G3078 E (D) € 14,90 | CHF 27,00 | (A, B, L) € 16,00

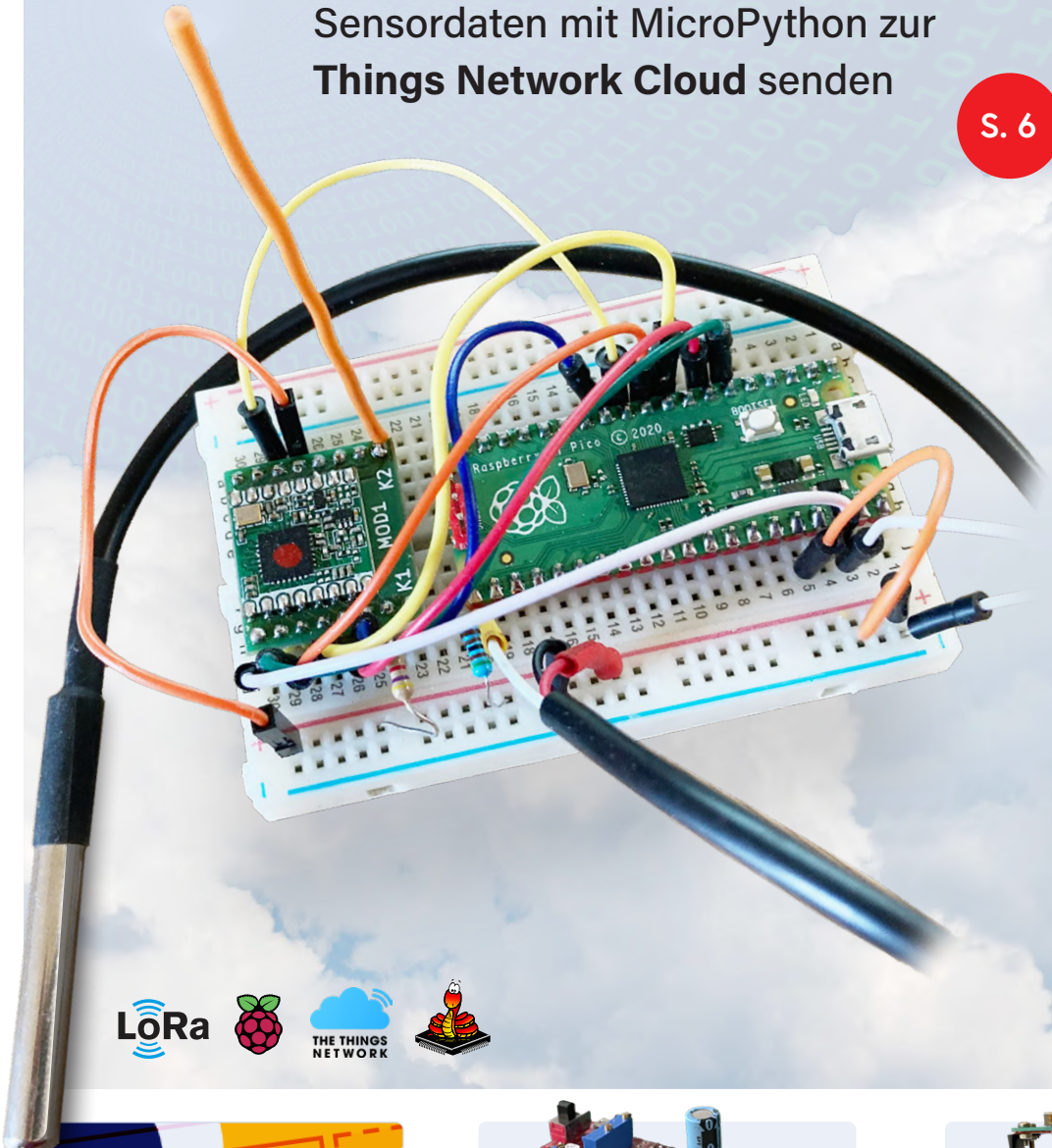
elektor MAG
design › share › sell
580
* SEIT 1961 *
JULI/AUGUST 2021
ELEKTORMAGAZINE.DE

60 years young

LoRa mit dem Raspberry Pi Pico

Sensordaten mit MicroPython zur **Things Network Cloud** senden

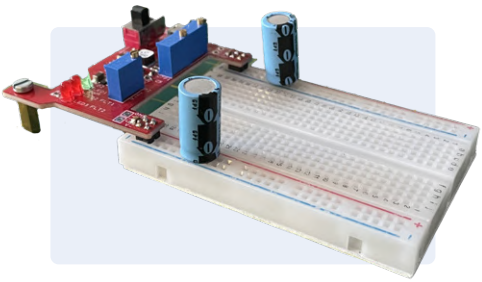
S. 6



In dieser Ausgabe

und vieles mehr!

Was ist RISC-V?
Warum die Industrie einen Prozessorkern so spannend findet
S. 17



Vielseitige Spannungsversorgung für Breadboards – Positive und negative 5-V-Ausgangsspannungen vom USB
S. 26

Magnetische Levitation auf die einfache Art
Schweben ohne Mikrocontroller!
S. 34

10008
4 198630 314904

Treten Sie jetzt der Elektor Community bei!

Jetzt elektor GOLD membership Mitglied werden!

- Komplettes Webarchiv ab 1970
- 6x Elektor Doppelheft (Print)
- 9x Digital (PDF) inkl. Elektor Industry
- 10% Rabatt im Online-Shop und exklusive Angebote
- Elektor Jahrgangs-DVD
- Mit Tausenden von Mitgliedern des Online-Labors gemeinsam entwickeln mit Zugang zu über 1.000 Gerber-Dateien und direktem Kontakt zu unseren Experten!
- Veröffentlichen Sie Ihr eigenes Projekt oder verkaufen Sie direkt über unseren Shop!

Auch erhältlich

Die digitale elektor GREEN membership Mitgliedschaft!

- Zugang zu unserem Webarchiv
- 10% Rabatt in unserem Online-Shop
- 6x Elektor Doppelheft (PDF)
- Exklusive Angebote
- Zugang zu über 1.000 Gerber-Dateien

www.elektor.de/mitglied

elektor design › share › sell

IMPRESSUM

52. Jahrgang, Nr. 580
Juli/August 2021
ISSN 0932-5468

Erscheinungsweise: 9x jährlich
(6x Elektor-Doppelheft + 3x Elektor Industry Magazin)

Verlag
Elektor Verlag GmbH
Kackertstraße 10
52072 Aachen
Tel. 0241 95509190

Technische Fragen bitten wir per E-Mail an redaktion@elektor.de zu richten.

Hauptsitz des Verlags
Elektor International Media
Postbus 11, 6114 ZG Susteren
Niederlande

Anzeigen
Raoul Morreau (Leitung)
Mobil: +31 6 440 399 07
E-Mail: raoul.morreau@elektor.com

Büsra Kas
Tel. 0241 95509178
E-Mail: busra.kas@elektor.com

Es gilt die Anzeigenpreisliste ab 01.01.2021.

Distribution
IPS Pressevertrieb GmbH
Postfach 12 11, 53334 Meckenheim
Tel. 02225 88010
Fax 02225 8801199

Druck
Pijper Media, Groningen (NL)

EDITORIAL

Von Jens Nickel

Chefredakteur ElektorMag

Kochshow à la Elektor

Die Erfahrung haben Sie sicher auch schon gemacht: Sie haben eine tolle Idee für eine eigene Entwicklung, und entdecken, dass jemand anders bereits viel Vorarbeit geleistet hat. Meist lohnt es sich nicht, das Rad nochmal neu zu erfinden. Stattdessen können Sie auf fertige Software- und Hardwaremodule zurückgreifen, und diese (nach etwas Anpassung) miteinander verbinden, um eine neue Applikation zum Laufen zu bekommen. Für das Hauptprojekt dieser Ausgabe ist mein Kollege Mathias Claußen diesen Weg gegangen. Sein LoRa-Sensorknoten kann im Feld Messwerte sammeln und über weite Strecken weitergeben, für die Erfassung von Umweltdaten, aber auch die Hausautomatisierung. Als Zutaten wurden verwendet: Ein kleiner Raspberry Pi Pico mit angeschlossenem Temperatursensor, ein LoRa-Modul von SeeedStudio, eine LoRa-MicroPython-Library, The Things Network als Cloud-Plattform und schließlich noch das Automatisierungsframework Node-RED, das sich auf einem Raspberry Pi oder PC installieren lässt. Nach Elektor'scher Art präsentieren wir Ihnen nicht nur das Endergebnis, sondern eine Art Kochshow, damit sie wissen, was vor sich geht, und das Rezept nach Ihren eigenen Wünschen anpassen oder verfeinern können. Serviert wird das Ganze auf einem Breadboard, doch wer lieber eine Platine möchte, findet auch hierzu die nötigen Daten.
Es ist ebenfalls Elektor'sche Art, dass wir nicht nur opulente Menüs, sondern auch leichtere Kost im Angebot haben. Wer beim Thema MicroPython noch Einsteiger ist, dem kann ich den Artikel auf Seite 92 empfehlen - für erste Gehversuche genügt ein kleines ESP32-Board. Und ab Seite 30 wird gezeigt, wie Sie den oben genannten Pico verwenden können, um eine LED von einem Smartphone aus zu schalten.
Bleiben Sie dran - und bleiben Sie gesund!

ELEKTOR WIRD 60: ES KOMMT NOCH MEHR!

Der Elektor Verlag feiert 60 Jahre Elektronik-Innovation mit einigen spannenden Sonderprojekten. Wir arbeiten an einem Jubiläumsbuch, einem Film, einem Live-Event (World Ethical Electronics Forum) und dem „Elektor-Mobilheimlabor". Wir wetten, Sie sind jetzt schon neugierig. Bleiben Sie dran und erfahren Sie mehr in den nächsten Wochen. Sie haben Ideen für weitere besondere Projekte? Schicken Sie mir Ihre Gedanken: denise.bodrone@elektor.com!

Denise Bodrone, „Elektor 60" Festkomitee-Koordinatorin

Unser Team

Elektor ist Mitglied des 1929 gegründeten VDZ (Verband Deutscher Zeitschriftenverleger), der „die gemeinsamen Interessen von 500 deutschen Consumer- und B2B-Verlagen vertritt."

Elektor ist Mitglied von FIPP, einer Organisation, die „über fast 100 Jahre gewachsen ist und Medienbesitzer und Content-Ersteller aus der ganzen Welt umfasst".

INHALT

52. Jahrgang
Elektor Nr. 580 - Juli/August 2021

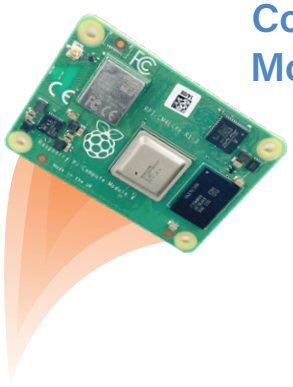
Portables Feinstaubmessgerät für **2,5-µm-Partikel**

85

Rubriken

Hintergrund

Review:
Raspberry Pi Compute Module 4



80

Solaranlage für Mähroboter

105

Projekte

Vorschau

Elektor September/Oktober 2021
Das nächste Heft ist wie immer randvoll gefüllt mit Schaltungsprojekten, Grundlagen sowie Tipps und Tricks für Elektroniker.

Aus dem Inhalt:
- Elektronische Last für DC und AC
- DIY-Kamerasystem für Raspberry Pi
- ESP32-Thermostat
- Magnetische Levitation: der digitale Weg
- Elektronischer Kompass mit GY-271
- MicroPython auf dem ESP32: LED-Matrix-Displays
- Displays in Raspberry-Pi-Projekten
- Nvidia Jetson Image Processing für Einsteiger

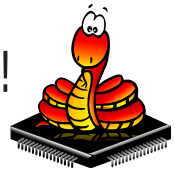
Und vieles mehr!

Elektor September/Oktober 2021 erscheint am 2. September 2021.
Änderungen vorbehalten.

elektor MAG
design › share › sell

LoRa mit dem Raspberry Pi Pico

Viel Spaß mit MicroPython!



Von **Mathias Claußen** (Elektor)

Ein Raspberry Pi Pico mit MicroPython ist eine einsteigerfreundliche Art zu programmieren. Verbunden mit einem LoRa-Modul RFM95 von SeeedStudio und einem Temperatursensor DS18B20 lässt sich mit MicroPython schnell ein LoRaWAN-Knoten auf einem Breadboard erstellen. Und da der „Feldeinsatz“ der Schaltung auf einem Breadboard etwas instabil sein kann, haben wir direkt ein echtes Platinenlayout für Sie entworfen.

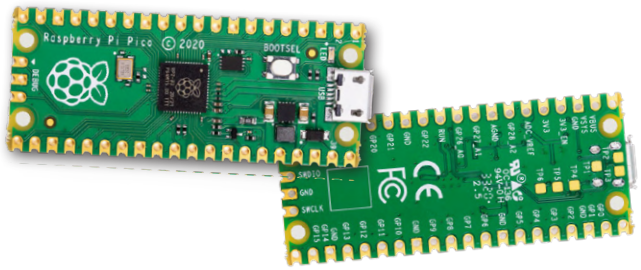


Bild 1. Vorder- und Rückseite des Raspberry Pi Pico.

Bild 2. Das RFM95-Modul.

Der Raspberry Pi Pico (**Bild 1**) ist ein neues Mikrocontroller-Board, das mit dem Controller RP2040 ausgestattet ist. Dieser Chip wurde als erstes eigenes „Silizium“ von der Raspberry Pi Foundation entworfen und entwickelt. Nachdem ich all die Neuigkeiten rund um das Board gehört und gesehen hatte – inklusive der Zweifel und der Faszination, die von ihm ausgeht - dachte ich mir, dass es an der Zeit wäre, ein paar Projekte damit zu starten. Das Board bietet zwar kein WLAN, kann aber mit einem Satz Batterien in portablen Anwendungen betrieben werden. Warum also nicht den Raspberry Pi Pico mit einem LoRa-Modem kombinieren? Das HOPERF RFM95 (**Bild 2**) ist ein preisgünstiges und gut unterstütztes Modul, das bereits in früheren Elektor-Projekten wie in *LoRaWAN - ein einfacher Einstieg* [1] verwendet wurde. Während aber in diesem Artikel ein STM32-BluePill-Controller verwendet wurde, werden wir dieses Mal mit dem Raspberry Pi Pico arbeiten.

Was wir bauen wollen, ist ein einfaches Temperaturmessgerät mit dem beliebten DS18B20-Sensor, das Daten an ein LoRaWAN-Gateway sendet, welches seinerseits die Daten an die Cloud von The Things Network weiterleitet. Von dort werden wir die Daten mit einem (klassischen) Raspberry Pi abgreifen, auf dem die (Heim-) Automatisierungsplattform Node-RED läuft.

Wir werden den temperatur-erfassenden LoRa-Knoten auf einem Breadboard aufbauen, aber wenn Sie einen dauerhaften Aufbau bevorzugen, finden Sie in diesem Artikel auch einen KiCad-Entwurf für eine richtige Platine.

In **Bild 3** sehen Sie den Datenfluss vom Raspberry Pi Pico über das LoRaWAN zum Server von The Things Network. Wie Sie sehen können, befindet sich ein Gateway in der Mitte, das die Funkdaten des LoRa-Knotens in etwas übersetzt, das über das Internet zu The Things Network transportiert werden kann. Die an das The Things Network gelieferten Daten sollen verarbeitet werden, da wir an der übertragenen Temperatur interessiert sind. Dann müssen wir diese Daten von der Cloud abgreifen und sie auf einer kleinen Webseite präsentieren. **Bild 4** zeigt den Datenfluss vom The Things Network zu einem Raspberry Pi, der eine Webseite mit der aktuell übertragenen Temperatur und auch ein Diagramm mit den zuletzt übertragenen Werten generiert.

Die Software, die verwendet wird, um die Daten von The Things

Network zu erfassen und eine Webseite zu generieren, ist Node-RED, ein Werkzeug, mit dem Datenflüsse grafisch angeordnet werden können und das verschiedene Daten verarbeitet. Die Darstellung erfolgt einfach in Ihrem Lieblingsbrowser. Sie müssen nur den Node-RED-Server auf dem Raspberry Pi installieren und den Browser öffnen, um loszulegen. Node-RED wurde bereits in der Vergangenheit in Elektor-Projekten eingesetzt und ermöglicht einen schnellen Einstieg in die Datenver- und -bearbeitung. Ein Node-RED Einsteigerbuch finden Sie im Elektor-Store (siehe **Passende Produkte** für weitere Informationen).
Die **Stückliste** zeigt, dass Sie für dieses Projekt nur wenige, einfach erhältliche und preiswerte Zutaten benötigen. Überzeugen Sie sich, dass ein LoRaWAN-Gateway in Reichweite ist, das Ihre Daten an The Things Network weiterleiten kann. Sollte dies nicht der Fall sein, können Sie auch ein eigenes Gateway erstellen, das zusätzlich eine bessere LoRaWAN-Abdeckung schafft. In [3] ist eine All-in-One-Lösung gezeigt, die Sie für nicht allzuviel Geld von der Stange kaufen können. Alternativ können Sie mit [4] Ihr eigenes Raspberry-Pi-basiertes Gateway bauen.

Breadboard first!

Zur Demonstration der grundlegenden Schaltung reicht erst einmal der Aufbau auf einem Breadboard, das einfach zu verdrahten ist (**Bild 5**) und schnelles Testen ermöglicht. Um das LoRa-Modem mit dem Raspberry Pi Pico zu verbinden, benötigen wir die vier typischen Anschlüsse der SPI-Schnittstelle: MOSI, MISO, SCK und CS (data in, data out, clock und chip select). Über diese Verbindungen vollzieht sich der Datenaustausch mit dem LoRa-Modem. Darüber hinaus werden RESET und DIO0 zur Steuerung des Modems benötigt. DIO1 bis DIO3 sind angeschlossen und werden von einigen anderen LoRa-Bibliotheken benötigt, etwa der LMIC-Bibliothek (wenn wir den Raspberry Pi Pico in C/C++ programmieren wollen). Wir können einen der SPI-Ports wählen, die der Raspberry Pi Pico bietet, und ihn mit dem RFM95-LoRa-Modem verbinden. Außerdem müssen wir nur RESET und DIO0 vom Modul mit einem der GPIOs des Pico verbinden. Für ein voll funktionsfähiges LoRa-Modem muss natürlich auch eine Antenne angeschlossen werden.
Als Antenne können wir einen einfachen Kupferdraht mit einem Durchmesser von 1 mm verwenden. Die benötigte Länge berechnet sich als λ/4-Antenne für 868 MHz, dem Arbeitsbereich des LoRa-Moduls, wie folgt: $\lambda/4 = (c_0/868\ \text{MHz})/4 = (299792458\ \text{m/s})/(868000000\ 1/\text{s}*4) = 0{,}08635\ \text{m} = 8{,}635\ \text{cm}$. Diese Länge gilt, wenn sich die Welle im Vakuum ausbreiten würde, aber in Kupfer ist die Geschwindigkeit der Welle geringer. Deshalb muss ein Verkürzungsfaktor von etwa 0,95 berücksichtigt werden. Damit erhält man schließlich eine Drahtlänge von ungefähr 8,2 cm.
Obwohl es nicht ausdrücklich empfohlen wird, hat sich in der Vergangenheit gezeigt, dass ein Pull-up-Widerstand an der Reset-Leitung für einen stabileren Betrieb erforderlich ist, da die RESET-Leitung möglicherweise Rauschen von den „fliegenden" Drähten aufnimmt und das Modul unerwartet zurücksetzt. Wenn Sie an Ihrem eigenen LoRa-Board arbeiten, sollten Sie einen Pull-up-Widerstand im Layout vorsehen und ihn später nur, wenn es erforderlich scheint, mit dem

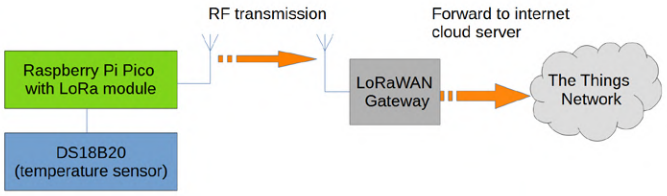


Bild 3. Datenfluss vom Raspberry Pi zum The Things Network.

Bild 4. Datenfluss vom The Things Network zur Benutzerwebseite.

Stückliste

- Raspberry Pi Pico*
- RFM95*
- Breadboard-Set*
- 9 Jumperdrähte Stecker-Stecker
- Breakout Board für RFM95
- 2x20-polige Stiftleiste
- 2x8-polige Stiftleiste
- Temperatursensor DS18B20
- 10 cm Kupferdraht (1 mm Durchmesser)

Die mit * gekennzeichneten Teile können im Elektor-Shop bestellt werden (siehe **Passende Produkte**). Für das Breakout-Board stehen Gerber-Dateien [15] zur Verarbeitung durch Ihren bevorzugten Platinenservice zur Verfügung.

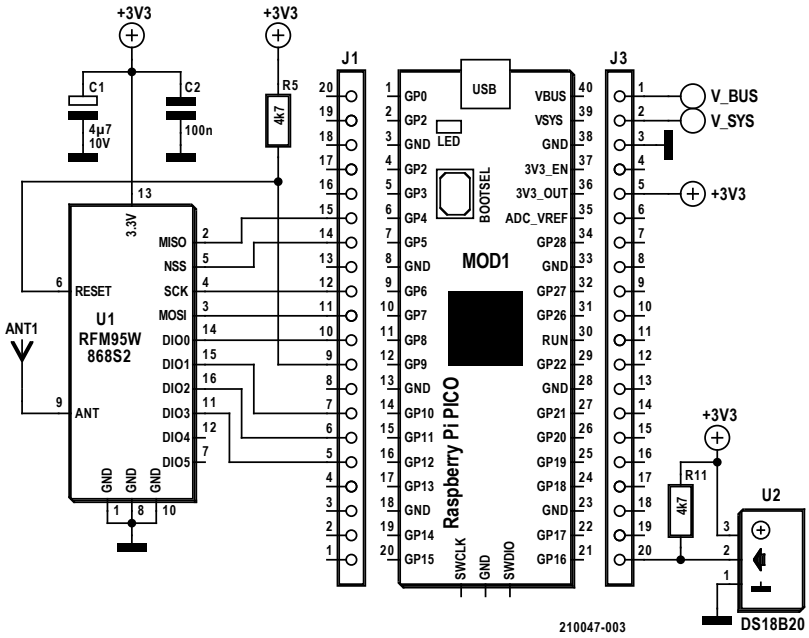


Bild 5. So wird die Elektronik miteinander verbunden.

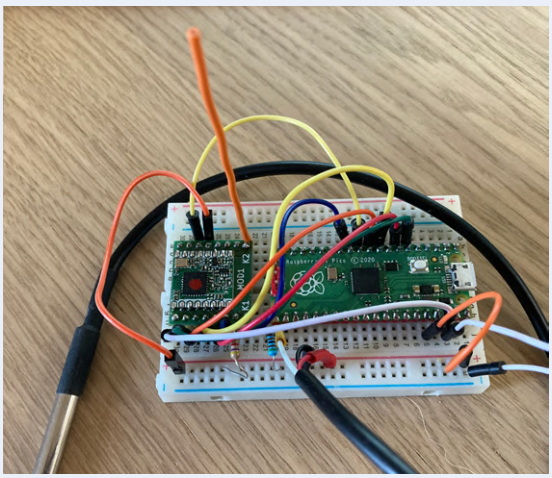


Bild 6. Der Aufbau auf einem Breadboard.

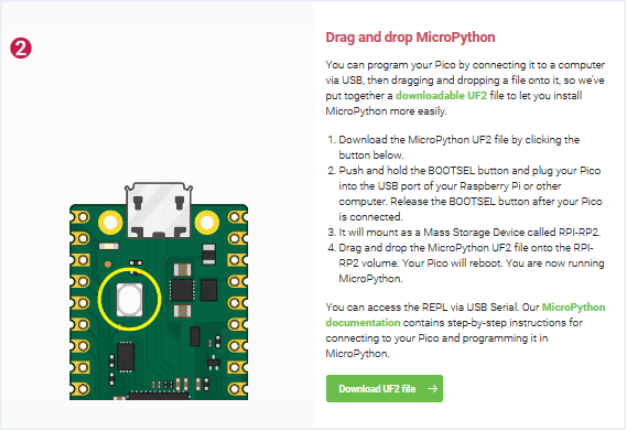


Bild 7. MicroPython-Download für den Raspberry Pi Pico.

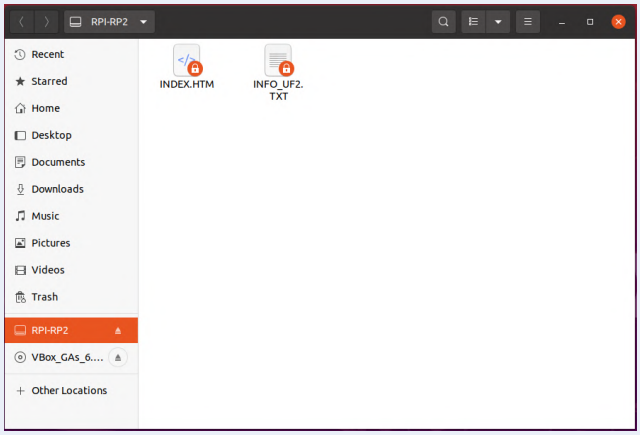


Bild 8. Der Raspberry Pi Pico im Bootloader-Modus.

Bild 9. Thonny ist in der Version 3.3.4 installiert.

Handlötkolben auf die Platine löten.

Der Temperatursensor ist über eine One-Wire-Verbindung angeschlossen. Wie der Name One-Wire schon sagt, benötigt die Verbindung nur einen GPIO-Pin für den Datenaustausch. Das Protokoll kann als reine Softwarelösung implementiert werden, so dass jeder GPIO-Pin zur Verwendung geeignet ist. Beim One-Wire-Bus *muss* ein Pull-Up-Widerstand vorhanden sein, je nach verwendetem Controller ein vom GPIO-Block in der MCU bereitgestellter interner oder ein externer Widerstand. Für eine zuverlässige Datenübertragung sollte man bevorzugt einen externen Pull-up-Widerstand einsetzen.

Wenn die gesamte Hardware eingerichtet ist, sieht der Aufbau so aus wie in **Bild 6**. Nun kann mit der Programmierung begonnen werden.

MicroPython und LoRa

Da wir bereits in früheren Projekten C/C++ für LoRa verwendet haben, wollen wir diesmal mit MicroPython arbeiten. Da der Raspberry Pi Pico ein gut geeignetes Board für Lernen und Ausbildung ist, macht es den Einsatz von MicroPython noch ein wenig einfacher. Wenn Sie neu im Geschäft mit MicroPython auf dem Raspberry Pi Pico sind, werfen Sie doch einen Blick in das (englischsprachige) Buch von Gareth Halfacree und Ben Everard *Get Started with MicroPython on Raspberry Pi Pico* (Raspberry Pi Foundation, 2021). Wenn Sie ein gedrucktes Exemplar möchten, besuchen Sie den Elektor-Shop [5], aber Sie können auch ein kostenloses elektronisches Exemplar von [6] herunterladen.

Die Kombination von LoRa und MicroPython gab es bereits auf anderen Plattformen, sie bringt aber einige zusätzliche Herausforderungen. MicroPython ist eine Interpreter-Sprache, wie es früher BASIC auf dem C64 oder anderen Heimcomputern war. Dies ermöglicht zwar einen einfachen Zugriff, führt aber zu einem gewissen CPU-Overhead bei der Ausführung und damit zu einigen Einschränkungen.

Die Bibliothek, die wir verwenden werden, ist eine leicht modifizierte Version von uLora von fantasticdonkey, zu finden auf GitHub unter [7]. Diese Bibliothek ist ein Fork der TinyLoRa-Bibliothek für CircuitPython von Adafruit. Die oben genannten Einschränkungen sind, dass wir nur Daten senden, aber nichts vom LoRaWAN zurückerhalten können. Mit dieser Einschränkung mussten wir auch APB (activation by personalization) für die Authentifizierung verwenden, was bedeutet, dass der Network Session Key und der Application Session Key fest in unserem Code gespeichert sind.

Da in diesem Artikel nur gezeigt werden soll, dass und wie man LoRa mit MicroPython auf dem Raspberry Pi Pico betreiben kann, ist das Beispiel nicht perfekt, aber es kann schnell realisiert und einsatzfähig gemacht werden. Die benötigten modifizierten Dateien können von der Elektor-GitHub-Seite [12] heruntergeladen werden. Denken Sie daran: Auch wenn die Software korrekt zu funktionieren scheint, könnte mangelnde Stabilität ein Problem sein, das irgendwann behoben werden muss.

MicroPython auf dem Raspberry Pi Pico installieren

Der Raspberry Pi Pico wird mit einem Bootloader ausgeliefert, der es einfach macht, die laufende Firmware auf dem Controller auszutauschen. Zunächst besorgt man sich die neuste Version von MicroPython, die unter [8] erhältlich ist (**Bild 7**), und lädt die UF2-Datei von

der Webseite herunter.
Um mit dem Raspberry Pi Pico in den Bootloader-Modus zu wechseln, trennen Sie das Board zunächst vom Computer, drücken die BOOTSEL-Taste und schließen es dabei wieder an Ihren Computer an. Der Raspberry Pi Pico sollte nun ein neues Massenspeichergerät angezeigt werden, wie in **Bild 8** zu sehen. Ziehen Sie einfach die heruntergeladene UF2-Datei auf dieses Laufwerk und schon wird die aktuelle Version von MicroPython installiert.
Nach einem Neustart sind Sie oder besser gesagt Ihr Raspberry Pi Pico startklar. Für die Softwareentwicklung wäre es von Vorteil, einen Editor oder eine IDE zu haben. Der nächste Schritt ist deshalb die Einrichtung von Thonny als Python-IDE.

Thonny einrichten

Die Installation der Thonny-IDE auf einem aktuellen Ubuntu 20.04 oder auch Windows ist mit wenigen Klicks erledigt. Für Windows können Sie einen Installer unter [9] herunterladen. In Ubuntu tippen Sie

```
wget -q -O - https://github.com/thonny/thonny/
releases/download/v3.3.4/thonny-3.3.4.bash
```

ins Terminal, um den Installer für Version 3.3.4 mit Unterstützung für den Raspberry Pi Pico zu erhalten. Nachdem die Datei heruntergeladen wurde, können Sie sie mit `bash thonny-3.3.4.bash` ausführen.
Wenn die Installation erfolgreich war, haben Sie die Thonny-IDE in der Version 3.3.4 installiert, wie Sie in **Bild 9** sehen können.
Nach dem ersten Start muss der Raspberry Pi Pico konfiguriert werden. Dies geschieht über das Menü mit *Tools → Options* und öffnet den Konfigurationsdialog (**Bild 10**). Nun sind alle Tools und die Hardware bereit, so dass wir ein paar Zeilen Code schreiben können.

Einrichten des Codes

LoRa kommuniziert in einem ISM-Frequenzband. Je nachdem, in welcher Region der Welt Sie sich befinden, kann das ein Band im Bereich 433 MHz, 868 MHz oder 915 MHz sein. In Europa wird vor allem das 868-Mhz-Band verwendet, in den USA ist es 915 MHz. Wir verwenden hier also das 868-Mhz-Frequenzband, aber wenn Sie sich irgendwo sonst auf diesem Globus befinden, stellen Sie sicher, dass Sie im ISM-Band Ihrer Region kommunizieren. Da uLoRa für die Verwendung auf dem Raspberry Pi Pico gepatcht ist, müssen Sie *ulora.py, ttn_eu.py, ulora_encryption.py* und *lora.py* über die Thonny-IDE öffnen und auf Ihrem Raspberry Pi Pico speichern. Falls Sie sich nicht in Europa befinden, müssen Sie statt *ttn_eu.py* die für Ihre Region geeignete *ttn_xx.py* auf Ihrem Board speichern. Damit werden dem Board nicht nur die benötigten Bibliotheken zur Verfügung gestellt, sondern auch eine grundlegende Beispielanwendung, die wir jetzt erklären werden. In *lora.py* befindet sich das Hauptprogramm für dieses Projekt, dessen Ablauf in **Bild 11** dargestellt ist. Nach der Initialisierung wird nach einem

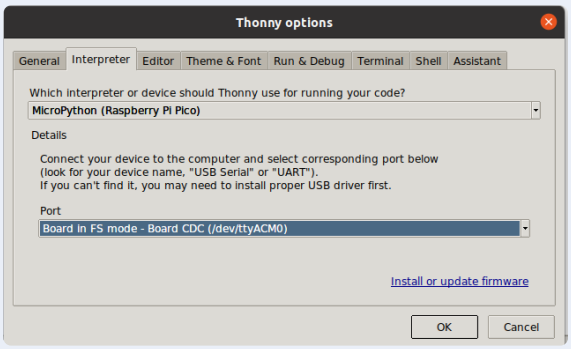


Bild 10. Thonny-Einstellungen für den Raspberry Pi Pico.

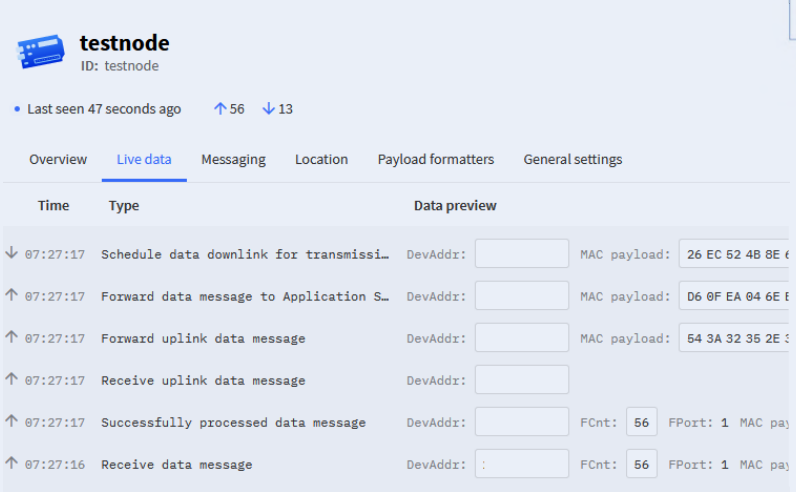


Bild 12. Empfangene Daten in der TTN-Konsole.

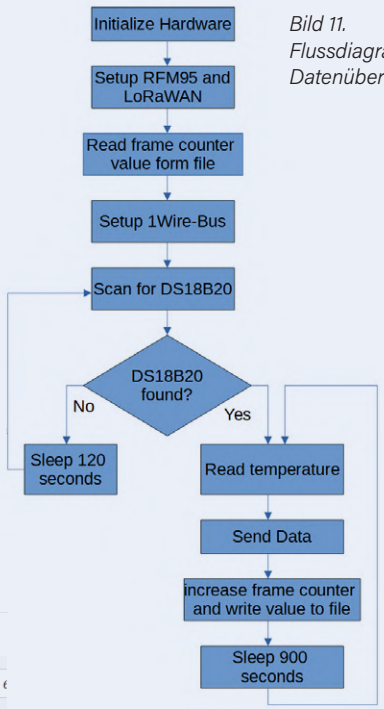


Bild 11. Flussdiagramm der Datenübertragung.

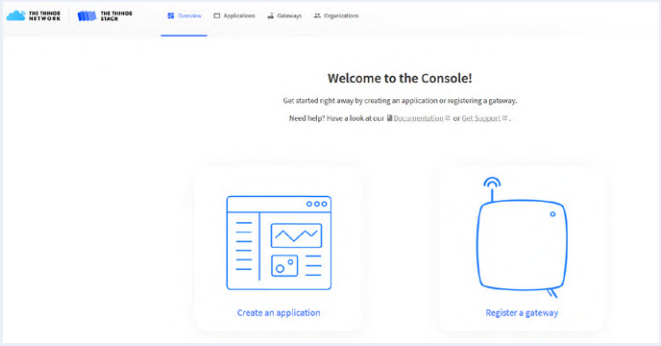


Bild 13. Erstellen einer neuen Anwendung im TTN.

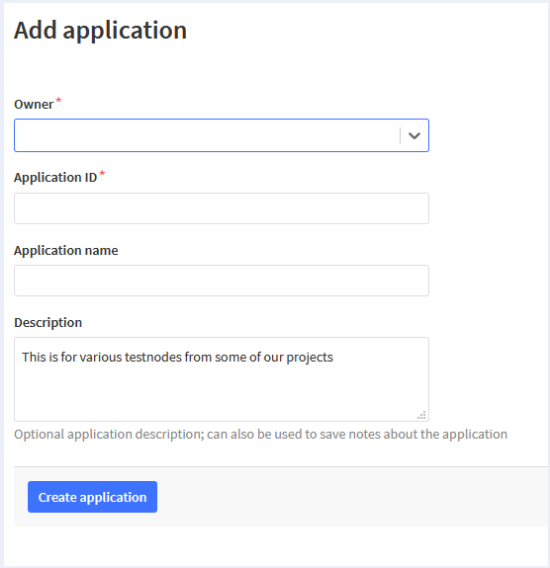


Bild 14. Assistent zum Hinzufügen einer neuen Anwendung.

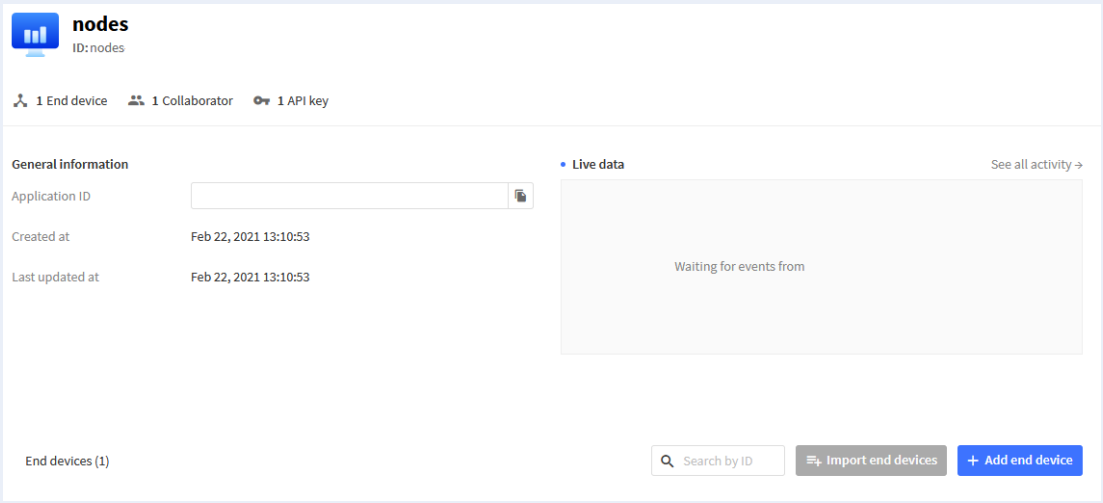


Bild 15. Neues Gerät hinzufügen.

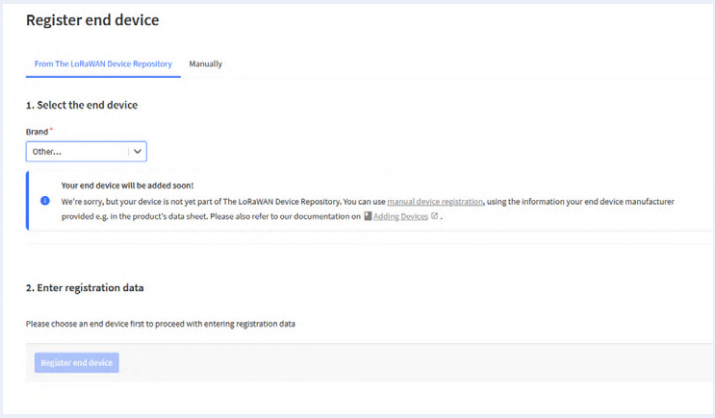


Bild 16. Assistent zum Anlegen eines neuen Knotens.

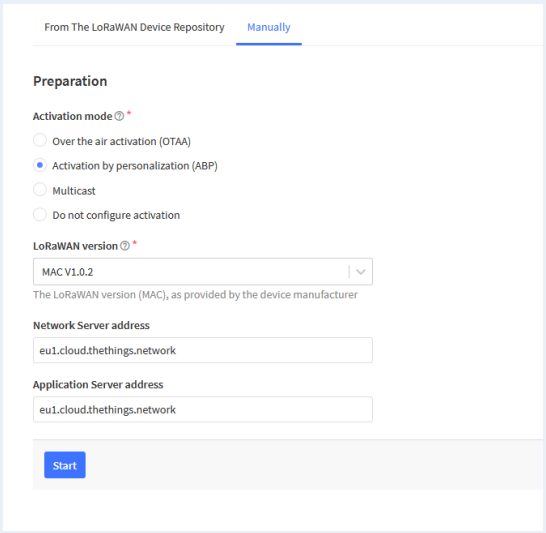


Bild 17. Manuelle Einstellungen für den ABP-Modus.

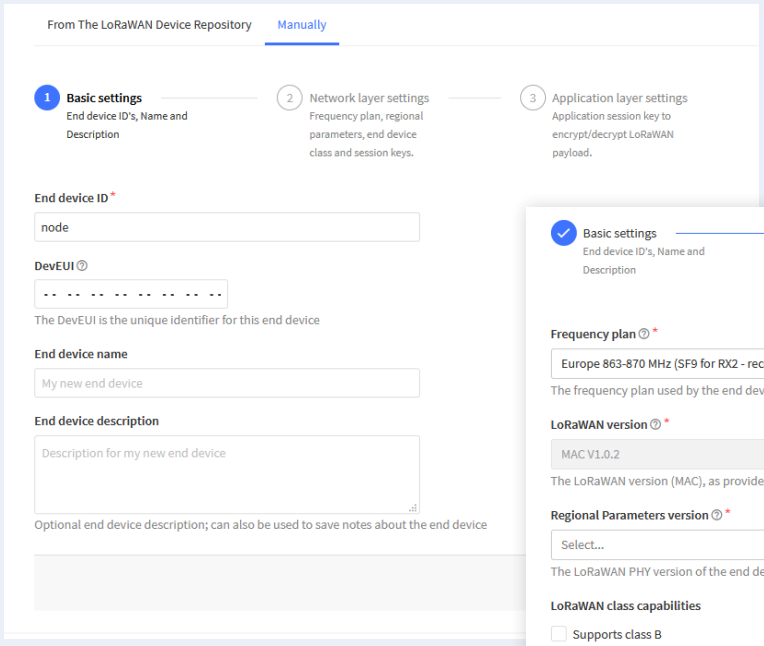


Bild 18. Einstellungen des Namens und der DevEUI.

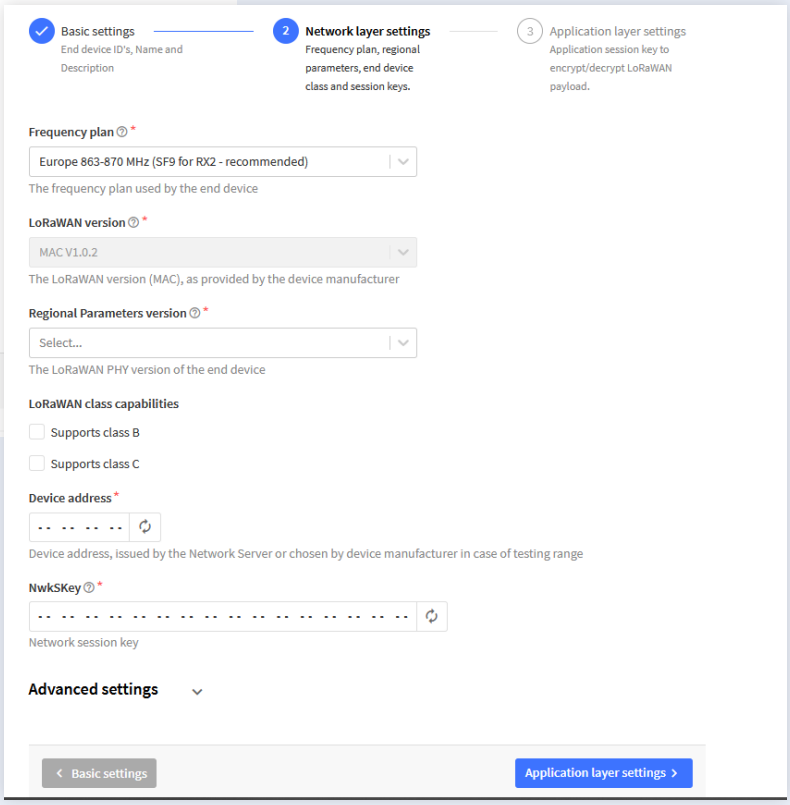


Bild 19. Einstellung des Frequenzbandes.

Temperatursensor am One-Wire-Bus gesucht. Wenn keiner vorhanden ist, wird nach 120 Sekunden im Sleep-Modus ein neuer Versuch unternommen, einen zu finden. Ist ein Sensor aber vorhanden, wird dessen Temperaturregister gelesen und anschließend übertragen. Die Art und Weise, wie der Wert kodiert wird, ist nicht sehr effizient hinsichtlich seiner Nutzlast, wird aber später gut verständlich für die Demonstration sein. **Bild 12** zeigt die übertragenen Daten in der Konsole von The Things Network. Nach einer Übertragung müssen wir 900 Sekunden warten, um die Fair Use Policy von The Things Network einzuhalten. Wenn Sie Ihren Code automatisch auf Ihrem Raspberry Pi Pico starten möchten, speichern Sie die l*ora.py* als *main.py*.

Vielleicht bemerken Sie im Code, dass stets ein Wert in eine Datei namens *current.txt* geschrieben wird. In dieser Datei wird der aktuelle Frame-Zähler für die Übertragung an The Things Network gespeichert. Würden wir den Wert nicht speichern, wäre er bei jedem Neustart des Knotens, zum Beispiel nach einem Batteriewechsel, wieder auf Null gesetzt und die übertragenen Daten würden aus Sicherheitsgründen von The Things Network zurückgewiesen werden. Nach jeder Übertragung wird deshalb die Datei mit dem aktuell verwendeten Wert aktualisiert und beim nächsten Neustart geladen. Wenn wir alle 15 Minuten einen neuen Wert schreiben, bedeutet dies 96 Schreibvorgänge pro Tag oder 35040 Schreibvorgänge pro Jahr, Dies ist nicht ideal und verkürzt zudem die Lebensdauer des Flash-Speichers, muss also bei einer echten Praxisanwendung verbessert werden.

Einmal Internet der Dinge und zurück

Wie man einen LoRa-Knoten einrichtet, wurde bereits in früheren Artikeln beschrieben, aber ich möchte hier dennoch ein paar Worte darüber verlieren. Der frühere Artikel zeigte den Stack von The Things Network in Version 2. Kürzlich wurde die Version 3 angekündigt, und wenn dieses Elektor-Heft erscheint, können Sie bestimmt Ihre Anwendungen und Gateways schon in den neuen Stack der Version 3 und ihrer Konsole übertragen.

Um Daten übertragen zu können, müssen Sie die mitgelieferten MicroPython-Dateien um einige wichtige Parameter ergänzen. Um die Communities-Infrastruktur der Cloud zu nutzen, ist ein Account bei The Things Network erforderlich. Wenn schon ein Konto erstellt wurde oder man eines erstellen möchte, kann man über [10] zur Stack-Version 3 wechseln. Nach der Anmeldung bei unserem Konto müssen wir einen neuen Knoten erstellen. Dazu müssen wir zunächst eine Anwendung einrichten. Wie **Bild 13** zeigt, klickt man dazu auf *Create an application* und folgt dem Benutzerdialog in **Bild 14**. Wählen Sie den *Owner* dieser Applikation, in diesem Fall Ihr Konto, und geben Sie die *Application-ID* ein. Ein Klick auf *Create application* schließt die Einrichtung der Applikation ab.

Jetzt können wir der Applikation einen neuen Knoten hinzufügen, der die Anmeldeinformationen bereitstellt, die wir für das MicroPython-Skript benötigen. Wählen Sie in der gerade erstellten Applikation *Add end device*, um den Assistenten zum Erstellen eines neuen Knotens zu starten, wie in **Bild 15** zu sehen.

Der Assistent in **Bild 16** fordert Sie auf, ein *end device* auszuwählen.

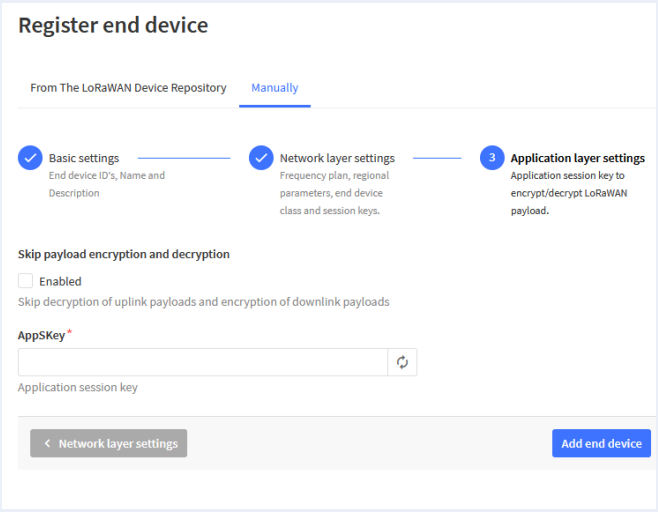


Bild 20. Generierung des Applikationsschlüssels.

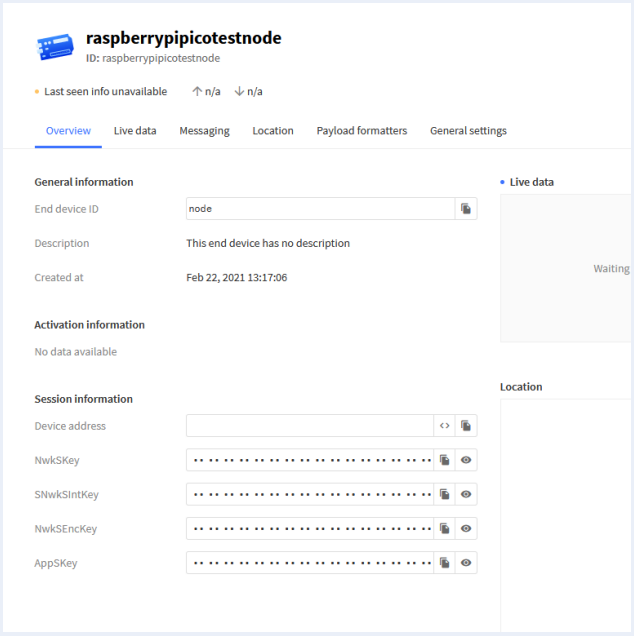


Bild 21. Geräteinformationen für den neuen Knoten.

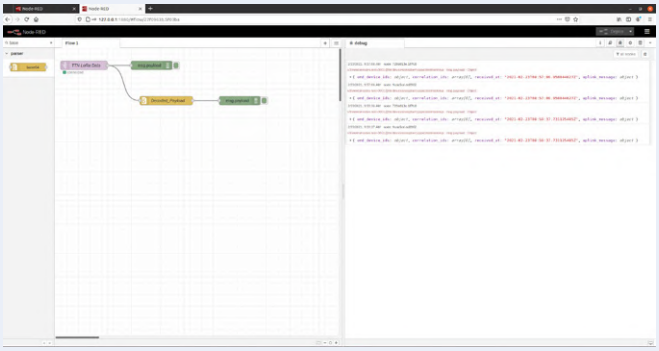


Bild 22. Grundlegender Ablauf für LoRaWAN-Daten in Node-Red.

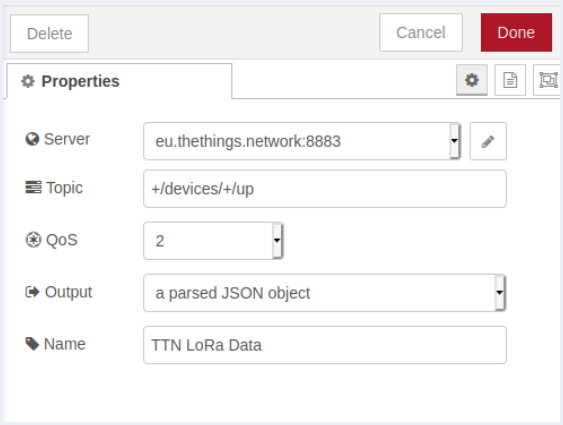


Bild 23. Server-Einstellung für den MQTT-Knoten.

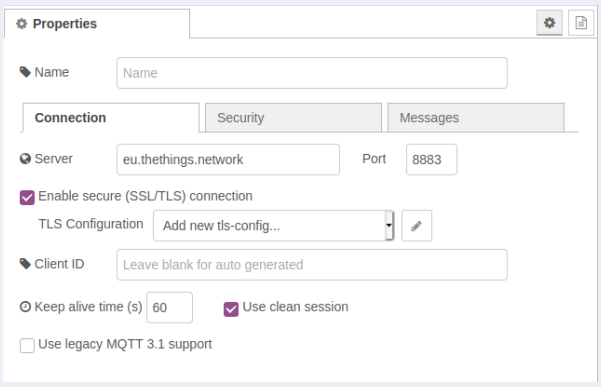


Bild 24. Server- und SSL-Einstellungen.

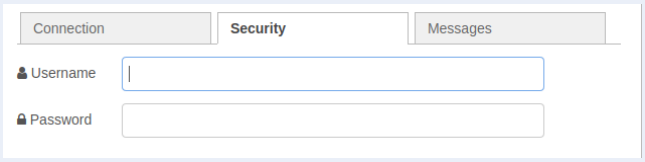


Bild 25. Angabe von Benutzernamen und Passwort.

Da unser Gerät nicht in der Dropdown-Liste enthalten ist, müssen Sie *Manually* wählen. **Bild 17** zeigt, dass *Activation by personalization (ABP)* und die *LoRaWAN version* für MAC V1.0.2 gewählt werden. Dann drückt man *Start*, um zu den Grundeinstellungen zu gelangen (**Bild 18**). Geben Sie die *End device ID* ein, eine eindeutige ID für Ihren Knoten. Die Felder *End device name* und *End device description* helfen Ihnen später zu unterscheiden, welche Knoten Sie angelegt haben. Auf der nächsten Seite in **Bild 19** müssen Sie den ISM-Frequenzbereich Ihrer Region festlegen. Da der von uns erstellte Knoten das LoRaWAN nur zum Übertragen von Daten nutzt, gibt es keine Unterstützung für Klasse B oder Klasse C. Mit *Application layer settings* setzen Sie die Einrichtung fort und generieren Sie einen *Application session key*

wie in **Bild 20** zu sehen. Schließen Sie den Assistenten mit *Add end device* ab.
Es wird wie in **Bild 21** zu sehen ein neu generiertes Device angezeigt. Wir benötigen für unser MicroPython-Skript die *Device address*, *NwkSKey* und *AppSKey*. Die Angaben werden in *lora.py*, wobei `DEVADDR` die *Device address*, `NWKEY` den *NwkSKey* und `APP` den *AppSKey* enthält. Wenn alle Daten korrekt eingegeben sind, wird im The Things Network unmittelbar die Übertragung angezeigt.

Daten mit Node-RED abgeholt

Um Node-RED auf einem Raspberry Pi zu installieren, öffnen Sie ein Terminal oder loggen sich per SSH ein und führen den folgenden Befehl aus:

```
bash <(curl -sL https://raw.githubusercontent.
com/node-red/linux-installers/master/deb/
update-nodejs-and-nodered
```

Um Node-RED als Dienst laufen zu lassen, müssen Sie nach der Node-RED-Installation zusätzlich den folgenden Befehl eingeben und ausführen:

```
sudo systemctl enable nodered.service
```

Eine ausführliche Installationsanleitung für Node-RED finden Sie unter [11]. Um den Dienst zu starten, führen Sie den Befehl `sudo systemctl start nodered.service` im Terminal aus. Node-RED ist nun eingerichtet und gestartet, jetzt müssen wir, um unsere übertragenen Daten zu erhalten, eine Flowchart einrichten, wie sie **Bild 22** zeigt. Auch wenn schon Knoten für The Things Network existieren, müssen Sie auf dem Raspberry Pi die generische MQTT-Konnektivität nutzen. Die Verwendung der vorhandenen Knoten für The Things Network kann auf einem Raspberry Pi zu Fehlern und zum Absturz Ihres Datenflusses führen. Dies dürfte an Kompatibilitätsproblemen mit den bereitgestellten Knoten liegen, da diese auf X86-basierter Hardware problemlos laufen. Für die MQTT-Konnektivität sind wir an den Daten interessiert, die unser Knoten überträgt, die sogenannte Nutzlast. Bild 22 zeigt den kompletten Node-RED-Ablauf zum Abgreifen der Daten, die wir übertragen. Die erforderliche Konfiguration beginnt mit dem MQTT-Knoten. Hier müssen wir den Server, der unsere Daten bereitstellt, und die Anmeldedaten angeben.
Bild 23 zeigt, wie der *Server* und das *Topic* in Node-RED eingegeben werden müssen. Ein Topic kann als ein Konversationsraum zu einem definierten Thema betrachtet werden. Auf diese Weise werden nur Nachrichten bereitgestellt, an denen ein Interesse besteht. Wir verwenden `+/devices/+/up` als Topic. Dies besagt, dass wir an allen Nachrichten interessiert sind, die von den in unserer Anwendung registrierten Knoten an uns gesendet werden. Als Ausgabe erwarten wir ein geparstes JSON-Objekt zur weiteren Verarbeitung.
Im nächsten Schritt werden die Einstellungen für die Server von The Things Network vorgenommen. Um eine Servereinstellung zu bearbeiten, klicken Sie auf das Bleistift-Symbol rechts neben Server. Es öffnet sich ein neuer Dialog wie in **Bild 24**. Da wir auf dem neuen V3-Stack arbeiten, verwenden Sie *eu1.cloud.thethings.network* mit Port *8883*. *Enable secure (SSL/TLS) connection* muss aktiviert sein. Auf dem Tab Security (**Bild 25**) muss ein Benutzernamen und ein Passwort eingegeben werden. Verwenden Sie als Benutzernamen den Namen der Anwendung, die wir zuvor in der

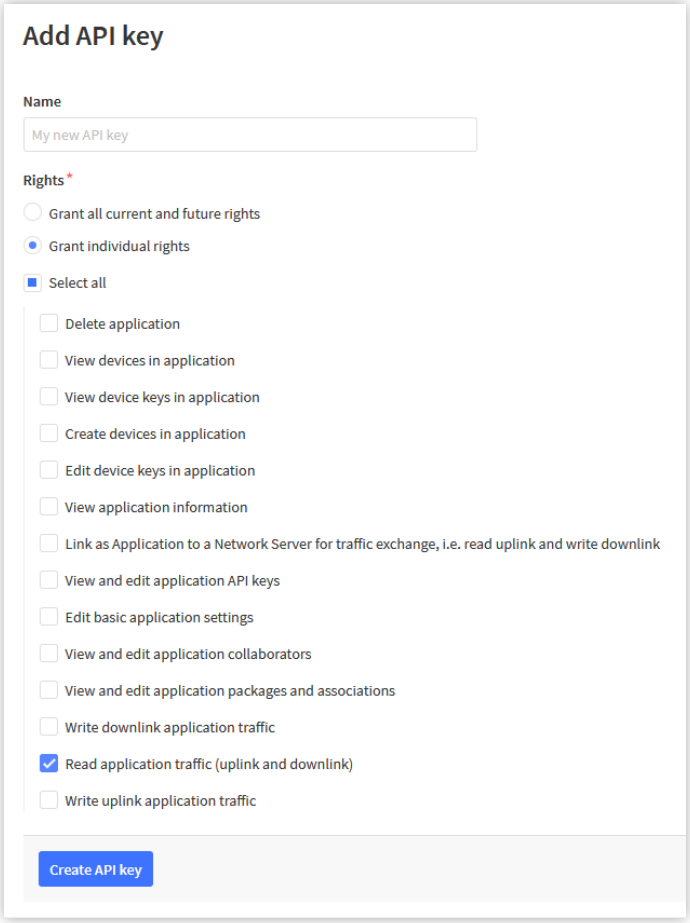


Bild 26. Assistent für einen neuen API-Schlüssel.

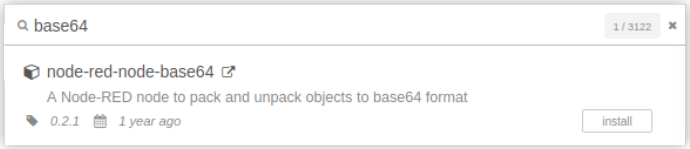


Bild 27. BASE64-Decoder-Einstellungen in Node-RED.

Konsole von The Things Network erstellt haben.
Um das Passwort zu erhalten, ist etwas mehr Arbeit nötig. Gehen Sie zurück in die Konsole von The Things Network und öffnen Sie Ihre Applikation. Klicken Sie auf den Menüeintrag *API keys* auf der linken Seite und starten Sie die Generierung eines API-Schlüssels mit *Add API key*. Es erscheint ein neuer Assistent wie in **Bild 26**. Wählen Sie die gewünschten Zugriffsrechte und beenden Sie den Dialog mit *Create API Key*. Als nächstes wird Ihnen ein neuer API-Schlüssel präsentiert, den Sie verwenden können. Bewahren Sie den Schlüssel gut auf, da Sie später nicht mehr darauf zugreifen können. Dieser Schlüssel ist das Passwort, das wir für Node-RED verwenden müssen.
Wenn alle Einstellungen in Node-RED angegeben sind, werden die Änderungen mit *Done* übernommen. Der MQTT-Knoten baut eine Verbindung auf und präsentiert neue Daten am Knoten. Bei den empfangenen Daten gibt es nur einen Haken. Die Payload, also die Daten, die unser Knoten sendet, sind mit BASE64 kodiert. Um die übertragenen Rohbytes abzurufen, müssen wir einen BASE64-Decoder einfügen, der wie in **Bild 27** konfiguriert ist. Nach der Konversion können wir auf die übertragenen Daten in Form eine Byte-Array

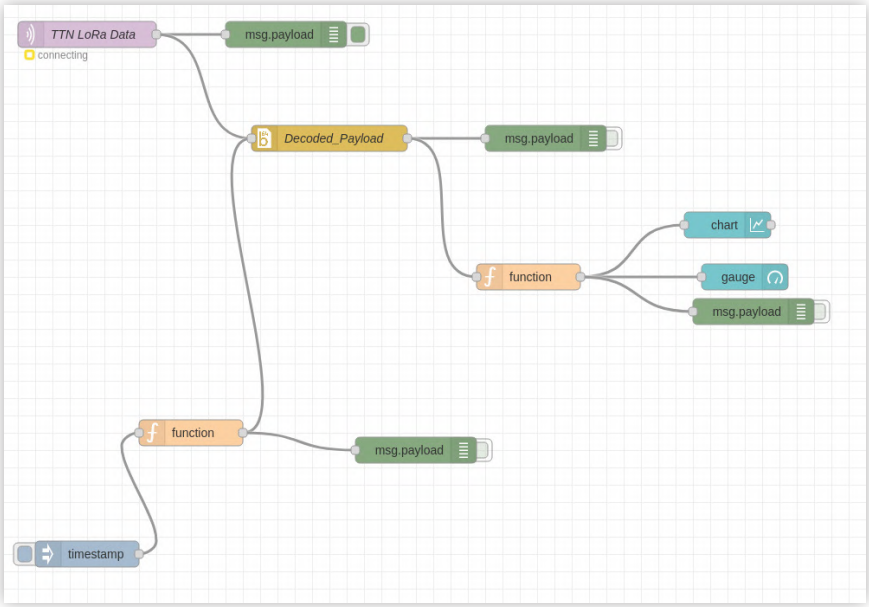


Bild 28. Beispiel für den Datenfluss in Node-RED.

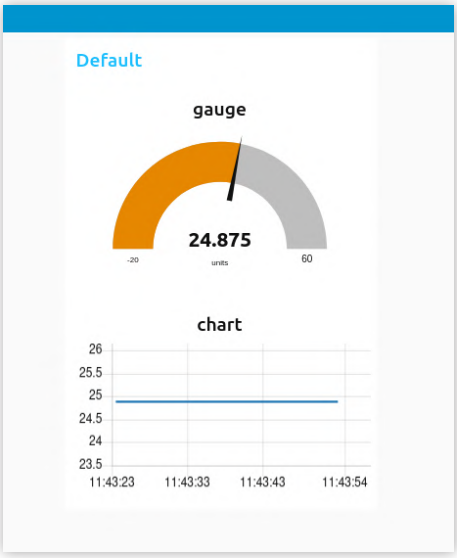


Bild 29. Die Daten werden grafisch auf der Webseite dargestellt.

von aufeinander folgenden Knoten zugreifen.
Der Einfachheit halber wurde ein Demo-Flowchart wie in **Bild 28** eingerichtet, das neu ankommende Daten anzeigt, die über einen Webbrowser zugänglich sind. **Bild 29** zeigt, wie die Webseite am Ende aussieht. Hier müssen Sie nur noch Ihre Anmeldedaten eingeben.

Einfach und anfängerfreundlich

Dies ist zwar nicht das anspruchvollste Projekt, aber es ermöglicht Ihnen, mit dem Raspberry Pi Pico und LoRaWAN zu spielen. MicroPython ist besonders für Anfänger ein sanfter Übergang in die Welt der eingebetteten Systeme. Der Zeitaufwand für den Bau und Betrieb dieses Projekts macht es geeignet für (virtuelle) Klassenzimmer und den Unterricht. Allerdings habe ich Ihnen zu Beginn eine Platine versprochen. Wenn Sie nur an Software interessiert sind, können Sie jetzt aufhören zu lesen. Doch wenn Sie Elektronik nicht nur theoretisch entwerfen, sondern auch bauen wollen, dann ist der nächste Teil etwas für Sie!

Ein Wort der Warnung

Wenn wir in Elektor Schaltpläne und Platinen vorstellen, wurden sie normalerweise aufgebaut und getestet (zumindest soweit, dass sie nicht Brände entfachen oder süße Kätzchen töten). Bei dieser Platine und diesem Schaltplan ist das anders. Die Platine und damit auch der Schaltplan sind noch „in Arbeit". Sie sollte zwar funktionieren, ist aber nicht getestet, so dass Sie sich stets bewusst sein sollten, dass noch einige Bugs vorhanden sein könnten. Sie können alle KiCad-Dateien von der Elektor-Seite oder dem Elektor-GitHub-Repository [12] herunterladen, um sie nach eigenem Gusto zu modifizieren oder zu verbessern. Wenn Sie einige Fehler im Design oder dumme Entscheidungen meinerseits (und damit meine ich nicht so etwas wie die Verwendung eines Raspberry Pi Pico) finden, wenn Sie meinen, die eine oder andere Funktion hätte noch aufgenommen werden sollen, können Sie uns gerne Ihre Vorschläge mitteilen. Es handelt es sich schließlich um „work in progress". Ihr Feedback ist willkommen!

Schaltplan: Teil 1

Der Schaltplan ist in zwei Teile aufgeteilt, auch wenn alle Komponenten auf einem KiCad-Blatt untergebracht sind. Der erste Teil in Bild 3 betrifft nur die Verbindungen zwischen dem LoRa-Transceiver RFM95, dem DS18B20-Temperatursensor und dem Raspberry Pi Pico. Für den RFM95 benötigen wir eine SPI-Verbindung, bestehend aus MISO, MOSI, SCK und nCS. Darüber hinaus müssen wir RESET und DIO0 für einen Minimalbetrieb hinzufügen. Die beiden Kondensatoren C1 (10 µF) und C2 (100 nF) stellen die im Datenblatt empfohlenen Energiespeicher für die Stromspitzen dar, die beim RFM95 im Sende- oder Empfangsmodus auftreten.
Außerdem sehen Sie R5, einen 4,7-kΩ-Widerstand an der Reset-Leitung. Eigentlich wird dieser nicht benötigt, da der RFM95 einen internen Pull-Up-Widerstand enthält. Es hat sich aber im praktischen Einsatz des Moduls gezeigt, dass ein externer Pull-up unerwünschte Resets verhindert, die aufgrund von Störeinstrahlungen sonst ab und zu auftreten können. Etwas, das die MicroPython-Software nicht nutzt, aber später von C/C++-Bibliotheken verwendet werden kann, sind DIO0, DIO1, DIO2 und DIO3. Während DIO0 für den allgemeinen Betrieb der meisten LoRa-Bibliotheken erforderlich ist, lassen sich die anderen DIOs für optionale Funktionen mit anderen Bibliotheken nutzen.
Vom „echten" One-Wire-Temperatursensor DS18B20 sind zahlreiche mehr oder weniger funktionskompatible Varianten im Handel. Während der originale Sensor eine so genannte parasitäre Versorgung erlaubt und der VCC-Pin nicht angeschlossen werden muss, zeigen etliche „falsche" Exemplare ein merkwürdiges und nicht vertrauenswürdiges Verhalten, will man sie im parasitären Modus betreiben. Zur Sicherheit haben wir deshalb den VCC-Anschluss fest mit 3,3 V verbunden. Für One-Wire benötigen wir zusätzlichen einen 4,7-kΩ Pull-Up-Widerstand auf der Datenleitung.
Der Raspberry Pi Pico ist ein fertiges Modul, an das Sie nicht viele Bauteile anschließen müssen. An die Leitungen mit den Bezeichnungen V_BUS (Pin 40) und V_SYS (Pin 39) kann eine wiederaufladbare Batterie angeschlossen werden. Dies bringt uns zu Teil 2 des Schaltplans.

