gerät im Smart Home gesendet werden (Wohnzimmer/Deckenlampe/On).

Raspberry Pi als Blackberry-Handheld

Mit einem handlichen 3D-gedruckten Gehäuse hat Maker Taylor Hay einen Handheld-Raspberry-Pi gebaut.

von Daniel Schwabe

Taylor Hay

In diesem Gehäuse findet ein vollständiger Raspberry Pi 4 Platz. An der Seite des Gerätes sind die USB-Ports und der LAN-Anschluss des Pi zugänglich, ebenso der SD-Karten-Slot auf der anderen Seite. HDMI- und 3,5-mm-Klinkenstecker für Audio-Ein- und -Ausgabe zeigen allerdings ins Innere des Gehäuses und sind nicht zugänglich. Der Pi wird mit einem Akkupack mit Strom versorgt, das über ein Controller-Board einer Powerbank geladen wird. Dieses Board verbindet den Akku dann auch mit dem Raspberry Pi. Über das Original-Display des Akkupacks kann an der unteren Seite des Handhelds der Ladestand abgelesen werden. Verbaut ist ein 1s3p-18650-Akkupack mit drei 3000-mAh-Zellen. Insgesamt sind es also 9000 mAh. Dieses Board erlaubt es, den Handheld über USB zu laden. Allerdings kann der Raspberry Pi aufgrund der Powerbank-Herkunft nicht während des Ladevorgangs verwendet werden. Dieses Manko will Taylor Hay in einer zukünftigen Revision beheben.

Die grafische Ausgabe geschieht über ein quadratisches HyperPixel-4.0-Touchdisplay mit 720 × 720 Pixeln. Das Display ist 4 Zoll (10,16 cm) groß und bietet eine Bildwiederholungsrate von 60 Hz und 18 Bit Farbraum. Verbunden wird das Display über ein Extra-Board, das über den 40-Pin-GPIO-Stecker des Raspberry Pi (von dem laut Hersteller auch alle Pins genutzt werden) angedockt wird.

Beim Vorbild dieses Projektes, dem Blackberry, war die Tastatur eine wichtige Eigenschaft. Die hier verwendete Blackberry-Q10-USB-/-Bluetooth-Tastatur ist optisch der Handytastatur nachempfunden und bietet neben einem QWERTY-Tastaturlayout auch eine Trackpad-Maus. Dieses Trackpad kann auch als Scrollrad verwendet werden. Die Tastatur wird als externes Eingabegerät vertrieben und verfügt über ein eigenes Gehäuse. Die Tastatur bietet ein Standard QWERTY-Layout mit mehrfacher Belegung der Tasten mit Sonderzeichen etc. Diese können durch das Durchschalten verschiedener Belegungsebenen mit einer Tastenkombination genutzt werden. Für die Verwendung in diesem Projekt wurde die Tastatur direkt in das Gehäuse eingebaut und an die USB-Pins des Pi angelötet. Die Tastatur verwendet die Open-Source-ZMK-Firmware.

Als Betriebssystem für den Blackberry-Pi-Handheld verwendet Taylor Hay Kali Linux. Allerdings sollten auch andere Raspberry-Pi-kompatible Systeme funktionieren. Trotzdem scheint die Einrichtung des Displays bei jedem Betriebssystem etwas anders zu erfolgen. Auf der Blackberry-Pi-Projektseite gibt es einen Abschnitt, wie es unter Kali funktioniert, und auf der Herstellerseite des Displays für Raspberry OS Bullseye und Buster.

Auf der Projektseite des Blackberry Pi gibt es weitere Infos. —*das*

▶ **Link zur Projektseite auf Hackaday**

Taylor Hay

Nicht nur Bildschirm und Tastatur: Auch die USB-Slots und der LAN-Anschluss sind zugänglich.

Taylor Hay

Geräumiges Innenleben: Neben dem Raspberry hat auch die Tastatur bequem Platz.

Taylor Hay

In dem 3D-gedruckten Gehäuse wird die Tastatur mit integriert.

Großformatkamera im Eigenbau

Meine Linhof-Master-Technika-Laufbodenkamera (Großformat 4×5 Zoll) ist nicht gerade ein Leichtgewicht und von den Verstellmöglichkeiten benutze ich meist nur den Objektivshift für Architekturaufnahmen. Daher habe ich eine leichtere Shiftkamera gebaut.

von Uwe Magnus

Uwe Magnus

Die Kamera stammt größtenteils aus dem 3D-Drucker: 1,3 kg Gewicht gegenüber der Master Technika mit 2,6 kg. Das drehbare internationale Rückteil stammt von Linhof (auf eBay gekauft). Kameragehäuse und die Frontstandarte wurden in einem 3D-Programm entworfen und 3D-gedruckt. Der Druck des Kameragehäuses hat ca. 45 Stunden gedauert.

Die Rückseite des Kameragehäuses wurde wegen der Planparallelität von Vorder- und Rückseite übergefräst. Für Gewinde wurden gerändelte Messing-Gewindebuchsen mit 2K-Kleber eingepresst. Die Frontstandarte ist mit vier kugelgelagerten Linearführungen am Kameragehäuse befestigt und mit einem schwarzen Samtband lichtdicht beklebt. Durch einen Schrittmotor kann die Frontstandarte in der Höhe verstellt werden oder, falls der Akku leer ist, über einen Rändelknopf.

In der Frontstandarte sind fünf gefederte und vergoldete Kontaktstifte, die eine binäre Kodierung in den Objektivplatinen abtasten. Das Kameragehäuse enthält zwei Hallsensoren, um die untere Stellung der Frontstandarte zu detektieren und um die Stellung der Planfilmkassette (Querformat oder Hochformat) zu erfassen.

Zur Verstellung hat ein Großformat-Objektiv je nach Brennweite einen bestimmten Bildkreis, der größer ist als das Negativformat. Dieses muss daher innerhalb des Bildkreises liegen, sonst kommt es zu Abschattungen. Jedes Objektiv lässt sich also nur um ein bestimmtes Maß nach oben schieben (shiften).

An die Kamera wird ein Bedienteil angeschlossen, das den Schrittmotor ansteuert, die Hallsensoren, die Objektivkodierung sowie die Lage der Planfilmkassette abfragt. In der Software ist für jedes Objektiv und die Kassettenlage der maximale Verstellweg des Objektivs hinterlegt. Shiftet man das Objektiv, wird dieses gestoppt, wenn der Rand des Bildkreises erreicht ist. Da die Shiftkamera keinen Balgen hat, müssen die Objektive in einer Schnecke sitzen, um scharf stellen zu können.

Es wurden vier Weitwinkel-Objektive mit Einstellschnecken auf Objektivplatinen montiert, 90 mm, 75 mm, 65 mm und 58 mm (bei 4×5 Zoll Negativen sind 150 mm die Normalbrennweite). Um bei einem Motiv die passende Brennweite auswählen zu können, bietet sich ein optischer Aufstecksucher an. Der Sucher meiner Master Technika lässt sich leider nicht verwenden, da dort die kürzeste Brennweite 75 mm ist. Daher benutzte ich ein iPhone SE mit Vorsatzobjektiv als Sucher, es kommt dabei die App „Mark II Artist's Viewfinder“ zum Einsatz. Das iPhone lässt sich für Hoch- oder Querformat um 90 Grad drehen. Einen Baubericht aller Komponenten mit vielen Fotos gibt es auf meiner Website unter dem Menüpunkt „Selbstbau Großformatkamera“. —*caw*

▶ **uwe-magnus.de**

UWE MAGNUS

Der Kamerakörper mit Schrittmotor und Anschlussstecker für das Bedienteil. Alle Verschraubungen sind mit Messing-Gewindeeinsätzen verstärkt.

UWE MAGNUS

Kamera von vorn: Am schwenkbaren iPhone-Sucher erkennt man die Aufsatzlinse.

UWE MAGNUS

Kamera von hinten mit Bedienteil. Das Rückteil ist im Querformat montiert.

Zeitgesteuerte Kamera mit KI-Überwachung

Im Gegensatz zu herkömmlichen Kameras sendet diese Kamera die geschossenen Bilder an einen KI-Dienst, um bei bestimmten Ereignissen im Bildausschnitt eine Meldung auf das Handy zu schicken.

von Daniel Schwabe

Hiroshi Nakajima

Auf Basis eines ESP32-S3 hat Maker Hiroshi Nakajima eine Fotokamera mit Zeitsteuerung und AI-Funktionen gebaut. Ausgestattet ist das Projekt mit einer OV5640-ESP-32-Kamera, die bis zu 2592 × 1944 Pixel große Aufnahmen machen kann. Nakajima nutzt kein fertiges ESP-Board, sondern hat ein eigenes entworfen, auf dem alle Bauteile platziert sind.

Eingepasst wird die Technik in ein 3D-gedrucktes Gehäuse. Mit einem integrierten Qi-Schaltkreis für kabelloses Laden kann der eingebaute 1100-mAh-Akku geladen werden.

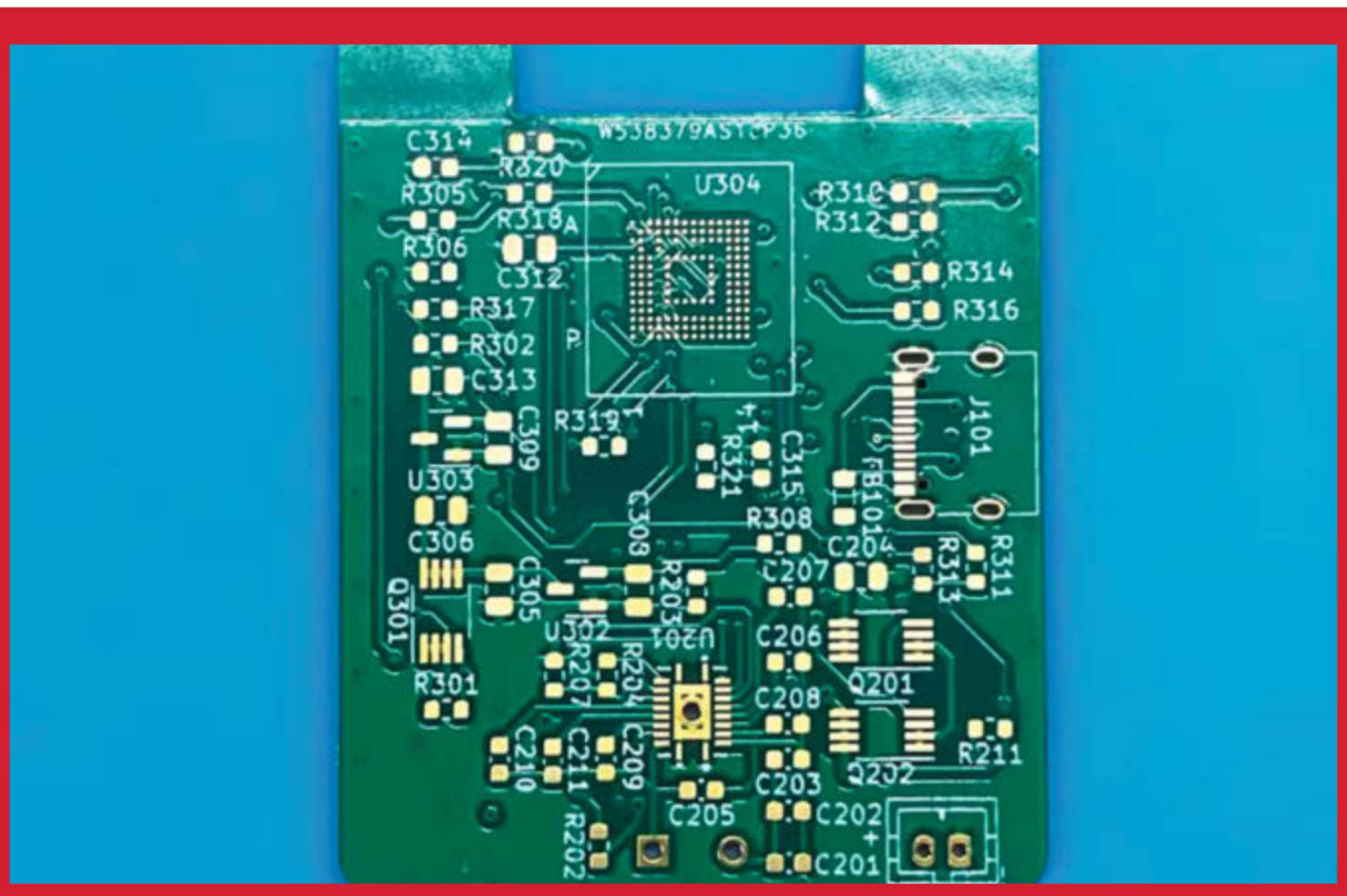
Die WiFi-Funktionalität des ESP32 wird genutzt, um die Kamera ins Netzwerk zu bringen. Über eine Weboberfläche können Einstellungen bezüglich des Aufnahmeintervalls der Fotos und Standby-Phasen gemacht werden. Beispielsweise kann die Kamera zu einer bestimmten Uhrzeit aus dem Schlafmodus aufwachen, ein Foto schießen, dieses an den KI-Dienst leiten und auf die Rückmeldung warten. Wenn auf dem Bild nichts Wichtiges erkannt wurde, schläft die Kamera bis zur nächsten programmierten Uhrzeit wieder ein. Erkennt die KI aber etwas, wird eine Nachricht an das Handy geschickt und das Bild zu einem Imagehoster hochgeladen. Danach schläft die Kamera bis zum nächsten Bild wieder ein. Auch Aufnahmen in bestimmten Intervallen sind möglich.

Diese Aufnahmen werden anschließend im eingebauten 64-Gigabyte-Speicher (eMMC) abgelegt. Die Bilder lassen sich dann mit OpenAI ChatGPT analysieren. Dafür werden sie über das Internet an den OpenAI-Service geschickt und dort auf ein bestimmtes Merkmal hin untersucht. Das geschieht über einen KI-Prompt. Wie genau der aussieht, steht weiter unten. Nakajima hat die Kamera entworfen, um sofort über einen Sturz seines Vaters informiert zu werden, damit er Hilfe holen kann.

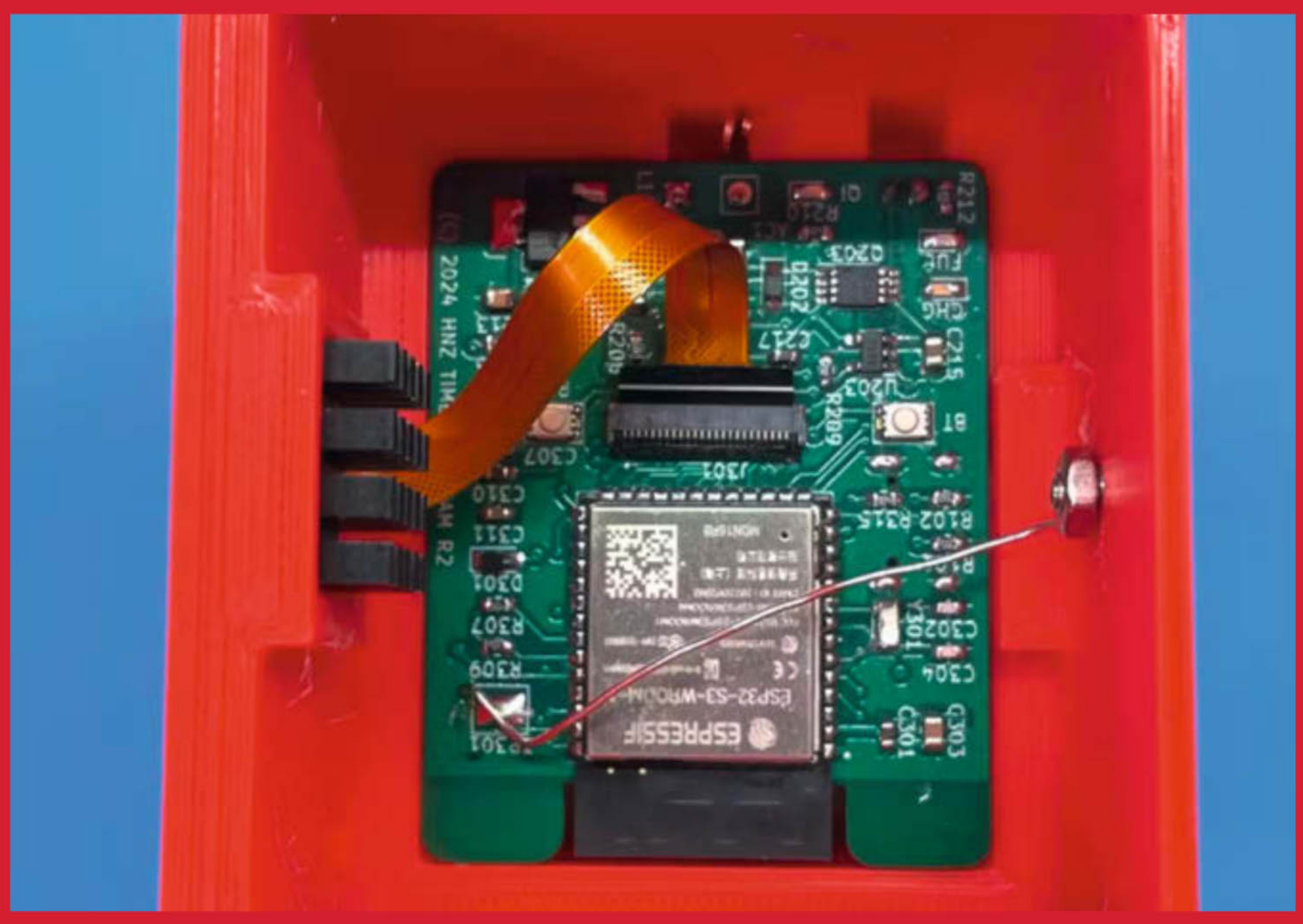
Dafür werden die Bilder mit einem Prompt (einer Anweisung für die KI) versehen. Beispielsweise: „If there is somebody, add 'NOTICE' in the reply and reason. If there is nobody, just reply 'NONE'." Erkennt die KI nun jemanden auf dem Bild, bekommt die Kamera „NOTICE" zurückgespielt. Daraufhin wird das Bild zu einem Imagehoster hochgeladen und der Link dazu per LINE (ein Messenger) verschickt. Dadurch wird eine Benachrichtigung ausgelöst und man kann direkt nachschauen, was auf dem Bildausschnitt zu sehen ist. Ist auf dem Bild dann wirklich eine Gefahrensituation zu erkennen, kann man dementsprechend reagieren.

Auf der Projektseite hat Hiroshi Nakajima eine Liste von weiteren Anwendungsfällen bereitgestellt – beispielsweise als Haustierüberwachung oder Sicherheitssystem. Auf dieser Seite sind auch detaillierte Schritte zum Nachbau der Kamera zu finden. Auf der GitHub-Seite des Projekts sind KiCad- und STL-Dateien sowie Code und Schaltpläne bereitgestellt. —*das*

▶ **make-magazin.de/xpqh**

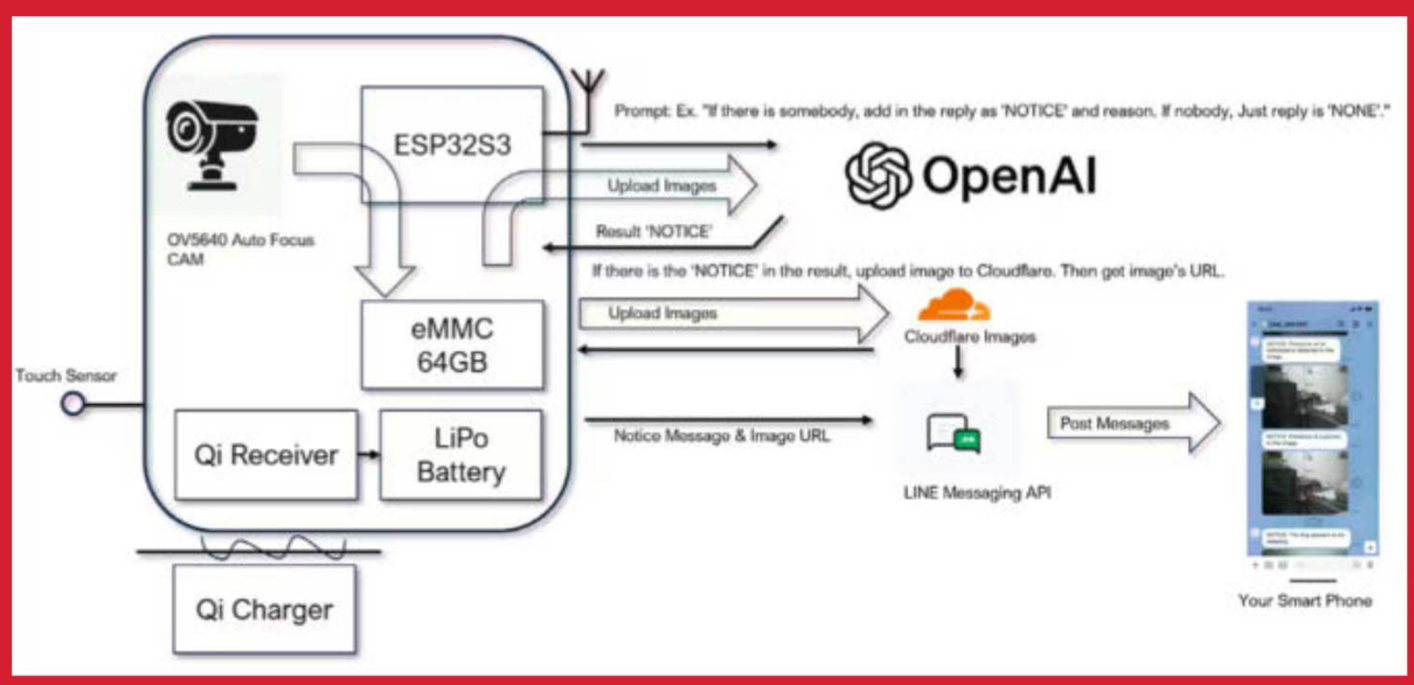


Ein selbst erstelltes PCB für das Projekt. Das ESP-32-Modul wird direkt auf diese Platine aufgelötet. Es wird kein fertiges Board verwendet.



Hiroshi Nakajima

Zur selbst designten Leiterplatte kommt noch ein 3D-gedrucktes Gehäuse, in das die Technik perfekt hineinpasst.



Hiroshi Nakajima

Der Programmablauf: Nachdem ein Bild geschossen wurde, wird es an einen KI-Dienst geschickt. Erkennt die KI dort etwas Bestimmtes, wird das Bild dann an einen Messenger weitergeleitet.

Der sprechende Hut

Gryffindor, Hufflepuff, Ravenclaw oder gar Slytherin? Wir wissen nicht, in welches Haus Sie der sprechende Hut aus der Zauberschule Hogwarts stecken würde. Aber wir wissen, was in ihm steckt. Und das zeigen wir Ihnen hier ganz ohne Zauberstab oder Denkarium.

von Heinz Behling

Um an das Innenleben dieser ursprünglich Godric Gryffindor gehörenden Kopfbedeckung aus den Harry-Potter-Geschichten zu kommen, waren keinerlei übernatürliche Kräfte, allerdings zunächst ein Schraubendreher mit dreieckigem Spezialbit notwendig, denn die Abdeckung des Batteriefachs war mit Spezialschrauben befestigt (Bild 1).

Auch eine Schere kam zum Einsatz, da das Hut-Innenleben nicht durch die Stofföffnung am Batteriefach passte. Offenbar wurde es bei der Produktion des Hutes eingenäht. Nach dem Auftrennen zweier Nähte (Längsnaht am Innenfutter und die Naht an der Hutkrempe direkt am Schalter) änderte sich das aber.

Das weiße Kunststoffchassis des Hutes (es kann übrigens in der Größe verstellt werden, aber nicht über Kinderkopfgröße hinaus) besteht aus zwei miteinander verschraubten Halbschalen. Im becherartigen Oberteil befinden sich oberhalb des Batteriefachs die Elektronik, der gemeinsame Antriebsmotor (läuft ab etwa 3 V Spannung) für die Bewegungen sowie der Lautsprecher (siehe Titelbild des Artikels). Vorn an der Hutkrempe ist eine per Draht mit der Elektronik verbundene Kupferfolie eingeklebt. Sie dient als Sensor, der den Hut beim Aufsetzen aktiviert und losplappern lässt, sofern er mit dem Schalter am Batteriefach, der gleichzeitig zur Sprachauswahl dient, eingeschaltet ist.

Nach dem Trennen der beiden Huthälften (Achtung: insgesamt 18 Schrauben am Gehäuse und an den Hebeln sind zu lösen) kommen die Mechanik, gut geschützt in einem braunen Kunststoffgehäuse, die kleine Platine und der Lautsprecher zum Vorschein (Bild 2). Mund, Augenbrauen und die Hutspitze werden über ein Getriebe und einen Hebel zwar unterschiedlich, aber immer gemeinsam von nur einem Motor bewegt. Mit etwas Geschick und selbst konstruierten 3D-Druck-Teilen könnte man die drei Elemente auch von je einem Servo bewegen lassen. So würde aus dem lediglich bekannte Sprüche zitierenden Hut in Kombination mit einem Raspberry o. Ä. und KI eventuell ein intelligenterer Gesprächspartner werden.

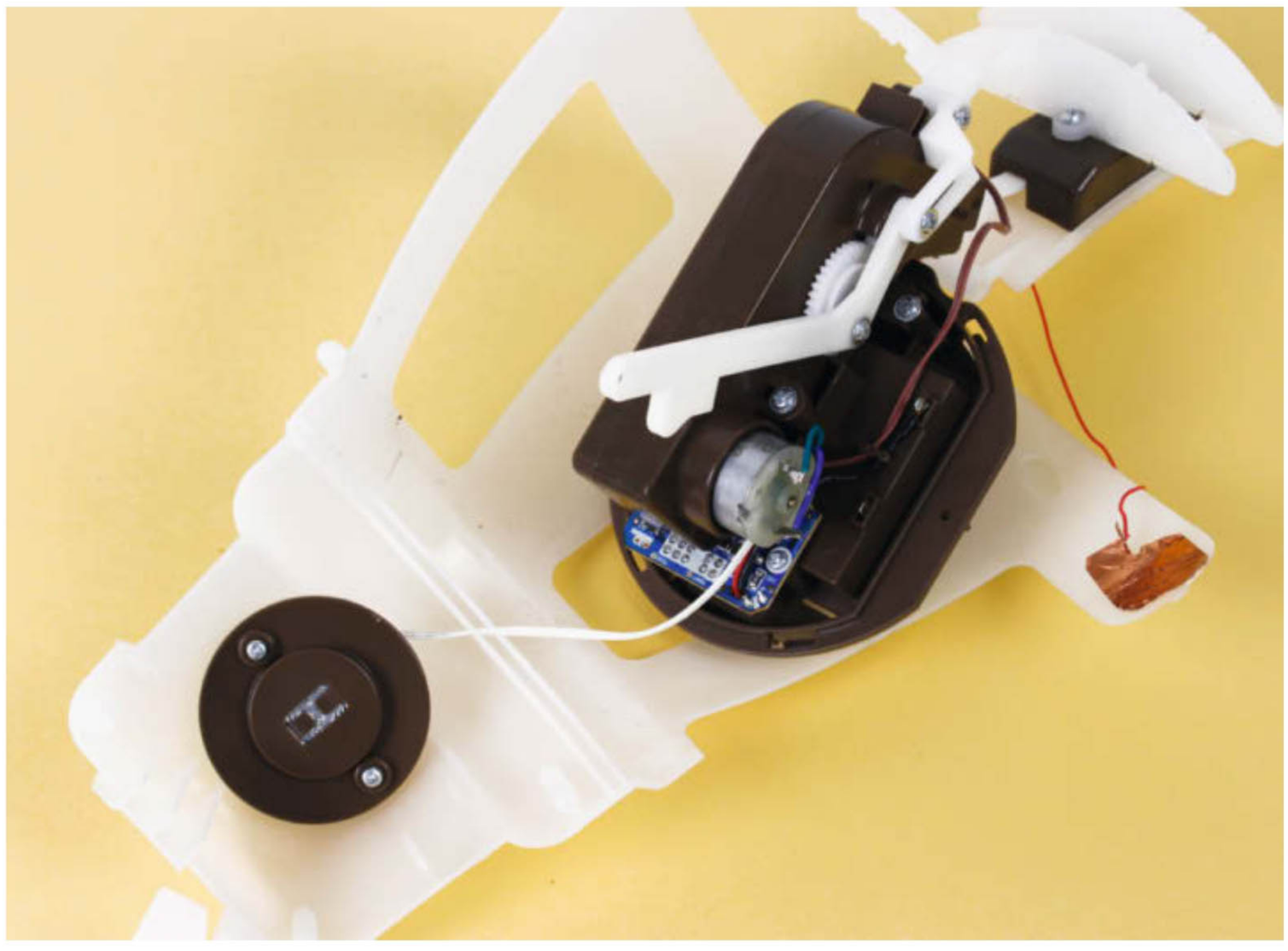


Bild 2: Im braunen Gehäuse sitzt das Getriebe, das über ein Hebelwerk alle Teile bewegt.

Vergossene Elektronik

Der Chip der Elektronik ist dafür aber nicht geeignet, denn er ist, wie bei solchen Geräten üblich, vergossen. Eine Weiterverwendung mit Umprogrammierung kommt daher nicht infrage (Bild 3). Das Batteriefach mit den drei AA-Zellen sollte aber in der Lage sein, einen Mikrocontroller (ESP) oder gar einen Raspi Zero mit genügend Strom zu versorgen.

Der Hut hätte sogar noch reichlich Platz für umfangreichere Elektronikteile: Kameras oder Mikrofone könnte man dann für viele zusätzliche Funktionen wie Sprachsteuerung oder Personenerkennung nutzen. Vielleicht baut sich ja jemand einen hörenden und sehenden persönlichen Assistenten daraus. Allerdings sollte Ihre Hutgröße unter 56 liegen, sonst wird es eng. —*hgb*

▶ **make-magazin.de/x9u8**

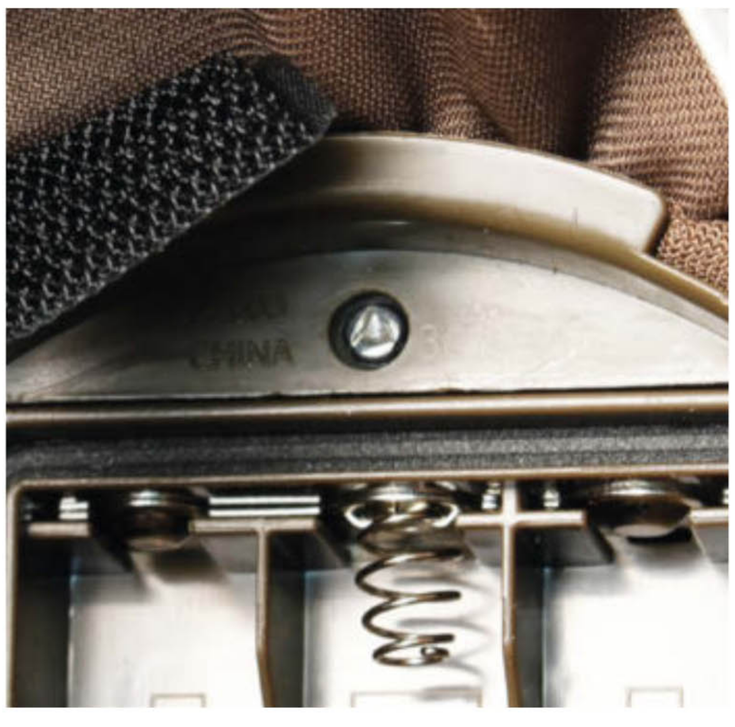


Bild 1: Für die ersten Schrauben am Batteriefach ist ein Spezialbit erforderlich.

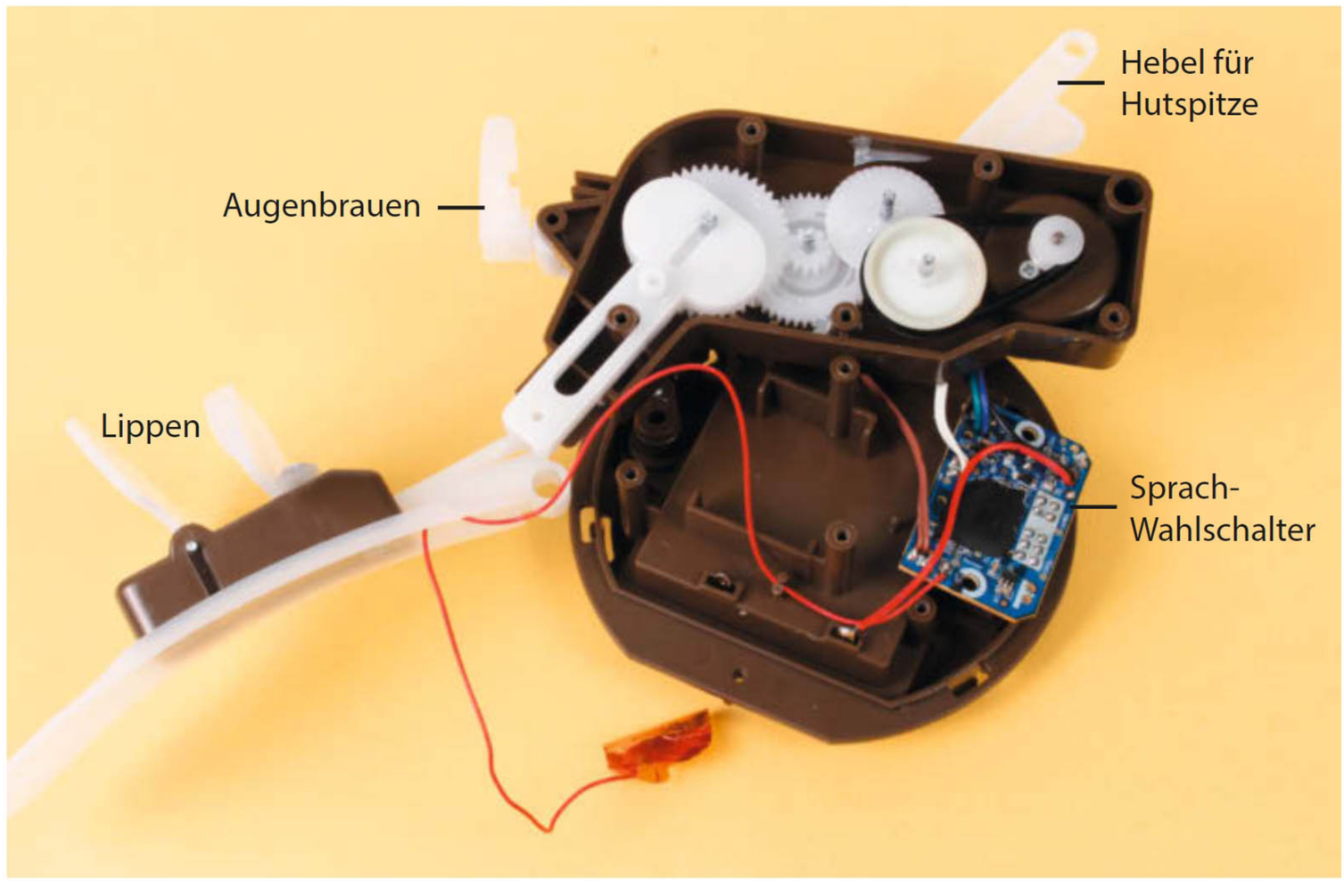
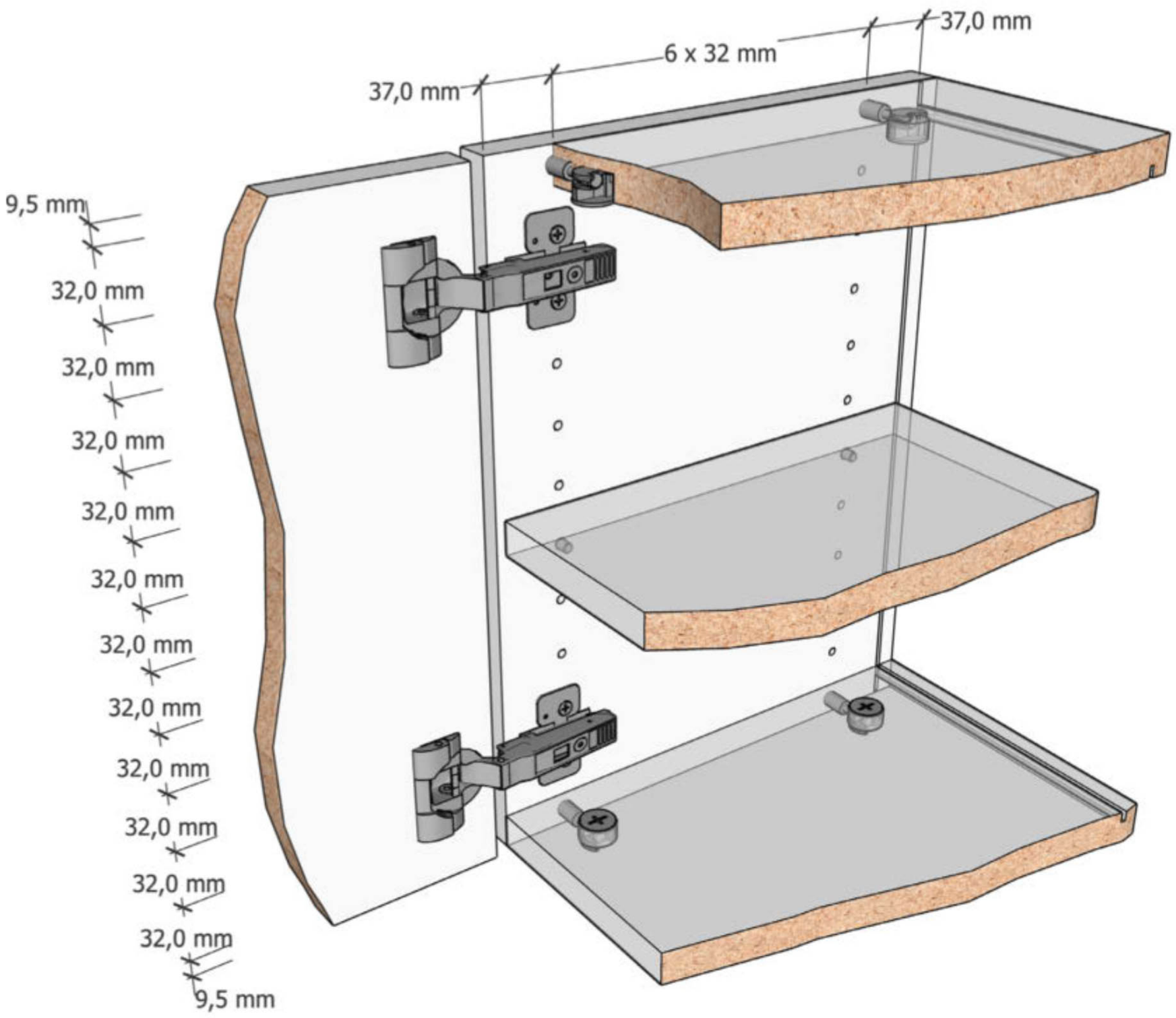


Bild 3: Die vergossene Elektronik und das geöffnete Getriebe

Einfacher Möbelbau dank System 32

Eigene Schränke und Regale zu bauen ist leichter, wenn man sich am Konstruktionsraster der Industrie orientiert. Wir haben drei günstige Bohrlehren für das System 32 ausprobiert und zeigen deren Verwendung.

von Johannes Börnsen



Die Maßangaben zur Montage von Topfscharnieren oder Schubladenauszügen können durchaus einschüchternd sein. Oft müssen Bohrungen mit unrunden Abständen wie 32 Millimeter oder 64 Millimetern gesetzt werden. Dabei entspringen diese Abstände nicht dem Kopf eines hinterlistigen Einwicklers eines Beschlagherstellers, sondern orientieren sich am System 32, einem international gängigen Prinzip zur Konstruktion von Möbeln wie Schränken und Regalen. Während Tischlereien und Möbelhersteller Möbel im System 32 vorwiegend auf großen CNC-Maschinen fertigen, lassen sich dieselben Prinzipien auch mit günstigen Bohrschablonen bei selbstkonstruierten Möbeln anwenden. Beim Zusammenbau ergeben sich die kompliziert wirkenden Bohrabstände für Topfscharniere oder Schubladenauszüge von alleine.

32 mm – das Maß aller Dinge

Das offensichtlichste Merkmal vom System 32 sind die Lochreihen in den Seitenwänden von Regalen oder Schränken, in welche die Fachbodenträger gesteckt werden. Namensgebend ist dabei der Rasterabstand von 32 mm, der den Mittenabstand der Lochreihenbohrungen angibt. Häufig zieht sich diese Lochreihe von oben bis unten durch den ganzen Schrank. Dabei ist jedoch nicht nur der Abstand der Bohrungen untereinander definiert, sondern auch ihre Position innerhalb des Schrankes. Das System 32 legt den Abstand der Lochreihe zur Korpusvorderkante auf 37 mm fest. Für die Fachbodenträger wäre dies nicht

so wichtig, wohl aber für Topfscharniere. Deren Montageplatten nutzen ebenfalls die Löcher der Lochreihenbohrung und benötigen meist 37 mm Abstand zur Vorderkante, damit die Schranktür im richtigen Verhältnis zum Korpus sitzt. Der Durchmesser der Lochreihenbohrungen im System 32 beträgt 5 mm, was dem Durchmesser von Fachbodenträgern und Euroschrauben zum Befestigen von Scharnieren und Auszügen entspricht.

Auch Exzenter-Verbinder wie Hettich Rastex sind Teil des Systems 32. Die dafür nötige Bohrung in der Schrankseite ist die oberste Bohrung der Lochrasterreihe. Sie wird vom Deckel (der Tischler spricht vom Oberboden) verdeckt, der wiederum das Gegenstück des Exzenters aufnimmt. Damit das passt, werden für Möbel im System 32 in der Regel Spanplatten mit 19 mm Stärke verwendet. Die oberste Lochreihenbohrung ist eine halbe Plattenstärke, also 9,5 mm, von der Oberkante entfernt. Das untere Ende des Schrankes folgt den gleichen Prinzipien, wodurch sich automatisch auch die Korpushöhe ohne Sockel oder Füße ergibt. Es setzt sich zusammen aus einem Vielfachen des Rastermaßes von 32 mm plus zweimal der halben Plattenstärke, einmal für oben, einmal für unten.

Kurzinfo

- **Überblick Möbelbau-Konstruktionsprinzip System 32**
- **Drei Bohrhilfen und eine Schablone zum systemgerechten Möbelbau**
- **Tipps zum Bearbeiten von Spanplatten und zum Bau von Korpussen**

Checkliste

Zeitaufwand:
ein Wochenende

Kosten:
je nach Möbelgröße ab 30 €

Material

- **Beschichtete Spanplatte, Umleimer**
- **Beschläge wie Topfscharniere oder Griffe**

Werkzeug

- **Kreissäge, Akkuschrauber, Bügeleisen** (für Umleimer)
- **Bohrschablonen für System 32,** wie im Artikel beschrieben

Alles zum Artikel im Web unter *make-magazin.de/x7sp*

Planungshilfe Rastermaß

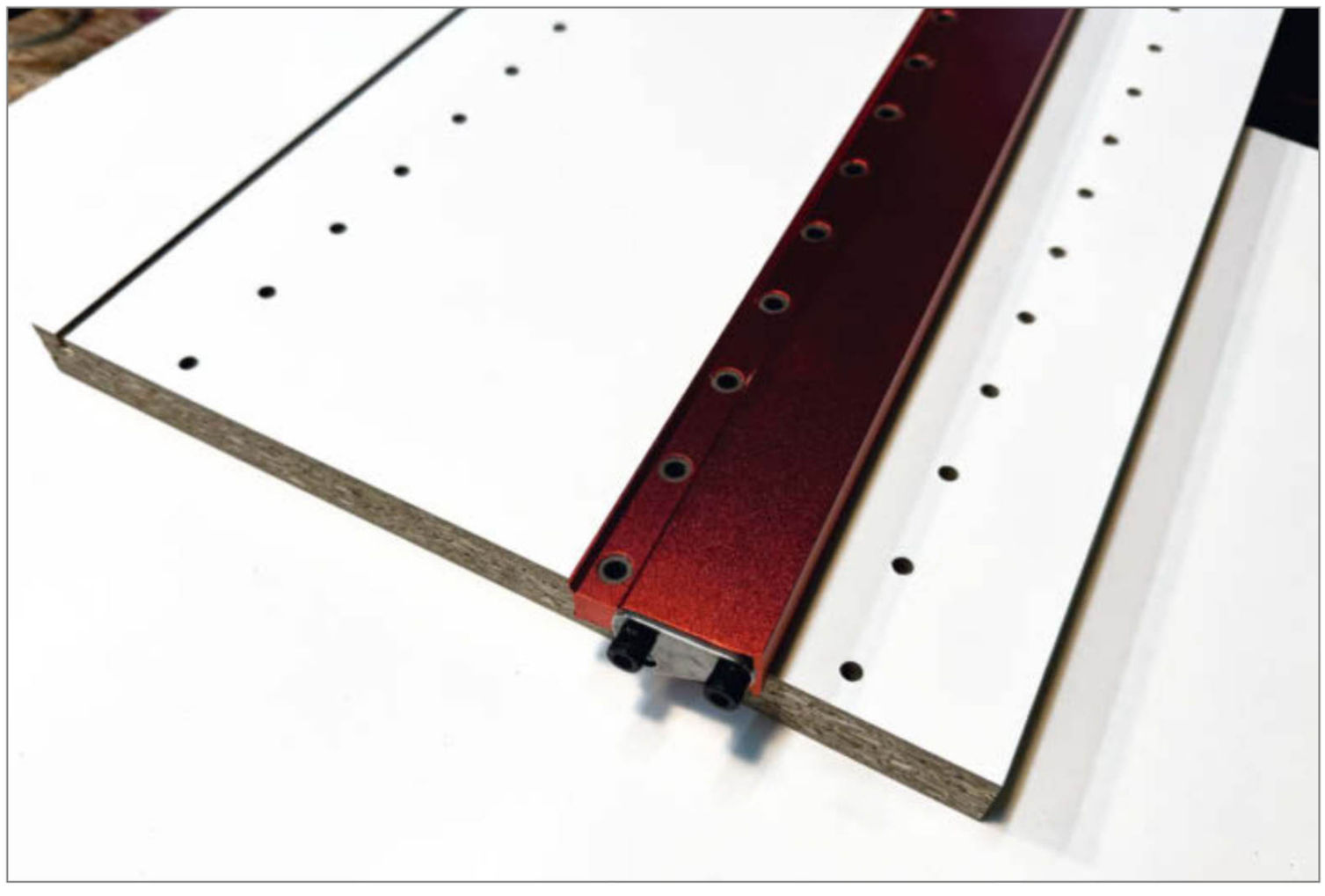
Das 32er-Raster bestimmt auch den Abstand zwischen vorderer und hinterer Lochreihe, in unserem Beispiel ist es 6 × 32 mm, was 192 mm ergibt. Der Abstand der Montagebohrungen von Schubladenauszügen beträgt ebenfalls ein Vielfaches von 32 mm, sodass sie sich in unserem Beispielkorpus problemlos montieren lassen würden (wobei das sehr kurze Schubladen wären und uns keine so kurzen Auszüge bekannt sind). Der Abstand von hinterer Lochreihe zur Innenseite der Rückwand beträgt wieder 37 mm. Die Korpustiefe unseres Beispiels setzt sich also zusammen aus 37 mm von der Vorderkante zur vorderen Lochreihe, 6 × 32 mm Abstand zwischen den Lochreihen, 37 mm von der hinteren Lochreihe zur Innenseite der Rückwand, 3 mm Rückwand und 10 mm hinter der Nut für die Rückwand bis zur hinteren Korpuskante (ergibt 279 mm). Die Rückwanddicke und der Abstand der Rückwandnut sind dabei in System 32 nicht definiert, sondern frei wählbar. Umleimer oder Dichtungslippen werden übrigens zum Korpusmaß gezählt, die Spanplatte muss also gegebenenfalls entsprechend kürzer gesägt werden.

Auch viele andere, typische Bauteile für Regale und Schränke orientieren sich am 32-mm-Raster. So sind die Bohrungen für Küchengriffe, Klappenhalter, Möbelfüße, Rückwandverbinder oder Verbindungswinkel meist 64, 96, 128, 160 oder 192 mm voneinander entfernt und auch Kleiderstangenhalter haben ihre Bohrungen im zum Lochraster passenden Abstand.

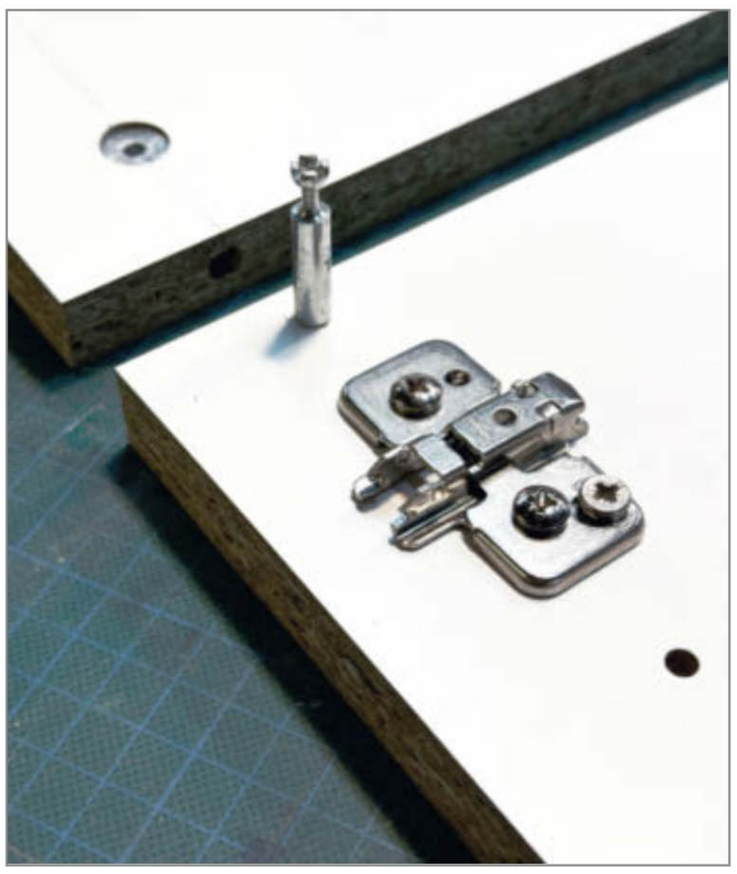
Schablone für Lochreihen

Um beim Selbstbau von Möbeln nicht jedes Loch einzeln ausmessen zu müssen, was unweigerlich zu systemsprengenden Ungenauigkeiten führen würde, gibt es Schablonen und Bohrhilfen für unterschiedliche Elemente des Systems 32. Die Bohrschablone BS-32-500 von Sotech (ca. 72 €) bietet beispielsweise 16 Löcher mit Hartmetall-Bohrführungen mit jeweils 32 mm Mittenabstand. Durch klappbare Anschläge an den Enden der Bohrschablone lässt sich das erste Loch genau 9,5 mm vom Ende der Platte platzieren.

Benötigt man mehr Löcher, als die Schablone in einem Durchgang erlaubt, verschiebt man sie und richtet sie mit einem zweiten 5-mm-Bohrer aus, den man durch die Schab-



Gleiche Abstände dank einfacher Schablone (Sotech BS-32-500). Solch eine Lochreihe dauert nur wenige Minuten und erfordert dank klappbarer Anschläge keinerlei Einstellarbeit.



Einschraubdübel und Topfscharnier-Montageplatte. Beide verschraubt in den Bohrungen der Lochreihe. Oben links ist das Gegenstück zum Einschraubdübel, der Exzenter, zu sehen.

Diese Schablone (Pocket Hole Jig Set) ist von zwei Seiten nutzbar. Zunächst wird wie abgebildet in die Stirnseite gebohrt, anschließend durch die dunkle Hülse auf der rechten Seite in die Fläche. Die Bohrungen dienen zur Aufnahme von Einschraubdübel und passendem Exzenter, bekannt von schwedigen Schränken.

lone in ein beim ersten Durchgang gebohrtes Loch steckt. Um den 37-mm-Abstand von der Plattenkante zu erreichen, befindet sich auf der Unterseite der Schablone eine kleine Lippe, die sich gegen die Plattenkante schieben lässt.

Zum Bohren der hinteren Lochreihe fräst man zuerst die Nut für die Rückwand, in die sich anschließend die Schablonenlippe legen lässt. Die BS-32-500 ist auch mit doppelt so vielen Bohrungen in einer 1-Meter-Version erhältlich (BS-32-1000). Vergleichbare Schablonen in anderen Längen gibt es auch von Wolfcraft und Kreg.

Das kommt mir schwedisch vor

Möchte man große Schränke für einen eventuellen Umzug zerlegbar fertigen, bieten sich Exzenterverbinder an, wie sie unter anderem bei Ikea-Möbeln zum Einsatz kommen. Diese Verbindung haben wir auch bei unserem Beispielkorpus dargestellt. Sie bestehen jeweils aus zwei Teilen: Der Einschraubdübel wird in das oberste Loch der Lochreihenbohrung geschraubt. Er befindet sich dann genau in der Mitte der Ober- oder Unterbodenplatte. Diese benötigt stirnseitig eine Bohrung, in die der Einschraubdübel beim Zusammenbau gefädelt wird. Eine zweite Bohrung auf der Fläche nimmt den Exzenter auf, der durch Verdrehen den verdickten Kopf des Einschraubdübels fixiert. Die benötigten Löcher lassen sich mit dem „Pocket Hole Jig Set" von Ceiever herstellen (ca. 71 €, wobei der Name irreführend ist, da der Begriff „Pocket Holes" synonym für eine andere Verbindungsart genutzt wird). Die Schablone wird stirnseitig an der Platte befestigt (schützen Sie den Druckpunkt der Feststellschraube mit Klebeband) und erlaubt – einmal korrekt justiert – das genaue Bohren beider Löcher. Zum Set gehören auch Bohrer mit Tiefenstopps sowie ein Forstnerbohrer mit 15 mm Durchmesser für die Flächenbohrung.

Topfscharniere frustfrei montieren

Mit den beiden bisherigen Schablonen können sämtliche Bohrungen für den Korpus, für Schubladenauszüge sowie die Montageplatten von Topfscharnieren gemäß System 32 angefertigt werden. Für Topfscharniere benötigt man jedoch auch eine 35-mm-Bohrung

Tipps für den Korpusbau

» In manchen Baumärkten gibt es Spanplatten mit bereits gebohrten Lochreihen in unterschiedlichen Breiten zu kaufen. Vielleicht passen diese ja für Ihr Projekt. Achten Sie beim Zuschnitt aber auf den Abstand zum ersten Loch, falls sie dieses für Einschraubdübel nutzen möchten.

» Manche Baumärkte bieten neben dem Holzzuschnitt auch das Verkleben von Umleimern als Service an. Hier bekommen Sie auch zwei Millimeter dicke Umleimer, die robuster sind als ihre dünnen Kollegen zum Selberaufbügeln.

