G3078E (D) € 14,90 | CHF 27,00 | (A, B, L) € 16,00

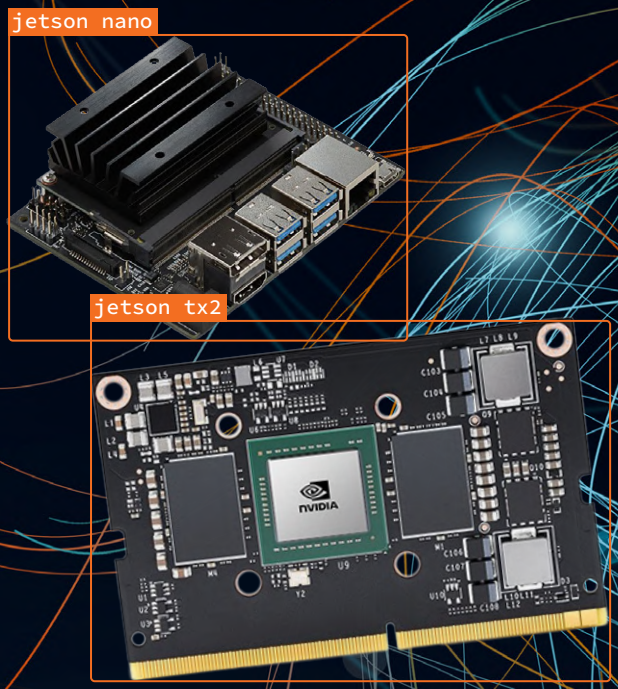
elektorMAG
design > share > earn

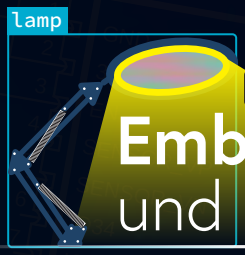
600
* SEIT 1961 *

MÄRZ/APRIL 2024
ELEKTORMAGAZINE.DE

AI-Spezialist

Machine Learning mit dem Jetson Nano



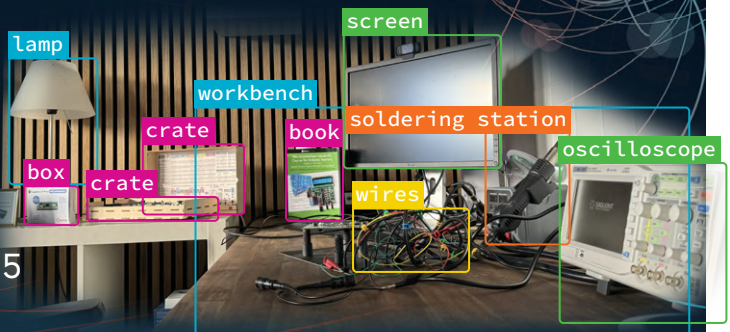


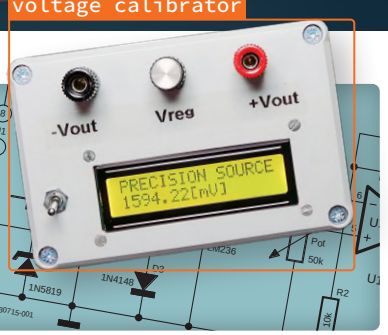
IM FOKUS
Embedded und KI

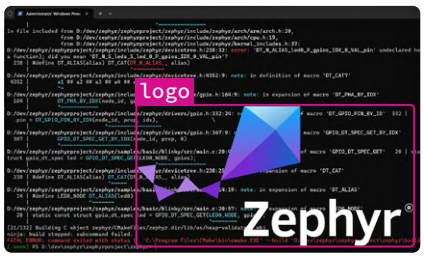
Ein intelligenter Objektzähler
Bilderkennung leicht gemacht mit Edge Impulse

ESP32-RS232-Adapter
Wireless-Anbindung klassischer Messgeräte

CaptureCount
Ein Objektdetektor und -zähler auf dem Raspberry Pi 5







4 198630 314904 40004

UNSER SORTIMENT
VON TECHNIKERN FÜR TECHNIKER

The best part of your project: www.reichelt.de

Nur das Beste für Sie – von über 1.500 Markenherstellern

Unsere Produktmanager sind seit vielen Jahren bei reichelt tätig und kennen die Anforderungen unserer Kunden. Sie stellen ein breites Spektrum an Qualitätsprodukten zusammen, optimal auf den Bedarf in Forschung & Entwicklung, Instandhaltung, IT-Infrastruktur und Kleinserienproduktion sowie auf Maker zugeschnitten.

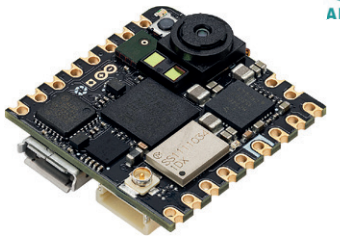
Robotik und Machine Learning

Die Kombination aus Robotik, Künstlicher Intelligenz und Machine Learning ermöglicht es Robotern komplexere Aufgaben autonom auszuführen und sich an neue Herausforderungen anzupassen. Finden Sie die passende Technologie in unserem stetig wachsenden Sortiment.

Arduino Pro Nicla Vision — überwachen - analysieren - erkennen

KI-Kameramodul mit Dual ARM CortexR M7 & M4

Nicla Vision ist eine einsatzbereite, eigenständige Kamera zur Analyse und Verarbeitung von Bildern on the Edge und eignet sich für Asset Tracking, Objekterkennung und vorausschauende Wartung.w



Bestell-Nr.:
ARD NIC VISION

98,40

ARM · DUAL CORE

Die neue Generation - Go2

Jetzt standardmäßig mit 4D ultra-wide LiDAR

Tauchen Sie ein in die faszinierende Welt der Robotik. Dieser fortschrittliche vierbeinige Roboter vereint modernste Technologie mit einer eleganten und agilen Bauweise und bringt die Zukunft der Robotik direkt in Ihre Hände.

- 30% verbesserte Motorleistung
- 150% erhöhte Ausdauer/Batteriekapazität
- ISS2.0 (ab Pro) Intelligentes Seitenführungssystem

Bestell-Nr.:
QR GO2 AIR

2.618,00

SCHLAUERE ROBOTER KÖNNEN MEHR!
ROBOTIK UND MACHINE LEARNING

Jetzt entdecken ▶
https://rch.lt/ml

NEUHEITEN IM SHOP
AUS ALLEN BEREICHEN DIE NEUSTEN ARTIKEL IM ÜBERBLICK!

Jetzt entdecken ▶
www.reichelt.de/neu

- Top Preis-Leistungs-Verhältnis
- über 130.000 ausgesuchte Produkte
- zuverlässige Lieferung – aus Deutschland in alle Welt

www.reichelt.de
Bestellhotline: +49 (0)4422 955-333

reichelt elektronik – The best part of your project

TAGESPREISE! Preisstand: 16. 2. 2024

IMPRESSUM

55. Jahrgang, Nr. 600
März/April 2024
ISSN 0932-5468

Das Elektor Magazin wird 8 Mal im Jahr herausgegeben von
Elektor Verlag GmbH
Lukasstraße 1, 52070 Aachen (Deutschland)
Tel. +49 (0)241 95509190
www.elektor.de | www.elektormagazine.de

Für alle Ihre Fragen
service@elektor.de

Mitglied werden
www.elektormagazine.de/abo

Anzeigen
Büsra Kas
Tel. +49 (0)241 95509178
busra.kas@elektor.com
www.elektormagazine.de/mediadaten

Druck
Senefelder Misset, Mercuriusstraat 35
7006 RK Doetinchem (Niederlande)

Distribution
IPS Pressevertrieb GmbH, Carl-Zeiss-Straße 5
53340 Meckenheim (Deutschland)
Tel. +49 (0)2225 88010

EDITORIAL

Jens Nickel

Chefredakteur ElektorMag

Setzen Sie sich Ziele!

In einem Editorial vor vielen Jahren habe ich einmal geschrieben, dass mich erst ein konkretes Ziel vor Augen dazu gebracht hatte, ambitioniert zu programmieren. Für unsere interne Verwaltung von Artikeln musste ich mich damals in VBA für Excel, MS Access, C#, HTML + Javascript und vieles mehr einarbeiten. Ich habe dabei unglaublich viel gelernt, was ich auch für meine Arbeit als Redakteur und diverse andere Projekte gebrauchen konnte. Mit dem trockenen Studium von Tutorials und einigen kleinen Übungsprojekten wäre ich niemals soweit gekommen. Ein paar Profis, die mein System evaluierten, waren dann auch verblüfft. Sie selbst hatten das Schreiben eines eigenen Redaktionssystems oft im Kopf gehabt, aber nie zu Ende gebracht. Und das Prinzip funktioniert auch „in Hardware". Wie viele von Ihnen wissen, bin ich auf Umwegen zur Elektronik gekommen. Ich habe dann auch gerne gebastelt und programmiert, doch das Platinendesign habe ich stets den Kollegen überlassen. Vor kurzer Zeit nahm ich mir mit einem Freund ein neues, größeres Elektronik-Projekt vor, nämlich die Fernsteuerung (und Fernvermessung) von gemoddeten Audio-Verstärkern. Und plötzlich war ich „gezwungen", mich mit KiCad, den verschiedensten Steckverbindern, der Programmierung von Touch-Displays und diversen Optionen von GitHub vertraut zu machen. Und das wird sicher noch nicht das Ende der Fahnenstange sein - in Kürze werde ich auf Elektor-Labs.com darüber berichten.
Was ich Ihnen damit sagen will? Setzen Sie sich Ziele, und wagen Sie sich auch mal an Dinge, die Ihnen auf den ersten Blick eine Nummer zu groß erscheinen. Sie werden getrieben sein vom Ehrgeiz, Ihre Idee zu verwirklichen; und damit bringen Sie genügend Eifer mit, sich auf neues Terrain vorzuwagen. Runden Sie Ihr Projekt zwischendrin einmal ab, so schlimm es auch noch aussieht und so holprig es noch läuft. Das Erfolgserlebnis wird Sie antreiben, das Ganze verbessern zu wollen.
Sie können gleich in diesem Heft beginnen, sich einmal mit der KI-gestützten Bildverarbeitung zu beschäftigen. Das hatten Sie schon lange vor? Prima! Es ist nämlich gar nicht so schwierig, eine eigene vorzeigbare Anwendung zu entwickeln (Seite 6 und Seite 86). Ein Tür-Öffner zu einer neuen Welt voller Ideen!

Veröffentlichen Sie bei Elektor!

Ihr Fachwissen über Elektronik ist willkommen: Schicken Sie uns Ihr Video, Ihren Artikelvorschlag oder Ihre Idee für ein Buch! Wir haben unseren Leitfaden für Autoren und Content Creator aktualisiert.
Alle Einzelheiten finden Sie unter:

www.elektormagazine.com/submissions

Elektor-Labs: Ideen und Projekte

Die Plattform Elektor Labs ist offen für jeden. Hier können Sie Ideen und Projekte zum Thema Elektronik veröffentlichen, technische Probleme diskutieren und mit anderen zusammenarbeiten.

www.elektormagazine.de/labs

AI-Spezialist

Machine Learning mit dem Jetson Nano

44

Rubriken

Hintergrund

Industry

CaptureCount
Ein Objektdetektor und -zähler auf dem Raspberry Pi 5

6

Bluetooth LE mit MAUI
Steuer-Apps für Android und Co. 30

ESP32-RS232-Adapter
Wireless-Anbindung klassischer Messgeräte 70

Projekte

Vorschau

Elektor Mai/Juni 2024
Das nächste Heft ist wie immer randvoll gefüllt mit Schaltungsprojekten, Grundlagen sowie Tipps und Tricks für Elektroniker. Schwerpunkt wird das Thema **Messen und Testen** sein.

- Radar-Detektor für menschliche Anwesenheit
- EKG-Monitor
- Gain-Phase-Analysator mit Soundkarten-Schnittstelle
- Reparatur elektronischer Geräte
- DDS-Signalgenerator
- In-Circuit-LC-Messgerät
- Power-Meter-Interface (5 A / 60 V)
- Aktive Stroboskopscheibe für Plattenspieler

Elektor Mai/Juni 2024 erscheint am 15. Mai 2024.

Änderungen vorbehalten!

Spannungsreferenz mit Arduino Pro Mini
Linearisieren und kalibrieren Sie analoge Eingänge 14

IM FOKUS

Embedded und KI

CaptureCount

Ein Objektdetektor und -zähler auf dem Raspberry Pi 5

Von Saad Imtiaz (Elektor)

Maschinelles Sehen ist eine großartige Technologie, die die Erkennung und Klassifizierung einer großen Anzahl von Objekten in unserer Umgebung ermöglicht. Dieser Artikel stellt ein faszinierendes Raspberry-Pi-Projekt vor, das das YOLO-Objekterkennungsmodell verwendet. Das Modell ist in der Lage, eine Vielzahl von Objekten zu erkennen, von gewöhnlichen Gegenständen wie Autos und Fahrrädern bis hin zu verschiedenen Tieren wie Katzen, Hunden und Vögeln.

Bild 1. CaptureCount läuft auf einem Raspberry Pi 5 mit dem Camera Module 3 Wide von Raspberry Pi.

Das Projekt begann mit einem klaren Ziel: ein vielseitiges Objekterkennungs- und -zählsystem zu entwickeln, das die Art des Objekts erkennt und dann die Objekte jeder Kategorie zählt, die es erkennt. Der Raspberry Pi war aufgrund seiner Fähigkeiten und seiner Unterstützung von KI-Projekten eine naheliegende Wahl. Da dies eines meiner ersten Projekte im Bereich der Computer Vision war, brauchte ich einige Zeit, um Python zu lernen und die richtigen Tools und Bibliotheken zu verwenden, um dieses Projekt zu entwickeln. Da es so viele Projekte, Ressourcen [1][2] und unser geliebtes ChatGPT gibt, war es aber nicht schwierig, ein solches Projekt zu verwirklichen.

Objekterkennung

Der erste Schritt zur Entwicklung dieses Projekts war die Suche nach dem richtigen Objekterkennungsmodell, vorzugsweise ein Modell, das eine Vielzahl von Objekten mit minimaler Trainingszeit erkennen kann. Doch bevor wir uns auf die Suche nach einem solchen Objekterkennungsmodell machen, sollten wir einen Schritt zurückgehen und darüber sprechen, wie Objekterkennung funktioniert.

Die Objekterkennung ermöglicht es Computern, Objekte in einem Bild oder Video zu identifizieren und zu lokalisieren. Im Gegensatz zur Bilderkennung, die Ihnen sagt, was sich in einem Bild befindet, geht die Objekterkennung weiter, indem sie genau feststellt, wo sich diese Objekte befinden und sie umreisst. Dies wird in der Regel durch zwei Schlüsselprozesse erreicht:

- Objekt-Lokalisierung: Bestimmt den Standort eines einzelnen Objekts in einem Bild. Das Modell gibt die Koordinaten der Bounding Box (Begrenzungsrahmen) aus, die das Objekt umgibt.
- Objekt-Klassifizierung: Identifiziert das Objekt in der Bounding Box aus einer Reihe bekannter Kategorien.

Fortschrittliche Objekterkennungsmodelle vereinen diese beiden Schritte und verarbeiten ein Bild effizient, um sowohl die Position als auch die Klassifizierung mehrerer Objekte darin zu ermitteln.

Erforschung potentieller Objekterkennungsmodelle

Auf der Suche nach dem idealen Objekterkennungsmodell wurden verschiedene Optionen mit ihren jeweiligen Stärken und Beschränkungen miteinander verglichen. Modelle wie R-CNN und seine Derivate boten eine hohe Genauigkeit, allerdings bei einer für Echtzeitanwendungen zu langsamen Verarbeitungsgeschwindigkeit. Der Single Shot Multibox Detector (SSD) stellte eine schnellere Alternative dar, allerdings mit Einbußen bei der Genauigkeit. Mask R-CNN bot eine präzise Erkennung, war aber für unsere Bedürfnisse zu komplex. Letztendlich fiel die Wahl auf YOLO (You Only Look Once), weil es einen optimalen Kompromiss zwischen Geschwindigkeit, Effizienz und passabler Genauigkeit bietet. Die Fähigkeit, Bilder in Echtzeit zu verarbeiten, seine Einfachheit und die starke Unterstützung durch die KI-Community machten YOLO zur idealen Wahl für die Ziele unseres Projekts.

Die Auswahl des YOLO-Modells war ein entscheidender Punkt in diesem Projekt. YOLO von Joseph Redmon [3] ist bekannt für seine Fähigkeit, Objekte in Echtzeit zu erkennen - eine entscheidende Funktion für jedes Erkennungssystem. Was mich an YOLO reizte, war der einzigartige Ansatz zur Objekterkennung.
Im Gegensatz zu herkömmlichen Methoden, die ein Bild in mehreren Schritten verarbeiten, macht YOLO dies in einem einzigen Durchgang. Dies beschleunigt den Prozess und verbessert die Fähigkeit des Modells, aus dem Training heraus zu verallgemeinern, was die Erkennung von Objekten in verschiedenen und unvorhersehbaren realen Umgebungen effektiver macht. Diese Fähigkeit war für dieses Projekt von entscheidender Bedeutung, da es darum ging, eine Vielzahl von Objekten zu erkennen.

Hardware-Software-Integration

Für das Projekt wurden der Raspberry Pi 5, der für seine hohe Rechenleistung bekannt ist, und ein kompatibles Kameramodul benötigt. Der Integrationsprozess umfasste die Einrichtung des Raspberry-Pi-OS, den Anschluss der Kamera und die Installation von YOLO. Um mit dem Raspberry Pi loszulegen, muss zunächst ein Betriebssystem auf seiner microSD-Karte installiert werden. Dazu laden Sie einfach den Raspberry-Pi-Imager von der offiziellen Website des Raspberry Pi herunter [4]. Führen Sie dann den Imager aus, wählen Sie das richtige Gerät, wählen Sie die neueste Version des Raspberry-Pi-OS (64 Bit), wählen Sie die microSD-Karte und klicken Sie schließlich einfach auf *Next*. Innerhalb weniger Minuten wird das Raspberry-Pi-OS auf der microSD-Karte installiert. Stecken Sie anschließend die SD-Karte in den microSD-Kartensteckplatz des Raspberry Pi und schalten Sie ihn ein. Sie benötigen außerdem einen Monitor, ein Mini-HDMI-auf-HDMI-Kabel, eine Tastatur und eine Maus, um zum ersten Mal mit dem Raspberry Pi zu interagieren. Nachdem Sie einen VNC-Server oder SSH eingerichtet haben, können Sie den Raspberry Pi auch über eine WLAN- oder Ethernet-Verbindung von Ihrem Computer aus fernsteuern.
Nun ist es an der Zeit, die erforderlichen Bibliotheken für dieses Projekt zu installieren. Die Installation der Bibliotheken ist einfach; sie erfolgt über das Terminal. Vor der Installation neuer Bibliotheken wird empfohlen, die Systempakete zu aktualisieren, durch Eingabe von:

```
sudo apt-get update && sudo apt-get upgrade
```

Dies kann einige Zeit in Anspruch nehmen. Danach können die Bibliotheken, die wir für das Projekt benötigen, mit den folgenden Befehlen installiert werden:

› Installieren von Image I/O-Paketen

```
sudo apt-get install libjpeg-dev libtiff5-dev
  libjasper-dev libpng12-dev
```

› Einrichten von Video-I/O-Paketen und der GTK-Bibliothek

```
sudo apt-get install libavcodec-dev libavformat-dev
  libswscale-dev libv4l-dev
sudo apt-get install libxvidcore-dev libx264-dev
sudo apt-get install libgtk2.0-dev
```

› Zusätzliche Abhängigkeiten für OpenCV

```
sudo apt-get install libatlas-base-dev gfortran
```

› Installieren von pip und pipx (Paketverwaltung)

```
sudo apt-get install python3-pip
sudo apt install pipx -y
```

› Installieren von Numpy, Pandas und Open CV

```
pip install numpy
pip install pandas
sudo apt install python3-opencv
```

Nach der Installation aller Bibliotheken geht es nun darum, eine nahtlose Kommunikation zwischen der Hardware und der Software zu gewährleisten, damit der Raspberry Pi die Live-Kamerafeeds effektiv verarbeiten kann. Um die Dinge einfach zu halten, wurde das Raspberry-Pi-Kameramodul an den MIPI-Kameraport des Raspberry Pi 5 angeschlossen. Jetzt ist alles eingerichtet, also können wir den Code des Projekts besprechen.

In den Code eintauchen

Der Kern dieses Projekts liegt in seinem Python-Code, der auf GitHub [4] verfügbar ist. Das Skript beginnt mit dem Import wichtiger Bibliotheken wie OpenCV für die Bildverarbeitung, Pandas für die Datenverarbeitung und Numpy für numerische Operationen.

```
import cv2
import pandas as pd
import numpy as np
import subprocess
import os
from datetime import datetime
```

Zu Beginn importiert das Skript wichtige Bibliotheken. *cv2* (OpenCV) ist entscheidend für Bildverarbeitungsaufgaben. *pandas* und *numpy* sind für die Datenbearbeitung beziehungsweise für numerische Berechnungen zuständig. *subprocess* und *os* sind Standard-Python-Bibliotheken für die Interaktion mit dem System, zum Beispiel für die Ausführung externer Befehle und die Handhabung von Dateipfaden. *datetime* wird für die Zeitstempelvergabe von Erkennungen verwendet.

```
model = './yolo/yolov3.weights'
config = './yolo/yolov3.cfg'
net = cv2.dnn.readNetFromDarknet(config, model)
```

Das Skript gibt die Pfade zu den vortrainierten Gewichtungen und der Konfigurationsdatei von YOLO an und lädt sie dann mit dem DNN-Modul (Deep Neural Network) von OpenCV. Dieser Schritt initialisiert das YOLO-Modell für die Objekterkennung.

```
classes = []
with open("./yolo/coco.names", "r") as f:
    classes = [line.strip() for line in f.readlines()]
```

Die Datei *coco.names*, die Namen von Objekten enthält, die YOLO erkennen kann, wird Zeile für Zeile gelesen, um eine Liste von Klassennamen

zu erstellen. Mehrere Funktionen werden definiert, um den Erkennungsprozess zu rationalisieren. Schauen wir uns in **Listing 1** die anfänglich festgelegten Parameter und Funktionen für die Hauptschleife an.
Die Funktion `get_output_layers(net)` extrahiert die Namen der Ausgabeschichten des YOLO-Netzes. Die Architektur von YOLO kennt mehrere Ausgabeschichten, und diese Funktion hilft, sie für die Verarbeitung der Erkennungen zu identifizieren.
`draw_bounding_box(...)` zeichnet Rechtecke (Bounding Boxes) um erkannte Objekte und beschriftet sie mit dem Namen des Objekts und dem Zuverlässigkeitswert.
`calculate_centroid(x, y, w, h)` berechnet den Mittelpunkt der Bounding Box des erkannten Objekts, ein wichtiger Schritt für die Verfolgung von Objekten über mehrere Bilder hinweg.

```
def capture_image(image_path):
    subprocess.run(["libcamera-still", "-o", image_path,
"--width", "1920", "--height", "1080", "-n", "-t", "1"],
stdout=subprocess.DEVNULL, stderr=subprocess.STDOUT)
```

Diese Funktion verwendet das Modul `subprocess`, um *libcamera-still* auszuführen, ein Kommandozeilentool auf dem Raspberry-Pi-OS, das ein Bild von der Kamera aufnimmt und es im angegebenen Pfad speichert. Die Auflösung ist auf 1.920×1.080 Pixel voreingestellt. Um die CPU- und GPU-Auslastung zu minimieren, kann auch eine niedrigere Auflösung verwendet werden. Zu diesem Zweck kann das obige Codefragment wie folgt angepasst werden:

```
def capture_image(image_path):
    subprocess.run(["libcamera-still", "-o", image_path,
"--width", "1280", "--height", "720", "-n", "-t", "1"],
stdout=subprocess.DEVNULL, stderr=subprocess.STDOUT)
```

Vor dem Sprung in die Haupterkennungsschleife initialisiert das Skript einen Pandas `DataFrame`, um Erkennungen mit Zeitstempeln zu versehen, und ein Dictionary, um die Anzahl der einzelnen Objekttypen zu verfolgen. Mit `centroid_tracking` werden die Bewegungen der Objekte verfolgt.

```
data_frame = pd.DataFrame(columns=['Timestamp', 'Type',
  'Count'])
object_counts = {cls: 0 for cls in classes}
centroid_tracking = {}
```

Der folgende Teil des Codes (siehe **Listing 2**) prüft, ob das erkannte Objekt neu ist oder bereits zuvor identifiziert wurde. Er verwendet `centroid_tracking`, um dies festzustellen. Wenn ein Objekt als neu identifiziert wird, wird die Anzahl für diesen Objekttyp erhöht, die Erkennungszeit protokolliert und ein Pandas `DataFrame` aktualisiert. Dieser `DataFrame` kann später zur weiteren Analyse als *.csv*-Datei exportiert werden.

Einrichten der Haupterkennungsschleife

Die Haupterkennungsschleife des Raspberry Pi 5 und des YOLO-Systems ist ein wichtiger Teil seiner Funktionalität. Sie beginnt mit der Aufnahme eines Bildes vom Raspberry-Pi-Kameramodul, das in ein für die YOLO-Verarbeitung geeignetes Format konvertiert wird. Das System verarbeitet dann dieses Bild durch YOLO und extrahiert wichtige Informationen wie Klassenzugehörigkeit, Zuverlässigkeitswerte und die Bounding-Box-Koordinaten.
Um die Erkennungen zu verfeinern, wird die Non-Maximum Suppression (NMS) [5] angewendet, die weniger sichere und überlappende Erkennungen herausfiltert. Das System zeichnet dann Bounding Boxes um die erkannten Objekte und speichert diese Bilder, wenn neue Objekte erkannt werden. Während dieses Prozesses aktualisiert das System die Tracking- und Logging-Datenstrukturen und führt so eine kontinuierliche Aufzeichnung der Erkennungen durch. Diese Schleife ist das Herzstück der Echtzeit-Objekterkennungsfunktionen des Raspberry Pi 5 und des YOLO-Systems. Siehe **Listing 3** für die Hauptschleife des vollständigen Codes.
Nach der Verarbeitung der Erkennungen kompiliert das Skript die Daten in einen DataFrame und speichert sie als *.csv*-Datei. Dies liefert eine detaillierte Aufzeichnung jedes Erkennungsereignisses.

CaptureCount in Aktion

In der Praxis zeigte CaptureCount eine gute Objekterkennung mit einigen faszinierenden Eigenheiten. Der Raspberry Pi wurde auf das Kamerastativ montiert und die Kamera blickte aus dem Fenster, wie in **Bild 2** zu sehen ist. Wie Sie sehen können, wohne ich in einem urbanen Teil der Stadt, und die einzigen Dinge, die CaptureCount erkennt, sind Autos, Lastwagen, Menschen und Motorräder. In einer ländlichen Umgebung wäre es viel besser gewesen, denn dann hätte ich auch einige Wildtiere aufgenommen.
Der Großteil der Leistung zeichnete sich durch eine bewundernswerte 80-prozentige Trefferquote bei der Identifizierung und Kategorisierung

Bild 2. Die Aussicht der Kamera.

Bild 3. Zwei von CaptureCount erkannte Autos.

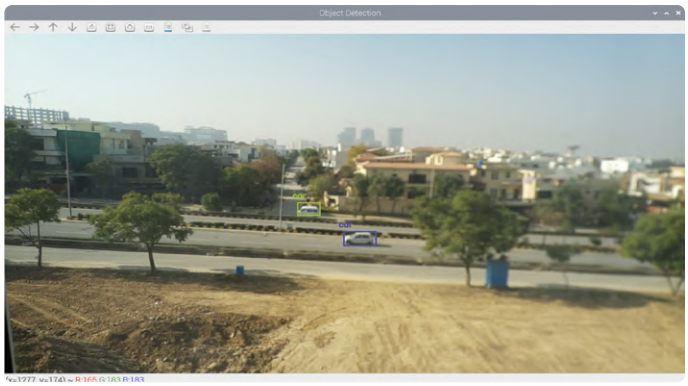


Bild 4. Bild der vom System erkannten Autos.

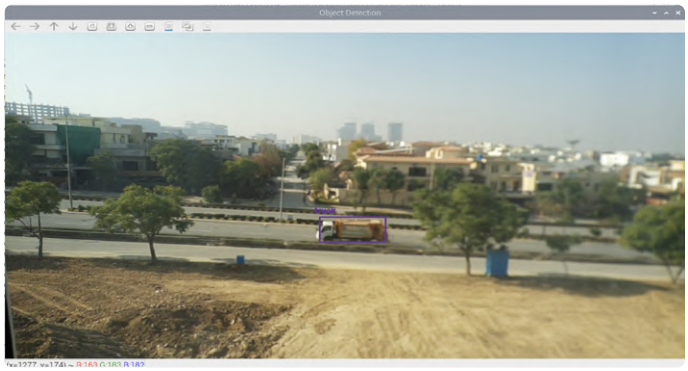


Bild 5. Lkw, der vom System erkannt wird.

verschiedener Objekte aus. Dieser Präzisionsgrad ist besonders bemerkenswert, wenn man bedenkt, dass die Kamera relativ weit von den Objekten entfernt war und ein Weitwinkelobjektiv verwendet wurde. **Bild 3**, **Bild 4** und **Bild 5** zeigen die erkannten Objekte, in **Bild 6** und **Bild 7** ist das gesamte Protokoll in einer *.csv*-Datei dargestellt, aus der hervorgeht, wann welches Objekt erkannt wurde und wie viele Objekte derselben Kategorie in verschiedenen Zeitintervallen erkannt wurden.

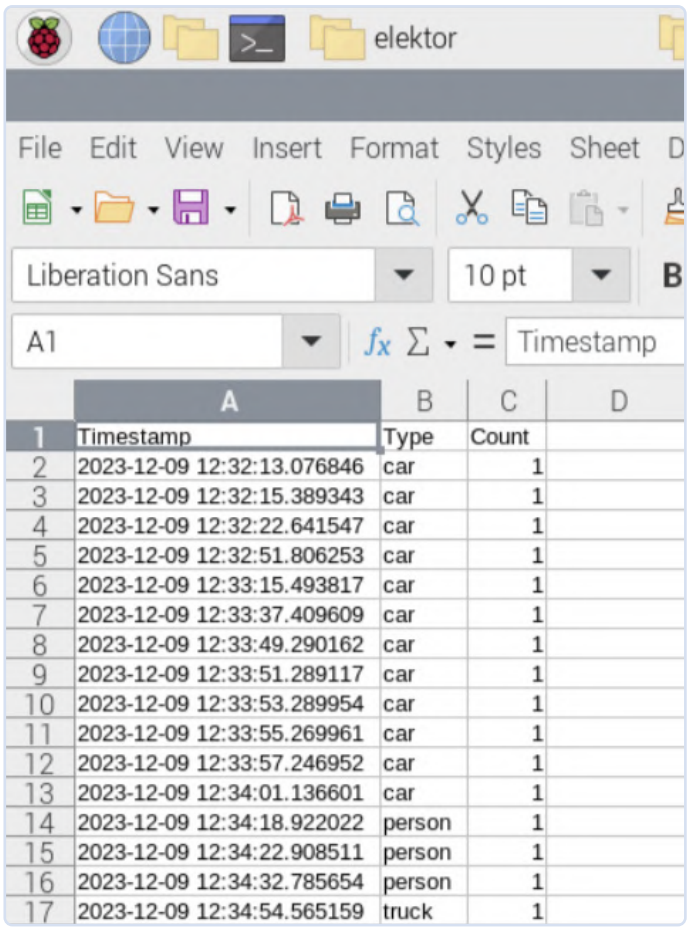
Besondere Beobachtungen

Trotz all dieser Erfolge stand das System vor einigen interessanten Herausforderungen:

- Selektive Erkennung in dynamischen Umgebungen: Es wurden Fälle festgestellt, in denen das System eine Person auf einem Motorrad erkannte, das Motorrad selbst aber nicht erkannte. Diese selektive Erkennung machte deutlich, dass es schwierig ist, eng zusammenhängende Objekte zu unterscheiden, insbesondere wenn sie in Bewegung sind.
- Gelegentliche Fehlklassifizierungen: Es gab seltene Fälle, in denen Fußgänger fälschlicherweise als Fahrräder klassifiziert wurden. Diese Fehleinstufung verdeutlicht die Komplexität der Objektklassifizierung, insbesondere wenn Objekte ähnliche räumliche Merkmale aufweisen.

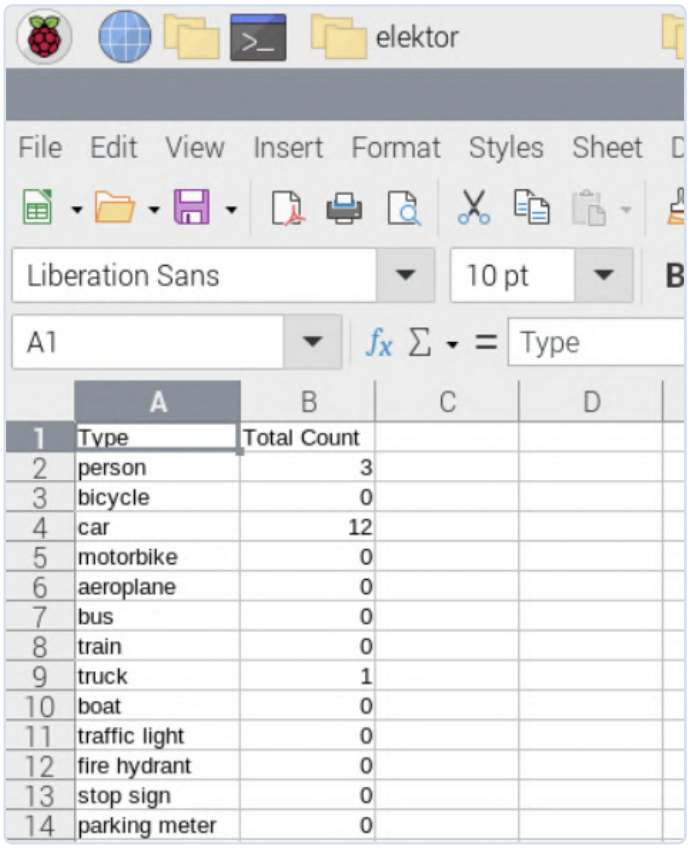
Ein Weg in die Zukunft für CaptureCount

Zusammenfassend lässt sich sagen, dass dieses Projekt, bei dem der Raspberry Pi 5 und YOLO zum Einsatz kamen, eine bemerkenswerte Leistung gezeigt hat, die Türen für verschiedene Erweiterungen und Anwendungen öffnet. Die gelegentlichen Erkennungsanomalien und Fehlklassifizierungen des Systems unterstreichen nicht nur Bereiche für zukünftige Verbesserungen, sondern eröffnen auch Möglichkeiten für weitere Forschung.
Zukünftige Erweiterungen könnten die Integration von Servomotoren



	A	B	C	D
1	Timestamp	Type	Count	
2	2023-12-09 12:32:13.076846	car	1	
3	2023-12-09 12:32:15.389343	car	1	
4	2023-12-09 12:32:22.641547	car	1	
5	2023-12-09 12:32:51.806253	car	1	
6	2023-12-09 12:33:15.493817	car	1	
7	2023-12-09 12:33:37.409609	car	1	
8	2023-12-09 12:33:49.290162	car	1	
9	2023-12-09 12:33:51.289117	car	1	
10	2023-12-09 12:33:53.289954	car	1	
11	2023-12-09 12:33:55.269961	car	1	
12	2023-12-09 12:33:57.246952	car	1	
13	2023-12-09 12:34:01.136601	car	1	
14	2023-12-09 12:34:18.922022	person	1	
15	2023-12-09 12:34:22.908511	person	1	
16	2023-12-09 12:34:32.785654	person	1	
17	2023-12-09 12:34:54.565159	truck	1	

Bild 6. Screenshot der .csv-Datei, die die erkannten Objekte mit ihren jeweiligen Zeitstempeln zeigt.



	A	B	C	D
1	Type	Total Count		
2	person	3		
3	bicycle	0		
4	car	12		
5	motorbike	0		
6	aeroplane	0		
7	bus	0		
8	train	0		
9	truck	1		
10	boat	0		
11	traffic light	0		
12	fire hydrant	0		
13	stop sign	0		
14	parking meter	0		

Bild 7. Screenshot der .csv-Datei, die die Gesamtsumme der in einer einzelnen Kategorie erkannten Objekte anzeigt.

zur gezielten Objektverfolgung und IoT-Konnektivität für automatische Reaktionen umfassen. Nur ein Beispiel: Das Blinken einer RGB-LED für eine bestimmte Art von Objekt. Anregungen, wie Sie Ihr Projekt mit dieser oder einer anderen Hardware-Steuerung erweitern können, finden Sie in **Listing 4**. Diese Funktion schaltet die rote LED für ein Auto ein, die grüne für eine Person und die rote und grüne (und damit gelbe) für einen Lastwagen. Nach einer Verzögerung schaltet sie alle LEDs aus. Darüber hinaus können Modifikationen für erweiterte Anwendungen in den Bereichen hochentwickelte Überwachung von Menschen und (Wild-) Tieren, Verkehrs- und Menschenmengenmanagement, Einzelhandelsanalyse und Gesundheitswesen vorgenommen werden. Die Fähigkeit dieses Systems zur präzisen Objekterkennung in Echtzeit bildet die Grundlage für innovative Lösungen in verschiedenen Branchen und zeigt die weitreichenden Möglichkeiten im Bereich der KI und der Computer Vision.

SG — 230749-02

Haben Sie Fragen oder Kommentare?

Wenn Sie technische Fragen oder Kommentare zu diesem Artikel haben, wenden Sie sich bitte per E-Mail an den Autor unter saad.imtiaz@elektor.com oder an das Elektor-Redaktionsteam unter redaktion@elektor.de.

Über den Autor

Saad Imtiaz (Senior Engineer, Elektor) ist ein Mechatronik-Ingenieur mit Erfahrung in eingebetteten Systemen, mechatronischen Systemen und Produktentwicklung. Er hat mit zahlreichen Unternehmen, von Start-ups bis hin zu globalen Konzernen, bei der Prototypenerstellung und Entwicklung zusammengearbeitet. Saad hat auch einige Zeit in der Luftfahrtindustrie verbracht und ein Technologie-Start-up-Unternehmen geleitet. Bei Elektor treibt er die Projektentwicklung sowohl im Bereich Software als auch Hardware voran.

Passende Produkte

- **Ultimatives Starterkit für den Raspberry Pi 5 (8 GB)**
  www.elektor.de/20721
- **Hochwertiges Kameramodul für den Raspberry Pi**
  www.elektor.de/19279
- **Camera Module 3 Wide von Raspberry Pi**
  www.elektor.de/20364

WEBLINKS

[1] Vereinfachte Implementierung von YOLOv3: https://analyticsvidhya.com/blog/2021/06/implementation-of-yolov3-simplified
[2] YOLOv3-Code erklärt: https://pylessons.com/YOLOv3-code-explanation
[3] YOLO: Objekterkennung in Echtzeit: https://pjreddie.com/darknet/yolo
[4] CaptureCount Github-Repository: https://github.com/ElektorLabs/CaptureCount
[5] NMS (Wikipedia): https://de.wikipedia.org/wiki/Canny-Algorithmus#Non-maximum_suppression

Listing 1: Erkennungsfunktionen

```
import cv2
import pandas as pd
import numpy as np
import subprocess
import os
from datetime import datetime

# Load pre-trained model and configuration
model = './yolo/yolov3.weights'  # Update this path to your model's path
config = './yolo/yolov3.cfg'     # Update this path to your config file's path
net = cv2.dnn.readNetFromDarknet(config, model)

# Load class names
classes = []
with open("./yolo/coco.names", "r") as f:  # Update to the path of your coco.names file
    classes = [line.strip() for line in f.readlines()]

# Function to get output layers
def get_output_layers(net):
    layer_names = net.getLayerNames()
    return [layer_names[i - 1] for i in net.getUnconnectedOutLayers().flatten()]
```

```
# Function to draw bounding box
def draw_bounding_box(img, class_id, confidence, x, y, x_plus_w, y_plus_h):
    label = str(classes[class_id])
    color = np.random.uniform(0, 255, size=(3,))
    cv2.rectangle(img, (x, y), (x_plus_w, y_plus_h), color, 2)
    cv2.putText(img, label, (x-10, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)

# Function to calculate centroid of a bounding box
def calculate_centroid(x, y, w, h):
    return (int(x + w/2), int(y + h/2))

# Function to capture image using libcamera-still
def capture_image(image_path):
    subprocess.run(["libcamera-still", "-o", image_path, "--width", "1920", "--height", "1080", "-n", "-t",
                                                "1"], stdout=subprocess.DEVNULL, stderr=subprocess.STDOUT)

# Initialize a DataFrame to store counts
data_frame = pd.DataFrame(columns=['Timestamp', 'Type', 'Count'])
object_counts =  # Object count per category
centroid_tracking = {}  # Tracks centroids of detected objects

# Ensure output directory exists
output_dir = "output"
if not os.path.exists(output_dir):
    os.makedirs(output_dir)
```

Listing 2: Centroid Tracking — erkennen, ob das Objekt neu oder bereits erkannt ist

```
# Check if this object was already detected
if class_id not in centroid_tracking or not any(np.linalg.norm(np.array(centroid) - np.array(old_centroid))
                                                < 50 for old_centroid in centroid_tracking[class_id]):
    object_counts[classes[class_id]] += 1
    centroid_tracking.setdefault(class_id, []).append(centroid)
    new_row = pd.DataFrame([{'Timestamp': datetime.now(), 'Type': classes[class_id], 'Count': 1}])
    data_frame = pd.concat([data_frame, new_row], ignore_index=True)
```

Listing 3: Hauptschleife

```
# Main loop
image_path = "temp.jpg"
object_id = 0  # Unique identifier for each object

try:
    while True:
        capture_image(image_path)
        frame = cv2.imread(image_path)
        if frame is None:
            continue

        Width = frame.shape[1]
        Height = frame.shape[0]
        scale = 0.00392
```

(Fortsetzung auf der nächsten Seite)

```
        # Create a blob and pass it through the model
        blob = cv2.dnn.blobFromImage(frame, scale, (416, 416), (0, 0, 0), True, crop=False)
        net.setInput(blob)
        outs = net.forward(get_output_layers(net))

        # Process the outputs
        class_ids = []
        confidences = []
        boxes = []
        centroids = []
        conf_threshold = 0.5
        nms_threshold = 0.4

        for out in outs:
            for detection in out:
                scores = detection[5:]
                class_id = np.argmax(scores)
                confidence = scores[class_id]
                if confidence > conf_threshold:
                    center_x = int(detection[0] * Width)
                    center_y = int(detection[1] * Height)
                    w = int(detection[2] * Width)
                    h = int(detection[3] * Height)
                    x = center_x - w / 2
                    y = center_y - h / 2
                    class_ids.append(class_id)
                    confidences.append(float(confidence))
                    boxes.append([x, y, w, h])
                    centroids.append(calculate_centroid(x, y, w, h))

        indices = cv2.dnn.NMSBoxes(boxes, confidences, conf_threshold, nms_threshold)

        for i in indices:
            box = boxes[i]
            x, y, w, h = box
            x, y, w, h = int(x), int(y), int(w), int(h)  # Ensure the values are integers

            centroid = centroids[i]
            class_id = class_ids[i]

            # Check if this object was already detected
            if class_id not in centroid_tracking or not any(np.linalg.norm(np.array(centroid) -
                            np.array(old_centroid)) < 50 for old_centroid in centroid_tracking[class_id]):
                object_counts[classes[class_id]] += 1
                centroid_tracking.setdefault(class_id, []).append(centroid)


                # Update data_frame using pandas.concat
                new_row = pd.DataFrame([{'Timestamp': datetime.now(),'Type': classes[class_id],'Count': 1}])
                data_frame = pd.concat([data_frame, new_row], ignore_index=True)

                # Save the entire frame when an object is detected
                frame_filename = os.path.join(output_dir, f"frame__.jpg")
                cv2.imwrite(frame_filename, frame)
                object_id += 1

            draw_bounding_box(frame, class_id, confidences[i], round(x), round(y), round(x+w), round(y+h))

        # Display the frame
        cv2.imshow("Object Detection", frame)

        # Break loop with 'q' key
```

```
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
finally:
    cv2.destroyAllWindows()
    if os.path.exists(image_path):
        os.remove(image_path)

# Print and save the total count of objects detected per category
total_counts = pd.DataFrame(object_counts.items(), columns=['Type', 'Total Count'])
print(total_counts)
total_counts.to_csv("total_object_counts.csv", index=False)

# Save detailed counts to CSV
data_frame.to_csv("object_counts.csv", index=False)

# Write the total count to a text log
with open("total_counts_log.txt", "w") as log_file:
    log_file.write(str(total_counts))
```

Listing 4: RGB-LED blinkt bei Objekterkennung

```
#include these libraries in the shared code
import RPi.GPIO as GPIO
import time

# GPIO pin setup
RED_PIN, GREEN_PIN, BLUE_PIN = 17, 27, 22

# Update with your GPIO pin numbers

# add this part in the initial part of the code
GPIO.setmode(GPIO.BCM)
GPIO.setup([RED_PIN, GREEN_PIN, BLUE_PIN], GPIO.OUT)

def blink_rgb_led(object_type):
    GPIO.output(RED_PIN, object_type == 'car' or object_type == 'truck')
    GPIO.output(GREEN_PIN, object_type == 'person' or object_type == 'truck')
    GPIO.output(BLUE_PIN, False)
    time.sleep(1)
    GPIO.output([RED_PIN, GREEN_PIN, BLUE_PIN], False)

# Usage of this function:

# ... [Previous code] ...
for out in outs:
    for detection in out:
# ... [add the following lines to the code to this 'for' loop]
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if confidence > conf_threshold:
            detected_object = classes[class_id]
             blink_rgb_led(detected_object) # Blink the LED based on the detected object

# ... [Rest of your detection and saving logic] ...
# ... [Remaining code] ...
```

Spannungsreferenz mit Arduino Pro Mini

Linearisieren und kalibrieren Sie analoge Eingänge

Von Giovanni Carrera (Italien)

Wenn Sie die analogen Eingänge Ihres bevorzugten Mikrocontrollers verwenden, müssen Sie früher oder später deren Linearität und Genauigkeit über den gesamten Bereich der möglichen Eingangswerte überprüfen. Dieser präzise und kompakte Spannungskalibrator auf Arduino-Basis ist in der Lage, stabile und genaue Referenzspannungen im Bereich von 0…5 V zu erzeugen und sie ebenso präzise zu messen.

Ein Spannungskalibrator ist ein Instrument, das bekannte Spannungen mit hoher Genauigkeit ausgibt und zum Überprüfen der Qualität und zum Kalibrieren von Voltmetern, ADC- und DAC-Wandlern, Erfassungssystemen und so weiter verwendet wird.

Bei Spannungsmessungen mit Mikrocontrollern ist es oft notwendig, die Wandlungskonstante zu kennen und die Linearität des integrierten Analog-Digital-Wandlers zu überprüfen. Wenn wir uns mit einem Fehler von etwa 5 % zufrieden geben, reicht es aus, die U_{ref} durch den maximalen Wert am Ausgang zu dividieren. Beim Arduino UNO wäre dieser Wert zum Beispiel $2^{10} - 1 = 1023$. Wenn wir die Versorgungsspannung (5 V) als U_{ref} verwenden, kann diese Referenz aber variieren, je nachdem, ob wir das Board über USB oder eine andere Spannungsquelle versorgen, und ist außerdem verrauscht. Wenn wir die interne U_{ref} verwenden (was definitiv vorzuziehen ist), zum Beispiel 1,1 V, kennen wir ihren exakten Wert aber nicht wirklich. Andere Controller (-Boards) wie der ESP32 verfügen nicht einmal über Analogeingänge, die bei 0 V beginnen, sondern irgendwo bei 80…150 mV, und der Skalenendwert liegt dann zwischen 950 mV und 1100 mV.

Um diese Ungenauigkeiten zu überwinden, müssen wir eine Kalibrierung durchführen, also eine stabile und rauscharme Spannungsquelle zusammen mit einem guten Digitalvoltmeter mit (viel) höherer Genauigkeit als der Wandler des Controllers verwenden. Dann führen wir etwa ein Dutzend Messungen verschiedener Werte innerhalb des Eingangsbereichs des ADCs durch, geben die Werte in eine Excel-Tabelle ein und führen eine lineare Regression durch, um die Steigung und den Achsenabschnitt abzuleiten. Wir benötigen diese Werte, um eine Ausgabe in Volt oder Millivolt zu erhalten. Dieses Projekt kombiniert die beiden benötigten Instrumente: einen sehr stabilen, rauscharmen Spannungsgenerator und ein extrem genaues Digitalvoltmeter mit einem Messbereich von 0…5 V und einer Auflösung von etwa 0,2 mV.

Die Spannungsquelle

Um eine stabile, das heißt thermisch wenig driftende Referenzspannung zu erhalten, reicht es nicht aus, eine gewöhnliche Z-Diode oder einen Spannungsregler zu verwenden, sondern ein Bauteil, das eine Ausgangsspannung liefert, die bei Änderungen der Versorgungsspannung, der Temperatur und auch im Laufe der Zeit möglichst konstant bleibt. Für dieses Projekt wurde eine Spannungsreferenzdiode LM236H-2.5 verwendet, die eine Temperaturdrift von nur 3,5 mV zwischen -25 °C und +85 °C aufweist, wenn sie wie hier mit einem Diffusionsstrom von 1 mA beaufschlagt wird. Wäre ihr Verhalten linear, so ergäbe sich eine Drift von etwa 32 µV/°C (etwa 12,8 ppm/°C). Hält man Temperatur und den Strom konstant, so beträgt die Spannungsänderung über einen Zeitraum von 1.000 Stunden etwa 20 ppm.

Wenn noch bessere Eigenschaften gewünscht werden, kann mit geringfügigen Änderungen der Schaltung stattdessen eine AD680-Referenzquelle verwendet werden; wären dagegen auch „schlechtere“ Chips ausreichend, könnte man diese Spannungsreferenz ohne

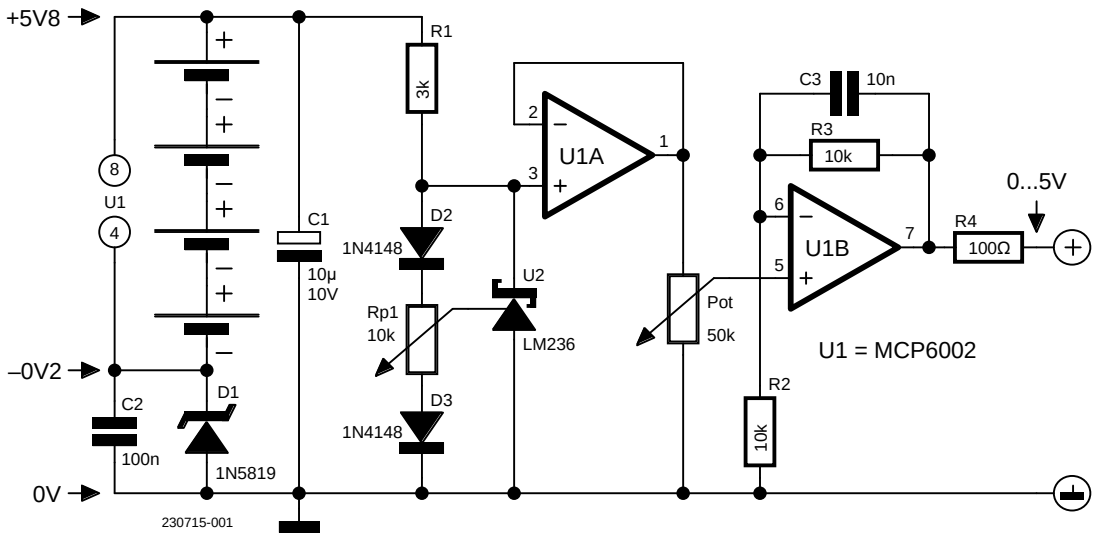


Bild 1. Schaltbild der variablen Spannungsquelle.

Schaltungsänderungen durch einen gebräuchlicheren TL431-Shuntregler ersetzen. **Bild 1** zeigt den Schaltplan. Die beiden Dioden D2 und D3 sollen die thermische Drift weiter verringern, und das Trimmpoti Rp1 sollte auf eine Ausgangsspannung von 2,49 V eingestellt werden, dem Wert, bei dem die besten Eigenschaften erzielt werden. Bei dem verwendeten Operationsverstärker (mehr zu seiner Funktion weiter unten) handelt es sich um einen MCP6002, einen Dual-Rail-to-Rail-Operationsverstärker, der in der Lage ist, bei Spannungspegeln zu arbeiten, die sehr nahe (einige zehn Millivolt) an den Rails liegen, im Gegensatz zu normalen Operationsverstärkern, die bei einer Differenz von weniger als einigen Volt gegenüber den Rails ihren Betrieb verweigern.

Die thermische Drift dieses Chips ist mit ±2 µV/°C sehr gering: und der Offset liegt bei einigen Millvolt. Um noch näher an Null heranzukommen, wurde die Schottky-Diode D1 in die Schaltung aufgenommen. Mit dieser Strategie liegt die negative Spannung der Operationsverstärker um etwa 100...200 mV unter null (Vorwärtsspannung dieser Dioden), und die minimale Ausgangsspannung wird nur durch den Offset der Operationsverstärker beeinflusst. Wie im Schaltbild zu sehen, ist das Nullpotential oder GND nicht der Minuspol der Batterien, sondern die Anode der Diode D1, und genau deshalb ist ein Differenzeingang des DVM erforderlich.

Nimmt man eine Verschlechterung der Eigenschaften dieses Entwurfs in Kauf, kann auch ein gewöhnlicher LM358 anstelle des MCP6002 verwendet werden. Der Opamp U1A arbeitet als Impedanzwandler mit hoher Eingangsimpedanz, und an seinem Ausgang ist das 10-Gang-Poti angeschlossen, mit dem das Potenzial zwischen Null und 2,5 V eingestellt werden kann. Der Widerstandswert ist nicht so wichtig; er kann zwischen 5 kΩ und 100 kΩ liegen. Der zweite Operationsverstärker U1B arbeitet als nicht-invertierender Verstärker mit einer Verstärkung von

$G = (1+R3/R2) = 2$

Die Ausgangsspannung liegt also zwischen etwa 0 V und 5 V. Der Kondensator C3 stellt ein Tiefpassfilter dar, das das thermische Rauschen der Halbleiter reduziert. Der Widerstand R4 begrenzt den Ausgangsstrom im Falle eines versehentlichen Kurzschlusses. Ein Reihenwiderstand von 100 Ω stellt kein Problem dar, da er im Vergleich zu den Eingangswiderständen der zu kalibrierenden Systeme vernachlässigbar sein dürfte. Die Spannungsversorgung ist batteriebetrieben, um Netzrauschen zu vermeiden, und ein Schaltnetzteil mit einem Ausgangsrauschen von einigen hundert Kilohertz mit Spitzenwerten von bis zu 100 mV sollte unbedingt vermieden werden.

Das hochpräzise DVM

Für die Herstellung eines Kalibrators benötigen Sie auch ein digitales Voltmeter mit einer guten Genauigkeit, die um einige Stellen besser ist als die des zu kalibrierenden Systems. Das hier vorgeschlagene DVM verfügt über einen 16-Bit-A/D-Wandler, der die positive Eingangsspannung in 2^{15} = 32.768 Stufen auflöst, mit einem maximal möglichen Skalenendwert von 32.767. In diesem System wurde der ADC mit einem Skalenendwert von 6.144 mV programmiert, so dass die Auflösung 6.144/32.767 = 0,1875 mV beträgt. Der Wandler lässt

Stückliste Spannungsquelle

Widerstände:
R1 = 3 k, 1%, 0,25 W
R2, R3 = 10 k, 1 %, 0,25 W
R4 = 100 Ω, 5%, 3 W (Wickelwiderstand)
Rp1 = 10 k, Mehrgang-Trimmpoti
Pot = 50 k, Mehrgang-Poti, linear, 10 Umdrehungen

Kondensatoren:
C1 = 10 µ, 25 V, Tantal
C2 = 100 n, keramisch
C3 = 10 n, keramisch

Halbleiter:
U1 = MCP6002, Operationsverstärker
U2 = LM236H-2.5, Spannungsreferenz
D1 = 1N5819, Schottky-Diode
D2, D3 = 1N4148, Diode

Außerdem:
SW1 = Schalter, 1xan
B1...B4 = 4 x AA, 1,5 V Alkaline-Zellen mit Batteriehalter

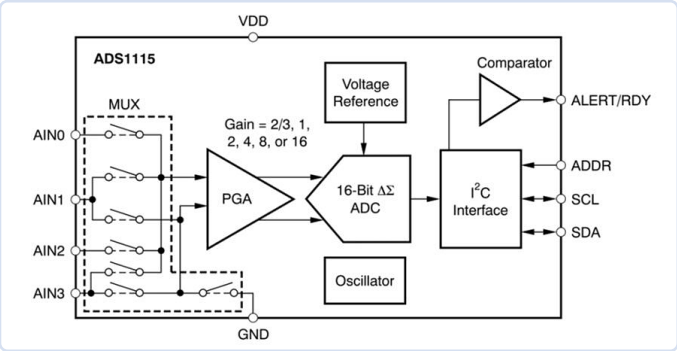


Bild 2. Innenschaltbild des ADS1115-Wandlers. (Quelle: https.//ti.com/lit/ds/symlink/ads1115.pdf)

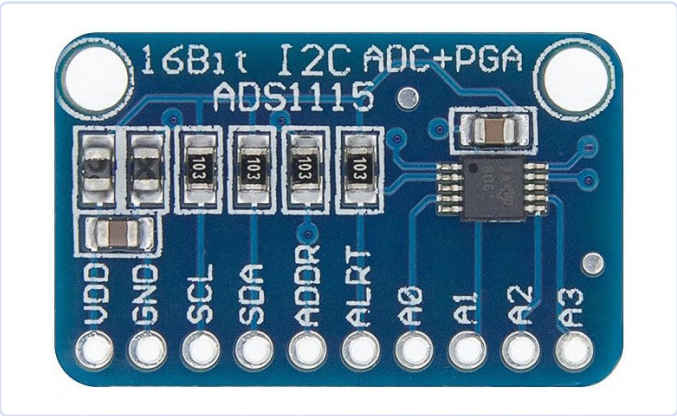


Bild 3. Das ADS1115-Breakout.

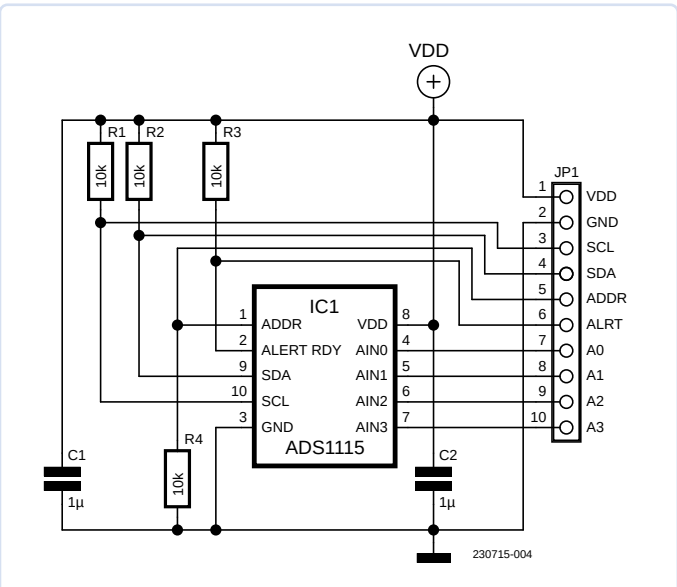


Bild 4. Schaltbild des ADS1115-Moduls.

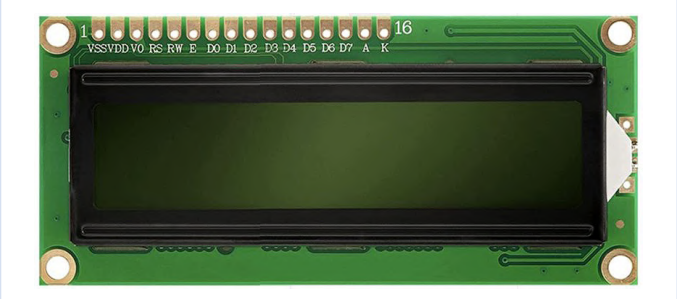


Bild 5. Das 16×2-LCD.

keine Eingangsspannungen zu, die 0,3 V höher sind als die Versorgungsspannung V_{cc}, also 5,3 V, und die Spannungsquelle ist für einen Maximalwert von etwa 5 V ausgelegt. Der maximale Wert, der 5 V entspricht, ist also 5.000/0,1875 = 26.666.
Dies ist eine sehr hohe Zahl im Vergleich zu den 1.999 eines normalen 3,5-stelligen Panel-DVMs. Mit einem zusätzlichen Schalter kann man das Voltmeter elektrisch abtrennen und es separat verwenden. Was nicht funktioniert, ist ein hochohmiger Eingangsteiler, um höhere Spannungen zu messen. Dies wurde bereits bei anderen Anwendungen versucht und führte zu Rauschproblemen, die sich dahingehend äußerten, dass selbst bei stabiler Eingangsspannung sich zwei Digits nicht stabil verhielten.

Der Wandler ADS1115

Das Herzstück des DVM ist der ADS1115, ein 16-Bit-Delta-Sigma-Analog-Digital-Wandler mit hervorragenden Spezifikationen, was Präzision und Fehler angeht. Er kann bis zu vier Eingänge haben. Von den 16 Bits ist das meist signifikante für das Vorzeichen reserviert, so dass der Chip sogar negative Werte messen kann, insbesondere wenn er im Differenzmodus konfiguriert ist, allerdings mit einigen Einschränkungen in Bezug auf die Versorgungsspannung. Dieser interessante Chip von Texas Instruments hat bemerkenswerte Eigenschaften:

- Eine Auflösung von 16 Bit, im Vergleich zu den 10 Bit eines Arduino
- Vier Single-Ended- (auf Masse bezogene) oder zwei Differenzkanäle
- Abtastrate von 8…860 Samples/Sekunde
- Verwendet einen programmierbaren Verstärker (PGA), mit dem auch kleine Spannungen gemessen werden können
- Verfügt über eine interne Referenzquelle mit geringer thermischer Drift
- Kann mit Spannungen von 2…5,5 V versorgt werden, bei einer Stromaufnahme von nur 150 µA, die im Single-Shot-Modus noch weiter reduziert wird
- I²C-Schnittstelle mit mehreren wählbaren Adressen
- Verfügt über eine Warnfunktion (ALERT/RDY) zur effizienten Überwachung der Spannungen. Dies kann die Stromaufnahme eines Geräts reduzieren, indem das Signal bei bestimmten Ereignissen den Mikrocontroller aufweckt und in Betrieb nimmt oder Interrupts erzeugt.

Bild 2 zeigt das Funktionsdiagramm des ADS1115-Chips, der über zwei Umwandlungsmodi verfügt:

- **Continuous conversion:** Der ADS1115 führt kontinuierlich Konvertierungen durch. Sobald eine Konvertierung abgeschlossen ist, trägt der ADS1115 das Ergebnis in das Konvertierungsregister ein und startet sofort eine weitere Konvertierung.
- **Single-shot conversion:** Der ADS1115 wartet, bis das OS-Bit auf High gesetzt wird. Sobald dies der Fall ist, wird das Bit auf 0 gesetzt, was anzeigt, dass eine Konvertierung im Gange ist. Sobald die Konvertierungsdaten fertig sind, wird das OS-Bit erneut bestätigt und das Bauteil schaltet sich ab. Das Schreiben einer 1 in das OS-Bit während einer Konvertierung hat keine Auswirkung.

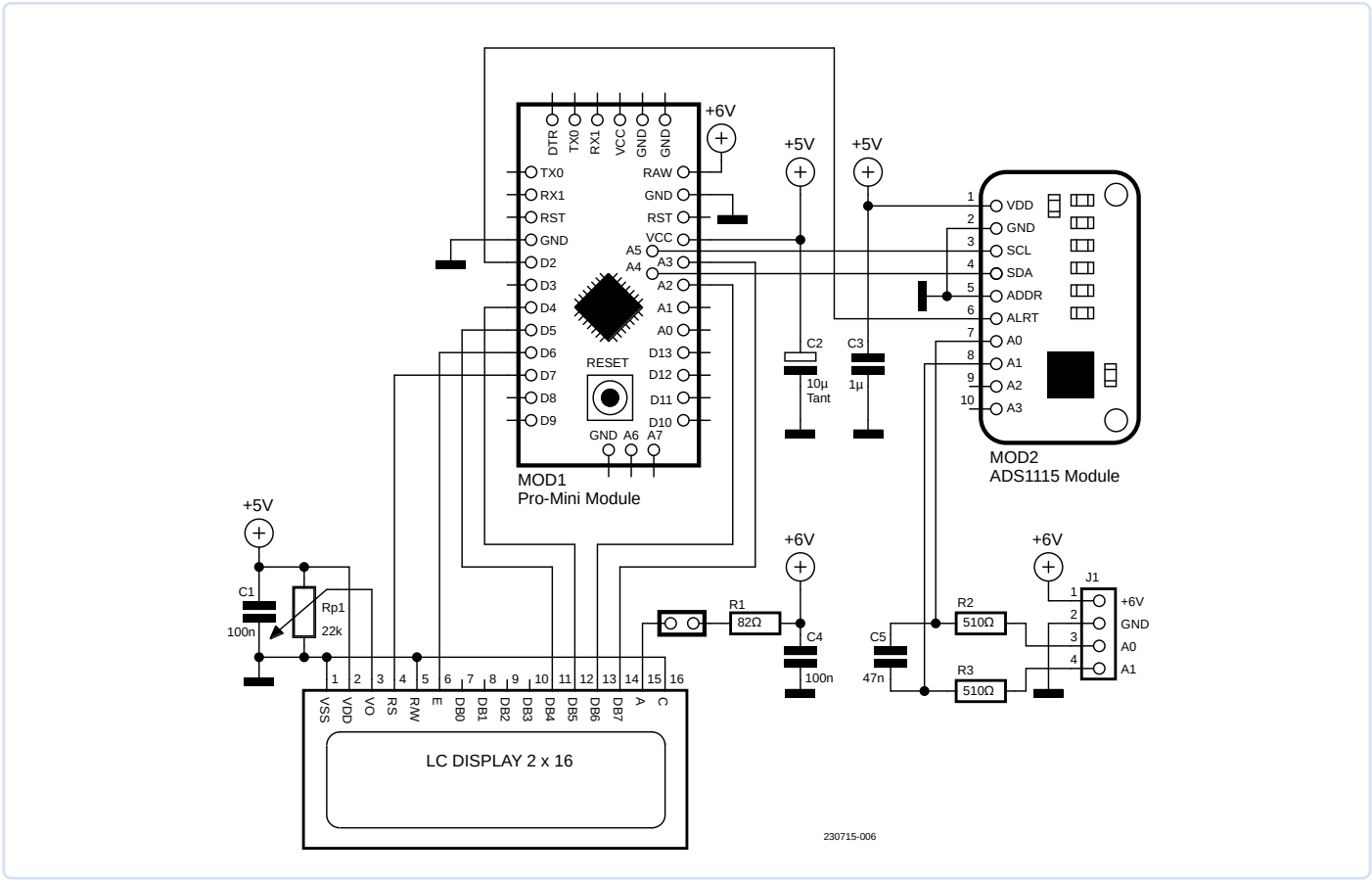


Bild 6. Schaltbild des DVMs.

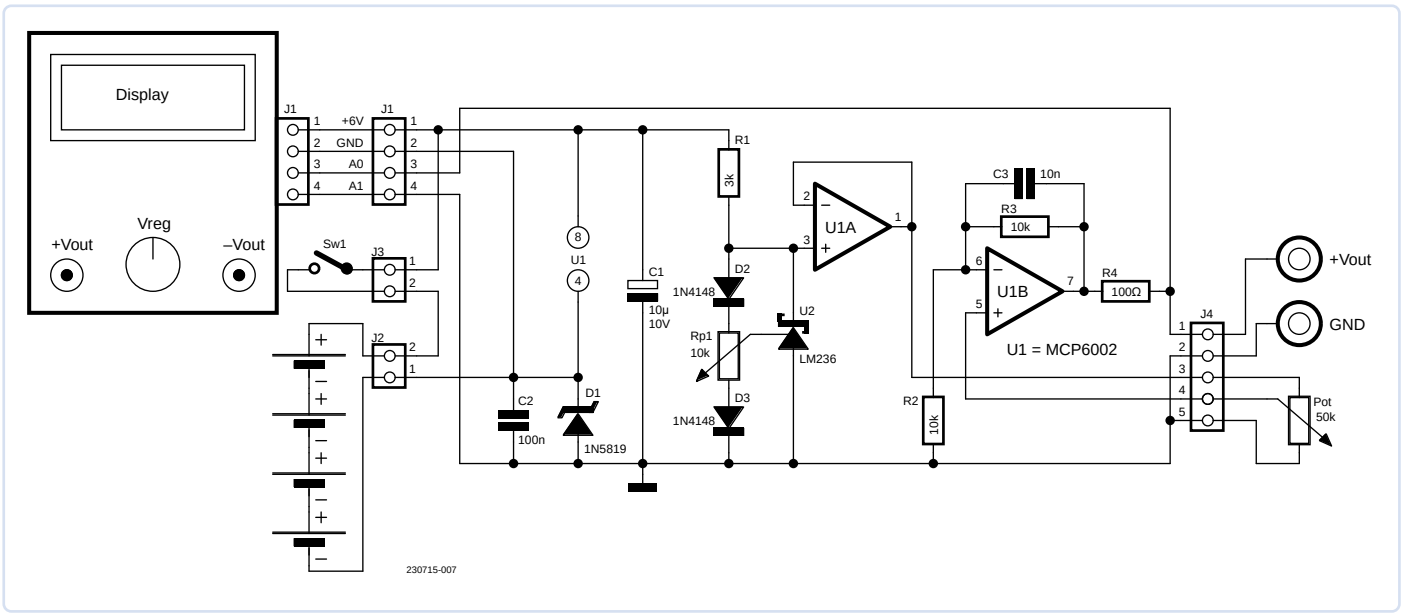


Bild 7. Verdrahtung zwischen dem DVM und der Spannungsquelle.

In diesem Projekt wird der energiesparende Single-Shot-Modus mit einer Wandlung pro halber Sekunde verwendet.

ADS1115-Modul

Auf dem Bauteilemarkt sind Module wie das in **Bild 3** gezeigte und ähnliche erhältlich. Das hier eingesetzte Modul ist eigentlich ein Breakout eines ADS1115 mit nur wenigen anderen Komponenten, wie man im Schaltplan in **Bild 4** sehen kann. Um diesen Chip mit der Arduino-IDE zu verwenden, wurde die Bibliothek ADS1115_WE von Wolfgang Ewald verwendet, die von der IDE selbst (im Bibliotheksmanager) oder aus dem Internet heruntergeladen werden kann [1]. Sie verfügt über zahlreiche Funktionen, mit denen alle Eigenschaften des Chips genutzt werden können.

Das Display

Es wurde ein LC-Display gewählt, das mit dem HD44780-Controller von Hitachi kompatibel ist. Um schöne große Zeichen zu erhalten, wurde das klassische zweizeilige 16-Zeichen-Display verwendet, das dem

in **Bild 5** gezeigten ähnelt. Das Display ist nicht nur sehr gut lesbar, sondern benötigt auch wenig Strom (wenn die Hintergrundbeleuchtung abgeschaltet ist). Eine 5-V-Versorgung und entsprechende Logikpegel machen das Display perfekt für die Zusammenarbeit mit dem Arduino Pro Mini. Die benötigte Arduino-Bibliothek finden Sie in [2].