Schaltplan: Teil 2

Der zweite Teil des Schaltplans in **Bild 30** betrifft die Akku-Stromversorgung mit einem MCP73871-1CC-Lithium-Lade-IC mit Strombypass. Wenn wir also einen ausreichenden Strom an seinem Eingang zur Verfügung stellen, gibt es diesen an den Ausgang weiter und benutzt

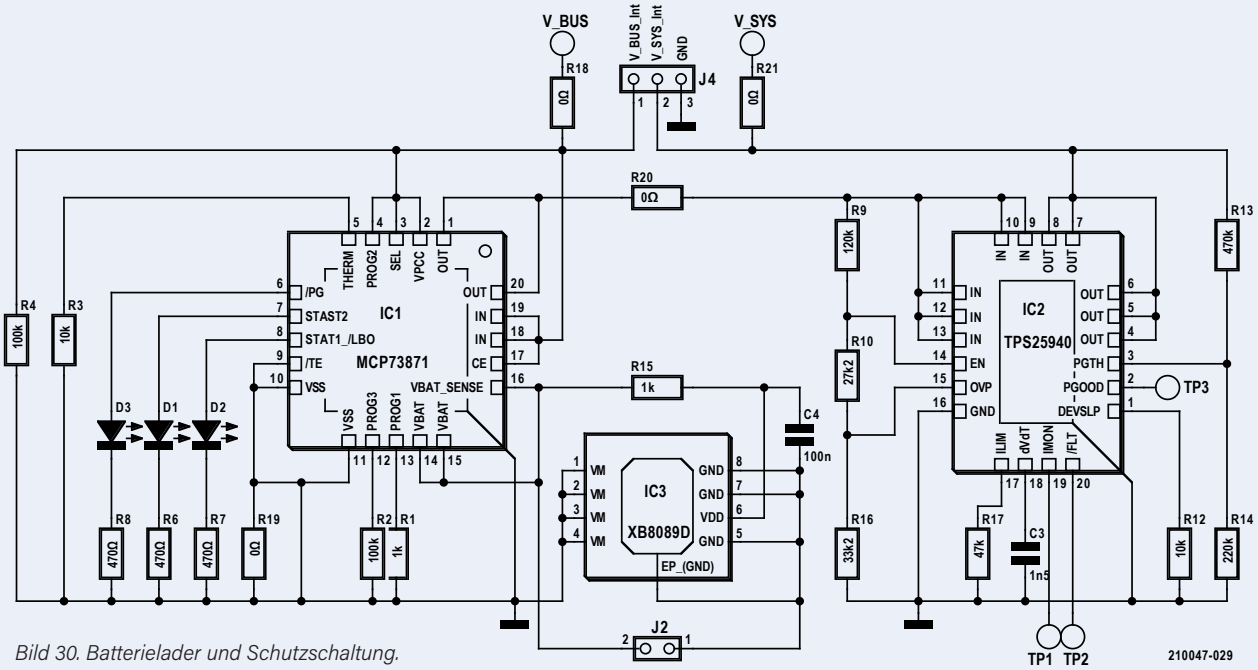


Bild 30. Batterielader und Schutzschaltung.

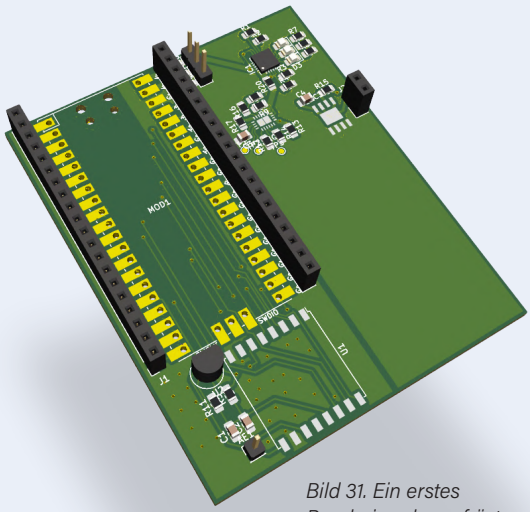


Bild 31. Ein erstes Rendering der gefrästen Leiterplatte.

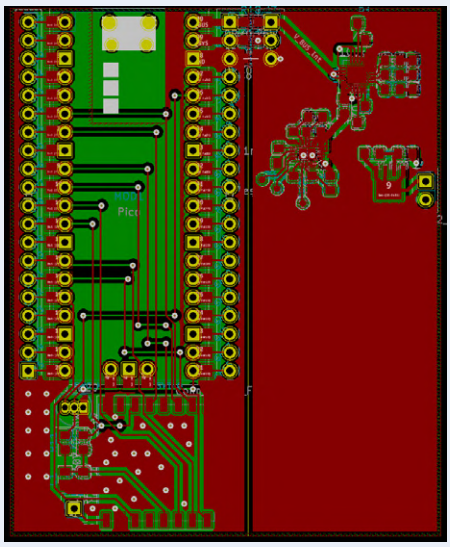


Bild 32. Vorschau auf das fertige Layout.

den Akku überhaupt nicht. Wenn aber die externe Stromversorgung unterbrochen wird oder nicht ausreicht, erfolgt eine automatische Umschaltung auf die Batterie. So wie gezeigt, ist das Lade-IC auf das Laden von Batterien mit 1000 mA eingestellt.

Dem Schutz des angeschlossenen Akkus dient ein XB8089D, ein IC, das bereits im GreatScott! DIY LiPo Supercharger Kit [14] Verwendung fand. Es verhindert Überladung, Tiefentladung, Überstrom und Verpolung. Das letzte Element TPS25940 ist eine eFuse von Texas Instruments, die als ideale Diode arbeitet und einen Stromrückfluss zum Lithium-Batterieladegerät verhindert. Sie arbeitet gleichermaßen als Überstrom- und Überentladungsschutz. Die Schwellwerte für Über- und Unterspannung werden durch R9 (120 kΩ), R10 (27,2 kΩ) und R16 (33,2 kΩ) auf 5,37 V respektive 2,957 V festgelegt. Dies sollte im sicheren Bereich für die Batterie und für den DC/DC-Wandler auf dem Raspberry Pi Pico liegen. Das Datenblatt liefert die erforderlichen Formeln zur Berechnung der gewünschten Werte. Mit R17 = 47 kΩ wird der Ladestrom auf 1,89 A begrenzt, was das Maximum für diesen Lithium-Lader darstellt. Warum es eine schlechte Idee wäre, hier eine Polysicherung zu verwenden, erfahren Sie im Elektor-Artikel [13]. Eine Diode verursacht einen Spannungsabfall von wenigstens 0,3 V, was bedeuten würde, wertvolle Energie als Wärme zu verheizen.

Etwas, das man nicht oft auf Platinen sieht, sind die 0-Ω-Widerstände R20, R21 und R18. Mit ihnen können Sie Teile des Ladegeräts leicht (er als mit Durchkratzen von Leiterbahnen) abtrennen, wenn einige Teile näher untersucht werden müssen oder gar nicht funktionieren wollen.

Eine schnelle Platine

Der Schaltplan ist fertig, alle Bauteile auf einer Platine platziert und geroutet. Das sieht zwar noch nicht so schön aus, aber für einen ersten Versuch erfüllt es den Zweck. **Bild 31** zeigt eine erste virtuelle Ansicht der Platine und **Bild 32** gibt einen Eindruck eines Layouts. Sie können auch sehen, dass die Platine so gestaltet ist, dass der Ladeteil komplett entfernt und auch für andere Zwecke verwendet werden kann. Wenn es nicht funktioniert oder funktionieren soll, kann man es auch von

der Platine entfernen.

Was ist der nächste Schritt?

Die nächsten Schritte hängen vom Feedback unserer Leser ab, also von Ihnen. Wenn Ihnen dieses Projekt gefällt und Sie möchten, dass wir es fortsetzen, Bauteile und/oder Funktionen hinzufügen oder Vorschläge zur Verbesserung haben, können Sie gerne einen Kommentar hinterlassen oder uns eine Nachricht schicken. Auch, wenn Sie Vorschläge für verschiedene Bauteile haben, sind wir interessiert. Sie können uns auch eine Nachricht über andere potentiell interessante Projekte hinterlassen. Die Mail-Adresse(n) finden Sie unter **Fragen oder Kommentare?**

Abschließende Überlegungen

Die Verwendung von MicroPython auf dem Raspberry Pi Pico und die Arbeit an einem LoRaWAN-Projekt ist recht einfach und kann auch einem Anfänger Spaß machen. Aber die Stabilität der Software kann ein Problem sein. Für zuverlässigere Setups ist der Weg über C/C++ vorzuziehen. Wenn die MicroPython-Skripte funktionieren, ist das aber ein schneller und bequemer Weg zu brauchbaren Resultaten. Welche physikalischen Größen Sie messen und übertragen, bleibt Ihnen überlassen, und der verwendete Code ist einfach genug, um ihn mit Ihren Kindern auszuprobieren. Warum nur Temperaturinformationen übertragen? Sie könnten ein LoRaWAN-Alarmsystem mit einigen PIR-Meldern entwickeln. Oder Sie können überwachen, ob Ihre Pflanzen Wasser brauchen. Lassen Sie Ihrer Kreativität freien Lauf! Dies ist das erste Projekt mit dem Raspberry Pi Pico, aber keinesfalls letzte. Wir arbeiten mit Hochdruck an weiteren Projekten, die bei ihrer Veröffentlichung aber eventuell genau wie dieses noch „work in progress" sind. Dennoch hoffen wir, dass sie Sie für Ihre eigenen Projekte inspirieren oder Ihnen einige Hintergrundinformationen geben, die in Zukunft nützlich sein könnten.

PASSENDE PRODUKTE

- **Raspberry Pi Pico**
  www.elektor.de/raspberry-pi-pico-rp2040-with-pre-soldered-headers
- **SeeedStudio RFM95 Ultra-long LoRa Transceiver-Modul (868 MHz)**
  www.elektor.de/seeedstudio-rfm95-ultra-long-lora-transceiver-module-868-mhz
- **Breadboard**
  www.elektor.de/breadboard-830-tie-points
  www.elektor.de/solderless-breadboard-1660-tie-point-zy-204
  www.elektor.de/pimoroni-maker-essentials-mini-breadboards-jumper-jerky
- **Node-RED-Buch (Papier und E-Book)**
  www.elektor.de/programming-with-node-red
  www.elektor.de/programming-with-node-red-e-book

Ein Beitrag von
Entwurf und Text: **Mathias Claußen**
Übersetzung: **Rolf Gerstendorf**
Redaktion: **Jens Nickel**
Layout: **Giel Dols**

Haben Sie Fragen oder Kommentare?
Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Schicken Sie eine E-Mail an den Autor unter mathias.claussen@elektor.com oder kontaktieren Sie Elektor unter editor@elektor.com.

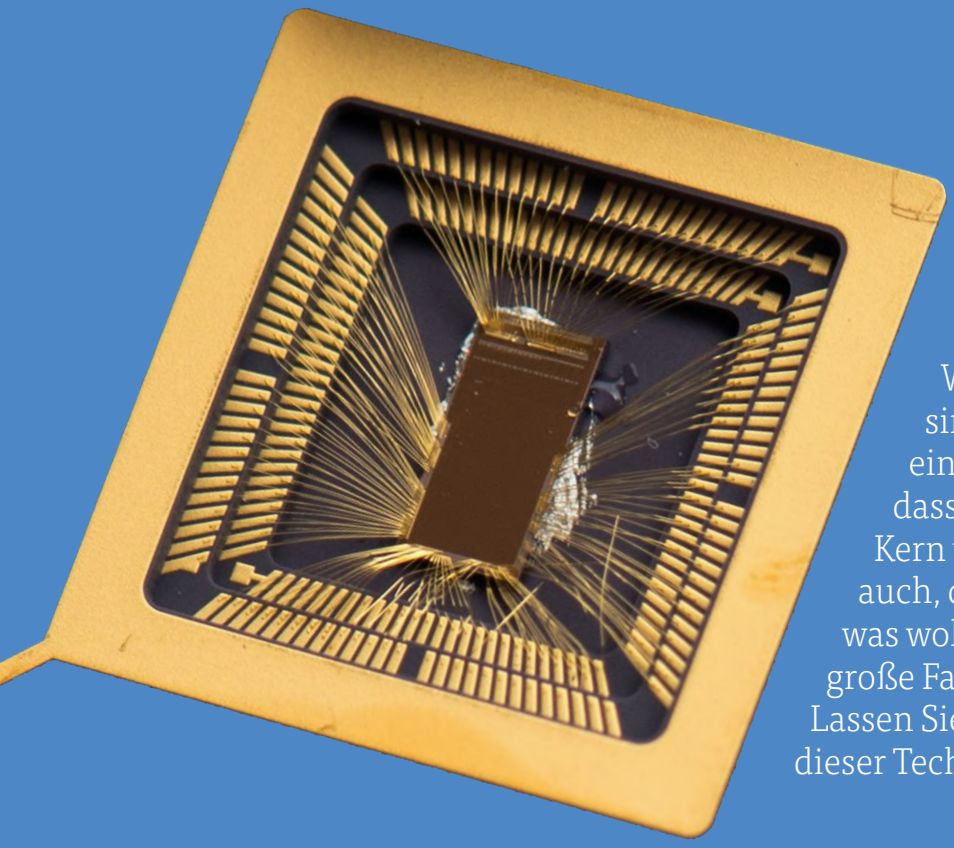
WEBLINKS

[1] **LoRaWAN - ein einfacher Einstieg**: www.elektormagazine.de/magazine/elektor-140/57101
[2] **Projektdateien auf GitHub**: https://github.com/ElektorLabs/210047-LoRa-with-the-Raspberry-Pi-Pico
[3] **Dragino LPS8 Indoor LoRaWAN Gateway**: www.elektor.de/dragino-lps8-indoor-lorawan-gateway
[4] **Dragino PG1301 LoRaWAN GPS Concentrator for Raspberry Pi**: www.elektor.de/dragino-pg1301-lorawan-gps-concentrator-for-raspberry-pi-868-mhz
[5] **Get Started with MicroPython on Raspberry Pi Pico**: www.elektor.de/get-started-with-micropython-on-raspberry-pi-pico
[6] **Kostenlos: Get Started with MicroPython on Raspberry Pi Pico PDF (EN)**: https://hackspace.raspberrypi.org/books/micropython-pico/pdf/download
[7] **GitHub-Repository von uLoRa**: https://github.com/fantasticdonkey/uLoRa
[8] **Raspberry Pi Pico MicroPython**: www.raspberrypi.org/documentation/pico/getting-started/
[9] **Download Thonny Python-IDE**: https://thonny.org/
[10] **The Things Network v3 Stack**: https://eu1.cloud.thethings.network/console/
[11] **Installationsanleitung für Node-Red auf dem Raspberry Pi**: https://nodered.org/docs/getting-started/raspberrypi
[12] **Elektor GitHub**: https://github.com/ElektorLabs
[13] **DIY LiPo Supercharger Bundle**: www.elektormagazine.de/articles/diy-lipo-supercharger-bundle
[14] **GreatScott! DIY LiPo Supercharger Kit**: www.elektormagazine.de/191188-B-01
[15] **Gerberfiles für RFM95-BOB**: https://github.com/ElektorLabs/191069-RFM95-BOB/

Was ist **RISC-V**?

Warum die Industrie einen neuen Prozessorkern so spannend findet

Von **Stuart Cording** (Elektor)



Die Elektronikindustrie scheint verrückt nach RISC-V geworden zu sein. Aber warum? Was ist das überhaupt? Und wie können Sie daran teilhaben? Wenn Sie schon etwas informiert sind, wissen Sie, dass es sich um einen Prozessorkern handelt, und dass es einige Chips gibt, die diesen Kern verwenden. Sie wissen vielleicht auch, dass RISC-V „frei und offen" ist, was wohl für die Aufregung und die große Fangemeinde verantwortlich ist. Lassen Sie uns herausfinden, was es mit dieser Technologie wirklich auf sich hat!

Zu Beginn ist es wichtig zu verstehen, dass RISC-V gar kein Prozessor ist, sondern eine Befehlssatzarchitektur (Instruction Set Architecture, ISA [1]). Das bedeutet, dass die Community, die hinter RISC-V steht, eine Beschreibung erstellt hat, wie ein Prozessordesign funktionieren sollte, wenn er auf der RISC-V-ISA aufbaut. Und wenn wir Design sagen, dann meinen wir wirklich die Erstellung des Prozessors mit all seinen Registern, Akkumulatoren, mathematischen Operationen, Speicherbussen und allem anderen.

Die ISA dokumentiert, um nur ein paar Punkte zu nennen, die unterstützten Operationen, die Möglichkeiten der Speicheradressierung, wie der Stack funktioniert und was passiert, wenn Interrupts auftreten. Zu den unterstützten Operationen wird erklärt, wie viele Bits zur Kodierung einer Anweisung und welche Bits zur Kodierung der Quelle der benötigten Operanden verwendet werden.

Der Grund für die ganze Aufregung um RISC-V ist, dass die ISA frei und offen ist. Offen bedeutet, dass jeder zu seiner Entwicklung beitragen kann, frei meint, dass es nicht die Bohne kostet, sie zu benutzen. Aber genauso wie die Arduino-Board-Designs zwar offen und kostenlos sind, bedeutet das nicht, dass ein RISC-V-Board kein Geld kostet. Das Gleiche gilt für den Aufbau Ihres Traumdesigns auf RISC-V-Basis.

Mit was konkurriert RISC-V?

Jeder Prozessor hat eine ISA, fast alle sind proprietär, und einige sind lizenzierbar. Microchip zum Beispiel stellt Chips her, die 8-Bit- und 16-Bit-PIC-Prozessoren verwenden, und irgendwo gibt es eine ISA, die sie beschreibt. Diese proprietären Kerne sind Eigentum von Microchip und werden von Microchip in ihren Mikrocontrollern verkauft. Wenn Sie Ihren eigenen Mikrocontroller entwerfen wollen, werden Sie sich wahrscheinlich zu Arm und MIPS kommen. Diese proprietären Kerne sind als geistiges Eigentum (intellectual property, IP) lizenzierbar. Die dahinter stehenden Unternehmen machen die ganze Arbeit, um die ISA in ein gutes Prozessordesign umzuwandeln, entwickeln Tools, um sie zu unterstützen, schaffen eine zugehörige Infrastruktur und verlangen eine Gebühr, damit Sie sie nutzen können. Probleme gibt es, wenn diese Optionen nicht ganz das tun, was Sie wollen.

Ihre neue Anwendung muss vielleicht eine

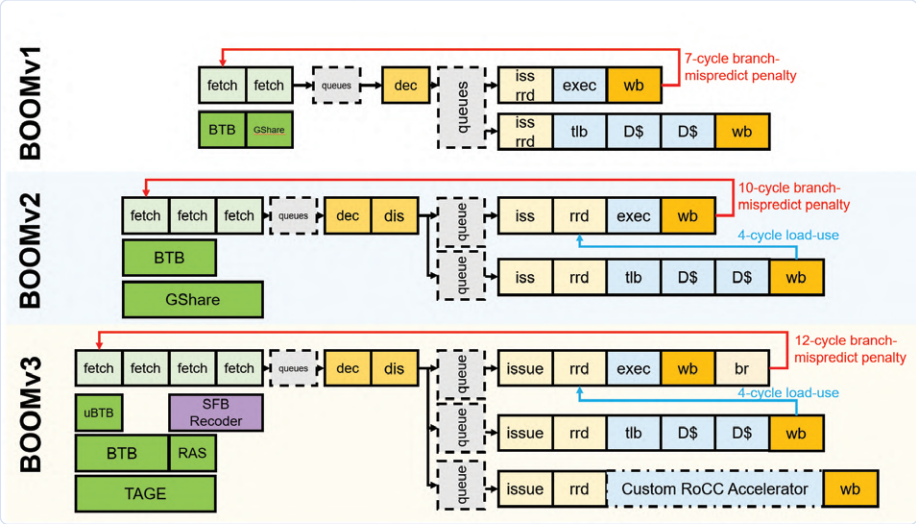


Bild 1. Entwicklungsprozess für die RISC-V-Pipeline-Implementierung, die im BOOM-Projekt [27] verwendet wurde. (Quelle: Mitglieder der University of California)

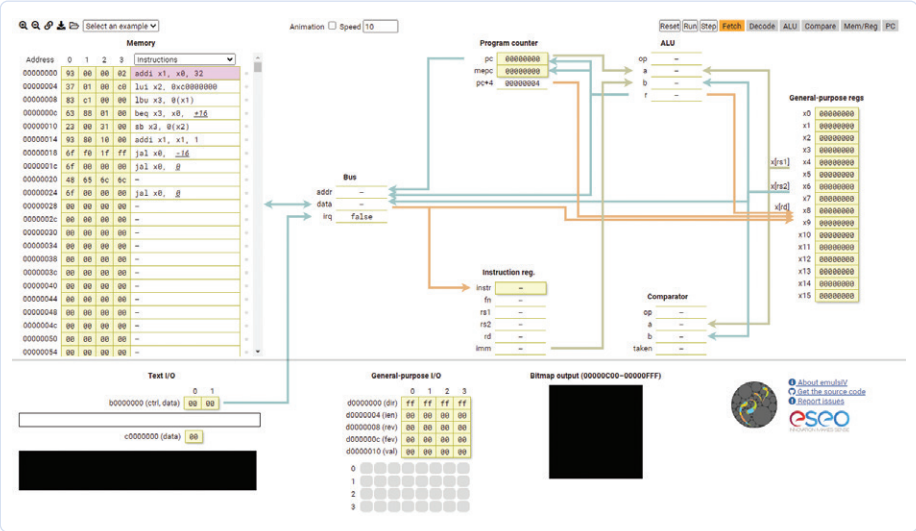


Bild 2. Der emulsiV-Simulator ermöglicht es, RISC-V in einem Webbrowser auszuprobieren.

Aufgabe, zum Beispiel eine Verschlüsselung, sehr schnell und mit minimaler Leistungsaufnahme ausführen. Eine potenziell lizenzierbare Prozessor-IP kann Ihre Aufgabe in 100 Befehlen ausführen. Wenn Sie nun die Stromaufnahme minimieren wollen, müssen Sie eine Fertigungsstätte (silicon fabrication facility, fab) finden, die sich auf Low-Power spezialisiert hat. Und das kann viel teurer sein als ein „Allzweck"-Fertigungsprozess, so dass ihr phantastisches Produkt zu teuer für den Zielmarkt wird.

Vielleicht haben Sie aber auch ein paar clevere Ingenieure, die die Ausführungszeit Ihres Codes optimieren könnten, indem sie neue Befehle für den Prozessor entwickeln. Da aber die ISA proprietär ist, dürfen Sie sie nicht verändern. Sie stecken also mit einem Problem fest, das Sie mit einem Fertigungsansatz lösen müssen. Mehr dazu später...

Was kann RISC-V „out-of-the-Box"?

Die kurze Antwort ist: „nicht viel, aber genug". Grundsätzlich müssen Sie damit beginnen, die genaue Architektur zu wählen, die Sie wollen. Derzeit sind 32-Bit- und 64-Bit-ISAs definiert, an einer 128-Bit-ISA wird gearbeitet. Diese Basisdefinitionen werden RV32I und RV64I genannt. Wenn Sie den RV32I wählen, stehen Ihnen 49 Instruktionen [2] zur Verfügung. Das „I" steht dabei für „Integer". Enthalten sind alle grundlegenden Integer-Arithmetik- und Logikbefehle (ADD, SUB, AND, OR, XOR), Shifts, Compare, Jump und Link sowie einige Systembefehle [3]. Wenn Sie kompakten Code bevorzugen, könnte die C-Option mit 16-Bit-Befehlskodierung, die dem Arm-Thumb-Modus ähnelt, für Sie interessant sein. Befehle zur Multiplikation und Division (M), Atom (A) und Gleitkomma (F, D und Q) können ebenfalls hinzugefügt werden.

Der nächste Schritt besteht darin, den Prozessorkern basierend auf den Eigenschaften der gewählten Optionen in einer Hardwarebeschreibungssprache (Hardware Description Language, HDL [4]) wie VHDL oder Verilog zu entwerfen. Dieser Schritt ist nicht einfach, aber man kann auf die Hilfe der Community bauen. Das Entwerfen von Prozessoren erfordert viel Geschick, daher gibt es eine Reihe von Leuten und Unternehmen, die Ihnen fertige Designs zur Verfügung stellen. Wenn Sie den „freien" Weg gehen wollen, sollten Sie sich die PULP-Plattform [5] ansehen, die von der ETH Zürich und der Università di Bologna entwickelt wurde. Deren CV32E40P RV32IM[F]C-Implementierung ist auf GitHub [6] und der Befehlsdecoder auf [7] verfügbar, falls Sie sehen möchten, wie so etwas implementiert wird. Eine weitere Implementierung ist das BOOM-Projekt, ein hochleistungsfähiger und parametrisierbarer Kern für die Architekturforschung, entwickelt von der University of California in Berkeley (**Bild 1**).

Wenn Sie es eilig haben und Unterstützung brauchen, dann müssen Sie etwas Geld auf den Tisch legen und eine Implementierung von jemandem wie SiFive [8] lizenzieren. SiFive hat eine Reihe von 32- und 64-Bit-Designs im Angebot [9], die auch angepasst werden können.

Wie kann ich RISC-V ausprobieren?

Obwohl RISC-V nicht neu ist, gibt es nur wenige Chips, die wir testen könnten. Im industriellen Kontext ist RISC-V noch relativ neu. Wenn Sie mit Mikrocontrollern vertraut sind, werden Sie wissen, dass der Großteil der Industrie auf Arm baut und sich von den proprietären Cores entfernt hat. Dies war eine strategische und langfristige Entscheidung. Ein Wechsel von Arm zu RISC-V würde nur die Lizenzgebühren einsparen und den Anwendern sonst wenig Vorteile bringen. Außerdem müssten sich ihre Entwicklungsteams in RISC-V einarbeiten und es mit ihrem sonstigen IP (Analog, Timer, Busse, Schnittstellen, Speicher) verbinden und die Tools wie die Entwicklungs-IDE, den Compiler, den Debugger und so weiter aktualisieren.

Wenn Sie eine Festplatte von Seagate [10] oder Western Digital [11] besitzen, „benutzen" Sie vielleicht schon RISC-V. Aber Sie wollen

natürlich Ihren eigenen Code auf diesem Kern ausführen. Am schnellsten funktioniert dies mit einem Simulator wie emulsiV [12] mit der RISC-V-Kern-Implementierung „Virgule“, der von ESEO angeboten wird (**Bild 2**). Zusätzlich zum Prozessor bietet der Simulator einige Textein- und -ausgänge, eine Bitmap-Ausgabe und einige Allzweck-E/A (GPIO). Dazu gibt es sieben Beispiele, die die Grundlagen von der Addition und Ausgabe von ASCII-Text bis zur Steuerung der GPIOs abdecken. Ein nettes Detail ist die Option „Animation“ (Checkbox oben in der Mitte), die zeigt, woher die Daten kommen und wohin sie gehen, während der Code ausgeführt wird. Wenn Sie möchten, können Sie den Code in **Listing 1** ausprobieren, indem Sie ihn in einen Texteditor kopieren, die Datei als *program.hex* speichern und in den Simulator laden.

Mit dem HiFive1 Rev B von CrowdSupply [13] können Sie RISC-V im „Arduino-Format“ ausprobieren. Das Board verwendet den SiFive-FE310-G002-Mikrocontroller [14]. Es handelt sich dabei um ein Bare-Bone-Device ausschließlich mit digitaler Peripherie (I²C, UART, SPI, PWM, GPIO) und etwas nichtflüchtiges SRAM in einem Off-Chip-QSPI-Flash. Das Board verfügt über ein WLAN- und ein Bluetooth-Modul und besitzt einen Segger-J-Link für USB-Debugging.

Am anderen Ende des Leistungsspektrums steht das PolarFire-SoC [15] von Microchip mit vier 64-Bit-RISC-V-Cores neben einem FPGA. Diese Plattform ist hoch konfigurierbar, kann Linux ausführen und gleichzeitig „harte“ Echtzeitanwendungen unterstützen.

Wie kann ich RISC-V anpassen?

Wie schon gesagt, besteht der praktische Vorteil von RISC-V darin, den Befehlssatz auf die individuellen Anforderungen der Anwendung abstimmen zu können. Das heißt, wenn die für optimal befundene Prozessor-Implementierung 95 % der Anforderungen erfüllt, können Sie die verbleibenden 5 % mit einigen raffinierten Zusatzfunktionen versehen. Nehmen wir an, dass Ihre Anwendung viel Gebrauch von der ChaCha-Variante [16] der Stromverschlüsselung Salsa20 [17] macht, wie in einer Application Note [18] von Imperas beschrieben, einem weiteren RISC-V-Anbieter, der Verifikations-, Analyse- und Profiling-Tools anbietet.

Sie haben Ihre ChaCha-Implementierung auf einem RISC-V-Kern ausgeführt und festgestellt, dass sie eine erhebliche Menge an Verarbeitungszeit verschlingt. Sie möchten aber nicht nur die Ausführungszeit verbessern, sondern auch von der Leistungsreduzierung profitieren, die sich durch die verringerte Ausführungszeit ergibt, und diese vielleicht nutzen, um in einen stromsparenden Schlafmodus zu wechseln.

Der Code in **Listing 2** verwendet ausgiebig XOR- und Rotate-Anweisungen in einem Schritt, der als „quarter rounds“ bezeichnet wird und für den vier `qrX_c()`-Funktionen in C geschrieben wurden. Eine `processLine()`-Funktion ruft diese vier Funktionen auf, um die Verschlüsselung durchzuführen. Die Analyse der Ausführungszeiten zeigt, dass der Prozessor etwa 55 % seiner Zeit für diese Aufgabe aufwendet und etwa 32 % verteilt auf die vier Quarter-round-Funktionen (**Bild 3**).

Mit RISC-V können wir einfach vier dedizierte Quarter-round-Funktionen implementieren, die in einem einzigen Zyklus ausgeführt werden, anstatt uns auf den Code zu verlassen, den der C-Compiler generiert. Das liegt daran, dass in der ISA ein Bereich für kundenspezifische Anweisungen reserviert ist. Zunächst können die neuen Befehle zu einem RISC-V-Entwurf hinzugefügt werden, wobei die Implementierung des Befehls in C kodiert ist. Dadurch können die Befehle simuliert werden, um ihre Funktionalität zu testen und zu prüfen, ob eine Leistungsverbesserung erreichbar ist. In diesem Fall, mit dedizierten Quarter-round-Befehlen, die auf dem angepassten RISC-V-Kern verfügbar

Listing 1. Roher HEX-Code für den emulsiV-Simulator zum Speichern und Hochladen als program.hex.

```
:10000000930000023701000c083c100006388010033
:10001000230031009380100006ff01fff6f0000007d
:1000200048656c6c6f20456c656b746f72210000c5
:00000001FF
```

Listing 2. C-Code zur Implementierung der ChaCha-Stream-Stromverschlüsselung.

```
unsigned int processLine(unsigned int res, unsigned int word) {
  res = qr1_c(res, word);
  res = qr2_c(res, word);
  res = qr3_c(res, word);
  res = qr4_c(res, word);
  res = qr1_c(res, word);
  res = qr2_c(res, word);
  res = qr3_c(res, word);
  res = qr4_c(res, word);
  return res;
}
```

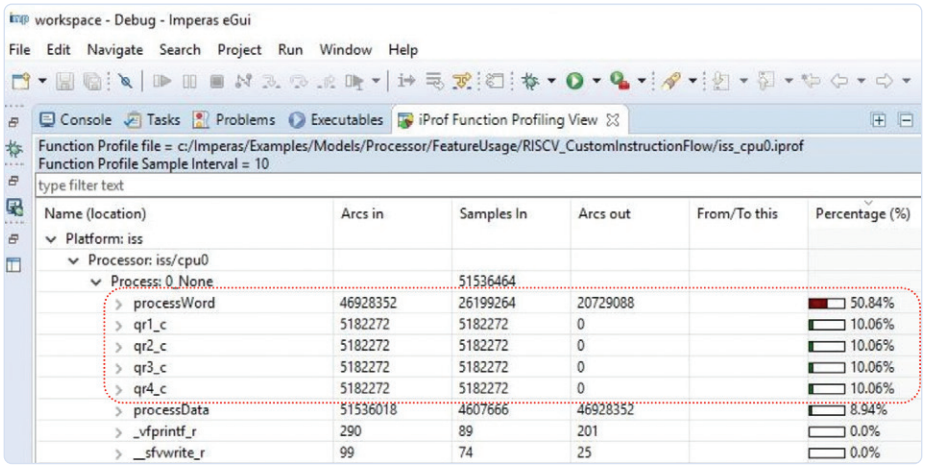
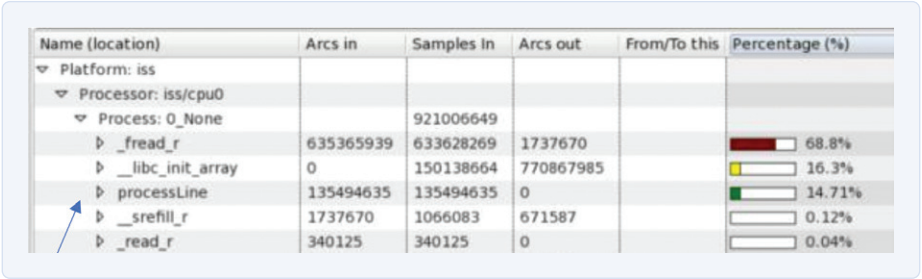


Bild 3. Der ChaCha-Verschlüsselungsstrom beanspruchte mit in Standard-C kompiliertem Code etwa 55 % der Prozessorzeit. (Quelle: Imperas Software Limited)



Name (location)	Arcs in	Samples in	Arcs out	From/To this	Percentage (%)
Platform: iss					
Processor: iss/cpu0					
Process: 0_None		921006649			
_fread_r	635365939	633628269	1737670		68.8%
__libc_init_array	0	150138664	770867985		16.3%
processLine	135494635	135494635	0		14.71%
__srefill_r	1737670	1066083	671587		0.12%
_read_r	340125	340125	0		0.04%

Bild 4. Durch die Verwendung dedizierter, neu entwickelter Befehle sinkt die Prozessorlast für den ChaCha-Verschlüsselungsstrom auf weniger als 15 %. (Quelle: Imperas Software Limited)

sind, benötigt die Funktion `processLine()` weniger als 15 % der verfügbaren Prozessorleistung (**Bild 4**). Wenn Sie damit zufrieden sind, kann das Entwicklungsteam anschließend die Hardware-Implementierung der Anweisungen in Verilog entwickeln.
Leider ist die Verwendung von neuen Befehlen nicht so einfach wie das Neukompilieren des C-Codes (**Listing 3**). Es ist ein enormer Aufwand, einen RISC-V-Compiler so zu modifizieren, dass die neuen Befehle verwendet werden können. Stattdessen werden die hex-kodierten Befehle mit Inline-Assembler auf die gleiche Weise aufgerufen wie handoptimierter Code.

Wie kann ich zu RISC-V beitragen?

Wenn Sie die Weiterentwicklung von RISC-V unterstützen wollen, haben Sie Glück! RISC-V International [19] ist die Organisation, die mit der Entwicklung und Förderung von RISC-V betraut ist (**Bild 5**). Einzelpersonen können als *Community Members* [20] beitreten, Sie können sich aber auch im Rahmen eines Unternehmens und einer Universität [21] aktiv beteiligen.

Wenn Sie nun bei all der Begeisterung eine große Auswahl an RISC-V-Mikrocontrollern auf dem Markt erwarten, werden Sie enttäuscht. GigaDevice bietet einige Controller an [22], und es gibt einen Chip eines russischen Anbieters, der auf den Smart-Metering-Markt abzielt [23]. Arm ist leider zu sehr mit den Big Players verbunden und Start-ups dürften es schwer haben, einen Fuß in den gesättigten Mikrocontrollermarkt zu bekommen, selbst mit dem finanziellen Vorteil eines Prozessors, für den keine Lizenzgebühren gezahlt werden müssen.
Stattdessen ist es wahrscheinlicher, dass RISC-V in spezialisierten Anwendungen eingesetzt wird, bei denen die Möglichkeit, den Kern anzupassen, enorme Vorteile mit sich bringt, etwa bei Anwenungen mit ultraniedrigem Stromverbrauch [24]. Angesichts der Probleme bei der Lizenzierung von Technologie nach China erweist sich RISC-V als beliebte Alternative zum Erwerb von IP aus den USA. Alibaba hat einen 64-Bit-16-Kern-RISC-V-Controller mit 2 GHz angekündigt, der in einem 12-nm-Prozess produziert werden soll [25]. Der Kern ist für den Einsatz in Server-Infrastrukturen vorgesehen. Schließlich hat sich die European Processor Initiative [26] mit

Bild 5. Offizielles Logo der Community RISC-V International, die die Entwicklung des ISA fördert und unterstützt.

Listing 3. Verwendung der neuen RISC-V-Befehle

Die Verwendung neuer RISC-V-Befehle erfordert Inline-Assembler. Die Befehle müssen hexadezimal kodiert sein, da der Assembler die Namen der neuen Befehle nicht kennt.

```
unsigned int processLine(unsigned int res, unsigned int word) {
  asm __volatile__("mv x10, %0" :: "r"(res));
  asm __volatile__("mv x11, %0" :: "r"(word));
  asm __volatile__(".word 0x00B5050B\n" ::: "x10"); // equivalent to qr1_c()
  asm __volatile__(".word 0x00B5150B\n" ::: "x10"); // equivalent to qr2_c()
  asm __volatile__(".word 0x00B5250B\n" ::: "x10"); // equivalent to qr3_c()
  asm __volatile__(".word 0x00B5350B\n" ::: "x10"); // equivalent to qr4_c()
  asm __volatile__(".word 0x00B5050B\n" ::: "x10"); // equivalent to qr1_c()
  asm __volatile__(".word 0x00B5150B\n" ::: "x10"); // equivalent to qr2_c()
  asm __volatile__(".word 0x00B5250B\n" ::: "x10"); // equivalent to qr3_c()
  asm __volatile__(".word 0x00B5350B\n" ::: "x10"); // equivalent to qr4_c()
  return res;
}
```

heterogenen Architekturen befasst, in denen Arm und RISC-V (und andere Kerne) parallel eingesetzt werden könnten. Das Ziel hierbei ist es, das Beste aus den jeweiligen Kernen zu ziehen, indem in Multicore-Designs der beste Prozessor für eine bestimmte Rechenherausforderung verwendet wird.
RISC-V ist nicht der erste freie und offene Versuch einer Prozessor-IP, aber bisher der erfolgreichste. In Anbetracht seiner Geschichte, der Flexibilität, des offenen Ansatzes, des Interesses der akademischen Welt und der umfangreichen Unterstützung durch die Industrie handelt es sich um eine Technologie, die eine oder mehrere Generationen von Ingenieuren während ihrer gesamten Laufbahn begleiten dürfte.

210223-02

Haben Sie Fragen oder Kommentare?
Haben Sie technische Fragen oder Anmerkungen zu diesem Artikel? Dann kontaktieren Sie den Autor direkt über stuart.cording@elektor.com oder die Redaktion über editor@elektor.com.

Ein Beitrag von
Text und Bilder: **Stuart Cording**
Übersetzung: **Rolf Gerstendorf**
Redaktion: **Jens Nickel, C. J. Abate**
Layout: **Giel Dols**

WEBLINKS

[1] **„Microarchitecture and Instruction Set Architecture“ GeeksforGeeks, Oktober2019**: http://bit.ly/3bBKAy2
[2] **„RISC-V“, Wikipedia**: https://de.wikipedia.org/wiki/RISC-V
[3] **J. Zhu, „RISC-V Reference“**: http://bit.ly/30IICXj
[4] **„Hardwarebeschreibungssprache“, Wikipedia**: https://de.wikipedia.org/wiki/Hardwarebeschreibungssprache
[5] **Webseite PULP-Plattform**: https://pulp-platform.org/
[6] **GitHub-Repository - CV32E40P RISC-V core**: https://github.com/openhwgroup/cv32e40p
[7] **SystemVerilog-Implementierung des CV32E40P-Befehlsdekoders**: https://github.com/openhwgroup/cv32e40p/blob/master/rtl/cv32e40p_id_stage.sv
[8] **SiFive-Website**: www.sifive.com
[9] **SiFive-Core-Designer**: http://bit.ly/3rKnkU7
[10] **RISC-V-Webseite von Seagate**: http://bit.ly/3etS0oU
[11] **RISC-V-Webseite von Western Digital**: http://bit.ly/3eyldyP
[12] **emulsiV-Simulator für RISC-V-Virgule-Prozessor**: http://bit.ly/30AzqmG
[13] **CrowdSupply-Webseite für HiFive1 Rev B Arduino-Board**: http://bit.ly/3eww7Fj
[14] **Datenblatt für den SiFive-FE310-G002-Mikrocontroller**: https://bit.ly/3bFANak
[15] **Produkt-Webseite PolarFire-SoC**: http://bit.ly/3bFAU5K
[16] **ChaCha-Variante der Salsa20-Stromverschlüsselung, Wikipedia**: http://bit.ly/3rHfZES
[17] **„Stromverschlüsselung“, Wikipedia**: https://bit.ly/336wmQC
[18] **„Imperas RISCV Custom Instruction Flow Application Note“, Imperas Software Limited, Oktober 2019**: http://bit.ly/3vguDVK
[19] **RISC-V - About**: https://riscv.org/about/
[20] **RISC-V - Membership**: https://riscv.org/membership/
[21] **RISC-V-Mitglieder**: https://riscv.org/members/
[22] **GD32 RISC-V-Microcontroller von GigaDevice**: http://bit.ly/2NbSgO9
[23] **Sergei2405, „Russian microcontroller K1986BK025 based on the RISC-V processor core for smart electricity meters“, Habr, Dezember 2020**: http://bit.ly/3qGPY7o
[24] **Webseite MicroMagic**: http://www.micromagic.com
[25] **J. Aufranc, „More Details about Alibaba XT910 64-bit RISC-V Core“, CNX Software, August 2020**: http://bit.ly/3eync6f
[26] **Webseite European Processor Initiative**: www.european-processor-initiative.eu/
[27] **GitHub repository - BOOM RISC-V core**: https://github.com/riscv-boom/riscv-boom

Sommer-Projekte

Von **Dr. Thomas Scherer**

Sechzig Jahre Elektor! 1961 wurde der Elektor-Verlag in den Niederlanden gegründet. Seitdem berichten wir über die neuesten Technologien, Geräte und Bauteile - mit der typischen Mischung aus Theorie und Praxis. Jahrzehntelang gab es im Sommer etwas Besonderes: Das sogenannte Halbleiterheft war eine Sammlung von vielen Projekten; oft waren es mehr als 100 in einer einzigen Ausgabe. Fast alle waren einfach zu bauen und elektronisch nicht allzu kompliziert, und trotzdem sehr, sehr nützlich ...
Hier ist eine kleine Auswahl!

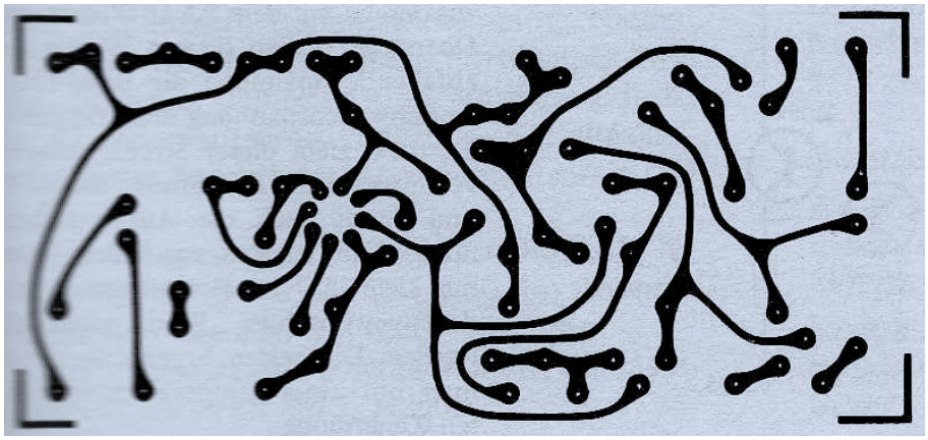


Bild 1. Handarbeit: Die Platine zum „100-W-Endverstärker (40411)“.
www.elektormagazine.de/magazine/elektor-197011/54723

Auch ich bin ein Elektor-Leser der ersten Stunde. Kaum dass ich in den 1970ern gelernt hatte, den Lötkolben nicht am heißen Ende anzufassen, war ich dabei. Was habe ich nicht alles nachgebaut! Am besten gefallen haben mir immer die Halbleiterhefte, und damit bin ich sicher nicht allein, oder? Selbstverständlich konnte man das niemals alles nachbauen. Aber darum ging es auch nicht. Diese Schaltungssammlungen waren nämlich eine einzigartige Fundgrube an Ideen. Viele Elektroniker – auch und gerade in der Industrie – haben sich von den oft unkonventionellen Lösungen inspirieren lassen. Motto: Da stand doch mal was in Elektor…

Verstärker…

Elektor und Edwin – das gehört für altgediente Elektroniker irgendwie zusammen. Die erste deutsche Ausgabe brachte diese simple „Endstufe“, deren besonderes Kennzeichen das Fehlen eines (driftenden) Ruhestroms war. Verstärker kann man ja bekanntlich nie genug haben. Und so kam es, dass in der ersten deutschen Ausgabe des Halbleiterhefts von 1970, das als Ausnahme mitten im Winter erschien, allein acht von 104 Schaltungen Audioverstärker waren [1]. Platinen wurde damals bei Elektor übrigens von Hand gezeichnet: **Bild 1** zeigt das beim „100-W-Endverstärker (40411)“. 100 W waren damals, als 2 x 6 W für die Auszeichnung „High Fidelity“ ausreichten, ein echter Hammer. Die Zahl in Klammern bezog sich auf den verwendeten NPN-Leistungstransistor im modernsten TO3-Gehäuse. Der Verstärker war zudem High-Tech, setzte er doch mit dem 709 schon auf ein Opamp-IC!

Fünf Jahre später war schon richtig Routine eingekehrt und die 103 Schaltungen dieses Halbleiterhefts erschienen bei sommerlichen Temperaturen als Doppelausgabe Juli/August 1975 [2]. Zwar taucht auch hier der Begriff „Verstärker“ in der Liste der Schaltungen gut 18 Mal auf, aber interessant fand ich damals vor allem die Nr. 16, den „Fledermaus-Konverter“ (**Bild 2**). Hier wurde der Ultraschall der Fledermaus mit Hilfe des Signals eines Oszillators auf hörbare Frequenzen „heruntergemischt“. Ein regelrechtes Transistorgrab - typisch für die Elektronik dieser Zeit.

1980 war ich dann in der Redaktion dabei. Als jüngster Elektor-Mitarbeiter verblüffte mich die psychedelisch anmutende Aufmachung das damals aktuellen Halbleiterheftes [3]. **Bild 3** zeigt das Titelblatt der holländischen Ausgabe. Natürlich gab es wieder massenweise Verstärker. Wissen Sie, was eine Kaskode-Schaltung (Nr. 47) ist? Falls nicht: Lesen (für Millenials: Googeln).

Verblüffter als von der Titelblattgestaltung aber war ich von einem anderen Dauerbrenner: Einem Ladegerät für die damals noch erlaubten NiCd-Akkus (**Bild 4**). Das Besondere daran? Es war „netzgespeist“, will meinen: „C-gekoppelt ohne Trafo“. Der damalige Chef-Redakteur Bob van der Horst stand regelrecht auf die Einsparung dieser Blechpakete mit Kupferwicklung. Der Totenkopf in der Schaltung hat seine Berechtigung. Aus heutiger Sicht würde man anmerken: „Don't try this @ home!“

Touch aus den 80ern

Das Phänomen Heim-Automatisierung hat nicht erst im Internet-Zeitalter begonnen. Im Halbleiterheft 1985 ist ein Doppeldimmer mit „Touch“-Bedienung zu finden. Dank der in zwei ICs steckenden Elektronik und einer sachgerechten runden Platine (**Bild 5**) passt diese Beleuchtungssteuerung in die üblichen Installationsdosen für Lichtschalter. Platinen wurden in dieser Zeit mit speziell für Elektor gefertigten Klebesymbolen realisiert.

Auch das Thema Modellbau kam in den Halbleiterheften nicht zu kurz. 1990 gab es zum Beispiel die Schaltung Nr. 13 mit dem bezeichnenden Titel „Schiffsdiesel“. Wie **Bild 6** zeigt, reichen ein Standard-Opamp, zwei Transistoren, ein LDR und ein Glühlämpchen dazu aus, Modellboote mit dem typischen nagelnden Geräusch eines Dieselmotors auszustatten. Der Klang war sogar geschwindigkeitsabhängig.

Mystisches

Eine ganz besonders beliebte Rubrik waren Schaltungen, die eigentlich niemals bis überhaupt nicht funktionieren können. Aber viele Elektroniker schwören drauf. Im Halbleiterheft 1995 erschien ein angeblicher Wasser-Entkalker (**Bild 7**), der ein elektrisches Wechselfeld von 700 Hz mit einer Amplitude von 400 V_{SS} an Wasserrohre anlegt, was angeblich das Entkalken von Haushaltsgerä-

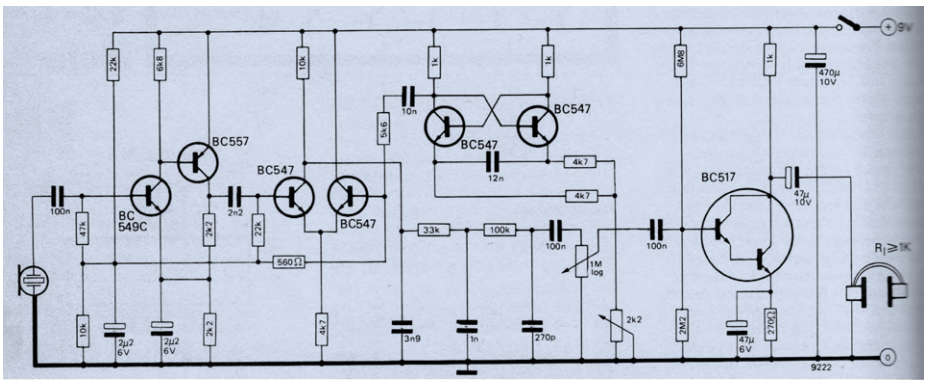


Bild 2. Schaltung des Fledermaus-Konverters.
www.elektormagazine.de/magazine/elektor-197507/55831

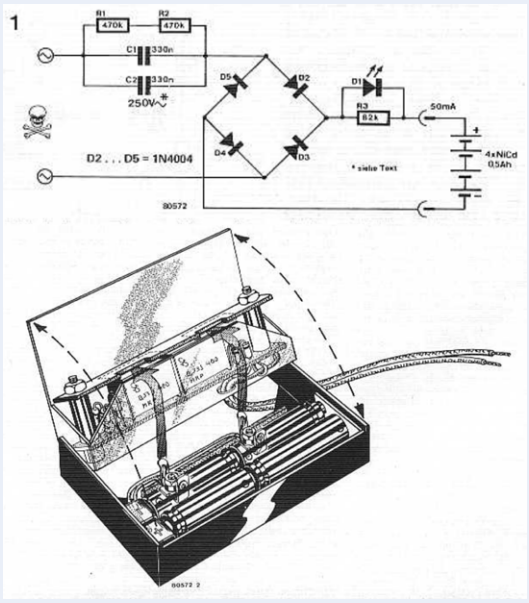


Bild 4. NiCd-Lader ohne Trafo. Unten ist ein zwingend isolierender Gehäusevorschlag zu sehen.
www.elektormagazine.de/magazine/elektor-198007/46122

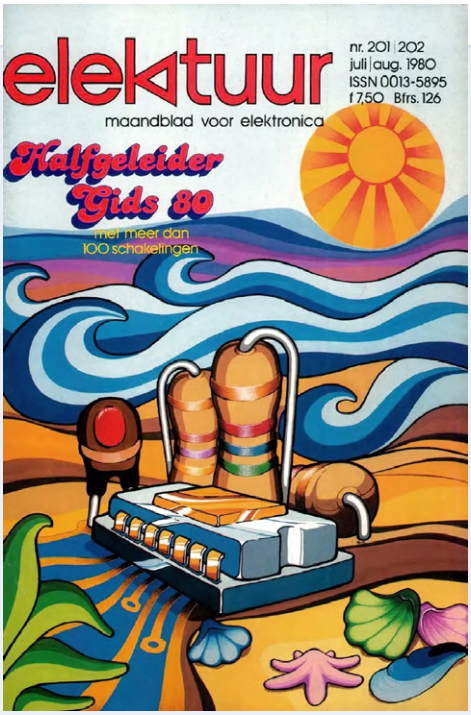


Bild 3. Titelblatt des holländischen Halbleiterhefts 1980.
www.elektormagazine.de/magazine/elektor-198007

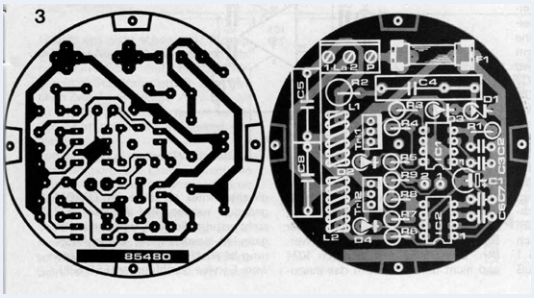


Bild 5. Platine des Zweifach-Dimmers.
www.elektormagazine.de/magazine/elektor-198507/48022

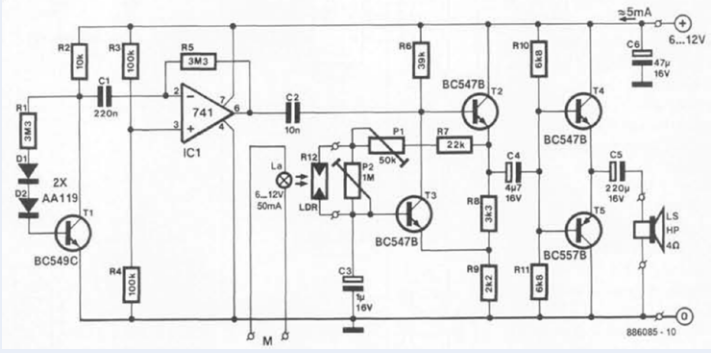


Bild 6. Ein paar Bauteile simulieren einen Schiffsdiesel.
www.elektormagazine.de/magazine/elektor-199007/29287

ten überflüssig machen sollte. Es war nicht der erste Entkalker dieser Art in Elektor. Er wurde tausendfach nachgebaut. In Baumärkten kann man heute sogar ähnliche Fertiggeräte kaufen. Manche davon sollen angeblich sogar durch das Metall der Rohre hindurch funktionieren. Ich weiß nicht, ob es das Fehlen im Physikunterricht war oder die Placebo-Effekt-Elektronik: 50 % der Kollegen von Elektor schwörten jedenfalls Stein und Bein, dass die Schaltung bei ihnen zuhause funktionieren würde!

PC und mehr

Jahrtausendwende: Aus der Reihe „Schaltungen mit fast keinen Bauteilen" stammt der Beitrag „Saft aus dem PC" (**Bild 8**) des langjährigen Elektor-Autors Burkhard Kainka. Man merkte, dass mittlerweile jeder Elektroniker über einen PC verfügte und manche nachgrübelten, was man damit „außerdem" noch anstellen könnte. Hier wird kurz vor der Verdrängung durch USB eine serielle Schnittstelle zum Laden von NiCd-Akkus missbraucht…

Auch das Thema Haus & Garten kam nie zu kurz. Die Terminologie der Schaltung „Schnecken-Firewall" weist nicht nur darauf hin, dass 2006 das Internet schon Alltag war, sondern dass es Elektor auch längst „elektronisch" als PDF-Datei gab, wie man am CAD-gestützt erstellten Schaltplan von **Bild 9** erkennt. Digitale ICs plus Endstufe ergaben einen speziellen Elektrozaun, wirksam gegen Schnecken.

Falls Sie bis jetzt höhere Frequenzen vermisst haben: Selbstverständlich gab es von Anfang an viele HF-Schaltungen und Tools für Amateurfunker. Besonders professionell ist das Elektor-DSP-Radio aus dem Jahr 2010 ausgefallen, dessen Prototyp Sie in **Bild 10** bewundern können. Dieser Selbstbau-Weltempfänger mit USB-Schnittstelle kommt sogar ohne den (gefürchteten) Abgleich aus.

Nur zum Spaß…

Langjährige Elektor-Leser werden an diesem Punkt etwas vermisst haben, nicht wahr? Gemeint sind die Spaß- oder Witzschaltungen. In jedem Halbleiterheft war mindestens eine Schaltung versteckt, die entweder nicht oder nicht so wie beschrieben funktionierte. Oder es kam noch schlimmer: Es wurde gelegentlich etwas vorgestellt, mit dem man seine Zeitgenossen ärgern konnte. Wenn mir jemand die „Elektronische Mücke" [4] aus dem Halbleiterheft 1990 unter die Matratze gejubelt hätte, wäre mein Humor doch arg strapaziert worden. Das war eine echt fiese Idee!

210256-02

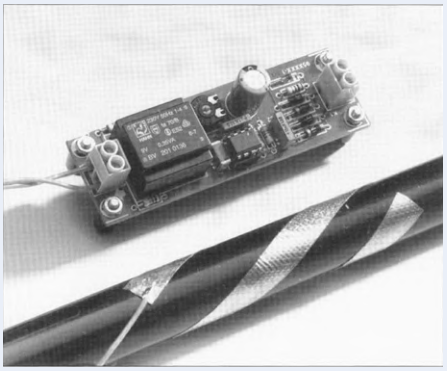


Bild 7. Prototyp des Wasser-Entkalkers.
www.elektormagazine.de/magazine/elektor-199507/30743

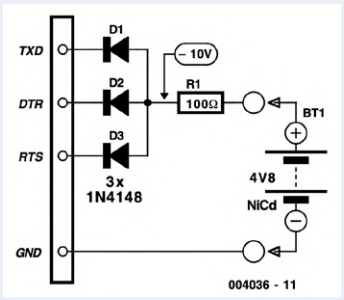


Bild 8. Saft aus dem PC.
www.elektormagazine.de/magazine/elektor-200007/963

Kostenloser Artikel-Download!

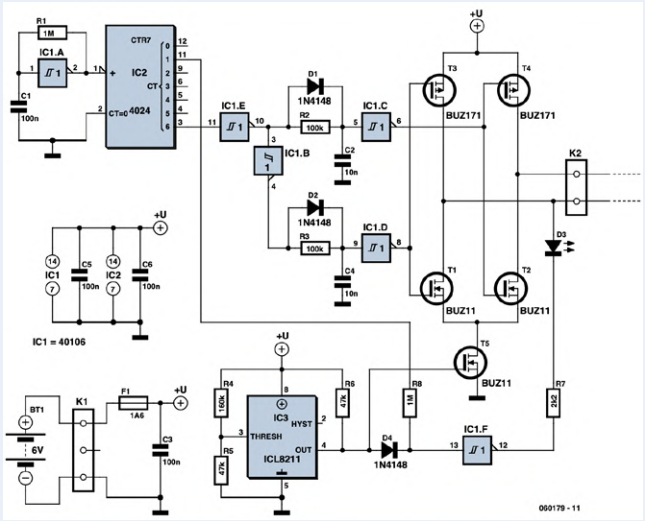


Bild 9. Schnecken-Firewall.
www.elektormagazine.de/magazine/elektor-200607/2418

Nachtrag

Das letzte reguläre Halbleiterheft mit über 100 Schaltungen erschien 2012. Das mag zwar schade sein, aber in den sechs Jahrzehnten seit der Gründung von Elektor ist die Elektronik stetig komplexer und industrieller geworden. „Nur eine Handvoll Bauteile" reicht heute nicht mehr als Begründung für eine Veröffentlichung aus. Darüber hinaus hat Elektor Bereiche wie Laden/Stromversorgung/Dimmen/LEDs/Senden&Empfangen in all den Jahren schon reichlich abgedeckt. Auch hat sich bei vielen Projekten die „Intelligenz" in die Software verlagert; eine Beschreibung benötigt hier einfach mehr Platz, denn wir möchten unsere Leser ja auch an die Installation der Entwicklungsumgebungen, Programmiersprachen und dergleichen mehr heranführen. Das heißt aber nicht, dass wir kleine Schaltungen nicht mehr zu würdigen wissen - ganz im Gegenteil, schließlich lernen hier vor allem Einsteiger eine Menge! Schauen Sie sich bei Gelegenheit doch einmal unsere neue Serie (und viele weitere Schaltungen) auf unserer Webseite an:
www.elektormagazine.de/tags/schaltung

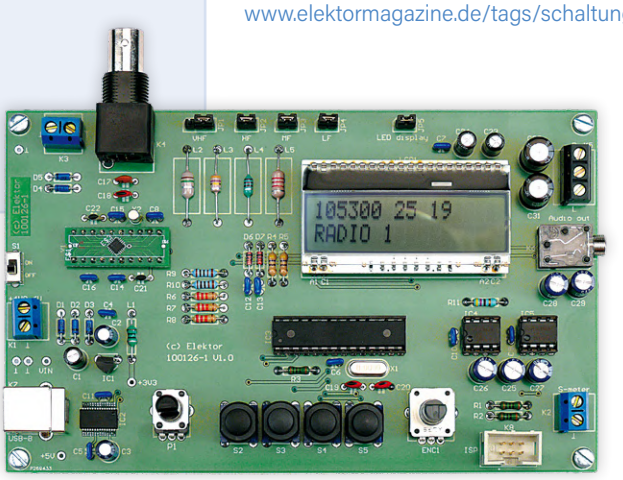


Bild 10. Das Elektor-DSP-Radio.
www.elektormagazine.de/magazine/elektor-201007/3531

Ein Beitrag von
Autor: **Dr. Thomas Scherer**
Redaktion: **Jens Nickel**
Layout: **Harmen Heida**

WEBLINKS

[1] **Elektor Ausgabe 4/1970**:
www.elektormagazine.de/magazine/elektor-197011

[2] **Elektor Ausgabe 7-8/1975**:
www.elektormagazine.de/magazine/elektor-197507

[3] **Elektor Ausgabe 7-8/1980**:
www.elektormagazine.de/magazine/elektor-198007

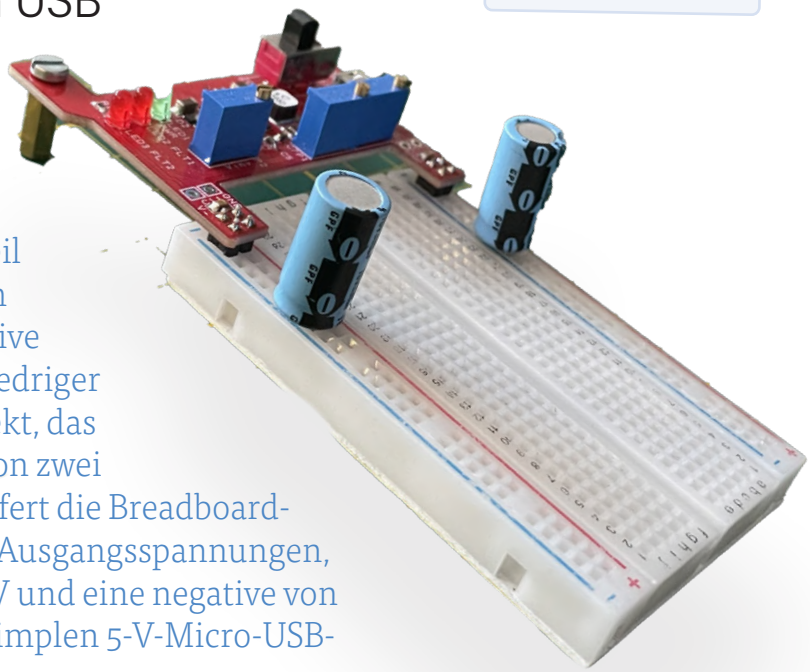
[4] **Elektronische Mücke**:
www.elektormagazine.de/magazine/elektor-199007/29328

Vielseitige Spannungsversorgung für breadboards

Positive und negative 5-V-Ausgangsspannungen von USB

Von **Fons Janssen** (Niederlande)

Breadboard-Stromversorgungen sind nützlich, wenn kein Desktop-Labornetzteil zur Verfügung steht. Die meisten auf dem Markt erhältlichen Module sind auf positive Ausgangsspannungen beschränkt, die niedriger als die Eingangsspannung sind. Das Projekt, das wir hier vorstellen, ist anders. Mit Hilfe von zwei integrierten Schaltkreisen von Maxim liefert die Breadboard-Stromversorgung bis zu vier einstellbare Ausgangsspannungen, drei positive im Bereich von 0,6 V bis 20 V und eine negative von –1,8 V bis –11 V. Und das alles aus einem simplen 5-V-Micro-USB-Anschluss!



PROJEKT-INFOS

Tags
Breadboard, Stromversorgung, Prototyping

Level
Experte

Zeit
etwa 1 Stunde

Werkzeuge
Heißluft-Lötstation oder Reflow-Ofen, Lötpaste

Kosten
etwa 20 €

Ein Breadboard ist nicht nur ein praktisches Werkzeug für erfahrene Elektroniker, die schnell etwas testen möchten, sondern auch für Anfänger, Bastler oder Studenten, die ihre ersten Schritte in die Welt der Elektronik machen. Für den Aufbau einer Schaltung ist kein Löten erforderlich und das Austauschen von Bauteilen oder Verdrahtungen ist sehr einfach. Da jede Schaltung auf die eine oder andere Weise mit Strom versorgt werden muss, haben wir eine vielseitige Stromversorgung entwickelt, die einfach auf einem Breadboard verwendet werden kann.

Obwohl ein Labornetzteil die bevorzugte Stromquelle beim Experimentieren mit Elektronik oder beim Prototyping ist, ist ein solches Gerät nicht immer verfügbar. Andererseits gibt es überall 5-V-Netzteile mit Micro-USB-Steckern. Wir alle haben wahrscheinlich mehr davon herumliegen, als wir für die Stromversorgung oder das Aufladen unserer Mobiltelefone und anderer elektronischer Geräte und Gadgets benötigen. Die meisten dieser Adapter sind kurzschlussfest, und in Kombination mit dem hier vorgestellten Projekt haben Sie die perfekte, kostengünstige Lösung für die Stromversorgung von Schaltungen auf Breadboards. Die Leistung ist begrenzt, aber wie wir alle wissen *sollten*: Breadboards sind nicht für High-Power-Schaltungen ausgelegt und auch definitiv nicht dafür geeignet.

Es gibt viele Breadboard-Stromversorgungsplatinen auf dem Markt, aber im Gegensatz zu dieser sind die meisten auf Ausgangsspannungen beschränkt, die niedriger als die Eingangsspannung sind. Negative oder symmetrische Ausgänge sind noch seltener bis nicht vorhanden. Die hier vorgestellte Schaltung ist in dieser Hinsicht eine Neuheit. Sie ist so konzipiert, dass Sie aus einem Standard-5-V-Netzteil schnell eine negative oder symmetrische Spannungsversorgung zum Beispiel für eine Opamp-Verstärkerschaltung mit +9 V/-9 V oder so ähnlich erzeugen können. Die positive Ausgangsspannung darf sogar bis zu 20 V betragen! Sie können aber auch niedrige 3,3 V für Ihren Mikrocontroller herstellen. Es ist alles einstellbar! Die Breadboard-Stromversorgungsplatine enthält nur zwei ICs und ist so ausgelegt, dass sie vier Ausgangsspannungen liefert: eine negative und drei positive.

Das Schaltbild

Bild 1 zeigt das Schaltbild der Breadboard-Stromversorgung. Das Herz der Schaltung ist der Maxim MAX8614B [1]. Er ist mit einem Aufwärtswandler ausgestattet, der eine Spannung (V_{BST}) erzeugen kann, die höher als die Eingangsspannung ist, und einem invertierenden

Abwärtswandler, der eine negative Spannung (V_{INV}) erzeugen kann. Das Poti P1 befindet sich in der Rückkopplung des positiven Aufwärtswandlers und erlaubt die Einstellung der Ausgangsspannung zwischen:

$$\left(\frac{120k}{33k}+1\right)\times 1,01V \approx +5V \quad und \quad \left(\frac{620k}{33k}+1\right)\times 1,01V \approx +20V$$

Der invertierende Abwärts/Aufwärtswandler hat ebenfalls ein Poti (P2) in seiner Rückkoppplung, mit dem die Ausgangsspannung eingestellt werden kann zwischen:

$$-\frac{39k}{27k}\times 1,25V \approx\sim -1,8V \quad und \quad -\frac{239k}{27k}\times 1,25V \approx\sim -11V$$

Die maximale Ausgangsleistung, die die Wandler liefern können, beträgt etwa 2 W für den Aufwärtswandler und 1 W für den Abwärtswandler. Das bedeutet, dass bei höheren Ausgangsspannungen der maximale Ausgangsstrom niedriger ist. Man sollte nicht vergessen, dass auch die Restwelligkeit bei höheren Lasten zunimmt. Daher empfiehlt es sich, einen Elektrolytkondensator von einigen hundert Mikro-Farad auf die Stromschienen des Breadboards zu stecken. Achten Sie auf die Polarität dieses Kondensators und kontrollieren Sie, ob er auch für die zu erwartende Spannung geeignet ist!