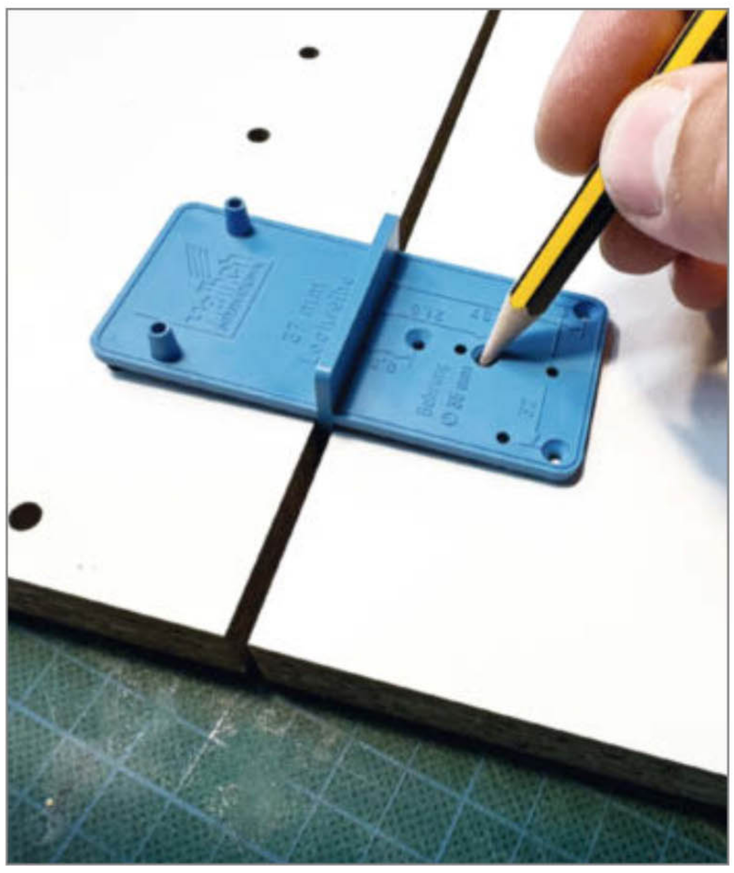
» Wenn Sie sich Platten im Baumarkt zusägen lassen, prüfen Sie die Maße vor dem Bezahlen und achten Sie auf ausrissfreie Kanten. Spätere Reklamationen sind nicht möglich. Zuschnittdienste aus dem Netz arbeiten nach der Erfahrung des Autors präziser, vermutlich nutzen sie CNC-gesteuerte Sägen oder fräsen die Platten CNC-gestützt aus.

» Bevor sie Umleimer aufbügeln, sollten Sie die Nut zum Einschieben der Rückwand fräsen. Sie können dann einfach über das Ende der Platte hinaus fräsen, der Umleimer deckt die Nut oben und unten anschließend ab.

» Wenn Sie beschichtete Spanplatten selber zuschneiden, achten Sie auf ein scharfes Sägeblatt mit ausreichend Zähnen, das für beschichtete Platten geeignet ist. Stumpfe Sägeblätter oder zu wenig Zähne ergeben Ausrisse. Auch das Aufkleben von Klebeband auf die Schnittlinie vor dem Sägen beugt Ausrissen vor. Es hilft, zunächst nur einen Millimeter tief zu sägen und erst im zweiten Durchgang die volle Plattenstärke zu durchtrennen. Professionelle Kreissägen haben hierfür einen Vorritzer. Das ist ein zweites Sägeblatt, das die Beschichtung im Gleichlauf durchtrennt, bevor die Platte auf das eigentliche Sägeblatt trifft.

» Für das Bohren beschichteter Platten sind scharfe Werkzeuge hilfreich. Lassen Sie den Bohrer außerhalb des Materials auf Drehzahl kommen und senken Sie ihn dann langsam auf das Material ab. Und: je mehr Drehzahl, desto besser. Für die kleinen 5-mm-Löcher der Lochreihen dürfen es ruhig 2000 Umdrehungen pro Minute sein.

» Exzenterverbinder mit Einschraubdübeln ziehen Schrankwand und Ober- beziehungsweise Unterboden zwar zusammen, bieten aber wenig Stabilität gegen Scherkräfte. Hier helfen klassische Holzdübel, auch wenn sie nicht eingeleimt werden. Holzdübel sind nicht Teil des Systems 32, lassen sich aber beispielsweise mit dem Wolfcraft Meisterdübler platzieren.



Diese einfache Schablone (Hettich Multiblue) dient zum Markieren der Topfbohrung für Topfscharniere. Im Bild links ist die Schrankseitenwand, rechts die Tür.

Wer viele Topfscharniere einlassen muss, wird diese Bohrhilfe (Kreg Bohrschbalone für Topfscharniere) zu schätzen wissen. Der Forstnerbohrer samt Tiefenstopp ist im Set enthalten.

Blick in den fertigen Schrank, auf das Topfscharnier. Das Ergebnis sind präzise passende Lochreihen, Scharniere und Schubladenauszüge, ohne einmal gemessen zu haben.

auf der Innenseite der Schranktür. Diese lässt sich mit der 3 Euro teuren Ankörnschablone „Multiblue“ von Hettich markieren (manche Baumärkte verkaufen eine baugleiche, graue Version, ebenfalls von Hettich). Dafür wird die Schranktür vor der Korpusmontage in geöffneter Position neben die Seitenwand gelegt. Die Ankörnschablone wird mit zwei Nasen in die Lochreihenbohrung gesteckt, gibt das passende Maß zwischen Schrankwand und Tür vor und bietet ein Ankörnloch für die Mitte der 35-mm-Bohrung. Dabei ergibt sich ein Abstand des Bohrlochs zur Außenkante der Tür von 4 mm. Für Hettich und viele andere Hersteller passt das.

Manche Scharnier-Montageanleitungen fordern jedoch, die Topfbohrung einen Millimeter weiter in die Fläche der Tür zu schieben, also einen Steg von fünf Millimetern zwischen Bohrloch und Außenkante zu lassen. Dies lässt sich mit Kregs „Bohrschablone für Topfscharniere“ (ca 45 €) erledigen. Die Multiblue-Schablone überträgt dann lediglich die senkrechte Position der Topfbohrung, den Abstand von der Kante stellt man an Kregs Bohrschablone von drei bis sechs Millimetern ein. Im Set befindet sich auch ein (scharfer) Forstnerbohrer mit 35 mm Durchmesser. Außerdem erlaubt die Schablone, die Bohrungen für die Befestigung des Topfscharniers in der Tür anzukörnen, die man dann aber freihändig bohren muss.

Alles kann, nichts muss

Natürlich gibt es keinen Zwang, sich dieses Systems zu bedienen. Und sie können auch frei wählen, welche Elemente für Ihr Projekt passen. Lochreihen für Fachböden können praktisch sein und sind mit der Lochreihenschablone schnell gebohrt, selbst wenn der Rest des Systems 32 nicht zum Einsatz kommt. Dreht man die Multiblue-Schablone um, lassen sich zwei Löcher für die Montageplatte von Topfscharnieren anzeichnen, ohne eine ganze Lochreihe bohren zu müssen. Der einfachste Einstieg ins System 32 beginnt also schon bei 3 Euro.

Im Übrigen nutzt Ikea nicht bei allen Möbeln das System 32. Entsprechend kommen bei Ikea auch nicht unbedingt System-32-kompatible Scharniere oder Auszüge zum Einsatz, vermutlich aus Kostengründen. Sollten Sie Ikea-Möbel umbauen oder Teile weiternutzen wollen, sollten Sie diese vorher auf Kompatibilität prüfen.

—jom

Elektrische Enduro tunen

Die Modifikationsmöglichkeiten für die elektrische Enduro von Surron sind vielfältig. Wir zeigen, wie man sie schöner und schneller macht.

von Roman Radtke

Kurzinfo

» Karosserie-Tuning
» Mehr Leistung durch Aftermarket-Akku
» Umbau des Controllers

Checkliste

Zeitaufwand:
Ein Wochenende

Kosten:
50 bis 4.000 Euro

Material

» **Upgrade-Akku** z. B. Gruber-Parts
» **Kit for Surron Light Bee** Controller- und Tacho-Combo
» **Marzocchi Bomber 58 Federgabel**
» **600-lbs-Heckfeder**
» **Ergotec Riser Bar 70/31,8**
» **SURRON MX Fußrasten**
» **Unterfahrschutz**
» **Scheinwerfer** Supernova M99 pure plus
» **Verstärkungsstrebe**
» **Schrumpfschlauch**

Werkzeug

» **Für Motorradarbeiten ausgestatteter Werkzeugkasten**

Mehr zum Thema

» Finn Koubek und Christian Koubek, E-Bike mit Akkuschrauber-Antrieb selber bauen: Alltagstauglich durch Mods
» Carsten Wartmann, Dominik Laa, Gerd Michaelis, Clemens Verstappen, Reparatureinsätze planen: Passendes Handgepäck für Maker

Alles zum Artikel im Web unter
make-magazin.de/xwb6

Die Surron Elektro-Enduro bietet mit einem Gewicht von nur 65 Kilogramm und hervorragender Geländegängigkeit viel fürs Geld. Der mit 60 V und 40 Ah relativ große Akku ermöglicht je nach Fahrweise eine realistische Reichweite von 40 bis 60 km. Auch wenn das Fahren damit auf der Straße schon Spaß macht – richtig Freude kommt erst im Gelände auf!

Was dürfen die zwei Modelle?

Die Surron Light Bee ist in Deutschland in zwei verschiedenen Versionen erhältlich – als LBX-Modell, ohne Straßenzulassung, das bis zu 75 km/h schnell ist, und die L1E-X-Version, die für die Nutzung auf der Straße ausgelegt ist.

Für knapp 5.000 Euro wurde es unter anderem mit zusätzlichem Zubehör wie Blinkern und einer anderen Beleuchtung ausgestattet. Leider wurde auch die Höchstgeschwindigkeit auf 45 km/h gedrosselt, um die Zulassungsklasse L1e-B (zweirädriges Kleinkraftrad vergleichbar mit einem 50-cm³-Benzinmotorrad) zu ermöglichen.

Im Folgenden soll auf das Tuning der L1E-X-Version eingegangen werden, da es auf jeden Fall sinnvoll ist, die straßenzugelassene Version des Bikes zu kaufen.

Karosserie-Tuning

Die erste Umrüstung, die jedem ans Herz gelegt werden kann, ist ein Austausch der Fußrasten, da die ab Werk montierten nur wenig Halt bieten.

Wenn man schon dabei ist, die Fußrasten zu tauschen, sollte auch auf jeden Fall eine Verstärkungsstrebe zwischen den Fußrasten montiert werden, da diese sonst bei einem Sturz schnell verbiegen.

Um auch größeren Fahrern eine bequeme Sitzposition zu ermöglichen, bietet sich der Austausch des Lenkers als einfache Maßnahme an. Wer ein breiteres und stärker gebogenes Modell verbaut, fühlt sich gleich schon viel weniger wie der Affe auf dem Schleifstein, und weitere Ausfahrten verlieren ihren Schrecken. Verbaut wurde ein Ergotec Riser Bar 70/31,8, welcher extrem stabil und daher auch für bis zu 45 km/h schnelle E-Bikes freigegeben ist.

Achtung auf öffentlichen Straßen

Die Regeln für die Nutzung von getunten Elektromotorrädern auf öffentlichen Straßen sind in allen Ländern unterschiedlich. Viele Länder beginnen, härtere Regelungen einzuführen. Wer seine Maschine tunen möchte, sollte sich unbedingt genau informieren, welche Regelungen wo gelten.

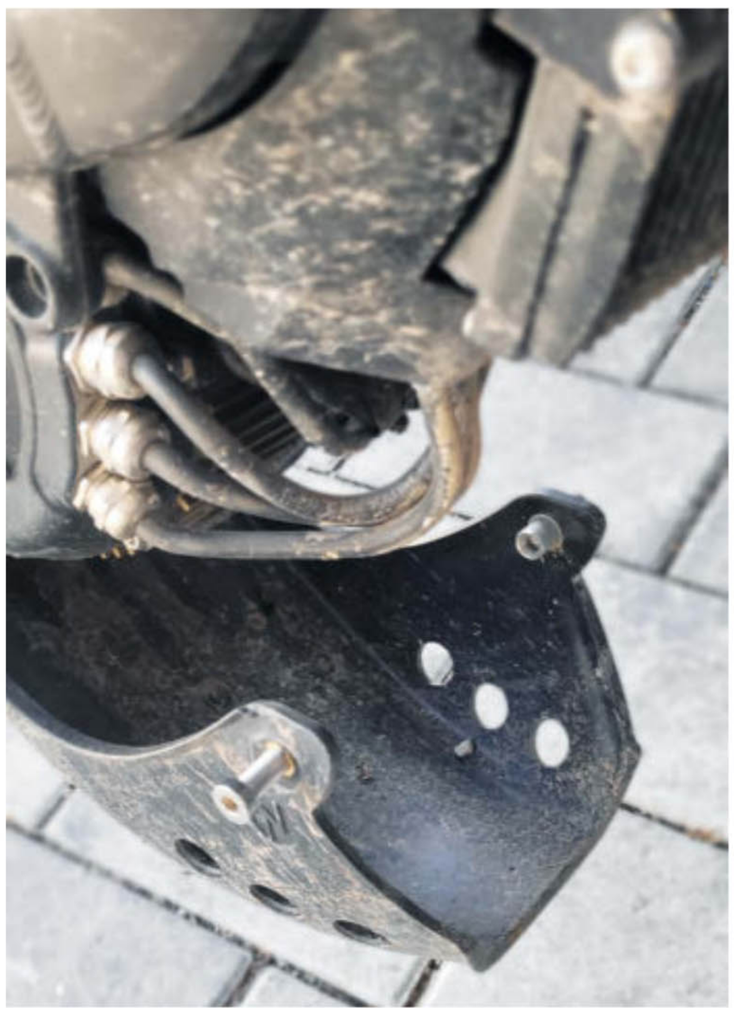
Neben dieser praktisch unverzichtbaren Modifikation sollte man – besonders wenn es ins Gelände geht – auch den vorhandenen Unterfahrschutz austauschen. Der ab Werk montierte ist aus dünnem Blech und bietet nur wenig Schutz für den Motor, falls man einmal beim Überwinden eines Hindernisses aufsetzt.

Bei den ersten Versionen des L1E wurden noch Reifen mit wenig Profil geliefert, die eine Umrüstung nötig machten, sobald man sich abseits gefestigter Straßen bewegen wollte. Der Hersteller hat hier nachgebessert: Die inzwischen montierten Reifen funktionieren sowohl auf der Straße als auch im Gelände gut.

Das Fahrwerk ist bei beiden Versionen gleich, lässt jedoch eine Menge Raum für Verbesserungen, besonders bei der Nutzung im Gelände. Ab Werk gibt es je nach Baureihe verschiedene Federgabeln und Dämpfer, welche alle mehr oder weniger ordentlich funktionieren, jedoch bei Weitem nicht das Ultimo darstellen.

Mit mehr Fläche und Spikes bieten neue Fußrasten einen deutlich sichereren Halt.

Eine neue Strebe sorgt für mehr Stabilität.



Der nachgerüstete Unterfahrschutz ist stabiler als der alte.

Der originale Heckdämpfer tut an sich ziemlich klaglos seinen Dienst. Je nach Gewicht des Fahrers und Einsatzzweck kann es jedoch sinnvoll sein, eine stärkere Feder zu verbauen, um ein Durchschlagen zu verhindern. Federn mit Federraten zwischen 500 lbs und 650 lbs sind im Zubehörhandel für unter 50 Euro erhältlich und schnell getauscht, wobei beachtet werden muss, dass bei höheren Federraten auch der Dämpfer stärker belastet wird.

Neben einer 600-lbs-Heckfeder wurde aufgrund des geplanten Einsatzes abseits der Straßen eine Marzocchi-Bomber-58-Federgabel verbaut. Mit rund 1.000 Euro sicher kein Schnäppchen, aber so modifiziert verlieren selbst größere Hindernisse und Sprünge ihren Schrecken.

Die neue Gabel ist nicht nur deutlich robuster gebaut, sie spricht auch viel besser an und lässt sich feinfühlig auf jede Anforderung einstellen.

Geschmack ist was gefällt – und die LBX-Version des Surron sieht aus wie ein zu groß geratenes Mountainbike – dynamisch, was man von der Straßenversion leider nicht unbedingt behaupten kann.

Besonders der hintere Schmutzabstreifer / Kennzeichen- und Lampenhalter wirkt deplatziert und wird daher kurzerhand demontiert.

Da Kennzeichen, Blinker und Rücklicht im Bereich der StVO vorgeschrieben sind, bietet sich für deren Montage der Griff zu einem der zahlreichen im Web verfügbaren Umbausätze an.

Da das Kennzeichen nach dem Umbau im Spritzbereich des Hinterrades liegt, sollte man es nach dem Geländeeinsatz, wenn nötig, reinigen.

Als Maker bietet sich alternativ der Eigenbau eines entsprechenden Halters an – hierzu finden sich diverse Anleitungen im Netz, wobei natürlich in jedem Fall auf eine zulässige und fachgerechte Montage zu achten ist.

Da auch die Front gewöhnungsbedürftig ist und Gabel und Lenker ohnehin schon umgebaut wurden, bot sich auch hier eine mehr oder weniger radikale Auffrischungskur an.

Um die Strecke auch bei schneller Fahrt im Dunkeln optimal auszuleuchten, wurde ein Supernova-M99-Pure-Plus-Scheinwerfer verbaut – dieser stellt so ziemlich das Optimum an E-Bike-Beleuchtung dar und hat vor allem die wichtige Zulassung für Bikes bis 45 km/h.

Um die neuen Anbauteile und die Blinker nach der Modifikation zu montieren, mussten neue Halter her. Erfreulicherweise konnte die Werkstatt vor Kurzem mit einem neuen 3D-Drucker – dem QiDi Tech X-MAX 3 – aufgerüstet werden, sodass dem Eigenbau eigener Bauteile nichts mehr im Wege steht. Damit ist es nun möglich, Offroad-taugliche Teile aus modernen Materialien wie Kohlefaser-Nylon zu drucken. Kurzerhand wurden passende Schellen für die Befestigung der Blinker sowie neue Halter für den Scheinwerfer und den Tacho gedruckt.

Neben den hier beschriebenen Modifikationen gibt es natürlich noch deutlich mehr legale Möglichkeiten, das Bike zu tunen.

Weg von der Straße

Selbstverständlich gibt es neben den für den Straßenverkehr zugelassenen Modifikationen noch weitere Tuning-Möglichkeiten, um das Fahrzeug für mehr Fahrspaß abseits der Straße zu trimmen – wobei hier die Betonung auf abseits der Straße liegt. Und – wie ein guter Freund von mir einst sagte: „Spaß kostet Geld." Wer also sein Surron leistungsfähiger und schneller machen will, muss einigermaßen tief in die Tasche greifen. Eine Alternative wäre, gleich zur größeren Maschine zu greifen. Ultra Bee und Storm Bee bieten zwar deutlich mehr Leistung, sind aber auch wesentlich schwerer und dürfen nur mit einem Motorradführerschein gefahren werden, was sie also zu keiner wirklichen Alternative macht.

Der erste Schritt des Leistungstunings ist das Aufrüsten des Bikes mit einem neuen Motorregler, denn der originale X-Controller hat eine maximale Nenndauerleistung von 2,05 kW. Die Wahl fiel auf den Nucular P24F Controller, da dieser vielversprechende Leistungsdaten hat und als komplettes Umbaukit zu einem für die gebotene Leistung recht moderaten Preis von rund 850 Euro erhältlich ist. Die mitgelieferten Teile des Kits sind von sehr guter Qualität und das große beiliegende Display macht alle Einstellungen sehr einfach. Mit einer Nennleistung von 10 kW ist er fast fünfmal so leistungsfähig wie der Original-Controller und das bei einer Spitzenleistung von bis zu 27 kW.

Welche Folgen solch ein Tuning für der Motor hat, darauf wird weiter unten eingegangen.

Wer sich den Umbau des Bikes in Echtzeit anschauen will, findet in der Online-Info einen Link zu einem YouTube-Video, das den Vorgang genau zeigt.

Für die Durchführung der Montagearbeit habe ich das Bike mittels Spanngurten an der Garagendecke angehoben. Dadurch konnte ich überall gut herankommen. Ist diese Möglichkeit nicht vorhanden, sind die Arbeiten auch mit einer Getränkekiste o. ä. zum Aufbocken des Fahrzeugs möglich. Mit dieser Alternative kann das Hinterrad ebenso frei rotieren, was besonders beim Einstellen des Controllers von Vorteil ist.

Generell sollte man sich vor so einem Umbau aber auf jeden Fall die Anleitung durchlesen. Die im Betrieb fließenden Ströme und auch die Akkuspannung sind nicht zu vernachlässigen.

Wie in dem verlinkten Installationsvideo in der Online-Info vorgeschlagen, lässt sich mit

Die neue Gabel ist sehr viel stärker als das Originalteil.

dem Einsatz eines Akku-Schlagschraubers viel Zeit sparen. Beim späteren Anziehen der Schrauben mit einem Schlagschrauber sollte man jedoch vorsichtig vorgehen, da diese meist genug Drehmoment besitzen, um die Schrauben zu beschädigen.

Sinnvoll für die Installation des Controllers ist zudem ein kleiner Drehmomentschlüssel, da die Schrauben für die elektrischen Anschlüsse am Controller selbst nur mit 6 Nm angezogen werden sollten. Auch für die Montage anderer Komponenten wie dem Vorbau, den Bremsscheiben etc. bietet sich der Einsatz eines Drehmomentschlüssels an. Um zu verhindern, dass die Kontakte im Laufe der Zeit korrodieren oder einen Wackelkontakt bekommen, können sie vor der Montage dünn mit einem korrosionshemmenden Kontaktspray eingesprüht werden. Ist man schon dabei, kann man auch gleich die Anschlüsse des Akkus so behandeln.

Auch der Einsatz eines Schraubensicherungslacks für die Metall-auf-Metall-Verbindungen ist sinnvoll, damit sich die Schrauben nicht im Gelände durch Vibrationen lösen. Aber Achtung: Diese Mittel sollten nicht an den elektrischen Kontakten oder an Kunststoffteilen eingesetzt werden.

Nachdem man den Akku entfernt hat, sollte man auf jeden Fall einmal die „Zündung“ einschalten, um die im Controller verbauten Kondensatoren ganz zu entladen – sicher ist sicher!

Generell ist es auch sehr hilfreich, von den einzelnen Schritten Fotos anzufertigen – besser haben, als irgendwann doch einmal brauchen.

Die Verkabelung der Steuerleitungen gelingt sehr einfach, weil alle Stecker kodiert sind und nur an den dafür vorgesehenen Platz passen. Um Verwirrung vorzubeugen: Ein Stecker bleibt nach der Montage des neuen Controllers unbelegt. Da die Kontakte offen liegen, sollte man diese mit etwas Elektroklebeband vor den Elementen schützen.

Der Tacho der Straßenversion unterscheidet sich von dem der Geländeversion, lässt sich aber genauso einfach durch Entfernen der Halteschrauben demontieren. Hat man nichts

Eine neue Heckfeder schützt den Fahrer bei ruppiger Fahrt.

Das Heck der originalen Straßenversion.

an Vorbau und Gabel verändert, kann der neue Tacho wie im Video beschrieben in den Vorbau geklemmt werden.

Bei Verwendung des gedruckten Lampenhalters ist die Montage des neuen Tachos besonders einfach – kurzerhand den mitgelieferten Halter mit Schraube und Mutter im mittigen Loch befestigen und schon hat der Tacho einen sicheren Platz.

Wenn man später beispielsweise aus rechtlichen Gründen die Option haben will, den Controller wieder zu demontieren und den zugelassenen Controller einzubauen, sollte man nicht den mitgelieferten Schrumpfschlauch an den Motorverlängerungen verwenden, da dieser innen mit Heißkleber beschichtet ist und sich nur extrem schwer wieder entfernen lässt. Nachteilig ist, dass der Anschluss so nicht hundertprozentig wasserdicht ist, was man aber verschmerzen kann.

Folgt man der Installationsanleitung, sollte es keine Probleme geben. Das Einzige, was etwas knifflig wird, ist, alle Kabel unterzubringen, ohne beim Zusammenbau etwas einzuklemmen.

Besonders der Platz zwischen Hupe und Montagerahmen ist sehr eng, wobei das nur auf die straßenzugelassene Version zutrifft. Man muss hier besonders darauf achten, aber keine Sorge, es passt!

Beim Anziehen des Montagerahmens sollte man sich Zeit lassen und vielleicht auch einen Helfer dabei haben, denn bis alle Leitungen an ihrem Platz sitzen, ist ein wenig Feinarbeit notwendig. Es kann auch hilfreich sein, die Leitungen mit Kabelbindern zu bündeln, wobei darauf zu achten ist, dass genügend Spielraum zum Montieren des neuen Controllers bleibt.

Der Anschluss der Leitungen an den Controller ist einfach und straight forward. Am besten montiert man zuerst die Motorleitungen, dann die Steuerleitungen und zuletzt die Batteriekabel. Wichtig ist, dass die Schrauben nicht zu fest angezogen werden, da sonst die Terminals aus dem Kunststoff ausreißen. Hierzu ist – wie erwähnt – der Einsatz eines Drehmomentschlüssels optimal. Alternativ benutzt man einen kurzen Inbusschlüssel, da man mit diesem nicht so leicht ein zu großes Drehmoment aufbringen kann.

Wer will, kann jetzt noch die Leitungen mit Kabelbindern sichern, bevor sie in den Montagerahmen gedrückt werden.

Nachdem alles wieder zusammengebaut ist, sind die mechanischen Arbeiten auch schon abgeschlossen.

Für den letzten Schritt – das Einstellen des Controllers – muss das Hinterrad frei drehen können. Hat man den Akku wieder angeschlossen (Vorsicht – ist die Zündung aus?), wird das Bike entsprechend aufgebockt. Da der Controller bereits für das Surron und den Standard-Akku vorprogrammiert ist, sind nur wenige Klicks (wie im Video beschrieben) nötig, um den Umbau abzuschließen. Aber Achtung: Das Hinterrad wird sich während der noch zu tätigenden Einstellungen drehen!

Folgt man dieser Anleitung Schritt für Schritt, ist man in wenigen Stunden mit dem Umbau fertig, besonders da der Controller bereits für den Standard-60-V-Akku vorprogrammiert geliefert wird.

Um eine derartige Leistung an den Motor liefern zu können, muss diese auch irgendwo herkommen. Leider ist der im Original verbaute Akku nicht auf derartige Leistungen ausgelegt, denn das im Akku verbaute BMS (Battery Management System) schaltet bei einer Stromentnahme von mehr als 90 Ampere ab, um die einzelnen Zellen des Akkus zu schützen. Da der Akku 60 V liefert, ist die Leistung somit auf eine Spitzenleistung von etwa 5,4 kW begrenzt (60 Volt x 90 Ampere = 5400 Watt).

Eine Möglichkeit, mehr Leistung aus dem Standard-Akku zu holen, besteht darin, das BMS zu überbrücken. Zu diesem Zweck gibt es verschiedene Kits, die im Prinzip aus dicken Drahtbrücken bestehen. Der Nachteil dieser Methode ist, dass auch die Schutzfunktionen des integrierten BMS überbrückt werden, was im wahrsten Sinne des Wortes brandgefährlich ist. Aus diesem Grund kann man von dieser Modifikation nur abraten!

Wer auf Nummer sicher gehen will, kommt daher nicht um ein Akku-Upgrade herum,

Das neue Heck (nach einer wilden Fahrt durchs Gelände).

Die neue Front macht ordentlich was her.

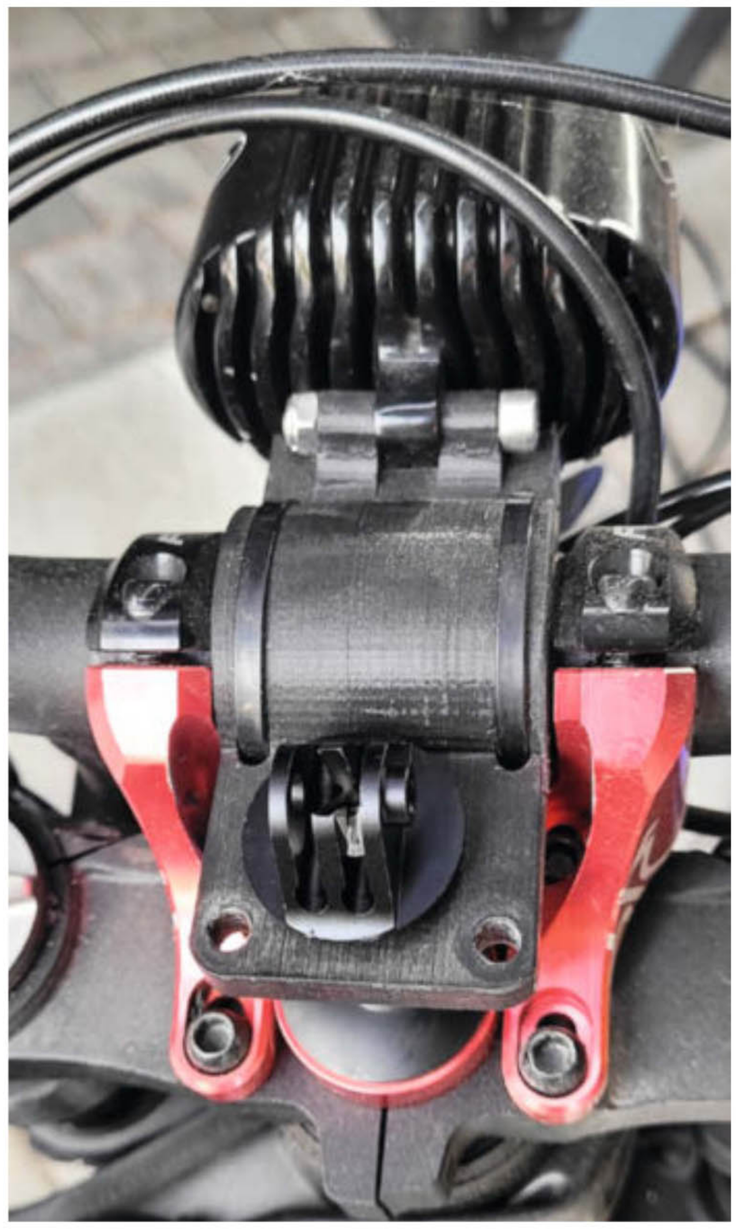
auch wenn der Controller allein, ohne Akku-Umbau, schon ein Leistungsplus von 20 bis 30 Prozent bietet.

Auch hier hat sich der Markt bereits auf den Wunsch nach mehr Leistung eingestellt. Viele Hersteller bieten verschiedenste Upgrade-Akkus für die Light Bee an: von Long-Range-Akkus mit 60 V und 65 Ah bis hin zu Race-Akkus mit satten 90 V sollte für jeden etwas dabei sein. Die grundlegende Frage ist dabei, ob man nur mehr Leistung, mehr Reichweite oder beides benötigt. Mehr Kapazität bedeutet mehr Reichweite, mehr Spannung sorgt für höhere Leistung, meist 72 V anstelle der originalen 60 V, und eine größere Strombelastbarkeit. Anzumerken ist dabei, dass ein Akku mit höherer Spannung bei gleicher Kapazität – bei gleicher Fahrweise – auch eine größere Reichweite ermöglicht, da die Stromaufnahme bei gleicher Leistung proportional sinkt.

Wichtig ist hier die richtige Beratung, denn ein neuer Akku schlägt leicht mit 2.000 Euro und mehr zu Buche. Wer sich als Maker einen Akku selbst bauen will, hat sich leider zu früh gefreut. Rechnet man die Kosten für die Zellen – bei einem großen Akku oft 20 Zellen in Reihe und 15 oder mehr parallel – zusammen, so kommt man bei geeigneten Hochleistungszellen bereits auf reine Materialkosten von deutlich über 1.500 Euro. Da ein geeignetes, hochstromfähiges BMS – selbst wenn man es direkt in China bestellt – auch mit deutlich über 100 Euro zu Buche schlägt, ist die Ersparnis marginal.

Darüber hinaus ist der Bau eines derart großen Akku-Packs kein Spaß. Sollte es zu einem Kurzschluss kommen, verschweißen sich die Zellen sofort und fangen nach kurzer Zeit Feuer oder explodieren sogar.

Da es mir wichtig war, den Akku aus Deutschland zu beziehen, entschied ich mich nach einigen Vorgesprächen für einen Akku, welchen Gruber Parts neben vielen anderen Tuning-Teilen im Angebot hat. Link dazu in der Online-Info.



Der Halter für Lampe und Tacho sieht nicht nur robust aus – er ist aus kohlefaserverstärktem Nylon gedruckt und damit so gut wie unzerstörbar.

Selbst wenn der neue Controller im Vergleich zu einem Copter ESC riesig wirkt, ist er für seine Leistung winzig.

So werden die Kondensatoren nach Entfernen des Akkus sicher entladen.

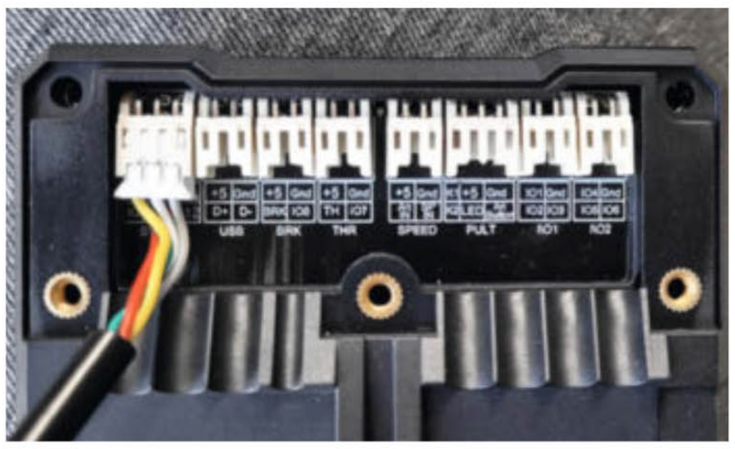
Auch eine Dokumentation darüber, welche Schrauben und Teile woher kommen, ist immer sinnvoll.

Verbaut sind in diesem Akku hochwertige 21700 Li-Ion-Zellen von Samsung, was eine Strombelastbarkeit des Packs von maximal 300 A ermöglicht, wobei die Zellen selbstverständlich durch ein hochwertiges BMS vor Überstrom und Unterspannung und so ziemlich alle anderen denkbaren Fehlerfälle geschützt sind.

Ein Nachteil der größeren Akkus ist, dass sie nicht ohne Modifikation in den Originalschacht passen. Sie sind meist etwas länger als das Original, was aber leicht durch einen

Der kleine Stecker bleibt unbelegt.

So sitzt der neue Tacho formschön und bombenfest.



Da das System erweiterbar ist, bleiben die meisten Anschlüsse am neuen Tacho unbelegt.

Wenn man die Kabel der Hupe möglichst weit nach oben biegt, sollte der Montagerahmen problemlos passen.

Will man den Controller später wieder entfernen können, empfiehlt sich die Verwendung eines anderen Schrumpfschlauchs.

längeren Bügel am Deckel des Akkuschachts ausgeglichen werden kann. Aufgrund der höheren Akkuspannung benötigt man selbstverständlich auch ein anderes Ladegerät.

Auch der Controller muss für die höhere Akkuspannung umkonfiguriert werden, was sich aber relativ leicht über das mitgelieferte Display erledigen lässt, wobei auch hier wieder diverse hilfreiche Videos auf YouTube zu finden sind.

Wer keine Kosten und Mühen scheut, um Controller und Akku umzubauen, wird mit schier unglaublicher Performance belohnt. Das Grinsen im Gesicht, das beim Beschleunigen entsteht, ist nicht nur ein aktives, sondern auch ein passives, denn neben der gesein geliebtes Surron und den Führerschein verzichten muss. Rechnet man die Kosten für ein Gutachten, das Verfahren und unter Umständen einen Anwalt und die Strafe zusammen, kommt man fast immer auf mehrere Tausend Euro.

Der schwerwiegendste Punkt ist jedoch, dass der Versicherungsschutz entfällt. Verursacht man einen Unfall, kann die Versicherung die Leistung verweigern und man muss die verursachten Kosten selbst tragen.

In anderen Ländern ist die Rechtsprechung zum Teil ähnlich, sodass man sich auch dort auf jeden Fall informieren muss, was erlaubt ist. Problemlos hingegen ist das Betreiben das Surrons auf einer Rennstrecke oder einem geeigneten abgesperrten Privatgelände mit Einwilligung des jeweiligen Eigentümers.

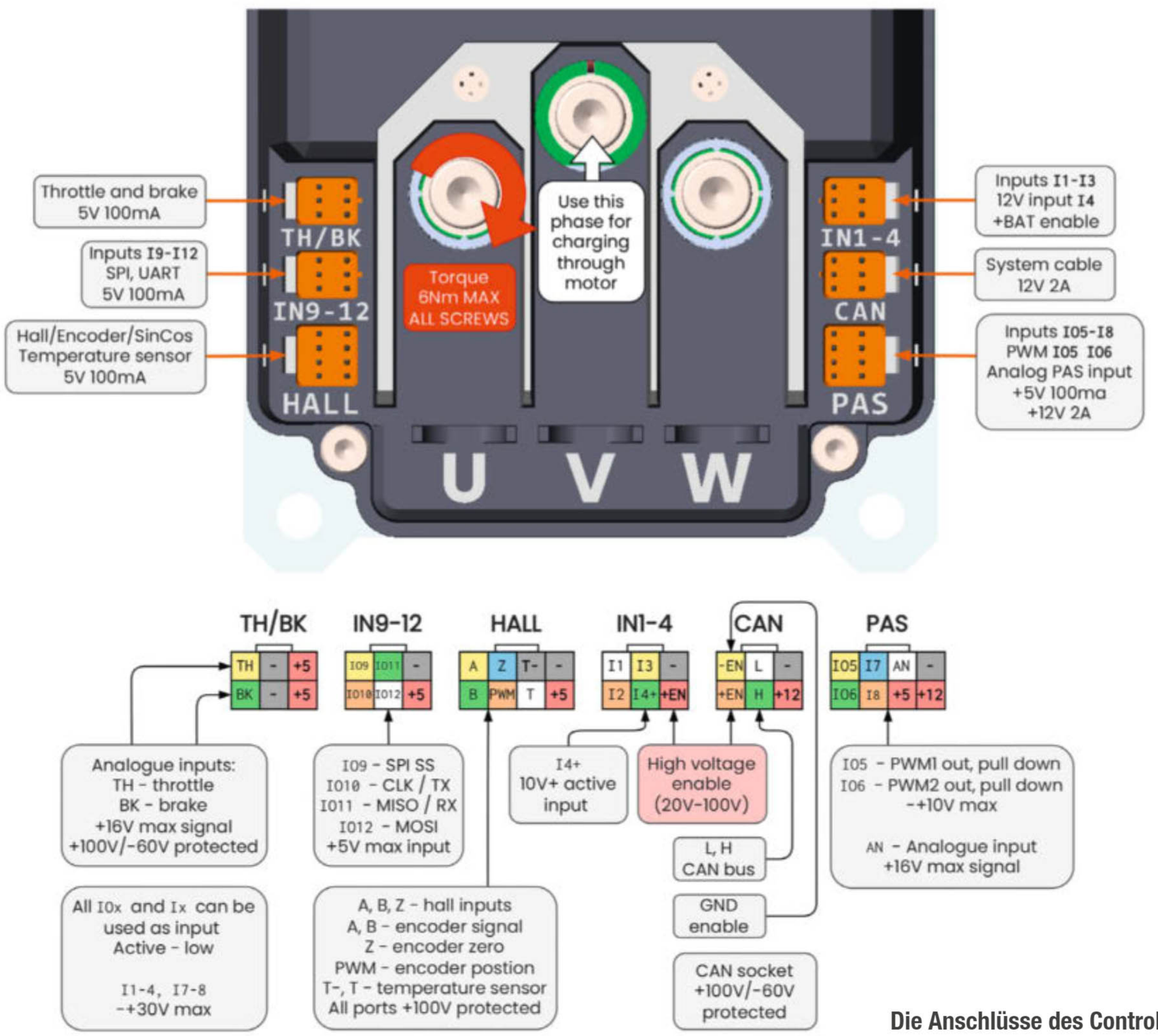
Abseits der Straße!

Der Gesetzgeber hat bei allen Umbauten am Motorrad einiges mitzureden! Natürlich kann an dieser Stelle keine Rechtsberatung erfolgen, daher muss jeder vor jedem noch so kleinen Umbau selbst prüfen, inwieweit dieser zulässig ist. Grundlegend gilt: Jedes Teil, welches das Fahrverhalten beeinflussen kann oder sicherheitsrelevant ist, braucht eine Zulassung und muss gemäß der StVO montiert werden.

Was leistungssteigernde Maßnahmen angeht, ist die Gesetzeslage in Deutschland darüber hinaus ziemlich rigoros: Illegales Tuning stellt fast immer nicht nur eine Ordnungswidrigkeit, sondern eine Straftat dar.

Wird man erwischt, kommen schnell einige Kosten zusammen, davon abgesehen, dass man zumindest kurz- bis mittelfristig auf

Mit etwas Geduld findet alles seinen Platz.



Die Anschlüsse des Controllers

fühlten Freude zieht der unglaubliche Schub die Mundwinkel nach hinten. Ist alles richtig eingestellt, sind Zeiten von 0 auf 100 km/h von deutlich unter 9 Sekunden möglich, was für ein so handliches Gefährt sensationell ist.

Zu beachten ist bei derart extremem Tuning, dass der Originalmotor auf Dauer nicht auf derartige Leistungen ausgelegt ist. Dauervollgas von mehr als wenigen Sekunden führt schnell zum Überhitzen und damit auch zum Tod des Motors – ein kurzer, teurer Spaß. Kurze Sprints sind hingegen kein Problem, auch nicht bei Leistungen, die deutlich über den als Belastbarkeit für den Motor angegebenen 6 kW liegen.

Auch hier bietet der Tuning-Markt Lösungen: Hersteller von Zubehör bieten inzwischen Motoren mit Dauerleistungen um die 20 kW an.

Videos auf YouTube, die zeigen, was alles möglich ist, scheinen vielversprechend, aber da sich meine Ambitionen als Rennfahrer in Grenzen halten, kann ich hier nicht aus eigener Erfahrung sprechen. Und auch wenn ein schnelles Surron jede Menge Spaß macht, möchte ich mit dem Bike nicht deutlich über 100 km/h fahren, denn dafür ist es auch nach diversen Upgrades an Fahrwerk, Bremsen und anderen Komponenten keinesfalls ausgelegt.

Ergibt das Sinn?

Wenn man davon absieht, dass der Gesetzgeber bei derartigen Maßnahmen ein gehöriges Wörtchen mitzureden hat, ist ein getuntes Surron ein Spaßgerät sondergleichen. Ein Zweirad mit der Leistung einer 125er, das nur etwa 60 kg wiegt, bietet ein wirklich unvergleichbares Erlebnis: Beschleunigung ohne Ende und besonders im Gelände ein Handling, wie es besser nicht sein kann.

Mit dem hier beschriebenen Tuning hat man mehr als genug Leistung und erreicht je nach Programmierung des Controllers und Übersetzung Höchstgeschwindigkeiten von 100 km/h und mehr und das bei jeder Menge Drehmoment vom Start weg.

Und selbst ohne Umrüstung ist das Surron als Nahverkehrsmittel wirklich extrem praktisch, umweltschonend und kostengünstig zu betreiben.

Nach den 5.000 € für die Anschaffung einer Surron ist das Limit für Folgeausgaben unbegrenzt: hier neue Fußrasten, dort eine andere Beleuchtungsanlage, bessere Bremsen, vielleicht noch ein kleines Fahrwerksupgrade. Selbst ohne Leistungstuning liegt man schnell bei zusätzlichen Kosten von einigen Tausend Euro. —*das*

Auch dank des sehr schön gemachten Montagerahmens sieht das Endprodukt sehr sauber aus.

Ein Veranstaltungsformat der Make: make-magazin.de

Maker Faire®

Das Format für Innovation & Macherkultur

Die nächsten Events

HANNOVER Maker Faire 17.-18. August 2024

8. SEPT 2024 Maker Faire SCHWEDT/O. UCKERMARK

Maker Faire 12. Oktober 2024 SINDELFINGEN

Maker Faire 9. November 2024 SALZBURG

... weitere folgen.

maker-faire.de

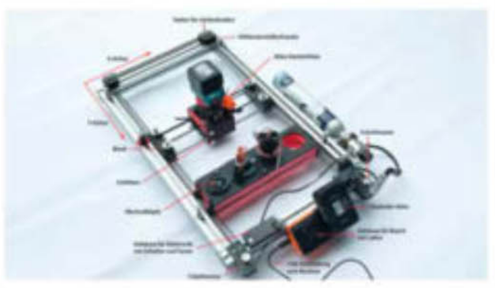
Make: Online

Beliebt auf heise+

Keller trocknen: Das Taupunktlüftungssystem im Smart Home betreiben

Nach dem Hochwasser bleiben oft die Kellerwände feucht. Da hilft das selbstgebaute Taupunkt-Lüftungssystem, hier in der Smart-Home-Version.

23.05.2024 66 | Make Magazin



CNC-Rahmen für die Oberfräse aus Aluprofil und 3D-Druck-Teilen bauen

Dieser handliche X-Y-Positionierer für die Fräse ist einfach nachgebaut – dank fertig auf Länge bestellter Aluprofile und Verbindungsteilen aus dem 3D-Drucker.

05.06.2024 13 | Make Magazin

Bastel-Projekt: Elektroniklabor für den Küchentisch

Dieser Koffer vereint alles, was man für kleinere Elektronikprojekte oder Reparaturen benötigt. Einfach aufklappen und loslegen. Ein Bauvorschlag zum Anpassen.

15.05.2024 | Make Magazin

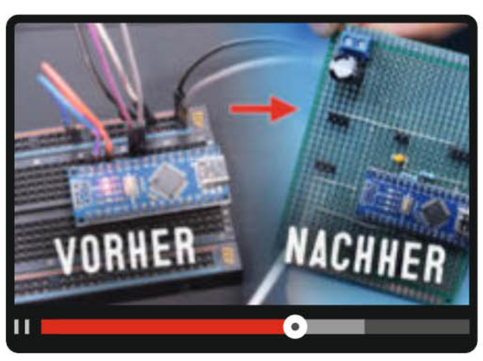
So kommen Sie als Make-Abonnent an Artikel hinter der Paywall von heise+: https://heise.de/-7363373

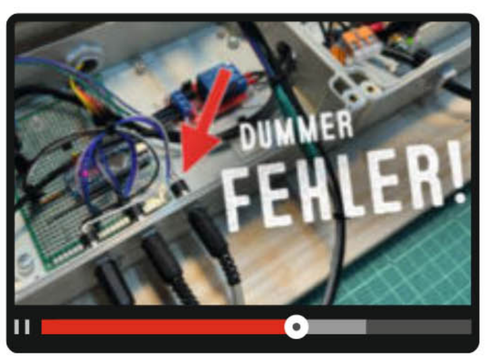
Taupunkt-Lüfter gegen feuchte Keller!

Auf unserem YouTube-Kanal gibt es eine brandneue Serie zu einem Thema, das derzeit sicherlich viele Menschen betrifft: der Taupunkt-Lüfter. Er hilft dabei, effizient zu lüften und so zum Beispiel feuchte Kellerräume trockenzulegen. Das Projekt stammt aus der 01/2022 und Johannes beschäftigt sich in der Videoreihe mit dem Nachbau des Projekts. Er zeigt den Bauprozess des Lüfters bis hin zu den Fehlern, die ihm unterlaufen sind. Außerdem hat er die Fragen, die sich unter den Videos angesammelt haben, in einem Video gebündelt als FAQ beantwortet. —*dus*

▶ www.youtube.com/@MakeMagazinDE

Weitere aktuelle Videos:





Bleib informiert:

www.make-magazin.de

@makemagazinde

Instagram: @makemagazinde
@makerfairedeutschland

Facebook: @makemagazin.de

X (Twitter): @MakeMagazinDE

TikTok: @makemagazinde

Bluesky: @makemagazin.de

WhatsApp Channel:
Make Magazin Deutschland

GitHub: MakeMagazinDE

Threads: @makemagazinde

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

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

Oder 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

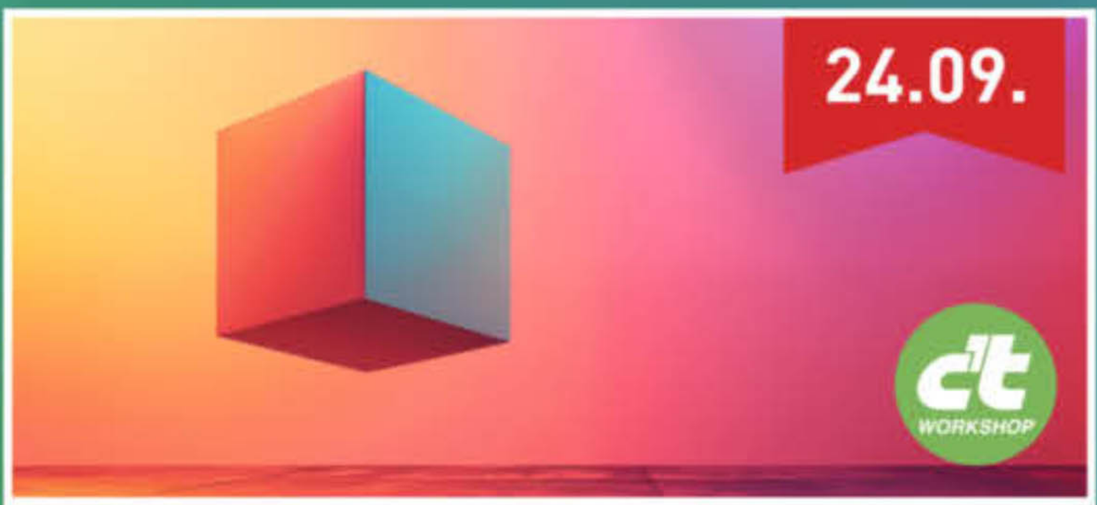
WIR TEILEN KEIN HALBWISSEN
WIR SCHAFFEN FACHWISSEN

Node.js intensiv

Der Workshop bietet einen umfassenden Einstieg in die Entwicklung serverseitiger Anwendungen mit Node.js. Sie lernen die wichtigsten Basics, die Architektur und Best Practices kennen.

Domänenmodellierung mit TypeScript

In diesem Workshop für (Web-)Entwickler und Product Owner werden moderne, kollaborative Methoden wie Event-Storming untersucht und die Domäne mithilfe des TypeScript-Typsystems modelliert.



3D im Web

Sie lernen Grundlagen und Best Practices für die Umsetzung interaktiver 3D-Szenen im Web. Von Licht, Kamera und 3D-Objekten bis hin zu Bibliotheken wie Three.js und A-Frame.

React 19 – Wichtige Neuerungen an einem Tag kennenlernen

Der Workshop bietet einen Überblick über die wichtigsten Neuerungen, reduziert die Einarbeitungszeit und unterstützt bei der Bewertung neuer Features.

Kluge Strukturen für Microsoft 365 entwickeln

Lernen Sie in dem Workshop, wie Sie gemeinsam mit Ihrem Team Leitlinien entwickeln, um in Zukunft das volle Potenzial für die Zusammenarbeit auszuschöpfen.

c‘t <webdev>

In Köln erwarten Sie auf der englischsprachigen Konferenz drei Tage voller Wissen, Einblicke, Spaß und eine großartige Community von Entwicklern. Über 40 Talks und ganztätige Workshops.

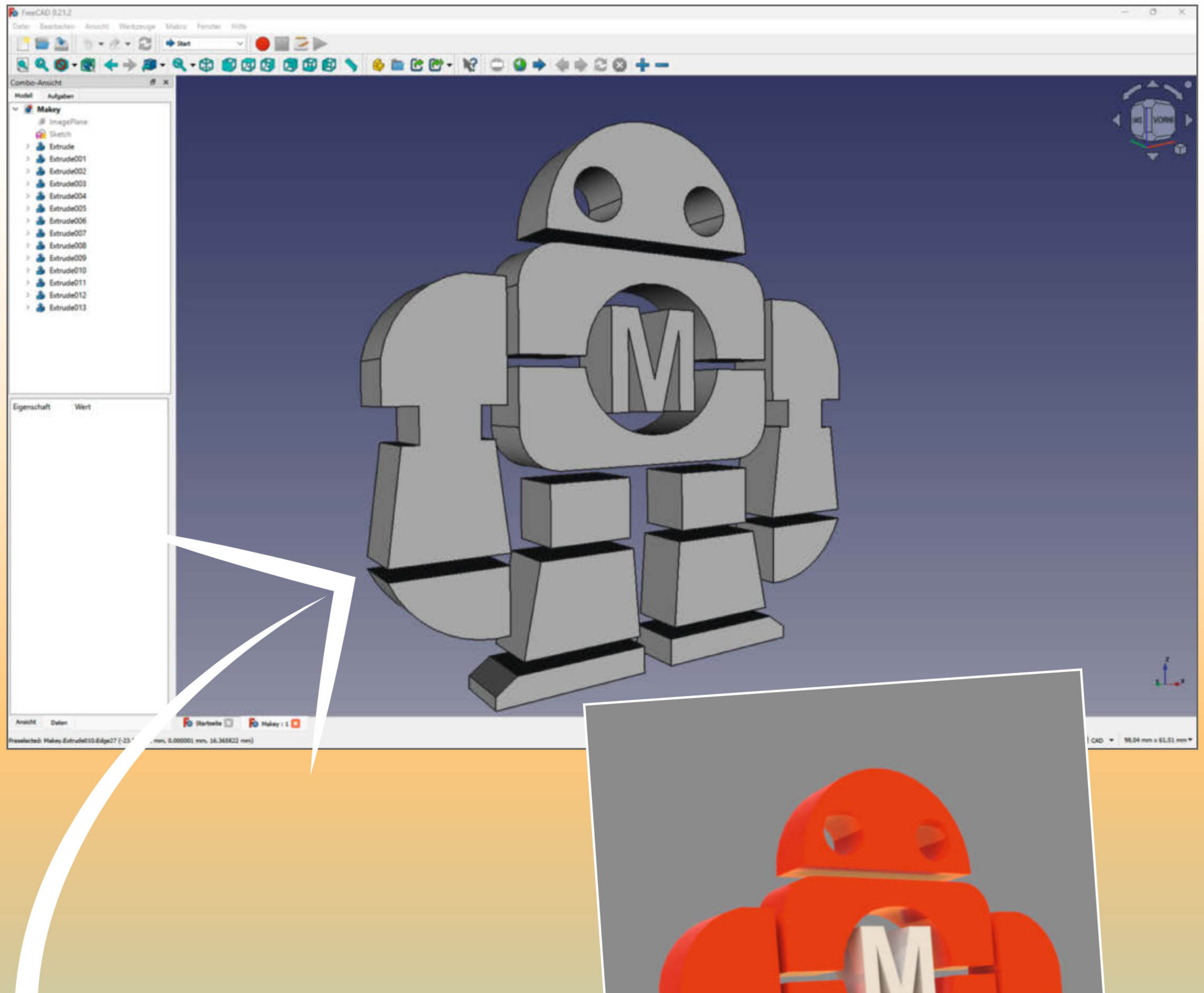
Sichern Sie sich Ihren Frühbucher-Rabatt:

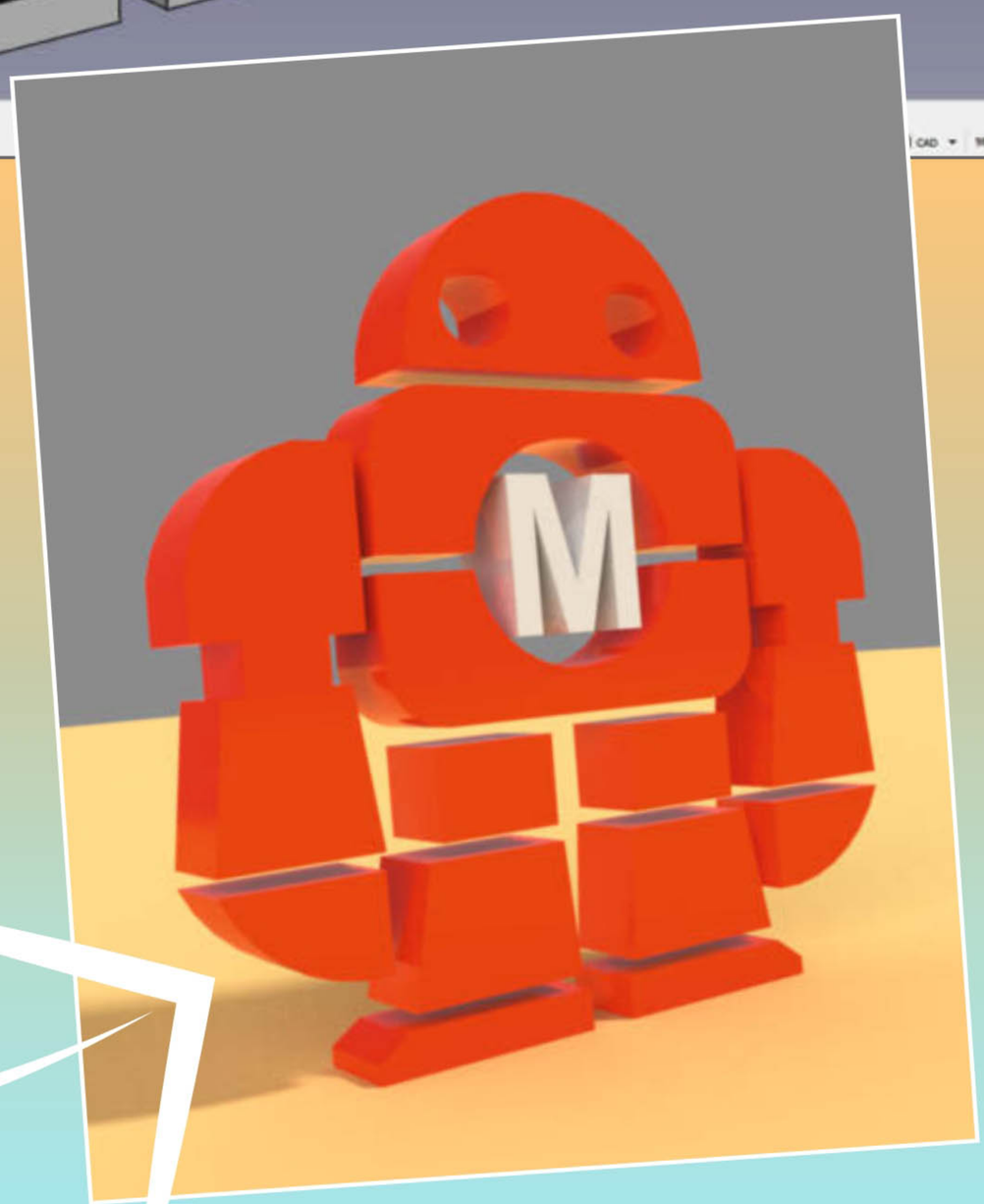
heise.de/ct/Events

Rendern mit FreeCAD

Mit der neuen Render-Workbench in der aktuellen Version der kostenlosen Konstruktionssoftware können Sie fotorealistische Bilder Ihrer FreeCAD-Projekte erzeugen. So gelingen die Installation und Integration verschiedener externer Rendering-Engines.

von Matthias Mett





Wenn man als Maker ein neues Projekt am Rechner in 3D entwirft, um es hinterher mit dem 3D-Drucker, dem Lasercutter, der Fräse oder klassisch nach Plan und Maßen anzufertigen, ist eine realistisch gerenderte perspektivische Ansicht als Zwischenstufe nützlich. Denn auf dem computergenerierten Bild sieht man sofort: Hat das geplante Gehäuse die richtigen Proportionen? Sind die geplanten Materialien stilecht oder ein Stilbruch? Springen Bedienelemente wie Knöpfe oder Schalter gleich ins Auge oder muss man sie suchen? Gefällt der Entwurf den anderen, falls es sich um ein Gemeinschaftsprojekt handelt? Nicht zuletzt ist ein solches Rendering für manche Maker eine Belohnung zwischendrin für die aufwendige Konstruktionsarbeit und gleichzeitig eine Motivation für die vielen Stunden Arbeit in der Werkstatt, die noch vor einem liegen ...