Anschlussschema des DVM

Bild 6 zeigt den Verdrahtungsplan des DVM. Als Mikrocontroller wurde der sehr kompakte Arduino Pro Mini ausgewählt, das auch die 5 V für die Versorgung des LCD und des Wandlers liefert. In diesem Projekt werden die als Differenzeingang konfigurierten Eingänge A0 und A1 des ADC-Moduls verwendet. Die Stromaufnahme des gesamten Systems liegt bei etwa 20...25 mA, wenn die LCD-Hintergrundbeleuchtung nicht eingeschaltet ist. Die ideale Spannung wäre 6 V; das System verträgt auch höhere Spannungen (bis maximal 12 V), aber der winzige eingebaute Regler des Arduino könnte dann überhitzen. Der Strom für die Hintergrundbeleuchtung sollte nicht diesem internen Regler entnommen, sondern von einem externen Regler mit dem Widerstand R1 in Reihe geliefert werden. R1 ist dann entsprechend dem erforderlichen Strom zu berechnen.
Im Prototyp wurde ein 82-Ω-Widerstand verwendet, es dürfen aber auch 100 Ω sein, so dass sogar noch einige Milliampere gespart werden. Ältere Anzeigen sollten Sie nicht verwenden, da diese mit LEDs alter Technologie ausgestattet sind, die einen Strom von bis zu 100 mA ziehen. Neuere Displays benötigen maximal etwa 20 mA, da sie von sehr hellen LEDs beleuchtet werden. Ein 4-poliger Stecker verbindet das DVM mit der Platine für die variable Spannungsquelle. Der Kondensator C5 und die 510-Ω-Widerstände R2 und R3 wurden zur Rauschunterdrückung hinzugefügt. Sie bilden ein Tiefpassfilter erster Ordnung. Die Zeitkonstante beträgt $\tau = 47{,}94\ \mu s$, das entspricht einer Grenzfrequenz von

$$f = 1/(\tau \times 2 \times \pi) = 3{,}320\ kHz$$

Verdrahtungsschema

Für den Prototyp wurden zwei Platinen hergestellt, eine für das DVM und eine für die Spannungsquelle (**Bild 7**), aber es spricht nichts dagegen, alle Bauteile/gruppen auf einer Platine zusammenzufassen. Die Stromversorgung für die digitalen Funktionen erzeugt aufgrund von Schaltströmen Rauschen, daher wurden Bypass-Kondensatoren wie der Tantal-Kondensator C2 verwendet, um Rauschspitzen zu reduzieren.

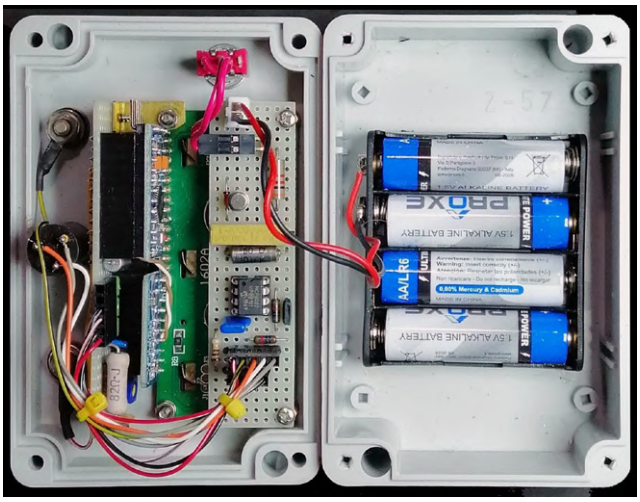


Bild 8. Innenansicht des fertigen Prototyps.

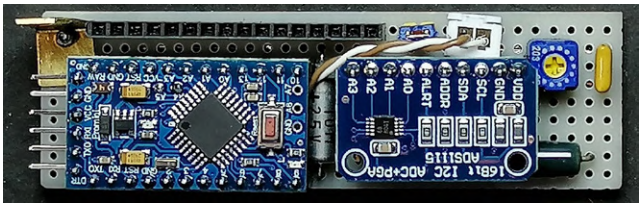


Bild 9. Die vollständig bestückte DVM-Platine.

Der Prototyp

Bild 8 zeigt den Prototyp: Die beiden einseitigen Lochrasterplatinen sind mit dem Display und dem ADC-Wandler mit 16- und 10-poligen Buchsen/Stiftstreifen und das Arduino Pro-Modul mit zwei 12-poligen Streifen verbunden.
Bild 9 zeigt die Platine des DVM. Leider sind die Pins A4 und A5 des I²C-Busses des Arduino Pro Mini nicht auf den 12+12-Anschlussreihen verfügbar und auch nicht einmal in einem 2,54-mm-Abstand, so dass zwei Drähte (weiß und braun) mit einem 2-poligen Stecker angelötet wurden müssen, um die Signale zum ADS1115-Modul zu übertragen.

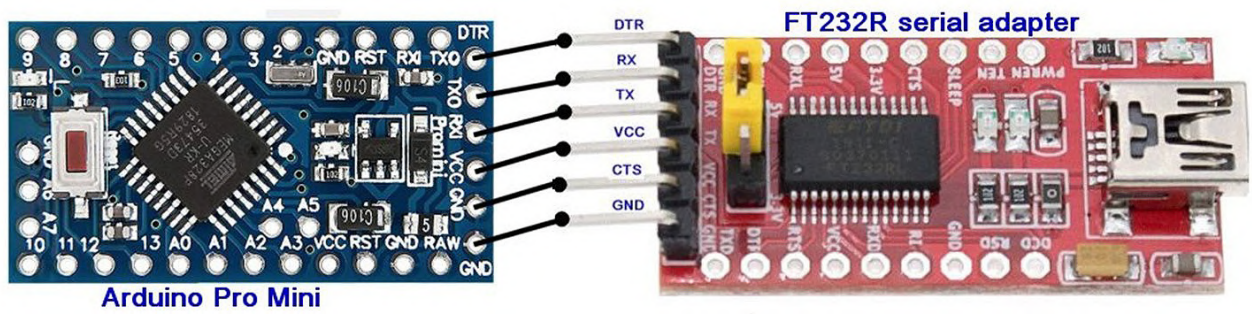


Bild 10. Verbindungen zwischen seriellem Adapter und Arduino Pro Mini.

Die 16-polige Steckerleiste oben links ist für das LCD, das im 90-Grad-Winkel verbunden ist. Die Lochrasterplatine ist sehr kompakt und wird mit einem kleinen Messingbügel (links oben im Bild) am Display befestigt.

Programmierung

Für die Programmierung benötigen wir einen seriellen USB-TTL-Adapter wie den in **Bild 10**. Natürlich müssen wir auch den Adaptertreiber laden, der neben den Rx- und Tx-Signalen auch das DTR-Signal verwaltet, das zum Zurücksetzen des Systems verwendet wird. Er dient auch als serielle USB-Schnittstelle, aber in diesem Fall kommt er vor allem zum Einsatz, um den von der Arduino-IDE kompilierten Sketch auf unser System zu übertragen. In der Arduino-IDE müssen Sie das Board als *Arduino Pro* mit 16 MHz und 5 V einstellen. Nach der Programmierung kann man, wenn alles gut verlaufen ist, das Programmieradapter entfernen und die Platine über die Batterien mit Strom versorgen.
Der Anschluss auf dem Arduino ist eine 6-polige Buchsenleiste im 0,1-Zoll-Raster, die direkt an den Stecker der Adapterplatine gelötet oder mit Female-Female-Breadboard-Kabeln verbunden wird. Doch Achtung, nicht bei allen seriellen Adaptern sind die Pins wie abgebildet angeordnet oder sitzt der gleiche Chip auf dem Platinchen. Bei einigen seriellen Adapterchips wie dem Prolific PL2303 sind unter Windows 10 Treiberprobleme aufgetreten.

Kalibrierung des DVM

Obwohl die Messergebnisse des ADS1115 bereits sehr genau waren, wurde die Kalibrierung mit einem Tischmultimeter HP 3478A mit 5,5 Stellen und 0,1 µV Auflösung durchgeführt. Um dies zu erreichen, müssen wir die Zeile in der Funktion `printData()` ersetzen:

```
// calibrated and result in millivolts
voltage = Ch0*1.000515+0.022;
```

durch

```
// not calibrated and result in millivolts
voltage = Ch0;
```

Es wurden etwa zehn Werte erzeugt, die Messwerte des Referenzmultimeters und des Kalibrators wurden in Excel eingegeben. Das Punktdiagramm (*scatter plot*) mit der Trendlinie (*trend line)* wurde mit

Stückliste DVM

Widerstände:
R1 = 82 Ω, ±5%, 1 W, siehe Text
R2, R3 = 510 Ω, ±1%, 0,25 W
Rp1 = 22 k, Trimmpoti

Kondensatoren:
C1, C4 = 100 n, 50 V, keramisch
C2 = 10 µ, 25 V, Tantal
C3 = 1 µ, 63 V, keramisch
C5 = 47 n, 63 V, Polyester

Module:
1 x Arduino Pro Mini (oder kompatibel)
1 x 16x2-LC-Display, mit Hintergrundbeleuchtung
1 x ADS1115 A/D-Wandler

Außerdem:
12+12+10+16-polige Buchsenleiste, teilbar

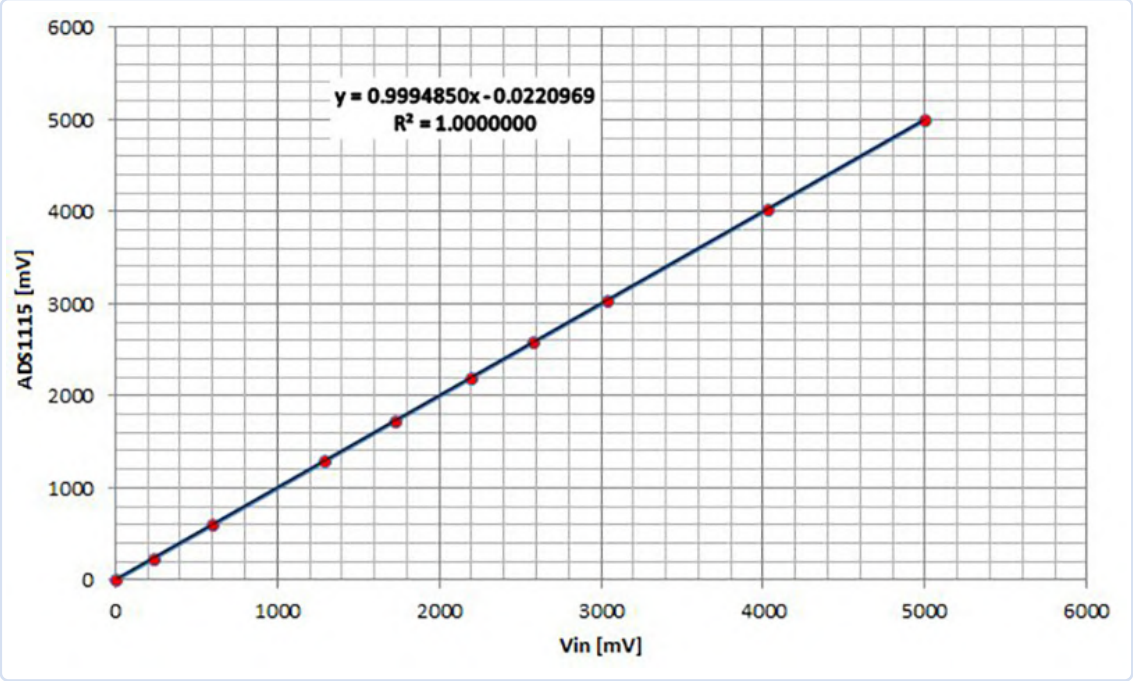


Bild 11. DVM-Kalibrierungsdiagramm (Linearisierung).

der Option *linear* eingegeben und die Optionen *Formel im Diagramm anzeigen* und *R-Quadrat-Wert im Diagramm anzeigen*. Das Ergebnis war ausgezeichnet, wie in **Bild 11** zu sehen ist. Der Wert von R2 war eine 1 mit sieben Dezimalstellen gleich Null: Dies entspricht einem extrem hohen Linearitätswert. Um die Daten zu korrigieren, müssen wir die inverse Formel anwenden, also die Achsen invertieren. So erhalten wir:

- Steigung = 1,000515286
- Achsenabschnitt = 0,022110099
- Fehler = 0,051941236

Wie Sie sehen können, ist der Fehler wirklich sehr gering. Wenn kein sehr präzises Messgerät zur Verfügung steht, kann man die Korrektur auch unterlassen und mit einem Standardfehler von 0,051941236 leben. Andernfalls korrigieren wir die Linie mit den Werten, die bei der Kalibrierung ermittelt wurden. Beim Prototyp betrug das Minimum der erzeugten Spannung 1,7 mV (wegen des Offsets der Operationsverstärker nicht gleich Null), das Maximum 4.997,03 mV.

Beschreibung des Programms

Das Programm ist relativ einfach: Nach der Initialisierung der I/O, der Anzeige und des ADC-Wandlers führt es alle 500 ms eine Messung durch. Die Funktion `readChannel(ADS1115_MUX channel)` ist die für den Single-Shot-Modus vorgeschlagene Funktion und führt die Messung in Millivolt durch. Die Funktion `LCDprintLine(String text, byte line)` schreibt einen String in die obere oder untere Zeile des Displays. Die Funktion `printData()` korrigiert die Messung mit den Ergebnissen der Erstkalibrierung und druckt sie aus.

RG — 230715-02

Listing 1

```
/*  program ArduCalibrator.ino for the ADS1115, 16 bit 4 ch ADC
a single differential input (A0,A1), Ch0 6144 mV (5000 generated)
read the  precision voltage source
uses an Arduino Pro Mini, LCD 16x2 display
Giovanni Carrera 23/11/2022 with calibration
*/
#include <Wire.h>
#include<ADS1115_WE.h>
#include <LiquidCrystal.h>
ADS1115_WE adc = ADS1115_WE();// uses Wire / ADC Address = 0x48
// LCD pins
#define rs 7
#define en 6
#define d4 5
#define d5 4
#define d6 A2
#define d7 A3
// initialize the library by associating any needed LCD interface pin
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
#define Alert 2  // AD Alert pin
const unsigned long deltat_ms = 500;
unsigned long cms, pms;
float Ch0,Ch1;
void setup(){
   Wire.begin();
   //Serial.begin(115200);
   pinMode(Alert,INPUT);
   // set up the LCD's number of columns and rows:
   lcd.begin(16, 2);
   lcd.print(F("ArduCalibrator"));
   lcd.setCursor(0, 1);// print on the second row
   lcd.print(F("GCar V231122"));
   delay(2000);
   if(!adc.init()){
```

```
    lcd.setCursor(0, 1);
    lcd.print(F("No ADS1115 found"));
    while( true );// ends here
  }
  adc.setVoltageRange_mV(ADS1115_RANGE_6144); // 6144 mV input range
  LCDprintLine("PRECISION SOURCE", 0);
}
void loop() {
  cms = millis();
  // checks if passed deltat milliseconds
  if(cms - pms > deltat_ms) {   // period timebase
      pms = cms;// update pms
    Ch0 = readChannel(ADS1115_COMP_0_1);// Ch0 differential A0(+) with A1(-)
    Ch0 = readChannel(ADS1115_COMP_0_1);// repeating helps
    printData();
  }
}
```

Haben Sie Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Bitte kontaktieren Sie den Autor oder die Elektor-Redaktion unter redaktion@elektor.de.

Über den Autor

Giovanni Carrera hat einen Abschluss in Elektrotechnik. Als Universitätsprofessor an der Fakultät für Schiffstechnik im italienischen Genua unterrichtete er in zahlreichen Kursen zum Beispiel über Schiffsautomatisierung und die Simulation von Schiffsantriebsanlagen. Carrera begann in den späten 1970er Jahren mit der 6502-CPU zu arbeiten und ging dann zu anderen Prozessoren über. Heute widmet er sich dem Entwurf und der Entwicklung analoger und digitaler elektronischer Schaltungen, über die er in seinen Blogs (ArduPicLab und GnssRtkLab) und in verschiedenen Zeitschriften geschrieben hat.

Passende Produkte

- **SparkFun Arduino Pro Mini 328 (5 V, 16 MHz)**
  www.elektor.de/20091
- **Standard-LCD mit 2x16 Zeichen und Hintergrundbeleuchtung**
  www.elektor.de/16414

WEBLINKS

[1] ADS1115-Bibliothek von Wolfgang Ewald: https://github.com/wollewald/ADS1115_WE
[2] LiquidCrystal-Bibliothek : https://github.com/arduino-libraries/LiquidCrystal
[3] Download-Link für dieses Projekt: https://elektormagazine.de/Arducalibrator
[4] G. Carrera, „Rauscharmer ADC-Kalibrator für moderne Mikrocontroller", Elektor Summer Circuits 2022: https://www.elektormagazine.de/magazine/elektor-262/60738
[5] Ultra-Small, Low-Power, 16-Bit Analog-to-Digital Converter with Internal Reference: https://ti.com/lit/ds/symlink/ads1114.pdf

FPGAs für Einsteiger

Der Weg vom Mikrocontroller zur FPGA-Programmierung

Von Theo Mulder (Niederlande)

Es gibt viele kostengünstige Mikrocontroller-Entwicklungsboards, aber FPGAs bleiben teuer. Für Anwendungen, die eine höhere Rechenleistung zum Beispiel für KI oder eine höhere I/O-Geschwindigkeit für schnelle Kameras, Displays, ADCs oder DACs benötigen, wären kleine und erschwingliche FPGA-Boards wünschenswert. Steigen wir nun ein in die aufregende Welt der FPGAs!

Bei der Suche nach preisgünstigen FPGAs ist es hilfreich, die aktuellen Geschäftstrends zu verstehen. Der FPGA-Markt wurde im Jahr 2022 auf 8,0 Mrd. US$ geschätzt und soll bis 2027 auf 15,5 Mrd. US$ anwachsen. Der Sektor ist wettbewerbsintensiv und hat eine erhebliche Konsolidierung der Anbieter erlebt. Die bis heute verbliebenen Hersteller sind AMD/Xilinx, Intel/Altera, Lattice, Microchip, QuickLogic, Efinix, Flex Logix, GOWIN, Achronix, S2C und Renesas. Durch Firmenübernahmen verändert sich die Landschaft ständig. Bereits 1999 erwarb Lattice Vantis, 2001 dann Acree und 2010 übernahm Microsemi Actel. Im Jahr 2013 waren die wichtigsten Akteure Xilinx, Altera, Actel, Vantis, Lattice, Lucent, QuickLogic und Cypress. Das Ballett der Fusionen und Übernahmen hat jedoch nie aufgehört; selbst die „Großen Zwei", Xilinx und Altera, wurden schließlich von AMD beziehungsweise Intel übernommen.

Im Jahr 2019 führte Xilinx den Markt mit einem Anteil von 52 % an, gefolgt von Altera mit 35 %. Kleinere Anteile entfielen auf Microchip, Lattice und andere. Trotz der Dominanz dieser größeren Unternehmen sind kleinere FPGA-Anbieter nach wie vor aktiv, was für die Unterstützung verschiedener Arten von Unternehmen und die Förderung von Innovationen hervorragend ist.

Der zunehmende Einsatz von FPGAs in der Automobilindustrie bietet Anbietern wie Lattice Chancen, da die Entwickler in der Branche nach kostengünstigen Alternativen zu den teureren Angeboten der großen Unternehmen suchen. Renesas entwickelt sich aufgrund seiner etablierten Präsenz im Automobilsektor zu einem bedeutenden Konkurrenten.

Die Dynamik des FPGA-Marktes mit seinen Wettbewerbsverschiebungen, Übernahmen und dem Eintritt neuer Akteure könnte auch Makern zugutekommen, die kostengünstige FPGA-Optionen für ihre Projekte suchen.

Erschwingliche FPGA-Boards für Einsteiger

Da sich die Toolchains der verschiedenen FPGA-Hersteller stark voneinander unterscheiden und die Einarbeitung in ein neues Software-Tool Zeit in Anspruch nimmt, ist es sinnvoll, die Boards nach FPGA-Herstellern zu unterscheiden. Auf diese Weise ist die Lernzeit gut investiert, wenn Sie sich in Zukunft für ein höherwertiges Modell desselben FPGA-Herstellers entscheiden, den Sie anfangs herausgepickt haben.

Die drei wichtigsten Hersteller, die in Frage kommen, sind Altera/Intel, Xilinx/AMD und Lattice. Letzterer bietet in der Regel weniger leistungsfähige, aber erschwinglichere Modelle an als die beiden erstgenannten.

Das Evaluierungsboard iCE40HX1K-EVB von Olimex mit FPGAs von Lattice Semiconductor bietet mit einem Preis von 15 € einen äußerst günstigen Einstieg, erfordert jedoch einen externen Programmierer. Dieses Board enthält einen iCE40HX1K mit 1280 Logikelementen (LE). Interessanterweise verfügt dieses FPGA über einen internen, nichtflüchtigen Konfigurationsspeicher, der dank des von Olimex bereitgestellten Sketches über die SPI-Schnittstelle eines (eventuell vorhandenen) Arduino-Boards programmiert werden kann. Das offizielle Programmiergerät von Lattice verwendet einen FT232H von FTDI, und es ist auch möglich, ein FT232H-Breakout-Board dafür einzusetzen. Eines der interessantesten Merkmale dieser FPGA-Familie ist die Kompatibilität mit dem grafischen Open-Source-Tool Icestudio, mit dem Benutzer mit blockbasierter grafischer Programmierung experimentieren können. Wer andere sehr kompakte, breadboardfähige und budgetfreundliche Optionen sucht, dürfte mit den interessanten Produkten der TinyFPGA-Familie gut bedient sein. Der TinyFPGA BX kostet 40 € und ist mit einem iCE40LP8K von Lattice ausgestattet, der ebenfalls mit Icestudio kompatibel ist. Die Modelle AX1 und AX2 für 13 € respektive 17 € sind kleinere und noch billigere Alternativen, wenn auch mit weniger Funktionen. Leider sind diese drei Modelle oft nicht auf Lager. Von einem anderen

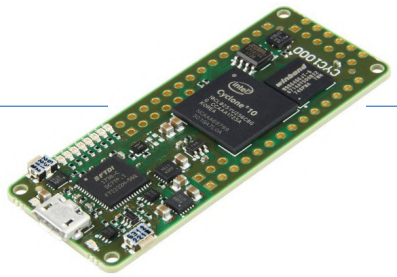


Bild 1. Das CYC1000-Board.

Hersteller stammt der Upduino v3.1, der 30 € kostet und ein iCE40UP5K-FPGA mit 5,3 kLE enthält.
Der iCEstick von Lattice ist eine kompakte Option im USB-Stick-Format mit einem iCE40HX1K-FPGA und einem Onboard-Programmierer für 48 €. Es ist das offizielle Tool von Lattice, funktioniert also gut mit der Lattice-Software und wird von vielen Tutorials unterstützt.
Um das Lattice-Angebot abzurunden, sei noch eine Open-Source-Plattform im Arduino-UNO-Formfaktor erwähnt, nämlich Alhambra II (iCE40HX4K, 3,52 kLE, 60 €) und das Go-Board (iCE40HX1K, 70 €), das der Hersteller Nandland.com mit ausgezeichneten Tutorials auf seiner Webseite unterstützt.
Das MAX1000-Board von Trenz Electronic ist ein MAX10-FPGA von Altera (jetzt im Besitz von Intel), das mit 8 kLE oder 16 kLE (34 € bis 49 €) erhältlich ist und ein breadboard-freundliches Design bietet. Das CYC1000 [1] in **Bild 1**, ebenfalls von Trenz, kostet 40 € und ist mit einem Cyclone-10-FPGA (25 kLE) ausgestattet.
Traditionellere, voll funktionsfähige Boards gibt es von Terasic. Einige der Produkte dieser Firma sind teuer, aber andere sind für Einsteiger sehr interessant, zum Beispiel das DE0-nano, das ein Cyclone-IV-FPGA mit 22 kLE und eine Reihe von Onboard-Funktionen für 110 € bietet. Das DE10-Lite für 140 € ist ein funktionsreiches Board mit einem MAX10-FPGA (50 kLE), verschiedenen I/O-Optionen, Siebensegmentanzeigen, LEDs und Schaltern sowie Unterstützung für Arduino-Shields.
Unter dem Dach von AMD (ehemals Xilinx) bieten der Digilent Cmod S7 und der Cmod A7-35T breadboard-fähige Designs mit Spartan-7- und Artix-7-FPGAs, 23,4 kLE beziehungsweise 33,3 kLE, zu einem Preis von jeweils 90 €.
Ein größeres und funktionsreiche Board ist das Digilent Basys 3 mit einem Artix-7-FPGA mit 33,3 kLE, umfangreichen I/O-Optionen einschließlich VGA-Ausgang und USB-Host, zu einem Preis von 155 €.
In diesem Artikel werde ich mich auf Intel/Altera-Produkte konzentrieren. Ich empfehle das oben abgebildete Board CYC1000. In den nächsten Abschnitten werden wir mit Intels Quartus Prime Lite experimentieren, der kostenlosen Version des offiziellen Entwicklungstools für FPGAs von Altera/Intel.

Installation

Auf der Software Seite für Intel Quartus Prime Design [2] finden Sie Download-Links für die neueste Version für Windows oder Linux. Eine gute Internetverbindung ist erforderlich, da die Datei etwa 5,5 GB groß ist. Das Installationsprogramm benötigt 15 GB freien Speicherplatz. Einige Teile von Quartus Prime Lite verwenden Code aus frühen Versionen von Quartus. Daher sollten Leerzeichen in Dateinamen vermieden werden, auch wenn das nach heutigen Maßstäben überraschend altmodisch klingt.
Nur das Low-Level-Design-Tool wird benötigt. Es gibt zusätzliche Pakete, die nicht in den Rahmen dieser Einführung fallen, zum Beispiel der DSP-Builder für Signalverarbeitungsanwendungen, High-Level-Synthese-Compiler (mit C++ als Eingabe, für fortgeschrittene Anwendungen) sowie die Embedded-Design-Suite Nios II zur Implementierung eines Nios-II-Softprozessors innerhalb eines FPGAs.

Ein neues Projekts

Erstellen Sie zunächst ein Verzeichnis - zum Beispiel *C:\Projects\IntelFpga\Elektor\BeginExample*. Öffnen Sie dann Quartus Prime Lite. Der Hauptbildschirm ist in **Bild 2** zu sehen. Gehen Sie zu *File* und wählen Sie *New Project Wizard*. Nachdem Sie den Willkommensbildschirm gewürdigt haben, klicken Sie auf *Next*. Wenn Sie zur Eingabe des Arbeitsverzeichnisses

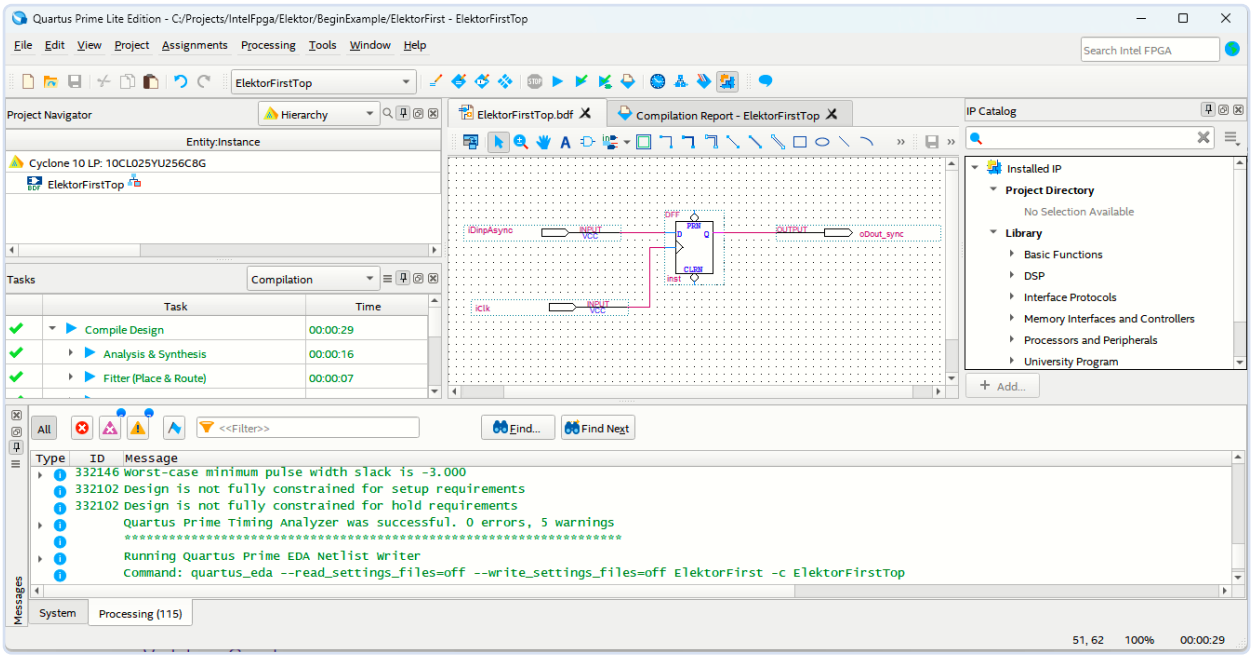
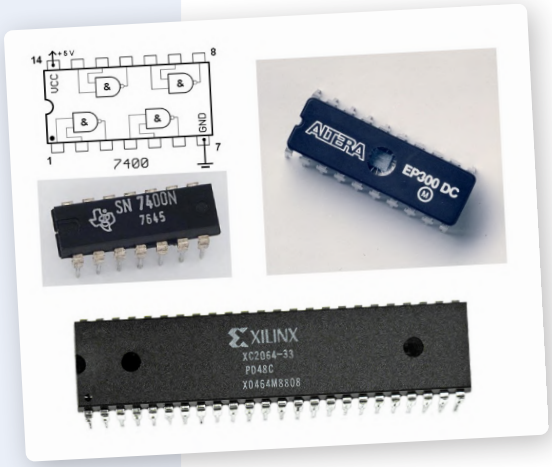


Bild 2. Quartus Lite Edition, mit ElektorFirstTop.bdf.

Geschichte der Logik: Von TTL zum FPGA

Vor dem Aufkommen von FPGAs wurde digitale Schaltungen hauptsächlich mit TTL-Chips und programmierbaren Bausteinen wie FPLDs, PLAs und PALs realisiert. Die TTL-Serie SN7400 arbeitete mit bis zu 20 MHz. Altera stellte vor 40 Jahren den EP300 vor, dem der XC2064 von Xilinx folgte. Der EP300 wurde in einem 5000-nm-Prozess mit Geschwindigkeiten bis zu etwa 10 MHz gebaut, während moderne Mid-Range-FPGAs einen 20-nm-Prozess verwenden und Taktgeschwindigkeiten von wenigstens 250 MHz erreichen.

Der EP300 war ein wiederprogrammierbarer 20-Pin-Baustein, der mit der Software A+PLUS auf IBM-PCs unter DOS lief. Der XC2064 von Xilinx verfügte über Logikzellen mit je 1200 Gattern und verwendete einen stromsparenden CMOS-Prozess mit einer Design-Software, die Future-Net, VIEW-logic und den XACT-Design-Editor umfasste. Der XC2064 wurde mit einer internen Toggle-Rate von 100 MHz beworben, erreichte aber typischerweise nur etwa 70 MHz. Anfänglich verwendeten FPGA-Softwaretools Schaltplanerfassung, die später durch textbasierte Hardwarebeschreibungssprachen (HDL) wie AHDL für Altera und VHDL und Verilog für Xilinx ergänzt wurde. Heute unterstützen beide Unternehmen VHDL und Verilog. Ältere Versionen des Quartus-Tools von Altera verwendeten AHDL mit eingebauten Simulationsfunktionen. Später wurde aufgrund der fehlenden AHDL-Unterstützung ModelSim für die Simulation erforderlich. Heute bieten alle Hersteller eine kostenlose Version ihrer Tools an.



aufgefordert werden, wählen Sie das von Ihnen erstellte Verzeichnis. Sie müssen einen Projektnamen sowie einen Namen für die Top-Level-Design-Entity wählen, die der Compiler als Root des Entwurfs verwenden soll. In diesem Beispiel wurden *ElektorFirst* und *ElektorFirstTop* verwendet. Im nächsten Schritt wählen Sie *Empty Project*. Dann werden Sie zu *Add files* weitergeleitet. Da wir keine Dateien haben, klicken Sie einfach wieder auf *Next*.

Sie gelangen dann zu einer Seite, auf der Sie über eine der beiden Tabs *Device* und *Board* ein *Device* auswählen können. Im letzteren Tab sind mehrere Entwicklungsboards der MAX10- und Cyclone-V-Familien aufgelistet, weitere Boards können später installiert werden. Auf der Registerkarte *Device* können Sie die genaue Familie und Teilenummer des verwendeten FPGAs auswählen, zum Beispiel *10LC025YU256C8G* in der Cyclone-10-LP-Familie, wenn Sie das CYC1000-Board haben. Die Cyclone-10-Familie besteht aus kostengünstigen und stromsparenden Mitgliedern, die vom Cyclone V abgeleitet wurden.

Wenn Sie noch keine Entwicklungsplatine haben, können Sie trotzdem diese auswählen, um mit dem Tutorial fortzufahren und mit dem Kompilierungsprozess zu experimentieren. Klicken Sie auf *Next*, nochmals *Next* und *Finish*.

Struktur des Projektverzeichnisses

Nehmen Sie sich einen Moment Zeit, um Ihr Arbeitsverzeichnis zu begutachten. Sie werden feststellen, dass es ein db-Verzeichnis gibt, das Quartus später verwenden wird. Die Datei *ElektorFirst.qpf* ist die Quartus-Projektdatei (QPF). Wenn Sie später zu Ihrer Arbeit zurückkehren wollen, können Sie einfach auf diese Datei doppelklicken, um die Quartus-Software erneut zu öffnen. Außerdem gibt es eine Datei namens *ElektorFirst.qsf*, das Quartus Settings File (QSF). Beides sind Textdateien, die Sie sich bei Interesse ansehen können - klicken Sie sie einfach mit der rechten Maustaste an und wählen Sie mit *Open with* einen Texteditor.

Öffnen des Projekts

In der oberen linken Ecke sehen Sie den *Project Navigator* mit ausgewählter *Hierarchy*. Suchen Sie nach dem zuvor benannten *ElektorFirstTop*. Wenn Sie darauf klicken, erscheint ein Pop-up-Fenster mit der Meldung *Can't find design entity ElektorFirstTop*, weil wir noch keine Datei erstellt haben. Klicken Sie auf *OK* und gehen Sie dann in das *File*-Menü. Hier sehen Sie verschiedene Dateitypen, die Sie erstellen können. Wählen Sie *Block Diagram/Schematic File* und klicken Sie auf *OK*.

Jetzt sehen Sie im mittleren Bereich eine weiße Fläche, die oben mit *Block1.bdf* beschriftet ist. Gehen Sie zu *File*, wählen Sie *Safe As*, und es wird vorgeschlagen, die Datei als *ElectorFirstTop.bdf* zu speichern. Das ist perfekt - klicken Sie auf *Save*. Wenn Sie nun unter *Project Navigator* und *Hierarchy* auf *ElektorFirstTop* klicken, landen Sie auf diesem leeren Schaltplanblatt.

Hinzufügen von Bauteilen

Damit alle Leser, auch diejenigen, die mit den Sprachen Verilog und VHDL nicht vertraut sind, diesem Tutorial folgen können, werden wir die Schaltplanprogrammierung verwenden. Klicken Sie auf die Registerkarte *ElektorFirstTop.bdf*, um das Blatt aufzurufen, klicken Sie mit der rechten Maustaste auf eine beliebige Stelle des Blattes und wählen Sie *Insert Symbol*. Dies öffnet das Menü *Libraries*. Klicken Sie auf das kleine Dreieck neben *Libraries*, um die Liste zu erweitern, in der Sie Kategorien wie *Megafunctions, Other, Primitives* und mehr finden. Hier gibt es eine Menge zu entdecken, also stöbern Sie ruhig herum.

Scrollen Sie zum unteren Rand des *Libraries*-Fensters und geben Sie im Feld *Name* den Begriff „Input" ein. Es wird ein Eingabesymbol angezeigt; klicken Sie auf *OK*. Platzieren Sie dieses Symbol irgendwo auf dem Blatt, am besten oben links.

Dann fügen Sie mit einem rechten Mausklick eine zweite Eingabe hinzu und wählen Sie *Insert Symbol*. Da im *Name*-Feld immer noch *Input* stehen sollte, klicken Sie erneut auf *OK* und platzieren Sie auf Ihrem Blatt diese neue Eingabe unter der ersten.
Als nächstes fügen wir ein D-FlipFlop hinzu. Klicken Sie mit der rechten Maustaste, wählen Sie *Insert Symbol* und geben Sie im *Name*-Feld *dff* ein. Wählen Sie das *dff* aus den angebotenen Optionen und klicken Sie auf *OK*, dann platzieren Sie es auf dem Blatt.
Um diesen Schritt abzuschließen, klicken Sie mit der rechten Maustaste, wählen Sie *Insert Symbol*, gehen in das *Name*-Feld den Begriff *Output* ein und klicken Sie auf *OK*. Platzieren Sie auch diesen Ausgang auf dem Blatt.

Pins umbenennen

Wir müssen den Pins aussagekräftige Namen geben. Es ist möglich, Präfixe oder Suffixe hinzuzufügen, um Details wie „i“ für Input, „o“ für Output, „sync“ oder „async“ für synchron beziehungsweise asynchron anzugeben. Wir wollen deshalb den Eingang *pin_name1* in *iDinpAsync* umbenennen. Doppelklicken Sie dazu auf *pin_name1* und geben Sie den neuen Namen ein. Benennen Sie auch *pin_name2* in *iClk* und den Ausgang *pin_name3* in *oDout_sync* um. Klicken Sie dann auf *File* und *Save*.

Hinzufügen von Verbindungen

Klicken Sie auf den Pin *iDinpAsync*. Dadurch wird automatisch das *Orthogonal Node Tool* zum Zeichnen von Verbindungen (Wires) aktiviert. Beginnen Sie mit dem Zeichnen an diesem Pin und verbinden Sie ihn mit dem D-Eingang des D-Flip-Flops (DFF). Verbinden Sie dann *iClk* mit dem Takteingang des DFF, der im Symbol durch ein Dreieck gekennzeichnet ist. Schließlich erstellen Sie eine Verbindung vom Q-Ausgang des DFF zum *oDout_sync*-Ausgangssymbol. Speichern Sie dann erneut.

Kompilieren

Um den Entwurf zu kompilieren, klicken Sie einfach auf das blaue Dreieckssymbol in der Symbolleiste. Behalten Sie das *Task*-Fenster auf der linken Seite im Auge, um den Fortschritt zu verfolgen, und achten Sie auf Meldungen, die am unteren Rand des Bildschirms angezeigt werden. Es ist ein recht kurzweiliger Vorgang.

Nach der Kompilierung wird im mittleren Fenster der Bereich *Flow Summary* angezeigt. Diese Zusammenfassung zeigt, dass nur eines der insgesamt 24.624 Logikelemente verwendet wurde. Es ist selten, dass 100 % der Logikelemente eines FPGAs verwendet werden; in größeren, gut optimierten Designs ist etwa 80 % verwendeter LEs ein realistischer Wert. Wenn diese Anzahl überschritten wird, kann es zu Routing-Schwierigkeiten oder aufgrund von Überlastung zu Timing-Problemen kommen.
Die Zusammenfassung zeigt auch, dass ein Register verwendet wurde und drei von 151 möglichen Pins in Gebrauch sind, was 2 % der Gesamtpins entspricht. Beachten Sie jedoch, dass einige Pins reserviert sein können und nicht zur allgemeinen Verwendung zur Verfügung stehen.
Sehen Sie sich Ihr Projektverzeichnis noch einmal an. Sie werden dort einige neue Unterordner entdecken. Werfen Sie einen Blick in den Ordner *db*, um einen Eindruck von der umfangreich geleisteten Hintergrundarbeit zu bekommen. Im Ordner *output_files* befindet sich die Datei *ElektorFirstTop.sof*, mit der Sie Ihr FPGA programmieren werden.
Als letztes sollten Sie die Datei *ElektorFirstTop.pin* mit einem Texteditor öffnen. Diese Datei zeigt die Pin-Verbindungen, die der

Auswahl des passenden FPGAs für Ihr Projekt

Bei der Auswahl von FPGAs und Boards sind mehrere Kriterien zu berücksichtigen. Eines davon ist die Anzahl der Logikelemente, die eine ungefähre Vorstellung von der Größe des FPGAs vermittelt. Außerdem muss man berücksichtigen, ob man I/Os mit geringer oder hoher Geschwindigkeit benötigt. High-End-FPGAs mit schnellem Speicher und Transceivern sind teuer, ebenso wie die zugehörigen Software-Tools. Natürlich sind die Klassifizierungen von High, Mid und Low-End subjektiv und können von einem Anbieter zum anderen variieren. Was bei den beiden führenden FPGA-Anbietern als mittlere Leistung gilt, könnte bei Lattice als High-End bezeichnet werden.
Wenn es um I/O geht, werden oft Transceiver erwähnt. Dabei handelt es sich um serielle Verbindungen, die hohe Übertragungsraten bieten (zum Beispiel 4...32 Gbit/s), die aber auch die Kosten stark beeinflussen. Sie sind nicht für alle Anwendungen erforderlich, so dass es für viele Anwender anstelle von Hochgeschwindigkeits-PCIe- oder COM-Express-Verbindungen eine kostengünstigere und bequemere Option ist, USB zu verwenden, möglicherweise mit einem FTDI-Wandler als Schnittstelle.
Für Hochgeschwindigkeits-ADC- oder -DAC-Verbindungen zu einem FPGA kann die JESD204B-Schnittstelle verwendet werden, aber dazu sind Transceiver sowohl auf der FPGA- als auch auf der ADC/DAC-Seite erforderlich, was teuer werden kann. Beim Routing der Platine muss zudem auf die Signalintegrität geachtet werden, aber das Layout wird durch die geringere Anzahl von Leiterbahnen etwas vereinfacht.
Eine andere Möglichkeit, wenn große Durchsätze benötigt werden, ist die parallele Verwendung mehrerer langsamerer LVDS-Paare (Low Voltage Differential Signaling). In jedem Fall ist es nützlich, eine ziemlich genaue Berechnung der Durchsatzanforderungen vorzunehmen, um die Auswahl eines geeigneten Bauteils zu erleichtern. Ich neige dazu, die Schnittstellen als Low-Speed zu betrachten, wenn sie unter 1000 Mbps arbeiten, und als High-Speed, wenn sie schneller sind. Die Kosten für FPGAs steigen mit der Anzahl der Hochgeschwindigkeitsschnittstellen. Es gibt jedoch auch viele Fälle, in denen selbst LVDS mit mittlerer Geschwindigkeit nicht erforderlich sind, weil Allzweck-I/O-Pins durchaus ausreichen.
Bei FPGAs stellen die Allzweck-I/O-Pins die langsamsten der verfügbaren Schnittstellen dar, aber sie haben immer noch schnellere Toggle-Raten als Allzweck-I/O-Pins bei Mikrocontrollern. Sie sind in der Regel entweder mit 3,3-V-, 2,5-V- oder 1,8-V Logik kompatibel. Wenn Sie für Ihr nächstes Projekt einen Mikrocontroller verwenden möchten, aber schnellere I/Os benötigen, als sie typische Mikrocontroller bieten können, können FPGAs eine kompakte Lösung sein, da es möglich ist, einen Prozessor (RISC-V oder andere) in die FPGA-Struktur aufzunehmen.

VHDL oder Verilog?

Die Entscheidung, ob ein FPGA-Neuling Verilog oder VHDL lernen sollte, wird von mehreren Faktoren beeinflusst. Für diejenigen, die in der Rüstungs- oder Avionikbranche arbeiten wollen, ist VHDL die maßgebliche Sprache, insbesondere in europäischen Ländern wie Deutschland und Frankreich. In den Vereinigten Staaten und in vielen Bereichen außerhalb des Rüstungssektors wird dagegen eher Verilog verwendet.

Wer jedoch ernsthaft eine Karriere im digitalen Design anstrebt, muss wahrscheinlich beide Sprachen beherrschen. Es ist wichtig, dass Sie sich an Ihren Lernkontext anpassen. Wenn es sich um akademische Zwecke handelt, sollten Sie sich an der Sprache orientieren, die in Ihren Kursen gelehrt wird, während es im beruflichen Umfeld am besten ist, die Sprache zu verwenden, die Ihre Kollegen benutzen.

Aus technischer Sicht ist Verilog tendenziell kompakter und ähnelt in einigen Aspekten C, was nicht immer von Vorteil ist, da es zu einer softwareorientierten Denkweise führen kann, anstatt den Fokus auf die Hardware zu legen. Die ausführliche und starke Typisierung von VHDL kann helfen, bestimmte Fehler zu vermeiden, die bei der schwachen Typisierung von Verilog möglicherweise übersehen werden. Diese Unterscheidung könnte VHDL zu einem besseren Ausgangspunkt für diejenigen machen, die eine Sprache bevorzugen, die einen hardware-zentrierten Denkprozess erzwingt, der für FPGA-Design entscheidend ist.

Compiler eingerichtet hat. Da wir keine besonderen Einschränkungen angegeben haben, hat Quartus die Pins nach dem Zufallsprinzip ausgewählt. Normalerweise würden Sie als Designer Ihre Pin-Zuweisungen definieren. Suchen Sie nach den Pins *iDinpAsync*, *iClk* und *oDout_sync*. In meinem Fall war *iDinpAsync* mit Pin T4 verbunden, arbeitet mit einem 2,5-V-Pegel und befindet sich in Bank 3 des FPGAs.

Programmierung des FPGA

Natürlich sind wir neugierig, ob dies im echten FPGA funktioniert. Probieren wir es also aus! Schließen Sie zunächst Ihr Board an einen freien USB-Port Ihres Computers an und folgen Sie den Schritten zum Hochladen der Konfigurationsdatei in das FPGA-SRAM. Alternativ können Sie die Datei auch in den Flash-Speicher hochladen, so dass das FPGA das Design auch nach einem Stromausfall beibehält.

Gehen Sie dazu zu *Tools* und dann auf *Programmer*, oder doppelklicken Sie einfach auf *Program Device* in der Task-Liste. Klicken Sie auf *Hardware Setup* und wählen Sie *USB-Blaster [USB-0]*. Klicken Sie dann auf *Add File* und wählen Sie die SRAM-Objektdatei *ElektorFirstTop.sof*, die nach der Kompilierung im Verzeichnis *output_files* erzeugt wurde. Klicken Sie auf *Start*. Denken Sie daran, die Konfiguration des Programmiergeräts zum Beispiel als *example1.cdf* zu speichern, damit es bei der nächsten Verwendung mit diesen Einstellungen geöffnet wird. Nach ein paar Sekunden ist die Übertragung abgeschlossen! Wir haben das Design erfolgreich auf das FPGA hochgeladen, wo es verbleibt, bis die Stromversorgung unterbrochen wird. Herzlichen Glückwunsch, Sie haben jetzt ein sehr schickes D-Flip-Flop, das Sie ausprobieren können, indem Sie richtige Drähte und echte Schalter an die entsprechenden Pins des CYC1000-Boards anschließen!

Hierarchischer Entwurf

Es kann sehr nützlich sein, einen Teil eines Systems einmal zu entwerfen, um ihn dann an verschiedenen Stellen des Systems mehrmals zu verwenden. Dazu verpackt man den gewünschten Teil in ein sogenanntes Symbol. Lassen Sie uns als Beispiel ein Symbol für unsere Schaltung entwerfen.

Erstellen Sie zunächst eine neue Blockdiagramm-/Schaltplandatei, wie im obigen Abschnitt „Öffnen des Projekts“ beschrieben: *File*, *New* und so weiter, alles wie gehabt. Nennen Sie das Projekt *Synchronizer.bdf*. Klicken Sie dann im Fenster *ElektorFirstTop.bdf* mit der Maus und halten Sie die Taste gedrückt, während Sie alle Komponenten in einem Rechteck auswählen. Kopieren Sie dann den Inhalt, fügen Sie ihn in die leere *Synchronizer.bdf* ein und speichern Sie. Gehen Sie dann, während das Fenster *Synchronizer.bdf* noch geöffnet ist, auf *File*, wählen Sie *Create/Update*, wählen Sie *Create Symbol Files for Current File* und klicken Sie auf *Save* (mit dem vorgeschlagenen Namen *Synchronizer.bsf*). Klicken Sie in dem Popup-Fenster auf *OK*.

Jetzt wurde ein Symbol für eine DFF-Datei mit dem Namen *Synchronizer.bsf* erstellt. Sie können das *Synchronizer*-Fenster schließen. In *ElektorFirstTop.bdf* können Sie mit diesem neu erstellten Symbol experimentieren und es verwenden. Entfernen Sie das D-Flip-Flop, verschieben Sie die Ein- und Ausgänge an die Seiten, um Platz zu schaffen, und klicken Sie mit der rechten Maustaste in den leeren Bereich, um das Symbol einzufügen, wie wir es im Abschnitt „Hinzufügen von Bauteilen“ getan haben. Das neu erstellte *Synchronizer*-Symbol ist damit im Menü *Insert Symbol* verfügbar. Jetzt können Sie das Symbol mit den entsprechenden Ein- und Ausgängen verbinden und erneut speichern. Dieser hierarchische Entwurf kann nun erneut kompiliert werden.

Verwendung von Hardware-Beschreibungssprachen

Schaltplanentwürfe sind mit zunehmender Größe immer schwieriger zu pflegen und zu aktualisieren. Deshalb bevorzugen die meisten Anwender statt der Eingabe von Blockschaltbildern eine HDL wie Verilog oder VHDL. Es gibt viele Ressourcen, die Sie beim Kennenlernen dieser Sprachen unterstützen.

Konvertieren wir deshalb nun die Datei *Synchronizer.bdf* in eine Verilog-Datei. Klicken Sie auf *Synchronizer.bdf*, navigieren Sie zu *File*, dann zu *Create/Update* und wählen Sie *Create HDL Design File from Current File*. Wählen Sie *Verilog HDL* und klicken Sie auf *OK*. Nun sollte eine *Synchronizer.v*-Datei in Ihrem Design-Verzeichnis erscheinen, die sie mit einem beliebigen Texteditor oder mit Quartus selbst öffnen können. Gehen Sie zu *File*, wählen Sie *Open*, suchen Sie *Synchronizer.v* und öffnen Sie sie. Voilà, jetzt haben Sie die Verilog-Beschreibung Ihrer Synchronizer-Schaltung direkt vor sich. Das kann für Lernzwecke sehr hilfreich sein! Natürlich ist es auch möglich, auf gleichem Wege stattdessen eine VHDL-Datei zu erstellen.

Sie müssen ein neues Symbol für diese neue *Synchronizer.v*-Datei erstellen. Führen Sie dazu die gleichen Schritte aus wie im

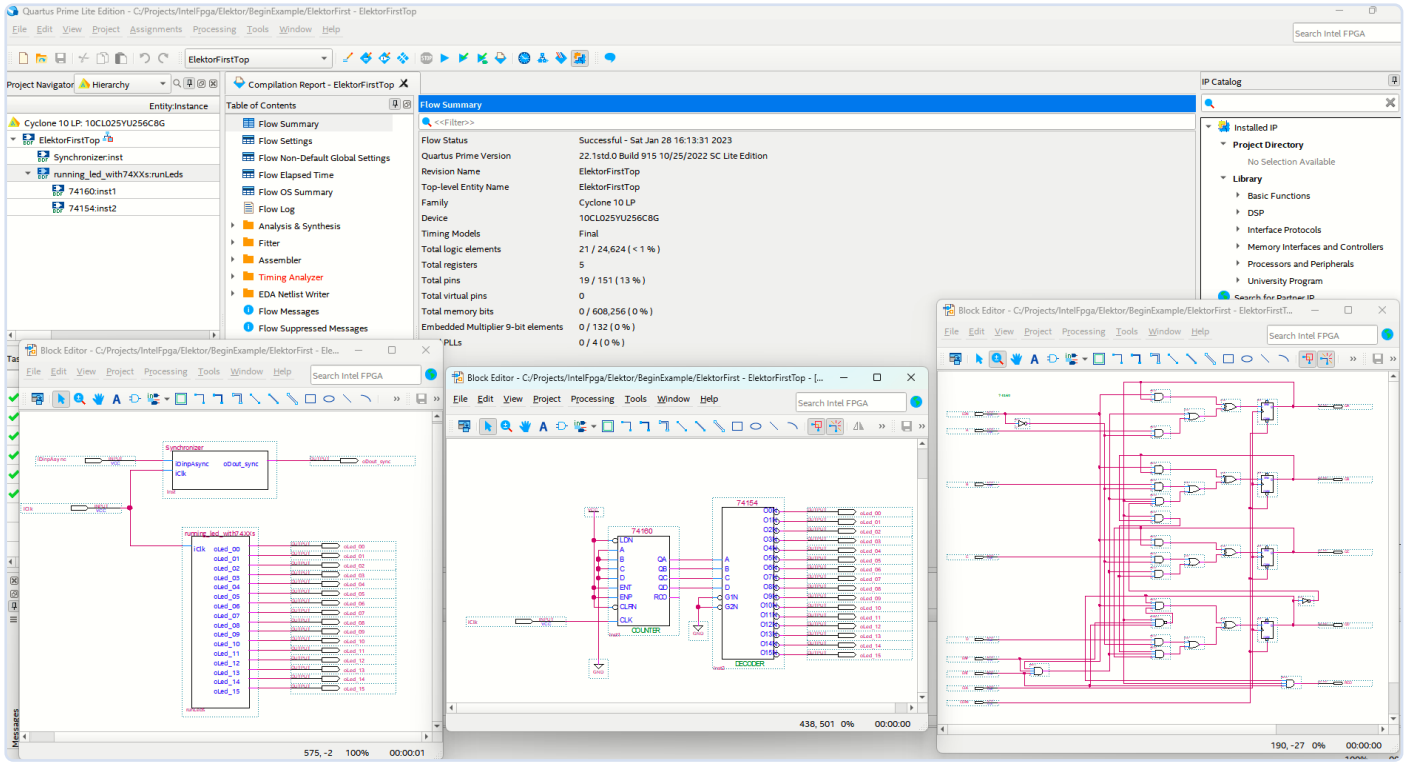


Bild 3. Quartus Lite Edition, mit dem Beispiel running_led_with74XXs.

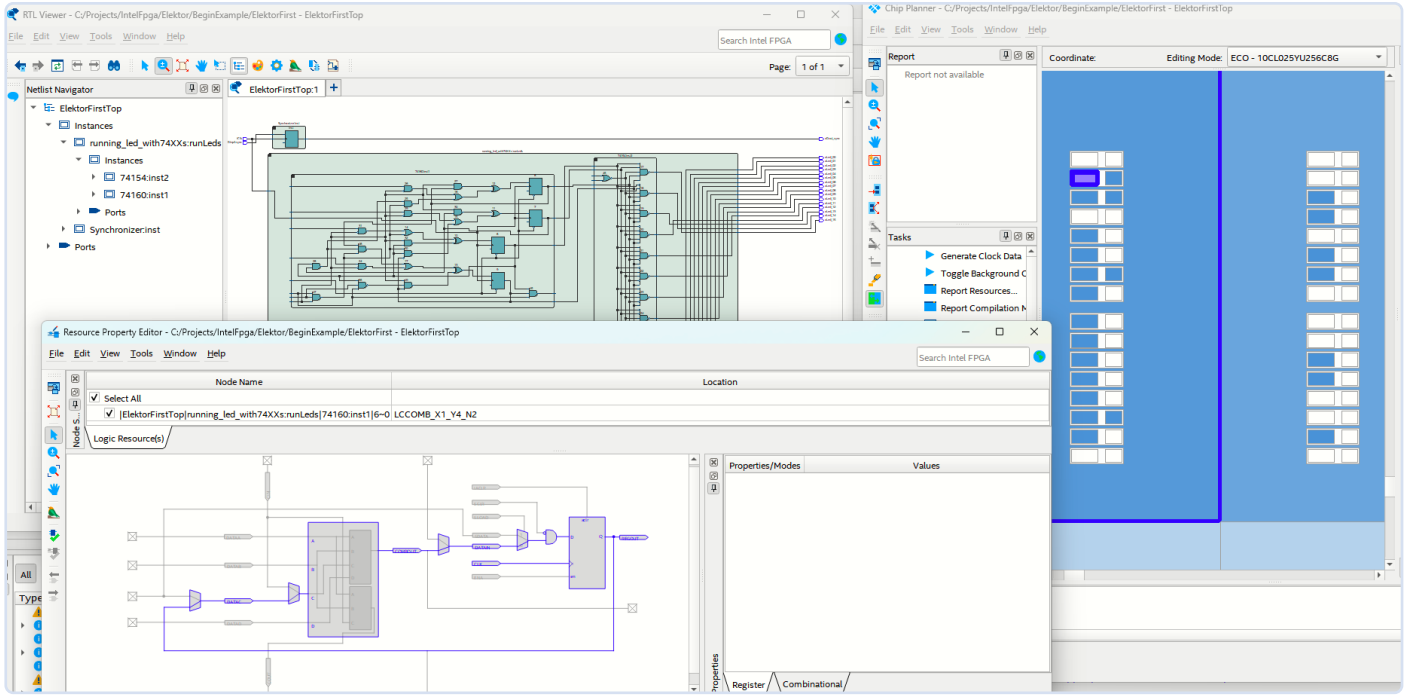


Bild 4. Quartus Lite Edition, mit einigen der Low-Level-Implementierungen des Beispiels running_led_with74XXs.

Abschnitt „Hierarchischer Entwurf“ und verwenden Sie den Befehl *Create Symbol for Current File*.

Eigene Experimente

Sie können nun selbst experimentieren. Klicken Sie mit der rechten Maustaste in *ElektorFirstTop.bdf* auf den Schaltplanbereich, wählen Sie *Insert Symbol* und klicken Sie auf das kleine Dreieck neben dem unteren Element. Tauchen Sie in die in die Logik ein, dann werden Sie eine Reihe von grundlegenden Logikkomponenten wie AND, OR, XOR und so weiter entdecken. Basteln Sie Ihre eigene Schaltung!

Um die Logikschaltungen der Serie 7400 in der Bibliothek zu finden, klicken Sie mit der rechten Maustaste auf den Schaltplanbereich in *ElektorFirstTop.bdf*, wählen Sie *Insert Symbol* und geben Sie „7400“ in das *Name*-Feld ein. Sie finden eine große Auswahl an ICs der 7400er-Serie. Sie können sie durchblättern und die Symbole im rechten Fensterbereich in der Vorschau anzeigen. Es gibt viele interessante Logikschaltungen, die die 7400er-Logik verwenden und die Sie auf einem FPGA emulieren können. Es ist vielleicht nicht die effizienteste Methode, aber es kann sicherlich eine unterhaltsame und lehrreiche Erfahrung sein! In **Bild 3** sehen Sie ein Beispiel für einen Entwurf mit Logik der Serie 74.

Sie können auch einen Blick auf das gesamte Design innerhalb des FPGAs werfen. Wählen Sie in der oberen Leiste *Tools, Netlist viewers, RTL viewer*. Das Ergebnis können Sie in **Bild 4** sehen mit den Schaltplänen oben links. Um die platzierten Elemente innerhalb des FPGAs zu untersuchen, wählen Sie *Tools, Chip planner*. Hier können Sie auf den Bereich zoomen, der die logischen Zellen des Musters enthält. Außerdem ist im Bild oben rechts eine kleine Zelle lila hervorgehoben. Wenn Sie eine logische Zelle auswählen und darauf klicken, können Sie deren Inhalt anzeigen, wie unten links dargestellt.

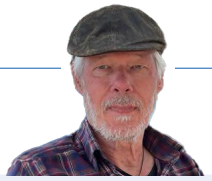
Weiter geht’s

Wir haben mit der Vorstellung des kostenlosen Entwicklungssoftware Quartus Prime Lite nur an der Oberfläche des FPGA-Designs gekratzt. Sie bietet eine komplexe Reihe von Funktionen, von denen wir viele in diesem einführenden Artikel nicht abdecken konnten, zum Beispiel die Simulation, den Signal Tap Logic Analyzer, den Platform Designer und mehr. Es gibt keinen Mangel an zugänglichen FPGA-Boards für Einsteiger, mit denen man experimentieren kann. Für Enthusiasten, die sich für Open-Source interessieren, bietet das Icestorm-Projekt und die grafische IDE Icestudio bemerkenswerte Einstiegsmöglichkeiten in dieses Gebiet. Darüber hinaus gibt es eine Fülle von Online-Ressourcen für das weitere Lernen. Websites wie fpga4fun [3], NandLand [4] und VHDLwhiz [5] sind hervorragende Ausgangspunkte. Joels Liste der budgetfreundlichen FPGA-Boards [6] kann ebenfalls bei der Auswahl der richtigen Hardware helfen.

Und was Bücher betrifft, so wird die digitale Version von *Free Range VHDL* von Bryan Mealy und Fabrizio Tappero von den Autoren völlig kostenlos angeboten, und *Getting Started with FPGA* von Russel Merrick (dem Autor der NandLand-Tutorials) ist ebenfalls eine umfassende Ergänzung für Ihr (virtuelles) Bücherregal.

Die FPGA-Welt liegt Ihnen zu Füßen, und es gibt mehr Lernwerkzeuge und Communities als je zuvor. Jetzt ist ein aufregender Zeitpunkt, um Ihre Reise ins digitale Design zu beginnen!

RG — 230067-02



Über den Autor

Theo Mulder ist ein Elektronikingenieur und sehr vertraut mit Signalverarbeitung sowie mit C++, DSPs und FPGAs. Er hat mit analoger und digitaler Elektronik gearbeitet, insbesondere in der medizinischen Elektronik für Ultraschallgeräte. Er setzt Forschungskonzepte in funktionierende Elektronik um, und einige seiner eigenen Konzepte wurden patentiert. Er ist ein Systemdenker, der auf der Ebene einer Elektronikplatine ein gutes Gleichgewicht zwischen analoger/digitaler Hardware und Software findet, um gut funktionierende Systeme mit einem guten Preis/Leistungsverhältnis zu realisieren.

Haben Sie Fragen oder Kommentare?

Wenn Sie technische Fragen oder Kommentare zu diesem Artikel haben, wenden Sie sich bitte per E-Mail an den Autor unter tf.mulder@outlook.com oder an die Elektor-Redaktion unter redaktion@elektor.de.

Passende Produkte

- **M. Dalrymple, *Microprocessor Design Using Verilog HDL***
  E-Buch, PDF, englisch:
  www.elektor.de/18518
- **Alchitry Au FPGA Entwicklungsboard (Xilinx Artix 7)**
  www.elektor.de/19641

WEBLINKS

[1] Trenz Electronic — CYC1000-Board: https://tinyurl.com/trenzcyc1000
[2] Intel Quartus Prime Design Software: https://tinyurl.com/downloadquartus
[3] fpga4fun: http://fpga4fun.com
[4] NandLand: https://nandland.com
[5] VHDLwhiz: https://vhdlwhiz.com
[6] Joels Liste der FPGA-Boards: https://t1p.de/fbq1w

UPDATE:

STM32 Wireless Innovation Design Contest 2024

Vom der Elektor-Redaktion

Im Rahmen des STM32 Wireless Innovation Design Contest 2024, bei dem Geldpreise im Wert von 5.000 € zu gewinnen sind, haben Innovatoren aus aller Welt in den letzten Monaten hart an STM32-Lösungen gearbeitet und eine Vielzahl kreativer Wireless-Anwendungen entwickelt. Die Nominierungen für die Gewinner werden im März 2024 öffentlich gemacht; die endgültige Bekanntgabe findet auf der embedded world 2024 statt.

Der *STM32 Wireless Innovation Design Contest* bietet Ingenieuren und Entwicklern die einmalige Gelegenheit, ihre Designfähigkeiten unter Beweis zu stellen, indem sie mit den leistungsstarken Entwicklungs- und Evaluierungsboards von STMicroelectronics drahtlose Anwendungen realisieren. Ganz gleich, ob Sie sich für IoT, Robotik, Spiele, Hausautomatisierung oder künstliche Intelligenz begeistern, die Möglichkeiten sind unbegrenzt. Es gibt Geldpreise im Wert von 5.000 € zu gewinnen!

Die Gewinner: Bleiben Sie dran!

Einsendeschluss für den Wettbewerb war der 1. März 2024. Wenn diese Elektor-Ausgabe erscheint, wertet die Jury gerade die eingereichten Projekte aus. Die Gewinner werden dann am Mittwoch, den 10. April um 17:00 Uhr MESZ live auf der embedded world 2024 in Nürnberg sowie online unter *elektormagazine.com/st-contest* bekannt gegeben. Wenn Sie an der embedded world 2024 teilnehmen, sollten Sie auf jeden Fall am Elektor- und am STMicroelectronics-Stand vorbeischauen (embedded-world.de).

Beurteilung

Eine unabhängige Jury wählt die drei besten Projekte anhand folgender Kriterien aus:

- **Kreativität und Innovation**: Die Einzigartigkeit und Originalität des Designs der drahtlosen Anwendung.
- **Technische Exzellenz**: Die technischen Fähigkeiten und Fertigkeiten, die bei der Verwendung des gewählten Entwicklungsboards gezeigt wurden.
- **Funktionalität und Zweckmäßigkeit**: Die Effektivität und Praktikabilität der drahtlosen Anwendung bei der Lösung eines realen Problems oder der Verbesserung der Benutzererfahrung.
- **Ästhetik und Benutzerfreundlichkeit**: Die visuelle Attraktivität, das Design der Benutzeroberfläche und die allgemeine Benutzerfreundlichkeit der Anwendung.
- **Dokumentation und Präsentation**: Die Klarheit, Vollständigkeit und Qualität der Projektdokumentation und -präsentation.

Wir wünschen allen Teilnehmern viel Glück!

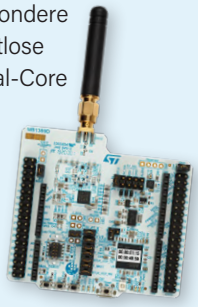
STM32-Technologie

Um an dem Wettbewerb teilzunehmen, mussten die Teilnehmer ein Projekt mit einem der folgenden Boards erstellen: NUCLEO-WBA52CG, STM32WB5MM-DK oder NUCLEO-WL55JC. Der Gedanke dahinter ist, die Fähigkeiten eines Boards zu nutzen, um drahtlose Anwendungen in einem beliebigen Bereich zu entwerfen und zu entwickeln. Die Teilnehmer konnten standardisierte Protokolle wie LoRaWAN, Sigfox und Bluetooth Low Energy (BLE) nutzen oder eigene Protokolle entwickeln.

- Das **NUCLEO-WBA52CG** ist ein drahtloses Bluetooth-Low-Energy- und Ultra-Low-Power-Board mit einer leistungsstarken Funkeinheit mit niedrigster Leistungsaufnahme, die mit der Spezifikation Bluetooth Low Energy SIG v5.3 konform ist. Das Board unterstützt die Konnektivität des ARDUINO Uno V3; die ST-Morpho-Header ermöglichen es, die Funktionalität der offenen Entwicklungsplattform STM32 Nucleo mit einer großen Auswahl an geeigneten Shields auf einfache Weise zu erweitern.

- Das **STM32WB5MM-DK Discovery Kit** ist eine Demonstrations- und Entwicklungsplattform für das STM32W5MMG-Modul von STMicroelectronics. Dieses Modul mit einer Dual-Core 32-Bit Arm Cortex-M4/M0+ CPU verfügt über eine Ultra-Low-Power-Funkeinheit, die mit Bluetooth Low Energy (BLE) 5.2, 802.15.4 mit Zigbee, Thread und proprietären Protokollen kompatibel ist.

- Das Board **Nucleo-WL55JC** ist ein Entwicklungsboard für die STM32WL-Reihe von Mikrocontrollern, insbesondere für den STM32WL55. Dieser sogenannte drahtlose Sub-GHz-Mikrocontroller basiert auf einer Dual-Core 32-Bit Arm Cortex-M4/M0+-CPU mit einer Taktfrequenz von 48 MHz. Er zeichnet sich durch eine extrem niedrige Stromaufnahme, einen HF-Transceiver mit einem Frequenzbereich von 150 MHz bis 960 MHz, 256 KB Flash-Speicher und 64 KB SRAM aus.



Suchen Sie weitere Informationen? Besuchen Sie die Website des STM32 Wireless Innovation Design Contests (*elektormagazine.com/st-contest*), um Details zu den Gewinnerprojekten und vieles mehr zu erfahren.

RG — 240010-02

Bluetooth LE mit MAUI

Steuer-Apps für Android und Co.

Von Tam Hanna (Ungarn)

Eine Smartphone-App eignet sich sehr gut zum Steuern vieler Elektronik-Geräte - die Kommunikation kann stromsparend über Bluetooth LE erfolgen. Jedoch bedeutet das Entwickeln und Warten von Software für Android und iOS einigen Aufwand. Hier kommt das bereits in Elektor vorgestellte MAUI-Framework ins Spiel, mit dem sich Apps plattformunabhängig programmieren lassen. In diesem Artikel zeigen wir, wie man die BLE-Schnittstelle eines Smartphones anspricht.

Der Aschenbecher BopSync, entwickelt von Icy Beats LLC nutzt Bluetooth LE zur Steuerung.

Insbesondere wer auf umfangreiche Kenntnisse im Bereich der .NET-Entwicklung zurückgreifen kann, findet in der Nutzung von MAUI einen bequemen Weg der Software-Entwicklung für Android und iOS - eine allgemeine Einführung ist im Artikel [1] zu finden.
In diesem Artikel wollen wir das Ganze vertiefen, um eine Bluetooth-Schnittstelle unter Android zu programmieren. Angemerkt sei, dass die hier verwendete Bibliothek auch unter iOS funktioniert - die notwendigen Adaptionen würden den Rahmen dieses Artikels aber sprengen, weshalb sie hier unterbleiben.

Einrichtung der Arbeitsumgebung

Dieser Artikel basiert auf einem praktischen Consultingprojekt des Autors, der als freier Entwickler tätig ist. Als „Gegenstelle" dient dabei ein ESP32, der ein auf dem Codebeispiel [2] basierendes Steuerungsprogramm ausführt.
Für die Entwicklung der Smartphone-Applikation kommt Visual Studio 2022 zum Einsatz; das eigentliche Beispielprojekt basiert auf der Vorlage .NET MAUI-App.

Microsoft entschied sich schon vor längerer Zeit, in MAUI (beziehungsweise Xamarin, auf dem MAUI basiert) nicht direkt in das Anbieten einer Bluetooth-API einzusteigen. Eine weise Entscheidung, da Microsoft den alles andere als glatt verlaufenden Entwicklungsprozess der Bluetooth-APIs unter Android wohl aus der ersten Reihe mitbeobachtet hat.
Da eine Schnittstelle existiert, die Xamarin den Zugriff auf native Elemente des Host-Betriebssystems erlaubt, lassen sich allerdings verschiedene NuGet-Pakete mit Bibliotheken verwenden, die diese Scharte auszuwetzen versuchen. In den folgenden Schritten wollen wir auf die unter [3] bereitstehende Bibliothek *Plugin.Ble* setzen.
Amtshandlung Nummero eins ist das Öffnen der NuGet-Konsole, wo Sie die fehlende Bibliothek herunterladen.

Anpassung der Manifestdatei

Cross-Plattform-Umgebungen können den Entwickler prinzipbedingt nur teilweise von der zugrunde liegenden Plattform isolieren. Im Fall der hier verwendeten Bibliothek artikuliert sich dies unter anderem dadurch, dass sowohl für Android als auch für das hier nicht weiter besprochene iOS Anpassungen in den jeweiligen Manifestdateien erforderlich sind. Diese teilen den Betriebssystemen mit, welche Capabilities die Applikation verwenden möchte.
Im Fall der hier verwendeten Visual-Studio-Version 17.6.0 Preview gilt, dass das Anklicken von *AndroidManifest.xml* normalerweise zum Öffnen eines grafischen Editors führt. Dieser ist aber noch nicht wirklich ausgereift, weshalb ein Rechtsklick auf die Datei und das

Öffnen des Manifest-Files in einem Texteditor die bequemere Vorgehensweise darstellt.
Im nächsten Schritt ersetzte beziehungsweise ergänzte der Autor die schon dort befindlichen Deklarationen um einige Strukturen (**Listing 1**). Dank des in Android 6.0 eingeführten „dynamischen Permission-Systems" reicht es an dieser Stelle nicht mehr aus, die Manifestdatei zu beeinflussen. Stattdessen muss die Applikation den Benutzer zur Laufzeit um Genehmigung zum Zugriff auf die empfindlichen Capabilities bitten. Um einem „Hase-und-Igel-Kampf" zwischen der MAUI-Runtime und Googles Einpflegen neuer Permissions zu vermeiden, implementiert Microsoft ein „allgemeines" Interface zur Beschaffung von Permissions. Im Prinzip handelt es sich dabei um eine Klassen-Struktur, die eine Liste der Permission-Konstanten aufnimmt. Der Entwickler muss zur Nutzung eines neuen Permission-Attributs lediglich die Konstante in eine Instanz von `Permissions.BasePlatformPermission` verpacken. Die sonstige Interaktion mit dem Benutzerinterface, die ja nach Schema F abläuft, erledigt die in MAUI enthaltene Runtime.
Unsere nächste Aufgabe ist deshalb das Anlegen einer neuen Klasse vom Typ `Permissions.BasePlatformPermission`, die die diversen benötigten Permissions aufnimmt (**Listing 2**).