Das zweite IC in diesem Schaltplan ist der MAX38903 (ebenfalls von Maxim, siehe [2]) ist ein linearer Spannungsregler (LDO), dessen Ausgangsspannung mit P3 auf eingestellt werden kann zwischen:

$$\left(\frac{0k}{68k}+1\right)\times 0,6V \approx +0,6V \quad und \quad \left(\frac{500k}{68k}+1\right)\times 0,6V \approx +5V$$

Bei einem LDO kann die Ausgangsspannung nicht höher sein als die Eingangsspannung. Die maximale Ausgangsspannung ist deshalb abhängig von der Spannung des Netzteils (hier 5 V), den Verlusten in den Kabeln/Steckern und der Drop-Out-Spannung des MAX38903. In der Praxis liegt dieses Maximum bei etwa 4,5 V. Der maximale Strom, den dieser LDO verarbeiten kann, beträgt 1 A. Die meisten Adapter können dies liefern, aber bei niedrigeren Ausgangsspannungen steigt die Verlustleistung im Chip deutlich an:

$$P_{diss} = (5V - V_{out}) \times I_{out}$$

Diese Verlustleistung sollte 2 W nicht überschreiten, so dass 1 A nur bei Ausgangsspannungen höher als 3 V geliefert werden kann. Glücklicherweise ist der MAX38903 mit verschiedenen Schutzschaltungen ausgestattet, so dass wir uns keine Sorgen machen müssen, dass unser Netzteil in Rauch aufgeht, weil die Leistungsgrenzen überschritten werden. Die Schaltung auf dem Breadboard selbst erfolgt natürlich auf eigene Gefahr.

Montage

Die Gerberdateien für die Bestellung der Leiterplatte sind in **Bild** 2 dargestellt. Die DesignSpark-Designdateien stehen unter [3] zum Download bereit.

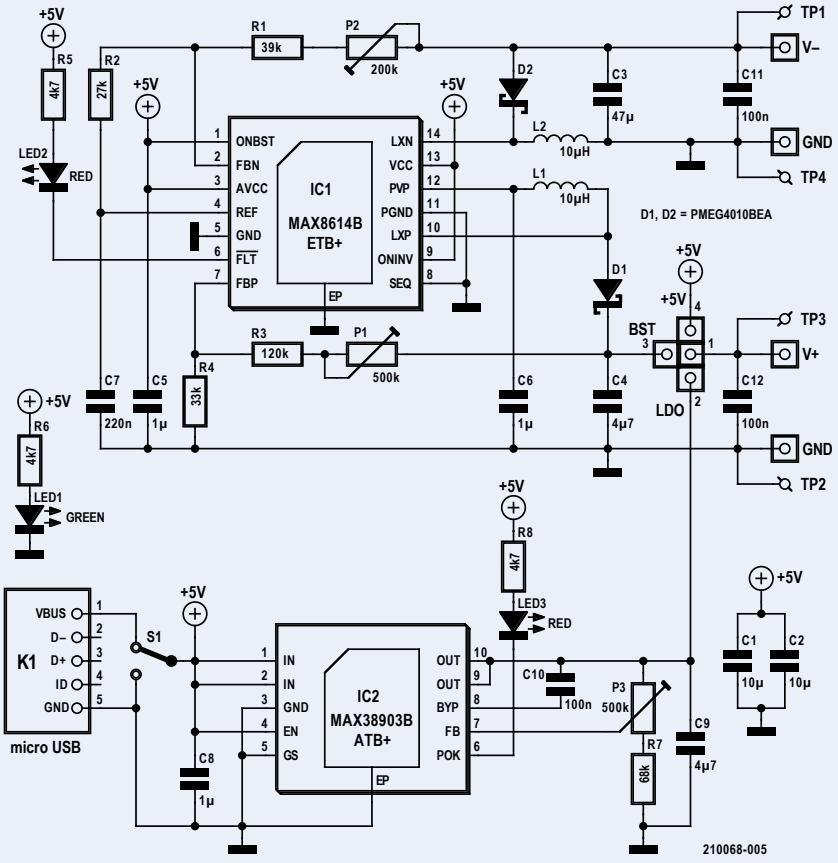


Bild 1. Schaltplan der Breadboard-Stromversorgung.

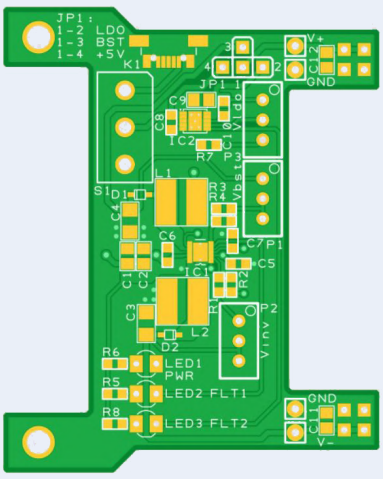


Bild 2. Die Design- und Gerberdateien für das Platinen-Layout können heruntergeladen werden [3].

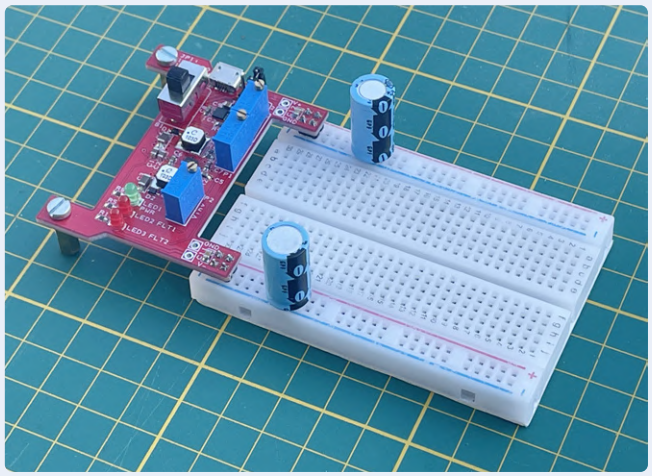


Bild 3. Die Stromversorgung auf einem MB102-Breadboard mit zusätzlichen Elektrolytkondensatoren.

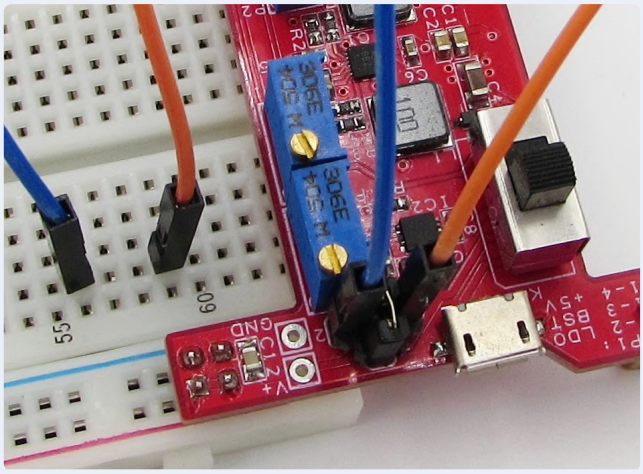


Bild 4. Positive Schiene gebrückt auf V_{BST}, die Brückendrähte verbinden V_{LDO} und +5 V mit dem Breadboard.

Die Platine wurde so kompakt wie möglich gehalten. Am besten werden zunächst die SMDs verlötet, beginnend mit den ICs. Das Löten dieser winzigen Bauteile in ihren TDFN-Gehäusen ist eine ziemliche Herausforderung und der Bau ist unerfahrenen Bastlern sicher nicht zu empfehlen. Es gibt sicher Lötgenies, die die Pads an den Seiten der Gehäuse mit einem kleinen Lötkolben löten können, aber das Löten der freiliegenden Pads darunter erfordert eine Heißluftstation oder einen Reflow-Ofen. Lötpaste wird bevorzugt, auch wenn es dem Entwickler gelungen ist, seine Prototypen so zu bauen, dass er die Pads mit normalem Lötdraht verzinnte, zusätzliches Flussmittel auftrug und eine Heißluftstation zum Reflowing verwendete. Eine Lupe, ein Digitalmikroskop oder sogar ein Stereomikroskop sind eine große Hilfe, um die Lötstellen zu prüfen, bevor Sie mit den anderen Bauteilen fortfahren. Das anschließende Löten des Micro-USB-Verbinders scheint da weniger anspruchsvoll zu sein.

Als Nächstes können die umgebenden Spulen, Dioden, Widerstände und Kondensatoren montiert werden und schließlich die Durchsteck-Bauteile wie der Schalter, die LEDs und die Stiftleisten, die mit einem normalen Lötkolben leicht zu befestigen sind. Für JP1 werden die beiden äußeren Pins einer der beiden Reihen einer 3 x 2-poligen Stiftleiste abgeschnitten, so dass eine dreieckige, 4-polige Jumper-Konfiguration entsteht.

Verwendung

Die Stromversorgung kann wie in **Bild 3** auf das Breadboard gesteckt werden, die Steckverbinder J1 und J2 in die horizontalen Stromschienen des Breadboards. Die obere Schiene wird dann mit V+, die untere mit V- und die beiden inneren Stromschienen mit GND verbunden (alle Stromversorgungen dieses Boards haben eine gemeinsame Masse). Die Platine kann auf der linken Seite mit zwei 12-mm-M3-Abstandshaltern oder Schrauben auf dem Labortisch abgestützt werden; dazu sind zwei Befestigungslöcher vorhanden. Dies verbessert die mechanische Stabilität der Verbindung zwischen Netzteil und Breadboard und verhindert ein Kippen der Platine beim Einstellen der Ausgangsspannungen. Mit dem Schalter S1 kann die gesamte Spannungsversorgung ein- und ausgeschaltet werden. Beim Einschalten des Netzteils leuchtet die LED1 als Power-On-Anzeige grün.

Da es drei positive Versorgungsspannungen auf der Platine gibt, muss der Anwender auswählen, welche an Jumper JP1 mit der oberen Versorgungsschiene des Breadboards (V+) verbunden ist. Stecken Sie den Jumper in Position 1-2 für die niedrige einstellbare Spannung (V_{LDO}), in Position 1-3 für die hohe einstellbare Spannung (V_{BST}) und in Position 1-4 für die feste +5 V. Die Jumperpositionen sind auch im Bestückungsaufdruck der Platine markiert. Die beiden verbleibenden Spannungen können noch mit Hilfe von Steckbrücken an die Schaltung auf dem Breadboard angeschlossen werden, wie in **Bild 4** gezeigt. Auf diesem Bild ist V_{BST} mit der positiven Schiene gebrückt, während die orangefarbenen und blauen Brückendrähte die +5 V und V_{LDO} mit dem Breadboard verbinden. Die negative Ausgangsspannung V_{INV} ist direkt mit der unteren Versorgungsschiene (V-) verbunden.

Die Spannungen werden mit den Mehrgang-Potis P1 bis P3 eingestellt. Der Bestückungsaufdruck der Platine zeigt, welcher Trimmer welchem Ausgang zugeordnet ist. Zum Messen von V+ (TP3) und V- (TP1) während des Abgleichs sind Messpunkte für ein Multimeter vorhanden. TP2 und TP4 sind mit GND verbunden.

Es gibt zwei rote LEDs, die eine Überlast oder einen Kurzschluss einer der Stromversorgungen anzeigen: LED2 (auf der Platine mit FLT1 gekennzeichnet) für V_{BST} und/oder V_{INV} und LED3 (FLT2) für V_{LDO}. Der Aufwärtswandler und der LDO begrenzen den Ausgangsstrom bei Überlast und die Ausgänge werden abgeschaltet, wenn thermische Grenzwerte der Chips überschritten werden. In beiden

WEBLINKS

[1] **Maxim Integrated, „MAX8614A/MAX8614B: Dual-Output (+ und -) DC-DC-Converters for CCD“, 12/2019**: https://datasheets.maximintegrated.com/en/ds/MAX8614.pdf

[2] **Maxim Integrated, „MAX38903A/MAX38903B/MAX38903C/MAX38903“, 4/2020**: https://datasheets.maximintegrated.com/en/ds/MAX38903A-MAX38903D.pdf

[3] **Projekt-Download**: https://elektormagazine.de/210068-02

Fällen leuchten die LEDs. Wenn der Abwärtswandler überlastet ist, werden beide Ausgänge von IC1 abgeschaltet und LED2 leuchtet. Ein Einschalten der Versorgung mit S1 setzt diesen Fehlerzustand zurück, die Ausgangsspannungen werden wiederhergestellt und die rote LED erlischt wieder, vorausgesetzt natürlich, es liegt kein Kurzschluss vor. Es gibt jedoch keine Überlastanzeige für die festen +5 V. In diesem Fall vertrauen wir darauf, dass das Netzteil kurzschlusssicher ist.
Obwohl dieses Projekt für den Einsatz auf Breadboards konzipiert und gedacht ist, kann es natürlich auch als einfache, stromsparende Stromversorgung für andere Anwendungen genutzt werden, indem man Drähte zwischen den Ausgangspins und anderen elektronischen (Prototyping-) Schaltungen anschließt. So können Sie das überflüssige 5-V-Micro-USB-Steckernetzteil auf dem Labortisch sinnvoll einsetzen!

210068-02

Dieses Projekt kann auch auf der Elektor-Labs-Website eingesehen und verfolgt werden: www.elektormagazine.com/labs/breadboard-power-supply

Haben Sie Fragen oder Kommentare ?
Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Dann schicken Sie bitte eine E-Mail an die Redaktion unter luc.lemmens@elektor.com oder kontaktieren Sie Elektor unter editor@elektor.com.

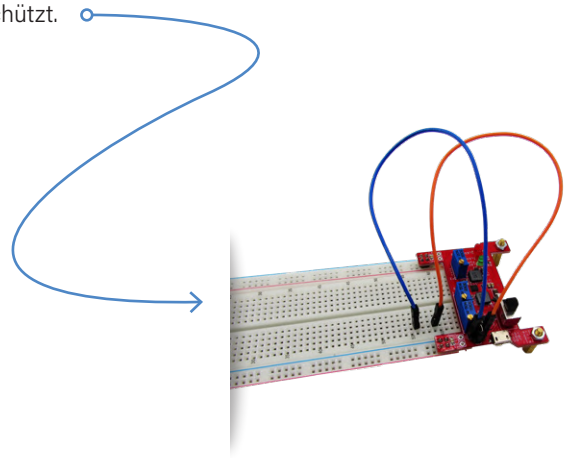
Ein Beitrag von
Design und Text: **Fons Janssen**
Redaktion: **Luc Lemmens**
Illustrationen: **Patrick Wielders, Fons Janssen, Luc Lemmens**
Layout: **Harmen Heida**
Übersetzung: **Textmaster**

Passende Produkte

- **Breadboard (830 Anschlusspunkte)**
  www.elektor.de/breadboard-830-tie-points
- **Andonstar AD407 HDMI-Digitalmikroskop mit 7"-LCD-Bildschirm**
  www.elektor.de/andonstar-ad407-hdmi-digital-microscope-with-7-lcd-screen

Technische Daten

- Passt genau auf MB102-Breadboards.
- Eine einstellbare negative Spannung von -1,8 V bis -11 V (V_{INV}).
- Eine einstellbare positive Spannung von +0,6 V bis +5 V (V_{LDO}).
- Eine einstellbare positive Spannung zwischen +5 V und +20 V (V_{BST}).
- Eine feste positive Spannung von +5 V (von der Spannungsversorgung).
- Kann mit einem handelsüblichen 5-V-Telefonladegerät mit Micro-USB-Anschluss betrieben werden.
- Ausgänge (außer +5 V) gegen Überlast und Kurzschluss geschützt.



STÜCKLISTE

Widerstände:
(Größe 0603, wenn nicht anders angegeben)
R1 = 39 k
R2 = 27 k
R3 = 120 k
R4 = 33 k
R5, R6, R8 = 4k7
R7 = 68 k
P1, P3 = 500 k Mehrgang-Trimmpoti (Vishay Typ 24 W)
P2 = 200 k Mehrgang-Trimmpoti (Vishay Typ 24 W)

Induktivitäten:
L1, L2 = 10 µ 1 A (Fastron 242408FPS)

Kondensatoren:
C1, C2 = 10 µ, 16 V X5R Größe 0805
C3, C4 = 4µ7, 50 V X7R Größe1206
C5, C6, C8 = 1 µ, 16 V X7R Größe 0603
C7 = 220 n, 16 V X7R Größe 0603
C9 = 4µ7, 25 V X5R Größe 0805
C7 = 220 n, 16 V X7R Größe 0603
C11, C12 = 100 n, 50 V X7R Größe 0805

Halbleiter:
D1, D2 = Schottky-Diode 40 V/1 A PMEG4010, SOD-323
LED1 = Low-current-LED 3 mm, grün
LED2, LED3 = Low-current-LED 3 mm, rot
IC1 = Dual-Output (+ und -) DC-DC-Wandler MAX8614BETD+
IC2 = 1,7 -5,5 VIN, 1 A einstellbarer LDO MAX38903BATB+

Außerdem:
J1,J2 = 2x2-polige Stiftleiste (Raster 2,54 mm)
JP1 = 2x3-polige Stiftleiste (Raster 2,54 mm, siehe Text)
K1 = Micro-USB-Verbinder, Amtek MIUSB-F5M-AGB-U
S1 = Schiebeschalter SDPT, NKK Switches CS12ANW03
Jumper für JP1

Raspberry Pi Pico Essentials

Ein Beispiel-Kapitel: WLAN mit dem Raspberry Pi Pico

Von **Dogan Ibrahim** (Großbritannien)

In dieser Ausgabe von Elektor Books stellen wir Ihnen einen Kapitelteil aus Dogan Ibrahims Buch *Raspberry Pi Pico Essentials* vor, das im März 2021 bei Elektor erschien und inzwischen ein Bestseller ist. Damals war es das erste Buch eines Drittanbieters, das den praxiserprobten Umgang mit dem RP2040-Mikrocontroller auf dem Raspberry Pi Pico-Modul samt Software zum Mitnehmen zeigte. Ist „der Pico" denn so zugänglich, wie behauptet wird, und wird er den Erwartungen gerecht, ein „Drop-in"-Controller-Board zu sein? Das wollen wir in diesem Artikel herausfinden!

Anmerkung der Redaktion: Dieser Artikel ist ein Auszug aus dem Kapitel 10 des Buchs *Raspberry Pi Pico Essentials*, neu formatiert und leicht bearbeitet, um den Standards und dem Seitenlayout der Zeitschrift *Elektor* zu entsprechen. Da es sich um einen Auszug aus einer größeren Publikation handelt, können sich einige Begriffe in diesem Artikel auf Diskussionen an anderer Stelle des Buchs beziehen. Der Autor und die Redaktion haben jedoch ihr Bestes getan, um solche Fälle zu vermeiden, und helfen bei Rückfragen gerne weiter. Kontaktinformationen finden Sie im Kasten Fragen oder Kommentare?

In diesem Artikel werden wir ein Projekt entwickeln, das eine WLAN-Verbindung nutzt, um die Kommunikation zwischen dem Raspberry Pi Pico und einem Smartphone herzustellen.

WLAN-Steuerung einer LED von einem Smartphone aus

Beschreibung: In diesem Projekt werden wir Befehle von einem Mobiltelefon über die WLAN-Verbindung senden, um eine LED zu steuern, die mit dem Raspberry Pi Pico verbunden ist. Die LED kann beispielsweise durch ein Relais ersetzt werden, um ein Gerät zu schalten. Die Befehle müssen mit einem Return (CR/LF oder newline) abgeschlossen werden. Gültige Befehle sind:

`LON`	LED einschalten
`LOFF`	LED ausschalten

Ziel: Das Ziel dieses Projekts ist es, die Verwendung der WLAN-Konnektivität auf dem Raspberry Pi Pico zu demonstrieren.

Pico WLAN-Konnektivität: Der Raspberry Pi Pico verfügt über kein eingebautes WLAN-Modul und kann daher nicht direkt mit einem WLAN verbunden werden. Dazu ist ein externes WLAN-Modul erforderlich. Die vielleicht einfachste und billigste Möglichkeit, dem Pico WLAN-Fähigkeit zu verleihen, ist eine ESP-01-Prozessorplatine. Das winzige Board (siehe **Bild 1**), das nur 2,7 cm × 1,2 cm misst, basiert auf einem ESP8266-Prozessor und kostet etwa 4...5 €. Der ESP-01 besitzt die folgenden zu Experimenten motivierenden Eigenschaften:

- Betriebsspannung: +3,3 V
- Schnittstelle: mit einfachen AT-Befehlen über serielle Schnittstelle/UART
- Integrierter TCP/IP-Protokollstack
- 802.11 b/g/n
- Keine externen Bauteile erforderlich

Das ESP-01 kommuniziert mit dem Host-Prozessor über die TX- und RX-Pins der seriellen Schnittstelle. Es handelt sich um eine 8-polige Platine mit der folgenden Pinbelegung:

VCC:	+3,3-V-Spannungsversorgung
GND:	Masse der Spannungsversorgung

GPIO0: I/O-Pin, der im normalen Betrieb an +3,3 V und für das Hochladen von Firmware auf den Chip an GND gelegt sein muss
GPIO2: Allzweck-I/O-Pin
RST: Reset-Pin, der im Normalbetrieb an +3,3 V angeschlossen werden muss
CH_PD: Enable-Pin, der im Normalbetrieb an +3,3 V angeschlossen werden muss
TX: Serieller Ausgangspin
RX: Serieller Eingangspin

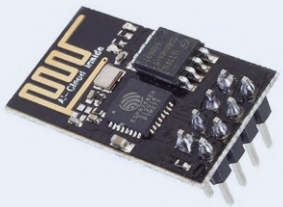
Die Pins des ESP-01 sind mechanisch nicht mit einem Standard-Breadboard-kompatibel, daher wird ein Adapter benötigt, wenn die Platine auf einem Steckboard befestigt werden soll (**Bild 2**).

Blockschaltbild: Bild 3 zeigt das Blockschaltbild des Projekts.

Schaltplan: Bild 4 zeigt den Schaltplan des Projekts. Für die Kommunikation mit dem ESP-01 werden die UART0-Pins TX und RX des Raspberry Pi Pico verwendet.

Programmlisting: Listing 1 zeigt das Listing des Programms **Picowifi**. Es ist in der Sammeldatei enthalten, die unter *Downloads* auf der Elektor-Webseite zum Buch zu finden ist [1]. In der Setup-Routine wird die serielle Kommunikationsgeschwindigkeit auf 115.200 eingestellt, was der Standard-Baudrate des SP-01 entspricht. Der LED-Pin wird als Ausgang konfiguriert und ausgeschaltet. Die Funktion `ConnectToWiFi` wird aufgerufen, um eine Verbindung mit dem lokalen WLAN-Router herzustellen. Es werden AT-ähnliche Befehle verwendet, um das ESP-01-Board für die Verbindung mit dem WLAN-Router zu konfigurieren.
Der Rest des Programms läuft in einer Endlosschleife, die mit einer `while`-Anweisung erzeugt wird. Innerhalb dieser Schleife werden Daten vom Smartphone empfangen und die LED entsprechend gesteuert. Die Befehle `LON` und `LOFF` schalten die LED ein und aus. Die Datenpakete werden mit der Funktion `readline` vom Smartphone empfangen. Die Funktion `find` sucht nach einer Teilzeichenkette in einem String und gibt einen Wert ungleich null zurück, wenn die Teilzeichenkette gefunden wird. Die Funktion `find` erkennt, wenn die Daten, die vom mobilen Gerät empfangen werden, das folgende Format haben: `+ID0,n: data` (zum Beispiel `+ID0,3:LON`), wobei `0` die Link-ID und `n` die Anzahl der empfangenen Zeichen ist. Mit der Funktion `find` können wir also einfach nach den Zeichenketten `LON` oder `LOFF` im empfangenen Datenpaket suchen.
Die Funktion `ConnectToWiFi` sendet die folgenden Befehle an den ESP-01, um eine Verbindung zum WLAN herzustellen:

AT+RST ESP-01 zurücksetzen
AT+CWMODE stellt den ESP-01-Modus ein (hier auf Station-Modus)
AT+CWJAP WLAN-SSID und Passwort einstellen
AT+CPIMUX stellt den Verbindungsmodus ein (hier auf mehrere Verbindungen)
AT+CIFSR gibt die IP-Adresse zurück (hier nicht verwendet)
AT+CIPSTART stellt den TCP- oder UDP-Verbindungsmodus, die Ziel-IP-Adresse und die Portnummer ein. Hier wird UDP verwendet und die Portnummer auf 5000 eingestellt. Die Ziel-IP-Adresse ist auf „0.0.0.0“ eingestellt, so dass jedes Gerät Daten senden kann, solange es



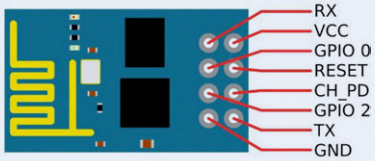


Bild 1. ESP-01-Prozessorplatine mit Anschlussbelegung.

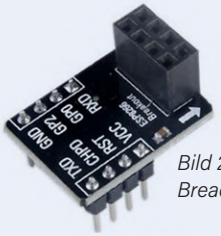


Bild 2. Das ESP-01-Breadboard-Adapter.

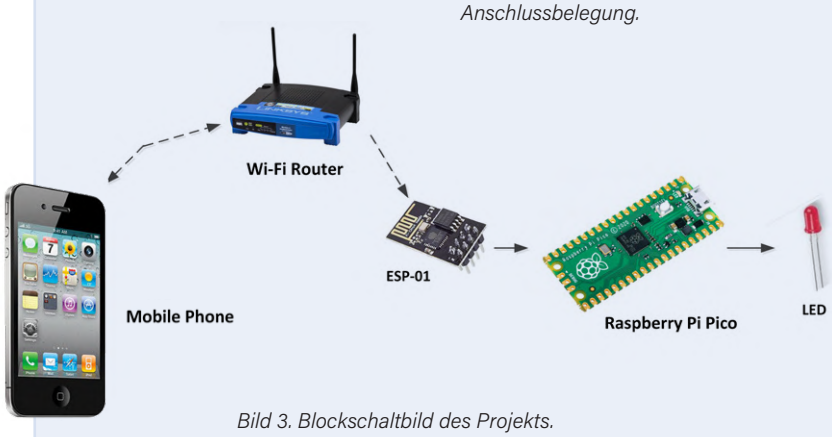


Bild 3. Blockschaltbild des Projekts.

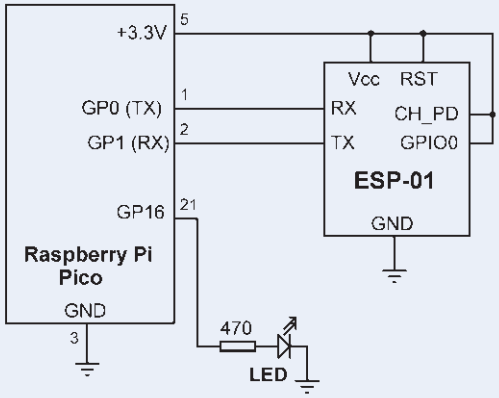


Bild 4. Schaltplan des Projekts.

Listing 1. Python-Programm zur Steuerung eines ESP-01 als WLAN-Modul

```
#------------------------------------------------------------
#           USING WI-FI
#           ===========
#
# In this project a ESP-01 chip is connected to the Raspberry
# Pi Pico. This chip is used to connect the Pico to the Wi-Fi
#
# Author: Dogan Ibrahim
# File  : Picowifi.py
# Date  : February 2021
#------------------------------------------------------------
from machine import Pin, UART
import utime
uart = UART(0, baudrate=115200,rx=Pin(1),tx=Pin(0))

LED = Pin(16, Pin.OUT)
LED.value(0)

#
# Send AT commands to ESP-01 to connect to local WI-Fi
#
def ConnectToWiFi():
    uart.write("AT+RST\r\n")
    utime.sleep(5)

    uart.write("AT+CWMODE=1\r\n")
    utime.sleep(1)

    uart.write('''AT+CWJAP="BTHomeSpot-XNH","49345xyzpq"\r\n''')
    utime.sleep(5)

    uart.write("AT+CPIMUX=0\r\n")
    utime.sleep(3)

    uart.write('''AT+CIPSTART="UDP","0.0.0.0",5000,5000,2\r\n''')
    utime.sleep(3)

ConnectToWiFi()

#
# Main program loop
#
while True:
    buf = uart.readline()          # Read data
    dat = buf.decode('UTF-8')      # Decode
    n = dat.find("LON")            # Includes LON?
    if n > 0:
        LED.value(1)               # LED ON
    n = dat.find("LOFF")           # Includes OFF?
    if n > 0:
        LED.value(0)               # LED OFF
```

Port 5000 verwendet. Sie können diesen Eintrag auf die IP-Adresse Ihres Smartphones ändern, damit nur Daten von Ihrem Telefon empfangen werden.

Beachten Sie, dass nach jedem Befehl eine kurze Verzögerung eingefügt wird. Der Befehl `AT+CWJAP` erfordert eine längere Verzögerung. Das Programm kann leicht modifiziert werden, indem die Verzögerungen entfernt und stattdessen die Antworten des ESP-01 überprüft werden. Auf diese Weise wird das Programm fortgesetzt, sobald die richtige Antwort eingetroffen ist. Eventuell müssen Sie einen Hardware-Reset des ESP-01 durchführen, also das Gerät aus- und wieder einschalten, bevor Sie das Programm starten.

Testen des Programms

Das Programm kann einfach mit dem Programm *PacketSender* (**Bild 5**) auf dem PC oder mit einem Android-Smartphone nach der Installation einer UDP-Server-App getestet werden. Von den vielen frei verfügbaren UDP-Apps im Play Store installieren und verwenden Sie *UDP/TCP Widget* von K.J.M (**Bild 6**).

Die Schritte zum Testen des Programms sind:

- Bauen Sie die Schaltung auf.
- Laden Sie das Programm auf Ihren Raspberry Pi Pico.
- Starten Sie die App *UDP/TCP Widget* auf Ihrem Mobiltelefon.
- Klicken Sie auf das Zahnradsymbol und stellen Sie das Protokoll auf *UDP*, die *IP-Address* auf die IP-Adresse Ihres Raspberry Pi Pico (*192.168.1.160* beim Pico des Autors) und den *Port* auf *5000* ein, wie in **Bild 7** gezeigt.
- Klicken Sie auf den Menüpunkt *MESSAGE*, wählen Sie *Text (UTF-8)* als Format und geben Sie den Befehl `LON` ein, um die LED einzuschalten. Wählen Sie *LF\n* als *Terminator* und klicken Sie auf das OK-Symbol (Häkchen), wie in **Bild 8** dargestellt.
- Klicken Sie nun auf *SEND* (**Bild 9**), um den Befehl an den Raspberry Pi Pico zu senden. Sie sollten die Meldung *Packet Sent* am oberen Rand Ihres Android-Bildschirms vorübergehend angezeigt sehen.

Die IP-Adresse des ESP-01 kann durch Scannen aller Geräte im lokalen WLAN-Router ermittelt werden. Sie können beispielsweise die Android-App „Who Uses My WiFi - Network Scanner“ von Phuongpn verwenden, um die IP-Adressen aller mit Ihrem Router verbundenen Geräte zu sehen. Der ESP-01 wird wie in **Bild 10** gezeigt mit dem Namen *Espressif* aufgelistet (IP: 192.168.1.160).

210198-02

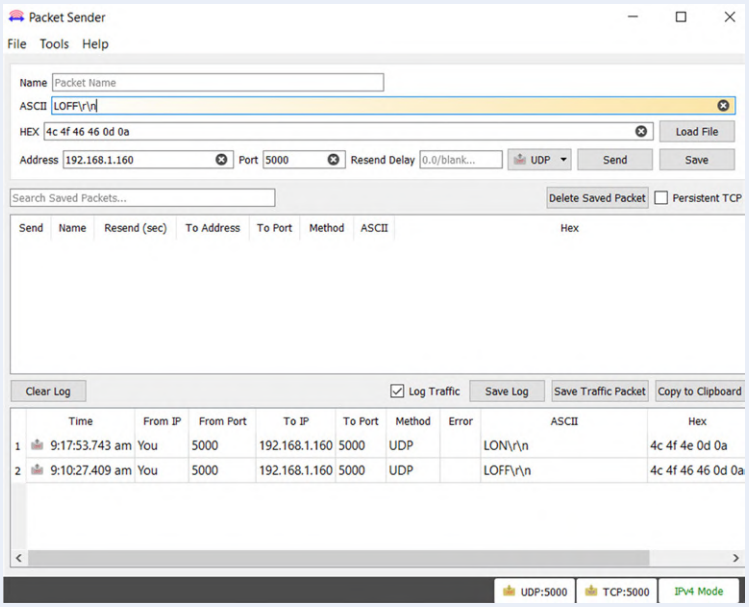


Bild 5. Der PacketSender wird zum Testen des Programms verwendet.

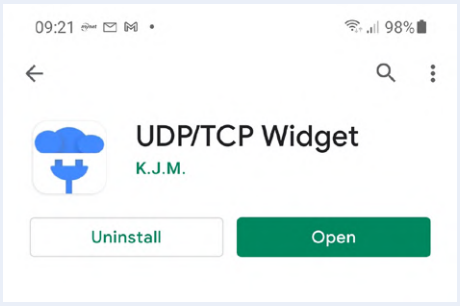


Bild 6. UDP/TCP-Widget-App für Android.

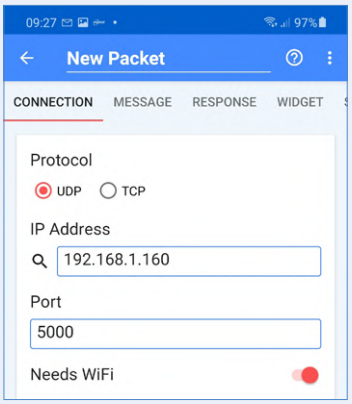


Bild 7. Konfigurieren der App.

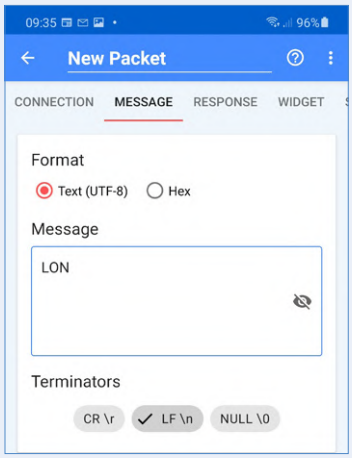


Bild 8. Befehl zum Einschalten der LED.

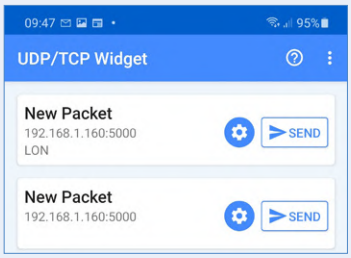


Bild 9. Klicken Sie auf SEND, um den Befehl zu übertragen.

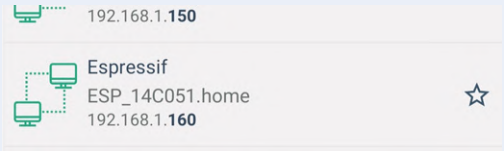


Bild 10. Ermitteln der IP-Adresse des ESP-01.

Passende Produkte

› Buch: „Raspberry Pi Pico Essentials“
www.elektor.de/raspberry-pi-pico-essentials

› E-Buch: „Raspberry Pi Pico Essentials“
www.elektor.de/raspberry-pi-pico-essentials-e-book

› Raspberry Pi Pico Mikrocontroller-Board
www.elektor.de/raspberry-pi-pico-rp2040

Haben Sie Fragen oder Kommentare?
Haben Sie technische Fragen oder Anmerkungen zu diesem Artikel? Dann schreiben Sie eine E-Mail an den Autor unter d.ibrahim@btinternet.com oder an Elektor unter editor@elektor.com.

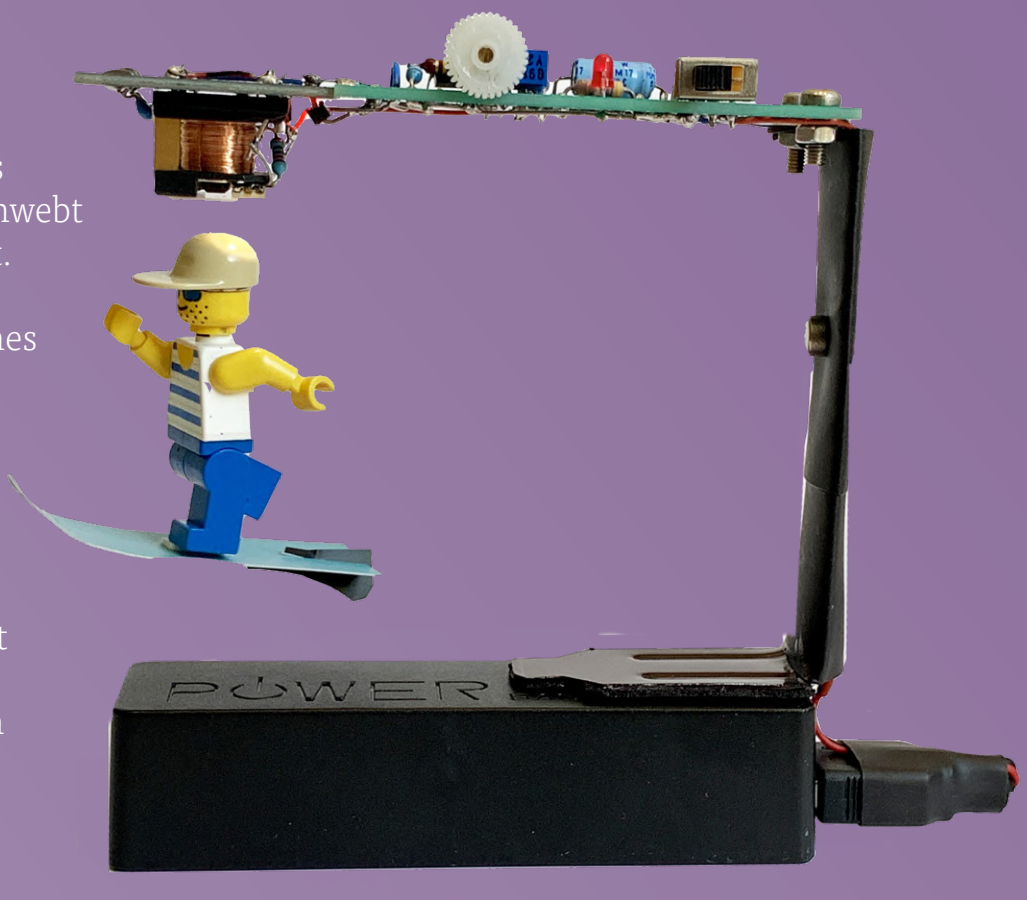
Ein Beitrag von
Text: **Dogan Ibrahim**
Redaktion: **Jan Buiting**
Übersetzung: **Rolf Gerstendorf**
Layout: **Giel Dols**

WEBLINK

[1] **Beispiel-Programmdateien**: www.elektor.de/raspberry-pi-pico-essentials

Magnetische Levitation die einfache Art

Von **Peter Neufeld** und **Luc Lemmens** (Elektor)



Zwei ähnliche DIY-Magnetschwebeprojekte von Peter Neufeld wurden schon im letzten Jahr auf Elektor Labs vorgestellt. In beiden Fällen schwebt ein Lego-Männchen in der Luft. Beide funktionieren nach dem gleichen Prinzip: Die Stärke eines Magnetfeldes zwischen einem Elektromagneten und einem Neodym-Magneten wird mit einem Hall-Sensor gemessen und der Elektromagnet abhängig von diesem Messwert gesteuert. Der Unterschied zwischen den Projekten liegt in der Steuerschaltung. In einem Projekt wird der Magnet von einem Mikrocontroller-Modul M5Stack Atom ESP32 Pico gesteuert, der andere Entwurf, der hier beschrieben wird, ist „hardware only“ und verwendet zur Steuerung nur einen analogen Komparator.

Auf Wikipedia wird Levitation beschrieben als „das freie Schweben eines Objektes. Dazu wird mithilfe einer Kraft die wirkende Gewichtskraft kompensiert und ein Objekt im Raum positioniert, wobei kein direkter Kontakt zum Boden oder zu festen Objekten besteht“ [1]. Auf das Objekt wird also eine nach oben gerichtete Kraft ausgeübt, die der Erdanziehungskraft entspricht. Wenn man ein elektronisches Schwebeprojekt entwerfen möchte, kann dies beispielsweise durch einen Elektromagneten geschehen. Theoretisch wäre auch eine Konstruktion aus zwei Permanentmagneten möglich, aber da es in der realen Welt alle möglichen störenden Einflüsse gibt, die dieses instabile Gleichgewicht stören könnten, werden in der Praxis Elektromagnete verwendet. Ein Regelkreis kann dabei dann die Stärke des Magnetfeldes variieren, um das schwebende Objekt in Position zu halten. Dies erfordert eine Art Rückmeldung über die Position des schwebenden Objekts, und in den beiden auf den ElektorLabs-Seiten vorgestellten Projekten wird ein Hall-Sensor verwendet, um die Stärke des Magnetfelds zwischen einem fest positionierten Elektromagneten und einem darunter schwebenden Permanentmagneten zu messen. Die Ausgangsspannung dieses Sensors wird als Maß für den Abstand zwischen der beiden Magneten verwendet.

Der Begriff „Regelkreis“ lässt aufwendige, recht komplexe Berechnungen und Schaltungen vermuten, um ein stabiles Regelsystem zu realisieren, aber die Projekte von Peter Neufeld zeigen, dass es nicht immer kompliziert sein muss.

Levitation-Projekte, bei denen Objekte einfach in der Luft schweben, sind immer faszinierende Hingucker. Es sieht aus, so als ob die Gesetze der Schwerkraft für das schwebende Objekt nicht mehr gelten, was natürlich nicht der Fall ist. Ich fand es ziemlich außergewöhnlich, dass dies mit so wenig Elektronik möglich war. Die meisten anderen DIY-Leviationen, die ich vor Peters Projekten gesehen hatte, sahen viel schwieriger zu bauen aus, oft mit selbstgewickelten Spulen, und erforderten hohe Ströme. Die Videos von Peter haben aber bewiesen, dass seine Projekte funktionieren, auch wenn es einiges an Präzision und Geschicklichkeit braucht, um die Kalibrierung richtig hinzubekommen. Glauben Sie nicht? Einfach mal ausprobieren!

Dieser Artikel konzentriert sich zunächst darauf, wie die Levitation überhaupt funktioniert und stellt dann das einfachste von Peters Projekten, genannt „easy way“ [2], mit einem Analogkomparator zur Steuerung des Elektromagneten vor. In einem folgenden Artikel wird die Version mit dem M5Stack Atom ESP32 Pico behandelt [3], die mehr Wert auf die „Kosmetik“ legt, um ein schön anzuschauendes Levitationsgerät zu schaffen.

Hardware für die analoge Version

Der Schaltplan in **Bild 1** zeigt, dass tatsächlich nur sehr einfache Hardware benötigt wird, um dieses Projekt zu bauen. L1 stellt die Spule eines Relais dar, wobei der Schaltmechanismus entfernt wurde. Sie ist Teil einer fertigen Standard-Relaisplatine, die mit dem Kern nach unten an einem Rahmen befestigt ist, unter dem das mit einem Permanentmagneten ausgestattete levitierte Objekt schwebt. R6 und LED3 ersetzen die Freilaufdiode D4 und dienen als optische Anzeige während der Kalibrierung der Levitationsschaltung. Der Hallsensor U1 ist auf dem Kern montiert und misst das Magnetfeld zwischen L1 und dem schwebenden Objekt. Der Sensorausgang ist mit dem invertierenden Eingang des Komparators U2, einem LM311, verbunden. Der nicht-invertierende Eingang von U2 ist mit dem Schleifer des Mehrgang-Trimmpotis RV1 verbunden, an dem der Regelkreis abgeglichen wird. R3 sorgt für eine kleine Hysterese des Komparators. Der Ausgang des Komparators schaltet über Q1 die Spule ein, wenn die Spannung am Ausgang des Hallsensors niedriger ist als der voreingestellte Wert am Schleifer von RV1.

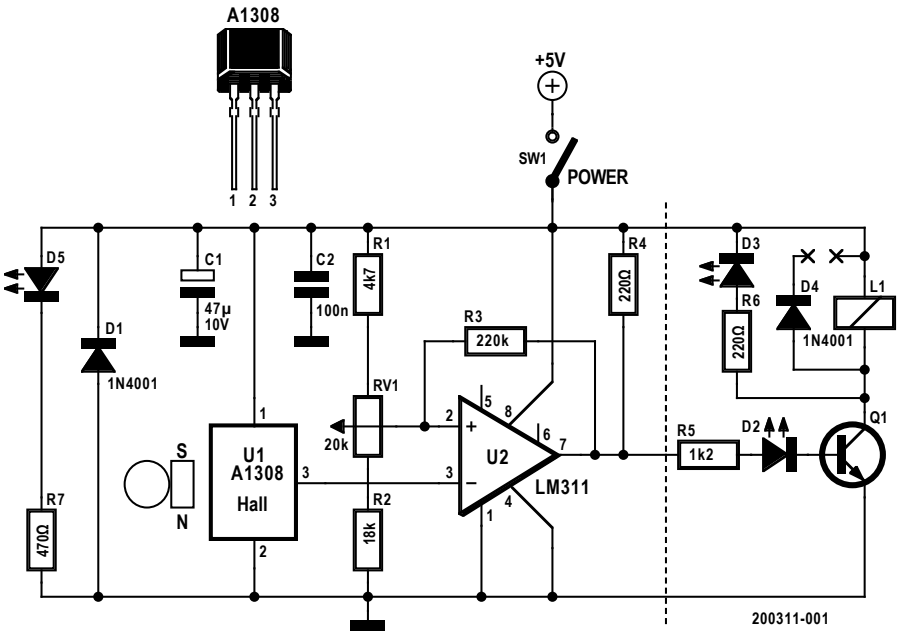


Bild 1. Schaltbild des Levitation-Projekts.

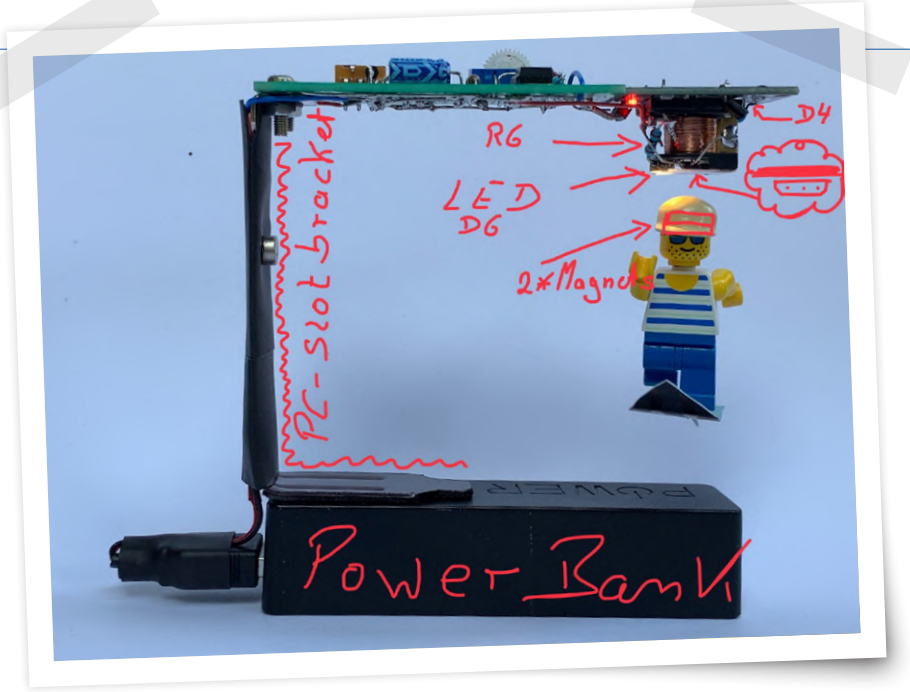


Bild 2. Bestandteile der fertig aufgebauten Schaltung.

Die LED D5 ist die Einschaltanzeige und SW1 der An/Aus-Schalter. D1 ist eine Klemmdiode zum Schutz der Schaltung, wenn die Versorgungsspannung versehentlich verkehrt herum angeschlossen wird.

Beschaffung der (richtigen) Bauteile

Die meisten Teile in Schaltplan und Stückliste sind Standardbauteile, nur die Hall-Sensoren A1302 oder A1308 von Allegro erwiesen sich als etwas schwieriger zu beschaffen. Billigere und weiter verbreitete Sensoren wie der SS49 funktionierten in den frühen Versuchen von Peters Levitation-Projekten nicht, wahrscheinlich weil das Ausgangssignal nicht schnell genug reagiert, wenn sich die Magnetfeldstärke ändert, aber das wurde nicht weiter untersucht. Überraschenderweise war das am schwierigsten zu beschaffende Teil die Platine mit

Bild 3a. Nach Auslöten der Platinenanschlussklemme wird der Relaisabdeckung zu Leibe gerückt.

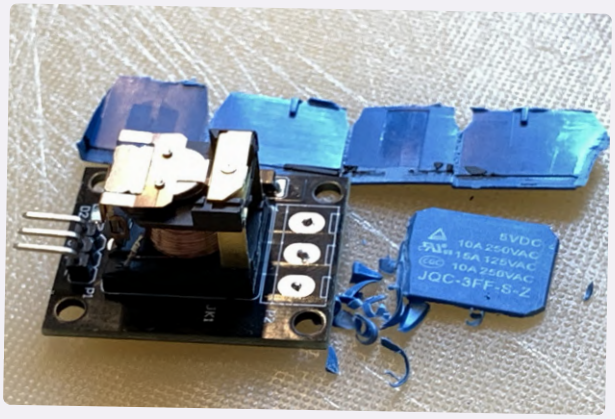


Bild 3b. Abrissarbeiten: das nackte Relais.

Bild 3c. Der Schaltkontakt wird entfernt und der Kern abgeflext.

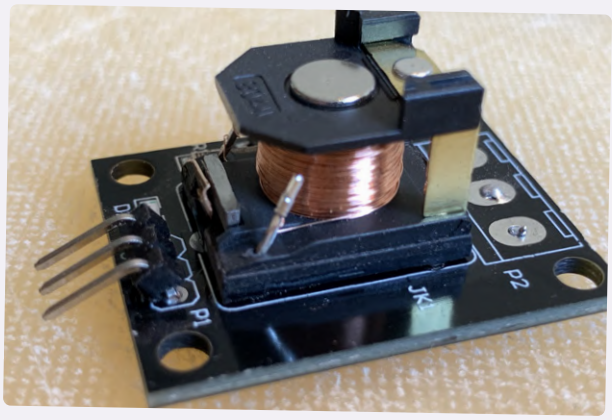


Bild 3d. Das Relais wurde in einen Elektromagneten umgewandelt.

Bild 3. Modifikation des Relaismoduls.

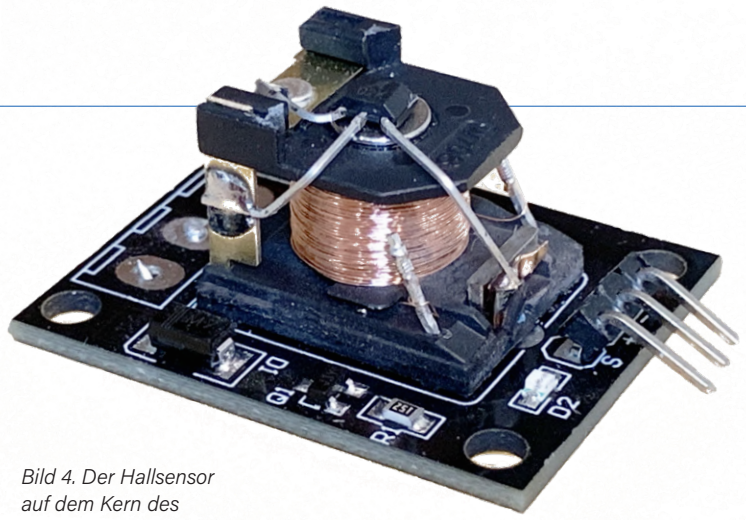


Bild 4. Der Hallsensor auf dem Kern des Elektromagneten montiert.

STÜCKLISTE

Widerstände:
R1 = 4k7
R2 = 18 k
R3 = 220 k
R4,R6 = 220 Ω
R5* = 1k2
R7 = 470 Ω
RV1 = 20 k
Mehrgang-Trimmpoti

Kondensatoren:
C1 = 47 µ, 10 V radial
C2 = 100 n

Halbleiter:
D1 = 1N4001
D2*,D5 = rote LED
D3 = weiße LED
D4* = nicht bestückt (oder von der Relaisplatine entfernen)
Q1* = BC550
U1 = Hallsensor A1302 oder A1308 (Allegro)
U2 = Komparator LM311

Außerdem:
5V-Relaisplatine*
SW1 = Schiebeschalter
Neodym-Scheibenmagnete*

*= siehe Text

dem Relais, das zu einem Elektromagneten umgebaut wird. Solche Module sind im Internet in Dutzenden von Maker- und Arduino-Webshops leicht erhältlich und es sollte doch kein Problem sein, eines zu finden und zu kaufen. Peter hatte Boards mit der Bezeichnung HW-482 mit den Relaistypen JQC-3FF-S-Z und JQC3F-05VDC getestet. Ich habe (wieder einmal) eine sehr wichtige Lektion beim Einkaufen im Internet gelernt: Traue niemals Fotos in Webshops! Ich sah mir die Fotos an, die Peter mir geschickt hatte, und fand genau die gleiche Platine bei Amazon. Ich erhielt aber Module mit der Platine mit der Bezeichnung HW-307, die ein Relais mit der Typennummer FL-3FF-S-Z tragen. Sie funktionieren wie beschrieben und im Shop versprochen, das heißt, als 5-V-SPDT-Relais-Board mit Treibertransistor, Freilaufdiode und Anzeige-LEDs. Allerdings entpuppte sich der Transistor auf dieser Platine als ein PNP-Typ, im Gegensatz zu dem NPN auf den von Peter getesteten Relaisplatinen. Ich habe es trotzdem ausprobiert, denn bis auf den Typ und die Marke sah das Relais dem im ursprünglichen Projekt verwendeten sehr ähnlich, und wie sich später herausstellte, auch im Inneren. Immerhin sind für dieses Projekt nur die Spule und der Kern des Relais wichtig.

Für die Magnete wurde ein Stapel von zwei oder drei scheibenförmigen Neodym-Magneten mit einem Durchmesser von 8 mm bis 12 mm und einer Dicke von 2 mm bis 3 mm geordert. Die Größe und Anzahl der benötigten Magnete hängt auch von den Maßen des Lego-Püppchens oder einer anderen Last ab. Ich habe mit einem Stapel von nur zwei Magneten begonnen, beide mit einem Durchmesser von 10 mm und einer Höhe von 2 mm, und ich würde empfehlen, mit einem solch einfachen Objekt zu experimentieren, um ein Gefühl dafür zu bekommen, wie man den Regelkreis einstellen muss. Es kann nützlich sein, die Oberseite des Magnetstapels mit einem Permanentmarker oder einem Stück Klebeband zu markieren, um es stets richtig auszurichten. Sobald Sie das hinbekommen haben, können Sie mit anderen Magneten und dem Hinzufügen von Objekten experimentieren. Natürlich gibt es Grenzen für die Größe und das Gewicht der Last, die mit dieser Hardware schweben kann. In diesem Artikel bezieht sich der Begriff „Magnet(e)" stets auf die komplette Last, also den Stapel von Magneten plus optionaler Last.

Aufbau

Es ist keine speziell entworfene Platine für dieses Projekt erforderlich, ein einfaches Stück Lochraster oder Veroboard oder sogar ein Breadboard ist für die wenigen Bauteile ausreichend. **Bild 2** zeigt, wie die Bestandteile des ursprünglichen Prototyps zusammengesetzt waren. Der Aufbau beginnt mit einem Abbau: Der 3-polige Block mit den Platinenanschlussklemmen wird von der Relaisplatine ausgelötet, um mehr Bewegungsfreiheit für die Arbeiten am Relais zu erhalten. Schauen Sie sich Peters Fotos in **Bild 3** genau an, um zu sehen, wie mit dem Relais umzugehen ist. Entfernen Sie die Abdeckung und den größten Teil des Schaltmechanismus'; nur die Spule und der Kern sind für uns wichtig. Kürzen Sie den U-förmigen Kern auf eine J-Form, um einen Kurzschluss des Magnetfelds zu vermeiden. Um die Metallteile zu schneiden, könnte sich eine Art kleine Schleifscheibe auf einem Dremel-Werkzeug als nützlich erweisen.

Unabhängig davon, welche Art von Relaisplatine Sie haben, muss die Freilaufdiode D4 entfernt und durch R6 und die weiße LED D3 ersetzt werden. Bei Peters Relaisplatine ist damit der rechte Teil des Schaltplans in Bild 1 einschließlich Q1, R5 und D2 abgehandelt. In meinem Fall habe ich nur die Relaisspule behalten, alle anderen Bauteile entfernt und diesen Teil der Schaltung mit bedrahteten Bauteilen neu aufgebaut. Für die Ansteuerung der Relaisspule reicht für Q1 ein BC550 oder ein anderer Standard-NPN-Transistor aus. Im Nachhinein betrachtet wäre es sinnvoller gewesen, nur das Relais zu kaufen, aber diese kleinen, in Massenproduktion hergestellten Relaismodule sind wahrscheinlich billiger als ein passendes separates Relais.

Bitte beachten Sie, dass die Bauteilbezeichnungen (L1, R5, D2 Q1 und D4) auf der Relaisplatine **nicht** *mit denen im Schaltplan übereinstimmen. Es dürfte jedoch nicht allzu schwierig sein, die zu entfernende Freilaufdiode D4 zu identifizieren.*

Der Hallsensor wird direkt und mittig auf dem Kern der Spule positioniert (**Bild 4**). Kleben Sie ein dünnes Stück Kunststoff über den Sensor und seine Pins, um einen Kurzschluss der Stromversorgung zu verhindern, wenn der Magnet an den Kern herangezogen wird.

Der Auf- und Zusammenbau der Schaltungen ist nicht allzu schwierig, beachten Sie aber, dass die Verdrahtung der Spule und die Ausrichtung des Permanentmagneten und des Hallsensors sehr wichtig sind, damit die Magnet-Levitation funktioniert:

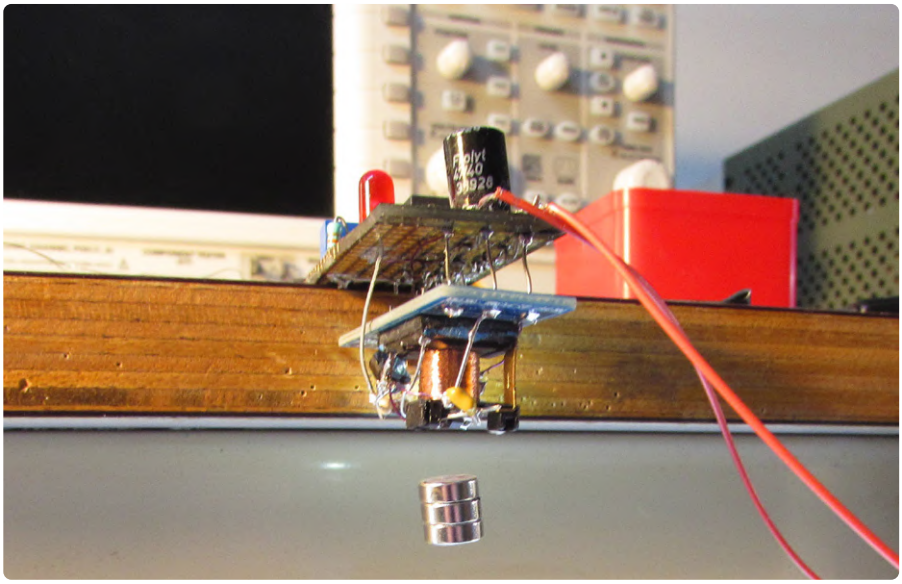


Bild 5. Dieser Prototyp funktioniert zwar, aber ein gefälliges Äußeres wäre wünschenswert.

- Wenn sich der Permanentmagnet der Spule und dem Hallsensor nähert, muss sich die Ausgangsspannung des Sensors erhöhen.
- Wenn die Spule eingeschaltet wird, muss ihr Magnetfeld den Magneten anziehen.

Wenn eine oder beide dieser Bedingungen nicht erfüllt sind, funktioniert es nicht. Die erste Bedingung lässt sich leicht überprüfen, indem man die Ausgangsspannung des Hallsensors mit einem Multimeter misst. Gegebenenfalls muss man einfach den Magneten umdrehen, wenn die Spannung bei Annäherung von Sensor und Permanentmagnet sinkt statt steigt. Und die zweite … Ich konnte die Kraft zwischen der stromführenden Spule und dem Magneten doch fühlen, das schien in Ordnung zu sein.

Es sah so aus, als ob die Steuerschaltung, die ich gebaut hatte, richtig funktionierte, da beide LEDs leuchteten, wie sie sollten: D2, wenn die Spule eingeschaltet ist, D3 blinkte bei ausgeschalteter Spule, wenn der Magnet sich der Spule nähert. Es sollte nur eine Sache der Einstellung von Trimpoti RV1 sein, um die richtige Spannungseinstellung am nicht-invertierenden Eingang des Komparators zu erhalten, um den Magneten am Lego-Püppchen wie im Video schweben zu lassen. Aber meine Bemühungen waren vergeblich: Der Magnet hing sich einfach an die Spule auf oder folgte der Gravitation auf die Tischplatte. Dann erinnerte ich mich an die zweite Bedingung, damit die Levitation funktioniert. Ich konnte definitiv spüren, dass der Magnet zwischen meinen Fingern von der Spule angezogen wurde, wenn sie mit Strom versorgt wurde. Das sollte doch in Ordnung sein. Oder doch nicht?

Wenn alles fehlschlägt: RTFM

Zuerst überprüfte ich die Stärke des Magnetfeldes der Spule. Zu meiner Überraschung stellte ich fest, dass ihre elektromagnetische Kraft fast vernachlässigbar zu sein schien: Direkt mit einer 5-V-Versorgung versorgt, konnte sie kaum den kleinsten Eisengegenstand anheben. Nicht gerade die Art von Kraft, die man erwarten würde, um etwas hochzuheben, geschweige denn eine Kraft, die stark genug ist, um eine relativ schwere Last wie einen Magneten und einen daran befestigten Gegenstand schweben zu lassen. Ich habe hier etwas sehr Wichtiges übersehen: Auch wenn die Spule nicht mit Strom versorgt wird, gibt es ja noch die statische Magnetkraft zwischen dem Permanentmagneten der Last und dem Metallkern der Spule. Und diese Kraft ist viel stärker als die elektromagnetische Kraft der Spule. Der Trick ist also: Wenn sich der Dauermagnet dem Kern nähert, gibt es einen Punkt, an dem die statische Kraft einfach nicht stark genug ist, um den Magneten zum Kern zu ziehen. Hier kommt das zusätzliche elektromagnetische Feld ins Spiel: Die Spule fügt

EINE POWERBANK ALS STROMQUELLE

Wenn sich kein Permanentmagnet in der Nähe des Kerns und des Hallsensors befindet, ist die Spule permanent eingeschaltet. Die gesamte Stromaufnahme dieser Schaltung beläuft sich dann auf etwa 75 mA, was hauptsächlich durch den Strom durch den Elektromagneten bestimmt wird. Wenn ein Magnet unter der Spule schwebt, reduziert sich der durchschnittliche Strom auf nur 50 mA bei 5 V Versorgungsspannung, was für ein Magnetlevitation-Projekt erstaunlich niedrig ist.

Peter Neufeld verwendet eine USB-Powerbank zur Stromversorgung, wodurch das Projekt auf einem Tisch oder Schreibtisch präsentiert werden kann, ohne dass ein Stromkabel angeschlossen werden muss. Gleichzeitig ist die Powerbank eine standfeste Basis für die Halterung, die die Hardware trägt. Die meisten Powerbanks schalten sich bei geringer Belastung nach ein paar Sekunden automatisch ab, und je nach Typ und Marke der Powerbank können 50 mA unter dieser Schwelle liegen. In solchen Fällen funktioniert diese Art von Stromversorgung nicht, aber das kann leicht behoben werden, indem man die Stromaufnahme dieser Schaltung erhöht, zum Beispiel durch Hinzufügen einer oder mehrerer Power-LEDs.

dem statischen Feld nur eine kleine Kraft hinzu, gerade stark genug, um den Magneten nach oben zu ziehen. Das vom Hallsensor gemessene Magnetfeld nimmt zu (und damit auch seine Ausgangsspannung), wenn sich der Magnet dem Kern nähert, wodurch bei korrekter Einstellung von RV1 die Spule abgeschaltet und verhindert wird, dass der Magnet vollständig zum Kern gezogen wird. Die Schwerkraft zieht den Magneten dann nach unten, wodurch das vom Sensor gemessene Feld sinkt und die Spule wieder eingeschaltet wird - und so weiter.

Was ich also falsch gemacht habe: Da das elektromagnetische Feld im Vergleich zum statischen Magnetfeld sehr schwach ist, konnte ich nicht sagen, ob die eingeschaltete Spule den Magneten anzieht oder abstößt. So ermittelte ich mit einem altmodischen Kompass, ob die Ausrichtung des elektromagnetischen Feldes korrekt war, um die zweite Bedingung für den Schwebeflug zu erfüllen. Als ich Peters Dokumentation erneut las, sah ich, dass er eine einfache Lösung erwähnte, um die richtige Ausrichtung der Spule zu finden: Die Ausgangsspannung des Hallsensors steigt nicht nur an, wenn sich der Permanentmagnet nähert, sondern auch, wenn die Spule mit Strom versorgt wird (zum Beispiel durch Verbinden des Kollektors von Q1 mit GND). Sie müssen also unter Umständen die Anschlüsse der Spule und der LED D3 vertauschen, um die richtige Ausrichtung zu erhalten.

Abgleich mit Fingerspitzengefühl

Wie bereits erwähnt, erfordert es Präzision und Fingerspitzengefühl, um den Punkt zu finden, an dem die Last schwebt, irgendwo im Abstand zwischen Spule und Last von 10 mm bis 15 mm. Peter beschreibt ein Verfahren mit einem Stapel von Notizzetteln, um den richtigen Abstand für die Kalibrierung zu finden, aber aus irgendeinem Grund - wohl wegen meines Mangels an Präzision und Geschicklichkeit - hat das bei mir nicht funktioniert. Mein Trick war, den Magneten auf meine Hand zu legen und sie ganz langsam in Richtung der Spule zu heben, bis ich spüren konnte, dass die Magnetkraft sie anzieht. Wenn die Spule abschaltet, bevor dieser Punkt erreicht ist, stellen Sie RV1 auf einen höheren Schwellenwert am nicht-invertierenden Eingang des Komparators ein, oder auf einen niedrigeren Pegel, wenn sie nicht abschaltet, bevor der Magnet zum Kern hochgezogen wird. Die weiße LED leuchtet kurz auf, wenn die Spule abgeschaltet wird. Stellen Sie das Potentiometer so ein, dass diese LED scheinbar ständig leuchtet (tatsächlich blinkt sie mit der Steuerfrequenz des Magneten). Am Anfang wird der Magnet wahrscheinlich am Kern kleben bleiben, aber wenn Sie den Dreh erst einmal

raus haben, wird es immer einfacher, die Kalibrierung für andere Lasten richtig hinzubekommen. Bei meinem Aufbau konnte ich dann sogar am Schalten des Regelkreises hören, wenn die richtige Einstellung erreicht ist. Und ja: Es kann einige Zeit dauern, bis man es richtig hinbekommt, aber nur Mut, Sie können es schaffen!

Meine erste Absicht war es, einen Prototyp zu bauen, der mindestens so gut aussieht wie der des Autors, aber mit all den Änderungen, die ich vornehmen musste, um ihn zum Laufen zu bringen, bin ich in dieser Hinsicht völlig gescheitert, wie man in **Bild 5** sehen kann. Aber mein Hauptziel habe ich erreicht: Ich habe die Levitationstechnik zum Laufen gebracht! Eines Tages, wenn mir die Ideen ausgehen, was ich mit meiner Freizeit anfangen soll ... Aber ich will mich mehr um die Kosmetik bemühen, wenn ich das zweite Levitationsprojekt von Peter Neufeld, der digitalen Lösung, aufbaue. Mit den Lektionen, die ich bei diesem „einfachen" analogen Aufbau lernen musste, sollten beim nächsten Projekt die meisten Fallstricke vermieden werden. Dann bleibt mir mehr Zeit, um mich um ein gefälliges Äußeres zu kümmern!