Klar, FreeCAD kann prinzipiell schon länger schicke Renderings von 3D-Entwürfen produzieren, die man in dieser Software konstruiert hat. Dafür war bis Version 0.20 die Raytracing-Workbench standardmäßig installiert. Mit ihr sendete man ein FreeCAD-Projekt an einen externen Renderer wie POV-Ray, der es dann renderte und das erzeugte Bild an FreeCAD zurücklieferte, wo es angezeigt wurde. Wollte man jedoch z. B. eigene Einstellungen für eine Szene mit Kamera und Licht vornehmen, musste man zuerst eine Textdatei exportieren und diese dann in POV-Ray weiter bearbeiten. Dort konnte man dann die Szene rendern und das Bild speichern. Alle weiteren Einstellungen fanden damit außerhalb von FreeCAD statt.

Kurzinfo

- **Fotorealistische Bilder direkt aus CAD-Daten erzeugen**
- **Umgang mit der Render-Workbench und externen Rendering-Engines: POV-Ray, LuxCoreRender, OSPRay, Cycles**
- **Szenenvorlagen, Material und Texturen verwenden**

Checkliste

Zeitaufwand:
etwa 2 Stunden

Kosten:
keine

Konstruieren:
CAD, aber keine Vorkenntnisse erforderlich

Computer:
Desktop oder Notebook mit Windows, macOS oder Linux

Mehr zum Thema

- Eine verlinkte Liste aller FreeCAD-Artikel aus der Make gibt es online unter der Kurz-URL.

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

Rendern aus FreeCAD

Die in der FreeCAD-Version 0.21 neue Render-Workbench hat nun die alte Raytracing-Workbench abgelöst. Darin kann man Einstellungen fürs Erstellen von Szenen oder für die Materialien direkt in FreeCAD vornehmen. Geblieben ist aber, dass nach wie vor ein externer Renderer das Bild erzeugt und es fertig an FreeCAD zurücksendet, von wo heraus man es speichern kann. Wie das mit unterschiedlichen Renderern geht, zeigen wir in diesem Artikel ganz praktisch.

Wir verwenden hier FreeCAD in der Version 0.21.2, in der die neue Render-Workbench zwar (noch) nicht Bestandteil der Standardinstallation ist, sie lässt sich aber mit dem Add-on-Manager einfach installieren. Dazu rufen Sie in der Menüleiste Werkzeuge/Add-on-Manager auf. Stellen Sie sicher, dass oben die Ausklappliste auf „Arbeitsbereiche“ als Filter steht und geben Sie rechts oben im Filter-Feld das Suchwort „Render“ ein. Klicken Sie auf das Ergebnis mit der Bezeichnung „Render“, wobei die Ansicht auf die GitHub-Seite des Projekts wechselt. Aber keine Bange: Mit einem Klick auf den Installieren-Button rechts oben können Sie nun die Workbench ganz einfach in Ihr FreeCAD integrieren. Nach einem Neustart der Software finden Sie die Render-Workbench in der Ausklappliste zur Auswahl der Arbeitsbereiche wieder, wo man zu ihr umschalten kann.

Start the Engines!

Die eigentlichen Rendering-Engines sind nicht Teil von FreeCAD, man muss sie zuerst separat herunterladen und installieren. Die Workbench unterstützt sechs verschiedene Engines. In der Render-Werkzeugleiste finden sich auf der linken Seite für jeden Renderer Projektvorlagen, die man dort einem Projekt hinzufügen kann – erreichbar über eine Ausklappliste mit den Icons der einzelnen Renderer; sie steht standardmäßig auf dem Stern, der für die Appleseed-Engine steht (Bild 1). Auf der rechten Seite der Werkzeugleiste erreicht

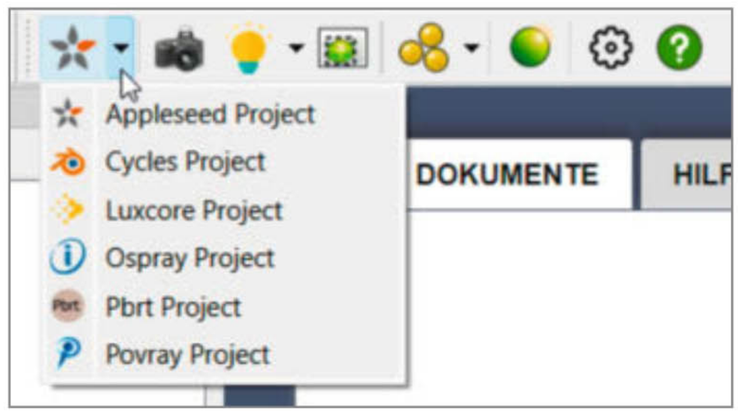


Bild 1: Die Render-Werkzeugleiste mit der Ausklappliste für die Projektvorlagen

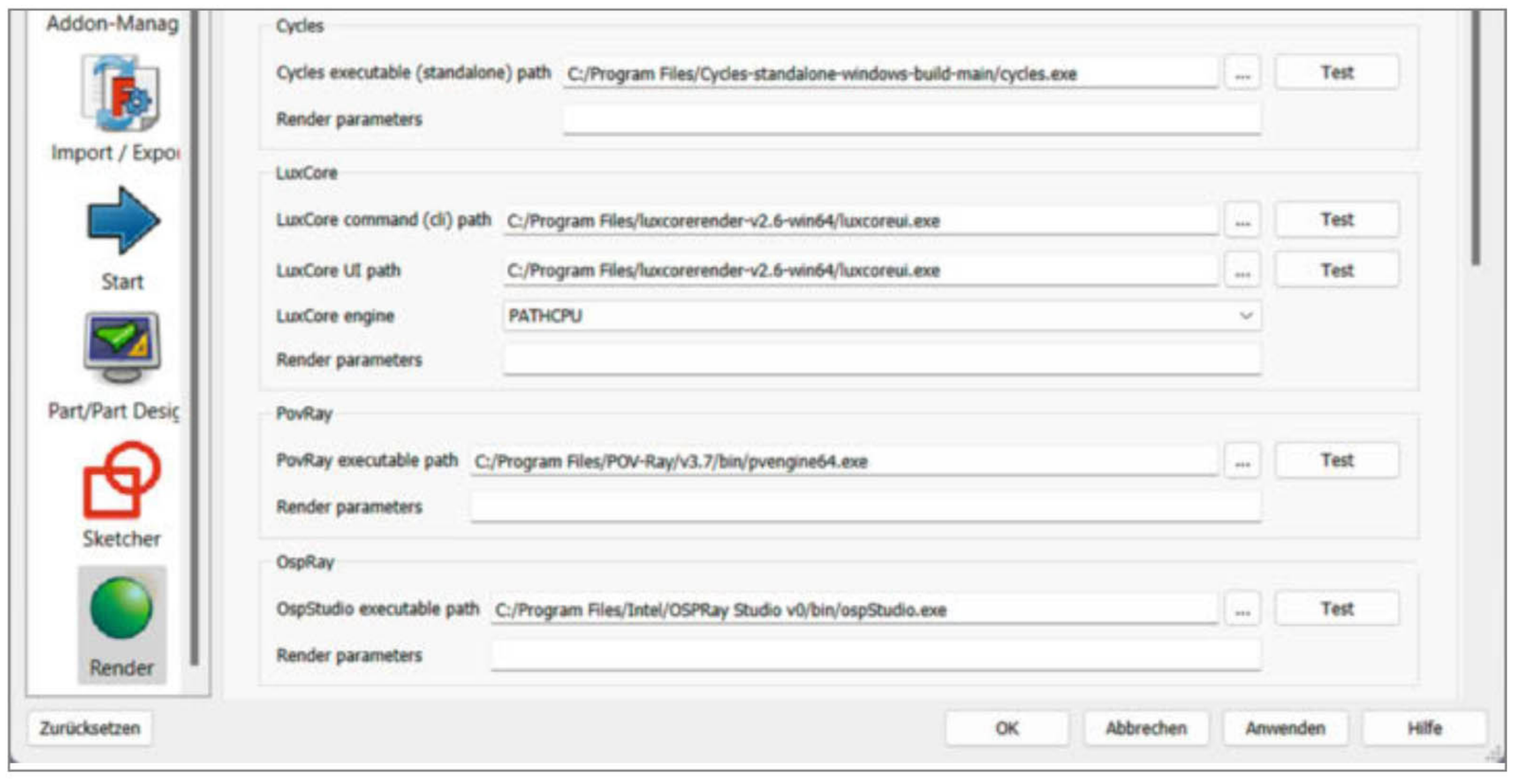


Bild 2: Render-Einstellungen mit Installationspfaden für die vier Engines, die wir für diesen Artikel ausprobiert haben

Alle FreeCAD-Projekte aus Make als PDF

Seit 2020 zeigen wir in loser Folge den praktischen Umgang mit der kostenlosen Software FreeCAD. Den Großteil der bisher in Make erschienenen FreeCAD-Artikel unseres Autors Matthias Mett bekommt man auch als elektronisches Dossier zusammengefasst. Auf 58 Seiten werden dort die Konstruktionen fünf kleinerer und größerer Projekte vom Stempel bis zum Raspberry-Pi-Gehäuse Schritt für Schritt und Klick für Klick beschrieben; zusätzlich gibt es insgesamt zwei Stunden Videotutorials direkt am digitalen Zeichenbrett.

Das Dossier ist im heise shop für 7,90 Euro zu kaufen, Make-Abonnenten zahlen nur 4,90 Euro. Am einfachsten finden Sie das Dossier über diesen Link:

▶ **make-magazin.de/xfgh**

man über das Zahnrad-Symbol die Einstellungen, in denen man die lokalen Installationspfade zu den externen Rendering-Engines angeben kann.

Im Prinzip benötigen Sie nur einen der möglichen Renderer. Wir haben für diesen Artikel natürlich mehrere Engines ausprobiert, namentlich Cycles (vielleicht manchen aus Blender bekannt), LuxCoreRender, OSPRay sowie POV-Ray, und beschreiben für jeden den Installationsvorgang. Alle Renderer stehen unter Open-Source-Lizenz und sind kostenlos verfügbar.

Erst mal verzichtet haben wir auf pbrt, denn dieser Renderer liegt nur im Sourcecode vor, daher müsste man die Quelldateien zuerst kompilieren. Bei Appleseed öffnete die Render-Workbench in unseren Versuchen zwar Appleseed Studio, startete aber nicht automatisch das Rendern. Dies könnte man manuell anstoßen, beispielsweise mit der Taste F5. Das erzeugte Bild gab Appleseed Studio jedoch nicht an FreeCAD zurück, weswegen wir die Arbeit mit diesem Renderer ebenfalls nicht weiterverfolgt haben. Die Download-Links zu allen unterstützten Rendering Engines finden Sie über die URL in der Kurzinfo.

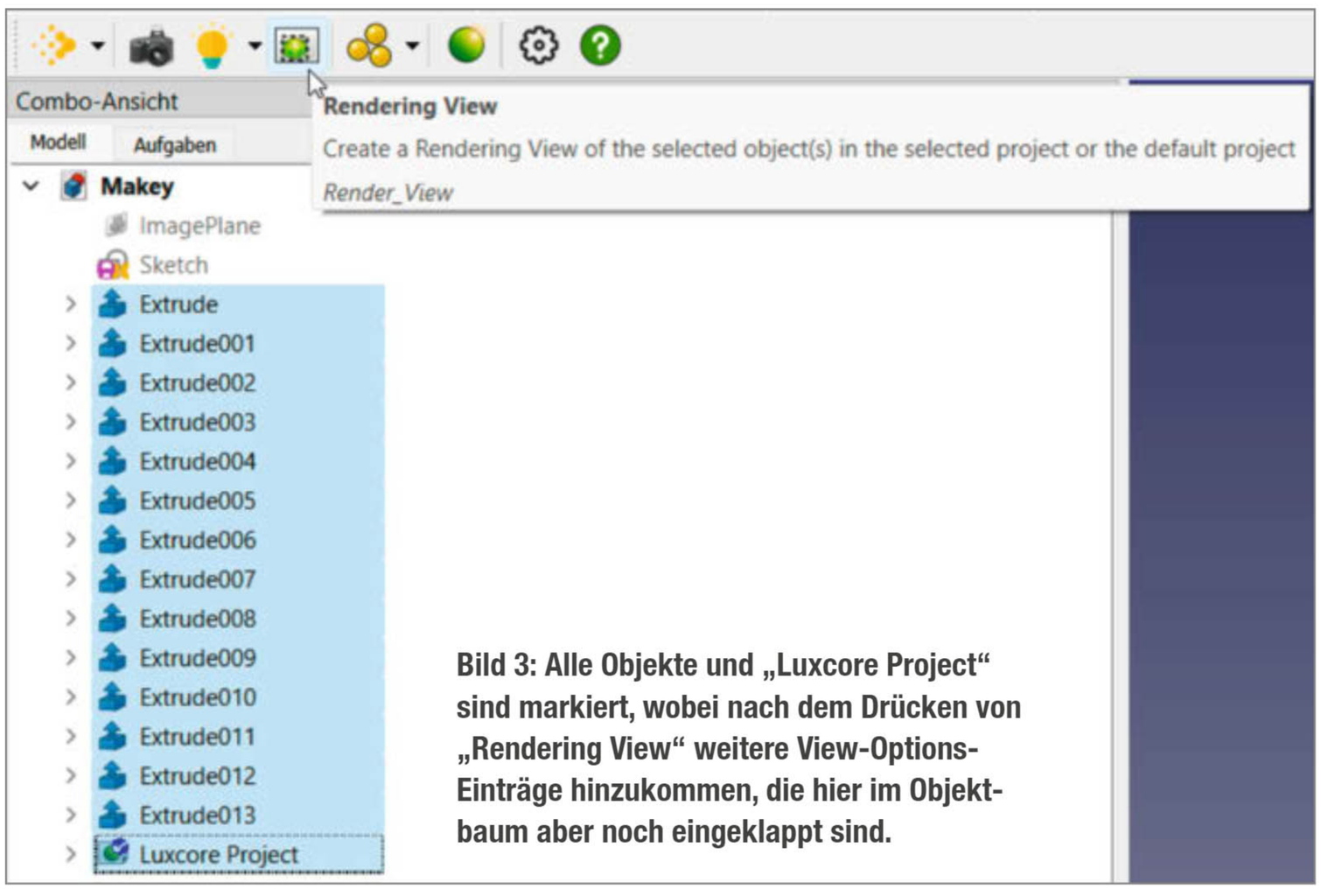


Bild 3: Alle Objekte und „Luxcore Project“ sind markiert, wobei nach dem Drücken von „Rendering View“ weitere View-Options-Einträge hinzukommen, die hier im Objektbaum aber noch eingeklappt sind.

OSPRay Studio

Bei Intels OSPRay Studio ist die Installation unter Windows recht einfach. Hierzu laden Sie die aktuelle Setup-Datei von der Homepage herunter und installieren die Software. In den Render-Workbench-Einstellungen (zu erreichen über das Zahnrad in der Werkzeugleiste, siehe Bild 1 rechts) geben Sie in das Feld neben der Beschriftung „OspStudio executable path“ den absoluten Pfad zur benötigten Datei /bin/ospStudio.exe im Installationsverzeichnis an (Bild 2). Anschließend können Sie die Installation mit einem Klick auf den Test-Button rechts daneben überprüfen.

POV-Ray

Bei POV-Ray funktioniert es im Prinzip genauso. Nach der Installation der Windows-Version der eigentlichen Engine öffnet sich allerdings ein weiteres Browserfenster mit einem Download-Link für die nötigen DLLs – das hat lizenzrechtliche Gründe. Nach deren Installation trägt man in FreeCAD wiederum bei den Render-Workbench-Einstellungen in das Feld neben der Beschriftung „PovRay executable path“ den absoluten Pfad zur benötigten Datei /bin/pvengine64.exe im Installationsverzeichnis ein (Bild 2).

Auf einem unserer Testrechner öffnete sich bei einem Klick auf den Test-Button zwar kurz der POV-Ray-Editor, beendete sich dann aber nach kurzer Anzeige mit der Fehlermeldung, dass Kommandozeilen-Parameter nicht geparst werden konnten. Trotz dieses fehlgeschlagenen Tests konnten wir mit der Engine anschließend aber auf normale Weise Renderings erzeugen; der Test in FreeCAD ist somit leider als nicht zuverlässig einzustufen.

Cycles

Cycles ist eine Rendering-Engine aus dem Blender-Projekt, lässt sich aber als eigenständiges Programm herunterladen, ohne dass Blender installiert sein muss. Allerdings ist Cycles auf der Projektseite nur als Quellcode verfügbar, sodass man das Programm selbst kompilieren müsste – gäbe es nicht auf GitHub Projekte, bei denen man sich eine kompilierte Cycles-Engine herunterladen kann (siehe Link in der Kurzinfo). Wie man ein GitHub-Repository als Zip-Archiv herunterlädt, beschreiben wir für Einsteiger in einem Online-Artikel, ebenfalls über diesen Link zu erreichen. Das Archiv entpackt man anschlie-

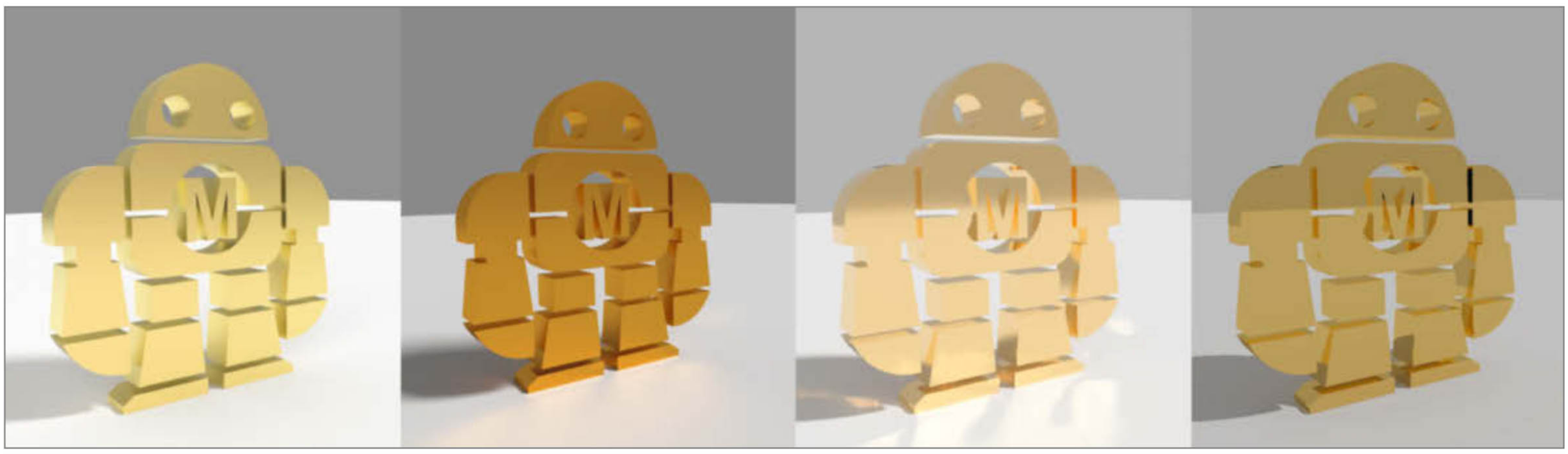


Bild 4: Vier Renderer im Vergleich bei Standardeinstellungen. Von links nach rechts: LuxCore, Cycles, OSPRay, POV-Ray

ßend und fügt den Pfad zur Datei cycles.exe in die Render-Workbench-Einstellungen in FreeCAD bei „Cycles executable (standalone) path“ ein (Bild 2).

Je nach Alter der fertig kompilierten Version muss man natürlich damit rechnen, nicht die frischeste Cycles-Version zu bekommen – als Teil von Blender wird der Renderer laufend weiterentwickelt; die Stand-alone-Version stufen die Entwickler aber als „work in progress“ ein. Wer die jüngste Fassung selber kompilieren mag, findet unter den Links in der Kurzinfo Informationen dazu.

LuxCore

LuxCore kann man ebenfalls als komprimierte Datei herunterladen und an einem Ort der Wahl auf der Festplatte entpacken. Auf der Downloadseite wählt man dazu „LuxCoreRender Standalone release“ und trägt den Pfad zur Datei luxcoreui.exe sowohl bei „LuxCore command (cli) path“ als auch unter „LuxCore UI path“ in FreeCAD ein (Bild 2). Bei unseren Versuchen ließ sich zwar auch eine Installation des LuxCore API SDK in den Einstellungen einrichten, beim Rendern erschien jedoch eine Fehlermeldung und der Vorgang brach ab. Die Stand-alone-Lösung funktioniert hingegen einwandfrei.

Für weitere Hilfe zur Installation und für andere Betriebssysteme als Windows ist die GitHub-Seite des Render-Workbench-Projektes hilfreich. Hier finden sich die direkten Links zu den jeweiligen Engines und eine Anleitung in englischer Sprache, wie man externe Renderer in die Render-Workbench einbindet, siehe Link in der Kurzinfo.

Makey als Test

Um ein erstes Bild zu erstellen, öffnen Sie das gewünschte FreeCAD-Projekt. Falls Sie noch keines haben, können Sie (wie wir im Folgenden) als Beispiel Makey verwenden, das Robotermaskottchen der Maker Faire, das Sie über den Link in der Kurzinfo als 3D-Datei fertig herunterladen und dann in FreeCAD öffnen können.

Wechseln Sie in den Render-Arbeitsbereich, falls Sie sich aktuell in einem anderen befinden, und wählen Sie dann aus der Ausklappliste auf der linken Seite der Render-Symbolleiste ein Beispielprojekt aus, für das Sie den Renderer installiert haben, beispielsweise „Luxcore Project“. Anschließend öffnet sich ein Dateidialog, um eine Projektvorlage auszuwählen.

Die jeweiligen Studio-Light-Vorlagen für die einzelnen Renderer eignen sich gut für den Anfang – wählen Sie daher „luxcore_studio_light.cfg“ aus. Hiermit erscheint im Objektbaum in FreeCAD auf der linken Seite ganz unten ein neues Objekt mit dem Namen „Luxcore Project“. Ein weißer Haken auf blauem Grund am Symbol sagt aus, dass man das Projekt neu berechnen, also aktualisieren muss. Dies erreichen Sie entweder mit dem Eintrag „Objekt neu berechnen“ aus dem Kontextmenü, mit der Taste F5 oder über Bearbeiten / Aktualisieren in der Menüleiste von FreeCAD.

Eine Projektvorlage kann man auch noch nachträglich ändern, wenn man mit der rechten Maustaste auf dem Projekt das Kontextmenü aufruft und „Change template“ anklickt. Es ist auch möglich, verschiedene Projektvorlagen in einem Projekt zu haben, indem man einfach die Projekte der verschiedenen Renderer hinzufügt. So braucht man nicht für jeden Renderer eine eigene FreeCAD-Datei zu erstellen. Gerendert wird dann jeweils mit der Engine, deren Objekt man gerade ausgewählt hat, wenn man auf den Render-Button klickt.

Perspektivfragen

Projektvorlagen enthalten vorgegebene Szenen mit voreingestellter Kamera und Lichtern. Man muss sich daher erst einmal keine Gedanken über Kameraposition oder Lichtintensität machen. Wechseln Sie jedoch zuerst in die perspektivische Ansicht, falls das 3D-Objekt aus einer der Grundansichten angezeigt wird, etwa von vorne oder von oben. Dies kann man entweder im Ansichtenwürfel oder in der Menüleiste unter „Ansicht/Perspektivi-

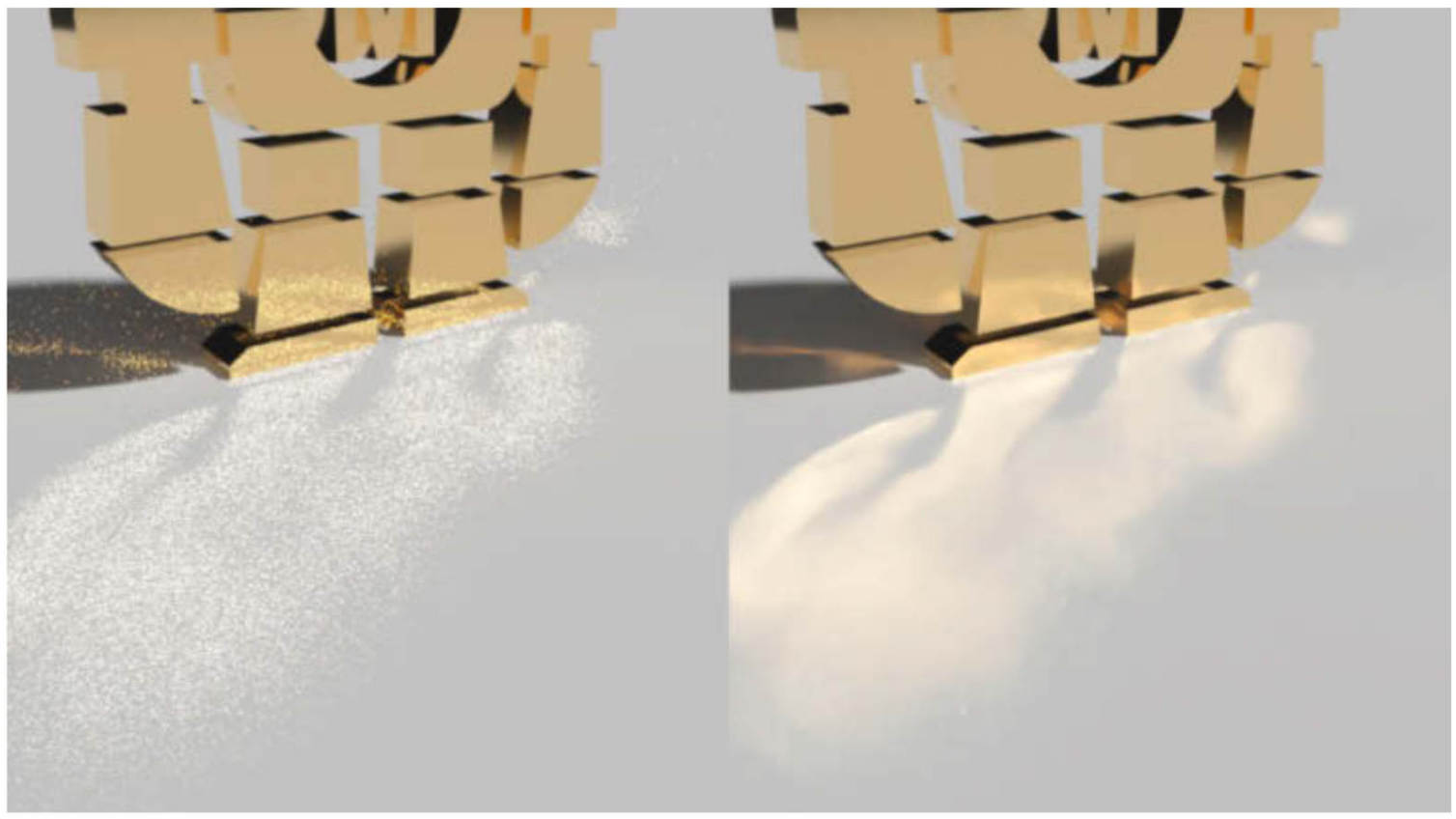


Bild 5: OSPRay-Projekt, links verpixelt und rechts mit eingeschaltetem Denoiser

sche Ansicht einstellen“. Jetzt zeigt das gerenderte Bild genau die Ansicht, die Sie im Bearbeitungsbereich sehen.

Fügen Sie nun zu dem Projekt alle Objekte aus dem Objektbaum hinzu, die der Renderer verarbeiten soll. Hierzu markieren Sie die Objekte in der Baumansicht mit der Maus und gedrückt gehaltener Steuerungstaste, inklusive des Projekteintrags „Luxcore Project“; anschließend betätigen Sie den Button „Rendering View“ (Bild 3). Dies führt dazu, dass FreeCAD unterhalb des Projekteintrags weitere Einträge mit der Endung „@Luxcore-Project“ hinzufügt. Diese werden eingeblendet, wenn man sie über einen Klick auf die kleine Pfeilspitze links neben dem „Luxcore Project“ ausklappt. Diese Einträge nennen sich „View options“, die wiederum Eigenschaften beinhalten, die man im Eigenschaftenfenster ändern kann. Diese beeinflussen, wie das Objekt gerendert wird.

Jetzt können Sie mit dem ersten Render-Durchgang starten. Markieren Sie hierzu nur den Eintrag „Luxcore Project“ in der Baumansicht und betätigen Sie den Render-Button in der Werkzeugleiste. Nun öffnet sich für kurze Zeit ein Kommandozeilenfenster und zusätzlich das User-Interface von Luxrender, worin die gerenderte Szene erscheint.

Rendern ist ein fortlaufender Prozess, wobei anfangs die Szene noch etwas verrauscht erscheinen kann. Das Bild verbessert sich jedoch nach ein paar Sekunden oder auch über einen längeren Zeitraum – je nach eigener Hardware und Komplexität der Szene. Generell ist das Rendern sehr rechenintensiv, ein leistungsstarker Computer mit dedizierter Grafikkarte ist hier empfehlenswert. In den meisten Fällen funktioniert das Rendern aber auch mit einfacherer Hardware, allerdings nimmt es dann entsprechend mehr Zeit in Anspruch.

Wenn Sie nun das Luxrender-Fenster schließen, übernimmt der Renderer das zuletzt gerenderte Bild in FreeCAD und öffnet es als neue Ansicht. Klicken Sie in FreeCAD mit der rechten Maustaste auf das Bild, können Sie es über das Kontextmenü speichern oder in die Zwischenablage kopieren. Zurück zur 3D-Ansicht in FreeCAD wechseln Sie dann über die Karteireiter unterhalb des gerenderten Bilds in FreeCAD.

Analog funktioniert das Rendern auch mit den anderen Engines, aber mit Unterschieden im Detail: So zeigt Cycles selbst kein Bild außerhalb von FreeCAD an, sondern meldet sich dort direkt mit dem Ergebnis zurück, während man in der Ansicht von OSPRey sogar die Kameraposition noch interaktiv verändern kann, durch Ziehen der Maus bei gedrückt gehaltenen Tasten.

Regie führen

Man kann auch eine eigene Szene erstellen, in die man selbst Kameras oder Lichter hinzufügt. Hierzu wählt man zuerst mittels Kontextmenü und „Change template“ die Standard-Vorlage für den jeweiligen Renderer aus. Eine Kamera ist nicht zwingend nötig, um eine Szene zu rendern: Falls keine Kamera vorhanden ist, nimmt FreeCAD die aktuelle Sicht und erstellt eine virtuelle Kamera. Sie können jedoch über das Kamera-Symbol in der Render-Werkzeugleiste eine oder mehrere Kameras explizit hinzufügen, um die Szene festzulegen. Bei einem Klick auf die Schaltfläche wird eine Kamera hinzugefügt, die die Szene aus der Perspektive zeigt, die der aktuellen 3D-Ansicht entspricht. Fügt man mehrere Kameras hinzu, verwendet FreeCAD die zuletzt aktive Kamera.

In den Eigenschaften einer Kamera in der Seitenleiste links unten können Sie beispielsweise die Position, die Brennweite oder den Abstand zum Objekt festlegen. Wenn man aus dem Zeichenfenster ein Stück herauszoomt, erscheint die Kamera selbst im 3D-Raum als Objekt. So kann man sehen, wo sich die Kamera befindet und wie sie ausgerichtet ist.

Man muss die Kamera vor dem Rendern aber noch dem Projekt hinzufügen, ansonsten exportiert FreeCAD sie nicht an den Renderer. Markieren Sie dazu das Render-Objekt und die Kamera mit gedrückt gehaltener Steuerungstaste und fügen Sie die Kamera wieder mit der Schaltfläche „Rendering View“ zum Projekt hinzu. Anschließend erscheint die Kamera als Objekt im Objektbaum unterhalb des Projektes. Dort lassen sich wiederum in den Eigenschaften renderspezifische Parameter verändern, wenn gewünscht.

Spot an!

In der Symbolleiste findet man außerdem ein Glühbirnen-Symbol für die Lichter. Ein Klick auf das kleine schwarze Dreieck daneben klappt die Liste der möglichen Lichttypen aus. Neben dem standardmäßig ausgewählten „Point Light“ gibt es noch „Area Light“, „Sunsky Light“, „Image Light“ und „Distance Light“. Punktlichter

Kleines Textur-Einmaleins

Hochwertige Texturen fürs 3D-Rendering bestehen nicht nur aus einer farbigen Grafik, mit der die Oberfläche eines 3D-Objekts belegt wird – auch wenn das optisch erst einmal den größten Teil der Wirkung ausmacht. Fünf hochaufgelöste Grafiken liefern in unserem Beispiel zusammen den fotorealistischen Rost-Effekt für Makey (siehe Bild): Die erste Datei bestimmt die Farben (Color), die zweite sorgt im Renderer für ein minimales Oberflächen-Relief der damit belegten Fläche, auch wenn die im CAD-Modell perfekt eben angelegt ist (Displacement). Die dritte Grafik bestimmt, welche Teile der Oberfläche wie stark glänzen (Metalness), die vierte (hier nur auf den ersten Blick eine einfarbige Textur) gibt die Richtung der sogenannten Normalen an – das sind Vektoren, die senkrecht auf der Oberfläche stehen und dem Renderer entscheidende Hinweise geben, wie einfallendes Licht reflektiert wird (NormalGL). Die letzte Textur bildet die Rauheit der Oberfläche ab, was ebenfalls Einfluss auf Glanzlichter und Lichtreflexionen hat (Roughness).

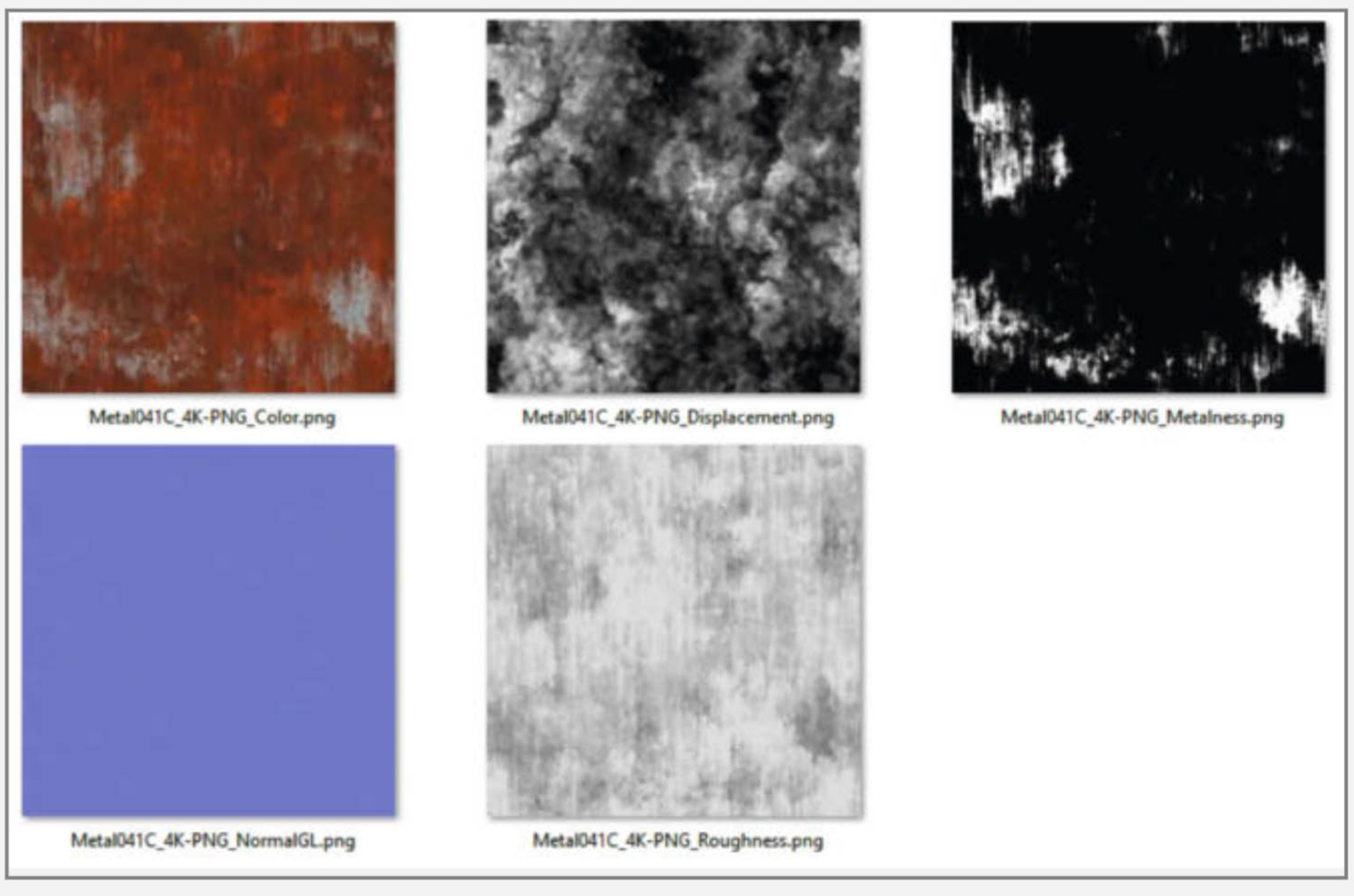


Bild 6: Dank Rost-Textur und Displacement in Luxrender wird aus Makey ein verrosteter und auseinanderfallender Haufen Blechschrott.

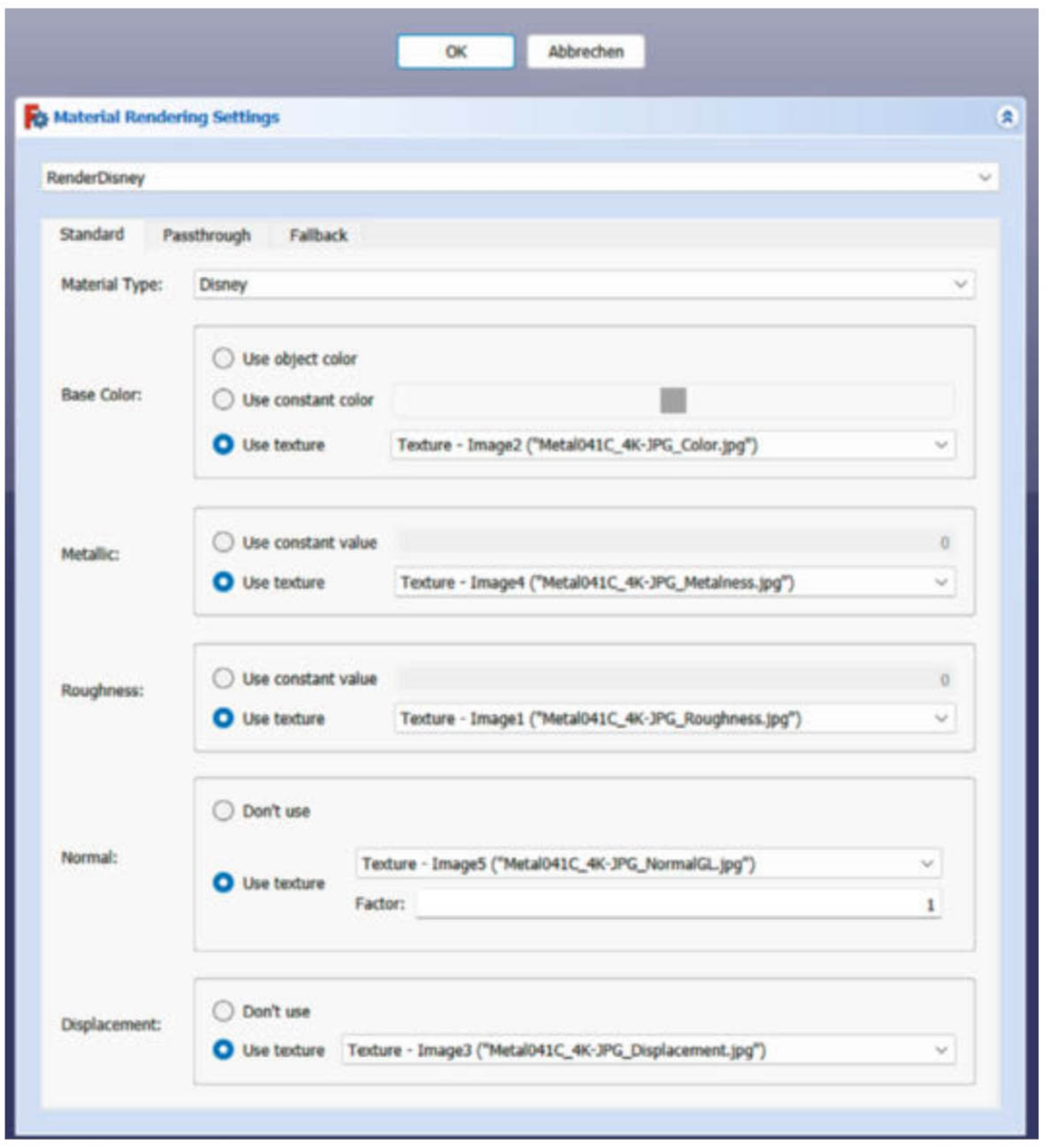


Bild 8: Die Einstellungen für die Texturen

erinnern ein wenig an Kerzenflammen, Flächenlichter erzeugen etwas diffusere Beleuchtung, ähnlich wie Lichtwannen im Fotostudio. Sonnenlicht kommt parallel aus einer Richtung, außerdem wird das 3D-Objekt andeutungsweise unter freien Himmel gesetzt.

Wenn man der Szene ein solches Licht hinzufügt, kann es sein, dass es im 3D-Zeichenfenster entweder in der Mitte oder am Fuß des zu rendernden 3D-Objekts als eigenes Objekt erscheint. Man kann jedoch in den Eigenschaften die Position des Lichts unter „Location" einfach ändern, ebenso weitere Eigenschaften wie die Leuchtkraft.

Die einzelnen Lichter haben unterschiedliche Darstellungen im 3D-Zeichenfenster, beispielsweise das Point Light mit einem gepunkteten Stern oder das Area Light mit einem ausgefüllten Rechteck. Die Auswirkungen des Lichtes sieht man direkt im Zeichenfenster auf dem Objekt, dies ist in der Nähe des Lichtes entsprechend heller.

Auch die Lichter muss man wie schon die Kamera dem Projekt hinzufügen. Markieren Sie dazu das Projekt und die Lichter mit gedrückter Steuerungstaste und fügen Sie sie mit der Schaltfläche „Rendering view" aus der Render-Werkzeugleiste dem Projekt hinzu.

Material zuweisen

Wir verwenden in unserem Beispiel für diesen Artikel der Einfachheit halber die eingangs erwähnten fertigen Studio-Light-Projektvorlagen ohne selbst platzierte Kameras und Lichter. Nach dem ersten Rendern erscheinen im Bild die einzelnen Teile allerdings einheitlich in Grau. Das liegt daran, dass FreeCAD ein Standardmaterial verwendet, das eher langweilig ist.

Um beispielsweise Makey in Gold zu rendern, fügen Sie mit dem Button „Create Material" aus der Render-Werkzeugleiste ein neues Material hinzu, das nach einem Klick auf das kleine schwarze Dreieck neben dem Material-Symbol erscheint, das drei gelbe Kugeln zeigt. Hier öffnet sich nun ein Aufgabenfenster, in dem Sie aus der Ausklappliste bei „Voreinstellungen auswählen…" den Eintrag „RENDER - Gold" wählen und die Eingabe mit OK bestätigen.

Markieren Sie nun in der Baumansicht alle 3D-Objekte der ursprünglichen FreeCAD-Datei (nicht die Objekte innerhalb des Render-Projekts!) mit der Maus bei gedrückt gehaltener Steuerungstaste, welche das Material erhalten sollen. Drücken Sie wieder in der Werkzeugleiste neben dem Button „Create

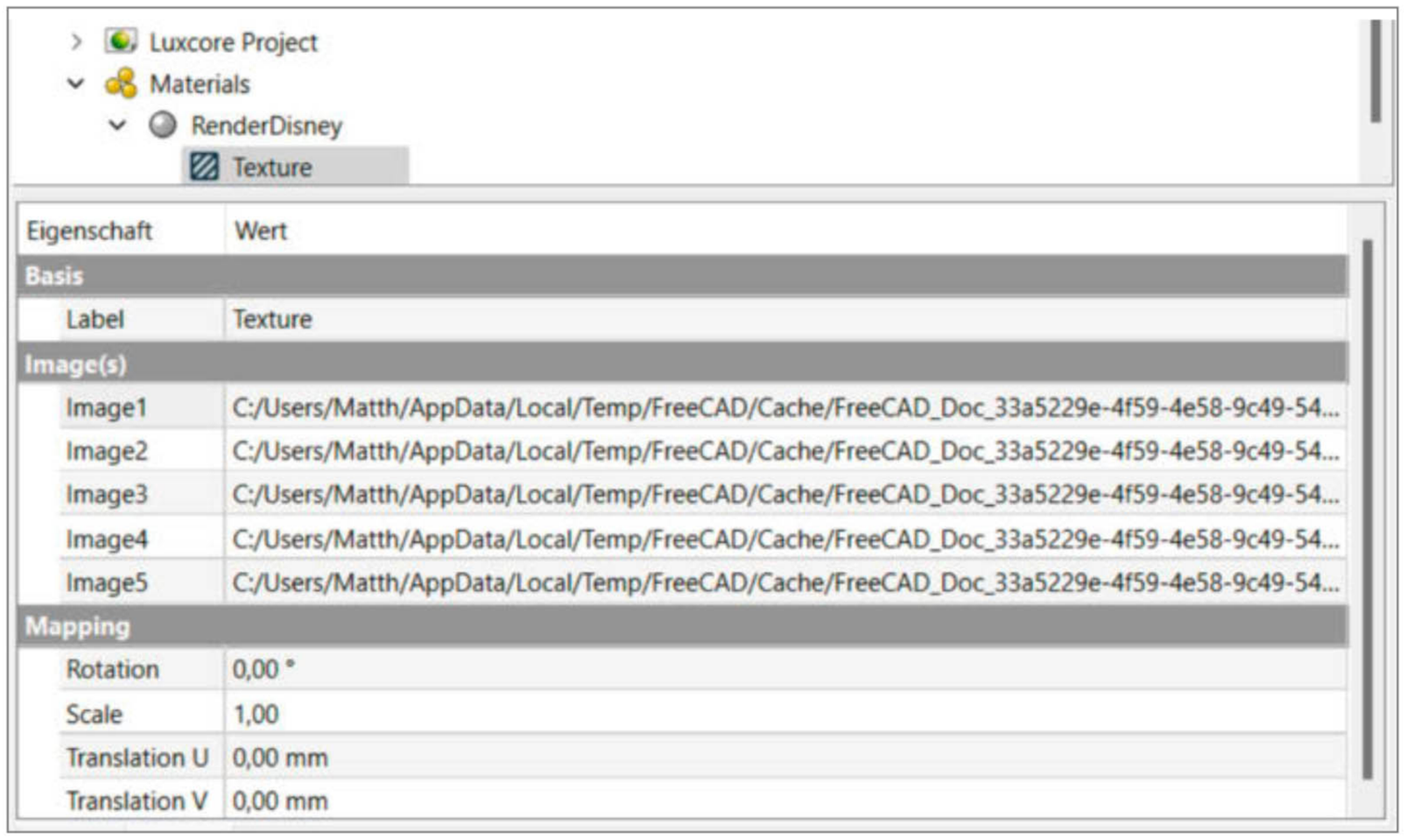


Bild 7: Mehrere Bilder mit Pfadangaben in den Textureinstellungen

Material“ die Dropdown-Auswahl und wählen Sie den Button „Apply Material“ aus. Nun erscheint ein Fenster, in dem Sie den einzigen Eintrag „RenderGold“ auswählen und mit OK bestätigen. Damit haben Sie allen Objekten das Material Gold zugewiesen. Wenn Sie jetzt das Luxcore-Projekt in der Baumansicht erneut markieren und neu rendern, erscheint Makey in Gold.

Sie können die Szene jetzt noch verbessern, indem Sie eine Bodenplatte hinzufügen. Wenn Sie wieder das Projekt markieren, finden Sie in den Eigenschaften einen Abschnitt mit dem Namen „Ground Plane“. Setzen Sie die Eigenschaft „Ground Plane“ von „false“ auf „true“. Unter „Ground plane colour“ können Sie die Farbe der Bodenplatte ändern und dadurch auch die Farbe, welche die Bodenplatte auf das Objekt zurückwirft. Unter „Ground plane size factor“ können Sie die Größe der Bodenplatte anpassen sowie die Höhe auf der Z-Achse unter „Z position for ground plane“ einstellen.

Obwohl es sich eigentlich um dieselbe Szene handelt: Wenn man das gleiche Objekt unter ähnlichen Bedingungen wie bei den jeweiligen Studio-Light-Vorlagen rendert, kommt bei jeder Engine ein anderes Ergebnis heraus. Einerseits kann die Beleuchtung zu tief sitzen wie bei OSPRay oder es entstehen treppenartige Artefakte und das Objekt scheint durchsichtig zu sein wie bei POV-Ray (siehe Bild 4).

Außerdem können Bereiche pixelig erscheinen, was an der Arbeitsweise der Rendering-Engines liegt. Bei OSPRay spiegelt sich Makey im vorderen Bereich der Grundplatte, was zu einer Verpixelung führt. Hier können Sie aber Abhilfe schaffen, indem Sie in den Eigenschaften des Render-Objekts unter dem Bereich „Execution Control“ den Eintrag „Denoiser“ auf den Wert „true“ setzen (Bild 5).

Farben und Texturen

Wenn Sie ein Objekt als Kunststoff rendern wollen, können Sie das Material „RENDER - RoughPlastic“ verwenden und zuweisen, wie vorhin beim Gold beschrieben. Hier verwendet FreeCAD jedoch die Standardfarben. Makey hätte damit zwar einen Plastik-Look, jedoch in der Farbe Grau. Um den Objekten eine andere Farbe zu geben, markieren Sie in der Baumansicht alle Objekte mit gedrückter Steuerungstaste, die eine neue Farbe erhalten sollen. Drücken Sie die rechte Maustaste auf den markierten Objekten und wechseln Sie in den Menüpunkt „Darstellung …“. Klicken Sie danach in den Aufgaben auf die Farbe des Punktes „Flächenfarbe“ und wählen Sie einen anderen Ton aus. Wir haben für Makey die Farbe Rot verwendet, nur für das M im Bauch Weiß und damit die Grafik des Titelbilds dieses Artikels gerendert.

In der Render-Workbench kann man nicht nur einzelne Materialien auswählen, sondern auch Texturen verwenden. Hierbei überzieht FreeCAD ein Objekt mit der gewünschten Oberfläche als Pixelgrafik. Wir haben als Beispiel Makey mit einem rostigen Überzug versehen (Bild 6). Dafür benötigen Sie zuerst eine Textur, die Sie auf verschiedenen Webseiten kostenlos herunterladen können, etwa bei AmbientCG und MaterialX (siehe Links in der Kurzinfo). Suchen Sie zum Beispiel auf der Webseite AmbientCG unter dem Menüpunkt „Materials“ nach dem Stichwort „Rust“ – wir haben für unseren Versuch das Material „Metal 041 C“ gewählt.