Listing 1. Deklarationen.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
              <application android:allowBackup="true" android:icon="@mipmap/appicon" android:roundIcon
               ="@mipmap/appicon_round" android:supportsRtl="true"></application>
              <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
              <uses-permission android:name="android.permission.INTERNET" />
              <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
              <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
              <uses-permission android:name="android.permission.BLUETOOTH" />
              <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
              <uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
              <uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
              <uses-permission android:name="android.permission.BLUETOOTH_ADVERTISE" />
              <uses-feature android:name="android.hardware.bluetooth_le" android:required="true"/>
</manifest>
```

Listing 2. Anlegen einer neuen Klasse vom Typ Permissions.BasePlatformPermission.

```
public class BluetoothLEPermissions : Permissions.BasePlatformPermission
{
  public override (string androidPermission, bool isRuntime)[] RequiredPermissions
  {
   get
   {
     return new List<(string androidPermission, bool isRuntime)>
     {
    (Manifest.Permission.Bluetooth, true),
    (Manifest.Permission.BluetoothAdmin, true),
    (Manifest.Permission.BluetoothScan, true),
    (Manifest.Permission.BluetoothConnect, true),
    (Manifest.Permission.AccessFineLocation, true),
    (Manifest.Permission.AccessCoarseLocation, true),
       //(Manifest.Permission.AccessBackgroundLocation, true),
    }.ToArray();
   }
 }
}
```

Ein wichtiger Hinweis: Diese Permissionauswahl funktioniert mit Android 13. Ältere Varianten und Kindle Fire benötigen ein anderes Permissionskomplement.

Scannen nach BLE-Gegenstellen

Danach können wir uns der eigentlichen Suche nach ansprechbaren Bluetooth-Geräten zuwenden. Der Autor geht in den folgenden Schritten davon aus, dass der Leser mit der Verwendung von Bluetooth LE im Allgemeinen vertraut ist. Unter [4] findet sich eine Kurzeinführung. Für die Auflistung aller in der Umgebung des Transmitters befindlichen Elemente wird der Autor in den folgenden Schritten auf eine ListView setzen: Öffnen Sie die Datei *MainPage.xaml* und fügen Sie an einer bequemen Stelle im Layout die folgende *ListView* ein:

```
<ListView x:Name="deviceList">
  <ListView.ItemTemplate>
   <DataTemplate>
    <ViewCell>
    <StackLayout Margin="20,0,0,0"
          Orientation="Horizontal"
          HorizontalOptions="FillAndExpand">
    <Label Text="{Binding}"
          VerticalOptions="Start"
          TextColor="White"/>
    </StackLayout>
   </DataTemplate>
  </ListView.ItemTemplate>
</ListView>
```

Mit Visual Basic 6 und Co. aufgewachsene Entwickler müssen bei der Arbeit mit MAUI-Listboxen umdenken. Eine derartige Listbox besteht nämlich nicht allein aus der Collection mit den anzuzeigenden Elementen, denn für die eigentliche Anzeige sind als `ItemTemplate` bezeichnete Zusatzinformationen erforderlich. Dabei handelt es sich um ein kleines Stück XAML-Markup, das vom GUI-Stack für jede in der ListView anzuzeigende Datenzeile verwendet wird. Unsere Instanz ist insofern einfach, als dass sie eigentlich nur ein Label auf den Bildschirm bringt. Der String `Text="{Binding}"` baut eine Binding-Beziehung auf, um dem XAML-Parser das direkte Beschaffen der im `Item` anzuzeigenden Elemente zu ermöglichen.
In der XAML-Markupdatei ist außerdem noch ein gewöhnlicher Knopf erforderlich, über den der Benutzer den Scanprozess lostreten soll.
Im nächsten Schritt können wir in den Code Behind zurückkehren, wo wir nach folgendem Schema einige Unterstützungsklassen anlegen:

```
public partial class MainPage : ContentPage
{
   int count = 0;
   IBluetoothLE myBLE = CrossBluetoothLE.Current;
   IAdapter myAdapter =
    CrossBluetoothLE.Current.Adapter;
   public ObservableCollection<String> BTLEDevices;
```

Neben den von der Bibliothek erforderten Variablen ist hier auch eine `ObservableCollection` notwendig. Sie dient der Data Binding-Engine als „Datenquelle" für die Bevölkerung der ListView. Angemerkt sei, dass die Entscheidung pro `ObservableCollection` nicht zufällig fiel — es handelt sich dabei um eine Collection-Klasse, die zum Absetzen von Notifications bei Änderungen am von ihr verwalteten Datenbestand befähigt ist.
Zur Fertigstellung der Data Binding-Beziehung ist im Konstruktor der Mainpage dann noch etwas Housekeeping erforderlich:

```
public MainPage() {
   InitializeComponent();
   BTLEDevices = new ObservableCollection<String>();
   deviceList.ItemsSource = BTLEDevices;
}
```

Im nächsten Schritt folgt die Code-Behind-Methode, die sich um das Anklicken des Scan-Knopfes kümmert. Android implementiert einen Permission-Cache, weshalb wir im ersten Schritt überprüfen, ob unsere Applikation die für die Interaktion mit dem Bluetooth-Transmittter nötigen Berechtigungen bereits aufweist. Ist dies der Fall, so rufen wir die `goScan`-Methode auf, die sich um den eigentlichen Scandurchlauf kümmert:

```
private async void OnScanClicked
   (object sender, EventArgs e) {
   PermissionStatus status = await
     Permissions.CheckStatusAsync
     <BopSyncNetPOC.Platforms.
       Android.BluetoothLEPermissions>();
   if (status == PermissionStatus.Granted){
    goScan();
   }
```

So der Rückgabewert von `CheckStatusAsync` nicht erfolgreich ausfällt, fragen wir den Benutzer nach der Permission:

```
   else {
   await DisplayAlert("Permission required",
          "Google requires the declaration
           of this permission to perform a BTLE scan.
           Please grant it!", "OK");
   status = await
         Permissions.RequestAsync
           <BopSyncNetPOC.Platforms.
              Android.BluetoothLEPermissions>();
   if (status == PermissionStatus.Granted) {
       goScan();
   }
   else {
      await DisplayAlert("Alert",
           "Permission denied.
            Please reinstall application!", "OK");
    }
  }
}
```

Die hier verwendeten umfangreichen Permission-Beschreibungstexte sind erforderlich, um den Benutzer zum Zustimmen zu bewegen.

Die praktische Erfahrung lehrt nämlich, dass insbesondere „technisch herausgeforderte“ Benutzer aufgrund der Medien- und Moral-Panik auf unerwartet aufpoppende Permission-Abfragen meist in höchstem Maße ablehnend reagieren. Außerdem gilt, dass moderne Android-Versionen Ablehnungen mitunter speichern - klickt der User einmal auf Nein, so muss er die Applikation im schlimmsten Fall deinstallieren und neu installieren, bevor er die Permission wieder gewähren kann. Sei dem wie es sei, kommt als nächstes die Methode `goScan` zum Einsatz:

```
private void goScan() {
   if (myBLE.State == BluetoothState.On) {
     myAdapter.ScanMode = ScanMode.LowLatency;
     myAdapter.DeviceDiscovered += FoundDevice;
     myAdapter.ScanTimeout = 12000;
     //MUST be called last or ignores settings
     myAdapter.StartScanningForDevicesAsync();
   }
```

So der Bluetooth-Transmitter bereits aktiv ist, parametrieren wir das Adapter-Objekt und befehlen durch Aufruf von `StartScanningForDevicesAsync` einen Scan-Lauf. Wichtig ist, dass alle für den Scan-Lauf notwendigen Einstellungen in die Klasse geschrieben werden müssen, bevor die Methode aufgerufen wird - spätere Änderungen werden nicht beachtet.
Sofern der Bluetooth-Transmitter nicht aktiv ist, zeigen wir dem Benutzer stattdessen ein Fenster an, das ihn zum Einschalten des Transmitters auffordert:

```
   else {
   DisplayAlert("Alert",
        "Please switch Bluetooth on!", "OK");
   }
}
```

Der Delegat kümmert sich um die „Lokalisierung“ von Geräten. Im Moment wollen wir uns auf die Bevölkerung der Liste konzentrieren:

```
private void FoundDevice(object sender,
                         DeviceEventArgs e){
   try
   {
     BTLEDevices.Add(e.Device.Name.ToString());
   }
   catch(Exception ex)            {
     var aD = e.Device.NativeDevice;
     //Dont muck around with the GUID in ID
     PropertyInfo aProp =
          aD.GetType().GetProperty("Address");
    BTLEDevices.Add("N/A " + aProp.GetValue(aD, null));
   }
}
```

Der Gutteil der Bluetooth-LE-Geräte weist keinen Namen auf. Die hier gezeigte Routine nutzt einen Try-Catch-Block, um in diesem Fall einen „primitiveren“ Default-String in die ListView zu schreiben.

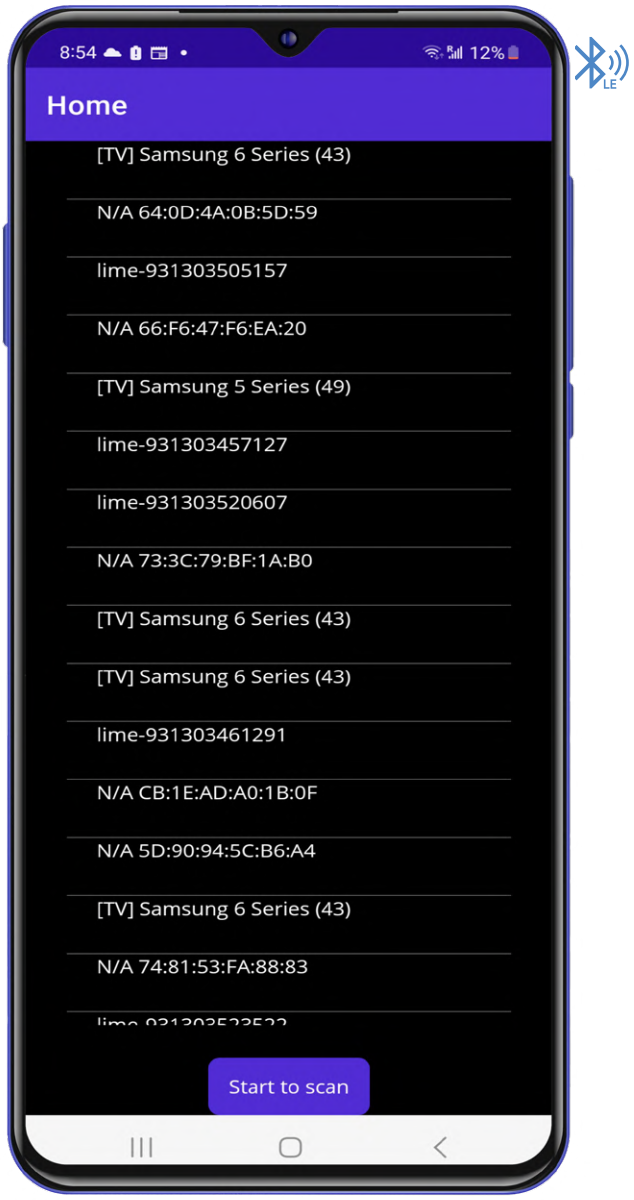


Bild 1. Die Bluetooth-LE-Erkennung funktioniert.

An dieser Stelle ist das Programm für einen Testlauf bereit - **Bild 1** zeigt, wie ein Scanlauf aussieht.
Angemerkt sei, dass das direkte Platzieren von nativen Elementen im allgemeinen Teil der Solution zu Lustigkeiten führt - spezifischerweise treten Fehler auf, die auf das Nichtvorhandensein des Namespace `Platforms.Android` verweisen. Ursache dafür ist, dass manche Operationen in Visual Studio nicht nur die Android-Variante, sondern auch die diversen anderen im MAUI-Projektskelett angelegten Zielplattform-Projekte zu kompilieren versuchen. Zur Umgehung dieses Problems reicht es aus, eine Präprozessorabsicherung um den plattformspezifischen Code zu platzieren:

```
private async void OnScanClicked
      (object sender, EventArgs e) {
   #if ANDROID
   PermissionStatus status . . .
  #endif
}
```

Bild 2. Der tägliche Kampf: Mann gegen Visual Studio.

Ärgerlich ist, dass Visual Studio mit derartigen Elementen seine liebe Not hat, und wie in **Bild 2** gezeigt die Syntax-Vervollständigung deaktiviert.

Identifikation und Verbindungsaufbau

Nun sind wir bereit, Kontrolle über das Peripheriegerät zu übernehmen. Der Autor erweiterte die Mainpage um eine zusätzliche Liste, in der er die einzelnen Bluetooth-Device-Instanzen unterbringt und die die vom Bluetooth-LE-Suchlauf gefundenen Gegenstellen repräsentiert Bei der Applikation des Autors galt, dass immer nur ein Gerät angesprochen werden muss. Aus diesem Grund bietet es sich an, die Instanz in der `Application`-Klasse zu persistieren. Öffnen Sie die Datei *App.xaml.cs* und fügen Sie ein globales Member ein:

```
public partial class App : Application
{
   public Plugin.BLE.Abstractions.
   Contracts.IService myBTLEService;
```

Die vollständige Qualifikation ist dabei vernünftig, weil es in der .NET-Welt oft mehr nur als eine Klasse mit dem Namen `IService` gibt. Wer vollständig deklariert, ist stets auf der sicheren Seite.
Der eigentliche Verbindungsaufbau beginnt dann mit dem Extrahieren der `IDevice`-Klasse aus der weiter oben angesprochenen Geräte-Liste:

```
private async void deviceList_ItemSelected
  (object sender, SelectedItemChangedEventArgs e)
{
   IDevice myDevice =
   BTLEDeviceClasses[e.SelectedItemIndex];
```

Der nächste Akt ist die Beschaffung einer Geräte-Instanz und die Auflistung aller Services:

```
try
{
   await myAdapter.ConnectToDeviceAsync(myDevice);
   var services = await myDevice.GetServicesAsync();
   services = services;
   foreach (IService serv in services) {
     String aString = serv.Id.ToString();
     if (aString.CompareTo
        ("000000ff-0000-1000-8000-00805f9b34fb") == 0)
     { //Swap view
       App curApp = (App) Application.Current;
       curApp.myBTLEService = serv;
       await Navigation.
         PushModalAsync(new MainWorkPage()) ;
     }
   }
}
```

Der Autor entscheidet sich hier dafür, die Identifikation der Gegenstelle durchzuführen, indem er die Verfügbarkeit eines bestimmten Services überprüft. Wenn ein Service mit der benötigten GUI mit aufgefunden wird, erfolgt ein Wechsel der gerade angezeigten Activity.
Wichtig ist die Nutzung der Methode `Navigation.PushModalAsync`. Ursache dafür ist, dass der Autor in seiner kommerziellen Applikation eine von [5] abgeleitete Seite verwendet. Die Nutzung von normalem `PushAsync`, das übrigens in der Microsoft-Dokumentation empfohlen wird, führt im Zusammenhang mit einem nach diesem Schema aufgebauten Fenster zu seltsamen Abstürzen.
Der Try-Catch-Block ist erforderlich, weil er gegen Kommunikationsprobleme absichert:

```
   catch (DeviceConnectionException ex)
   {
```

```
    // ... could not connect to device
    }
}
```

Die nächste Aufgabe der Applikation ist die Kommunikation mit den Charakteristiken. Im Interesse der Reduktion der in den Views enthaltenen Logik setzt der Autor auf die im Flussdiagramm **Bild 3** gezeigte Struktur.
Wer die Kommunikations-Funktion in einer (idealerweise statischen) Klasse platziert, muss sie in den einzelnen Seiten nicht anlegen. Der Autor entschied sich in seiner Applikation abermals für die Nutzung der `App`-Klasse, im nächsten Schritt folgt die Deklaration:

```
public async void setLightMode(int _what)
{
   var x = await myBTLEService.
      GetCharacteristicAsync(Guid.Parse
         ("0000ff01-0000-1000-8000-00805f9b34fb"));
```

Erste Amtshandlung ist dabei die Nutzung der Bibliotheksmethode `GetCharacteristicAsync`, die eine bestimmte Charakteristik anhand ihrer jeweiligen GUID ermittelt. Da wir weiter oben eine Identifikation durchgeführt haben, verzichtet der Autor in diesem Codestück darauf, den Wert von `x` abzusichern. In der Praxis wäre es natürlich vernünftig. Angemerkt sei, dass die Bibliothek auch zur „globalen" Suche befähigt wäre. Der hierfür notwendige Code würde folgendermaßen aussehen:

```
public async void setLightMode(int _what)  {
  var ibx = await myBTLEService.GetCharacteristicsAsync();
```

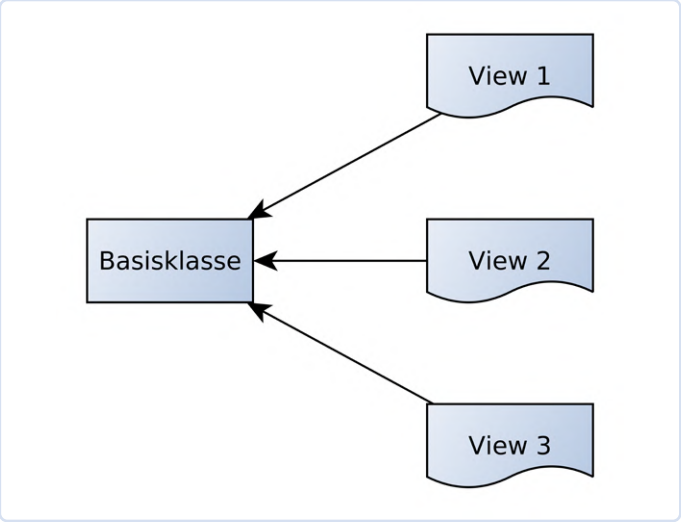


Bild 3. Das „Vereinzeln" der Kommunikations-Logik reduziert die Redundanz im System.

```
  foreach (var chara in ibx) {
     var str = chara.Id.ToString();
    str = str;
  }
...
```

Die in diesen Snippet platzierte tautologische Anweisung ist nützlich, weil sie das „Abernten" der GUIs der einzelnen Charakteristiken erlaubt. Wenn Sie die GUI-Charakteristik nicht in Form von einem String aus

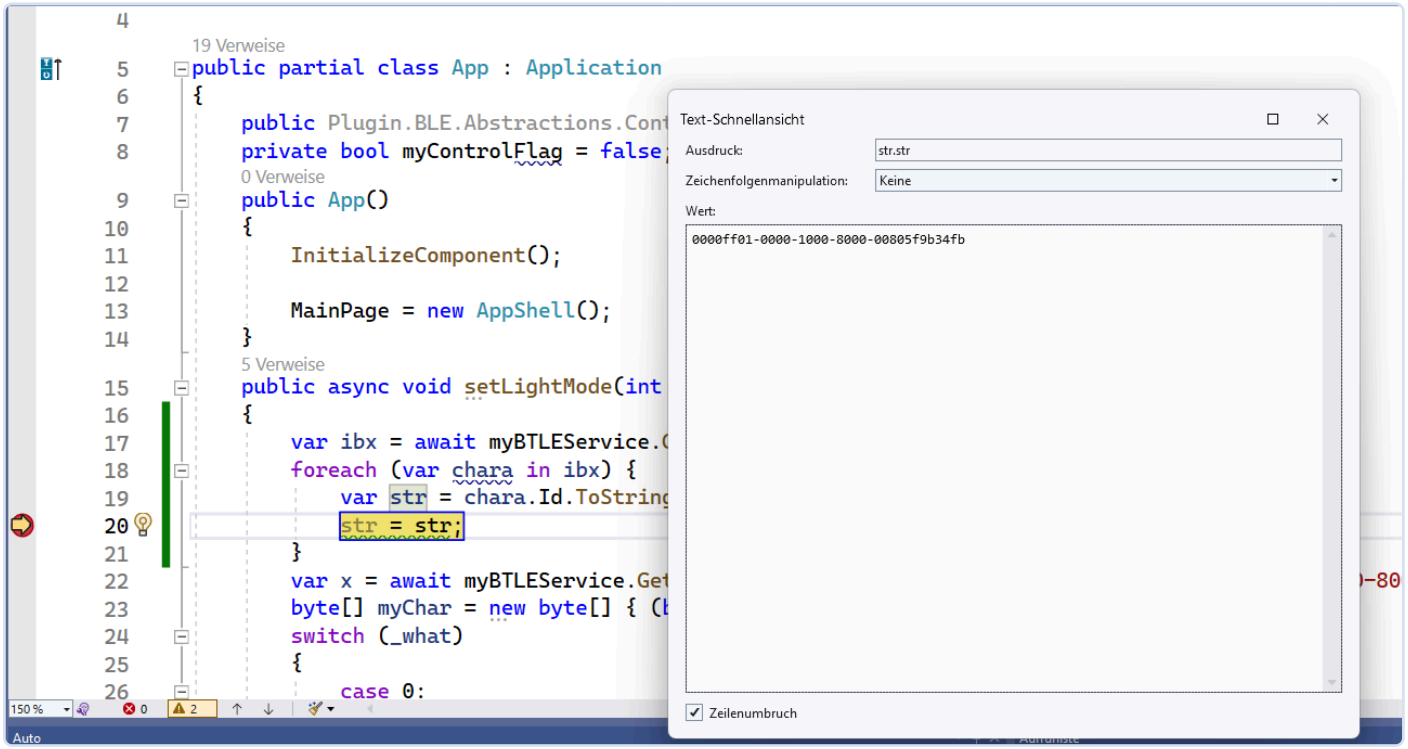


Bild 4. Tautologische Anweisungen ersparen Tipparbeit!

dem ESP32-Programm entnehmen können, ist es eine bequeme Vorgehensweise, auf der tautologischen Anweisung einen Breakpoint zu setzen und das Programm danach auszuführen.
Bei jedem Durchlauf können Sie dann, wie in **Bild 4** gezeigt, im Debugger die jeweilige Charakteristik extrahieren, und (bei Bedarf) mit den im Bluetooth-Scanner angezeigten Informationen vergleichen.

Bluetooth-Scanner sind hilfreich

Wer eine Bluetooth-LE-Applikation realisiert, ist gut beraten, auf seinem Android-Smartphone einen Bluetooth-LE-Scanner zu installieren. Dabei handelt es sich um eine Applikation, die sich mit beliebigen Gegenstellen verbindet und die dort befindlichen Services und Charakteristiken auflistet. Der Autor verwendet in seinem Unternehmen gerne die Applikation *nRF Connect*, die im Playstore kostenlos [7] bezogen werden kann. Aufgepasst: iOS16 und iOS17 liefern hier falsche Ergebnisse!

Sei dem wie es sei, im nächsten Schritt folgt die Zusammenstellung der Payload. Im Fall des Autors basiert das proprietäre Kommunikations-Protokoll auf Datenströmen, die ASCII-Werte übertragen. Das Zusammenbauen der Payload erfolgt dann in Abhängigkeit vom angelieferten Integer-Wert unter Nutzung von Array-Konstanten:

```
byte[] myChar = new byte[] { (byte)'0' };
switch (_what)
   {
   case 0:
     myChar= new byte[] { (byte)'0' };
     break;
   case 1:
     myChar = new byte[] { (byte)'1' };
     break;
   case 2:
     myChar = new byte[] { (byte)'2' };
     break;
   case 3:
     myChar = new byte[] { (byte)'3' };
     break;
   case 4:
     myChar = new byte[] { (byte)'4' };
     break;
...
}
```

Zu guter Letzt ist dann noch ein Verbindungsaufbau-Befehl erforderlich, der sich nach folgendem Schema präsentiert:

```
     var y = await x.WriteAsync(myChar);
   }
```

Daten vom ESP32 zum Android-Telefon übertragen

Als letzte Aufgabe wollen wir das Anliefern von im ESP32 befindlichen Werten über die Luftschnittstelle realisieren. An dieser Stelle möchte der Autor ein wenig mehr auf den ESP-IDF-Code eingehen. Die auf dem GATT-Tabellensystem basierenden Beispiele sind nach Ansicht des Autors miserabel dokumentiert; im Forum finden sich einige verwirrte und nur teilweise beantwortete Fragen.
Am wichtigsten ist in diesem Bereich die Frage, wie die Initialisierung der Charakteristiken erfolgt. Wird wie im Beispiel von Haus aus vorgegeben die Konstante `ESP_GATT_AUTO_RSP` übergeben, so handhabt der Stack die in der Charakteristik befindlichen Werte.
Die Applikation bekommt zwar ebenfalls nach folgendem Schema ein Lese-Ereignis angeliefert; die zurückgelieferten Werte entnimmt der Bluetooth-Stack allerdings aus einem internen Speicher und sendet sie normalerweise sogar vor dem Abfeuern des Events auf die Reise:

```
case ESP_GATTS_READ_EVT:
   ESP_LOGI(GATTS_TABLE_TAG, "ESP_GATTS_READ_EVT");
   break;
```

In der Theorie steht mit der Methode `esp_ble_gatts_set_attr_value` eine Funktion bereit, die das Anpassen der im Bluetooth-Stack vorgehaltenen Speicherwerte erlaubt. Eine naive Implementierung würde sich dann folgendermaßen präsentieren:

```
void updateBTLECache() {
   esp_err_t ret;
   ret = esp_ble_gatts_set_attr_value
     (heart_rate_handle_table[IDX_CHAR_VAL_A],
        1, (const uint8_t *)"1");
}
```

Problematisch ist an dieser Methode, dass sie auch dann den Wert null zurückgeliefert, wenn der übergebene Handle ungültig oder noch nicht vom Bluetooth-Stack initialisiert worden ist.
Damit ist das Problem auch schon beschrieben: Der Aufbau beziehungsweise die Bevölkerung des Handle-Arrays erfolgt nämlich erst vergleichsweise spät. Ein im Unternehmen des Autors für gut befundener Weg besteht darin, die im BTLE-Cache befindlichen Werte immer dann zu aktualisieren, wenn sich ein neues Gerät mit dem ESP32 verbindet. Hierzu ist der nach folgendem Schema aufgebaute Code ideal:

```
case ESP_GATTS_CONNECT_EVT:
   ESP_LOGI(GATTS_TABLE_TAG, "ESP_GATTS_CONNECT_EVT,
            conn_id = %d", param->connect.conn_id);
   updateBTLECache();
   esp_log_buffer_hex(GATTS_TABLE_TAG,
            param->connect.remote_bda, 6);
```

Zu guter Letzt dann hier noch ein Code-Snippet, das das Einlesen der Werte auf MAUI-Seite illustriert:

```
public async void setLightMode(int _what)
{
   var x = await myBTLEService.
        GetCharacteristicAsync(Guid.Parse
             ("0000ff01-0000-1000-8000-00805f9b34fb"));
   byte[] z = await x.ReadAsync();
   z = z;
...
```

Die in Form eines Byte-Arrays angelieferten Werte können Sie dann naturgemäß mehr oder weniger nach Belieben weiter verarbeiten.

Entwicklung beschleunigen

Unsere Experimente zeigen, dass die Bluetooth-LE-API von MAUI problemlos mit anderen Zielsystemen interagieren kann. Wer für sein Embedded-System eine Companion-Applikation benötigt und diese mit MAUI baut, kann in C# oder Visual Basic programmieren und erspart sich einen Gutteil des Kampfes mit den nativen Plattformen. Insbesondere wenn im Unternehmen bereits Kompetenz im Bereich des .NET-Frameworks vorhanden ist, kann dies zu einer wesentlichen Beschleunigung des Entwicklungs-Prozesses führen.
Selbstverständlich ist eine Bluetooth-LE-Steuerung auf dem Smartphone nur die halbe Miete. In einem bereits erschienenen Artikel [6] zeigt der Autor, wie man einen STM32-Mikrocontroller mit BLE-Interface programmieren kann.

230381-02

Sie haben Fragen oder Kommentare?

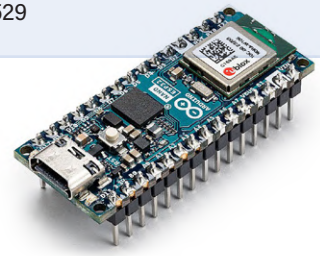
Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Wenden Sie sich bitte per E-Mail an den Autor unter tamhan@tamoggemon.com oder an die Elektor-Redaktion unter redaktion@elektor.de.

Über den Autor

Ingenieur Tam Hanna befasst sich seit mehr als 20 Jahren mit Elektronik, Computern und Software; er ist freiberuflicher Entwickler, Buchautor und Journalist (www.instagram.com/tam.hanna). In seiner Freizeit beschäftigt sich Tam unter anderem mit 3D-Druck und dem Vertrieb von hochwertigen Zigarren.

Passendes Produkt

Arduino Nano ESP32 mit Header
www.elektor.de/20529



WEBLINKS

[1] Veikko Krypczyk, „MAUI: Programmieren für PC, Tablet und Smartphone“, Elektor 1-2/2024: https://www.elektormagazine.de/220442-02
[2] GATT Server Service Table: https://t1p.de/xaumf
[3] Bibliothek Plugin.Ble: https://t1p.de/67fs2
[4] Kurzeinführung in Bluetooth Low Energy (BLE): https://t1p.de/fxprt
[5] Methode Navigation.PushModalAsync: https://t1p.de/fasqw
[6] Tam Hanna, „Bluetooth LE auf dem STM32“, Elektor 1-2/2024: http://www.elektormagazine.de/230698-02
[7] Applikation nRF Connect: https://t1p.de/8nq1e

Breakout-Board für Port-Erweiterung

Mehr I/Os für Ihr Entwicklungssystem

Von Alessandro Sottocornola (Italien)

Elettronica In
WWW.ELETTRONICAIN.IT

Sind Sie es leid, bei der Entwicklung mit einer bevorzugten Plattform Kompromisse aufgrund fehlender I/O-Ports einzugehen? Möchten Sie mehr, um eine Vielzahl von Signalen und Aktoren anschließen zu können? Mit diesem Breakout-Board, das über den I²C-Bus gesteuert wird, können Sie jeden GPIO des Controllers in 16 I/Os verwandeln, bis zu einem Maximum von 128 I/Os.

Es gibt in der Praxis verschiedene Situationen, in denen mehrere Eingangssignale und Aktoren von einem Mikrocontroller gelesen respektive gesteuert werden sollen, der nicht über genügend I/O-Pins verfügt. In solchen Fällen kann man ICs verwenden, die als „I/O-Expander" bekannt sind und deren Funktion darin besteht, eine bestimmte Anzahl von Eingangs- und Ausgangsleitungen hinzuzufügen, die über einen seriellen Bus wie I²C- oder SPI gesteuert werden.

Um den Bedarf an einer Vielzahl von I/O-Leitungen zu decken, haben wir ein Breakout-Board auf der Basis des 16-Bit I/O-Expanders MCP23017 von Microchip [1] entwickelt. Zusätzlich haben wir auf der Platine einige Bauteile untergebracht; einige Pull-up-Widerstände und vor allem einen dreifachen DIP-Schalter, an dem sich die I²C-Adresse des Chips einstellen lässt. Um Ihnen eine Vorstellung davon zu geben, was mit einem I/O-Expander möglich ist, wollen wir in einer praktischen Anwendung des Breakout-Boards eine Reihe von Relais (insgesamt acht) auf einer Platine über TTL-Pegel oder Tastendruck steuern. Aber gehen wir der Reihe nach vor und schauen wir uns zuerst an, wie dieser Chip funktioniert.

Der MCP23X17

Diese vielseitige integrierte Schaltung bietet eine generische serielle/parallele 16-Bit-I/O-Erweiterung und ist in zwei Versionen erhältlich: Der hier verwendete MCP23017 ist mit einer I²C-Bus-Schnittstelle ausgestattet; der MCP23S17 ist die SPI-Variante. Der Chip selbst enthält einen 16-Bit-I/O-Expander, der in zwei Ports zu je 8 Bit aufgeteilt ist und über den I²C-Bus gesteuert wird. Das bedeutet, dass mit nur zwei Drähten und einer Masseleitung die Erfassung des Zustands von bis zu 16 Leitungen (Eingabemodus) oder die Einstellung des logischen Zustands jeder dieser Leitungen im Ausgabemodus möglich ist. Die I/O-Leitungen sind standardmäßig als Eingänge geschaltet.

Der MCP23017 besteht aus mehreren Registern in 8-Bit-Konfiguration für die Auswahl von Eingang, Ausgang und Polarität. Der Master des Systems kann die I/O-Pins als Eingänge oder Ausgänge aktivieren, indem er die entsprechenden I/O-Konfigurationsbits im Register IODIRA/B schreibt. Die Daten für jeden Eingang oder Ausgang werden im entsprechenden Eingangs- oder Ausgangsregister gespeichert. Die Polarität der Eingangsportregister kann in einem Rutsch mit Hilfe des Polarity-Inversion-Registers umgekehrt werden. Alle Register können vom Master des Systems auch gelesen werden. Der 16-Bit-I/O-Port besteht strukturell aus zwei 8-Bit-Ports, nämlich Port A und Port B, die mit den Pins 21...28 beziehungsweise den Pins 1...8 verbunden sind. Der MCP23X17 kann so konfiguriert werden, dass er entweder im 8-Bit- oder im 16-Bit-Modus arbeitet. Außerdem verfügt der Chip über die beiden Interrupt-Pins INTA und INTB, die zwei Betriebsarten zugeordnet werden können:

- Die Interrupt-Pins arbeiten unabhängig voneinander. INTA legt die Interrupt-Bedingung für den Port A und INTB für den Port B fest
- Beide Interrupt-Pins werden aktiv, wenn an einem der beiden Ports ein Interrupt auftritt.

Der Schaltplan

Wie Sie im Schaltplan in **Bild 1** sehen können, ist die Breakout-Platine sehr einfach, denn auf der Platine befindet sich nur die integrierte Schaltung MCP23017 in DIP-Ausführung, bei der alle Pins mit Stiftleisten verbunden sind, die auf der Platine einen Abstand von 2,54 mm haben, damit sie in andere Platinen oder Breadboards eingesetzt werden können. Die drei Pins zur Einstellung der niederstwertigen Bits der I²C-Bus-Adresse der Peripherie sind mit dem dreifachen Dip-Schalter SW1 verbunden, um die Kommunikationsadresse auf dem I²C-Bus einzustellen. Die Pins A0, A1 und A2 werden von den Widerständen R1 bis R3 auf logisch high gehalten, wenn die Schalter offen sind.

Die I/O-Pins von Register A und Register B sind auf beiden Seiten des ICs und auch der Platine angeordnet, so dass diese bequem zum Beispiel auf ein Experimentierboard stecken können. An diese digitalen I/O-Pins dürfen Sie anschließen, was Sie wollen, natürlich innerhalb der Grenzen der vom MCP23017 unterstützten Stromstärke und Spannung. Zusätzlich zu den seitlichen Stiftleisten im 2,54-mm-Raster ist eine vierpolige Stiftleiste (ebenfalls im 2,54-mm-Raster) mit der Bezeichnung I2C vorhanden, so dass wir den Bus direkt an der Stirnseite des Moduls anschließen können. Die Leitungen SDA und SCL sind schon mit Pull-up-Widerständen ausgestattet.

Der I²C-Bus ist inklusive der positiven 5-V-Versorgung und Masse auch auf eine der seitlichen Stiftleitungen gelegt und kann alternativ zum I2C-Anschluss verwendet werden. Auf der Stiftleiste an der anderen Seite der Platine befinden sich statt SDA und SCL die beiden Interrupt-Anschlüsse INTA und INTB. Der Reset-Anschluss des I/O-Expanders wird in unserem Breakout-Board nicht verwendet; um ihn zu deaktivieren, haben wir den entsprechenden Pin 18 (/RES) über den Widerstand R4 auf logisch high gelegt. Die gesamte Schaltung wird über einen der insgesamt drei 5-V-Kontakte (plus Masse) versorgt.

Der I/O-Expander

Das Hauptelement der Schaltung ist natürlich der I²C-Bus/Parallelwandler MCP23017 von Microchip (mit U1 gekennzeichnet). Das IC, dessen internes Blockdiagramm in **Bild 2** zu sehen ist, funktioniert als Empfänger (Slave) des I²C-Busses und unterstützt zwei Eingangs- und Ausgangsmodi. Im ersten Modus können die I/O-Zustände der Register A und Register B auf Anforderung des I²C-Bus-Masters, des Controllers in seriellem Format (ein Byte pro Register) auf den Bus übertragen werden. Im zweiten Modus werden die I/O-Leitungen

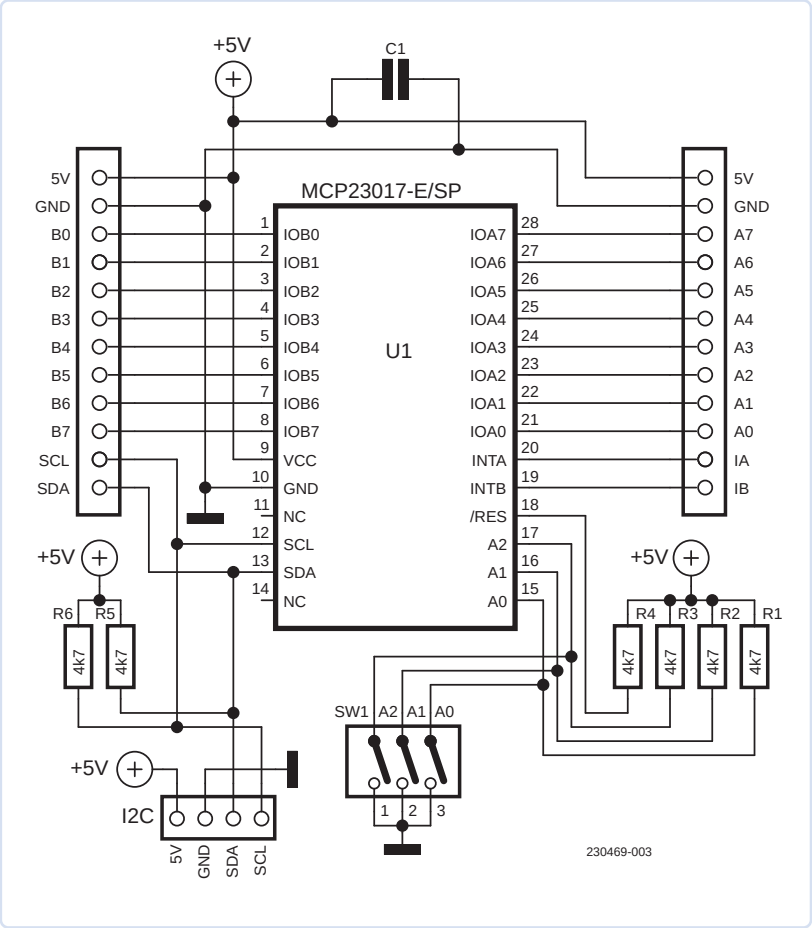


Bild 1. Der Schaltplan des Breakout-Boards.

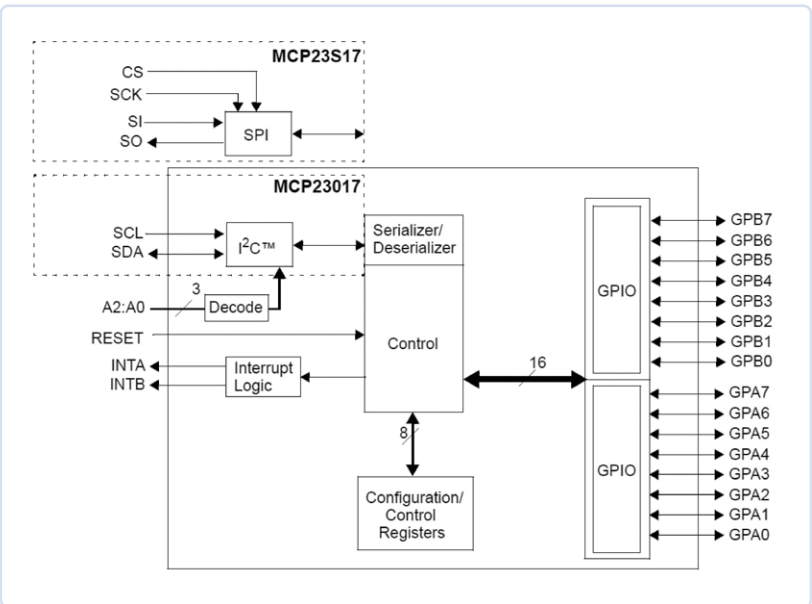


Bild 2. Blockschaltbild des MCP23017. Die SPI-Variante, die die Bezeichnung MCP23S17 trägt, ist ebenfalls abgebildet. (Quelle: Microchip [3])

Der MCP23017 in Kürze

Der Chip auf dem hier beschriebenen Breakout-Board ist ein I/O-Expander mit 16 Bit, aufgeteilt in zwei Ports zu je 8 Bit, die über den I²C-Bus miteinander verbunden sind. Das bedeutet, dass man mit einem Zweidrahtbus und einer Masseleitung den Status von bis zu 16 Leitungen erfassen oder als Ausgang setzen kann.

- Die Hochgeschwindigkeits-I²C-Datenschnittstelle arbeitet mit einer Busfrequenz von 100 kHz, 400 kHz oder 1,7 MHz
- Die I²C-Busadresse ist in acht Varianten einstellbar
- Konfigurierbare Interrupt-Pins, nach Pegel und logischer Funktion
- Konfigurierbare Interrupt-Quelle
- Register für Polaritätsumkehr

Für die Eingänge:

- Externer Reset-Eingang
- Standby-Strom von max. 1 µA
- Versorgungsspannung von 1,8...5,5 V

durch Umwandlung der auf dem I²C-Bus eingehenden Daten in den entsprechenden Zustand der Leitungen von Register A und Register B eingestellt.
Der Interrupt-Ausgang kann so konfiguriert werden, dass er unter zwei (sich gegenseitig ausschließenden) Bedingungen auslöst:

- Wenn sich ein Eingangszustand von seinem entsprechenden Eingangsport-Registerzustand unterscheidet. Diese Bedingung wird verwendet, um dem System-Master anzuzeigen, dass sich ein Eingangszustand geändert hat.
- Wenn sich der Zustand eines Eingangs von dem vorkonfigurierten Wert des DEFVAL-Registers unterscheidet.

Die Interrupt-Leitungen INTA und INTB können als Active-High, Active-Low oder Open-Drain konfiguriert werden. Das Interrupt Capture Register erfasst die Werte der Ports zum Zeitpunkt der Auslösung des Interrupts und speichert so die Bedingung, die den Interrupt ausgelöst hat. Der Power-On-Reset (POR) setzt die Register auf ihre Standardwerte zurück und initialisiert die Zustandsmaschine des Geräts. Die Notwendigkeit des bidirektionalen Betriebs ergibt sich aus der Tatsache, dass jeder I²C-Bus-Peripheriebaustein in der Lage sein muss, sowohl (zum Beispiel Befehle) zu lesen als auch erfasste 8+8-Bit-Daten über den Bus zum Master zu senden.
Wie die meisten Bausteine für den I²C-Bus ermöglicht der MCP23017 die Einstellung seiner Adresse. Damit mehrere der gleichen ICs individuell über den I²C-Bus angesprochen werden können, verfügt es über die Pins A0, A1, A2 zur Einstellung der unteren drei Bits des Adresswortes. Jede dieser Leitungen wird über den Dip-Schalter SW1 eingestellt: Jeder geschlossene Schalter setzt die logische Null auf die jeweilige Adresse, während ein offener Schalter den logischen Zustand Eins bestimmt. Es lassen sich acht Adressen einstellen, also acht gleiche ICs am Bus anschließen, was bedeutet, dass bis zu 128 I/Os mit nur drei Leitungen gesteuert werden können. Wenn Sie mehrere Breakout-Boards einsetzen wollen, geben wir Ihnen in **Tabelle 1** den Zusammenhang Adressen und Einstellung der Dip-Schalter an.

Tabelle 1: Einstellung der Peripherieadresse des MCP23017

ADDR	A2	A1	A0
0x20	EIN	EIN	EIN
0x21	EIN	EIN	AUS
0x22	EIN	AUS	EIN
0x23	EIN	AUS	AUS
0x24	AUS	EIN	EIN
0x25	AUS	EIN	AUS
0x26	AUS	AUS	EIN
0x27	AUS	AUS	AUS

Der MCP23017 funktioniert wie folgt: Immer wenn er einen String auf der SDA-Leitung des I²C-Busses empfängt (getaktet durch das Taktsignal auf der SCL-Leitung), führt der MCP23017 den darin enthaltenen Befehl aus (in diesem Fall den, der das Laden des Datenbytes anzeigt) und ordnet die acht Ausgangsleitungen IOA0...IOA7 und IOB0...IOB7 als die entsprechenden Bits an. Zum Beispiel nimmt IOA0 den Zustand des ersten Bits von Byte 1 an, IOA1 den des zweiten Bits und so weiter. Dasselbe geschieht mit IOB0...IOB7, die genau die Bits des zweiten Datenbytes wiedergeben.
Umwandlung und Ausgabe erfolgen natürlich nur unter der Voraussetzung, dass der empfangene String die I²C-Bus-Adresse enthält, die der mit den Dip-Schaltern von SW1 für U1 eingestellten Adresse entspricht. Beim Empfang jedes Strings aktualisiert das IC den Status seiner Ausgänge, und die jeweiligen logischen Pegel bestimmen, ob die nachgeschalteten Schaltungen (LEDs, Segmentanzeigen oder halt Relais) eingeschaltet werden oder im ausgeschalteten Zustand verbleiben. Wenn anschließend kein String gesendet wird, behält der Ausgangsstatus das letzte Bytemuster bei, da die Ausgänge des MCP23017 verriegelt sind. Dies gilt für den Ausgabemodus, das heißt das Schreiben der Status der beiden I²C-Bus-Bytes in die Ausgaberegister A und B. Erreicht das IC über den Bus dagegen ein Lesebefehl, so erfasst der MCP23017 den I/O-Status jedes Registers und generiert zwei Bytes, von denen das erste den Status von IOA0...IOA7 und das zweite den logischen Zustand von IOB0...IOB7 enthält, und sendet sie als Antwort über den I²C-Bus.

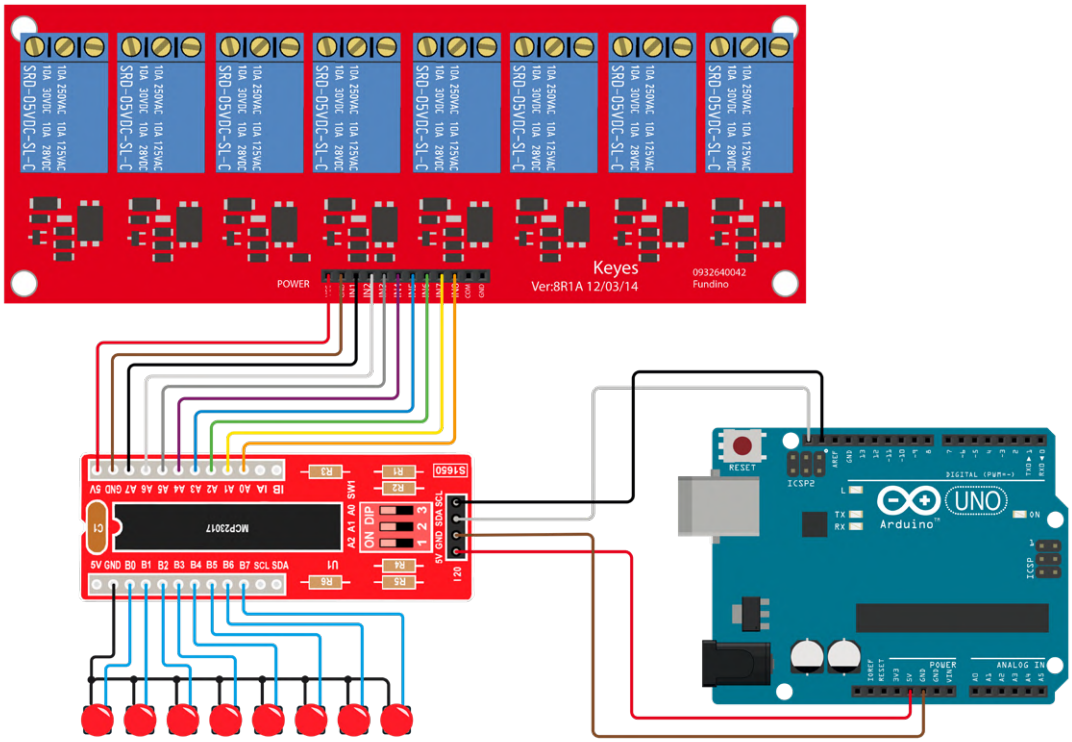


Bild 3. Verdrahtungsplan der Anwendung zur Steuerung der Relais.

Stückliste

Widerstände:

R1…R6= 4k7

Kondensator:

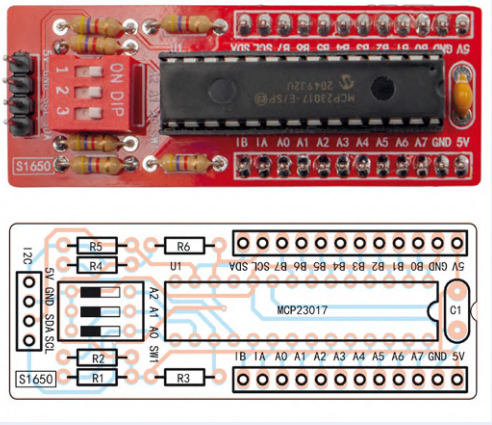
C1 = 100 n, keramisch

Halbleiter:

U1 = MCP23017-E/SP

Außerdem:

SW1 = 3-poliger Dip-Schalter
1 St. 2×14-polige DIL-Fassung
2 St. 1×12-polige Stiftleiste
1 St. 1×4-polige Stiftleiste
Platine (siehe Text)



In der Praxis

Nun, da wir den Schaltplan beschrieben haben, können wir zur Konstruktion übergehen. Danach werden wir ein Anwendungsbeispiel auf der Grundlage einer Schnittstelle mit einem Arduino-Board vorschlagen, zusammen mit dem entsprechenden Sketch. Wie üblich haben wir eine (doppelseitige) Platine entworfen, deren beide Layouts auf der Elektor-Labs-Projektseite unter [2] zum Download zur Verfügung stehen. Anhand dieser Entwürfe können Sie die Platine selber ätzen und bohren und die wenigen benötigten Bauteile montieren. In diesem Projekt kommen nur traditionelle bedrahtete Bauteile zu Einsatz, zur Freude derjenigen, die das kleine SMD-Hühnerfutter nicht so sehr mögen. Zuerst werden die Widerstände und der Kondensator eingesetzt und verlötet, dann folgt die IC-Fassung (mit der Kerbe nach rechts, wie im Bestückungsplan zu sehen) und der dreifache Dip-Schalter mit dem Schalter 1 nach links.

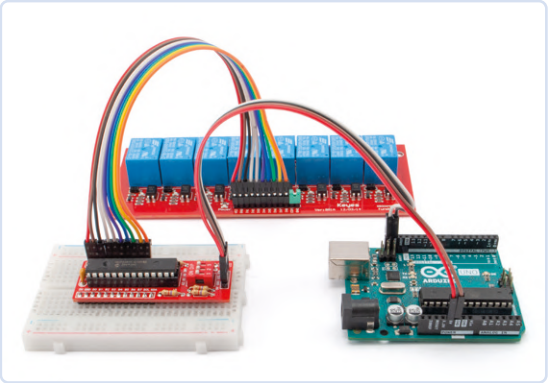


Bild 4. Die Hardware zum Testen des Beispielsketches.

Listing 1: Demoprojekt

```
#include <Adafruit_MCP23X17.h>.
#include <Adafruit_MCP23X17.h>
Adafruit_MCP23X17 mcp;
int i = 0;
int OUT[] = ;      //Represents MCP23017 PIN (A7...A0)
int IN[] = ;       //Represents MCP23017 PIN (B0...B7)
int STATO[] = ;    //For each output, every toggle the status is being saved
void setup()
{
 Serial.begin(9600);
 Serial.println("MCP23017 INPUT/OUTPUT");
 if (!mcp.begin_I2C(0x20))         //0x20 is MCP23017's address with A0=A1=A2 > ON(GND)
 {
    Serial.println("MCP Error!");  //If MCP is not found, the error is visualized
    while (1);
 }
 //bank A Pin set as outputs and B bank as inputs
 //The STATO variable to 0 to indicate idling outputs
 for (i=0; i<8; i=i+1)
 {
    mcp.pinMode(OUT[i], OUTPUT);
    mcp.pinMode(IN[i], INPUT_PULLUP);
    STATO[i] = 0;
 }
}

//******************* L O O P **********************
void loop()
{
 String Testo_Debug = "";
 for (i=0; i<8; i=i+1)
 {
    //If button pressed or output not activated, I activate it
    if ((mcp.digitalRead(IN[i])==0) && (STATO[i]==0))
    {
      STATO[i] = 1;
      Testo_Debug = "Pulsante " + String(i+1) + " premuto";
      Serial.println(Testo_Debug);
      mcp.digitalWrite(OUT[i], HIGH);
    }
    //If button released and output is active,I de-activate it
    if ((mcp.digitalRead(IN[i])==1) && (STATO[i]==1))
    {
      STATO[i] = 0;
      Testo_Debug = "Pulsante " + String(i+1) + " rilasciato";
      Serial.println(Testo_Debug);
      mcp.digitalWrite(OUT[i], LOW);
    }
 }
 delay(10);
}
```

Schließlich wird die vierpolige Stiftleiste an der Aufschrift I2C in die entsprechenden Löcher gesteckt und verlötet, dann kommen die beiden 1×12-poligen Stiftleisten an den Langseiten an die Reihe. Die Steckleisten ermöglichen die Montage auf Lochrasterplatinen oder das Einsetzen auf andere Platinen, um schließlich mit Hilfe von klassischen männlichen/weiblichen Drahtbrücken oder DuPont-Drähten eine Verbindung mit dem Arduino herzustellen. Sobald Sie mit dem Löten der Bauteile fertig sind, setzen Sie den MCP23017 in die Fassung, mit der Kerbe wie im Montageplan gezeigt. Damit ist Ihr Breakout-Board bereit für Experimente oder Prototypen.

Der Arduino steuert

Das Breakout-Board wurde für den Anschluss an einen Mikrocontroller entwickelt, da I/O-Expander typischerweise von Controllerschaltungen verwendet werden, die über eine serielle I²C-Bus-Schnittstelle verfügen. Da der Arduino diesen Bus unterstützt, haben wir einen Beispielcode erstellt, um die I/Os des MCP23017 über Arduino zu lesen und zu steuern. Der Sketch kann ebenfalls unter [2] heruntergeladen werden.

Er nutzt, wie man in der ersten Zeile des Sketches (**Listing 1**) sehen kann, die Vorteile der Adafruit-Bibliothek für den MCP23017. Der Sketch ermöglicht es im Grunde, den Zustand des Registers von Port A in Abhängigkeit von einem vom Arduino über den Bus gesendeten Byte zu schreiben, dessen Bits dem an Port B gelesenen Zustand entsprechen, der dieses Mal als Eingang dient. Um dem Beispiel eine konkrete Anwendung zu geben, wollen wir die logischen Zustände der I/Os von Port A, die hier als digitale Ausgänge fungieren, zur Ansteuerung einer Relaisplatine verwenden. Dazu müssen wir die acht Relaisausgangs-Steuerleitungen einer 8-Kanal-Relaisplatine an die I/O-Bank von Port A anschließen. An Port B werden acht normalerweise offene Drucktaster angeschlossen, wobei der jeweils freie Pol gemeinsam mit GND verbunden ist. Für diese Anwendung müssen der Arduino UNO, die Breakout-Platine, die Relaisplatine und die Taster wie in **Bild 3** angeschlossen werden. **Bild 4** zeigt den Prototyp unserer Schaltung in natura. Da es sich um ganz gewöhnliche Drucktasten ohne externe Elektronik handelt, wurden (über die Bibliothek) die internen Pull-ups des MCP aktiviert. Das reicht aus, um die Taster zu verarbeiten und jede Zustandsänderung zu erkennen, so dass wir den entsprechenden Ausgang aktivieren können, wenn die Taste gedrückt wird und die Verbindung zu Masse (GND) herstellt.

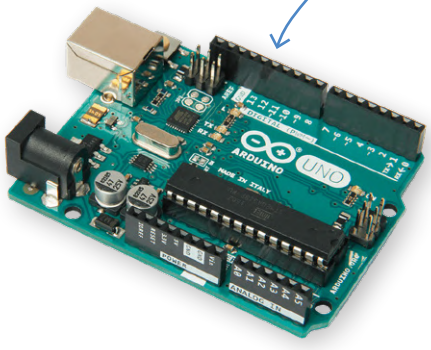
Bevor wir den Code in den Programmspeicher unseres Arduino-Boards laden, ist es wichtig, die Bibliothek von [5] herunterzuladen und sie mit dem in der IDE enthaltenen Bibliotheksmanager zu installieren oder einfach den Inhalt der ZIP-Datei zu entpacken und dann den gesamten Ordner *Adafruit_MCP23017_Arduino_Library* in das Bibliotheksverzeichnis zu kopieren, das sich normalerweise im Betriebssystem unter dem Pfad *Documents\Arduino\libraries* befindet. Nach dem Laden der Bibliothek reicht es aus, unseren Beispielcode in das Board zu laden, nachdem der richtige COM-Port aus dem *Tools*-Menü der IDE ausgewählt wurde.

Im Code wird ein Byte angefordert, das an den MCP23017 gesendet wird und den Status der Tasten enthält. Die entsprechenden Daten werden verarbeitet und dann in ein Byte geschrieben, das an den MCP23017 gesendet wird, der den Status der Port-A-Leitungen bis zur folgenden Auffrischung stabil einstellt. Damit die Schnittstelle funktioniert, müssen die Dip-Schalter A0, A1 und A2 korrekt eingestellt sein, denn wenn dem IC nicht die Adresse 0x20 zugewiesen wurde, würde eine Fehlermeldung auf der seriellen Schnittstelle erscheinen. Der einstellbare Adressteil der Breakout-Platine, also die drei Bits A0, A1, A2, ist auf 000 eingestellt, das heißt, alle drei Dip-Schalter sind mit Masse verbunden. Wenn Sie die Adresse ändern wollen, sehen Sie in Tabelle 1 nach, wobei Sie natürlich auch Adresse im Sketch ändern müssen.

RG — 230469-02

Passendes Produkt

› **Arduino UNO Rev3**
www.elektor.de/15877



WEBLINKS

[1] Microchip-Webseite für den MCP23017: https://microchip.com/en-us/product/mcp23017
[2] Dieses Projekt auf Elektor Labs: https://t1p.de/7ueor
[3] Datenblatt MCP23017/MCP23s17 (PDF): https://tinyurl.com/mcp23017datasheet

KI-Spezialist

Machine Learning mit dem Jetson Nano

Von Tam Hanna (Ungarn)

KI auf GPUs - das muss nicht immer sündhaft teure Grafikkarten und immensen Energieverbrauch bedeuten. NVIDIA geht seit einiger Zeit in Richtung kleinerer Systeme, sicher auch aufgrund der Konkurrenz durch die Mikrocontrollerhersteller. In diesem Artikel wollen wir einen Blick auf den Jetson Nano werfen. Natürlich gehört auch eine kleinen Demo-Anwendung dazu.

Der Markt für KI-Beschleunigung in Embedded-Systemen ist im Fluss: Unternehmen wie Canaan und Maxim Integrated kämpfen mit harten Bandagen um die Vorherrschaft; und auch ARM hat mit der Ankündigung des Cortex M52 einen Gladiator in diese Arena getrieben.

Ziel ist durch die Bank das Anbieten von kleinen KI-Systemen, die an der viel besungenen *Egde* grundlegende KI-Aufgaben ohne Verbindung zum Internet erledigen. Dies führt zu einer wesentlich stabileren Systemarchitektur, weil die ML- beziehungsweise KI-Aufgaben dann auch bei Verlust der Verbindung zum Mutter-Server weiterbearbeitet werden können. Mit der Jetson-Serie versucht NVIDIA seit einiger Zeit, in diesem Marksegment mitzuspielen.

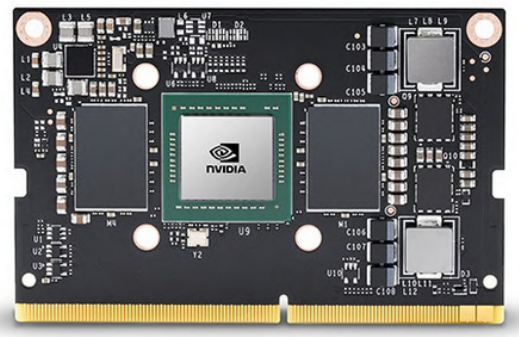
Im Interesse der didaktischen Ehrlichkeit sei angemerkt, dass eigene Platinendesigns mit diesen Systemen schon wegen der extremen Bandbreite beim Zugriff auf den DDR-Arbeitsspeicher für kleinere Unternehmen nicht wirklich handhabbar sind.

NVIDIA ist sich dieser Problematik bewusst. Auf der unter [1] bereitstehenden Portfolio-Übersicht, die dem Elektroniker einen Überblick über das gesamte Jetson-Ökosystem anbietet, finden sich deshalb auch verschiedene „Compute cards“, so wie der in **Bild 1** gezeigte Jetson TX2.

Angemerkt sei außerdem noch, dass das von NVIDIA aufgrund der Dominanz im GPU-Bereich gut ausgebaute Drittanbieter-Ökosystem mittlerweile auch die Jetson-Produktpalette für sich entdeckt hat. Unter [2] findet sich eine Liste verschiedenster Drittanbieter-Produkte, deren Integration in eine hauseigene Lösung sehr viel Entwicklungszeit (man denke beispielsweise an Gehäuse-Design und Auswahl von Kamera und Co.) einsparen kann.

Inbetriebnahme des Systems

Zum „Quereinstieg“ bietet NVIDIA das in **Bild 2** (neben einem Raspberry Pi und einem Orange Pi 5+) gezeigte Jetson Nano Developer Kit an. Zum Zeitpunkt der Drucklegung dieses Artikels beträgt der OEMSecrets-Bestpreis (siehe [3]) 155 €. Das ist zwar



Jetson TX2 Series

The extended Jetson TX2 family of embedded modules gives you up to 2.5X the performance of Jetson Nano in as little as 7.5W. Jetson TX2 NX offers pin- and form factor compatibility with Jetson Nano, while Jetson TX2, TX2 4GB, and TX2i all share the original Jetson TX2 form factor. The rugged Jetson TX2i is ideal for settings including industrial robots and medical equipment.

Learn More ›

Bild 1. Die Nutzung dieses vorgefertigten Moduls erleichtert die Integration des Jetson in hauseigene Schaltungen erheblich.

etwas teurer als ein Raspberry Pi, doch ist die NVIDIA-Technologie besser in das allgemeine KI-Ökosystem eingebunden. Beim Kauf des NVIDIA Jetson Nano ist es empfehlenswert, ein Netzgerät mitzubestellen, das einen DC-Hohlstecker mit den gebräuchlichen Maßen 5,5/2,1 mm aufweist. In den folgenden Schritten verwendet der Autor ein MeanWell-Fabrikat (GST25E05-P1J).
Bei sorgfältiger Betrachtung des Systems (siehe auch **Bild 3**) fällt auf, dass es sich um ein zweiteiliges Produkt handelt. Neben dem Träger-Board, das die diversen Schnittstellen exponiert, findet sich das eigentliche Rechenmodul mit dem Kühlkörper. Relevant ist, dass eine mit dem Betriebssystem ausgestattete Micro-SD-Karte in das Rechenmodul gesteckt werden muss.

Theoretisch auch Micro-USB möglich

Wer ein sehr leistungsfähiges Micro-USB-Netzgerät im Haus hat und den Micro-USB-Port nicht zum Anschließen externer Hardware verwendet, kann auch diesen Weg der Stromversorgung wählen. Aufgrund der „Quereleien" mit dem Raspberry Pi scheut der Autor als gebranntes Kind allerdings das Feuer und setzt auf das klassische DC-Netzgerät.

Das verbaute SoC ist ein Mehrkernsystemen, das neben dem eigentlichen KI-Beschleuniger auch vier vollwertige Arm Cortex A57-Kerne mitbringt. Daraus folgt ein an Prozessrechner erinnerndes Setup, auf dem normalerweise Embedded Linux läuft. NVIDIA empfiehlt die Verwendung einer MicroSD-Karte mit mindestens 32 GB Kapazität und einer Mindestgeschwindigkeit von UHS1. Unter [4] findet sich die Imagedatei, die Sie - wie gewohnt - unter Nutzung eines Cardreaders auf die auserkorene Speicherkarte extrahieren.
Für die erstmalige Inbetriebnahme ist außerdem eine USB-Maus und eine USB-Tastatur erforderlich; im Bereich der Bildschirm-Ausgabe unterstützt Jetson sowohl HDMI als auch Displayport. Dass das Anschließen eines Ethernet-Kabels ebenfalls „vernünftig" ist, wollen wir in den folgenden Schritten als gegeben annehmen.

Grobmotoriker halt!

Der Micro-SD-Halter des Jetson-Moduls basiert auf dem Formschlussprinzip. Das Einschieben und Entfernen erfolgt durch „tiefes Hereindrücken" - wer eine Elektronikerpinzette und einen feinen Schraubenzieher verwendet, bekommt die Operation ohne Zerlegen des Evaluationsboards hin.

Problematisch erweist sich bei der Inbetriebnahme das Fehlen einer Kamera. Statt einem CCD-Sensor verbaut man im Hause NVIDIA zwei der von Raspberry Pi 4 und Co. bekannten Steckverbinder. Manche Raspberry-Pi-Kameras lassen sich direkt mit dem Jetson verbinden. Der Autor verwendet in den folgenden Schritten eine Kamera für den Raspberry Pi 2, die über ein für den Raspberry Pi vorgesehenes FPC-Kabel mit dem Port CAM0 Kontakt aufnimmt.

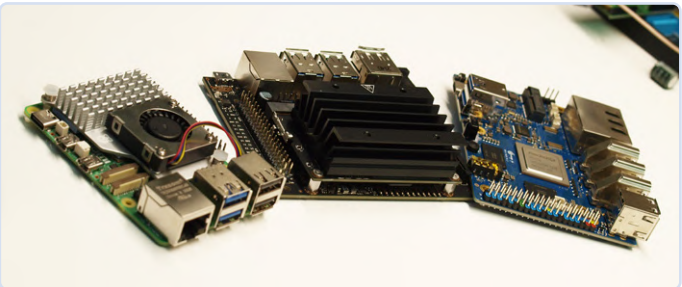


Bild 2. Der Jetson, neben einem Raspberry Pi 5 und einem OrangePi 5 Plus.

Bild 3. Wer diese Schraube und ihr Pendant löst, kann sein Entwicklerkit auseinandernehmen.

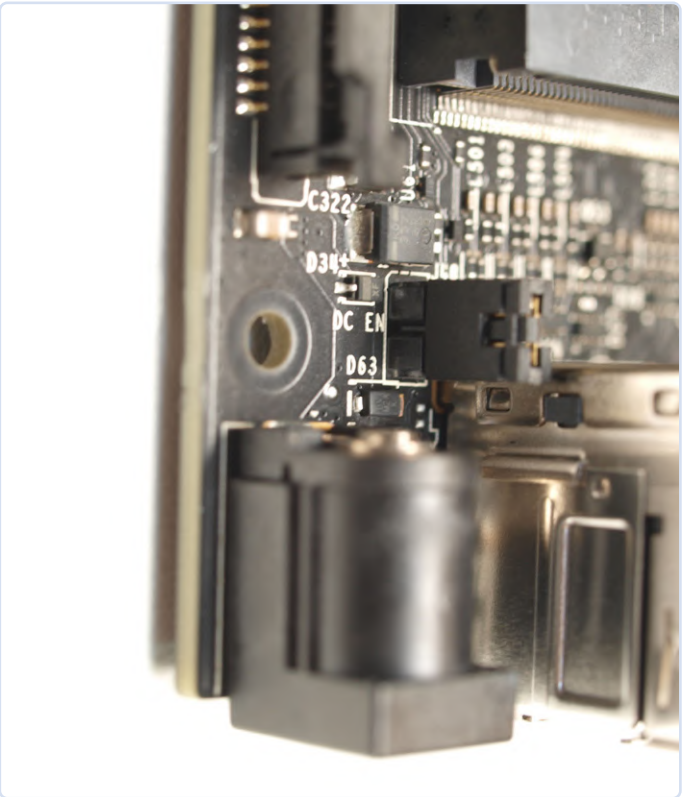


Bild 4. Dieser Jumper ist von eminenter Bedeutung.

Zu beachten ist lediglich, dass die für den Raspberry Pi 5 vorgesehene Variante des Kabels mit dem Jetson nicht kompatibel ist. Für die Positionierung der Kamera stehen in ThingiVerse verschiedene 3D-Modelle zur Verfügung, deren Ausdrucken das Leben des Entwicklers verbessert.
Wer den NVIDIA Jetson unter Nutzung des soeben erwähnten MeanWell-Netzgeräts in Betrieb nehmen möchte, muss eine kleine Anfängerfalle beachten. Der in **Bild 4** gesteckt gezeigte Jumper ist im Auslieferungszustand nicht gesteckt, so dass der DC-Anschluss nicht mit der Energieversorgung verbunden ist. Beim Einstecken des Netzteils leuchtet die Status-LED nicht auf, was verwirrend sein kann.
Im Rahmen des ersten Starts gilt, dass der Prozessrechner den HDMI-Bildschirm sofort einschaltet. Die Rekonfiguration der MicroSD-Karte und einige andere „Einführungsrituale" nehmen dann etwas Zeit in Anspruch. Nach getaner Arbeit präsentiert das System einen Ubuntu-Desktop und fordert danach zur Abarbeitung des Welcome-Assistenten auf.
Im Allgemeinen handelt es sich dabei um den gewöhnlichen Ubuntu-Set-up-Assistenten, der von NVIDIA aber unter anderem um einen Schritt zum Abnicken der hauseigenen Softwarelizenzen erweitert wurde. Wichtig ist der Assistent zur Auswahl des Energiemodells: Hier ist es empfehlenswert, die ausgewählte Option beizubehalten. Sie sorgt dafür, dass der Jetson die maximale Menge an Energie entgegennimmt.
Nach der erfolgreichen Abarbeitung des Assistenten gilt jedenfalls, dass noch etwas Zeit ins Land geht, bevor das Jetson-Board gefechtsbereit ist, denn auch am Desktop erscheinen mitunter noch einige Aufforderungen zum Rebooten des Systems.

Erste Experimente

NVIDIA setzt im Jetson auf ein mehr oder weniger komplett standardkonformes Ubuntu 18.04. Der Autor konfiguriert derartige Prozessrechner gerne für den Remote-Zugriff, um sich das Wechseln zwischen der PC- und der Prozessrechnertastatur zu ersparen.
Die in Ubuntu enthaltene Settings-Applikation ist für diese Aufgabe allerdings ungeeignet, weshalb wir stattdessen im ersten Schritt `sudo apt-get update` und `sudo apt-get upgrade` zur Aktualisierung des Paketbestands ausführen. Eventuelle Abfragen sind durch Drücken von *Enter* zu quittieren; ein Begehr im Bezug auf den Neustart des Docker-Systems ist zu erlauben. Danach ist ein Neustart des Jetson erforderlich. Die Eingabe von `sudo apt install vino` stellt sicher, dass der VNC-Server gefechtsbereit ist.
Zu guter Letzt sind dann noch die folgenden Befehle erforderlich, um die Konfiguration in einen brauchbaren Zustand zu bringen. Der String `thepassword` ist hierbei natürlich durch ein für Ihre Installation geeignetes Passwort zu ersetzen:

```
tamhan@tamhan-desktop:~$ mkdir -p ~/.config/autostart
tamhan@tamhan-desktop:~$ cp /usr/share/applications/
  vino-server.desktop ~/.config/autostart
tamhan@tamhan-desktop:~$ gsettings set org.gnome.Vino
  prompt-enabled false
tamhan@tamhan-desktop:~$ gsettings set org.gnome.Vino
  require-encryption false
tamhan@tamhan-desktop:~$ gsettings set org.gnome.Vino
  authentication-methods "['vnc']"
tamhan@tamhan-desktop:~$ gsettings set org.gnome.Vino
  vnc-password $(echo -n 'thepassword'|base64)
```

Nach dem nächsten Reboot lässt sich der Jetson mit dem VNC-Client Reminna ansprechen, wenn ein Benutzer „eingeloggt" ist. Zu beachten ist dabei allerdings, dass das Applet für den Remote-Zugriff in der Settings-Applikation nach wie vor nicht funktioniert.
Im nächsten Schritt bietet sich ein erster Test der Kameraverbindung durch Eingabe des folgenden Befehls an:

```
tamhan@tamhan-desktop:~$ gst-launch-1.0 nvarguscamerasrc
  ! nvoverlaysink
```

Die mit diesem primitiven Befehl aktivierbare Kamera-Preview erscheint ausschließlich auf einem physikalisch mit dem Jetson verbundenen Monitor - die per VNC mit dem Rechner kommunizierende Anlage sieht stattdessen die Terminalausgabe. Zur Beendigung des Spuks reicht es aus, mittels *Control* + *C* ein Interrupt-Event zu schicken.
Der eigentliche Kamerazugriff erfolgt dann allgemein durch die von PC beziehungsweise Workstation bekannten Methoden. Besonders interessant ist die Datei [5], die die Einrichtung einer Open-CV-basierten Pipeline demonstriert.
Zu ihrer Ausführbarmachung sind die folgenden Befehle erforderlich:

```
tamhan@tamhan-desktop:~$ git clone https://github.com/
  JetsonHacksNano/CSI-Camera
tamhan@tamhan-desktop:~$ cd CSI-Camera/
tamhan@tamhan-desktop:~/CSI-Camera$ python3 simple_
  camera.py
```

Das gestartete Kamera-Fenster ist dann übrigens auch über VNC sichtbar, weil es die Informationen nicht direkt in den Framebuffer der Tegra-GPU wirft.
Sollte das Kamerabild auf dem Kopf stehen, dann sorgt das Adjustieren des Parameters `flip_method` in der Datei für Abhilfe:

```
def show_camera():
    window_title = "CSI Camera"
    print(gstreamer_pipeline(flip_method=2))
     video_capture = cv2.VideoCapture(gstreamer_
            pipeline(flip_method=2), cv2.CAP_GSTREAMER)
    if video_capture.isOpened():
    . . .
```

Interaktion mit GPIO-Pins (via Python)

Systeme der künstlichen Intelligenz verlangen vom Diensthabenden einen komplett anderen Skillset, der mit klassischer Embedded-Entwicklung nicht wirklich verträglich ist.
Die praktische Erfahrung lehrt, dass fachfremde Personen mit Hintergrund im Bereich der klassischen Großrechnertechnik oft besser mit KI zurechtkommen als Embedded-Entwickler.

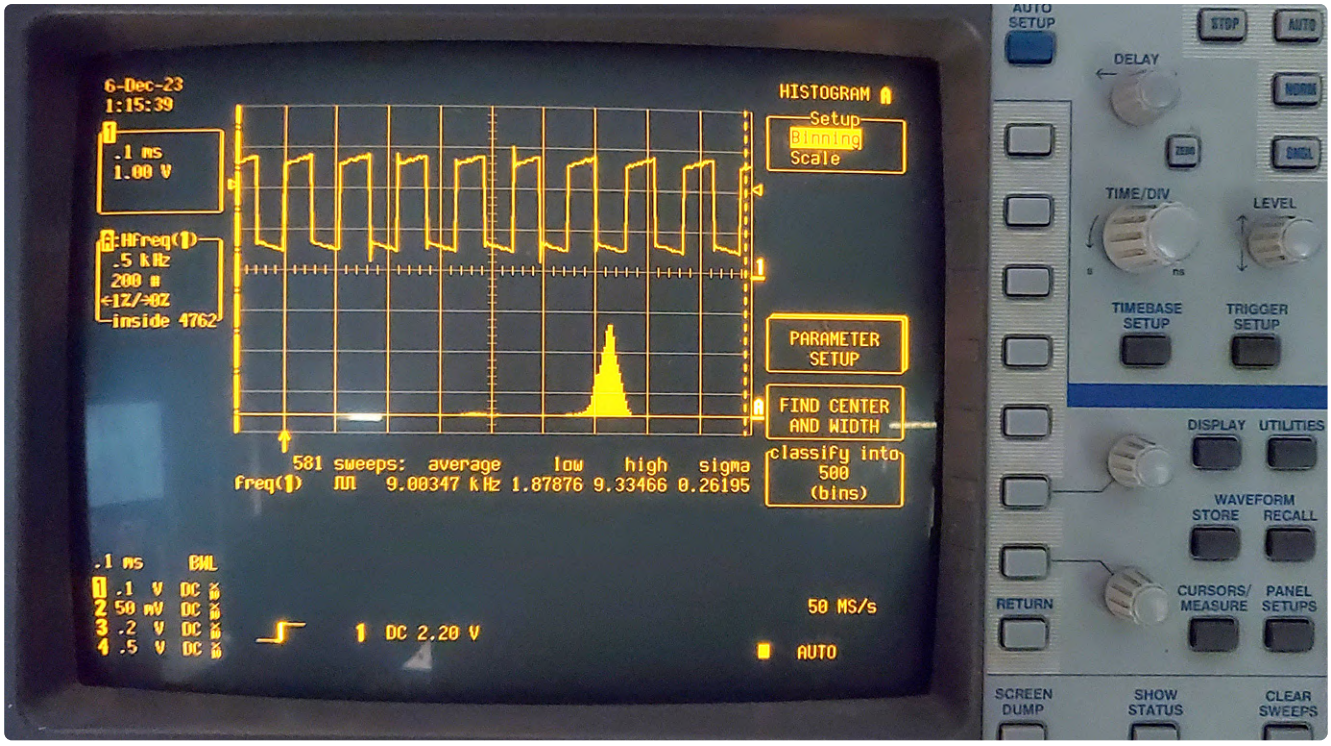


Bild 5. Der NVIDIA Jetson produziert unter Python Rechtecksignale.

Die in Java sehr effizient programmierende Lebensgefährtin des Autors löst KI-Aufgaben schneller – von Assembler hat sie indes nur wenig Ahnung.
Sinn dieses softwarearchitekturalen Exkurses ist die Feststellung, dass wir den GPIO-Zugriff auf dem Jetson in den folgenden Schritten absichtlich unter Python illustrieren. Ursache dafür ist, dass der Gutteil der anwenderorientierten Erzeugung von KI-Systemen unter Python erfolgt: Wer hier auf C setzt, handelt sich ein unnötiges natives Interface ein.
Die Standard-Distribution von Python bringt am Jetson keine Paketverwaltung mit, weshalb die Eingabe der Befehle `sudo apt install python-pip` und `sudo apt install python3-pip` erforderlich ist. Im nächsten Schritt bietet sich das Überprüfen der korrekten Installation der GPIO-Module an. Auch hier sind zwei Befehle erforderlich, weil die Python-Umgebungen (logischerweise) nicht wirklich zum Teilen ihres Bibliotheksschatzes befähigt sind:

```
tamhan@tamhan-desktop:~$ sudo pip install Jetson.GPIO
tamhan@tamhan-desktop:~$ sudo pip3 install Jetson.GPIO
```

Schon aus Platzgründen können wir die GPIO-API des Jetson hier nicht vollständig vorstellen. Unter [6] findet sich eine Gruppe vorgefertigter Beispiele, die die API vollumfänglich erklären.
Für unseren kleinen Test reicht dann folgendes Programm aus:

```
import RPi.GPIO as GPIO
import time

output_pin = 18  # BCM pin 18, BOARD pin 12

def main():
    GPIO.setmode(GPIO.BCM)
# BCM pin-numbering scheme from Raspberry Pi
    GPIO.setup(output_pin, GPIO.OUT, initial=GPIO.HIGH)
    print("Starting demo now! Press CTRL+C to exit")
    try:
        while True:
            GPIO.output(output_pin,  GPIO.HIGH)
            GPIO.output(output_pin,  GPIO.LOW)
            GPIO.output(output_pin,  GPIO.HIGH)
            GPIO.output(output_pin,  GPIO.LOW)

    finally:
        GPIO.cleanup()

if __name__ == '__main__':
    main()
```

Interessant ist hier vor allem, dass NVIDIA dem angehenden Jetson-Entwickler „mehrere“ Wege zur Ansprache der Pins anbietet. Wir nutzen hier die Option `GPIO.BCM`, die sich am Raspberry-Pi-Pinout orientiert. Lohn der Mühen ist dann das in **Bild 5** gezeigte Schirmbild. Die Frequenzstabilität mag nicht besonders hoch sein, reicht aber zum Auslösen von IoT-Ereignissen mehr als aus.
Angemerkt sei außerdem, dass die Ausführung von GPIO-Programmbeispielen ohne Superuserrechte mitunter Probleme verursacht - unter [7] findet sich eine detaillierte Besprechung der Thematik.

Experimente mit der ML-Funktion

Die Verfügbarkeit eines vollwertigen Linux-Betriebssystems und eines (sehr schnellen) USB3-Interfaces animiert eigentlich zur Ausführung von Experimenten. Ein mögliches Beispiel wäre Stable Diffusion: In der Praxis ist es unter Nutzung eines (adaptierten) Runners möglich, den Jetson als Bildgenerator einzuspannen.
In der Praxis ist diese Vorgehensweise allerdings nicht empfehlenswert: Sowohl der mit nur 4 GB VRAM vergleichsweise kleine Grafik-Speicher als auch die kleine Zahl von „nur” 128 Kernen sorgt dafür, dass die Bild-Generierung Geduld voraussetzt.

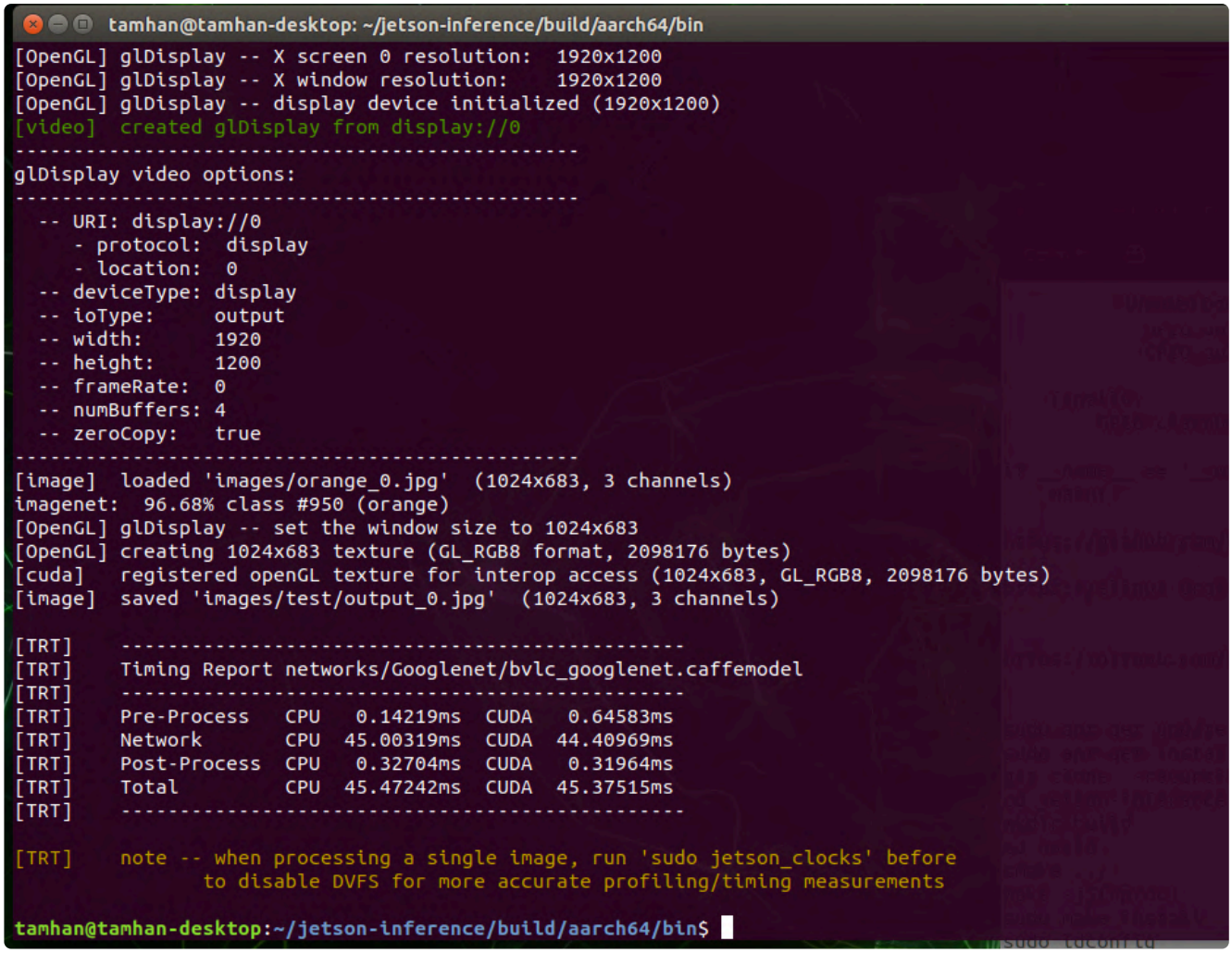


Bild 6. Das mit dem Jetson ausgelieferte Demoprogramm zeigt sich auf Kommandozeilen-Ebene nur wenig kommunikativ.

Auf Reddit finden sich Berichte von ML-Experimentatoren, die bei einer Clustergröße von 512 × 512 Pixel von bis zu fünf Stunden Rechenzeit pro Bild ausgehen.

Starthilfe für populäre Frameworks

NVIDIA versucht, Entwicklern die Inbetriebnahme verschiedener weitverbreiteter ML-Frameworks so weit wie möglich zu erleichtern. Unter [8] findet sich eine Liste der unterstützten Produkte samt optimierten Installationsanweisungen.

Ähnliches gilt auch für die gerne demonstrierten „End-to-end"-Lösungen: Das Trainieren eines Modells setzt so viel Rechenleistung und Ressourcen voraus, dass NVIDIA in den meisten Tutorials die Auslagerung auf einen Desktop oder einen Großrechner empfiehlt und der Jetson dann mit den schlüsselfertigen Modellgewichten parametriert wird.
Unter [9] finden sich mehr oder weniger schlüsselfertige Beispiele, die die Leistungsfähigkeit des Jetson auf eine unbürokratische Art und Weise illustrieren.
Zur Nutzung dieser Modell-Powershow ist es erforderlich, ein NVIDIA-Softwarepaket herunterzuladen und zu deployen. In der Theorie bietet sich hier auch die Nutzung eines Docker-Containern, für mit der Container-Technik fremdelnde Personen aber alternativ das lokale Kompilieren an, was durch Eingabe der folgenden Befehle bewerkstelligt werden kann:

```
sudo apt-get update
sudo apt-get install git cmake libpython3-dev
  python3-numpy
git clone --recursive --depth=1 https://github.com/
  dusty-nv/jetson-inference
cd jetson-inference
mkdir build
cd build
cmake ../
make -j$(nproc)
sudo make install
sudo ldconfig
```

Die Frage nach der Installation der Trainingskomponenten können Sie nach Belieben beantworten. Der Autor hat sie in den folgenden Schritten verneint, um auf seiner nur 32 GB großen Micro SD-Karte etwas Speicher einzusparen.
Nach dem erfolgreichen Durchlaufen des Kompilations-Prozesses findet sich im Verzeichnis `~/jetson-inference/build/aarch64$/bin` eine vergleichweise komplexe Projektstruktur, die neben diversen Binärdateien Python-Files bereitstellt. Interessant ist, dass NVIDIA hier sogar einige fertige Testbeispiele platziert.

Als Erstes wollen wir den Klassifikator verwenden - er analysiert Bilddateien und stellt fest, was auf dem angelieferten Bild zu sehen ist:

```
tamhan@tamhan-desktop:~/jetson-inference/build/aarch64/
  bin$ ./imagenet.py images/orange_0.jpg images/test/
  output_0.jpg
```

Die erstmalige Ausführung dieses Befehls nimmt dabei etwas mehr Zeit in Anspruch, weil die bereitgestellten Module für die Bedürfnisse des Jetson-Systems optimiert werden. Danach erfolgt die Ausgabe der **Bild 6** gezeigten Timinginformationen.
Zur eigentlichen Sichtbarmachung der vom ML-Prozess angelieferten Ergebnisse müssen Sie stattdessen die generierte Bilddatei ansehen - die Eingabe des folgenden Befehls öffnet den Ausgabe-Ordner direkt im Nautilus-Dateimanager:

```
tamhan@tamhan-desktop:~/jetson-inference/build/aarch64/
  bin$ nautilus images/test/output_0.jpg
```

Lohn der Mühen ist das Erscheinen des in **Bild 7** gezeigten Schirmbilds.

Analyse des Python-Files

Als nächstes Element wollen wir einen kurzen Blick auf das soeben verwendete Beispielprogramm werfen. Sein Quellcode lässt sich im Gedit-Editor durch Eingabe des folgenden Befehls in der Kommandozeile öffnen:

```
tamhan@tamhan-desktop:~/jetson-inference/build/aarch64/
  bin$ gedit imagenet.py
```

Schon auf den ersten Blick fällt auf, dass die hier verwendete Bibliothek mit dem klassischen *ImageNet* verwandt ist - im Jetson-Starterkit verpackt NVIDIA verschiedene weit verbreitete Systeme der künstlichen Intelligenz.
Der Kern des Programmbeispiels ist dabei eine Endlosschleife, die den Eingangs-Strom um ein Bild erleichtert, dieses dem ML-Modell

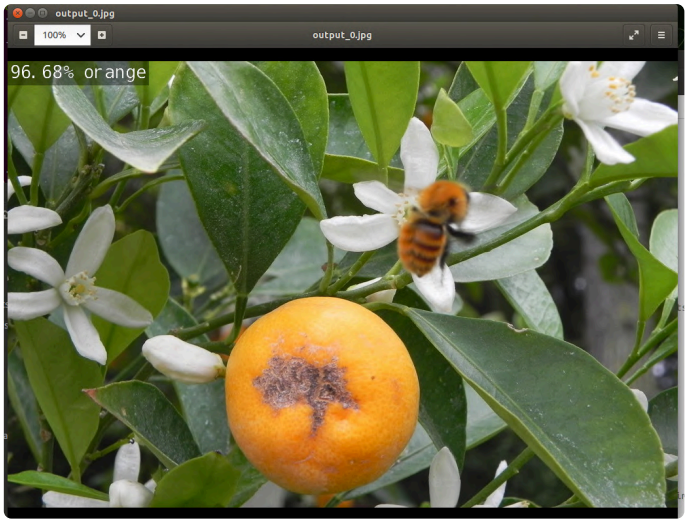


Bild 7. Die Obst-Erkennung gelang dem NVIDIA-Beispielprogramm vorbildlich.

vorwirft und zu guter Letzt die „errechneten" Informationen ausgibt (**Listing 1**).
Die Initialisierung der Datenströme und des zu verwendenden Modells erfolgt weiter oben nach folgendem Schema:

```
net = imageNet(args.network, sys.argv)
input = videoSource(args.input, argv=sys.argv)
output = videoOutput(args.output, argv=sys.argv)
font = cudaFont()
```

Interessant ist dann noch die Beschaffung beziehungsweise Bevölkerung des Parameters `network`, der den Namen des zu verarbeitenden Modells anliefert. NVIDIA setzt hierbei auf die `ArgParser`-Klasse, die normalerweise auf die Verarbeitung von über die Kommandozeile angelieferten Parameter spezialisiert ist. Im Fall der hier vorliegenden Deklaration wird ein Default-Wert eingeschrieben, der normalerweise das GoogLeNet aktiviert und lädt:

```
parser.add_argument("--network",  type=str,
default="googlenet", help="pre-trained model to load
(see below for options)")
```

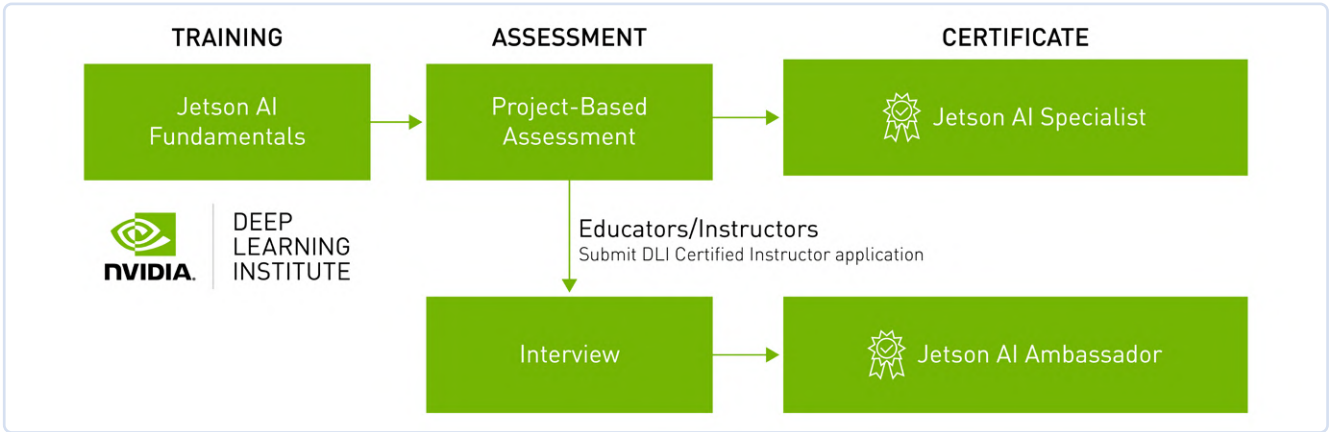


Bild 8. Zweigleisiger Bildungsweg à la NVIDIA! (Bildquelle: NVIDIA [10].)

Listing 1. Klassifikation.

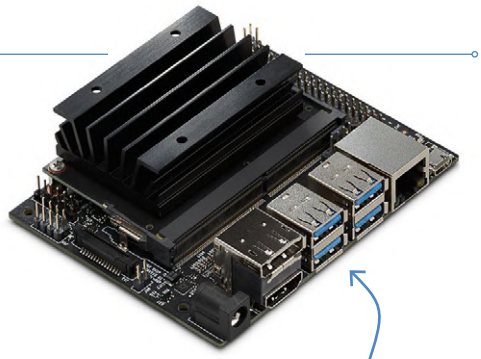
```
while True:
    img = input.Capture()

    if img is None: # timeout
        continue
        predictions = net.Classify(img, topK=args.topK)
        for n, (classID, confidence) in enumerate(predictions):
            classLabel = net.GetClassLabel(classID)
            confidence *= 100.0

            print(f"imagenet: % class # ()")

            font.OverlayText(img, text=f"% ",
                             x=5, y=5 + n * (font.GetSize() + 5),
                             color=font.White, background=font.Gray40)

    output.Render(img)
```



Passendes Produkt

› **NVIDIA Jetson Nano Developer Kit (B01)**
www.elektor.de/19001

Exkurs: „Geführte“ Online-Fortbildung samt Zertifikation

Die Erfolge der Sowjetunion in diversen arabischen und afrikanischen Staaten lassen sich unter anderem auf die enge Ausbildungspartnerschaft zurückführen - womit der Kadett lernt, das setzt er später gerne im Beruf ein. NVIDIA ist sich dieser Situation offensichtlich bewusst, weshalb mit Jetson AI Certification ein zweigeteilter und komplett kostenloser ML-Kurs wie in der **Bild 8** zu sehen zur Verfügung steht. Hervorzuheben ist dabei außerdem noch, dass das erfolgreiche Absolvieren von NVIDIA sogar mit einem Zertifikat honoriert wird.
Sollten Sie daran Interesse haben, empfiehlt sich der Besuch von [10]. Dort finden Sie weitere Informationen dazu, wie man die Teilnahme am NVIDIA-Fortbildungskurs am effizientesten gestaltet.

Vorteile durch Linux

Mit dem Jetson schickt NVIDIA ein Zwitterwesen ins Rennen, das sich im Bereich der künstlichen Intelligenz zwischen alle Stühle setzt. Einerseits gilt, dass dedizierte Low-Power-Mikrocontroller wie der Maxim MAX78000 einen wesentlich geringeren Energieverbrauch bieten. Andererseits leiden solche Controller darunter, dass sie keine Unterstützung für CUDA bieten: Ein auf dem PC oder im Großrechner-Bereich lauffähiges Modell benötigt deshalb Adaption, bevor es sich unter Nutzung dieser Chips im IoT verwenden lässt. Andererseits ist der NVIDIA Jetson keine vollwertige GPU: Das Modul ist sowohl vom Energieverbrauch als auch von der unterstützten CUDA-Variante (CUDA 11 läuft nicht) kein vollwertiger Ersatz für eine RTX 4000.
Unterm Strich gilt, dass sich ein Deployment auszahlt, wenn ein am Rechner oder Großrechner problemlos funktionierendes Modell aufwandsarm mobilisiert werden muss und es mit den vom Jetson angebotenen Ressourcen auskommt. Wegen des Linux-Betriebssystems gilt, dass dies für einen Data Scientist vergleichsweise wenig Aufwand bei der Umstellung bedeutet - die Einarbeitung in die von Maxim und Co. verwendeten Embedded-APIs erfordert erheblich mehr Mannstunden. Die höheren Kosten der Hardware lassen sich so, insbesondere in kleineren Serien, schnell und unbürokratisch amortisieren.

230740-02

Sie haben Fragen oder Kommentare?

Gerne können Sie sich an den Autor unter der E-Mail-Adresse tamhan@tamoggemon.com oder an die Elektor-Redaktion unter der E-Mail-Adresse redaktion@elektor.de wenden.

WEBLINKS

[1] Portfolio-Übersicht: https://www.nvidia.com/en-eu/autonomous-machines/embedded-systems/
[2] Liste verschiedenster Drittanbieter-Produkte: https://developer.nvidia.com/embedded/ecosystem
[3] OEMSecrets-Bestpreis: https://www.oemsecrets.com/compare/%20945-13450-0000-100
[4] Imagedatei herunterladen: https://developer.nvidia.com/jetson-nano-sd-card-image
[5] Open CV-basierte Pipeline-Datei: https://github.com/JetsonHacksNano/CSI-Camera/blob/master/simple_camera.py
[6] Vorgefertigte Beispiele: https://github.com/NVIDIA/jetson-gpio/tree/master/samples
[7] Die Ausführung von GPIO-Programmbeispielen ohne Superuserrechte: https://github.com/NVIDIA/jetson-gpio/issues/20
[8] Liste der unterstützten Produkte: https://elinux.org/Jetson_Zoo
[9] Schlüsselfertige Beispiele: https://github.com/dusty-nv/jetson-inference
[10] Jetson KI-Kurse und Zertifikate: https://developer.nvidia.com/embedded/learn/jetson-ai-certification-programs

embeddedworld

Exhibition&Conference

UNITED

CONNECTING THE
EMBEDDED COMMUNITY

9. – 11.4.2024

Get your
free ticket now!

embedded-world.de/code

Use the voucher code **ew24ELE**

Medienpartner

Markt&Technik
Elektronik
computer & automation
Elektronik automotive
Elektronik medical
elektroniknet.de

NÜRNBERG MESSE

2024 Eine Odyssee in die KI

Erste Gehversuche mit TensorFlow

Von Brian Tristam Williams (Elektor)

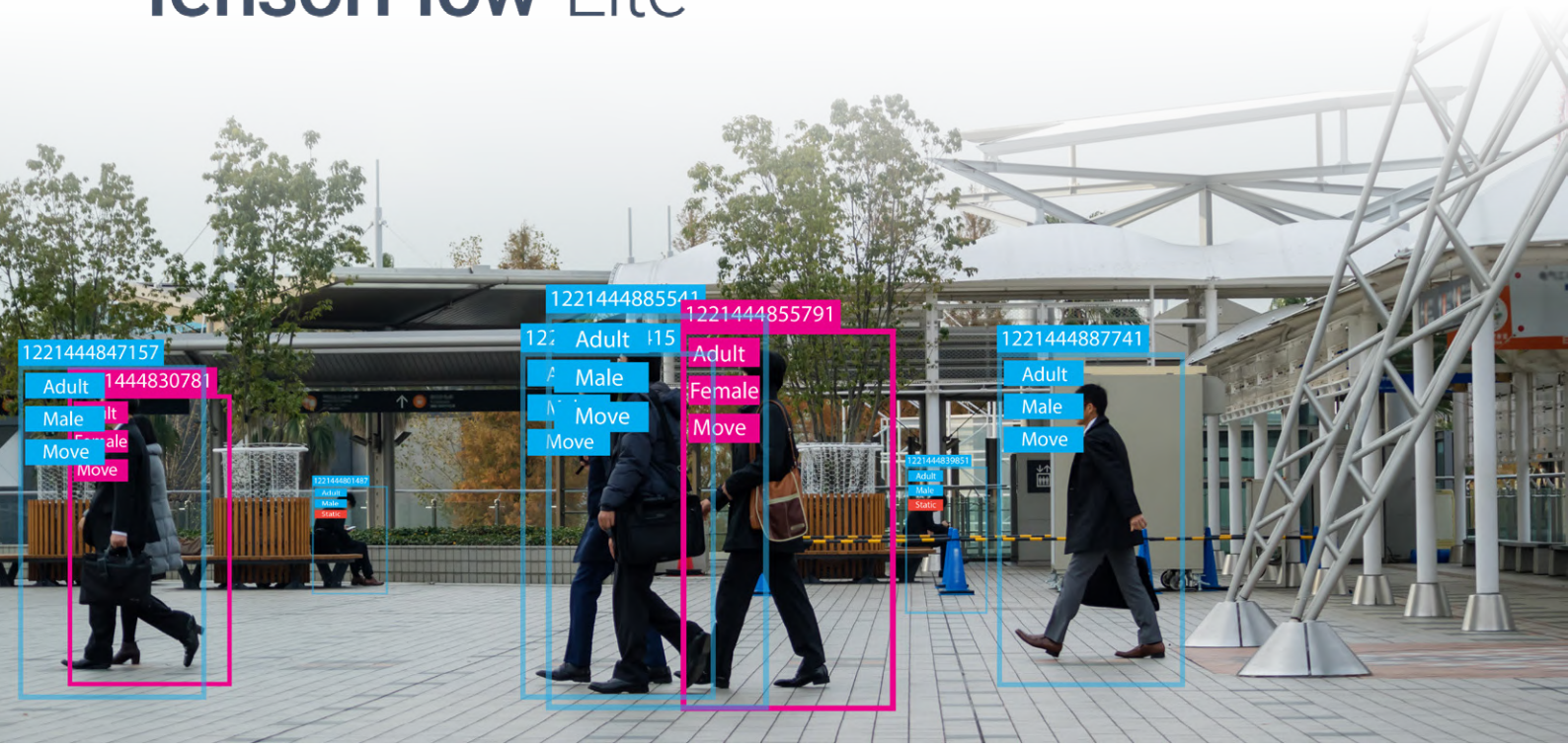
Diese Folge erkundet das Potenzial von TensorFlow Lite auf dem Raspberry Pi und wirft einen kompakten Blick auf die praktischen Möglichkeiten von KI, indem sie die Grenzen dessen testet, was auf Geräten mit geringerer Stromaufnahme möglich ist.

TensorFlow Lite

Was ist TensorFlow?

TensorFlow wurde vom Google-Brain-Team entwickelt und ist eine Open-Source-Bibliothek für numerische Berechnungen und maschinelles Lernen. Die Bibliothek spielt eine entscheidende Rolle im Ökosystem der breiteren KI, das auch generative KI-Technologien wie ChatGPT umfasst. Sie ist ein Eckpfeiler des Deep Learning und ermöglicht die Erstellung von Modellen, die „Big Data" verarbeiten und daraus lernen können.

TensorFlow und generative KI-Modelle ergänzen sich oft gegenseitig, wobei TensorFlow die grundlegende Plattform für den Aufbau und das Training einer Vielzahl von KI-Modellen darstellt, einschließlich solcher, die generative Anwendungen antreiben können. Die Vielseitigkeit erlaubt es, eine breite Palette von Aufgaben von einfachen Klassifizierungen bis hin zu komplexen Prozessen der Entscheidungsfindung zu bewältigen, was TensorFlow zu einer beliebten Wahl sowohl im akademischen als auch im industriellen Bereich macht. Die Fähigkeit, große Datenmengen zu verarbeiten, Muster zu erkennen und daraus zu lernen, macht TensorFlow zu einem leistungsstarken Werkzeug im



breiteren Kontext der generativen KI, die sich auf die Erstellung neuer Inhalte und Datenmodelle konzentriert.

TensorFlow Lite: Weniger ist mehr

In diesem nächsten Teil unserer KI-Reise wähle ich einen etwas unkonventionellen Weg. Obwohl ich Zugang zu einem einigermaßen fähigen PC habe, der mit einer anständigen RTX-4070-GPU und genügend RAM ausgestattet ist, fühle ich mich zur Welt von TensorFlow Lite [1] auf dem Raspberry Pi hingezogen. Es liegt nämlich ein gewisser Reiz in der Herausforderung, KI-Prozesse für ein so kompaktes, weniger leistungsfähiges Gerät zu optimieren. Aber es geht nicht nur darum, mit weniger mehr zu erreichen, sondern auch darum, intelligente, in sich geschlossene Lösungen zu schaffen, die portabel und effizient sind.
Manchmal benötigt ein Projekt nicht die Größe und Leistung eines großen Desktop-Towers oder man hat keinen Platz für ein solches Mainframe. In solchen Fällen bietet sich der Raspberry Pi als perfekte Alternative an. Die Ausführung von TensorFlow Lite auf diesem winzigen, aber fähigen Gerät macht KI zugänglicher und anpassungsfähiger für verschiedene Umgebungen, sei es für Bildungszwecke, Hobbyprojekte oder reale Anwendungen, bei denen Platz und Stromaufnahme eine Rolle spielen.
Mit dem Aufkommen des Edge Computings ist der Bedarf an leistungsstarken und effizienten KI-Tools noch deutlicher geworden. Hier kommt TensorFlow Lite ins Spiel, eine leichtere und effizientere Version von TensorFlow, insbesondere für Geräte wie den Raspberry Pi. Der Raspberry Pi ist ein kleiner, erschwinglicher und dennoch leistungsfähiger Computer, der die perfekte Plattform für die Ausführung von TensorFlow Lite darstellt und die Fähigkeiten des maschinellen Lernens in die Hände von Makern, Hobbybastlern, Pädagogen und Profis gleichermaßen legt.
Die Implementierung von TensorFlow Lite auf einem kleinen Einplatinencomputer bietet erhebliche Vorteile gegenüber KI-Anwendungen auf größeren Rechnern. TensorFlow Lite ist für die Leistung auf leichtgewichtiger Hardware optimiert und eignet sich daher für eine breite Palette praktischer Anwendungen über die Objekterkennung hinaus. Dazu gehören Echtzeit-Datenverarbeitung, lokale Entscheidungsfindung in IoT-Geräten und die Ausführung von KI-Modellen für Aufgaben wie Gestenerkennung, Umweltsensorik und Gesundheitsüberwachung. Dies ermöglicht es Raspberry-Pi-Herstellern, komplexe KI-Modelle auf kostengünstige und leicht zugängliche Weise einzusetzen, wodurch sich eine Welt der Möglichkeiten für Innovation und Erforschung von KI eröffnet, selbst bei kleineren Projekten.
In den folgenden Abschnitten sehen wir uns an, wie man TensorFlow Lite auf einem Raspberry Pi installiert und sich auf zukünftige innovative Anwendungen vorbereitet, die es ermöglicht.

Startblöcke

Ich habe eine Schublade voller Raspberry Pis, vom 1 Mod. A. über den Zero bis zum aktuellen Modell 5, aber ich will für dieses Projekt einen Raspberry Pi 4 verwenden. Das ist ein guter Mittelweg, was die Zugänglichkeit für unsere Leser angeht, und liegt genau zwischen den 3ern, die ich manchmal noch im Angebot sehe, und dem neuen Kid on the Block, dem 5er. Außerdem habe ich nur den einen Raspberry Pi 5, den ich oft anderweitig gebrauche. Also, was jetzt auf meinem Arbeitstisch liegt, ist:

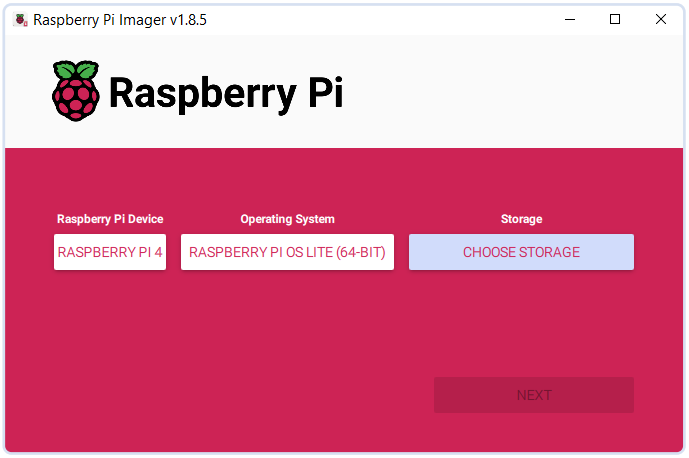


Bild 1. Der Raspberry-Pi-Imager hat eine übersichtliche, leicht zu bedienende Oberfläche.

- **Raspberry Pi 4 Model B**: Wir werden sehen, wie er mit seinen 4 GB RAM zurechtkommt. Ich habe sicherheitshalber mit `free -h` in der Kommandozeile den Speicherplatz ermittelt.
- **32 GB große microSD-Karte**: Ich weiß nicht, ob ich unbedingt so viel brauche, aber die Karte war die erste, die mir in die Finger kam, als ich auf der Suche nach einer Karte war.
- **Raspberry Pi OS Lite (64-Bit)**: Ich bin nicht wirklich ein Fan des Overheads, den GUIs mit sich bringen, und ich bevorzuge die Ja/Nein-Ein/Aus-Bestätigung von der Textschnittstelle, also habe ich mich für die Portierung von Debian Bookworm ohne Desktop-Umgebung entschieden. Da ich lange Zeit mit Webservern gearbeitet habe, habe ich zwar nie wirklich eine GUI in Linux gebraucht, aber lassen Sie mich wissen, ob ich mit der Zeit gehen sollte!

Ich habe *Raspberry Pi Imager v1.8.5* installiert, die zum Zeitpunkt der Erstellung dieses Artikels aktuelle Version von [2]. Laden Sie einfach die Version für Ihr Betriebssystem herunter und führen Sie sie aus. Das Programm vereinfacht den Installationsprozess wirklich hervorragend. Wählen Sie einfach die Version Ihres Raspberry Pi-Boards, das gewünschte Betriebssystem und die Karte, die Sie in den Kartenleser Ihres Computers gesteckt haben. Die Benutzeroberfläche ist wirklich sauber und einfach (**Bild 1**), und noch besser: Sie funktioniert.
Ich habe meine Versionsnummer nach der Installation mit `cat /etc/os-release` überprüft. Bei Ihnen kann das anders sein, aber das war mein Ergebnis:

```
briantw@raspberrypi:~ $ cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 12 (bookworm)"
NAME="Debian GNU/Linux"
VERSION_ID="12"
VERSION="12 (bookworm)"
VERSION_CODENAME=bookworm
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
```

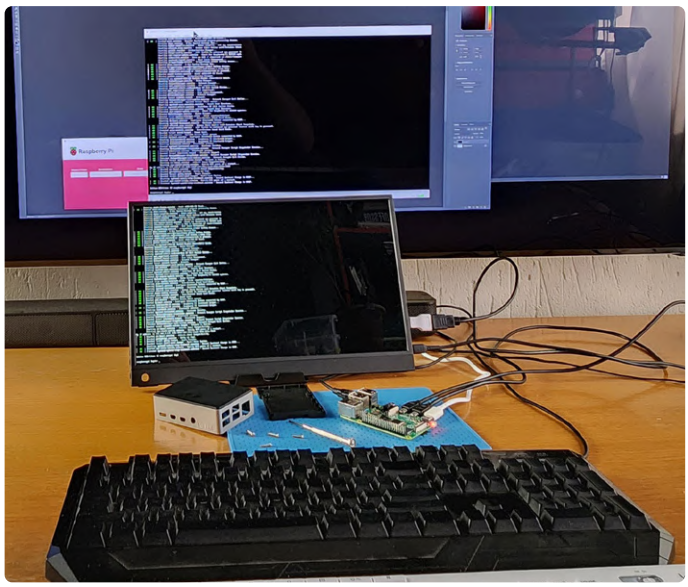


Bild 2. Die grundlegende Einrichtung des Raspberry Pi, den ich für diese Untersuchung verwende.

Was das Zubehör betrifft, so habe ich lediglich eine Tastatur (keine Maus erforderlich - juhu!), ein Raspberry-Pi-4-Netzteil und beide Micro-HDMI-Ausgänge angeschlossen (**Bild 2**). Zurzeit geht eine HDMI-Verbindung zu einem großartigen kleinen tragbaren Monitor und der andere an ein Aufnahmegerät an meinem PC, das ich im Moment für einige Screenshots verwende. Die Idee ist, meinen Code auf einem Bildschirm laufen zu lassen und die Videoausgabe - zum Beispiel von einem Objekterkennungsskript - auf dem anderen zu sehen.

Installation von TensorFlow Lite

Nun, da wir unseren Ausgangspunkt haben, wollen wir TensorFlow Lite auf dem Raspberry Pi zum Laufen bringen. Erstens, etwas Hausarbeit. Führen wir einen Update/Upgrade-Zyklus auf dem Raspberry Pi durch, um sicherzustellen, dass wir die neuesten Versionen von allem, was installiert ist, haben. Beginnen Sie mit:

```
sudo apt-get update
```

Wie lange Sie warten müssen, hängt davon ab, wie weit Sie mit den letzten Aktualisierungen zurückliegen. Bei mir wurden nur zwölf Elemente aktualisiert. Weiter:

```
sudo apt-get upgrade
```

Bei mir führte dies zu einem ganzen Bildschirm voller Aktualisierungen, was mich immer ein wenig überrascht, wenn ich gerade das neueste offizielle Betriebssystem installiert habe. Haben Sie Geduld mit dem Prozess – die Wartezeit kann variieren.
Was die Installation von TensorFlow Lite angeht, habe ich verschiedene Google-Suchergebnisse durchforstet, und die zugänglichste Anleitung, die ich gefunden habe, stammt von Evan von EdjeElectronics auf GitHub [3]. Natürlich sind seit Evans ursprünglicher Beschreibung des Prozesses einige Betriebssystemversionen und -upgrades ins Land gezogen, so dass ich auf ein paar kleinere Probleme gestoßen bin, die ich hier für Sie dokumentieren werde. Der erste Schritt war das Klonen des GitHub-Repositorys mit:

```
git clone https://github.com/EdjeElectronics/TensorFlow-
  Lite-Object-Detection-on-Android-and-Raspberry-Pi.git
```

Die entmutigende Antwort meines Pi:

```
-bash: git: command not found
```

Ach ja, es ist ja eine Neuinstallation. Installieren Sie also *git* mit:

```
sudo apt install git
```

Das hat bei mir funktioniert. Da wir nun *git* installiert haben, gehen wir zurück zum obigen Befehl `git clone...`. Wenn Sie erfolgreich waren, werden Sie ein Unterverzeichnis mit dem Namen *TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi* in Ihrem Home-Verzeichnis entdecken. Der Name ist etwas unhandlich, also benennen Sie es mit dem Befehl `mv` (move) in *tflite1* um:

```
mv TensorFlow-Lite-Object-Detection-on-Android-and-
  Raspberry-Pi tflite1
```

Gehen Sie dann in dieses Verzeichnis mit:

```
cd tflite1
```

EdjeElectronics schlägt vor, eine virtuelle Umgebung zu erstellen, also folgen wir diesem Rat:

```
sudo pip3 install virtualenv
```

Die Antwort des Pi ist fast vorhersehbar:

```
-bash: pip3: command not found
```

Die Installation von *pip3* ist fast genauso einfach gelöst wie die von *git*. Ich habe zuerst ein weiteres `sudo apt-get update` durchgeführt, bei dem neun Dinge aktualisiert wurden, und dann dies:

```
sudo apt-get install python3-pip
```

Dabei wurde eine Menge installiert, was zu einem weiteren Bildschirm voller Text führte.
Als nächstes habe ich den Befehl `sudo pip3...` erneut versucht, aber Debian mochte ihn nicht. Nach einigem Suchen fand ich heraus, dass ich meine eigene virtuelle Umgebung erstellen musste. Angenommen, Sie befinden sich noch im *tflite1*-Verzeichnis, dann verwenden Sie:

```
python3 -m venv tflite1-env
```

Aktivieren Sie dann diese virtuelle Umgebung:

```
source tflite1-env/bin/activate
```

Nun installieren Sie die Abhängigkeiten von TensorFlow Lite und OpenCV. Wie bereits erwähnt, hat TensorFlow Lite viele Anwendungsbereiche, aber diese Installation wird es für einen sehr populären Bereich von Anwendungen vorbereiten, nämlich Maschinelles Sehen und Objekterkennung. Daher ist die Installation der Open-Source-Bibliothek OpenCV, die für Computer Vision und maschinelles Lernen entwickelt wurde, sehr nützlich.
Glücklicherweise müssen Sie hier nichts mit `curl`, `wget` oder `git` klonen, denn der Ersteller des Repo hat ein nettes 40-zeiliges Shell-Skript namens *get_pi_requirements.sh* für uns erstellt, das alles automatisch erledigt. Wenn Sie neugierig sind, was das Skript macht, bevor Sie es starten, können Sie sich den Inhalt ansehen, indem Sie `cat get_pi_requirements.sh` ausführen, aber wir werden es direkt durch Eingabe von:

```
bash get_pi_requirements.sh
```

ausführen. Ich habe den Fehler gemacht, dies zu versuchen, nachdem ich den Raspberry Pi neu gestartet hatte, und es wurden einige Fehlermeldungen und Warnungen ausgegeben. Der Grund dafür? Der Quellcode-Befehl, den wir zuvor im tflite1-Verzeichnis ausgeführt haben, muss bei jedem Neustart oder bei einer neuen Terminal-Sitzung ausgeführt werden. Außerdem dauerte die Ausführung ein paar Minuten, also haben Sie noch einmal Geduld!
Jetzt haben wir die Möglichkeit, ein TensorFlow-Lite-Beispielmodell von Google zu verwenden oder eines, das wir selbst trainiert haben. Als Neuling habe ich mich für das von Google entschieden:

```
wget https://storage.googleapis.com/download.tensorflow.org/models/tflite/coco_ssd_mobilenet_v1_1.0_quant_2018_06_29.zip
```

Das ist eine lange Zeile, die ohne Zeilenumbruch nicht hierher passt, aber beachten Sie einfach, dass das einzige Leerzeichen in der Zeile das nach `wget` ist - ab `https` gibt es keine mehr.
Als nächstes entpacken Sie die heruntergeladene Datei:

```
unzip coco_ssd_mobilenet_v1_1.0_quant_2018_06_29.zip -d
  Sample_TFLite_model
```

Und das war's, wir haben alles installiert, was wir brauchen, um mit der Objekterkennung und Bildklassifizierung mit TensorFlow Lite auf dem Raspberry Pi zu spielen. Jetzt können wir anfangen zu experimentieren!

In unserer nächsten Folge: Als Datensammler, insbesondere von historischen Bildern und Videos, die oft nirgendwo anders verfügbar sind, möchte ich das, was ich zur Verfügung habe, mit der Welt teilen. Aber ich werde nie die Zeit haben, jedes einzelne Bild und jedes einzelne Video durchzugehen, um Metadaten hinzuzufügen und den Suchmaschinen mitzuteilen, was ich habe. Deshalb möchte ich diese Tools nutzen, um meinen Medienberg zu analysieren und zu klassifizieren. Diese Phase beginnt jetzt, und ich werde Sie beim nächsten Mal über meine Erfolge und Misserfolge informieren!

RG — 230181-E-02

Haben Sie Fragen oder Kommentare?

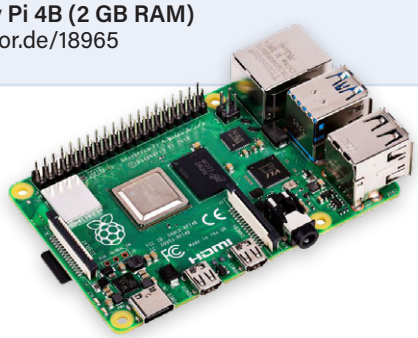
Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Schreiben Sie dem Autor eine E-Mail an brian.williams@elektor.com.

Über den Autor

Brian Tristam Williams ist von Computern und Elektronik fasziniert, seit er im Alter von zehn Jahren seinen ersten „Mikrocomputer" bekam. Seine Reise mit Elektor begann, als er mit 16 Jahren seine erste Ausgabe kaufte. Seitdem verfolgt er die Welt der Elektronik und Computer, erforscht und lernt ständig dazu. Seit 2010 arbeitet er bei Elektor und hält sich gerne über die neuesten Techniktrends auf dem Laufenden, insbesondere über künstliche Intelligenz und Einplatinencomputer wie den Raspberry Pi.

Passende Produkte

- **Raspberry Pi 4B (2 GB RAM)**
  www.elektor.de/18965



WEBLINKS

[1] Offizielle Webseite TensorFlow Lite: https://tensorflow.org/lite
[2] Download Raspberry Pi Imager: https://raspberrypi.com/software
[3] TensorFlow Lite, Objekterkennung auf Android und Raspberry Pi (GitHub): https://tinyurl.com/edjetflite

262.144 Wege, das Spiel des Lebens zu spielen

Ein Leserprojekt in Kürze

Von Brian White (USA)

Ich bin seit langem von Conways Spiel des Lebens fasziniert. Folgen Sie mir auf meinem Weg, dieses Programmierrätsel aus den 1970er Jahren in ein fesselndes visuelles Spektakel zu verwandeln. Dabei verwende ich eine RGB-LED-Matrix in Verbindung mit einzigartigen Codierungsmethoden, die es ermöglichen, jede denkbare Spielregel zu simulieren!

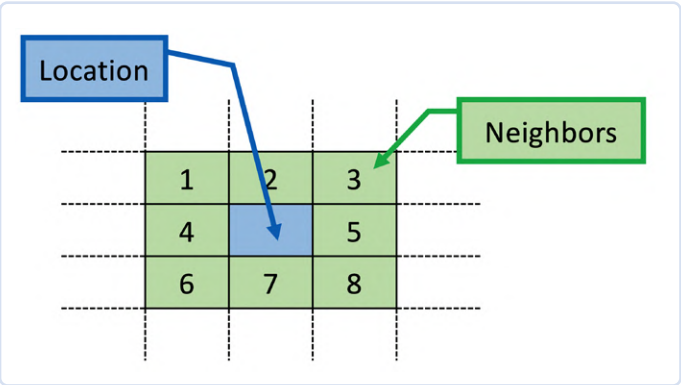


Bild 1. Ein Ort in der Welt und seine acht benachbarten Orte.

Dieses Projekt ist eine Geschichte in vielen Teilen, von denen einige schon seit 50 Jahren in meinem Kopf kreisen und die nun endlich zu einem sehr befriedigenden Ergebnis zusammenkommen. Es beginnt mit Conways Spiel des Lebens, von dem ich in den späten 1970er Jahren zum ersten Mal hörte. Damals belegte ich meinen ersten (und einzigen) Programmierkurs - BASIC-Programmierung an einem DECwriter-Terminal, das mit einem PDP-11 in meiner High School verbunden war - und das Schreiben von Code für „Das Spiel des Lebens“ war eine der Übungen. Es ist ein unterhaltsames Beispiel für verschachtelte Schleifen (`FOR...NEXT` in BASIC), eine großartige Übung für Programmieranfänger, und es ergab faszinierende, wechselnde Muster. Als sich die Computertechnik in den darauffolgenden Jahren von Druckerterminals über Röhrenmonitore zu Laptops und iPads weiter entwickelte, faszinierte mich Conways Spiel.

Das Spiel des Lebens

Kurz gesagt, Conways Spiel des Lebens [1] ist eine einfache regelbasierte Simulation zellulärer Automaten. Es wird auf einer rechteckigen Matrix von beliebiger Größe gespielt. Die Spielfiguren sind Zellen, die jeweils einen Platz in der Matrix einnehmen. Im Spiel können die Zellen von einer Generation zur nächsten fortbestehen, sterben oder neu geboren werden. Ob eine bestimmte Zelle in der nächsten Generation vorhanden ist oder nicht, hängt nur davon ab, wie viele Nachbarn dieser Ort in der aktuellen Generation hat. In einer rechteckigen Matrix hat jeder Ort zwischen null und acht Nachbarn (**Bild 1**).
Die Regeln von Conway lauten:

- Eine Zelle bleibt in der nächsten Generation bestehen, wenn sie zwei oder drei Nachbarn hat. Zellen mit weniger Nachbarn sterben an Einsamkeit, Zellen mit mehr Nachbarn sterben an Überbevölkerung.
- Eine Zelle kann in der nächsten Generation auf einem leeren Platz geboren werden, wenn sie genau drei Nachbarn hat

Conway wählte diese Regeln nach einer Reihe von Experimenten aus, um die folgenden in [2] beschriebenen Kriterien zu erfüllen:

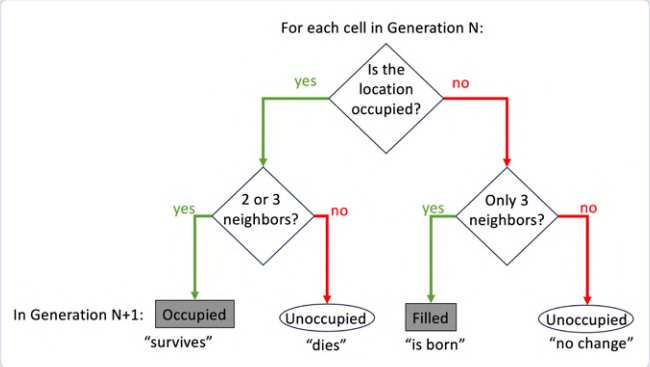
- Es sollte kein initiales Muster geben, für das es einen einfachen Beweis gibt, dass die Population unbegrenzt wachsen kann
- Es sollte initiale Muster geben, die offensichtlich unbegrenzt wachsen
- Es sollte einfache initiale Muster geben, die über einen beträchtlichen Zeitraum wachsen und sich verändern, bevor sie auf drei mögliche Arten enden: Sie verschwinden vollständig (weil sie überfüllt oder zu spärlich werden), sie

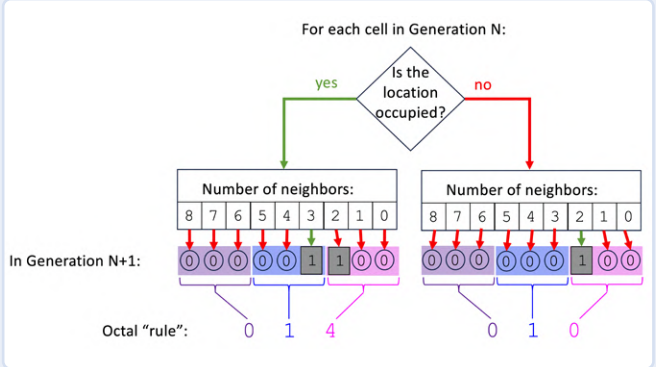
Kodierung der Regeln in sechs Oktalziffern

Die ursprünglichen Regeln des Spiels des Lebens können kurz so beschrieben werden:

- Eine Zelle überlebt, wenn sie zwei oder drei Nachbarn hat, ansonsten stirbt sie.
- Eine Zelle wird an einem leeren Ort mit genau drei Nachbarn geboren.

Dieselben Regeln können auch so dargestellt werden, wobei eine „1“ in der Regel einen besetzten Platz in der nächsten Generation (Überleben oder Geburt) für diese Anzahl von Nachbarn anzeigt und eine „0“ in der Regel einen unbesetzten Platz in der nächsten Generation (Tod oder Leerbleiben) anzeigt.





Das Bitmuster gibt den Zustand der Zelle in der nächsten Generation an; der Index des Bitmusters ist die Anzahl der Nachbarn in Verbindung mit der Angabe, ob die Zelle derzeit besetzt ist oder nicht. Im Folgenden wird die Dekodierung der Regel 256020 als Beispiel gezeigt:

Tabelle 1: Interpretation des oktalen Regelsatzes „256020“.

Ursprünglicher Standort	Belegt									Nicht belegt								
Nachbarn	8	7	6	5	4	3	2	1	0	8	7	6	5	4	3	2	1	0
Nächste Generation	0	1	0	1	0	1	1	1	0	0	0	0	0	1	0	0	0	0
Oktal	2			5			6			0			2			0		

Übersetzt in Textform würde dies bedeuten:

- Derzeit lebende Zellen überleben, wenn sie 1, 2, 3, 5 oder 7 Nachbarn haben; andernfalls sterben sie.
- Eine Zelle wird an jedem derzeit unbesetzten Ort geboren, der genau vier Nachbarn hat.

Mit diesem Kodierungsformat kann jeder der 262.244 möglichen Regelsätze für das Spiel des Lebens angegeben werden.

pendeln sich in einer stabilen Konfiguration ein, die danach unverändert bleibt, oder sie treten in eine oszillierende Phase ein, in der sie einen endlosen Zyklus von zwei oder mehr Perioden wiederholen.

In der ursprünglichen Version von 1970 wurde das Spiel mit physischen Spielsteinen auf einem Schachbrett gespielt. Bald darauf wurde das Verfahren auf Computer umgestellt, und die Innovationen halten bis zum heutigen Tag an [3].

Interessante Algorithmen

Das nächste Puzzleteil war das Buch „A New Kind of Science“ von Stephen Wolfram (ja genau, der Begründer von Mathematica), in dem er verschiedene Algorithmen für zelluläre Automaten beschreibt, die sehr interessante visuelle Muster erzeugen. Einer der wichtigsten Punkte von Wolframs Analyse bestand darin, die Regeln für die Übergänge von Generation zu Generation als Binärzahlen zu kodieren. Dadurch ist es möglich, die Konsequenzen aller möglichen Regelsätze auf eine wohldefinierte und wiederholbare Weise zu untersuchen [4].

Dies brachte mich dazu, über eine Möglichkeit nachzudenken, die Regeln des Spiel des Lebens so zu kodieren, dass es möglich ist, alle möglichen Regelsätze zu untersuchen. Mir wurde klar, dass man die Regeln in einer 18-Bit-Binärzahl kodieren könnte (siehe Details im obigen Kasten). Jedes der neun Bits niedriger Ordnung (0…8) gibt den Zustand des Ortes in der nächsten Generation (1 = besetzt; 0 = unbesetzt) für einen unbesetzten Ort mit null bis acht Nachbarn an. Bit 0 gibt den Zustand in der nächsten Generation an, wenn der Ort derzeit null Nachbarn hat, Bit 1 für einen Nachbarn und so weiter In ähnlicher Weise geben die neun Bits höherer Ordnung (9…17) den Zustand der nächsten Generation für einen belegten Ort mit null bis acht Nachbarn an. Bit 9 für null Nachbarn, Bit 10 für einen Nachbarn und so fort. Mit dieser Kodierung ist es möglich, beliebige Regeln festzulegen, die auf der Anzahl der Nachbarn eines Ortes basieren.

Interessanterweise müssen diese Regeln keinen „biologischen“

Sinn ergeben. Ein Beispiel dafür ist der Regelsatz 256020, bei dem Zellen überleben, wenn sie 1, 2, 3, 5 oder 7 Nachbarn haben. Dieser Regelsatz ergibt ein interessantes Verhalten, obwohl es keinen biologischen Grund gibt, warum 4, 6 und 8 Nachbarn überfüllt sein sollten, 3, 5 und 7 dagegen aber nicht.

Da alle möglichen Regelsätze in 18 binären Bits kodiert werden können und 2^{18} = 262.144, bedeutet dies, dass es 262.144 Möglichkeiten gibt, das Spiel des Lebens zu spielen. Damit ist es möglich, ein Gerät zu bauen, mit dem ein Benutzer all diese Möglichkeiten erforschen und beobachten kann, wie sich eine Population der verschiedenen „Arten" im Laufe der Zeit entwickelt.

Der letzte Teil des Softwareentwurfs entstand in vielen Gesprächen mit der Elektronikkünstlerin Kelly Heaton [5], die ich durch einen Elektor-Artikel [6] kennengelernt habe. Sie ermutigte mich, über das Kanonische hinauszudenken und Werke zu schaffen, die sowohl dem Algorithmus treu bleiben als auch visuell überzeugend sind. Dies veranlasste mich dazu, das Standard-Farbschema für Spielsimulationen zu ändern, bei dem besetzte Orte farbig und unbesetzte Orte schwarz sind. Ich wollte den Mustern einen stärkeren Sinn für Veränderung geben und färbte daher neugeborene Zellen blau; Zellen, die über mehr als eine Generation bestehen, grün; und Zellen, die sterben, hinterlassen einen tiefvioletten „Geist" für eine Generation (diese „Geister" werden nicht als Nachbarn gezählt). Dies ist in **Bild 2** detailliert dargestellt.

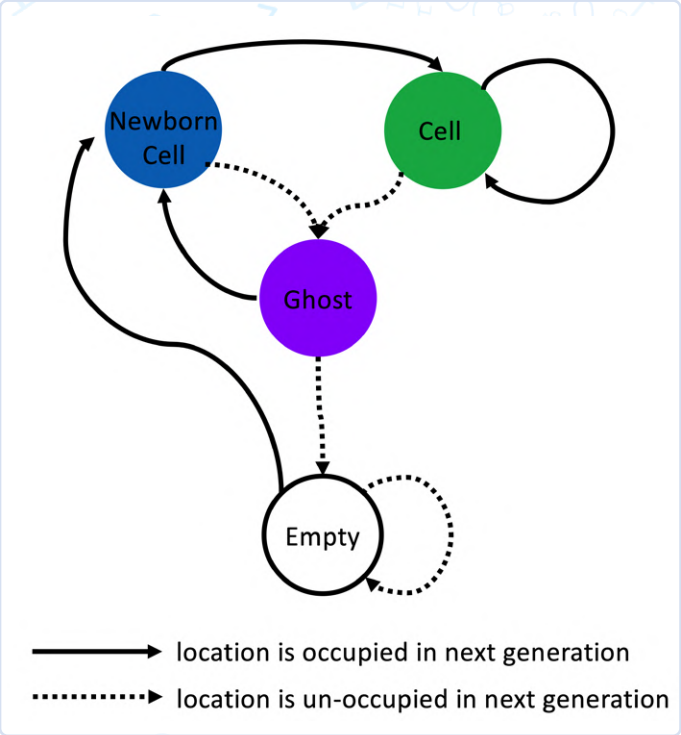


Bild 2. Zustandsdiagramm für ein mehrfarbiges Zellenmuster.

Zusammenbau der Hardware

Das letzte Teil, die Hardware, kam als Weihnachtsgeschenk von meinen Söhnen: ein Matrix-Portal von Adafruit [7] und eine 32×64-RGB-LED-Matrix [8]. Ich habe dann die Teile in einem Acrylgehäuse zusammengebaut, so dass man alle Innereien sehen kann. Das Endprodukt ist in **Bild 3** zu sehen.

Auf der linken Seite befinden sich drei Sätze von drei Daumenradschaltern, mit denen die Regeln und Parameter für den Lauf festgelegt werden. Der obere und mittlere Satz von drei Schaltern legt die Regeln in Form von sechs Oktalziffern fest. Jede Oktalziffer besteht aus drei Binärbits; was multipliziert mit sechs Ziffern die erforderlichen 18 Binärbits ergibt. Die oberen drei Ziffern geben die Regeln für besetzte Plätze an, also die Regeln dafür, welche Zellen

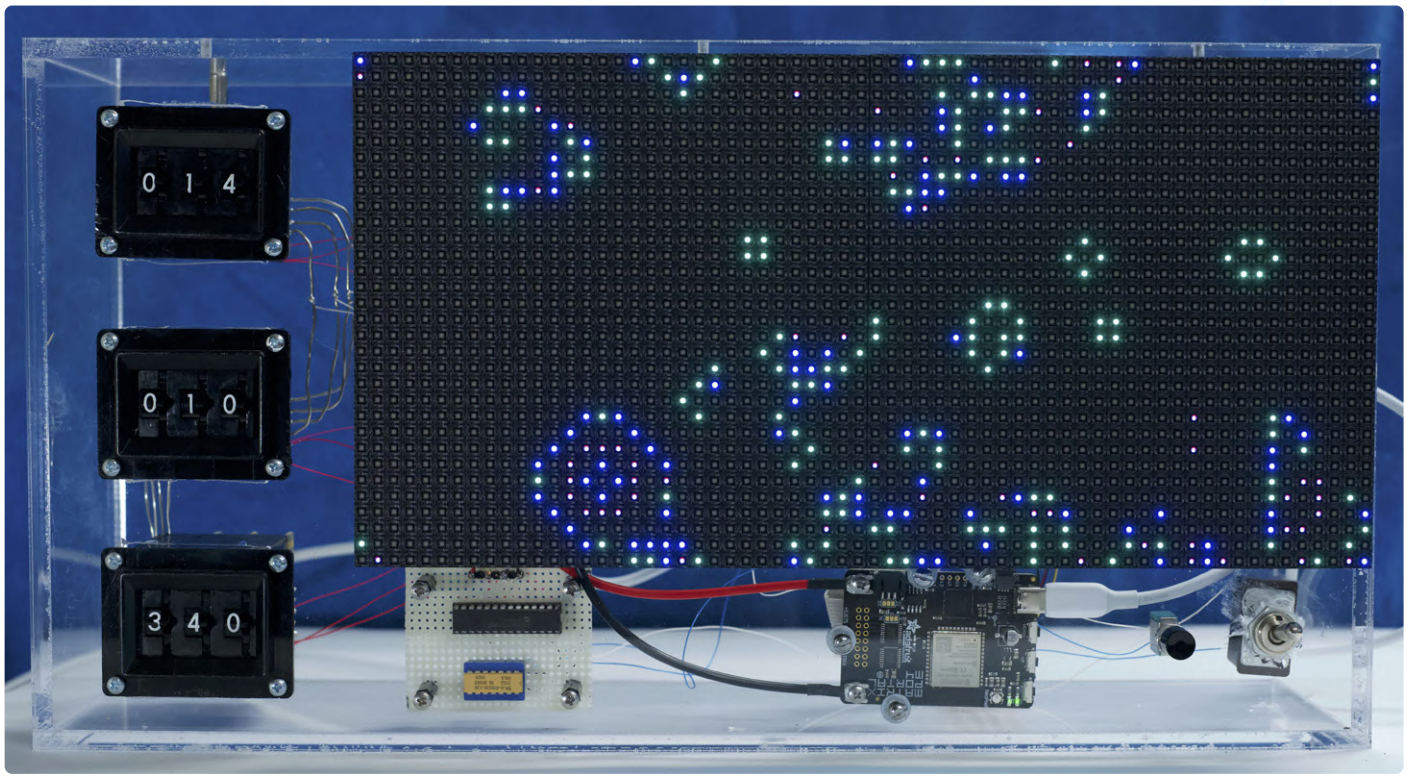


Bild 3. Demo des zusammengesetzten Projekts.

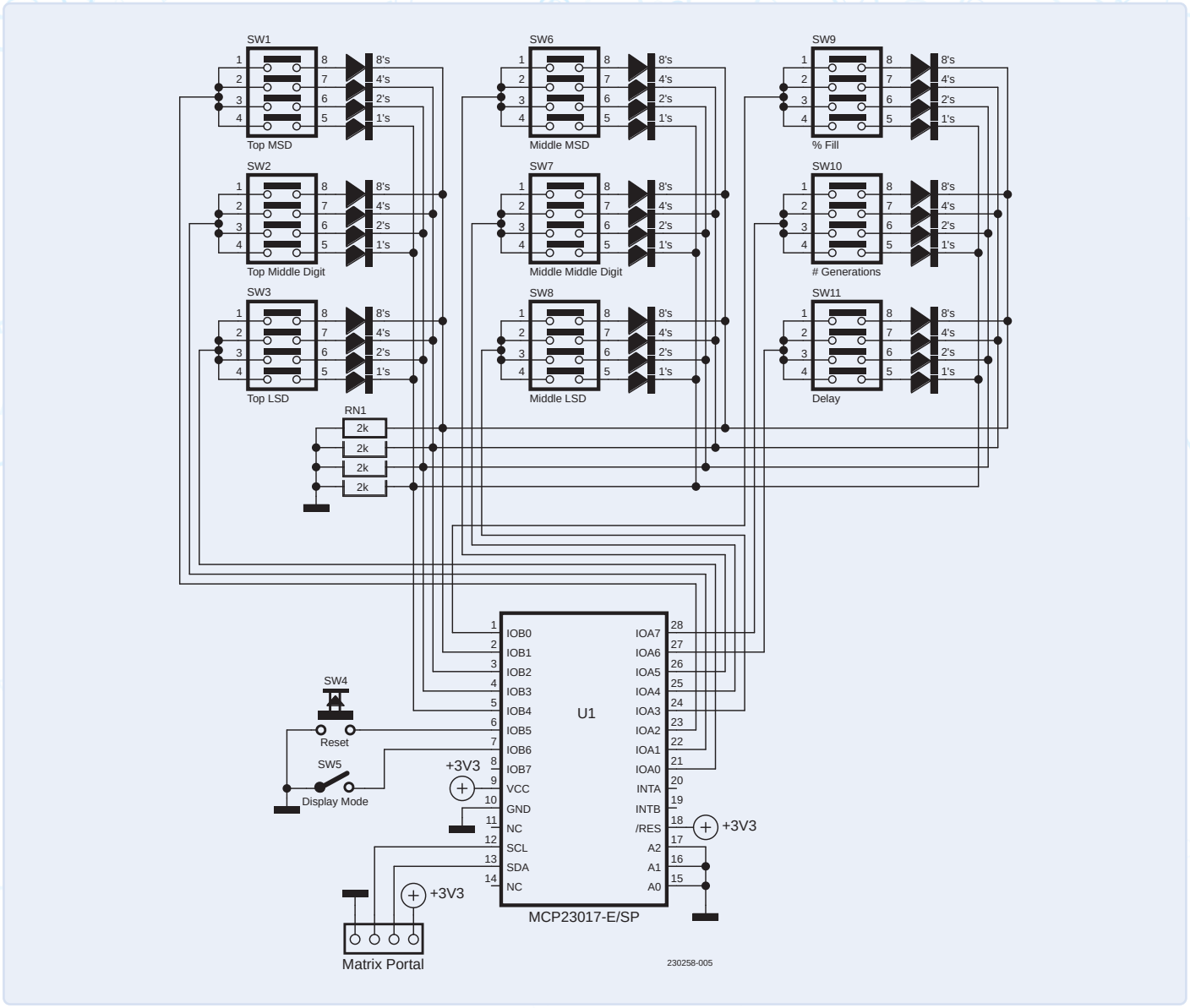


Bild 4. Schaltplan ohne Matrix-Portal, 64×32-Display und USB-Stromversorgung.

in der nächsten Generation überleben. Die mittleren drei Ziffern spezifizieren die Regeln für unbesetzte Plätze, also die Regeln für die Geburt neuer Zellen. Als Beispiel würden die kanonischen Regeln von Conway als 014010 ausgedrückt werden. Im Kasten **Kodierung der Regeln in sechs Oktalziffern** wird dies genauer beschrieben. Unterhalb der Anzeige befinden sich von links nach rechts die Port-Erweiterungsplatine mit dem MCP23017, die Matrix-Portal-MCU-Platine, der Reset-Taster und der Umschalter für den Anzeigemodus.

Jeder Durchlauf beginnt mit einer zufälligen Anordnung besetzter und unbesetzter Plätze und läuft dann für eine festgelegte Anzahl von Generationen weiter, bevor es mit einem neuen zufälligen Feld von Zellen neu beginnt. Die unteren drei Ziffern geben die Details dieses Prozesses an. Die linke Ziffer gibt die Dichte der Organismen in der Anfangspopulation an. Die Chance, dass ein Ort zu Beginn besetzt ist, beträgt das 10-fache der Einstellung dieses Daumenrädchens. Eine Einstellung von „3“ bedeutet also, dass die Welt zu Beginn zu etwa 30 % mit Zellen gefüllt ist. Da einige Regelsätze interessantere Ergebnisse mit mehr oder weniger besetzten Orten liefern, können so die Auswirkungen von unterschiedlichen initialen Zelldichten erforscht werden. Die mittlere Ziffer gibt die Anzahl der Generationen an, die bis zum Neustart durchlaufen werden. Hierfür habe ich eine exponentielle Skala gewählt, um eine große Bandbreite an Lauflängen zu ermöglichen. Genauer gesagt, läuft das Programm $2\times10^{(N/2)}$ Generationen; eine Einstellung einer „4“ würde also 200 Generationen lang laufen. Einige Muster lösen sich schnell auf, während andere sich nie auflösen; diese Steuerung trägt dazu bei, die Anzeige in einem interessanten Zustand zu halten. Die Ziffer ganz rechts schließlich gibt die Verzögerung in Sekunden zwischen den Generationen an; bei einer „0“ wird demnach mit maximaler Geschwindigkeit gearbeitet. Wenn die Generationen langsam aufeinander folgen, kann man die Regeln, wenn gewünscht, detaillierter im Detail verfolgen. Die letzten beiden Steuerelemente sind ein Druckknopf, der den Durchlauf sofort neu startet und die Population neu randomisiert, und ein Kippschalter, mit dem man die Standardfärbung (blau = besetzt; weiß = unbesetzt) oder mein mehrfarbiges Schema für die Orte in der Matrix auswählen kann.

Selberbauen

Der Schaltplan in **Bild 4** ist sehr einfach. Das Matrix-Portal und das Display sind wie in den Anleitungen von Adafruit beschrieben angeschlossen. Die gezeigten Erweiterungen erhalten ihre +3,3-V-Betriebsspannung und die I²C-Kommunikation vom

Stückliste

Adafruit Matrix Portal - CircuitPython-basiertes Internet-Display (Adafruit 4745)
64x32-RGB-LED Matrix – 5-mm-Raster (Adafruit 2277)
3x3-stelliger BCD-kodierter Daumenradschalter (Beispiel: Littlefuse 3P-3-23-3-1-0-2)
Port-Erweiterung MCP23017
4 Pull-up-Widerstände 2 kΩ
Druckknopf-Taster
Kippschalter 1×an (SPST)
USB-Steckernetzteil
Acrylglas-Gehäuse

Matrix-Portal über dessen Stemma-QT-Anschluss. Die Hauptarbeit wird von einem MCP23017-I²C-Port-Expander erledigt. Die BCD-Schalter (SW1 bis SW3 und SW6 bis SW11) werden einzeln durch Ausgänge des MCP23017 aktiviert, und die BCD-Ausgänge des aktivierten Schalters werden dann eingelesen. Da der MCP23017 nur über schwache (hochohmige) Pull-Downs verfügt, werden die BCD-Leitungen über externe Festwiderstände auf Masse gezogen. Die letzten beiden Eingänge des MCP23017 werden mit schwachen Pull-ups zum Einlesen der Reset- und Display-Modus-Schalter verwendet. Die Stromversorgung erfolgt über ein Steckernetzteil (5 V, 4 A), das auch das Matrix-Portal über ein USB-C-Kabel versorgt. Der Code basiert auf dem Spiel-des-Lebens-Code, der von Adafruit für das Matrix-Portal und das Display [9] zur Verfügung gestellt wurde und einige sehr clevere Tricks verwendet, um den verschachtelte Schleifen-Code sehr schnell auszuführen. Ich habe den Code so modifiziert, dass die Regeln statt auf dem Standard auf den Zahlen basieren, die auf den Daumenradschaltern eingegeben werden. Kurz gesagt, ich lese die binären Werte der Daumenräder ein und verwende sie, um eine 18-elementige Folge von Einsen und Nullen zu erstellen, die den Zustand des Ortes in der nächsten Generation repräsentiert, abhängig von der aktuellen Belegung des Ortes und der Anzahl der nicht leeren und nicht „geisterhaften" Nachbarn. Dann zähle ich die Nachbarn und verwende diese Zahl, wobei ich „9" addiere, wenn der Ort gerade besetzt ist, und verwende dies als Index für die Regelliste. Ich habe auch das oben beschriebene Multicolor-Muster im Code eingebaut. Der Code ist auf der Elektor-Labs-Projektseite [10] verfügbar.

Weitergehend

Die Ergebnisse sind faszinierend. Ich habe eine YouTube-Playlist [11] mit einer Sammlung kurzer Videos erstellt, die die Muster zeigen, die bei verschiedenen Einstellungen der Schalter entstehen. Es ist bemerkenswert, wie manche Änderungen der Regeln große Auswirkungen haben, während andere nur subtile Veränderungen in der Art und Weise bewirken, wie die Muster wachsen, sich verändern, sich bewegen und aussterben. Ich habe gerade erst begonnen, die Möglichkeiten zu untersuchen (262.144 ist eine ziemlich große Anzahl!), aber hier sind einige Dinge, die mir aufgefallen sind:

- Das Setzen der Bits niedrigster Ordnung (Bit 0 und Bit 9) und damit das Zulassen von Geburten und/oder Überleben mit Null-Nachbarn führt zu dramatischen Positiv/Negativ-Umkehrungen in jeder Generation.
- Die Einstellung von Bits höherer Ordnung, die die Geburten und das Überleben mit einer höheren Anzahl von Nachbarn steuern, hat tendenziell weniger Auswirkungen, vielleicht weil Orte mit vielen Nachbarn seltener sind.
- Der ursprüngliche Regelsatz 014010 entwickelt sich in der Regel zu einigen relativ stabilen Strukturen oder oszillierenden Figuren. Andere Regeln, zum Beispiel 114110, sind die Lieblingsregeln meiner Frau, da sie sich nie aufzulösen scheinen.
- Man kann Regeln erstellen, bei denen Strukturen entstehen und sich allmählich auf interessante Weise füllen, wenn man das Überleben bei vielen Nachbarn und die Geburt bei nur wenigen zulässt; zum Beispiel 256020.

Im Moment lebt das Gerät auf unserem Küchentisch und ich stelle oft fest, dass jemand im Haus einen neuen Regelsatz gefunden hat, der ein faszinierendes neues Muster ergibt.
Wir haben mit diesem Stück biologischer Elektronik gelebt und es mit anderen geteilt; und wir hatten mehrere Ideen für diejenigen, die vielleicht selber ein Spiel des Lebens bauen möchten. Erstens wäre es interessant, dem mehrfarbigen Schema weitere Farben hinzuzufügen. Zum Beispiel könnten die Farben allmählich durch mehr Farbtöne wechseln, wenn die Zellen „reifen". Kelly Heaton schlug außerdem vor, die Farben ineinander übergehen zu lassen, anstatt sie abrupt zu wechseln, um einen fließenden Effekt zu erzielen. Schließlich schlagen die nicht-oktalen Mitglieder meines Haushalts vor, eine Möglichkeit zu finden, die Entsprechung zwischen den Bits in den Regeln und der Anzahl der Nachbarn deutlicher darzustellen. Man könnte eine Reihe von 18 LEDs hinzufügen, um das binäre Äquivalent der oktalen Einstellungen anzuzeigen, oder, noch einfacher, 18 einzelne Schalter verwenden, einen für jedes Bit in den Regeleinstellungen.

Ich hoffe, dass die Leser dieses Artikels inspiriert werden, eine Version dieses Projekts zu bauen, entweder mit einer speziellen Anzeige und einem Mikrocontroller oder komplett auf einem Computer, und die Regelsätze, die sie interessant finden, mit anderen zu teilen. Wer hätte gedacht, dass das Teilen von sechsstelligen Oktalzahlen so interessant sein kann?

RG — 230258-02

Über den Autor

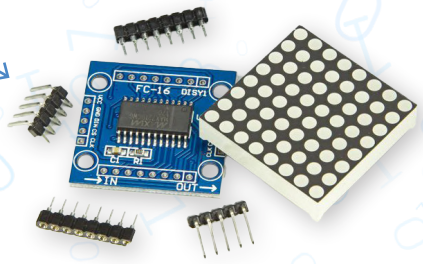
Brian White ist Professor im Fachbereich Biologie an der US-amerikanischen University of Massachusetts, Boston. Er hat ein Biologiestudium am MIT und in Stanford absolviert. Als Lehrer forscht er auch in den Bereichen Biologie und Biologieunterricht und entwickelt entsprechende Software. Brian ist außerdem Elektronik-Enthusiast, Musiker und Funkamateur (KA1TBQ). Weitere Einzelheiten zu seinen Elektronikprojekten finden Sie unter [12].

Haben Sie Fragen oder Kommentare?

Wenn Sie technische Fragen oder Kommentare zu diesem Artikel haben, wenden Sie sich bitte per E-Mail an den Autor unter brian.white@umb.edu oder an das Elektor-Redaktionsteam unter redaktion@elektor.de.

Passende Produkte

- **MAX7219 Punktmatrix-Modul (acht Stück)**
  www.elektor.de/18422
- **Adafruit Feather RP2040**
  www.elektor.de/19689
- **ESP32-C3-DevKitM-1**
  www.elektor.de/20324



WEBLINKS

[1] Conways Spiel des Lebens (Wikipedia): https://t1p.de/o9dy0
[2] Martin Gardner, „MATHEMATICAL GAMES: The fantastic combinations of John Conway's new solitaire game 'life'", Scientific American 223 (Oktober 1970): 120-123: https://web.stanford.edu/class/sts145/Library/life.pdf
[3] LifeWiki: https://conwaylife.com/wiki
[4] Stephen Wolfram, A New Kind of Science, see especially this section: https://t1p.de/izcs2
[5] Kelly Heaton Studio: https://kellyheatonstudio.com
[6] C.J. Abate, „Mit Elektrizität Kunst machen, Q&A mit Kelly Heaton", Elektor Circuit Special 2022: https://t1p.de/xlbw6
[7] Matrix-Portal — Internet-Display gesteuert von CircuitPython: https://adafruit.com/product/4745
[8] 64×32-RGB-LED-Matrix im 5-mm-Raster: https://adafruit.com/product/2277
[9] Adafruit-Beispiel von Conways Spiel des Lebens: https://t1p.de/dtu25
[10] Projektseite bei Elektor Labs: https://elektormagazine.de/labs/262144-ways-to-play-the-game-of-life
[11] Videos des Spiel des Lebens: https://t1p.de/ijuks
[12] Webseite des Autors: https://brianwhite94.wixsite.com/electronics

Aus dem Leben gegriffen

Der Chinesische Drache

Von Ilse Joostens (Belgien)

Feines chinesisches Kraak-Porzellan mit blau-weißen Motiven war in den Niederlanden sehr beliebt und wurde seit Anfang des 17. Jahrhunderts von der Niederländischen Ostindien-Kompanie importiert, die auch einen Standort in Delft hatte. Dieses Porzellan wurde vor allem als Dekorationsartikel in wohlhabenden Haushalten verwendet, und als der Import nach dem Fall der Ming-Dynastie 1644 eingestellt wurde, übernahmen die Delfter Töpfereien die Produktion von billigeren Kopien aus Steingut, die zunächst typische Ming-Motive auf weißer Zinnglasur zeigten. Nun, 380 Jahre später, scheinen die Rollen vertauscht zu sein, und jetzt ist es China, das westliche Produkte kopiert.

Am Morgen des 12. Oktober 1654 zerstörte Cornelis Soetens versehentlich einen großen Teil der Stadt Delft, als er eine Schießpulverprobe aus dem als „Secreet van Holland" (Geheimnis von Holland) bekannten Schießpulverdepot holen wollte. Der „Delfter Donnerschlag" und der Wiederaufbau der Stadt lösten eine rasante Entwicklung in der bereits florierenden Steingutindustrie aus, und Delfter Blau ist bis heute weltweit bekannt. Das Depot wurde wiederaufgebaut, allerdings weit außerhalb der Stadtmauern.

Brauchen Sie eine Kopie?

Natürlich war Delfter Blau [1] mehr als nur eine Kopie des chinesischen Porzellans, und die Delfter Keramikmaler haben nicht nur die Qualität ihrer Produkte auf die Spitze getrieben, sondern auch immer wieder mit neuen Techniken überrascht. Andererseits kopiert China den Westen buchstäblich - der 2019 eröffnete Forschungscampus des chinesischen Technologieunternehmens Huawei in Dongguan sieht aus wie ein gefälschtes Europa [2]. Der Zug, der an Bahnhöfen wie Heidelberg, Bologna und Granada hält, ähnelt deutlich der Schweizer Jungfraubahn. Schon früher wurden Teile von Paris wie die Champs-Élysées und das malerische österreichische Dorf Hallstatt [3] als Touristenattraktionen nachgebaut. Leider geht das Kopieren noch viel weiter, und zwar bei allen Arten von Produkten, einschließlich Kunstwerken. Chinesische Geschäfte ahmen das Aussehen, die Atmosphäre und den Service erfolgreicher westlicher Einzelhandelskonzepte nach, zum Beispiel eines bekannten schwedisches Möbelhauses oder Geschäfte einer Computermarke mit einer halb gegessenen Frucht als Logo.

Diese Geschäfte wurden nach einigen juristischen Auseinandersetzungen geschlossen, aber das Kopieren von Produkten und Ideen geht weiter. Vor nicht allzu langer Zeit hat es mich fast vom Stuhl gehauen, als ich bei Aliexpress eine zwei Meter hohe Nachbildung der Skulptur *Kindred Spirits* von Alex Pentek [4] sah. Auch der Elektroniksektor ist nicht verschont geblieben. Viele kennen die gefälschten ICs und recycelten Bauteile, die als neu verkauft werden. Neben dubiosen Bauteilen können Sie alle Arten von elektronischen Modulen, Kits, Gadgets, Werkzeugen und Maschinen in allen Größenordnungen zu sehr niedrigen Preisen bestellen. Sogar Nischenprodukte werden ins Visier genommen, und Dalibor Farný, der tschechische Hersteller der Nixie-Röhren R|Z568M, fürchtet nun sicherlich eine gewaltige Konkurrenz. 2016 habe ich für Elektor einen Artikel über 7-Segment-Anzeigen [5] geschrieben, die auf (damals noch relativ neuen) LED-Fäden basieren. Offenbar kann man jetzt bei Onkel Ali Bausätze für Uhren mit LED-Fäden zu Preisen bestellen, bei denen ich nicht einmal die Bauteile zusammenbekomme. Auch die Versandkosten sind minimal, da China mit dem Status eines Entwicklungslandes beim Weltpostverein viel billiger versenden kann [6]. Es scheint, dass dieser Verein den Teil übersehen hat, in dem China erfolgreich ein Raumschiff auf der dunklen Seite des Mondes gelandet hat. Ich habe nichts dagegen, dass eine Idee verwendet wird, aber wenn

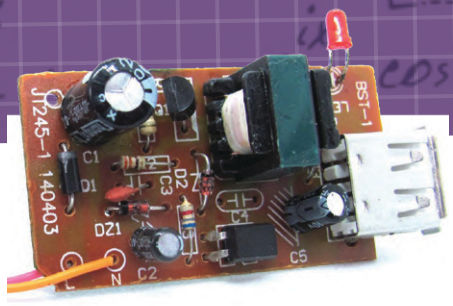


Bild 1. Billiges USB-Steckernetzteil, Bauteilseite…

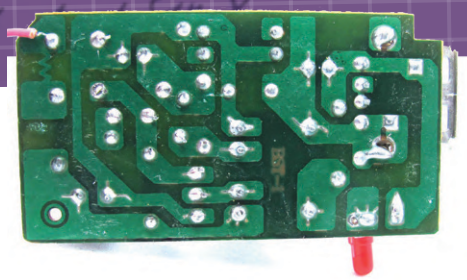


Bild 2. …und Kupferseite. (Bilder 1 und 2: Fotos von Leo Potjewijd).

Bild 3. Das Leuchtmittel aus der in China hergestellten Salzlampe.

die eigene Idee verwendet wird, um selbst vom Markt verdrängt zu werden, ist das schmerzhaft.

Tofu-Dreg-Projekte

Wie in vielen Gesellschaften gibt es auch in China Korruption, Bestechung, Lügen und Betrug auf allen Ebenen, obwohl die Mehrheit der Bürger freundlich und aufrichtig ist. Die Probleme, die diese hinterhältigen Machenschaften mit sich bringen, sind nicht unerheblich, wozu Gebäude zählen, die aus minderwertigen Materialien gebaut wurden und häufig einstürzen. Benannt nach den Resten, die bei der Herstellung von Tofu übrig bleiben, dem „Tofu-Dreg", ist dies auch in der Umgangssprache ein synonym für schlecht ausgeführte Projekte und schlampige Arbeit [7]. Neben gefährlicher Infrastruktur und sogar gefälschten Lebensmitteln hat auch der Elektroniksektor mit diesen Problemen zu kämpfen. Viele chinesische Konsumgüter, die hierher importiert und verkauft werden, sind nicht nur von schlechter Qualität, sondern manchmal sogar lebensbedrohlich.

Immer dabei sind billige Netzteile und USB-Ladegeräte, die aus einer minimalen Anzahl von Bauteilen bestehen. Ein Isolationsabstand zwischen Primär- und Sekundärteil ist praktisch nicht vorhanden, und eine dünne Leiterbahn dient als Sicherung. So ist es nicht verwunderlich, dass gelegentlich jemand in der Badewanne einen tödlichen Stromschlag durch ein Smartphone erleidet. Die Medien stellen dies als Bagatelle dar: Abgesehen von der allgemeinen Warnung, das Smartphone beim Baden nicht an das Ladegerät anzuschließen, wird den Gründen dafür keine Aufmerksamkeit geschenkt. In einem Forum sah ich kürzlich ein weiteres Foto vom Innenleben eines solchen Netzteils (**Bild 1** und **Bild 2**). Neben den bereits erwähnten Mängeln hat der Optokoppler auf der Platine keine Funktion, wenn man sich die gedruckte Schaltung genau ansieht. Also ein völlig nutzloses Bauteil.

Auch über die Salzlampe, die ich zu Weihnachten geschenkt bekommen habe, war ich nicht erfreut. Sie war dimmbar, flackerte aber nervenaufreibend, wenn der Dimmer nicht maximal aufgedreht war, und ich habe schließlich alles auseinandergenommen und überarbeitet. Die interne LED-Leuchte (**Bild 3**) bestand aus nichts weiter als einem Brückengleichrichter, einem IC, das glühend heiß wurde, und einer LED-Matrix - nicht einmal ein Glättungskondensator war vorhanden. Das Gehäuse des Dimmers (**Bild 4**) konnte ohne Werkzeug zerlegt werden, und das Netzkabel war nichts weiter als verkupfertes Aluminium mit einer Querschnittsfläche von weniger als einem Zehntel Quadratmillimeter pro Leiter (**Bild 5**). In Kombination mit einem gefälschten chinesischen Leitungsschutzschalter [8] sind fantastische pyrotechnische Effekte fast garantiert (**Bild 6**). Es ist nicht immer einfach, aber mehr lokal produzierte Waren zu kaufen und unseren eigenen Leuten eine Chance zu geben, scheint nicht die schlechteste Idee überhaupt zu sein.

SG — 230609-02

Bild 4. Der Dimmer der Salzlampe.

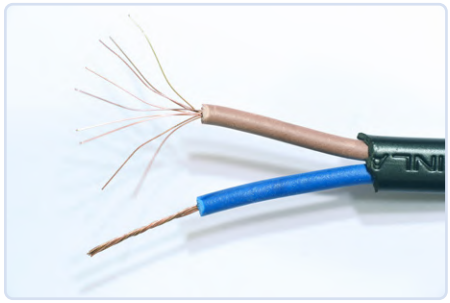


Bild 5. Das erschreckend dünne Netzkabel der Salzlampe…

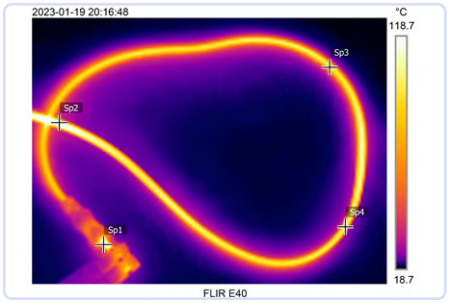


Bild 6. …wurde bei 6 A beängstigend heiß (fast 118°C bei Sp2).

WEBLINKS

[1] Wikipedia: Delfter Blau: https://de.wikipedia.org/wiki/Delfter_Blau
[2] The Atlantic, „Photos of Huawei's European-Themed Campus in China": https://tinyurl.com/atlanticoxhorn
[3] Wikipedia: Hallstatt (China): https://de.wikipedia.org/wiki/Hallstatt_(China)
[4] Wikipedia: Kindred Spirits: https://de.wikipedia.org/wiki/Kindred_Spirits
[5] Peter S'heeren, „LEDitron", Elektor 4/2016: https://www.elektormagazine.de/magazine/elektor-201604/28882
[6] RTL News: „Waarom AliExpress zo goedkoop pakketjes kan versturen" (Niederländisch): https://tinyurl.com/rtlaliexpress
[7] China Insights, „Fragile steel bars/Tofu-dreg project in China/Shaky building/Collapsing buildings/Poor quality" (YouTube): https://youtu.be/s-2DtL-Wjkc
[8] bigclivedotcom, „Inside a fake un-trippable circuit breaker" (YouTube): https://youtu.be/2TJEzdqtXlQ

Bringen Sie Ihren **DC-Bürstenmotor zum Laufen!**

Beispielprojekte aus dem Motor Control Development Bundle von Elektor

Von Dogan Ibrahim (Großbritannien)

Es gibt keinen besseren Einstieg ins Programmieren und in die praktische Elektronik, als Dinge zum Laufen, Summen oder Blinken zu bringen. Zum „Laufen“ gehört oft ein Motor: Ein kleiner Gleichstrom-Bürstenmotor eignet sich hervorragend, um einige Grundlagen zu erlernen, bevor man sich an ernsthafte Robotertechnik und Mechatronik heranwagt. Hier machen wir die ersten Schritte zur intelligenten Steuerung von Gleichstrommotoren, unterstützt von dem fantastischen Entwicklungskit von Elektor, das Hardware, Software und ein solides Handbuch kombiniert.

Anmerkung der Redaktion: *Dieser Artikel ist ein Auszug aus dem 192-seitigen Elektor-Buch, das Teil des* Motor Control Development Bundles *ist. Dieser Auszug wurde formatiert und leicht bearbeitet, um den Konventionen und dem Seitenlayout der Zeitschrift Elektor zu entsprechen. Der Autor und die Redaktion stehen für Rückfragen gerne zur Verfügung. Kontaktinformationen finden Sie im Kasten* **Fragen oder Kommentare**.

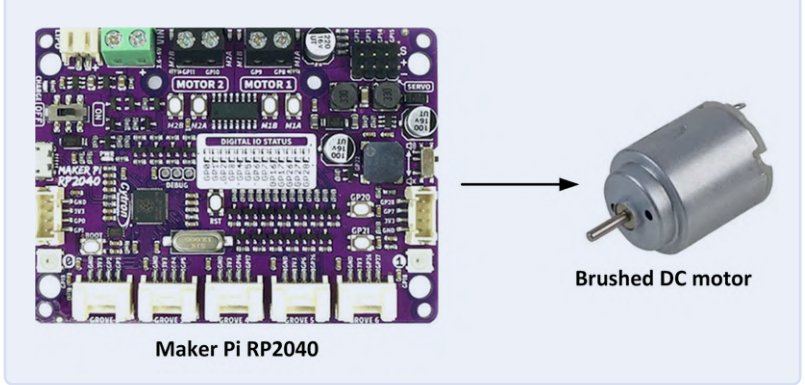


Bild 1. Die beiden Hauptbestandteile des Projekts.

In diesem Artikel werden drei einfache Projekte mit einem kleinen, bürstenbehafteten Permanentmagnet-Gleichstrommotor beschrieben, der von der Entwicklungsplatine *Maker Pi RP2040* der Firma Cytron angesteuert wird. Sowohl Motor als auch Entwicklungsplatine sind im *Motor Control Development Bundle* enthalten, das im Elektor-Store [1] erhältlich ist. Die Projekte sind eher experimentell und lehrreich als voll ausgereifte, praxistaugliche Beispiele.

DC-Motor Ein/Aus-Steuerung

Dies ist ein einfaches Projekt, das zeigt, wie ein kleiner, bürstenbehafteter Gleichstrommotor, der an einer Gleichspannung von 1...6 V betrieben wird, an eine der DC MOTOR-Klemmenleisten des Maker Pi RP2040 angeschlossen wird. Der Motor wird für 5 s in Vorwärtsrichtung eingeschaltet und dann für 5 s angehalten. Danach wird er 5 s lang in umgekehrter Richtung betrieben und wieder 5 s lang angehalten. Dieser Vorgang wird so lange wiederholt, bis er manuell gestoppt wird. Das Ziel dieses Projekts ist es, den Anschluss, den Betrieb und die Steuerung eines solchen bürstenbehafteten Gleichstrommotors mit dem Entwicklungsboard Maker Pi RP2040 zu zeigen. **Bild 1** zeigt das „Blockschaltbild“ des Projekts und **Bild 2** den tatsächlich verwendeten Motor. Es handelt sich um einen kleinen DC-Bürstenmotor, der für den Betrieb bei +1 V bis +6 V mit einer empfohlenen Betriebsspannung von +3 V ausgelegt ist. Schauen wir uns an, wie man ihn zum Drehen bringt.

Auf dem Entwicklungsboard befindet sich ein Zweikanal-H-Brücken-Motortreiber, der bis zu zwei DC-Bürstenmotoren M1 und M2 oder einen Schrittmotor steuern kann. Die I/O-Pin-Belegung ist:

- M1A: GP8
- M1B: GP9
- M2A: GP10
- M2B: GP11

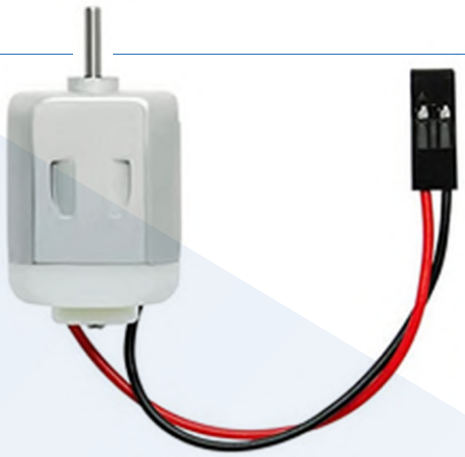


Bild 2. Der verwendete kleine Gleichstrommotor.

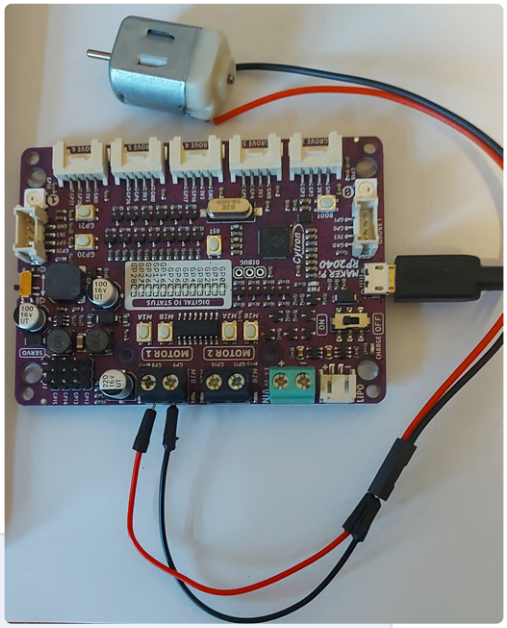


Bild 3. Anschluss des Motors an das Dev-Board RP2040 Maker Pi.

Die Wahrheitstabelle des Motortreibers ist in **Tabelle 1** dargestellt. In diesem Projekt werden die Motoreingänge PWM1A (Port GP8) und PWM1B (Port GP9) mit den Motorausgängen M1A und M1B verwendet. Diese Ausgänge (*Output A*) sind an den Schraubklemmen MOTOR 1 im mittleren oberen Teil der Entwicklungsplatine verfügbar. Tabelle 1 zeigt, dass der Motor von den Anschlüssen GP8 und GP9 gesteuert wird. Setzt man beispielsweise GP8 auf HIGH und GP9 auf LOW, dreht sich der Motor vorwärts, dreht man die Pegel um (GP8 auf LOW und GP9 auf HIGH), dreht sich auch der Motor in die andere Richtung. **Bild 3** zeigt das Projekt, bei dem der Motor an die Ausgänge MOTOR 1 angeschlossen ist. Das Programm-Listing für diese einfache Demo zum Einstieg ist in **Listing 1** enthalten. Das Programm mit dem Namen *DCMotor1.py* ist in einer großen Software-Archivdatei enthalten, die auf der Elektor-Website [1] unter *Downloads Software_Motor Control Development Kit* kostenlos erhältlich ist.

Der Motor wird mit PWM-Signalen gesteuert, wie es an anderer Stelle im genannten Lehrbuch beschrieben wird. In diesem Projekt wird die schon in der IDE integrierte DC-Motor-Bibliothek *adafruit_motor* verwendet. Zu Beginn des Programms werden die Bibliotheksmodule *board*, *time*, *pwmio* und *motor* in das Programm importiert. Die Anschlüsse GP8 und GP9 von Motor 1 werden dann so konfiguriert, dass sie mit einem PWM-Signal mit einer Frequenz von 50 Hz arbeiten. Der Rest des Programms läuft in einer Endlosschleife ab.

Die Eigenschaft der Klasse `throttle` im Programm kann einen Wert zwischen -1 und +1 annehmen und steuert den Motor wie folgt:

`throttle = 0`	Motor im Leerlauf
`throttle = 1`	Motor dreht sich mit voller Geschwindigkeit vorwärts
`throttle = 0.5`	Motor dreht sich mit 50% seiner vollen Geschwindigkeit vorwärts
`throttle = -1`	Motor dreht sich mit voller Geschwindigkeit rückwärts
`throttle = -0.5`	Motor dreht sich mit 50% seiner vollen Geschwindigkeit rückwärts

In diesem Programm dreht der Motor mit 25 % seiner vollen Geschwindigkeit, weil `throttle` auf +0.25 in einer Richtung und -0.25 in der entgegengesetzten Richtung eingestellt wird.

Listing 1: DCMotor1.py

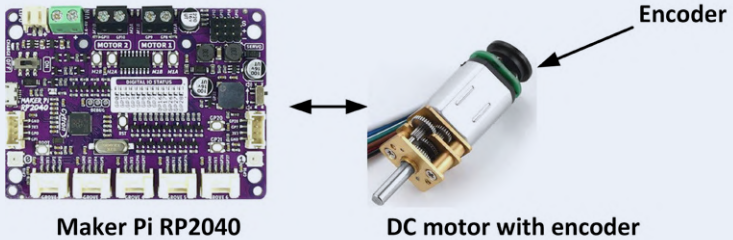
```
#---------------------------------------------------------
#                BRUSHED DC MOTOR CONTROL
#
# In this program a brushed DC moor is controlled as follows:
# The motor is turned ON in forward direction for 5 seconds
# and then stopped for 5 seconds, then in reverse direction
# for 5 seconds and then stopped again for 5 seconds. This
# process is repeated forever
#
# Author: Dogan Ibrahim
# File  : DCMotor1.py
# Date  : February, 2023
#---------------------------------------------------------
import board
import time
import pwmio
from adafruit_motor import motor
#
# Initialize DC Motor 1
#
m1a = pwmio.PWMOut(board.GP8, frequency=50)
m1b = pwmio.PWMOut(board.GP9, frequency=50)
motor1 = motor.DCMotor(m1a, m1b)
while True:
    motor1.throttle = 0.25              # 25% of full speed
    time.sleep(5)
    motor1.throttle = 0                 # Idle
    time.sleep(5)
    motor1.throttle = -0.25             # 25% of full speed
    time.sleep(5)
    motor1.throttle = 0                 # Idle
    time.sleep(5)
```

Tabelle 1: Motortreiber-Wahrheitstabelle.

Eingang A	Eingang B	Ausgang A	Ausgang B	Motor-Aktion
Low	Low	Low	Low	Bremse
High	Low	High	High	Vorwärts
Low	High	Low	High	Rückwärts
High	High	Hi-Z (offen)	Hi-Z (offen)	Auslaufen

Listing 2: DCMotor2.py

```
#-------------------------------------------------------------
#        TWO SPEED BRUSHED DC MOTOR CONTROL
#
# In this program a brushed DC moor is controlled as follows:
# The motor normally rotates at 25% of its full speed. Pressing
# button at port GP20 increses the speed to 50% of its full
# speed. Releasing the botton returns the speed to 25%
#
# Author: Dogan Ibrahim
# File  : DCMotor2.py
# Date  : February, 2023
#-------------------------------------------------------------
import board
import time
import pwmio
import digitalio
from adafruit_motor import motor
#
# Initialize DC Motor 1
#
m1a = pwmio.PWMOut(board.GP8, frequency=50)
m1b = pwmio.PWMOut(board.GP9, frequency=50)
motor1 = motor.DCMotor(m1a, m1b)
#
# Configure button at port GP20. The button state is
# normally at logic 1
#
btn = digitalio.DigitalInOut(board.GP20)
btn.direction = digitalio.Direction.INPUT
btn.pull = digitalio.Pull.UP
while True:
    motor1.throttle = 0.25              # 25% of full speed
    while btn.value == 0:               # If button pressed
        motor1.throttle = 0.5           # Increase speed
```



▲

Bild 4. Das Blockschaltbild des Projekts.

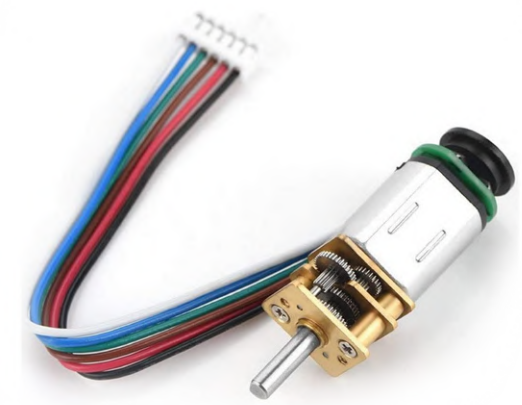


Bild 5. Die Motor/Encoder-Baugruppe mit der Typenbezeichnung GBMQ-GM12BY20.

▶

DC-Motor-Drehzahlregelung mit zwei Geschwindigkeiten

Auch hier handelt es sich um ein sehr einfaches Projekt, bei dem ein kleiner DC-Bürstenmotor an das Entwicklungsboard Maker Pi RP2040 angeschlossen wird. Der On-Board-Taster an Port GP20 wird ebenfalls in diesem Projekt verwendet. Normalerweise dreht sich der Motor mit 25 % seiner vollen Geschwindigkeit. Durch Drücken des Tasters wird er auf 50 % seiner vollen Drehzahl hochgefahren.

Die „Zutaten" und die Verbindungen zwischen Motor und Entwicklungsboard sind für dieses Projekt die gleichen wie im vorherigen Projekt.

Listing 2 zeigt das Programm *DCMotor2.py*. Das Hauptprogramm läuft in einer Schleife, in der der Zustand der Taste an GP20 überprüft wird. Solange der Taster gedrückt ist, wird die Motordrehzahl auf 50 % der vollen Drehzahl erhöht.

Anzeige der DC-Motordrehzahl mit Hilfe eines Drehgebers

Dieses Projekt zeigt, wie ein Drehgeber verwendet werden kann, um die Drehzahl eines Gleichstrommotors zu messen. Die Drehzahl wird dann auf dem Monitor angezeigt. Beachten Sie, dass ein Motor wie in **Bild 4** mit einem eingebauten Drehgeber nicht im *Motor Control Development Kit* enthalten ist und separat erworben werden muss. Bezugsquellen sind eBay, AliExpress und andere.

Es wird ein kleiner DC-Bürstenmotor mit Getriebe und eingebautem Drehgeber verwendet. **Bild 5** zeigt ein Bild des Motors, bei dem der Drehgeber hinten an der Welle des Motors befestigt ist. In diesem Projekt wird ein Motor mit folgenden Eigenschaften verwendet: 6 V_{DC}-Getriebemotor, 2 W, Typ GBMQ-GM12BY20 mit Encoder, Leerlaufdrehzahl 70 U/min. In diesem Projekt verwenden Sie eine externe Spannungsversorgung von nur +5 V für den Motor, so dass die Leerlaufdrehzahl etwas unter 70 U/min liegen dürfte. Die Stromaufnahme des Motors kann sich auf bis zu 200 mA belaufen.

Ein Drehgeber (oder Drehencoder) ist ein Gerät, das die Winkelposition eines rotierenden Bauteils wie eines Motors in ein elektrisches Signal umwandelt. In Motorsteuerungen werden oft Drehgeber eingesetzt, um die Drehzahl des Motors oder die Position der Motorwelle zu erfassen. Es gibt grundsätzlich zwei Arten von Drehgebern: optische und Hall-Effekt-Drehgeber. Bei optischen Drehgebern scheint Licht durch Schlitze (oder Löcher) in einer Metall- oder anderes gearteten Scheibe auf eine Fotodiode. Indem man die Anzahl der Löcher zählt, die in einer bestimmten Zeit vor der Fotodiode vorbeihuschen, kann man (will heißen, der Mikrocontroller) die Drehzahl des Motors berechnen.

In diesem Projekt wird aber ein auf dem Hall-Effekt basierender Drehgeber verwendet. Zwei auf der Encoderpla-

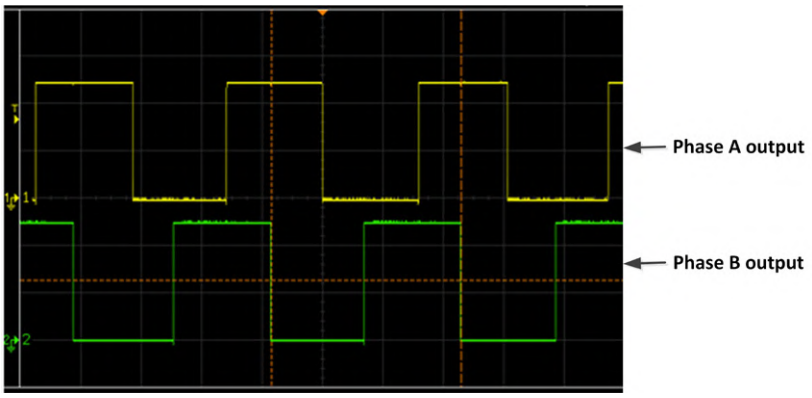


Bild 6. Ausgangssignal des Drehgebers.

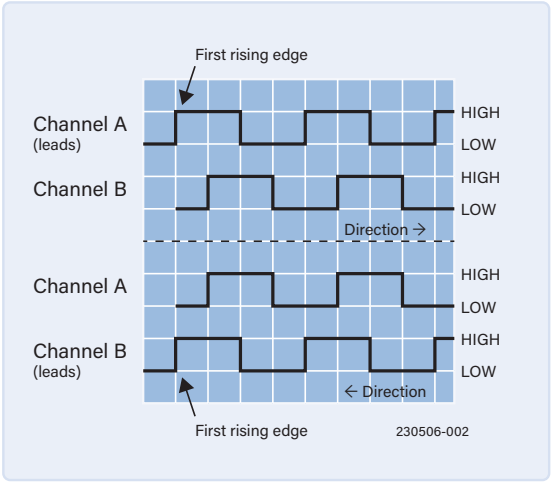


Bild 7. Bestimmung der Drehrichtung (Quelle: robotoid.com).

tine fest angebrachte Magnetsensoren (Phase A und Phase B genannt) werden zusammen mit einer rotierenden Scheibe mit Magneten. Die Sensoren liefern Impulse, wenn sich der Motor dreht. Durch Zählen der Impulse über eine bestimmte Zeit hinweg lässt sich die Drehzahl des Motors berechnen. **Bild 6** zeigt die Ausgangssignale des in diesem Projekt verwendeten Motors. Die obere Wellenform stellt die Phase A und die untere die Phase B dar. Diese Art von Encodern wird auch als Quadratur-Encoder bezeichnet, da die vier Magnetsensoren in einem Winkel von 90 Grad zueinander angeordnet sind und es vier mögliche Ausgangszustände gibt. Die Drehrichtung lässt sich auf diese Weise leicht feststellen, indem man die Reihenfolge ermittelt, in der die Ausgangssignale von 0 auf 1 wechseln. Wenn beispielsweise das Signal der Phase A von seinem LOW-Zustand aus ansteigt, dreht sich der Motor in eine Richtung. Steigt dagegen das Phase-B-Signal an, während das Phase-A-Signal LOW ist, dann dreht sich der Motor in die entgegengesetzte Richtung - siehe **Bild 7**.
Die Eigenschaften des in diesem Projekt verwendeten Motors sind wie folgt (abhängig von der angelegten Spannung):

- 6-V-Gleichstrombetrieb
- Drehzahl nach dem Getriebe: 70 U/min (bei 6 V)
- Leistung: 2 W
- Stromaufnahme: 170...200 mA
- Zwei eingebaute Hall-Effekt-Sensoren
- Gewicht: 14 g

An der Motorrückseite ist ein 6-poliger Stecker mit folgender Anschlussbelegung angebracht.

Schwarzes Kabel	Motorleistung GND
Rotes Kabel	Motorspannung (+5 V in diesem Projekt)
Yellow	Encoder-Spannung (+3.3 V in diesem Projekt)
Gelb	Encoder-Spannung GND
Blau	Encoder-Ausgang Phase A
Weiß	Encoder-Ausgang Phase B

Bild 8 zeigt den Schaltplan des Projekts. Beachten Sie, dass der Motor mit einer externen +5-V-Stromversorgung betrieben wird, die bis zu 500 mA liefern kann. Der Encoder wird dagegen an der +3,3-V-Leitung des Entwicklungsboards angeschlossen, damit die Spannung mit den Eingangsspannungspegeln des RP2040-Prozessors kompatibel ist. In diesem Projekt wird nur der Phase-A-Ausgang des Motors verwendet.
Leider geben die Hersteller und Vertreiber von Elektromotoren meist weder das Übersetzungsverhältnis noch die Anzahl der vom Motor pro Sekunde ausgegebenen

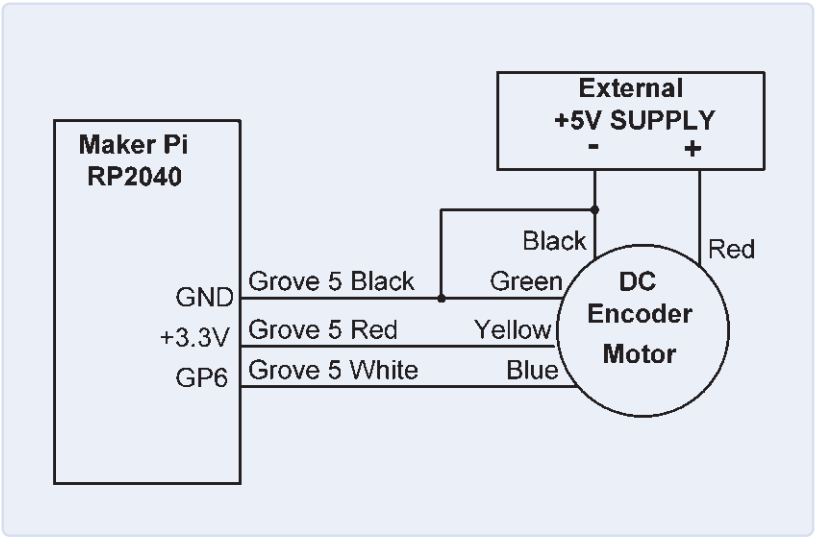


Bild 8. Verdrahtungsplan des Projekts.

Encoderimpulse an, obwohl dies wichtige Angaben sind. In diesem Abschnitt wird deshalb ein Programm beschrieben, das diese wichtigen Parameter für den verwendeten Motor ermittelt. Möglicherweise müssen Sie diese Berechnungen durchführen, um die wichtigen Spezifikationen des von Ihnen verwendeten Motors zu ermitteln. **Listing 3** zeigt das Programm *MotorPulses.py*, das die Anzahl der Motorimpulse angibt, die jede Sekunde von Phase A des Drehgebers ausgegeben wird. In diesem Programm wird der Port GP6 dem Objekt `Encoder` zugewiesen und als Eingang konfiguriert, die Anzahl der vom Encoder empfangenen Impulse wird berechnet und am Ende jeder Sekunde angezeigt. Dieses Programm verwendet die Funktion `time.monotonic()`, um das

Listing 3: MotorPulses.py

```
#-------------------------------------------------------
#       DISPLAYING THE NUMBER OF ENCODER PULSES
#
# In this program a geared DC motor with Hall Effect sensors
# is connected to the Maker Pi. This program calculates
# and displays the number of pulses received from the encoder
# every second. Only Phase A encoder output is used here
#
# Author: Dogan Ibrahim
# File  : MotorPulses.py
# Date  : March, 2023
#-------------------------------------------------------
import board
import digitalio
import time
Encoder = digitalio.DigitalInOut(board.GP6)
Encoder.direction = digitalio.Direction.INPUT
while Encoder.value == 0:          # Wait while 0
    pass
strtime = time.monotonic()         # Start time,rising edge
                                   # detected
count = 0                          # Intialize count
#
# Main program loop
#
while True:
    while Encoder.value == 1:      # Wait while 1
        pass
    while Encoder.value == 0:      # Wait while 0
        pass
    endtime = time.monotonic()     # End time
    if endtime-strtime > 1.0:      # Just over a second
        print(count)               # Display count
        strtime = time.monotonic() # Start time
        count = 0
    else:
        count = count + 1          # Increment count
    while(Encoder.value) == 1:     # Wait while 1
        pass
```

Listing 4: MotorSpeed.py

```
#-------------------------------------------------------
#         DISPLAYING THE MOTOR SPEED
#
# This program displays the motor speed in RPM using the
# number of encoder pulses received every second and the
# formula given in the text
#
# Author: Dogan Ibrahim
# File  : MotorSpeed.py
# Date  : March, 2023
#-------------------------------------------------------
import board
import digitalio
import time
Encoder = digitalio.DigitalInOut(board.GP6)
Encoder.direction = digitalio.Direction.INPUT
while Encoder.value == 0:                Wait while 0
    pass
strtime=time.monotonic()                # Start time
count=0                                 # Intialize count
#
# Main program loop
#
while True:
    while Encoder.value == 1:           # Wait while 1
        pass
    while Encoder.value == 0:           # Wait while 0
        pass
    endtime=time.monotonic()            # End time
    if endtime-strtime > 1.0:           # Just over a second
        RPM = count * 60 / 880          # Motor speed
        print("Speed=%6.2f RPM" %RPM)   # Display speed
        strtime=time.monotonic()        # Start time
        count=0
    else:
        count=count+1                   # Increment count
    while(Encoder.value) == 1:          # Wait while 1
        pass
```