200311-02

Ein Beitrag von

Entwurf: **Peter Neufeld**
Text und Bearbeitung: **Luc Lemmens**
Illustrationen: **Peter Neufeld, Patrick Wielders, Luc Lemmens**
Übersetzung: **Rolf Gerstendorf**
Layout: **Harmen Heida**

Haben Sie Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Wenden Sie sich bitte an die Redaktion unter luc.lemmens@elektor.com!

Passende Produkte

- **Magnetsphere Kit**
  www.elektor.de/magnetsphere-kit

WEBLINKS

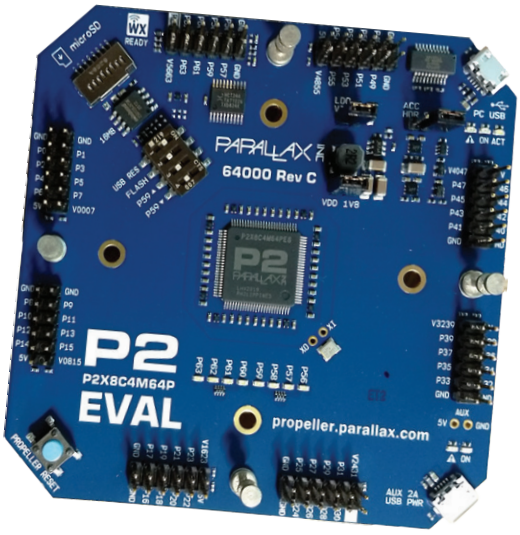
[1] **Wikipedia-Seite zur Levitation**: https://de.wikipedia.org/wiki/Levitation_(Technik)

[2] **P. Neufeld, „Magnetic Levitation - The Easy Way", ElektorMagazine.de, Juni 2020**: www.elektormagazine.de/labs/magnetic-levitation-the-easy-way

[3] **P. Neufeld, „Magnetic Levitation - The Digital Way", ElektorMagazine.de, Juli 2020**: www.elektormagazine.de/labs/magnetic-levitation-the-digital-way

Parallax Propeller 2

Teil 3: Smart Pins und serielle Daten (UART)



Von **Mathias Claußen** (Elektor)

In der letzten Folge wurde eine LED zum Leuchten gebracht. Jetzt soll sie blinken. Weiter geht es mit den Funktionen der Smart Pins, mit denen sich ein UART bilden lässt, der Zeichen übertragen kann.

In den vorherigen Folgen dieser Serie ging es um Grundsätzliches der MCU Propeller 2 und die Ansteuerung einer LED via SPIN2. Jetzt wollen wir diese LED blinken lassen. Dabei werden Funktionen der IO-Pins gestreift.

Eine einfache Lösung

Am einfachsten ist sicher das Schema „Aus- und wieder Einschalten". Dazu braucht es folgende Zutaten:

- `pinwrite()`
- `repeat()`
- `waitms()`

Die notwendigen Schritte umfassen:

- Einschalten der LED
- Warten für 500 ms
- Ausschalten der LED
- Warten für 500 ms
- Wiederholung des Vorhergehenden

Der Code sieht dann so aus wie in **Bild 1**. Sie können ihn von der Elektor-Webseite zu diesem Artikel [1] herunterladen.
Nachdem die Pins als Ausgang verwendet wurden, stellt sich eigentlich die Frage, wie man sie als Eingänge verwenden kann. Dieses Thema wird später behandelt. Zunächst geht es darum, eine Art seriellen Datenausgang zu realisieren. Dieser soll Daten an einen PC schicken können, denn so kann man sich auch anzeigen lassen, welchen Status ein I/O-Pin hat. Das bringt hier mehr, als lediglich eine LED blinken zu lassen. Außerdem lässt sich dieser Ausgang zum Versenden von Statusdaten nutzen und ermöglicht eine Art Debugging für den Code. Der nächste Punkt dreht sich daher um sogenannte Smart-Pins.

Smart-Pins

Heutzutage drapieren Marketing-Teams gerne alle möglichen und unmöglichen Produkte mit dem Etikett „smart". Smart-Pins aber sind nicht bloß Marketing-Speak. Sie können weit mehr als gewöhnliche IO-Pins. Bei MCUs gibt es häufiger die Möglichkeit, aus mehreren Funktionen für einen Pin zu wählen. Manche MCUs wie der ESP32 haben sogar eine IO-Matrix, die jedes interne IO-Signal auf jeden beliebigen Pin routen kann. Doch selbst hier bleibt ein IO-Pin immer noch ein IO-Pin. Die eigentlichen Funktionen wie UART, SPI oder ADC befinden sich in einer dedizierten Hardware auf dem Die, die auf diese Weise schlicht mit Pins verbunden werden. Die Smart-Pins des Propeller 2 sind aber anders: Die integrierte Peripherie besteht eben nicht aus dedizierten Funktionsblöcken der MCU, die durch eine IO-Matrix geroutet werden. Stattdessen sind viele Funktionen zumindest teilweise direkt in jeden IO-Pin integriert. Die Bezeichnung *Smart Pin* ist daher wirklich gerechtfertigt.

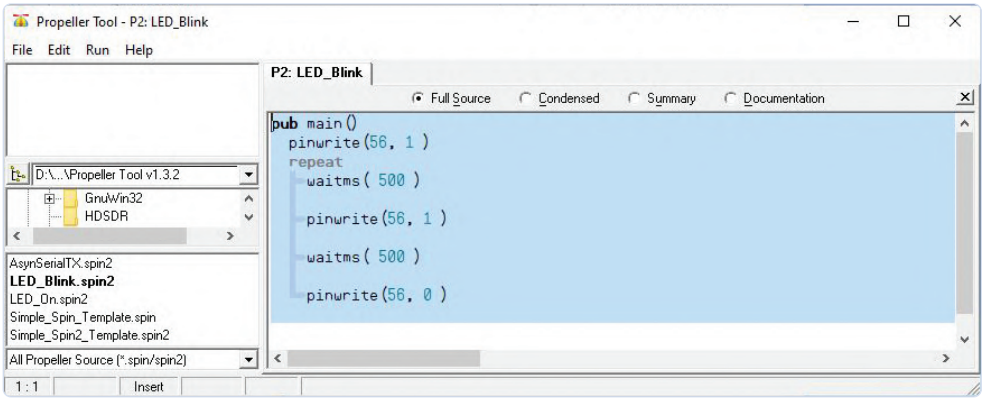


Bild 1. Code zum Blinken einer LED.

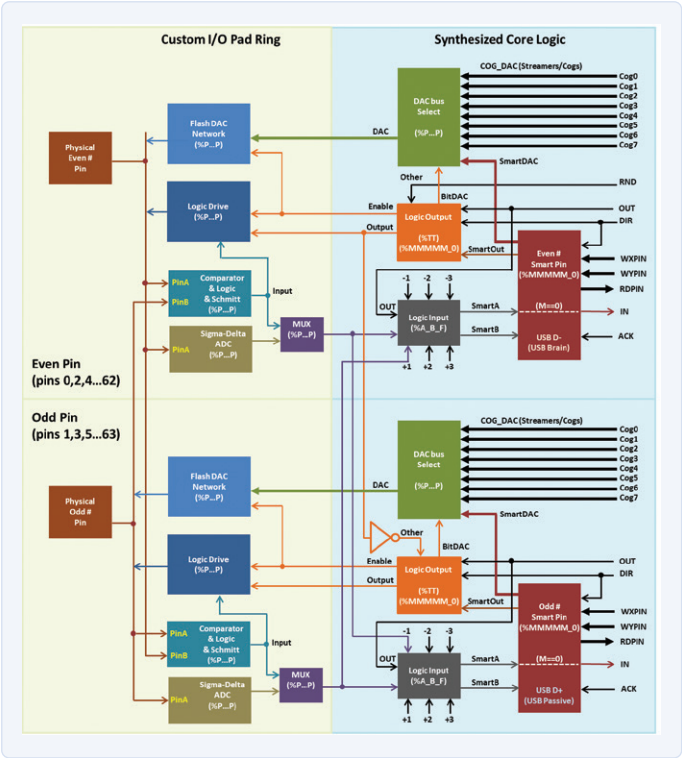


Bild 2. Smart-Pin-Übersicht.

Netterweise hat der User *rayman* im Parallax Forum [2] einen tollen Job gemacht, indem er der Community einen Überblick über Smart-Pin-Interna gewährt. Sein Beitrag findet sich samt Bild unter [3]. Sie können auch einen Blick auf **Bild 2** werfen. Der nebenstehende Kasten enthält einen Auszug aus dem Datenblatt. Sie sehen, dass ein Pin viele Funktionen haben kann. Im Moment aber interessiert vor allem *11110* = asynchrones serielles Senden*, da man so Daten für Debugging-Zwecke verschicken kann. Hierzu sollte klar sein, wie man den passenden Modus für den Pin einstellt und ob SPIN2 dazu reicht, oder ob eine Prise Assembler nötig ist.

UART-Konfiguration

Dies ist der Punkt, wo man unwillkürlich die Augenbrauen hebt, wenn man zum ersten Mal mit SPIN2 und Propeller 2 zu tun hat. Das Datenblatt ist relativ vollständig, doch alles zu lesen und richtig zu verstehen kann länger dauern. Erstes Ziel ist daher eine einfache Funktion `tx()`, die ein Zeichen an einen als UART operierenden Pin schickt. Die Parameter sind: 115.200 Baud, 8 Daten-Bits, keine Parität und ein Stopp-Bit. Zunächst muss man den Pin konfigurieren:

- Pin auf *async serial transmit* setzen
- Einstellung der Baudrate und der Daten-Bits
- Aktivieren des Smart Pins

Mit diesen drei Schritten wird ein sendender UART-Pin realisiert. Da der Code später wiederverwendet und modifiziert werden soll, wird er in eine Funktion verpackt. Eine Funktion enthält einfach Code, der innerhalb eines Programms häufiger verwendet wird. Das vermeidet viel redundantes „Copy and Paste" von Code und erlaubt es, die Änderungen an einer einzigen Stelle vorzunehmen. Die Funktion erhält den Namen `serial_start`. Sie nimmt keine Argumente entgegen und konzentriert sich auf die drei oben genannten Schritte. Der verwendete Ausgang ist fest als Pin 57 kodiert (einer der am Rand zugänglichen LED-Pins, siehe **Bild 3**).

Die Funktion beginnt mit einem `PUB`, gefolgt von ihrem Namen und leeren geschweiften Klammern. Letzteres bedeutet, dass keine Argumente übernommen werden.

Übersicht der Smart-Pin-Funktionen

00000	smart pin off (default)
00001	long repository (P[12:10] != %101)
00010	long repository (P[12:10] != %101)
00011	long repository (P[12:10] != %101)
00001	DAC noise (P[12:10]= %101)
00010	DAC 16-bit dither, noise (P[12:10] = %101)
00011	DAC 16-bit dither, PWM (P[12:10] = %101)
00100*	pulse/cycle output
00101*	transition output
00110*	NCO frequency
00111*	NCO duty
01000*	PWM triangle
01001*	PWM sawtooth
01010*	PWM switch-mode power supply, V and I feedback
01011	periodic/continuous: A-B quadrature encoder
01100	periodic/continuous: inc on A-rise & B-high
01101	periodic/continuous: inc on A-rise & B-high / dec on A-rise & B-low
01110	periodic/continuous: inc on A-rise (/ dec on B-rise)
01111	periodic/continuous: inc on A-high (/ dec on B-high)
10000	time A-states
10001	time A-highs
10010	time X A-highs/rises/edges -or- timeout on X A-high/rise/edge
10011	for X periods, count time
10100	for X periods, count states
10101	for periods in X+ clocks, count time
10110	for periods in X+ clocks, count states
10111	for periods in X+ clocks, count periods
11000	ADC sample/filter/capture, internally clocked
11001	ADC sample/filter/capture, externally clocked
11010	ADC scope with trigger
11011*	USB host/device (even/odd pin pair = DM/DP)
11100*	sync serial transmit (A-data, B-clock)
11101	sync serial receive (A-data, B-clock)
11110*	async serial transmit (baudrate)
11111	async serial receive (baudrate)

* OUT signal overridden

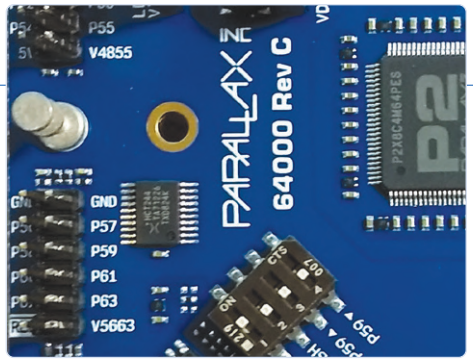


Bild 3. Lage der Stiftleiste für den Zugriff auf die LED-Pins.

```
PUB serial_start()
  WRPIN( 57, %01_11110_0  )              'set async tx mode for txpin
  WXPIN( 57, ((217<<16) + (8-1 )) )      'set baud rate to sysclock/115200 and word size to 8
  org
   dirh    #57
  end
```

Bild 4. Code von serial_start()

```
PUB tx(val)

      WYPIN(57,val) 'load output word
org

      WAITX   #1              'wait 2+1 clocks before polling busy
      wait
      RDPIN   val,#57  WC     'get busy flag into C
      IF_C    JMP     #wait           'loop until C = 0

end
```

Bild 5. Code für die Funktion tx(val).

Zeile 1 in **Bild 4** enthält den oben erwähnten Funktionskopf. Die nächste Zeile setzt Pin 57 in den Modus für asynchrones serielles Senden (via 11110). Das erste Bit ist immer Null; die beiden höchstwertigen Bits (hier 01) bedeuten, dass der Pin als GPIO- oder Smart-Pin angesteuert werden soll. Dies wird mit der SPIN2-Funktion `WRPIN` im ersten Schritt erreicht. Die nächste Funktion `WXPIN` konfiguriert den gewünschten Takt-Teiler und die zu verwendenden Daten-Bits für einen Smart-Pin im asynchronen seriellen Modus. Die Sache mit dem Teiler für die Baudrate wird zunächst vertagt. Kurz gesagt: Der Wert für die Baudrate ergibt sich als Quotient von systemclock / baudrate

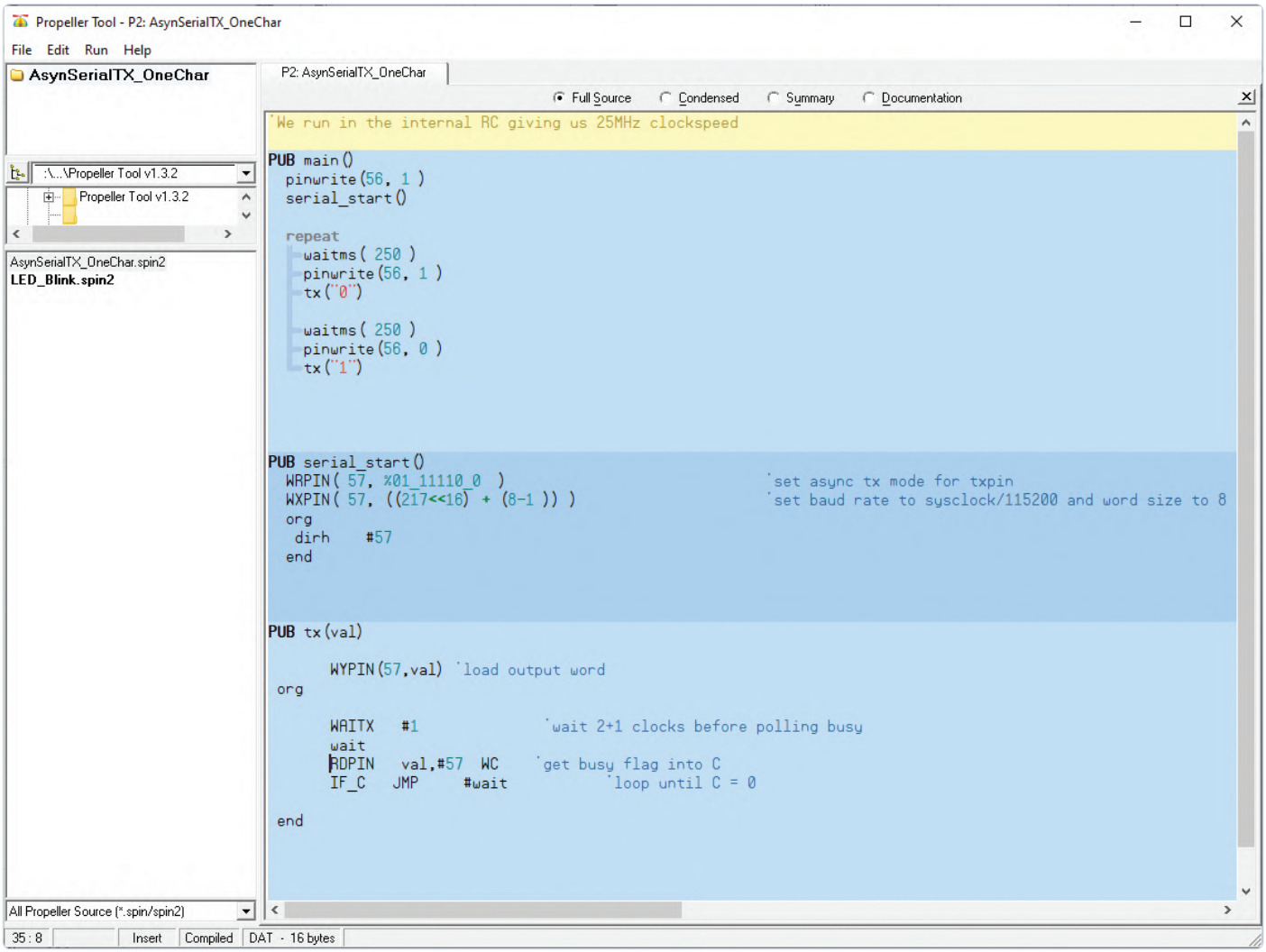


Bild 6. Kompletter Code zusammengesetzt.

- hier 25 MHz / 115.200 bd ≈ 217. Das Ergebnis muss noch um 16 nach rechts verschoben werden. Für die Anzahl der zu übertragenden Bits gilt die Formel: gewünschte Bits - 1, also 8 – 1 Bits.
Mehr braucht es nicht, um die Übertragungsgeschwindigkeit und die Daten-Bits einzustellen. Die nächsten drei Zeilen unterscheiden sich vom bisherigen Code. Der hier auftauchende Assembler-Code bedarf wohl zusätzlicher Erklärung. Mit `org` startet ein Abschnitt mit Assembler-Befehlen. Der Assembler-Befehl `dirh` aktiviert die Smart-Pin-Funktionen, wie es für den dritten Schritt erforderlich ist. Die Festlegung der Nummer des genutzten Pins sieht etwas anders aus, da sie mit einem `#` begonnen werden muss.
Der Assembler-Abschnitt wird mit einem `end` in der letzten Zeile abgeschlossen, was in diesem speziellen Fall auch gleich als Ende der Funktion selbst wirkt. Es wäre schön, wenn man Assembler vermeiden könnte, aber derzeit gibt es keine SPIN2-Entsprechung für die Assembler-Anweisung `dirh`.
Da der Pin jetzt konfiguriert ist, kann man eine Funktion einrichten, die ein Zeichen sendet und wartet, bis sie fertig ist. Die Funktion kann größtenteils aus Seite 91 des Datenblatts [4] entnommen werden. Die vorige Funktion kommt ohne übergebene Argumente aus. Doch wenn ein Zeichen gesendet werden soll, wäre es von Vorteil, wenn es der Sende-Funktion übergeben werden könnte. Um Assembler möglichst zu vermeiden, werden hierzu die SPIN2-Funktionen verwendet.

Datenübertragung

Die Sende-Funktion `tx` sieht `serial_start()` recht ähnlich, wie **Bild 5** beweist. Außer dem anderen Namen befindet sich hier das Argument `val` innerhalb der Klammern. Dieser Parameter enthält nach Übergabe (beziehungsweise Funktionsaufruf) das zu sendende Zeichen. Im Funktionsrumpf wird der Wert des Parameters mit dem Befehl `WYPIN` in das Übertragungsregister von Pin 57 kopiert. Nun folgen wieder ein paar Zeilen Assembler-Code. Es wird gewartet, bis das Busy-Flag für den Sender nicht mehr gesetzt ist und die Übertragung somit abgeschlossen ist. Laut Datenblatt muss man zunächst drei CPU-Zyklen warten, um das Flag konsistent zu lesen. Dies wird durch die Anweisung `waitx` mit dem Parameter `#1` erreicht, da dessen Ausführung zwei Zyklen plus den der Funktion übergebenen Wert von hier einem Zyklus benötigt. Die nächste Zeile ist ein Label `wait`, zu dem später gesprungen werden kann. `RDPIN` (Assembler) liest den Pin-Status mit Übertrag und wird durch `WC` am Ende der Anweisung abgeschlossen. Das Carry-Bit dient hier als Busy-Flag und zeigt an, ob die Übertragung abgeschlossen ist oder nicht.
`RDPIN val, #57 WC` liest den Status inklusive Carry-Bit in `val` ein. Da der Inhalt von `val` gerade übertragen wird, lässt sich sein Speicherplatz dazu nutzen, um den Smart-Pin-Status dorthin zu kopieren. Beim letzten Befehl `IF_C JMP #wait` handelt es sich um einen bedingten Sprung. In BASIC entspricht das dem berüchtigten `GOTO` kombiniert mit einer `IF`-Anweisung. Der Code übersetzt in Klartext:
Wurde irgendein Wert gelesen, bei dem das Carry-Bit (hier Busy-Flag) gesetzt ist? Wenn ja: Springe zurück zum Wait-Label und beginne von dort aus erneut. Wenn nein: Weitermachen. Die Übertragung ist beendet, wenn das Carry-Bit nicht mehr gesetzt ist. Dann wird die Funktion bis zu ihrem Ende laufen und schließlich dorthin zurückspringen, wo sie aufgerufen wurde.

Alle Teile zusammen

Wir können nun den Code zusammenbauen und nach jedem `pinwrite()` eine „0“ oder „1“ übertragen, indem wir an diesen Stellen ein tx(„0“) oder ein tx(„1“) im Code einfügen (siehe **Bild 6**). Um die Ausgaben an einen PC zu übertragen, schließt man nun noch einen Seriell/USB-Konverter an. Ich habe das mit einem Logic 16 erledigt und so die LED-Ansteuerung via serieller Übertragung aufgezeichnet. Das Ergebnis sehen Sie in den **Bildern 7** und **8**.

Doch was ist mit Strings? Wäre es nicht schön, einfach ein print(„Hello World“) über die serielle Schnittstelle zu übertragen, wie es in der Arduino-Welt möglich ist? Auch das ist möglich. In der nächsten Folge geht es um solche Funktionen.

200479-C-02

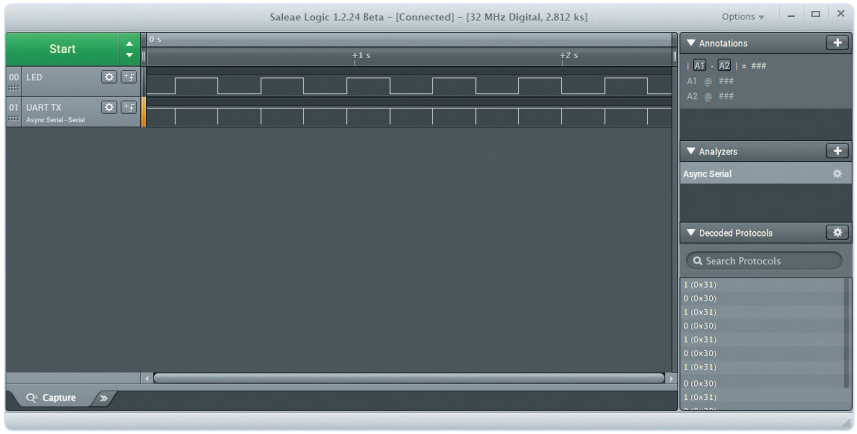


Bild 7. Der Logik-Analyzer registriert alle 500 ms ein gesendetes Zeichen.

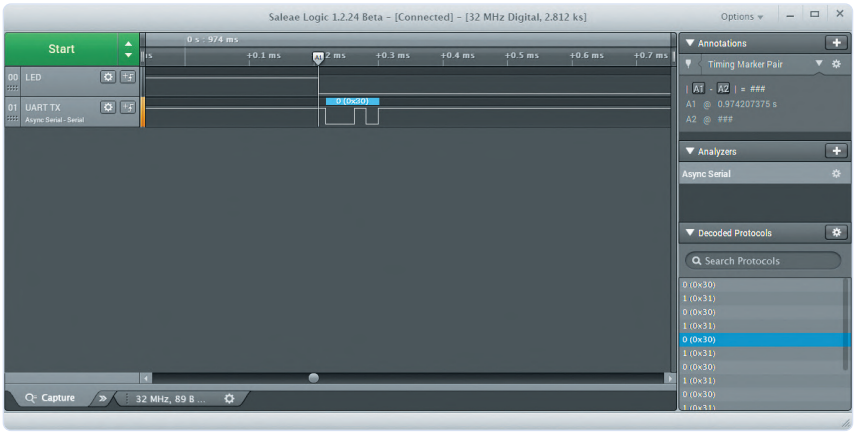


Bild 8. Gezoomter Trace mit dem übertragenen Zeichen.

Sie haben Fragen oder Kommentare?
Haben Sie Fragen oder Anmerkungen zu diesem Artikel? Schicken Sie eine E-Mail an den Autor mathias.claussen@elektor.com oder kontaktieren Sie Elektor unter editor@elektor.com.

Ein Beitrag von
Projekt und Text: **Mathias Claußen**
Redaktion: **Jens Nickel, C. J. Abate**
Übersetzung: **Dr. Thomas Scherer**
Layout: **Giel Dols**

WEBLINKS

[1] **Artikel-Webseite**: www.elektormagazine.de/200479-C-01
[2] **Parallax-Forum**: https://forums.parallax.com
[3] **Smart-Pin-Übersicht von rayman**: https://forums.parallax.com/discussion/171420/smartpin-diagram-now-with-p-p-bit-mode-table/p8
[4] **Datenblatt zu Propeller 2 (vorläufig)**: https://docs.google.com/document/d/1gn6oaT5Ib7CytvlZHacmrSbVBJsD9t_-kmvjd7nUR6o/edit#heading=h.1h0sz9w9bl25

Nucleo-Boards programmieren mit der STM32Cube-IDE

Ein Beispiel-Kapitel: FreeRTOS für die STM32 MCU

Von **Dogan Ibrahim** (Großbritannien)

Diesen Monat präsentieren wir in der Rubrik *Elektor Bücher* einen Auszug aus dem Elektor-Buch *Nucleo Boards Programming with the STM32CubeIDE* von Dogan Ibrahim. Herzstück aller im Buch beschriebenen Projekte ist das kostengünstig erhältliche Entwicklungsboard Nucleo-L476RG und die kostenlose Software STM32CubeIDE. Hard- und Software bilden eine perfekte Kombination für fortgeschrittene Embedded-Anwendungen. In diesem Kapitel führen wir das STM32-Mikrocontrollerboard, die CubeIDE und das freie RTOS zu einem Lernprojekt zusammen.

Anmerkung der Redaktion: Dieser Auszug aus dem Buch *Nucleo Boards Programming with the STM32CubeIDE - Hands-on in More Than 50 Projects* wurde formatiert und leicht bearbeitet, um den redaktionellen Standards und dem Seitenlayout der Zeitschrift *Elektor* zu entsprechen. Da es sich um einen Auszug aus einem umfangreichen Buch handelt, können sich einige Begriffe in diesem Artikel auf Erläuterungen an anderer Stelle des ursprünglichen Buchs beziehen. Autor und Redaktion haben jedoch ihr Bestes getan, um solche Fälle zu vermeiden, und sind gerne bereit, Ihnen bei Fragen zu helfen - die Kontaktangaben finden Sie im Kasten Fragen oder Kommentare?

In den meisten komplexen Echtzeitsystemen muss eine Reihe von Tasks (oder Programmen) fast gleichzeitig abgearbeitet werden. Betrachten wir ein extrem einfaches Beispiel: Eine LED soll in bestimmten Intervallen blinken und gleichzeitig eine Taste überwacht werden. Man könnte in einer Schleife die Taste in regelmäßigen Abständen abfragen und dabei die LED blinken lassen, aber was in diesem einfachen Beispiel funktionieren mag, dürfte in den meisten komplexeren Echtzeitsystemen als Multitasking-Ansatz implementiert werden.

Der Begriff Multitasking bedeutet, dass mehrere Tasks (oder Programme) parallel auf der gleichen CPU abgearbeitet werden. Das ist bei einer einzigen CPU mit einem Kern aber ein Ding der Unmöglichkeit. Daher wird ein Task-Switching durchgeführt, bei dem sich mehrere Tasks die CPU-Zeit teilen. In vielen Anwendungen können Tasks aber nicht unabhängig voneinander laufen, so dass sie in irgendeiner Weise zusammenarbeiten müssen. So kann beispielsweise die Ausführung eines Tasks von der Fertigstellung eines anderen abhängen, oder ein Task benötigt Daten von einem anderen. In solchen Fällen müssen die beteiligten Tasks über eine Art von übergeordneter Kommunikationseinheit synchronisiert werden. Echtzeitsysteme sind zeitabhängige Systeme, bei denen die CPU nie überlastet ist. In solchen Systemen haben Tasks in der Regel Prioritäten, die strikt befolgt werden. Ein Task mit höherer Priorität kann die CPU von einem Task mit niedrigerer Priorität an sich reißen und dann die CPU exklusiv nutzen, bis er die CPU wieder freigibt. Wenn ein Task mit höherer Priorität seine Verarbeitung abschließt oder wenn er darauf wartet, dass eine Ressource verfügbar wird, kann ein Task mit niedrigerer Priorität die CPU an sich reißen und die Verarbeitung an dem Punkt wieder aufnehmen, an dem er unterbrochen wurde. Von Echtzeitsystemen wird auch erwartet, dass sie auf Ereignisse so schnell wie möglich reagieren.

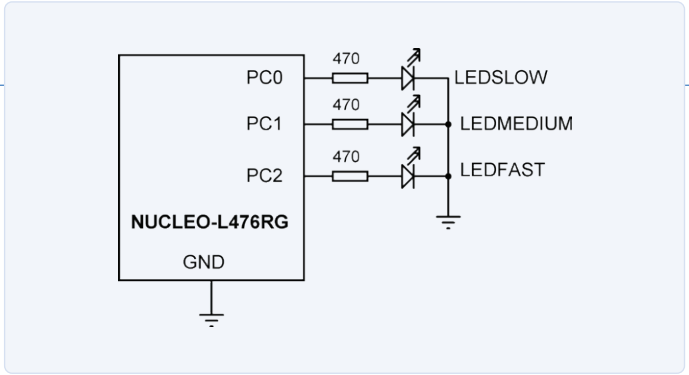


Bild 1. Schaltplan des Beispiels zum Ausführen von Free RTOS auf einem STM32-Mikrocontroller-Board.

Externe Ereignisse werden deshalb in der Regel über externe Interrupts mit möglichst kurzer Interrupt-Latenz verarbeitet, damit die Interrupt-Service-Routine ausgeführt wird, sobald ein Interrupt auftritt.
Es ist nicht allzu schwer, eine Liste von Vorteilen eines Multitasking-Kernels zu erstellen:

- Ohne einen Multitasking-Kernel können zwar mehrere Tasks in einer Schleife ausgeführt werden, aber dieser Ansatz führt zu einer sehr schlechten Echtzeit-Performance, bei der die Ausführungszeiten der Tasks nicht kontrolliert werden können.
- Es ist möglich, die verschiedenen Tasks als Interrupt-Service-Routinen zu programmieren. Dies kann zwar in der Praxis funktionieren, aber wenn eine Anwendung viele Tasks hat, wächst die Anzahl der Interrupts, wodurch der Code weniger handhabbar wird.
- Ein Multitasking-Kernel erlaubt es, problemlos neue Tasks hinzuzufügen oder vorhandene aus dem System zu entfernen.
- Das Testen und Debuggen eines Multitasking-Systems mit einem Multitasking-Kernel ist im Vergleich zu einem Multitasking-System ohne entsprechenden Kernel einfach.
- Mit einem Multitasking-Kernel wird der Speicher besser verwaltet.
- Die Kommunikation zwischen den Tasks ist mit einem Multitasking-Kernel einfacher zu handhaben.
- Die Task-Synchronisation ist mit einem Multitasking-Kernel einfacher zu steuern.
- Die CPU-Zeit lässt sich mit einem Multitasking-Kernel leichter verwalten.
- Die meisten Multitasking-Kernel bieten Speichersicherheit, bei der ein Task nicht auf den Speicherbereich eines anderen Tasks zugreifen kann.
- Die meisten Multitasking-Kernel bieten eine Task-Priorität, bei der Tasks mit höherer Priorität die CPU in Beschlag nehmen und die Ausführung von Tasks mit niedrigerer Priorität stoppen können. Dadurch können wichtige Tasks immer dann ausgeführt werden, wenn es erforderlich ist.

Die Notwendigkeit eines RTOS

Ein RTOS (Real-Time Operating System) ist ein Programm, das die Systemressourcen verwaltet, die Ausführung verschiedener Tasks in einem System plant, die Ausführung von Tasks synchronisiert, die Ressourcenzuweisungen verwaltet und den Nachrichtenaustausch zwischen den Tasks ermöglicht. Jedes RTOS besteht aus einem Kernel, der die Low-Level-Funktionen bereitstellt, hauptsächlich Scheduling, Task-Erstellung, Inter-Task-Kommunikation, Ressourcenverwaltung und so weiter.

Die meisten komplexen RTOSs bieten auch Dienste für die Dateiverarbeitung, Lese- und Schreiboperationen auf der Festplatte, Interrupt-Services, Netzwerkmanagement, Benutzerverwaltung und so weiter.
Ein Task ist ein unabhängiger Ausführungsstrang in einem Multitasking-System, normalerweise mit einem eigenen lokalen Datensatz. Ein Multitasking-System besteht aus mehreren Tasks, die jeweils ihren eigenen Code ausführen, miteinander kommunizieren und sich synchronisieren, um Zugriff auf gemeinsame Ressourcen zu gewähren. Unser (vielleicht einfachstes) Beispiel für ein Multitasking-System besitzt drei LEDs, die in unterschiedlichen regelmäßigen Intervallen blinken sollen. Die Programmierung eines solchen einfachen Systems könnte ohne einen Multitasking-Kernel eine komplizierte Aufgabe werden. Wir werden in diesem Artikel sehen, wie ein Multitasking-Betriebssystem wie FreeRTOS verwendet werden kann, um die MCU-Ressourcen aufzuteilen.
Das einfachste RTOS besteht aus einem Scheduler, der die Ausführungsreihenfolge der Tasks im System bestimmt. Jede Task hat seinen eigenen Kontext aus dem Zustand der CPU und den zugehörigen Registern. Der Scheduler schaltet von einem Task zu einem anderen, indem er eine Kontextumschaltung durchführt, bei der der Kontext des laufenden Tasks gespeichert und der Kontext des folgenden Tasks entsprechend geladen wird, damit die Ausführung des nächsten Tasks ordnungsgemäß fortgesetzt werden kann. Die Zeit, die die CPU für die Kontextumschaltung benötigt, wird als Kontextumschaltzeit bezeichnet und ist normalerweise vernachlässigbar im Vergleich zu den eigentlichen Ausführungszeiten der Tasks.

Das FreeRTOS

FreeRTOS ist ein Echtzeit-Betriebssystem, das auf vielen High-End-Mikrocontrollern und auch auf den Controllern der STM32-Familie läuft. Die STM32CubeIDE hat die FreeRTOS-Software gebündelt, obwohl wir FreeRTOS nicht direkt aufrufen. ARM hat die Bibliothek CMSIS-RTOS erstellt, die es ermöglicht, Aufrufe an ein zugrundeliegendes RTOS wie FreeRTOS zu machen. Das bedeutet, dass wir das zugrundeliegende FreeRTOS über Aufrufe an CMSIS-RTOS (Version 2) steuern.
Multitasking mit FreeRTOS ist ein sehr umfangreiches Thema und würde ebenfalls umfangreiche Literatur erfordern, um alle Funktionen zu erklären. Es gibt mehrere Bücher, Tutorials und Application Notes im Internet, die für Leser hilfreich sein können, die mit den Konzepten des Multitasking nicht vertraut sind. Das Buch *ARM-Based Microcontroller Multitasking Projects - Using the FreeRTOS Multitasking Kernel* kann sehr nützlich sein, um die Konzepte des Multitasking zu verstehen und die Verwendung von FreeRTOS in ARM-basierten MCUs zu erlernen [1].

Ein FreeRTOS-Projekt mit der STM32MCubeIDE

Im weiteren Verlauf dieses Artikels werden wir eine einfache Anwendung mit drei LEDs entwickeln, die an das Entwicklungsboard NUCLEO-L476RG angeschlossen sind. Die LEDs haben die Namen *LEDSLOW*, *LEDMEDIUM* und *LEDFAST*. Die Anschlüsse dieser LEDs und ihre Blinkraten sind (siehe auch **Bild 1**):

LED	Blinkrate	an GPIO-Pin
LEDSLOW	jede Sekunde	PC0
LEDMEDIUM	alle 500 ms	PC1
LEDFAST	alle 250 ms	PC2

Ein erstes Programm

Folgen Sie diesen Schritten:

- Starten Sie STM32CubeIDE
- Starten Sie eine neue Applikation
- Erstellen Sie einen neuen Workspace
- Wählen Sie STM32L476RG als Prozessor.
- Benennen Sie das Programm: *FREE*
- Konfigurieren Sie *PC0*, *PC1*, und *PC2* als *GPIO_Output*. Klicken Sie mit der rechten Maustaste auf die Pins und setzen Sie die User-Labels auf *LEDSLOW*, *LEDMEDIUM* beziehungsweise *LEDFAST* (**Bild 2**)
- Konfigurieren Sie den MCU-Takt auf 80 MHz
- Klicken Sie auf der linken Seite auf *Middleware* und wählen Sie *FreeRTOS*
- Setzen Sie das Interface auf *CMSIS_V2*.
- Es gibt viele Parameter, die auf dem Tab *Configuration* eingestellt werden können. Dies erfordert aber gute Kenntnisse in FreeRTOS, so dass wir in diesem Beispielprojekt besser alle Standardwerte übernehmen (**Bild 3**).
- Klicken Sie auf *File*, dann auf *Save*, und generieren Sie den Code mit *YES*.
- Klicken Sie auf *Core*, dann auf *Src* und zeigen Sie das Hauptprogramm durch einen Doppelklick auf *main c* an

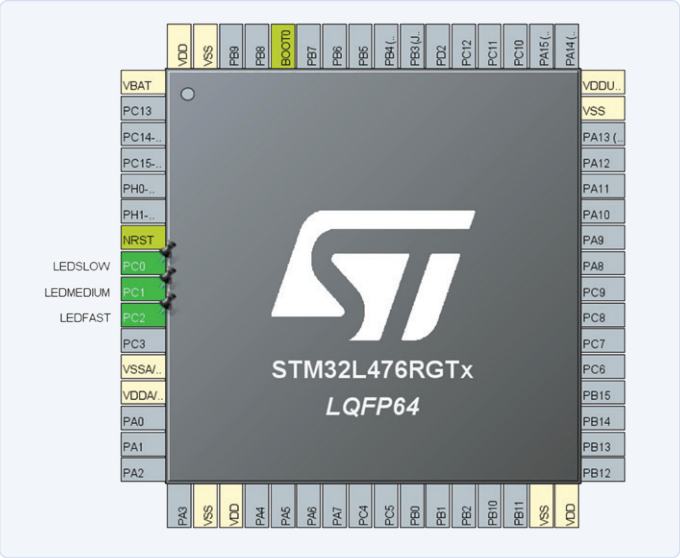


Bild 2: Konfigurieren der STM32-Ausgänge in der STMCube-IDE.

In diesem Programm haben wir drei Tasks, auch Threads genannt. Die Grundfunktionen in einem FreeRTOS-Programm sind wie folgt:

- Thread-IDs definieren
- Thread-Attribute definieren
- den Scheduler initialisieren
- Threads erzeugen
- Starten des Schedulers.

Mit `osThreadId_t` werden die Thread-IDs definiert. Die Thread-IDs in diesem Programm sind:

```
osThreadId_t LEDSTaskHandle; // Langsame LED
osThreadId_t LEDMTaskHandle; // Mittelschnelle LED
osThreadId_t LEDFTaskHandle; // Schnelle LED
```

Die Thread-Attribute definieren verschiedene Parameter eines Threads, zum Beispiel seinen Namen, seine Priorität, die Stackgröße und so weiter. Die Thread-Attribute für den langsamen Task sind zum Beispiel:

```
const osThreadAttr_t LEDSTask_attributes =
{
.name = “LEDSTask”,
.priority = (osPriority_t) osPriorityNormal,
.stack_size = 128 * 4
};
```

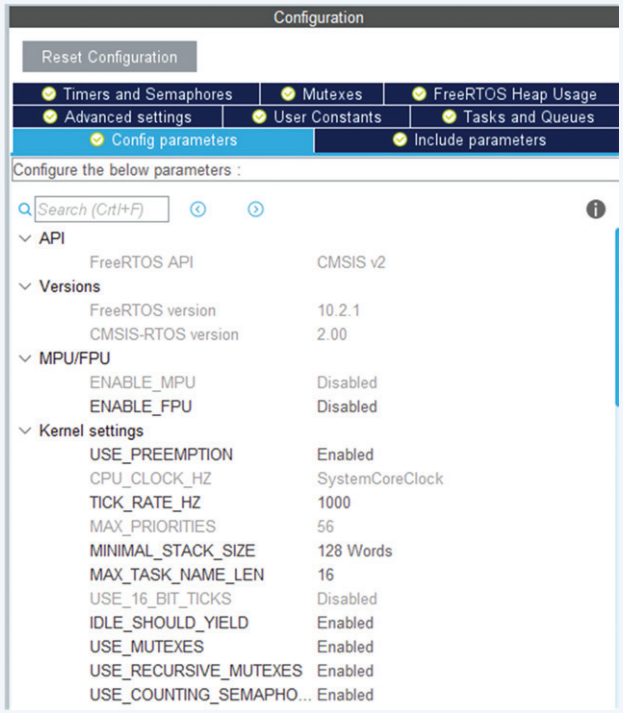


Bild 3. Übernehmen Sie die Standardwerte unter Config parameters.

WEBLINK & LITERATUR

[1] **Dogan Ibrahim: „ARM-Based-Microcontroller-Multitasking-Projects - Using the FreeRTOS“ (Newness, 2020)**

[2] **Programmpaket zum Download**: www.elektor.de/nucleo-boards-programming-with-the-stm32cubeide

Mit den Funktionsaufrufen wird der Scheduler initialisiert und gestartet:

```
osKernelInitialize();
osKernelStart();
```

Threads werden mit dem Aufruf der Funktion `osThreadNew()` erzeugt. Der Thread für die langsame LED beispielsweise wird wie folgt angelegt:

```
LEDSTaskHandle = osThreadNew(StartLEDSTask, NULL,
   &LEDSTask_attributes);
```

Dabei ist `StartLEDSTask` der Name der Funktion, die vom Scheduler aufgerufen wird, um die langsam blinkende LED zu realisieren. Ihr Inhalt ist:

```
void StartLEDSTask(void *argument)
{
for(;;)
{
HAL_GPIO_TogglePin(GPIOC, LEDSLOW_Pin);
osDelay(1000);
}
}
```

Anstelle von `HAL_Delay()` verwenden wir `osDelay()`, da `HAL_Delay()` in einem hochprioren Task einen Kontextwechsel verhindern könnte. Mit `osDelay()` dagegen sagt er dem Scheduler, dass er zu einem anderen Task wechseln soll, während er wartet.

Kompilieren Sie das Programm im Release-Modus und ziehen Sie die Binärdatei *FREE.bin* per Drag & Drop auf das Device *NUCLEO-L476RG*. Jetzt laufen alle drei Threads gleichzeitig in ihren eigenen Endlosschleifen, so dass die drei LEDs gleichzeitig mit unterschiedlichen Geschwindigkeiten blinken. Der Scheduler schaltet dabei die Threads ein und aus, um den Anschein zu erwecken, dass drei Threads gleichzeitig ausgeführt werden.

Das Programm

Listing 1 zeigt das Programm namens FREE aus dem freien Software-Archiv (.zip), das von der Elektor-Webseite für das Buch [2] heruntergeladen werden kann. Scrollen Sie auf dieser Seite nach unten zu *Downloads* und wählen Sie die Datei *Software_Nucleo Boards Programming with the STM32CubeIDE*. Entpacken Sie das Archiv auf Ihrer Festplatte und gehen Sie dann in den Ordner *FREE*.

210236-02

Haben Sie Fragen oder Kommentare?
Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Schicken Sie eine E-Mail an den Autor unter d.ibrahim@btinternet.com oder Elektor unter editor@elektor.com.

Ein Beitrag von
Text: **Dogan Ibrahim**
Übersetzung: **Rolf Gerstendorf**
Redaktion: **Jan Buiting**
Layout: **Giel Dols**

Listing 1. Das Programm FREE.

```
/* USER CODE BEGIN Header */
/**
******************************************************************************
* @file : main.c
* @brief : Main program body
******************************************************************************
* @attention
*
* Copyright (c) 2020 STMicroelectronics.
* All rights reserved.
*
* This software component is licensed by ST under Ultimate Liberty license
* SLA0044, the «License»; You may not use this file except in compliance with
* the License. You may obtain a copy of the License at:
* www.st.com/SLA0044
*
******************************************************************************
*/
#include «main.h»
#include «cmsis_os.h»
//
// Define Thread IDs
//
osThreadId_t LEDSTaskHandle;          // Slow LED
osThreadId_t LEDMTaskHandle;          // Medium LED
osThreadId_t LEDFTaskHandle;          // Fast LED
```

```
//
// Slow LED task. Flash every second
//
void StartLEDSTask(void *argument)
{
   for(;;)
   {
     HAL_GPIO_TogglePin(GPIOC, LEDSLOW_Pin);
     osDelay(1000);
   }
}
//
// Medium LED task. Flash every 500ms
//
void StartLEDMTask(void *argument)
{
   for(;;)
   {
     HAL_GPIO_TogglePin(GPIOC, LEDMEDIUM_Pin);
     osDelay(500);
   }
}

//
// Fast LED task. Flash every 250ms
//
void StartLEDFTask(void *argument)
{
   for(;;)
   {
     HAL_GPIO_TogglePin(GPIOC, LEDFAST_Pin);
     osDelay(250);
   }
}

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
void StartDefaultTask(void *argument);
//
// Start of main program
//
int main(void)
{
   HAL_Init();
   SystemClock_Config();
   MX_GPIO_Init();
   //
   // Slow LED Task attributes
   //
   const osThreadAttr_t LEDSTask_attributes =
   {
      .name = «LEDSTask»,
      .priority = (osPriority_t) osPriorityNormal,
      .stack_size = 128 * 4
   };
   //
   // Medium LED Task attributes
   //
   const osThreadAttr_t LEDMTask_attributes =
   {
      .name = «LEDMTask»,
```

```
        .priority = (osPriority_t) osPriorityNormal,
        .stack_size = 128 * 4
    };
    //
    // Fast LED Task attributes
    //
    const osThreadAttr_t LEDFTask_attributes =
    {
        .name = «LEDFTask»,
        .priority = (osPriority_t) osPriorityNormal,
        .stack_size = 128 * 4
    };

    /* Init scheduler */
    osKernelInitialize();

    /* creation of Tasks */
    LEDSTaskHandle = osThreadNew(StartLEDSTask, NULL, &LEDSTask_attributes);
    LEDMTaskHandle = osThreadNew(StartLEDMTask, NULL, &LEDMTask_attributes);
    LEDFTaskHandle = osThreadNew(StartLEDFTask, NULL, &LEDFTask_attributes);

    /* Start scheduler */
    osKernelStart();

    while (1)
    {
    }
}

void SystemClock_Config(void)
{
    RCC_OscInitTypeDef RCC_OscInitStruct = ;
    RCC_ClkInitTypeDef RCC_ClkInitStruct = ;

    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
    RCC_OscInitStruct.HSIState = RCC_HSI_ON;
    RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
    RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
    RCC_OscInitStruct.PLL.PLLM = 2;
    RCC_OscInitStruct.PLL.PLLN = 20;
    RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV7;
    RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
    RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
    if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
    {
      Error_Handler();
    }

    RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                                  |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
    RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
    RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
    RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
    RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_4) != HAL_OK)
    {
      Error_Handler();
    }
```

```
  if (HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1) != HAL_OK)
   {
     Error_Handler();
   }
}

static void MX_GPIO_Init(void)
{
   GPIO_InitTypeDef GPIO_InitStruct = ;

  /* GPIO Ports Clock Enable */
   __HAL_RCC_GPIOC_CLK_ENABLE();
   /*Configure GPIO pin Output Level */
   HAL_GPIO_WritePin(GPIOC, LEDSLOW_Pin|LEDMEDIUM_Pin|LEDFAST_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pins : LEDSLOW_Pin LEDMEDIUM_Pin LEDFAST_Pin */
   GPIO_InitStruct.Pin = LEDSLOW_Pin|LEDMEDIUM_Pin|LEDFAST_Pin;
   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
   GPIO_InitStruct.Pull = GPIO_NOPULL;
   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
   HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
}

void Error_Handler(void)
{

}

#ifdef USE_FULL_ASSERT

void assert_failed(uint8_t *file, uint32_t line)
{

}
#endif
/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

PASSENDE PRODUKTE

- Buch: Nucleo Boards Programming with the STM32CubeIDE
  www.elektor.de/nucleo-boards-programming-with-the-stm32cubeide
- E-Buch: Nucleo Boards Programming with the STM32CubeIDE
  www.elektor.de/nucleo-boards-programming-with-the-stm32cubeide-e-book
- STM32 Nucleo L476RG Board
  www.elektor.de/stm32-nucleo-l476rg-board

Erweiterung des MT3608-Gleichspannungs-Aufwärtsschaltregler-Moduls

Von **Johannes Sturz**

Im Internet lassen sich alle möglichen interessanten preiswerten Elektronikmodule finden. Das Angebot deckt so ziemlich alles ab, von Sensoren und Stromversorgungen bis hin zu Audio-Verstärkern und Mikrocontroller-Boards.

Eines dieser Module ist ein Gleichspannungs-Aufwärtswandler auf Basis des ICs MT3608 von Aerosemi (**Bild 1**). Es wandelt eine Eingangsspannung im Bereich von 2..24 V in eine mindestens 0,6 V höhere und maximal 28 V betragende Spannung um. Die Ausgangsspannung wird mit einem Trimmpoti eingestellt. Das IC kann - natürlich bei entsprechender Kühlung - bis zu 2 A liefern. Um die Schaltung des Step-up-Moduls einfach zu halten (**Bild 2**), wurden einige Funktionen des ICs deaktiviert, aber nicht so, dass man sie mit einem Lötkolben nicht wiederbeleben könnte.

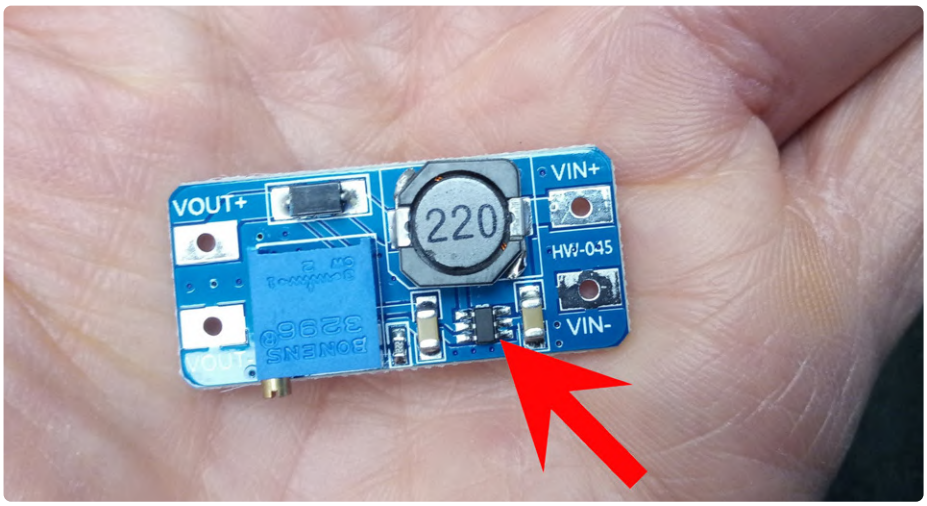


Bild 1. Das preiswerte Aufwärtswandler-Modul MT3608 kann aus einer Eingangsspannung von nur 2 V bis zu 28 V erzeugen. Der Pfeil zeigt auf Pin 4.

Hinzufügen eines Enable-Eingangs

Der MT3608 verfügt über einen Enable-Eingang, um den Wandler ein- und auszuschalten. Wenn die Spannung am Enable-Pin 1,5 V beträgt oder höher ist, schaltet der Wandler ein; wenn sie niedriger ist, beträgt die Ausgangsspannung V_{IN} minus die Durchlassspannung der Schottky-Diode D1 (etwa 0,3 V). Der Enable-Pin ist nicht nur nützlich, weil er das Ein- und Ausschalten des Moduls ermöglicht, sondern auch eine PWM-Steuerung ermöglicht. Wie wir weiter unten sehen werden, ist dies praktisch, um beispielsweise die Helligkeit einer oder mehrerer LEDs zu steuern.

Beim Step-Up-Modul ist der Enable-Pin (Pin 4) durch eine Leiterbahn mit dem Eingang des ICs (Pin 5) verbunden. Um den Enable-Pin zu verwenden, muss diese Verbindung mit einem Skalpell durchtrennt (oder Pin 4 entlötet und isoliert) werden. Eine Lupe oder ein Mikroskop kann dabei hilfreich sein.

Konstantstrom-Ausgang oder LED-Treiber

Der Feedback-Pin des MT3608 ist über einen Spannungsteiler (P1 und R1) mit der Ausgangsspannung verbunden. Dadurch kann das IC seine Ausgangsspannung so einstellen, dass die Spannung an seinem Feedback-Pin konstant bei 0,6 V bleibt.

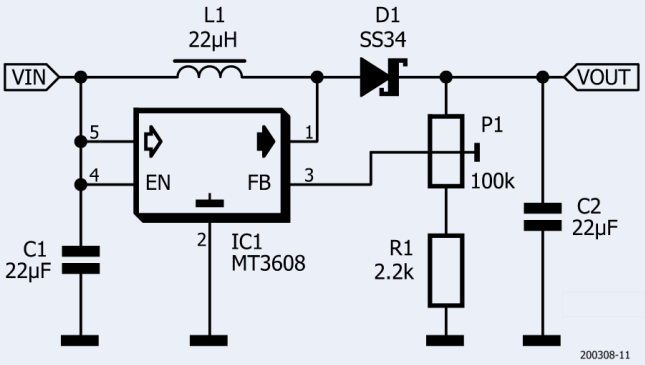


Bild 2. Die Schaltung des preiswerten und beliebten Aufwärtswandler-Moduls auf Basis des MT3608.

Bild 4. Bei defekten LED-Lampen ist in der Regel die Steuerschaltung beschädigt. Die LEDs sind meist nicht betroffen und dürfen weiterleben.

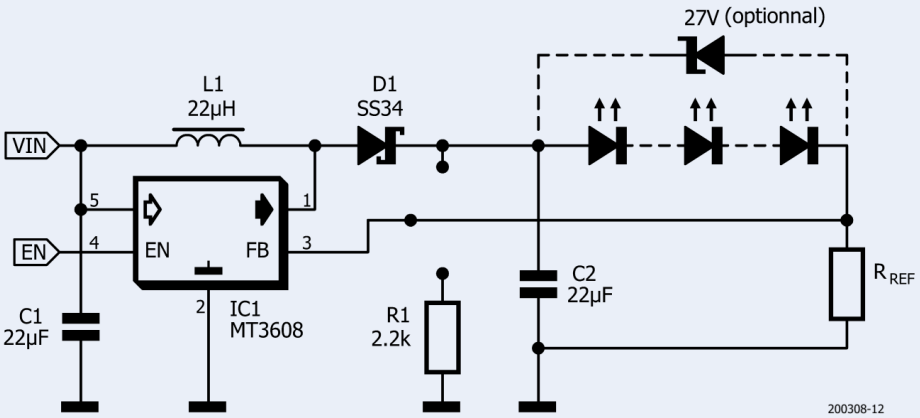


Bild 3. Das MT3608-Aufwärtswandler-Modul als dimmbarer LED-Treiber. Dazu ist es notwendig, Pin 4 des ICs von VIN+ zu trennen und das Trimmpoti zu entfernen.

Wenn man den oberen Teil des Spannungsteilers durch eine oder mehrere LEDs ersetzt, wird der Aufwärtswandler zu einem Konstantstromtreiber, bei dem der Strom durch die untere Hälfte des Spannungsteilers bestimmt wird. Das ist so, weil die Spannung über einer LED (fast) unabhängig vom Strom ist, der durch sie fließt, so dass nur die Spannung über der unteren Hälfte des Spannungsteilers einer variierenden Ausgangsspannung folgt. Diese wiederum bestimmt den durch ihn fließenden Strom, der nach dem ersten Kirchhoff'schen Gesetz auch der Strom durch die LED(s) ist.

Entfernt man das Trimmpoti, ist der Feedback-Pin des ICs abgeklemmt und kann mit einem externen Widerstand zur Stromeinstellung (R_{ref}) in Reihe mit einer LED-Kette ausgestattet werden, wie in **Bild 3** gezeigt. Der Strom wird berechnet mit:

$$I_{LED} = 0{,}6\ V / R_{ref}$$

Die Summe der Spannungsabfälle über den LEDs plus 0,6 V darf dabei 28 V nicht überschreiten, ansonsten stirbt der MT3608.

Zum Beispiel: Dimmen von LEDs mit Arduino

Aus einer kaputten 9-W-LED-Lampe mit sieben LEDs wurden die noch funktionsfähigen LEDs ausgeschlachtet (**Bild 4**). Da aber jede LED eigentlich aus drei in Reihe geschalteten LED-Chips besteht, zählt der String insgesamt 21 LED-Chips. Der Spannungsabfall über den LEDs beträgt bei etwa 2,4 V Spannungsabfall pro LED-Chip etwa 50 V, was einen LED-Strom von etwa 9 W/50 V = 180 mA ergibt.

Schaltet man drei LEDs, also insgesamt neun LED-Chips zu einer Kette zusammen, ergibt sich ein Spannungsabfall von etwa 22 V, was im Ausgangsbereich des MT3608 liegt. Für einen Strom von 180 mA durch die LEDs muss R_{ref} = 0,6 V/0,180 A = 3,3 Ω betragen. Die Verlustleistung beträgt rund 110 mW, so dass die LEDs eine gute Kühlung benötigen, um nicht den Hitzetod zu sterben.

Mit einem einfachen Arduino-Sketch [1] ist es möglich, die Helligkeit der LEDs zu steuern, indem Werte von 0 bis 255 über den Seriellen Monitor gesendet werden. Pin 9 des Arduino-Boards wird dabei als PWM-Ausgang konfiguriert.

Beachten Sie, dass bei den Experimenten das Aufwärtswandlermodul über den USB-Anschluss mit Strom versorgt wird. Um ein

Ansprechen der Sicherung auf dem Arduino-Board zu vermeiden, wurde der LED-Strom auf 20 mA reduziert. Für höhere Ströme wird also eine externe Stromversorgung benötigt. Vergessen Sie auch nicht, auch den MT3608 zu kühlen, da sein winziges Gehäuse nur für 0,6 W Verlustleistung spezifiziert ist.
Dieses Beispiel zeigt, dass mit ein paar einfachen Modifikationen ein preiswertes Aufwärtswandler-Modul in einen universellen LED-Treiber verwandelt werden kann.

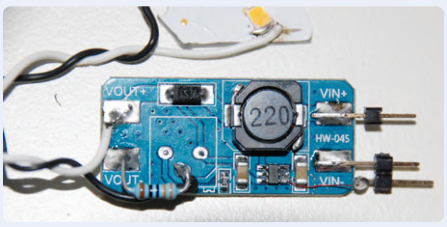
Tipps:

- Streng genommen ist es nicht notwendig, das Trimmpoti zu entfernen, aber es verbessert den Wirkungsgrad ein wenig. Wenn Sie das Trimmpoti an seinem Platz belassen, sollten Sie V_{OUT} auf Maximum stellen.
- Bei einem Ausgangsstrom von 100...300 mA ist der Wirkungsgrad besser als 90%.
- Der Spannungsabfall über den LED-Strängen muss höher sein als die Eingangsspannung, sonst können sie nicht vollständig abgeschaltet werden.
- Wenn der Enable-Pin low ist, reduziert sich die Stromaufnahme auf wenige Nanoampere (vorausgesetzt, das Trimmpoti wurde entfernt).
- Wenn der modifizierte Aufwärtswandler ohne Last verwendet wird (zum Beispiel aufgrund eines Kabelbruchs), kann der MT3608 durchschlagen und beschädigt werden. Eine 27-V-Zener-Diode zwischen V_{OUT+} und FB, die die Leistung aufnehmen können muss, kann dies verhindern.
- Wenn der Spannungsabfall über dem LED-String mehr als 27 V betragen würde, teilen Sie den LED-String in mehrere Segmente auf. Dann können Sie entweder jedes String-Segment mit einem eigenen Modul ansteuern (mit verbundenen Enable-Eingängen) oder alle Segmente parallel mit einem Modul ansteuern, indem Sie einen Widerstand in Reihe mit jedem LED-String-Segment hinzufügen. Das verringert natürlich die Effizienz.
- Es ist wichtig, für eine ausreichende Kühlung der LEDs zu sorgen. Die meisten LED-Lampen sterben durch Überhitzung. Sorgen Sie auch für die Kühlung des MT3608-Moduls.

200308-02

Haben Sie Fragen oder Kommentare?
Haben Sie Fragen oder Kommentare? Dann schicken Sie eine E-Mail an den Redakteur: clemens.valens@elektor.com.

Ein Beitrag von
Beitrag: **Johannes Sturz**
Redaktion: **Clemens Valens**
Übersetzung: **Rolf Gerstendorf**
Layout: **Harmen Heida**



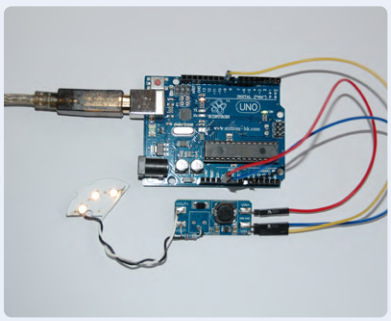


Bild 5. Ein Arduino Uno und ein modifiziertes MT3608-Modul steuern gemeinsam die Helligkeit einer LED-Kette.

WEBLINKS

[1] **Downloads für diesen Artikel**: www.elektormagazine.de/200308-02

MyVanitar - Tutorials für Elektronik-Enthusiasten

Elektor-Labs-User Hesam Moshiri stellt auf seinem YouTube-Kanal MyVanitar Video-Tutorials und Projekte für Elektronik-Enthusiasten vor.

Bauteil-Tester
Die meisten Oszilloskope verfügen über einen XY-Modus, der die Spannung auf einem Kanal als Funktion der Spannung auf einem anderen Kanal und nicht als Funktion der Zeit anzeigt. Dieser Modus wird verwendet, um die berühmten Lissajous-Figuren zu erstellen, aber mit etwas Einfallsreichtum können Sie ihn auch als Kennlinienschreiber verwenden, um Bauteile zu prüfen.
Dieses Video zeigt, wie Sie den XY-Modus, einen Funktionsgenerator und einen 10-Ω-Widerstand verwenden, um einen Bauteiltester oder V-I-Kennlinienschreiber einzurichten. Damit können Sie zum Beispiel die Kniespannung einer Zener-Diode oder die Durchlassspannung einer unbekannten LED bestimmen.

www.elektor-labs.com/4243

I²C-Dekodierung
In einem weiteren Video zeigt Hesam, wie man mit einem Oszilloskop mit der Fähigkeit zur Dekodierung eines seriellen Protokolls (was heutzutage Standard ist) Signale auf einem I²C-Bus dekodieren kann. Das Einrichten erfordert oft etwas Tüftelei, aber wenn man es einmal geschafft hat, ist Bob's your uncle (wie Hesam es ausdrückt).

www.elektor-labs.com/4281

Digitale Pinzette
Miniware DT71

Von **Harry Baggen** (Niederlande)

Zur Identifikation und Vermessung von Bauteilen sind sogenannte Messpinzetten ein sehr handliches Werkzeug. Besonders bei SMDs sind diese eine tolle Lösung. Die DT71 von Miniware kann alle gängigen Arten von passiven Bauteilen automatisch identifizieren und messen und bietet darüber hinaus einige zusätzliche Funktionen. Das wahre Highlight ist aber eine einzigartige Konstruktion, ein Display, das sich drehen lässt.

Es gibt verschiedene Arten von „Pinzetten" zum Bestimmen von passiven Bauteilen. Die besseren Versionen sind recht teuer und liegen irgendwo zwischen 200 € und 300 €, die billigen sind schon für 20...30€ zu bekommen. Allerdings können diese billigeren Pinzetten meist keine Induktivitäten messen und weisen eine geringe mechanische Qualität auf. Als ich die technischen Daten der DT71-Pinzette von Miniware studierte, ordnete ich sie direkt unter die erste Kategorie ein, da sie wirklich alles bietet, was man von einem solchen Instrument erwarten darf. Und dennoch kostet sie kaum mehr als die Low-Budget-Modelle!

Design

Das erste, was Ihnen beim Auspacken der DT71 auffällt, sind die Abmessungen. Sie ist 14 cm lang und wiegt weniger als 25 g. Das Gerät besteht aus dem Pinzettenteil und dem Displayteil mit einem kleinen OLED-Display. Die Teile werden über einen 4-poligen 3,5-mm-Klinkenverbinder zusammengesteckt. Auf der Oberseite des Displayteils befindet sich ein Touch-Sensor, der zur Steuerung aller Funktionen dient. Das Displayteil kann sich wegen der Steckverbindung gegenüber der Pinzette drehen. Außerdem ist ein Neigungssensor eingebaut, der erkennt, ob Sie die Pinzette in der linken oder rechten Hand halten und der die Displayausrichtung entsprechend anpasst. Das gesamte Gerät ist aus Kunststoff gefertigt und besitzt ein annehmbares Äußeres.
Die Messarme besitzen rote und blaue Polaritätsanzeigen. Die Metallmessspitzen sind vergoldet und austauschbar. Eine Besonderheit sind die Federung in den Messarmen. Anstelle eines Federmechanismus, der die Messarme auseinander hält, werden zwei Magnetpaare verwendet, bei denen sich zwei Magnete anziehen und zwei Magnete abstoßen. Durch diese Anordnung der Magnetpaare ergibt sich eine sehr weiche Federwirkung.
Das DT71 wird in einer kleinen Kunststoffbox geliefert (**Bild 1**), die neben der Pinzette und dem Displayteil auch zwei Ersatzmessspitzen und ein Adapterkabel mit USB-C-Stecker enthält,

Bild 1. Die DT71-Pinzette besteht aus zwei Teilen und wird mit einem Satz Ersatzmessspitzen und einem speziellen Lade-/Anschlusskabel geliefert.

der einerseits zum Laden der eingebauten Lithium-Akkus (**Bild 2**) im Pinzettenteil verwendet wird, andererseits den Anschluss des Displayteils an einen Computer zum Ändern der Einstellungen und für Firmware-Upgrades ermöglicht. Ein USB-Kabel und ein Steckernetzteil sind nicht im Lieferumfang enthalten, aber die meisten von uns sollten so etwas irgendwo herumliegen haben.

Messfähigkeiten

Der Hersteller hat sich sehr bemüht, so viele Messmöglichkeiten wie möglich im DT71 unterzubringen. Da wäre zunächst einmal die Messung von Widerständen, Dioden, Kondensatoren und Induktivitäten. In der Auto-Einstellung ermittelt das Gerät selbstständig, welche Art von Bauteil am wahrscheinlichsten ist und zeigt dessen Wert auf dem Display an. Außerdem kann das DT71 Frequenzen bis zu 20 MHz und Gleichspannungen bis zu 40 V messen.
Zusätzlich ist auch ein einfacher Frequenzgenerator eingebaut, der Sinus, Rauschen und Impulse mit einem Spitze-Spitze-Wert von etwa 3 V erzeugen kann. Es gibt sogar eine Benutzereinstellung für beliebige Wellenformen (maximal 100 Punkte)! Dazu ist es allerdings erforderlich, den Anzeigeteil an einen PC anzuschließen und die *CAL.INI*-Datei im Speicher der Pinzette in hexadezimaler Notation zu modifizieren (**Bild 3**). Nettes Feature, aber mir erscheint es nicht ganz einfach, auf diese Weise schnell eine Wellenform zu definieren. Ein kleines Zusatzprogramm dafür wäre sehr praktisch gewesen!
Die *CAL.INI*-Datei enthält auch einige Parameter, die Sie nach Belieben einstellen können, zum Beispiel die Zeit, nach der sich das Gerät automatisch ausschaltet, die Displayausrichtung, die Displayhelligkeit und die verschiedenen vorprogrammierten Frequenzwerte für die Sinus-, User- und Pulssignale. Alle Einstellmöglichkeiten sind detailliert im Handbuch beschrieben, das Sie ebenso wie die aktuelle Firmware im Forum von Miniware [1] finden.

In der Praxis

Ich habe den DT71 mit einer Handvoll Bauteile (bedrahtet und SMD) aus meiner Sammlung getestet und mit einem anderen Bauteiltester und einem genauen Multimeter verglichen. Miniware gibt einen maximalen Fehler von 0,5 % für Widerstände, 2 % für Kondensatoren, 5 % für Induktivitäten und 1 % für Gleichspannungen an. Dies ist mehr als ausreichend zur groben Bemessung von Bauteilen. Es werden ohnehin nur drei Stellen (in manchen Fällen vier) angezeigt.
Bei der Arbeit mit dem DT71 ist mir aufgefallen, dass das Display (**Bild 4**) zwar schön scharf, aber doch sehr klein ist. Ich hätte es gerne etwas größer gehabt. Das federnde Verhalten der Pinzettenarme mit den Magneten ist dagegen sehr angenehm, aber die Metallmessspitzen sind nicht scharf genug: Sie rutschen leicht vom Bauteil ab, besonders wenn es auf einer Platine verlötet ist. Der Hersteller hat aber angedeutet, dass es in naher Zukunft verschiedene Arten von Messspitzen geben soll.
Die wichtigste Funktion eines solchen Pinzetteninstruments ist meiner Meinung nach, den Bauteiltyp zu identifizieren. Gerade bei SMDs kann man oft nicht erkennen, um was es sich eigentlich

Bild 2. Display und Pinzettenteil lassen sich mit dem Adapterkabel verbinden und über ein USB-C-Kabel aufladen.