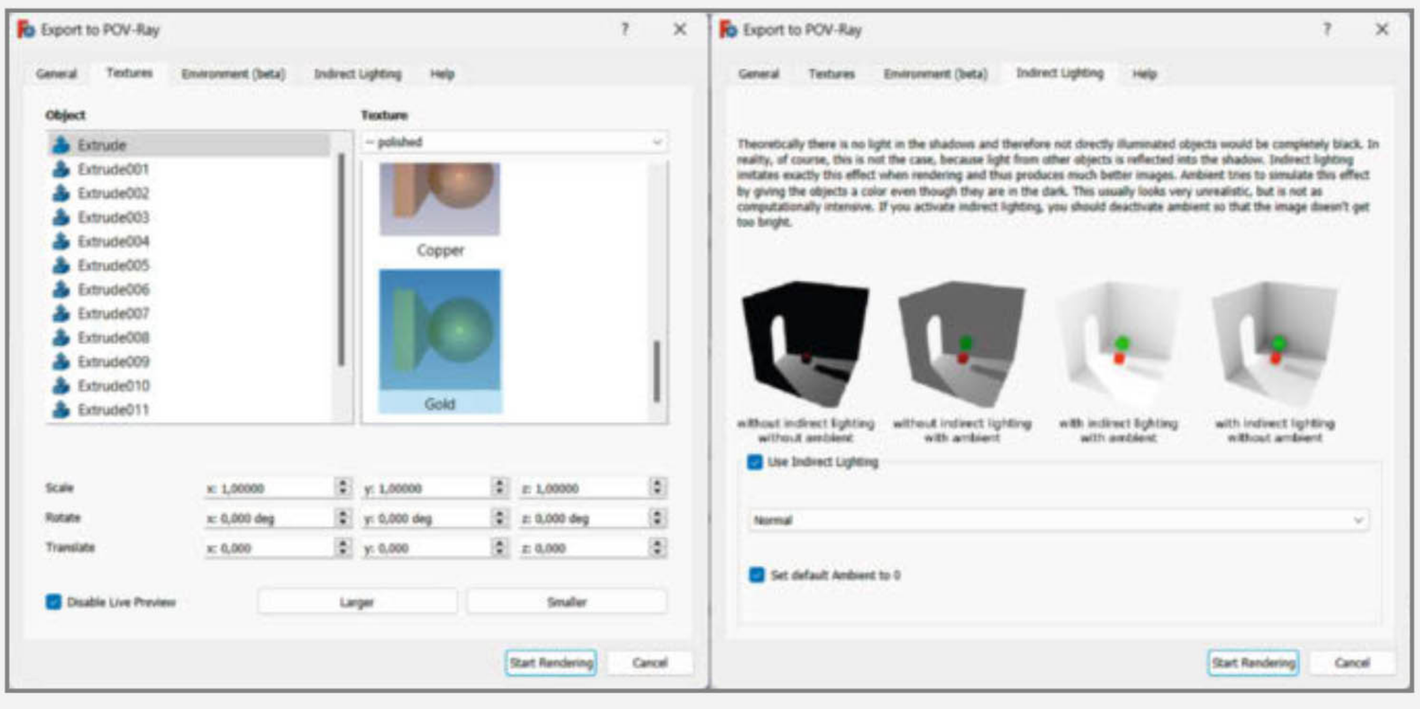
Auf der Materialseite kann man nun verschiedene Dateien herunterladen, gestaffelt nach Auflösung, die daher unterschiedliche Dateigrößen haben. Je nach Objektgröße ist eine Auflösung von 1K oder 2K zu klein, wodurch im Ergebnis die Texturen zu grob erscheinen. Laden Sie daher die Datei 4K-PNG.zip mit einer Downloadgröße von 229 MB herunter. Entpacken Sie die Zip-Datei in ein Verzeichnis Ihrer Wahl. Die Dateien mit den Endungen MTLX und USDC können Sie direkt löschen,

POV-Ray-Alternative

Sucht man nach der Render-Workbench, wie eingangs im Artikel beschrieben, zeigt der Add-on-Manager auch ein Ergebnis mit dem Namen „POV-Ray-Rendering“ an. Wie der Name schon sagt, ist diese Workbench speziell auf die POV-Ray-Engine zugeschnitten. Auch sie lässt sich wie gewohnt über den Add-on-Manager installieren.

Diese Workbench bringt eine Symbolleiste mit vier Buttons mit. Mit drei Buttons kann man Punkt-, Flächen und Spotlichter einfügen. Hinter dem Button „Export Model“ versteckt sich ein Fenster mit mehreren Karteireitern. Unter „General“ nimmt man allgemeine Einstellungen wie die Bildgröße vor. Im Karteireiter „Textures“ sind auf der linken Seite die Objekte aufgelistet, denen man auf der rechten Seite vorgegebene Materialien wie Gold zuweisen kann. Oberhalb der Materialzuordnung unter „Textures“ kann man nach Materialgattungen wie „Glass“, „Metal“, „Stone“ und weiteren filtern. Im nächsten Karteireiter kann man ein HDR-Image als Hintergrundbild verwenden und entsprechende Einstellungen vornehmen. Unter „Indirect Lightning“ lassen sich aus einer Auswahlliste vorgegebene Einstellungen für indirektes Licht vornehmen.

Nachdem man alle Einstellungen vorgenommen hat, beginnt man das Rendern mit dem Button „Start Rendering“. Hierbei wechselt das Programm in den POV-Ray-Renderer und rendert das Objekt mit den vorgegebenen Einstellungen. Das fertige Bild gibt er jedoch nicht wieder an FreeCAD zurück; wenn man das Bild speichern möchte, muss man dies im POV-Ray-Renderer tun.



Materialzuordnung und Anpassung des indirekten Lichts

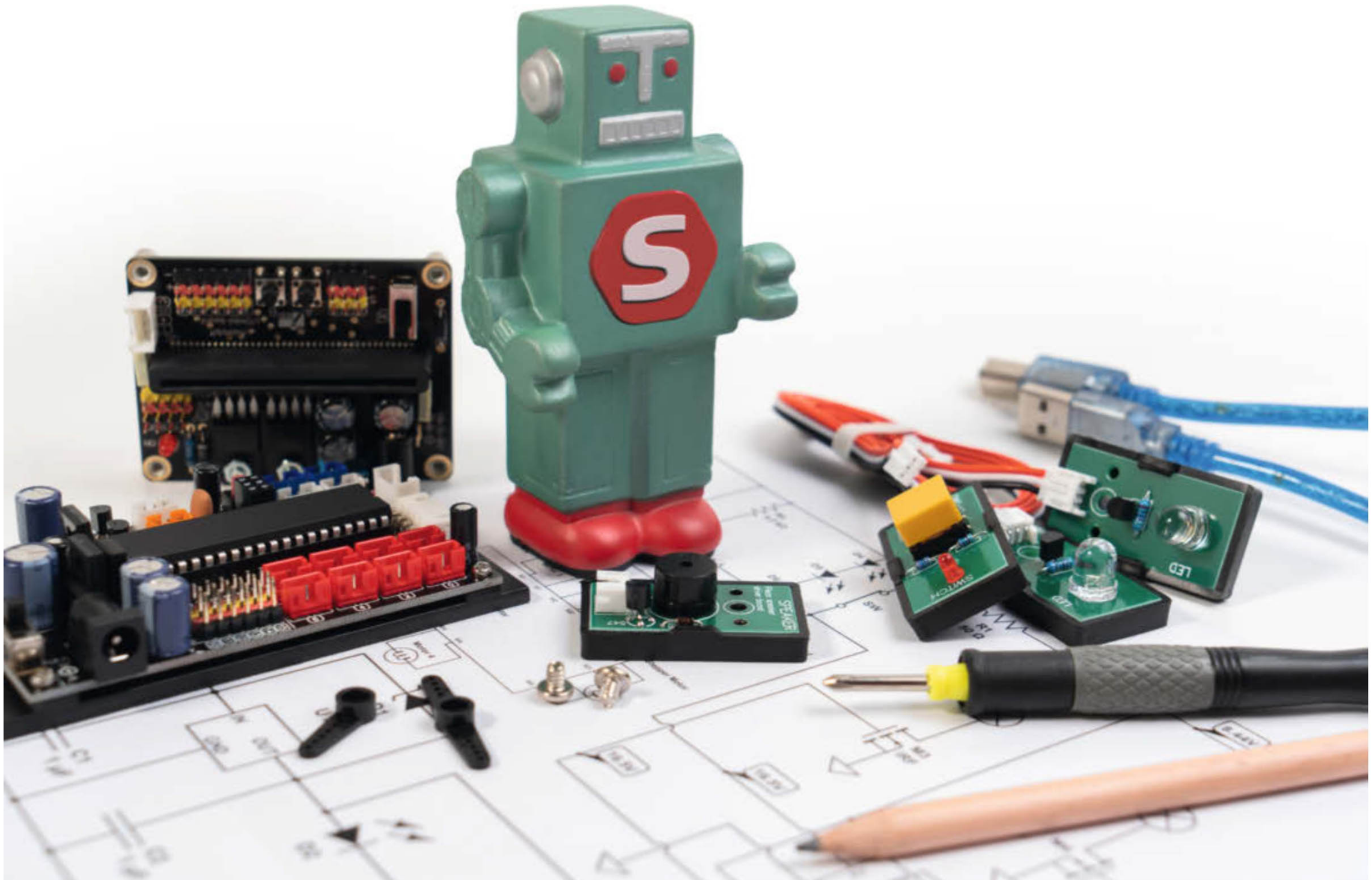
ebenso die Datei, die auf _NormalDX.jpg endet. Das Bild Metal041C.png dient nur als Vorschau, damit man weiß, um welches Material es sich handelt. Sie können es ebenfalls löschen, das erleichtert im Folgenden die Übersicht.

Fügen Sie nun mit dem Button „Create Material“ ein neues Material dem Projekt hinzu. Wählen Sie in der Aufgabenansicht unter „Render Material“ aus der Ausklappliste das Material „RENDER - Disney“ aus und bestätigen Sie mit OK. Im Objektbaum ist nun der Eintrag „Materials“ (falls noch nicht vorhanden) und darunter „RenderDisney“ hinzugekommen. Weisen Sie das Material wieder den Objekten zu, indem Sie diese mit gedrückter Steuerungstaste markieren und unter „Material“ in der Render-Werkzeugleiste den Button „Apply Material“ betätigen. Wechseln Sie die Auswahl zu „RenderDisney“ und bestätigen diese mit OK.

Gehen Sie mit der rechten Maustaste im Objektbaum auf den Eintrag „RenderDisney“ und wählen Sie aus dem Kontextmenü „Add Texture“ aus. Jetzt ist unterhalb des Eintrags „RenderDisney“ im Objektbaum der Eintrag „Texture“ hinzugekommen, den Sie gegebenenfalls mit dem kleinen Pfeil im Objektbaum neben „RenderDisney“ ausklappen müssen. Fügen Sie nun die Bilder in die Eigenschaften ein, indem Sie unter „Image(s)“ und „Image“ klicken und dort das erste Bild Metal041C_4K-PNG_Color.png auswählen. Fügen Sie das nächste Bild hinzu, indem Sie mit der rechten Maustaste auf den Eintrag „Texture“ klicken und das Kontextmenü aufrufen, wobei Sie „Add Image Entry“ auswählen. Hierbei ist in den Eigenschaften unter „Images(s)“ ein weiterer Eintrag mit dem Namen „Image1“ hinzugekommen. Klicken Sie dort wieder hinein und wählen das nächste Bild aus. Wiederholen Sie diesen Vorgang, bis Sie alle fünf Bilder hinzugefügt haben, die Texturen für Color, Displacement, Metalness, Normals sowie Roughness definieren (Bild 7, siehe auch Kasten).

Verknüpfen Sie nun die Texturen mit den Render-Einstellungen, indem Sie mit der rechten Maustaste auf den Eintrag „RenderDisney“ gehen und dort „Edit Render Settings“ auswählen. Im Aufgabenfenster können Sie nun die Texturgrafiken einzelnen Einstellungen zuweisen, wie in Bild 8 gezeigt. Achtung: Um Platz zu sparen, haben wir Aspekte, für die wir keine Textur zur Verfügung hatten, aus dem Screenshot herausgeschnitten. Bestätigen Sie mit OK, wenn alle Werte gesetzt sind. Dann können Sie direkt mit dem Rendern beginnen.

Generell kann man auch Materialien und Texturen kombinieren, indem man jedem Objekt ein anderes Material oder eine andere Textur zuweist. Wem diese Möglichkeiten noch nicht genug sind, der kann auch eigene Materialkarten für das Rendering schreiben. Dies ist auf der GitHub-Seite des Projektes für die Render-Workbench ausführlich erklärt. Jetzt sind Sie aber erst mal an der Reihe, Ihre Projekte mit Materialien und Texturen fotorealistisch zum Leben zu erwecken. —*akf*



ShutterStockStudio/Shutterstock.com

Programmieren mit Sming

Die Arduino IDE reicht nicht mehr, aber Sie möchten doch für unterschiedliche Zielplattformen offen bleiben? Sming vereint Komfort und Kontrolle, ist gut dokumentiert und lässt sich leicht installieren. Sming läuft unter Linux, macOS und per WSL2 unter Windows.

von Carsten Wartmann

Sming stellt ein Open-Source-Framework zur schnellen asynchronen Programmierung von Mikroprozessoren für Embedded-Systeme und Internet of Things (IoT) zur Verfügung. Es stehen Funktionen und Bibliotheken bereit, die die typischen Schritte, etwa um ein Board mit einem Sensor ins IoT zu bringen, vereinfachen. Für Einsteiger gibt es Beispiele, die sich beliebig kombinieren lassen, für die Fortgeschrittenen ist die ausführliche Dokumentation der richtige Ausgangspunkt. Weitere Kernkomponenten umfassen einen asynchronen Netzwerk-Stack mit Server und Client, SSL, ein Filesystem mit Zugriff auf SD-Karten, Timer, Watchdog und vieles mehr. Es wird direkt mit den Herstellerwerkzeugen kompiliert, dabei wird alles durch ein ausgefeiltes make-System zusammengehalten, das viele der Schritte sehr vereinfacht. Vom Übersetzen des Quellcodes über die Konfiguration der Hardware bis zum Aufspielen der Firmware auf das Board wird alles mit nur wenigen immer ähnlichen, leicht merkbaren Befehlen erledigt.

Warum Sming?

Das Sming-Framework erzieht geradezu zu sauberen Programmen, die die Hardware ausnutzen, Strom sparen und, statt stumpf zu warten, in der Zwischenzeit andere Aufgaben erledigen. Was bei AVR-Prozessoren, mit denen die Arduino IDE groß geworden ist, noch als schneller Hack durchging, sollte bei den immer komplexeren Anforderungen an IoT-Geräten vermieden werden.

Das einfache Blink-Programm sieht erst einmal nicht so viel anders aus (siehe Listing Basic_Blink) als in der Arduino IDE. Statt zwischen den LED-Umschaltungen einfach mit delay() zu warten, wird ein Timer verwendet, der die `blink()`-Funktion alle 1000 Millisekunden (1 s) aufruft. Das mag erst einmal etwas komplizierter zu verstehen sein als die Sequenz in Arduino, wird aber in dem Moment so viel einfacher, wenn eine zweite LED dazukommt, die mit 5 Hz blinken soll und währenddessen das Board noch alle 30 Sekunden Messwerte per Internet übertragen muss.

Ist der Programmcode geschrieben, reicht der einfache Befehl `make flash` und das Projekt wird kompiliert und auf das Board geflasht.

Die Installation von Sming ist nicht kompliziert und die Konfiguration von Projekten ist simpel, ohne Abstriche für aufwendigere Projekte zu machen. Es besteht jederzeit die Möglichkeit, mit C++ auf alle Interna und Möglichkeiten der Entwicklungskits zugreifen zu können. Die nicht immer einfache Installation der Entwicklungskits (Espressif und RP2040) und deren Einrichtung werden auch automatisch übernommen; neue Boards und Cores etwa bei ESP32 werden so zeitnah mit dem Erscheinen der neuen Entwicklungskits unterstützt.

Nicht zuletzt gibt es die Möglichkeit, Sming in die großen Entwicklungsumgebungen wie VSCode und Eclipse einzubinden. Unterstützte Architekturen sind aktuell ESP82656, ESP32 (diverse Cores), RP2040 (Raspberry Pico) sowie ein Host-Emulator, der ohne Hardware auf dem PC läuft.

Sming wurde ursprünglich für Unix-artige Systeme (Linux, macOS, BSD …) entwickelt und benutzt die dort vorhandenen Tools ausgiebig, um seine Arbeit zu erledigen. Mit WSL2 von Microsoft gibt es nun ein System, das unter Windows läuft, wenig Ressourcen braucht und sich hervorragend in Windows integriert. In unserem Artikel „WSL2 für Maker“ in Make 1/24 (siehe „Mehr zum Thema“) haben wir dieses bereits behandelt und die Vor- und Nachteile anhand von Beispielen aufgezeigt. WSL2 ist eine gute und einfache Methode, wenn man die Tools von Linux braucht oder einfach nur mal Linux ausprobieren möchte, ohne auf das vielleicht nötige Windows zu verzichten. Hat man macOS oder Linux auf dem Rechner, so funktionieren die Anweisungen in diesem Artikel natürlich ebenso. Wir gehen im Weiteren auch davon aus, dass Sie WSL2 bereits eingerichtet haben.

Kurzinfo

- C++-Framework für ESP und RP2040
- Ausgefeiltes, leicht zu beherrschendes make-System
- Kompatibel zu Standard- und Arduino-Bibliotheken

Checkliste

Zeitaufwand:
1 Stunde

Kosten:
0 Euro

Mehr zum Thema

- Carsten Wartmann, WSL2 für Maker, Make 1/24, S. 94
- Ákos Fodor, Mikrocontroller-Projekte simulieren mit Wokwi, Make 3/23, S. 88
- Carsten Wartmann, Internet-of-Things-Dienste für Maker, Make 3/21, S. 32

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

Frisches WSL2

Starten Sie Ihre WSL2-Distribution über die Powershell, das Windows-Menü (Bild 1) oder die Windows-Suche. Ich benutze hier ein frisch installiertes Ubuntu 22.04; haben Sie die Distribution schon eine Weile in Benutzung, so erzeugen einige Befehle eventuell andere Ausgaben, wenn Softwarepakete bereits installiert sind. Auf Debian-Systemen muss „git“ nachinstalliert werden.

Im Prinzip können mehrere Sming-Versionen (wenn man e xperimentieren möchte) gleichzeitig installiert sein. In der offiziellen Dokumentation wird nach `/opt/` installiert und Sming lädt dorthin auch die nötigen Dateien für ESP und Rasperry Pico, damit sie nicht mehrfach auf dem System sein müssen. Es hat aber durchaus Vorteile, nicht nach `/opt` zu installieren.

Das Linux-System sollte aktuell sein, die Distributionen aus dem Windows-Store sind teils schon etwas abgelagert, daher würde ich die folgenden Befehle zur Aktualisierung des Systems ausführen:

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

Sudo wird nach Ihrem root-Passwort fragen. Beim zweiten Befehl mit Y (oder Return) bestätigen.

Bei meinem Ubuntu fehlte noch das Paket python3.10-venv und wurde von Sming auch nicht nachinstalliert, also zur Sicherheit:

```
$ sudo apt install python3.10-venv
```

Listing Basic_Blink

```
#include <SmingCore.h>

#define LED_PIN 2 // GPIO2 (ESP8266)

SimpleTimer procTimer;
bool state = true;

void blink()
{
    digitalWrite(LED_PIN, state);
    state = !state;
}

void init()
{
    pinMode(LED_PIN, OUTPUT);
    procTimer.initializeMs<1000>(blink).start();
}
```

Sming-Entwicklung

Sming wird seit 2015 (und auch heute noch) entwickelt, wie man an den Commits im GitHub sieht. Die Entwickler nehmen an den „Discussions“ teil und ein Vorschlag von mir, der die Nutzung von USB-Geräten angeht, führte innerhalb von 48 Stunden zu einem Pull-Request, der dann zügig in den Development-Zweig eingebaut wurde.

Im Artikel wird ja die in der Entwicklung befindliche Wokwi-Integration angesprochen, eine sehr spannende Möglichkeit mit viel Potenzial, schnell ohne Hardware wie Boards und Sensoren Code auf Mikroprozessoren zu testen und zu debuggen.

Apropos Debugging: Dieses komplexe Thema wurde hier ausgelassen; erfahrenen Anwendern von Eclipse und VSCode bietet Sming die passenden Einstellungen und Skripte. Auch aus der Bash heraus ist Debugging möglich. Für hardwarenahe und komplexe Projekte unabdingbar, für diese Einführung aber zu viel. Mit Debugging auf AVR beschäftigt sich der Artikel „AVR-Programme debuggen“ in diesem Heft.

Schaut man sich im Code um und spricht mit den Entwicklern, kommen noch weitere teils sehr experimentelle Code-Zweige zum Vorschein: So gibt es die Graphics-Library, die komplexe Zeichenfunktionen und Pixelgrafik-Unterstützung bietet und das Zeug hat, eine vom physischen Display unabhängige Grafik zu erstellen. Im Moment werden leider nur zwei Displaytypen und die Anzeige auf dem Host-Rechner (auch per Netzwerk) unterstützt, aber auch das kann schon hilfreich sein, wie jeder weiß, der schon mal Grafiken und User-Interfaces auf Mikrocontrollern erstellt hat. Es gibt sogar einen grafischen Editor zur Erstellung von solchen Benutzeroberflächen.

https://github.com/DominicMe

Ich habe Sming in mein Heimatverzeichnis `/home/cw` installiert.

```
$ pwd
/home/cw
...
$ git clone \
    https://github.com/SmingHub/Sming
Cloning into 'Sming'...
... done....
$ ls

Sming
```

Nun den Installer aufrufen, das Skript wird eventuell nach dem root-Passwort fragen, wenn Sie nicht gerade `sudo` verwendet haben, um die Abhängigkeiten installieren zu können.

```
$ source ./Sming/Tools/install.sh all
```

Nach ein paar Minuten ist der Prozess mit der Meldung „Installation complete“ beendet. Installiert wurde der develop-Zweig (aktuelle Entwicklungsversion) aus dem GitHub, wir leben hier bei der Make gern gefährlich. Nein, Quatsch beiseite, tatsächlich ist die Entwicklungsversion sehr stabil und enthält eine Funktion zu USB-Geräten, die uns die Arbeit gleich viel einfacher macht.

Vor der Benutzung von Sming muss man ein paar Umgebungsvariablen für die Bash setzen. Dies erledigt ein Bash-Skript im Tools-Verzeichnis der Sming-Installation.

```
$ source ./Sming/Tools/export.sh
```

Diesen Befehl können Sie in Ihrer Bash-Voreinstellungsdatei (`~/.bashrc`) einbauen oder einfach manuell vor der Arbeit mit Sming aufrufen. Dies hat den Vorteil, dass man in verschiedenen Shells („Eingabeaufforderungen“) verschiedene Sming-Versionen nebeneinander benutzen kann, etwa Stable, Development und noch einen eigenen experimentellen Zweig.

Mach mal!

Nun geht’s aber los mit dem obligatorischen Blink, dem Hello-World-Programm für Mikrocontroller.

```
$ cd ./Sming/samples/Basic_Blink

$ make

...

LDGEN out/Esp8266/debug/build/rom0.ld
Basic_Blink: Linking out/.../app_0.
out
...
Total Used RAM : 1898
Free RAM : 80022
Free IRam : 25513
ESPTOOL2 out/.../rom0.bin
make[1]: Nothing to be done ...
make[1]: Leaving directory ...
```

Das erste Mal werden noch ein paar Komponenten hinzugeladen und kompiliert, die nächsten Male wird es dann schneller und kann noch zusätzlich durch `make -j 3` beschleunigt werden, was drei CPU-Kerne zum Kompilieren verwendet.

Beim Kompilieren wird eine „rom0.bin“-Datei geschrieben, die die Firmware enthält, sowie weitere nötige Partitionen für das ESP-Board. Die Firmware kann dann im nächsten Schritt per USB auf das Board geflasht werden.

Flashen

Beim Flashen der Firmware per USB taucht wieder (siehe WSL2-Artikel in „Mehr zum Thema“) das Problem auf, dass WSL2-Instanzen keinen von Microsoft vorgesehenen Weg haben, um auf USB-Ports zuzugreifen. Ein Weg wäre, auf Windows-Seite ein Flash-Tool zu benutzen, dies ist aber genauso umständlich wie eine Powershell aus WSL2 heraus, die ein Pythonskript unter Windows zum Flashen aufruft.

Für dieses und ähnliche Probleme hat sich der Entwickler Frans van Dorsselaer daran gemacht, eine Lösung zu finden: Sein usbipd-win

Eingabe der Befehle

Wenn die Kommandozeile mit der Shell „Bash“ unter Linux auf WSL2 gezeigt wird, ist dies durch ein $ angezeigt. Dieses Zeichen darf nicht eingegeben werden. Bei Ihnen sieht dieser Prompt (die Eingabeaufforderung) sicher anders aus, da normalerweise noch weitere Informationen wie das aktuelle Verzeichnis oder der Benutzer- und Systemname ausgegeben werden.

Wenn Sie diesen Artikel im Heft lesen, müssen Sie das Zeichen \ (Backslash) in langen umgebrochenen Zeilen nicht eingeben und können einfach in der Zeile weitertippen. Wenn Sie den Befehl aus dem Online-Artikel kopieren, können Sie Befehle aus mehreren zusammengehörigen Zeilen kopieren und einfügen, die Bash fügt die Zeilen dann korrekt zusammen.

gibt USB-Ports vom Host zum Gastsystem weiter, sei es eine andere Maschine im Netzwerk oder eine virtuelle Maschine wie Hyper-V oder WLS2.

Ja, auch usbipd erfordert eine einfache Installation. Allerdings kann man so praktisch alle USB-Geräte weitergeben und muss sich als Linux-Entwickler keine Gedanken machen, ob es nun COMx, ttyUSBx oder ttyACMx ist. Oder gar eine USB-Kamera oder ein USB-Netzwerkgerät. Die Installation und Bedienung sind im Kasten „USB für WSL“ erläutert.

In der WSL-Bash kann man kontrollieren, welches USB-Seriell-Device benutzt wird:

```
$ dmesg  |grep USB
...
[...] usb 1-1: Product: CP2102 USB...
[...] usb 1-1: ...attached to ttyUSB0
...

$ ls -l /dev/ttyUSB*
crw-rw---- 1 root dialout 188,
    0 Jun 18 10:38 /dev/ttyUSB0
```

Unter Ubuntu etwa gehören die Benutzer des Systems leider nicht automatisch der Gruppe „dialout“ an und dürfen keine USB-Ports benutzen. Also müssen wir dem Linux beibringen, das unser normaler Arbeitsaccount (bei mir `cw`, bitte ändern) berechtigt ist.

```
$ sudo adduser cw dialout
```

Damit gehört der Benutzer `cw` zur Gruppe `dialout`, die auf die Ports zugreifen darf. Leider müssen wir uns jetzt wieder ausloggen, erneut einloggen und die Sming-Umgebungsvariablen (s. o.) neu setzen.

Flash! A-ha!

Nun kann geflasht werden! Der Befehl `make flash` (siehe Kasten „Flash-Prozess“) schreibt die Firmware auf den Flash-Speicher des Boards und startet dann ein Terminal (miniterm von Python), um Ausgaben auf dem seriellen Port sichtbar zu machen. Beenden können Sie das Terminal mit Strg+T Q. Um ohne Flashen ein Terminal zu einem angeschlossenen Board zu bekommen, kann `make terminal` auch einzeln aufgerufen werden.

Der Haufen wilde (hier nicht abgedruckte) Zeichen kommt daher, dass die Boot- und Debugmeldungen auf ESPs mit einer eher ungewöhnlichen Baudrate von 74880 ausgegeben werden und Sming die eher übliche Rate von 115200 benutzt. Ich habe mir daher angewöhnt, das Kompilieren mit `make COM_SPEED_SERIAL=74880` aufzurufen. Dies wird dann an den Befehl zum Öffnen der seriellen Schnittstelle weitergereicht und die Ausgabe ist so klarer. Versuchen Sie die Schritte doch noch einmal mit diesem Parameter, die Abarbeitung sollte jetzt nur noch ein paar Sekunden dauern.

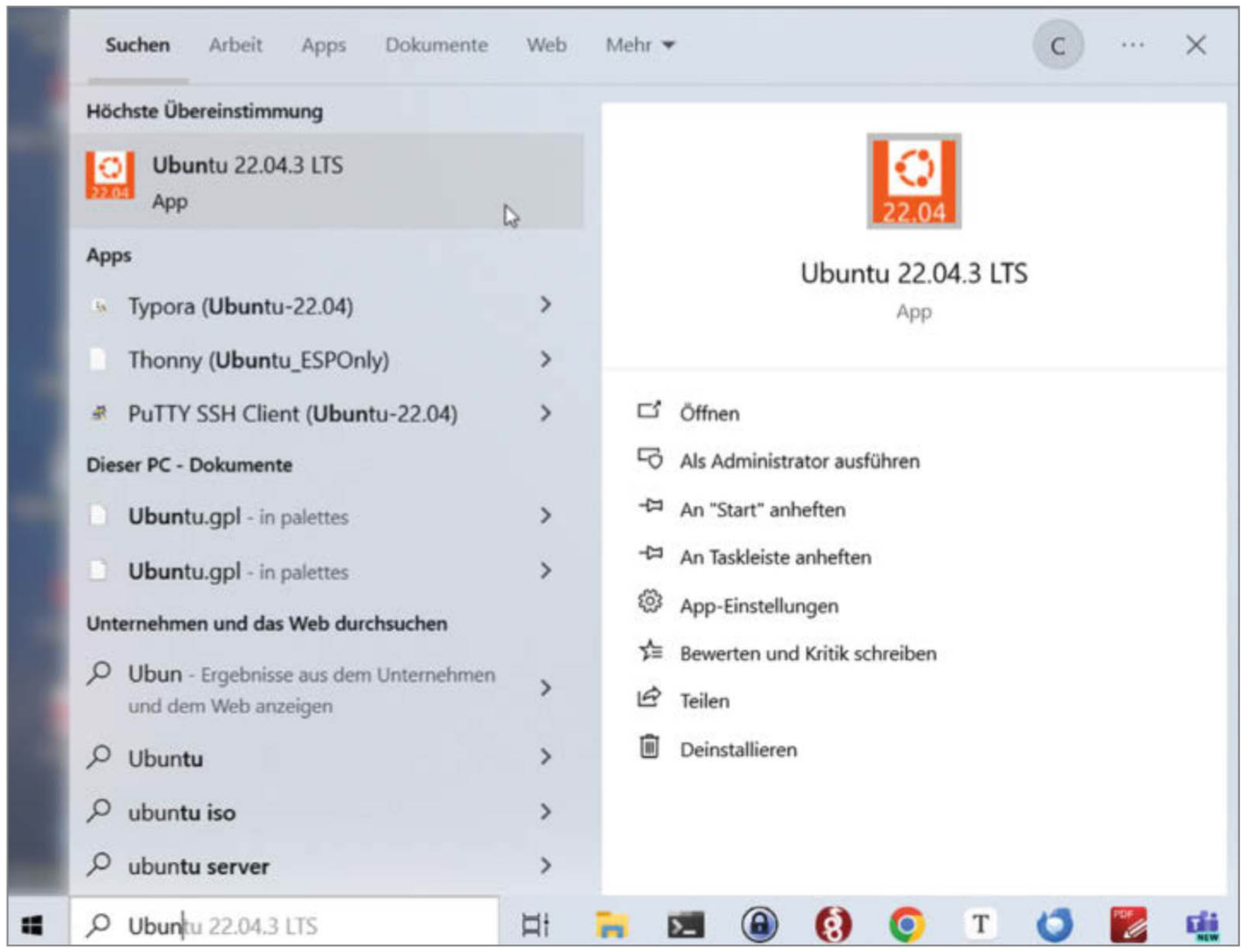


Bild 1: Der Start von WSL2.

VSCode

Benutzt man bereits VSCode (Visual Studio Code) unter Windows, möchte man das sicher auch für Sming tun. Das ist dank der Integration von WSL2 und Powershell auch kein Problem: Sming stellt auch hier die passenden make-Files und Skripte zur Verfügung. Neben VSCode (oder dem freien VSCodium für Linux) wird auch eine Integration von Eclipse CDT angeboten.

VSCode sollte schon installiert sein; die folgenden Schritte konfigurieren es von WSL2 aus und laden eventuell noch nicht installierte Komponenten (wie die C/C++-Extension) automatisch hinzu. Aus dem Projektverzeichnis, das in VSCode bearbeitet werden soll (also etwa `samples/Basic_Blink`), für die gewünschte Architektur Folgendes aufrufen:

```
$ make ide-vscode SMING_ARCH=Esp8266

$ code .
```

Die zweite Zeile ruft VSCode mit dem aktuellen Verzeichnis auf, wenn es im System-Pfad von Windows ist. Weitere Architekturen sind in Host, ESP32 oder RP2040. Beim ersten Aufruf wird der VSCode-Server installiert:

```
$ code .
Installing VS Code Server for
```

Flash-Prozess

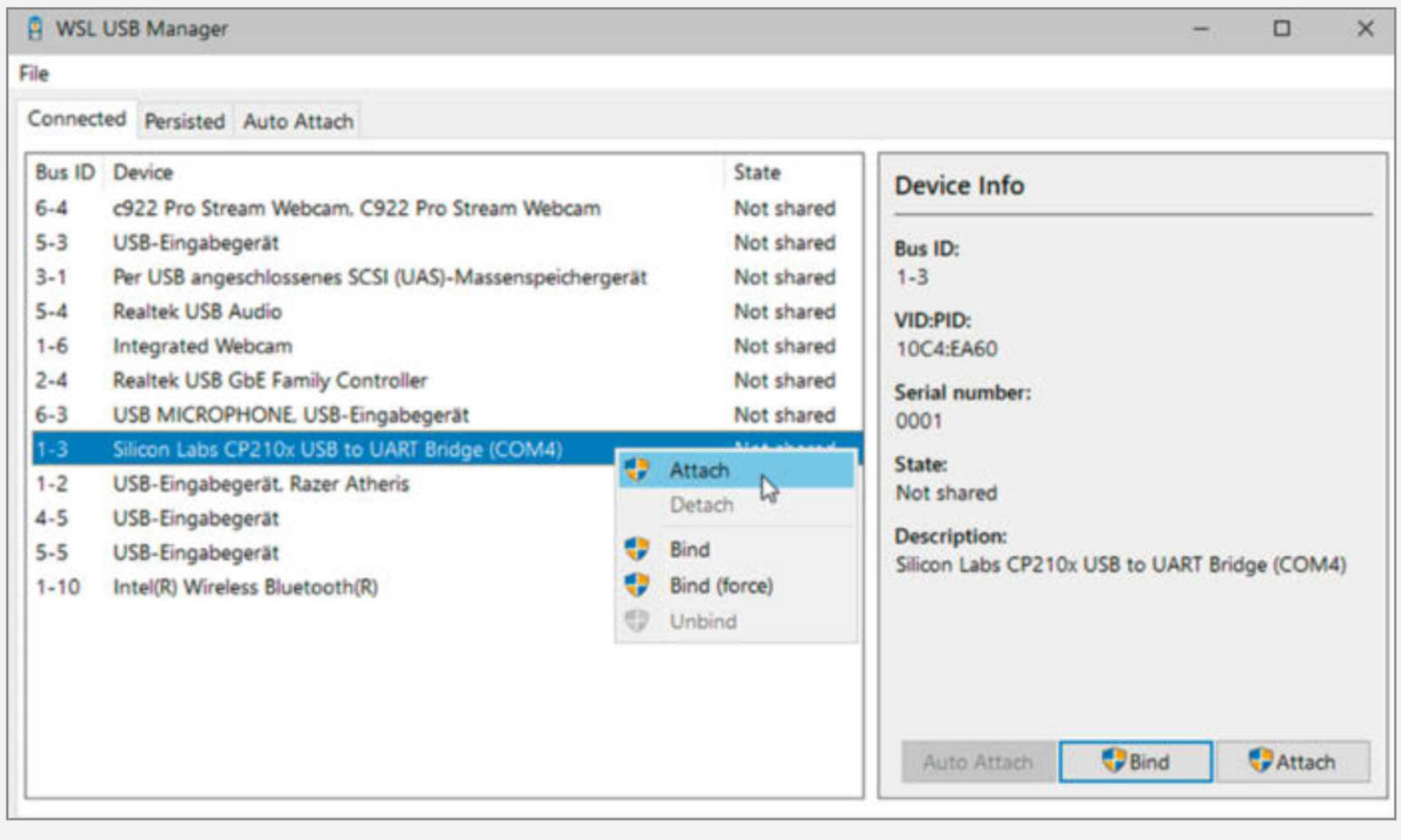
```
$ make flash
...
Wrote 128 bytes (75 compressed) at 0x000fc000 in
0.1 seconds (effective 19.8 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...

make terminal
make[1]: Entering directory
'/home/cw/Sming/samples/Basic_Blink'
Killing Terminal to free /dev/ttyUSB0
...
/usr/bin/python3 -m serial.tools.miniterm
--raw --encoding ascii --rts 0 --dtr 0 /dev/ttyUSB0 115200
--- forcing DTR inactive
--- forcing RTS inactive
--- Miniterm on /dev/ttyUSB0  115200,8,N,1 ---
--- Quit: Ctrl+] | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
```

USB für WSL

Laden Sie den usb-ipd-Installer herunter (siehe Kurzlink) und installieren Sie das Programm durch einen Doppelklick auf den MSI-Installer. Jetzt können in der Powershell USB-Geräte an WSL weitergereicht werden. Das Vorgehen für die Powershell ist im Artikel „WSL2 für Maker“ (siehe „Mehr zum Thema“) und der Dokumentation genau beschrieben. Inzwischen gibt es aber verschiedene GUIs (siehe Kurzlink), die die Arbeit erleichtern. Ich benutze den WSL-USB-Manager von Niccolò Betto. Er stellt eine Oberfläche zur Verfügung, in der man die Geräte einfach per Mausklick weiterleiten kann, ist sehr schlank und besteht nur aus einer einzelnen .EXE, die man einfach irgendwo vorhalten kann.

In der Liste der Devices erscheint das eingesteckte Gerät. Ein Klick auf „Attach“ verbindet es nach einer Windows-Benutzerkonten-Abfrage. Danach taucht das Gerät z. B. als `/deb/ttyUSB0` (je nach Linux-Distribution, hier Ubuntu) in WSL auf. Das war es auch schon. Wird das Gerät wieder ausgesteckt, erscheint es im Persited-Tab, wo es auch gelöscht werden kann. Mit „Auto Attach“ wird es bei jedem Einstecken wieder eingebunden.

Manchmal taucht ein eingestecktes Gerät nicht im Connected-Tab auf, meist wird es aber trotzdem eingebunden, wenn es entsprechend konfiguriert ist. Hier hilft dann ein File/Refresh aus dem Menü. Schließt man das „WSL USB Manager“-Fenster, so beendet man das Programm nicht; es wurde minimiert und findet sich jetzt im Windows-Tray.




```
Linux...
Downloading: 100%
...

Running compatibility check script

Compatibility check successful (0)
```

Öffnet man nun einen Quellcode (.c, .cpp, .h) in VSCode (Bild 2), so wird ein Hinweis rechts unten eingeblendet, dass der Ordner einen Arbeitsbereich enthält (aus der ide-vscode-Konfiguration von oben). Wechselt man in diesen Arbeitsbereich, so ist alles eingerichtet: Links unten erscheint der Hinweis, dass wir uns im Kontext „WSL: Ubuntu-22.04“ befinden (entsprechend der jeweiligen Distribution). Die C/C++-Extensions werden, falls nötig, zur Nachinstallation angeboten.

Öffnen Sie ein Terminal in VSCode (Strg+Ö), so können dort ganz normal die make-Befehle ausgeführt werden. Erfahrende VSCode-Hasen werden sich ab hier zurechtfinden. Die Aufgaben sind in VSCode integriert, sodass man diese wie üblich verwenden kann. In der Suche (F1) findet man die Befehle und Tastenkombinationen.

Mir reicht für meine kurzen Programme allerdings einfach der nano-Editor (Bild 3) in einem Terminalfenster und eine Bash in einem anderen Terminal.

Host Emulator

Immer wieder stolpert man in der Dokumentation oder den Parametern über den „Host Emulator“. Dieser ermöglicht die Erstellung von Sming-Anwendungen als ausführbare Datei, die auf dem Entwicklungs-PC läuft. Dies ist ein Emulator auf Quellcodeebene, mit dem neuer Framework-Code entwickelt und getestet werden kann, bevor er auf ein reales Gerät übertragen wird. Hierzu wird der make-Prozess mit dem Parameter `SMING_ARCH=Host` aufgerufen. Die erste Kompilierung wird wieder etwas länger dauern. Aufgerufen wird das Programm dann mit `make run`. Beenden können Sie das laufende Programm mit Strg+C.

```
$ make SMING_ARCH=Host
$ make run
Basic_Blink: Invoking 'run' for
Host..
make components application
...
Welcome to the Sming Host emulator
Opened "out/.../flash.bin", size =...
...
>> Starting Sming <<

pinMode: 2, 1
digitalWrite: 2, 1
digitalWrite: 2, 0
```

Es wird nicht die unterliegende Hardware (also die CPU und die darum liegende Architektur) emuliert, also können natürlich auch keine hardwarenahen Sachen oder gar Sensoren getestet werden. Ein klassisches Blink gibt aber dennoch den Status der GPIO-Pins aus. Mit der Graphics-Umgebung (siehe auch Kasten „Sming-Entwicklung“) in Sming können die Ausgaben sowohl auf Displays auf dem Host-Computer als auch auf einigen Displaytypen mit echter Hardware angezeigt werden; Ersteres funktioniert auch unter WSL2 einwandfrei.

Dies ist kein Ersatz für echte Emulatoren wie QEMU oder Wokwi (Bild 4). Tatsächlich gibt es schon eine erste Anbindung an Wokwi – einem Emulator, der komplett im Browser läuft –, mit der man die virtuelle Hard- und Firmware in Woki eingebettet in VSCode ausführen, testen und debuggen kann. Das Vorgehen inkl. Debugging ist in der Dokumentation beschrieben.

Arduino-Libraries

In Sming sind schon eine Menge Libraries integriert. Viele Sensoren sucht man hier aber vergebens. Dies ist kein Problem, die meisten Arduino-Libraries sind schnell hinzugefügt.

Man kopiert den Quellcode (siehe Kurzlink) in ein Unterverzeichnis nach `Libraries/`

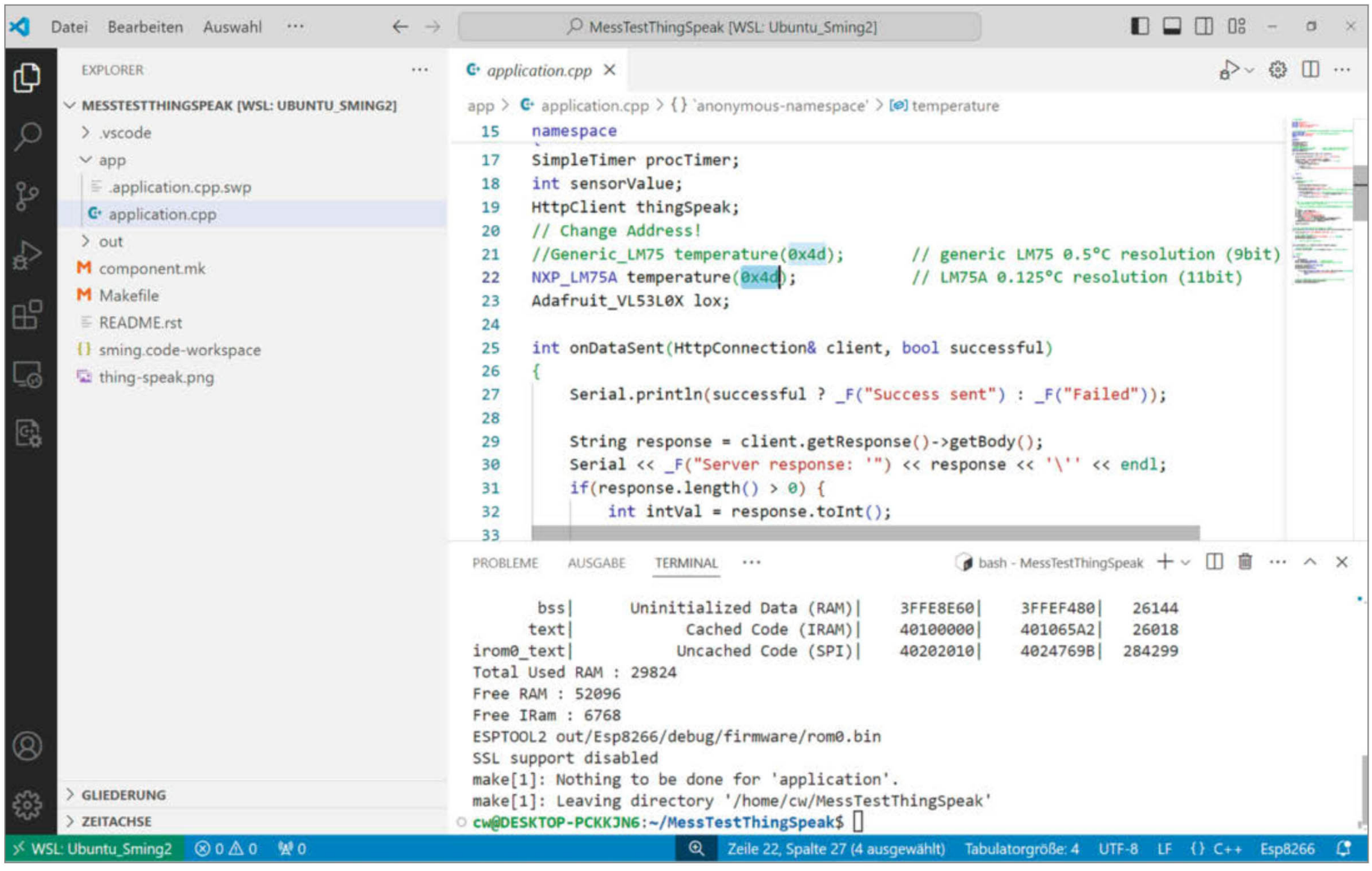


Bild 2: VSCode unter Windows bei der Arbeit in WSL2.

im Sming-Verzeichnis oder man checkt sie per git-Befehl in das Libraries-Verzeichnis aus. Unter WSL2 findet man die WSL-Distributionen im Windows-Explorer unter Linux/Distributionsname und kann hier nach Herzenslust kopieren.

Befindet sich der Quellcode im Hauptverzeichnis der Arduino-Library, so ist alles ok, wenn aber ein Unterverzeichnis wie src/ existiert, muss das in der component.mk-Datei vermerkt werden. In diese Datei können auch Abhängigkeiten von weiteren Bibliotheken eingetragen werden:

```
$ cd $SMING_HOME/Libraries/
$ ls Temperature_LM75_Derived/
CHANGES.md  LICENSE  README.md
component.mk  doc  examples
library.properties  src
$ cd Temperature_LM75_Derived/
$ cat component.mk
COMPONENT_INCDIRS := src
COMPONENT_DEPENDS := OneWire
COM_SPEED_SERIAL  := 74880
```

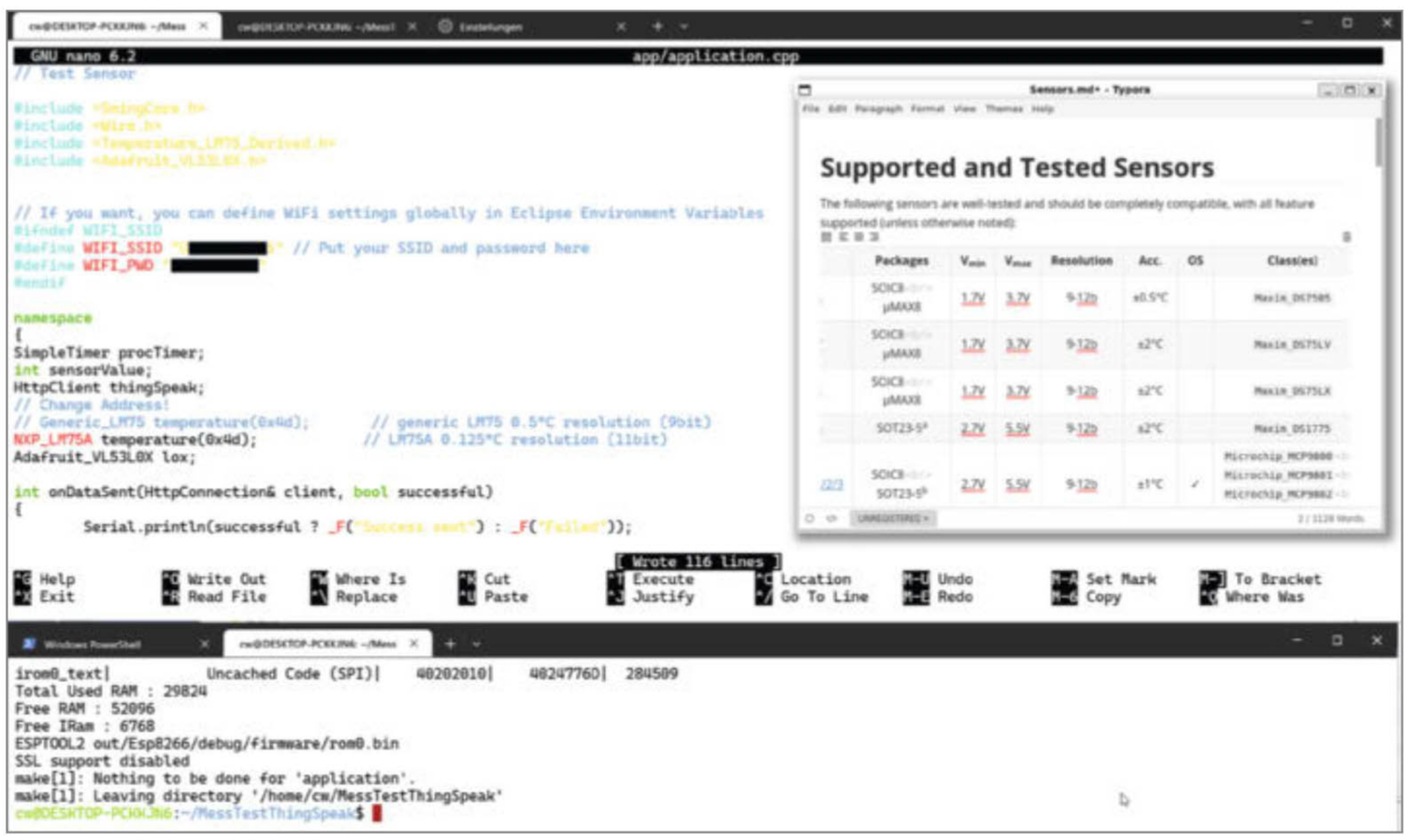


Bild 3: Nano-Editor und Bash. Der Typora-Markdown-Editor läuft übrigens auf dem Linux unter WSL2.

Die Anweisung `COMPONENT_INCDIRS` lässt die make-Skripte die Quellen in `src/` suchen, `COMPONENT_DEPENDS := OneWire` sorgt dafür, dass für die weiters benötigte Bibliothek „OneWire" automatisch inkludiert wird. In dieser Datei können auch noch weitere Aspekte der Kompilierung gesteuert werden, etwa die Geschwindigkeit der seriellen Ports oder für welche Architektur das Programm gebaut werden soll.

Temperatursensor LM75

Mit der beispielhaften Library „Temperature_LM75_Derived" aus dem Abschnitt zuvor kann nun ein LM75-Temperatursensor (I^2C) abge-

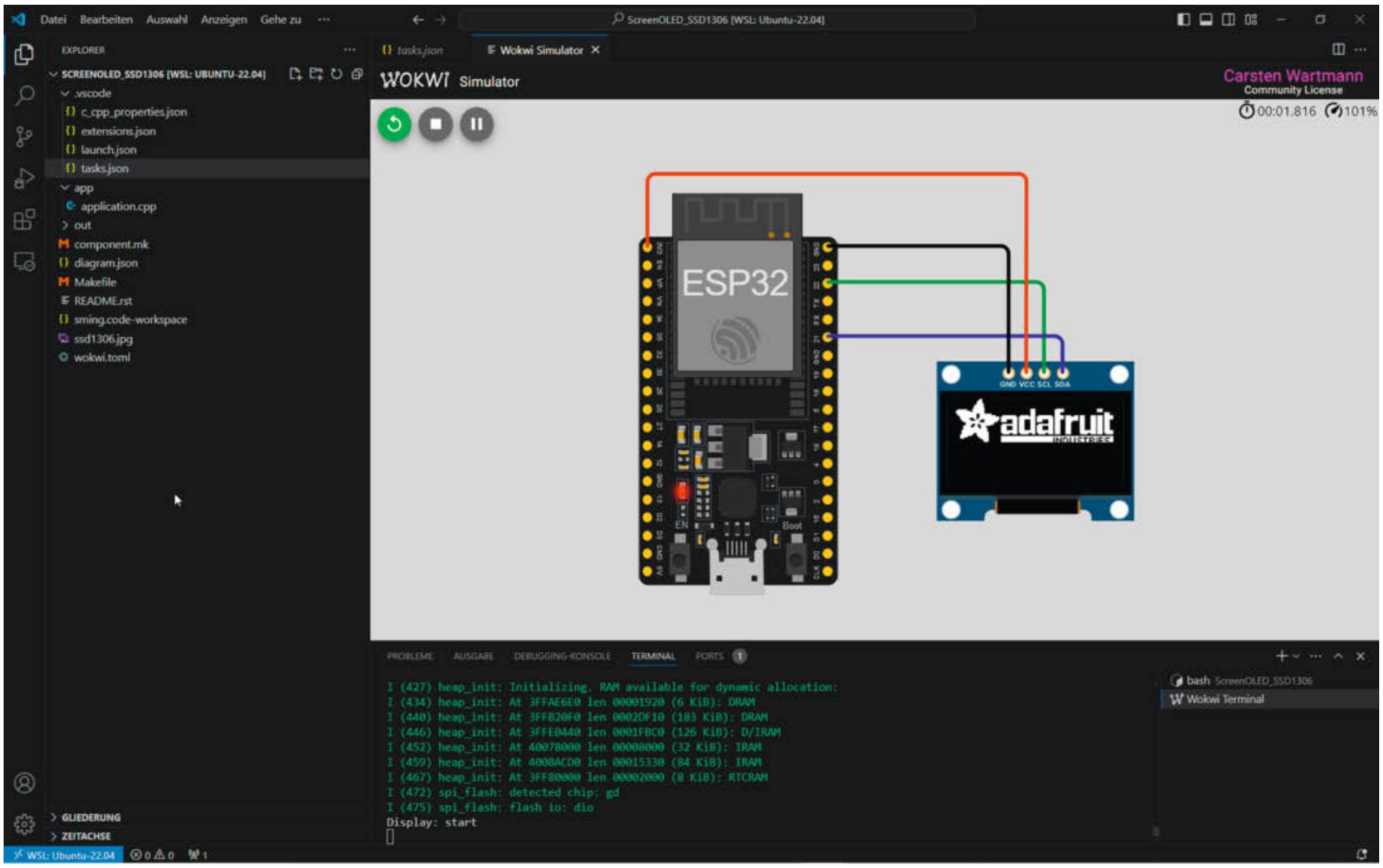


Bild 4: Wokwi-Simulator integriert in VSCode.