```
CircuitPython REPL
Speed= 63.95 RPM
Speed= 64.02 RPM
Speed= 64.02 RPM
Speed= 63.95 RPM
Speed= 63.89 RPM
Speed= 63.95 RPM
```

Bild 9. Ausgabe des Programms.

Bild 10. Der digitale Drehzahlmesser DT-2234C.

Passendes Produkt

- **Motor Control Development Bundle**
  SKU 20534: www.elektor.de/20534

Timing zu berechnen, die möglicherweise nicht sehr genau ist. Besser wäre es, die Impulse als externe Interrupts zu behandeln und die Anzahl der empfangenen Impulse mit Hilfe von einsekündigen Timer-Interrupts zu berechnen und anzuzeigen.
Wenn der Motor läuft, zeigt das Programm *MotorPulses.py* 938 Impulse/Sekunde an. Ein rundes Objekt (ein kleiner Reifen) wurde an der Motorwelle befestigt und eine Markierung darauf angebracht, um die Umdrehungen zählen zu können. Die Anzahl der Umdrehungen der Motorwelle im Leerlauf wurde mit 64 U/min festgestellt. Die Anzahl der pro Minute empfangenen Impulse beträgt nun 938 × 60 = 56.280 Impulse/Minute, was einer Leerlaufdrehzahl von 64 U/min entspricht. 56.280/64 ist gleich 879,38 (oder 880 als nächste Ganzzahl), und dann kann die Motordrehzahl mit der folgenden Formel berechnet werden:

Motordrehzahl (min^{-1}) = (gemessene Anzahl von Impulsen pro Sekunde × 60) / 880

Wenn die Anzahl der empfangenen Impulse beispielsweise 450 s^{-1} beträgt, ist die Motordrehzahl wie folgt:

Motordrehzahl = 450 × 60 / 880 = 30,68 min^{-1}

Mit der obigen Formel wird die Motordrehzahl berechnet. Das Programm *MotorSpeed.py* in **Listing 4** wird verwendet, um die Motordrehzahl anhand der obigen Formel zu berechnen und dann anzuzeigen. Dieses Programm ähnelt dem in Listing 3, aber hier wird die Motordrehzahl berechnet und auf dem Bildschirm angezeigt. **Bild 9** zeigt einige Beispielausgaben des Programms.
Wie bereits erwähnt, hätten genauere Ergebnisse erzielt werden können, wenn externe Interrupts zum Zählen der Encoderimpulse und Timer-Interrupts zum Messen der Zeit verwendet worden wären.

Misst der RP2040 richtig?

Die Motordrehzahl kann leicht mit einem digitalen Drehzahlmesser gemessen **und überprüft** werden. Es gibt viele solcher Geräte zu unterschiedlichen Preisen. Der Autor benutzte das Modell DT-2234C (**Bild 10**), das ihn etwa 10 € plus die Kosten für eine 9-V-Blockbatterie kostete. Bevor die Überprüfung beginnen kann, muss ein Stück reflektierendes Papier an der Motorwelle befestigt werden, wie in **Bild 11** zu sehen. Die Motordrehzahl wird mit dem Drehzahlmesser DT-2234C wie folgt überprüft:

- Lassen Sie den Motor drehen.
- Drücken Sie die *TEST*-Taste am Drehzahlmesser und richten Sie das Licht des Geräts auf das reflektierende Papier.
- Die Motordrehzahl wird kontinuierlich auf dem LC-Display angezeigt.
- Sie können die Messwerte durch Drücken der *MEM*-Taste speichern.

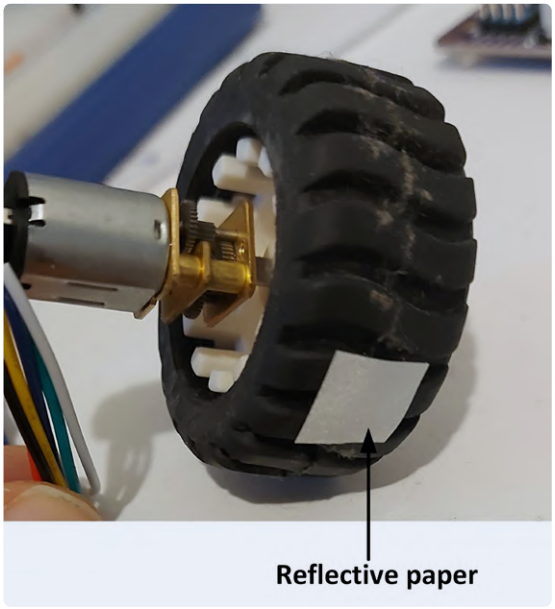


Bild 11. Auf dem Reifen wird ein Stück reflektierendes Papier angebracht.

Hinweis: In diesem Projekt wird nur eine Phase des Encoders (Phase A) verwendet. Genauere Ergebnisse für die Motordrehzahl können erzielt werden, wenn die beiden Geberphasen Phase A und Phase B verwendet werden. Es ist auch möglich, den Motor über die Schraubklemmen MOTOR 1 oder MOTOR 2 des Entwicklungsboards Maker Pi RP2040 zu betreiben. Dadurch wird die maximale Motorspannung auf +3,3 V begrenzt und die Motordrehzahl muss durch Senden von PWM-Signalen an den Motor über die Ports GP8/GP9 (MOTOR 1) beziehnungsweise die Ports GP10/GP11 (MOTOR 2) gesteuert werden.

RG – 230506-02

Haben Sie Fragen oder Kommentare?

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

Über den Autor

Dogan Ibrahim hat einen BSc (Hons) in Elektrotechnik, einen MSc in automatischer Steuerungstechnik und einen PhD in digitaler Signalverarbeitung und Mikroprozessoren. Dogan Ibrahim hat in vielen Organisationen gearbeitet und ist *Fellow der Institution of Engineering and Technology* (IET) in Großbritannien sowie zugelassener Elektroingenieur. Dogan hat über 100 Fachbücher und mehr als 200 Fachartikel über Elektronik, Mikroprozessoren, Mikrocontroller und verwandte Bereiche verfasst. Dogan ist ein zertifizierter Arduino-Experte und verfügt über langjährige Erfahrung mit fast allen Arten von Mikroprozessoren und Mikrocontrollern.

WEBLINK

[1] Motor Control Development Bundle: https://elektor.de/motor-control-development-bundle

ESP32-RS232-Adapter

Wireless-Anbindung klassischer Messgeräte

Von Georg Hofstetter (Deutschland)

RS232 goes WLAN! Ein modernes und preiswertes ESP32-Modul verleiht der seriellen Schnittstelle Flügel. Darüber hinaus bindet der Adapter altehrwürdige Mess- und Testgeräte über MQTT ins Heimnetz ein.

Wenn man über viele Jahre hinweg regelmäßig an Elektronikprojekten arbeitet, sammeln sich nach und nach einige Messgeräte an. So besitze ich beispielsweise ein SourceMeter 2400 von Keithley, das ich gerne nutze, um Fehler in elektronischen Baugruppen zu finden, diese in Betrieb zu nehmen oder sogar, um Eigenschaften von Bauteilen zu messen. Es lässt sich auch prima zum Testen von Elektrolytkondensatoren auf Leckströme oder zum Messen ihrer Nennkapazität nutzen. Auch ein Empfänger AR5000 von AOR ziert mein Büro. Diesen habe ich oft genutzt, um dem Amateurfunkgeschehen in der Welt zu folgen. Beide Geräte können, wie viele andere im Labor, dank einer RS232-Schnittstelle auch vom PC aus gesteuert werden. Es gibt zahlreiche USB-RS232-Adapter im Handel, mit denen die Anbindung an den heimischen PC eigentlich kein Problem sein sollte. Doch während mein Adapter beim AR5000 prima funktionierte, war die Kommunikation beim Keithley SourceMeter deutlich instabiler. Verlorene Zeichen beim Senden von SCPI-Befehlen sorgten immer wieder für Verbindungsabbrüche.

Das ist RS232

Der in den 1960er Jahren definierte Standard RS232 überträgt die Datenbits eines Zeichens mittels Spannungspegeln. Die elektrischen Eigenschaften dazu wurden in der V.28 definiert, die übergeordnete Funktionalität ist in V.24 geregelt. Die V.28 sieht vor, dass senderseitig je nach logischem Bitwert in einem Fenster von 5 V bis 12 V beziehungsweise -5 V bis -12 V gearbeitet wird. Empfängerseitig ist dieses Fenster breiter und erstreckt sich bis hinunter zu 3 V respektive hinauf bis -3 V, um Effekte wie Rauschen, Spannungsabfall sowie durch Leitungskapazitäten oder -induktivitäten verschliffene Flanken zu berücksichtigen.

In älteren, billigen USB-zu-Seriell-Adaptern sowie in manchen Laptops ist man senderseitig schon gefährlich nahe an die ±5 V oder ±3 V gegangen, um den Entwicklungs- und Bauteilaufwand möglichst gering zu halten. Dies hat zu den berühmt-berüchtigten Weisheiten geführt, dass Laptops und RS232-Geräte oft nicht zusammen funktionieren. Die Zeichnung in **Bild 1** aus einem TI-Dokument veranschaulicht die Spannungspegel, die anliegen müssen, um dem Standard zu entsprechen.

Ob das oben beschriebene Problem mit dem Keithley-Gerät nun an einem USB-Verlängerungskabel, an einer Erdschleife oder gar an den elektrischen Eigenschaften des verbauten Transceivers lag, war an dieser Stelle irrelevant, da ein USB-Kabel quer durch den Raum ohnehin nicht nur umständlich ist, sondern auch zum Stolpern einlädt. Daher musste eine schlanke, kompakte Wireless-Lösung her!

Bei Recherchen im Internet stößt man entweder auf fertige Module für Hutschienenmontage oder auf DIY-Aufbauten, die meist sehr auf eine bestimmte Anwendung ausgerichtet sind. Kompakte Lösungen waren keine zu finden, so dass ich selbst entwickeln durfte.

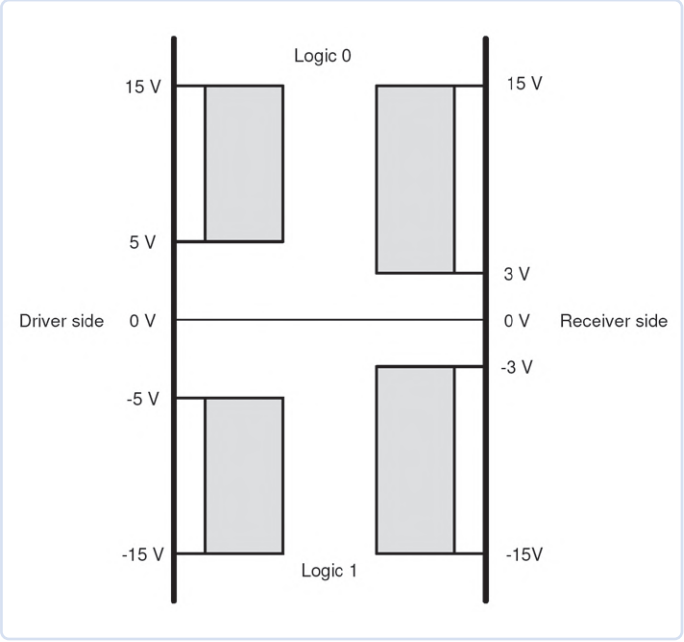


Bild 1. Erlaubte RS232-Signalpegel beim sendenden und empfangenden Gerät (Quelle: Texas Instruments [9]).

Designkriterien

Zwar ist die Lösung bewusst für uns „Maker" und nicht für Endkunden gedacht, dennoch sollten wie vor jeder Entwicklung ein paar Kriterien festgelegt werden, die das Endprodukt erfüllen soll.

Basierend auf dem ESP32

Das RS232-Gateway soll primär Daten vom und hin zum Netzwerk übertragen. Hierfür wird üblicherweise der TCP-Port 23 (Telnet) genutzt. Allerdings sollen auch anwendungsspezifische Software-Anpassungen möglich sein, über die der Nutzer selbst entscheiden darf. Für mich war ein SCPI-zu-MQTT-Service wichtig, damit das SourceMeter über Nacht seine Messwerte an Grafana (siehe unten) zur Auswertung übertragen kann.

Der ESP32 bietet genügend Rechenleistung über das Versenden einfacher TCP-Pakete hinaus, um das angeschlossene Endgerät bequem per MQTT in Home Assistant oder einen vergleichbaren Dienst einzubinden oder mittels eines eigenen Frontends über den ESP32-Webserver freizugeben.

Eine Alternative wäre der ESP8266, der etwas weniger Platz in Anspruch nimmt und ein wenig günstiger ist. Allerdings stehen dem Platzgewinn eine deutlich schwächere CPU und weniger RAM gegenüber. Außerdem fehlt dem ESP8266 ein Bluetooth-Modul, was neben der eigentlich gewünschten WLAN-Funktion noch ein Ass im Ärmel dieser Lösung wäre.

Nachteil des ESP32 ist natürlich der deutlich höhere Strombedarf, den man mit circa 500 mA auf der 3,3-V-Schiene ansetzen darf. Freilich wird man die CPU nicht permanent zum Rechnen nutzen, aber je nachdem, wie gut optimiert die Software am Ende ist, wird der Spitzenstrom schon in diese Größenordnung reichen, wie es in Tabelle 4.2 des Datenblatts angegeben ist [1]. Ein Vergleich mit dem Datenblatt des ESP8266 [2] zeigt, dass die Stromaufnahme des ESP32 im Sendebetrieb etwa 35%, im Empfangsbetrieb etwa 80% höher liegt.

Kompaktes Design

Auf den Labortischen und unter den Schreibtischen gibt es schon genug Kabelsalat. Daher sollte die Schaltung zum einen direkt am DE-9 Stecker angelötet werden und von der Länge her nicht größer sein als ein Kaltgerätestecker plus dem Biegeradius des Netzkabels. Dadurch vergrößert die Platine im eingesteckten Zustand nicht den Platzbedarf des RS232-Endgeräts. Wenn ein Gehäuse benötigt wird, sollte dies einfach zu drucken sein; idealerweise könnte die Schaltung sogar in ein handelsübliches Standard-Gehäuse passen.

Spannungsversorgung über Micro-USB

Um die Platine zu versorgen, bietet sich eine Micro-USB-Buchse an, wie sie bereits für viele Kleingeräte Standard ist. Die vom ESP32 benötigten +3,3 V erzeugen wir mit einem AMS1117 von Advanced Monolithic Systems [3] aus den +5 V vom USB-Port.

Der AMS1117 ist zwar für bis 1 A ausgelegt, verursacht aber prinzipbedingt eine Menge Abwärme auf der doch recht kleinen Platine. Eine effizientere Alternative wäre der TPS62291 (Texas Instruments, [4]), ein Step-Down-Wandler im WSON-6-Gehäuse mit 2×2 mm, der ebenso bis 1 A arbeitet und bei geschickter Anordnung auch etwas Platz sparen würde. Allerdings ist er durch das winzige Gehäuse etwas anspruchsvoller zu löten, wenn man kein Mikroskop hat oder noch etwas ungeübt ist. Beim Löten unsichere Zeitgenossen finden im Video [5] Trost und eine anschauliche Hilfe.

Zudem war die Verfügbarkeit dieses Bauteils in den letzten zwei Jahren zeitweise problematisch, wodurch die Bauteilkosten statt der üblichen 1,70 € bei deutlich über 20 € lagen (je nach Distributor ist das immer noch so). Wenn Sie im Internet Preise vergleichen, werden Sie aber auch Exemplare zu Preisen wie zu Omas Zeiten wiederfinden. Um es Ihnen nicht zu schwer zu machen, habe ich in dieser Version aber den AMS1117 verbaut, was das Handlöten dann doch einfacher gestaltet.

Spannungsversorgung über 9-pol D-Sub

Die Platine sollte alternativ aber auch über den 9-poligen D-Sub-Stecker zu versorgen sein. Erfahrene Elektroniker können, wenn sie diese Methode bevorzugen, eine kleine Anpassung am Endgerät beziehungsweise den Endgeräten vornehmen, um via RS232 die benötigten 5 V zur Verfügung zu stellen (und sich so ein zusätzliches Netzteil ersparen). Allerdings sieht der Standard keine solche Möglichkeit der Spannungsversorgung über den Stecker vor – hier müssen wir uns also außerhalb des Standards bewegen.

Da nach V.28 alle Signalleitungen Spannungen von +15 V bis -15 V führen können, liegt es nahe, eine der Signalleitungen zu verwenden. Pin 9, der im Normalfall vom Ring Indicator (RI) belegt ist, wird nur von Modems zur Signalisierung eines Anrufs verwendet. Da RS232-Modems ein wenig aus der Mode gekommen sind, wäre RI ein geeigneter Kandidat für die Spannungsversorgung.

Vorab müssen wir aber erst einmal prüfen, was passiert, wenn wir unerwartet die vollen Spannungen anlegen, weil wir beispielsweise ein Endgerät anschließen, das unsere Versorgungsleitung entsprechend V.28 bedient.

Die +15 V wären für den AMS1117 selbst kein Problem; diese Spannung liegt innerhalb der Spezifikation. Sollten jedoch -15 V anliegen, würden die internen Schutzdioden des AMS die üblicherweise 10...20 mA des RS232 ableiten. Empirische Beobachtungen zeigen,

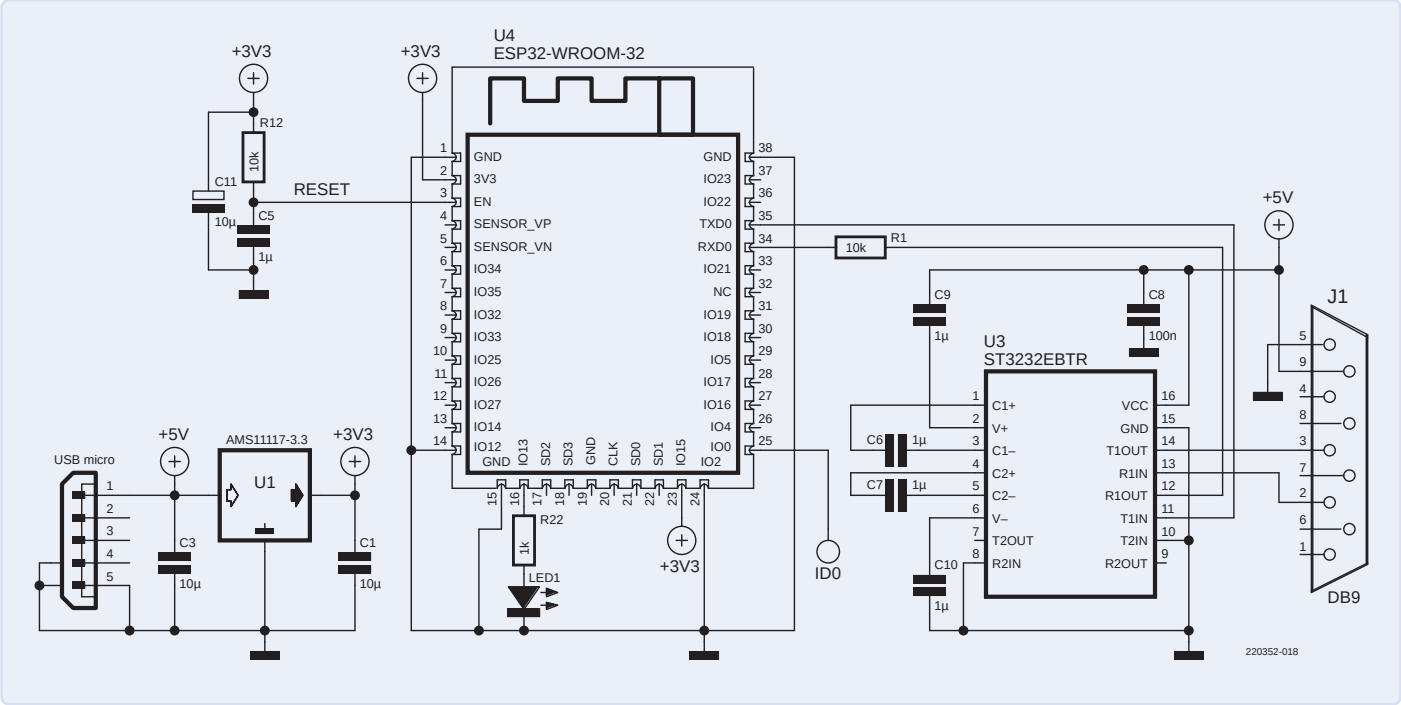


Bild 2. Die kleine vollständige Schaltung des ESP32-RS232-Dongles.

dass diese Dioden hiermit kein Problem haben. Und ein RS232-Treiber ist laut Spezifikation kurzschlussfest, so dass auch hier kein Schaden entsteht.

Allerdings hängt an den +5 V auch die VDD unseres RS232-Transceivers, der nur bis 5,5 V spezifiziert ist. Glücklicherweise fällt in diesem Fall die Spannung durch die Strombegrenzung der RS232-Leitungen auf circa 3,5 V ab. Das klingt zunächst abenteuerlich; doch diese Überlegungen haben wir ja nur für den seltenen Fall angestellt, dass der RI auf Pin 9 durch das Endgerät doch bedient wird.

True RS232

Wie eingangs angesprochen, wollen wir möglichst die V.28 erfüllen, weswegen ein Transceiver zwingend erforderlich ist. Der ST232EBTR von ST [6] ist im TSSOP-16-Gehäuse verfügbar und zudem mit circa 1 € recht preisgünstig. Durch seine internen Spannungsverdoppler- und -inverterschaltungen erzeugt er Versorgungsspannungen von +10 V und -10 V, die zum Treiben der RS232-Leitungen verwendet werden, wovon im Regelfall in etwa 9 V gemessen werden können.

Die Schaltung des ESP32-RS232-Dongles mit dem ESP32-WROOM-Modul (oder auch dem WROVER mit PSRAM) und den beiden ICs ST232EBTR und AMS1117 mit allen peripheren Bauteilen ist in **Bild 2** zu sehen.

Bestückung

Für Platinen mit einer so kurzen Stückliste hat eine Bestückung beim Hersteller von Kosten und Aufwand her gesehen wenig Sinn, ebenso wie die Herstellung von Stencils und die anschließende Bestückung mittels heimischer Hotplate. Nein, bei dieser Platine in **Bild 3** bietet sich das altmodische Handlöten förmlich an – die Bauteilanzahl ist gering, die Pads sind zugänglich.

Als erstes zu bestückendes Bauteil empfiehlt sich der ST232 (U3). Er ist relativ flach und die Pins benötigen unter Umständen etwas Nacharbeit. Danach darf das Hühnerfutter – Kondensatoren, Widerstände und LED – bestückt werden, anschließend der AMS1117 (U1) und die USB-Buchse. Als nächstes kommt der ESP32 an die Reihe und am Ende der 9-polige D-Sub-Verbinder.

Firmware

Die Grundfunktionen der Firmware sind zum einen natürlich das Routen der über das Netzwerk empfangenen Zeichen an den RS232-Port und zurück. Aber dank der Vielseitigkeit des ESP32 ist hier deutlich mehr möglich.

Die Firmware, wie sie im Repository [7] liegt, bietet aktuell zwei Arbeitsmodi – Telnet und MQTT/SCPI – die im Folgenden ausgeführt werden.

Telnet / Raw TCP-Socket

Wie schon in den Designkriterien ausgeführt, ist der Hauptzweck, die Daten vom Netzwerk auf den RS232-Port auszugeben. Bei solchen Adaptern ist es üblich, die zu übertragenden Zeichen roh über den TCP-Port 23 zu übertragen. Dieser Port wurde früher für Telnet-Verbindungen zu Servern genutzt, findet heute jedoch (abgesehen von eingebetteten Systemen) kaum noch Anwendung. Für unseren Zweck, der Übertragung von Befehlen oder Messwerten an serielle Endgeräte oder zurück, ist dies im Heimnetz absolut ausreichend.

Telnet ist ein Netzwerkprotokoll, das entwickelt wurde, um eine Fernsteuerung von Computern über ein Netzwerk zu ermöglichen. Es bietet einen textbasierten Kommunikationskanal, der es einem Nutzer erlaubt, auf einen entfernten Rechner zuzugreifen und dort Befehle auszuführen. Diese Verbindung erfolgt bidirektional, was bedeutet, dass sowohl Eingaben als auch Antworten über das Netzwerk übertragen werden. Ursprünglich im Kontext früher

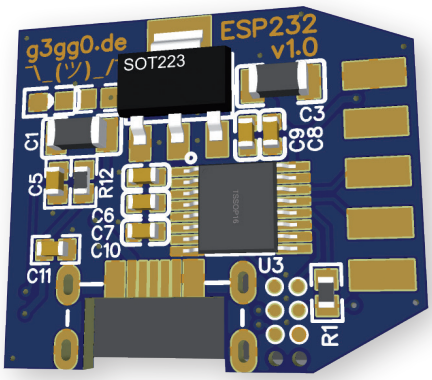


Bild 3. Die kleine Platine mit gut erreichbaren Pads ist prädestiniert für die SMD-Handbestückung.

Bild 4. Ein in Grafana erstelltes Dashboard, das Messwerte von diversen Umweltsensoren anzeigt (gehört nicht zum Projekt).

Internet-Standards entwickelt, wird Telnet häufig in Umgebungen verwendet, in denen eine einfache, textbasierte Kommunikation ausreichend ist. Allerdings gilt es als unsicher, da es keine Verschlüsselung bietet und somit alle übertragenen Daten, einschließlich Passwörter und anderer sensible Informationen, potenziell abgefangen werden können. Für die ersten Kommunkationstests kann man das Tool *telnet.exe* nutzen. Bei älteren Windows-Versionen war das Tool Teil der Standardinstallation, falls nicht bei Ihnen vorhanden, bietet es sich an, die freie und universellere Alternative PuTTY [8] zu verwenden.

MQTT / SCPI

Wenn man Messwerte über mehrere Stunden hinweg erfassen will, kann man geeignete Python-Skripte entwickeln oder gar fertige Tools auf seinem PC starten und per Telnet-Port auslesen. Wer aber bereits eine geeignete Infrastruktur in seinem Heimnetz besitzt, spart sich den stromfressenden PC und vereinfacht den Messaufbau.
In vielen Haushalten werden bereits Dienste wie MQTT, InfluxDB und Grafana zur Erfassung von Messwerten auf einem Raspberry Pi oder in Docker-Containern des heimischen NAS ausgeführt. Ohne zu weit auszuschweifen und in einfachen Worten:

- **MQTT** ist ein sogenanntes Broker-Protokoll, das von vielen Geräten unterstützt wird, um einfache Schlüssel/Wert-Paare zu publizieren.
- **InfluxDB** ist eine Datenbank, die darauf ausgelegt ist, Schlüssel/Wert-Paare, insbesondere deren zeitliche Verläufe, effizient zu speichern.
- **Grafana** liest Daten aus der Datenbank und stellt sie benutzerfreundlich mit beliebigen Zeitskalen auswertbar dar.

Wer ein solches Setup zuhause hat, für den gibt es mit dieser Firmware die Option, die vom SCPI-sprechenden Endgerät erzeugten Messwerte direkt an einen MQTT-Broker zur weiteren Auswertung zu senden.
Da sich die Befehle für die unterschiedlichen SCPI-Geräte deutlich unterscheiden, ist das einzig unterstützte Gerät momentan das Keithley Sourcemeter 2400. Dieser Code bietet aber einen guten Startpunkt für eigene Firmware-Varianten. Versierte Embedded-Entwickler sind gerne willkommen, den Funktionsumfang zu erweitern und mit der Community zu teilen.
Wie Messwert-Darstellungen in verschiedenen Anwendungen aussehen können, sieht man in **Bild 4** (Grafana), **Bild 5** (MQTT Explorer, ein umfassender MQTT-Client für verschiedene Plattformen [10]) und **Bild 6** (ebenfalls Grafana).

Um das all dies zu bewerkstelligen, muss der ESP32 sich um ein paar Dinge kümmern. Dazu braucht er Informationen, die wir per Webinterface zur Verfügung stellen müssen.

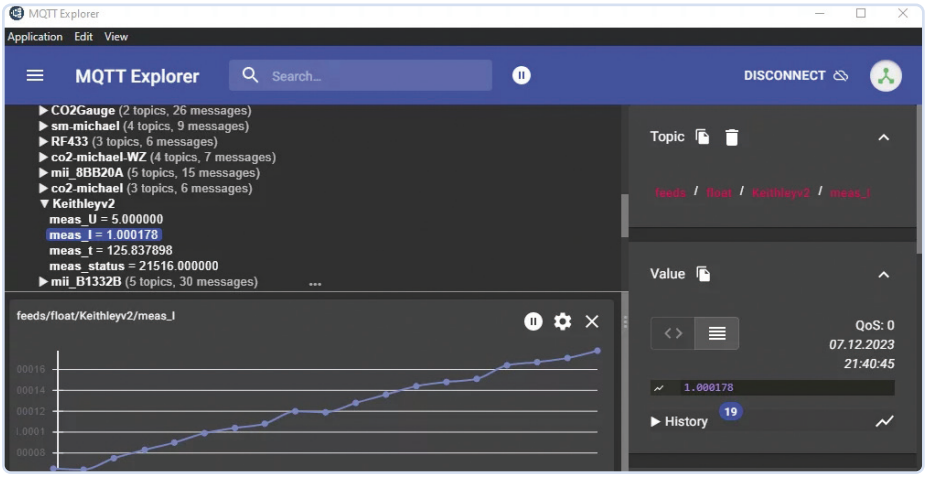


Bild 5. Der MQTT Explorer zeigt die vom ESP32-RS232-Adapter via MQTT publizierten Messwerte.

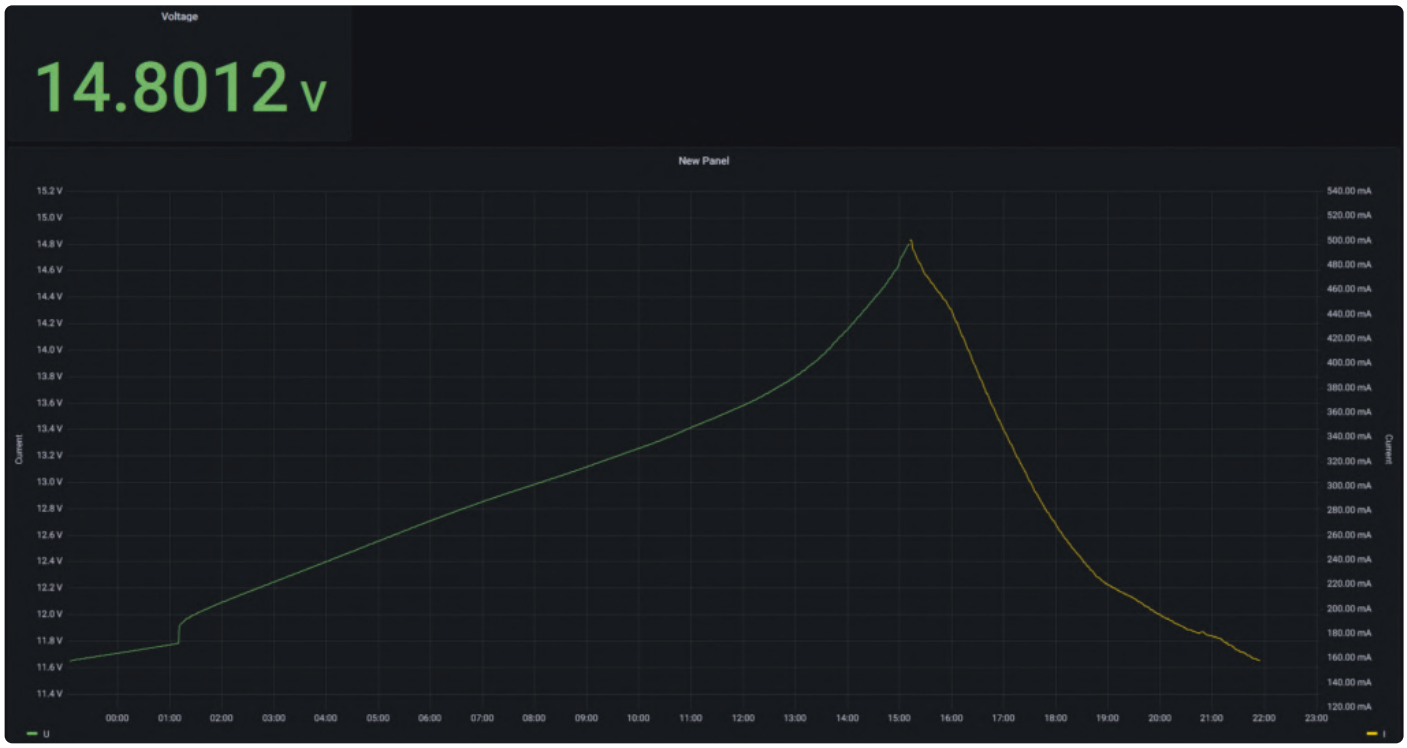


Bild 6. Spannungsverlauf (grün) und Stromverlauf (gelb) beim Laden eines Bleiakkus, gelesen von einem ESP32-RS232-Adapter.

WLAN-Setup

Wird der konfigurierte Access-Point (AP) nicht gefunden oder ist noch keiner konfiguriert, wird der ESP32 aktiv und bietet sich als AP mit dem Namen *esp232-config* an. Am besten verbindet man sich mit dem Handy mit diesem AP. Nachdem man sich angemeldet hat, kann man per Browserzugriff auf http://192.168.4.1/ die wichtigsten Parameter konfigurieren. Die im ESP32 gehostete Webseite stellt sich wie in **Bild 7** dar. **Tabelle 1** erläutert, was in die einzelnen Zeilen eingetragen werden muss.

OTA – Updates Over-The-Air

In der Arduino-IDE oder in dem von mir bevorzugten PlatformIO kann der ESP32 auch „remote" aktualisiert werden. Das unterliegende Protokoll wird durch die Komponente ArduinoOTA implementiert. Zwar erleichtert dieses Feature ein hochfrequentes Aktualisieren des Mikrocontrollers, bei der praktischen Anwendung zeigten sich aber auch wesentliche Nachteile. Durch ein ständiges Allozieren und Freigeben von Puffern bei jedem empfangenen UDP-Paket fragmentierte der Speicher des ESP32, was völlig unvorhersehbar zu Resets geführt hat. Zu allem Überfluss passierte das unabhängig davon, ob OTA-Pakete unterwegs waren. Man kann sich vorstellen, dass eine Crash-Meldung, die über den UART des ESP32 ausgegeben wird, vom angeschlossenen Endgerät selten gut aufgenommen wird. Glücklicherweise gibt es mittlerweile Verbesserungen [11], die die

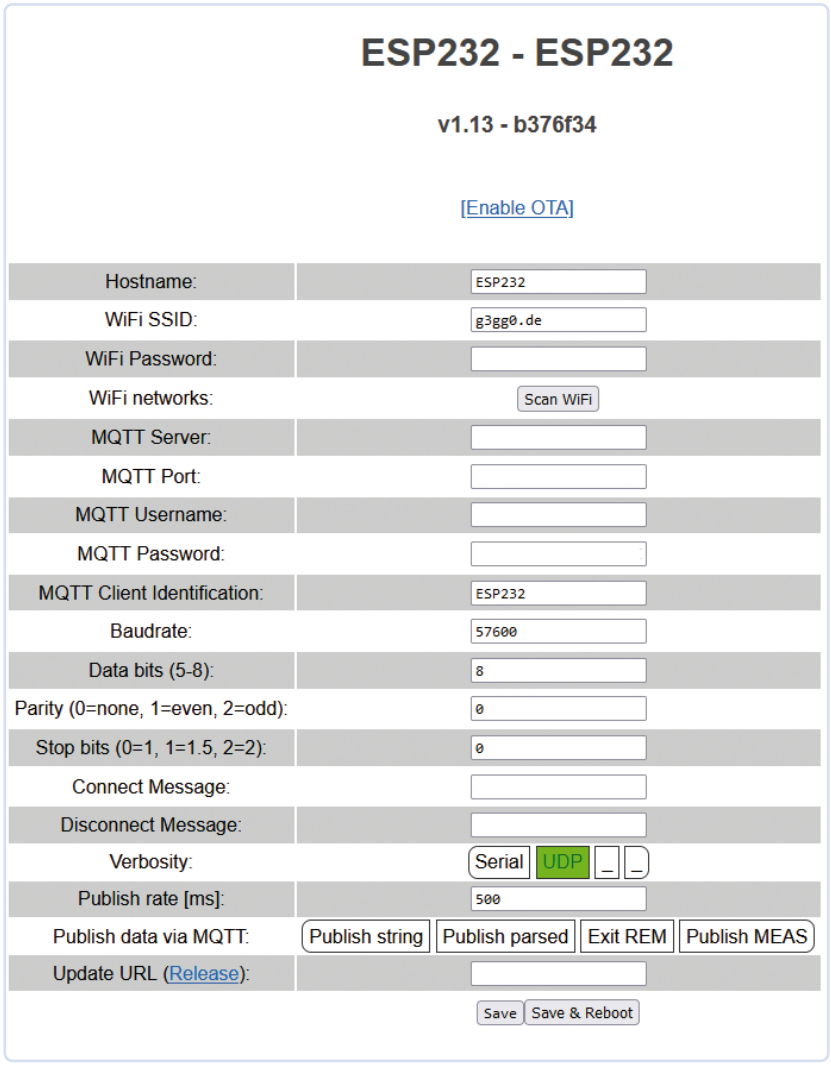


Bild 7. Das WLAN-Setup im ESP32.

Tabelle 1: Einträge im WLAN-Setup des ESP32.

Hostname	Name, mit dem sich das Gerät via MDNS im Netzwerk bekannt macht
WiFi SSID/Password	WLAN, mit dem sich das Gerät beim Start verbinden soll
WiFi-Networks	Zeigt eine Liste an gefundenen WLANs an, ein Klick übernimmt den Namen
MQTT [...]	Konfigurationsparameter für den MQTT-Client, in Verbindung mit SCPI unten
Baudrate / Databits / Stop bits	Die entsprechenden Parameter für die RS232-Kommunikation
Dis-/Connect Message	[Spezialfälle] bei TCP-Verbindung via RS232 zu sendende Nachricht
Verbosity	[Experten] Debugging der Kommunikation via UDP
Publish data via MQTT	[Experten] Für Keithley Source Meter 2400: Parsen der Displaydaten bzw. Starten einer Messung und Publizieren der Messwerte via MQTT
Update URL	Für Firmware-Update einfach die HTTP-URLs einsetzen und das hier erwartete .bin wird heruntergeladen und geflasht

Tabelle 2: Anschlussbelegung der Programmierschnittstelle.

Pin-Name	Zweck	Pegel
IO0	Festlegen des Boot-Modus, zeigt erst Effekt, wenn ein RESET ausgeführt wurde	Flashen: GND Normal: floating (3,3 V)
RESET	RESET-Pin des ESP32 (CHIP_PU)	aktiv: GND inaktiv: floating (3,3 V)
TXD0	Transmit-Leitung des ESP32	0V / 3,3 V
RXD0 (U3_34)	Receive-Leitung des ESP32	0V / 3,3 V
+5V	Versorgungspannung	5 V
GND	Masse	GND

Crashrate deutlich reduzieren. Gerade wegen dieser Unsicherheit wird das OTA-Feature immer nur kurz und auf Benutzerwunsch aktiviert, weswegen es oben im Webinterface einen Link (Enable OTA) dafür gibt.

Flashen

Wenn man nicht schon einen Pogo-Adapter für die ESP32-Module hat, muss der Mikrocontroller nach dem Bestücken mit der passenden Firmware versehen werden. Dazu sollte man den Sourcecode von [7] herunterladen und in PlatformIO importieren, kompilieren und letztendlich flashen.

Im Entwurf ist hierfür unten rechts ein Programmierport vorgesehen. Für kleine Stückzahlen reicht es, die Leitung IO0 auf GND zu ziehen und sowohl TXD0 als auch den etwas unglücklich benannten Pin U4_34 (RXD0 direkt am µC, siehe **Bild 8**) an einen USB-UART-Adapter zu löten.

Der ESP32 hat, wie **Tabelle 2** zeigt, ein paar „Strapping-Pins", die das Verhalten beim Start des ROM-Codes festlegen. In diesem Setup müssen wir uns nur um einen (IO0) kümmern. Ist die Firmware hochgeladen, sollte nach circa 30 s eine blinkende LED auf sich aufmerksam machen.

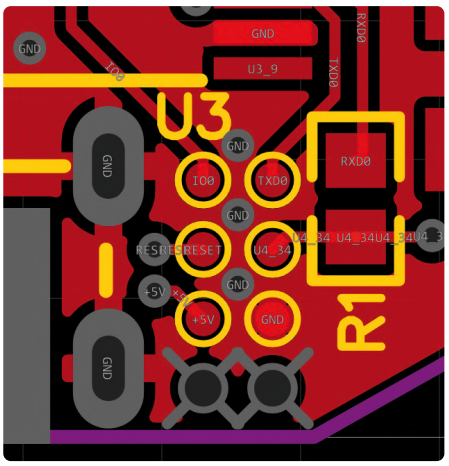


Bild 8. Programmieranschluss auf der kleinen Platine.

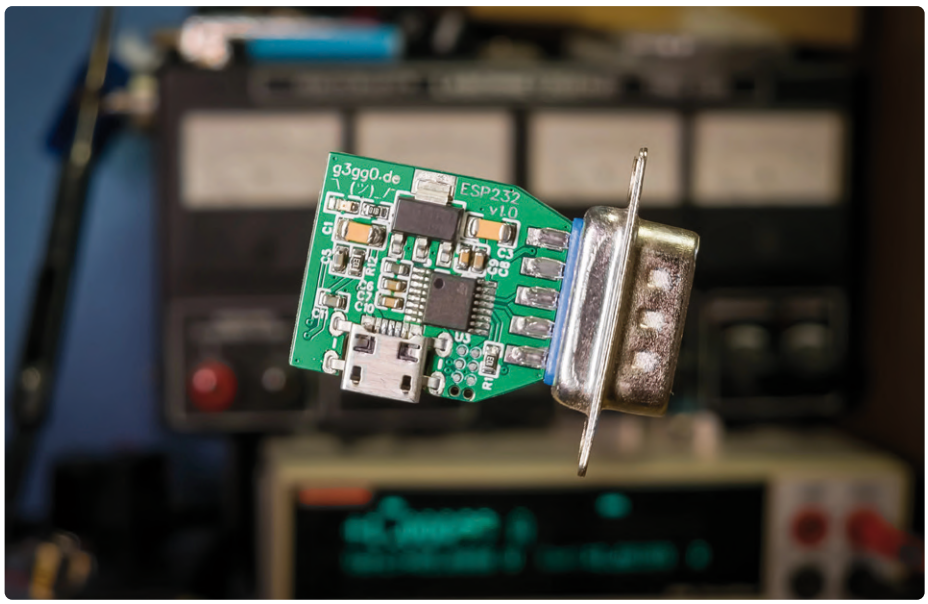
The Ugly

Der ESP32 hat sich in meinen Projekten mittlerweile sehr bewährt. Zwar gab es anfangs diverse Probleme in der verwendeten Arduino-Bibliothek hinsichtlich des UART-Puffers, weswegen der Code teilweise direkt auf die Bibliotheken des IDF-Frameworks zugreift. Auch Bugs in der ArduinoOTA-Bibliothek verursachten lange Debugging-Nächte, um dem Problem auf die Schliche zu kommen. Dadurch gibt es im Quellcode einige Stellen, an denen Workarounds zum Einsatz kamen. Da diese Firmware aber Open Source ist, wird sie weiterleben und erfreut sich an Verbesserungen, die von der Community kommen.

The Next Generation

Was mich an der Version v1.0 des Projekts bis heute stört, ist die doch recht umständliche Flashbarkeit bei der Erstinbetriebnahme (oder nach dem Flashen von fehlerhaften Entwicklungsständen). Dank eines Flashadapters mit Pogo-Pins, den ich für alle meine Projekte nutze, hält sich der Aufwand zwar in Grenzen, allerdings hat nicht jeder einen passenden Adapter parat oder hantiert gerne mit Fädeldrähten.

Aus diesem Grund geht der ESP32-RS232-Adapter in die nächste Runde und wird aktuell als Version 2.0 mit einem ESP32-S3-PICO entwickelt. Es gibt schon die ersten Prototypen, und die sehen vielversprechend aus. Der PICO ist ein komplettes Modul mit SPI-Flash und RAM, alles integriert in ein LGA-56-Gehäuse. Zwar schaut der PICO auf den ersten Blick nach einem QFN aus, die in



Über den Autor

Gestartet in der Ära des C64 und der Lochrasterplatinen hat der Autor nach einer Ausbildung zum Elektroniker ein Studium in der Informatik abgeschlossen. Seitdem entwickelt oder reverse-engineered er Software und eingebettete Systeme und veröffentlicht ausgewählte Projekte auf seiner Webseite *www.g3gg0.de*. Zu den bekannteren Projekten gehören Firmware-modifikationen für Canon DSLRs namens Magic Lantern (*www.magiclantern.fm*) oder für die Toniebox (*https://gt-blog.de/toniebox-hacking-how-to-get-started*).

der Regel gut per Hand lötbar sind, allerdings muss GND über das Exposed-Pad versorgt werden, weswegen bei Handbestückung das Pad per Bohrung verlötet werden muss. Weiter hat dieses Package keinen Anschlussrahmen und man kann die Pins nur sehr schlecht von der Seite verlöten. Letztendlich bleibt hier doch wohl nur eine (Mini-) Heizplatte als Ausweg [12].

Der große Vorteil des S3 ist, dass er über eine vollwertige USB-Peripherie verfügt, über die geflasht oder debugged werden kann. Notfalls lässt sich der ESP232 v2.0 dann auch als USB-zu-RS232-Adapter verwenden.

Ob wir damit im Kreis gehen und wieder da stehen, wo wir am Anfang losgelaufen sind, lassen wir bewusst offen. Denn wenn man es genau betrachtet, für was der ESP8266 damals gedacht war, nämlich als eine WLAN-zu-Seriell-Adapter, fragt man sich, warum es noch kein Gerät dieser Art zu kaufen gibt.

RG — 220352-02

Sie haben Fragen oder Kommentare?

Haben Sie technische Fragen oder Anregungen zu diesem Projekt? Dann kontaktieren Sie bitte den Autor unter elektor@g3gg0.de oder die Elektor-Redaktion unter redaktion@elektor.de.

Passende Produkte

- **ESP32-WROOM-32**
  www.elektor.de/18421
- **ESP32-S2-WROVER**
  www.elektor.de/19693

WEBLINKS

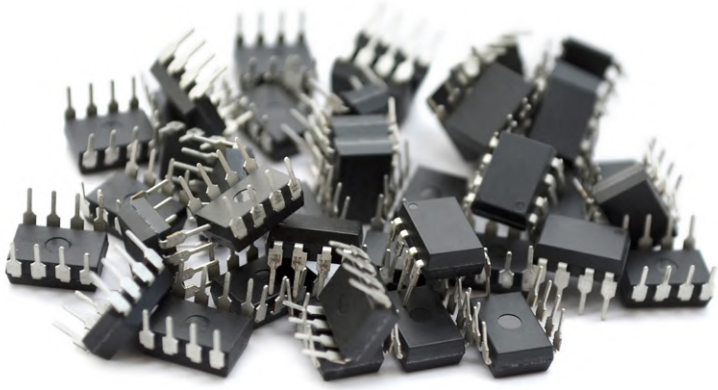
[1] Datenblatt ESP32: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
[2] Datenblatt ESP8266: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf
[3] AMS1117: http://www.advanced-monolithic.com/pdf/ds1117.pdf
[4] TPS62291: https://www.ti.com/product/de-de/TPS62291
[5] Händisches SMD-Löten im Elektor-Video: https://www.elektormagazine.de/news/uberzeugendes-video-loten-von-smd-bauteilen
[6] ST232: https://www.st.com/resource/en/datasheet/st202eb.pdf
[7] Projekt-Repository: https://github.com/g3gg0/ESP232
[8] PuTTY: https://www.putty.org/
[9] MQTT Explorer: https://mqtt-explorer.com/
[10] WiFiUDP-Crash: https://github.com/espressif/arduino-esp32/issues/4104
[11] Texas Instruments Application Report: RS-232 Frequently Asked Questions: https://www.ti.com/lit/an/slla544/slla544.pdf
[12] Massimo Divito, „Mini-Reflow-Platte", Elektor 11-12/2023: https://www.elektormagazine.de/230456-02

Aller Anfang...

...muss nicht schwer sein: Mehr über Operationsverstärker

Von Eric Bogers (Elektor)

In der letzten Folge haben wir einen Blick auf diskrete FETs geworfen und mit einer Diskussion über das wahrscheinlich wichtigste Bauteil in der analogen Elektronik begonnen: den Operationsverstärker. Es gibt eine ganze Menge über Operationsverstärker zu sagen, wie Sie sehen werden, und es gibt eine Menge Formeln, die wir im Zusammenhang mit Operationsverstärkern verwenden können (und werden).



Der Operationsverstärker als Black Box

Wenn Sie einen Operationsverstärker in einer praktischen Anwendung einsetzen wollen, interessiert es Sie als Neuling in der Elektronik kaum, wie er von innen aussieht. Tatsächlich braucht man nur einige wenige wichtige Parameter, um einen Operationsverstärker zu benutzen. Das heißt, Sie können Operationsverstärker als eine Art Black Box mit Eingängen und Ausgängen betrachten (**Bild 1**). Das Wichtigste, worauf Sie achten müssen, ist die Leerlaufverstärkung, die in der Regel zwischen 10.000 und 100.000 liegt. Anhand dieses Parameters können Sie die Ausgangsspannung mit der folgenden Formel berechnen, wobei U_{ni} die Spannung am nicht-invertierenden Eingang und U_i die Spannung am invertierenden Eingang ist:

$$U_{out} = V_0 \cdot (U_{ni} - U_i)$$

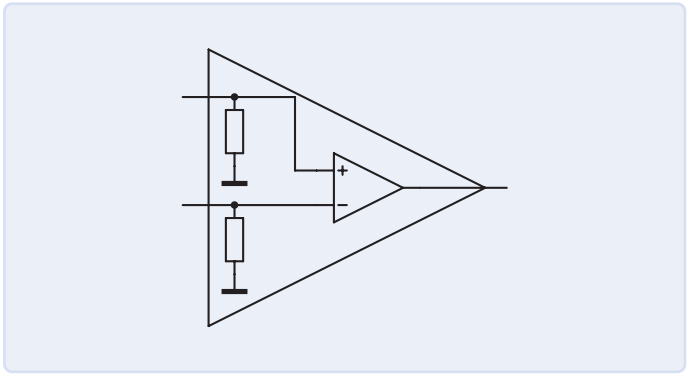


Bild 1. Der Operationsverstärker als Black Box.

Daraus könnte man den Eindruck gewinnen, dass der genaue Wert der offenen Schleifenverstärkung (das heißt die Verstärkung ohne Gegenkopplung) von entscheidender Bedeutung ist. Dies ist jedoch nicht der Fall, wie Sie gleich sehen werden.

Bei Operationsverstärkern aus bipolaren Transistoren liegt die Eingangsimpedanz in der Größenordnung von 1 MΩ. Werden in der Eingangsstufe des Operationsverstärkers Feldeffekttransistoren (FETs) verwendet, so liegt die Eingangsimpedanz um mehrere Zehnerpotenzen höher.

Die maximale Versorgungsspannung liegt in der Regel zwischen ±16 V und ±22 V, kann aber bei Hochvolt-Opamps bis zu ±150 V betragen. Der maximale Ausgangsstrom liegt normalerweise in der Größenordnung von 20 mA, obwohl Leistungs-Operationsverstärker mehrere Ampere verkraften können, wenn sie ausreichend gekühlt werden. Auf die Grenzfrequenz wird später noch näher eingegangen.

Die beiden gebräuchlichsten Schaltungen mit Operationsverstärkern sind der invertierende und der nichtinvertierende Verstärker. Wie der Name schon andeutet, kehrt der invertierende Verstärker die Polarität des Eingangssignals um, während der nichtinvertierende Verstärker dies nicht tut.

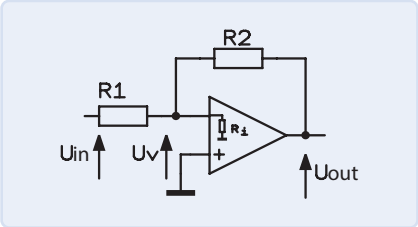


Bild 2. Der invertierende Verstärker.

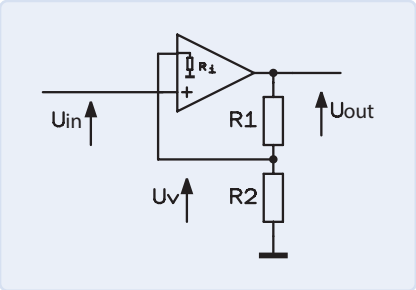


Bild 3. Der nicht-invertierende Verstärker.

Der invertierende Verstärker

Bild 2 zeigt das grundlegende Schaltbild des invertierenden Verstärkers. Er benötigt nur zwei externe Komponenten in Form von zwei Widerständen. In Audioanwendungen wird in der Regel noch ein Kondensatorpaar hinzugefügt.
Wie viel Verstärkung bietet diese Schaltung? Der nicht-invertierende Eingang ist mit Masse verbunden, so dass die offene Schleifenverstärkung der erste Parameter ist, der die Verstärkung bestimmt:

$$V_0 = -\frac{U_{out}}{U_V} \quad \Rightarrow U_V = -\frac{U_{out}}{V_0}$$

Für die Eingangsspannung gilt:

$$U_{in} = U_{R1} + U_V = I_{R1} \cdot R_1 - \frac{U_{out}}{V_0}$$

Die Verstärkung ist definiert als die Ausgangsspannung geteilt durch die Eingangsspannung. Setzt man das oben ermittelte Ergebnis in diese Formel ein, erhält man:

$$V = \frac{U_{out}}{U_{in}} = \frac{U_{out}}{I_{R1} \cdot R_1 - \frac{U_{out}}{V_0}}$$

Die folgende Formel gilt für die Ströme, die in den und aus dem Knoten vor dem invertierenden Eingang fließen:

$$I_{R1} = I_{R2} + I_{Ri} = I_{R2} + \frac{U_V}{R_i} = I_{R2} - \frac{U_{out}}{R_i \cdot V_0}$$

Setzt man dies in die Formel für die Verstärkung ein, erhält man:

$$V = \frac{U_{out}}{U_{in}} = \frac{U_{out}}{I_{R2} \cdot R_1 - \frac{U_{out} \cdot R_1}{R_i \cdot V_0} - \frac{U_{out}}{V_0}}$$

Für das Netzwerk mit dem Widerstand R_2 gilt die folgende Formel:

$$U_{out} = U_V - U_{R2} = -\frac{U_{out}}{V_0} - I_{R2} \cdot R_2$$

Diese kann für den Strom I_{R2} aufgelöst werden:

$$I_{R2} = \frac{U_{out}}{V_0 \cdot R_2} - \frac{U_{out}}{R_2}$$

Setzt man dies schließlich in die Formel für die Verstärkung ein und löst U_{out} überall dort auf, wo es auftaucht, erhält man die genaue Formel für die Verstärkung des invertierenden Verstärkers:

$$V = \frac{1}{-\frac{R_1}{V_0 \cdot R_2} - \frac{R_1}{R_2} - \frac{R_1}{R_i \cdot V_0} - \frac{1}{V_0}}$$

Jetzt wird es interessant. Man kann nämlich wie folgt argumentieren: Die Open-Loop-Verstärkung ist normalerweise sehr viel größer als das Verhältnis R_2/R_1, während die interne Impedanz des Operationsverstärkers ebenfalls sehr viel größer ist als die Werte der externen Widerstände. Vor diesem Hintergrund kann man eine gute Annäherung für die Verstärkung erhalten, und zwar in vereinfachter Form als:

$$V \approx -\frac{R_2}{R_1}$$

Dies ist eine sehr schöne und praktische Formel zur Berechnung der Verstärkung des invertierenden Operationsverstärkers. Anhand eines Zahlenbeispiels wollen wir sehen, wie groß der Unterschied in der Praxis ist. Nehmen wir eine Verstärkerschaltung mit $R2 = 100\ k\Omega$ und $R1 = 10\ k\Omega$ an und verwenden die vereinfachte Formel, so ergibt sich eine Verstärkung von -10. Nach der zuvor abgeleiteten exakten Formel (unter der Annahme einer offenen Schleifenverstärkung von 100.000 und einer Eingangsimpedanz von 1 MΩ) ergibt sich eine Verstärkung von etwa -9,99889 (wenn Sie daran interessiert sind, können Sie die Berechnung selbst durchführen). Das bedeutet, dass die Differenz um mehrere Zehnerpotenzen kleiner ist als die Toleranz selbst von Präzisionswiderständen. Man sieht, der Unterschied kann leicht vernachlässigt werden! Übrigens haben wir eine Reihe anderer Faktoren wie die Ausgangsimpedanz, die Temperaturdrift und so weiter völlig außer Acht gelassen.

Der nicht-invertierende Verstärker

Bild 3 zeigt das grundlegende Schaltbild des nicht-invertierenden Verstärkers. In dieser minimalistischen Version benötigt man auch hier nur zwei externe Widerstände. Für die Ausgangsspannung eines Operationsverstärkers gilt die folgende Formel:

$$U_{out} = (U_{in} - U_V) \cdot V_0 \quad \Rightarrow \quad (U_{in} - U_V) = \frac{U_{out}}{V_0}$$

Die Verstärkung der gesamten Schaltung kann wie folgt ausgedrückt werden:

$$V = \frac{U_{out}}{U_{in}} = \frac{U_{out}}{(U_{in} - U_V) + U_V}$$

Im Laufe der Herleitung wird klar, warum wir den scheinbar bedeutungslosen Ausdruck U_V in die obige Formel aufgenommen haben. Wenn man diese Formel umdreht und die Formel für die Ausgangsspannung des Operationsverstärkers einsetzt, erhält man Folgendes (wir haben hier einige Schritte weggelassen haben, um diese Erklärung nicht zu lang werden zu lassen):

$$\frac{1}{V} = \frac{U_{out}}{V_0 \cdot U_{out}} + \frac{U_V}{U_{out}} = \frac{1}{V_0} + \frac{U_V}{U_{out}}$$

Hier bilden R_1 und R_2 einen Spannungsteiler, der durch den Widerstand R_i belastet ist. Für U_V bedeutet dies:

$$U_V = \frac{U_{out} \cdot (R_2 \parallel R_1)}{R_i + (R_2 \parallel R_1)} = \frac{U_{out}}{R_1 \cdot \left(\frac{1}{R_2} + \frac{1}{R_i}\right) + 1}$$

Setzt man dies in die (umgekehrte) Formel für die Verstärkung ein, erhält man:

$$\frac{1}{V} = \frac{1}{V_0} + \frac{1}{R_1 \cdot \left(\frac{1}{R_2} + \frac{1}{R_i}\right) + 1}$$

Durch Umkehrung dieses Ergebnisses erhält man die genaue Formel für die Verstärkung:

$$V = \frac{1}{\frac{1}{V_0} + \frac{1}{R_1 \cdot \left(\frac{1}{R_2} + \frac{1}{R_i}\right) + 1}}$$

Auch in diesem Fall ist zu bedenken, dass die Leerlaufverstärkung viel größer ist als das Verhältnis R_2/R_1, und außerdem ist R_i sehr viel größer als R_2. Wendet man diese Überlegungen auf die obige Formel an, erhält man schließlich die vereinfachte Formel für die Verstärkung des nicht-invertierenden Operationsverstärkers:

$$V \approx \frac{R_1}{R_2} + 1$$

Auch dies ist eine sehr schöne, kompakte Formel, für deren Anwendung man oft nicht einmal einen Taschenrechner benötigt.
In der nächsten Folge werden wir etwas tiefer in die Verstärkertheorie einsteigen, und danach werden wir uns endlich echten Verstärkerschaltungen zuwenden.

RG — 230756-02

Anmerkung der Redaktion: Diese Artikelserie „Aller Anfang...“ basiert auf dem Buch „Basiskurs Elektronik“ von Michael Ebner, das auf Deutsch und Niederländisch bei Elektor erschienen ist.

Haben Sie Fragen oder Kommentare?

Wenn Sie technische Fragen oder Kommentare zu diesem Artikel haben, wenden Sie sich bitte an die Elektor-Redaktion unter redaktion@elektor.de.

Passendes Produkt

- **B. Kainka, *Elektronik-Grundlagen und Einsteigerprojekte* (Elektor, 2019)**
  Buch, kartoniert, deutsch: www.elektor.de/19035
  E-Buch, PDF, deutsch: www.elektor.com/19036

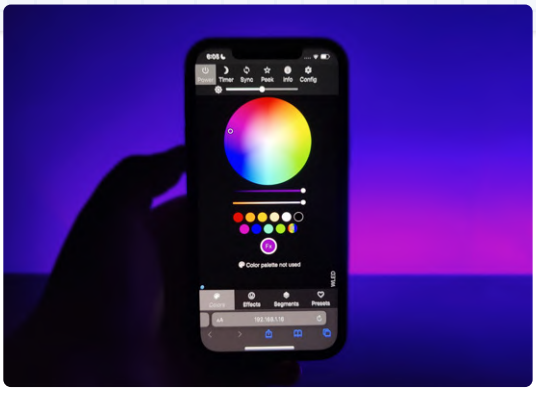
Empfehlungen für ESP-Bibliotheken

Von Saad Imtiaz und Jean-François Simon (Elektor)

Sie suchen nach Bibliotheken für ESP? Schauen Sie sich diese Empfehlungen des Entwickler-Teams von Elektor an.

WLED (github.com/Aircoookie)

WLED von Christian Schwinne ist eine schnelle und funktionsreiche Implementierung eines ESP8266/ESP32-Webservers zur Steuerung von NeoPixels-LEDs (WS2812B, WS2811, SK6812). Die Bibliothek unterstützt auch SPI-basierte Chipsätze wie den WS2801, APA102 und viele andere. Sie bietet über 100 Spezialeffekte für NeoPixels, 50 Paletten, FastLED-Rauscheffekte und vieles mehr! WLED hat eine moderne Benutzeroberfläche mit ebenso modernen Steuerelementen. Die Konfiguration erfolgt über das Netzwerk. Zu den bemerkenswerten Vorteilen gehört, dass es bis zu zehn LED-Ausgänge pro Instanz unterstützt und mit RGBW-Streifen arbeiten kann. Es lassen sich bis zu 250 Benutzervoreinstellungen zum einfachen Speichern und Laden von Farben/Effekten verwenden. Auch das Wechseln zwischen den Einstellungen wird unterstützt. Voreinstellungen lassen sich auch verwenden, um API-Aufrufe automatisch auszuführen. Es gibt zudem eine Nightlight-Funktion (Nachtlicht), die die LEDs allmählich abdunkelt. Updates können Sie Over the Air (HTTP + ArduinoOTA) einspielen und ein Passwortschutz ist ebenfalls möglich. Wenn Sie Ihre RGB-Beleuchtung steuern oder Ihrem Zimmer in einem neuen Thema illuminieren möchten, ist diese Bibliothek genau richtig für Sie! Schauen Sie auch mal unten auf *Adafruit_NeoPixel*.
https://github.com/Aircoookie/WLED

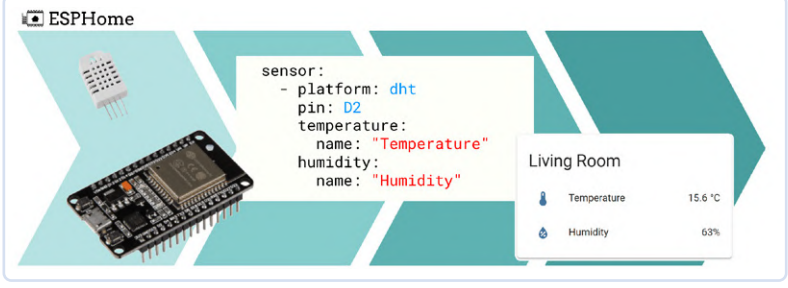


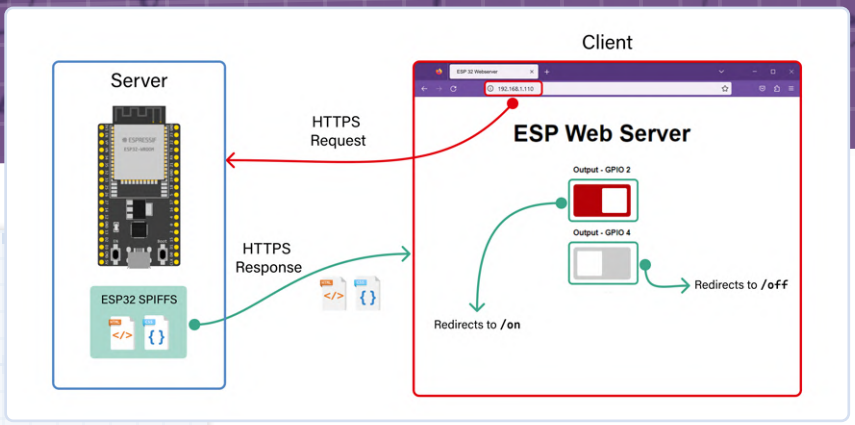
ExpressLRS (github.com/ExpressLRS)

ExpressLRS ist eine Open-Source-Funkverbindung für RC-Anwendungen (Radio Control). Sie wurde entwickelt, um mit dem LoRa-Chip SX127x/SX1280 in Kombination mit einem Espressif- oder STM32-Mikrocontroller eine hervorragende Leistung zu erzielen. Mit ExpressLRS profitieren Sie von einer guten Reichweite und einer sehr geringen Latenzzeit. Das liegt an der LoRa-Modulation sowie einer reduzierten Paketgröße. ExpressLRS unterstützt Hardware von vielen Herstellern: AxisFlying, BETAFPV, Flywoo, FrSky, HappyModel, HiYounger, HGLRC, ImmersionRC, iFlight, JHEMCU, Jumper, Matek, NamimnoRC, QuadKopters und SIYI. Die Bibliothek bietet Paketrate mit 1 kHz, Telemetrie, WLAN-Updates, zwei Frequenzen für die RC-Verbindung (2,4 GHz oder 900 MHz) und mehr. Es ist zweifellos sehr interessant für viele RC-Projekte, mit unterschiedlicher Hardware und unterschiedlichen Anforderungen.
https://github.com/ExpressLRS/ExpressLRS

ESPHome (github.com/esphome)

ESPHome ist ein Open-Source-Framework, das die Konfiguration und Verwaltung von Geräten auf Basis von ESP8266 und ESP32 vereinfacht. Es ermöglicht Anwendern, benutzerdefinierte Firmware für diese Geräte zu erstellen, ohne dass eine umfangreiche Programmierung erforderlich wäre. Mit ESPHome können Sie Gerätefunktionen und -einstellungen mit einer benutzerfreundlichen YAML-Konfigurationsdatei definieren, sodass es sowohl für Anfänger als auch für erfahrene Entwickler zugänglich ist. Zu den wichtigsten Funktionen von ESPHome gehören die Unterstützung einer breiten Palette von Sensoren und Komponenten sowie die automatische Erkennung und Integration mit beliebten Hausautomatisierungsplattformen wie Home Assistant. Auch Over-the-Air-Updates (OTA) für nahtlose Firmware-Upgrades sind möglich. ESPHome fördert die Erstellung von Smart-Home-Lösungen zum Selbermachen und ermöglicht es den Nutzern, ihre Geräte auf bestimmte Bedürfnisse anzupassen. Damit sind etwa Temperaturüberwachung, Lichtsteuerung oder Haussicherheit gemeint. ESPHome hat in der Smart-Home-Community an Popularität gewonnen. Schauen Sie es sich unbedingt an, wenn Sie es bisher nicht kennen!
https://github.com/esphome/esphome



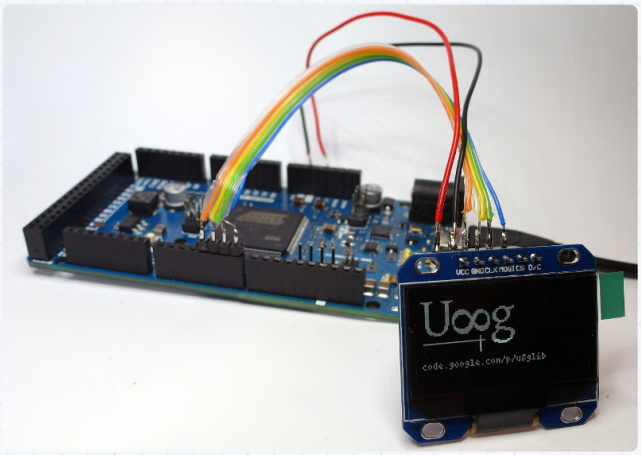


ESPAsyncWebServer (github.com/me-no-dev)

ESPAsyncWebServer ist ein asynchroner HTTP- und WebSocket-Server für ESP8266, der in der Arduino-Welt verwendet wird. Er lässt sich in PlatformIO und der Arduino-IDE einsetzen. Die Verwendung eines asynchronen Netzwerks bedeutet, dass Sie mehr als eine Verbindung gleichzeitig verarbeiten können. Sie werden aufgerufen, sobald die Anfrage fertig ist und geparst wurde. Senden Sie die Antwort, lassen sich sofort andere Verbindungen verarbeiten, während sich der Server im Hintergrund um das Senden der Antwort kümmert. Die Verarbeitungsgeschwindigkeit ist sehr hoch! ESPAsyncWebserver bietet eine benutzerfreundliche API und ist kompatibel mit *HTTP Basic* und *Digest MD5 Authentication* (als Standard) sowie *Chunked Response*. Es gibt verschiedene Plug-ins, die verschiedene Vorteile bieten, zum Beispiel verschiedene Standorte, Senden von Ereignissen an Browser, erweitertes URL-Rewrite und mehr.
https://github.com/me-no-dev/ESPAsyncWebServer

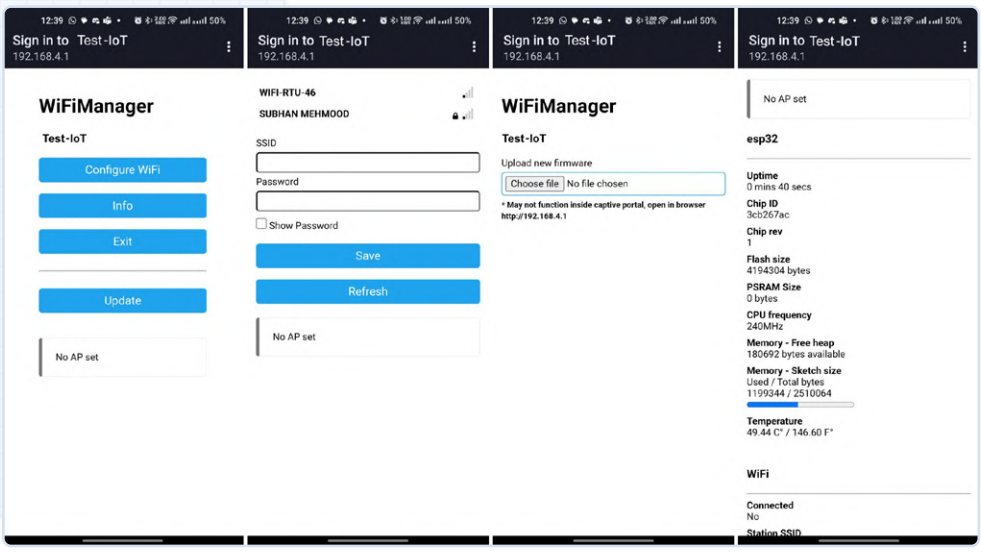
U8glib (github.com/olikraus)

U8glib ist eine Grafikbibliothek, die viele monochrome Displays unterstützt. Die neuste Version von U8glib für Arduino ist im Library Manager verfügbar und kann auch von GitHub heruntergeladen werden. Sie ist kompatibel mit Arduino (ATmega und ARM), AVR, ARM (mit Beispiel für LPC1114) und vielen anderen Geräten wie SSD1325, ST7565, ST7920, UC1608, UC1610, UC1701, PCD8544, PCF8812, KS0108 und mehr. Die Bibliothek unterstützt viele Schriftarten (sowohl Monospace-Schrift als auch proportional), Mauszeigermodus, Quer- und Hochformat ... Alles in allem eine ziemlich umfassende Bibliothek, die sich für viele Projekte als nützlich erweisen dürfte.
https://github.com/olikraus/u8glib



WifiManager (github.com/tzapu)

WiFiManager ist ein Verbindungsmanager für ESP-basierte Projekte. Damit haben Nutzer die Möglichkeit, WLAN-Anmeldedaten und benutzerdefinierte Parameter zur Laufzeit über ein Captive-Portal zu konfigurieren. Wie es funktioniert? Sobald Ihr ESP startet, wird er in den Station-Modus versetzt und versucht, eine Verbindung zu einem zuvor gespeicherten Access Point herzustellen. Gelingt das nicht, versetzt die Firmware den ESP in den Access-Point-Modus und richtet einen DNS- sowie Webserver ein. Dann können Sie mit einem beliebigen WLAN-fähigen Gerät mit einem Browser (Computer, Telefon, Tablet) eine Verbindung zu dem neu erstellten Access Point etablieren und die Zugangsdaten festlegen. Der ESP wird sich dann mit dem Netzwerk Ihrer Wahl verbinden! Sie können dieses Verhalten auch ändern oder das Konfigurationsportal und das Webportal unabhängig voneinander manuell starten. Zudem können Sie sie den ESP im Non-Blocking-Modus laufen lassen.
https://github.com/tzapu/WiFiManager

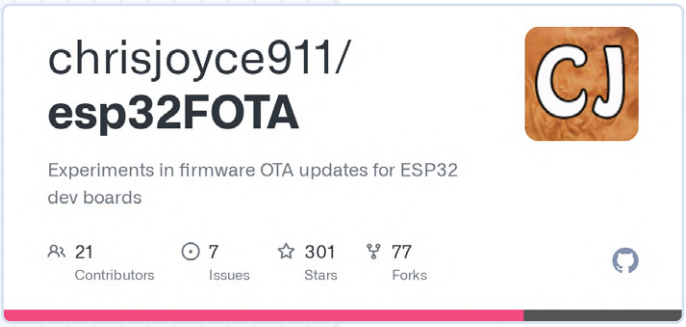


Tasmota (github.com/arendst)

Tasmota ist eine Open-Source-Firmware für ESP-Geräte, die speziell für die Heimautomatisierung und Smart Homes entwickelt wurde. Sie bietet Unterstützung für die Mikrocontroller ESP8266, ESP32, ESP32-S oder ESP32-C3. Theo Arends initiierte das Projekt im Jahr 2016 unter dem Namen Sonoff-MQTT-OTA. Sein Hauptziel war es, ESP8266-basierte Geräte, die von ITEAD (Sonoff) hergestellt wurden, mit MQTT- und Over-the-Air-Firmware-Funktionen (OTA) auszustatten. Was als einfache Lösung begann, um ein Cloud-gebundenes Sonoff-Basisgerät in ein lokal verwaltbares Gerät umzumodeln, hat sich zu einem vollständigen Ökosystem entwickelt. Es eignet sich für fast jedes ESP-basierte Gerät. Tasmota wurde für PlatformIO entwickelt und ermöglicht eine einfache Konfiguration über eine webbasierte Benutzeroberfläche (webUI). Updates lassen sich drahtlos durchführen. Benutzer können Geräte mithilfe von Timern oder benutzerdefinierten Regeln automatisieren. Vollständige lokale Kontrolle und Erweiterbarkeit sind über MQTT-, HTTP-, serielle oder KNX-Protokolle umsetzbar.
https://github.com/arendst/Tasmota

Esp32FOTA (github.com/chrisjoyce911)

esp32FOTA ist eine einfache Bibliothek, mit der Sie Ihrem ESP32-Projekt Unterstützung für OTA-Updates (Over-the-Air) spendieren können. Die Bibliothek versucht, auf eine JSON-Datei zuzugreifen, die auf einem Webserver gehostet wird. Sie analysiert den Inhalt und stellt damit fest, ob es eine neuere Firmware-Version gibt. Ist das der Fall, lädt sie sie herunter und installiert sie. Damit dieser Aktualisierungsprozess funktioniert, benötigen Sie einen Webserver mit einer gültigen JSON-Datei (die optional mit zlib oder gzip komprimiert werden kann). Die vollständige Liste der detaillierten Anforderungen (einschließlich Details zu HTTPS) finden Sie auf GitHub. Die Bibliothek ist recht umfangreich und bietet Unterstützung für komprimierte Firmware, SPIFFS/LittleFS-Partitionsupdates, Kompatibilität mit verschiedenen Dateisystemen für die Speicherung von Zertifikaten sowie Signaturen. Außerdem ermöglicht esp32FOTA webbasierte Updates (mit einem Webserver), Batch-Firmware-Synchronisation und die Möglichkeit, Firmware-Updates zu erzwingen. Schließlich verfügt die Bibliothek auch über einige Luxusfunktionen wie Signaturprüfungen für heruntergeladene Firmware-Images, Signaturüberprüfung und Unterstützung für semantische Versionierung. Kurz gesagt, es ist eine Bibliothek, die mehr als einen Blick wert ist, wenn Sie OTA-Updates für Ihr nächstes Projekt einsetzen möchten.
https://github.com/chrisjoyce911/esp32FOTA

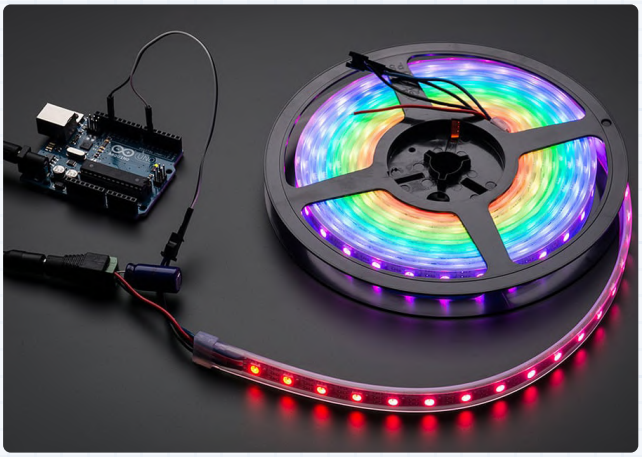


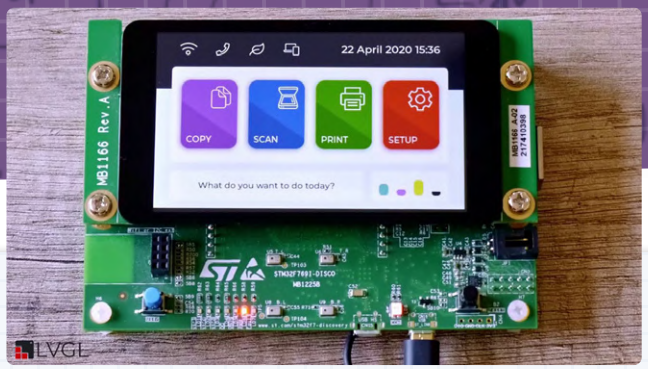
Willow (github.com/toverainc)

Willow ist ein Espressif-IDF (IoT Development Framework), womit Sie eine ESP32-S3-BOX in einen vielseitigen Sprachassistenten mit verschiedenen Funktionen verwandeln können. Er lässt sich durch benutzerdefinierte Weckwörter wie „Hi ESP“ oder „Alexa“ starten und hört auf Sprachbefehle. Dabei erkennt er automatisch, wann er zuhören soll und wann nicht. Willow lässt sich gut in gängige Heimautomatisierungsplattformen wie Home Assistant, openHAB und generische REST-APIs integrieren. Der Sprachassistent bietet auch in schwierigen Umgebungen eine hervorragende Leistung. Er erkennt gesprochene Wörter und Sprache aus einer Entfernung bis etwa 7,5 m. Mit Funktionen wie automatischer Verstärkungsregelung, Echounterdrückung, Rauschunterdrückung und Quellentrennung sorgt er für eine hohe Audioqualität. In Umgebungen mit hoher WLAN-Dichte kann Willow eine Audiokompression einsetzen, um die Nutzung der Sendezeit zu optimieren. Zudem bietet Willow eine geräteeigene Sprachbefehlserkennung, womit Sie bis zu 400 Befehle lokal konfigurieren dürfen. Alternativ können Sie den quelloffenen Willow Inference Server für umfangreichere Sprachtranskriptionsfunktionen einsetzen. Es handelt sich tatsächlich um ein sehr beeindruckendes Projekt!
https://github.com/toverainc/willow

Adafruit_NeoPixel (github.com/adafruit)

Dies ist eine Arduino-Bibliothek zur Steuerung von, nun ja, NeoPixeln. Dabei handelt es sich, im Jargon von Adafruit, um individuell adressierbare RGB-Farbpixel und -streifen, die auf den LEDs WS2812, WS2811 und SK6812 basieren. Diese LEDs besitzen integrierte Treiber und verwenden ein Ein-Draht-Steuerprotokoll. Aufgrund der engen Timing-Anforderungen für die Steuersignale und des spezifischen Protokolls ist die Nutzung normalerweise etwas kompliziert. Doch mit dieser Bibliothek wird alles viel einfacher! Adafruit entwickelt und pflegt sie kostenlos. Im Gegenzug können die Benutzer einige ihrer Produkte kaufen. Die Hauptziele von Adafruit_NeoPixel sind die benutzerfreundliche Handhabung und die Flexibilität bezüglich der unterstützten Chipsätze. Die Bibliothek unterstützt AVRs (ATmega und ATtiny), Teensy, Arduino Due, Arduino 101, ATSAMD21/51, Adafruit STM32 Feather, ESP8266, ESP32, Nordic nRF51/52 sowie einige ICs der XMC-Serie von Infineon. Unter gibt es vielen erprobte Funktionen wie `setBrightness()`, `setPixelColor()` und zwölf weitere. Damit ist die Bibliothek für schnelles Prototyping geeignet.
https://github.com/adafruit/Adafruit_NeoPixel



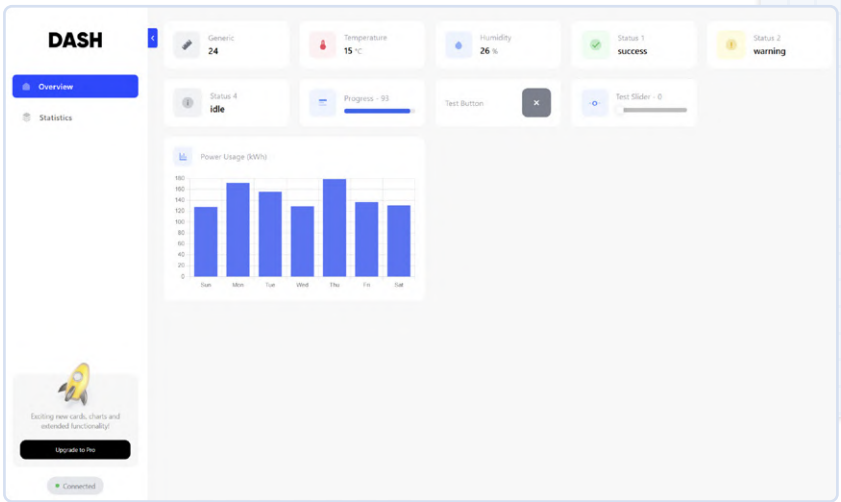


LVGL (github.com/lvgl)

LVGL ist eine eingebettete Grafikbibliothek, mit der Sie grafische Benutzeroberflächen für eine breite Palette von Mikrocontrollern erstellen können. Dazu gehören natürlich auch ESP32 und Arduino; eine vollständige Liste finden Sie auf GitHub). Die Bibliothek wurde in C geschrieben und ist portabel, also nicht auf externe Abhängigkeiten angewiesen. Sie lässt sich mit oder ohne Echtzeit-Betriebssystem (RTOS) verwenden und verfügt über eine umfangreiche Display-Unterstützung, Monochrom, ePaper, OLED und mehr. Das Projekt benötigt 32 Kbyte RAM und 128 Kbyte Flash-Speicher. LVGL bietet eine breite Palette an eingebauten Widgets, Stilen, Layouts und mehr und ist so stark anpassbar. Animationen, Anti-Aliasing, Deckkraftkontrolle, sanftes Scrollen, Schatten, Bildtransformation … die Liste ließe sich fortsetzen. Die Bibliothek unterstützt auch verschiedene Eingabemethoden wie Maus, Touchpad und Tastatur. Sowohl Make als auch CMake werden unterstützt, sodass Sie auf einem PC entwickeln und denselben UI-Code auf eingebetteter Hardware verwenden können. Für einige Benutzer kann das eine interessante Funktion sein.
https://github.com/lvgl/lvgl

ESP-DASH (github.com/ayushsharma82)

ESP-DASH ist eine Hochgeschwindigkeits-Bibliothek zur Entwicklung eines funktionalen Echtzeit-Dashboards für ESP8266- und ESP32-Mikrocontroller. Sie wird vom GitHub-Benutzer ayushsharma82 und anderen Mitwirkenden entwickelt. Derzeit gibt es Version 4. Die Bibliothek bietet viele Funktionen, darunter Diagramme, Anzeigekarten, interaktive Schaltflächen und zahlreiche andere Komponenten, womit sich schöne Dashboards erstellen lassen. Sie sind lokal zugänglich und erfordern keine Internetverbindung, da alle Daten auf dem Chip gespeichert sind. ESP-DASH bietet eine automatische Erstellung einer entsprechenden Webseite und Echtzeit-Updates für alle angeschlossenen Clients. Die Benutzer müssen sich nicht in HTML, CSS oder JavaScript einarbeiten, da es eine benutzerfreundliche C++-Schnittstelle bietet. Es gibt vorkonfigurierte Komponenten für die Verwaltung Ihrer Daten und das Projekt ist sehr flexibel. Komponenten können Sie direkt auf der Webseite hinzufügen oder entfernen. Zudem gibt es integrierte Unterstützung für Diagramme. Das erhöht die Funktionalität.
https://github.com/ayushsharma82/ESP-DASH



Übersetzung von Jürgen Donauer (230588-02)

Piezoelektrische Bauelemente

Bemerkenswerte Bauteile

Von David Ashton (Australien)

Der piezoelektrische Effekt ist seit den 1880er Jahren bekannt, aber wie er von einer wissenschaftlichen Kuriosität zu einem Grundnahrungsmittel der modernen Elektronik wurde, ist eine Geschichte der Innovation und Entdeckung. Diese einzigartige Eigenschaft bestimmter Kristalle hat dazu geführt, dass der piezoelektrische Effekt in vielen Bereichen genutzt wird, von Audiowandlern bis hin zu Präzisionsinstrumenten in der Astronomie und digitalen Bildgebung.

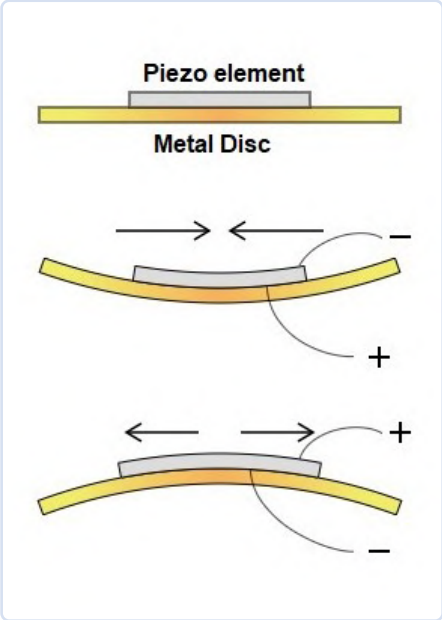


Bild 1. Piezoeffekt. (Quelle: Sonitron Support - CC BY-SA 3.0, commons.wikimedia.org/w/index.php?curid=115322872)

Bild 2. Eine Auswahl von Piezo-Schallgebern. Es handelt sich um Piezoelemente, die mit einem Wechselstromsignal angesteuert werden müssen. A: ein typischer Piezo-Summer, B: ein ähnliches Gerät, das einen Blick in sein Inneres gewährt, C: ein Kristall-Ohrhörer und D: ein 100-W-Piezohorn-Hochtöner für den PA-Einsatz.

Wenn bei bestimmten kristallinen keramischen Substanzen zwei gegenüberliegende Seiten des Kristalls mit Elektroden versehen sind und eine Spannung angelegt wird, verformt sich der Kristall leicht. Dieser Effekt wurde erstmals 1880 von den Brüdern Pierre und Jacques Curie nachgewiesen, obwohl er bereits 1861 von Gabrielle Lippmann mathematisch vorhergesagt worden war. In den folgenden Jahrzehnten wurde der Effekt jedoch lediglich als Laborkuriosität betrachtet.

Dann wurde der Effekt aber in der Elektronik auf vielfältige Weise genutzt. Am bekanntesten ist den Elektronikfreunden wohl der Tonwandler. Ich erinnere mich, dass ich vor über 50 Jahren mein erstes Kristallradio mit einem Kristallhörer gebaut habe, der nur ein Piezo-Wandler auf einer kleinen Metallscheibe war. **Bild 1** zeigt das Funktionsprinzip. Sie haben den Vorteil, dass sie eine relativ hohe Impedanz haben, was für ein Quarzgerät wichtig ist, damit es nicht sehr belastet wird. Außerdem sind sie sehr dünn, was dazu führt, dass sie in Grußkarten verwendet werden, die eine Melodie spielen, wenn man sie öffnet! Es gibt größere Geräte, die einen halbwegs ordentlichen Frequenzgang haben, und auch einige HiFi- und PA-Hochtöner verwenden Piezotechnologie (**Bild 2**). Die meisten dieser Geräte haben eine Resonanzfrequenz, bei der sie am effizientesten sind, und diese wird in Schallgebern mit einem eingebauten Oszillator genutzt, der die Scheibe bei oder nahe der Resonanzfrequenz anregt. Sie werden als „Piepser“ und Alarmgeber (**Bild 3**) verwendet, um Warnsignale zu geben, und sie haben den Vorteil, dass sie sehr effizient sind und wenig Strom für den von ihnen abgegebenen Krach verbrauchen.

Andere Anwendungen von Piezo-Wandlern sind Tintenstrahldrucker (eine kleine

Bild 3. Eine Auswahl von Piezo-Summern (Piepser) mit Elektronik im Inneren, die das Element ansteuert. Hinten: Eine 12-V-Piezosirene, Mitte: drei verschiedene Piezo-Buzzer, vorne: Ein AC/DC-Multispannungs-Piezosummer für den Einsatz in industriellen Schalttafeln.

Bewegung wird verwendet, um ein Tintentröpfchen auszustoßen) In astronomischen Teleskopen werden Piezo-Aktuatoren verwendet, um sehr kleine Korrekturen an der Form des schweren Spiegels vorzunehmen, der das Licht sammelt, so dass die Verzerrung minimiert wird. Piezokristalle können relativ große Kräfte über sehr kleine Entfernungen ausüben. Eine weitere Anwendung ist die Bildstabilisierung in Digitalkameras, wo eine kleine, präzise Bewegung mit einer schnellen Reaktionszeit erforderlich ist.

Der Effekt ist reversibel, das heißt, wenn ein Piezoelement verbogen oder verformt wird, erzeugt es eine Spannung an seinen Elektroden. Dies wird in Gasanzündern verwendet, bei denen ein Element langsam verformt wird und dann mit Hilfe eines Ratschenmechanismus schnell in seine normale Position zurückkehrt (**Bild 4**). Dabei wird ein Impuls mit sehr hoher Spannung erzeugt, der ausreicht, um einen Funken zwischen zwei Elektroden zu erzeugen und das Gas zu entzünden. Sie werden auch in Vibrationssensoren eingesetzt, wo ihre geringe Größe und niedrigen Kosten von großem Vorteil sind.

RG — 230684-02

Über den Autor

David Ashton wurde in London geboren, wuchs in Rhodesien (heute Simbabwe) auf, lebte und arbeitete in Simbabwe und heute in Australien. Er interessiert sich für Elektronik, seit er „kniehoch zu einem Grashüpfer“ war. Rhodesien war nicht das Zentrum des Elektronikuniversums, so dass das Anpassen, Ersetzen und Schnorren von Bauteilen zu den Fähigkeiten gehörte, die er sich früh aneignete (und auf die er immer noch stolz ist). Er hat ein Elektroniklabor geleitet, war aber hauptsächlich in der Telekommunikation tätig.

Haben Sie Fragen oder Kommentare?

Wenn Sie technische Fragen oder Kommentare zu diesem Artikel haben, können Sie sich gerne per E-Mail an die Elektor-Redaktion wenden: redaktion@elektor.de.

Bild 4. Ein Piezo-Zünder aus einem Gasfeuerzeug. Drückt man den Knopf am Ende, macht es „klick“ und es wird ein Hochspannungsimpuls erzeugt, der einen Funken zwischen zwei Elektroden schlägt.

Ein intelligenter Objektzähler

Bilderkennung leicht gemacht mit Edge Impulse

Von Somnath Bera (Indien)

Entdecken Sie, wie Sie einen Raspberry Pi und eine Kamera mit Hilfe der Edge-Impulse-Plattform in ein intelligentes Werkzeug zur Objektzählung verwandeln können! Dieses unterhaltsame und leicht zugängliche Projekt zeigt, wie einfach es ist, Edge Impulse auf einem Raspberry Pi zu betreiben. Es eignet sich sowohl für Anfänger als auch für erfahrene Enthusiasten.

Edge Impulse ist auf die Bereitstellung von Tools und Plattformen für die Entwicklung von maschinellen Lernmodellen für insbesondere auf eingebettetes Edge-Computing spezialisiert. Edge-Computing bedeutet die Verarbeitung von Daten in der Nähe der Datenquelle anstatt auf einem entfernten Server. Das ist perfekt für die Implementierung auf einem Raspberry Pi! In diesem Beispiel wollen wir kleine Objekte zählen, und zwar gewöhnliche Knöpfe auf gewöhnlichem Stoff.

Plattformen für maschinelles Lernen wie Edge Impulse verwenden so genannte Modelle, das heißt, bestimmte Arten von Algorithmen, die zur Datenanalyse und Mustererkennung eingesetzt werden. Diese Modelle werden darauf trainiert, Muster zu erkennen, Vorhersagen zu treffen oder Aufgaben auf der Grundlage von Eingabedaten auszuführen.

Die Klassifizierung von Objekten ist mit Edge-Impulse-Modellen leicht möglich. Sie können zwischen Menschen und Tieren, zwischen Fahrrädern und Autos und so weiter unterscheiden. Außerdem können Sie eine bestimmte Art von Objekten leicht von anderen Arten von Objekten unterscheiden. Alles, was Sie brauchen, ist eine Kamera von guter Qualität, ausreichend Licht, eine gute Fokussierung und schließlich einen einigermaßen gut gebauten Computer wie einen Raspberry Pi 3 oder Raspberry Pi 4. Dieses Projekt war von Anfang an für die Installation auf Mikrocontroller-Ebene gedacht, einem Espressif ESP32, einem Arduino Nicla Vision oder ähnlichem. Aus diesem Grund wurde es für einen sehr kleinen Zählbereich (120 Pixel × 120 Pixel) mit einem relativ kleinen Knopf als Objekt der Begierde entwickelt. Letztendlich stellte sich heraus, dass diese MCUs selbst für den kleinsten Bereich überhaupt nicht geeignet waren. Die maschinellen Lernmodelle werden auf den Edge-Impulse-Servern vortrainiert, und es wird eine so genannte Modelldatei erstellt, die auf dem eingebetteten Gerät gespeichert wird. Dabei ist die Modelldatei selbst etwa 8 MB groß! Daher wurde das Projekt schließlich auf einem Raspberry Pi installiert, wo es problemlos funktioniert.

Wissen und Weisheit

Wenn Sie Edge Impulse [1] kennen, dann, das können Sie mir glauben, ist die Hälfte der Arbeit bereits erledigt. Für die andere Hälfte müssen Sie Ihr Modell nur noch optimieren, um es auf ein akzeptables Leistungsniveau abzustimmen. Ein Computer-KI-Modell funktioniert, entschuldigen Sie, wie ein Kind. Stellen Sie sich vor, wie Sie Dinge wie „A steht für Apfel“ und „B steht für Ball“ gelernt haben. Man hat Ihnen einen Apfel aus verschiedenen Blickwinkeln gezeigt und Ihnen dann beigebracht, das Objekt „Apfel“ zu nennen. Das Gleiche gilt für „Ball“. Heute können Sie einen Apfel und einen Ball aus allen möglichen Blickwinkeln relativ leicht erkennen! Und auch eine künstliche Intelligenz kann diese Objekte leicht identifizieren.

Stellen Sie sich vor, es gibt einen Korb, in dem apfelgroße Bälle und ballgroße Äpfel gemischt sind, die alle gleich aussehen. Was würden Sie als Kind tun? Da es nur die Begriffe Apfel und Ball kennt, würde es die Objekte einfach nicht unterscheiden können! Auch eine KI würde danebenliegen. Aber stellen Sie sich vor, der Obstkorb steht bei einem Obst- und Gemüsehändler. Dann ist höchstwahrscheinlich keines der Objekte ein Ball, und einige oder alle davon könnten Äpfel sein. Diesen „Trick“, einen Apfel mit einem Obst- und Gemüsehändler in Verbindung zu bringen, nennt man Weisheit, die man weder von einem Kind noch von einer KI erwarten kann, bis man ihm/ihr den Bezug lehrt. Der Mensch lernt im Laufe seines Lebens viele solcher Bezüge, so dass er schließlich weise genug ist, einen Apfel mit einem Obst- und Gemüsehändler in Verbindung zu bringen.

KI entwickelt sich so schnell weiter, dass sie eines Tages ebenfalls über die nötige Weisheit verfügen dürfte. Aber bis dahin muss man

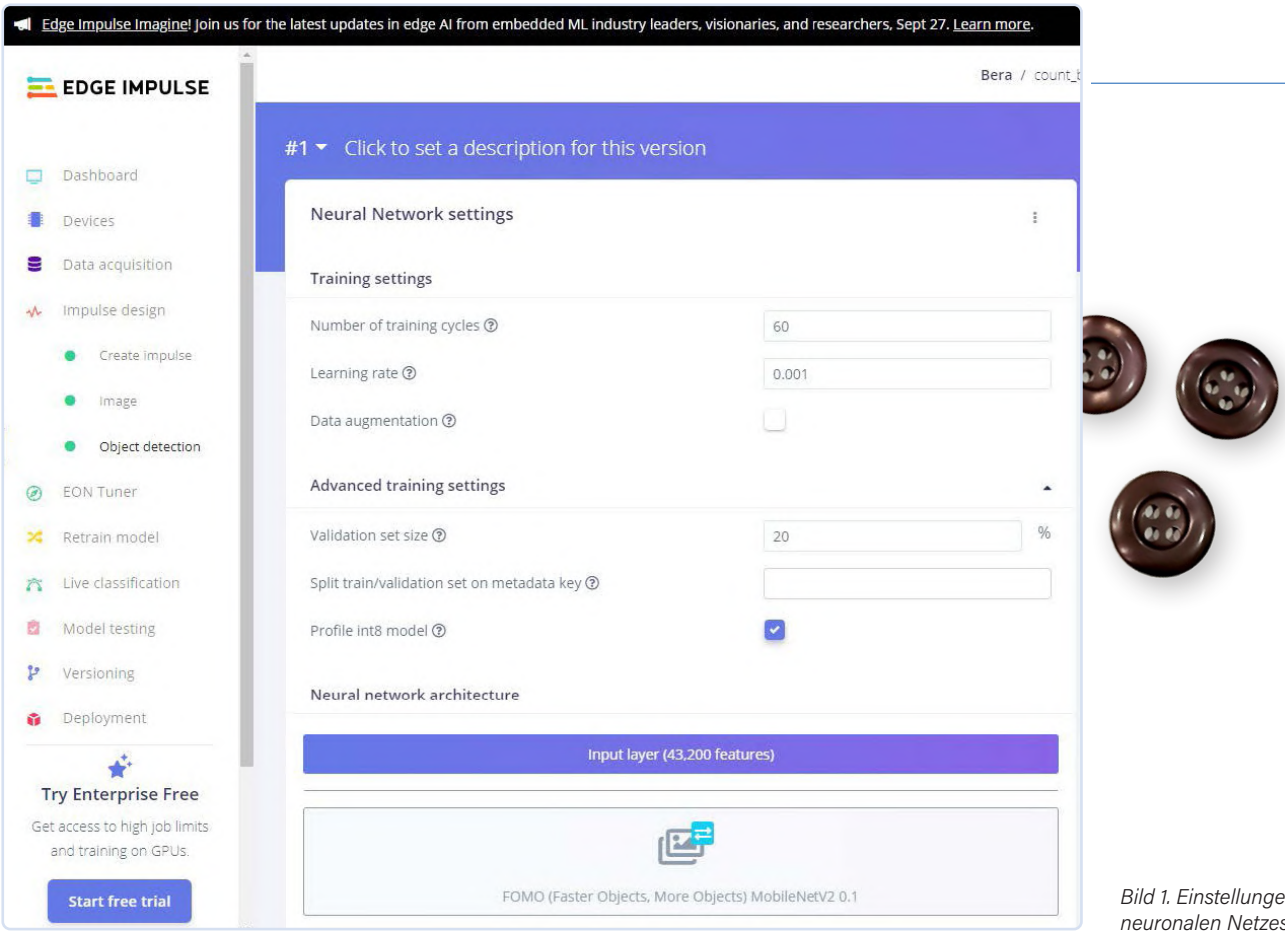


Bild 1. Einstellungen des neuronalen Netzes.

dem ML-Modell für Äpfel und Bälle aus allen möglichen Blickwinkeln beibringen, sie ohne Verwirrung zu erkennen, zum Beispiel durch das Texturprofil eines Apfels, seines Stiels, seine Rillen an der Oberfläche, den Blick von oben und unten und vieles mehr. Auf jeden Fall stehen in Edge Impulse viele verschiedene Modelle mit unterschiedlichen Fähigkeiten zum Testen und Experimentieren zur Verfügung.

Erste Schritte mit Edge Impulse

Eröffnen Sie zunächst ein Konto bei Edge Impulse [1], wofür Sie eine E-Mail-ID benötigen. Sammeln Sie eine Handvoll ähnlicher Knöpfe. Wenn Sie die Website von einem Raspberry Pi aus öffnen und dessen Kamera verwenden (angeschlossen entweder über USB oder über den Cam-Port), können Sie Bilder der Knöpfe aus verschiedenen Winkeln aufnehmen (was erforderlich ist, wenn das Modell in einem realen Arbeitsfeld eingesetzt werden soll). Edge Impulse bietet auch die Möglichkeit, Ihr Mobiltelefon oder Ihren Laptop als Eingabegerät für die Datenerfassung zu nutzen, was für die Datenerfassung mancher Edge-Impulse-Projekte sehr praktisch ist.

Das Projekt

Das Edge-Impulse-Projekt ist grob gesagt in die folgenden Schritte unterteilt, die alle auf der Edge-Impulse-Website ausgeführt werden müssen.

1. Datenerfassung (data acquisition): Das können Bilder, Töne, Temperaturen, Entfernungen und so weiter sein. Ein Teil der Daten wird als Testdaten abgetrennt, während alle anderen Daten als Trainingsdaten verwendet werden.
2. Impulsentwurf (impulse design): Der Hauptteil davon wird als *Create Impulse* bezeichnet. In diesem Zusammenhang bezieht sich ein „Impuls“ auf eine Art Pipeline oder einen Arbeitsablauf zur Erstellung eines maschinellen Lernmodells. Dieser Impuls umfasst mehrere Stufen, darunter die Anpassung der Eingabeparameter im Zusammenhang mit den gerade erfassten Daten, die Signalverarbeitung, die Feature-Extraktion und das maschinelle Lernmodell selbst. „Features“ sind einzelne messbare Eigenschaften oder Merkmale des beobachteten Phänomens. Im Wesentlichen sind Features die Datenattribute, die von Modellen verwendet werden, um Muster zu erkennen und Entscheidungen zu treffen. Die Impulse-Pipeline ist unterteilt in:

   - Eingabeparameter (input parameters): Bild (Breite, Höhe), Ton (Tonparameter)
   - Verarbeitungsblock (processing block): Wie werden die Daten verarbeitet?
   - Lernblock (learning block): Objektdaten dieses Modells

   Sie müssen diese drei Schritte auswählen und konfigurieren.

3. Bildverarbeitung (image processing): Erzeugen von Merkmalen aus den gesammelten Bildern.
4. Objekterkennung (object detection): Wählen Sie Ihr neuronales Netzwerkmodell und trainieren Sie es.

Im letzten Teil, der Objekterkennung, ist Ihre Expertise gefragt (ich würde es eher als Versuch und Irrtum bezeichnen), damit die Genauigkeit des Modells 85 % oder mehr beträgt. Manchmal müssen Sie einige schlechte Bilder (auch Ausreißer genannt) aus dem Modell werfen, um seine Effizienz zu verbessern.
Es gibt eine Handvoll Modelle, die Sie ausprobieren können, um

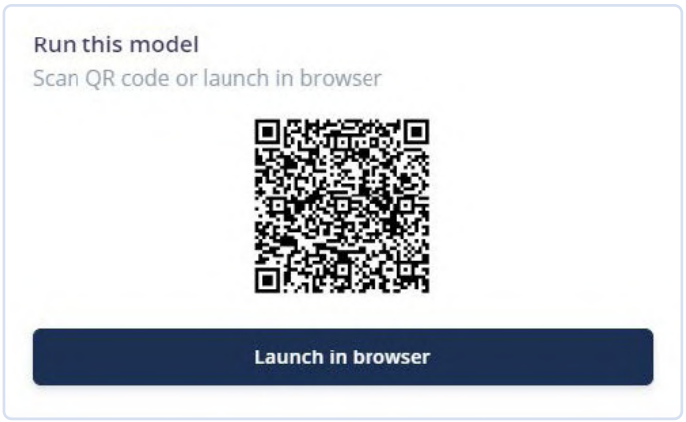


Bild 2. Scannen Sie den QR-Code, um dieses Modell auszuführen.

die Genauigkeit des Modells zu ermitteln. Alles, was über 90 % liegt, ist großartig, aber natürlich sollte es nicht 100 % genau sein! Wenn das der Fall ist, dann stimmt etwas mit Ihren Daten nicht. Es könnten zu wenige Daten oder unzureichende Merkmale vorhanden sein. Überprüfen und korrigieren Sie die Daten und versuchen Sie es erneut! Bei diesem Projekt lag die Genauigkeit bei 98,6 %. Sicherlich war die Anzahl der Daten (etwa 40) gering, aber für ein Einsteigerprojekt ist das ziemlich gut (siehe **Bild 1**). Die Dateien für dieses Projekt sind auf der Elektor-Labs-Webseite des Projekts [4] verfügbar.

Testen des Modells

Sie können Ihr Modell zunächst mit den Testdaten testen. Beginnen Sie dort, und gehen Sie mit Ihrem Gerät dann zu lebensechten Daten über, um zu sehen, ob es funktioniert!

Auf dem Dashboard der Edge-Impulse-Startseite ist die Testfunktion verfügbar. Sie können das Modell direkt im Browser ausführen, oder Sie können es mit Ihrem Smartphone testen. Dazu bietet Edge Impulse einen QR-Code, den Sie mit dem Smartphone scannen können (**Bild 2**). Richten Sie die Kamera auf die Knöpfe (**Bild 3**, **Bild 4** und **Bild 5**) und sehen Sie, ob sie gezählt werden oder nicht!

Einsatz des Raspberry Pi

Um das Modell auf einem Raspberry-Pi-Computer auszuführen, müssen Sie die **.eim*-Datei herunterladen. Aber anders als bei Hardware wie Arduino, Nicla Vision oder ESP32, wo man die Datei direkt herunterladen kann, muss man hier zuerst Edge Impulse auf dem Raspberry Pit installieren und innerhalb der Edge-Impulse-Daemon-Software diese Datei herunterladen. Aber keine Sorge, Edge Impulse hat eine ganze Seite der Installation von Edge Impulse auf dem Raspberry Pi gewidmet. Es gibt ein paar Abhängigkeiten, die zuerst installiert werden müssen. Schauen Sie sich [2] an, wo der Vorgang ziemlich gut beschrieben ist. Es ist wirklich ziemlich einfach.

OK, jetzt haben Sie Edge Impulse auf dem Raspberry Pi installiert; der Spaß kann beginnen. Vergessen Sie nicht, den Raspberry Pi mit dem Internet zu verbinden.

Führen Sie im Terminal des Raspberry Pi den Befehl `edge-impulse-linux-runner` aus. Dadurch wird ein Assistent gestartet, der Sie auffordert, sich anzumelden und ein Edge-Impulse-Projekt auszuwählen. Wenn Sie später zwischen den Projekten wechseln möchten, führen Sie den Befehl erneut mit der Option `--clean` aus. Dieser Befehl kompiliert und lädt das KI-Modell Ihres Projekts automatisch herunter und startet es dann auf Ihrem Raspberry Pi. Zeigen Sie der Kamera, die an Ihren Raspberry Pi angeschlossen ist, die Knöpfe, und sie sollte sie zählen. Das ist gut so! Im Folgenden

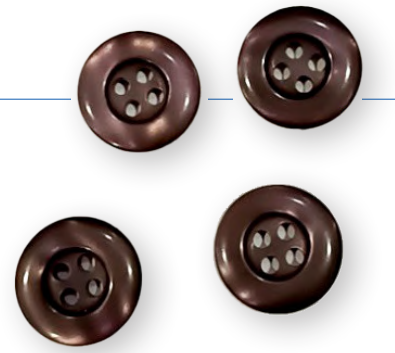


Bild 3. Datenerfassung: Probe-1.

Bild 4. Datenerfassung: Probe-2.

Bild 5. Datenerfassung: Probe-3.

werden wir das System mit Hilfe von Python und einem Sprachsynthesizer modifizieren, der nach dem Zählen die Anzahl der Knöpfe, die er zu zählen hat, ausspricht.

Einsatz des Modells in Python

In der obigen Implementierung würde es so funktionieren, wie es im Edge-Impulse-Modell vorgesehen ist. Damit Sie es für Ihren speziellen Zweck anpassen, zum Beispiel um einen akustischen Alarm auszulösen oder eine LED aufleuchten zu lassen, wenn der Zähler „2 oder mehr“ erreicht, müssen Sie einen anderen Weg finden! Hier kommt Python 3 hinzu, um Ihnen dabei zu helfen, wenn *Linux-sdk-python* auf Ihrem Raspberry Pi installiert ist.

Das Edge Impulse SDK Software Development Kit (SDK) ist für viele Modelle verfügbar, einschließlich Python, Node.js, C++ und so weiter. Informieren Sie sich auf der SDK-Python-Seite [3].

Sobald *linux-sdk-python* installiert ist, gehen Sie in das Verzeichnis *linux-sdk-python/examples/image* und führen Sie die Python-Datei zur Bilderkennung aus. Lassen Sie sich nicht verwirren: Im Beispielverzeichnis gibt es drei Unterverzeichnisse - jeweils eines für Audiodaten, Bilddaten und benutzerdefinierte Daten. Im Bildverzeichnis ist die Videoklassifizierungsdatei auch für

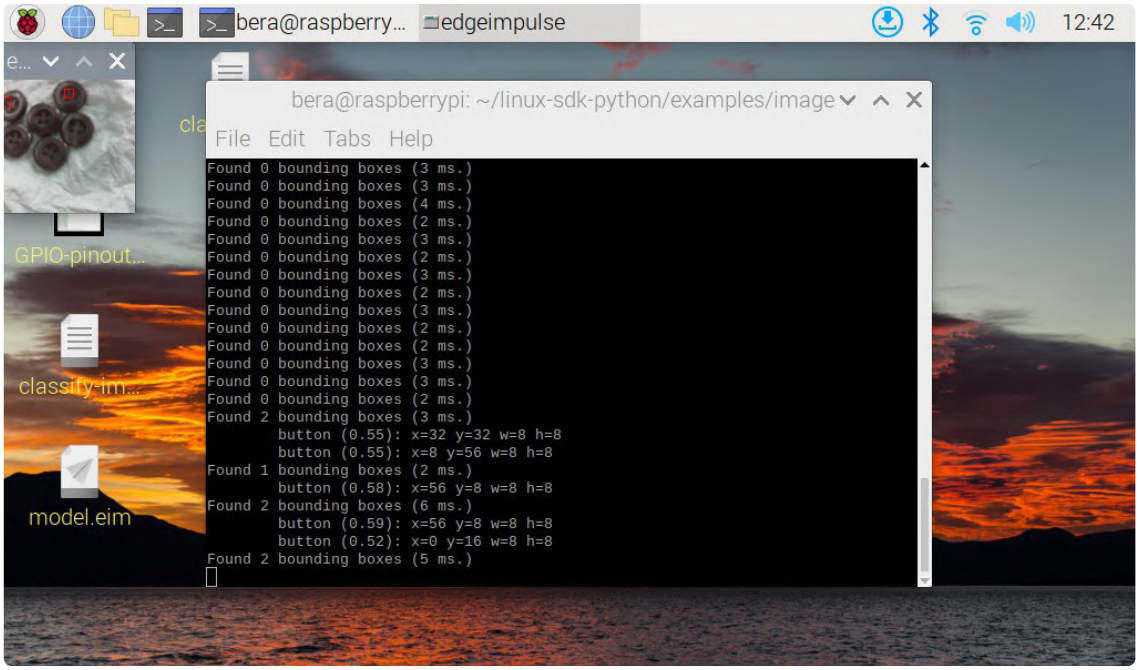


Bild 6. Vier Knöpfe werden wegen unzureichender Fokussierung und Lichtproblemen übersehen.

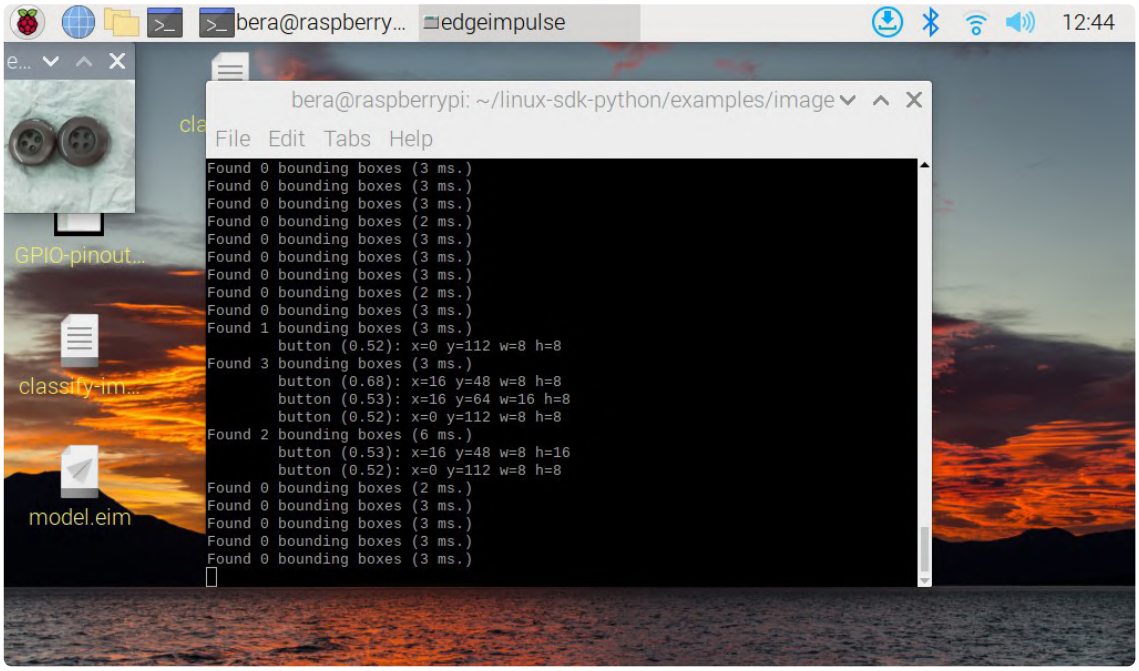


Bild 7. Nachdem ein Knopf entfernt wurde, hat das Modell sie zunächst richtig gezählt.

Videoeingangsdaten verfügbar. Das Verzeichnis *custom* ist für die Anpassung anderer Daten (nur für Experten!). Führen Sie nun den Befehl aus:

```
python3 classify-image.py /home/bera/downloads/model.eim
```

Die Modelldatei **.eim* ist aus dem jeweiligen Verzeichnis ihres Speicherortes zu laden. Wenn Sie möchten, können Sie sie auch in das SDK-Verzeichnis kopieren!
Auf diese Weise müssen Sie die Python-Datei mit der heruntergeladenen *model.eim*-Datei laden. Das Programm findet automatisch das über USB oder den Cam-Port angeschlossene Kameramodul und startet! In der oberen linken Ecke öffnet sich ein kleines Kamera-Fenster (120 × 120), und die identifizierten Knöpfe werden mit einer kleinen roten Markierung versehen. Die identifizierte Anzahl wird im Terminal angezeigt. Achten Sie auf ausreichend Licht und darauf, dass die Kamera für die Knöpfe richtig fokussiert ist. Dies ist besonders bei billigen Kameras wichtig. Wenn Sie das Modell auf Ihrem Smartphone laufen lassen, liefert es weitaus bessere Bilder und zählt viel schneller. Achten Sie dennoch auch hier auf das richtige Licht und die richtige Fokussierung, dann werden Sie bessere Ergebnisse erzielen.
In den obigen Bildschirmfotos des Raspberry Pi ist oben links das kleine 120 × 120-Fenster zu sehen, in dem alle drei Knöpfe vom Modell identifiziert und gezählt werden. Sehen Sie die deutlich sichtbare rote Markierung auf allen drei Knöpfen!
In **Bild 6** fehlen vier Knöpfe, weil der Fokus und das Licht unzureichend sind. Die Kamera hat auch kein Stativ, das man für diese Arbeit verwenden sollte! Deshalb empfehle ich, die Kamera auf einem Stativ zu befestigen, wie bei einem „Mikroskop-Setup“. Achten Sie darauf, dass Sie nicht schräg von oben auf die Objekte schauen.
In **Bild 7** habe ich einen Knopf entfernt, und das Modell hat ihn

Bild 8. Mein Prototyp.

zuerst richtig gezählt. Es zeigte zwei Knöpfe an, aber als ich die Auslösetaste drückte, wurde die Kamera falsch ausgerichtet und verlor die Knöpfe. Die Kamera sollte deshalb über ein langes (Flachband-) Kabel wie in **Bild 8** verfügen, damit sie sich nicht bewegt, wenn man am Computer etwas drückt. Solche Kabel sind unter anderem bei Amazon erhältlich. Sobald die Kamera dann auf einem Stativ befestigt ist und ausreichend (Tages-) Licht auf die Objekte fällt, wird das Modell unfehlbar funktionieren.

Passen Sie Ihr Modell an

Schauen Sie sich bitte die Datei *classify-image.py* an. Es handelt sich um eine einfache Python-Datei, die mit wenig Aufwand angepasst werden kann. In dieser Python-Datei habe ich ein *espeak*-Modul eingefügt, das, sobald ein Knopf oder mehrere gefunden werden, deren Anzahl über einen Lautsprecher ansagt. Um *espeak* auf Ihrem Raspberry Pi zu installieren, führen Sie den Befehl aus:

```
sudo apt-get install espeak
```

In **Listing 1** ist die Python-Datei einschließlich meiner Änderungen dargestellt. Der offene Sprachsynthesizer Espeak ist ein eigenständiges Text-to-Speech-Modul für Python. Er benötigt keine Internetverbindung, um zu funktionieren.

Geänderte Ausführung

Sie haben das Python-Programm modifiziert. Wenn Sie die Python-Datei jetzt ausführen, findet sie die Knöpfe (oben links öffnet sich unser kleines Kamerafenster). Die Zahlen werden im Terminalfenster angezeigt und Espeak sagt die Anzahl an: „Found five buttons“ und so weiter. Wenn Sie ein Relais ansteuern möchten oder eine LED aufleuchten soll, importieren Sie die GPIO-Bibliothek von Python und feuern dann den zugehörigen GPIO, um (wegen des hohen Spulenstroms über einen Schalttransistor) das Relais anzusteuern.

Nachspiel

Edge Impulse wurde 2019 mit dem Ziel gegründet, Entwicklern die Möglichkeit zu geben, die nächste Generation von intelligenten Geräten zu entwickeln. Seitdem sind KI-gesteuerte Programme und Geräte auf ESP32, Jetson Nano, Raspberry Pi, Orange Pi, Maixduino, OpenMV, Nicla Vision und vielen anderen erschienen. Dieser Trend wird sich in den kommenden Tagen noch verstärken! Vorbei sind die Zeiten der Super- oder der großen Markencomputer. Kleine, modulare Geräte mit geringer Stromaufnahme nehmen diesen Raum schnell ein. Und wer weiß? Vielleicht können wir schon bald die eingebaute Weisheit direkt aus der Verpackung heraus installieren!

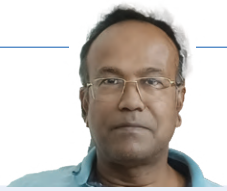
RG — 230575-02

Haben Sie Fragen oder Kommentare?

Wenn Sie technische Fragen oder Kommentare zu diesem Artikel haben, wenden Sie sich bitte per E-Mail an den Autor unter berasomnath@gmail.com oder an die Elektor-Redaktion unter redaktion@elektor.de.

Listing 1: Das Python-Programm nutzt das Tool espeak, um die Anzahl der Knöpfe anzusagen.

```
#!/usr/bin/env python
import device_patches    # Device Specific patches - taken care by the software
import cv2  #import Computer Vision
import os
import sys, getopt
import signal
import time
from edge_impulse_linux.image import ImageImpulseRunner
import subprocess    #this one have been added by Bera
…
        elif "bounding_boxes" in res["result"].keys():
                print('Found %d bounding boxes (%d ms.)' % (len(res["result"]["bounding_boxes"]),
                res['timing']['dsp'] + res['timing']['classification']))
                if  (len(res["result"]["bounding_boxes"])>0):
                        exitCode = subprocess.call(["espeak","-ven+f3","-a200"," Found %d Buttons" %
                        len(res["result"]["bounding_boxes"])  ])   #This one have been added by Bera
…
```



Über den Autor

Somnath Bera, ein Maschinenbauingenieur vom Jalpaiguri Govt. Engg. College, Indien, arbeitet als General Manager bei NTPC, dem größten Stromerzeuger des Landes. Er hat eine tiefe Leidenschaft für Elektronik, was durch seine mehr als 60 innovativen Projekte bei Elektor Labs bewiesen wird, von denen schon mehr als zehn in Elektor vorgestellt wurden. Seine Projekte beschäftigen sich oft mit Problemlösungen in Bereichen wie der Abfallwirtschaft und dem Umgang mit natürlichen Ressourcen. Somnath verwendet gerne innovative Ansätze und Plattformen wie Arduino, Raspberry Pi und ESP32 in Verbindung mit verschiedenen Arten von Sensoren und drahtlosen Systemen, um effiziente und kostengünstige Lösungen zu schaffen.

Passende Produkte

- **Raspberry Pi 4 B (1 GB RAM)**
  www.elektor.de/18966
- **G. Spanner, *Machine Learning mit Python für PC, Raspberry Pi und Maixduino* (Elektor)**
  Buch, kartoniert, deutsch: www.elektor.de/19981
  E-Buch, PDF, deutsch: www.elektor.de/19982
- **D. Situnayake, Jenny Plunkett, *AI at the Edge Rande* (O'Reilly)**
  Buch, englisch: www.elektor.de/20465

WEBLINKS

[1] Edge Impulse: https://edgeimpulse.com/
[2]] Installation von Edge Impulse auf dem Raspberry Pi 4: http://tinyurl.com/ysc6mtuz
[3] Linux Python-SDK: http://tinyurl.com/2bat4w6z
[4] Projektseite bei Elektor Labs: https://t1p.de/mfnqr

Meistern Sie die kniffligsten Probleme Ihrer Embedded-Entwicklungen!

Von Stuart Cording, für Mouser Electronics

Die Entwicklung eingebetteter Systeme bietet Ingenieuren nicht nur viele Möglichkeiten, sondern auch mindestens doppelt so viele Herausforderungen! Und die Herausforderungen sind so vielfältig und abwechslungsreich wie es nur geht. Dank der einzigartigen Kombination von Elektronik, Software und Systemen können Sie in den analogen Bereich eintauchen, ein digitales Timing-Problem lösen, komplexe Regelkreisprobleme untersuchen oder die Qualität von Sensormesswerten beurteilen. Die Hersteller von Mess- und Prüfgeräten haben dies längst erkannt und erleichtern mit cleveren Features, neuen Funktionen, Konnektivität und Smartphone-Apps Ihr Leben als Entwickler.

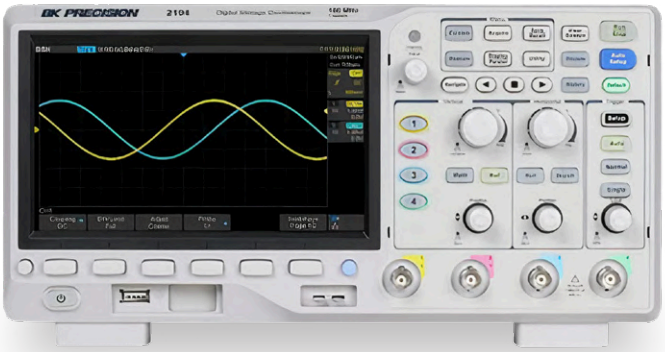


Bild 1. Das B&K Precision Modell 2194 bietet eine beeindruckende Speichertiefe von 14 Mpts. Eine Suchfunktion erleichtert das Auffinden von Signalanomalien.

Ein Beispiel: Oszilloskope zeigen nicht mehr nur Spannungen auf einer Zeitachse an. Sie können komplexe mathematische Berechnungen an den Eingangssignalen durchführen, um die Analyse zu unterstützen, serielle Datenkommunikation dekodieren und Trigger auf komplexe Signale auslösen. Dann gibt es Soft-Oszilloskope, die auf FPGA-Technik basieren und eher ein Miniaturlabor sind. Sie bieten von Haus aus die gewohnten Standard-Messfunktionen, können aber dank ihrer Programmierbarkeit auf viel mehr abgestimmt werden. Wenn Sie mit Ihrem Laptop als Bildschirm zufrieden sind und Ihnen der Platz auf Ihrem Labortisch fehlt, sind solche Geräte eine gute Alternative.

Oszilloskope suchen Runt-Bits

Mit dem Aufkommen des digitalisierenden Oszilloskops wurde die Suche nach den Ursachen von Problemen viel einfacher als die alte Methode, nämlich ein Foto des Röhrenbildschirms zu machen. Die heutigen Geräte lassen sich leicht an Netzwerke und Computer anschließen, so dass die aufgezeichneten Signale im Team begutachtet und genutzt werden können, was die Konfiguration und automatisierte Prüfung vereinfacht. In dieser Hinsicht ähnelt das B&K Precision Modell 2194 [1] vielen anderen vierkanaligen Oszilloskopen auf dem Markt (**Bild 1**). Wie bei allen bedeutenden Anschaffungen lohnt es sich jedoch, vor dem Kauf das Benutzerhandbuch zu lesen, um den vollen Funktionsumfang zu entdecken.

Mit einer respektablen Eingangsbandbreite von 100 MHz, einer maximalen Abtastrate von 1 GSa/s und einem 7-Zoll-TFT-Display mit 800 × 480 Pixeln ist das Oszilloskop 2194 genauso gut ausgestattet wie vergleichbare Geräte, die mehr als doppelt so viel kosten. Mit einem Gewicht von 2,6 kg und einer Größe von 31 × 15 × 13 cm ist es leicht zu transportieren und nimmt nur wenig Platz auf dem Labortisch ein. BNC-Anschlüsse und USB für Speichermedien sind an der Vorderseite leicht zugänglich. Auf der Rückseite befinden sich nur eine USB-Buchse für die Steuerung, eine Ethernet-Buchse und der BNC-Trigger-Ausgang (der auch als Pass/Fail-Ausgang für die Testautomatisierung dient). Das Gerät verfügt außerdem über einen Befestigungspunkt für die bei Laptops

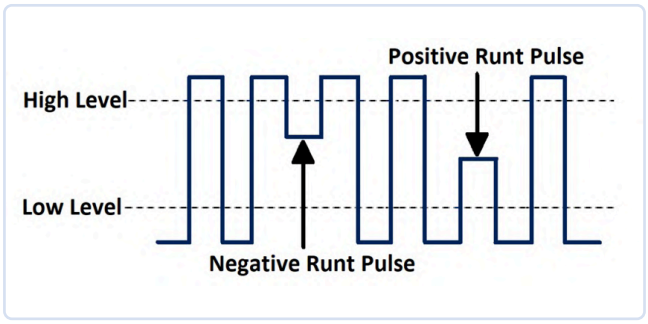


Bild 2. Unregelmäßige Impulse, die durch Probleme mit einer Treiberschaltung verursacht werden, können als Trigger verwendet oder mit der Suchfunktion des 2194 in den erfassten Signalen gefunden werden.

Bild 3. Das Moku:Go gleicht eher einem Miniaturlabor und bietet die Möglichkeiten von acht verschiedenen Prüfgeräten in einem Gerät, das nur ein wenig größer als Ihre Hand ist.

verwendeten Kabelschlösser. Für die Signalanalyse bietet das Gerät 38 Arten von Messparametern und Statistiken, darunter Zählung, Standardabweichung, Anstiegs- und Abfallzeiten sowie Spitze-Spitze. Das Oszilloskop dekodiert auch I²C, SPI, UART, CAN und LIN.

Eine Aktualisierungsrate von 100.000 wfms/s hilft bei der Suche nach Störungen und anderen seltenen Ereignissen. Beeindruckend ist auch die Speichertiefe. Mit 14 Mpts können lange Zeiträume von Signalen erfasst und mit der Zoom-Funktion analysiert werden, um Anomalien zu finden. Die Suche nach Anomalien in solch umfangreichen Datensätzen kann jedoch eine Herausforderung darstellen. Dank der Suchfunktion des 2194 kann eine Reihe von Ereignissen gefunden werden, darunter Flanken, Steigungen, Pulse und Intervalle.

Ein weiteres Problem für Entwickler ist es, digitale Signale zu finden, die nicht korrekt ausgebildet sind. Diese so genannten Runt-Impulse (**Bild 2**) treten auf, wenn die Anstiegsgeschwindigkeit des Treibers nicht ausreicht, um den gewünschten Logikpegel in der erforderlichen Zeit zu erreichen. Runt-Impulse können mit dem 2194 gesucht und gefunden, aber auch als Trigger verwendet werden. Dies wird erreicht, indem sowohl ein oberer als auch ein unterer Triggerpegel definiert wird. Überschreitet ein Signal den einen, aber nicht den anderen Schwellenwert, erfasst der Trigger die Wellenform zur Überprüfung und vereinfacht so die Suche nach den schwer zu findenden zeitweisen Fehlern.

Moku:Go von Liquid Instruments

Für diejenigen, die ständig unterwegs sind oder im Außendienst arbeiten, kann ein herkömmliches Oszilloskop zu schwer und zu sperrig sein. Es gibt jedoch Alternativen, wenn Sie bereit sind, Ihren Laptop als visuelle Schnittstelle zu verwenden. Ein solches Gerät ist das Moku:Go von Liquid Instruments [2], ein kompaktes Gerät mit den Abmessungen 24 × 3,8 × 13 cm und einem Gewicht von nur 750 g (**Bild 3**). Dank seines FPGA-basierten Designs umfasst das Moku:Go eigentlich acht Geräte in einem Gehäuse; ein 30-MHz-Oszilloskop, ein Echtzeit-Spektrumanalysator, ein 20-MHz-Wellenformgenerator, PID-Controller und ein Arbiträrwellenformgenerator, um nur einige der Funktionen zu nennen.

Das Moku:Go M0 besitzt zwei analoge Eingänge, zwei analoge Ausgänge und 16 digitale I/Os. Außerdem lässt es sich über USB-C problemlos an Windows- und Mac-Rechner anschließen. Das M1 fügt eine programmierbare Zweikanal-Stromversorgung (PSU) für -5 V bis +5 V und 0 V bis 16 V bei 150 mA zusammen mit den erforderlichen Kabeln hinzu. Das M2 ist mit zwei weiteren programmierbaren Netzteilen für 0,6 V bis 5 V bei bis zu 1 A pro Kanal ausgestattet, zusammen mit Ethernet-Konnektivität.

Programmierbare Netzteile mögen zwar nicht wie eine Killer-Funktion erscheinen, haben aber Sinn, wenn man den entnommenen Strom als Parameter in Testaufbauten verwenden möchte. Dies wird in der Application Note „Converter Evaluation Using TI TPS63802EVM“ [3] demonstriert, in der der Wirkungsgrad eines Abwärtswandlers gemessen wird. Die mit den beiden Analogkanälen gekoppelten Netzteile sind mehr als ausreichend, um den Wirkungsgrad zu ermitteln, und der Math-Kanal liefert, wenn er mit dem Lastwiderstand versehen ist, bequem die Leistung in Watt. Die Automatisierung der Messungen ist auch über Python, MATLAB und LabView-Programmier-APIs möglich. Es ist schwierig, auf dem hier zur Verfügung stehenden Platz genügend Hintergrundinformationen über das Moku:Go zu geben. Neben den beeindruckenden Application Notes [4] muss jedoch auch die App des Tools gelobt werden [5]. Erstens wirkt die Benutzeroberfläche sauber und aufgeräumt, und dank des Demo-Modus können Sie alle verschiedenen Funktionen ausprobieren. So bekommen Sie ein Gefühl dafür, wie die X- und Y-Achse vergrößert und verkleinert wird und wie Kanäle und mathematische Funktionen konfiguriert werden. Wenn Sie Zweifel an der Kompetenz eines solchen Tools haben, sollten Sie diese App ausprobieren, um eine Entscheidung zu treffen.

Testen und Binning von Komponenten

Das Design von Stromversorgungen erfordert ein hervorragendes Verständnis der verwendeten Induktivitäten und Kondensatoren. In manchen Fällen ist es aus Gründen der Genauigkeit erforderlich, jedes Bauteil zu testen und zu Qualitätsgruppen zusammenzufassen (Binning). Unabhängig davon, ob die Herausforderung im Entwicklungslabor oder in der Fabrikhalle liegt, können die LCR-Messgeräte der Serie T3LCR [6] von Teledyne LeCroy dabei helfen (**Bild 4**). Das Gerät verfügt über ein großes 3,5"-TFT-Display

Bild 4. Im Entwicklungslabor kann der T3LCR von Teledyne LeCroy zur Charakterisierung von Bauteilen eingesetzt werden, während er in einer Produktionsumgebung für das Binning von Bauteilen verwendet werden kann.

Bild 5. Mit den Messgeräten der 250W-Serie von Extech sind eine Reihe von Umweltmessungen möglich, darunter Schallpegel, Drehzahl, Lichtstrom, relative Luftfeuchtigkeit und Luftgeschwindigkeit.

und lässt sich über die Frontplatte oder über die USB- oder RS-232C-Anschlüsse leicht für Vierleitermessungen konfigurieren. Die drei T3LCR-Modelle unterstützen Testfrequenzen von 10 Hz bis 2 kHz, 100 kHz oder 300 kHz und bieten jeweils eine Grundgenauigkeit von 0,05 %.
Funktionen wie die automatische Pegelregelung (ALC) eignen sich für die konstante Prüfspannung von Keramikkondensatoren, während der einstellbare Prüfstrom für Induktivitätsmessungen geeignet ist. Eine einstellbare interne DC-Vorspannung von ±2,5 V kann verwendet werden, um AC und DC gleichzeitig zu simulieren, um Kapazitätsschwankungen zu bewerten. In der Betriebsart *List Measurement* können bis zu zehn automatische Messparameter erfasst werden, die eine Charakterisierung von Bauteilen und deren Veränderungstrends ermöglichen. Auf der Rückseite des T3LCR befindet sich ein DB-25-Anschluss für die Verbindung mit einem Handler für das Binning von Bauteilen, der bis zu zehn Bins unterstützt. Sowohl Serien- als auch Parallel-Äquivalentmodellmessungen sind für Widerstand, Induktivität und Kapazität verfügbar. Ein weiteres Dutzend Parameter kann ebenfalls erfasst werden, darunter Verlustfaktor (D), Qualitätsfaktor (Q), Phasenwinkel und Gleichstromwiderstand.

Behalten Sie Ihre Umwelt im Auge

Bei der Entwicklung eingebetteter Systeme ist es oft erforderlich, die vom Mikrocontroller erfassten Sensordaten mit den Messungen professioneller Testgeräte zu vergleichen. In anderen Fällen sind bei Umwelttests Langzeitmessungen erforderlich. *Die Bluetooth Connected Environmental Meter* der Reihe 250W von Extech [7] eignen sich ideal für solche Situationen (**Bild 5**) und bieten Messungen von Drehzahl, Lichtintensität und relativer Luftfeuchtigkeit bis hin zu Schallpegel und Luftgeschwindigkeit. Mit ihren Abmessungen von 54 × 28 × 120...176 mm liegen die Geräte gut in der Hand und sind mit ihren großen und hell beleuchteten LCDs gut ablesbar.
Das RPM250W ist ein Laser-Drehzahlmesser, der Drehzahlen zwischen 10 U/min und 99.999 U/min mit einem Fehler von 0,04 % misst, während das LT250W die Lichtintensität bis zu 100.000 Lux erfasst. Das kompakte Luftströmungsmessgerät AN250W liefert seine Ergebnisse in ft/min, m/s und Knoten zusammen mit der Umgebungstemperatur. Für A-bewertete Schalldruckpegel bietet das SL250W „Human Hearing"-Frequenzmessungen mit Aufzeichnung der Max/Min-Werte. Schließlich können mit dem RH250W auch die relative Luftfeuchtigkeit und die Temperatur gemessen und aufgezeichnet werden. Jedes Gerät bietet auch Bluetooth-Konnektivität, so dass sie mit einem iOS- oder Android-Gerät gekoppelt und mit der ExView-App von Extech verwendet werden können. Diese App kann gleichzeitig Daten von bis zu acht Messgeräten der Reihe 250W erfassen und so konfiguriert werden, dass sie bei hohen/niedrigen Werten einen akustischen Alarm erzeugt. Die Daten werden lokal protokolliert und können im .csv-Format weitergegeben werden. Darüber hinaus können Fotos von Messstellen in PDF-Berichte mit den Messdaten eingebettet werden.

Ein Messwerkzeug für jeden Embedded-Entwickler

Während die Entwicklung eingebetteter Systeme vielfältig und interessant ist, erfordern einige der Probleme, mit denen man dabei konfrontiert wird, clevere Testgeräte, um die Ursache für sporadische Ausfälle zu finden, die tatsächliche Effizienz von Stromwandlern zu bestimmen oder Umweltmessungen mit den Ergebnissen eingebetteter Sensoren zu vergleichen. In anderen Fällen ist es von entscheidender Bedeutung, den genauen Wert der verwendeten Bauteile zu kennen oder es kann während der Produktion ein Binning erforderlich sein. Wir hoffen, dass Ihnen eine der hier vorgestellten Testlösungen bei der Lösung Ihrer Aufgaben in der Embedded-Entwicklung helfen wird.

RG — 230750-02

Über den Autor

Stuart Cording ist ein freiberuflicher Journalist, der unter anderem für Mouser Electronics schreibt. Er ist auf Videoinhalte spezialisiert und konzentriert sich auf tiefgehende technische Analysen und Einblicke. Daher interessiert er sich besonders für die Technologie selbst, ihre Einbindung in Endanwendungen und Vorhersagen über zukünftige Fortschritte.

Mouser Electronics ist autorisierter Distributor für Halbleiter und elektronische Komponenten, der sich auf neue Produkteinführungen seiner führenden Herstellerpartner konzentriert.

WEBLINKS

[1] B&K Precision Modell 2194: https://eu.mouser.com/new/bk-precision/bk-2194-oscilloscope/
[2] Moku:Go von Liquid Instruments: http://tinyurl.com/Moku-Go
[3] Bewertung des Wandlers mit dem TI TPS63802EVM: http://tinyurl.com/AppNoteConverter
[4] Application Notes von Liquid Instruments: https://www.liquidinstruments.com/blog/category/application-notes/
[5] Windows- und macOS Anwendungen von Liquid Instruments: https://www.liquidinstruments.com/products/desktop-apps/
[6] T3LCR von Teledyne LeCroy: http://tinyurl.com/TeledyneT3LCR
[7] Extech 250W: https://eu.mouser.com/new/extech/extech-250w-meters/

ESP32-Terminal

Ein Handheld-Gerät mit Touch-Display

Von Johan van den Brande (Belgien)

Das ESP32-Terminal von Elecrow ist ein ESP32-S3-gesteuertes Handheld-Gerät mit einem kapazitiven TFT-Touch-Display mit einer Diagonalen von 3,5 Zoll, einer Auflösung von 480 × 320 Pixeln und einer Vielzahl von Möglichkeiten. Das Gerät kann Ihre Projekte gut ergänzen, wenn Sie ein berührungsfähiges Display mit Schnittstellenfunktionen benötigen.

Das Display mit dem Displaytreiber ILI9488 und dem Touch-Controller FT6236 weist eine Farbtiefe von 16 Bit auf. Beide Chips werden von der Arduino-Community gut unterstützt. Das ESP32-Terminal ist in einem schwarzen Acryl-Gehäuse untergebracht, das sich nicht fummelig anfühlt, sondern einen sehr soliden Eindruck macht (**Bild 1**). Ein Batterieanschluss mit LiPo-Ladeschaltung ist vorhanden, aber das Gerät wäre mit einer Batterie *im* Gerät mobiler. Wenn Sie Ihr Projekt wirklich tragbar machen wollen, müssen Sie daran einen Akku anbringen. Auf der Rückseite befinden sich zwei M3-Befestigungslöcher, an denen Sie ein Zusatzgehäuse befestigen oder das Gerät an einer Wand befestigen können.

Der Controller ESP32-S3

Wenn ich mir anschaue, was das Gerät steuert, bin ich immer wieder erstaunt über die unglaubliche Rechenleistung einer modernen MCU. Das ESP-Terminal wird von einem ESP32-S3 [2] von Espressif gesteuert, in dem eine 32-Bit-Dual-Core-MCU XTensa LX7 mit 240 MHz arbeitet. Vergleichen Sie das einmal mit dem ursprünglichen Arduino UNO mit seinem 8-Bit-ATMega328 von Microchip, der mit 16 MHz läuft! Die MCU besitzt 512 KB SRAM und 8 MB PSRAM (Pseudo-static RAM, eine Art externes statisches RAM), das hier über einen SPI-Bus mit der MCU verbunden ist.

Was die drahtlose Konnektivität betrifft, so haben wir 2,4 GHz-WLAN (2,4 GHz 802.11 b/g/n) und Bluetooth 5 LE. Ich war mir dessen nicht bewusst, aber Bluetooth 5 hat durchaus Long-Range-Fähigkeiten, und dieses Modul behauptet, es zu unterstützen. Der Long-Range-Modus erweitert die Reichweite von 10...30 m auf mehr als 1 km.

Verbindung mit der echten Welt

Das ESP32-Terminal hat einen Batterieanschluss und ein integriertes LiPo-Ladegerät. Sie können das Gerät über den USB-C-Anschluss mit Strom zu versorgen, aber auch den LiPo-Akku laden. Es gibt auch einen micro-SD-Kartensteckplatz, der praktisch ist, wenn Sie ein paar Bilder oder andere Dateien wie Webseiten für Ihre Anwendung speichern möchten.

Bild 1. Das ESP32-Terminal in seinem soliden schwarzen Gehäuse zeigt Wetterdaten.

Bild 2. Die vier „Crowtail"-Ports sind auf der Rückseite sichtbar.

An den Seiten des Moduls befinden sich insgesamt vier „Crowtail"-Ports [3] (**Bild 2**), 4-Draht-Schnittstellen, die mit Grove von Seeed Studio kompatibel sind. Es gibt einen digitalen, einen analogen, einen seriellen (UART) und einen I²C-Anschluss, was für einfache Projekte ausreicht. Wenn Sie mehr benötigen, können Sie den I²C-Anschluss als Erweiterung verwenden.

Programmierung des ESP32-Terminals

Das Hochladen der Firmware erfolgt über das USB-C-Kabel, über eine serielle Verbindung, die durch einen USB-zu-Seriell-Konverter CH340 [4] herausgeführt wird. Ich hatte kein Glück bei dem Versuch, dies mit meinem MacBook zu erledigen, bei dem der offizielle Treiber installiert war. Zum Glück besitze ich noch meinen alten Linux-Laptop, auf dem Ubuntu läuft, und das funktionierte perfekt!

Zuerst wollte ich eine Variante von Python auf dem Gerät laufen zu lassen. Das erwies sich als schwierig, denn obwohl in der Dokumentation steht, dass das Gerät mit Python und MicroPython programmiert werden kann, konnte ich keine fertige Firmware zum Herunterladen finden. Auf der Website von elecrow gibt es neben einem Tutorial für Arduino [5] zahlreiche Beispiele, die ich für meine Demoanwendungen genutzt habe.

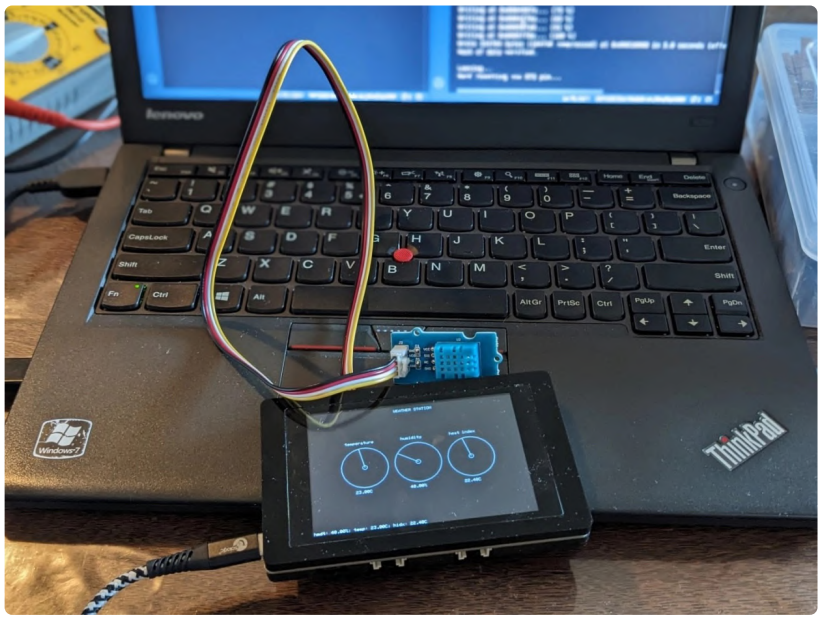


Bild 3. Ein DHT-11-Sensor, etwas Code, und schon haben Sie eine einfache Wetterstation.

UI-Design

Laut der Webseite des ESP32-Terminals ist das Gerät LVGL-zertifiziert. LVGL [6][7] steht für *Light and Versatile Graphics Library*, eine Open-Source-Grafikbibliothek für die Erstellung von Benutzeroberflächen für verschiedene MCUs und Displaytypen. Obwohl die Bibliothek selbst quelloffen ist, ist der Drag-and-Drop-Layout-Editor namens Squareline Studio [8] closed source und kostenintensiv.

Ich habe die Testversion installiert, und obwohl sie auf den ersten Blick etwas überwältigend ist, ist es mir schnell gelungen, eine einfache Benutzeroberfläche für das untenstehende Servoprojekt zu erstellen. Wenn die Erstellung von Benutzeroberflächen für eingebettete Geräte Ihr Ziel ist, dann lohnt es sich, etwas Zeit zu investieren, um diese Produkte kennenzulernen.

Zwei kleine Projekte

Ich habe beschlossen, zwei Beispielprojekte zu erstellen. Das erste Projekt ist eine kleine Wetterstation mit einer Benutzeroberfläche mit Linien- und Kreisprimitiven. Das zweite Projekt ist ein Beispiel für die Steuerung eines Servos über eine mit Squareline Studio entworfene Benutzeroberfläche.

Wetterstation

Für die Wetterstation kommt ein DHT-11-Sensor zum Einsatz, der Temperatur und Feuchtigkeit messen kann. Er besitzt eine digitale Ein-Draht-Schnittstelle und wird von Arduino gut unterstützt. Für dieses Projekt habe ich deshalb die DHT-Bibliothek von Adafruit [9] gewählt. Sie ist Teil einer Reihe von Sensorbibliotheken, die einen gemeinsamen Code haben, und deshalb müssen wir auch die gesamte *Adafruit Unified Sensor Driver Library* [10] installieren.

In diesem Projekt wollte ich meine eigene Benutzeroberfläche von Grund auf erstellen und dazu einfache Grafikelemente zu verwenden, um Text und einige Liniengrafiken darzustellen. Dazu verwendete ich die offene LCD- und E-Ink-Grafikbibliothek *LovyanGFX* [11][12] (**Bild 3**).

Nach der Initialisierung des Bildschirms, für die ich einfach eines der Beispiele von der ESP32-Terminal-Website kopiert habe, wird der DHT-11-Sensor ausgelesen. Dieser liefert drei Werte: Temperatur, Luftfeuchtigkeit und den Hitzeindex, ein um die aktuelle Luftfeuchtigkeit bereinigter Temperaturwert (gefühlte Temperatur). Diese drei Werte werden auf einer einfachen Skala angezeigt, die aus drei Kreisen und je einer Linie als Zeiger besteht. Die Anzeige wird alle zwei Sekunden mit einem neuen Wert aktualisiert.

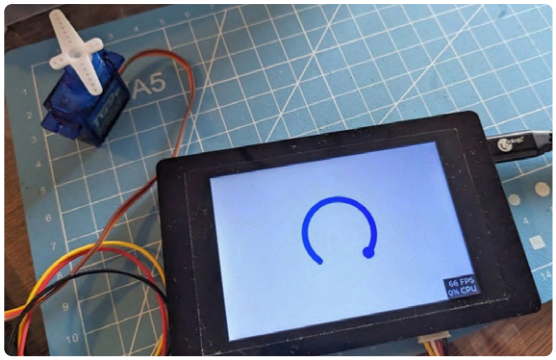


Bild 4. Der runde Schieberegler steuert die Position des Servos.

Servo-Controller

Für die Servosteuerung wollte ich das Tool Squareline Studio ausprobieren, um eine Benutzeroberfläche zu erstellen. Ein Schieberegler in Form eines Bogens ist, wie in **Bild 4** zu sehen, das einzige verwendete Grafikelement. Der Bereich des Schiebereglers wird von 0…180 Grad eingestellt, was dem Winkel des Servos entspricht.

Will man mit diesem Tool eine Benutzeroberfläche für den ESP32 entwerfen, beginnt man der Vorlage *Arduino with TFT_eSPI* und stellt die Farbtiefe auf 16 Bit und die Auflösung auf 480 × 320 ein. Der generierte Code wird in ein Verzeichnis namens *ui* in Ihrem Arduino-Ordner *libraries* exportiert und dort entpackt. Normalerweise befindet sich dieses *Libraries*-Verzeichnis in demselben Verzeichnis wie die Arduino-Sketches. Ich habe einige Zeit gebraucht, um das herauszufinden.

Der Servo wird von der ESP32Servo-Bibliothek [13] gesteuert. Die standardmäßige Arduino-Servo-Bibliothek funktioniert nicht mit dem ESP32.

Der Code ist einfach: In der Funktion `Loop` wird der Winkel des Bogenschiebers gelesen, der eine Zahl zwischen 0 und 180 zurückgibt, und dieser Wert an die Funktion `servo.write` gesendet.

Den Quellcode beider Projekte finden Sie auf der Elektor-Website [14]. Den ganzen Ordner *Libraries* habe ich nicht in den Download aufgenommen, da er (viel) zu groß ist. Es ist aber nicht kompliziert, die Abhängigkeiten selbst zu installieren.

Eine Menge Potential

Das ESP32-Terminal von Elecrow ist nicht so einfach zu zähmen, wie ich gehofft hatte, aber es bietet eine Menge Potenzial. Wenn Sie die Zeit investieren wollen, um die LVGL-Bibliothek oder eine andere UI-Bibliothek zu erkunden, und wenn Sie mit Arduino vertraut sind, dann können Sie Ihrem nächsten Projekt mit dem ESP32-Terminal eine Benutzeroberfläche mit Touch-Steuerung spendieren.

Es wäre einfacher gewesen, hätte die MicroPython-Firmware [15] auf der Produktseite zum Download zur Verfügung gestanden. Ich habe ein wenig im Internet gesucht, konnte aber nichts finden. Es ist aber sicherlich machbar, eine eigene zu kompilieren. Vielleicht eine nette Idee für Ihr nächstes Projekt…

RG — 230755-02

Haben Sie Fragen oder Kommentare?

Bei technischen Fragen können Sie sich gerne an die Elektor-Redaktion wenden: redaktion@elektor.de.

Passendes Produkt

- **ESP-Terminal (ESP32-S3 basiertes Entwicklungsboard mit kapazitivem 3,5-Zoll-TFT-Touchdisplay)**
  www.elektor.de/20526

WEBLINKS

[1] ESP32-Terminal von Elecrow: http://www.elektor.de/20526
[2] Datenblatt ESP32-S3: https://t1p.de/igbvh
[3] Crowtail-Ports: https://t1p.de/h79uh
[4] USB-nach-Seriell-Wandler CH340: https://t1p.de/dy2ch
[5] Tutorial für Arduino: https://t1p.de/u2ta4
[6] LVGL: https://lvgl.io/
[7] LVGL-Dokumentation: https://docs.lvgl.io/latest/en/html/index.html
[8] Squareline Studio: https://squareline.io/
[9] DHT-11-Bibliothek: https://github.com/adafruit/DHT-sensor-library
[10] Unified-Sensor-Bibliothek: https://github.com/adafruit/Adafruit_Sensor
[11] LCD- und E-Ink-Grafikbibliothek LovyanGFX: https://github.com/ropg/LovyanGFX
[12] Dokumentation LovyanGFX: https://lovyangfx.readthedocs.io/en/latest/index.html
[13] ESP32-Servo-Bibliothek: https://www.arduino.cc/reference/en/libraries/esp32servo/
[14] Software-Download: https://www.elektormagazine.de/news/elecrow-esp32-terminal-review
[15] Günter Spanner, „MicroPython für den ESP32 und Co. – Teil 1: Installation und erste Programme“, 2021: https://www.elektormagazine.de/articles/micropython-fur-den-esp32-und-co

Erste Schritte mit dem Zephyr RTOS

Sehr leistungsfähig, aber schwer zu beherrschen

Von Clemens Valens (Elektor)

Zephyr ist ein kleines, skalierbares Echtzeitbetriebssystem, das für ressourcenbeschränkte Controller verschiedener Architekturen optimiert ist. Das von der Linux Foundation betriebene Projekt ist eine Open-Source-Gemeinschaftsarbeit mit dem Ziel, ein erstklassiges RTOS zu entwickeln. Das Zephyr-Betriebssystem hat sich in den letzten Jahren im Bereich der eingebetteten Computer immer mehr durchgesetzt, und heute sind Hersteller neuer Mikrocontroller und Entwicklungsplatinen stolz darauf, Zephyr zu unterstützen.

Das von der Linux Foundation [1] ins Leben gerufene RTOS Zephyr ist skalierbar und eignet sich daher für eine Vielzahl von Controllern mit unterschiedlichen Ressourcenbeschränkungen. Die Skalierbarkeit wird durch eine modulare Architektur erreicht, die es den Entwicklern ermöglicht, nur die benötigten Komponenten einzubinden und so den Platzbedarf des Systems zu minimieren. Auf der Zephyr-Website wird behauptet, dass die Software auf Systemen mit nur 8 KB Speicher bis hin zu Systemen mit einem Gigabyte Speicher läuft.

Breite Hardware-Unterstützung

Zephyr unterstützt eine breite Palette von Architekturen, darunter ARM, x86, RISC-V, Xtensa, MIPS und andere, auch FPGAs mit Nios2- und MicroBlaze-Soft-Cores. Als dieser Artikel geschrieben wurde, listet Zephyr über 600 verwendbare Boards auf, darunter Arduino UNO R4 Minima, GIGA R1 WIFI und Portenta H7, mehrere Varianten des ESP32, beide Versionen des BBC micro:bit, den Raspberry Pi Pico (und sogar den Raspberry Pi 4B+), nRF51- und nRF52-Boards, die NXP MIMXRT1010-EVK und -Familie sowie die STM32-Nucleo und -Discovery-Familien. Und dies sind nur die bei Elektor üblichen Boards, aber es gibt noch viele weitere.
Neben den Prozessorplatinen unterstützt Zephyr auch viele Zusatzplatinen (so genannte Shields) und enthält Treiber für alle möglichen Schnittstellen und mehr als 150 Sensortypen.

Multitasking, Vernetzung und Energieverwaltung

Als Echtzeitbetriebssystem (RTOS) bietet Zephyr Funktionen wie präemptives Multitasking, Inter-Thread-Kommunikation und Unterstützung für Echtzeit-Timer. Das Betriebssystem verfügt außerdem über Netzwerktechnologien und -protokolle wie TCP/IP, Bluetooth und IEEE 802.15.4 (zum Beispiel Zigbee), MQTT, NFS und LoRaWAN. Zusammen mit den Netzwerkfähigkeiten eignen sich die in Zephyr integrierten Energieverwaltungsfunktionen für energieeffiziente IoT-Anwendungen und batteriebetriebene Geräte. Eine Reihe von Bibliotheken und Middleware vereinfachen gängige Aufgaben wie Kommunikationsprotokolle, Dateisysteme und Gerätetreiber. Zephyr ist auch für Sicherheitszertifizierungen wie ISO 26262 ausgelegt und sich somit bereit für sicherheitskritische Anwendungen.

Inspiriert von Linux

Zephyr ist kein Linux, nutzt aber Konzepte, Techniken und Werkzeuge, die auch Linux verwendet. So wird beispielsweise Kconfig für die Konfiguration des Betriebssystems verwendet, und die Hardwareeigenschaften und -konfigurationen werden mithilfe der Device Tree Specification (DTS) [2] beschrieben. Daher werden sich Linux-Entwickler schnell heimisch fühlen, wenn sie für Zephyr programmieren.

Offene Quelle

Nicht zuletzt wird Zephyr unter der freizügigen Lizenz Apache-2.0 veröffentlicht, die sowohl die kommerzielle als auch die nicht-kommerzielle Nutzung erlaubt. Die Benutzergemeinschaft, zu der auch Sie beitreten können, bietet Unterstützung und Dokumentation.

Ausprobieren von Zephyr

Zephyr auszuprobieren steht schon seit mehreren Jahren auf meiner To-Do-Liste, aber meine ersten Erfahrungen damit waren nicht sehr ermutigend, so dass ich mich nicht näher damit beschäftigt habe. Damals war eines der Hauptprobleme (neben der nicht fehlerfreien Kompilierung), dass man einen Programmier-Pod zur Programmierung des Ziel-Controllers benötigte, was es für

Bild 1. Das winzige Board BBC micro:bit ist ein hervorragendes Ziel, um das Zephyr-RTOS auszuprobieren. Wer hätte gedacht, dass dieses kleine Board, das 10-Jährigen das Programmieren mit MakeCode, einer Scratch-ähnlichen grafischen Sprache, beibringen sollte, auch ein hervorragendes Werkzeug für erfahrene Entwickler von eingebetteter Software sein würde, die ein industrietaugliches Echtzeitbetriebssystem erlernen möchten?

Entwickler weniger geeignet machte. Dank Arduino und seinem Bootloader haben wir uns daran gewöhnt, dass wir keine speziellen Programmiertools brauchen, und so fühlte es sich wie ein Rückschritt an, eines zu benötigen.

Auswahl eines Boards

Die Dinge haben sich seitdem weiterentwickelt. Wie bereits erwähnt, unterstützt Zephyr heute über 600 Mikrocontroller-Boards, und die Chancen stehen gut, dass Sie bereits ein oder mehrere kompatible Boards besitzen. Bei der Durchsicht der Liste habe ich festgestellt, dass ich mehr als ein Dutzend verschiedener unterstützter Boards zur Verfügung habe!

Lang lebe der BBC micro:bit!

Ich habe die meisten von ihnen ausprobiert und mich schließlich für das BBC micro:bit entschieden, um damit zu experimentieren (**Bild 1,** von Zephyr als `bbc_microbit` oder `bbc_microbit_v2` bezeichnet, je nach Version des Boards). Im Vergleich zu meinen anderen Boards ist das BBC micro:bit nicht nur leicht erhältlich, sondern hat auch die beste Zephyr-Unterstützung: Alle Peripheriegeräte sind zugänglich und werden durch einige Beispiele unterstützt, und das Beste von allem ist, dass es ohne zusätzliche Tools programmiert und debugged werden kann.

Der beliebte ESP-WROOM-32 (von Zephyr als `esp32_devkitc_wroom` bezeichnet) ist ebenfalls ein geeigneter Kandidat, aber zum Debuggen ist ein externes Tool erforderlich. Das Board Arduino GIGA R1 WIFI ist eine weitere gute Option, aber sein Bootloader wird bei der Verwendung von Zephyr abgeschossen. Sie können ihn natürlich wiederherstellen, aber das ist dennoch eine missliche Begleiterscheinung.
Offiziell benötigt das Arduino UNO R4 Minima einen SWD-fähigen Programmier-Pod (wie viele andere Boards, einschließlich des Raspberry Pi Pico), aber ich habe mit *dfu-util* einen Weg gefunden, dies zu umgehen (siehe unten). Wie beim GIGA R1 wird allerdings der Arduino-Bootloader von Zephyr zertrampelt.

Verwenden Sie einen Emulator

Falls Sie kein passendes Board haben, aber Zephyr unbedingt ausprobieren wollen, sollten Sie wissen, dass der Emulator QEMU (nur unter Linux/macOS) Zephyr unterstützt. So können Sie Anwendungen virtuell ausführen und testen. Renode von Antmicro soll zu ähnlichen Leistungen fähig sein; ausprobiert habe ich es nicht.

Installation von Zephyr

Ich habe Zephyr auf einem Computer mit Windows 11 installiert — Linux oder macOS habe ich nicht ausprobiert. Eine detaillierte und funktionierende Installationsanleitung ist online verfügbar [3]. Die einzelnen Schritte sind klar beschrieben und bedürfen keiner weiteren Erläuterung. Ich habe, wie vorgeschlagen, eine virtuelle Python-Umgebung verwendet. Das bedeutet, dass Sie sich den Befehl zum Aktivieren der virtuellen Umgebung irgendwo im PC notieren müssen, da Sie ihn jedes Mal benötigen, wenn Sie mit der Arbeit beginnen wollen. Wenn Sie die PowerShell von Windows verwenden, sollten Sie das Skript *activate.ps1* starten; in der Eingabeaufforderung ist es die Batch-Datei *activate.bat*. Windows PowerShell ist besser in der Lage, Compiler- und Linker-Ausgaben zu verarbeiten (**Bild 2**).
Zephyr besteht aus zwei Teilen, dem Betriebssystem selbst und einem SDK, das eine Sammlung von zum Zeitpunkt der Erstellung dieses Artikels 21 MCU-Toolchains enthält. Das Betriebssystem und

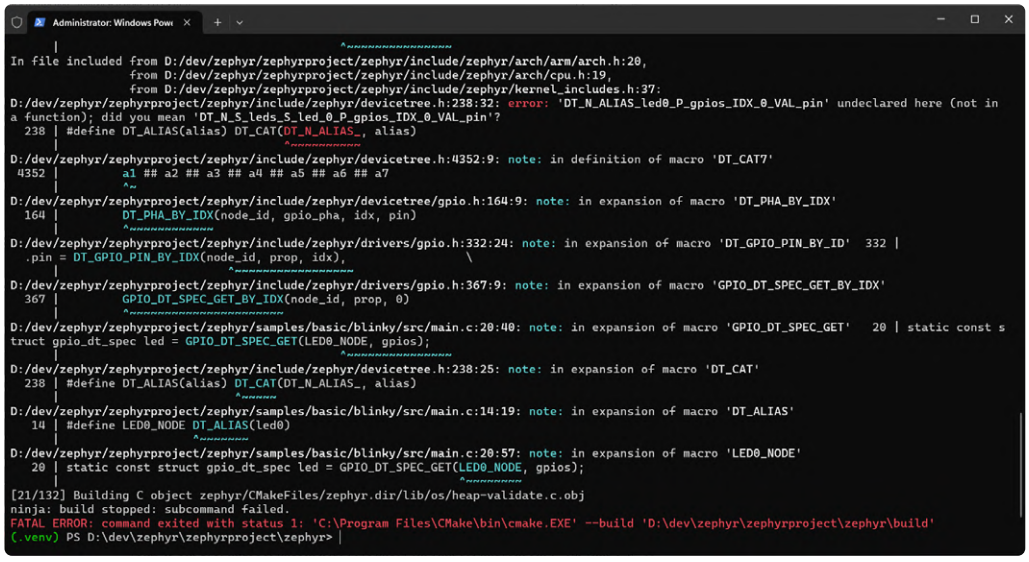


Bild 2. Das Zephyr-Build-Tool west ist für die Ausführung in einem Terminal vorgesehen. cmd.exe (Windows) kann zwar verwendet werden, ist aber kein Terminal. Die Windows-PowerShell a.k.a. Terminal ist daher besser geeignet.

Bild 3. In vielen (aber nicht allen) Situationen ist ein JTAG- oder SWD-Programmier/Debug-Pod für die Zephyr-Experimente erforderlich.

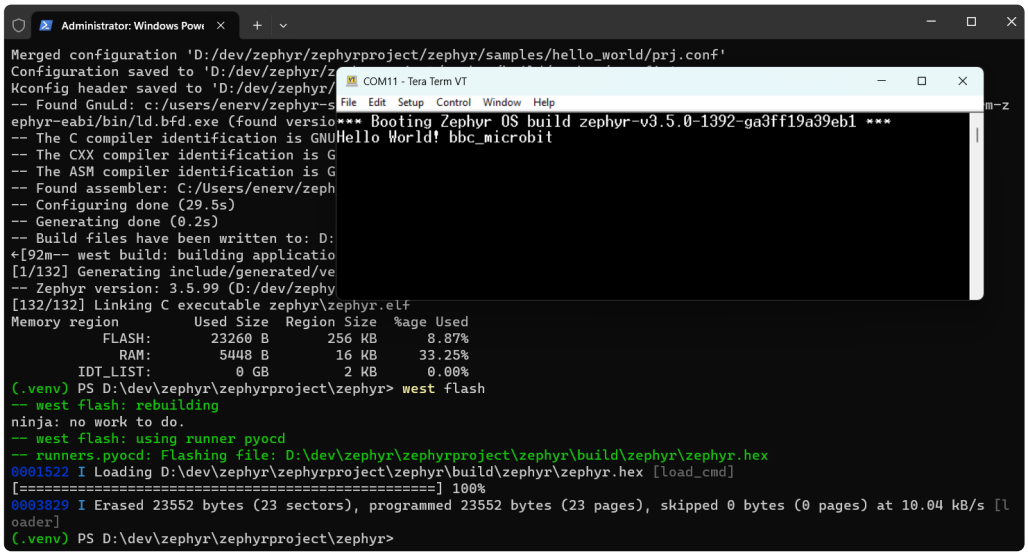


Bild 4. Das Hello-World-Beispiel ist nicht sehr umfangreich. Wenn Sie das serielle Terminal zu spät öffnen, werden Sie diese Begrüßungsnachricht nicht sehen.

das SDK müssen nicht an der gleichen Stelle installiert werden. Insgesamt verbrauchte es bei mir etwa 12 GB kostbaren Festplattenspeicherplatz. Um etwas Platz zu gewinnen, können Sie nicht benötigte Tool-Chains löschen.

Testen Sie nach der Installation, ob es funktioniert, indem Sie ein Beispiel erstellen und es mit dem folgenden Befehl auf das Board hochladen. Ersetzen Sie `<my_board>` durch den Namen Ihres Boards, zum Beispiel `arduino_uno_r4_minima`:

```
west build -p always -b <my_board> samples/basic/blinky
```

Wenn Sie den Pfad zum Beispiel nicht ändern wollen, müssen Sie diesen Befehl innerhalb des Ordners ausführen (wobei `(.venv)` anzeigt, dass Sie sich in einer virtuellen Umgebung befinden):

```
(.venv) <my_path_to_zephyr>\zephyrproject\zephyr
```

Wenn das Beispiel ohne Fehler gebaut wurde, können Sie es mit

```
west flash
```

hochladen, so dass die Standard-LED des Boards mit einer Frequenz von 0,5 Hz zu blinken beginnt.

Wie bereits erwähnt, kann zum Flashen ein externes Programmiertool erforderlich sein, zum Beispiel ein J-Link-Adapter oder ein anderes JTAG- oder SWD-fähiges Programmiergerät (**Bild 3**), und die Treibersoftware dafür muss zugänglich sein (im Suchpfad oder *%PATH%* unter Windows). Ist dies nicht der Fall, werden Sie durch eine (meist lange und kryptische) Meldung darüber informiert.

Auf dem BBC micro:bit V2 musste ich beim ersten Mal die HEX-Datei manuell mit dem Standardprogrammierverfahren für micro:bit auf das Board kopieren, danach funktionierte der Flash-Befehl aber einwandfrei. Die ausführbare Datei *zephyr.hex* befindet sich in *zephyrproject\zephyr\build\zephyr*.

Der Standard-Flash-Befehl für die Arduino-Boards UNO R4 Minima und GIGA R1 WIFI erfordert, dass sich das Programmierprogramm *dfu-util* im Suchpfad des Betriebssystems befindet (bevor Sie die virtuelle Umgebung aktivieren, falls Sie eine verwenden). Dieses Dienstprogramm ist in der Arduino-IDE enthalten. Wo genau es sich auf Ihrem Computer befindet, müssen Sie jedoch selbst herausfinden (standardmäßig in *%HOMEPATH%\AppData\Local\Arduino15\packages\arduino\tools\dfu-util\<Ihre aktuelle Arduino-IDE-Version>*). Das Board muss außerdem in den DFU-Modus versetzt werden. Dies geschieht durch zweimaliges Drücken des Reset-Knopfes. Wenn die LED anfängt zu „pulsieren“, können Sie das Programm flashen.

Blinky-Kompatibilität

Sie haben vielleicht bemerkt, dass ich für das Blinky-Beispiel den Arduino UNO R4 Minima als Board vorgeschlagen habe und nicht den BBC micro:bit, von dem ich erst ja so begeistert war. Der Grund dafür ist, dass das Board zwar 25 LEDs besitzt (die Einschaltanzeige nicht mitgerechnet), aber keine LED, die mit dem Blinky-Beispiel kompatibel ist. Der ESP Wroom 32 hat auch keine, aber der R4 Minima schon.

Das GIGA R1 ist ebenfalls Blinky-kompatibel. Die MCU auf diesem Board hat zwei Kerne (Cortex-M7 und -M4) und Zephyr lässt die Wahl offen, welchen Sie verwenden, indem Sie entweder `arduino_giga_r1_m4` oder `arduino_giga_r1_m7` für den Build-Befehl auswählen. Sie können zeigen, dass die Kerne tatsächlich unabhängig sind, indem Sie das Blinky-Beispiel zweimal flashen, einmal für den -M4 und einmal für den -M7. Die GIGA hat eine RGB-LED und Blinky verwendet unterschiedliche Farben für jeden Kern: blau für den M4 und rot für den M7. Um die beiden Blinkies deutlicher zu unterscheiden, können Sie die Blinkrate für einen der beiden ändern (ändern Sie in *samples\basic\blinky\src\main.c* den Wert von `SLEEP_TIME_MS`).

Hallo Welt!

Für Boards ohne Blinky-LED gibt es das Beispiel `hello_world`, das einen String auf der seriellen Schnittstelle ausgibt.