```
/*****************************************************

          DT71 calibration parameter file

*****************************************************/
SLEEP_TIME=60
DISPLAY_DIRECTION=4
OLED_BRIGHTNESS=2
SINE_FREQ_OPT=0
NOISE_FREQ_OPT=1
USER_FREQ_OPT=2
PULSE_FREQ_OPT=3
USER_WAVEFORM = {
 0x7FF, 0x87F, 0x8FF, 0x97E, 0x9FC, 0xA77, 0xAF0, 0xB66, 0xBD9, 0xC48,
 0xCB2, 0xD18, 0xD78, 0xDD3, 0xE29, 0xE77, 0xEC0, 0xF01, 0xF3C, 0xF6F,
 0xF9A, 0xFBE, 0xFDA, 0xFEE, 0xFFA, 0xFFE, 0xFFA, 0xFEE, 0xFDA, 0xFBE,
 0xF9A, 0xF6F, 0xF3C, 0xF01, 0xEC0, 0xE77, 0xE29, 0xDD3, 0xD78, 0xD18,
 0xCB2, 0xC48, 0xBD9, 0xB66, 0xAF0, 0xA77, 0x9FC, 0x97E, 0x8FF, 0x87F,
 0x7FE, 0x77E, 0x6FE, 0x67F, 0x601, 0x586, 0x50D, 0x496, 0x424, 0x3B5,
 0x34B, 0x2E5, 0x285, 0x22A, 0x1D4, 0x186, 0x13D, 0x0FC, 0x0C1, 0x08E,
 0x063, 0x03F, 0x023, 0x00F, 0x003, 0x000, 0x003, 0x00F, 0x023, 0x03F,
 0x063, 0x08E, 0x0C1, 0x0FC, 0x13D, 0x186, 0x1D5, 0x22A, 0x285, 0x2E5,
 0x34B, 0x3B5, 0x424, 0x497, 0x50D, 0x586, 0x601, 0x67F, 0x6FE, 0x77E,
 0x000, 0x000, 0x000, 0x000, 0x000, 0x000, 0x000, 0x000, 0x000, 0x000,
 0x000, 0x000, 0x000, 0x000, 0x000, 0x000, 0x000, 0x000, 0x000, 0x000,
 0x000, 0x000, 0x000, 0x000, 0x000, 0x000, 0x000, 0x000,
}

                LV       MV       HV       RL       RM       RH       CX1      CX2      CX3      CX4      CX5
    CX6      CX7      CX8      CX9      CH1      CH2      LX1      LX2      LX3      LX4      LX5      LX6
CALRB_K0 =   -1.856,   5.286, 109.890, -10.059,  -9.349, 167.794,  -5.263,  -5.263,  -5.263,   0.000,   0.000,
  0.000,   0.000,   0.000,   0.000,   0.000,   0.000,   0.000,   0.000,   0.000,   0.000,   0.000,   0.000,
CALRB_K1 =    1.045,   1.011,   1.009,   1.020,   1.021,   1.008,   1.000,   1.000,   1.000,   1.000,   1.000,
  1.000,   1.000,   1.000,   1.000,   1.000,   1.000,   1.000,   1.000,   1.000,   1.000,   1.000,   1.000,
```

Bild 3. Die Konfigurationsdatei CAL.INI enthält eine Reihe benutzerkonfigurierbarer Parameter, eine Hexadezimaltabelle für den Funktionsgenerator und einige Kalibrierwerte.

Bild 4. Das OLED-Display zeigt normalerweise nur den Wert an, aber in der Auto-Einstellung erscheint wie bei dieser Induktivität manchmal auch ein zweiter Wert.

Bild 5. Der Sinus (hier 5 kHz) des Signalgenerators ist etwas abgeflacht und die Quantisierungsstufen sind deutlich sichtbar.

handelt. Die DT71-Pinzette erfüllt im Allgemeinen diesen Job sehr gut, aber es gibt ein paar Situationen, in denen es schief geht. Das passiert vor allem bei solchen Bauteilen, bei denen der Unterschied zwischen Induktivität und Kapazität schwer zu erkennen ist (zum Beispiel bei sehr kleinen Induktivitätswerten). Aber hier muss ich erwähnen, dass dies auch bei anderen Bauteiltestern nicht ganz unproblematisch ist. Wenn man aber weiß, um welche Art von Bauteil es sich handelt und auf manuellen Betrieb umschaltet, dann wird der richtige Wert angezeigt.

Die Genauigkeit erweist sich als viel besser, als ich erwartet hatte. Bei Widerständen und Induktivitäten lag sie gut innerhalb der Spezifikationen. Bei den Kondensatoren lagen die Messergebnisse der verschiedenen Tester um einige Prozent auseinander. Das hat wohl unter anderem mit der Messmethode zu tun. Der DT71 zeigte in der Regel ein paar Prozent zu wenig an, was aber nicht weiter schlimm war. Die Induktivitäten, die gemessen wurden, lagen alle innerhalb einer Toleranz von 5 %. Bei Dioden muss man darauf achten, dass die Polarität in Bezug auf die Plus- und Minusspitze stimmt, sonst zeigt die DT71 nichts an. Eine LED blinkt, wenn sie richtig angeschlossen ist, aber bei blauen und weißen LEDs funktioniert diese Messung nicht, weil die Messspannung für diese Diodentypen nicht hoch genug ist.

Bei Gleichspannungen lag mein Gerät nur um 0,1 % daneben. Auch hier muss man die Polarität im Auge behalten, sonst zeigt das Messgerät *Negative* an. Bei den Frequenzmessungen lag der Fehler innerhalb von 0,1 %. Der Signalgenerator erzeugt eine Sinusform, bei der die „Spitzen" etwas abgeflacht sind. Man kann auch die Quantisierungsstufen vor allem bei tiefen Frequenzen noch deutlich erkennen (**Bild 5**). Für Audiomessungen ist die Pinzette daher nicht unmittelbar geeignet, aber als Produzent für ein Testsignal ist sie auf jeden Fall brauchbar. Das Impulsssignal ist in Wirklichkeit eine Rechteckwelle, die auch bei 100 kHz noch gut in Form ist.

Ich fand die DT71 sehr angenehm zu bedienen. Die Pinzette schaltet sich automatisch ein, wenn man sie in die Hand nimmt (ab Softwareversion 1.08), und die Displayanzeige dreht sich automatisch um, wenn man die DT71 in die andere Hand nimmt. Der Federdruck zwischen den Armen ist sehr gering, so dass in Verbindung mit ihrem geringen Gewicht diese Pinzette sehr angenehm in der Handhabung ist.

Ein vielseitiges Instrument

Die DT71 von Miniware ist ein sehr komfortables Smart-Tweezers-Messgerät, das nicht nur verschiedene passive Komponenten identifiziert, sondern auch viele zusätzliche Funktionen wie Frequenz- und Spannungsmessungen bietet. Es kann auch als Mini-Signalgenerator fungieren. Der zweiteilige Aufbau der DT71, das drehbare Display und die Magnetfedern machen das Gerät einzigartig. Mein einziger Kritikpunkt ist, dass ich das Display etwas klein fand; es hätte durchaus etwas größer ausfallen können. Aber ansonsten ist DT71 ein vielseitiges Messgerät, über das sich jeder Elektronikbegeisterte bestimmt an seinem nächsten Geburtstag freuen würde!

210182-03

WEBLINK

[1] **Miniware-Forum**: https://minidso.com/forum.php?mod=viewthread&tid=4244

Haben Sie Fragen oder Kommentare?
Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Senden Sie eine E-Mail an editor@elektor.com!

Ein Beitrag von

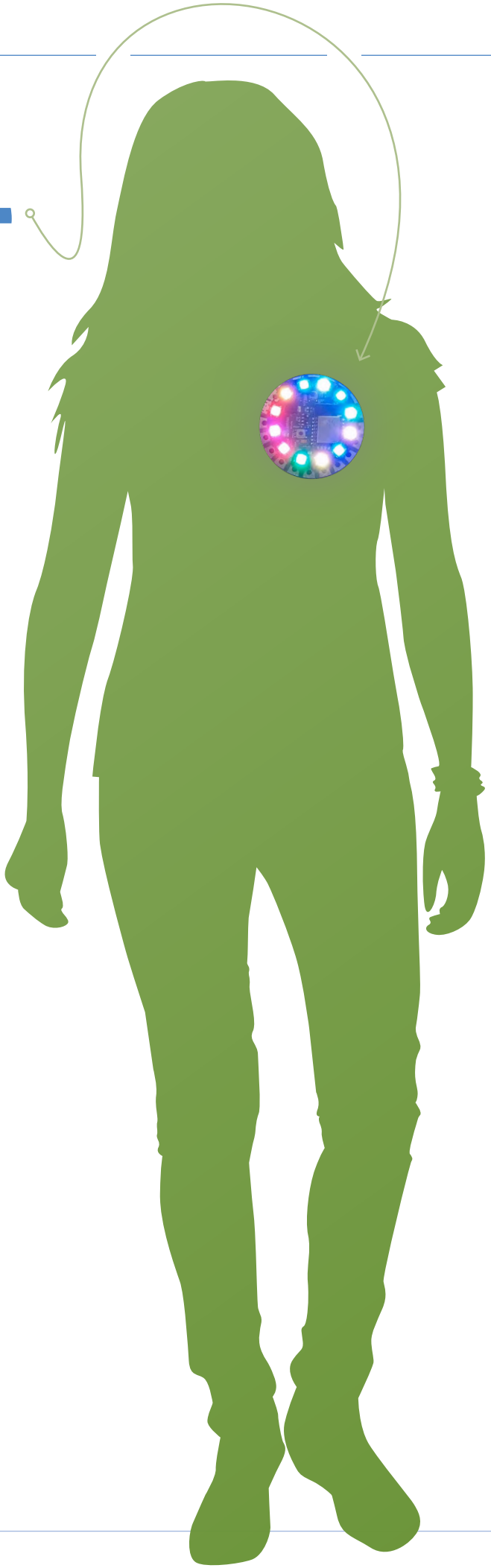
Text und Abbildungen: **Harry Baggen**
Übersetzung: **Rolf Gerstendorf**
Redaktion: **Jens Nickel**
Layout: **Giel Dols**

PASSENDE PRODUKTE

› **Miniware DT71 Mini Digital Tweezers**
www.elektor.com/miniware-dt71-mini-digital-tweezers

Wearable-WLAN Gadget

ESPHome wieder im Einsatz!



Von **Clemens Valens** (Elektor)

Kennen Sie den? Den Karton voller Platinen, die Sie für die Zukunft aufbewahren, wenn Sie mal mehr Zeit haben? Da ich dringend ein ESP8266-Modul für meine Hausautomatisierungs-Experimente benötigte, habe ich einen Blick hinein geworfen und eine ESP8266-basierte Wearable-Platine gefunden, die ich vor einigen Jahren gebaut habe. Nichts Spektakuläres, im Grunde nur ein ESP-12E-Modul mit einer USB-zu-Seriell-Brücke und einem Treiber für eine Kette von adressierbaren WS2812-LEDs. Ich ließ mein ursprüngliches Projekt liegen und habe dieses Board zum Leben erweckt - natürlich mit Hilfe von ESPHome!

Vor einigen Jahren bat mich ein Kollege, ein tragbares Mikrocontroller-Board mit einem ESP8266-basierten WLAN-Modul zu entwickeln. Die Firmware des Projekts würde er übernehmen, da er große Pläne für ein solches Gerät hatte. Als der Prototyp fertig war, probierte er das Board aus und verließ dann die Firma. Hatte ihn mein Design so sehr enttäuscht oder entmutigt? Das habe ich nie herausgefunden. Das Projekt wurde aufgegeben und wäre in Vergessenheit geraten, wenn ich nicht vor ein paar Monaten dringend ein ESP8266-Modul

für meine Experimente zur Heimautomatisierung benötigt hätte. Beim Durchstöbern von Kisten mit Dingen, die „mal eines Tages nützlich sein“ könnten, fiel mir der Wearable-ESP8266-Prototyp in die Hände. Und da ich zu dieser Zeit besessen von ESPHome [1] war, wusste ich sofort, dass ich nun die Werkzeuge hatte, um endlich die Software für dieses Board zu entwickeln.

Eine NodeMCU-ähnliche Schaltung

Der Schaltplan des Boards (**Bild 1**) ist im Grunde ein NodeMCU-Entwicklungsboard, bei dem die USB-zu-Seriell-Brücke CP2101 von Silicon Laboratories (Silabs) durch den wesentlich günstigeren FT231XS von FTDI ersetzt wurde. Außerdem wurde ein Port für eine adressierbare WS2812-basierte LED-Kette (a.k.a. NeoPixels) hinzugefügt.

Da das Wearable-Gadget quasi ein NodeMCU-Board ist, gilt alles im Folgenden gesagte auch für ein normales NodeMCU-Modul. Auch die vorgestellte Software funktioniert genauso gut.

Die verfügbaren GPIO-Ports und die Stromversorgung sind auf spezielle Pads mit großen Löchern herausgeführt, die kreisförmig am Rand der runden Platine angeordnet sind. Diese Pads sind für die Verwendung von leitfähigem Garn (conductive thread) vorgesehen, aber man kann natürlich auch dünnen Schaltdraht anlöten oder gar Krokodilklemmen verwenden.

Die Stromversorgung der Schaltung erfolgt entweder über den Micro-USB-Anschluss oder durch ein externes 5-V-Netzteil an einem der 5-V-Pins. Letzteres ist bei langen LED-Strings sinnvoll, die mehr Strom benötigen, als ein normaler USB-Anschluss liefern kann. Eine USB-Powerbank mit hoher Kapazität ist ebenfalls eine Option. Beachten Sie, dass a) die LED-Schnittstelle für 5-V-Strings ausgelegt ist und b) der 5-V-Eingang keinen Verpolungsschutz bereitstellt.

Software-Entwicklung mit ESPHome

Für meine Experimente habe ich eine ringförmige Kette von zwölf WS2812-LEDs an den Port K2 angeschlossen, siehe **Bild 2**. Natürlich können Sie die Software für dieses Board von Grund auf neu schreiben, wie es mein ehemaliger Kollege vorhatte und wahrscheinlich auch tun musste, da damals nicht so viel ESP8266-Code im Netz verfügbar war wie heutzutage. Die Übernahme eines Open-Source-Projekts wie ESPHome erspart Ihnen aber eine Menge Arbeit.

Ich habe es schon einmal gesagt, und ich will es gerne noch einmal wiederholen: Mit ESPHome können Sie eine vernetzte Anwendung

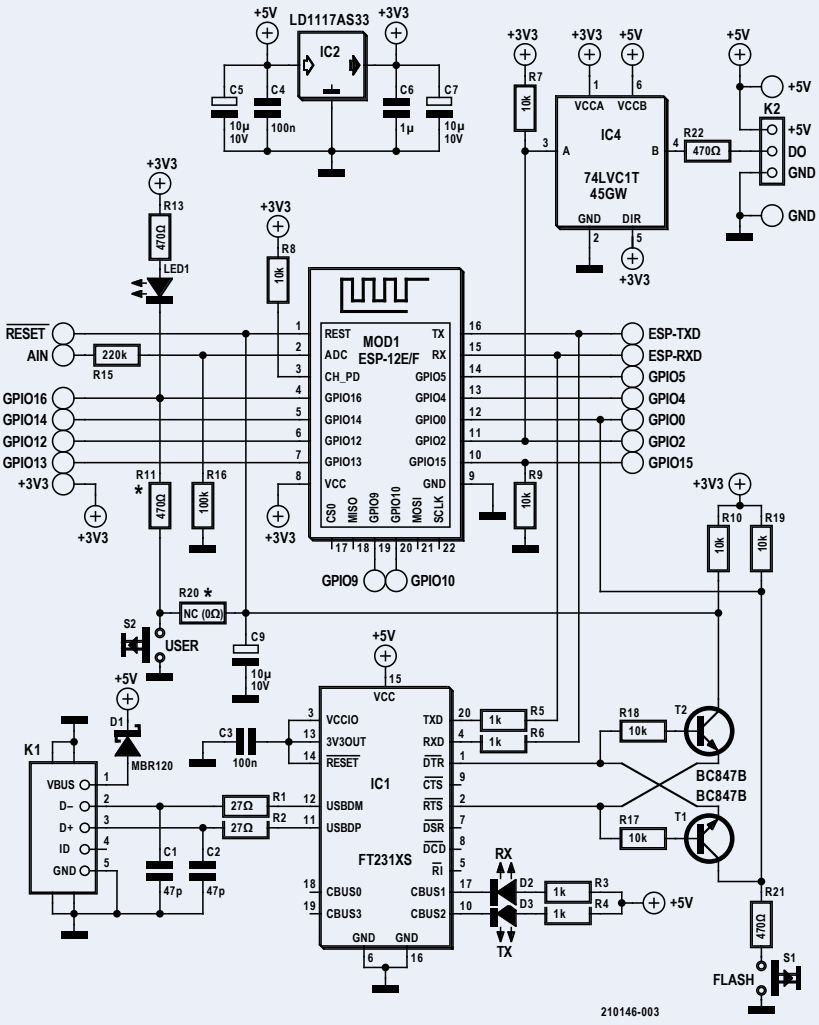


Bild 1. Schaltplan des Wearable-ESP8266-Boards.

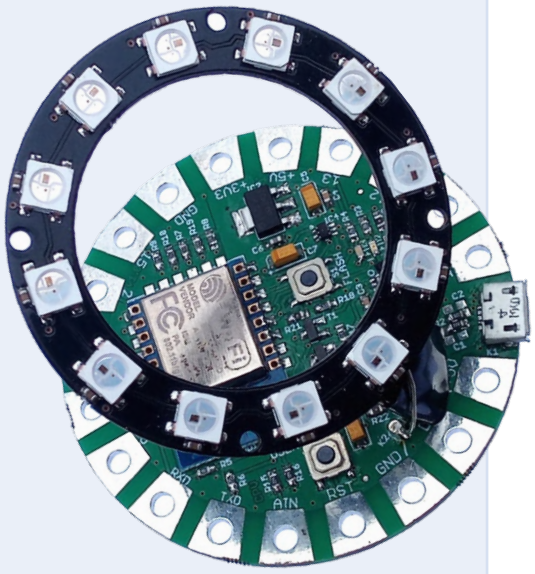


Bild 2. Der Prototyp des Wearable-WLAN-Gadgets mit einem adressierbaren 12-Pixel-RGB-LED-Ring.

Listing 1. Die YAML-Konfigurationsdatei [2].

```
# Elektor 160112 Wearable ESP8266
 # Configuration file for ESPHome

esphome:
   name: wearable
   platform: ESP8266
   board: nodemcu

wifi:
   ssid: "my_ssid"
   password: "my_passphrase"
  ap:
     ssid: "Wearable Fallback Hotspot"
     password: "12345678"

captive_portal:

# Enable logging
 logger:

# Enable Home Assistant API
 api:

# Enable Over-the-Air updates.
 ota:

output:
   - platform: gpio
     id: "blue_led"
     pin:
       number: GPIO16
       inverted: True

light:
   - platform: binary
     name: "Blue LED"
     output: "blue_led"
   - platform: neopixelbus
     name: "Light Ring"
     num_leds: 12
     type: GRB
     pin: GPIO2
     method: ESP8266_UART1
     effects:
       - addressable_color_wipe:
           name: "Color Wipe"
       - addressable_fireworks:
           name: "Fireworks"
       - flicker:
           name: "Flicker All"
       - addressable_flicker: # Doesn't work?
           name: "Flicker Individually"
       - addressable_rainbow:
           name: "Rainbow"
       - random:
           name: "Random All"
       - addressable_scan:
           name: "Scan"
       - strobe:
           name: "Strobe All"
       - addressable_twinkle:
           name: "Twinkle"
       - addressable_random_twinkle:
           name: "Twinkle Random"

# GPIO11 is somehow related to flash and should
# not be used.
 switch:
   - platform: gpio
     name: "GPIO4"
     pin: GPIO4
   - platform: gpio
     name: "GPIO5"
     pin: GPIO5
   - platform: gpio
     name: "GPIO12"
     pin: GPIO12
   - platform: gpio
     name: "GPIO14"
     pin: GPIO14
   - platform: gpio
     name: "GPIO15"
     pin: GPIO15

# Pushbutton on GPIO0.
 binary_sensor:
   - platform: gpio
     name: "Flash"
     pin:
       number: GPIO0
       inverted: True

sensor:
   - platform: adc
     name: "Analog Input"
     pin: A0
     update_interval: 60s
     filters:
       - multiply: 3.2
# voltage divider is 100k/(220k+100k)
```

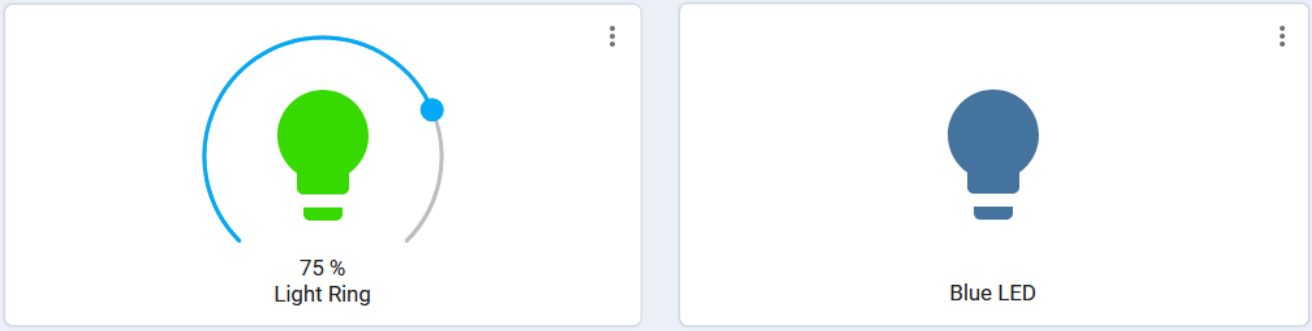


Bild 3. Diese Karten im Home Assistant ermöglichen die Steuerung des LED-Rings und der blauen LED des Wearable-WLAN-Gadgets.

für den ESP8266 oder ESP32 erstellen, die Over-the-Air-Programmierung (OTA), Fallback-Hotspot, Webserver-Benutzeroberfläche und Schnittstellen für über 200 Geräte bietet, und das in kürzester Zeit. Wirklich. Siehe [1] für weitere Details.

ESPHome verwendet einen modularen Ansatz, bei dem vorgefertigte Codeblöcke zu einer Anwendung kombiniert werden. Die von der Anwendung benötigten Blöcke werden in einer Konfigurationsdatei, der sogenannten YAML-Datei, aufgelistet (YAML kommt von „Yet another markup language" und ist keine Programmiersprache, sondern ein Satz von Formatierungsregeln für Textdateien zur Angabe von Parametern und Werten [1]). Jeder Baustein wird individuell konfiguriert, um zum Beispiel den oder die zu verwendenden GPIO-Pin(s), deren Typ oder ihr Kommunikationsprotokoll festzulegen.

Die Konfigurationsdatei wird von ESPHome gelesen und in C++-Code umgewandelt, der dann wiederum in eine ausführbare Datei kompiliert und in den Flash-Speicher des Moduls programmiert werden kann. Wenn Sie einmal verstanden haben, wie man eine Konfigurationsdatei zusammenstellt, sind Sie im Geschäft. Bitte lesen Sie [1] für weitere Details.

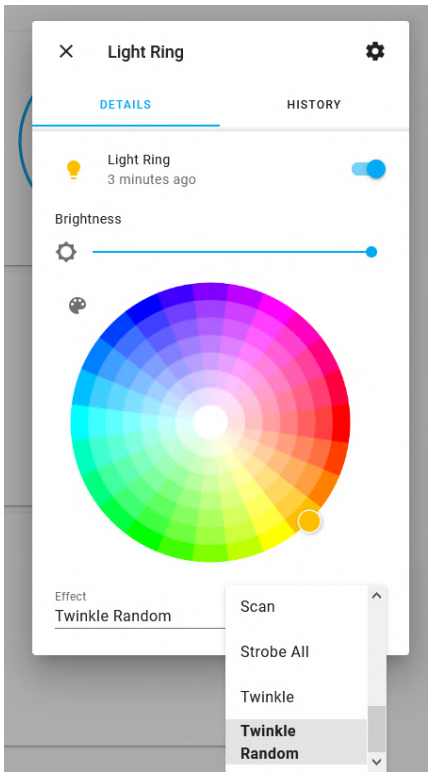


Bild 4. Die in der YAML-Konfigurationsdatei von ESPHome deklarierten Lichteffekte werden im Home Assistant als Dropdown-Liste angezeigt, aus der Sie den gewünschten Effekt auswählen können.

Die Konfigurationsdatei genau betrachtet

Die YAML-Konfigurationsdatei (**Listing 1**) beginnt mit dem obligatorischen Abschnitt *esphome:*, in dem der Projektname, die verwendete MCU (ESP8266) und der Boardtyp (NodeMCU) angegeben werden. Als nächstes folgt der WLAN-Abschnitt *wifi:*, um das Netzwerk, mit dem eine Verbindung hergestellt werden soll, zu definieren und um Rettungsoptionen im Falle von Netzwerkproblemen anzugeben. Die Reihenfolge der Sektionen spielt keine Rolle.

Durch die Angabe der Option *logger:* wird die Statusausgabe auf der seriellen Schnittstelle aktiviert. Die Option *api:* ermöglicht die einfache Integration mit der freien und quelloffenen Heimautomatisierungs-Controller-Software *Home Assistant* (siehe [1]). Und *ota:* ist für die Over-the-Air-Programmierung (durch zum Beispiel den Home Assistant) zuständig. OTA ist sehr praktisch, da dadurch keine physische Verbindung zum Gerät erforderlich ist.

Dann wird es ernst! Es beginnt mit der Spezifikation, dass GPIO16 ein Ausgang sein muss. Dies ist erforderlich, will man die daran angeschlossene LED als *light*: verwenden. Das ist interessant, denn *lights* haben ganz andere Möglichkeiten als beispielsweise Schalter (**Bild 3**).

Lights

Als *Lights* habe ich die blaue LED an GPIO16 und den LED-String definiert. Letzterer wird von der *platform: neopixelbus* verwaltet, die ein paar Optionen hat, die angegeben werden müssen, etwa die Länge des Strings und der Port, an den er angeschlossen wird. In diesem Fall ist der Port GPIO2, was die Verwendung der *ESP8266_UART1*-Methode ermöglicht. Eigentlich müssen Sie bei dieser Methode den GPIO2 gar nicht angeben, da es impliziert ist.

Lights können Effekte haben, und ESPHome hat ein paar eingebaut, die Sie verwenden können (wenn Ihr *light:* unterstützt wird, natürlich). Selbstverständlich können Sie auch Ihre eigenen Lichteffekte programmieren. Ich habe die meisten der eingebauten Effekte für den LED-Ring genutzt. Auch die Effekte können Parameter haben, aber wenn Sie keine explizit angeben, werden Standardwerte verwendet. Im Home Assistant können Sie wählen, welcher Effekt aktiv ist (**Bild 4**). Ich mag besonders den Effekt *random twinkle*.

Mehr Ein- und Ausgänge

Die unbenutzten GPIO-Ports sind als Schalter deklariert, so dass Sie sie im Home Assistant ein- und ausschalten können. In der Hausautomatisierung ist ein Schalter ein Gerät, das vom System gesteuert wird (zum Beispiel ein Relais). Ein Schalter, der durch den Benutzer (oder

Bewohner) betätigt wird, ist ein binärer Sensor. An GPIO0 ist ein „richtiger“ Taster angeschlossen, daher ist er als binärer Sensor spezifiziert. Schließlich besitzt das Board einen analogen Eingang an Pin A0 mit einem Spannungsteiler davor (R15 und R16), der für eine Abschwächung von 3,2 sorgt. Wenn die maximale Eingangsspannung von 3,3 V an AIN anliegt, bleibt davon am Analogeingang des ESP12 etwa 1 V übrig.

Abstandsalarm

ESPHome verfügt über Automatisierungsfunktionen. Sie könnten der Platine zum Beispiel einen Näherungssensor hinzufügen, um die Farbe des LED-Rings zu ändern, je nachdem, was der Sensor so sieht. Auf diese Weise könnte das Board zum Beispiel als „Social-Distancing-Alarm“ fungieren (wenn so etwas noch nötig sein sollte, wenn Sie dieses Heft im Sommer in den Händen halten). Es kann aber auch einfach ein dekorativer elektronischer Anstecker oder eine Brosche sein; lassen Sie Ihrer Fantasie freien Lauf!

Eine interessante Möglichkeit bietet hier die Nutzung des Home Assistant. Mit ein paar dieser tragbaren ESP8266-Boards, die in den Home Assistant integriert sind, lassen sich ausgefallene Lichteffekte erzeugen. Auch wenn der Home Assistant für eine Heimautomatisierung optimiert ist, kann er auch ganz andere Dinge tun, beispielsweise ein Spiel auf der Geburtstagsparty Ihres Kleinkindes steuern. Hängen Sie jedem Kind ein Board um und nutzen Sie den Home Assistant, um Teams zu bilden und zu steuern oder zu entscheiden, welcher Kandidat eine Frage beantworten darf oder wer beim Fangen spielen dran ist. Ich bin sicher, dass Sie mit ein bisschen Kreativität viele lustige Anwendungen finden können.

Passende Produkte

- **ESP-12F, ESP8266-basiertes WLAN-Modul**
  www.elektor.de/esp-12f-esp8266-based-wi-fi-module-160100-92
- **NodeMCU ESP8266 Mikrocontrollerboard**
  www.elektor.de/nodemcu-microcontroller-board-with-esp8266-and-lua
- **H. Henrik Skovgaard, *IoT Home Hacks with ESP8266*, Elektor, 2020**
  www.elektor.de/iot-home-hacks-with-esp8266

Alle Dateien zu diesem Projekt können unter [2] heruntergeladen werden.

210146-02

Haben Sie Fragen oder Kommentare?
Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Schicken Sie eine E-Mail an den Autor unter clemens.valens@elektor.com oder kontaktieren Sie Elektor unter editor@elektor.com.

Ein Beitrag von
Idee, Design, Text und Fotografien: **Clemens Valens**
Redaktion: **Jens Nickel, C. J. Abate**
Schaltplan: **Patrick Wielders**
Übersetzung: **Rolf Gerstendorf**
Layout: **Harmen Heida**

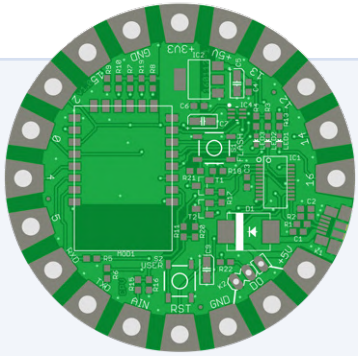
STÜCKLISTE

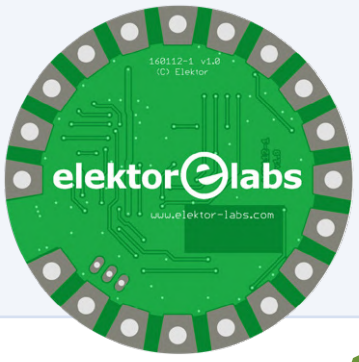
Widerstände:
Alle 5%, 50 V, 0,1 W, 0603
R20 = 0 Ω
R1,R2 = 27 Ω
R11,R13,R21,R22 = 470 Ω
R3...R6 = 1 k
R7...R10,R17...R19 = 10 k
R16 = 100 k
R15 = 220 k

Kondensatoren:
C1,C2 = 47 p, 0603
C3,C4 = 100 n, 0603
C6 = 1 µ, 0603
C5,C7,C9 = 10 µ, 16 V, Case-A

Halbleiter:
D1 = MBRS540
IC4 = 74LVC1T45GW
IC1 = FT231XS
IC2 = LD1117AS33
LED1 = LED, blau, 0603
LED2 = LED, gelb, 0603
LED3 = LED, rot, 0603
T1,T2 = BC847C

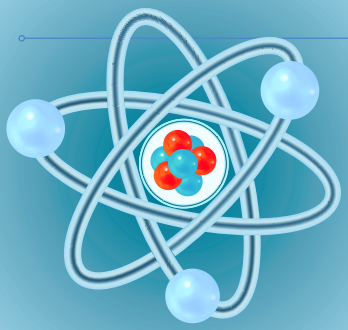
Außerdem:
K1 = Micro-USB-Buchse Typ B, gewinkelt, Platinenmontage
K2 = 1x3-polige Stiftleiste, 0,1“-Raster
S1,S2 = SMD-Taster, 5,1 mm x 5,1 mm
MOD1 = ESP-12F
Platine 160112-1





WEBLINKS

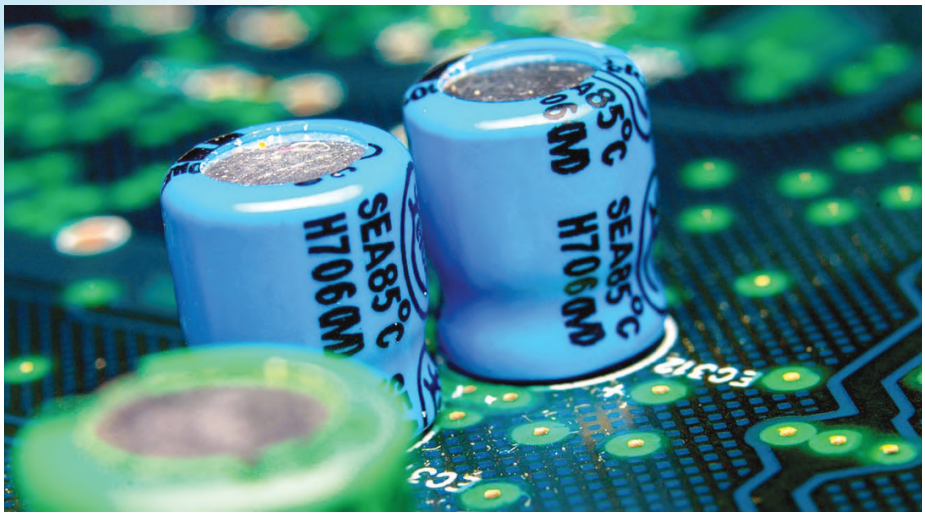
[1] **C. Valens, „Hausautomation leicht gemacht“, Elektor Sep/Okt 2020**: www.elektormagazine.de/magazine/elektor-154/58936
[2] **Wearable Wi-Fi Gadget bei Elektor Labs**: www.elektormagazine.com/labs/4382



Aller Anfang …

muss nicht schwer sein!

Von **Eric Bogers** (Elektor)



In dieser Elektor-Ausgabe wird das Thema „Kondensatoren“ abgeschlossen. Beim letzten Mal [1] haben wir gesehen, dass Kondensatoren Gleichstrom blockieren, aber Wechselstrom durchlassen. Damit ist es aber noch lange nicht getan - Spannung und Strom sind auch noch phasenverschoben, was die Berechnungen etwas komplizierter machen kann. Aber das ist noch lange kein Grund, sich davon abschrecken zu lassen!

Phasenwinkel

Phase? Was ist denn das? Um das zu verdeutlichen, kehren wir kurz zu unserem bescheidenen Widerstand zurück. Wenn wir eine Spannung über einen Widerstand anlegen, fließt sofort ein Strom. Hier sind Spannung und Strom gleichphasig: In dem Moment, in dem die Spannung ansteigt, steigt auch der Strom an. Und anders herum.

Bei einem Kondensator ist das jedoch anders, und wir können das wie folgt erklären: In dem Moment, in dem wir eine Spannung an einen entladenen Kondensator anlegen, verhält er sich nahezu wie ein Kurzschluss. Es fließt der maximale Ladestrom, aber die Spannung am Kondensator ist null (wie es sich für einen Kurzschluss gehört). Während sich der Kondensator auflädt, steigt die Spannung über dem Kondensator und der Strom nimmt ab. Wenn der Kondensator vollständig aufgeladen ist, hat die Spannung über dem Kondensator ihren Maximalwert erreicht, während der Strom auf null gesunken ist. Man kann also sagen: Erst kommt der Strom, dann folgt die Spannung. Im Fachjargon: Der Strom eilt der Spannung voraus. Wenn eine reine Sinusspannung an einen Kondensator gelegt wird, gibt es einen *Phasenverschiebungswinkel* (auch einfach *Phasenverschiebung* oder *Phasendifferenz* genannt) von genau 90° zwischen der Spannung und dem Strom.

Wenn wir eine Wechselspannung an einen Kondensator anlegen, fließt ein Wechselstrom durch diesen Kondensator - das haben wir schon kennengelernt. Das bedeutet, dass wir auch einen (Wechselstrom-) Widerstand für diesen Kondensator berechnen können. Um diesen nicht mit dem ohmschen Widerstand zu verwechseln, wird er als Impedanz bezeichnet und das Symbol X verwendet. Die Einheit der Impedanz ist aber wieder das Ohm. Es gilt der folgende Ausdruck (wir wollen Sie nicht mit der Herleitung belästigen):

$$X_C = \frac{1}{2 \cdot \pi \cdot f \cdot C}$$

Je größer der Kondensator ist, desto mehr Ladung kann er speichern, desto größer ist die Strommenge, die in ihn fließt und desto kleiner ist seine Impedanz. Außerdem gilt: Je höher die Frequenz, desto häufiger (pro Zeiteinheit) wird der Kondensator geladen und entladen, desto mehr Strom fließt und wiederum: desto kleiner ist seine Impedanz. Wichtig ist: Die Impedanz eines Kondensators ist kein fester Wert, sondern hängt von der Frequenz ab.

Sie können natürlich mit den Schultern zucken und denken: „Okay, das ist ja alles ganz nett, diese Phasendifferenz, aber darüber werde ich mir keinen Kopf machen.“ Wenn es doch nur so einfach wäre… In der praktischen Elektronik werden Kondensatoren und Widerstände selten „alleine“ verwendet; meistens haben wir es mit ein Reihen- oder Parallelschaltungen verschiedener Bauteile zu tun. Und diese Phasendifferenz ist schuld daran, dass man zum Beispiel in einer Reihenschaltung aus einem Kondensator und einem Widerstand die Impedanz und den Widerstand nicht

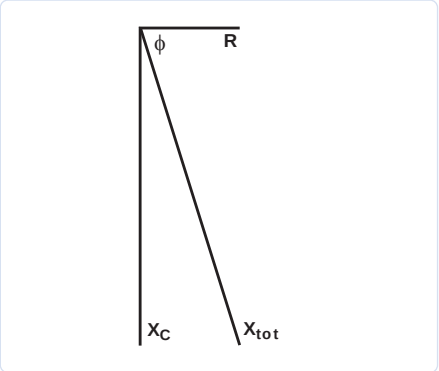


Bild 1. Reihenschaltung eines Widerstandes und eines Kondensators.

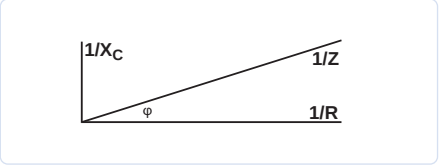


Bild 2. Parallelschaltung eines Widerstandes und eines Kondensators.

einfach so addieren kann. Impedanz und Widerstand müssen *vektoriell* addiert werden. Wir wollen das am Beispiel einer Reihenschaltung aus einem Kondensator mit einem Wert von 0,1 µF und einem Widerstand mit einem Wert von 10 kΩ durchexerzieren. Wir möchten die Gesamtimpedanz bei einer Frequenz von 50 Hz wissen.
Zunächst berechnen wir die Impedanz des Kondensators:

$$X_C = \frac{1}{2 \cdot \pi \cdot f \cdot C} = \frac{1}{2 \cdot \pi \cdot 50\,Hz \cdot 0{,}1\,\mu F} = 31{,}830\,k\Omega$$

Für die Gesamtimpedanz der Reihenschaltung gilt folgendes:

$$X_{tot} = \sqrt{X^2{}_C + R^2} = \sqrt{(31{,}830\,k\Omega)^2 + (10\,k\Omega)^2} = 33{,}364\,k\Omega$$

Die so berechnete Gesamtimpedanz wird oft mit dem Buchstaben *Z* (anstelle von X_{tot}) angegeben. Auch die Impedanz *Z* besitzt einen bestimmten Phasenwinkel (der logischerweise irgendwo zwischen den -90° des Kondensators und den 0° des Widerstandes liegt). Es gilt:

$$\tan\varphi = \frac{X_C}{R}$$

Der Phasenwinkel ist dann (sorry, dafür brauchen Sie wahrscheinlich einen „wissenschaftlichen" Taschenrechner):

$$\varphi = \arctan\frac{X_C}{R} = \arctan\frac{-31{,}830\,k\Omega}{10\,k\Omega} = -72{,}56^\circ$$

In einer Reihenschaltung fließt in allen Bauteilen derselbe Strom, in einem Kondensator jedoch hinkt die Spannung dem Strom hinterher, weshalb der Phasenwinkel negativ ist. Wenn gewünscht, kann dieses Problem auch grafisch gelöst werden (**Bild 1**). Da die Drehrichtung (im mathematischen Sinne) gegen den Uhrzeigersinn gerichtet ist, muss der negative Phasenwinkel nach unten zeigen.
Natürlich können der Widerstand und der Kondensator aus der Reihenschaltung auch parallel geschaltet werden. Die Gesamtimpedanz *Z* ist dann:

$$Z = \frac{1}{\sqrt{\frac{1}{X^2{}_C} + \frac{1}{R^2}}} = \frac{1}{\sqrt{\frac{1}{(31{,}830\,k\Omega)^2} + \frac{1}{(10\,k\Omega)^2}}} = 9{,}54\,k\Omega$$

Für den Phasenwinkel haben wir:

$$\varphi = \arctan\frac{R}{X_C} = \arctan\frac{10\,k\Omega}{31{,}830\,k\Omega} = 17{,}44^\circ$$

Auch hier ist eine grafische Lösung möglich, siehe **Bild 2**. In einer Parallelschaltung liegt an allen Bauteilen die gleiche Spannung an. Der Strom durch den Kondensator eilt jedoch der Spannung voraus, so dass in diesem Fall der Phasenwinkel positiv ist und nach oben zeigt.
Eine abschließende Bemerkung: Der Unterschied zwischen den Phasenwinkeln in einer Reihenschaltung und einer Parallelschaltung (mit denselben Komponenten) beträgt genau 90°.

Hoch- und Tiefpässe

In den Elektor-Ausgaben von September 2020 bis Februar 2021 haben wir uns ausgiebig mit (dem Entwurf von) Filterschaltungen beschäftigt. Für den angehenden Elektroniker waren diese drei Artikel aber wohl etwas zu viel des Guten. Dennoch sind Filterschaltungen in der Elektronik von so eminenter Bedeutung, dass dieses Thema hier kurz gestreift werden muss, wenn auch ohne zu viel Theorie.
Aus der Formel für die Impedanz eines Kondensators folgt, dass diese Impedanz bei einer Frequenz von 0 Hz (also einer Gleichspannung) ins Unendliche geht und mit steigender Frequenz kleiner wird. Deshalb wird ein Kondensator verwendet, um den Gleichanteil aus einer Mischung von Gleich- und Wechselspannung zu entfernen (zu blocken).
Eine solche Schaltung benötigen wir zum Beispiel in der Eingangsstufe eines Mikrofonverstärkers: Die Wechselspannung (das elektrische Äquivalent des akustischen Signals), die vom Mikrofon erzeugt wird, überlagert sich mit der Gleichspannung, mit der das Mikrofon versorgt wird (wir sprechen von einer Phantomspeisung, die separate Kabel für Signal und Stromversorgung überflüssig macht). Wir wollen natürlich nicht die Versorgungsspannung verstärken, sondern nur die Signalspannung.
Um diese Gleichspannung zu entfernen, verwenden wir ein Hochpassfilter (**Bild 3**). Es heißt so, weil es die hohen Frequenzen durchlässt und die niedrigen Frequenzen sperrt. In **Bild 4** sehen wir die Frequenz- und Phasenkennlinien eines Hochpassfilters (das Messsystem, mit dem das Diagramm erstellt wurde, ist an den Rändern des Frequenzbereichs leider nicht sehr genau). Dennoch können wir deutlich drei Regionen unterscheiden:

- Der Durchlassbereich beginnt (in diesem Diagramm) bei etwa 1 kHz. In diesem Bereich läuft das Signal praktisch ungehindert durch und es gibt auch kaum eine Phasenverschiebung.
- Der Stopp- oder Sperrbereich liegt unterhalb von etwa 100 Hz. Hier nimmt die Signalstärke mit abnehmender Frequenz ab - genauer gesagt mit 6 dB pro Oktave beziehungsweise mit 20 dB pro Dekade. Der Phasewinkel nähert sich asymptotisch der 90°-Grad-Marke.
- Zwischen diesen beiden Bereichen befindet sich die Eck- oder Grenzfrequenz. Diese ist definiert als die Frequenz, bei der der Signalpegel um 3 dB gegenüber dem Durchlassbereich reduziert ist. Bei dieser Eckfrequenz ist die kapazitive Impedanz gleich dem Wert des Widerstandes, so dass wir schreiben können:

$$f_{cut-off} = \frac{1}{2 \cdot \pi \cdot R \cdot C}$$

Das Verhalten eines solchen einfachen Filters kann nicht weiter als bis zur Eckfrequenz spezifiziert werden; dies würde mehrstufige Filter (Filter höherer Ordnung) erfordern. Diese Materie wurde in der oben genannten Artikelserie in Elektor [2] bereits ausführlich behandelt.

Wo es Hochpassfilter gibt, gibt es auch Tiefpassfilter. Diese sehen gleich aus, nur der Widerstand und der Kondensator haben ihre Plätze getauscht (siehe **Bild 5**). Je höher die Frequenz, desto mehr wird das Signal durch den Kondensator gegen Masse kurzgeschlossen. Diese Art von Filter wird (unter anderem) dazu verwendet, hochfrequentes Rauschen aus einem Signal zu entfernen.

In **Bild 6** sehen Sie den Frequenz- und Phasenverlauf eines Tiefpassfilters (in diesem speziellen Fall haben wir andere Bauteilwerte verwendet als im Beispiel von Bild 4). Auch hier unterscheiden wir drei Bereiche: den Durchlassbereich unterhalb der Eckfrequenz, den Sperrbereich oberhalb der Eckfrequenz und die Eckfrequenz selbst, wo die kapazitive Impedanz gleich dem Wert des Widerstandes ist.

Damit schließen wir unsere Beschreibung der Kondensatoren ab. Nächstes Mal geht es weiter mit Induktivitäten - Bauteile, die von vielen Elektronikern (zu Unrecht) verschmäht werden…

210183-03

Die Artikelserie „Aller Anfang …“ basiert auf dem Elektor-Buch „Basiskurs Elektronik“ von Michael Ebner.

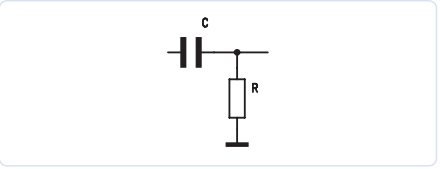


Bild 3. Hochpassfilter

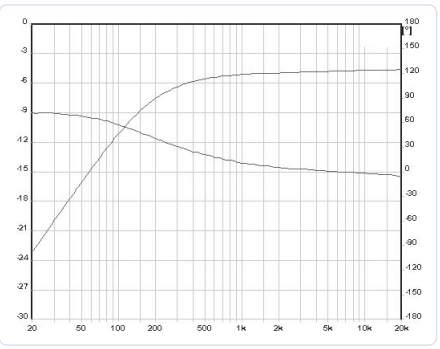


Bild 4. Frequenz- und Phasenverlauf eines Hochpasses.

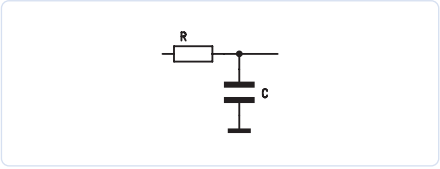


Bild 5. Tiefpassfilter

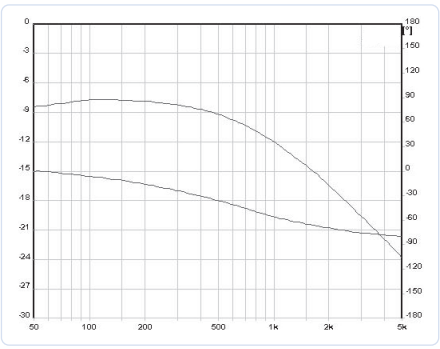


Bild 6. Frequenz- und Phasenkennlinien eines Tiefpassfilters.

Ein Beitrag von
Idee und Illustrationen: **Michael Ebner**
Text und Redaktion: **Eric Bogers**
Übersetzung: **Rolf Gerstendorf**
Layout: **Giel Dols**

Haben Sie Fragen oder Kommentare?
Haben Sie technische Fragen oder Anregungen zu diesem Artikel? Senden Sie eine E-Mail an den Autor oder an die Elektor-Redaktion über redaktion@elektor.de.

WEBLINKS

[1] **E. Bogers, „Aller Anfang …“, Elektor, Mai/Juni 2020**: www.elektormagazine.de/magazine/elektor-147/58576
[2] **A. Rosenkränzer, „Design analoger Filter“, dreiteilig ab Elektor September/Oktober 2020**: www.elektormagazine.de/magazine/elektor-154/58927

Einfache Sketches mit dem Qwiic-Ökosystem

Von **Marcus Stevenson** (USA)

Das Qwiic-System ist eine SparkFun-Innovation, die den I²C-Bus über gepolte vierpolige JST-Steckverbindungen führt, um modulare Schaltungen mit jedem der über 100 kompatiblen Breakout-Boards und Mikrocontrollern zu erstellen. Das bedeutet, dass Qwiic das mühsame Löten von Anschlusskabeln und das Entschlüsseln von unübersichtlichen Schaltplänen überflüssig macht, so dass es sogar für einen Anfänger extrem einfach ist, einen Arduino mit einem beliebigen Sensor zu verbinden und sofort mit dem Experimentieren zu beginnen. Den Beweis liefern die folgenden einfachen Sketches, die das Qwiic-Pro-Micro-Controllerboard und eine Handvoll anderer Breakout-Boards von SparkFun verwenden.

Dieser Artikel geht davon aus, dass Sie die Arduino-IDE bereits eingerichtet und die erforderlichen Bibliotheken installiert haben. Falls Sie Hilfe benötigen: Auf SparkFun.com finden Sie Anleitungen zur Einrichtung aller hier verwendeten Produkte. Die Verdrahtung ist bei jedem der folgenden kleinen Projekte im Wesentlichen gleich.

Drehgeber mit Farbwechsel

ⓘ Schaltung: **Bild 1,** Programmcode: **Weblink [1]**.

Für dieses Beispiel benötigen Sie das Controllerboard *Qwiic Pro Micro*, den *Qwiic Twist RGB Rotary Encoder* und das *Qwiic Mirco OLED Breakout*. Verwenden Sie Qwiic-Kabel, um den Pro Micro mit dem Drehgeber und dem OLED-Display zu verbinden. Die Reihenfolge spielt keine Rolle, solange sie nur alle miteinander verbunden sind. Dann schließen Sie den *Pro Micro* an Ihren Computer an und laden aus der Arduino-IDE den Sketch *QwiicTwistMicroOLEDDisplay.ino* hoch. Achten Sie darauf, dass unter *Prozessor* der *ATMEGA 32U4 (5V, 16MHz)* ausgewählt ist, sonst erhalten Sie einen Upload-Fehler! Wenn alles erfolgreich war, sollten Sie kurz das SparkFun-Flammen-Logo auf dem OLED-Display sehen, dann leuchtet die RGB-LED des Drehgebers entsprechend der eingestellten Farbe. Das Display zeigt den Namen der Farbe an. Wenn Sie am Enkoder drehen, ändert

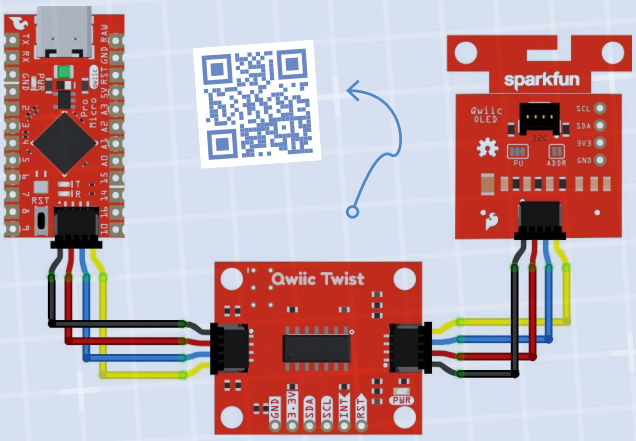


Bild 1. Anschluss der drei Platinen für den Drehgeber mit Farbwechsel.

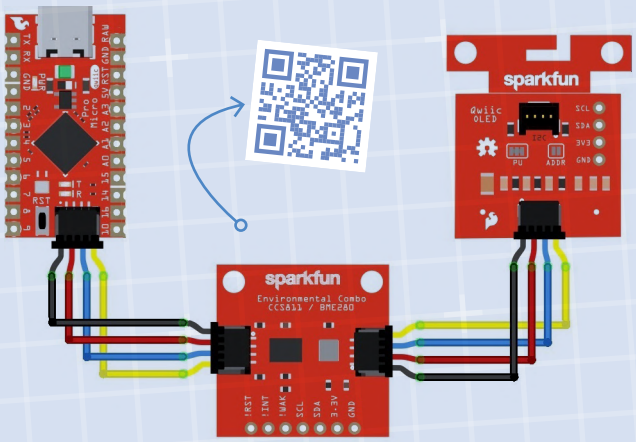


Bild 2. Anschluss der drei Platinen für das Qwiic-Umgebungsthermometer.

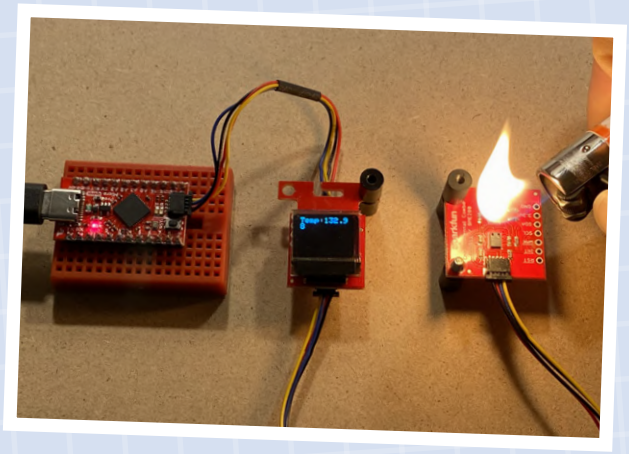


Bild 3. Feuer und Flamme für die Elektronik. Versuchen Sie das besser nicht!

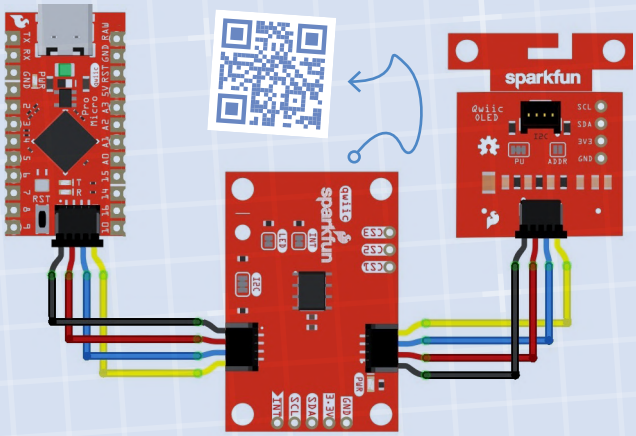


Bild 4. Verbindung der Platinen bei der „Kapazitiven Toucheingabe leicht gemacht".

sich die Farbe der LED, was auch in der Inschrift des OLED-Displays zu sehen ist. Wenn Sie den Drehencoder drücken, sollte auf dem Bildschirm außerdem *Pressed!* erscheinen.
Dieser Sketch verwendet die Funktion `getCount()`, um die aktuelle Stellung des Drehgebers zu ermitteln, und die Funktion `setColor()`, um die Farbe der RGB-LED des Drehgebers entsprechend der Stellung zu ändern, in der Folge rot, grün, violett, gelb, rosa, blau und orange. Außerdem werden die Methoden `clear()`, `setCursor()`, `print()` und `display()` der OLED-Bibliothek verwendet, um die Farbnamen auf das Display auszugeben.

Qwiic-Umweltthermometer

ⓘ Schaltung: **Bild 2**; Programmcode: **Weblink [2]**.

Dieses Beispiel verwendet ebenfalls das *Qwiic Micro OLED Display* und den *Pro Micro*, aber diesmal tauschen wir den Drehgeber gegen das *Qwiic Environmental Combo Breakout*. Dieser Sensor kann eine Reihe von Umweltdaten sammeln, einschließlich Luftdruck, Luftfeuchtigkeit, Temperatur, TVOCs (flüchtige organische Verbindungen) und äquivalente CO_2-Werte (eCO2). In diesem Beispiel werden wir uns aber nicht zu sehr ins Zeug legen, sondern nur die aktuelle Temperatur auf dem kleinen OLED-Display ausgeben, und zwar in Fahrenheit (exotisches fernwestliches Temperaturmaß). Angenommen, Sie haben die vorherige Schaltung gebaut, dann nabeln Sie den Drehgeber ab und schließen am Qwiic-Kabel das *Environmental Combo Breakout* an.

Jetzt können Sie den Sketch *QwiicEnvironmentalComboMicroOLED-Display.ino* auf den *Pro Micro* hochladen. Wieder sehen Sie kurz das Flammenlogo, dann erscheint *Temp:* gefolgt von der aktuellen Temperatur in Grad Fahrenheit. Wenn Sie auf den Sensor pusten oder eine Flamme darüber halten (**Bild 3**), sollten Sie die Temperaturschwankungen sehen. Dieser Sketch verwendet die Funktion `readTempF()`, um den Temperaturwert des BME280-ICs anzuzeigen. Natürlich könnten Sie die `readTempC()`-Methode verwenden, um die Temperatur auch in richtigen Grad Celsius darzustellen..

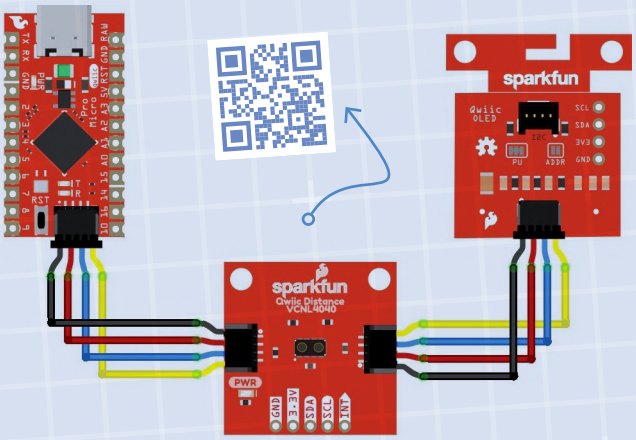


Bild 5. Platinenanschluss für den Qwiic-Anwensenheitsdetektor.

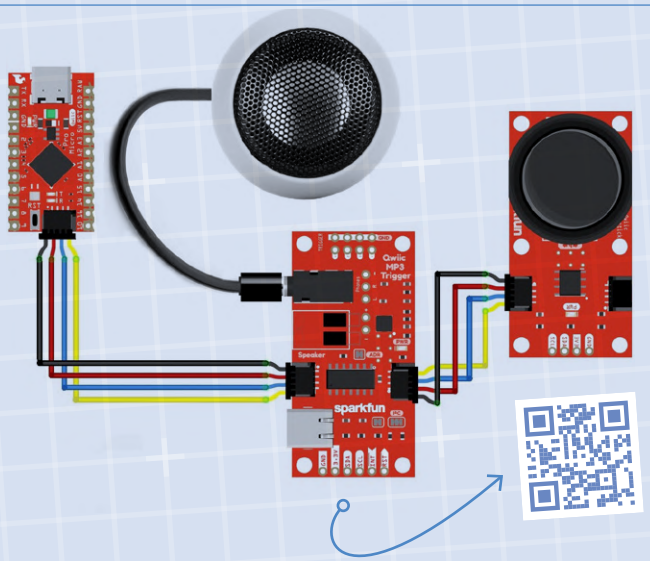


Bild 6. Für die Schaltung „Taktiler Eingang, akustischer Ausgang" ist ein kleiner Lautsprecher erforderlich.

Kapazitive Toucheingabe leicht gemacht

Schaltung: **Bild 4**; Programmcode: **Weblink [3]**.

Knöpfe werden oft überschätzt. Ihre mechanische Natur macht sie irgendwann malade. Mit kapazitiven Toucheingängen können Sie mit Ihren Projekten interagieren, ohne bewegliche Teile. In diesem Sketch verwenden wir weiterhin das *Micro OLED Display* und benutzen als Eingabegerät den *Qwiic capacitive touch slider*. Trennen Sie einfach das *Environmental Combo Breakout* ab und ersetzen Sie es durch den *Capacitive Touch Slider*. Dieses kleine Breakout hält drei Pads bereit, die Sie individuell als Eingabetasten oder gemeinsam zur Erfassung einfacher Wischgesten verwenden können. Außerdem können Sie die Pads aus der Platine brechen und für eigene Anwendungen nutzen.

Sobald Sie alles verdrahtet haben, laden Sie den Sketch *CapSliderMicroOledExample.ino* auf den *Pro Micro* hoch. Bestaunen Sie wieder kurz unser Flammenlogo, dann sollte das Display leer sein. Wenn Sie eines der drei Pads auf dem Breakout berühren, leuchtet ein entsprechendes vertikales Rechteck auf dem OLED-Display auf. SparkFun verwendet kapazitive Touchpads auf allen seinen Produktionsplätzen, weil sie im Gegensatz zu taktilen Schaltern nie verschleißen und nie neu platziert werden müssen. Im Gegensatz zu taktilen Schaltern erhalten Sie allerdings auch nie dieses befriedigende „Klick"...

Qwiic-Anwesenheitsdetektor

Schaltung: **Bild 5**; Programmcode: **Weblink [4]**.

Sie möchten wissen, ob ein Objekt vorhanden ist, das vorher nicht vorhanden war? Oder ob sich ein Objekt über einen bestimmten Punkt hinaus in den Raum bewegt hat? Dieser Sketch kann beides. Auch hier behalten wir das *Micro OLED Display* aus dem vorherigen Beispiel bei und tauschen den *Capacitive Touch Slider* gegen das *SparkFun Proximity Sensor Breakout*. Dieser kleine Sensor ist sehr einfach und ebenso nützlich, da er einen Infrarotsender, einen Umgebungslichtsensor und einen Näherungssensor verbindet, um Objekte im Nahbereich zu erkennen.

Wenn alles angeschlossen ist, laden Sie den Sketch *ProximityMicroOLEDexample.ino* auf den *Pro Micro*. Auch hier sollten Sie kurz das SparkFun Flammenlogo sehen. Sollte die OLED-Anzeige dabei einfrieren, drücken Sie kurz die Reset-Taste am *Pro Micro*. Wenn dies nicht hilft, liegt wahrscheinlich ein Problem mit den Qwiic-Kabeln oder -Steckern vor. Verkabeln Sie die Angelegenheit neu und richten ein kurzes Gebet an den Elektronikgott, bevor Sie den Controller wieder einschalten. Das Display bleibt leer, leuchtet aber auf, wenn Sie Ihre Hand oder einen Gegenstand innerhalb etwa 20 Zentimetern um das *Proximity Sensor Breakout* halten. Diese Sensoren sind in allen Arten von berührungslosen Geräten wie automatischen Papierhandtuchspendern üblich.

Taktile Eingabe, akustische Ausgabe

Schaltung: **Bild 6**; Programmcode: **Weblink [5]**.

Qwiic bietet mehr als nur die Ausgabe von Daten aus Encodern und Sensoren. Durch das modulare Konzept erlaubt Ihnen das System, die Funktionen, die Ihr Projekt benötigt, im Plug-and-Play-Verfahren zu kombinieren. In diesem Beispiel werden wir das OLED-Display und die Sensoren komplett weglassen. Mit dem *Sparkfun Qwiic MP3 Trigger* und dem *Qwiic Joystick* zeigt dieser Sketch, wie Sie mit Qwiic auf einfache Weise hochgradig interaktive Objekte erstellen können.

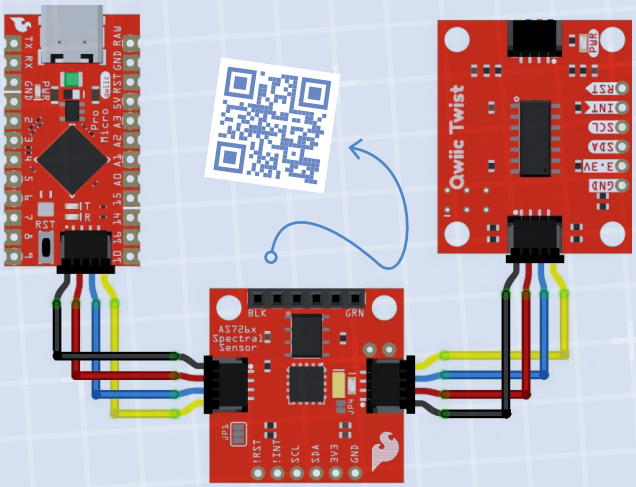


Bild 7. Platinenanschluss für „Rot, Grün oder Blau?“

Ähnlich wie bei den vorherigen Beispielen schließen Sie einfach die beiden Breakouts mit Qwiic-Kabeln an den *Pro Micro* an. Für diese Schaltung benötigen Sie einen Lautsprecher oder einen Kopfhörer an der Kopfhörerbuchse des *MP3 Trigger*. Außerdem müssen Sie vier MP3-Dateien mit den Namen *F001.mp3*, *F002.mp3*, *F003.mp3* und *F004.mp3* in das Rootverzeichnis einer Micro-SD-Karte kopieren und diese SD-Karte in den SD-Slot des *MP3 Triggers* stecken. Jetzt können Sie den Sketch *joystickmp3example.ino* auf den *Pro Micro* laden.

Wenn alles richtig angeschlossen ist, sollten Sie die MP3-Dateien auf der microSD-Karte über Ihren Kopfhörer/Lautsprecher hören, wenn Sie den Joystick nach oben, unten, links oder rechts drücken. Wenn Sie die Audiotracks aus SparkFuns GitHub-Repository *Simple Sketches* herunterladen, wird ein „up, down, left, right“ ertönen, je nachdem, in welche Richtung der Joystick gekippt wird.

Rot, Grün oder Blau?

ⓘ Schaltung: **Bild 7**; Programmcode: **Weblink [6]**.

Bei der Unzahl der möglichen Kombinationen von Qwiic-Breakouts kann es Spaß machen, sich wahllos ein paar davon zu schnappen und zu versuchen, ein Gadget daraus zu machen. In diesem Sketch werden wir wieder den *Qwiic Twist RGB Rotary Encoder* einsetzen und ihn mit dem *SparkFun Qwiic Spectral Sensor Breakout* (und natürlich dem *Pro Micro*) verbinden. Diesen Spektralsensor gibt es in mehreren Varianten, aber für diesen Sketch wollen wir die Version für das sichtbare Spektrum verwenden. Er soll erkennen, ob eine transparente farbige Murmel rot, grün oder blau ist. Sie benötigen also einige rote, grüne und blaue Objekte zum Testen. Ich habe Murmeln verwendet, aber so ziemlich alles, was über den Sensor passt, wird funktionieren.

Laden Sie die Datei *VisSpectrumTwistColorExample.ino* auf den *Pro Micro* hoch. Wenn alles richtig eingestellt ist, sollten Sie in der Lage sein, verschiedene Objekte auf ihre Farben zu testen, indem Sie sie über den Spektralsensor halten und dabei den Knopf des Drehencoders drücken. Je nach vom Sensor erkannter Farbe ändert sich die RGB-LED des Drehgebers. Wenn Sie ein Objekt testen, das nicht rot, nicht grün und gar nicht blau ist, wird das Ergebnis diejenige Farbe (R, G oder B) sein, die den größten Farbanteil besitzt.

200696-02

WEBLINKS

[1] **Programmcode Drehgeber mit Farbwechsel**: https://bit.ly/2XHeLfv

[2] **Programmcode Qwiic-Umweltthermometer**: https://bit.ly/35BP5VJ

[3] **Programmcode Kapazitive Toucheingabe leicht gemacht**: https://bit.ly/2XB5FAO

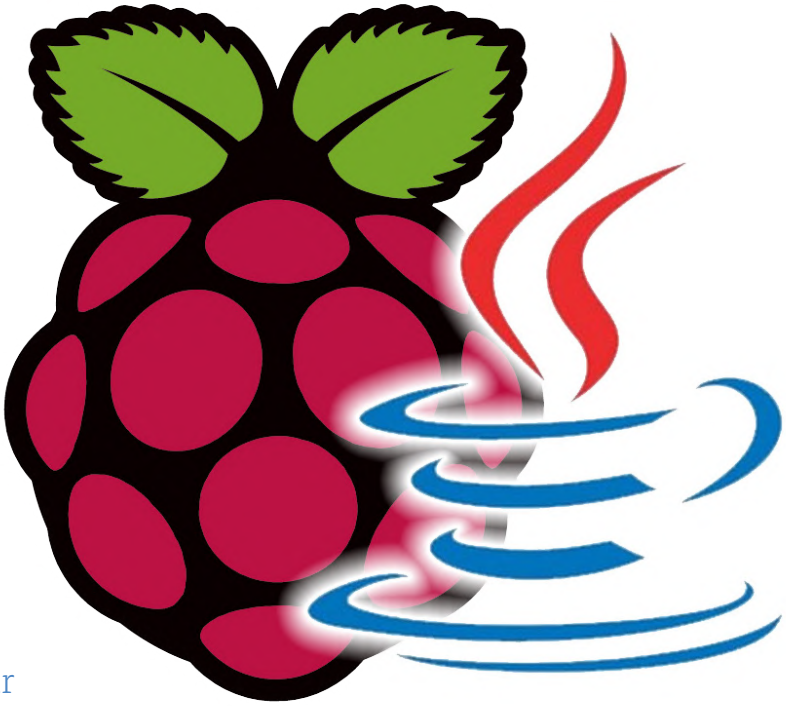
[4] **Programmcode Qwiic-Anwesenheitsdetektor**: https://bit.ly/3qjqjBN

[5] **Programmcode Taktile Eingabe, akustische Ausgabe**: https://bit.ly/3idt4BU

[6] **Programmcode Rot, Grün oder Blau?**: https://bit.ly/3ibXnJa

Java auf dem Raspberry Pi

Teil 2: Steuerung von GPIOs mit einem Spring-REST-Service



Von **Frank Delporte** (Belgien)

Ist Java eine geeignete Programmiersprache für den Raspberry Pi? Im vorherigen Artikel wird dies mit einem klaren „Ja“ beantwortet! Nachdem wir einige Java-Demoapplikationen mit nur einer Datei kennengelernt haben, ist es an der Zeit, etwas tiefer einzusteigen. Wir gehen nun ein paar Schritte weiter und konstruieren eine vollständige Anwendung mit mehreren Klassen, die uns mit REST-Webdiensten versorgen, die die GPIOs eines Raspberry Pi steuern.

Die Betriebssystem-Version *Raspberry Pi OS Full (32-bit)* enthält bereits das *Java Development Kit* (JDK) in der Version 11. Um Java-Anwendungen zu entwickeln, hilft uns jedoch eine IDE dabei, Anwendungen zu schreiben, die einfach zu warten sind. Wie im vorherigen Artikel [1] beschrieben, kann Visual Studio Code auf dem Raspberry Pi verwendet werden. Alternativ können Anwendungen in Visual Studio Code auch auf einem PC entwickelt und dann auf dem Raspberry Pi kompiliert und ausgeführt werden.
Für diesen Artikel werden wir den Code direkt auf dem Raspberry Pi schreiben. Dazu benötigen wir einige zusätzliche Tools, also installieren wir diese zunächst.

Maven

Wir werden Maven verwenden, um die Anwendung auf unserem Raspberry Pi zu erstellen. Maven kompiliert den Code zusammen mit den erforderlichen Abhängigkeiten in eine einzige JAR-Datei. Dies ist dank der Konfigurationsdatei *pom.xml* möglich, die sich im Stammverzeichnis des Projekts befindet.
Maven wird mit einem einzigen Befehl installiert. Danach überprüfen wir die Installation, indem wir die Version wie folgt abfragen:

```
$ sudo apt install maven
$ mvn -v
Apache Maven 3.6.0
Maven home: /usr/share/maven
```

Pi4J

Um die GPIOs und verschiedene daran angeschlossene elektronischen Komponenten zu steuern, verwenden wir die Pi4J-Bibliothek, die eine Brücke zwischen unserem Java-Code und den GPIO-Pins (GPIO = General Purpose Input/Output) des Raspberry Pi schlägt. Um die volle Unterstützung der Pi4J-Bibliothek auf dem Raspberry Pi zu erhalten, müssen wir eine zusätzliche Software installieren. Auch dazu brauchen wir nur einen einzigen Befehl:

```
$ curl -sSL https://pi4j.com/install | sudo bash
```

Update von WiringPi

Ein letzter Schritt der Vorbereitung ist erforderlich. Wenn Sie einen Raspberry Pi 4 verwenden, müssen Sie die von Pi4J verwendete native Bibliothek *WiringPi* zur Ansteuerung der GPIOs aktualisieren. Da sich die Architektur des System-on-Chip (SoC) mit der Version 4 geändert hat, wird eine neue Version von WiringPi benötigt. Leider wurde dieses Projekt im letzten Jahr „deprecated“, aber dank der Open-Source-Community ist eine inoffizielle Version verfügbar, die durch ein von Pi4J bereitgestelltes Skript installiert werden kann. Führen Sie dazu den Befehl aus:

```
sudo pi4j -wiringpi
```

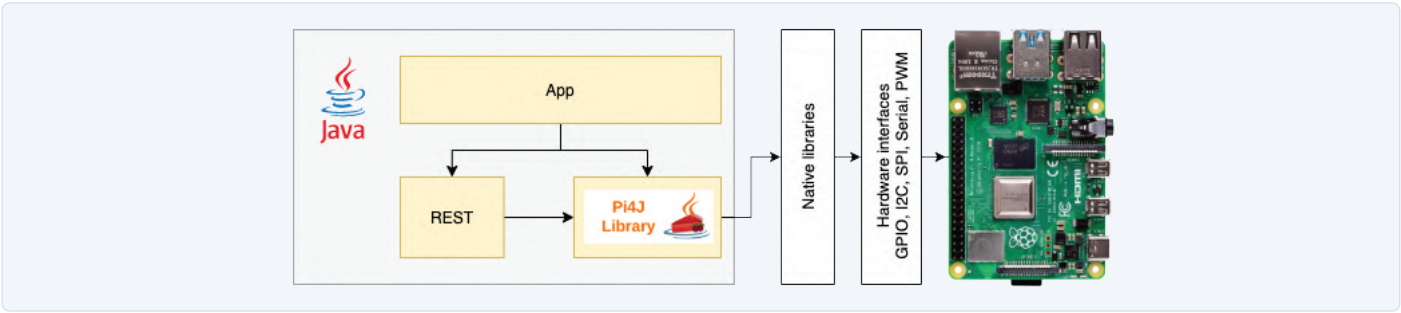


Bild 1. Überblick über die Anwendung, die den Spring-REST-Service und die Pi4J-Bibliothek verwendet.

Der Befehl bezieht das Projekt von GitHub, kompiliert und installiert es auf Ihrem Raspberry Pi in einem Rutsch. Wenn Sie die Version abfragen, werden Sie sehen, dass sie von der Standardversion 2.50 auf 2.60 hochgestuft wurde:

```
$ gpio -v
gpio version: 2.50
$ sudo pi4j -wiringpi
$ pi4j -v
-------------------------------------------
              THE Pi4J PROJECT
-------------------------------------------
  PI4J.VERSION     : 1.3
  PI4J.TIMESTAMP   : 2021-01-28 04:14:07
-----------------------------------------
  WIRINGPI.PATH : /usr/lib/libwiringPi.so /usr/local/lib/
    libwiringPi.so
  WIRINGPI.VERSION : 2.60
-------------------------------------------
```

Bild 2. Erforderliche Konfigurationseinstellungen für Spring Initializr.

Die Applikation

Der vollständige Code dieser Anwendung ist im GitHub-Repository dieses Artikels [2] im Ordner *elektor/2106* verfügbar. Dieses Projekt ist eine Proof-of-Concept-Anwendung, die die GPIOs über einen REST-Webdienst steuert. Es verwendet das Spring-Framework, eine Software, die viele Werkzeuge zum Erstellen leistungsfähiger Anwendungen mit minimalem Code bietet (**Bild 1**). Der Prozess zum Erstellen der verschiedenen Dateien, aus denen dieses Projekt besteht, wird im Folgenden beschrieben. Wenn dies nicht funktioniert, sollte aber der Repository-Code out-of-the-box funktionieren.
Das fertige Projekt wird wie folgt von GitHub heruntergeladen:

```
pi@raspberrypi:~ $ git clone https://github.com/FDelporte/
    elektor
Cloning into 'elektor'...
remote: Enumerating objects: 34, done.
remote: Counting objects: 100% (34/34), done.
remote: Compressing objects: 100% (24/24), done.
remote: Total 34 (delta 2), reused 34 (delta 2), pack-reused 0
Unpacking objects: 100% (34/34), done.
pi@raspberrypi:~ $ cd elektor/2106
pi@raspberrypi:~/elektor/2106 $ ls -l
total 8
-rw-r--r-- 1 pi pi 1720 Feb 15 14:23 pom.xml
drwxr-xr-x 3 pi pi 4096 Feb 15 14:23 src
```

Was sind Spring und seine Werkzeuge?

Spring ist ein Framework, das die Entwicklung von (Business-) Java-Anwendungen vereinfacht und beschleunigt. *Spring Boot* ist eine Schicht, die auf Spring aufsetzt und „fertige" Pakete bereitstellt, die es ermöglichen, eigenständige Spring-basierte Anwendungen zu erstellen, die man „einfach ausführen" kann. Dies wird durch ein Prinzip „Convention over Convention" erreicht, was bedeutet, dass standardmäßig alles nach einer vordefinierten Konvention funktioniert. Wenn Sie etwas anders machen wollen, können Sie Ihren eigenen Weg konfigurieren. Als wichtigste Fähigkeiten nennt die Spring-Boot-Website [3]:

- Einfache Erstellung von eigenständigen Spring-Anwendungen.
- Ermöglicht die Einbettung eines Webservers in Ihre Anwendung (Tomcat, Jetty oder Undertow).
- Stellt „Starter"-Abhängigkeiten zur Verfügung, um Ihre Build-Konfiguration zu vereinfachen.
- Konfiguriert Spring und Bibliotheken von Drittanbietern automatisch, wann immer dies möglich ist.
- Bietet produktionsreife Funktionen wie Metrics, Health Checks und externalisierte Konfiguration.
- Keine Notwendigkeit für Code-Generierung oder XML-Konfiguration.

Schließlich gibt es noch *Spring Initializr* [4], ein Online-Tool zur schnel-

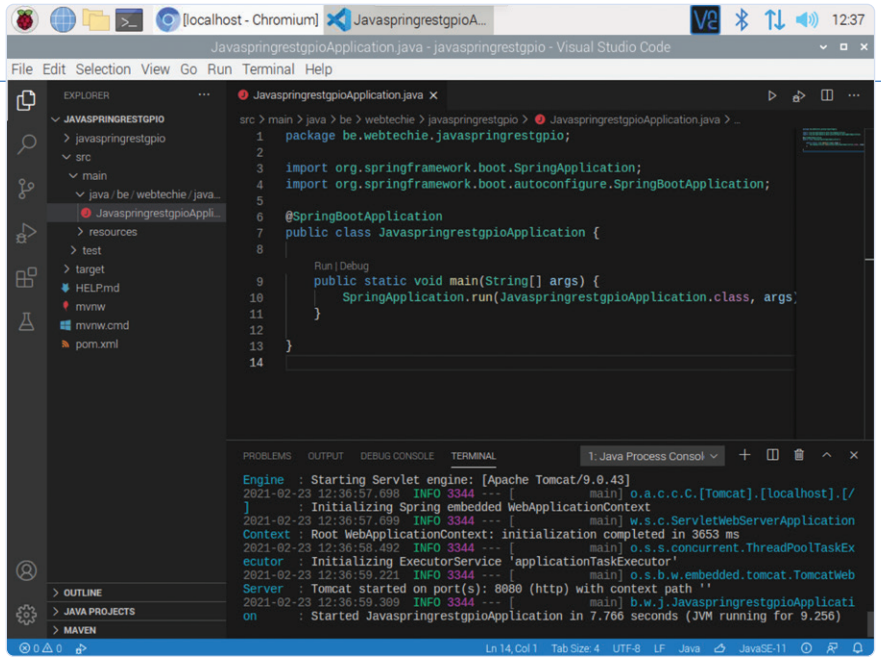


Bild 3. Ausführbares Projekt in Visual Code Studio.

len Erstellung einer Starter-Anwendung, die alle erforderlichen Spring-Boot-Pakete enthält.

Erste Schritte

Um dieses Projekt von Grund auf zu erstellen, verwenden wir zunächst Spring Initializr, ein webbasiertes Konfigurationsformular zur Erstellung eines Einstiegspaket. Folgen Sie dem Link in [4] und geben Sie im Formular die in **Bild 2** gezeigten Informationen an. Achten Sie darauf, dass Sie auf der rechten Seite auch die Schaltfläche *Dependencies* anklicken. In dem sich öffnenden Fenster müssen wir die Abhängigkeit *Spring Web* hinzufügen, die sich über das Suchfeld schnell finden lässt. Mit einem Klick auf diesen Eintrag wird die Abhängigkeit dem Projekt hinzugefügt. Sollte versehentlich eine unerwünschte Abhängigkeit hinzugefügt worden sein, kann diese durch einen Klick auf den roten Button (ganz rechts) wieder entfernt werden.

Listing 1. Ausgabe einer Starter-Applikation.

```
°  .°°  ____  °°°°°°°°°°  _°°°°°°°°°°°  __ _ _
°/\\ / ___'_ __ _ _(_)_ __°  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _' | \ \ \ \
°\\/° ___)| |_)| | | | | || (_| |° ) ) ) )
°  '° |____| .__|_| |_|_| |_\__, | / / / /
°=========|_|==============|___/=/_/_/_/
°:: Spring Boot ::°°°°°°°° (v2.4.2)
°
Starting JavaSpringRestApplication
No active profile set, falling back to default profiles: default
Tomcat initialized with port(s): 8080 (http)
Starting service [Tomcat]
Starting Servlet engine: [Apache Tomcat/9.0.41]
Initializing Spring embedded WebApplicationContext
Root WebApplicationContext: initialization completed in 1607 ms
Initializing ExecutorService 'applicationTaskExecutor'
Tomcat started on port(s): 8080 (http) with context path ''
Started JavaSpringRestApplication in 4.582 seconds (JVM running for 7.196)
Initializing Spring DispatcherServlet 'dispatcherServlet'
Initializing Servlet 'dispatcherServlet'
Completed initialization in 9 ms
```

Das Hinzufügen der Abhängigkeit *Spring Web* ermöglicht es, eine REST-Anwendung [5] zu erstellen, die es ermöglicht, uns über Webseiten im Browser mit der Anwendung zu verbinden. Ein Klick auf die Schaltfläche *Generate* liefert eine ZIP-Datei mit einem fertigen Maven-Projekt. Nachdem es in ein geeignetes Verzeichnis entpackt wurde, kann es in Visual Studio Code [6] sowohl auf dem Raspberry Pi als auch auf einem PC geöffnet werden. Verwenden Sie einfach *File -> Open folder...* und wählen Sie den entpackten Ordner. Als nächstes suchen Sie die Datei *JavaSpringRestApplication.java* (unter *src/main/java*). Im Kontextmenü dieses Dateinamens wählen Sie die Option *Run,* so dass die Anwendung gestartet wird (**Bild 3**). Wenn Ihre Installation von Visual Studio Code nicht über die erforderliche Java-Unterstützung verfügt, erscheint ein Popup-Fenster, das zur Installation auffordert.

Im Terminal-Fenster erhalten wir die Ausgabe in **Listing 1** (hier ohne Zeitmarken). Was sagt uns diese Ausgabe?

- Die JVM (Java Virtual Machine) ist sieben Sekunden lang gelaufen, wovon 4,5 s zum Starten der Anwendung benötigt wurden.
- Tomcat (der Webserver) ist in die Anwendung eingebettet und läuft auf Port 8080.

Öffnen wir also einen Browser und gehen auf *http://localhost:8080/* (**Bild 4**). Große Enttäuschung! Wir erhalten eine Fehlerseite. Doch das liegt nur daran, dass wir noch nichts erstellt haben, was auf dieser Seite angezeigt werden könnte! Gleichzeitig beweist diese Fehlerseite aber, dass unsere Anwendung funktioniert und als Webserver arbeitet. Sie weiß nur noch nicht, was sie uns mitteilen soll.

Unsere eigene Anwendung

Wir werden nun dieses Spring-Starterpaket um die Funktionalität erweitern, die wir in unserer Anwendung haben wollen, nämlich die Steuerung der GPIOs über Java-Code. Dazu müssen wir die Pi4J-Bibliothek in unser Projekt einbinden. Da wir den vollen Satz an Java-Methoden dieser Bibliothek nutzen können, müssen wir die Hardware nicht selbst ansteuern. Es ist jedoch zu bedenken, dass wir aufgrund der Abhängigkeit dieser Bibliothek vom Raspberry Pi diese Anwendung nicht vollständig auf dem PC ausführen oder testen können. Dazu muss die erzeugte JAR-Datei auf dem Raspberry Pi ausgeführt werden. In den Code-Beispielen, die hier zur Verfügung gestellt werden, ist jede Variable und Methode im JavaDoc-Stil (beginnend mit `/**`) kommentiert, um ihren Zweck zu erklären.

Hinzufügen der Abhängigkeiten

Dem mit Spring Initializr erstellten Projekt fügen wir die `pi4j-core`-Abhängigkeit zur *pom.xml*-Datei hinzu, um auf die pi4j-Methoden zugreifen zu können:

```
<dependency>
<groupId>com.pi4j</groupId>
<artifactId>pi4j-core</artifactId>
<version>1.3</version>
<scope>compile</scope>
</dependency>
```

Und wenn wir schon dabei sind, können wir gleich auch die Open-API-Abhängigkeit (`springdoc-openapi-ui`) hinzufügen, die wir später zum Testen der REST-Dienste verwenden:

```
<dependency>
<groupId>org.springdoc</groupId>
<artifactId>springdoc-openapi-ui</artifactId>
<version>1.5.1</version>
</dependency>
```

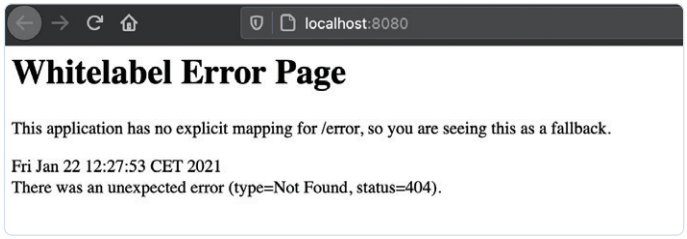


Bild 4. Die Standard-Fehlerwebseite beweist, dass der Webserver funktionsfähig ist.

Hinzufügen eines Information-REST-Controllers

Das erste, was wir mit dieser Anwendung zeigen werden, sind die von der Pi4J-Bibliothek bereitgestellten Informationen über unseren Raspberry Pi. Wir erstellen zunächst ein Paket *controller* mit einer Datei *InfoRestController.java*. In Visual Studio Code klicken Sie einfach mit der rechten Maustaste auf den Eintrag *java\be\webtechie\javaspringrestgpio* und wählen *New Folder*. Sie taufen den neuen Ordner *controller*, klicken mit der rechten Maustaste auf den Ordner und wählen *New File*. Nennen Sie die Datei *InfoRestController.java*.

In den Material zu dem Artíkel finden Sie den vollständigen Code, aber **Listing 2** zeigt schon einmal einen kurzen Ausschnitt. Jede Methode ist ein REST-Mapping, das einen bestimmten Satz von Schlüssel-Wert-Paaren mit Informationen über den Raspberry Pi zurückgibt.

Listing 2. Sektion der Datei InfoRestController.java.

```
/**
 * Provides a REST-interface to expose all board info.
 */
@RestController
@RequestMapping("info")
public class InfoRestController {
    private Logger logger = LoggerFactory.getLogger(this.getClass());

    /**
     * Get the OS info.
     */
    @GetMapping(path = "os", produces = "application/json")
    public Map<String, String> getOsInfo() {
        Map<String, String> map = new TreeMap<>();
        try {
            map.put("Name", SystemInfo.getOsName());
        } catch (Exception ex) {
            logger.error("OS name not available, error: {}",
                        ex.getMessage());
        }
        try {
            map.put("Version", SystemInfo.getOsVersion());
        } catch (Exception ex) {
            logger.error("OS version not available, error: {}",
                        ex.getMessage());
        }
        return map;
    }

    /**
     * Get the Java info.
     */
    @GetMapping(path = "java", produces = "application/json")
    public Map<String, String> getJavaInfo() {
        Map<String, String> map = new TreeMap<>();
        map.put("Vendor ", SystemInfo.getJavaVendor());
        map.put("VendorURL", SystemInfo.getJavaVendorUrl());
        map.put("Version", SystemInfo.getJavaVersion());
        map.put("VM", SystemInfo.getJavaVirtualMachine());
        map.put("Runtime", SystemInfo.getJavaRuntime());
        return map;
    }

    ...
}
```

Hinzufügen des GPIO-Managers

Bevor wir den GPIO-REST-Controller erstellen können, müssen wir eine Datei namens *GpioManager.java* hinzufügen, um die Pi4J-Aufrufe zu verarbeiten. Diese wird in einem *manager*-Paket abgelegt, das wie zuvor beschieben als *New Folder* und *New File* angelegt wurde. Dieser Manager soll die initialisierten GPIO-Pins speichern und die Pi4J-Methoden aufrufen, die mit den GPIOs interagieren. Auch hier ist ein kurzer Ausschnitt des Codes in **Listing 3** zu sehen, während der vollständige Code für diese Klasse im Repository zu finden ist. In dieser Klasse wird `@Service` verwendet, das das Spring-Framework anweist, eine einzelne Instanz dieses Objekts im Speicher zu halten. Damit haben wir jederzeit eine Liste der bereitgestellten GPIOs zur Verfügung.

Hinzufügen des GPIO-REST-Controllers

Abschließend fügen wir dem *controller*-Paket (Ordner) einen GPIO-Controller mit einer REST-Schnittstelle hinzu. Diese Datei heißt *GpioRestController.java* und stellt die Pi4J-GPIO-Methoden zur Verfügung, die

Listing 3. Sektion der Datei GpioManager.java.

```
/**
 * Service instance managing the {@link GpioFactory}.
 */
@Service
public class GpioManager {
    private Logger logger = LoggerFactory.getLogger(this.getClass());

    /**
     * The GPIO controller.
     */
    private final GpioController gpio;

    /**
     * List of the provisioned pins with the address as key.
     */
    private final Map<Integer, Object> provisionedPins =
            new HashMap<>();

    /**
     * Constructor which initializes the Pi4J {@link GpioController}.
     */
    public GpioManager() {
         String osName = SystemInfo.getOsName();
         if (osName.toLowerCase().contains("raspberry") || osName.toLowerCase().contains("linux")) {
             this.gpio = GpioFactory.getInstance();
         } else {
             logger.error("GPIO could not be initialized. Not running on Raspberry Pi but '{}'", osName);
             this.gpio = null;
         }
     }

    /**
     * Get the pin for the given address.
     *
     * @param address The address of the GPIO pin.
     * @return The {@link Pin} or null when not found.
     */
    private Pin getPinByAddress(int address) {
        Pin pin = RaspiPin.getPinByAddress(address);
        if (pin == null) {
            logger.error("No pin available for address {}", address);
        }
        return pin;
    }

    /**
     * Provision a GPIO as digital output pin.
     *
     * @param address The address of the GPIO pin.
```

wir in der Klasse *GpioManager.java* definiert haben. Auch hier ist ein Ausschnitt des Codes in **Listing 4** gezeigt, während der vollständige Code im Repository verfügbar ist.

Ausführen der Anwendung

Dieser Schritt kann sowohl auf dem PC als auch auf dem Raspberry Pi durchgeführt werden. Im ersten Fall ist es am besten, die Anwendung aus Visual Studio Code heraus zu starten, denn es macht die Installation von Maven auf dem PC überflüssig.

Da wir die *springdoc-openapi-ui*-Abhängigkeit der *pom.xml*-Datei hinzugefügt haben, stellt die Anwendung eine sehr nützliche Swagger-Webseite zum Testen der REST-Dienste zur Verfügung. Swagger ist ein weiteres Open-Source-Projekt, das eine einfache Webpage-Oberfläche zum Testen unseres Java-Codes generiert, indem es die von uns erstellten Controller automatisch visualisiert.

Es gibt zwei Möglichkeiten, die Anwendung zu starten. In Visual Studio Code kann sie mit *Run* (Raspberry Pi oder PC) gestartet werden. Eine Alternative für den Raspberry Pi ist es, die Anwendung mit *mvn package* in eine jar-Datei zu überführen. Ist dies geschehen, wird die Datei wie folgt von der Kommandozeile aus gestartet:

```
java -jar target/javaspringrestgpio-0.0.1-SNAPSHOT.jar
```

Öffnen Sie in einem Browser auf dem Raspberry Pi die Swagger-Seite mit `http://localhost:8080/swagger-ui.html`. Alternativ können Sie auf einem beliebigen PC im gleichen Netzwerk `http://<IP_ADDRESS_RASPBERRY_PI>:8080/swagger-ui.html` verwenden. Die beiden Controller mit ihren Methoden werden wie in **Bild 5** dargestellt.

```
     * @param name The name of the GPIO pin.
     * @return True if successful.
     */
    public boolean provisionDigitalOutputPin(final int address,
            final String name) {
        if (this.provisionedPins.containsKey(address)) {
            throw new IllegalArgumentException("There is already"
                  + " a provisioned pin at the given address");
        }

        final GpioPinDigitalOutput provisionedPin = this.gpio
            .provisionDigitalOutputPin(
                this.getPinByAddress(address), name, PinState.HIGH);
        provisionedPin.setShutdownOptions(true, PinState.LOW);

        this.provisionedPins.put(address, provisionedPin);

        return true;
    }

    /**
     * Toggle a pin.
     *
     * @param address The address of the GPIO pin.
     * @return True if successful.
     */
    public boolean togglePin(final int address) {
        logger.info("Toggle pin requested for address {}", address);

        Object provisionedPin = this.provisionedPins.get(address);

        if (provisionedPin == null) {
            throw new IllegalArgumentException("There is no pin"
                + " provisioned at the given address");
        } else {
            if (provisionedPin instanceof GpioPinDigitalOutput) {
                ((GpioPinDigitalOutput) provisionedPin).toggle();

                return true;
            } else {
                throw new IllegalArgumentException("The provisioned pin"
                        + " at the given address is not of the type"
                        + " GpioPinDigitalOutput");
            }
        }
    }

    ...

}
```

Bild 5: Die Swagger-Seite bietet Zugriff auf die Info- und GPIO-REST-Controller-APIs.

Listing 4. Sektion der Datei GpioRestController.java.

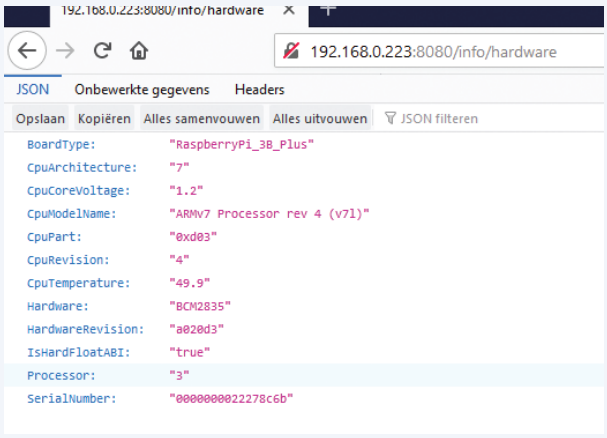
```
/**
 * Provides a REST-interface to interact with the GPIOs.
 */
@RestController
@RequestMapping("gpio")
public class GpioRestController {
    private Logger logger = LoggerFactory.getLogger(this.getClass());

    /**
     * Reference to the GPIO manager service
     */
    private final GpioManager gpioManager;

    /**
     * Constructor used by Spring to "inject" the GPIO manager
     * into this class.
     *
     * @param gpioManager {@link GpioManager}
     */
    public GpioRestController(GpioManager gpioManager) {
        this.gpioManager = gpioManager;
    }

    ...

    /**
     * Provision a GPIO as digital output pin.
     *
     * @param address The address of the GPIO pin.
     * @param name The name of the GPIO pin.
```



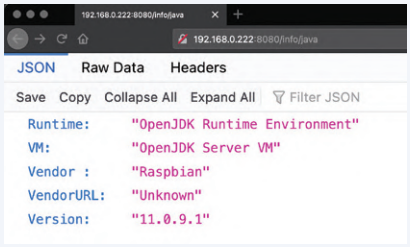


Bild 6. Die JSON-Daten können direkt durch Anhängen des Methodennamens an die URL abgerufen werden.

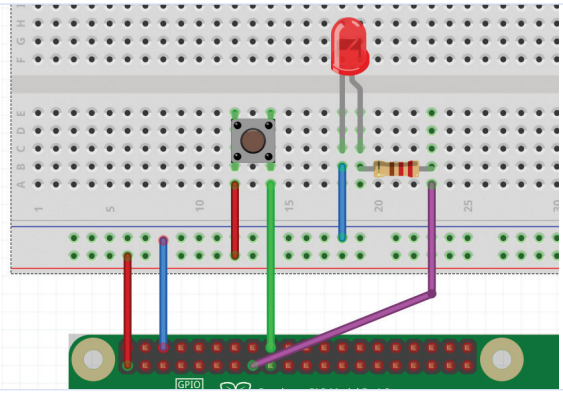


Bild 7. Schaltplan zum Anschluss des Schalters und der LED an den Raspberry Pi.

Testen des Information-REST-Controllers

Wir können auf die Schaltflächen der Swagger-Seite klicken und die verfügbaren Optionen ausführen, zum Beispiel im Abschnitt *info-rest-controller* neben der Methode *info/hardware* auf *GET* klicken. Klicken Sie dann auf *Try it out* und auf *Execute*, so dass im Abschnitt *Response body* die Antwort angezeigt wird.

Alle diese Methoden können auch direkt aus dem Browser heraus aufgerufen werden. Hängen Sie einfach den Methodennamen an die URL: Für *info/hardware* geben Sie einfach `http://localhost:8080/info/hardware` ein, und für *info/java* wäre es `http://localhost:8080/info/java`. Die Daten aus dem Aufruf dieser Methoden werden im JSON-Format angezeigt, wie in **Bild 6** zu sehen.

```
     * @return True if successful.
     */
    @PostMapping(
        path = "provision/digital/output//",
        produces = "application/json")
    public boolean provisionDigitalOutputPin(
            @PathVariable("address") int address,
            @PathVariable("name") String name) {
        return this.gpioManager
            .provisionDigitalOutputPin(address, name);
    }

    ...

    /**
     * Toggle a pin.
     *
     * @param address The address of the GPIO pin.
     * @return True if successful.
     */
    @PostMapping(
        path = "digital/toggle/",
        produces = "application/json")
    public boolean togglePin(@PathVariable("address") long address) {
        return this.gpioManager.togglePin((int) address);
    }