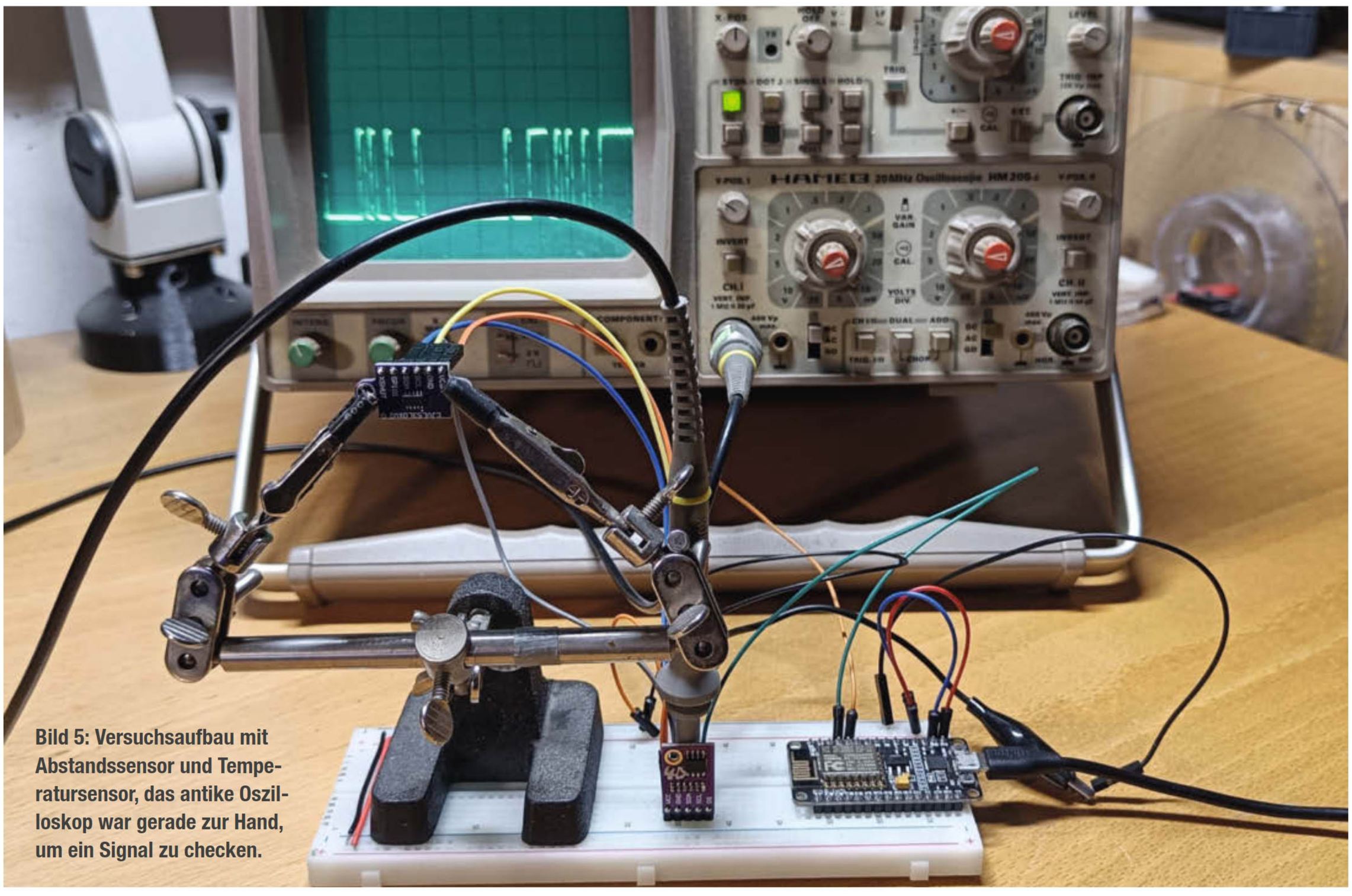


Bild 5: Versuchsaufbau mit Abstandssensor und Temperatursensor, das antike Oszilloskop war gerade zur Hand, um ein Signal zu checken.

fragt werden. Dafür muss zunächst die Verwendung der Library (auch Komponente in Sming genannt) in die Datei „component.mk“ im Projekt-Ordner eingetragen werden. Dazu genügt eine Zeile mit dem Inhalt `ARDUINO_LIBRARIES := Temperature_LM75_Derived`.

Weitere Komponenten werden einfach durch Leerzeichen getrennt an die obige Zeile angefügt. Der Quellcode sieht dann auszugsweise so aus:

```
#include <SmingCore.h>
#include <Temperature_LM75_Derived.h>

namespace
{
SimpleTimer procTimer;
// Change Address!
// LM75A (11bit)
NXP_LM75A temperature(0x4d);
}

...
```

In dem Beispiel für den LM75-Sensor mit oben vorgestellter Arduino-Library wird also das Include-File eingebunden und dann wie in Arduino benutzt. Den Quellcode finden Sie auf unserem GitHub (siehe Kurzlink). In dem doc/-Verzeichnis der Library ist aufgelistet (siehe auch Bild 3, nano-Editor), welche Typen von LM75-Sensoren Sie mit welchem Include-Befehl benutzen können: Im Beispiel wird ein NXP LM75A benutzt, der eine 11-Bit-Auflösung bereitstellt, also 0,125-°C-Schritte misst, wo ein LM75 nur 0,5-°C-Schritte misst. Die Adresse von I²C-Geräten kann man übrigens sehr einfach mit dem Beispiel „Basic_ScannerI2C“ aus dem samples-Verzeichnis von Sming herausbekommen.

IoT und so

Jetzt musste aber noch ein ESP seine WLAN-Fähigkeiten zeigen. Beispiele gibt es in dem „samples“-Ordner von Sming genug, da ich aber kein klassisches Smarthome besitze, fiel mir ein Beispiel besonders ins Auge: Daten an ThingSpeak senden. Für den Artikel „IoT für Maker“ (Siehe „Mehr zum Thema“) hatte ich mir seinerzeit mal ein Konto eingerichtet. So war es dann auch nur eine Sache von Minuten, das Passwort zu suchen, die WLAN- und ThingSpeak-Anmeldedaten in den Code einzutragen, und schon sendete mein ESP munter Daten zu ThingSpeak.

Der Code war fix erweitert und sendet nun die Daten des LM75A-Temperatursensors und eines VL53L0X-Time-of-Flight(ToF)-Abstandssensors. Die Unterstützung für den VL53L0X-Sensor ist bereits in Sming integriert, muss aber auch in `compoments.mk` gelistet sein, also `ARDUINO_LIBRARIES := Temperature_LM75_Derived Adafruit_VL53L0X`.

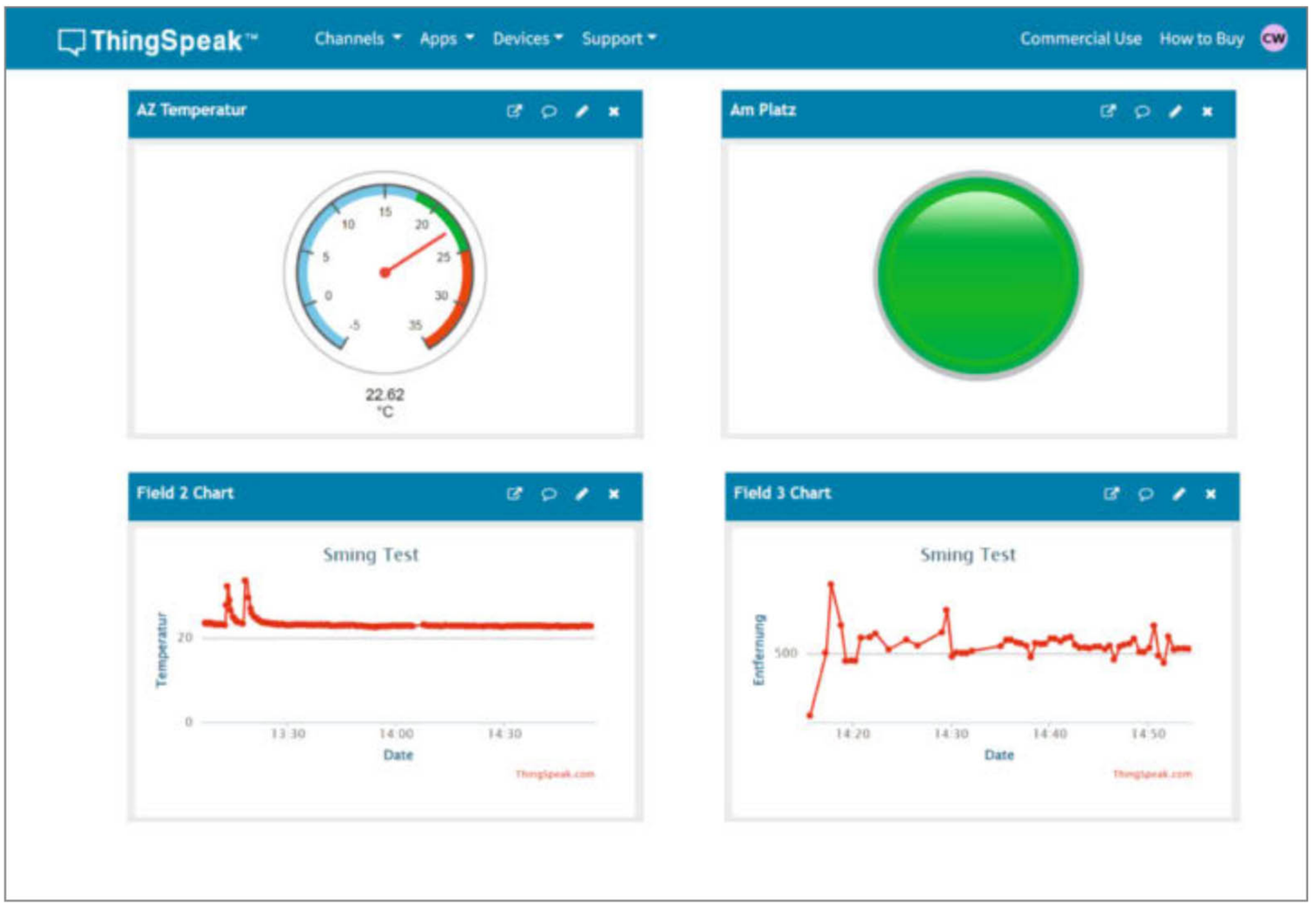


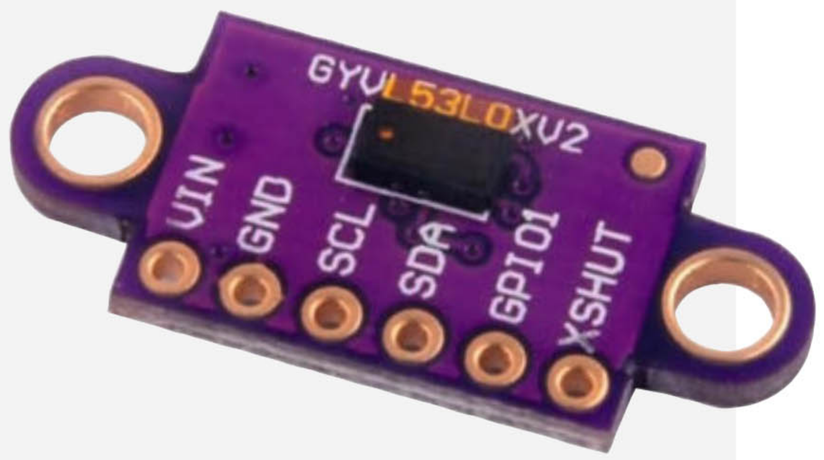
Bild 6: Das ThingSpeak-Dashboard

Damit war schnell der Prototyp (Bild 5) einer Raumüberwachung meines Büros gemacht. So könnte man im Smarthome seine Anwesenheit vor dem Rechner feststellen und entsprechend die Heizung, Klimaanlage, Luftfilter oder Ähnliches steuern oder nur den Maus-Wiggler anwerfen, wenn man nicht am Platz ist, um Anwesenheit vorzutäuschen. Das ThingSpeak-Dashboard (Bild 6) ist übrigens recht spartanisch ausgerüstet, was die Verarbeitung und Visualisierung der Daten angeht; möchte man mehr, so wird man mit Mathlab konfrontiert, das mir aber eher für rein statische Auswertungen gemacht erscheint. Aber für den Versuch hat es gut funktioniert und die anderen Dienste, sei es lokal oder im Internet, arbeiten praktisch alle gleich.

—caw

Time-of-Flight-Sensor

Time-of-Flight-Sensoren (ToF) sind Sensoren zur Entfernungsmessung und der räumlichen Erfassung von Szenarien. Diese Sensoren senden Laserimpulse aus und messen dann die Zeit, die das Licht benötigt, um von einem Objekt reflektiert zu werden und zum Sensor zurückzukehren. Die Entfernung zum Objekt wird durch die Lichtgeschwindigkeit und die gemessene Zeitdauer berechnet. Die Formel ist dann Entfernung = (Lichtgeschwindigkeit × Zeit)/2.



ToF-Sensoren unterscheiden sich von anderen Entfernungsmessgeräten wie Ultraschall- oder Infrarotsensoren durch ihre höhere Genauigkeit, größere Reichweite, höhere Geschwindigkeit und die Fähigkeit, in einer Vielzahl von Umgebungen und Materialien zu funktionieren.

Der im Artikel verwendete Sensor VL53L0X von STMicroelectronics ist mit seinem kleinen IR-Laser (viele Smartphonekameras können das IR-Licht sehen) und dem integrierten Prozessor eine sehr moderne Variante, die kompakt, energieeffizient und günstig ist.

Neuer Input für Maker

Make Elektronik Special

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

Heft + Experimentierset für 44,95 €

shop.heise.de/make-elektronik21

Inklusive Experimentierset und Breadboard

Make Operationsverstärker Special

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

Heft + Experimentierset für nur 49,95 €

shop.heise.de/make-opv

Inklusive Experimentierset

Make PI Pico Special

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

Heft + Raspberry Pi Pico für 24,95 €

shop.heise.de/make-pico

Inkl. Raspberry Pi Pico RP2040

WILLKOMMEN IM NEUEN IOT-ÖKOSYSTEM

Mit LoRaWAN und C-Programmierung über lange Distanzen messen und steuern

Heft + LoRaWAN-Set

Im Make Special LoRaWAN führen Sie 15 Artikel Schritt für Schritt in die Hardware, LoRaWAN, The Things Network und ihre Programmierung ein. Es wird Schritt-für-Schritt erklärt, wie Sie aus den mitgelieferten LoRaWAN- und Sensormodulen einen Umweltsensorknoten entwickeln und noch vieles mehr.

DARUM GEHT'S:

- Einstieg in STM-Mikrocontroller
- Programmieren mit der STM32CubeIDE
- Spannungen, Temperatur und Luftfeuchte messen
- Mit LoRaWAN senden und empfangen
- Daten im The Things Networks verarbeiten
- Werte mit TagoIO visualisieren
- Refresher: Programmieren in C

JETZT BESTELLEN!

Make Special LoRaWAN inkl. Experimentierset für 64,90 €

shop.heise.de/make-lorawan24

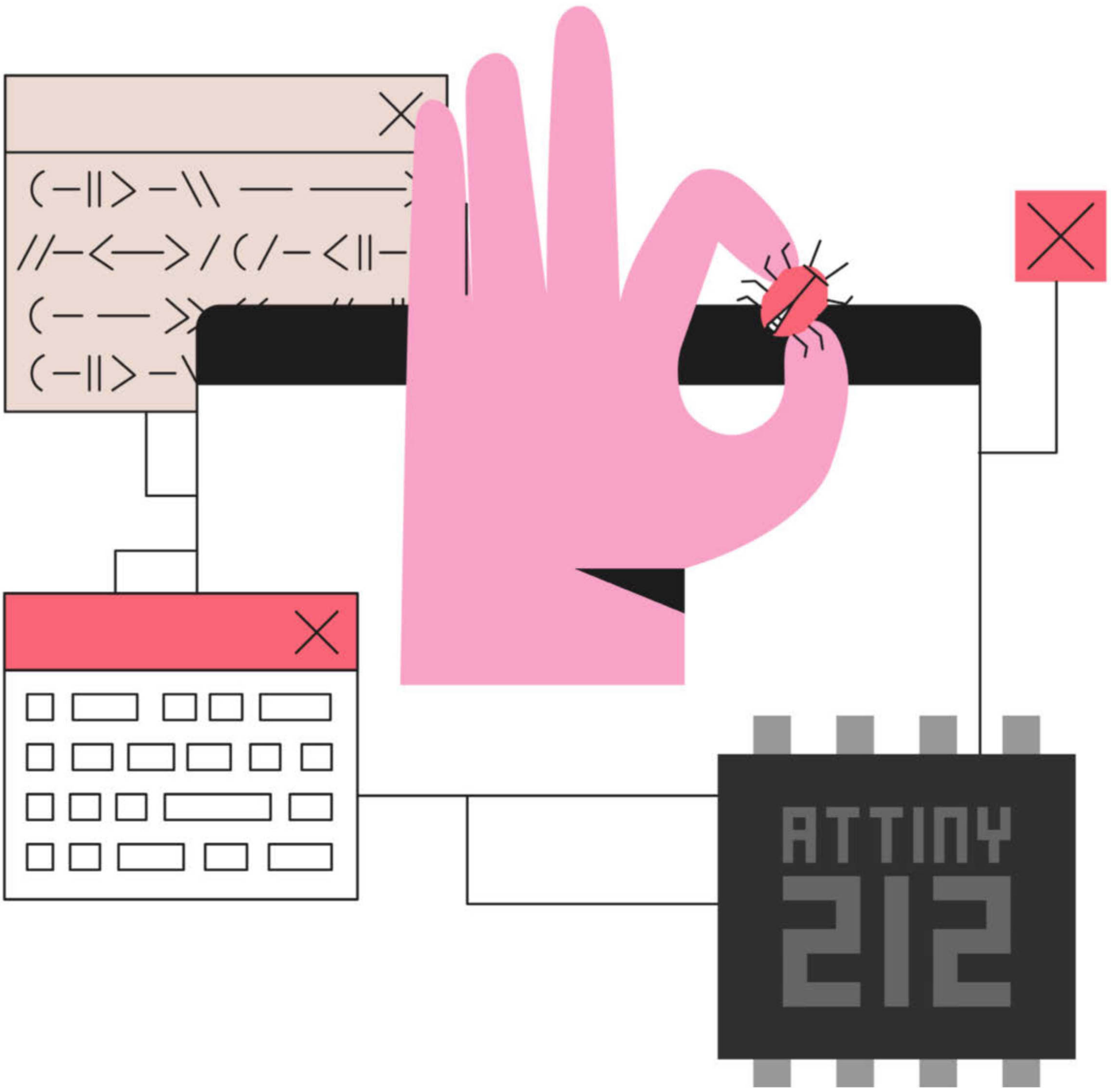
Generell portofreie Lieferung für Heise Medien- oder Maker Media Zeitschriften-Abonnenten oder ab einem Einkaufswert von 20 € (innerhalb Deutschlands). Nur solange der Vorrat reicht. Preisänderungen vorbehalten.

heise shop

AVR-Programme debuggen, Teil 1

Mit einem Debugger kann man professionell Programmierfehler finden, die sich sonst kaum aufspüren oder reproduzieren lassen. Wie das bei AVR-Mikrocontrollern mit dem MPLAB Snap In-Circuit Debugger funktioniert, zeigen wir Ihnen an einem ATtiny212 und begleiten Sie auf Ihrer ersten Pirsch nach Bugs.

von Florian Schäffer



shutterstock.com/ KatePilko

Wenn ein Code nach dem Flashen auf den Mikrocontroller nicht funktioniert, wie er soll, geht es an die Fehlersuche: Liegt es an einem vergessenen Semikolon, einem Speicherüberlauf, einem Denkfehler oder etwas anderem? Je geübter man beim Programmieren wird, desto einfacher fällt es einem, oft triviale Fehler zu erkennen. Der Compiler hilft auch dabei, indem er viele Probleme schon vorab erkennt und abbricht oder wenigstens Warnungen ausgibt. Logikfehler im Programmablauf oder falsch gesetzte Register zu erkennen, erfordert aber mehr Aufwand und ein Verständnis für die internen Abläufe in der Hardware.

Dabei kann ein Debugger hilfreich sein, mit dem sich die Codeausführung an jeder beliebigen Stelle anhalten und der Speicher untersuchen lässt. In der Make 2/24 haben Sie bereits gelernt, wie man AVR-Chips mit Microchip Studio programmiert. Dieses Mal erkläre ich exemplarisch am ATtiny 212, wie man den Hardware-Debugger MPLAB Snap in Microchip Studio verwendet, um Fehler im Code aufzuspüren. Die verwendeten Programmbeispiele gibt es als Download im GitHub-Repository des Projekts (siehe Link in der Kurzinfo).

Kurzinfo

- **Einstieg ins Debugging mit ATtiny212**
- **Debugger-Hardware kennenlernen**
- **Mit Microchip Studio und dem MPLAB Snap Fehler finden**

Checkliste

Zeitaufwand:
3 bis 5 Stunden

Kosten:
60 bis 80 Euro

Material

- **MPLAB Snap In-Circuit Debugger/ Programmer**
- **ATtiny212 auf SMD-Adapter**
- **MicroUSB-Kabel**
- **Breadboard mit Zubehör**
- **Kondensator,** mit 100 nF
- **Widerstände,** 2 × 220 Ω, 1 × 10 kΩ
- **LEDs**, rot und grün
- **5-V-Spannungsquelle** ggf. als Aufsatz fürs Breadboard oder in Form eines Arduino Uno

Werkzeug

- **Lötkolben**
- **Seitenschneider, Cutter, Dremel o. Ä.**
- **Spitze Metallpinzette oder Drahtbrücke**

Mehr zum Thema

- Florian Schäffer, ATtiny statt Arduino, Make 2/24, S. 72
- Felix Pfeifer, ARM Debuggen mit Eclipse, Make 4/16, S. 74
- Daniel Bachfeld, Speicherverbrauch in Mikrocontrollern, Make 3/24, S. 92

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

Debugging mit Textausgabe

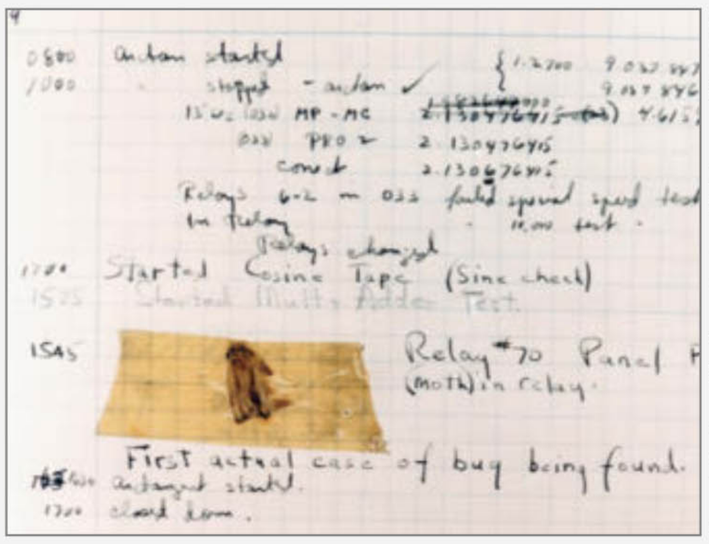
Wenn man Fehler in einem Code finden möchte, kann man über einen seriellen Monitor mit `Serial.print()` in der Arduino IDE oder mit `print()` bei MicroPython Werte aus dem Programm ausgeben – und so oft schon feststellen, an welcher Stelle der Code hakt. Das ist sehr bequem, denn die Methode benötigt kein zusätzliches Tool und bietet eine intuitive Nutzung. Dadurch ist der serielle Port aber belegt oder kann nicht für Ausgaben genutzt werden, wenn er für eine serielle Kommunikation mit anderen Komponenten benötigt wird – was auch zu Problemen bei der Programmübertragung führen kann.

Bei einem Mikrocontroller wie dem ATtiny mit nur vier I/O-Pins wäre eine serielle Schnittstelle, die allein für das Debugging zwei Leitungen belegt, in den meisten Fällen eine Verschwendung von Anschlüssen. Außerdem kostet es Speicherplatz, die Funktion einzubinden, und die Ausgaben benötigen selbst bei hohen Baudraten relativ viel Zeit, was bei zeitkritischen Aktionen stört.

Probleme mit Speicherüberläufen etc. (dazu später mehr) sind sowieso schon schwer

Entwanzen Sie Ihr Programm

Die Begriffe Debuggen und Debugger stammen vom englischen Wort „bug“ ab, das für Käfer oder Wanze steht – das Programm wird also von Ungeziefer befreit. Als informationstechnische Anekdote kann das abgebildete Betriebsprotokoll aus den 1940ern angesehen werden, in dem tatsächlich eine Motte – also im weitesten Sinn ein Bug – in einem Computer (Harvard Mark II) gefunden wurde. Wie auch Computerviren sind Bugs aber nur eine Allegorie für ein (unerwünschtes) Phänomen oder Verhalten in der IT.



Naval Surface Warfare Center, Dahlgren, VA., 1988

Bild 1: Der erste echte Computer-Bug (Bildausschnitt)

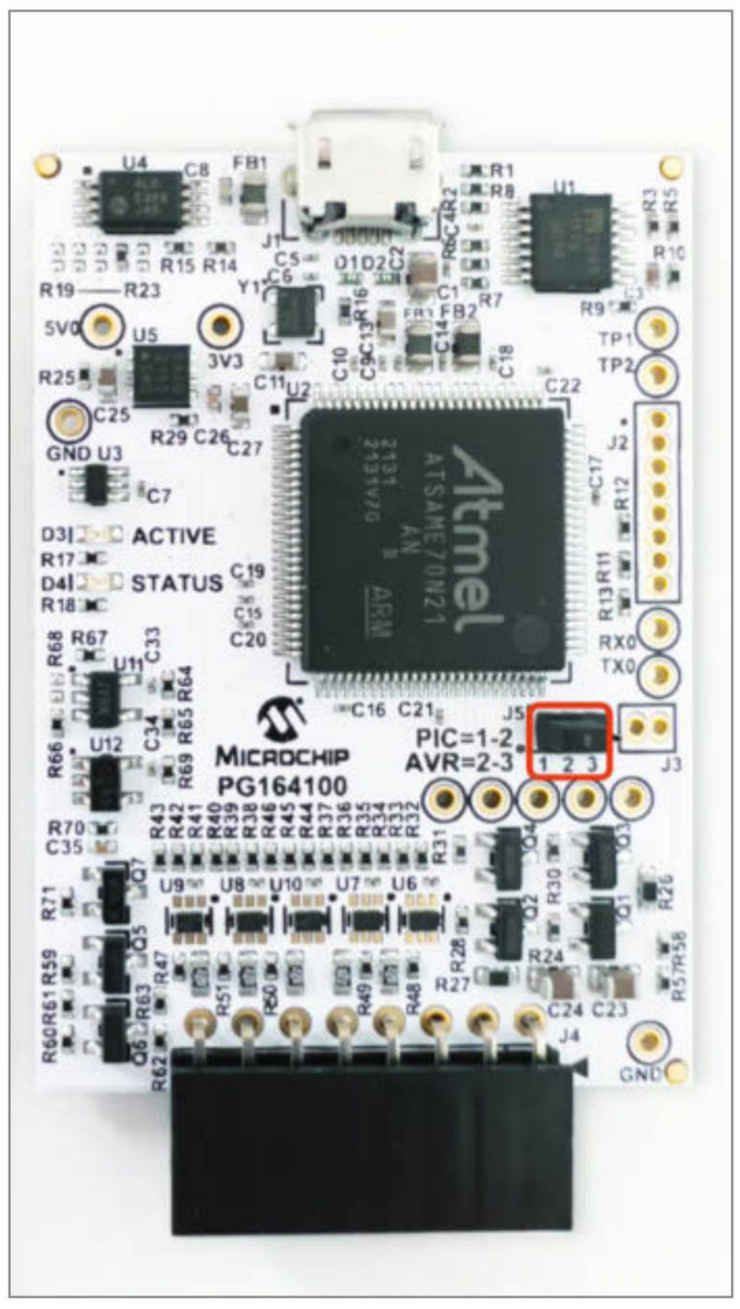


Bild 2: Den aktuellen MPLAB Snap erkennen Sie am Jumper J5 (rot markiert), der beim Vormodell fehlte.

	ICSP	MIPS EJTAG	CORTEX SWD	JTAG	ISP	dW	UPDI	PDI
1	MCLR	MCLR	MCLR					
2	VDD	VIO_REF	VTG	VTG	VTG	VTG	VTG	VTG
3	GND	GND	GND	GND	GND	GND	GND	GND
4	DAT	TDO	SWO	TDO	MISO		DAT	DAT
5	CLK	TCK	SWCLK	TCK	SCK			
6	AUX			RESET	RESET	RESET		CLK
7		TDI		TDI	MOSI			
8		TMS	SWDIO	TMS				

Bild 3: Je nach Art der Debugging-Methode verwendet man unterschiedliche Leitungen des MPLAB Snap.

zu finden und wenn sich der Code ändert, weil Debug-Ausgaben hinzugefügt oder entfernt werden, ändert sich auch das Speicherabbild. Das kann dazu führen, dass ein Fehler, der vorher nicht auftrat, plötzlich in Erscheinung tritt oder wieder zu verschwinden scheint.

DIY mit viel drumherum

Wer sich nach einer Low-Cost-Lösung fürs Debuggen umschaut, wird schnell fündig: Ein weiteres Arduino-Board oder ein USB-Seriell-Wandler und etwas Software machen eingeschränktes Debugging möglich. Dazu muss man aber immer eine Bibliothek (etwa GDB Remote Serial Protocol) in den eigentlichen Code einbinden und diesen mit Anweisungen ergänzen, die das Programm anhalten oder Werte von Variablen ausgeben. Dadurch wird der eigentliche Code aufgebläht und modifiziert, sodass im Prinzip wieder gar nicht das reine Programm untersucht wird. Zudem benötigt man wie beim Ansatz mit Textausgabe die serielle Schnittstelle – insgesamt also keine befriedigende Methode.

Zu schön, um wahr zu sein

Falls Sie sich doch nach anderen Debuggern umschauen, sollten Sie zumindest bei billigen Geräten für unter zehn Euro vorsichtig sein, die manchmal als JTAG ICE oder gar STK500 beworben werden. Der echte Atmel-ICE-Basic-Debugger kostet gut 200 Euro, ebenso der STK-500, der eigentlich auch kein kleiner USB-Stecker ist, sondern als Platine in einem Starterkit verkauft wird, auf der man AVR-Chips im DIL-Gehäuse debuggen kann.

Die günstigen Nachbauten beherrschen in der Regel ausschließlich JTAG und sind somit nur für wenige große AVRs geeignet, nicht für den Arduino Uno oder ATtiny.

Eigenständige Debugger

Die optimale Lösung, um Fehlern in seinem Code auf die Spuren zu kommen, ist ein Hardware-Debugger, der mit dem Mikrocontroller zusammenarbeitet, keine Veränderungen am Programm erfordert und kein zusätzliches Byte im Speicher belegt. Dafür muss der Mikrocontroller eine dedizierte Debugging-Schnittstelle mit interner Hardware besitzen. Dann kann man mit ihm auch auf alle internen Register und Variablen zugreifen und zu jedem beliebigen Zeitpunkt sehen, was in ihnen gespeichert ist.

Den Debugging-Vorgang steuert dabei die IDE (hier Microchip Studio) und nutzt Breakpoints (Haltepunkte), die den Programmablauf pausieren, damit man ihn schrittweise analysieren kann. Das nennt man auch In-Circuit Debugging. In der IDE lässt sich dann verfolgen, welche Anweisung gerade ausgeführt wird, um so (unerwartete) Sprünge oder Verzweigungen zu erkennen.

Zudem kann man jeden Debugger auch als Programmer nutzen, um kompilierten Code auf einen beliebigen (kompatiblen) Mikrocontroller zu übertragen. Damit sparen Sie sich einen zusätzlichen Programmier-Adapter und können auch ohne Arduino-Bootloader Controller flashen, die dafür nicht einmal einen USB-Anschluss besitzen müssen.

Ein Pin reicht

Sie haben im Zusammenhang mit Debugging vielleicht schon von dem weit verbreiteten Debugging-Standard JTAG gehört (benannt nach den Entwicklern: Joint Test Action Group). Mit ihm lassen sich sehr viele Mikrocontroller unterschiedlicher Hersteller debuggen, man benötigt dafür allerdings vier bis fünf Signalleitungen, was vor allem bei Prozessoren mit wenigen Anschlüssen verschwenderisch sein kann, da die Leitungen während des Debuggens belegt sind.

Um auch die kleinen ATtiny und andere AVR-Chips vernünftig debuggen zu können, hat der Hersteller Microchip daher zwei eigene Hardware-Debugging-Schnittstellen entwickelt, die beide mit nur einem einzelnen Anschluss auskommen: Unified Program and Debug Interface (UPDI) und debugWIRE. Sie kommunizieren über den Reset-Pin, aber in unterschiedlicher Weise, sodass man die Funktion abhängig vom Mikrocontroller wählen muss. ATtiny lassen sich mit UPDI debuggen und, wie bereits in der Make 2/24 gezeigt, auch programmieren. DebugWIRE empfiehlt sich etwa für den ATmega328, der auf den Arduino Uno bis einschließlich Rev. 3 sitzt (mehr dazu in der nächsten Ausgabe).

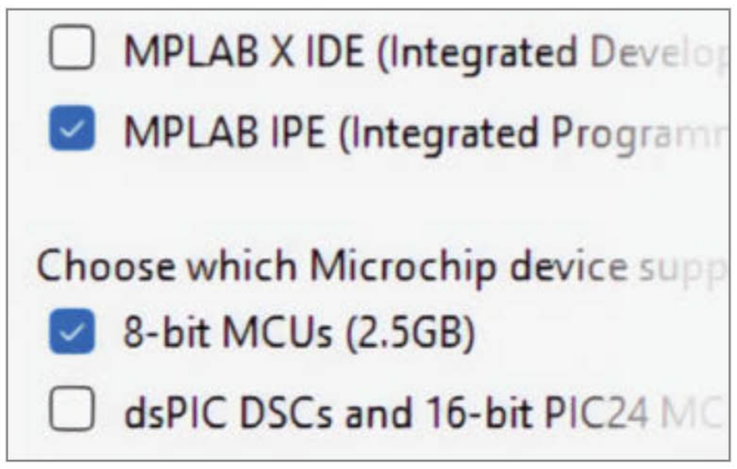


Bild 4: Installieren Sie nur die MPLAB IPE mit der Unterstützung für 8-Bit-MCUs.

Professionelle Debugger vom Hersteller

Es gibt von Microchip mehrere Debugger in unterschiedlichen Preis- und Leistungsklassen. Der aktuelle PICKit 5 wäre eigentlich meine erste Wahl für den Einstieg gewesen, weil er für etwas über 100 Euro die meisten Funktionen bietet – und wenn man sich schon einen Debugger kauft, dann sollte es eine langfristige Investition sein. Allerdings wird er derzeit nicht von Microchip Studio unterstützt, sondern nur von der MPLAB X IDE, die wiederum nicht mit Arduino-Code kompatibel ist.

Also fiel die Wahl auf Microchips günstigsten Debugger, den MPLAB Snap für ca. 50 Euro (Bild 2), der fast alle Debugging-Arten beherrscht. Lediglich die High-Voltage-Programmierung fehlt, die aber nur für einen eher seltenen Spezialfall benötigt wird. Das Gerät kommt als offene Platine ohne Gehäuse, sodass man sich noch selbst eines basteln muss. Eine passende Hülle können Sie sich im GitHub-Repository des Projekts für Ihren 3D-Drucker herunterladen. Wenn Sie zusätzlich die kleine Übersicht aus Bild 3 rückseitig auf das Gehäuse kleben, wissen Sie immer, welche Pins Sie für die unterstützten Kommunikationsprotokolle benötigen. Das fehlende USB-Kabel und die Jumper-Kabel dürften sich bei jedem Maker finden.

Kurz vor Redaktionsschluss haben wir festgestellt, dass es eine neue Revision des Snap gibt. Leider finden sich beim Hersteller und den Händlern dazu keinerlei Informationen. Zurzeit wird überall noch das Foto der vorherigen Platine gezeigt. Das neue Modell erkennen Sie nach dem Auspacken an einem aufgelöteten Pinheader als Jumper J5.

Firmware auf den Snap flashen

Frisch gekaufte Snap werden mit einer Firmware ausgeliefert, die nicht mit aktuellen Versionen des Microchip Studio funktioniert. Daher muss man erst ein Firmware-Update durchführen. Um die neue Version aufzuspielen, benötigen Sie die (kostenlose) Software MPLAB IPE (Integrated Programming Environment). Laden Sie sich dazu die MPLAB X IDE 6.05 herunter, in der die IPE enthalten ist

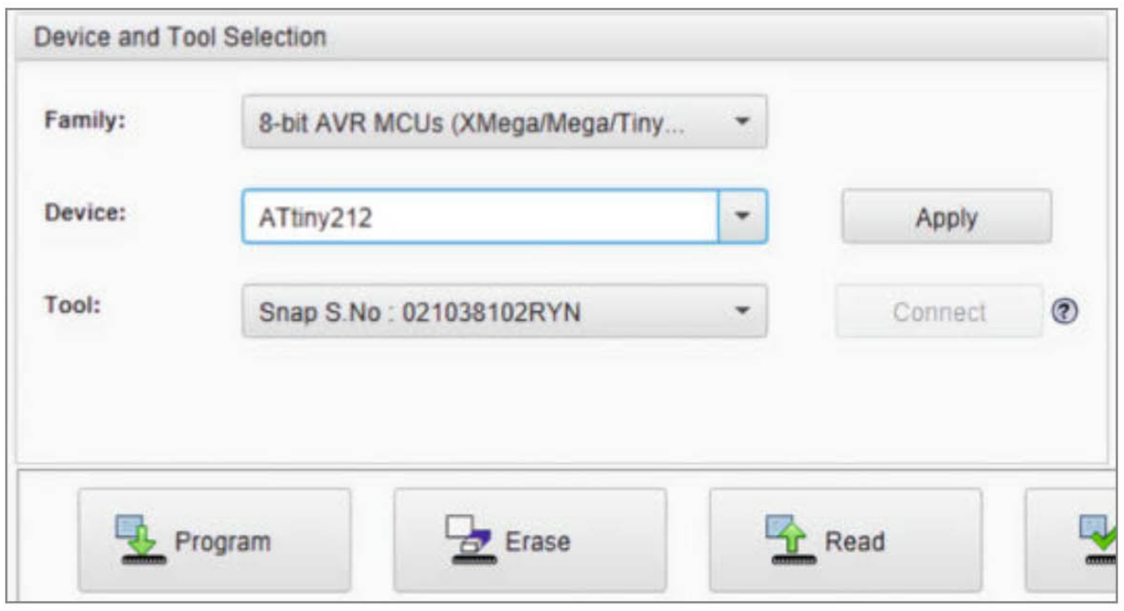


Bild 5: Nach dem Update im Recovery-Modus müssen Sie in der IPE noch angeben, welche Produkt-Familie Sie programmieren möchten.

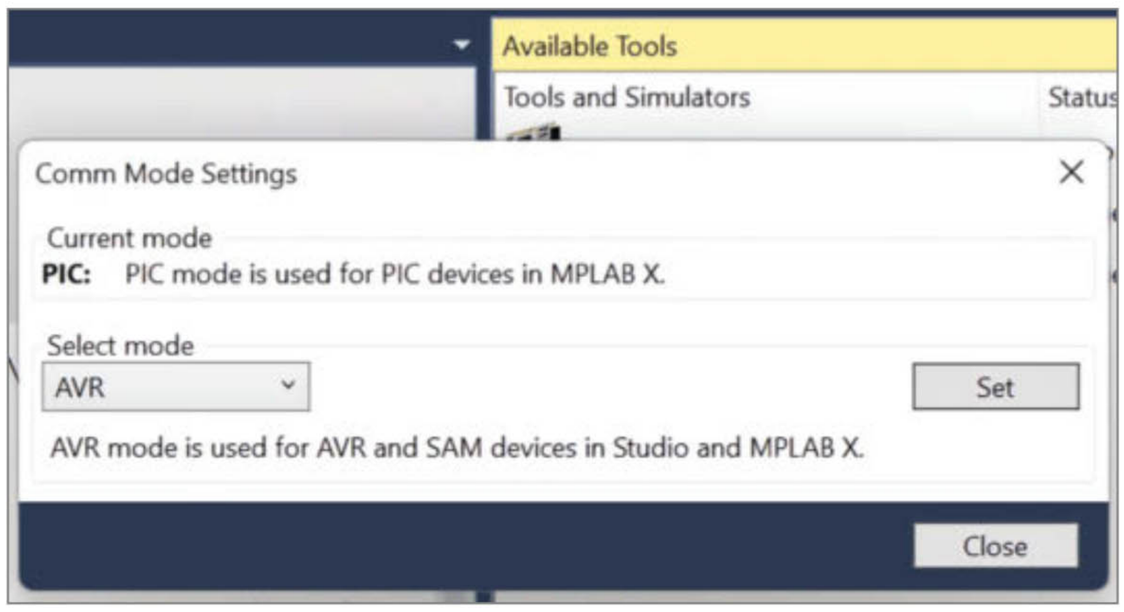


Bild 6: In Microchip Studio lässt sich der Kommunikationsmodus auf AVR einstellen.

(siehe Link in der Kurzinfo). Das „X" ist wichtig und diese Version funktioniert garantiert auch mit dem älteren Snap. Wählen Sie bei der Installation nur die IPE und die Unterstützung für 8-Bit-MCUs aus (Bild 4).

Schließen Sie nach der Installation den Snap mittels USB-Kabel an Ihren PC an und starten Sie die MPLAB IPE. Dort rufen Sie im Menü „Tools" den Punkt „Hardware Tool Emergency Boot Firmware Recovery" auf, um das Firmware-Update zu starten. Lassen Sie sich nicht von der gefährlich klingenden Bezeichnung beirren: Der Ablauf ist zwar etwas umständlich, aber die Schritte sind leicht auszuführen und erlauben nach der Erfahrung des Autors auch Fehlversuche, sodass Sie einen gescheiterten Flash-Vorgang wiederholen können.

Sobald Sie in dem Recovery-Fenster auf „Next" klicken, führt das Programm Sie schrittweise durch den Prozess. Klicken Sie auf „Next", um zur Auswahl des Debuggers zu kommen, und selektieren Sie dort „MPLAB Snap". Danach versetzen Sie den Snap in den Recovery-Modus.

- Schließen Sie die beiden Kontakte an J3 mit einer spitzen Metallpinzette oder einem Draht für eine Sekunde kurz.
- Warten Sie mindestens 10 Sekunden.
- Trennen Sie den Snap vom USB-Kabel
- … und stecken Sie ihn gleich wieder an.

Danach befindet sich der Snap im Recovery-Boot-Modus, den Sie daran erkennen, dass die beiden LEDs (ACTIVE und STATUS) ausgeschaltet sind.

Klicken Sie auf „Next", woraufhin Sie gebeten werden, den Snap wieder vom USB-Kabel abzuziehen. Nach einem weiteren „Next" schließen Sie ihn entsprechend der Meldung wieder an und starten mit „Next" den eigentlichen Update-Vorgang.

Nach dem erfolgreichen Updaten müssen Sie den Snap erneut einmal vom PC entfernen und wieder verbinden. Wählen Sie anschließend im IPE-Fenster unter „Device und Tool

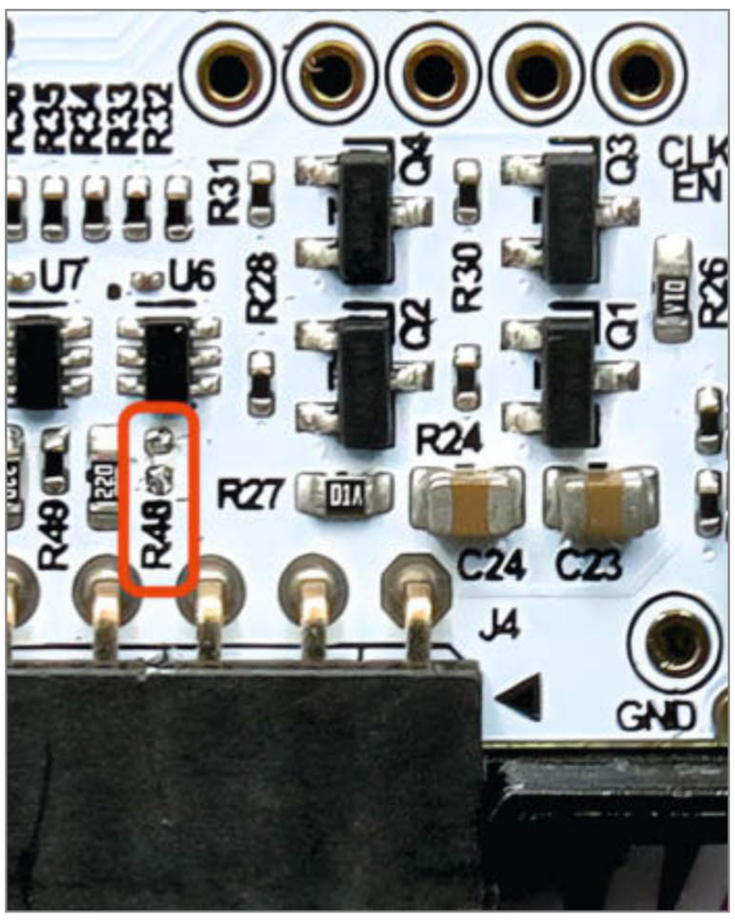


Bild 7: Entfernen Sie den Widerstand R48 nur bei dem älteren MPLAB Snap.

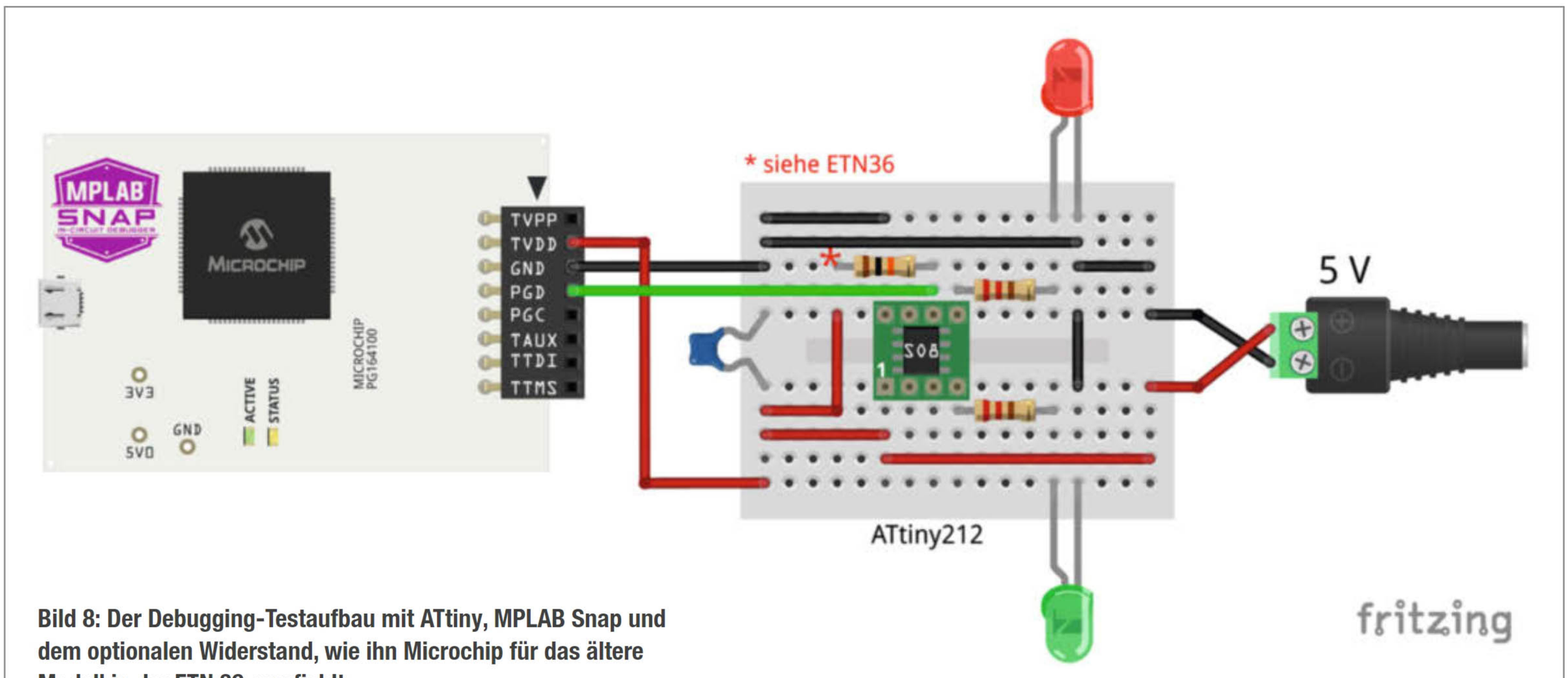


Bild 8: Der Debugging-Testaufbau mit ATtiny, MPLAB Snap und dem optionalen Widerstand, wie ihn Microchip für das ältere Modell in der ETN 36 empfiehlt.

attiny1 main.c

```
#define F_CPU 3333333UL
#include <avr/io.h>
#include <util/delay.h>
int8_t value;

int main(void){
  value++;
  PORTA.DIRSET = (1 << 2);

  for (int8_t cnt = 120;
            cnt < 200;
            cnt++){

    PORTA.OUTSET |= (1 << 2);
    _delay_ms(50);

    PORTA.OUTCLR |= (1 << 2);
    _delay_ms(50);
  }
  while (1);
}
```

Selection“ bei „Family“ den Eintrag „8-Bit AVR MCUs…“ aus und bei Device „ATtiny212“ (Bild 5). Klicken Sie danach auf „Apply“ und dann auf „Read“. In der Ausgabe darunter sehen Sie anschließend, dass die IPE noch die notwendige Firmware für die AVR-Programmierung auf den Snap lädt.

Danach ist der Debugger einsatzbereit, aber derzeit noch auf die Kommunikation mit PIC-Mikrocontrollern eingestellt. Wechseln Sie zu Microchip Studio, um den Modus auf AVR zu ändern.

Auf AVR-Debugging umstellen

Klicken Sie in Microchip Studio im Menü „View“ auf „Available Microchip Tools“. Auf der rechten Seite des Programmfensters erscheint daraufhin eine Auflistung der verfügbaren Werkzeuge. Klicken Sie mit der rechten Maustaste auf den MPLAB Snap und wählen Sie im Menü „Communication mode“ aus. Daraufhin erscheint ein Pop-up-Fenster, das Ihnen zunächst anzeigt, welcher Modus gerade aktiv ist. Wählen Sie in dem darunterliegenden Dropdown-Menü „AVR“ aus und bestätigen Sie mit „Set“ (Bild 6). Danach startet der Snap sich neu und ist auf den AVR-Modus umgestellt.

Snap für UPDI anpassen

Als Letztes müssen Sie den Snap noch leicht modifizieren, damit er über UPDI debuggen kann. Wenn Sie ein neueres Modell (02-10381/02) haben, wie in Bild 2 gezeigt, stecken Sie den Jumper bei J5 so, dass er die Pins 2 und 3 miteinander verbindet.

Bei dem älteren Modell (02-10381-R1) gibt es diese Pins nicht. Hier müssen Sie den Widerstand R48 (Bild 7) mit einem feinen Lötkolben entfernen oder mit einem Schneidewerkzeug durchtrennen (ohne benachbarte Teile zu beschädigen). Dieser Widerstand behindert sonst die Kommunikation. Bedenken Sie aber, dass Sie die Gewährleistung durch diese Anpassung verlieren werden.

Zusätzlich empfiehlt Microchip für das ältere Modell, einen Widerstand im Bereich von 1 kΩ bis 10 kΩ zwischen TVDD und Pin 4 der Buchsenleiste einzubauen, um die UPDI-Kommunikation zu verbessern. Diesen können Sie aber auch in der Zielschaltung einsetzen, wie es im nächsten Abschnitt gezeigt wird.

Wer mehr über den technischen Hintergrund erfahren möchte, findet einen Link zu der Engineering Technical Note 36 von Microchip in der Kurzinfo.

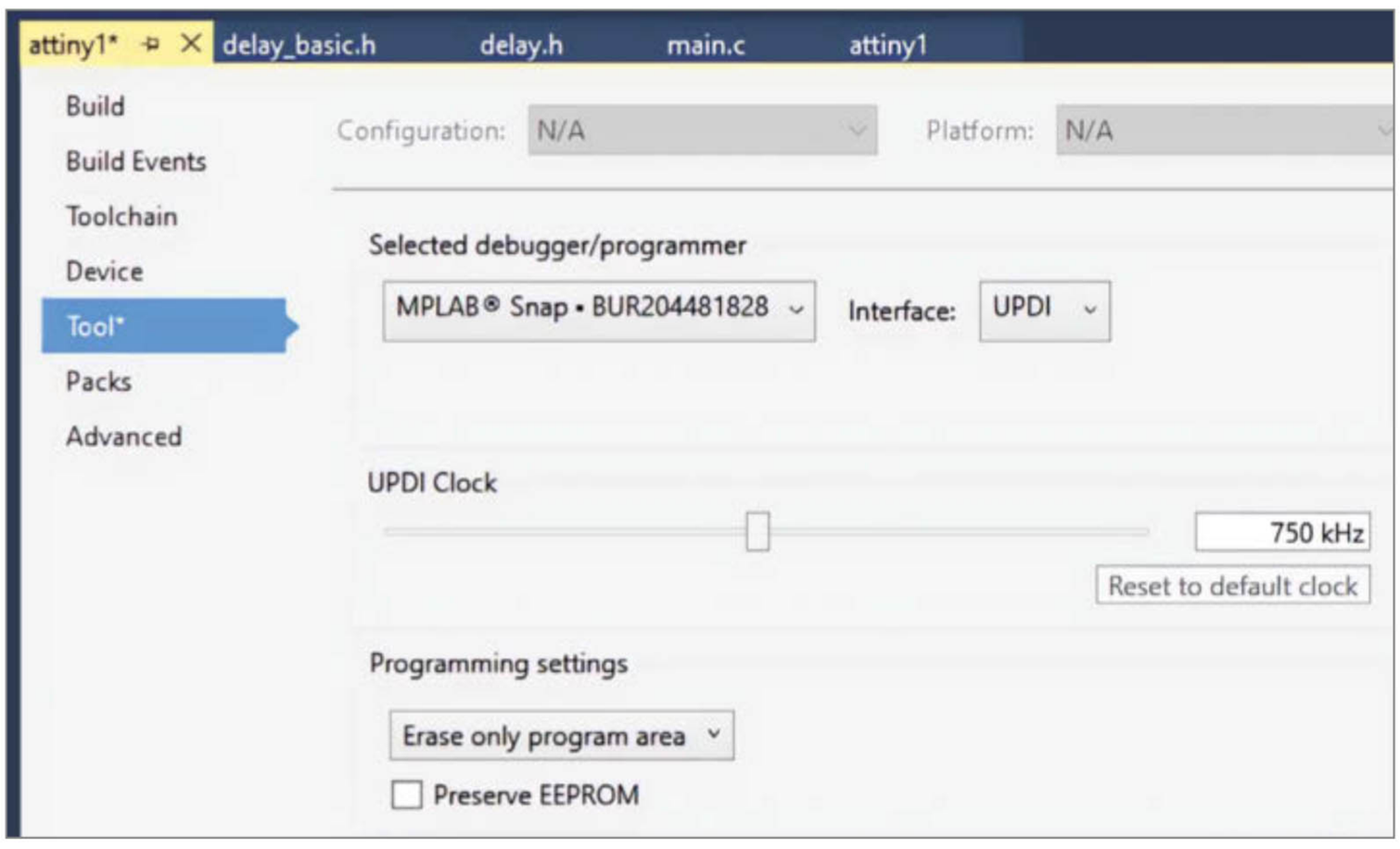


Bild 9: Wählen Sie den MPLAB Snap als Debugger aus und stellen Sie die „Programming settings“ wie gezeigt ein.