```
west build -p always -b <my_board> samples/hello_world
west flash
```

Dieses Beispiel funktioniert sowohl mit dem BBC micro:bit als auch mit dem ESP-WROOM-32-Modul. Um den Ausgabestring zu sehen, öffnen Sie ein serielles Terminalprogramm auf Ihrem Computer. Die Datenrate ist normalerweise 115.200 Baud (115200,n,8,1). Möglicherweise müssen Sie das Board zuerst zurücksetzen, da die Meldung nur einmal ausgegeben wird und Sie sie vielleicht übersehen haben (**Bild 4**).

Auf dem R4 Minima und dem GIGA R1 ist der serielle Ausgang der Pin 1 und nicht, wie man naiverweise erwarten könnte, der USB-C-Port, wie es in der Arduino-IDE der Fall wäre. Das liegt daran, dass der USB-Port ein Peripheriegerät der MCU ist und kein separater Chip. Da Zephyr ein modulares und skalierbares Betriebssystem ist, muss die USB-Unterstützung - wie jedes andere Modul oder Peripheriegerät - explizit für das Projekt aktiviert werden, bevor es verwendet werden kann. Dies geschieht in den Konfigurationsdateien des Projekts. Mehr zu diesen Dateien später.
Bei Boards ohne eingebauten Seriell-zu-USB-Konverter müssen Sie die serielle Schnittstelle finden (normalerweise Port 0, falls die MCU mehr als einen hat) und sie über einen externen Seriell-zu-USB-Konverter mit Ihrem Computer verbinden.

Noch ein bisschen weiter

Wenn Sie es geschafft haben, sowohl das Blinky- als auch das *hello_world*-Beispiel auf Ihrem Board zum Laufen zu bringen, sind Sie in einer ziemlich guten Position, um eine funktionierende Anwendung auf dem Zephyr zu erstellen. Wenn aber nur eines der Beispiele funktioniert und Sie möchten, dass das andere auch funktioniert, sind die Dinge etwas komplizierter.
Ich habe das BBC micro:bit als mein bevorzugtes Board für Zephyr-Experimente ausgewählt, auch wenn es nicht mit dem Blinky-Beispiel kompatibel ist. Das ist aber kein wirkliches Problem, denn das Board wird mit einigen Beispielen geliefert (im Unterordner *bbc_microbit* des Ordners *samples\boards*), von denen eines (Display) viel schöner ist als ein Blinky mit nur einer LED. Es gibt auch Beispiele für andere Boards, aber insgesamt doch nur für sehr wenige angesichts der Zahl von über 600 unterstützten Boards (nicht einmal 5 %). Außerdem betreffen die meisten dieser Beispiele fortgeschrittene oder obskure Anwendungsfälle.
Wenn Sie versuchen, Blinky für den BBC micro:bit (oder den ESP-WROOM-32 oder ein anderes inkompatibles Board) zu bauen, werden Sie eine unverständliche Fehlermeldung erhalten. Sie versucht Ihnen mitzuteilen, dass `led0` eine unbekannte Entität ist. Dies ist die standardmäßige Blinky-LED (ein bisschen wie `LED_BUILTIN` auf Arduino). Da der micro:bit einen Erweiterungsport hat, an den man LEDs und anderes anschließen kann, wollen wir versuchen, einen dieser Portpins als `led0` zu definieren.
Bevor wir dies tun, erstellen Sie eine Sicherungskopie des Ordners *samples/basic/blinky* oder eine Kopie mit einem neuen Namen und verwenden Sie diese stattdessen. Im Folgenden wird *samples/basic/blinky* verwendet.

Der Gerätebaum

Die Definition einer `led0` führt uns zum Gerätebaum (device tree), der bereits kurz erwähnt wurde. Der Gerätebaum in einer oder mehreren Textdateien listet die auf einem Board oder in einem Controller verfügbare Peripherie und den Speicher auf. In Zephyr haben diese Dateien die Endung *.dts* oder *.dtsi* („i“ für include), und die Dateien können andere Dateien enthalten. *.dtsi*-Dateien für Prozessoren befinden sich im Ordner *dts*, *.dts*- und *.dtsi*-Dateien für Boards im Ordner *boards*. Beide Ordner sind nach Prozessorarchitektur geordnet.
Um DTS(I)-Dateien auf eine etwas komfortablere Weise zu betrachten, können Sie das DeviceTree-Plugin für Visual Studio Code [4] verwenden. Dieses Plugin bietet Syntax-Highlighting

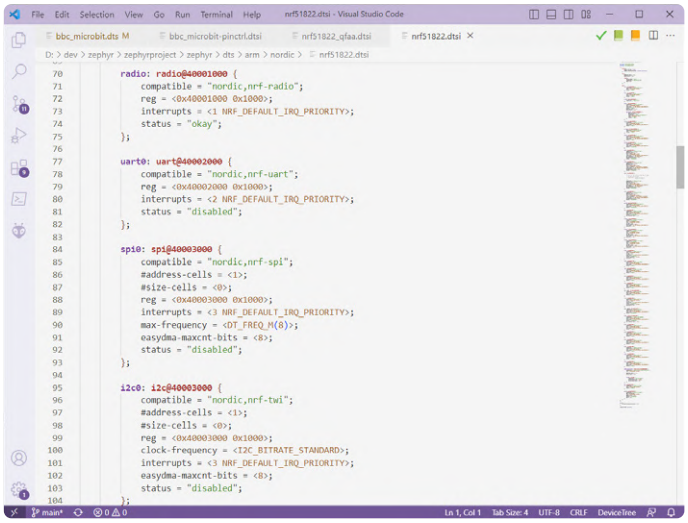


Bild 5. Ein Auszug aus der Datei nrf51822.dtsi. Die Ansicht rechts zeigt, wie lang diese Datei ist.

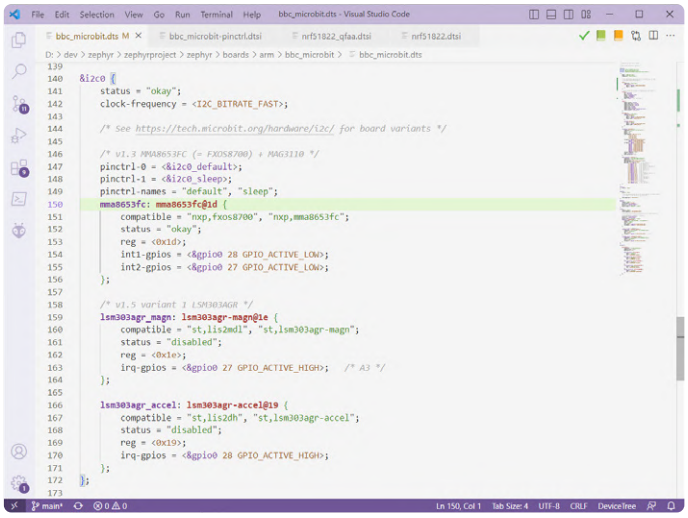


Bild 6. Dieser Abschnitt des Gerätebaums, entnommen aus der Datei bbc_microbit.dts, stellt den I²C-Bus des BBC-micro.bit-Boards dar, nicht den seines Prozessors.

und Codefaltung, wodurch die Dateien leichter zu lesen sind (DTS-Dateien verwenden eine Syntax im Stil der Sprache C). **Bild 5** zeigt einen Auszug aus der *.dtsi*-Datei für den nRF51822, der das Herzstück des Boards BBC micro:bit V1 ist. Diese Datei ist in der DTS-Datei des Boards enthalten. Sie sollten beispielsweise beachten, dass der Status von `uart0` auf `disabled` gesetzt ist. Dieser Status wird in der DTS-Datei des Boards überschrieben, wo er auf `okay` gesetzt wird, was bedeutet, dass er verwendet werden kann. Das gleiche gilt für `gpio0` und `i2c0`.

I²C im Gerätebaum

Ein weiterer Schnipsel aus der *.dts*-Datei für den BBC micro:bit ist in **Bild 6** zu sehen. Er zeigt den Gerätebaum für den I²C-Bus. micro:bit verfügt über einen oder zwei Sensoren, die an den Bus angeschlossen sind (je nach Boardvariante) und im Baum durch `mma8653fc` und `lsm303agr` repräsentiert werden (letzterer umfasst zwei Sensoren, weshalb er zweimal im Baum erscheint). Der erste hat den Status `okay`, während die beiden anderen `disabled` sind. Dies ist korrekt für meine Boardvariante, die zur allerersten micro:-bit-V1-Generation zählt.
Wie das Snippet zeigt, ist dieser Sensor mit dem FXOS8700 und dem MMA8653FC kompatibel, seine Adresse auf dem I²C-Bus ist 0x1d, und es sind zwei `int`-Signale (Interrupt) deklariert, die mit

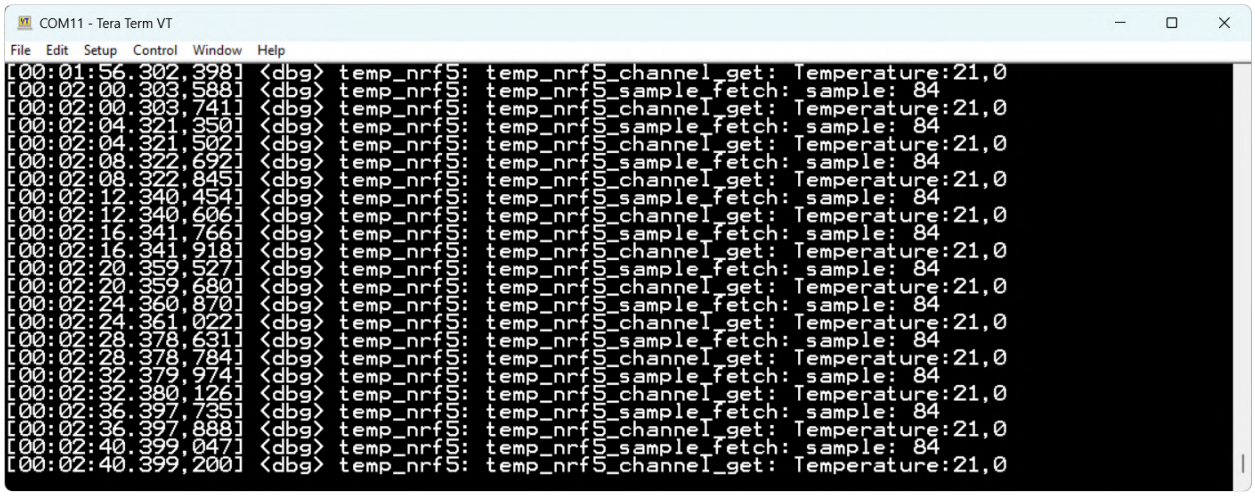


Bild 7. Die Ausgabe der samples/sensor/fxos8700-Demo, die auf dem BBC micro.bit läuft.

GPIO-Pin 27 und GPIO-Pin 28 verbunden sind. Wenn Sie es ausprobieren möchten, steht Ihnen ein Demoprogramm zur Verfügung:

```
west build -p always -b bbc_microbit samples/sensor/
  fxos8700
west flash
```

Beachten Sie, dass dies nicht mit dem BBC micro:bit V2 funktioniert, da dieser einen anderen Sensor in seinem Gerätebaum hat. Die Ausgabe der Demo ist in **Bild 7** zu sehen, aber wir schweifen ab.

Überlagern des Gerätebaums

Zurück zu unserer LED, `led0`. Im Gerätebaum des Boards wird `led0` erwartungsgemäß nicht erwähnt, also müssen wir sie hinzufügen. Wir könnten sie direkt in die Gerätebaumdatei des Boards eintragen, aber das wäre nicht korrekt, da das Board keine `led0` hat. Der richtige Weg, einen Gerätebaum zu erweitern, ist das Hinzufügen eines Overlays. Der Inhalt einer Overlay-Datei wird in den Gerätebaum eingefügt. Im Baum vorhandene Abschnitte werden erweitert (im Falle eines neuen Elements) oder überschrieben (falls das Element bereits im Baum vorhanden ist); neue Abschnitte werden hinzugefügt.

Overlays müssen innerhalb des Projektordners in einem Unterordner mit dem Namen *boards* abgelegt werden. Wenn dieser Unterordner vorhanden ist, sucht der Erstellungsprozess darin nach einer Datei mit dem Namen *<my_board>.overlay*. In meinem Fall lautet der Dateiname *bbc_microbit.overlay* (*bbc_microbit_v2.overlay* für V2-Anwender). **Bild 8** zeigt den Inhalt der Datei.

Hinzufügen einer Blinky-LED

Zephyr hat ein spezielles Gerätebaum-Schlüsselwort für LEDs, nämlich `leds`, also erstellen wir einen Knoten (branch) dafür. Sie können ihn nennen, wie Sie wollen, aber bleiben Sie bei `leds`, wenn er einem bestehenden `leds`-Knoten überlagert werden soll. Dieser Knoten soll zum Root des Gerätebaums hinzugefügt werden; daher wird ihm ein Schrägstrich „/“ vorangestellt, da dies in der DT-Sprache Root bedeutet. Die nächste Zeile besagt, dass dieser Zweig mit dem in Zephyr integrierten `gpio-leds`-Treiber kompatibel ist. Die Schnittstelle für diesen Treiber finden Sie in *zephyr\include\zephyr\drivers\led.h*.

Child-Knoten

Als nächstes kommt eine Liste von LED-Child-Knoten. Da ich nur eine LED habe, gibt es nur den einen Kindknoten `led_0`, den ich mit `led0` bezeichnet habe. Die Bezeichnung eines Knotens ist zwar optional, aber sie ermöglicht es, den Knoten an anderer Stelle im Baum zu referenzieren, was wir ein paar Zeilen weiter unten auch tun werden. Außerdem können sie von der Anwendung (dem Entwickler) verwendet werden, um Zugriff auf Knoten und deren Eigenschaften zu erhalten.

Ein untergeordneter Knoten muss die Eigenschaften des Geräts angeben. Im Falle einer LED ist nur der GPIO-Pin eine erforderliche Eigenschaft, aber die optionale Eigenschaft namens `label` kann hinzugefügt werden. Solche Labels können zur Dokumentation verwendet werden oder um für Menschen lesbare Informationen für die Anwendung bereitzustellen. Labels haben darüber hinaus keine andere Funktion.

Als GPIO-Pin habe ich `1` gewählt, was dem großen Loch/Pad 2 auf dem micro:bit-Erweiterungsanschluss entspricht. Wenn Sie einen BBC micro:bit V2 haben, verwenden Sie (anstelle von `1`) `4` als GPIO-Pin.

Einen Alias erstellen

Der nächste Schritt ist notwendig, weil das Blinky-Beispiel ihn erwartet. Er besteht darin, den `led0`-Alias für unsere LED zu erstellen. Man könnte meinen, dass es ausreicht, den Kindknoten

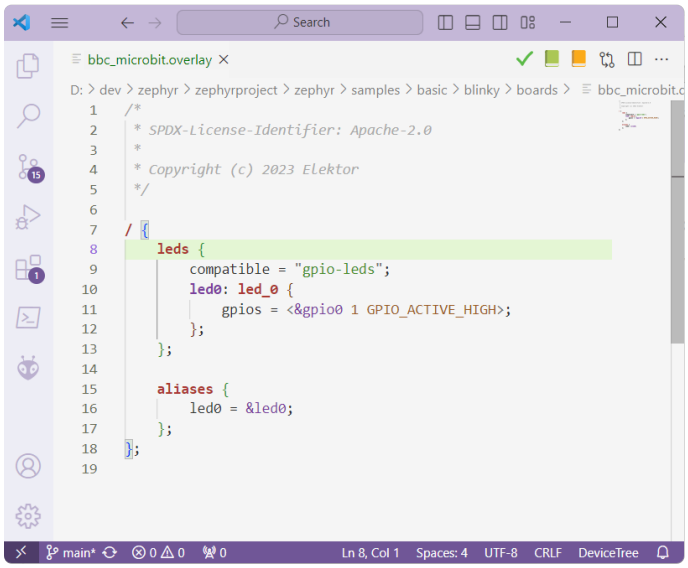


Bild 8. Diese Gerätebaum-Overlay-Datei macht den BBC micro.bit V1 kompatibel mit dem Blinky-Beispiel.

zu „belabeln“, aber das ist nicht der Fall, denn Blinky verwendet das Makro `DT_ALIAS`, um Zugriff auf den LED-Kindknoten zu erhalten. Daher müssen wir etwas bereitstellen, das dieses Makro verdauen kann, was in diesem Fall ein Alias ist. Es befindet sich innerhalb des Blocks `aliases`. Hätte Blinky stattdessen das Makro `DT_NODELABEL` verwendet, dann wäre ein Alias überflüssig gewesen, da `DT_NODELABEL` das Label des `led0`-Kinderknotens direkt übernimmt. Ich weiß, dass es etwas verwirrend ist, Labels und Aliase mit demselben Namen zu haben, aber es ist für meine Erklärung notwendig.

Zephyr-Makros

Auch wenn Makros in der C/C++-Programmierung verpönt sind, werden sie in Zephyr häufig eingesetzt. Makros wie `DT_ALIAS` und `DT_NODELABEL` ermöglichen es den Werkzeugen zur Anwendungs- und Projektkonfiguration, Informationen aus dem Gerätebaum zu extrahieren, und es gibt sie in Hülle und Fülle. Sie können die Beschreibungen der Makros im Zephyr-Handbuch im Kapitel „Devicetree API“ [5] finden.

Interessant ist die Tatsache, dass viele (alle?) Zephyr-Makros erwarten, dass ihre Argumente klein geschrieben werden, wobei alle Zeichen, die keine Buchstaben (a bis z) oder Zahlen (0 bis 9) sind, durch Unterstriche ersetzt werden. Dies wird als „lowercase-and-underscores-kompatibel“ bezeichnet. Stellen Sie sich zum Beispiel vor, ich hätte den LED-Unterknoten von vorhin mit `LED-0` statt `led0` bezeichnet. Dann wäre das Argument für `DT_NODELABEL led_0` gewesen, das heißt `DT_NODELABEL(led_0)`, denn der Bindestrich ist weder ein Buchstabe noch eine Zahl, und Buchstaben müssen klein geschrieben werden. Mit anderen Worten: Für den Entwickler der Anwendung, der Makros für den Gerätebaum verwendet, ist der Unterstrich ein Platzhalter. So kann sich `led_0` in der Anwendung entweder auf `led_0`, `led-0`, `Led_0`, `LED-0` und `ledé0` (und jede andere Variation, die Sie sich ausdenken können) im Gerätebaum beziehen. In diesem Sinne ist es sehr empfehlenswert, die Dokumentation der Zephyr-Makros sorgfältig zu lesen!

Beachten Sie, dass Fehler im Gerätebaum dadurch bestraft werden, dass der Compiler Sie mit „FATAL ERROR“ anschreit, ohne weitere Informationen zu liefern.

Makellose Builds

Wenn Sie mit dem Device-Tree oder Ihrer Anwendung herumspielen, werden Sie Ihr Projekt wahrscheinlich häufig neu erstellen (müssen). Um die Dinge etwas zu beschleunigen, entfernen Sie das `-p always` („p“ von pristine, makellos) aus dem Build-Befehl. Dadurch wird verhindert, dass alles von Grund auf neu erstellt wird. Wenn Sie hingegen viele verschiedene Beispiele nacheinander ausprobieren, sollten Sie den Befehl beibehalten, da er Ihnen eine lästige Fehlermeldung erspart, dass der Build-Ordner nicht für Ihr Projekt bestimmt ist.

Beachten Sie, dass der Flash-Befehl auch den letzten Build-Befehl auslöst, so dass es ausreicht, nur den Flash-Befehl jedes Mal auszuführen, wenn Sie etwas ändern.

Einen Device-Treiber verwenden

Das Blinky-Beispiel ruft die Funktion `gpio_pin_toggle_dt()` auf, um den Zustand der LED umzuschalten. Dies ist eine Funktion des GPIO-Treibers. Das ist natürlich völlig in Ordnung, aber Zephyr enthält auch eine Sammlung von LED-Treibern. Die Verwendung eines LED-Treibers macht das Programm nicht nur lesbarer, sondern auch flexibler und portabel, da ein LED-Treiber durch einen anderen ersetzt werden kann, ohne die Anwendung selbst zu ändern. Hier kommen die Skalierbarkeit und Modularität von Zephyr ins Spiel.

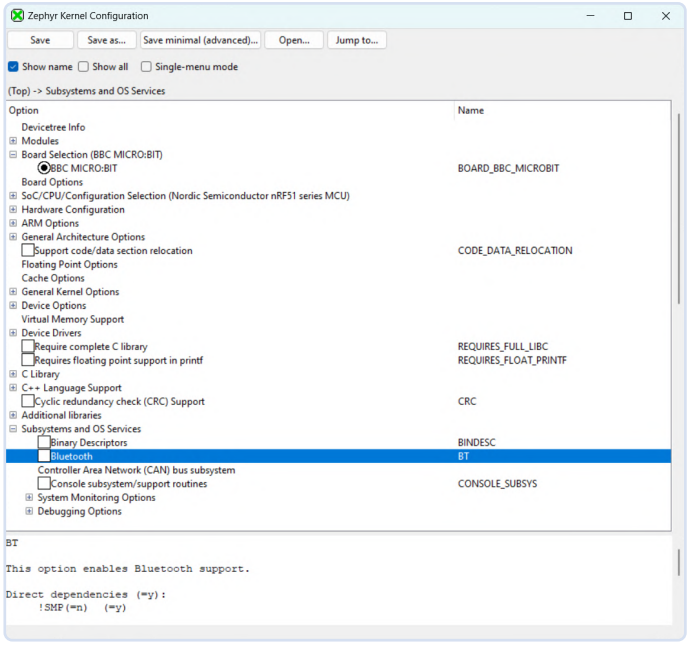


Bild 9. Die Benutzeroberfläche des Kconfig-Projektkonfigurationswerkzeugs. Es gibt viele Optionen.

Kconfig hat eine GUI

Die Einbindung eines LED-Treibers in unser Programm erfordert einige Schritte. Zunächst muss das Projekt neu konfiguriert werden, damit es den Treiber enthält. Die Projektkonfiguration wird von Kconfig vorgenommen, dem Kernel-Konfigurations-System, das auch von Linux verwendet wird. Es gibt mehrere Möglichkeiten, mit diesem System zu interagieren, und eine davon ist eine grafische Benutzeroberfläche (GUI). In Zephyr öffnen Sie sie wie folgt:

```
west build -t guiconfig
```

Es dauert eine Weile, bis sich die grafische Benutzeroberfläche öffnet, aber dann sieht sie aus wie in **Bild 9**. Die Oberfläche zeigt eine Menge Informationen über das in Entwicklung befindliche Projekt. Beachten Sie den *Project-under-development*-Teil. Damit Kconfig tatsächlich an Ihrem Projekt arbeitet, führen Sie einen ursprünglichen Build (mit dem Flag `-p always`) Ihres Projekts durch, bevor Sie das Kconfig-GUI starten.

So viele Optionen...

Nehmen Sie sich etwas Zeit, um den Konfigurationsbaum zu erkunden. Klappen Sie Zweige auf, indem Sie auf die +-Symbole klicken. Erforschen Sie die Optionen, indem Sie darauf klicken. Dadurch werden einige Informationen im unteren Bereich angezeigt. Beachten Sie, dass die Fließkomma-Unterstützung für `printf()` eine Konfigurationsoption ist, ebenso wie die Unterstützung der Sprache C++. In ähnlicher Weise finden Sie unter *Build and Link Features* Optionen zur Optimierung des Compilers. Es gibt eine Unmenge von Konfigurationsoptionen. Die für uns interessanten befinden sich im Zweig *Device Drivers*. Klappen Sie ihn auf und scrollen Sie nach unten, während Sie sich alle verfügbaren Optionen ansehen. Der LED-Treiber befindet sich etwa auf

halber Strecke nach unten: *Light-Emitting Diode (LED) drivers*. Aktivieren Sie das Kontrollkästchen und belassen Sie die Optionen im Unterzweig auf ihren Standardwerten (**Bild 10**). Klicken Sie auf den *Save*-Knopf und beachten Sie den Pfad der Konfigurationsdatei, der am unteren Rand des Fensters angegeben ist. Es ist sehr aufschlussreich, diese Datei zu inspizieren, um zu sehen, was sie enthält (eine Menge). Schließen Sie das GUI.

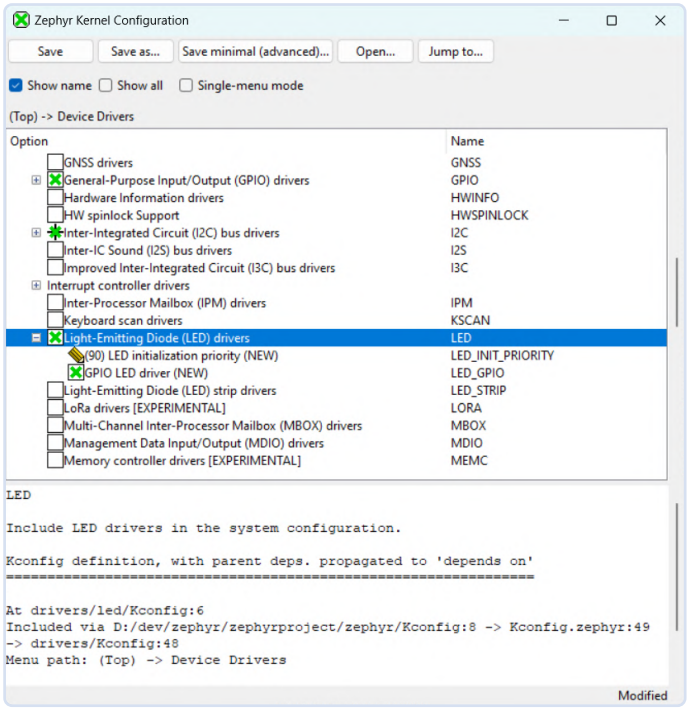


Bild 10. Wählen Sie Light-Emitting Diode (LED) drivers und lassen Sie die Child-Werte unverändert.

```c
#include <zephyr/kernel.h>
#include <zephyr/device.h>
#include <zephyr/drivers/led.h>

#define SLEEP_TIME_MS   (1000) /* 1000 msec = 1 sec */

int main(void)
{
    printf("\nBlinky with LED device driver\n");

    const struct device *const led_device = DEVICE_DT_GET_ANY(gpio_leds);
    if (!led_device)
    {
        printf("No device with compatible 'gpio-leds' found\n");
        return 0;
    }
    else if (!device_is_ready(led_device))
    {
        printf("LED device %s not ready\n", led_device->name);
        return 0;
    }

    while (1)
    {
        int result = led_on(led_device,0);
        if (result<0)
        {
            printf("led_on failed\n");
            return 0;
        }
        k_msleep(SLEEP_TIME_MS);

        result = led_off(led_device,0);
        if (result<0)
        {
            printf("led_off failed\n");
            return 0;
        }
        k_msleep(SLEEP_TIME_MS);
    }

    return 0;
}
```

Bild 11. Das Blinky-Programm wurde für einen LED-Treiber umgeschrieben. Beachten Sie, dass es keinen board-spezifischen Code gibt. Dieses Programm sollte auf jedem Board laufen, das einen Treiber `gpio_leds` *(oder* `gpio-leds`*) in seinem Gerätebaum hat.*

Von nun an sollten Sie in den Build-Befehlen nicht mehr das Flag `-p always` angeben, da es die oben vorgenommenen Änderungen rückgängig machen würde. Ich werde Ihnen gleich zeigen, wie man die Konfigurationsänderung dauerhaft macht.

Blinky mit LED-Treiber

Jetzt können wir das neue Blinky-Programm schreiben - siehe **Bild 11**. Es beginnt mit der Einbindung der Header-Dateien für das Gerät und den LED-Treiber. Dann verwenden wir in der Main-Datei das Makro `DEVICE_DT_GET_ANY`, um eine Gerätereferenz für die LED aus dem Gerätebaum zu erhalten. Beachten Sie, dass das Argument `gpio_leds` des Makros kleingeschrieben und unterstrichkompatibel ist, so dass es mit dem `gpio-leds`-Wert der Eigenschaft `compatible` des `leds`-Knotens im Gerätebaum übereinstimmt (wie oben erklärt). Wenn es kein Gerät findet, weil Sie sich vertippt haben, gibt das neue Blinky die Meldung „No device with compatible gpio-leds found“ aus. Diese Meldung wird auch ausgelöst, wenn die Eigenschaft `status` eines Geräts auf `disabled` gesetzt wird.

Die Verwendung des Wortes „compatible“ als Substantiv in Zephyr ist etwas gewöhnungsbedürftig. Daher bedeutet die obige Fehlermeldung nicht, dass es kein kompatibles Gerät gibt, sondern dass es kein Gerät gibt, das eine Eigenschaft mit dem Namen `compatible` mit dem Wert `gpio-leds` hat (und auch nicht mit `gpio_leds`, denken Sie daran, dass der Unterstrich jedes Zeichen außer a...z und 0...9 ersetzt).

Eine zweite Prüfung stellt fest, ob das Gerät ordnungsgemäß initialisiert wurde. Wenn dies der Fall ist, fahren wir fort.

In der Endlosschleife `while()` wird die LED mit den vom Treiber [6] bereitgestellten Befehlen `led_on` und `led_off` ein- und ausgeschaltet. Das Argument `0` steht für das erste (und einzige) Gerät, das vom Makro `DEVICE_DT_GET_ANY` gefunden wurde, nämlich `led0`.

Rückgabewerte prüfen

Da wir einen Gerätetreiber verwenden, anstatt einen GPIO-Pin direkt auf der Ebene der Hardware-Register umzuschalten, ist es eine gute Praxis, die Rückgabewerte aller Funktionsaufrufe des Treibers zu überprüfen, da sie aus irgendeinem Grund fehlschlagen können. Ein Treiber muss bestimmte Funktionen und Rückrufe bereitstellen, kann aber auch optionale Eigenschaften haben. Ein LED-Treiber muss zum Beispiel `led_on` und `led_off` implementieren, `led_blink` ist jedoch optional. Wenn Sie `led_blink` in unserem Projekt aufrufen, wird nichts passieren, weil es nicht implementiert ist. Es existiert zwar, aber es ist leer. Der Rückgabewert zeigt Ihnen das an. Im Allgemeinen ist es eine gute Programmierpraxis, den Rückgabewert jedes Funktionsaufrufs zu überprüfen.

Führen Sie ein Build des Programms durch und laden Sie es hoch (beachten Sie, dass das Flag `-p always` nicht vorhanden ist).

```
west build -b bbc_microbit samples/basic/blinky
west flash
```

Konfigurieren des Projekts

Wenn die LED mit 0,5 Hz zu blinken beginnt, haben wir ein funktionierendes Programm. Damit das so bleibt, müssen wir die aktuelle

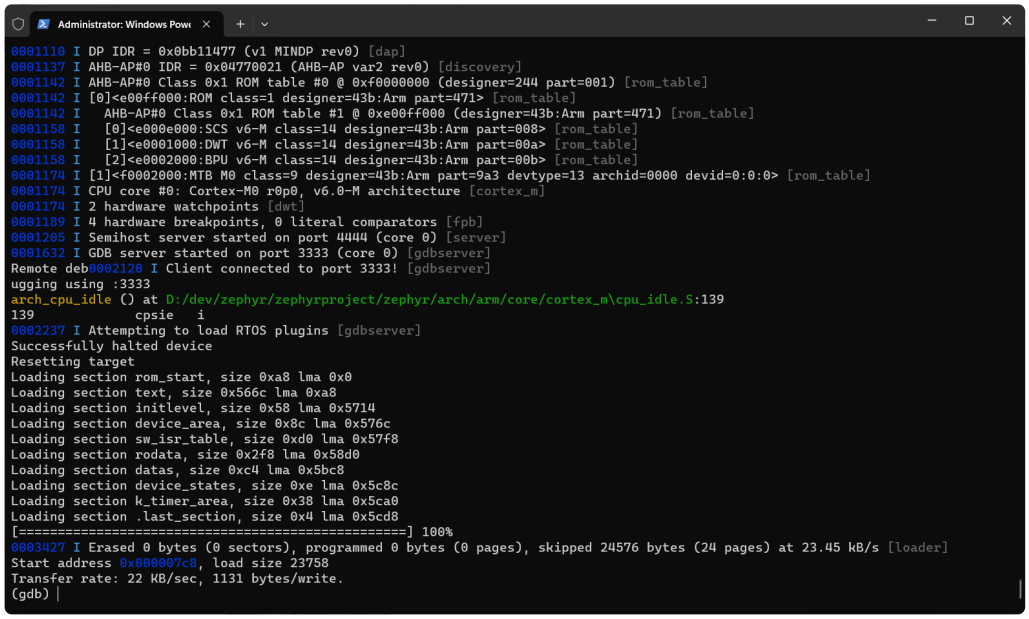


Bild 12. Das Board BBC micro. bit unterstützt das Debuggen mit gdb von Haus aus, so dass keine zusätzlichen Tools benötigt werden.

Konfiguration dauerhaft machen. Dazu öffnen wir die Datei *prj.conf* im Ordner unseres Blinky und fügen die folgende Zeile ein (im Gegensatz zu Gerätebaumdateien, die eine C-Syntax verwenden, verwenden Kconfig-Konfigurationsdateien eine Python-Syntax):

```
CONFIG_LED=y
```

Um zu überprüfen, ob es funktioniert, führen Sie einen Pristine-Build Ihres Projekts aus und laden Sie die ausführbare Datei auf das Board.

Debugging

Wenn Ihr Board es zulässt (wie das BBC micro:bit) oder Sie ein geeignetes Debugging-Tool haben, können Sie die Anwendung debuggen mit

```
west debug
```

Dadurch wird ein gdb-Server gestartet und ein Terminal geöffnet (**Bild 12**). Konsultieren Sie das Internet, um zu lernen, wie man mit gdb arbeitet, da dies den Rahmen dieses Artikels sprengen würde.

Zephyr vs. Arduino

Nachdem Sie nun gesehen haben, wie man mit dem Zephyr-Betriebssystem arbeitet, fragen Sie sich vielleicht, warum Sie es verwenden sollten. Ist es nicht viel einfacher, Arduino zu verwenden? Wie Zephyr unterstützt auch Arduino mehrere Prozessorarchitekturen und Hunderte von Platinen. Tausende von Treibern und Bibliotheken sind für Arduino geschrieben worden. Wenn eine Anwendung oder Bibliothek die Arduino-Core-API verwendet, kann sie, genau wie eine Zephyr-Anwendung, leicht auf jede andere unterstützte Plattform mit ähnlicher Peripherie portiert werden. Beides ist Open Source. Und nun?

Nun, Zephyr ist als industrietaugliches, robustes RTOS mit Funktionen wie Aufgabenplanung, Speicherverwaltung und Gerätetreiber gedacht. Darüber hinaus ist Zephyr für ein breites Spektrum an Schwierigkeitsgraden der Projekte ausgelegt, von kleinen IoT-Geräten bis hin zu komplexen eingebetteten Systemen. Es bietet mehr Flexibilität, erfordert aber ein tieferes Verständnis der Embedded-Entwicklung.

Arduino bietet zwar einige Echtzeitfähigkeiten, ist aber kein RTOS, sondern ein Framework für Single-Thread-Anwendungen mit dem Schwerpunkt auf Einfachheit und Benutzerfreundlichkeit. Arduino abstrahiert viele Low-Level-Details, wodurch es auch für Anfänger zugänglich ist. Für komplexere Anwendungen kann es jedoch zusätzlich zu einem RTOS wie Mbed OS verwendet werden. Es wird daran gearbeitet, die Arduino-Core-API auf Zephyr lauffähig zu machen [7].

Es liegt an Ihnen zu entscheiden, ob Sie Zephyr für Ihr nächstes Projekt benötigen oder nicht. Sie könnten es zumindest ausprobieren, denn dies macht sich gut im Lebenslauf eines jeden Embedded-Entwicklers.

Weitere Lektüre

Das war's für den Moment. Wie Sie gesehen haben, ist das Zephyr-Betriebssystem kompliziert und hat eine etwas steile Lernkurve. In diesem Artikel habe ich versucht, den Aufstieg ein wenig zu erleichtern. Es gibt jedoch noch viel, viel mehr über Zephyr zu sagen und zu lernen, also denken Sie nicht, dass Sie jetzt schon alles wissen. Die Verweise [8] und [9] enthalten Links zu zwei Themen, die Sie interessieren könnten.

RG — 230713-02

Haben Sie Fragen oder Kommentare?

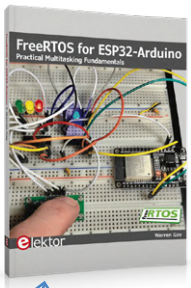
Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Schicken Sie eine E-Mail an den Autor unter clemens.valens@elektor.com oder kontaktieren Sie das Elektor-Team unter redaktion@elektor.de.

Über den Autor

Nach einer Karriere in der Schiffs- und Industrieelektronik begann Clemens Valens 2008 als Chefredakteur von Elektor Frankreich für Elektor zu arbeiten. Seitdem hatte er verschiedene Positionen inne und wechselte vor kurzem in die Abteilung Produktentwicklung. Seine Hauptinteressen sind Signalverarbeitung und Klangerzeugung.

Passende Produkte

- **BBC micro:bit V2**
  www.elektor.de/19488
- **Get Started with the NXP i.MX RT1010 Development Kit**
  (Buch und Entwicklungsboard) www.elektor.de/20699
- **Warren Gay, *FreeRTOS for ESP32-Arduino*, Elektor 2020**
  Buch, Paperback, englisch: www.elektor.de/19341
  E-Buch, PDF, englisch: https://www.elektor.de/freertos-for-esp32-arduino-e-book



Elektor Webinars

MÄRZ 28 2024 16:00 MEZ

Besuchen Sie unser Zephyr-Webinar
für eine praktische Einführung in IoT-Projekte

Details unter
elektormagazine.com/webinars

WEBLINKS

[1] Über das Zephyr-Projekt - Ein Projekt der Linux Foundation: https://zephyrproject.org/learn-about
[2] Gerätebaum-Spezifikationen: https://devicetree.org/specifications
[3] Erste Schritte mit Zephyr: https://t1p.de/turgj
[4] Devicetree, Syntax-Highlighting-Plugin für Visual Code Studio: https://github.com/plorefice/vscode-devicetree
[5] Devicetree-API: https://docs.zephyrproject.org/latest/build/dts/api/api.html
[6] LED-API: https://docs.zephyrproject.org/latest/hardware/peripherals/led.html
[7] Arduino unter Zephyr betreiben: https://dhruvag2000.github.io/Blog-GSoC22
[8] Whitepaper von Eli Hughes: „Vom Hardware-Konzept zum Zephyr-Update“: https://zephyrproject.org/white-papers
[9] Ausgabe einer Zephyr-RTOS-Anwendung über die serielle Konsole (USB): https://www.gnd.io/zephyr-console-output

Preisgekrönte Ethik

Ein Gespräch mit CTO Alexander Gerfer von Würth Elektronik eiSos, Innovation und achtsames Verhalten zu ermöglichen

Fragen von Shenja Panik
Ethics in Electronics / Elektor

Alexander Gerfer, Technischer Direktor von Würth Elektronik eiSos, spricht in einem exklusiven Interview mit Shenja Panik über sein Engagement für Ethik und Innovation. Das Interview behandelt die Initiativen des Hightech-Innovationszentrums für Umweltverantwortung, Fairness und Berechenbarkeit und zeigt auf, wie Ethik mit dem geschäftlichen Erfolg von Würth Elektronik verbunden ist.

Shenja Panik: Herzlichen Glückwunsch, Herr Gerfer, zum Gewinn des *Ethics in Electronics Award 2023* für Ihren unermüdlichen Einsatz für Nachhaltigkeit, Grüne Technologie und Vertikale Landwirtschaft. Vielen Dank, dass Sie uns heute in Ihr Hightech-Innovationszentrum in München eingeladen haben und dass Sie sich die Zeit genommen haben, einige Fragen zu beantworten. Können Sie bitte Ihre Rolle auf dem Gebiet der Ethik in der Elektronik beschreiben?
Alexander Gerfer: Um das zu erklären, ist es wichtig, meinen Hintergrund zu kennen: Wenn man auf einem Bauernhof mit Kühen und Hühnern aufwächst, muss man schon sehr früh Verantwortung übernehmen. Ein weiterer entscheidender Aspekt ist das Sprichwort „Was du nicht willst, das man dir tu, das füg' auch keinem andern zu". Das sind die Grundlagen meiner Erziehung. Bei Würth Elektronik steht der Kunde immer im Mittelpunkt. Wir behandeln unsere Kunden nicht nach Unternehmensgröße oder Bedeutung, sondern pflegen mit allen ein sehr partnerschaftliches Verhältnis. Die Kunden profitieren von unserem Know-how durch Beratung, Musterdesigns, Online-Präsentationen, Entwicklungskits und Software-Tools, die bei der Dimensionierung helfen. Dafür oder für Labormuster unserer Bauteile verlangen wir kein Geld. Für uns ist jeder gleich wichtig. Wir unterstützen jeden, der Wissen über Auswahl und Anwendung benötigt. Wir beliefern und unterstützen jeden, vom Existenzgründer bis zum etablierten Unternehmen, vom Kleinmengenbesteller bis zum Großserienabnehmer. Bei uns gibt es keine Mindestbestellmengen. Das liegt an unserer Unternehmenskultur. Als Start-up kann man während der Entwicklung an einer Cent-Komponente scheitern. Das weiß ich aus eigener Erfahrung. Deshalb sind wir immer bestrebt, Probleme schnell zu lösen - und geben nicht dem den Vorzug, der den größten Umsatz verspricht.

Shenja Panik: Nicht jeder hat die gleichen Chancen als Start-up, deshalb ist es toll, dass Würth Elektronik aktiv daran arbeitet, die Chancengleichheit zu erhöhen. Welche weiteren Initiativen oder Strategien werden in München umgesetzt, um nachhaltiges und ethisches Wirtschaften zu fördern?
Alexander Gerfer: In unserem neuen Hightech-Innovationszentrum in München prüfen wir Prototypen elektronischer Geräte und Baugruppen auf ihre elektromagnetische Verträglichkeit und machen

gezielt Verbesserungsvorschläge. Denn die EMV-Zertifizierung ist für Entwickler oft eine große Hürde.
Fairness, Berechenbarkeit, Gleichberechtigung, gegenseitige Unterstützung - das sind die Grundsätze für die Führung im Unternehmen und für den Umgang der Mitarbeiter untereinander.
Als Teil der Würth-Gruppe tragen wir auch aktiv zum Erhalt unserer Umwelt bei, sowohl durch unsere Produkte als auch durch umweltfreundliche Geschäftspraktiken. Heutzutage ist es wichtiger denn je, sich um die Umwelt zu kümmern.
Wir denken immer einen Schritt voraus. Nehmen Sie zum Beispiel Vertical Farming: Der Anbau von Lebensmitteln in mehrstöckigen Gewächshäusern leistet bereits einen Beitrag zur lokalen Erzeugung hochwertiger Lebensmittel. Dieser Beitrag muss aber noch deutlich ausgeweitet werden. Nach Schätzungen der Ernährungs- und Landwirtschaftsorganisation der Vereinten Nationen (FAO) werden im Jahr 2050 10 Milliarden Menschen auf der Erde leben. Dies erfordert eine Steigerung der weltweiten landwirtschaftlichen Produktion um 50 % bis zum Jahr 2050. Die landwirtschaftliche Nutzfläche ist jedoch in den letzten Jahrzehnten zurückgegangen, von fast 40 % der weltweiten Landfläche im Jahr 1991 auf nur noch 37 % im Jahr 2018. Die Pflanzen werden buchstäblich in den Himmel wachsen müssen, wenn wir sicherstellen wollen, dass sich alle Menschen gut ernähren können. Der Klimawandel und die demografische Entwicklung stellen uns vor enorme Herausforderungen, und wir müssen jetzt an Lösungen arbeiten - auch wenn diese zunächst als ineffizient und „zu teuer“ empfunden werden.
An guten Ideen, wie man unsere Welt besser machen kann, mangelt es nicht. Doch jede bedeutende Idee braucht heute Elektronik für ihre Umsetzung. Hier leisten wir unseren Beitrag als zuverlässiger Lieferant, aber vor allem als Unterstützer kleiner und mittlerer Unternehmen und ambitionierter Start-ups. Obwohl wir als Unternehmen natürlich Gewinne erwirtschaften müssen, steht bei Würth Elektronik nicht der kurzfristige Gewinn, sondern die Sinnhaftigkeit des Projektes im Vordergrund.

Bild 1. Übergabe des Ethics in Elecronics Award 2023 an den Gewinner Alexander Gerfer.

Shenja Panik: Was bedeutet diese Auszeichnung für Sie und Ihr Team?
Alexander Gerfer: Für mich persönlich ist die Auszeichnung eine ganz besondere Ehre. Wir haben ihn auch mit dem Team geteilt, denn ich mache das nicht alleine. Es ist eine Ehre für das ganze Team, das an diesen Projekten arbeitet - aber vor allem ist es ein Ansporn. Es wäre wohl eine Übertreibung zu behaupten, dass die Welt in letzter Zeit nur besser geworden ist. Leider gibt es in vielen Bereichen Zeichen der Konfrontation, und leider werden auch die grundlegendsten ethischen Prinzipien missachtet. Umso wichtiger ist es, dass wir unseren Prinzipien treu bleiben, selbstlos helfen, gemeinsam handeln, respektvoll miteinander umgehen, unseren Lebensraum bewahren und Innovationen zum Wohle der Allgemeinheit aktiv vorantreiben. Wir alle stehen vor großen Herausforderungen. Wir haben enorme Probleme zu lösen, und es ist am besten, wenn wir sie von heute an gemeinsam angehen.

Shenja Panik: Sie haben uns von Ihrem Werdegang und der Art und Weise erzählt, wie Sie zu diesem Punkt gekommen sind. Was sind nun Ihre wichtigsten Inspirationsquellen, wenn Sie ethische Entscheidungen treffen?
Alexander Gerfer: An erster Stelle steht natürlich mein ganz persönlicher ethischer und werteorientierter Kompass, den ich von zu Hause mitbringe. Das Verantwortungsbewusstsein gegenüber der Natur und den Mitmenschen wurde mir von klein auf in die Wiege gelegt. Deshalb liegt mir zum Beispiel das Thema Vertical Farming mit LEDs sehr am Herzen.
Wichtig für uns in der Würth-Gruppe sind unsere Unternehmenswerte, die unsere Einstellung und unser Geschäftsgebaren in allen wichtigen Bereichen bestimmen. Gegenseitiges Vertrauen, Berechenbarkeit, Ehrlichkeit und Geradlinigkeit nach innen und außen sind Grundprinzipien, die Prof. Dr. h. c. mult. Reinhold Würth bereits in den 1970er Jahren formulierte.
Der Compliance-Kodex der Würth-Gruppe umfasst 32 Seiten, in denen allgemeine Verhaltensgrundsätze, Richtlinien für den Umgang mit Geschäftspartnern, Anweisungen zur Vermeidung von

Interessenkonflikten und zur Umsetzung des Compliance-Kodex beschrieben werden.
Bevor ich eine Entscheidung treffe, fragen meine Kollegen und ich: Steht meine Entscheidung oder Handlung im Einklang mit den geltenden Gesetzen? Steht die Entscheidung im Einklang mit den bekannten Werten und Regeln von Würth? Angenommen, meine Kollegen und meine Familie wüssten von meiner Entscheidung, hätte ich dann ein reines Gewissen? Wenn alle so entscheiden würden, könnte ich dann mit den Konsequenzen leben? Wenn meine Entscheidung morgen in der Zeitung bekannt würde, könnte ich sie rechtfertigen?
Wie ich schon sagte, muss man bei seinen Entscheidungen fair sein und für jeden Beitrag respektvoll und dankbar sein, aber sehr wichtig ist auch, dass man berechenbar und zuverlässig ist.

Shenja Panik: Wie viel Widerstandskraft ist erforderlich, um ethische und nachhaltige Initiativen in einem großen Unternehmen umzusetzen?
Alexander Gerfer: In unserem Tagesgeschäft? Wenig bis gar keine, da wir unsere Mitarbeiter sehr sorgfältig auswählen. Compliance bedeutet für uns nicht nur die Einhaltung aller geltenden Vorschriften, Gesetze und Unternehmensrichtlinien, sondern auch eine entsprechende innere Haltung der Mitarbeiter, die ein entscheidender Baustein für den nachhaltigen Geschäftserfolg von Würth Elektronik ist.
Wir erwarten daher von unseren Führungskräften, Mitarbeitern und Geschäftspartnern, dass sie die national und international geltenden Gesetze und Vorschriften einhalten. Darüber hinaus verpflichten wir uns im Rahmen unseres Wertekanons zur Einhaltung der im Code of Compliance klar definierten Standards, Richtlinien und Normen. Diese gemeinsamen Werte ziehen sich durch alle Bereiche des Unternehmens und werden von allen Mitarbeitern verinnerlicht.
Bei der Auswahl beurteilen wir Bewerber nicht nur nach ihrer fachlichen Qualifikation, sondern vor allem nach ihrer persönlichen Einstellung.
Stimmen sie mit unseren Grundsätzen überein? Identifizieren sich die potenziellen Mitarbeiter mit unseren Unternehmenswerten oder haben sie diese nur auswendig gelernt? Sind sie eher auf nachhaltigen Erfolg ausgerichtet oder nur auf Quartalszahlen? Betrachten sie Kollegen als Partner oder arrogant? Denken sie ganzheitlich und umweltbewusst oder egozentrisch und opportunistisch?
Diese und viele weitere Fragen müssen vor der Einstellung beantwortet werden. Wissen kann man sich durch im Job aneignen, und Fertigkeiten kann man von erfahrenen Kollegen lernen. Die unverzichtbare Grundlage sind jedoch ethische Richtlinien, die jeder in die Gruppe einbringen muss. Es gibt also nicht viel Widerstandsfähigkeit - die Menschen, die hier arbeiten, glauben an das, was wir tun und wie wir es tun. Und wir erleben täglich, dass unsere Mitarbeiter sehr bewusst mit Verantwortung umgehen. Hier ziehen die Unternehmensführung, die Führungskräfte und die Mitarbeiter an einem Strang. Ethik wird nicht erzwungen, sondern in unserer Organisation gelebt.

Shenja Panik: Es geht nicht darum, von Projekt zu Projekt zu gehen, sondern eine Unternehmensphilosophie zu pflegen. Jeder von Ihnen ist an der Frage beteiligt, wie man die Welt besser machen kann.
Alexander Gerfer: Wir befähigen nicht nur unsere eigenen Mitarbeiter, sondern auch Menschen außerhalb unseres Unternehmens. Aus meiner eigenen Erfahrung mit Start-ups und den damit verbundenen Herausforderungen habe ich erkannt, wie wichtig es ist, in der Anfangsphase auf Entwickler zuzugehen, insbesondere im Bereich der Hardware, die zu Beginn eine der größten Herausforderungen darstellen kann. Eine falsche Entscheidung an dieser Stelle kann selbst die beste Idee zunichtemachen. Deshalb bieten wir Hilfe und Unterstützung an. Und wir lernen dabei selbst... Unsere übergreifende Vision ist also ein vollständig vorhersehbares Verständnis der Kundenbedürfnisse. Ethik und geschäftlicher Erfolg sind für uns keine Gegensätze, sondern untrennbar miteinander verbunden.

RG — 240013-02

Haben Sie Fragen oder Kommentare?

Haben Sie Fragen oder Kommentare zu diesem Artikel? Schicken Sie eine E-Mail an Shenja unter shenja.panik@elektor.com.

Über Shenja Panik

Shenja Panik ist eine leidenschaftliche Soziologin, die für das Inhalts-Team von EiE schreibt und sich auf Ethik in der Elektronikindustrie spezialisiert hat. Ihre Arbeit dreht sich um die Erforschung und Förderung ethischer Überlegungen in der sich ständig weiterentwickelnden Technologielandschaft.

Über Alexander Gerfer

Alexander Gerfer ist der Technische Direktor von Würth Elektronik eiSos, einem großen europäischen Hersteller von elektronischen Bauteilen. Er ist seit 27 Jahren für das Unternehmen tätig, hat verschiedene Positionen inne und fördert Technologiepartnerschaften in Bereichen wie Lösungen mit LED-Licht, Energy Harvesting und E-Mobilität. Mit seinem Fachwissen unterstützt und berät er leidenschaftlich gerne innovative Start-ups. Gerfer ist auch Autor von Fachbüchern, Dozent und Hauptredner auf Konferenzen auf der ganzen Welt.

elektorstore

www.elektor.de

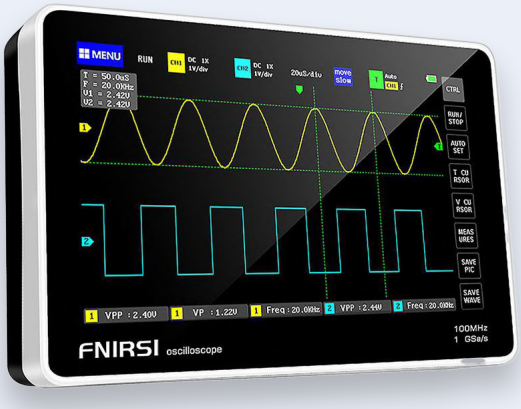
Der Elektor Store

Nie teuer, immer überraschend!

Der Elektor Store hat sich vom Community-Store für Elektor-eigene Produkte wie Bücher, Zeitschriften, Bausätze und Module zu einem umfassenden Webshop entwickelt, der einen großen Wert auf überraschende Elektronik legt.

Wir bieten die Produkte an, von denen wir selbst begeistert sind oder die wir einfach ausprobieren wollen. Wenn Sie einen Produktvorschlag haben, sind wir hier erreichbar (sale@elektor.de).

FNIRSI 1013D 2-Kanal Tablet-Oszilloskop (100 MHz)

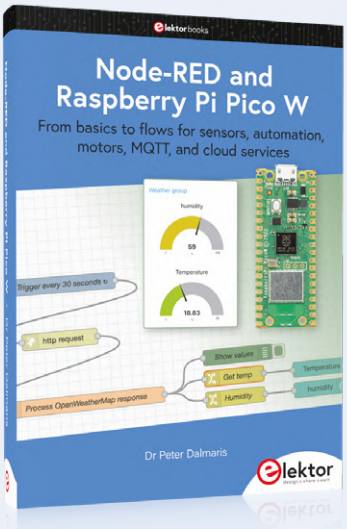


Das FNIRSI 1013D ist ein voll ausgestattetes 2-Kanal Tablet-Oszilloskop mit einem hochauflösenden 7-Zoll-LCD-Bildschirm (800 x 480 Pixel). Das Oszilloskop verfügt über eine Echtzeit-Abtastrate von 1 GSa/s und eine analoge Bandbreite von 100 MHz.

Preis: 159,95 €

Mitgliederpreis: 143,96 €

www.elektor.de/20644



Node-RED and Raspberry Pi Pico W

Dieses Buch ist eine Lernanleitung und ein Nachschlagewerk, mit dessen Hilfe Sie Node-RED, den Raspberry Pi Pico W und MicroPython kennenlernen und Ihr Technologie-Toolkit um diese hochmodernen Werkzeuge erweitern.

Preis: ~~44,95 €~~

Mitgliederpreis: 40,46 €

www.elektor.de/20745

Oxocart Connect Innovator Kit

Preis: 89,95 €

Mitgliederpreis: 80,96 €

www.elektor.de/20718

SunFounder Kepler Kit (Ultimate Starter Kit für Raspberry Pi Pico W)

Preis: 69,95 €

Mitgliederpreis: 62,96 €

www.elektor.de/20730

Temperaturgesteuerte Lötstation ZD-931

Preis: 49,95 €

Mitgliederpreis: 44,96 €

www.elektor.de/20623

FNIRSI DSO152 Mini-Oszilloskop (200 kHz)

Preis: 39,95 €

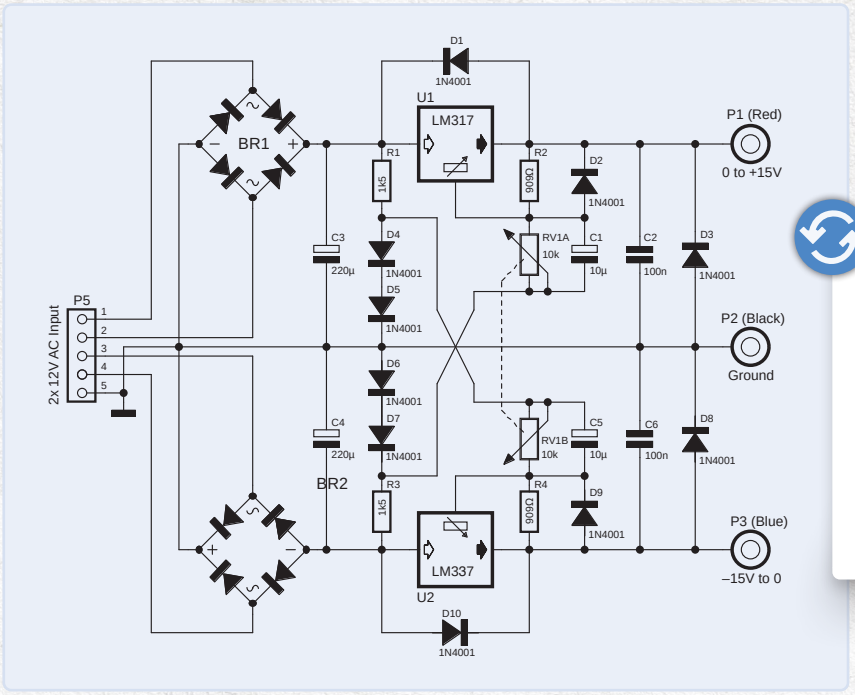
Mitgliederpreis: 35,96 €

www.elektor.de/20642

Projekt 2.0

Korrekturen, Updates und Leserbriefe

Zusammengestellt von Jean-François Simon (Elektor)



Variable lineare Stromversorgung

Elektor 1-2/2024, S. 26 (220457)

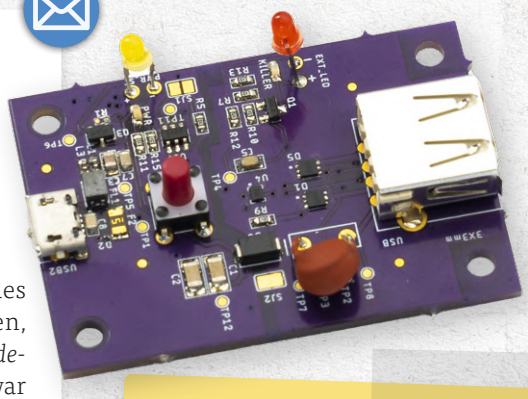
Es gibt einen Fehler im Schaltplan des dualen variablen Netzteils (S. 29). Der negative Spannungsregler am unteren Rand der Zeichnung ist ein LM337, kein LM317. Im Text des Artikels wird korrekt von einem LM337 gesprochen.

USB-Killer-Detektor

Elektor 11-12/2023, S. 32 (220549)

Gibt es den USB-Killer-Detektor als fertiges Gerät zu kaufen?
N. D. (Deutschland)

Sie finden alle Dateien, die zum Bau des USB-Killer-Detektors benötigt werden, auf GitHub (*https://github.com/instantdevices/USB-Killer-Detector*). Ich wäre zwar in der Lage, Teile zu liefern, aber ich suche zunächst die Unterstützung der Community aufgrund eines anhaltenden Problems, bei dem ein bestimmtes Killer-Modell den Detektor außer Gefecht setzt. Ich würde also sagen, dass das Gerät noch nicht ganz marktreif ist. Sie können mich unter instant.devices@yahoo.com erreichen.
Carlos Guzman (Autor des Artikels)



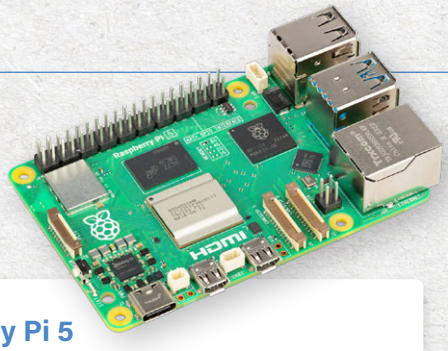
Haben Sie eine gute Idee oder wertvolles Feedback für Elektor? Melden Sie sich bei uns unter redaktion@elektor.de - wir freuen uns darauf, von Ihnen zu hören!

Einrichten einer SMT-Fertigungsstrecke

Elektor 11-12/2023, S. 108 (230593)

In diesem Artikel haben Sie den Whizoo Controleo3 vorgestellt. Können Sie uns eine Bezugsquelle, eventuell in Deutschland, nennen?
Wolf-Peter Kleinau (Deutschland)

Der Artikel wurde von unseren Partnern bei Opulo, einem in den USA ansässigen Unternehmen, geschrieben. Whizoo ist ebenfalls ein US-amerikanisches Unternehmen und leider gibt es meines Wissens keine Wiederverkäufer in der EU. Außerdem impliziert der Artikel von Opulo, dass „Controleo3“ eine fertige Reflow-Lösung von der Stange ist. Um genau zu sein, war der Controleo3 zunächst als Bausatz für einen eigenen Reflow-Ofen bekannt, und zufälligerweise verkauft Whizoo auch einen komplett montierten Ofen für rund 1200 €. Leider ist dieses Produkt nur in den USA und Kanada erhältlich und wird mit einem 115-V-Ofen geliefert. Wenn Sie sich für Controleo3-Produkte entscheiden, können Sie diese in den USA immer noch als Bausatz kaufen, aber Sie müssen neben dem enormen „Shipping“ auch Mehrwertsteuer und Einfuhrzoll bezahlen, was sehr lästig sein kann, und Sie müssen Ihren eigenen Ofen bereitstellen und zusammenbauen. So gesehen ist der Bausatz nicht „reflow ready“.
Auf die Frage „Welchen Reflow-Ofen soll ich kaufen?“ gibt es keine einfache Antwort. Dies ist ein weites Feld, und ich würde Ihnen raten, sich umfassend zu informieren und zu entscheiden, was Ihren Bedürfnissen am besten entspricht. Persönlich kann ich keine Empfehlungen aussprechen, aber ich habe schon von der Firma Beta Layout gehört. Zwei beliebte Optionen anderer Herstellern sind der Ofen T962 (in vielen Varianten) und der ZB3530HL. Einige Leute im Internet haben auch einige Möglichkeiten gefunden, den T962 zu verbessern, um eine bessere Wärmeverteilung oder eine bessere Benutzeroberfläche zu erreichen.
Jean-François Simon (Elektor)



Der Raspberry Pi 5

Elektor 11-12/2023, S. 6 (230635)

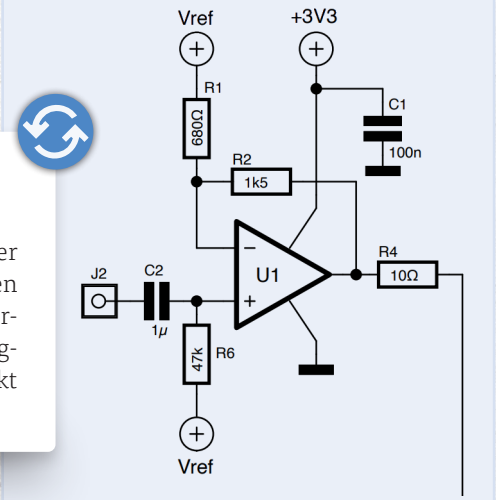
Vielen Dank für den informativen Artikel. Bei den verschiedenen Raspberry Pi 4, die ich besitze, bin ich es gewohnt, die SD-Karten zu kopieren, um alles, was mühsam installiert wurde (zum Beispiel VS-Code, Qt 5, CAN-Bus und so weiter), einfach auf andere Geräte zu portieren. Können Sie mir und den anderen Lesern sagen, wie man diese auf den Pi 5 portiert? Ich habe die kopierte SD-Karte in den Pi 5 eingelegt, aber es funktioniert nicht. Herzlichen Dank!
Hanspeter Schären (Deutschland)

Vielen Dank für Ihr E-Mail. Das hat mit der Version des Raspberry-Pi-OS zu tun, die auf Ihren SD-Karten vorhanden ist, die Sie mit dem Pi 4 verwendet haben. Der Pi 5 benötigt die OS-Version Bookworm und kann nicht mit früheren Versionen betrieben werden. In Ihrem Fall müssen Sie also leider von Grund auf eine neue SD-Karte erstellen, zum Beispiel mit dem einfach zu bedienenden Raspberry-Pi-Imager. Und Sie müssen die Software, die Sie benötigen, manuell neu installieren. Wenn Sie das getan haben, sollte diese SD-Karte mit Bookworm darauf im Prinzip sowohl mit dem Pi 5 als auch mit dem Pi 4 funktionieren, falls Sie jemals von einem Raspberry zum anderen wechseln müssen, aber Sie sollten Ihre eigenen Tests durchführen, um zu sehen, ob alles funktioniert. Viel Erfolg!
Jean-François Simon (Elektor)

MEMS-Mikrofon

Elektor 11-12/2023, S. 70 (230188)

Beim Neuzeichnen des Schaltplans (Bild 3, S. 72) ist uns ein Fehler unterlaufen. Wir entschuldigen uns dafür! Die nicht-invertierenden Eingänge der Operationsverstärker U1 bis U6 sind über 47-kΩ-Widerstände mit Vref verbunden, nicht mit Masse (siehe das Schaltplanfragment rechts). Der Originalschaltplan im verlinkten Hackaday.io-Projekt ist natürlich korrekt.



Kaltkathoden-Röhren

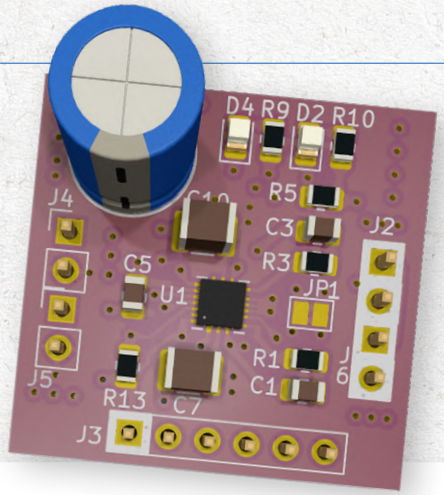
Elektor 1-2/2024, S. 74 (230373)

Ich habe gehört, dass einige Kaltkathodenröhren schwach radioaktiv sind. Stimmt das? Zum Beispiel die Röhren von Elesta und andere wie die ER21A, GT21, 5823 oder die GR16?
Adolf Parth (Italien)

Sie haben Recht. Einige dieser Röhren enthalten radioaktives Material, um das Gas in ihnen zu ionisieren! Im Internet finden Sie mehrere Websites von Radioaktivitäts- und Geigerzähler-Enthusiasten, die Ihnen genauere Informationen darüber geben, welche spezifischen Röhren welche radioaktive Quelle enthalten.
Jean-François Simon (Elektor)

Die Funktion einer Kathode (in Heiz- oder Kaltkathodenröhren) besteht normalerweise darin, Elektronen auszusenden. Bei Heizkathoden werden Elektronen ausgesandt, die in der Regel durch ein Gitter moduliert und von einer Anode aufgefangen werden. Bei Kaltkathodenröhren werden die Elektronen in der Regel zur Ionisierung eines Gases abgegeben, entweder zum Schutz vor Überspannungen, zur Anzeige (Neons und Nixies) oder zum Schalten, wie bei den von Herrn Parth erwähnten Kaltkathoden-Thyratronröhren.
Aus diesem Grund werden die Kathoden sehr oft mit Substanzen beschichtet, die die Emission von Elektronen unterstützen. Bei Standard-Heizkathodenröhren wird in der Regel Thorium verwendet; in Kaltkathodenröhren kommen Seltenerdmetallen und -verbindungen zum Einsatz. Einige von ihnen weisen eine geringe Radioaktivität auf. Im Periodensystem gehören die Metalle der Seltenen Erden zur Reihe der Lanthanoide und Thorium zur verwandten Reihe der Actiniden. Lanthanoide sind nur selten radioaktiv, die meisten Aktiniden dagegen schon. Radioaktives Thorium wird auch in den Mänteln von Gaslampen verwendet. Viele gewöhnliche Elemente haben einen Anteil an radioaktiven Isotopen, zum Beispiel das bekannte Element Kalium.
David Ashton (Autor des Artikels)

Quelle: https://en.wikipedia.org/wiki/Nixie_tube#/media/File:ZM1210-operating_edit2.jpg



Motor Driver Breakout Board

Elektor 9-10/2023, S. 56 (210657)

Hallo zusammen! Ist es möglich, mit dem Motortreiber MP6619 eine temperaturabhängige Drehzahlregelung für einen Lüfter zu erstellen, der dann von 3 V_{DC} bei 20°C auf 12 V_{DC} beispielsweise bei 40°C ausgibt, unter Verwendung eines 10-kΩ-NTCs? Für Informationen oder Schaltungsbeispiele wäre ich sehr dankbar. Oder gibt es vielleicht sogar eine andere Möglichkeit, eine solche Drehzahlregelung zu bauen?
Matthias Seesemann (Deutschland)

Es ist möglich, das, was Sie beschreiben, mit einem MP6619 zu machen, aber Sie brauchen auch andere Bauteile. Der MP6619 allein ist nur eine H-Brücke, was bedeutet, dass er seine Ausgangstransistoren ein- oder ausschaltet, abhängig vom Zustand der Eingangspins IN1, IN2 und ENABLE. Um eine Ausgangsspannung von 3 V bis 12 V zu erhalten, müssen Sie PWM verwenden, um diese Eingangspins des MP6619 zu steuern. Der EN-Pin sollte mit einem Pull-up-Widerstand auf High gezogen werden, dann kann man IN2 je nach gewünschter Drehrichtung auf High oder Low ziehen und gleichzeitig ein PWM-Signal an IN1 senden, um die Geschwindigkeit des Motors zu variieren. Um das PWM-Signal zu erzeugen, können Sie einen beliebigen Mikrocontroller (Arduino oder andere) verwenden. Ich empfehle einen Arduino Uno R3, für den es viele Code-Beispiele im Internet gibt. Dort wird gezeigt, wie man eine Temperatur mit einem NTC erfassen kann und wie man PWM-Funktionen auf dem Arduino verwendet. Vielleicht interessieren Sie sich auch für einige Online-Projekte, die den „Fan speed controller“ TC648 verwenden: Ich glaube, er macht genau das, was Sie brauchen. Möglicherweise müssen Sie ein oder zwei Widerstandswerte ändern, damit er sich genau so verhält, wie Sie es wünschen. Sie sollten Ihr Projekt auf unserer Elektor-Labs-Plattform (www.elektormagazine.de/labs) veröffentlichen, damit andere Mitglieder der Elektor-Community sie kommentieren und Ihnen helfen können. Viel Spaß damit!
Jean-François Simon (Elektor)

RG — 240020-02

Treten Sie jetzt der Elektor Community bei!

Jetzt elektor GOLD membership Mitglied werden!

- Zugang zum kompletten Online-Archiv (1970-heute)!
- 8x Elektor Magazin (gedruckt)
- 8x Elektor Magazin (PDF)
- 10% Rabatt im Elektor Store und exklusive Angebote
- Zugriff auf über 5.000 Gerber-Dateien

u.v.m. aus der Projektplattform Elektor Labs

Auch erhältlich

Die digitale elektor GREEN membership Mitgliedschaft!

- Zugang zum kompletten Online-Archiv
- 8x Elektor Magazin (PDF)
- 10% Rabatt im Elektor Store und exklusive Angebote
- Zugriff auf über 5.000 Gerber-Dateien

u.v.m. aus der Projektplattform Elektor Labs

www.elektormagazine.de/Abonnement

elektor design > share > earn

Erweitern Sie Ihr Fachwissen

Schnelle Tipps, Tools und Artikel für Einkäufer

mou.sr/purchasing-resources

MOUSER ELECTRONICS