    ...
}
```

Test des GPIO-REST-Controllers mit LED und Taster

Um zu demonstrieren, wie die Anwendung mit den GPIOs interagieren kann, verwenden wir einen sehr einfachen Breadboard-Aufbau wie in **Bild 7** gezeigt:

- LED an GPIO-Pin 15, BCM-Controller Pin 22, WiringPi Nr. 3
- Taster an GPIO-Pin 18, BCM-Controller Pin 24, WiringPi Nr. 5

Um die angeschlossene LED ansteuern und den Tasterzustand auslesen zu können, müssen wir zunächst die GPIOs initialisieren. Zur Konfiguration des Ausgangs wird die Methode `/gpio/provision/digital/`

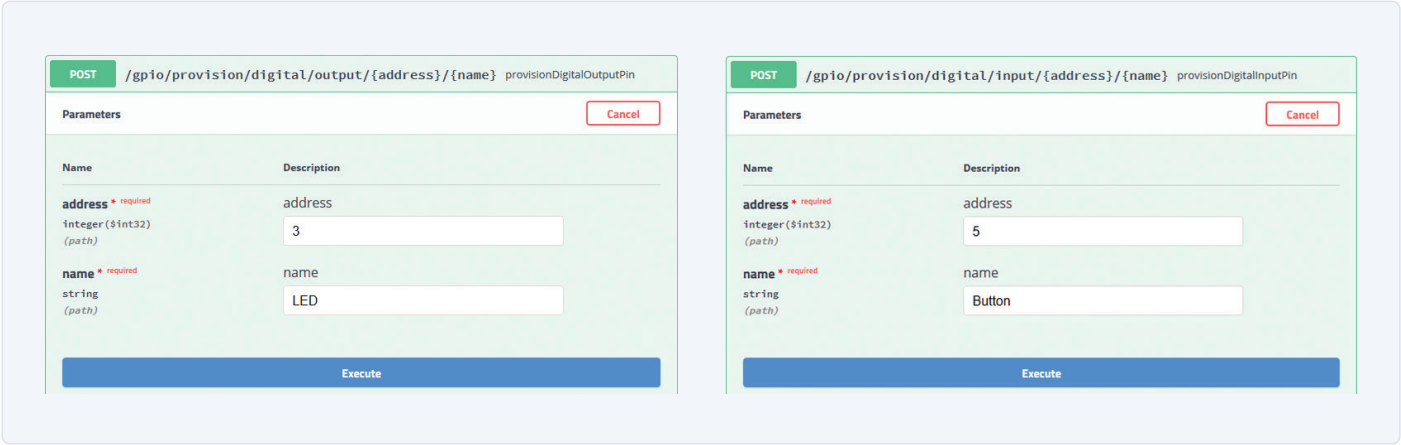


Bild 8. Initialisierung der GPIOs mit den Methoden `/gpio/provision/digital/`.

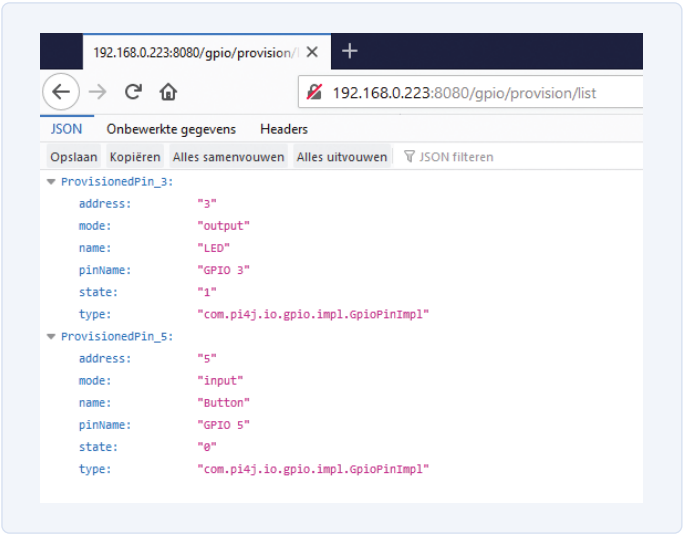


Bild 9. Abfrage der GPIO-Konfiguration mit der Methode `/gpio/provision/list`.

`output` verwendet. Kehren Sie auf die Seite `http://localhost:8080/swagger-ui.html` zurück, klicken bei dieser Methode auf *GET, Try it out* und tragen dann *3* in *address* sowie *LED* in *string* ein. Bestätigen Sie die Konfiguration durch *Execute* (**Bild 8** links).

Der Eingang wird mit der Methode `/gpio/provision/digital/input` konfiguriert. Klicken Sie bei dieser Methode auf *GET, Try it out* und tragen dann *5* in *address* sowie *Button* in *string* ein. Bestätigen Sie die Konfiguration durch *Execute* (Bild 8 rechts).

Sobald die GPIOs initialisiert sind, können wir mit der Methode `/gpio/provision/list` eine Liste der GPIOs erhalten. Verwenden Sie einfach *GET, Try it out* und *Execute* oder geben Sie die URL `http://localhost:8080/gpio/provision/list` direkt im Browser an (**Bild 9**).

Nachdem wir nun überprüft haben, dass die GPIOs einsatzbereit sind, können wir die LED mit der Methode `/gpio/digital/toggle` ein- und ausschalten, indem wir wiederholt auf die Schaltfläche *Execute* klicken (**Bild 10**). Es gibt eine weitere Methode, die die LED für eine bestimmte Zeit einschalten kann, zum Beispiel für 2 s (**Bild 11**). Die Dauer muss in Millisekunden angegeben werden.

Die Abfrage des Zustands der Taste kann über die Swagger-Schnittstelle

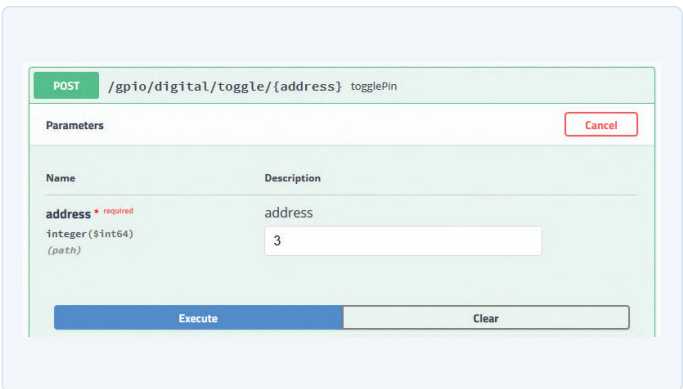


Bild 10. Verwendung der Schaltfläche ***Execute*** *in der Swagger-Schnittstelle zum Umschalten der LED.*

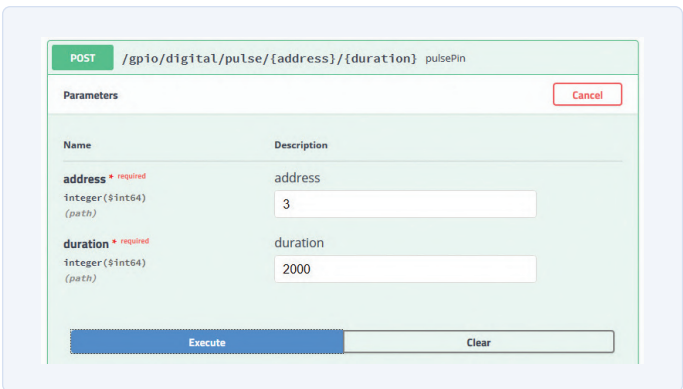


Bild 11. Die LED kann auch für eine bestimmte Dauer gepulst werden.

erfolgen, ist aber auch direkt über die URL `http://localhost:8080/gpio/state/5` verfügbar. Im Fall von **Bild 12** ist die Taste gedrückt und gibt eine 1 zurück.

Wer rastet, der rostet, wer RESTet, forscht weiter!

In dieser Anwendung wurden nur einige der Pi4J-Methoden als REST-Services gezeigt, um die Möglichkeiten und die Leistungsfähigkeit dieses Ansatzes zu zeigen. Abhängig von dem Projekt, das Sie bauen wollen, können Sie dieses Beispiel erweitern oder überarbeiten, um es Ihren Bedürfnissen auf den Leib zu schneidern.

Die Pi4J-Bibliothek wird derzeit neu geschrieben, um eine noch bessere Unterstützung für den Raspberry Pi 4 und zukünftige Versionen zu bieten. Dadurch wird sie auch mit den neusten Versionen von Java auf den neuesten Stand gebracht. In der Zwischenzeit können Sie mit der aktuellen Version loslegen und Anwendungen erstellen, bei denen Sie die Steuerung der Hardware über eine REST-API integrieren möchten.

200617-B-02

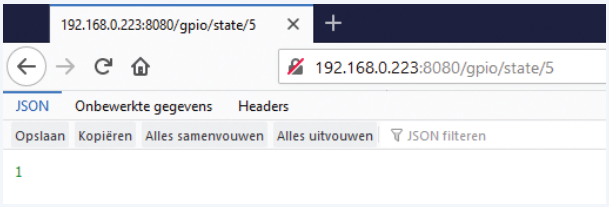


Bild 12. Die URL zusammen mit der Methode `gpio/state/5` erfasst den Zustand des Button-Eingangs.

Ein Beitrag von

Idee, Text und Illustrationen: **Frank Delporte**
Übersetzung: **Rolf Gerstendorf**
Redaktion: **Stuart Cording**
Layout: **Giel Dols**

Haben Sie Fragen oder Kommentare?

Haben Sie technische Fragen oder Anmerkungen zu diesem Artikel? Dann kontaktieren Sie den Autor direkt über javaonraspberrypi@webtechie.be oder die Redaktion über editor@elektor.com.

WEBLINKS

[1] **F. Delporte, „Java auf dem Raspberry Pi, Teil 1: GPIOs“, Elektor, Mai/Juni 2021**: www.elektormagazine.de/200617-02
[2] **GitLab-Repository für diesen Artikel**: https://bit.ly/3i9bP4v
[3] **Spring-Boot-Dokumentation**: https://spring.io/projects/ spring-boot
[4] **Spring Initialzr**: https://start.spring.io/
[5] **„What Is REST?“, Codecademy**: https://bit.ly/31odThv
[6] **Visual Studio Code**: https://code.visualstudio.com/

Raspberry Pi Compute Module 4

Ein Raspberry Pi für die Industrie

Von **Mathias Claußen** (Elektor)

Das neue Raspberry Pi Compute Module 4 ist eingetroffen. Es hat einen neuen Formfaktor und neue Peripherie, aber ist es auch gut? In Kombination mit dem Raspberry Pi Compute Module IO Board ergeben sich neue Möglichkeiten für RPi-basierende Systeme, die mit älteren Modulen nicht machbar waren. Dank der PCI-Express-Schnittstelle kann man jetzt auch wirklich schnelle Peripherie an den Raspberry Pi anschließen.

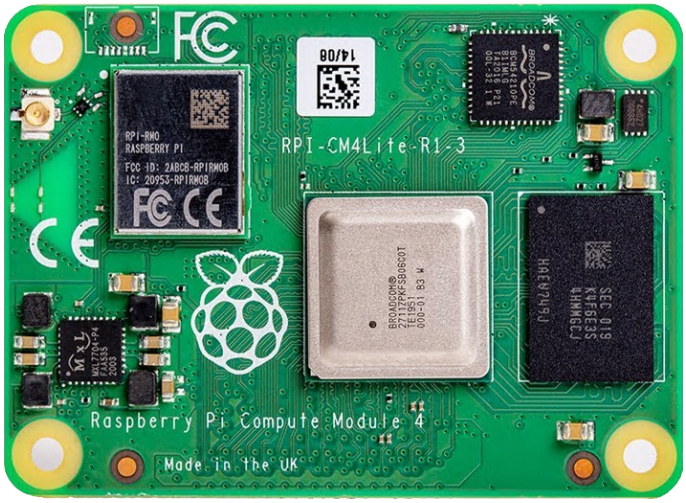


Bild 1. Die Oberseite der CM4-Platine.

Die Gerüchte kursieren schon seit einiger Zeit, doch jetzt gibt es den schlagenden Beweis für ein auf dem Raspberry Pi 4 basierendes Compute Module: Ein CM4 = Raspberry Pi Compute Module 4 liegt direkt vor mir. Beim CM bzw. Compute Module handelt es sich um ein abgespecktes RPi-Modul, das in andere Produkte integriert werden soll, die mehr Design-Flexibilität benötigen, als ein „normaler" Raspberry Pi bieten kann. Außerdem ermöglicht es die enge Integration kundenspezifischer Elektronik rund um das RPi-SOC (z.B. für industriellen Einsatz oder digitale Anzeigen). Die CMs mit den auf Raspberry Pi Zero und Raspberry Pi 3B(+) basierenden SoCs wurden im SO-DIMM-Formfaktor produziert und lassen sich daher wie RAM-Riegel einfach stecken. Diese Form bot zwar große Flexibilität, hatte aber auch einige Nachteile, da weder Ethernet noch ein USB-Hub integriert waren. Man musste diese Teile als Entwickler also bei Bedarf selbst hinzufügen. Außerdem gab es kein WLAN, was daher gegebenenfalls selbst auf der eigenen Basisplatine installiert werden musste und die Kosten erhöhte. Diese Einschränkungen disqualifizierten sowohl das CM1 als auch das CM3 für einige Anwendungen. Außerdem fehlten Hochgeschwindigkeitsschnittstellen, da beide Vorgänger-Module nur einen einzigen USB-2.0-Port hatten. Der Anschluss von Peripherie mit hoher Bandbreite oder niedriger Latenz war folglich nicht leicht möglich. Angesichts der Preise für CM1 und CM3 war für einige Projekte ein normaler Raspberry Pi besser geeignet, da das bei diesen CMs Fehlende bereits auf den Boards mit dem Standard-Formfaktor installiert war. Insofern läutet das neue CM4 eine dramatische Wende zum Besseren ein, da viel Energie auf die Optimierung gegenüber seinen Vorgängern verwendet wurde.

Erster Blick auf die Platine

Die **Bilder 1** und **2** geben einen ersten Eindruck von der Platine. Das CM4 gibt es jetzt in 32 (!) Varianten. Man bekommt es mit WLAN oder

ohne, mit 8/16/32 GB EMMC oder kein EMMC sowie mit 1/2/4 GB und sogar 8 GB RAM. Man kann nun die exakt zur Anwendung passende Variante auswählen. Im **Kasten Spezifikationen** finden Sie die Eigenschaften, die mit denen des Raspberry Pi 4 identisch sind. **Bild 3** zeigt die Bestückung eines CM4 mit WLAN, aber ohne EMMC. Auf der linken Seite befindet sich der für die komplette Stromversorgung des CM4 zuständige PMIC (**Bild 4**). Direkt darüber sieht man das WLAN-Modul, das den Betrieb mit 5 GHz und 2,4 GHz unterstützt und zudem Bluetooth integriert hat. In der Mitte (**Bild 5**) befindet sich das SoC BCM2711. Dieses zentrale Bauteil bietet vier Cortex-A72-Kerne. Oben rechts (**Bild 6**) steckt mit dem BCM54210PE eine 1000Base-T Ethernet-Elektronik samt IEEE1588-2008-Unterstützung für präzises Timing. Eine eigene Basisplatine benötigt also lediglich noch einen MagJack, um volle Ethernet-Unterstützung zu realisieren (etwas zusätzlicher ESD-Schutz wäre ebenfalls wünschenswert). Auf der rechten Seite (**Bild 7**) ist das RAM-Modul und darüber noch ein kleiner Flash-Chip für grundlegende Boot-Informationen zu sehen.

Auf der Unterseite der Platine (Bild 2) sind die üblichen Kondensatoren und Oszillatoren zu sehen. Oben und unten an den Längsseiten sind zwei 100-polige Hirose-Stecker angebracht, die zur Befestigung des CM4 auf einer Basisplatine gedacht sind. Dies ist eine wichtige Änderung gegenüber den früheren CMs. Auf der einen Seite des Steckverbinders sind alle Spannungen und Signale mit niedrigen Geschwindigkeiten angeschlossen; die andere Seite führt alle Hochgeschwindigkeitssignale, was das Routing der Basisplatine vereinfacht. Für die Stromversorgung braucht das CM4 lediglich 5 V, denn der integrierte DC/DC-Wandler erzeugt alle für das CM4 erforderlichen Spannungen. Weitere Informationen finden Sie im Kasten Spezifikationen. Die USB-3.0-Schnittstelle des RPi4B ist nicht aufgeführt – dafür gibt es einen guten Grund. Beim Raspberry Pi 4B war ein USB-3.0-Controller über eine PCIe-2.0 Single-Lane-Verbindung an den BCM2711 angeschlossen. Diese Schaltung ist nicht im CM4 enthalten, weshalb die PCIe-Lane für den eigenen Gebrauch zur Verfügung steht. Dies schafft neue Möglichkeiten für den Einsatz eines RPi auch dort, wo das bisher nicht denkbar war. Im nächsten Abschnitt geht es darum, wie man mit dem CM4 loslegt und wie man eigene Projekte angeht.

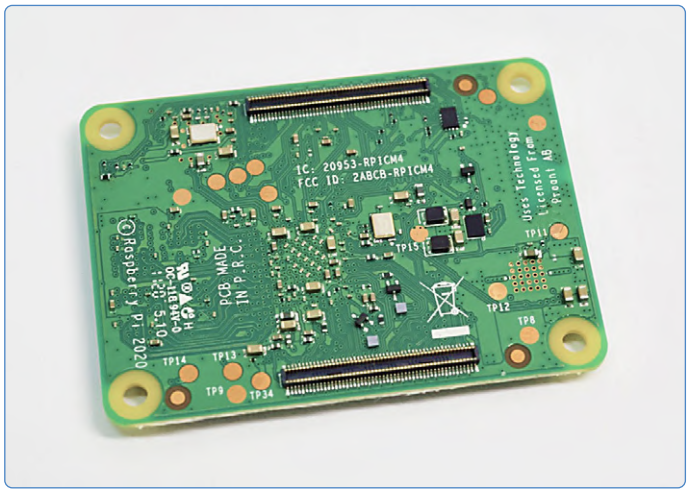


Bild 2. Die Unterseite der CM4-Platine.

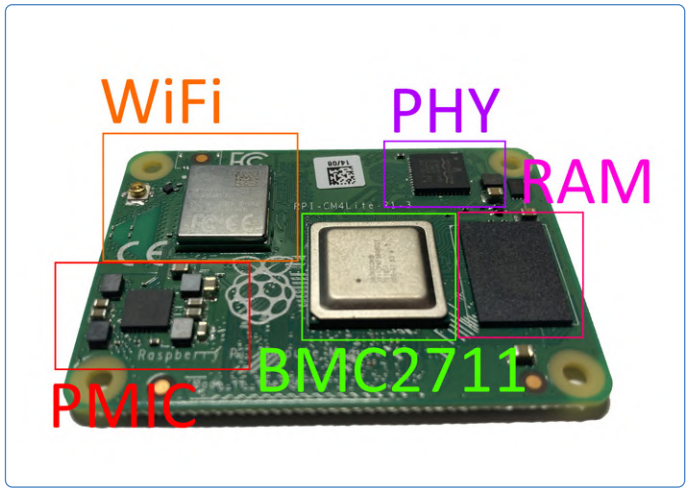


Bild 3. Bauteile auf der Platine des CM4.

Bild 4. PMIC.

Bild 5. Das SOC BCM2711.

Bild 6. Ethernet.

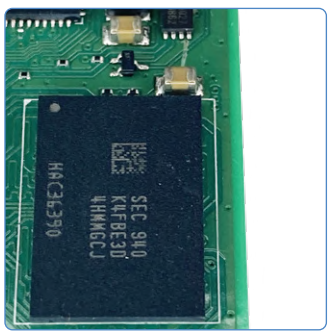


Bild 7. RAM-Chip und EEPROM.

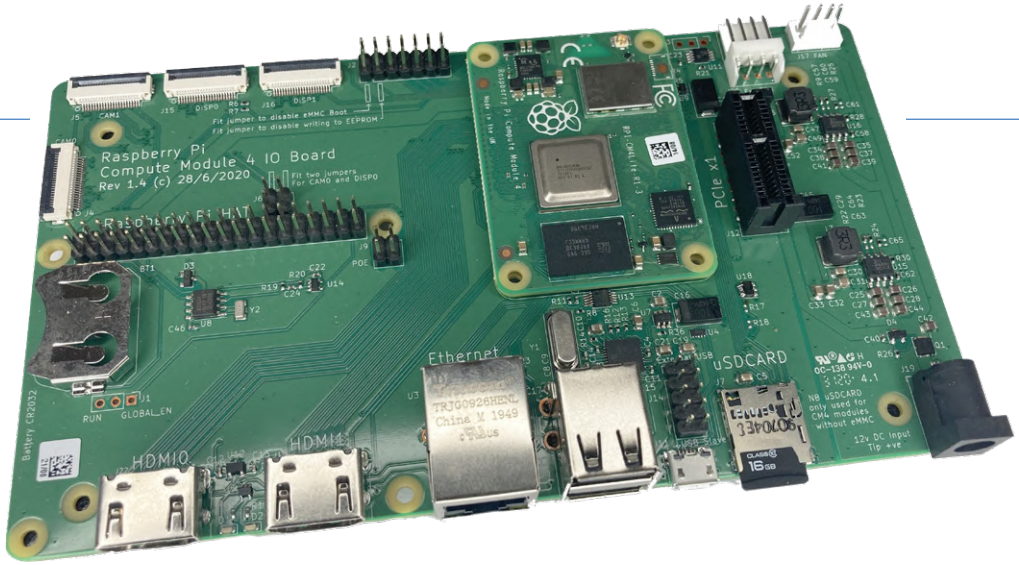


Bild 8. I/O-Board.

Bild 9. CSI-Ports.

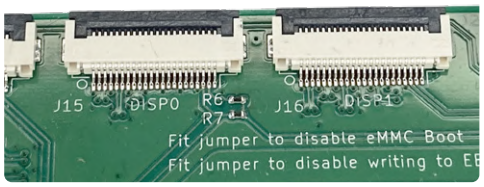


Bild 10. DSI-Ports.

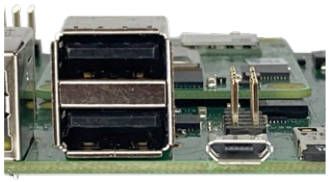


Bild 11. USB-Anschlüsse.

I/O-Board für das CM4

Für erste Experimente mit dem CM4 hat die Raspberry Pi Foundation auch ein I/O-Board (**Bild 8**) entwickelt, das alle schönen Eigenschaften des CM4 zugänglich macht und auch einen PCIe-1x-Steckplatz beherbergt. Das Board bietet mehr Konnektivität als die anderen Raspberry-Pi-4-Produkte. Dazu gehören zwei Kameraeingänge (CSI-2-Schnittstelle mit zwei oder vier Kanälen, siehe **Bild 9**) und zwei Display-Anschlüsse (DSI mit zwei oder vier Kanälen, siehe **Bild 10**). Mehrere Monitore und computergestützte Bildverarbeitung sind also kein Problem. Möchten Sie eine Maus und eine Tastatur anschließen? Der interne USB-2.0-Anschluss des CM4 wurde durch einen USB 2.0-Hub auf dem I/O-Board ergänzt, sodass damit vier USB-2.0-Anschlüsse zur Verfügung stehen (**Bild 11**). Das I/O-Board enthält auch ein RTC-Modul, das als Watchdog oder zum Booten des Systems in vorgegebenen Intervallen verwendet werden kann.

Für die Stromversorgung benötigt das Board nur ein ausreichend dimensioniertes 12-V-Netzteil, wenn man PCIe-Karten verwendet. Wenn der PCIe-Steckplatz nicht verwendet wird, kann das Board mit bis zu 26 V versorgt werden. Eine willkommene Entscheidung betrifft die beiden HDMI-Ports normaler Größe auf dem I/O-Board. Man kann jetzt einfach die üblichen HDMI-Kabel verwenden (**Bild 12**). Schließlich gibt es auch noch einen LAN-Port und einen Steckplatz für microSD-Karten. Was das I/O-Board besonders interessant macht, sind die frei zugänglichen KiCad-Dateien. Sie erlauben einen schnellen Start in eigene Projekte, wenn man KiCad verwendet. Das CM4 wird hier einfach als „Bauteil" hinzugefügt und kann so ohne eigene umständliche Designarbeit in KiCad zu eigenen Boards und Projekten hinzugefügt werden (**Bild 13**). Nur ein 3D-Modell fehlt.

Spezifikationen

- Broadcom BCM2711 Quad-Core Cortex-A72 (ARM v8) 64-Bit-SoC @ 1,5 GHz
- 1, 2, 4 oder 8 GB LPDDR4 (je nach Variante)
- Optionales WLAN mit 2,4 und 5,0 GHz nach IEEE 802.11b/g/n/ac
- Bluetooth 5.0, BLE mit Optionen für integrierte und externe Antennen
- Integrierter Gigabit Ethernet PHY nach IEEE1588
- 1 × USB 2. 0-Schnittstelle
- PCIe Gen 2 x1-Schnittstelle
- 28 GPIO-Signale
- Interface für eine SD-Karte oder externe eMMC (nur zur Verwendung mit CM4 ohne eMMC)
- Dual-HDMI-Schnittstelle (bis zu 4Kp60)
- 2-Lane MIPI DSI-Display-Schnittstelle
- 2-Lane MIPI CSI-Kamera-Schnittstelle
- 4-Lane MIPI DSI-Display-Schnittstelle
- 4-Lane MIPI CSI-Kamera-Schnittstelle
- Multimedia: H.265 (4Kp60-Decodierung); H.264 (1080p60-Decodierung, 1080p30-Codierung)
- OpenGL ES 3.0-Grafik
- Versorgungsspannung: 5 V DC
- Betriebstemperatur: -20 bis +85°C
- Produktverfügbarkeit: Das Raspberry Pi Compute Module 4 wird mindestens bis Januar 2028 in Produktion bleiben.

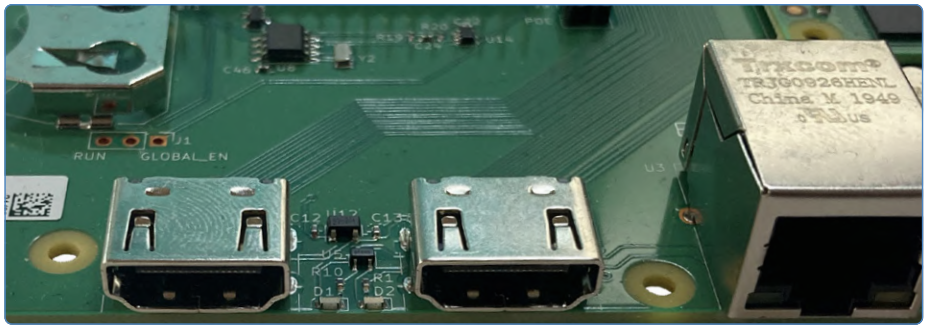


Bild 12. HDMI-Anschlüsse.

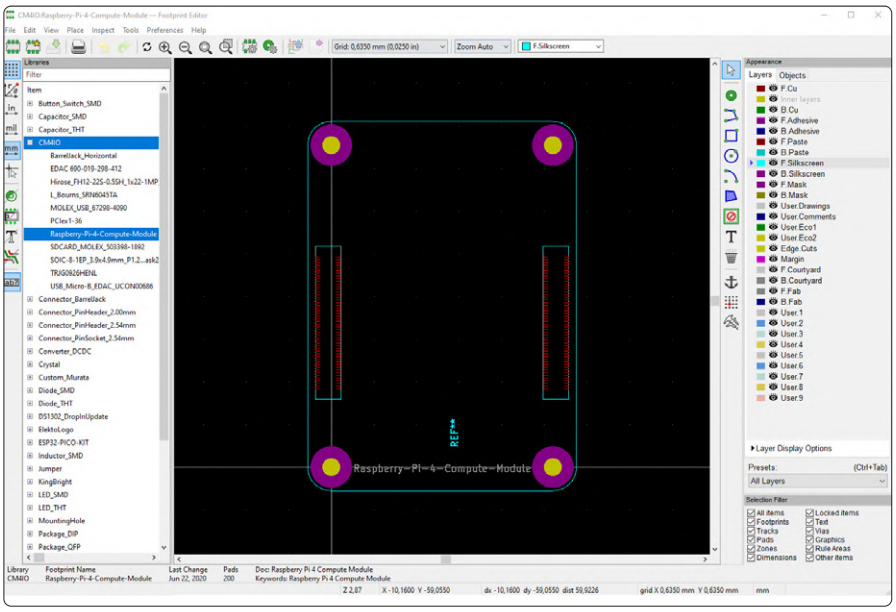


Bild 13. Screenshot des CM4-Layout-Symbols bei KiCad.

Teile kombinieren

Wenn Sie das CM4 einsetzen wollen, ist das I/O-Board eine tolle Ergänzung, da es alle Schnittstellen auf eine angenehme und brauchbare Art zugänglich macht und zudem die Nutzung eines PCIe-Steckplatzes ermöglicht. Der Einstieg ist so einfach wie bei einem gewöhnlichen Raspberry Pi 4: Man schiebe ein geeignetes OS-Image auf eine SD-Karte und boote von dieser. Wenn eine Maus und eine Tastatur angeschlossen sind, werden sie – anders als beim Raspberry Pi 4 – weder angezeigt noch funktionieren sie. Wenn man USB-Ports für Peripherie verwenden möchte, muss man eine Zeile in der Datei *config.txt* hinzufügen, die sich in der BOOT-Partition der SD-Karte befindet. Fügt man am Ende von *config.txt* die Zeile `dtoverlay=dwc2,dr_mode=host` hinzu, dann werden die USB-Ports beim nächsten Booten aktiviert.

Dank des PCIe-1x-Steckplatzes ist die Verwendung entsprechender Karten kein Problem – man muss lediglich daran denken, dass diese Karten eine Versorgung mit 12 V und ausreichender Belastbarkeit benötigen. Die Spezifikation von PCI Express erlaubt den Betrieb von Karten, die für 4x-, 8x- oder 16x-Steckplätze ausgelegt sind, auch mit weniger Lanes. Prinzipiell sollten sie daher auch in 1x-Steckplätzen funktionieren. Aufgrund mechanischer Einschränkungen kann man aber eine solche Karte nicht direkt auf das I/O-Board stecken.

Interessanterweise ist dieses Problem beim Hype um GPU-basiertes Crypto Mining auch bei modernen PC-Systemen aufgetreten. Folglich wurden schon passende Lösungen entwickelt. Eine geeignete Adapterkarte mit eigener Stromversorgung kann für weniger als 10 $ erworben (**Bild 14**) und an die I/O-Karte angeschlossen werden. Nachdem das CM4 gebootet hat und Linux läuft, ist mit `lspci` ein kurzer Blick auf die erkannten PCIe-Karten möglich. Wie man in **Bild 15** sehen kann, ist eine Intel Dual-Port-Netzwerkkarte angeschlossen. Hierfür gibt es einen funktionierenden Linux-Treiber und der sollte dem CM4 zusätzliche Netzwerkports hinzufügen. Die Ausgabe von `lspci` sieht zunächst vielversprechend aus: Die Karte wird erkannt, aber es ist kein Treiber geladen. Der Grund dafür ist ganz einfach: Der Standard-Kernel des

Passende Produkte

- **Raspberry Pi Compute Module 4 (CM4)**
  www.elektor.de/raspberry-pi-compute-module-4-cm4
- **Raspberry Pi Compute Module 4 IO-Board**
  www.elektor.de/raspberry-pi-compute-module-4-io-board
- **Raspberry Pi Compute Module 4 Antennen-Kit**
  www.elektor.de/raspberry-pi-compute-module-4-antenna-kit

Bild 14. PCIe-Adapterkarte.

Bild 15. CM4 + I/O-Board + weitere Komponenten.

Raspberry Pi OS hat nur Treiber eingebaut, die für den täglichen Gebrauch benötigt werden. Fragen Sie sich, ob man auf diese Weise auch eine PCIe-Grafikkarte anschließen könnte? Der Anschluss ist zwar möglich, aber die Sache wird trotzdem nicht funktionieren, wie Sie im Video unter [1] sehen können. Damit ein Ethernet-Controller und andere PCIe-Geräte wie etwa SATA-Karten funktionieren, muss den RPI4-Kernel neu kompiliert werden [2]. Da es sich „nur" um ein Software-Problem handelt, ist es prinzipiell lösbar. Leider ist es zurzeit noch nicht möglich, einen neuen Kernel für den Basistreiber zu erstellen. Software, Kernel-Builds und Treiber sind ein separates Thema, das in der Zukunft behandelt werden soll. Außerdem spielen einige PCIe-Geräte überhaupt nicht mit dem CM4 zusammen, da der Treiber von etwas ausgeht, das nur in der X86/AMD64-Welt zu finden ist – ganz wie derzeit beim Treiber für Grafikkarten.

Auf zu Neuem

Das Compute Module kommt jetzt in einem neuen Formfaktor und das macht das Layout eigener Basis-Boards einfach. Die frei zugänglichen KiCad-Symbole und Schaltpläne für das I/O-Board vereinfachen die Produktentwicklung. Warum nicht ein CM4 in eine Set-Top-Box oder ein selbstgebautes Handheld-Device einbauen? Mit der frei nutzbaren PCIe-Lane könnte man Ethernet-Karten mit vier Ports hinzufügen, um insgesamt fünf Gigabit-Schnittstellen bereitzustellen. Das klingt ganz nach dem Rezept für einen CM4-basierten Router, oder etwa nicht?
Angesichts günstiger Preise für das CM4 und das I/O-Board ist der Einstieg nicht allzu teuer. Wenn Sie eigene RPi-basierte Projekte entwickeln möchten, ist das mit der aktuellen CM-Generation einfacher denn je.

200590-02

Ein Beitrag von

Review und Text: Mathias Claußen
Redaktion: Jens Nickel
Übersetzung: Dr. Thomas Scherer
Layout: Giel Dols

Sie haben Fragen oder Kommentare?

Bei technischen Fragen können Sie sich gerne an die Elektor-Redaktion wenden unter der E-Mail-Adresse: redaktion@elektor.de.

WEBLINKS

[1] **GPU auf dem Raspberry Pi**: www.youtube.com/watch?v=ikpgZu6kLKE
[2] **Quad port NIC auf dem Raspberry Pi 4**: www.youtube.com/watch?v=KL0d68j3aJM

Portables autonomes Feinstaubmessgerät

Für 2,5-µm-Partikel

Behalten Sie Ihre Gesundheit im Auge

Von **Laurent Labbe** (Frankreich)

In dieser Zeit der COVID-19-Pandemie könnte man fast vergessen, dass es noch andere Gründe gibt, einen Mundschutz zu tragen. Und dass es Orte gibt, an denen dies schon viel länger ratsam, wenn nicht sogar notwendig ist. Wenn man in Asien in Großstädten lebt, ist Sport im Freien aufgrund der Luftverschmutzung oft problematisch, insbesondere wegen der lungengängigen 2,5-µm-Partikel, auch PM2,5 genannt. Hohe Konzentrationen dieser Partikel in der Atemluft können schädlich und sogar gefährlich für Ihre Gesundheit sein. Das in diesem Artikel vorgestellte PM2,5-Messgerät kann zur Überwachung der Luftqualität eingesetzt werden.

Leider ist die Luftverschmutzung ein Problem, nicht nur in den Millionenmetropolen in Fernost. Die Umweltämter der großen Städte in Deutschland wissen ein Lied davon zu singen. Da ist es gut, eine $PM_{2,5}$-Anzeige zur Hand zu haben, um die Luftqualität zu überwachen, aber um ehrlich zu sein: Es ist eigentlich sehr schlecht, dass wir so etwas überhaupt brauchen.

In der Vergangenheit hatte ich ein $PM_{2,5}$-Messgerät entworfen, das mit einem Solarpanel geladen werden konnte, aber nicht für den Einsatz in Innenräumen geeignet war. Ich hatte auch eine ESP32-basierte Version für verschiedene Sensortypen aufgebaut, deren Daten über WLAN an Thingspeak gesendet wurden, aber diese war nicht portabel. Das logische Ziel dieses Projekts ist es also, ein möglichst kompaktes Gerät zur Anzeige des $PM_{2,5}$-Index der Umgebungsluft zu bauen, dessen Batterie mehrere Wochen permanenten Betriebs durchhalten würde. Eine solches portables Gerät könnte man in Innenräumen verwenden, aber auch zum Beispiel mit einem Saugnapf außen an ein Fenster befestigen, so dass der $PM_{2,5}$-Wert von innen sichtbar ist.

Konstruktive Überlegungen

Die Wahl des Staubpartikel-Sensors ist sehr wichtig. In einem früheren Projekt hatte ich verschiedene Sensortypen getestet, für die Outdoor-Version zum Beispiel den Sharp GP2Y10. Für einen kompakten Entwurf scheint von all diesen Sensoren nur der Sensor PMS7003 von Plantower [10] brauchbar zu sein. Er verfügt über eine serielle Schnittstelle zum Senden von Messdaten. Wenn Sie diesen Sensor bestellen, achten Sie darauf, dass Sie ihn inklusive des passenden Steckers kaufen! Da der Sensor bis zu 100 mA bei 5 V zieht (hauptsächlich wegen seines internen Mini-Lüfters), kann er nicht im Dauerbetrieb arbeiten, sonst wären die Batterien in kürzester Zeit leer. Die Versorgungsspannung wird von einer Lithium-Batterie geliefert, daher ist ein DC/DC-Aufwärtswandler (3,8 V auf 5 V) mit einer „echten" Abschaltung erforderlich.

Zur Anzeige habe ich ein 8x2-zeiliges monochromes alphanumerisches LCD gewählt. Es benötigt weniger als 1 mA bei permanenter Anzeige. Ein Farb- oder OLED-Display kann aufgrund des hohen Strombedarfs nicht verwendet werden; die Autonomie des Geräts wäre ernsthaft beeinträchtigt. Für einen schnellen Überblick über den $PM_{2,5}$-Index habe ich einen LED-Bargraph hinzugefügt. Der Indexwert wird durch eine leuchtende LED dargestellt: Sechs Bereiche von $PM_{2,5}$-Quailitätsstufen mit fünf LEDs entsprechend den in **Tabelle 1** gezeigten

Air Quality Index (AQI Values)	Levels of Health Concern	Colors
0 to 50	Good	Green
51 to 100	Moderate	Yellow
101 to 150	Unhealthy for Sensitive Groups	Orange
151 to 200	Unhealthy	Red
201 to 300	Very Unhealthy	Purple
301 to 500	Hazerdous	Maroon

Tabelle 1. Die AQI-Indizes in Zahl, Wort und Farbe.

Farben, wobei mangels kastanienbrauner LEDs bei der schlechtesten Stufe die rote und die blaue LED gemeinsam leuchten. Für alle LEDs sollten lichtstarke Low-Current-Typen zum Beispiel aus der Kingbright-WP71xx-Reihe mit einem Durchlassstrom von 2 mA gewählt werden.

Die (optionale) Anzeige von Temperatur und Luftfeuchtigkeit ist ebenfalls möglich, wenn ein I²C-Sensor SHT20/SHT21 eingesetzt wird. Die Software unterstützt auch den 1-Wire-Temperatursensor DS1820, der jedoch in dieser Anwendung nicht getestet wurde. Das Herzstück des Messgeräts ist ein PIC18F2520, dem ein MAX931 zur Batteriespannungsüberwachung und ein Lithium-Batterielader LTC4054 zur Seite gestellt werden.

Die Gesamtleistungsaufnahme des Messgeräts hängt hauptsächlich vom Intervall zwischen den Messungen ab. Nach der Spezifikation des PMS7003 muss er 30 Sekunden lang eingeschaltet sein, bevor die erste zuverlässige Messung durchgeführt werden kann. Standardmäßig ist das Messintervall auf 20 Minuten eingestellt, es kann aber über das Menü geändert werden. Mit dieser Standardeinstellung und einem 1000-mAh-Akku hält das Messgerät etwa zwei Wochen lang ohne Nachladen durch. Die Größe des Akkus hängt vom Gehäuse ab. Es kann jeder Lithium-, LiPo- oder prismatische Akku verwendet werden. Da ich ein kompaktes Gehäuse für dieses Projekt selbst entwickelt habe, kann es natürlich perfekt an die Abmessungen des gewünschten Akkus angepasst werden. Bei meinem Messgerät spendet ein 1100-mAh-Akku der Elektronik seine Energie.

Berechnung des Luftqualitätsindex

Es gibt mehrere, zum Teil nicht triviale Methoden zur Berechnung des Luftqualitätsindexes. In diesem Projekt werden zwei Versionen implementiert. Die erste ist eine arithmetische Mittelwertberechnung über die letzten 24 Stunden nach der Berechnungsmethode der USA. Details zu dieser Methode, im Folgenden IQA genannt, finden Sie unter [1]. Der Nachteil dieser Methode ist, dass sie einen gleitenden Mittelwert über 24 Stunden verwendet, so dass der angezeigte Wert durch relativ alte Luftqualitätswerte beeinflusst wird.

Aus diesem Grund wird eine neue US-Berechnungsmethode (NowCast) mit einem gewichteten Mittelwerte verwendet. Kurz gesagt, hat die Messung der letzten Stunde in der endgültigen Berechnung ein größeres Gewicht als die Messung von vor 11 Stunden. Es handelt sich also ebenfalls um einen gleitenden Mittelwert, jedoch über 12 Stunden. Dies wird hier NQI (New Quality Index) genannt, hat aber die gleichen Indexgrenzen (**Tabelle 1**) wie die alte Version. Diese Methode wird bei [2] detailliert erläutert. Der Einfachheit halber wird in der Software dem Indexwert der vorherigen Stunde ein Gewicht von 50 %, dem der Stunde -2 ein Gewicht von 25 %, dem der Stunde -3 ein Gewicht von 12,5 % und so weiter zugemessen. Eine praktische Implementierung des Algorithmus wird in Link [3] diskutiert. In unserem Projekt können die Messwerte nach beiden Methoden berechnet und angezeigt werden, wobei die neue Berechnungsversion NQI standardmäßig eingestellt ist.

Die 2,5-µm-Partikelmenge wird von unserem Messgerät in regelmäßigen Abständen gemessen. Das Intervall kann im Menü zwischen 10 Minuten und 60 Minuten eingestellt werden, wobei die voreinstellten 20 Minuten einen guten Kompromiss zwischen Messgenauigkeit und Stromverbrauch darstellen. Jede Stunde berechnet das Programm den Durchschnitt der letzten Messungen, nur dieser Durchschnitt wird in einer Tabelle gespeichert und später nach der gewählten Methode bewertet. Der Index der Partikelmessung auf der LED-Leiste kann angezeigt werden als:

- IQA oder NQI
- jüngste Messung (Last)
- letzter stündlicher Durchschnittswert (Hour)

Die LEDs können während der Nacht abgeschaltet werden, um Energie zu sparen. Ein LDR misst die Umgebungshelligkeit und wechselt entsprechend den Modus (Tag/Nacht).

Eine interessante Option ist die Anzeige des aktuellen $PM_{2,5}$-Pegels in Echtzeit. Der Sensor ist dabei permanent aktiv und der angezeigte Wert wird jede Sekunde aktualisiert. Diese Option benötigt natürlich viel Energie und entleert die Batterie in kurzer Zeit.

Andere Werte, die angezeigt werden können:

- Temperatur- und Luftfeuchtigkeitsanzeige
- 3-stellige Batteriespannungsanzeige („384" zum Beispiel bedeutet 3,84 V).

Bei zu niedriger Batteriespannung blinkt die LED-Anzeige und zeigt damit an, dass ein Aufladen erforderlich ist. Für den Außeneinsatz ist ein Anschluss für eine optionale Ladung über eine 5-V-Solarzelle vorgesehen. Die Menüeinstellungen werden im internen EEPROM des Mikrocontrollers gespeichert. Alle anderen Werte, Tabellen und Variablen werden zurückgesetzt, wenn das Gerät ausgeschaltet oder neu gestartet wird.

Die Hardware

Die Schaltung des Luftqualitäts-Messgerät in **Bild 1** zeigt einen DC/DC-Aufwärtswandler LT3525ESC6-5 (U6) für den Sensor PMS7003. Dieser Wandler garantiert mit einer „True-Shutdown"-Funktion niedrigste Stromaufnahme. Die meisten Gleichspannungswandler haben zwar eine Shutdown-Funktion, die aber lediglich den Schalttransistor an der Induktivität stilllegen. Die Ausgangsspannung folgt dann (über Induktivität und Diode) aber immer noch dem Eingang, so dass weiterhin Strom fließt. Echte Shutdown-Schaltungen schalten den Wandler wirklich komplett ab: 0,00 mA! Das einzige wirkliche Problem mit diesem IC ist, es zu verlöten. Es ist wirklich sehr klein, Lötpaste und eine Heißluft-Lötstation werden empfohlen …

Der 3,3-V-Regler LP2980 (U7) wird von U1, einem Komparator mit Hysterese namens MAX931 gesteuert. Er schaltet den Regler ab, wenn die Batteriespannung unter 3,2 V fällt

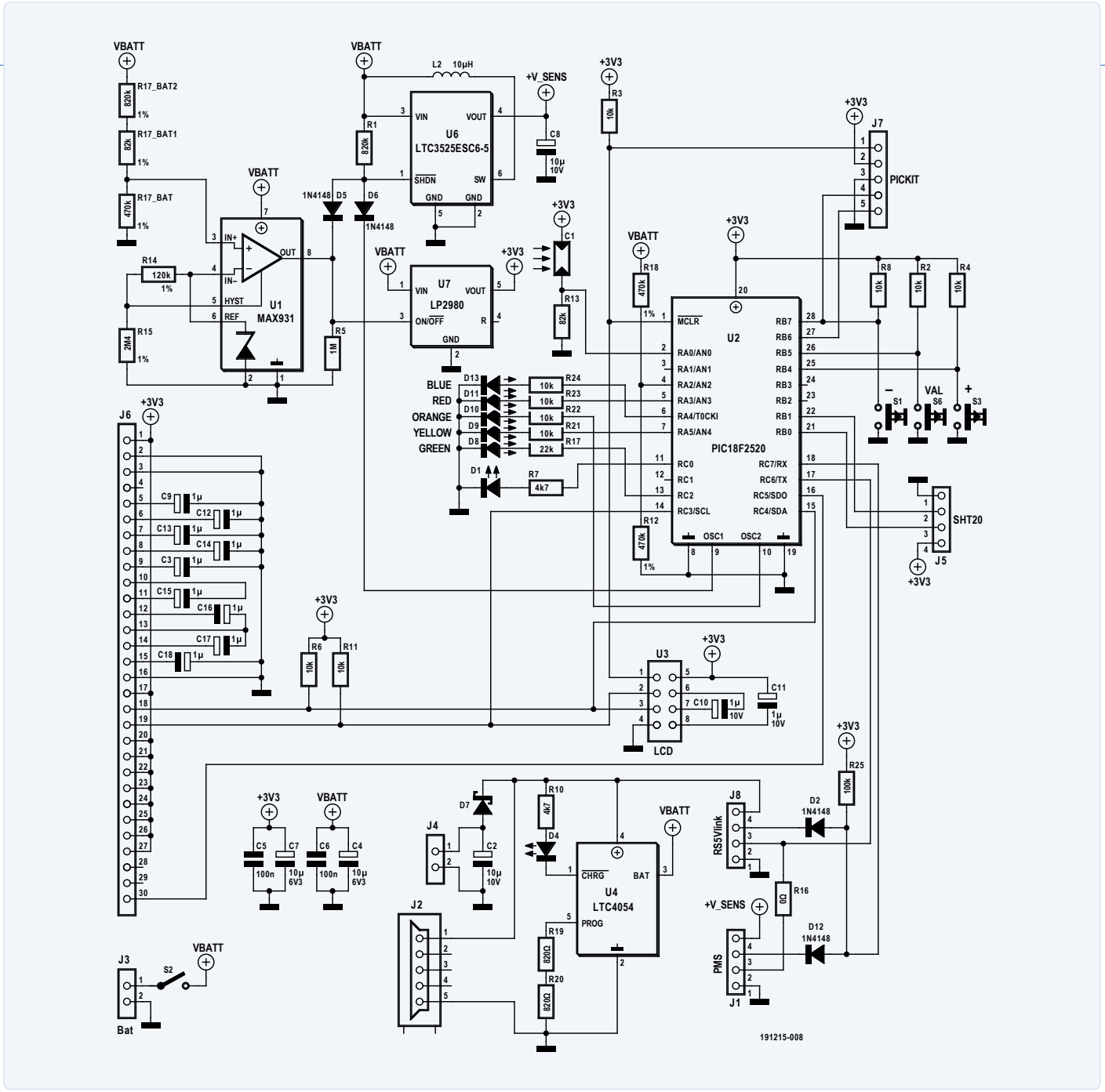


Bild 1. Die Hardware des $PM_{2,5}$-Messgeräts.

und kann erst wieder aktiviert werden, wenn diese Spannung über 3,8 V ansteigt. Diese Funktion ist im Normalbetrieb nicht unbedingt sinnvoll, aber notwendig, wenn ein Solarpanel zum Aufladen verwendet wird. Der MAX931 sperrt außerdem den DC/DC-Wandler über die Diode D5, um ungewollte Starts bei niedriger Batteriespannung zu verhindern.

Die serielle Schnittstelle wird auf die Verbinder J8 (für Tracking und Firmware-Upload) und J1 (für den PMS7003) aufgeteilt. Das Tx-Signal vom Mikrocontroller ist an beiden Verbindern angeschlossen, während die Rx-Signale ein Wired-or-Gatter (D2, D12 und R25) durchlaufen und zum Controller gelangen. Da die aktuelle Software keine Befehle an den PMS-Sensor sendet, ist der 0-Ω-Widerstand R16 nicht bestückt.

Für den SHT20/21-Sensor wird eine I²C-Schnittstelle per Software über portB.0 und portB.1 simuliert; die Hardware-I²C-Schnittstelle des PICs bleibt der LC-Anzeige vorbehalten. Das hier verwendete LCD, ein „Eastrising Serial COG 8x2 LCD Module I2C Character Display“ [7] mit I²C-Schnittstelle wird über U3 angeschlossen.

Die Ladeschaltung für den Lithium-Akku ist mit einem LTC4054 (oder dem kompatiblen EUP8054, U4) aufgebaut. Die Eingangsspannung erhält dieses IC entweder vom USB-Anschluss (J2) oder von einem externen Solarpanel an J4. Der Ladestrom wird durch die beiden Widerstände R19 und R20 auf 400 mA begrenzt. LED D4 in der Nähe des USB-Anschlusses zeigt an, dass der Akku geladen wird. Ein eventuelles 5-V-Solar-Panel an J4 lädt den Akku über die Schottky-Diode D7 auf.

Die fünf Bargraph-LEDs (D8...D11 und D13) werden durch direkt vom PIC-Controller gesteuert. Der Strombegrenzungswiderstand jeder LED hängt vom verwendeten LED-Typ (Low-Power oder nicht) und von der Durchlassspannung der einzelnen Farb-LED ab. Die rote LED D1 leuchtet, wenn der DC/

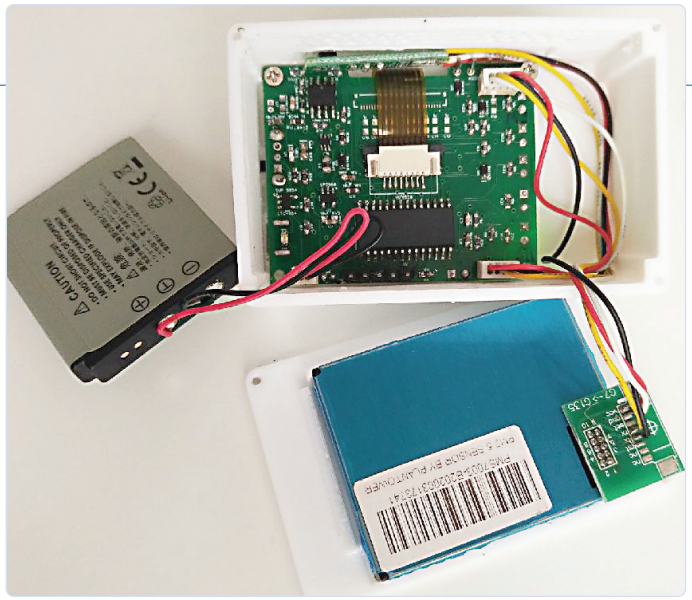


Bild 2. Platine, Batterie und PMS7003-Sensor bei geöffnetem Gehäuse.

Bild 3. Vorderansicht des $PM_{2,5}$-Messgeräts im 3D-gedruckten Gehäuse.

DC-Wandler arbeitet und zeigt an, dass das Messgerät eingeschaltet ist.

Für C10 und C11 am Display können Kondensatoren zwischen 1 µF und 2,2 µF mit einer Spannungsfestigkeit von mindestens 10 V verwendet werden. Die auf der Platine des Displays aufgeklebte Hintergrundbeleuchtung wird nicht verwendet und kann vor der Montage entfernt werden. Es wird empfohlen, das Display zusammen mit seinem Stecker zu kaufen.

Fehlt noch der „Tageslichtsensor“: Dafür wird ein simpler LDR (C1) verwendet, der aus der Frontplatte herausguckt. Er ist mit dem Analogeingang AN0 des Mikrocontrollers verbunden.

Bestückung der Platine

Der Schaltplan und die Platine wurden mit der CAD-Software DipTrace von Novarm entworfen, die Designdateien können von der Elektor Labs-Projektseite [9] heruntergeladen werden. Alle Bauteile außer den LEDs sind SMDs. Das am schwierigsten zu lötende Teil ist der DC/DC-Wandler, ich habe dafür Lötpaste und eine Heißluft-Lötstation verwendet. Aufgrund der Kompaktheit des Gehäuses ist es schwierig, Standardstecker zum Anschluss der zwei Sensoren zu montieren, so dass man besser die Anschlussdrähte direkt an die Schaltung lötet. Nur die Batterie und das Display werden mit Steckverbindern angeschlossen.

Zur Programmierung des Mikrocontrollers gibt es zwei Möglichkeiten. Entweder schließen Sie ein PicKit3 an J7 an oder Sie verwenden den seriellen Anschluss J8. Im zweiten Fall muss der Mikrocontroller vor dem Einlöten mit einem Bootloader programmiert werden. Zudem muss bei jedem Programmiervorgang des Controllers der PMS7003 abgezogen werden. Ich persönlich verwende immer einen Bootloader namens *Tiny Bootloader* [6]. Es ist notwendig, dass die im PIC eingebettete Bootloader-Version den richtigen Wert für den Watchdog kennt (hier 256 ms) und daher muss der Sourcecode damit neu kompiliert werden. Die .asm- und .hex-Dateien dieses Bootloaders für den PIC18F2520 sind auch Teil des Software-Downloads auf der Elektor-Labs-Projektseite [9].

Software

Die Software wurde mit einer älteren Version des MikroC-Compilers in C entwickelt. Sie kann wohl auf die neuere Version MikroC Pro portiert werden, aber ich habe es nicht ausprobiert. Der Quelltext und die HEX-Datei können von der Seite dieses Projekts auf Elektor Labs [9] heruntergeladen werden. Die Firmware ist recht einfach:

- Initialisierung
- Endlosschleife mit Tastenabfrage
- Anzeige der verschiedenen Displays, der Pegel der LEDs (je nach LDR-Wert für Tag/Nacht-Umschaltung)
- Alle *xx* Minuten Einschalten des Gleichspannungswandlers für den Partikelsensor. Nach 30 s erfolgt das Auslesen des $PM_{2,5}$-Werts. Der Wandler wird wieder abgeschaltet.
- Jede Stunde Berechnung des Mittelwerts
- Speicherung dieses Wertes in einer 24-reihigen gleitenden Durchschnittstabelle (für 24 Stunden)
- Berechnung des IQA (gleitender 24-Stunden-Mittelwert) oder NQI (gleitender gewichteter 12-Stunden-Mittelwert)

Die Funktion `calculation_aqi()` berechnet den Wert des Luftqualitätsindexes nach dem Prinzip des einfachen Mittelwertes über 24 Stunden. Die letzten 24 Werte in der Tabelle werden dazu einfach gemittelt. Die Funktion `calculation_nqi()` berechnet den gewichteten Wert des Luftqualitätsindex. Die „praktische“ Formel wird in Referenz [3] erklärt.

Die Umrechnung dieser Werte in die Indizes (0…500) erfolgt durch die Funktion `conversion_aqi()`, entsprechend der im Dokument [1] für die USA angegebenen Formel.

Die Hauptschleife der Software wird durch eine `sleep`-Anweisung beendet. Der Controller erwacht, wenn ein Watchdog ausgelöst wird, das Watchdog-Intervall wird auf 256 ms eingestellt (im Falle des PIC18F2520 muss dies bei der Programmierung des Mikrocontrollers geschehen). Dieses Intervall ist kurz genug, um keine Tastendrücke zu verpassen. Eine längere Zeit wäre besser, was den Stromverbrauch angeht, aber dann könnte man auch Tastendrücke verpassen. Die Software funktioniert auch ohne den angeschlossenen Temperatur-/Feuchtesensor, aber natürlich zeigt das Display dann ungültige Werte für Temperatur und Feuchte an.

Gehäuse aus dem Drucker

Das Gehäuse wurde in ThinkerCad entworfen und mit einem 3D-Drucker gedruckt. Es ist recht einfach, mit Öffnungen für den LDR und den SHT-Sensor auf der Oberseite sowie für den PMS7003-Sensor an der Seite, eine für den Lufteinlass, eine für den Auslass. Die aktuelle Version des Gehäuseentwurfs kann von [4] heruntergeladen werden, ebenso wie eine Halterung für eine Solarzelle [5],

BEDIENUNG

Durch einen langen Druck auf die VAL-Taste (S6) gelangt man in das Konfigurationsmenü, das dem Benutzer folgende Möglichkeiten bietet:

- Ändern der Messperiode (10...60 min) mit den Tasten + und -, bestätigen durch Drücken der VAL-Taste
- Aktivieren der LED-Pegelanzeige während der Nacht (ja/nein)
- Wählen, welcher Wert die LEDs anzeigen sollen (letzte Messung, AQI/NQI, Durchschnitt der letzten Stunde)
- Wahl des Indextyps, der angezeigt werden soll. (AQI/NQI)

Durch Drücken der Taste VAL können Sie durch die verschiedenen Menüs blättern. Auf dem LCD gibt das Zeichen vor dem Wert an, was aktuell angezeigt wird:

L: Letzte, jüngste Messung
H: Hour, der aktuelle stündliche Durchschnittswert
Q: Qualität, Luftqualitätsindex, IQA oder NQI (je nach Menüeinstellung)
T: Temperatur (in Grad Celsius)
H: Luftfeuchtigkeit (in %)
I: Momentanwert, Anzeige wird jede Sekunde aktualisiert

Der LED-Bargraph zeigt den im Menü festgelegten Wert an, außer bei der Anzeige des Sofortwerts. In diesem Fall zeigt der Bargraph den Index des letzten vom Sensor ermittelten Wertes an, der jede Sekunde aufgefrischt wird.
Im Momentanwert-Modus wird in der zweiten Zeile der Wert der vom Sensor zurückgegebenen Partikelanzahl ausgegeben. Es folgt der berechnete Luftqualitätsindexwert, dem der Buchstabe Q vorangestellt ist. Eine Anzeige zeigt auch den minimalen und maximalen Wert des Indexes während der letzten 24 Stunden an.

die wird mit kleinen Haken an der Rückseite des Gehäuses befestigt wird
Die kompakte, batteriebetriebene Hardware funktioniert hervorragend, aber Hard- und Software können immer noch erweitert und/oder verbessert werden. So gibt es im Schaltplan und auf der Platine bereits einen Anschluss (J6) für ein Low-Power-Grafik-LCD mit 128x64 Pixeln [8], aber dies ist nicht getestet und wird in der aktuellen Version der Firmware auch nicht unterstützt. Hat man dies nicht vor, kann man J6 samt der Vielzahl von Kondensatoren in seiner Nähe einfach weggelassen.

191215-03

Ein Beitrag von
Idee, Entwurf, Text und Illustrationen: **Laurent Labbe**
Schaltplan: **Patrick Wielders**
Redaktion: **Luc Lemmens, CJ Abate**
Übersetzung: **Rolf Gerstendorf**
Layout: **Giel Dols**

Haben Sie Fragen oder Kommentare?
Haben Sie Fragen oder Kommentare zu diesem Artikel? Schicken Sie eine E-Mail an den Autor unter laurent.elektor@gmail.com oder kontaktieren Sie Elektor unter editor@elektor.com.

WEBLINKS

[1] **Luftqualitätsindex**: https://en.wikipedia.org/wiki/Air_quality_index
[2] **Neuberechnung des Luftqualitätsindex (NowCast)**: https://en.wikipedia.org/wiki/NowCast_(air_quality_index)
[3] **Praxisnahe Formel zur Berechnung des NQI**: www.epa.gov/airnow/faq/Nowcast-formula.pptx
[4] **Gehäuse**: www.tinkercad.com/things/gznm3a4WifP-new-boitier-dust-11
[5] **Solarpanel-Halterung**: www.tinkercad.com/things/b3mohI2Dzla-copy-of-copy-of-boitier-dust-6
[6] **PIC-Bootloader**: www.etc.ugal.ro/cchiculita/software/picbootloader.htm
[7] **Alphanumerisches Display**: www.buydisplay.com/character-display/character-display-panel?interface=461&resolution=135
[8] **Grafisches Display**: www.buydisplay.com/1-4-inch-graphic-128x64-lcd-module-serial-spi-st7565-black-on-white
[9] **Elektor-Labs-Seite zu diesem Projekt**: www.elektormagazine.de/labs/portable-display-pm25-1-1
[10] **Datenblatt PMS7003**: www.pdf-archive.com/2017/04/12/plantower-pms-7003-sensor-data-sheet/plantower-pms-7003-sensor-data-sheet.pdf

Aus dem Leben gegriffen

Die Zukunft war in der Vergangenheit besser

Von **Ilse Joostens** (Belgien)

Die Sonne stand hoch über dem Horizont. Es war brütend heiß und warme, schwüle Luft waberte über dem Rollfeld, auf dem die Flugzeuge entlang rollten. Andrew starrte, sichtlich müde und mit glasigem Blick, durch das Fenster der Business-Lounge auf die geschäftige Szene, während er sich den Schweiß wegwischte, der ihm auf der Stirn perlte. Als er das Papiertaschentuch wieder wegstecken wollte, erfasste sein Blick versehentlich einen japanischen Geschäftsmann, der einige Meter entfernt Platz genommen hatte. Rechts von seinem Sessel standen zwei riesige Koffer auf dem Teppich und der Mann schien in eine hitzige Diskussion mit seiner Uhr verwickelt zu sein. Wie hypnotisiert starrte Andrew den Mann weiter an, und als dieser sein Gespräch beendete und aufstehen wollte, beschloss Andrew, verwirrt und neugierig wie er war, den Mann anzusprechen.

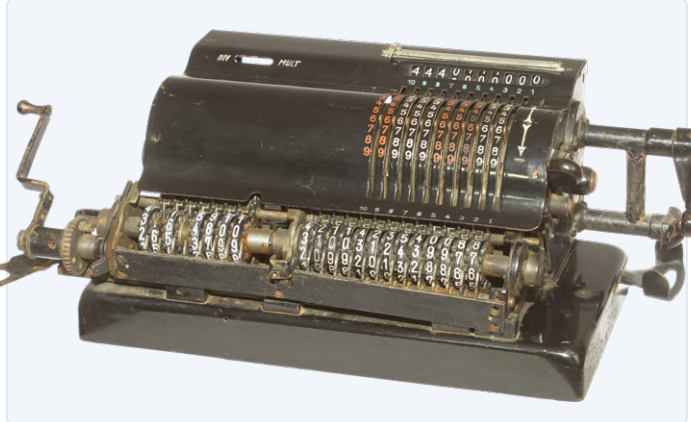
Napoleons Erfindungen

Wenn Sie sich beim Lesen der Einleitung gefragt haben, ob Sie einen Roman oder eine Elektronikzeitschrift in die Hand genommen haben, dann bitte ich Sie aufrichtig um Entschuldigung für diesen Schwall literarischen Wahnsinns. Ich wollte Ihnen nur den Witz von dem Japaner mit den zwei Koffern erzählen. Den Rest der Geschichte will ich Ihnen deshalb nicht vorenthalten, wenn auch in einem etwas vertrauteren Schreibstil.

Sagt der Amerikaner zu dem Japaner: „Was machst du da? Was ist denn das für ein Ding?“ „Das ist das allerneueste Gadget von Casio“, antwortet der Japaner. „Alles in einem, Radio-Atomuhr, Laserstift, Kamera, man kann Videotelefonate führen, ins Internet gehen, faxen, seinen Terminkalender verwalten und man kann sogar damit schwimmen, weil es wasserdicht ist.“ „Fantastisch!“, sagt der Amerikaner. „So ein Ding brauche ich auch. Wie viel kostet es?“ „Zehntausend Dollar“, sagt der Japaner. „Abgemacht“, sagt der Amerikaner, holt sein Portemonnaie heraus und bezahlt den Japaner. Zufrieden schnallt sich der Amerikaner seine Neuerwerbung um das Handgelenk und bemerkt, dass der Japaner zufrieden davonläuft. „He, Sie haben Ihre Koffer vergessen!“, ruft der Amerikaner. „Keineswegs“, sagt der Japaner. „Das sind Ihre Koffer, die gehören zur Uhr!“

Als ich diesen Witz zum ersten Mal hörte, gab es GSM-Telefone erst seit ein paar Jahren und im Internet konnte man nur quälend langsam über eine teure Wählverbindung mit einem altmodischen piepsenden und kreischenden Modem surfen. Damals konnte ich mir noch nicht vorstellen, dass das Gadget des Japaners aus dem Witz eine alltägliche Sache werden würde, zwar nicht gerade in Form einer Uhr, sondern in Form eines Smartphones. Es gibt wohl

Der mechanische Taschenrechner Curta (Foto: Ilse Joostens).



Ein mechanisches Rechenmonster.

Smartwatches, mit denen man quatschen kann, aber ich benutze lieber ein Smartphone. Fotos machen, im Internet surfen, Filme drehen, Videos anschauen, den Terminkalender verwalten, all das ist mühelos möglich und Gott sei Dank hat man die Möglichkeit, Faxe zu versenden, weggelassen. Wasserdicht sind sie auch, vor allem in Fernost, denn geben Sie es zu, wer will nicht unter der Dusche telefonieren?
Das Smartphone erinnert mich an das „Appendiscoop", ein Sender/Empfänger-Amulett, das Bilder und Töne weiterleitet, das Napoleon in der flämischen Kinderfernsehserie Merlina der 1980er Jahre erfunden hat. Auch andere Erfindungen von Napoleon sind heute mehr oder weniger Realität geworden, wie die Videobrille, den allwissenden „Evarist de Computer", das geräuschabsorbierende „Contradecibel" und den „Holomat" für die Aufnahme von 3-D-Fotos. Dinge, von denen man als Kind nicht einmal zu träumen wagte.
Eines der faszinierendsten Dinge aus dieser Zeit ist der Taschenrechner Clover 2001, den ich als Zehnjährige vom Nikolaus der Bank, bei der mein Vater arbeitete, geschenkt bekam. Das Gerät ist augenscheinlich sehr sparsam mit Energie und es muss auch sehr gute Batterien haben. Ich kann nämlich damit heute noch immer noch Berechnungen durchführen, obwohl die Anzeige sehr schwach geworden ist. Es ist einfach unglaublich, dass nach fast 40 Jahren (!) noch genügend Energie in den Originalbatterien ist.
Vor dieser Zeit benutzte man mechanische Taschenrechner. Diese waren nicht nur aufwendig und teuer, sondern oft auch sperrig und schwer. Ein bemerkenswerter mechanischer Taschenrechner (ja, den gibt es tatsächlich!) und heute ein beliebtes Sammlerobjekt ist der Curta-Rechner, der wegen seiner Form auch Pfeffermühle oder Mathe-Granate genannt wird. Die Curta, die bis in die 80er Jahre von Rallye-Piloten benutzt wurde, ist nicht nur ein verblüffendes Beispiel für Feinmechanik, der Kontrast zum Taschenrechner, den man in einer Werbeaktion von einem Versicherungsvertreter geschenkt bekommt, könnte nicht größer sein.

Hirngespinste

In den oft lustigen Vorstellungen in der Vergangenheit sah die Zukunft des Jahr 2000 viel besser aus als das, was sich in der Realität entwickelte. Aber das soll mich nicht davon abhalten, selbst ein paar Vorhersagen zu machen!
Ich erwarte, dass die Zukunft eine Revolution bei Implantaten und Prothesen bringen wird - ein bionischer Mensch, könnte man sagen.

Telefonieren unter der Dusche - wie hat die Menschheit ohne diese Fähigkeit überleben können?

Auch 3D-gedruckte Organe und weniger invasive Operationstechniken werden meiner Meinung nach eine (R)Evolution erleben. Und die künstliche Intelligenz wird eine immer größere Rolle spielen - mit dem Risiko, dass auch 1984 von George Orwell immer mehr zur Realität wird. Des Weiteren sehe ich auch eine wichtige Rolle für kleine Kernreaktoren wie den „Travelling Wave Reactor" (TWR) und vielleicht werden wir dem „Perseverance" folgen und eines Tages in einer bemannten Marsmission den roten Planeten betreten.
Was denken Sie, wird die Zukunft bringen? Polieren Sie Ihre Glaskugel und lassen Sie es uns wissen!

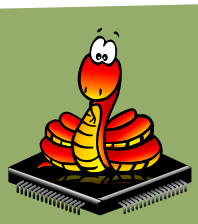
210185-03

Haben Sie technische Fragen oder Kommentare?
Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Schicken Sie eine E-Mail an den Autor oder an die Redaktion von Elektor, über redaktion@elektor.de.

Ein Beitrag von
Autor: **Ilse Joostens**
Redaktion: **Eric Bogers**
Übersetzung: **Rolf Gerstendorf**
Layout: **Giel Dols**

WEBLINKS

[1] **Der Klang einer Einwahlverbindung**: www.youtube.com/watch?v=gsNaR6FRuO0
[2] **Merlina**: https://nl.wikipedia.org/wiki/Merlina
[3] **Curta-Rechner**: www.vcalc.net/cu.htm
[4] **Curta-Rechner**: https://satadorus.eu/x_ite/yacs_2_0/yacs_2_0.html
[5] **Obsoleszenz der Technik**: https://youtu.be/zbsq4-KwnE4?t=51



MicroPython

für den ESP32 und Co.

Teil 1: Installation und erste Programme

Von **Dr. Günter Spanner** (Deutschland)

Die Sprache C als Klassiker unter den Programmiersprachen wird immer mehr von Python verdrängt. Dieser Trend setzt sich nun auch im Mikrocontrollerbereich fort. Der folgende Artikel zeigt am Beispiel des ESP32, wie man einen modernen Controller in MicroPython programmieren kann.

Benötigtes Material

1. Breadboard klein *
2. ESP32-PICO-Kit V4 *
3. LED rot
4. Widerstand 150 Ω
5. SSD1306 Display-Modul *
6. Drahtbrücken

* Im Elektor-Store erhältlich.

Python hat in den letzten Jahren einen enormen Aufschwung erlebt. Nicht zuletzt haben verschiedene Einplatinensysteme wie der Raspberry Pi zu dessen Bekanntheitsgrad beigetragen.
Aber auch in anderen Gebieten wie der künstlichen Intelligenz oder dem Machine Learning hat Python weite Verbreitung gefunden. Es ist daher naheliegend, Python beziehungsweise die Variante MicroPython auch für den Einsatz auf Mikrocontrollern zu verwenden. In diesem Artikel soll auf die Grundlagen von MicroPython eingegangen werden, unter anderem auf die wichtigsten Programmbefehle und Bibliotheken. Für ein paar kleine Demo-Anwendungen verwenden wir einen ESP32-Controller, den wir mit einer Firmware ausstatten müssen, die Python-Befehle interpretiert. Diese Befehle stellen wir mit einer Entwicklungsumgebung zusammen, die auf einem PC läuft. Die Python-Befehle können einzeln oder als ganzes Programm vom PC zum Controller geschickt werden - das funktioniert über USB, aber auch ein WLAN-Netzwerk. Doch der Reihe nach!

Programmier- und Entwicklungsumgebungen

Im Gegensatz zum Arduino-System stehen für die Arbeit mit MicroPython mehrere Entwicklungsumgebungen (IDEs: **I**ntegrated **D**eveloping **E**nvironment) zur Verfügung. Die zwei am weitesten verbreiteten Programmierumgebungen sind aktuell

1. µPyCraft
2. Thonny

Darüber hinaus kann auch der vor allem im Bereich der Künstlichen Intelligenz eingesetzte Anaconda Navigator für die Programmierung von Controllern eingesetzt werden. Jede Methode hat naturgemäß spezifische Vor- und Nachteile. So bietet die µPyCraft-IDE nur eine vergleichsweise einfache Oberfläche. Sie arbeitet mit simplen Grafikelementen und erinnert an textorientierte Betriebssysteme. Thonny dagegen verfügt über eine vollständig grafische Oberfläche im Windows-Stil (**Bild 1**). Die IDE ist bei Makern sehr beliebt,

insbesondere auch deshalb, weil sie unter dem Raspbian-Betriebssystem auf dem Raspberry Pi verfügbar ist. Viele RasPi-Anwender sind daher bereits bestens mit Thonny vertraut. Thonny steht für die wichtigsten Betriebssysteme wie Windows, Mac OS X oder Linux Ubuntu kostenlos im Internet zur Verfügung [1].
Um mit Thonny arbeiten zu können, muss Python 3 auf dem Entwicklungsrechner installiert sein. Falls dies noch nicht der Fall ist, kann die Installation über die entsprechende Webseite [2] nachgeholt werden.
Danach kann Thonny über [1] installiert werden. Hierzu wird auf der Webseite oben rechts das passende Betriebssystem ausgewählt. Nach dem Starten der heruntergeladenen exe-Datei läuft der übliche Installationsprozess ab. Anschließend steht die IDE für erste Python-Anwendungen zur Verfügung.

Installation des Interpreters

Als Nächstes wird die aktuelle MicroPython-Firmware geladen. Diese findet sich auf der offiziellen MicroPython-Webseite [3]. Hier stehen eine Reihe von Controller-Optionen zur Verfügung. In diesem Artikel soll insbesondere auf den ESP32 eingegangen werden. Deshalb muss die aktuelle stabile Version (*latest*) für diesen Controllertyp heruntergeladen werden. Dazu klickt man auf der Seite unter *Espressif ESP-based boards* auf das passende Image mit der Unterschrift *Generic ESP32 module*. Auf der sich nun öffnenden Seite stehen mehrere Firmware-Varianten zur Verfügung. Dabei ist zu beachten, dass die jeweils aktuelle Version meist mit dem Zusatz „*unstable*“ versehen ist. Diese Option ist eher für Entwickler der Interpreter-Firmware selbst oder experimentierfreudige Naturen geeignet. Möchte man mit einem stabilen System arbeiten, sollte man die erste Variante ohne den *unstable*-Zusatz wählen, zum Beispiel

GENERIC : esp32-idf3-20200902-v1.13.bin

Mit einem Klick auf den entsprechenden Link wird die Firmware heruntergeladen.
Nun kann das Controller-Board (zum Beispiel ein ESP32-Pico-Kit, siehe Kasten **Benötigtes Material**) via USB mit dem PC verbunden werden. Anschließend wird die Thonny-IDE gestartet. Hier muss als Erstes im Menü *Run* der Unterpunkt *Select interpreter* aufgerufen werden. Daraufhin öffnet sich das Fenster *Thonny options* (**Bild 2**). Auf der Registerkarte *Interpreter* können mehrere Optionen unter anderem auch für den ESP32 aufgerufen werden.
Anschließend ist unter *Port* der USB-Port auszuwählen, an welchem der ESP32 aktiv ist. Alternativ kann man die Option < *Try to detect port automatically* > selektieren. Diese arbeitet jedoch nicht in allen Fällen zuverlässig.
Unter dem Eintrag *Firmware* wird nun der Installer gestartet. Der Port wird nicht automatisch übernommen und muss nochmals eingetragen werden. Danach ist die oben heruntergeladene Firmware auszuwählen. Mit *Install* startet der Installationsprozess. Nach dem Schließen der noch offenen Fenster steht der Programmierung des ESP32 unter MicroPython nichts mehr im Weg.

Bibliotheken

Mit Python zu arbeiten heißt, Bibliotheken zu nutzen. Alle Programme von Grund auf neu zu schreiben wäre im besten Fall höchst ineffizient. Der überragende Erfolg von Python beruht zu

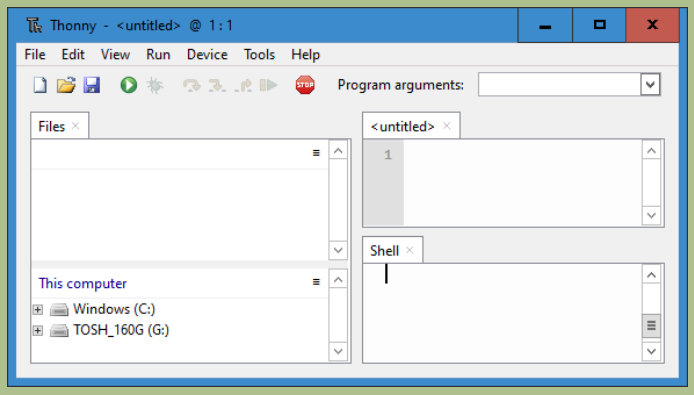


Bild 1. Die Thonny-IDE.

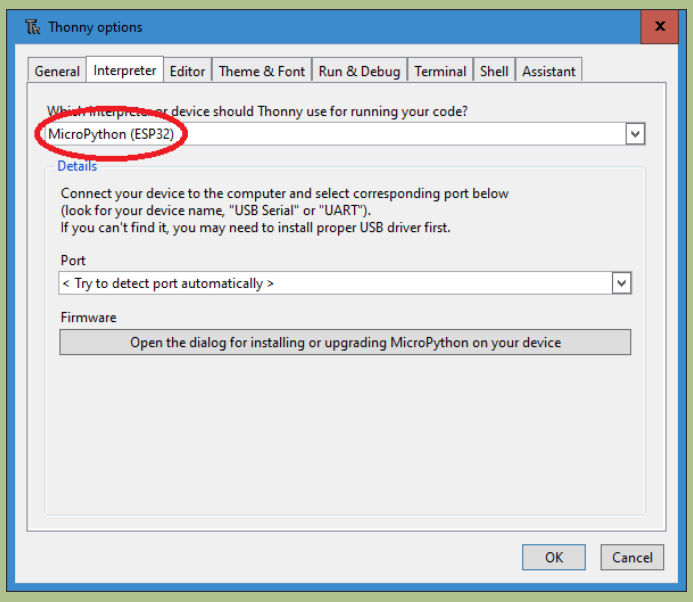


Bild 2. Thonny Options mit Firmware-Installer.

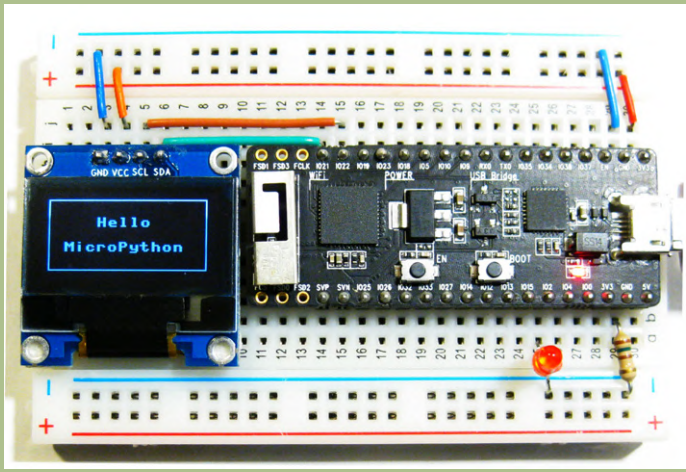


Bild 3. ESP32-Controller mit OLED-Display und LED an Pin 2.

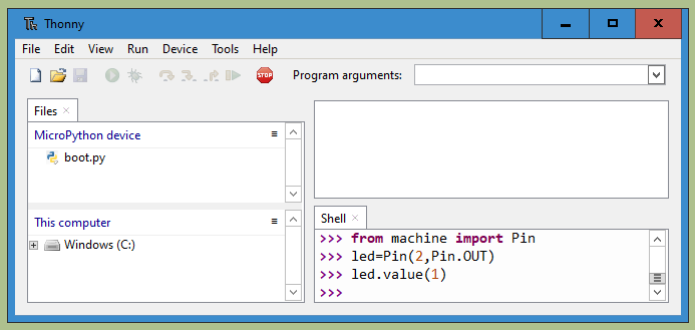


Bild 4. Schalten eines I/O-Ports über die REPL-Konsole.

einem wesentlichen Teil auf den dazu verfügbaren Bibliotheken. Leider lassen sich nicht alle Libraries mit den eingeschränkten Ressourcen eines Mikrocontrollers verwenden. Daher wurde für MicroPython eine eigene Auswahl an Libraries entwickelt. Viele davon stehen als Standard-Bibliotheken bereits mit dem Download der Thonny-IDE zur Verfügung.
Die beiden wichtigsten Standard-Libs in MicroPython sind *machine* und *time*. Über die `import`-Anweisung werden die Libs für den Controller verfügbar. Um die Bibliotheksfunktionen zugänglich zu machen, stehen unter anderem die folgenden Optionen zur Verfügung:

```
import module
from module import name
```

Im ersten Fall wird das komplette Modul eingebunden, im zweiten nur die jeweils benötigten Routinen.
Das *machine*-Modul enthält die Funktionen mit Bezug auf die Hardware eines bestimmten Controllers. Die Funktionen in diesem Modul ermöglichen also den direkten und uneingeschränkten Zugriff auf die Steuerung von Hardwareeinheiten wie CPU, Timer, Busse, I/O-Pins und so weiter. Es ist zu beachten, dass bei falscher Verwendung dieses Moduls Fehlfunktionen, Abstürze und in extremen Fällen sogar Hardwareschäden auftreten können.
Die Klasse `Pin` ist eine der wichtigsten Funktionen im *machine*-Modul. Ein Pin-Objekt wird zur Steuerung von Eingabe/Ausgabe-Pins verwendet. Pin-Objekte sind üblicherweise einem physikalischen Pin des Controllers zugeordnet. Damit lassen sich also Ausgangsspannungen steuern oder Eingangspegel einlesen.
Die Pin-Klasse verfügt über Methoden zum Einstellen des Pin-Modes, wie `IN` und `OUT`, mit welchen ein Pin als Ein- oder Ausgang definiert werden kann.

Das *time*-Modul bietet verschiedene zeitbezogene Funktionen. Die Klasse `sleep` aus diesem Modul unterbricht die Ausführung des aktuellen Programms für die angegebene Anzahl von Sekunden. Das Argument kann auch eine Gleitkommazahl sein, um eine exakte Ruhezeit bis auf Sekundenbruchteile anzugeben. Die Anweisungen

```
from machine import Pin
from time import sleep
```

stellen die beiden Funktionen `Pin` und `sleep` zur Verfügung. Damit lassen sich zum einen die einzelnen Port-Pins eines Controllers ansprechen, zum anderen wird eine einfache Zeitsteuerung möglich. Über die Anweisung

```
led = Pin(2, Pin.OUT)
```

wird ein Objekt `led` erzeugt, das dem I/O-Pin Nr. 2 zugeordnet ist und diesen als Ausgabe-Pin definiert. Diesem Objekt können nun verschiedene Werte zugeordnet werden. So wird dem Objekt über

```
led.value(1)
```

beispielsweise der Wert 1 zugeordnet. Dies bedeutet, dass der zugehörige Pin 2 nun High-Potential führt, im Falle des ESP32 also 3,3 V.

Die REPL-Konsole

Um den Zustand eines Ports sichtbar zu machen, kann man dort eine LED mit Vorwiderstand anschließen. **Bild 3** zeigt darüber hinaus auch bereits den Anschluss eines OLED-Displays, das erst

weiter unten zum Einsatz kommt.
Der Port und damit die LED können nun ganz einfach über die sogenannte REPL-Konsole gesteuert werden. REPL steht für „**R**ead **E**valuate **P**rint **L**oop“. Über diese interaktive MicroPython-Eingabeaufforderung kann direkt auf den ESP32 zugegriffen werden. Die Verwendung von REPL stellt somit eine sehr einfache Methode dar, um Anweisungen zu testen oder Befehle auszuführen.
In der Thonny-IDE wird die REPL-Konsole als *Shell* bezeichnet und steht im unteren rechten Fenster zur Verfügung. Hier können die Anweisungen aus dem letzten Abschnitt direkt eingegeben werden (**Bild 4**).
Nach der Ausführung der letzten Anweisung sollte die LED aufleuchten. Mit `led.value(0)` kann die LED wieder ausgeschaltet werden.
Die REPL-Konsole verfügt über einige interessante Features, die bei der Arbeit mit MicroPython sehr nützlich sein können. So werden etwa zuvor eingegebene Textzeilen gespeichert. Mit den Aufwärts- und Abwärtspfeiltasten können bereits früher eingegebene Zeilen bei Bedarf wieder aufgerufen werden.
Ein weiteres hilfreiches Feature ist die Tab-Ergänzung. Durch Drücken der Tabulatortaste wird das aktuell eingegebene Wort automatisch vervollständigt. Dies kann auch genutzt werden, um Informationen über Funktionen und Methoden eines Moduls oder Objekts zu erhalten.
Nach der Eingabe von „ma” und Drücken der Tabulatortaste erfolgt beispielsweise die Ergänzung zu „machine”, vorausgesetzt, das machine-Modul wurde wie oben gezeigt zuvor importiert. Wird danach der Punkt (`.`) eingegeben und die Tabulatortaste erneut gedrückt, erscheint eine vollständige Liste aller im Maschinenmodul verfügbaren Funktionen (**Bild 5**). Damit wird ein Nachschlagen von Anweisungen, Objekten oder Methoden in einer Dokumentation in den meisten Fällen überflüssig.

WLAN-Zugriff über WebREPL

Einer der großen Vorteile des ESP32 ist seine hervorragende WLAN-Konnektivität. Was liegt also näher, als die REPL-Konsole drahtlos über WLAN zur Verfügung zu stellen. Hierfür kann das WebREPL-Interface genutzt werden.
Der erste Schritt zur Verwendung von WebREPL besteht darin, sicherzustellen, dass es auf dem ESP32 verfügbar und aktiviert ist. WebREPL ist standardmäßig nicht aktiviert und muss mit einem einmaligen Befehl

```
import webrepl_setup
```

über die serielle Schnittstelle eingeschaltet werden. Es folgt eine Aufforderung, die Funktion zu aktivieren oder zu deaktivieren und ein Kennwort festzulegen. Anschließend muss ein Reboot durchgeführt werden.
Um WebREPL in einem WLAN-Netzwerk zu verwenden, ist der ESP32 zunächst mit dem Netzwerk zu verbinden. Dazu sind in der seriellen REPL die folgenden Befehle auszuführen:

```
import network
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect('ssid', 'password')
```

Für die Platzhalter `ssid` und `password` sind natürlich die korrekten Zugangsdaten des vorhandenen WLANs einzusetzen. Die Anweisung

```
wlan.ifconfig()
```

liefert dann die IP-Daten, mit denen der ESP im Netz angemeldet wurde (**Bild 6**). Über

```
import webrepl
webrepl.start()
```

wird der WebREPL-Client aktiviert.
Danach kann in einem Browser die Adresse

```
http://micropython.org/webrepl/#xxx.xxx.xxx.xxx:8266
```

aufgerufen werden. Für `xxx.xxx.xxx.xxx` ist die über `wlan.ifconfig()` ermittelte IP einzusetzen. Danach kann man sich über den Reiter *Connect* mit dem oben im WebREPL-Setup festgelegten Kennwort anmelden.
Nach dem Start von WebREPL befindet sich die Konsole eventuell im „raw REPL“-Modus, der der direkten Eingabe von Befehlszeilen

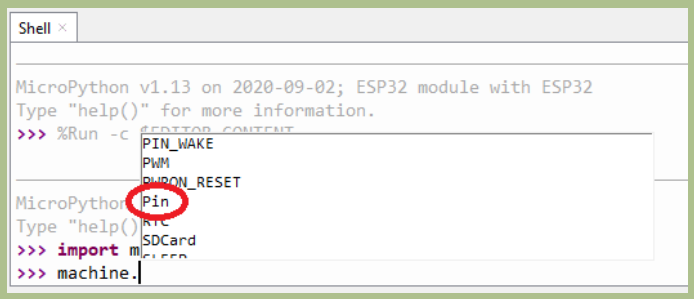


Bild 5. Auto-Vervollständigung.

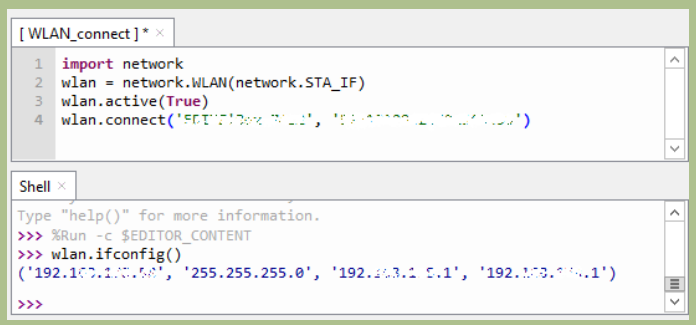


Bild 6. Der ESP32 ist im lokalen WLAN angemeldet.

über „copy und paste“ dient. Unter Umständen ist hier noch einmal mit CTRL-B in den Normal-Modus umzuschalten. Dann können die Anweisungen wie gewohnt eingegeben werden.
Auf diese Weise ist man nun in der Lage, den ESP vollständig drahtlos zu steuern. Über einfache Schaltfunktionen lassen sich so sogar bereits grundlegende Heimautomatisierungsfunktionen realisieren (**Bild 7**). Falls ein geeignetes Dateisystem auf dem ESP32 installiert wird, können sogar Softwareupdates drahtlos, also OTA (**O**ver **T**he **A**ir) vorgenommen werden.
Bei der Arbeit mit WebREPL sollte man beachten, dass es sich hierbei noch um ein „experimentelles Feature” handelt, welches noch nicht in allen Situationen absolut zuverlässig arbeitet.

Programmsteuerung

Die REPL- oder WebREPL-Konsole ist bestens für Testzwecke geeignet. Für klassische Programmabläufe steht dagegen das darüber liegende Editor-Fenster zur Verfügung. Dort kann beispielsweise das folgende Programm für eine automatische Nachlaufsteuerung (Nachtlicht, Treppenhaus-Beleuchtung) erstellt werden (**Bild 8**).
Nach der Eingabe und dem Starten des Programms mit dem Start-Icon (weißer Pfeil im grünen Kreis) erfolgt eine Aufforderung zum Speichern des Programms. Hier wird die Option *MicroPython device* gewählt und ein Programmname (etwa *Automatic_LED*) eingegeben und das Programm abgespeichert. Anschließend leuchtet die LED für 3 Sekunden auf und erlischt dann selbständig. Durch nochmaliges Betätigen des Start-Buttons kann das Programm nun direkt gestartet werden.
Um das klassische Demo-Programm (eine blinkende LED) umzusetzen, fehlt nur noch die `while`-Anweisung. Diese sorgt dafür, dass ein Befehl oder ein Befehlsblock wiederholt ausgeführt wird. Der Spezialfall `while True:` führt zu einer endlosen Wiederholung, also einem dauerhaften Blinken der LED:

```
from machine import Pin
from time import sleep

led = Pin(2, Pin.OUT)

while True:
  led.on()
  sleep(0.5)
  led.off()
  sleep(0.5)
```

Dabei ist zu beachten, dass die `while`-Anweisung mit einem Doppelpunkt abgeschlossen werden muss. Der darauf folgende Funktionsblock ist gemäß der allgemeinen Python-Konvention durch eine Einrückung über Leerzeichnen zu markieren. Thonny verfügt jedoch über eine automatische Einrückung. Sobald nach dem Doppelpunkt die Return-Taste gedrückt wird, erscheint der Cursor in der nächsten Zeile bereits an einer eingerückten Position. Selbstverständlich steht die TAB-Vervollständigung auch im Programm-Editor zur Verfügung. Laufende Programme können über *CTRL-C* beendet werden.

MicroPython in a Nutshell

Obwohl MicroPython von seinen Bibliotheken lebt, ist ein gewisses Verständnis der Elementar-Anweisungen erforderlich, um Programme zu verstehen oder selbst entwickeln zu können. Im Folgenden werden daher die wichtigsten MicroPython-Anweisungen in aller Kürze vorgestellt.
Einfache Kommentare werden mit dem #-Zeichen eingeleitet. Sie beginnen mit # und enden mit dem Zeilenende:

```
>>> print("hello ESP32")          # this is a comment
  hello ESP32
```

Ein Kommentar wird im Programmablauf ignoriert, da er nur Informationen für die Programmentwickler liefern soll. Über die hier ebenfalls verwendete `print()`-Anweisung können Informationen auf das Terminal ausgegeben werden. `Print()` ist einerseits direkt

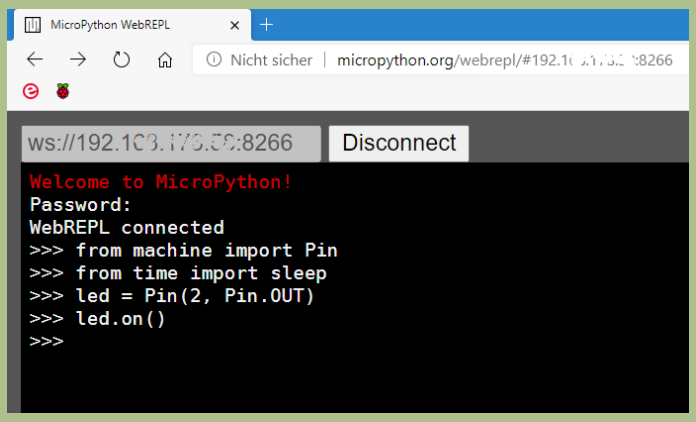


Bild 7. WebREPL im Browser.

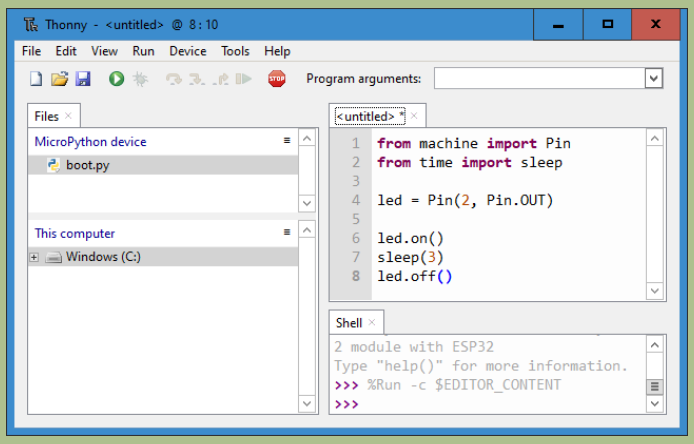


Bild 8. Automatische LED.

Passende Produkte

- **Buch: MicroPython für Mikrocontroller**
  www.elektor.de/micropython-fur-mikrocontroller
- **ESP32-PICO-Kit V4**
  www.elektor.de/esp32-pico-kit-v4
- **Breadboard**
  www.elektor.de/breadboard-830-tie-points
- **SSD1306 Display-Modul**
  www.elektor.de/blue-0-96-oled-display-i2c-4-pin

im Terminalfenster ausführbar, zum anderen dient die Anweisung in Programmen der Ausgabe von textbasierten Informationen. Wie bereits beim LED-Blinkprogramm klar wurde, werden in MicroPython verschiedene Blöcke durch Einrückungen gekennzeichnet. Damit sind geschweifte Klammern `{}` oder Ähnliches nicht mehr notwendig. Der Vorteil dieser Methode ist, dass man sozusagen bis zu einem gewissen Maß zur strukturierten Programmierung gezwungen wird.

Variablen lassen sich in Python besonders einfach erstellen. Es ist nicht erforderlich, einen Datentyp anzugeben. Zudem sind Variablen auch direkt in der Konsole einsetzbar:

```
>>> a=17
>>> b=12
>>> print(a*b)

            204
```

In MicroPython haben arithmetische Operatoren die aus der Mathematik bekannten Bedeutungen. Neben Addition, Subtraktion, Multiplikation und Division sind auch die Operatoren `//` für Ganzzahlige Division, `%` für Modulo (also Rest der Division) und `**` für Exponenten vorhanden.
Darüber hinaus stehen in MicroPython die üblichen Verzweigungs- und Schleifenanweisungen zur Verfügung. In einer Verzweigung wird im Programm eine Bedingung über die Schlüsselworte `if` und `else` definiert. Je nachdem, ob diese Bedingung wahr oder falsch ist, wird das Programm an unterschiedlichen Stellen fortgesetzt. Die `else`-Anweisungen werden nur ausgeführt, wenn die `if`-Abfrage falsch ist:

```
if True:
  # block 01
  print ("True")
else:
  # block 02
  print ("False")
```

Schleifen dienen dazu, Anweisungen zu wiederholen. Die Ausführung wird solange fortgesetzt, bis eine vorgegebene Bedingung erfüllt ist. Es sind zwei Varianten verfügbar:

- while-Schleifen
- for-Schleifen

Sollen beispielsweise die Zahlen von 1 bis 9 auf der Konsole ausgegeben werden, kann die folgende `while`-Schleife verwendet werden:

```
number=1
while number<10:
  print(number)
  number=number+1
```

Der zu wiederholende Code wird durch die Einrückung angezeigt. Die Aufgabe kann auch mit einer `for`-Schleife erledigt werden:

```
for number in range(1, 10):
  print(number)
```

Automatische Notsignale

Mit den jetzt bekannten Programmstrukturen kann bereits eine erste praktische Anwendung umgesetzt werden. Mit dem Programm entsteht eine automatische SOS-Bake, die bei See- oder Bergnot eingesetzt werden kann:

```
from machine import Pin
from time import sleep
led=Pin(2,Pin.OUT)

while True:
  for n in range(3):
    led.value(1)
    sleep(.1)
    led.value(0)
    sleep(.5)
  sleep(1)
  for n in range(3):
    led.value(1)
    sleep(.4)
    led.value(0)
    sleep(.5)
  sleep(1)
  for n in range(3):
    led.value(1)
    sleep(.1)
    led.value(0)
    sleep(.5)
  sleep(2)
```

Das OLED-Display: Ein Bildschirm im Kleinformat

Auch mit einer einzigen LED kann man also bereits sinnvolle Informationen ausgeben, wenn man beispielsweise, wie in der SOS-Bake, Morsecode verwendet. Wesentlich moderner ist natürlich die Datenausgabe über ein OLED-Display. Ein weit verbreiteter Typ sind Displays, die von einem SSD1306-Displaycontroller gesteuert werden. Wir verwenden eine Anzeigeeinheit, die bei einer Größe von nur 0,96 Zoll (etwa 2,5 cm) über eine Auflösung von 128 x 64 Pixel verfügt. MicroPython bringt bereits standardmäßig die Bibliothek für SSD1306-Displays mit. Diese ermöglicht sowohl die Darstellung von Texten und numerischen Daten als auch die Anzeige von einfachen Grafiken.

Die einfachsten Versionen der SSD1306-Module besitzen lediglich vier Pins. Dies ist ausreichend, um das Display über den sogenannten I²C-Bus anzusteuern. Der Anschluss des Displays ist bereits in Bild 3 dargestellt. Im Folgenden sind die erforderlichen Verbindungen in tabellarischer Form zusammengefasst:

OLED-Pin	ESP32
VDD	3V3
GND	GND
SCK	GPIO 22
SDA	GPIO 21

Das Skript in **Listing 1** gibt eine Textnachricht und ein einfaches Grafikelement in Form eines Rahmens auf das Display aus (siehe Bild 3).
Die zugehörige Library steht als Standardbibliothek zur Verfügung (*ssd1306.py* im Downloadpaket [5]) und kann separat auf das Board hochgeladen werden. Die Pin-Deklaration für den I²C-Bus erfolgt über:

```
i2c = I2C (-1, scl = Pin (22), sda = Pin (21))
```

Der Parameter `-1` gibt an, dass das verwendete Modul weder über Reset- noch über Interrupt-Pins verfügt. Die Anzahl der Pixel des angeschlossenen Moduls wird mit

```
oled = SSD1306_I2C(128, 64, i2c)
```

erfasst. Damit ist das Display betriebsbereit. Über die Funktion `text()` werden Informationen auf die Anzeige ausgegeben. Mit der Methode `show()` wird die Anzeige aktualisiert. Die `text()`-Funktion akzeptiert die folgenden Argumente:

- Nachricht (Typ String)
- X-Position und Y-Position des Textfeldes in Pixeleinheiten
- Optionale Textfarbe: 0 = schwarz (dunkel) und 1 = weiß (hell)

Die `show()`-Methode lässt die Änderungen auf dem Display sichtbar werden. Die `rect()`-Methode erlaubt beispielsweise die grafische Darstellungen eines Rechtecks. Sie akzeptiert die folgenden Argumente:

- X/Y-Koordinate der unteren linken Ecke des Rechtecks
- Y/Y-Koordinate der oberen rechten Ecke des Rechtecks
- Pixelfarbe: 0 = schwarz, 1 = weiß

Die Anweisung

```
oled.rect(5, 5, 116, 52, 1)
```

zaubert also einen rechteckigen Rahmen auf die Anzeige. Damit steht einer Anwendung des Displays für die Ausgabe von Informationen, vom einfachen Text bis hin zu Darstellung komplexer Sensordaten nichts mehr im Weg.

Listing 1. Nachricht auf dem OLED-Display.