Erster Testaufbau

Nachdem Sie Ihren Snap-Debugger mit der neuen Firmware geflasht und auf AVR-Programmierung eingestellt haben, können Sie als Nächstes das Beispielprojekt auf einem Breadboard aufbauen. Wenn Sie den Artikel „ATtiny statt Arduino“ gelesen haben (siehe Link in der Kurzinfo), kennen Sie bereits die neuen ATtiny-Typen und wissen, wie Sie SMD-ICs auf einem Breadboard nutzen können. Für das erste Debuggen habe ich den dort verwendeten Aufbau auf das absolute Minimum reduziert, sodass der Controller mit der internen Taktquelle läuft und zwei LEDs ansteuern kann – der Stützkondensator bleibt weiterhin empfohlen (Bild 8).

Zwischen Reset/UPDI-Data und VCC sitzt der Widerstand mit 10 kΩ, den Microchip für eine stabilere Verbindung mit älteren, modifizierten Snaps empfiehlt. Man kann ihn nach dem Debuggen entfernen. Das neuere Modell des Snap benötigt den Widerstand nicht.

Der Snap kann die Zielschaltung nicht mit Strom versorgen. Mit dem TVDD-Pin misst er lediglich die Spannung der Zielschaltung – auch wenn es Debugger gibt, die über denselben Pin auch Strom liefern können.

Sie müssen den Debugger also per USB mit dem Computer verbinden und die Schaltung auf dem Breadboard mit einer separaten 5-V-Spannungsquelle versorgen. Wenn Sie kein Netzteil benutzen wollen, können Sie auch einen Arduino Uno an USB anschließen und von dort 5 V und GND verbinden. Benutzen Sie aber nur eine der beiden Spannungsquellen auf einmal!

Microchip Studio vorbereiten

Das erste Programm, das wir unter die Lupe nehmen, soll die rote LED mehrmals blinken lassen und dann dauerhaft ausschalten. Erstellen Sie dazu in Microchip Studio ein neues Projekt vom Typ „GCC C Executable Project“ für den ATtiny212 oder öffnen Sie die Datei attiny1.atsln aus dem GitHub-Repository des Projekts.

Wenn Sie den Code anschauen (siehe auch Listing „attiny1 main.c“), werden Sie erkennen, dass er dem aus der Make 2/24 ähnelt: Es wird die interne Taktquelle des Mikrocontrollers benutzt. Ein Pin (PA2) wird als Ausgang gesetzt und dann in der Schleife ein- und ausgeschaltet. Ist die Schleife durchgelaufen, wird das Programm angehalten, und weil die letzte Anweisung den Ausgang auf Low geschaltet hat, sollte die LED aus bleiben. Wie Sie gleich feststellen werden, wurde der Infinitiv bei der Formulierung mit Absicht benutzt.

Öffnen Sie die projektbezogenen Einstellungen über das Menü „Project/…Properties“ und wechseln Sie auf die Registerkarte „Tool“ (Bild 9) oder klicken Sie auf das Symbol mit dem Hammer in der Symbolleiste. Stellen Sie dort

über das Dropdown-Menü den Snap als Debugger ein. Weil der ATtiny nur UPDI beherrscht, ist keine Änderung bei Interface möglich. Wählen Sie danach bei „Programming settings" den Eintrag „Erase only program area" aus und deaktivieren Sie „Preserve EEPROM".

Wechseln Sie auf die Registerkarte „Toolchain" und stellen Sie bei „Configuration" den Eintrag „Debug" aus. Wählen Sie im Baum mit den Einstellmöglichkeiten den Eintrag „AVR/GNU C Compiler/Optimization" aus. Achten Sie darauf, wirklich im Bereich mit den Compiler-Einstellungen zu sein und nicht beim Linker. Ändern Sie den Wert für „Optimization Level" auf „None". Ansonsten können diese Optimierungen das Debugging behindern, weil sie etwa den Zugriff auf Variablen einschränken. Wenn Sie später die finale Version des Programms auf den Mikrocontroller flashen wollen, können Sie eine der Optimierungen davor wieder aktivieren.

Compiler-Konfigurationen für jeden Zweck

Microchip Studio bietet die Möglichkeit, Compiler-Einstellungen in verschiedenen Konfigurationen zu speichern. So lassen sich beispielsweise in einer „Release"-Konfiguration alle Einstellungen vorgeben, die für die Erzeugung des Produktivsystems benutzt werden sollen.

In der „Debug"-Konfiguration können Sie dann andere Parameter festlegen und je nach Bedarf auch noch weitere Konfigurationen erstellen. Das erspart Ihnen, die zahlreichen Einstellungen jedes Mal einzeln umstellen zu müssen. Stattdessen lässt sich mit einem Mausklick in die Auswahlliste in der Symbolleiste bequem zwischen den gewünschten Konfigurationen wechseln. Sie müssen aber bei der Änderung von Parametern darauf achten, welche Konfiguration Sie gerade bearbeiten.

Auf die Plätze ...

Es hat zwar etwas gedauert, aber jetzt geht es richtig los: Kompilieren Sie das Programm und übertragen Sie den erzeugten Maschinencode zunächst mit „Debug/Start Without Debugging" oder über den grünen Pfeil ohne Füllung (in der Symbolleiste) auf den ATtiny.

Die rote LED blinkt jetzt im schnellen Rhythmus, schaltet sich aber nicht aus, wie sie sollte – in diesem Fall ist das natürlich genau richtig, weil der Fehler gesucht werden soll. Wie immer gibt es dafür viele Möglichkeiten und Ansätze.

Eine einfache Debugging-Technik besteht darin, Breakpoints (Haltemarken) zu nutzen: Wie bereits eingangs erwähnt, legt man mit ihnen im Code fest, bis wohin das Programm laufen bzw. wo es anhalten soll, damit man es untersuchen kann.

Vermutlich haben Sie in einem Code-Editor selbst schon mal unbeabsichtigt Haltemarken angelegt, als Sie versucht haben, eine Zeile zu markieren. Je nachdem, wo man am linken Rand neben den Code klickt, reagieren die Editoren unterschiedlich, und hier werden in der Regel auch die Breakpoints gesetzt und gelöscht.

In Microchip Studio geschieht das ganz links am grau-blauen Rand, wenn der Mauszeiger sich in einen nach rechts oben zeigenden Pfeil verändert. Mit dem ersten Klick wird der Haltepunkt für eine Zeile gesetzt, was durch einen roten Punkt und eine rot-braun unterlegte Codezeile markiert wird. Mit dem nächsten Klick entfernen Sie den Haltepunkt wieder.

Setzen Sie die drei Haltepunkte, die Sie in Bild 10 sehen. Je nach Debugger und Mikrocontroller lassen sich Breakpoints nur vor dem Flashen ändern. Der Snap, zusammen mit dem ATtiny212, kann die Haltepunkte aber jederzeit auch während des Debuggens ändern. Übertragen Sie danach das Programm im Debug-Modus, indem Sie das Symbol mit dem gefüllten grünen Pfeil oder in der Menüleiste „Debug/Continue" auswählen.

Das Programm wird neu kompiliert, übertragen und in der IDE erscheint der Programmablauf. Eventuell öffnen sich einige Fenster als neue Tabs, aber die können Sie ignorieren und immer wieder auf main.c zurückkehren. Die Anweisungen werden bis zum ersten Breakpoint ausgeführt, und **vor** diesem bleibt der Programmzeiger stehen. Die auf den Breakpoint folgende Anweisung wird gelb markiert (Bild 11).

Während das Programm angehalten ist, können Sie den aktuellen Wert jeder Variable abfragen, indem Sie mit dem Mauszeiger über den Variablennamen fahren. Möchten Sie

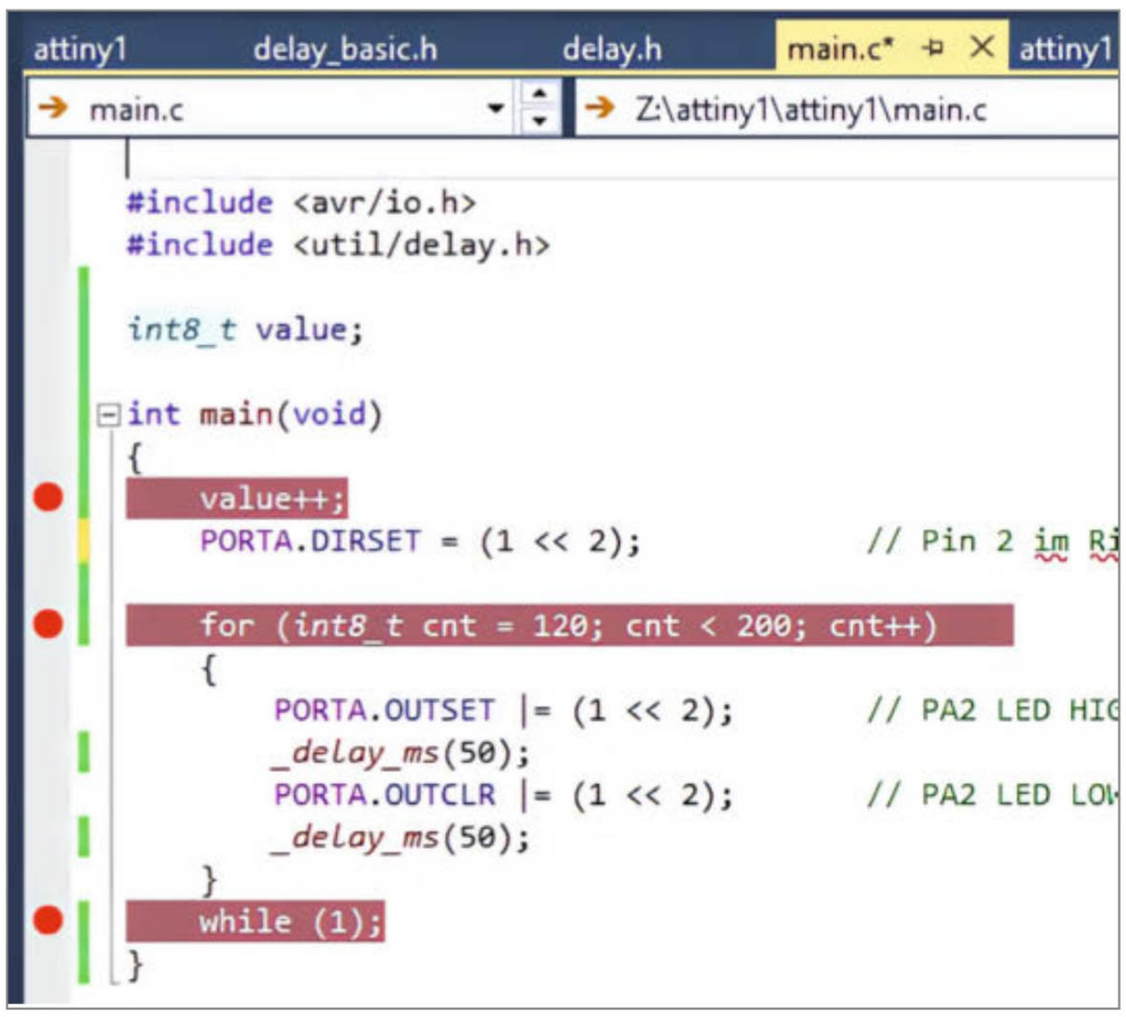


Bild 10: Es sind drei Breakpoints gesetzt, die das Programm pausieren, sobald man den Debugging-Vorgang startet.

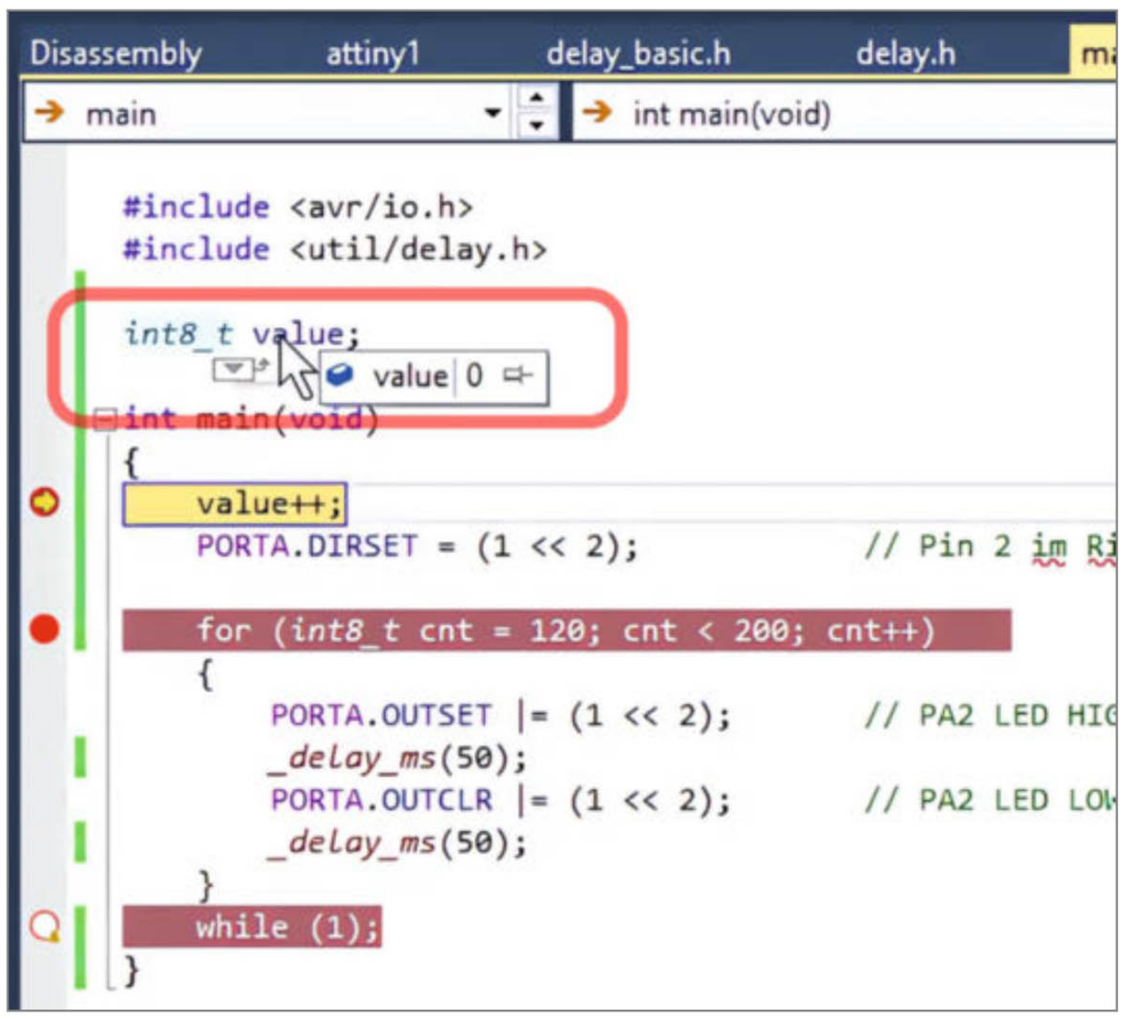


Bild 11: Das Programm hat am Breakpoint (vor value++) angehalten. Mit der Maus kann man nun die Variablen untersuchen (rot markiert).

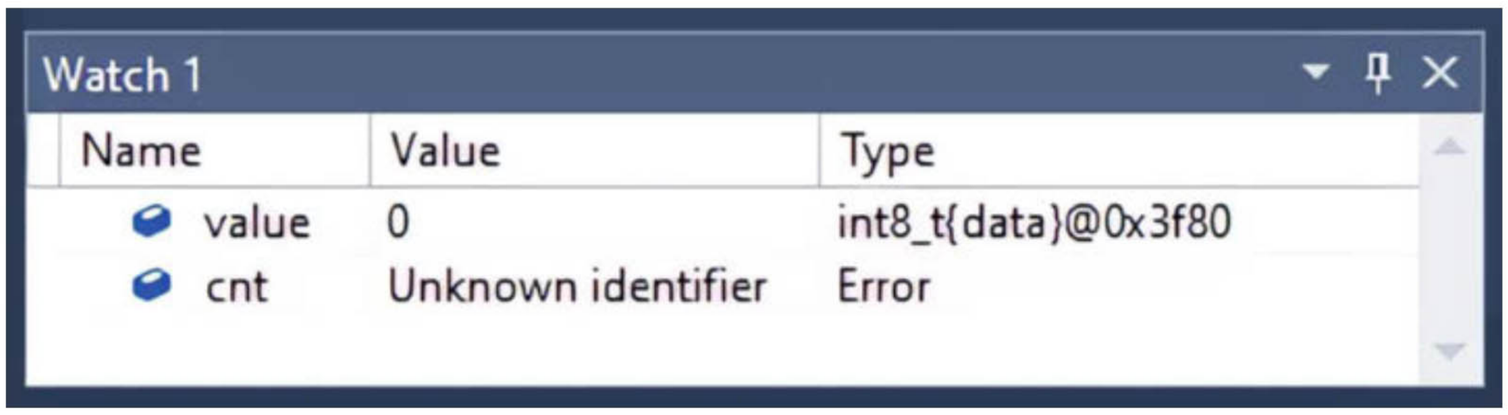


Bild 12: Im Watch-Fenster sehen Sie den Inhalt der beobachteten Variable.

etwa die Variable `value` die ganze Zeit im Auge behalten, klicken Sie mit der rechten Maustaste auf den Namen und wählen Sie im Kontextmenü den Eintrag „Add Watch“ aus. Am unteren Fensterrand dockt automatisch ein Fenster an, in dem von nun an immer der aktuelle Variablenwert zu sehen ist (Bild 12). Wird das Fenster nicht angezeigt, rufen Sie „Debug/Windows/Watch/Watch1“ auf. Fügen Sie auch eine Watch für die Variable `cnt` hinzu.

Die Variable `value` wurde im Code deklariert und ist mit `0` initialisiert worden. Weil die Variable `cnt` an der aktuellen Position des Programmablaufs noch gar nicht deklariert wurde (nur die IDE kennt sie derzeit), gibt es dort noch einen Fehler, der im Moment unwichtig ist. Aber es gibt eine andere typische Fehlerquelle, die Sie gleich mithilfe des Debuggers finden werden.

Variablen unter der Lupe

Klicken Sie auf das Symbol mit dem grünen Pfeil in der Symbolleiste oder F5 auf der Tastatur, um das Programm weiter auszuführen. Der nächste Breakpoint liegt bei der `for`-Schleife – genau genommen dort, wo `int8_t cnt` deklariert wird. Werfen Sie aber erst einmal einen Blick in die Watch-Liste (Bild 13): Der Wert für `value` hat sich geändert und steht nun bei `1`. Der Mikrocontroller hat also alle Befehle ausgeführt, die nach dem vorherigen Haltepunkt bis zu dem jetzigen kamen. Dies war unter anderem `value++`, was den Wert in der Variable um eins erhöht hat: von `0` zu `1`. Zudem sehen Sie, dass `cnt` jetzt auch deklariert wurde und in ihr ein Wert gespeichert ist – welcher das bei Ihnen ist, hängt mehr oder weniger vom Zufall ab und wird gleich erläutert.

Halten Sie die Programmausführung mit „Debug/Stop Debugging“ an (oder mit dem roten Quadrat in der Symbolleiste). Das ist notwendig, weil der Quellcode geändert werden muss, was nicht im laufenden Debug-Modus geht. Die IDE würde Sie später aber ohnehin darauf hinweisen. Verschieben Sie die Zeile mit der Deklaration für die Variable `value` in die `main()`-Routine, wie es in Bild 14 zu sehen ist.

Starten Sie wieder das Debugging, wodurch der Code neu kompiliert und auf den Mikrocontroller übertragen wird. Wieder läuft das Programm bis zum ersten Breakpoint und hält an. Schauen Sie sich die Ausgaben im Watch-Fenster an. Fällt Ihnen etwas auf? In der Variable `value` steht jetzt ein zufälliger Wert, der sich auch jedes Mal ändern wird, wenn Sie das Debugging anhalten und neu starten.

In der ersten Programmversion handelte es sich um eine globale Variable, weil sie außerhalb jeder Funktion deklariert wurde. Solche Variablen (und `statics`) werden von AVR-GCC in Anlehnung an den C-Standard mit dem Wert `0` initialisiert. Das ist aber nicht der Fall, wenn eine Variable innerhalb einer Funktion deklariert wird. Dann ist der Startinhalt beliebig.

Wenn Sie jetzt weiterrechnen, also den Code bis zum nächsten Breakpoint ausführen, wird auch mit diesem Startwert weitergerechnet und der Wert um eins erhöht.

So eine fehlende Initialisierung kann ganz böse nach hinten losgehen, wenn Sie einfach davon ausgehen, dass deklarierte Variablen immer mit `0` starten. Als Erkenntnis aus dem Debugging können Sie also schon einmal mitnehmen, dass es ratsam ist, jede Variable ohne Ausnahme immer gleich bei Deklaration auch mit einem Startwert – und sei es `0` – zu initialisieren.

Genau das ist auch der Grund, warum beim zweiten Breakpoint `cnt` mit einem zufälligen Wert initialisiert wird: Der Debugger hat angehalten, nachdem die Variable deklariert wurde, aber noch nicht initialisiert ist. Erst wenn das Programm fortgesetzt wird, kommt das `= 120`. So etwas

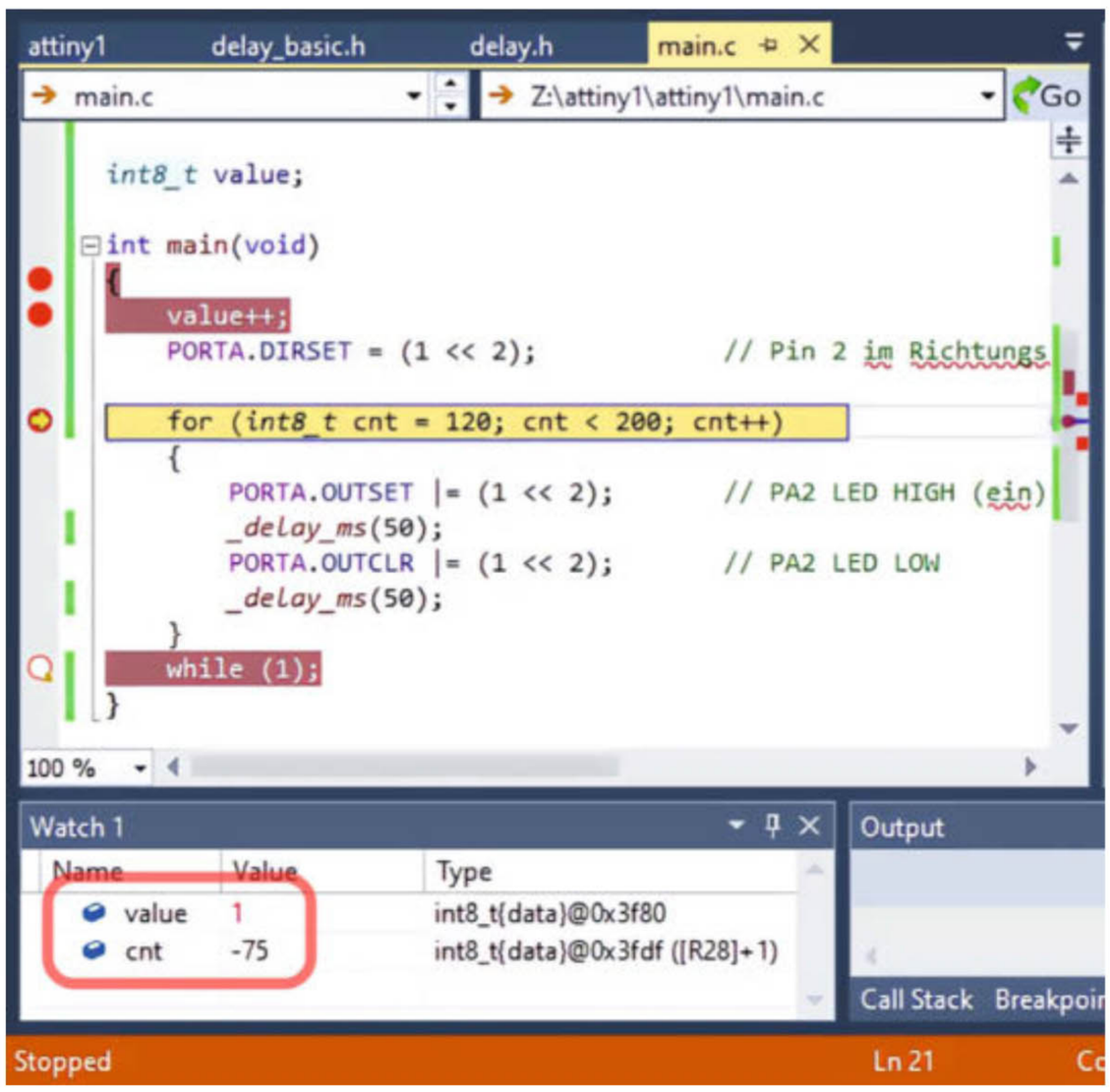


Bild 13: Die Variable `value` wurde inkrementiert und `cnt` deklariert, wie im Watch-Fenster links unten zu sehen

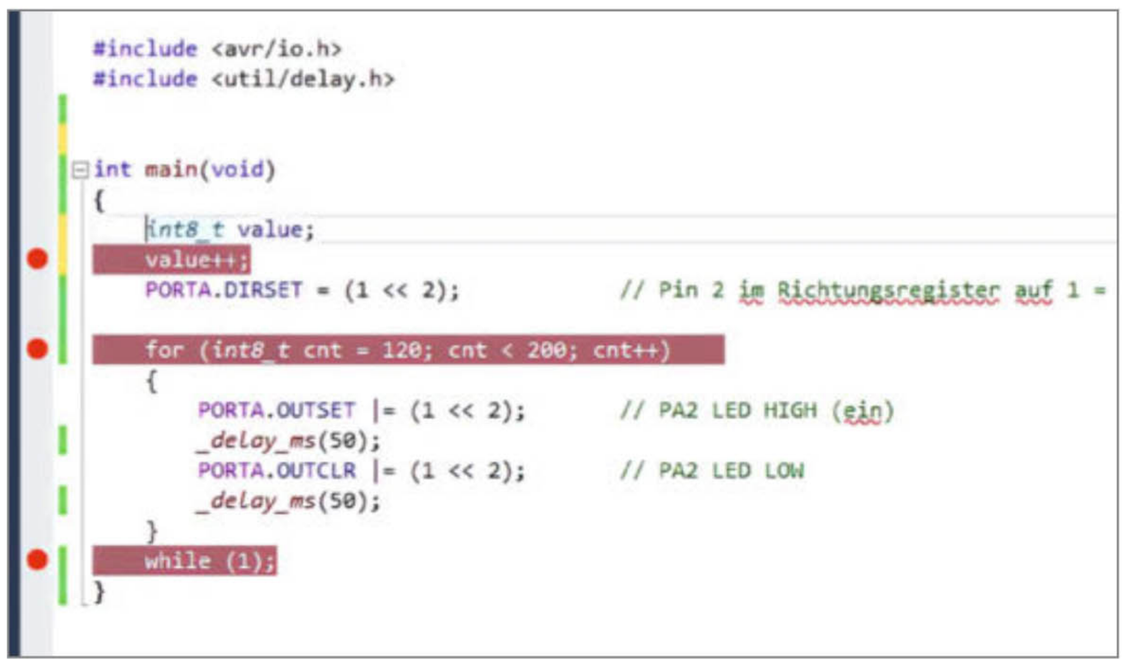


Bild 14: Eine kleine Änderung im Code und die Variable ist nun nicht mehr global gültig.

können Sie mit keiner anderen Debugging-Lösung erkennen.

Übergelaufene Variablen

Die Variable `value` wird nicht weiter benötigt – sie diente nur zur Veranschaulichung der bisherigen Ausführungen. Klicken Sie einmal auf den grünen Pfeil (F5), aber wirklich nur einmal. Scheinbar hat sich nichts getan, aber bei genauer Betrachtung sehen Sie, dass jetzt auch die Schleifenvariable `cnt` mit `120` initialisiert wurde. Sofern sie kleiner als `200` ist, sollte sie bei jedem Klick auf den Pfeil um `1` hochzählen, die LED kurz (50 ms) leuchten lassen und dann wieder ausschalten. Klicken Sie noch maximal fünfmal auf den Pfeil, bis `cnt` den Wert `125` hat.

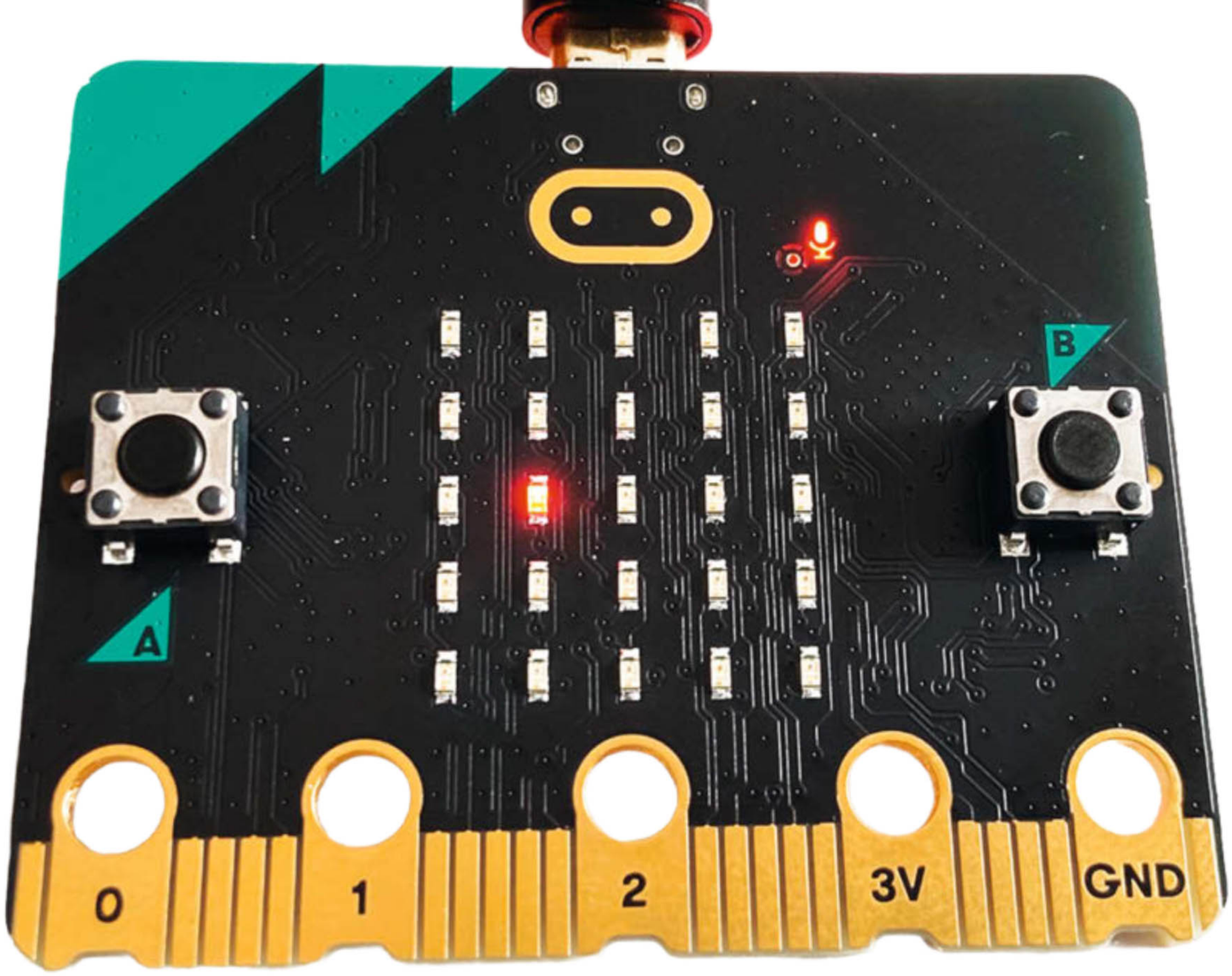
Denken Sie daran, dass wir uns immer noch auf der Suche nach dem Grund befinden, warum die LED endlos weiterblinkt. Also lassen Sie die Schleife jetzt noch ein paar Mal durchlaufen: `cnt` wird zu `126`, `127` und `-128` (!), und nicht etwa zu plus `128`. Dann geht's weiter: `-127`, `-126`, und irgendwann wieder mit `0`, `1` usw. Mit dieser Zahlenfolge bleibt `cnt` immer kleiner als `200` und die Schleife kann kein Ende finden.

Aber wieso ist das so? Das kann Ihnen der Debugger leider nicht beantworten, sondern hier muss jetzt Ihr Programmierwissen helfen. Schuld an der Misere ist der Variablentyp `int8_t`. Wie Sie vielleicht wissen, ist diese Schreibweise sehr beliebt, weil man erkennen kann, wie groß die Variable ist – also wie viele Bits sie belegt. Stutzig könnte Sie jetzt machen, dass in 8 Bit doch `255` unterschiedliche Werte gespeichert werden können. Also sollte es kein Problem sein, bis `200` zu zählen. Der Typ `int8_t` ist aber vorzeichenbehaftet, d. h. er kann in einem Bereich von `-128` bis `127` Werte speichern. Genau, wie Sie es mit dem Debugger sehen konnten, findet nach `127` ein Überlauf statt. Ein kleines „u" vor der Typangabe macht aus der Variable einen unsigned (vorzeichenlosen) Typen für ausschließlich positive Zahlen (`0` bis `255`).

Wir haben den Fehler gefunden! Halten Sie das Programm an, ändern Sie die Deklaration in `uint8_t cnt = 120` (das „u" ergänzen) und starten Sie das Programm neu. Damit Sie nicht 80-mal die Ausführung fortsetzen müssen, entfernen Sie zuvor aber noch alle Breakpoints bis auf den letzten bei `while()`. Jetzt macht das Programm genau das, was es soll: Die LED blinkt eine Weile und bleibt dann dauerhaft ausgeschaltet.

Es geht noch weiter

Sie haben erfolgreich Ihr erstes Programm auf einem ATtiny debuggt und ein paar Werkzeuge kennengelernt, die dabei helfen. Wenn Sie bis hierher Spaß hatten, schauen Sie sich doch noch den Online-Artikel mit einem weiteren Beispiel an. Er verwendet den hier gezeigten Aufbau, geht aber bei der Fehlersuche auf die Register ein und erklärt, wie man diese überwachen und manipulieren kann (siehe Link in der Kurzinfo). In der nächsten Make-Ausgabe debuggen wir dann Interrupts auf einem Arduino Uno mithilfe von debugWIRE. *—akf*



Den BBC Micro:bit in Python programmieren

Der BBC Micro:bit bietet zahlreiche Funktionen für den Einstieg in die Mikrocontroller-Welt: Sensoren für Bewegung oder Lautstärke, Taster, 25 LEDs zur Anzeige, Soundgenerator, Steuer-Ein-/-Ausgänge und vieles mehr. Um sich damit auseinanderzusetzen, ist ein Spiel ein motivierender Einstieg. Wer eine Echtzeitanwendung wie ein Spiel in MicroPython erstellen kann, kann Programme mit Steueraufgaben verfassen, bei denen das richtige Timing eine wesentliche Rolle spielt.

von Gerhard Völkl

„Die Zerstörung der Erde durch einen Meteoritenschauer steht kurz bevor. Nur du kannst es mit deinem Raumschiff verhindern." So oder so ähnlich beginnen die Beschreibungen vieler klassischer Videospiele. In den Anfängen der Computer waren nur einige helle Punkte als Raumschiff oder Meteoriten zu sehen. Für ein ähnliches Spiel ist der BBC Micro:bit bereits super ausgestattet. Er hat einen ARM Cortex-M4 Mikrocontroller mit 64 MHz und 32 Bit. Der Speicher besitzt einen beschreibbaren Bereich von 512 KB und 128 KB statisches RAM. Im folgenden Artikel wird die Erstellung eines solchen Raumschiffspiels begleitet.

Auf der Platine sind 25 LEDs vorhanden, die von einem Programm als Display genutzt werden können. Zwei Taster eignen sich zur Steuerung. Interessant sind die bereits auf der Platine vorhandenen Beschleunigungs- und Lautstärkesensoren. Sie können im Spiel zur Steuerung der Bewegung des Raumschiffs verwendet werden. Mit einem Soundgenerator und einem kleinen Lautsprecher steht auch einer musikalischen Untermalung nichts mehr im Wege. Die komplette Hardware für ein einfaches Spiel ist also vorhanden. Mit MicroPython als Programmiersprache ist alles für das neue Weltraumspiel parat.

Dieser Artikel zeigt das Einrichten der Micro:bit-Entwicklungsumgebung, wie man Codeblöcke erstellt und erklärt, was bestimmte Befehle im Code genau machen. Das komplette Programm findet man auf der GitHub-Seite in der Online-Info.

Entwicklungsumgebungen

Wer den Micro:bit per Micro-USB an seinen Rechner angesteckt hat, steht vor der Entscheidung: Welche Software verwende ich, um mein Programm in MicroPython zu entwickeln? Am schnellsten geht es mit der Webanwendung python.microbit.org. Den Link findet man noch einmal in der Online-Info.

Spielebasis

Das hier entwickelte Spiel wird aus folgenden drei Abschnitten bestehen:

- START: Titel und Musik im Startbildschirm stimmen auf das Spiel ein.
- SPIELE-SCHLEIFE: In einer Endlosschleife holt sich das Spiel die Eingabe, reagiert darauf und ändert dementsprechend die Anzeige.
- ENDE: Das Spiel teilt mit, ob man gewonnen oder verloren hat.

In MicroPython lassen sich diese drei Abschnitte und ihr Wechsel über eine Variable `game_state` abbilden. Diese hat den Wert 0 im Startabschnitt, bei der Spiele-Schleife den

Wert 1 und beim Ende den Wert 2. Damit alles etwas leserlicher wird, gibt es die entsprechenden Konstanten (`STATE_START = 0`, `STATE_RUN = 1` und `STATE_END = 2`), wie im Listing der Spieleabschnitte auf Seite 115 gezeigt.

Die Hauptschleife mit `while True` läuft endlos durch. Je nach aktuellem Status springt das Programm zu der Funktion, die in diesem Abschnitt auszuführen ist. Der Code ab `if game_state == STATE_START:` fragt ab, in welchem Spielstadium sich der aktuelle Spieldurchlauf gerade befindet. Ist es `STATE_START`, wird das Spiel gestartet und der Spielstatus auf `STATE_RUN` gesetzt (ab `if game_state == STATE_START:`). Falls der Status `STATE_RUN` ist, läuft der Game-Loop des Spiels (ab `elif game_state == STATE_RUN:`). Im Status `STATE_ENDE` wird das Spiel beendet (ab `elif game_state == STATE_END:`)

Musik zum Start

Am Anfang ist die Themenmelodie des Spiels in einer Endlosschleife zu hören und eine einfache Raumschiffanimation zu sehen, bis ein Tastendruck erfolgt und das eigentliche Geschehen startet. Zur Ausgabe von Musik bringt MicroPython das Modul `music` mit. Dieses bindet man über

```
import music
```

in das Spiel ein.

Die Funktion `play` startet ein Musikstück. Beispielsweise das vorgefertigte `music.FUNK` über die Codezeile `music.play(music.FUNK)`. MicroPython enthält einige dieser fertigen Melodien wie `music.ENTERTAINER`, `music.PRELUDE` oder `music.ODE`.

Im Weltraumspiel soll jedoch eine eigene Melodie zu hören sein, deren einzelne Töne in einer Liste definiert sind.

```
THEME_MELODY = ['C4:4','G4:4']
music.play(THEME_MELODY)
```

Die Melodie in der Liste `THEME_MELODY` besteht aus zwei Tönen, die `play` abspielt. Die Zeichenkette `C4:4` bezieht sich auf den Ton `C` und seine Oktave, in diesem Fall `4`. Damit ist die Tonhöhe festgelegt. Die Zahl nach dem Doppelpunkt definiert die Dauer des Tons. Je höher die Zahl ist, desto länger ist der Ton zu hören.

Normalerweise wartet das Programm, bis Micro:bit die Melodie abgespielt hat und geht dann weiter zum nächsten Befehl über. In unserem Spiel soll die Melodie jedoch in einer Endlosschleife im Hintergrund laufen.

```
music.play(THEME_MELODY, wait=False,
loop=True)
```

Dies erreicht man über die zusätzlichen Parameter `wait` und `loop`. Wenn der Parameter `wait` den Wert `False` hat, wartet das Programm nicht, sondern arbeitet den nächsten Befehl ab. Die Musik wird weiter abgespielt. Der zweite Parameter `loop` legt fest, dass die Melodie in einer Schleife abgespielt wird. Wenn die Musik im Hintergrund nicht mehr erwünscht ist, kann sie mit dem Befehl `music.stop()` ausgeschaltet werden.

Aktuell spielt das Spiel nach dem Start die selbst geschriebene Musik ab. Der Code ist wie im Bild für die Funktion `start_game` dargestellt.

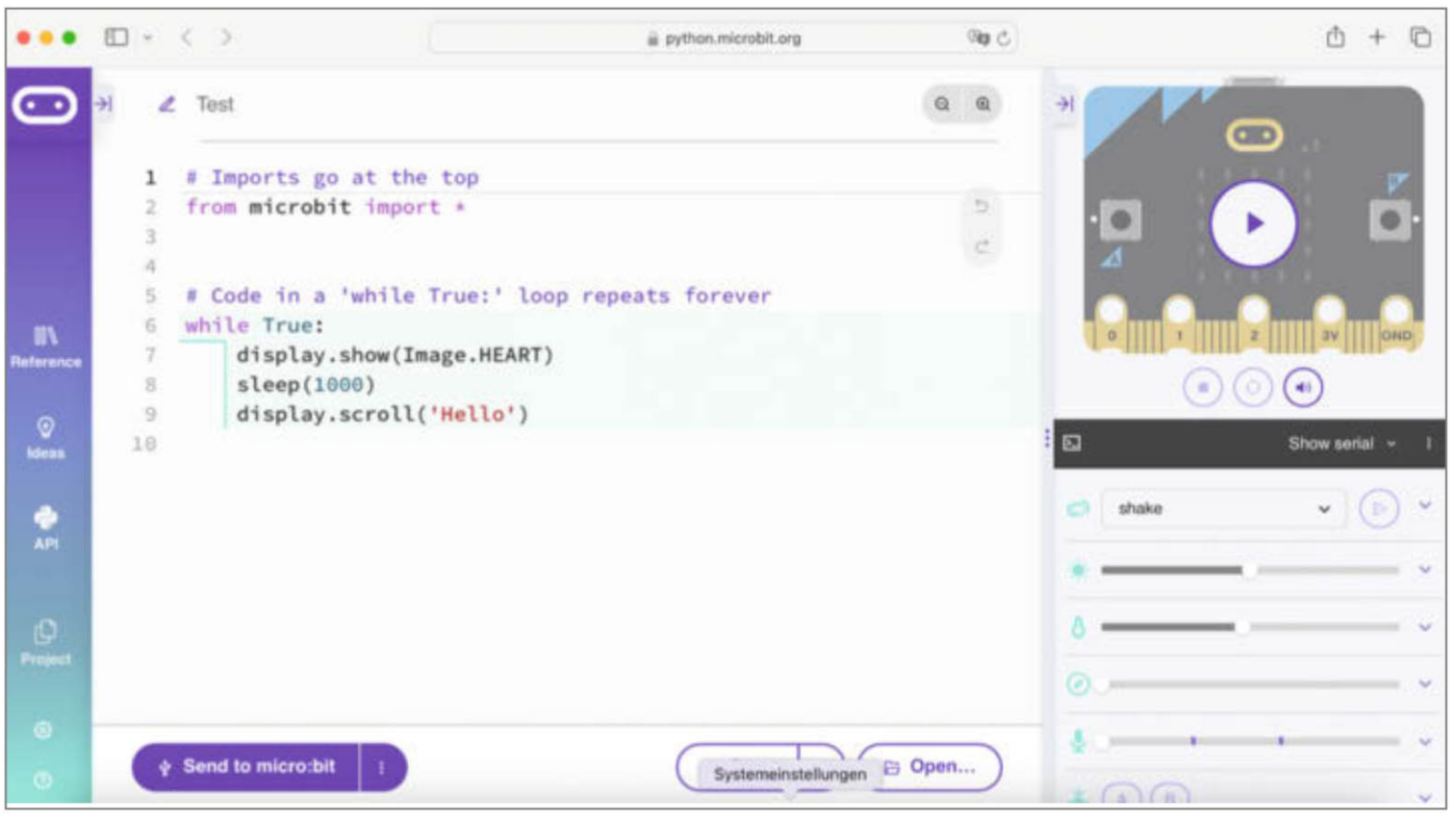
Bilder und Animation

Für die Ausgabe von Bildern verfügt der Micro:bit über 25 LEDs, die in einem Rechteck von fünf mal fünf LEDs angeordnet sind. Für jede dieser LEDs kann die Helligkeit separat festgelegt werden. Dies geschieht in 10 Stufen (von 0 bis 9). Je höher der gesetzte Wert, desto heller leuchtet die LED.

Mit diesen Einstellungen lassen sich kleine Bilder erstellen.

```
ship = Image("00000:"
             "00000:"
             "00900:"
             "09490:"
             "94449")
display.show(ship)
```

Jede Zeile innerhalb der Klammer steht für eine Reihe an LEDs auf dem Micro:bit. Die eingegebene Zahl bestimmt die Helligkeit der LEDs. In diesem Fall ist es ein einfaches dreieckiges Raumschiff. Wo eine 0 steht, ist nichts zu sehen. Am Rand des Raumschiffs stehen Neunen, damit es dort möglichst hell leuchtet.



Wer noch keinen Micro:bit hat, kann trotzdem mit dem simulierten Gerät online loslegen.

Kurzinfo

- **Einstieg in MicroPython**
- **Einrichten von VS-Code für Arbeit mit dem Micro:bit**
- **Nutzung verschiedener Sensoren**

Checkliste

Zeitaufwand:
15 Minuten

Kosten:
30 Euro

Material

- **BBC Micro:bit**

Mehr zum Thema

- Peter König, Ausprobiert: Wappsto:bit – Erweiterungsplatine für den BBC micro:bit
- Ákos Fodor, Calliope mini 3 ausprobiert, Make 2/24. S. 46
- Ákos Fodor, MicroPython-Boards, Make 7/23, S. 12

Alles zum Artikel im Web unter *make-magazin.de/x7nn*

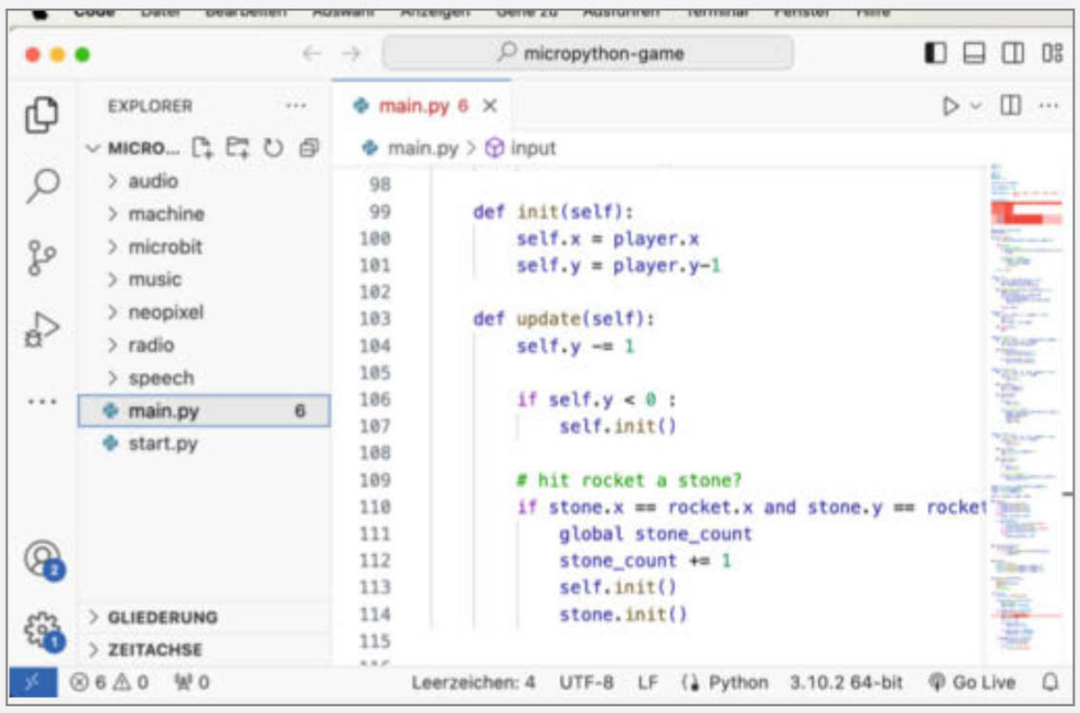
Entwickeln mit einer universellen IDE

Will man öfter Software entwickeln oder Programme für unterschiedliche Hardware erstellen, dann sollte man die kostenlose Entwicklungsumgebung Visual Studio Code von Microsoft auf dem eigenen Rechner installieren, die es für Microsoft Windows, macOS und Linux gibt. Diese Entwicklungsumgebung ist vielseitiger und lässt sich mit verschiedenen Plug-ins für unterschiedliche Programmiersprachen nutzen. In diesem Fall fügt die Erweiterung „micro:bit Python“ von MAKinteract die Kompatibilität für den Micro:bit hinzu.

Darüber hinaus gibt es Befehle, um mit dem Micro:bit zu kommunizieren. Die Tastenkombination STRG + Umschalt + P (auf dem Mac CMD + Shift + P) öffnet ein Eingabefenster, in das man `micro:bit-python` eintippt. Danach erscheint eine Liste von Befehlen. Der Befehl `micro:bit-python: Flash MicroPython environment on micro:bit` muss zuerst ausgeführt werden. Dieser überträgt die MicroPython-Umgebung auf den Micro:bit.

Jetzt sind sowohl Visual Studio Code als auch der Micro:bit bereit für die Python-Programmierung. Um in Visual Code ein erstes kleines Programm zu erzeugen, das man als Startpunkt verwenden kann, gibt man den Befehl `micro:bit-python: Initialize the Workspace` ein. Dabei fragt die Erweiterung ab, mit welchen Sensoren und Modulen man arbeiten möchte und bindet diese gleich ein. Die anschließend generierte Programmdatei, eine einfache Textausgabe, hat den Namen `main.py`. Diese führt der Micro:bit automatisch aus, sobald man ihn startet.

Mit dem Befehl `micro:bit-python: Flash sketch on the micro:bit` kann man die MicroPython-Anwendung auf den Micro:bit übertragen. Falls das Programm einen Fehler enthält, ist ein Hinweis auf den 25 LEDs zu sehen. Normalerweise startet das neue Programm sofort. Falls nicht, muss man den Reset-Knopf auf der Rückseite der Platine drücken. Danach startet immer das letzte übertragene Programm.



Visual Studio Code ist nicht nur ein Texteditor, sondern ermöglicht auch, komplexe Projekte zu verwalten.

Das Innere des Raumschiffs ist dagegen nicht so hell, hier steht eine 4.

Die Funktion `show` stellt die LEDs auf die eingestellten Werte ein. Aber nicht nur Standbilder kann man darstellen, auch Animationen lassen sich über das Display jagen. Dafür benötigt man eine Liste von Bildern (`SHIP_IMAGES`), die das Programm nacheinander abspielt.

Konstanten

Konstanten sind in einem Programm feste Werte. Sie können von überall im Code abgerufen werden und sind unveränderlich.

Sie haben einen Namen und einen Wert. Wenn ich zum Beispiel mein Geburtsjahr in einem Programm verwenden will, könnte ich das als Konstante machen, da mein Geburtsjahr unveränderlich ist.

Das würde im oberen Teil eines Programms in Großschreibung mit `GEBURTSJAHR = 1997` passieren.

Diese legt man wie folgt fest:

```
SHIP_IMAGES = [
    Image("00000:"
          "00000:"
          "00900:"
          "09490:"
          "94449"),
    Image("00000:00900:09490:94449:09
490"),
    Image("00900:09490:94449:09490:00
900"),
    Image("09490:94449:09490:00900:00
000"),
    Image("94449:09490:00900:00000:00
000")

]
```

In der Liste `SHIP_IMAGES` sind jetzt fünf verschiedene Bilder gespeichert. Dieses Codebeispiel zeigt die beiden Möglichkeiten, die Bilder zu definieren. Entweder schreibt man die einzelnen Zeilen untereinander – wie im ersten Image – oder direkt nebeneinander.

Mit einer sogenannten For-Schleife kann man diese Bilderliste jetzt animieren.

```
for image in SHIP_IMAGES:
            display.show(image)
            sleep(100)
```

Die For-Schleife holt jeweils ein Bild aus der Liste `SHIP_IMAGES` in die Variable `image` und zeigt es an. Der Befehl `sleep(100)` sorgt dafür, dass das Programm 100 Mikrosekunden lang nichts macht, bevor es weitergeht. Diese Wartezeit verhindert, dass die Bilder zu schnell hintereinander angezeigt werden.

Das Bild Animationen auf Seite 116 zeigt unseren aktuellen Spielstart. Zuerst wird ein Musikstück definiert, dann eine Reihe von Raumschiffbildern. Danach wird festgelegt, was passieren soll, wenn der Befehl `start_game()` ausgeführt wird. Die Musik beginnt zu spielen und das Raumschiff wird animiert.

Innerhalb der Funktion `start_game` befindet sich die For-Schleife, die unsere Raumschiffgrafik animiert. Damit das kontinuierlich passiert, ist die For-Schleife in eine Endlosschleife (`while True`) eingebettet. Um diese Schleife wieder zu verlassen, bauen wir als Nächstes die Abfrage für einen Tastendruck ein.

```
if button_a.was_pressed() or
button_b.was_pressed():
    break
```

Das Programm überprüft über die Funktion `was_pressed`, ob eine der Tasten (`button_a` oder `button_b`) gedrückt ist. Falls ja, wird die Schleife mit `break` abgebrochen, was die kom-

plette Funktion `start_game` beendet. Dann geht es wieder eine Stufe nach oben zur Schleife, in der die Variable `game_status` verarbeitet wird (siehe Listing Spielabschnitte). Nachdem der Start beendet wurde, setzt das Programm diese Variable auf `STATE_RUN`, sodass das eigentliche Spiel beginnen kann.

Damit ist unser erster Spielstatus fertig. Der Code dafür sieht wie im Bild Spielstatus Start auf Seite 117 aus.

Game-Loop

Das eigentliche Spiel besteht aus drei Abschnitten, die das Programm ständig bis zum Spielende durchläuft:

- INPUT: Das Programm prüft, ob Tasten gedrückt sind oder ob sich an den Sensoren zur Steuerung des Spiels etwas geändert hat.
- UPDATE: Verändert entsprechend der Eingabe oder anderer Vorgaben die interne Spielwelt.
- PAINT: Stellt die interne Spielwelt im Display mit den LEDs dar.

Im Programm wird dies durch die Funktionen `game_loop`, den eigentlichen Game-Loop, und die Funktion `do_gameloop`, die sich um die Zeitverwaltung kümmert, abgebildet (siehe Listing Game-Loop).

Da fast alles im Spiel zeitabhängig ist – die Bewegungen, was wann passiert – braucht es eine einfache Zeitverwaltung. Dazu verwenden wir die im Micro:bit vorhandene Funktion `running_time`, welche die Zeit seit dem Programmstart zurückgibt, denn das Spiel muss wissen, wie viel Zeit seit dem letzten Durchlauf des Game-Loop vergangen ist.

Die Variable `now` enthält den aktuellen Wert von `running_time`, und `last_run` enthält immer den Wert des letzten Durchlaufs. Die Differenz ist die seitdem vergangene Zeit. Die Funktion `update` erhält die Differenz als Parameter und kann diese bei der Änderung der internen Spielwelt berücksichtigen.

Spielabschnitte

```
STATE_START = 0
STATE_RUN = 1
STATE_END = 2

game_state = STATE_START

while True:
    last_run = running_time()

    if game_state == STATE_START:
        start_game()
        game_state = STATE_RUN

    elif game_state == STATE_RUN:
        do_gameloop(last_run)

    elif game_state == STATE_END:
        end_game()
```

Update der internen Spielwelt

Das Raumschiffspiel besteht grundsätzlich aus 3 verschiedenen Akteuren: einem Raumschiff, einer Rakete und einem Meteoriten. Diese haben alle sehr ähnliche Eigenschaften. Alle haben eine Position, bewegen sich und reagieren auf Veränderungen in ihrer Umgebung. Aber sie unterscheiden sich natürlich in ihrem Verhalten. Das Raumschiff bewegt sich beispielsweise immer von links nach rechts und der Meteorit fällt immer von oben nach unten.

Um Sachen zu verallgemeinern und trotzdem die Möglichkeit zu haben, spezielle Dinge zu programmieren, eignen sich Klassen (class) in MicroPython. Klassen sind so etwas wie Bau-

Variablen

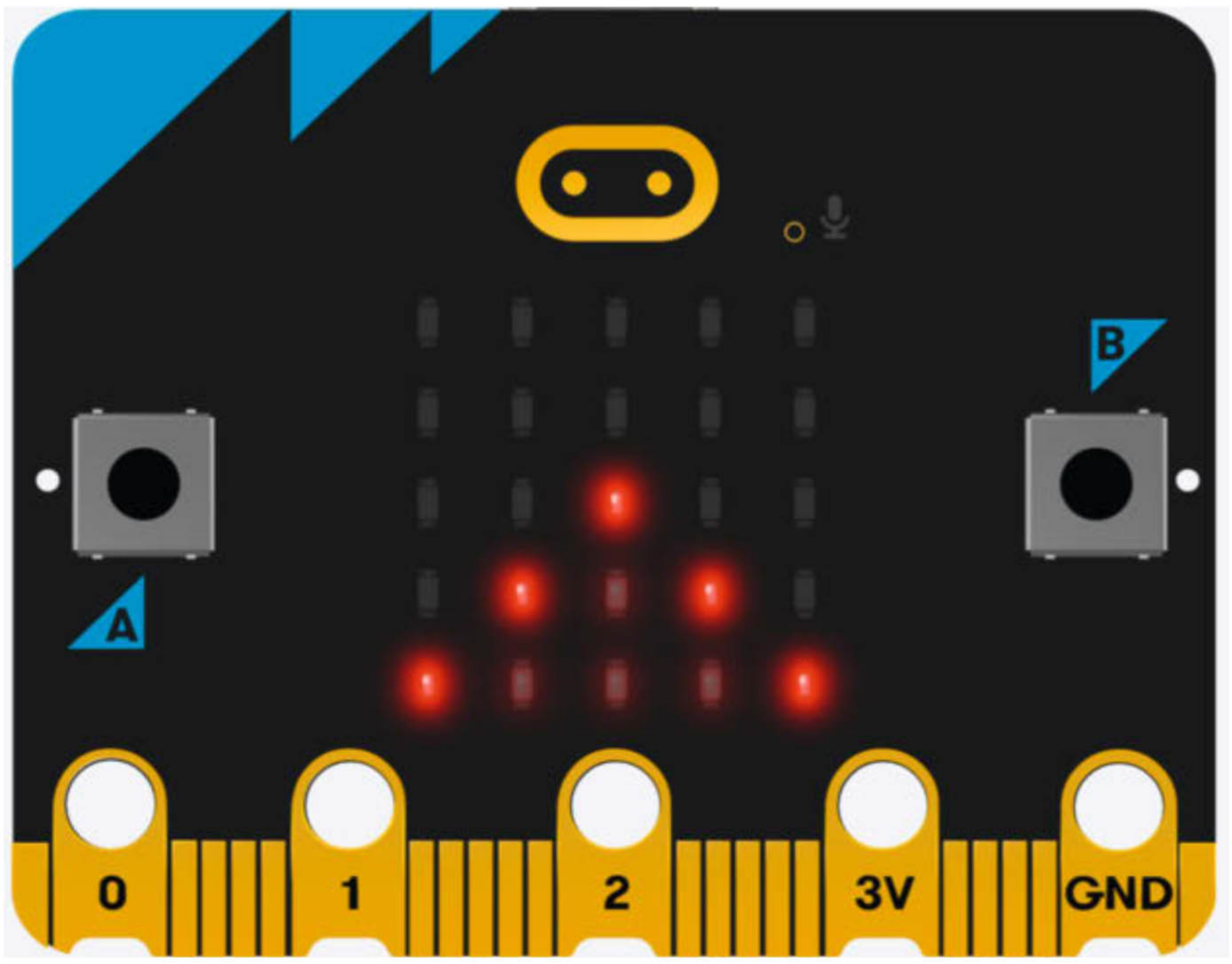
Variablen sind, wie der Name schon sagt, variable Werte. Sie haben ebenfalls einen Namen und einen Wert. Während der Laufzeit eines Programms können Variablen verändert werden. Das aktuelle Jahr ist zum Beispiel veränderlich. `aktuelles_jahr = 2024`

Funktionen

Funktionen sind eine Reihe von Anweisungen im Code, die eine immer wiederkehrende Aufgabe erledigen und unter einem Namen abrufbar sind. Bei der hier verwendeten Funktion `play`, gibt es einen Codeabschnitt, der den Namen `play` erhält und in dem festgelegt wird, wie genau der Sound abgespielt wird. Über diesen Namen `play` kann dann dieser Code an anderer Stelle einfach aufgerufen werden.

```
THEME_MELODY = [ 'G4:4', 'G4:4', 'G4:4', 'D4:2','A4:8', 'G4:2', 'C5:8', 'B4:4',
                 'A4:2']

def start_game():
    music.play(THEME_MELODY, wait=False,loop=True)
```

Funktion start_game: Der Code spielt eine Melodie ab.



Das animierte Raumschiff fliegt jetzt von unten nach oben über den Bildschirm.

pläne, aus denen im Verlauf der Laufzeit eines Programms spezifische Entitäten generiert werden können. Beispielsweise würde in einer Klasse für einen Stuhl festlegen, dass ein Stuhl eine unbestimmte Anzahl von Beinen hat, dass er eine Lehne haben kann und eine Farbe. Ein bestimmter Stuhl wird dann mit 4 Beinen, einer Lehne und der Farbe Rot erzeugt.

Die Klasse `Actor` steht für ein beliebiges Element, einen beliebigen Akteur, im Spiel. Sie besteht aus zwei Werten `x`, `y`, welche die aktuelle die Position auf dem Display beschreiben, sowie einem `Timer`, den man wie einen Wecker aufziehen kann. Dazu später mehr.

```
class Actor:
    def __init__(self, x, y,
speed=0):
        self.x = x
        self.y = y
        self.timer = Timer(speed)
```

Die Funktion `__init__` initialisiert ein neues Element der Klasse und legt dabei die Variablen der Klasse fest. Im Falle der Actor-Klasse von oben wird die X- und Y-Position eingestellt und die Geschwindigkeit festgelegt. Mit der Zeile

```
actor = Actor(0,4 ,speed=100)
```

erzeugen wir einen Actor mit dem Namen `actor`, der auf der X-Position 0 und der Y-Position 4 steht und eine Geschwindigkeit von `100` hat. Die Klasse `Actor` hat darüber hinaus eine Funktion `update`, die der Game-Loop immer aufruft, wenn sich etwas geändert hat. In dieser Funktion in der Klasse `Actor` passiert gar nichts, diese ist je nach Element zu ergänzen.

Zur Definition der Klasse für den Meteoriten (`Stone`) kommt die Klasse `Actor` zum Einsatz. Die Klasse `Stone` ist somit eine Erweiterung der Klasse `Actor`. Die Klasse Stone verfügt über sämtliche Eigenschaften der Klasse `Actor` und bastelt noch weitere Elemente dazu.

```
class Stone(Actor):
    def __init__(self, x=-1, y=-1,
speed=0):
        super().__init__(x, y, speed)
        if x == -1:
            self.init()
```

Die Initialisierungs-Funktion der Klasse `Stone` ruft mit `super().__init__` die Initialisierung der Klasse `Actor` auf. Wenn `x` den Standardwert `-1` hat und damit nicht vom Programm festgelegt wurde, setzt die Klasse in ihrer Funktion `init` den Wert selbst fest.

```
    def init(self):
        self.x = random.randint(0,4)
        self.y = 0
```

Die Funktion `randint` generiert in diesem Fall eine ganze Zufallszahl zwischen 0 und 4. Bei `y = 0` erscheint der Meteorit in der obersten Zeile der LEDs und mit dem Zufallswert für `x` in einer beliebigen Spalte. Die Funktion `Update` der Klasse `Stone` überschreibt die gleichnamige Funktion der Klasse `Actor`. Allerdings nur im Kontext der Stone-Klasse. Die Actor-Klasse kann weiterhin normal benutzt werden.

```
def update(self):
    self.y += 1

    if self.y > 4:
        self.init()
```

Diese Funktion verschiebt die Position bei jedem Aufruf um eins nach unten. Bei 4 ist der Rand des Spielfelds erreicht. Dann beginnt der Meteorit wieder von ganz oben. Zusätzlich überprüft die Funktion, ob der Meteorit die gleiche Position wie das Raumschiff hat. Wenn ja, hat er es getroffen und das Spiel ist verloren. Um dies abzufragen, erstellen wir später eine neue Klasse, welche die Actor-Klasse erweitert. Diese Klasse wird `PLAYER` heißen und unser konkretes Raumschiff wird den Namen `player` bekommen. Mit folgendem Code können wir dann Daten von diesem Raumschiff abfragen.

```
THEME_MELODY = [ 'G4:4', 'G4:4', 'G4:4', 'D4:2','A4:8', 'G4:2', 'C5:8', 'B4:4',
                 'A4:2']

SHIP_IMAGES = [
    Image("00000:"
          "00000:"
          "00900:"
          "09490:"
          "94449"),
    Image("00000:00900:09490:94449:09490"),
    Image("00900:09490:94449:09490:00900"),
    Image("09490:94449:09490:00900:00000"),
    Image("94449:09490:00900:00000:00000")
]

def start_game():
    music.play(THEME_MELODY, wait=False,loop=True)

    while True:

        # Raumschiff zeigen
        for image in SHIP_IMAGES:
            display.show(image)
            sleep(100)
```

Animationen: Die Melodie wird erstellt, abgespielt und animiert.

```
def start_game():
    music.play(THEME_MELODY, wait=False,loop=True)

    while True:
        # Taste abfragen
        if button_a.was_pressed() or button_b.was_pressed():
            break

        # Raumschiff zeigen
        for image in SHIP_IMAGES:
            display.show(image)
            sleep(100)

    music.stop()
```

Spielstatus Start: Solange keine Taste gedrückt ist, wird das animierte Raumschiff angezeigt.

Es wurde geprüft, ob die X- und Y-Werte des `player`-Raumschiffs mit den eigenen (Meteoriten-)Werten übereinstimmen. Sind sie gleich, hat man verloren.

```
if self.x == player.x and self.y ==
player.y:
    lost_or_win = LOST
```

Jetzt kümmern wir uns um die PLAYER-Klasse und die ROCKET-Klasse. Beide erweitern wie der Meteorit die ACTOR-Klasse.

Die Player-Klasse erhält zusätzlich den Wert `direction`. Damit wird festgelegt, wohin sich das Raumschiff bewegt. Nach links verringert den eigenen X-Wert, nach rechts erhöht ihn. In der Liste `actors` sind alle aktuell aktiven Spielelemente enthalten. Das Ergebnis steht im Bild Player.

```
player =
Player(0,4,direction=STOP,speed=100)
rocket = Rocket(speed=50)
stone  = Stone(speed=200)
actors = [player, rocket, stone]
```

Die Funktion `update` durchläuft diese Liste und aktualisiert die Elemente.

```
for a in actors:
    if a.timer.update_and_
check(difference):
        a.update()
```

Der `Timer` jedes Spielelements erhält über die Variable `difference` die Zeit, die seit dem letzten Aufruf vergangen ist. Ist die voreingestellte Zeit des Timers abgelaufen, gibt dieser den Wert `True` zurück und das Spiel startet die Funktion `update` des entsprechenden Spielelements. Der Timer setzt sich automatisch auf seinen vorgegebenen Wert zurück und alles beginnt von vorne. Durch diese Vorgehensweise laufen die Ereignisse im Spiel zu genau definierten Zeitpunkten ab, egal wie lange der Durchlauf eines Game-Loops tatsächlich gedauert hat.

Game-Loop

```
def game_loop(difference):
    input()
    update(difference)
    paint()

def do_gameloop(last_run):
    global game_state, lost_or_win

    now = running_time()
    game_loop(now-last_run)
    last_run = now
```

Steuerung mit Funktion input

Das Raumschiff kann sich nach links oder rechts bewegen. Am einfachsten geht die Steuerung dafür mit den beiden Tastern am Micro:bit.

```
if button_a.is_pressed():
        player.direction = LEFT
```

Ist der linke Taster (`button_a`) gedrückt, gibt die Funktion `is_pressed` den Wert `True` zurück. Dies wirkt sich auf die Variable `direction` (Richtung) des Objekts `player` für das Raumschiff aus. Beim Taster `button_a` setzt das Programm die Variable `direction` auf die Konstante LEFT, die dem Wert -1 entspricht. Bei

```
class Player(Actor):
    def __init__(self, x, y, direction=STOP, speed=0) -> None:
        super().__init__(x, y, speed)
        self.direction = direction

    def update(self):
        if self.direction == LEFT:
            self.x = max(0,self.x-1)

        if self.direction == RIGHT:
            self.x = min(4,self.x+1)
```

Player: Die Spielfigur, also das Raumschiff, erweitert die allgemeine Actor-Klasse.

```
ACCELEROMETER = False
MICROPHONE = True

LEFT = -1
STOP = 0
RIGHT = 1


def input():
    if button_a.is_pressed():
        player.direction = LEFT
    elif button_b.is_pressed():
        player.direction = RIGHT
    else:
        player.direction = STOP

    if ACCELEROMETER:
        if accelerometer.is_gesture("left"):
            player.direction = LEFT
        elif accelerometer.is_gesture("right"):
            player.direction = RIGHT
        else:
            player.direction = STOP

    if MICROPHONE:
        if microphone.is_event(SoundEvent.QUIET):
            player.direction = LEFT
        elif microphone.is_event(SoundEvent.LOUD):
            player.direction = RIGHT
        else:
            player.direction = STOP
```

Input: Der micro:bit hat verschiedene Sensoren. Diese können auch für die Steuerung benutzt werden.

dem anderen Taster `button_b` erhält die Variable den Wert RIGHT und sonst den Wert STOP.

Die Funktion `update` von `Player` reagiert auf den Wert der Variablen `direction` und ändert die Position des Raumschiffs entsprechend.

Der Micro:bit enthält einen Beschleunigungssensor, der feststellen kann, in welche Richtung man die Platine kippt. Diesen kann man ebenfalls zur Steuerung des Raumschiffs verwenden. Dieser Sensor erkennt mit MicroPython unterschiedliche Gesten wie auf, ab, links, rechts, nach vorne, nach hinten, schütteln und einiges mehr.

```
if accelerometer.is_gesture("left"):
    player.direction = LEFT
```

Zur Steuerung des Raumschiffs fragt das Spiel den Beschleunigungssensor (accelerometer) ab, ob die Platine nach links gekippt ist. Ist dies der Fall, liefert die Funktion `is_gesture` den Wert `True` für `left` zurück.

Genauso kann man `right` abfragen und entsprechend reagieren. Dies reicht für die Steuerung des Spiels aus. Wer e s genauer wissen will, kann die Beschleunigungswerte für die X-, Y- und Z-Richtung über `accelerometer.get_values()` abfragen. Der Beschleunigungssensor liefert für jede Richtung einen Wert zwischen 0 und 1024 zurück.

Eine weitere Möglichkeit zur Steuerung ist das Mikrofon des Micro:bit. Je lauter man singt, desto mehr geht das Raumschiff nach rechts und je leiser nach links.

```
if microphone.is_
event(SoundEvent.QUIET):
    player.direction = LEFT
```

Das Mikrofon arbeitet mit der Klasse `SoundEvent`. Der `SoundEvent.QUIET` entspricht einem Übergang von laut nach leise, der `SoundEvent.LOUD` von leiser zu laut. Die Funktion `is_event` ermittelt, was gerade passiert ist, und das Programm setzt den passenden Wert bei `player.direction`. Die Abfragen für die Eingaben sehen jetzt so aus wie im Bild Input.

Vollständigkeit

Was jetzt zum vollständigen Spiel noch fehlt, sind die Funktionen `paint` und `end_game`. Die Funktion `paint` schaltet bei jedem Durchlauf des Game-Loops alle LEDs mit `display.clear()` aus. Für das Raumschiff und andere Spielelemente schaltet sie die entsprechenden LEDs mit der passenden Helligkeit wieder an.

Dies geschieht durch den Aufruf von z. B. `display.setpixel(player.x,player.y,9)`. Diese Funktion erwartet als Parameter die Zeile und die Spalte der entsprechenden LED sowie einen Helligkeitswert zwischen 0 und 9.

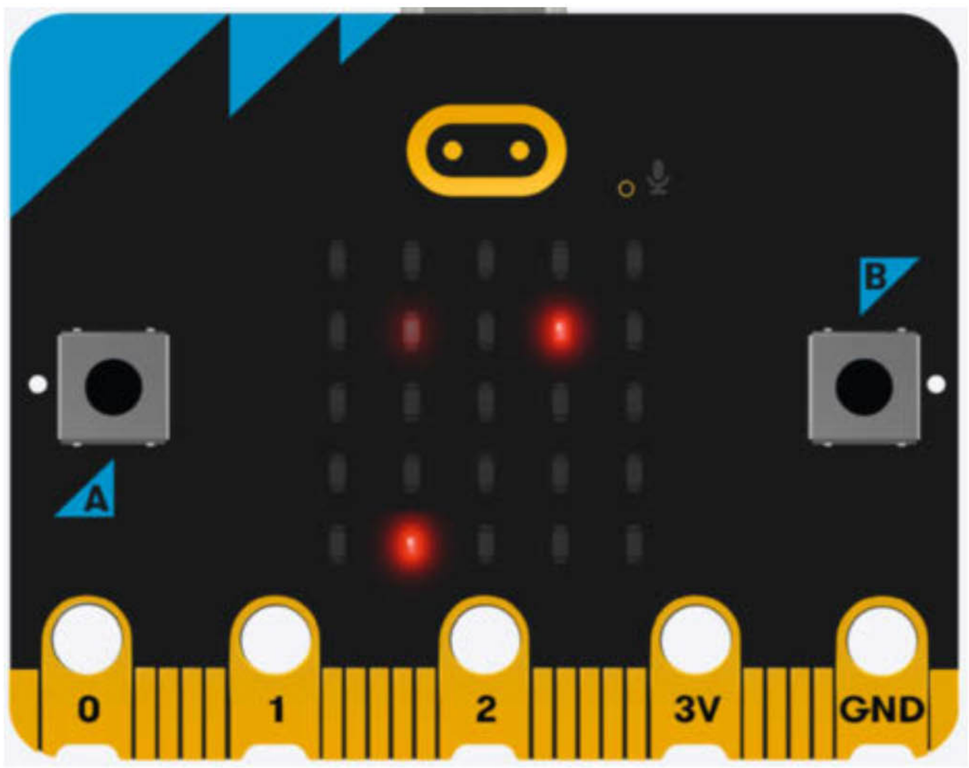
Am Ende des Spiels gibt die Funktion `end_game` bei Erfolg den Text **Win**, sonst **Lost** zurück.

Mit der Funktion `display.show("Win")` bewegt sich der Text auf den 25 LEDs von rechts nach links und ist überraschenderweise lesbar. Der Code-Block zur Darstellung befindet sich im Bild Funktion `paint` und Funktion `end_game`.

Ausblick

Der BBC Micro:bit erlaubt zusammen mit MicroPython einen interessanten und durchaus unbeschwerlichen Einstieg in die Welt der Sensoren, da er einige davon gleich selbst mitbringt. Über seine Ein-/Ausgänge lassen sich viele weitere integrieren. Durch die Programmierung mit MicroPython steht einem auch später die Welt der Mikrocontroller auf. Viele große Projekte können mit MicryPython umgesetzt werden. Auch viele Plattformen unterstützen die Programmiersprache, für die dann auch unterschiedliche Erweiterungen programmiert wurden. Diese bringen dann schon fertigen Code mit, um noch einfacher alles aus bestimmter Hardware heraus zu holen. *—das*

Jetzt ist unser Spiel fertig. Das Raumschiff fliegt am unteren Bildschirmrand und schießt schwächer leuchtende Raketen ab, während hell brennende Meteoriten von oben nach unten fallen.




```
def paint():
    display.clear()
    display.set_pixel(player.x,player.y,9)
    display.set_pixel(rocket.x,rocket.y,4)
    display.set_pixel(stone.x, stone.y, 8)

def end_game():
    if lost_or_win == WIN:
        display.show("Win")
    else:
        display.show("Lost")
```

Funktion paint und Funktion end_game: Das Spiel zeigt bei verschiedene Animationen bei unterschiedlichen Spielausgängen, gewonnen oder verloren.

Cheat-Sheet zum Coden

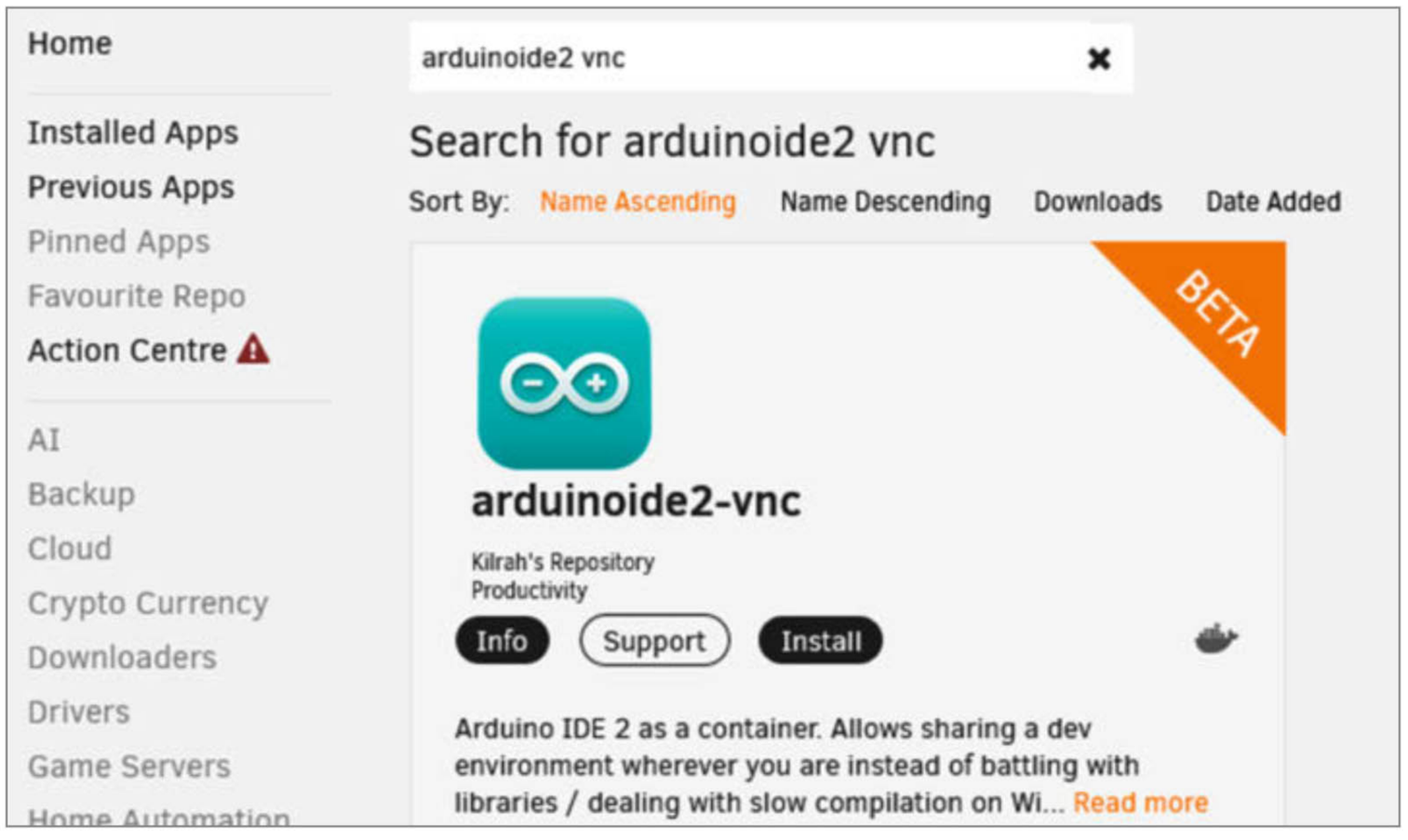
Musik und Sound	
`music.play(theme_song,wait=TRUE,loop=False)`	Spielt Melodie, entweder vordefiniert (`music.FUNK`) oder eine selbst zusammengestellte Liste von Noten, z. B. `A2:2`, Tonhöhe A2 mit der Länge 2 .Mit `wait = TRUE` wartet Micro:bit, bis die Melodie zu Ende ist, sonst führt es das Programm weiter aus. Mit `loop = TRUE` wiederholt Micro:bit die Melodie.
`music.pitch(frequenz,dauer)`	Die Frequenz, z. B. 440, entspricht dem Ton A, der mit der entsprechenden Dauer von Millisekunden zu hören ist.
`music.stop()`	Musik anhalten.
Display	
`display.show(image)`	Bild in den 5 × 5 LEDs anzeigen. `Image.HEART` oder selbst definieren.
`display.clear()`	Alle LEDs ausschalten und auf den Wert 0 setzen.
`display.set_pixel(x,y,wert)`	Für die LED in Spalte x und Zeile y den Wert `wert` setzen (einschalten).
`display.get_pixel(x,y)`	Den aktuellen Wert der LED in Spalte x und Zeile y auslesen.
`display.show(text)`	Text im Display zeigen.
Zeit	
`microbit.running_time()`	Zeit in Millisekunden seit dem Start oder Reset des Micro:bits.
Zufall	
`randint(start,end)`	Zufällige ganze Zahl zwischen `start` und `end`.
`random()`	Zufällige Nachkommazahl zwischen 0.0 und 1.0.
Beschleunigung	
`accelerometer.get_values()`	Liefert die Beschleunigungswerte für x-, y- und z-Richtung.
`accelerometer.current_gesture()`	Aktuelle erkannte Geste.
`accelerometer.is_gesture(name)`	Es wird gerade die Geste `name` erkannt.
Mikrofon	
`microphone.is_event(event)`	Liefert True zurück, falls das Ereignis eintritt. Es gibt die Ereignisse `SoundEvent.LOUD` und `SoundEvent.QUIET`.
`microphone.sound_level()`	Gibt einen Wert zwischen 0 und 255 zurück, die in etwa der aktuellen Lautsprecherausgabe entspricht.

Zentrale Arduino IDE mit Unraid

Auf dem NAS-Betriebssystem Unraid lässt sich eine Umgebung für die Entwicklung in der Arduino IDE hosten, die per Browser genutzt wird und über das Netzwerk auf lokal angeschlossene Boards zugreifen kann.

von Daniel Schwabe

Das NAS-Betriebssystem Unraid bietet nicht nur die Möglichkeit, einen Netzwerkspeicher zu betreiben, sondern gibt auch Zugriff auf verschiedene Serverdienste. In folgendem Artikel wird gezeigt, wie man eine Arduino IDE auf Unraid installiert und über den Browser remote nutzen kann. Außerdem beschreibt der Text, wie lokal angeschlossene Arduino-Boards über diese Remote-IDE bespielt werden können. Dieser Artikel setzt eine funktionierende Unraid-Installation voraus.



In den Community-Apps gibt es nur eine Arduino-App.

Die Arduino IDE

Die Arduino IDE ist der einfachste Weg, um Arduino-Boards zu programmieren. Neben der

normalen Code-Bearbeitungsfunktion bietet die IDE noch einen Manager für externe Libraries, also Code, der z. B. für bestimmte Hardware oder Funktionen vorausgesetzt wird. Durch die Nutzung der IDE über einen Server sind die Konfiguration der Software, die installierten Libraries sowie die eigenen Projekte auch auf unterschiedlichen Geräten immer gleich.

Die Installation der IDE beginnt im Apps-Tab von Unraid. Sollte das die erste App sein, die auf dem Server installiert wird, werden Sie zuerst aufgefordert, das Plugin für Community-Apps zu installieren. In diesem Fall ist den Anweisungen auf dem Bildschirm zu folgen.

In der Suchleiste des Unraid-App-Stores muss man jetzt nach „arduinoide2 vnc“ suchen.

Mit einem Klick auf install öffnet sich ein Assistent für Installationsoptionen.

Diese Einstellungen können bis auf eine alle so gelassen werden. Bei der letzten Option – Device – klickt man auf Remove, um die Zeile zu entfernen.

Am Ende bestätigt man mit Done. Jetzt dauert es ein bisschen, denn alle Docker-Bestandteile des Programms werden heruntergeladen (ja, natürlich basieren auch Unraid-Apps auf Docker).

Sobald am unteren Rand des Installationsfensters ein Button mit dem Wort Done erscheint, ist die Installation abgeschlossen und man kann mit einem Klick auf diesen Knopf zu Unraid zurückkehren.

Im Docker-Tab von Unraid erscheint jetzt ein Arduino-Logo in der Liste. Mit einem Linksklick darauf öffnet sich ein Drop-down-Menü. Dort kann über WebUI die Browseroberfläche des Add-ons geöffnet werden.

Es öffnet sich ein neues Browserfenster. Dort erscheint jetzt ein Kasten noVNC und ein Connect-Button. Auf diesen Button ist zu klicken, damit die Remote-Oberfläche des Unraid-Add-ons erscheint. Das funktioniert alles im Browser ohne extra Client-Software. Die Arduino IDE ist direkt geöffnet und installiert beim ersten Start die Updates.

Man befindet sich jetzt auf einem Remote-Desktop. Die IDE ist ein normales Fenster, das darauf frei bewegt werden kann. Um auf diesem System richtig zu arbeiten, sind zusätzlich noch ein Browser, ein Dateiexplorer, ein Texteditor und Git installiert. Mit einem Rechtsklick auf den schwarzen Hintergrund öffnet sich ein Menü, in dem diese Programme aufgerufen werden können.

Mit dieser Installation kann jetzt schon gearbeitet werden. Es können Programme geschrieben, in ein Git-Repository geladen und auch kompiliert und auf Fehler überprüft werden. Wer nicht mehr braucht, ist jetzt fertig. Wer diese Remote-Arbeitsumgebung von unterwegs nutzen möchte, sollte sich eine VPN-Verbindung in das eigene Netzwerk einrichten. Wie das funktioniert, steht im Artikel „Netzwerksicherheit mit Raspberry Pi“ in Make 1/24 auf Seite 100. Wenn man beim Remote-Arbeiten lokale Arduino-Boards flashen will, sollte man jetzt weiterlesen.

Kurzinfo

» **Eine zentrale IDE-Konfiguration**
» **Lokal angeschlossenen Arduino über Netzwerk flashen**
» **Toolkit zur Codeverwaltung auf dem Server**

Checkliste

Zeitaufwand:
30 min

Kosten:
49 Euro

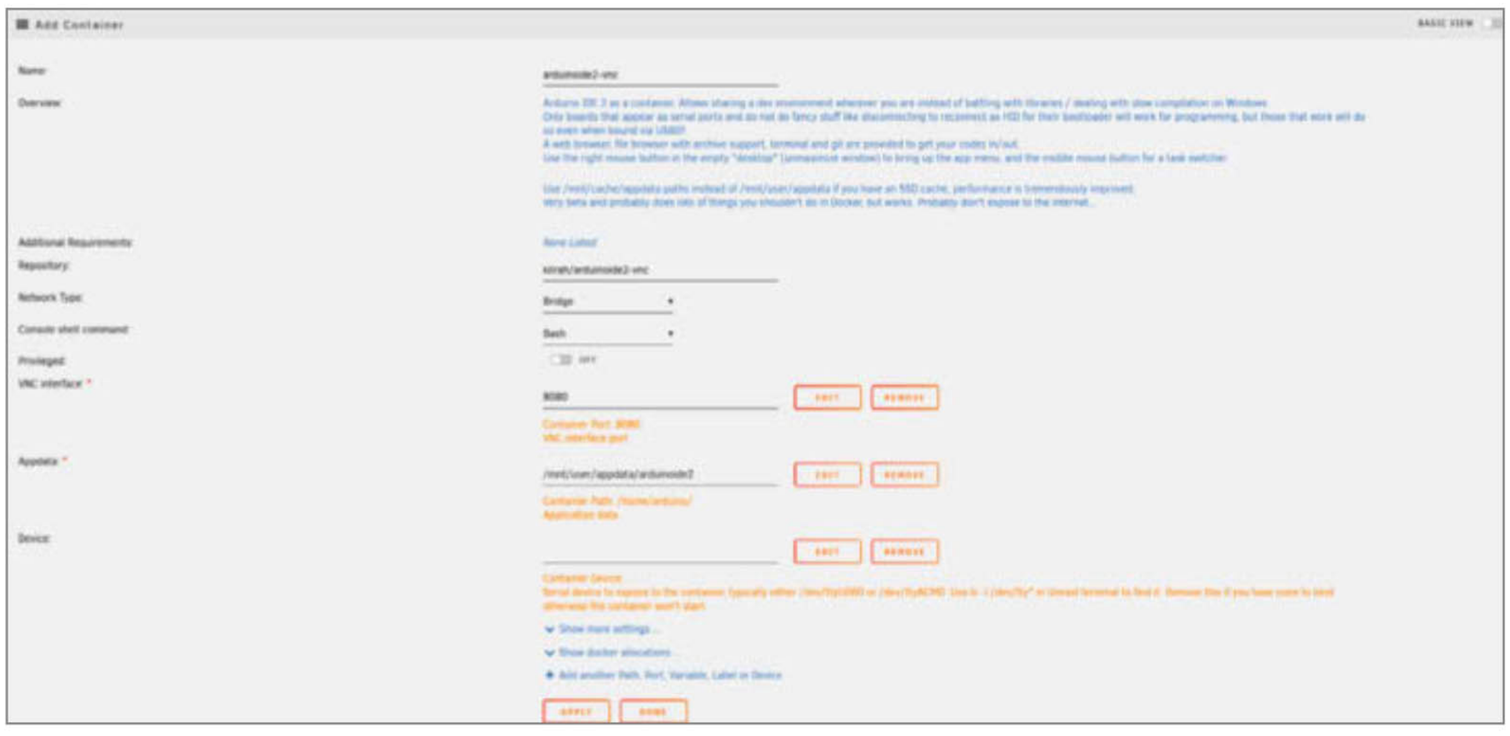
Material

» **Ein Unraid-Server**

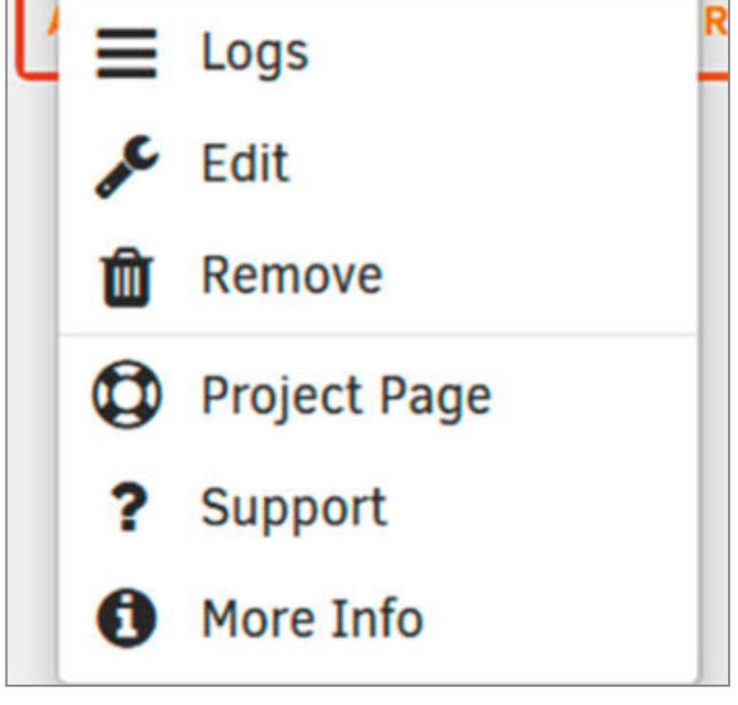
Mehr zum Thema

» Daniel Schwabe, Eigene Serverdienste mit einem Klick, Make 2/24, S. 98
» Daniel Schwabe, Reverse-Proxy auf dem Raspberry Pi, Make 2/24, S. 116

Alles zum Artikel im Web unter
make-magazin.de/x42q



Das sind die Docker-Optionen. In der Option „VNC Interface“ kann der Port geändert werden, über den auf die IDE zugegriffen wird.



Über dieses Menü können noch weitere Optionen genutzt werden. Am wichtigsten ist WebUI.

Ein simples Menü für die Programme im Remote-Desktop.

Die IDE kann wie auf einem normalen Desktop bewegt und die Fenstergröße geändert werden.

Remote arbeiten – lokal flashen

Um einen lokal angeschlossenen Arduino mit dieser Remote-IDE zu flashen, verbindet man den lokalen PC mit einem speziellen Programm mit dem Server. Der bekommt dann Zugriff auf den über USB angeschlossenen Arduino und behandelt ihn, als ob er physisch mit dem Server verbunden ist.

Als Erstes installiert man ein weiteres Add-on auf dem Unraid-Server – wieder über das Apps-Tab. Dort dann nach „USB Manager“ suchen.

Auch dieses Add-on wieder mit einem Klick auf Install installieren und warten, bis im Installationsfenster der Done-Button erscheint.

Im Unraid-Tab „Settings/User Utilities/USB Devices“ auf das Zahnrad oben rechts klicken (oder nach einem Reload der Unraid-Seite erscheint auch direkt im Menüband der Punkt USB). Jetzt ist man in den Einstellungen des USB-Plugins.

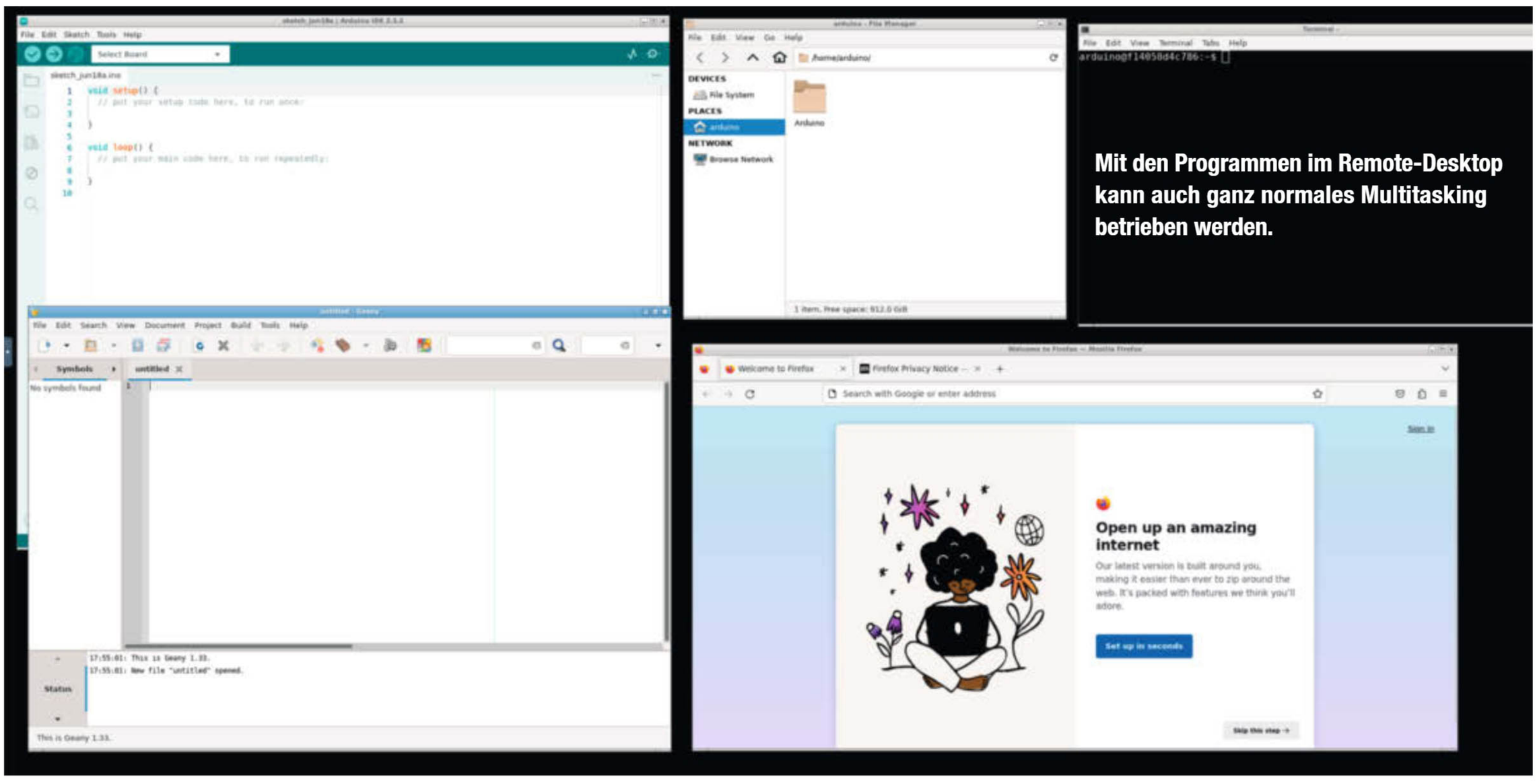
Die Option „Enable USBIP“ muss man jetzt auf Enabled stellen. Falls sie schon Enabled sein sollte, obwohl das eine frische Installation ist, einmal die Option Disablen, Apply anklicken und anschließend wieder auf Enabled stellen. Danach erscheinen viele neue Optionen auf der Seite.

Als Erstes muss unter „Addon Plugin/Install/remove addon plugin for USBIP“ INSTALL gewählt werden. Danach die Einstellungen aus demScreenshot von der nächsten Seite kopieren und unter allen veränderten Bereichen auf APPLY klicken.

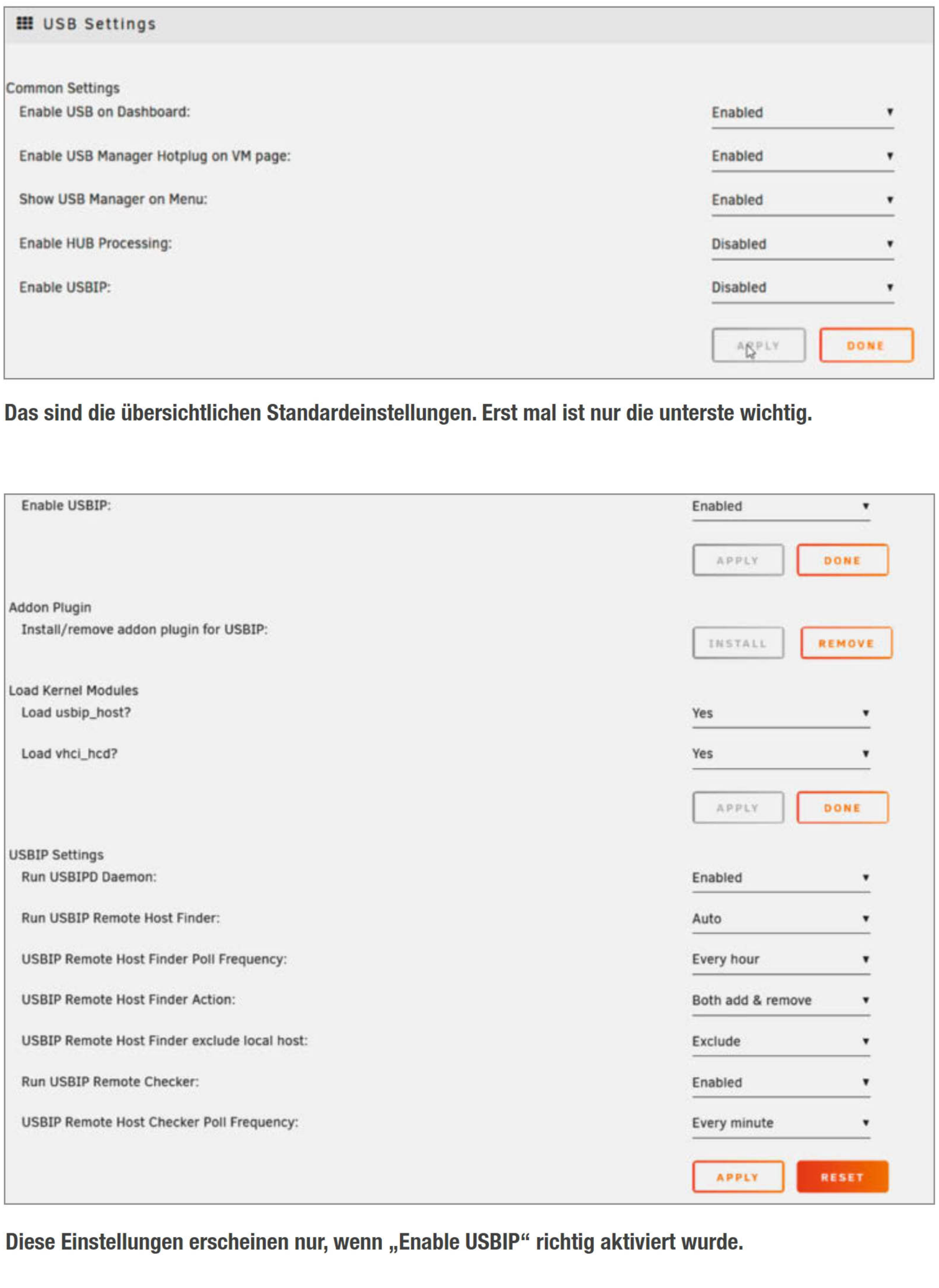
Jetzt startet das USBIP-Plugin und man kann sich mit einem Computer, der ein Gerät bereitstellt, verbinden. Und um diesen Computer muss man sich jetzt kümmern.

Unter Windows muss als Erstes die Software usbipd-win installiert werden. Der Link dazu befindet sich in der Online-Info. Nach der Installation braucht man jetzt ein Arduino-Board (in diesem Artikel wird ein Arduino Mega als Beispiel verwendet). Den Arduino jetzt per USB an den Computer anschließen.

Danach braucht es ein Terminal. Dafür im Windows-Startmenü nach Terminal suchen und das erste Ergebnis als Administrator öffnen. Dafür muss man nicht direkt auf das Symbol, sondern in der Liste daneben auf „Als Administrator ausführen“ klicken.

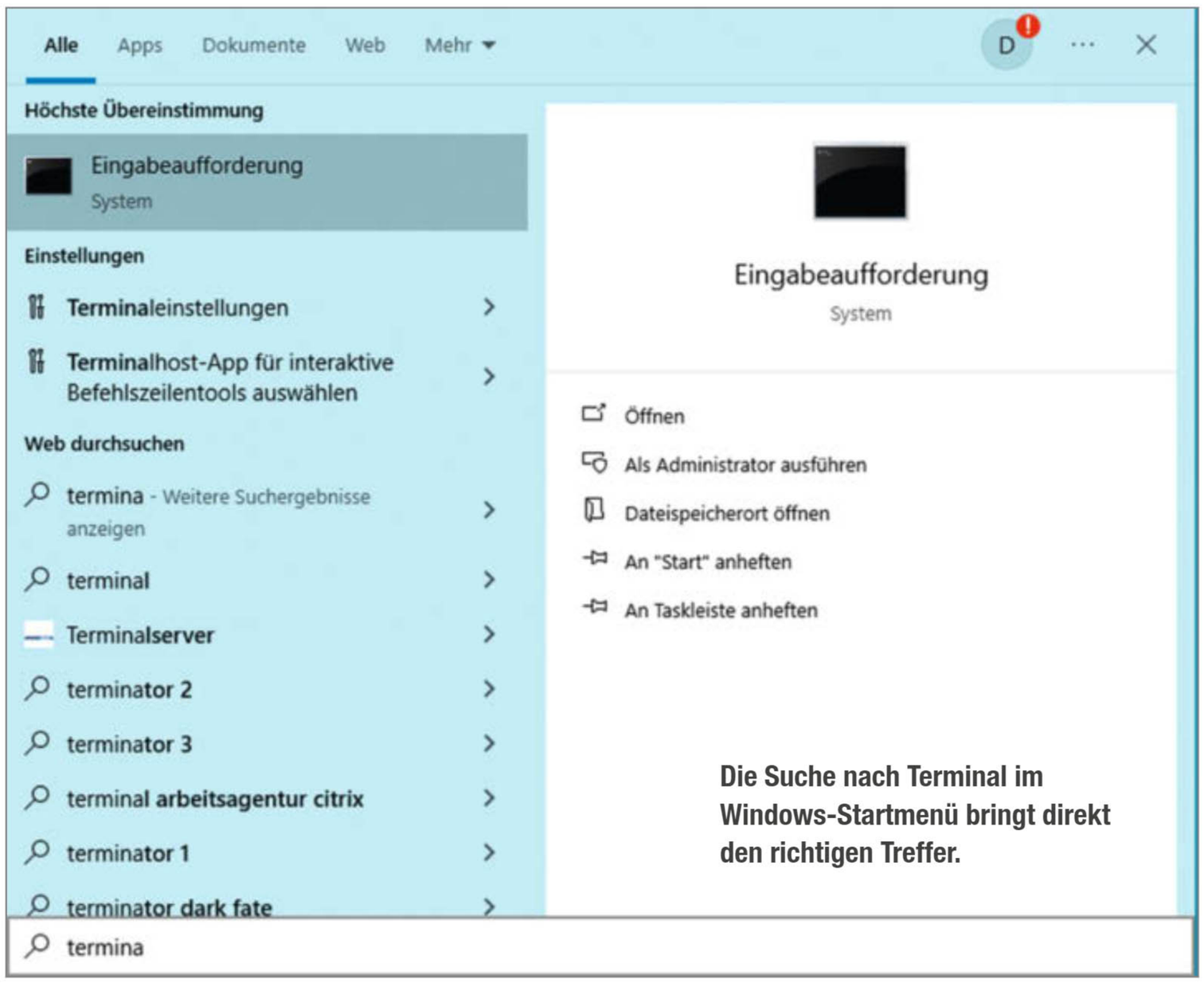


Bei der Suche nach USB Manager erscheint noch ein zweites Ergebnis. Das hat allerdings einen anderen Namen. Nur dieses Add-on wird gebraucht.

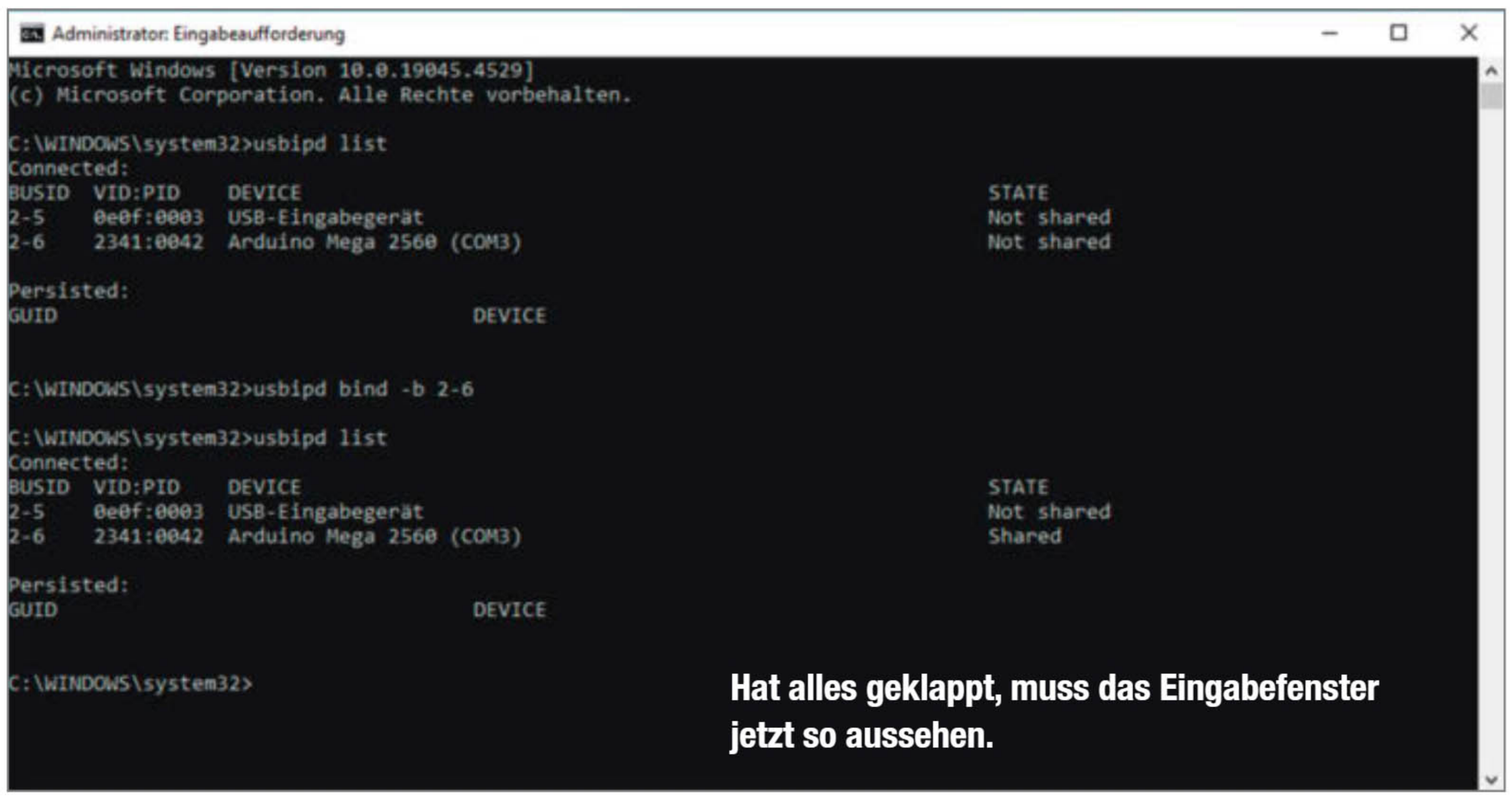


Das sind die übersichtlichen Standardeinstellungen. Erst mal ist nur die unterste wichtig.