```
from machine import Pin, I2C
from ssd1306 import SSD1306_I2C

i2c=I2C(-1,scl=Pin(22),sda=Pin(21))
oled = SSD1306_I2C(128, 64, i2c)
lin_hight = 9
col_width = 8

oled.fill(0)
oled.text("Hello", 5*col_width, 2*lin_hight)
oled.text("MicroPython", 2*col_width,
4*lin_hight)

oled.rect(5, 5, 116, 52, 1)
oled.show()
```

Zusammenfassung und Ausblick

Mit MicroPython steht eine moderne und leistungsfähige Programmiersprache zur Verfügung. Über Bibliotheken können auch komplexere Projekte schnell und einfach umgesetzt werden. Nachdem in diesem ersten Artikel die Installation der zugehörigen IDE und einige einfache Anwendungsbeispiele vorgestellt wurden, werden in einem zweiten Beitrag weitere Aspekte von MicroPython erläutert. Als Praxisanwendung soll dann unter anderem ein LED-Dotmatrix-Display im Großformat vorgestellt werden.

Weitere Informationen zum Thema MicroPython, zum ESP32-Controller und den hier vorgestellten Beispielen finden sich im Buch *MicroPython für Mikrocontroller* [4].

210179-02

Sie haben Fragen oder Kommentare?
Sie haben technische Fragen oder Kommentare zu diesem Artikel? Schreiben Sie uns unter redaktion@elektor..de!

Ein Beitrag von
Text und Bilder: **Dr. Günter Spanner**
Redaktion: **Jens Nickel**
Layout: **Harmen Heida**

WEB LINKS

[1] **Thonny**: https://thonny.org/
[2] **Python**: www.python.org/downloads
[3] **MicroPython**: http://micropython.org/download
[4] **Dr. Günter Spanner: „MicroPython für Mikrocontroller“**: www.elektor.de/micropython-fur-mikrocontroller
[5] **Software**: www.elektormagazine.de/210179-02

Ladungsgekoppelte Bauteile in Oszilloskopen

Bemerkenswerte Bauteile

Von **Neil Gruending** (Kanada)

Ich wette, das erste Ladungsgekoppelte Bauteil (Charge-Coupled Device, CCD), das Ihnen einfällt, ist der CMOS-Bildaufnahmesensor. Doch wussten Sie, dass sie nicht nur in Millionen von Kameras eingesetzt werden, sondern auch in anderen Bereichen verwendet wurden? Schauen wir uns an, wie Tektronix sie genutzt hat, um die Abtastrate von frühen digitalen Abtast- beziehungsweise Sampling-Oszilloskopen zu verbessern.

Ein ladungsgekoppeltes Bauteil ist genau so aufgebaut, wie es die Bezeichnung suggeriert: Es handelt sich, wie beim in **Bild 1** gezeigten TDA1022 von Philips [1] zu sehen, um ein Array von Metalloxidkondensatoren (MOS), die durch MOSFETs miteinander verbunden sind. Wenn ein analoges Signal an Pin 5 angeschlossen ist, wird es nach rechts verschoben, wenn die Gates der MOSFETs durch zwei phasenverschobene Takte getaktet werden, die an Pin 1 und 4 anliegen. Solche Bauteile wurden ursprünglich entworfen, um eine Verzögerungsleitung für analoge Schaltungen zu erzeugen, aber man erkannte bald, dass sie auch als Speicher dienen könnten.
Wie wir alle wissen, zeichnen traditionelle analoge Oszilloskope das gemessene Signal auf die (nachleuchtende) Kathodenstrahlröhre (CRT), was bei sich wiederholenden Signalen ein sehr genaues Ergebnis liefert. Wenn jedoch sehr langsame Signale oder schnelle Transienten gemessen werden, kann man dem Signalverlauf ohne ein digitales Oszilloskop nur schwer auf die Spur kommen.
Eines der ersten Oszilloskope mit digitaler Abtastung war das Tektronix 2440 (**Bild 2**). Als es entwickelt wurde, standen die Ingenieure vor dem ernsthaften Problem, dass die damaligen Analog-Digital-Wandler (ADCs) nicht schnell genug waren, um digitale Hochgeschwindigkeitssignale abzutasten. Die clevere Lösung der Tektronix-Entwickler war ein analoges FISO-Schieberegisters (Fast-in, Slow-out) vor dem ADC. Das FISO-Register war eigentlich eine CCD-Zeile, die das zu messende Signal mit hohen Abtastraten zwischenspeicherte (fast-in) und dann für die spätere Verarbeitung langsam ausgab (slow-out). Im Tektronix 2440 war das FISO-Register groß genug, um alle Abtastpunkte eines Frames zu puffern.
Dieser FISO-Ansatz wurde auch in den Oszilloskopen TDS300 und TDS600 verwendet, da diese über einen relativ kleinen Sample-Speicher verfügten. Leider ist es schwierig, den Abtastspeicher ohne viel schnellere ADCs zu vergrößern und die abgetasteten Daten direkt in einem RAM zu speichern, wie dies bei der TDS700-Serie der Fall war. Diese Baureihe besaß nicht nur einen größeren Speicherraum und hatte bessere Abtastfähigkeiten, sondern auch digitale Phosphordisplays, die den analogen Displays sehr nahe kamen. Sie dürften diesen Entwürfen aber nur sehr selten begegnen, außer, Sie besitzen eines dieser Old-Style-Oszilloskope. Wenn ein solches Gerät einmal preiswert angeboten wird, sollten Sie unbedingt zugreifen!

210181-02

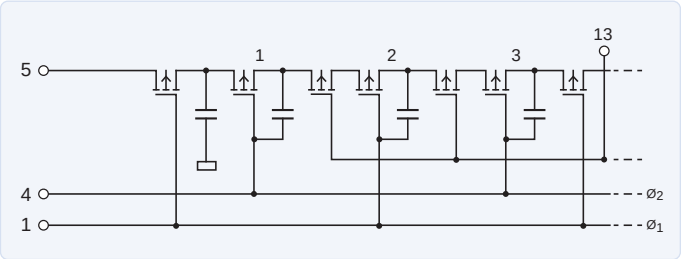


Bild 1. Die MOS-Verzögerungsleitung des TDA1022 [1].

Bild 2. Das zweikanalige Digital Sampling Oszilloskop Tektronix 2400 (300 MHz, 500 MS/s) aus den späten 1980er Jahren (Quelle: tekwiki [2]).

Haben Sie Fragen oder Kommentare?
Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Schicken Sie eine E-Mail an Elektor: editor@elektor.com.

WEBLINKS

[1] **Datenblatt TDA1022 Bucket Brigade Delay Line for Analog Signals:** http://bit.ly/3cUUQAW

[2] **Tektronix 2440 auf tekwiki:** https://w140.com/tekwiki/wiki/2440

ESD
der unsichtbare Blitz
Zerfetzt Halbleiter wie ein Blitz einen Baum

Bild 1. Die „Hindenburg" beim Andocken in Lakehurst (Foto: Sam Shere).

Von **Dipl.-Ing. Peter Beil**

Die immer kleiner werdenden Strukturen in Chips werden immer empfindlicher gegen elektrostatische Entladung. Lesen Sie hier, was es damit auf sich hat und wie man den damit verbundenen Problemen begegnet.

Bekanntlich endete der Atlantikflug des Luftschiffs Hindenburg 1937 in einer Katastrophe: Beim Andocken an den Ankermast im amerikanischen Lakehurst entzündeten sich explosionsartig 200.000 Kubikmeter Wasserstoff, die bei der Hindenburg als Traggas dienten (**Bild 1**). Die bis heute offiziell unbestätigte Ursache war ein durchaus bekannter physikalischer Vorgang: Die spontane Entladung statischer Elektrizität. Der korrekte Terminus dafür ist ESD (**E**lectro**S**tatic **D**ischarge).

Was ist ESD?

Es geht also um elektrische Potentialdifferenzen und deren Ausgleich durch spontane Entladungen. Grund für solche hohen Spannungsunterschiede ist oft der sogenannte „triboelektrische Effekt" – mit anderen Worten: Die Reibungselektrizität oder Influenz. Bekannt ist der Effekt, wenn man sich bei trockener Luft (daher im Winter) die Haare kämmt, über einen synthetischen Teppich geht oder aus einem Auto aussteigt. Bei Kontakt mit anderen leitfähigen Objekten mit anderem Potential gleichen sich die Potentiale schlagartig aus. Dabei kann bei entsprechend hoher Spannungsdifferenz eventuell schon kurz vor der Berührung ein Funke entstehen. Im Beispiel mit dem Teppich spürt man den

Funken bzw. die Entladung im Finger, wenn dieser sich z.B. einem Türgriff annähert (**Bild 2**). Beim Andocken der Hindenburg hatte vermutlich der dabei entstandene Funke die hochentzündliche Gasfüllung zur Explosion gebracht. Auf Grund der großen Oberfläche und der Luftreibung dürfte die Spannungsdifferenz zwischen Hülle und Mast leicht einige hunderttausend Volt betragen haben und die bei der Entladung aufgetretene Stromspitze ziemlich groß ausgefallen sein.
In kleineren Dimensionen begegnet uns das Phänomen jeden Tag: Beim Ausziehen eines Baumwoll-Pullovers, den man über einem Unterhemd aus Synthetikfasern getragen hat, hört man ein deutliches Knistern. Die dafür verantwortlichen Spannungsdifferenzen betragen locker einige kV. Im Dunkeln kann man diese Blitze sogar schon ab 5 kV sehen. Das Gehen mit Turnschuhen auf Kunststoff-Hartböden produziert bis zu 15 kV und auf Teppichböden bis zu 25 kV.
Wenn wir uns bewegen, stehen wir praktisch immer „unter Spannung". Bei geringer Luftfeuchtigkeit, etwa in klimatisierten Räumen oder in geheizter Luft im Winter, sind die Spannungen besonders hoch. Vielfach ist uns dies nicht bewusst: Erhebt man sich kurz von einem Stuhl, um nach etwas zu greifen und setzt sich dann wieder hin, hat man schnell bis zu 15 kV Potentialdifferenz produziert. Das Entscheidende aber ist: Spüren kann der Mensch solche Entladungen erst ab etwa 2000 bis 3000 V. Kleinere, aber für manche Dinge durchaus relevante Potentialdifferenzen bleiben also unbemerkt.
Dabei gilt es noch einen Faktor zu berücksichtigen: Laut dem sogenannten HBM (**H**uman **B**ody **M**odel) [1] geht man bei der Kapazität des menschlichen Körpers von 100…300 pF aus. Bei einem durchschnittlichen Haut-Übergangswiderstand von etwa 1,5 kΩ beim Entladen einer statischen Spannung von z.B. leicht erreichbaren 15 kV muss man mit Stromspitzen bis zu 10 A für einige zig Nanosekunden an den Übergangstellen rechnen.

Relevanz für Elektronik

Für Elektroniker besonders relevant: Auch wenn, wie schon erwähnt, viele elektrostatische Entladungen unter der Wahrnehmbarkeitsschwelle des Menschen liegen, können diese besonders für aktive elektronische Bauelemente schädlich sein. ESD hat infolge mangelnder Sorgfalt schon unzähligen Chips ein tödliches Ende bereitet. Das Berühren eines modernen ICs mit bloßen Händen kann zu einer für das Bauteil gefährlichen Entladung statischer Elektrizität führen. Da geht es formal der Elektronik nicht besser als einem Baum, in den ein Blitz einschlägt (**Bild 3**).
Relevant ist natürlich nicht nur die Höhe der Spannung, sondern vor allem die beim Potentialausgleich freiwerdende elektrischen Energie. Typischerweise wirkt ESD mit ein paar Nanosekunden nur sehr kurz – aber bei ICs auf einen winzigen räumlichen Bereich im Chip von nur wenigen Mikrometern Ausdehnung. Bei 15 kV und 150 pF stecken nach der Formel $E = ½C * U^2$ rund 17 mWs im menschlichen Kondensator. Trotz dieser eigentlich recht geringen elektrischen Energie resultiert aufgrund der Kürze der Entladung eine sehr hohe elektrische Leistung im kW-Bereich und aufgrund der kleinen Strukturen eine sehr hohe Leistungsdichte.
Betrachtet man ESD-Schäden unter dem Elektronenmikroskop, sieht man verkohlte Verbindungen, richtige Krater und verschmorte Reste einer kleinen Explosion auf dem Silizium (**Bild 4**), was durchaus an einen Blitzeinschlag erinnert. Diese Gefahren bedrohen übrigens nicht

Bild 2. Potentialausgleich per Funke, z.B. an einer Türklinke (dieses und weitere Bilder: Beil & Kaiser).

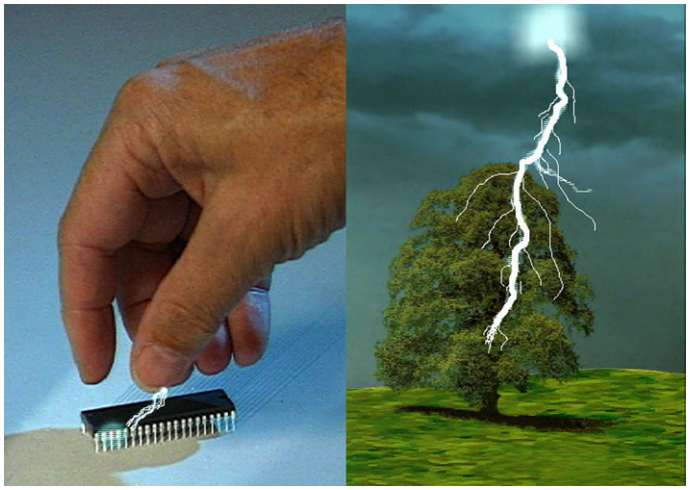


Bild 3. Was dem Baum die geladene Wolke ist dem IC ein geladener Finger.

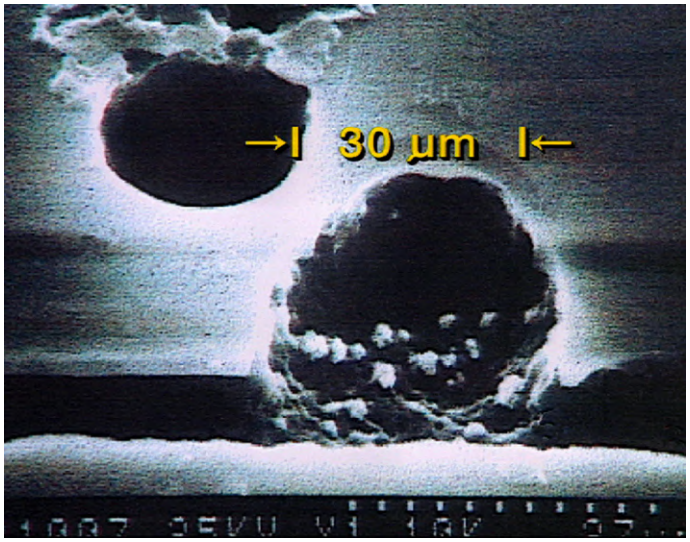


Bild 4. Geschädigtes IC unter dem Elektronenmikroskop.

Bild 5. Modell einer ESD-Schutzzone.

Bild 6. Offizielles ESD-Logo.

nur hochintegrierte ICs, wo man inzwischen bei Strukturen im Nanometerbereich angelangt ist, sondern auch Dioden, LEDs, MOSFETs und sogar Leistungshalbleiter.
Viele aktive elektronische Bauelemente können durchaus bei „kleineren" Spannungen um 1 kV zerstört werden, die wir selbst ja nicht wahrnehmen können. Deshalb bleiben solche Schäden oft unbemerkt. Fällt das ganze Bauteil aus, zeigt sich der Fehler schneller und eindeutiger. Doch wenn nur ein Teil eines ICs oder eine bestimmte Funktion beschädigt ist, bleibt das u.U. länger verborgen. Und trotzdem kann Letzteres zu großen Folgeschäden führen: Stellen Sie sich nur vor, bei einer Herz-OP setzt der Halbleiter-Laser aus, ein Airbag im Auto geht nicht bzw. zu spät auf oder Ihr neues autonomes Auto erkennt ein Hindernis nicht bzw. zu spät...

Vor ESD schützen

Ein zuverlässiger Schutz vor ESD beginnt damit, dass allen Beteiligten bewusst ist, wo die Gründe und Ursachen von ESD liegen. In der Produktion von Halbleitern sind demnach Schutzmaßnahmen von der Wafer-Fertigung bis zum Back-End und Bauteiletest notwendig. Beim Abnehmer dieser Produkte müssen Schutzmaßnahmen von der Entwicklung und Bestückung bis zur Auslieferung gewährleistet sein. Selbst im Hobby-Labor reduzieren ausreichende ESD-Kenntnisse potentiellen Frust.
Da das Problem so wichtig ist, sorgen Halbleiter-Hersteller durch entsprechende Design-Maßnahmen für eine sogenannte ESD-Härtung integrierter Schaltungen. Diese Schutzschaltungen wirken innerhalb der Baugruppe. Sie können allerdings nur begrenzte ESD-Events unschädlich machen. Mit der zunehmenden Verkleinerung von Halbleiterstrukturen schrumpft leider auch deren Schutzwirkung, da die Siliziumflächen zu klein und somit weniger belastbar geworden sind. Bei Mikrocontrollern setzt man daher auch auf externe Klemmschaltungen auf der Basis von Suppressor-Dioden. Diese beginnen, ähnlich wie Zenerdioden, ab einer gewissen Schwellspannung zu leiten – nur tun sie das sehr viel schneller und vertragen dabei sehr viel höhere Energieimpulse. Leider garantieren auch solche Lösungen keinen hundertprozentigen Schutz. Zu diesem Thema wurde in Elektor bereits ein ausführlicher Artikel [2] veröffentlicht.
Eine Grundvoraussetzung für die Arbeit mit hochintegrierten ICs ist eine speziell gekennzeichnete ESD-Schutzzone. Zur Vermeidung zu großer Potentialunterschiede sorgt man dort für einen kontrollierten Ausgleich von Ladungen (**Bild 5**). Diese Zone darf nur von entsprechend geschultem und ausgestattetem Fachpersonal betreten werden und wird mit dem Logo von **Bild 6** gekennzeichnet.
Eine solche ESD-Schutzzone ist mit einem leitfähigen und „geerdeten" Boden ausgestattet. Regalfächer und Arbeitsflächen weisen Ableitwiderstände < 100 kΩ auf, um riskante Ladungen weder zu schnell und heftig noch zu langsam abfließen zu lassen. Werkstoffe mit einem niedrigen Oberflächenwiderstand minimieren die auftretenden Potentialdifferenzen und sorgen dafür, dass entstehende Ladungen wieder gegen das Bezugspotential Erde abfließen und daher in ungefährlichen Bereichen bleiben.
Bewegliche Ablagen und Stühle haben neben leitfähigen Oberflächen auch leitfähige Räder. Ähnliches gilt auch für Werkzeuge wie Lötkolben etc. Isolierende Kunststoffgriffe können ja große Potentialunterschiede erzeugen. Hier empfehlen sich daher elektrostatisch leitfähige Materialien, die einen definiert langsamen Ladungsausgleich zwischen Mensch, Werkzeug und Bauteil bewirken.
Wichtig zu wissen: Diese Spezialwerkzeuge sind nicht geeignet für Arbeiten an Spannungen > 25 V und sie sind nicht VDE-konform!
Besondere Aufmerksamkeit verdienen Pinzetten oder andere spitze Werkzeuge. Dort kommt es auch bei geringer Aufladung zu einer Konzentration von elektrischen Ladungsträgern an der Spitze, die schon bei geringeren Spannungen zu Entladungen durch Funken führen können. Für die Arbeit mit sensiblen Bauteilen ist ein Handgelenkband (**Bild 7**) unerlässlich, das an einen vorgegebenen Erdungspunkt angeschlossen ist.
In manchen Umgebungen wie etwa Reinräumen ist der Einsatz von hochaufladbaren Isolationsmaterialien aus prozesstechnischen Gründen unvermeidbar. Diese Nichtleiter können nur durch den Einsatz sogenannter Ionisatoren neutralisiert werden. Elektrische Ladungen auf (meist hochisolierenden) Objekten kann man nicht durch Erdung beseitigen. Sie müssen über ionisierte Luft beschleunigt abgebaut werden (**Bild 8**). Dies kann noch durch Luftleit- und Ventilationssysteme unterstützt werden. Ionisation ist aber nur eine Ergänzung – keinesfalls ein ausreichender Ersatz für ESD-Schutz!

Passende Produkte

- **Bernstein 4-620 ESD Schraubenzieher-Set (6 Stück)**
www.elektor.de/bernstein-4-620-esd-screwdriver-set-6-pieces
- **Bernstein 2100 ESD Tool Holder VARIO (6 Werkzeuge)**
www.elektor.de/bernstein-2100-esd-tool-holder-vario-6-tools

Schutzmaßnahmen in der Fertigung

In der Fertigung hat man bei Bestückungsautomaten von in Rohren gleitenden Bauteilen Abstand genommen, da durch die Reibung beim Nachrutschen unerwünschte Effekte auftraten. Heute verwendet man hauptsächlich das „Pick-and-place"-System, bei dem die Bauteile mit Greifarmen auf der Platine platziert werden.

Das Betreten von ESD-Zonen ist nur mit entsprechendem Schuhwerk und in besonderer Arbeitsbekleidung erlaubt. Um Gewöhnungseffekten vorzubeugen, gibt es in der Industrie meist eine Schranke am Eingang. Diese öffnet sich nur nach einem Test der Ableitfähigkeit der Person: Durch Betreten einer Bodenplatte und Auflegen der Hand auf eine Messeinrichtung (**Bild 9**). Besucher erhalten ein Ableitband für die Schuhe und einen Arbeitsmantel aus Baumwolle. Dieser muss übrigens komplett geschlossen sein. Statische Elektrizität lässt sich quasi einsperren: Die Ladungen bleiben so innerhalb des Mantels.

Besondere Bedingungen gelten in der Halbleiterfertigung im Reinraum. Dort trägt man Spezialanzüge, die bis auf Augen und Hände den ganzen Körper bedecken. In diese Anzüge sind zusätzlich leitfähige Kohlefasern integriert (**Bild 10**). Auch kann dort das Tragen von ableitenden Fingerlingen oder Handschuhen nötig sein.

Sogar in ganz anderen Bereichen wird ESD sicht- und hörbar. Ein Beispiel wären Filmaufnahmen mit analogen Kameras. Beim Abwickeln der 35-mm-Filmrolle in der Kamera entstehen bei trockenem Wetter kleine Blitze, die das Negativ vorbelichten und so unerwünschte Bildstörungen verursachen. Dies geschieht wie oft vorranging bei Kälte und trockener Umgebungsluft. Im kopierten Positiv sieht man dann eine Art weißes Spinnennetz über dem Filmbild. Ähnliches geschieht bei der Magnettonaufzeichnung mit höheren Bandgeschwindigkeiten und breiteren Bandmaterialien. Dort entsteht ein leises Knistern, das aber relativ leicht durch einen Tiefpass entfernt werden kann. Aber zurück zur Elektronik...

Erst seit einigen Jahren kann man alle relevanten Ladungs- und Spannungswerte mit entsprechenden Mess-Ausrüstungen wie schnellen Oszilloskopen mit Sampling-Raten > 1 GHz erfassen. Für ESD-Ent-

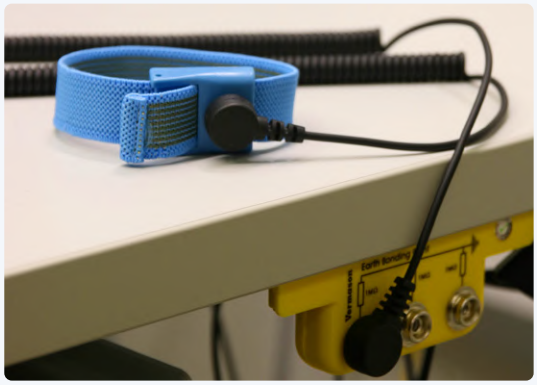


Bild 7. Handgelenk-Erdungsband.

Bild 8. Ionisatoren an der Decke einer Fertigungsanlage.

Bild 9. Zutrittsschranke zum Produktionskomplex mit ESD-Test.

Bild 10. Bekleidung in Reinräumen.

ladeimpulse benötigt man aufgrund der Kürze der Ereignisse Abtastabstände < 1 ns. Feldstärken misst man z.B. mit einem Elektrofeldmeter. All das ist mittlerweile in Regeln formalisiert [3] festgehalten.

ESD im kleinen Labor

Maker und sonstige Betreiber kleinerer Elektronik-Labore werden sich fragen, ob sie dieses Thema wirklich so ernst nehmen müssen. Zunächst: Auch zuhause gelten die gleichen physikalischen Gesetze wie in den Fertigungshallen der Großindustrie. Daher sollte man selbst in kleinerem Rahmen auf ESD-Schutzmaßnahmen nicht verzichten. Vielleicht muss man nicht ganz den gleichen technischen und finanziellen Aufwand betreiben, aber etwas abgemagerte Maßnahmen sind keinesfalls überflüssiger Luxus:

- Auch in einem kleinen Elektronik-Labor kann man durch eine leitfähige Bodenmatte mit Erdungsanschluss schon mal den Boden „entschärfen".
- Gleiches gilt auch für die Arbeitsfläche, wofür man ableitfähige Matten schon für kleines Geld bekommt.
- Ein Handgelenkband als Lebensversicherung für empfindliche Bauteile sollte an einem gemeinsamen Erdungspunkt angeschlossen sein.
- Jeder auch nur halbwegs professionelle Lötkolben hat einen Erdungsanschluss und Werkzeuge gibt es mit statisch ableitenden Griffen.
- Große Vorsicht sollte man bei spitzen Pinzetten walten lassen.
- Den Stuhl oder bewegliche Ablagen sichert man mit ableitfähigen ESD-Rollen.
- Empfindliche Bauteile befinden sich üblicherweise in ableitfähiger Verpackung und sollten bis zum Einbau darin verbleiben.
- Man achte darauf, dass ESD-Beutel keine Löcher oder Aufbewahrungskästen keine Risse aufweisen.

Eine sichere ESD-Strategie ist es, die ausgepackten Bauteile möglichst wenig zu bewegen und diese natürlich nicht an den Anschlüssen zu berühren. Ferner kann das Ausschütten von Bauteilen auf eine nicht ableitfähige Fläche zu unerwünschten Entladungen führen. Für das Bestücken von Bauteilen gibt es entsprechende Klammern oder Greifwerkzeuge. An diesen zu sparen kann recht teuer werden. Man sollte sich auch nicht zu fein sein, eine Art Arbeitsmantel zu tragen, der nicht aus synthetischen Fasern besteht. Außerdem sollte man diesen immer komplett schließen.
Nicht zu vernachlässigen ist letztlich die Sauberkeit: Also Arbeitsflächen reinigen, Löt- und Metallreste entfernen und die Stuhlrollen sowie den Boden regelmäßig reinigen.
Leider wird die Handhabung durch die hohe Komplexität und Miniaturisierung moderner Bauelemente in kleinen Elektronik-Laboren ohne teure und aufwändige Hilfsmittel immer schwieriger. Aber vielleicht verhelfen die Hinweise in diesem Beitrag dem einen oder anderen teuren IC zu einem längeren Leben.

200607-02

Typische Begriffe

ESD	ElectroStatic Discharge (elektrostatische Entladung)
ESA	ElectroStatic Area (ESD-Schutzzone)
ESDS	ElectroStatic Discharge Sensitive (empfindlich für elektrostatische Entladung)
Reinraum	In sich geschlossener, nahezu partikelfreier Produktionskomplex bei der Chip-Herstellung
Wafer	Runde Siliziumscheibe; die Ausgangsbasis integrierter Schaltungen.
Front End	Produktionsabschnitt; Erzeugung von Strukturen auf Wafern.
Back End	Produktionsabschnitt; Entstehung elektronischer Bauteile aus Wafern.
HBM	Human Body Model; ESD-Eigenschaften des menschlichen Körpers.

Ein Beitrag von
Text: **Peter Beil**
Redaktion: **Dr. Thomas Scherer**
Layout: **Giel Dols**

Sie haben Fragen oder Kommentare?
Mit technischen Fragen können Sie sich gerne an die Elektor-Redaktion wenden unter der E-Mail-Adresse: redaktion@elektor.de.

WEBLINKS

[1] **HBM**: https://de.wikipedia.org/wiki/ESD-Simulationsmodelle
[2] **„Aktive ESD-Schutzbeschaltung", Elektor 1/2014, Seite 93**: www.elektormagazine.de/magazine/elektor-201401/24335
[3] **Standards der ESD Association**: www.esda.org/about-esd/esd-fundamentals/part-6-esd-standards/

Solaranlage für Mähroboter

Ökologisch - preiswert - einfach!

Von **Dr. Thomas Scherer**

So ein Mähroboter ist eine feine Sache, bequem und praktisch dazu. Aber ökologisch besteht noch Optimierungspotential, denn er braucht dauernd Strom. Außerdem: Nicht überall kann man einfach ein 230-V-Kabel zwecks Stromversorgung verlegen. Eine passende Solaranlage löst beide Probleme.

Es war einmal (vor drei Jahren), als aus dem Nachbarsjungen ein junger Student wurde, der seine Zeit fürderhin besser mit Lernen als mit Rasenmähen in meinem Garten verbrachte. Also habe ich einen Mähroboter angeschafft. Diese Maßnahme hatte den zusätzlichen Vorteil, dass der Rasenschnitt und damit die Nährstoffe schon recycelt werden, statt mir in kurzer Zeit den Komposthaufen in einem Kompostberg zu verwandeln. Nach anfänglichen Schwierigkeiten bei der Verlegung des Begrenzungs- und Suchkabels passend zur recht komplexen Rasenform, tat der Roboter was er sollte. Und wenn er nicht kaputt gegangen ist, mäht er noch heute…

Solar-Betrieb

Alles wäre prima gewesen und das Märchen hier schon zu Ende, hätte meine Freundin – ihres Zeichens promovierte Chemikerin – mich nicht eines Tages geneckt. Es trug sich nämlich zu, dass ein guter Freund sein Motorboot mit einem elektrischen Kühlschrank bestücken wollte, der mit Solarstrom versorgt werden sollte. Als „Elektrikus“ meines Clans war ich natürlich für die grundlegende Berechnung der Solaranlage zuständig. Und kaum dass Alexandra das mitbekam, spitzte sie die Lippen und fragte mich mit gespielter Unschuld: „Und warum läuft Dein Rasenroboter noch nicht mit Solarenergie?“
Peng! Das konnte ich natürlich nicht auf mir sitzen lassen. Also überschlug ich die Sache schnell im Kopf, informierte mich über aktuelle Komponentenpreise im Internet und schon eine halbe Stunde später teilte ich ihr mit: „Das geht ziemlich einfach und ist auch noch recht preiswert. Mit etwa 100 € kommt man da hin!“
Sie schmunzelte skeptisch, denn als überzeugte Naturwissenschaftlerin operiert das in ihrem Gehirn installierte, Goethe-orientierte Betriebssystem nach dem Motto: *„Die Botschaft hör ich wohl, allein mir fehlt der Glaube.“* Für sie zählen demnach keine schönen Worte, sondern nur Taten. Also musste ich den Beweis antreten…

Basisüberlegungen

Ich habe einen Mähroboter von Gardena, der eine preiswerte Variante der bekannten Geräte von Husquarna ist. Für ca. 550 m² Rasen muss er in der Wachstumsphase ca. 4 h/d mähen – im Frühjahr und Herbst sowie im trockenen Hochsommer kommt er mit 2 h/d hin. Glücklicherweise gibt es viel Sonne genau dann, wenn sie auch benötigt wird.
Der Roboter hat einen Akku, der etwa für eine Stunde Mähen reicht. Anschließend muss er wieder an seine Basisstation und nach einer Stunde Laden ist er wieder mähbereit. Will ich ihn 2 h/d mähen lassen, muss ich ihm (per Funk) ein Fenster von

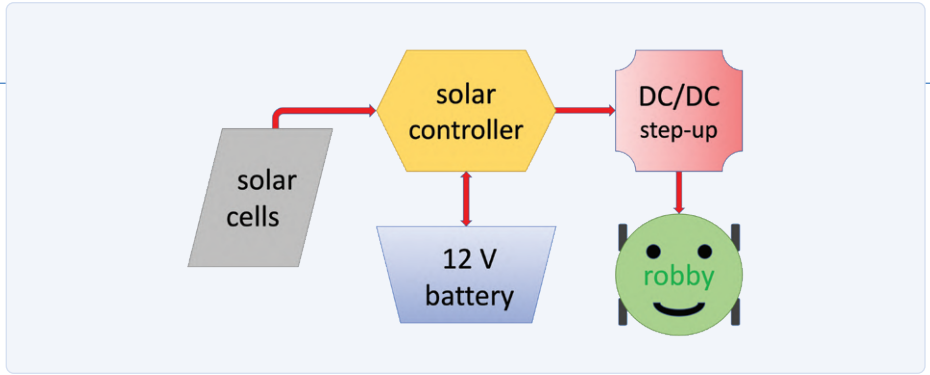


Bild 1. Die Blockschaltung einer Solaranlage für Mähroboter mit Energiefluss (rot).

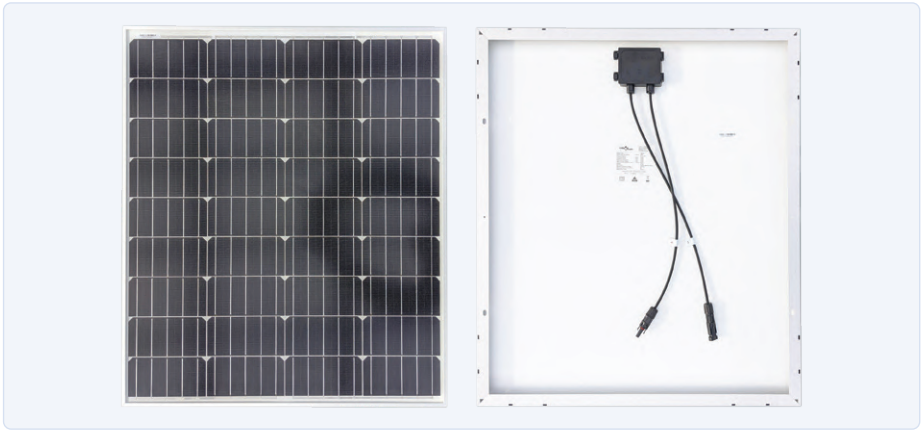


Bild 2. Vorder- und Rückseite meines 12-V-Solarpanels mit 80 W_{peak}.

3 h einprogrammieren, denn eine Stunde davon ist ja Ladepause.
Um abschätzen zu können, wie viel Energie pro Ladevorgang benötigt wird, habe ich das Internet befragt: Das zugehörige Netzteil liefert angeblich 28 V bei maximal 1,3 A. Da HTML geduldig ist, habe ich nachgemessen: Tatsächlich lieferte mein Netzteil eine Gleichspannung von 28,1 V im Nichtladebetrieb. Als der Roboter seinen Akku lud, zeigte mein Zangen-Amperemeter einen Strom von 1,29 A an. Außerdem war dabei die Spannung auf 26,5 V eingebrochen. „Das Netzteil ist aber weich!" dachte ich mir, was sich aber später noch als falsch herausstellen sollte ;-).
Wie dem auch sei: 26,5 V x 1,3 A macht nach Adam Ries und der Leistungsformel rund 35 W. Pro einstündigem Ladevorgang werden also etwa 35 Wh benötigt. Für die vier Mähvorgänge eines Frühsommertages ist demnach mit 140 Wh zu rechnen. So viel elektrische Energie pro Tag sollte die Solaranlage daher auch bei wenig Sonne liefern und auch speichern können.
Hinzu kommt noch der Energiebedarf der Basisstation, die ja rund um die Uhr die Begrenzungsdrähte mit einem gepulsten Signal ordentlicher Stärke versorgt, dessen Magnetfeld vom Roboter im Betrieb detektiert wird (siehe mein Artikel unter [1]). Ich hatte einen Ruhestrom (ohne Ladung) von 85 mA bei 28,1 V gemessen. Zusätzlich zur Ladung werden folglich etwa 2,4 W (≈ 58 Wh/d) benötigt.
Auch wenn diese Werte so nur für meinen Mähroboter gelten, kann man doch berechtigt davon ausgehen, dass sie mehr oder weniger für die meisten der aktuell verfügbaren Geräte gelten. Mein Roboter ist für 1.100 m² geeignet und daher „Mittelklasse". Kleinere Roboter brauchen sicher weniger Energie und größere entsprechend mehr. Eigenes Messen ist allerdings besser als bloßes Schätzen.

Solaranlage

Rechne ich etwa 200 Tage mit durchschnittlich drei Mähstunden pro Jahr, so komme ich auf 600 1-h-Ladezyklen, was einen jährlichen Energiebedarf von 21 kWh ergibt. Addiert man noch die 2,4 W x 24 h x 200 ≈ 11,5 kWh dazu, ergeben sich bei 30 ¢/kWh Stromkosten von 9,75 €. Eine Solaranlage für die angenommenen 100 € hätte sich also „schon" in zehn Jahren amortisiert. Lohnt sich das? Ökonomisch ist das vielleicht nicht der Brüller, aber ökologisch schon, denn in der Zeit wurden etwa 105 kg CO_2 beim Stromlieferanten eingespart. Außerdem ist man unabhängig vom Netzanschluss in der Wahl des Aufstellungsorts für die Ladestation des Mähers. Drittens Alexandra…
Wie dem auch sei - die Solaranlage für den Mähroboter benötigt vier Komponenten (**Bild 1**): Neben dem **Solarpanel** braucht es einen **Akku** als Energiespeicher, da ansonsten der Mäher nicht geladen werden kann, wenn gerade keine Sonne scheint. Aus Letzterem ergibt sich die Notwendigkeit für einen **Laderegler**, der die Energie vom Panel in den Akku transferiert und dabei Ladeschluss- und Entladeschlussspannungen des Akkus beachtet. Last not least ist noch ein **DC/DC-Konverter** erforderlich, der die Spannung des Akkus auf das von der Ladestation des Mähers benötigte Niveau anhebt. Zu jeder Komponente sind Überlegungen und eine Überschlagrechnung erforderlich, um sie adäquat zu dimensionieren. Basis aller Rechnungen ist der maximale Energiebedarf pro Tag, der in meinem Fall bei ≤200 Wh liegt. Nun zu den Kalkulationen.

Solarpanel

Im Idealfall sollte das Panel in zwei Stunden so viel Energie liefern, wie der Mäher in einer Stunde Mähen verbraucht, wenn man die einstündige Ladepause berücksichtigt. Rechnet man mit 15 % Verlusten durch Laderegler und DC/DC-Konverter, liegt man bei rund 40 W pro 2 h = 20 W Dauerleistung in der Zeit, in der der Mäher mäht (und lädt). Dann bleibt der Akku meistens voll, wenn die Sonne scheint. Da die Leistungsabgabe eines Solarpanels in W_{peak} gemessen wird und man selten optimalste Bedingungen ohne Abschattung und eine ideale Panelneigung haben wird, ist eine Leistungsverdoppelung sicherlich nicht verkehrt. Reicht also ein Panel mit 40 W_{peak}? Meiner Meinung nach nicht, da die Sonne im Frühling und Herbst sehr schräg scheint, wäre das immer noch eine sehr optimistische Auslegung. Ein weiterer Faktor 2 als Sicherheit ist sicherlich nicht übertrieben. Also 80 W_{peak} in meinem Fall als Minimum. Ich habe solch ein Panel bei eBay für 55 € geordert (**Bild 2**), denn es hatte – was für ein Zufall! – mit 77 x 66,5 cm genau die richtigen Maße um als Dach für die zur Mähergarage zweckentfremdete Hundehütte zu dienen.

Messungen ergaben, dass Ende Oktober in Südbaden um die Mittagszeit bei waagrechter Montage mit etwa 22 W Leistungsabgabe zu rechnen ist. Die Überschlagsrechnung stimmte!

Akku

Der Energiespeicher sollte für mindestens einen Tag dichter Bewölkung ausreichen. Er muss also ≥140 Wh speichern können. Ein AGM-12-V-Blei-Akku mit 12 Ah ≈ 150 Wh kostet bei eBay etwa 25 €. AGM muss sein, um bei dieser Anwendung möglichst viele Ladezyklen und damit hohe Haltbarkeit zu erzielen. Ein 80-W-Panel liefert einen Spitzenstrom von knapp unter 6 A. Es wird also mit maximal 0,5 C geladen (meistens mit viel weniger; siehe Kasten **Ströme in C**), was der Lebensdauer zugute kommt. Entladen wird mit etwa 40 W, was etwa 0,25 C entspricht. Da außerdem viele kurze Teilladezyklen vorkommen, kann man gerechtfertigt jahrelangen Betrieb erwarten. Ein 12 Ah-Akku ist für diese Anwendung das Minimum – mehr wäre besser – aber ich wollte es genau so ausprobieren und habe deshalb den Akku in **Bild 3** bestellt.

Bild 3. Der 12-V-AGM-Bleiakku mit 12 Ah.

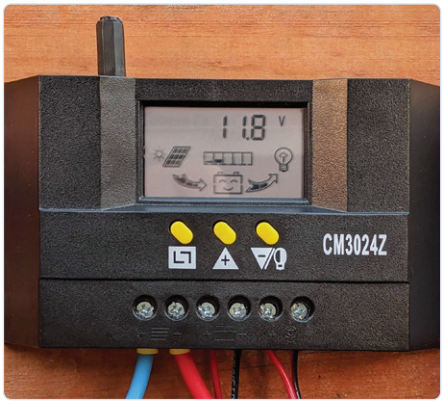


Bild 4. Preiswerter Solar-Laderegler in PWM-Technik mit Strömen bis zu 30 A.

Solar-Laderegler

Der Markt ist überschwemmt mit Massen an preiswerten Ladereglern kleiner Leistung, die für meine Zwecke in Frage kommen. Ich habe mit für einen mit 13,50 € vergleichsweise „teuren" Regler entschieden, da er ein Metallgehäuse und Kühlrippen auf der Rückseite hat. Vielleicht übertrieben, da meine Anlage die möglichen 30 A höchstens zu 20 % ausreizt.
Preiswerte Exemplare arbeiten durchweg mit einfacher PWM-Regelung, was nicht ganz optimal ist. Will man einen besseren MPPT-Regler (Maximum Power Point Tracking) für optimalen Wirkungsgrad, muss man hierfür schon gut 75 € aufwärts berappen. Um unter den projektierten 100 € zu bleiben, habe ich mich für die preiswertere Variante von **Bild 4** entschieden.

DC/DC-Konverter

Zuerst dachte ich, dass eine 24-V-Solaranlage mit zwei in Serie geschalteten 12-V-Akkus die simplere Lösung wäre. Vermutlich kommt nämlich die Ladestation des Mähers auch gut mit 24 V zurecht, auch wenn sie für 28 V spezifiziert ist. Man würde dann einen Boost-Konverter einsparen, der bei einer 12-V-Anlage für das Hochsetzen der Betriebsspannung zuständig ist. Dabei würde der Wandlungsverlust und der Ruhestrom des Konverters wegfallen. 24-V-Solarpanel sind aber teurer. Wie sich später noch zeigen sollte, habe ich mit der Entscheidung für 12-V-Technik eine glückliche Hand gehabt.
Also musste ein sogenannter Step-up-Konverter ausreichender Leistung her, der die 12...14 V des Akkus auf die geforderten 28 V bringt. Mindestens 40 W sollte er liefern können. Im Internet findet man passende Module ohne Hülle in Fülle. Ich entschied mit für ein 150-W-Modul für knapp unter 10 €, was ausreichende Reserven versprach. **Bild 5** zeigt die modifizierte Version, eingebaut in ein Kunststoff-Gehäuse. Als Wirkungsgrad werden 95 % versprochen. Und das stimmte, wie meine Messungen ergaben. Und es ist auch kein Wunder, denn IC1 ist mit dem Typ UC3843 sicher keine schlechte Wahl.
Insgesamt habe ich also 55 + 25 + 13,5 + 10 = 103,5 € ausgeben – ganz gut geschätzt, nicht wahr? Hinzu kamen noch ein paar Euro für Kleinteile und Edelstahlschrauben, Scharniere und das Kunststoffgehäuse.

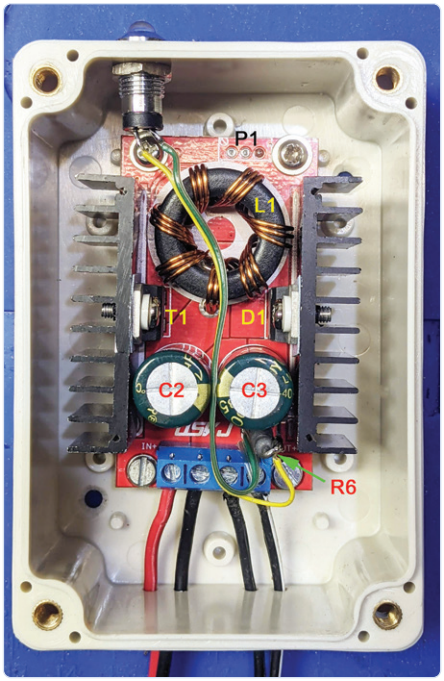


Bild 5. Dieser einfache Step-Up-Konverter wurde noch um eine Strombegrenzung bei 1,3 A erweitert.

Bild 6. Die vier Stadien des Umbaus einer zweckentfremdeten Hundehütte zur Solargarage für meinen Mähroboter.

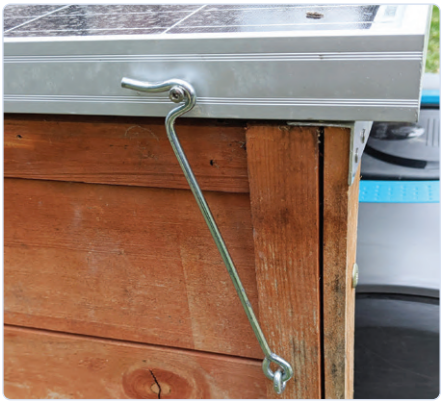


Bild 7. So wird das Panel windsicher arretiert.

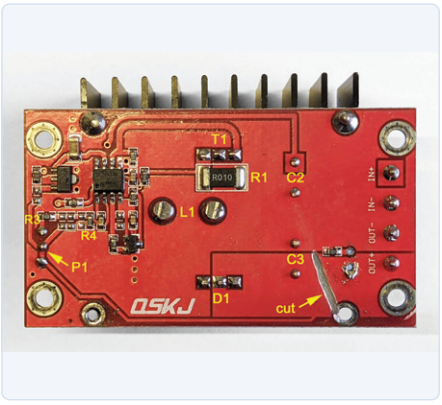


Bild 8. Rückseite der Platine des Step-up-Konverters.

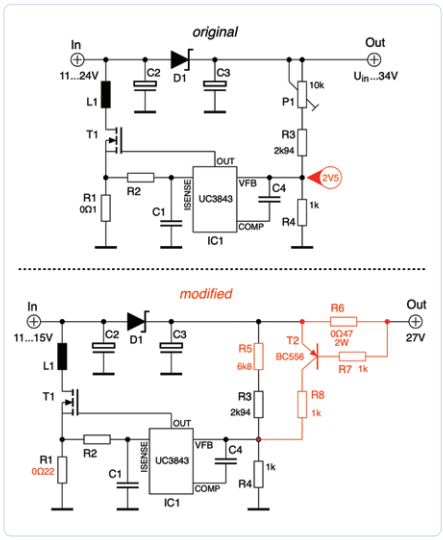


Bild 9. Originale und modifizierte (Teil-) Schaltung des Step-up-Konverters.

Bild 10. Alles fertig: Solargarage mit eingebauter Elektronik.

Umbau mit Hindernissen

Zunächst kamen die von meinem Elektronikerherz ungeliebten mechanischen Arbeiten: Der Umbau der bisherigen Behausung der Ladestation in eine Garage mit Solardach. Die vier Teilbilder von **Bild 6** illustrieren mein Vorgehen.

Zunächst wurden die beiden Dachhälften (**a**) entfernt und dann wie in (**b**) zu sehen die Giebel abgesägt. Dann kamen Alu-Winkelprofile zur Stabilisierung des nunmehr etwas wackeligen Restes, und schließlich wurde das Solarpanel aufklappbar befestigt (**c**). Edelstahl ist bei Scharnieren und Schrauben nicht übertrieben, wenn man dem Rosten vorbeugen will. In (**d**) ist das zugeklappte Resultat mit geparktem Mäher zu sehen. **Bild 7** zeigt, wie ich das Panel windsicher verriegelt habe, was ein **absolutes Muss** ist.

Die nächste Stufe ist schon interessanter gewesen: Endlich wurden Akku, Laderegler und Boost-Konverter eingebaut und verkabelt. Und sofort leuchtete die LED der Ladestation grün, obwohl ich den Konverter eingedenk des Spannungsabfalls der Originalnetzteils auf lediglich 27 V eingestellt hatte. Zunächst schien also alles zu funktionieren und tatsächlich erkannte der Mähroboter die verlegten Begrenzungsdrähte und mähte fröhlich vor sich hin.

Um zu sehen, ob er auch schön lädt, habe ich dann den Mäher in seine Ladestation geschoben. Schon machte mein Smartphone „Ping“, da seine App eine Fehlermeldung vom Mäher erhielt…

Angezeigt wurde: „Ladestrom zu hoch“.

Oha! „Das könnte ins Auge gehen!“ dachte ich mir und zog schnell den Mäher aus der Ladestation. Abends dann fragte Alexandra „Hat alles geklappt mit deinem Rasenmäher?“ Kleinlaut musste ich zugeben, dass nur eine Statusmeldung im Stile von Radio Eriwan berechtigt war: „Im Prinzip ja, aber…“

Ich erzählte, dass der Ruhebetrieb gut funktioniert, aber die Fehlermeldung nicht so lustig sei. Aber ich hätte einen Verdacht, dass nämlich der Roboter darauf baut, dass sein Netzteil eine eingebaute Strombegrenzung habe, und deshalb darauf aufbauend schlicht eine Ladung mit einem Konstantstrom erfolgen würde. Morgen würde ich das Netzteil vermessen und gegebenenfalls den Boost-Konverter mit einer Strombegrenzung nachrüsten oder eine extra Mimik dafür bauen. Hätte ich hier direkt 24 V aus Akkus ohne Strombegrenzung angeschlossen, hätte es wohl Rauchwölkchen gegeben. Glück im Unglück gehabt.

Strombegrenzung

Am nächsten Tag zeigte sich, dass meine Vermutung zutraf. An einer Last von 24 Ω lieferte das originale Netzteil einen Strom von 1,17 A bei stabilen 28,05 V. Bei Belastung mit 12 Ω konnte ich nur noch 15,5 V messen. Der Strom betrug 1,29 A. Die Gegenprobe mit 15 Ω führte zu 19,4 V bei ebenfalls 1,29 A. Demnach war das ein 28-V-Netzteil mit einer eingebauten Strombegrenzung von etwa 1,3 A.

Also habe ich mit die Platine des Konverters genauer angeschaut (**Bild 8**) und mir die Schaltung soweit nötig erschlossen. Die obere Hälfte von **Bild 9** zeigt die originale Schaltung. In der unteren Hälfte sind meine Modifikationen rot eingezeichnet.

Die Ausgangsspannung wird durch IC1 so eingestellt, dass an seinem Eingang *VFB* eine Spannung von 2,5 V anliegt. Sie ist mit P1 zwischen U_{in} und 34 V einstellbar. Kleiner als am Eingang kann die Spannung bei diesem Wandlertyp prinzipbedingt nicht werden, da auch beim Abschalten der Wandlung Strom durch L1 und D1 zum Ausgang fließt. Aber wenn die Spannung am Ausgang zwischen 14 V und 28 V variieren kann, reicht das auch, dachte ich mir. Man muss also nur einen mit R6 einen Shunt-Widerstand einschleifen und die Spannung an Pin VFB bei Überlast durch einen PNP-Transistor erhöhen, dann geht die Spannung am Ausgang in die Knie.

Zunächst habe ich P1 durch einen Festwiderstand von 6,8 kΩ ersetzt. Es ergaben sich damit 27 V – passt. Dann wurde der Stromfühler R1 durch höherohmige 0,22 Ω (SMD 2512) ersetzt, da 150 W gar nicht erforderlich sind. Mit 0,47 Ω für R6 ergibt sich

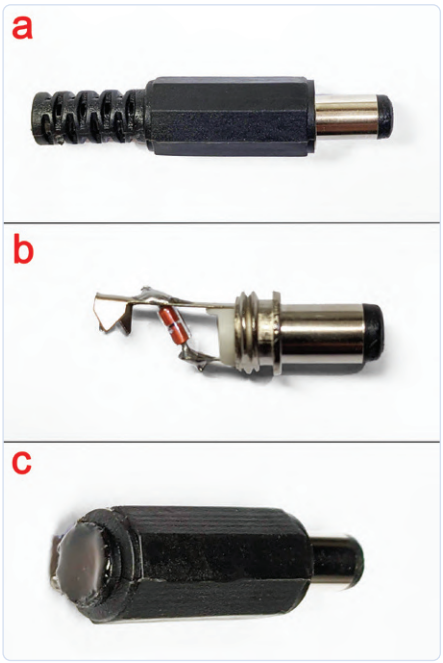


Bild 11. In diesem 3,5-mm-Klinkenstecker (a) steckt eine Diode (b) als Temperatursensor. Rechts (c) ist er wasserdicht mit Heißkleber vergossen.

bei einer BE-Spannung von etwa 650 mV des zusätzlichen T2 eine Strombegrenzung von rechnerisch 1,38 A. Auch das müsste passen. Also baute ich die roten Bauteile ein (T2, R7 und R8 direkt auf der Unterseite der Platine) und zückte das Amperemeter: genau 1,30 A. Treffer!

Ende gut, alles gut!

Ich baute den Modifizierten Konverter wieder in sein Gehäuse und platzierte dieses wieder in die Garage des Mähers (**Bild 10**) und schaltete ein: Die grüne LED der Ladestation leuchtete und Tataa: Beim Einschieben des Mähers kam keine Fehlermeldung mehr. Der Laderegler zeigte an, dass etwa 37 W Richtung Mähroboter unterwegs waren. Was wollte ich mehr?

Bevor ich Alexandra triumphierend von meinem Erfolg berichtete, habe ich den Mäher mehrere Male mähen und laden lassen. Es traten keine Probleme mehr auf. Und falls Sie sich jetzt fragen, was passiert, wenn es im Herbst oder im Frühling mal eine Woche lang regnet – diese Frage ist berechtigt: Bei einer Mähzeit von 2 h/d hält der Akku gerade mal 1,5...2 Tage, dann ist er leer und der Mäher bleibt solange in der Garage, bis die Sonne wieder ausreichend scheint. Leichte Bewölkung reicht aus, dass er wieder erwacht. Will man mehr Reserve, muss man den Akku und gegebenenfalls das Panel größer dimensionieren. Ein 100-W-Panel und ein 20-Ah-Akku ist sicher keine übertriebene Verschwendung. Der Laderegler und auch der Konverter vertragen noch deutlich größere Leistungen.

Der Winter ist übrigens kein Problem, denn da kommt der Mäher (samt Akku) von seiner Garage in meine Garage.

Nachtrag

Ein Blei-Akku bzw. seine Schlussspannungen haben einen Temperaturgang. Folglich haben manche Laderegler einen Temperatursensor, um dem Rechnung zu tragen. Mein Laderegler hatte einen anscheinend leeren 3,5-mm-Klinkenstecker beiliegen (**Bild 11a**). Beim Abschrauben der Hülle kam eine simple Silizium-Diode zum Vorschein (**b**). Klar, ein PN-Übergang hat einen Temperaturkoeffizienten von etwa -1,7 mV/K, und das ist ausreichend. Der primitive Aufbau war aber wegen möglicher Feuchtigkeit und kleinen Tierchen, die eine Behausung suchen, keine gute Idee. Deshalb schnitt ich den Knickschutz ab und füllte die Tülle einfach mit Heißkleber (**c**). Jetzt ist das wasser- und tierdicht.

Falls Sie etwas Ähnliches vorhaben: Es gibt auch fertige Step-up/down- bzw. Boost/Buck-Konverter mit einstellbarer Strombegrenzung für nur wenige Euro Aufpreis. Damit können Sie sich die von mir vorgenommene Modifikation sparen...

200553-02

Sie haben Fragen oder Kommentare?
Gerne können Sie sich an die Elektor-Redaktion wenden unter der E-Mail-Adresse: redaktion@elektor.de.

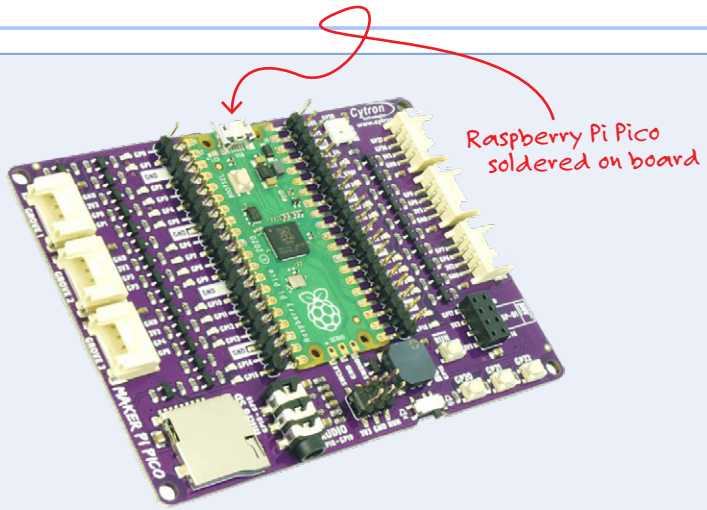
Ein Beitrag von
Idee, Durchführung und Text:
Dr. Thomas Scherer
Redaktion: **Jens Nickel**
Layout: **Giel Dols**

STRÖME IN C

Bei Akkus wird der relative Lade- bzw. Entladestrom in der gebräuchlichen „Nicht-SI-Einheit" C angegeben. Wenn man den Strom in A durch die Kapazität in Ah teilt, erhält man den relativen Lade- bzw. Entladestrom in der Einheit C. Bei einem C von 0,5 ergibt sich bei einem 12-Ah-Akku ein Strom von 6 A. Der Vorteil der Einheit C ist, dass er die Belastung des Akkus indiziert. Akkus sind für bestimmte maximale C-Werte konzipiert. Je größer die sich real ergebenden C-Werte, desto geringer die Lebensdauer. Bei gegebenem Strom leben größere Akkus also länger.

WEBLINK

[1] **Elektor-Artikel „Leitungen aufspüren":**
www.elektormagazine.de/magazine/elektor-140/57096



Cytron Maker Pi Pico (inkl. Raspberry Pi Pico RP2040)

Preis: 19,95 €
Mitgliederpreis: 17,96 €

www.elektor.de/19706

Andonstar AD409 Digital-Mikroskop mit 10,1“ LCD-Bildschirm

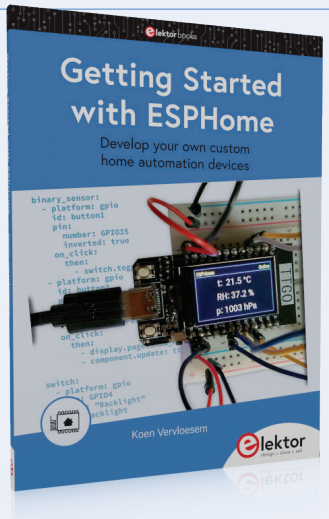
Preis: 399,00 €
Mitgliederpreis: 359,10 €

www.elektor.de/19681

The Complete Linear Audio Library (USB-Stick)

Preis: 149,95 €
Mitgliederpreis: 89,95 €

www.elektor.de/19672



Getting Started with ESPHome

Preis: 29,95 €
Mitgliederpreis: 26,96 €

www.elektor.de/19738

Europas Bemühungen, Big Tech zu zähmen

Von **Tessel Renzenbrink** (Niederlande)

Die Europäische Kommission arbeitet für ein menschenfreundliches Internet, in dem die Grundrechte der Nutzer respektiert werden, und strebt einen digitalen Binnenmarkt mit gleichen Wettbewerbsbedingungen an. Ist das möglich, oder wird Big Tech ungezähmt bleiben?

Europa will sein Internet zurück. Es versucht, die Macht über die digitale Sphäre den Klauen der Big-Tech-Unternehmen zu entreißen. Als Teil ihrer Strategie für die digitale Zukunft der EU strebt die Europäische Kommission (EC) einen digitalen Raum an, der auf europäischen Werten basiert - ein menschenzentriertes Internet, in dem die Grundrechte der Menschen respektiert werden, und einen digitalen Binnenmarkt mit gleichen Wettbewerbsbedingungen. Um dies zu erreichen, arbeitet die EC an neuen Gesetzen zur Regulierung des digitalen Raums. Kernstück der Gesetzgebung ist der *Digital Services Act* (DSA), der im letzten Dezember vorgeschlagen wurde.

Neben dem Digital Services Act (DSA), der darauf abzielt, die Bürger und ihre Grundrechte online zu schützen, ist es der *Digital Markets Act* (DMA), der die beträchtliche Macht der „sehr großen Online-Plattformen" oder „Gatekeeper" (Informationsregulatoren) eindämmen soll. Das Gesetzespaket führt ein gestaffeltes Verantwortungsschema ein, das auf der Größe eines Unternehmens beruht. Die unterste Stufe betrifft kleine Unternehmen. Sie haben die geringste Verpflichtung, ihre regulatorische Einflussnahme zu begrenzen. Die zweite Stufe ist für größere Plattformen reserviert. Die dritte Stufe zielt auf die sehr großen Online-Plattformen wie Facebook und Google ab, die als Gatekeeper bezeichnet werden, weil sie als wichtige Schnittstelle zwischen Unternehmen und Kunden dienen. Das gibt ihnen die Macht, Regeln aufzustellen, die andere Unternehmen benachteiligen und zu unlauterem Wettbewerb führen können. Aus diesem Grund weist der Vorschlag ihnen die meisten Verantwortlichkeiten zu. Eine Plattform gilt als Gatekeeper, wenn sie mindestens 10% der EU-Bevölkerung bedient.

Der Digital Service Act

Eines der Probleme, die der DSA angehen will, ist die Verbreitung von illegalen Inhalten wie Hassreden, Rachepornos, Online-Stalking und das Anbieten von gefälschten Waren. Durch das vorgeschlagene Gesetz erhalten sowohl die Behörden der Mitgliedsstaaten als auch ihre Bürger mehr Kontrolle, um Anbieter digitaler Dienste über solche illegalen Inhalte zu informieren. Service-Provider müssen benutzerfreundliche Mechanismen einrichten, die es jedem ermöglichen, als illegal erachtete Inhalte zu melden. Der Anbieter muss die Meldung zügig bearbeiten und den Meldenden über seine Entscheidung informieren, wie er mit dem Inhalt umgehen wird. Wenn die Entscheidung durch einen automatisierten Prozess getroffen wird, muss auch dies dem Meldenden mitgeteilt werden. Außerdem müssen sie eine natürliche Person in ihrem Unternehmen benennen, die als Ansprechpartner für die Behörden der Mitgliedsstaaten dient.

Es ist nicht ohne Risiko, Online-Plattformen zu verpflichten, illegale Inhalte zu entfernen und sie zur Rechenschaft zu ziehen, wenn sie es nicht tun. Es kann einen Anreiz für eine Art Überreaktion schaffen und zu einer übermäßigen Entfernung von Inhalten führen. Organisationen für digitale Rechte haben auf die Gefahr hingewiesen, dass private Unternehmen de facto zu Schiedsrichtern darüber werden, was online geäußert werden darf. Dies verstößt aber gegen die Grundrechte der Meinungsfreiheit und des Rechts auf Information. Um einer solchen Überreaktion entgegenzuwirken, behält der DSA zwei Kernprinzipien bei, die in der E-Commerce-Richtlinie festgelegt sind, dem rechtlichen Rahmen, der das europäische Internet seit dem Jahr 2000 regelt. Diese Kernprinzipien sind die beschränkte Haftung und das Verbot einer allgemeinen Überwachungsverpflichtung. Ersteres besagt, dass Plattformen nicht für nutzergenerierte Inhalte haftbar gemacht werden können, es sei denn, sie haben „tatsächliches Wissen“, dass der Inhalt illegal ist. Letzteres besagt, dass Plattformen nicht gezwungen werden können, ihrer Systeme pauschal zu überwachen, um illegale Inhalte zu identifizieren.

Der DSA führt weitere Sicherheitsvorkehrungen ein, um Nutzer vor der ungerechtfertigten Entfernung von Inhalten oder vor der Löschung des Kontos bei einer Plattform ohne triftigen Grund zu schützen. Die Anbieter müssen die Nutzer über die Entfernung von Inhalten informieren und den Grund dafür nennen. Große Plattformen müssen ein internes Beschwerdeverfahren anbieten, damit Nutzer die Entscheidung anfechten können. Kleine Unternehmen sind von dieser Verpflichtung ausgenommen. Große Plattformen müssen außerdem Streitbeilegungsmechanismen einrichten, die von einer unabhängigen Stelle überwacht werden.

Der DSA versucht, das Machtgleichgewicht zwischen Plattformen und Gesellschaft durch die Einführung von Transparenzregeln mehr in Richtung letzterer zu verschieben. Derzeit sammeln viele Plattformen viele Daten über ihre Nutzer, während die Nutzer ihrerseits sehr wenig darüber wissen, wie die Plattformen arbeiten. Durch den DSA müssen große Plattformen offener werden, was gezielte Werbung angeht. Die Nutzer müssen darüber informiert werden, welches Unternehmen oder welche Person die Werbung an sie richtet und warum sie dies tut. Große Plattformen müssen auch offener bezüglich ihrer Recomendation Engines (die berüchtigten Algorithmen) werden. Sie müssen den Nutzern die wichtigsten Parameter, die sie für die Empfehlung von Inhalten verwenden, klar erklären. Die Nutzer erhalten auch das Recht, diese Parameter zu ändern oder sich komplett aus dem Recommender-System abzumelden.

In ihrem Kampf um die Macht mit Big Tech fährt die EU schwere Geschütze auf. Strafen für Gesetzesverstöße können schwerwiegend sein.

Der Digital Markets Act

Der DMA richtet sich speziell an Gatekeeper und soll gleiche Wettbewerbsbedingungen auf dem digitalen Markt schaffen. Zu diesem Zweck schlägt das DMA vor, dass sehr große Online-Plattformen ihre eigenen Produkte nicht mehr denen der Wettbewerber bevorzugen dürfen (Google beispielsweise listet seine eigenen Dienste ganz oben in den Suchmaschinenergebnissen). Daten von Unternehmen sollen besser geschützt werden: Unternehmen müssen auf die Daten zugreifen dürfen, die sie auf der Plattform des Gatekeepers generiert haben. Sie müssen auch die Möglichkeit haben, ihre Daten aus dem Ökosystem herauszulösen, um sie an anderer Stelle zu hosten. Der Gatekeeper darf diese Daten nicht nutzen, um mit dem Unternehmen zu konkurrieren. Der Gatekeeper darf auch nicht mehr seinen Vermittlerstatus durchsetzen: Wenn Verbraucher und Unternehmen außerhalb der Plattform Kontakt aufnehmen wollen, darf der Gatekeeper dies nicht mehr verhindern.

In ihrem Machtkampf mit Big Tech fährt die EU die großen Geschütze auf. Die Strafen für Verstöße gegen das Gesetz können massiv sein. Gatekeeper, die sich nicht an die DSA halten, können mit einer Geldstrafe von bis zu 6 % ihres Jahresumsatzes belegt werden. Bei fortgesetzter Nichteinhaltung können sie gezwungen werden, alle Aktivitäten in den europäischen Netzwerken vorübergehend einzustellen. Die Bußgelder für Verstöße gegen das DMA sind mit 10 % des Jahresumsatzes noch höher. Fortgesetztes unseriöses Verhalten kann eine Zerschlagung des Unternehmens zur Folge haben.

Das Digital Services Act-Paket durchläuft derzeit das Gesetzgebungsverfahren der EU. Die EC, das Europäische Parlament EP und die Mitgliedsstaaten müssen dem endgültigen Text zustimmen, bevor er verabschiedet wird.

210177-02

EU-Flaggen vor dem Berlaymont-Gebäude der Europäischen Kommission. (Foto von Guillaume Périgois via Unsplash)

Hexadoku Sudoku für Elektroniker

Wie in jeder Ausgabe finden Sie auch in diesem Heft unser ganz spezielles Sudoku. PC, Oszilloskop und Lötkolben können sich erholen, während Ihre kleinen grauen Zellen auf Hochtouren arbeiten. Wenn Sie alle Hex-Ziffern in den grauen Kästchen herausgefunden haben, sollten Sie uns diese gleich zumailen – denn hier warten fünf Elektor-Gutscheine!

Die Regeln dieses Rätsels sind ganz einfach zu verstehen: Bei einem Hexadoku werden die Hexadezimalzahlen 0 bis F verwendet, was für Elektroniker und Programmierer ja durchaus passend ist. Füllen Sie das Diagramm mit seinen 16 x 16 Kästchen so aus, dass alle Hexadezimalzahlen von 0 bis F (also 0 bis 9 und A bis F) in jeder Reihe, jeder Spalte und in jedem Fach mit 4 x 4 Kästchen (markiert durch die dickeren schwarzen Linien) **genau einmal** vorkommen. Einige Zahlen sind bereits eingetragen, was die Ausgangssituation des Rätsels bestimmt. Wer das Rätsel löst – sprich die Zahlen in den grauen Kästchen herausfindet – kann einen von fünf Gutscheinen im Wert von 50 Euro gewinnen!

EINSENDEN

Schicken Sie die Lösung (die Zahlen in den grauen Kästchen) per E-Mail, Fax oder Post an:

Elektor Redaktion
Kackertstr. 10
52072 Aachen

Fax: 0241 / 955 09-013
E-Mail: hexadoku@elektor.de

Als Betreff bitte nur die Ziffern der Lösung angeben!

Einsendeschluss ist der 16. August 2021.

Die Gewinner des Hexadokus aus der Ausgabe Mai/Juni stehen fest!

Die richtige Lösung ist: **F9E18**

Aus allen Einsendungen mit der richtigen Lösung haben wir die fünf Gewinner eines Elektor-Wertgutscheins über je 50 € gezogen.

Die Namen der Gewinner werden unter www.elektormagazine.de/hexadoku bekannt gegeben.

Herzlichen Glückwunsch!

	4		F	2		0			6		1	7		5	
	3				1	7	8	0	C	A				2	
	A	0					B	5					E	3	
B		E	7			6			4			8	0		1
	B	2	5	0							7	E	C	8	
	F	6		B							5		D	1	
8					7					D					5
				5		F			2		9				
F	D		9	C	6					1	A	0		7	E
A		B	4			E			9			D	8		3
		C			3	4	0	6	5	7			B		
0		F	6									1	2		8
9						3	7	8	0						D
	1		B	D	0					F	2	6		C	
		D	8				2	A				3	9		

E	7	C	1	F	6	D	2	B	A	0	3	8	9	4	5
A	4	0	F	9	E	1	8	5	C	2	D	7	B	6	3
6	9	B	8	0	3	4	5	E	F	1	7	C	A	2	D
D	2	3	5	7	A	B	C	6	4	8	9	E	F	0	1
3	5	A	2	4	B	7	E	9	D	C	6	F	0	1	8
9	6	F	D	2	0	3	1	8	7	4	5	A	C	E	B
4	1	7	C	8	D	9	A	F	B	E	0	5	2	3	6
8	B	E	0	5	C	6	F	1	2	3	A	9	4	D	7
5	D	4	9	1	2	8	B	C	3	7	E	0	6	F	A
2	C	6	3	A	7	5	4	D	0	F	1	B	E	8	9
F	E	8	B	3	9	C	0	A	5	6	2	D	1	7	4
0	A	1	7	6	F	E	D	4	9	B	8	2	3	5	C
B	F	D	6	C	5	0	3	7	E	9	4	1	8	A	2
7	3	5	4	E	1	F	9	2	8	A	B	6	D	C	0
C	8	9	A	D	4	2	6	0	1	5	F	3	7	B	E
1	0	2	E	B	8	A	7	3	6	D	C	4	5	9	F

Jede Bewertung spiegelt ein persönliches Erlebnis wider

Wir lieben Elektronik und Projekte, und wir setzen alles daran, die Bedürfnisse unserer Kunden zu erfüllen

Der Elektor-Store: '**Never expensive, always surprising**'

Elektor Deutschland is rated **Excellent**

Based on 30 reviews

Trustpilot

Sehen Sie sich weitere Bewertungen auf unserer Trustpilot-Seite an: www.elektor.de/TP

Oder bilden Sie sich selbst eine Meinung und besuchen Sie unseren Elektor Store, www.elektor.de

elektor
design › share › sell

JETZT ABONNIEREN UND SIE ERHALTEN

WILLKOMMENSGESCHENK

ALLE 2 MONATE, EINE NEUE AUSGABE MIT DEN BESTEN ÜBER RASPBERRY PI

NUR 54,95 € PRO JAHR (6 AUSGABEN)

Mit einem MagPi Abo erhalten Sie:

- 6 gedruckte Ausgaben der MagPi
- ein Willkommensgeschenk
- Zugang zu dem Online Archiv mit allen MagPi Ausgaben seit 2016

Ihre Vorteile:

- 6 gedruckte Ausgaben der MagPi
- ein Willkommensgeschenk
- Zugang zu dem Online Archiv mit allen MagPi Ausgaben seit 2016
- Günstiger als Einzelkauf
- Jede Ausgabe direkt zu Ihnen nach Hause
- Alle Ausgaben auch als PDF
- Willkommensgeschenk mit tollen Inhalten

JETZT ABONNIEREN: WWW.MAGPI.DE