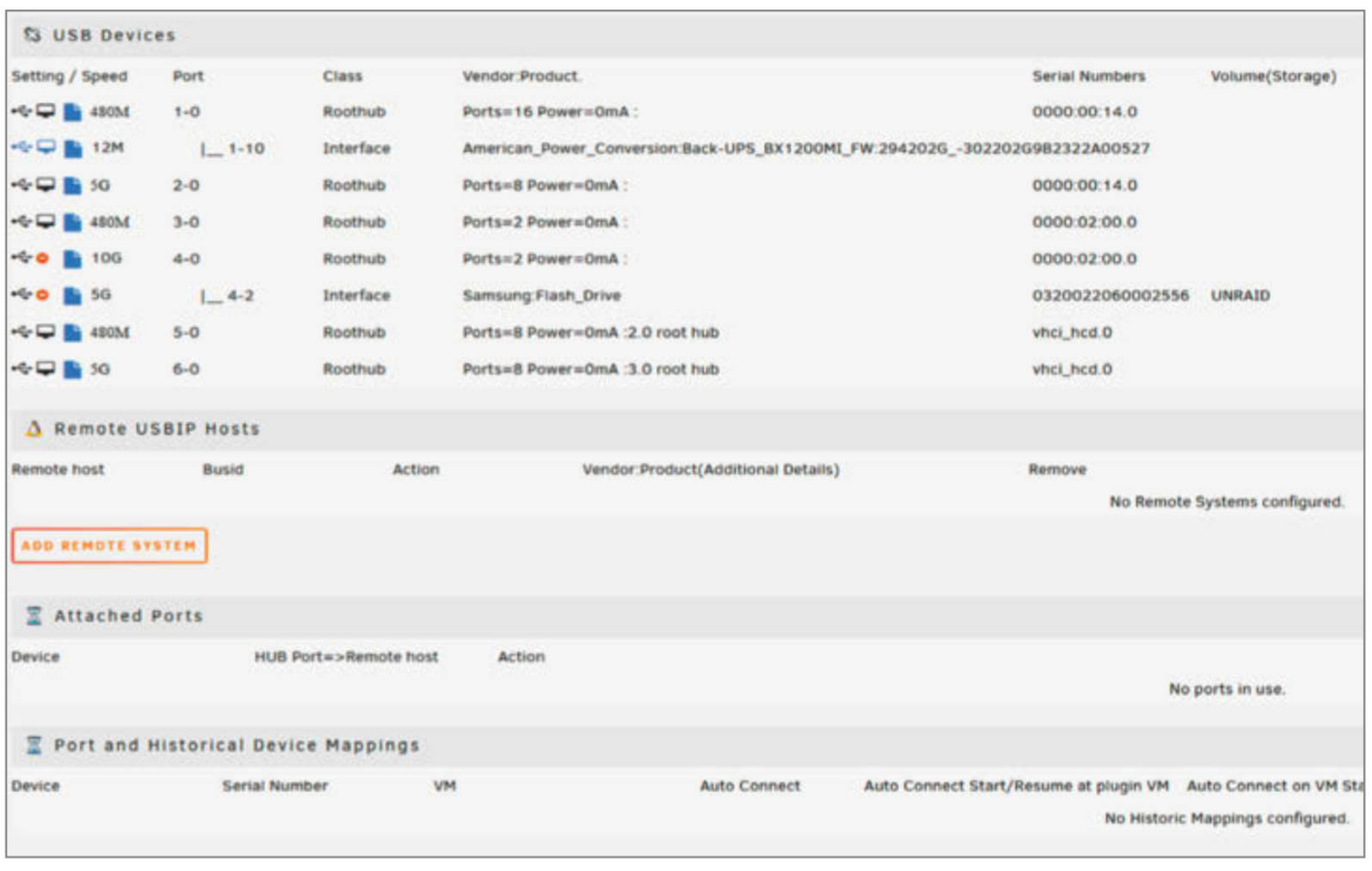
Diese Einstellungen erscheinen nur, wenn „Enable USBIP“ richtig aktiviert wurde.



Die Suche nach Terminal im Windows-Startmenü bringt direkt den richtigen Treffer.



Hat alles geklappt, muss das Eingabefenster jetzt so aussehen.



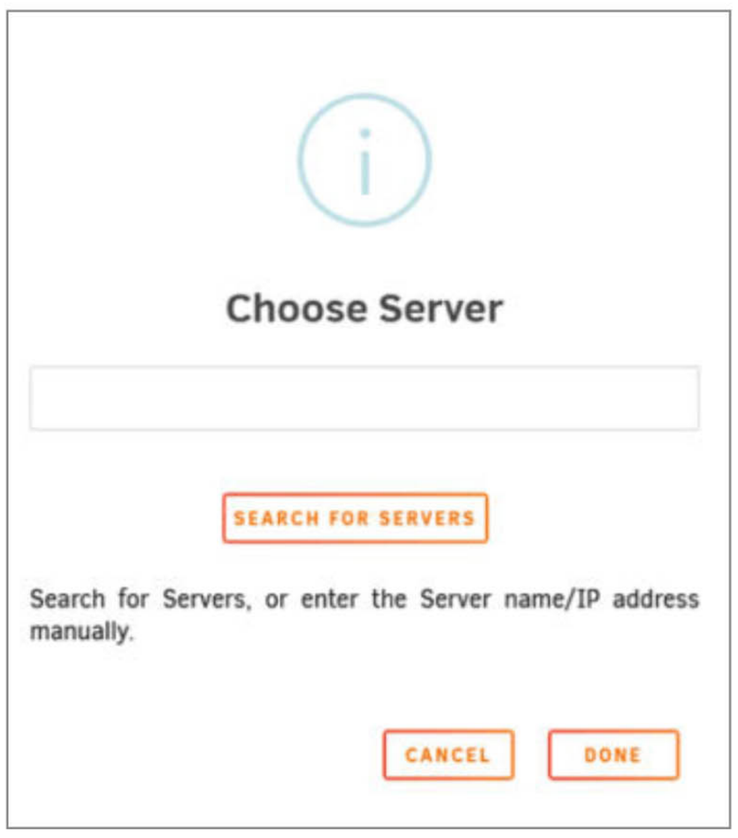
In diesem Terminal gibt man jetzt den Befehl `usbipd list` ein. Dadurch erscheinen alle angeschlossenen USB-Geräte. Der Arduino Mega in diesem Beispiel erscheint in der Liste als „Arduino mega 2560 (COM3)“. Ganz links neben jedem Eintrag in der Liste stehen zwei Zahlen, bei diesem Arduino ist es 2-6. Über diese Zahlen wird das USB-Gerät jetzt freigegeben. Mit dem Befehl `usbipd bind -b 2-6` (2-6 muss durch die Zahlen des Gerätes ersetzt werden) wird der Arduino geteilt. Gibt man noch einmal `usbipd list` ein, erscheint rechts in der Liste neben dem Arduino das Wort Shared.

Sollte eine Fehlermeldung „usbipd: error: Access denied“ zu sehen sein, wurde das Terminal nicht als Administrator gestartet.

Auf dem PC ist man jetzt fertig. Jetzt muss das Gerät mit Unraid und dem Arduino-Add-on verbunden werden.

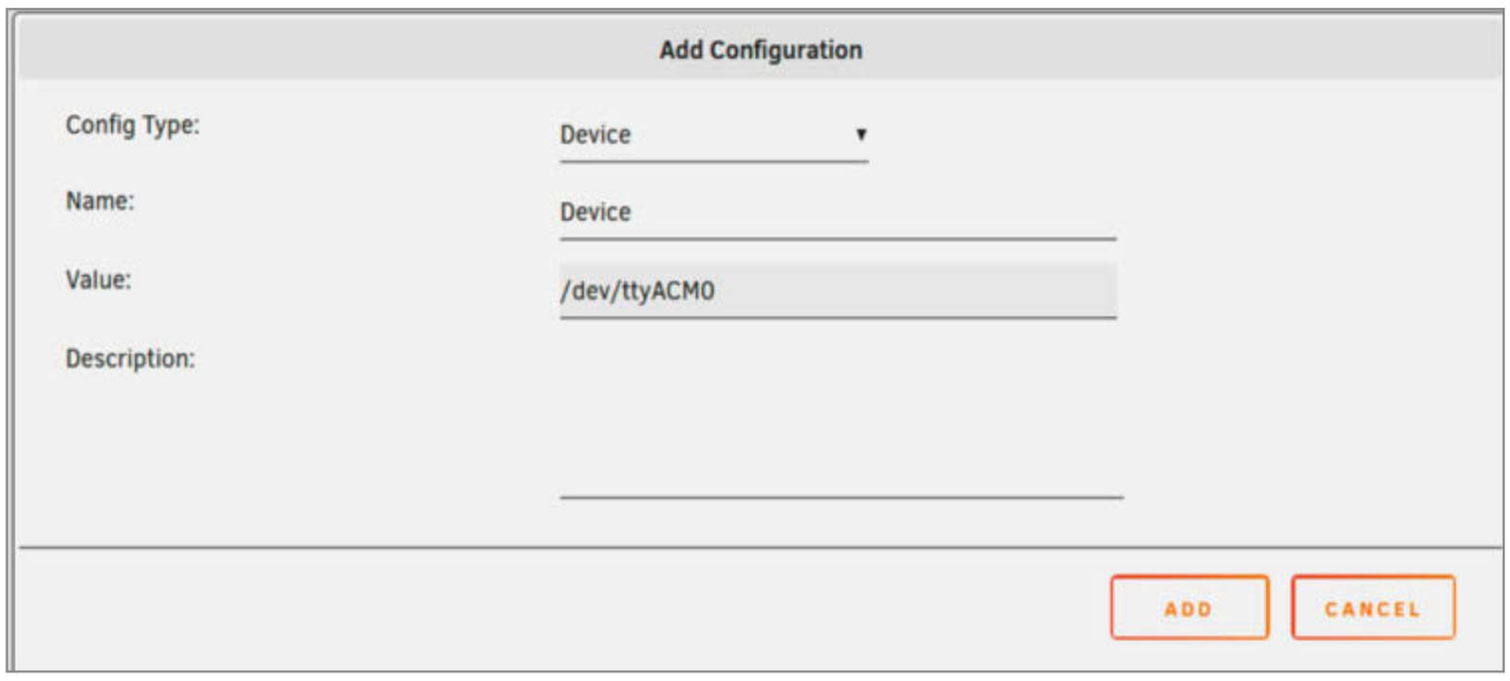
Dafür geht man in das neue USB-Tab von Unraid. Dort erscheinen mehrere Abschnitte. Der Abschnitt „Remote USBIP Hosts“ ist der wichtige. Dort klickt man auf „ADD REMOTE SYSTEM“. Es öffnet sich ein neues Fenster, in dem ein USBIP-Server (in diesem Fall unser PC) angegeben werden muss. Mit einem Klick auf „SEARCH FOR SERVERS“ sollte das Programm den Server automatisch finden. Die IP erscheint bei einem gefundenen Server danach im Eingabefeld. Auf dem PC kann im Terminal mit dem Befehl `ipconfig` überprüft werden, ob die IP stimmt. Mit einem Klick auf Done erscheinen jetzt der PC und der Arduino in der Liste der verfügbaren Geräte und er wird auch direkt an den Server angeschlossen.

Jetzt muss man dieses Gerät noch in den Arduino-Container weiterleiten. Dafür in der



Die Suche nach dem Server nimmt einem das Plugin ab.

Hier tauchen alle Geräte auf, die mit dem Server verbunden sind. Die Geräte oben sind echte physische Geräte.



Mit diesen Einstellungen erkennt die IDE den Arduino.

Unraid-Übersicht ganz rechts oben das Konsolenfenster öffnen.

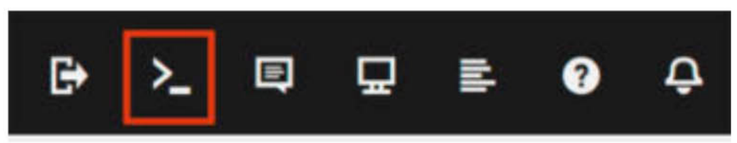
In dieser Konsole ist dann der Befehl `ls -l /dev/tty*` einzugeben. Es öffnet sich eine lange Liste mit Geräten. Am unteren Teil befindet sich ein Eintrag, der anders aussieht als die anderen. In diesem Fall `/dev/ttyACM0`. Falls unklar ist, welcher Eintrag der richtige ist, kann man das USB-Gerät in den Einstellungen noch einmal mit detach entfernen, den Befehl ausführen, wieder attachen zum Verbinden und dann schauen, welcher Eintrag in der Geräteliste neu dazukommt.

Diese Zeichenfolge `/dev/ttyACM0` muss jetzt in die Einstellung des Docker-Containers der Arduino IDE eingetragen werden. Das kann man entweder direkt bei der Installation machen und die Zeichenfolge in den Bereich Device eintragen (anstatt diesen zu löschen). Oder man bearbeitet den bereits installierten Container. Dafür wieder im Unraid-Docker-Tab das Drop-down-Menü des Arduino-Containers öffnen und auf Edit klicken. Es öffnet sich wieder das Einstellungsfenster, das man bei der Installation schon hatte. Jetzt muss man den Device-Eintrag wieder hinzufügen. Dafür klickt man ganz unten auf „+ Add another Port, Variable, Label or Device“. Im sich öffnenden Dialog stellt man die Art des Eintrages auf Device, gibt ihm den Namen Device und tippt die Zeichenfolge des USB-Gerätes ein. Danach klickt man auf ADD und auf der großen Übersicht wieder auf Done.

Der Container wird jetzt neu erstellt. Wenn er fertig ist und man wieder auf das WebUI zugreift, kann man in der Arduino IDE unter Tools den Port und das Board auswählen, als ob alles physisch miteinander verbunden wäre. Über File/Examples/01.Basics/Blink kann man auch direkt ein kleines Beispiel in die IDE laden und mit einem Klick auf den Upload-Pfeil auf das Arduino-Board aufspielen. Hat alles geklappt, blinkt der Arduino jetzt fröhlich vor sich hin. —*das*

```
crw--w---- 1 root tty      4, 34 May  7 18:20 /dev/tty34
crw--w---- 1 root tty      4, 35 May  7 18:20 /dev/tty35
crw--w---- 1 root tty      4, 36 May  7 18:20 /dev/tty36
crw--w---- 1 root tty      4, 37 May  7 18:20 /dev/tty37
crw--w---- 1 root tty      4, 38 May  7 18:20 /dev/tty38
crw--w---- 1 root tty      4, 39 May  7 18:20 /dev/tty39
crw--w---- 1 root tty      4,  4 May  7 18:21 /dev/tty4
crw--w---- 1 root tty      4, 40 May  7 18:20 /dev/tty40
crw--w---- 1 root tty      4, 41 May  7 18:20 /dev/tty41
crw--w---- 1 root tty      4, 42 May  7 18:20 /dev/tty42
crw--w---- 1 root tty      4, 43 May  7 18:20 /dev/tty43
crw--w---- 1 root tty      4, 44 May  7 18:20 /dev/tty44
crw--w---- 1 root tty      4, 45 May  7 18:20 /dev/tty45
crw--w---- 1 root tty      4, 46 May  7 18:20 /dev/tty46
crw--w---- 1 root tty      4, 47 May  7 18:20 /dev/tty47
crw--w---- 1 root tty      4, 48 May  7 18:20 /dev/tty48
crw--w---- 1 root tty      4, 49 May  7 18:20 /dev/tty49
crw--w---- 1 root tty      4,  5 May  7 18:21 /dev/tty5
crw--w---- 1 root tty      4, 50 May  7 18:20 /dev/tty50
crw--w---- 1 root tty      4, 51 May  7 18:20 /dev/tty51
crw--w---- 1 root tty      4, 52 May  7 18:20 /dev/tty52
crw--w---- 1 root tty      4, 53 May  7 18:20 /dev/tty53
crw--w---- 1 root tty      4, 54 May  7 18:20 /dev/tty54
crw--w---- 1 root tty      4, 55 May  7 18:20 /dev/tty55
crw--w---- 1 root tty      4, 56 May  7 18:20 /dev/tty56
crw--w---- 1 root tty      4, 57 May  7 18:20 /dev/tty57
crw--w---- 1 root tty      4, 58 May  7 18:20 /dev/tty58
crw--w---- 1 root tty      4, 59 May  7 18:20 /dev/tty59
crw--w---- 1 root tty      4,  6 May  7 18:21 /dev/tty6
crw--w---- 1 root tty      4, 60 May  7 18:20 /dev/tty60
crw--w---- 1 root tty      4, 61 May  7 18:20 /dev/tty61
crw--w---- 1 root tty      4, 62 May  7 18:20 /dev/tty62
crw--w---- 1 root tty      4, 63 May  7 18:20 /dev/tty63
crw--w---- 1 root tty      4,  7 May  7 18:20 /dev/tty7
crw--w---- 1 root tty      4,  8 May  7 18:20 /dev/tty8
crw--w---- 1 root tty      4,  9 May  7 18:20 /dev/tty9
crw-rw---- 1 root dialout 166,  0 Jun 18 16:39 /dev/ttyACM0
crw-rw---- 1 root dialout   4, 64 May  7 18:20 /dev/ttyS0
crw-rw---- 1 root dialout   4, 65 May  7 18:20 /dev/ttyS1
```

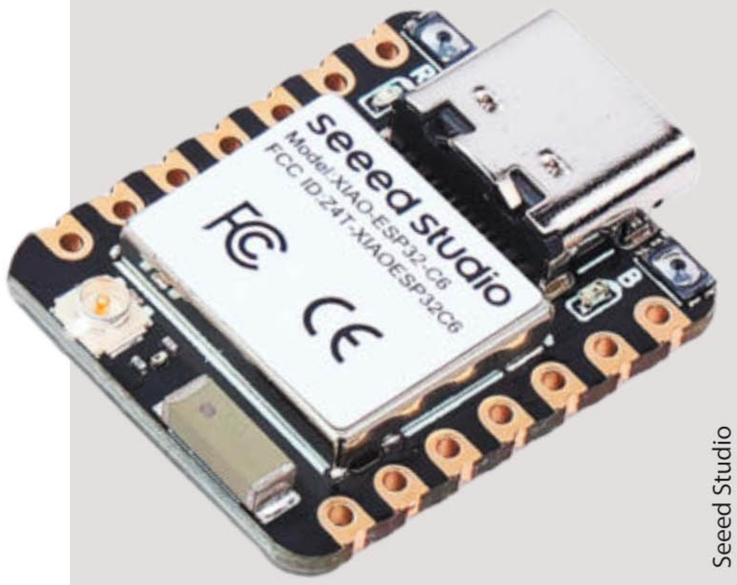
So sieht diese Liste dann aus.



Oben rechts in Unraid befinden sich diese Icons. Das eingerahmte öffnet die Konsole.

XIAO ESP32C6

Miniboard für Matter, Zigbee und mehr



Seeed Studio

Dieses ESP32-Mikrocontroller-Board von Seeed Studio ist nur halb so groß wie eine SD-Karte, beherrscht aber nahezu jeden Funkstandard fürs SmartHome, nämlich 2.4GHz-WiFi 6 (802.11ax), Bluetooth 5(LE) und den für Bluetooth Mesh, Thread und Zigbee notwendigen Standard IEEE 802.15.4. Auf der kleinen Platine stehen außerdem elf digitale Ein-/Ausgänge und sechs analoge Anschlüsse sowie eine Buchse für eine externe Antenne zur Verfügung. Die zum Teil mehrfach belegten Anschlüsse verteilen sich auf die beiden Platinenkanten (castellated Pads zum direkten Auflöten und für Löcher für Pin Header) und die Unterseite des Boards.

Die Onboard-Antenne soll für bis zu 80 m Reichweite bei Bluetooth sorgen. I²C sowie UART sind ebenfalls an Bord. Der ESP32-C6 enthält zwei RISC-V-Prozessoren, die mit 160 bzw. 20 MHz getaktet werden. Ihnen stehen 512 kByte SRAM sowie 4 MByte Flash-Speicher zur Verfügung. Im Deepsleep-Modus soll der Stromverbrauch bei 15 Mikroampere liegen. Die Stromversorgung erfolgt per USB-C-Buchse oder über einen (nicht mitgelieferten) Lithium-Akku. Die Lötanschlüsse dafür befinden sich an der Platinenunterseite. Eine entsprechende Ladeelektronik ist onboard verbaut.

Für das Board gibt es eine sehr ausführliche Online-Dokumentation, unter anderem auch für das Installieren der Matter-Firmware. —*hgb*

Hersteller	Seeed Technology Co.,Ltd.
URL	make-magazin.de/xjnm
Preis	5,20 US-$

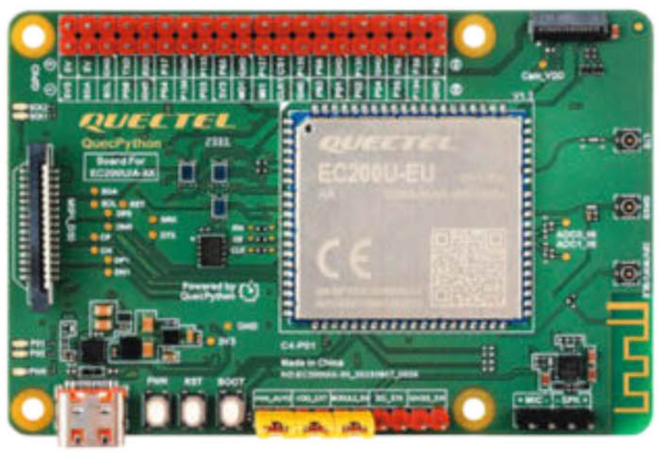
EC200U-EU

LTE-fähiges Mikroprozessorboard im Raspi-Format

Im gleichen Format wie der Raspberry Pi 4/5 und mit einer GPIO-Leiste, die zu den Raspis kompatibel sein soll, das ist das Board EC200U-EU von Waveshare. Auch Bluetooth (4.2) und WLAN (nur 2,4 GHz) sind an Bord. Ein LCD-Display oder eine Kamera können per MIPI-Connector angeschlossen werden.

Zum Raspi fehlt ein HDMI-Anschluss, dafür gibt es LTE für die Datenübertragung per Mobilnetz, vorausgesetzt, man stattet den kleinen Computer mit einer entsprechenden kostenpflichtigen Mikro- oder Nano-SIM-Karte aus. Zwei Slots dafür stehen auf der Unterseite der Platine bereit. Im Upload werden damit bis zu 5 Mbps, beim Download 10 Mbps erreicht.

Auf dem Board arbeitet eine EC200U-EU von Quectel. Das EU steht in diesem Fall für die europäische Version, was Funkkanäle usw. angeht. Auf diesem Prozessor arbeitet kein Linux, stattdessen gibt es vom Hersteller eine Python-Version: QuecPython, zu der online eine ausführliche Dokumentation zur Verfügung steht. Die Programmiersprache ist eine Erweiterung von MicroPython. Auch für VSCode steht ein Plug-in zur Verfügung. Für das Programm stehen 832 KByte zur Verfügung.



Waveshare

Das Board bietet drei UARTs, USB 2.0, PCM als Audioausgang, zwei I²C, SPI sowie einen Mikrofon- und Lautsprecheranschluss. Das Board kann außerdem die Signale von Navigationssatelliten (GPS, GLONASS, BDS, Galileo und QZSS) empfangen. Für externe Antennen von GPS sowie fürs Mobilnetz, WLAN und Bluetooth sind Anschlüsse vorhanden. —*hgb*

Hersteller	Waveshare
URL	make-magazin.de/xjnm
Preis	59 US-$

SSD-Shield X1011

Vier NVMe-SSDs für Raspi 5

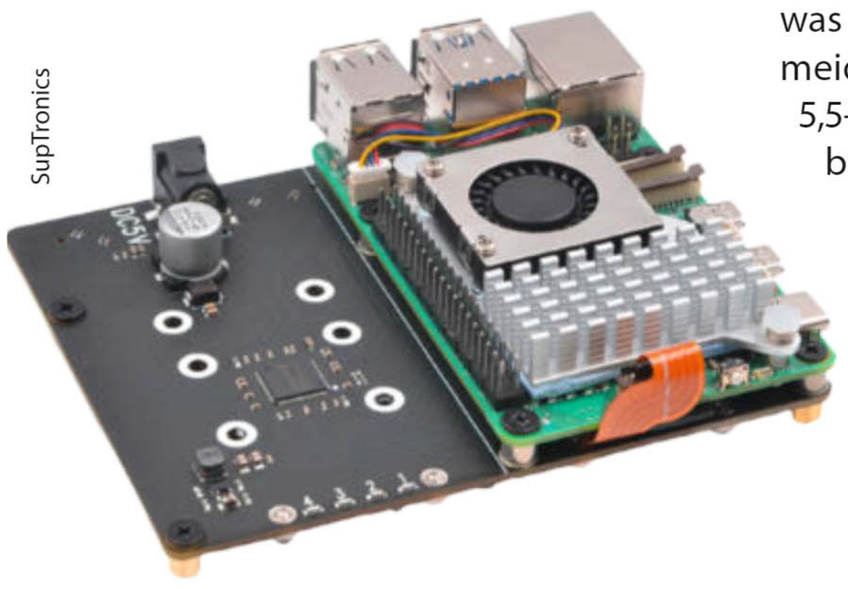
Suptronics (ja, mit „p“) bietet ab sofort das Shield X1011 für den Raspberry Pi 5 an. Vier Steckplätze für M.2 NVMe-SSDs (Fullsize, Format bis zu 2280) stellt die Platine bereit. Die Datenübertragung erfolgt per Folienkabel (mitgeliefert) über die PCIe-2.0-Buchse des Raspberry Pi, sodass Datenraten bis 5 Gbps möglich werden. In den technischen Infos des Herstellers werden Testergebnisse mit knapp 400 MB/s beim Lesen der Daten angegeben.

Die Montage erfolgt mit dem Schraubensatz unter dem Raspberry, der GPIO-Port bleibt daher frei für Erweiterungen. Außerdem steht so weiterhin Platz über der Raspberry-

SupTronics

Platine für Kühlkörper und Lüfter zur Verfügung. Mit 109 x 87.2 mm ragt die Platine aber seitlich deutlich über den Raspberry hinaus.

Booten von den Laufwerken (NVMe boot) ist seit Firmware EEPROM 2024/05/17 möglich, was die langsamen Boot-Zeiten von SD vermeidet. Für die Stromversorgung ist eine 5,5-mm-Hohlbuchse vorhanden. Der Raspberry wird dann über Pogo-Pins mitversorgt. Ein Netzteil (5 V/mind. 5A) gehört nicht zum Lieferumfang. —*hgb*



SupTronics

Hersteller	SupTronics Technologies Limited
URL	make-magazin.de/xjnm
Preis	51 US-$

Risc-V-Assembler-Buch

Wie in alten Zeiten

Für die meisten Anwendungen spielt Assembler zwar kaum noch eine Rolle, aber bei der Programmierung von Mikrocontrollern ist die Sprache immer noch wichtig. Das gilt auch für RISC-V-Systeme, weil die noch relativ neu sind und viel Software noch angepasst werden muss.

Edson Borin möchte seinen Lesern die Kunst der Assembler-Programmierung auf RISC-V-CPUs beibringen. Er beginnt mit den Grundlagen und erklärt die Repräsentation von Zahlen, Arrays und Strukturen im Speicher. Anschließend geht er über zu den Formaten für Objekt-Dateien und ausführbare Dateien. Mit Tools wie objdump und readelf demonstriert er dabei die Bedeutung von Symbolen und Programm-Sektionen.

Testen und ausprobieren kann man das Gelernte in einem JavaScript-basierten RISC-V-Simulator, der im Browser läuft, den man ebenfalls auf der Website findet. Dazu gibt es auch noch ein „RISC-V ALE Exercise Book v1.0“, mit dem man in diversen Aufgaben seinen Lernfortschritt testen kann.

Mit diesem Rüstzeug wendet er sich dann der Assembler-Syntax zu. Konkret stellt er die Programmierung mit dem RV32IM-Befehlssatz vor, der dem grundlegenden RISC-V-Befehlssatz – ergänzt um die Multiplikation und Division von Ganzzahlen – entspricht.

Schrittweise stellt er Register und Instruktionen zum Laden und Speichern von Werten vor. Weiter geht es dann mit bedingten und unbedingten Verzweigungen. Viele Konstrukte demonstriert er, indem er C-Anweisungen, wie zum Beispiel, `if`, `for` oder `while`, in entsprechenden Assembler-Code übersetzt. Ausführlich erklärt er die Programmierung von Funktionen.

In den letzten Kapiteln beschreibt er fortgeschrittene Techniken, wie den Zugriff auf externe Peripherie mittels Port-IO und Memory-mapped IO.

Insgesamt vermittelt der Autor die Grundzüge der RISC-V-Programmierung auf kompakte und ansprechende Weise. Das Buch ist auf der Website kostenlos verfügbar. In Papierform gibt es das Buch bei Amazon als Print-on-Demand. *—Maik Schmidt*

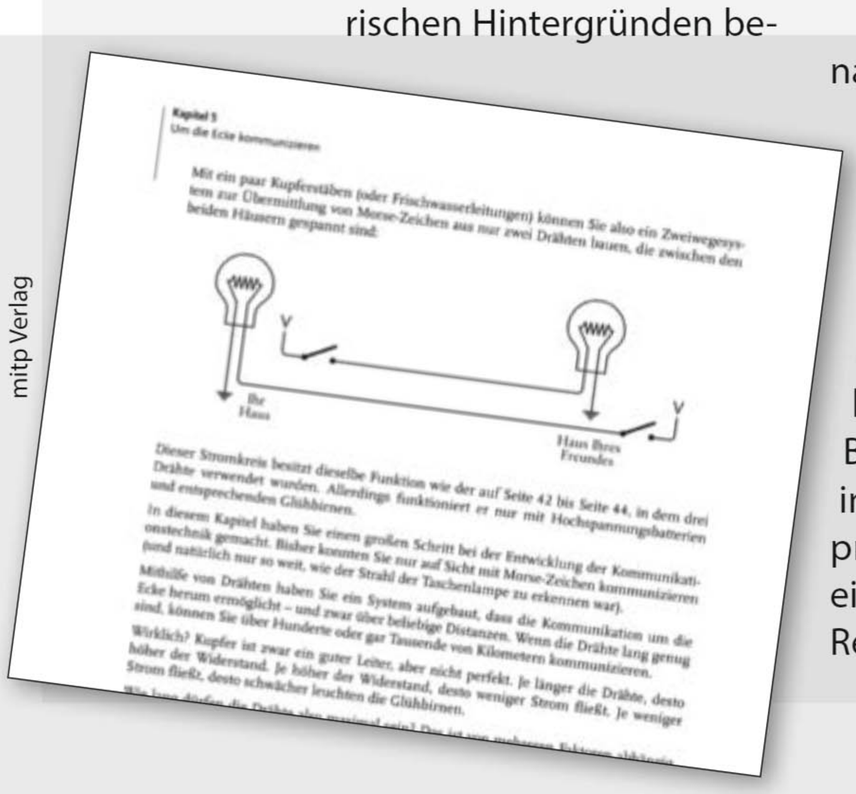
Edson Borin

Titel	An Introduction to Assembly Programming with RISC-V
Autor	Edson Borin
Verlag	Selbstverlag
Umfang	188 Seiten
ISBN	978-6500158113
Preis	E-Book kostenlos, Print-on-Demand 10 €

Code

Wie Computer funktionieren

Computer lassen sich in der Regel auch dann problemlos bedienen, wenn man nichts von Digitaltechnik versteht. Programmiert man jedoch für die Hardware – z. B. in Projekten mit Mikrocontrollern –, kann es helfen, sich einmal damit beschäftigt zu haben, wie ein Computer grundlegend funktioniert. Dafür lohnt sich ein Blick in das nun auch auf Deutsch verfügbare Buch „Code“ von Charles Petzold. Von Anekdoten und historischen Hintergründen begleitet, nimmt der Autor die Leser darin mit zu den Anfängen des Computerzeitalters und erklärt Schritt für Schritt die technische Funktionsweise von Rechnern.



mitp Verlag

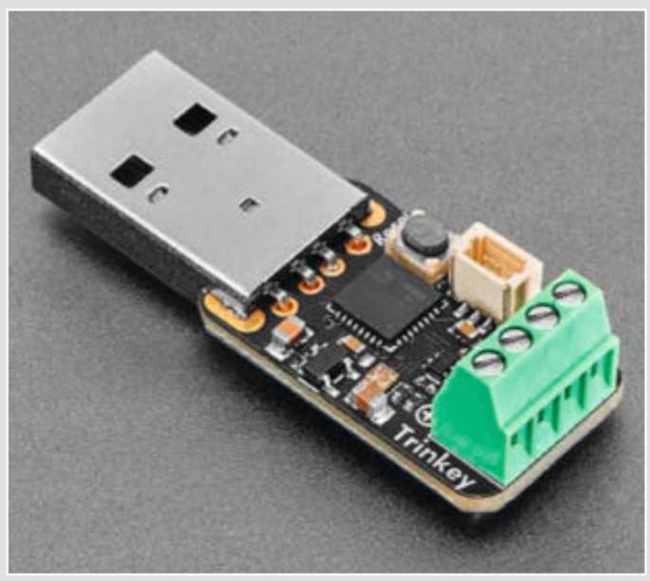
mitp Verlag

Insgesamt liefert „Code“ mit der gelungenen Art zu erklären wertvolle Grundlagen für Fortgeschrittene, die ihr Wissen über Computerhardware noch weiter vertiefen möchten. *—akf*

Autor	Charles Petzold
Verlag	mitp Verlags GmbH & Co. KG
Umfang	472 Seiten
ISBN	978-3-7475-0628-8
Preis	39,99 € (Softcover, deutsche Fassung)

Adafruit Pixel Trinkey

Mini Board für smarte LEDs



Adafruit

Das Pixel-Trinkey-Controllerboard ist nicht viel größer als ein klobiger USB-Stecker und soll an normalen USB-Ports ungefähr 150 adressierbare LEDs versorgen und ansteuern können, wenn man sie nicht zu hell leuchten lässt. Die von Adafruit mit dem Markennamen „Neopixel“ versehenen, adressierbaren LEDs vom Typ SK6812 oder WS2812 sind smarte LEDs, die sich einzeln durch einen Mikroprozessor ansteuern lassen. Für die Montage reicht dann ein Schraubendreher, solange die LED-Segmente die üblichen drei oder vier Kabel für Stromversorgung, Daten und Clock (je nach LED-Typ) haben.

Für kleinere Installationen ist der Pixel Trinkey sehr praktisch, nimmt er doch wenig Platz ein, ist leicht anzuschließen und kann USB-Power nutzen, was heutzutage fast allgegenwärtig ist. Pixel Trinkey bietet einen Prozessor vom Typ ATSAMD21E18 32-bit Cortex M0+ mit 48 MHz, 256 KByte Flash und 32 KByte RAM. Natives USB ist mit an Bord sowie eine einzelne RGB-LED für schnelle Tests.

Möchte man mehr LEDs versorgen, müssen die 5V für die LEDs von einem externen Netzteil kommen, das dann aber natürlich auch an GND am Block angeschlossen werden muss. Weiterhin gibt es einen JST-SH-3-Pin-Verbinder, der GPIO-D4, 3V und GND anschließt und an den etwa ein weiterer Taster, Potenziometer oder Infrarotdioden angeschlossen werden können. *—caw*

Hersteller	Adafruit
URL	make-magazin.de/xjnm
Preis	9,95 US-$

Pendrive S3 128 MByte

Ausprobiert — von Make:

USB-Stick mit ESP32-S3 für Pentesting und Entwicklung

Das Pendrive S3 ist ein ESP32-S3-Board mit USB-C, WS2812B-RGB-LED. Mithilfe von TinyUSB kann sich der ESP32-S3 unter anderem als USB-Datenstick, USB-Tastatur, USB-Maus, Audio-, Video- oder Netzwerkgerät präsentieren. Das Kit enthält ein Spritzguss-Plastik-Gehäuse und ist dann äußerlich kaum von einem normalen USB-Stick zu unterscheiden.

Die RGB-LED bleibt durch das Plastikgehäuse hindurch sichtbar. Technisch interessant ist ebenfalls der kapazitive Touch-Button, der mit einer Feder an das Gehäuse herangeführt wird und ohne Durchführung im Gehäuse eine Bedienung von außen erlaubt. Das Pendrive S3 wird auch von CircuitPython unterstützt.

Inzwischen gibt es vier Firmware-Images, die man bequem aus Chrome-basierten Browsern heraus von der Website des Herstellers flashen kann. Es gibt etwa ein Circuit-Python- und eine BadUSB-Firmware. Wie der große, aber teurere Bruder von Hak5 kann das Pendrive S3 verschiedene USB-Geräte simulieren, etwa Keyboard, Maus oder Netzwerk-Adapter. Mit den Erweiterungen von SuperWiFiDuck können so Tastatureingaben automatisiert, die

ThingPulse

LED gesteuert sowie Mausbewegungen programmiert werden. Das alles über ein Web-Interface und mit der Ducky-Scriptsprache. Scripts lassen sich entweder automatisch beim Einstecken ausführen oder bei Berührung des kapazitiven Buttons. Neben Pentesting (Sicherheitstests) lassen sich damit natürlich auch andere nützliche Sachen machen: etwa Rechner aufräumen und zurücksetzen nach Workshops oder Ähnliches.

Wünschen würden wir uns, dass ein paar GPIOs auf der Platine erreichbar wären und es ein Gehäuse gäbe, bei dem die Knöpfe BOOT und RST erreichbar bleiben. *—caw*

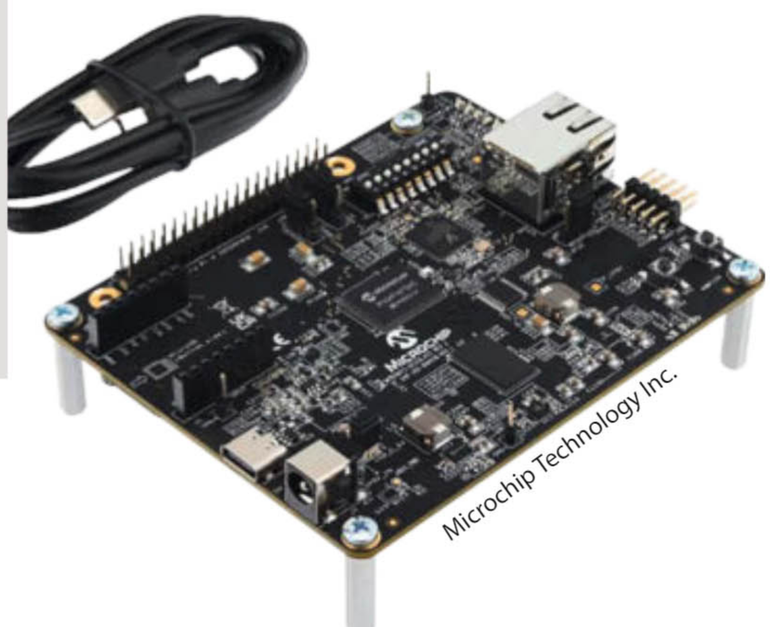
Hersteller	ThingPulse GmbH
URL	25 US-$ (+Versand+Steuer)
Preis	make-magazin.de/xjnm

PolarFire SoC Discovery Kit

RISC-V- und FPGA-Board für Einsteiger

Das Microchip-RISC-V-FPGA-Board basiert auf Microchips 5-Core, 667 MHz MPFS095T-1FCSG325E FPGA mit 95K Logikelementen und vier 64-Bit-RISC-V-Kernen, einem 1-Gigabyte-Hauptspeicher und einem Mikro-SD-Kartenslot. Über den SD-Kartenslot kann auf dem Board Linux installiert werden. Außerdem verfügt das PolarFire SoC Discovery Kit über Gigabit-Ethernet.

Die Ports werden durch drei UART-Anschlüsse über USB-Type-C, einen MIPI-Anschluss (Mobile Industry Processor Interface für Kameras etwa) sowie einen MicroBUS-Connector ergänzt, über den sich Boards aus dem MikroE-Click-Ökosystem anschließen lassen. Dazu kommen ein 7-Segment-Display-Anschluss und ein 40-Pin-Steckverbinder, der als „Raspberry Pi Connector“ bezeichnet wird.



Microchip Technology Inc.

Inwieweit das PolarFire SoC Discovery Kit mit Raspberry Pi Hardware kompatibel ist, wird nicht angegeben.

Das Board ist für Einsteiger in FPGA- und RISC-V-Entwicklungen gedacht. Deshalb steht als Unterstützung eine Sammlung an Webinaren zur Verfügung, welche die Handhabung und Entwicklung des Einplatinenrechners erläutern. Diese Watch-on-Demand-Webinare werden von verschiedenen Industriepartnern, immer mit Bezug auf den PolarFire SoC, gehalten.

Für die Programmierung ist ein FlashPro 5 direkt auf dem Board verbaut. Um loszulegen, muss das PolarFire SoC nur mit einem USB-auf-J4-Kabel mit dem Computer verbunden werden. Dann kann mit FlashPro Express bespielt werden.

Das Board ist 10,4 cm x 8,3 cm groß und direkt im Herstellershop erhältlich. Hier gibt es auch Links zu Designfiles und der offiziellen GitHub-Seite. Dort ist bereits das Referenzdesign-Kit hinterlegt. *—das*

Hersteller	Microchip Technology Inc.
URL	make-magazin.de/xjnm
Preis	123 €

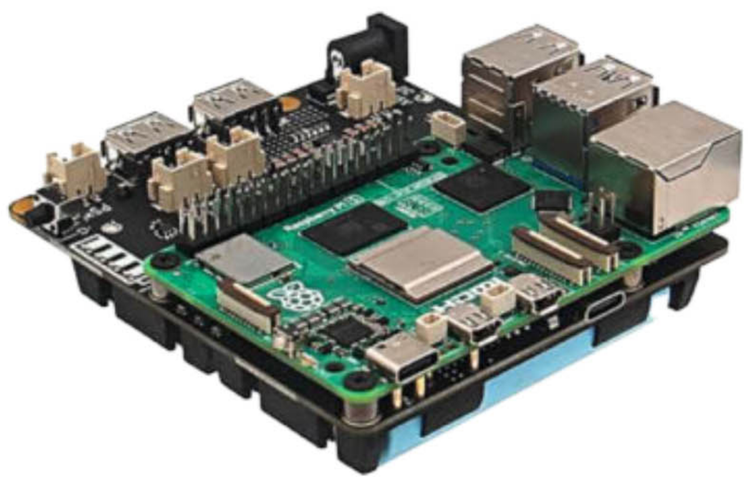
18650 Battery UPS HAT

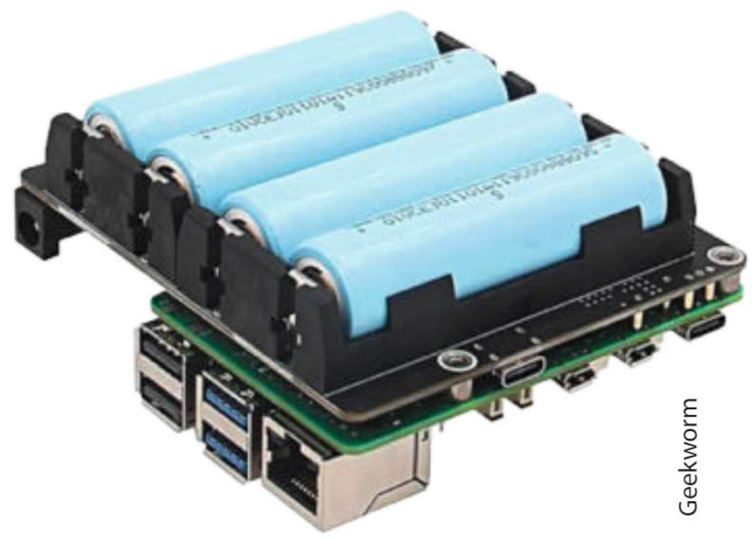
Akku-Stromversorgung mit viel Power für Raspi 5

Damit der Server auch bei Netzstromausfall weiter arbeitet und das Dateisystem konsistent bleibt, gibt es akkubetriebene Notstromversorgungen. Speziell für den recht leistungshungrigen Raspi 5 gibt es jetzt eine maßgeschneiderte unterbrechungsfreie Stromversorgung (USV).

Eine 98 × 85 mm große Adapterplatine namens Raspberry Pi 5 18650 Battery UPS HAT übernimmt die Energieversorgung des Raspi 5 unterbrechungsfrei, sobald die Spannung des per USB-C-Buchse angeschlossenen Netzteils ausfällt. Dazu besitzt sie an der Unterseite vier Halter für Standard-18650-Akkuzellen. Auf der Oberseite kann der Raspberry Pi 5 mittels Abstandshalter und Schrauben direkt angedockt werden. Die elektrischen Verbindungen zwischen HAT und Raspi (Stromversorgung sowie die Datenleitungen) entstehen dabei durch die auf der Platine angebrachten Federkontakt-Stifte (Pogo-Stifte). Eine zusätzliche Verkabelung ist nicht notwendig.

Verwendet man Akkuzellen mit 3400 mAh Kapazität, kann der Raspi bei Volllast (5A Stromaufnahme) theoretisch bis zu 10 Stunden versorgt werden. In der Praxis sollte man eher mit 7 bis 8 Stunden rechnen. Der Hersteller gibt einen Wirkungsgrad von 95 Prozent an, allerdings sollte man Akkus nicht bis zum letzten mA entladen.





Geekworm

Mit einfachen Python-Skripten kann man den Akku-Ladezustand sowie die Art der Stromversorgung (normal bzw. Akku-Versorgung) über den I²C-Bus abfragen. Auch ein automatischer, einstellbarer Shutdown bei entladenen Akkus ist einrichtbar. Die Wiki-Seiten des Herstellers stellen dazu alle Informationen und Programmbeispiele zur Verfügung.

Auf dem Board sitzen noch je zwei USB-A- sowie XH2,54-Buchsen, die 5,1V mit bis zu 5,5A für zusätzliche Geräte zur Verfügung stellen. Weiterhin können externe Ladegeräte bis 18 V angeschlossen werden, wenn das übliche Raspberry-Netzteil nicht zum Einsatz kommen soll. Der Ladevorgang erfolgt mit bis zu 3 A. Über diese USB-C-Buchse erfolgt auch die normale Stromversorgung des Einplatinencomputers. Weitere Bestandteile des Boards sind eine Ladezustandsanzeige mit vier LEDs sowie ein Power-Button, mit dem der Raspi ein- und ausgeschaltet werden kann. *—hgb*

Hersteller	Geekworm
URL	make-magazin.de/xjnm
Preis	43 US-$ / 70 € (Herstellershop/dt. Onlinehandel)

Arduino Alvik

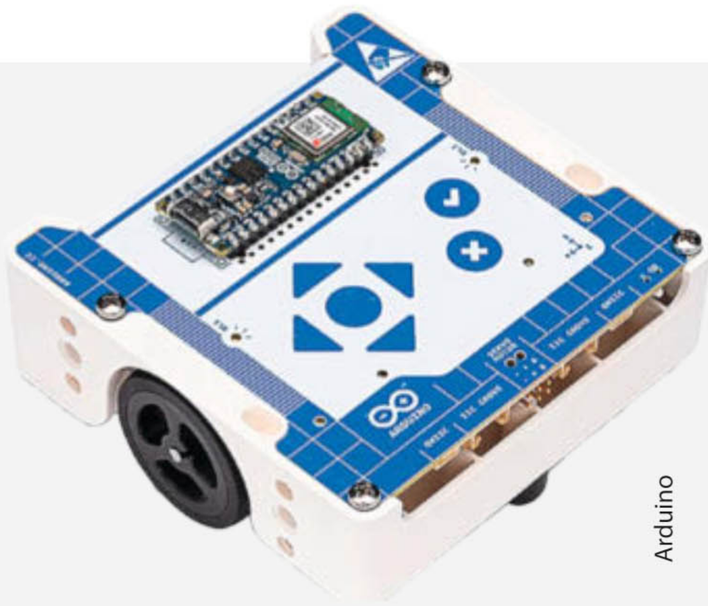
Arduino-Lernroboter für Kids

Der Arduino Alvik ist ein kleiner rollender Roboter, der über einen Arduino Nano ESP32 programmiert wird und über einen STM32 Arm Cortex-M4 als Rechenhirn verfügt. Alvik ist als Lernplattform für Kinder gedacht, die einen Einstieg in ESP32-Programmierung in Zusammenspiel mit Sensoren und Motoren ermöglicht.

Out of the box bringt Alvik verschiedene Sensoren mit. Dabei handelt es sich unter anderem um einen Time-of-Flight-Sensor zur Entfernungsmessung für bis zu 350 cm und 90° Sichtfeld, eine Kombination aus Beschleunigungssensor und 6-Achsen-Gyroskop und einen Farbsensor. Dazu kommen noch Motoren für die Räder des Roboters, ein Line Follower Array und Touch-Buttons auf der Oberseite.

Zusätzlich zu diesen fest verbauten Sensoren lassen sich über Steckverbinder weitere Module anschließen. Es stehen auch Ports für I²C und zusätzliche Servomotoren zur Verfügung. Über zu Klemmbausteinen kompatible Steckverbinder im Gehäuse können für diese externen Sensoren auch Halterungen an den Alvik angebracht werden. Neben diesen Bauteilen verfügt Alvik noch über Bluetooth-LE-5.0- und WLAN-2,4-GHz-Konnektivität. Angetrieben wird die Lernplattform über einen 18650-Li-Ion-Akku. Dieser kann auch einfach gewechselt werden.

Die Lehrmaterialien auf der Website von Arduino bilden klassische Anfangsprojekte ab: Nutzung des Line Follower Arrays, um einem aufgemalten Weg zu folgen, oder Einsetzen des Time-of-Flight-Sensors, um vor Hindernissen anzuhalten und diesen auszuweichen. Mit dem Gyroskop und dem Beschleunigungssensor kann die Ausrichtung des Roboters abgefragt und nachjustiert werden. Generell drehen sich Alvik-Projekte darum, den Roboter bestimmte Wege fahren bzw. ihn selbstständig Wege finden zu lassen, Hindernissen auszuweichen oder Gegenstände im Weg zu erkennen.



Arduino

Der Roboter lässt sich aktuell mit MicroPython- und Arduino-Sprache (C/C++) programmieren. Eine visuelle Blockprogrammierung soll demnächst hinzugefügt werden. Für den M4-Core steht eine API in Arduino und Micropython zur Verfügung. Einen Einstieg in Alvik bietet aktuell die Docs-Seite von Arduino. Dort sind auch schon Handbücher etc. für den Roboter verlinkt. *—das*

Hersteller	Arduino
URL	make-magazin.de/xjnm
Preis	130 €

IMPRESSUM

Redaktion

Make: Magazin
Postfach 61 04 07, 30604 Hannover
Karl-Wiechert-Allee 10, 30625 Hannover
Telefon: 05 11/53 52-300
Telefax: 05 11/53 52-417
Internet: www.make-magazin.de

Leserbriefe und Fragen zum Heft: info@make-magazin.de

Die E-Mail-Adressen der Redakteure haben die Form xx@make-magazin.de oder xxx@make-magazin.de. Setzen Sie statt „xx“ oder „xxx“ bitte das Redakteurs-Kürzel ein. Die Kürzel finden Sie am Ende der Artikel und hier im Impressum.

Chefredakteur: Daniel Bachfeld (dab)
(verantwortlich für den Textteil)

Redaktion: Heinz Behling (hgb), Johannes Börnsen (jom), Ákos Fodor (akf), Marcus Hansson (mch), Daniel Schwabe (das), Dunia Selman (dus, Social Media), Carsten Wartmann (caw)

Mitarbeiter dieser Ausgabe: Jahn Moritz Behnken, Benno Lottenbach, Uwe Magnus, Matthias Mett, Roman Radtke, Uwe Rohne, Florian Schäffer, Maik Schmidt, Gerhard Völkl, Dirk Wahl

Assistenz: Susanne Cölle (suc), Martin Triadan (mat)

Layout und Satz: Steffi Martens, Lisa Reich, Nicole Wesche, Heise Medienwerk GmbH & Co. KG

Korrektorat: Dörte Bluhm, Lara Bögner, Marei Stade, Christiane Tümmler, Heise Medienwerk GmbH & Co. KG

Titel: Lisa Reich, Nicole Wesche

Fotografie und Titelbild: Andreas Wodrich

Digitale Produktion: Melanie Becker, Thomas Kaltschmidt, Pascal Wissner

Hergestellt und produziert mit Xpublisher:
www.xpublisher.com

Verlag

Maker Media GmbH
Postfach 61 04 07, 30604 Hannover
Karl-Wiechert-Allee 10, 30625 Hannover
Telefon: 05 11/53 52-0
Telefax: 05 11/53 52-129
Internet: www.make-magazin.de

Herausgeber: Christian Heise, Ansgar Heise

Geschäftsführung: Ansgar Heise, Beate Gerold

Anzeigenleitung: Daniel Rohlfing (-844)
(verantwortlich für den Anzeigenteil),
mediadaten.heise.de/produkte/print/das-magazin-fuer-innovation

Leiter Vertrieb und Marketing: André Lux (-299)

Service Sonderdrucke: Julia Conrades (-156)

Druck: Dierichs Druck + Media GmbH & Co.KG,
Frankfurter Str. 168, 34121 Kassel

Vertrieb Einzelverkauf:
DMV DER MEDIENVERTRIEB GmbH & Co. KG
Meßberg 1
20086 Hamburg
Telefon: +49 (0)40 3019 1800
Telefax: +49 (0)40 3019 1815
E-Mail: info@dermedienvertrieb.de
Internet: dermedienvertrieb.de

Einzelpreis: 13,50 €; Österreich 14,90 €; Schweiz 26.50 CHF; Benelux 15,90 €

Abonnement-Preise: Das Jahresabo (7 Ausgaben) kostet inkl. Versandkosten: Inland 80,50 €; Österreich 88,90 €; Schweiz 123.90 CHF; Europa 95,20 €; restl. Ausland 100,80 €

Das Make-Plus-Abonnement (inkl. Zugriff auf die App, Heise Magazine sowie das Make-Artikel-Archiv) kostet pro Jahr 6,30 € Aufpreis.

Make: Nächste Ausgabe erscheint am 20. September 2024

Abo-Service:

Bestellungen, Adressänderungen, Lieferprobleme usw.:
Maker Media GmbH
Leserservice
Postfach 24 69
49014 Osnabrück
E-Mail: leserservice@make-magazin.de
Telefon: 0541/80009-125
Telefax: 0541/80009-122

ISSN 2364-2548

Nachgefragt

Wir bauen im Heft ein (fast echtes) Orakel. Welche Frage würdest Du ihm stellen?

Jahn Moritz Behnken
Leer, macht den Taupunktlüfter auf Seite 36 Smarthome-fähig.

Aus den unzähligen Projektideen und Bauteilen die richtige Wahl zu treffen, fällt nicht immer leicht. Deshalb möchte ich vom Orakel wissen: Welcher Sensor wird mein nächstes Projekt revolutionieren?

Roman Radtke
Künzelsau, tuned auf Seite 74 seine elektrische Enduro.

Ich bin Fan des Films „Pulp Fiction“. Deshalb will ich wissen: „Was war WIRKLICH in Marsellus Wallace' Koffer?“

Uwe Rohne
Sehnde, baut auf Seite 8 einen Preisrahmen für Tibber.

Liebes Orakel, ich bin leidenschaftlicher Segler. Leider bin ich nicht so schnell wie die Skipper auf vergleichbaren Booten. Was muss ich anders machen?

Gerhard Völkl
Pentling, programmiert ein Spiel auf Seite 112.

Wo und wann kann ich einen Termin im nächsten Holodeck buchen?

Inserentenverzeichnis

Ein Teil dieser Ausgabe enthält Beilagen der DIMABAY GmbH, München.

Ein Veranstaltungsformat der Make:
make-magazin.de

10 JAHRE

Maker Faire®

HANNOVER

17.–18. Aug.

Hannover Congress Centrum

maker-faire.de

UNTER DER SCHIRMHERRSCHAFT VON
BETTINA STARK-WATZINGER MDB

Bundesministerium
für Bildung
und Forschung

sponsored by

BASIC SOLAR

Hochmobiler Businessbegleiter der Premiumklasse

TUXEDO InfinityBook Pro 15 - Gen9

AMD oder
Intel CPU

Mobilität

Matt-schwarzes
oder silbergraues
Aluminiumgehäuse

Bis zu 8 TB SSD und 96 GB RAM
2x PCIe 4.0 SSD | 2x DDR5-5600-RAM

Akkulaufzeit

Linux kompatibel

Bis zu 5 Jahre Garantie

Sofort einsatzbereit

TUXEDO

tuxe.do/make0424

Gefertigt in Deutschland

Deutscher Datenschutz

Deutscher Tech Support