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

elektor MAG

design > share > earn

592
MÄRZ/APRIL 2023
ELEKTORMAGAZINE.DE
* SEIT 1961 *

ESP32-basierter

Wecker

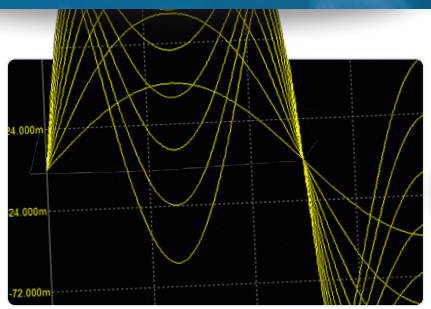
Gesteuert von Time Servern,
Steuert Radio, HiFi und TV

IM FOKUS
Embedded und KI

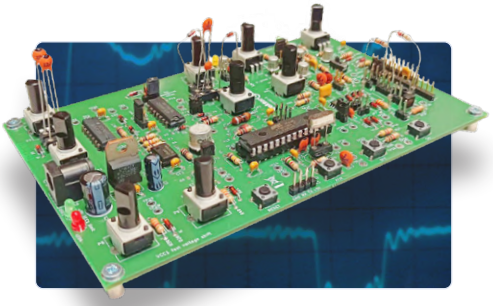
KI und eingebettete Systeme – Was kommt als Nächstes?
Werkzeuge, Plattformen und das Ende der Schriftstellerei

RP2040 PIO in der Praxis
Experimente mit programmierbaren I/Os

Videoausgabe mit Mikrocontrollern
VGA und DVI
S. 92



Schaltungssimulation mit Micro-Cap Erste Schritte in einer komplizierten Welt
S. 48



ChipTweaker des armen Mannes
Hardware einfach selber attackieren
S. 22

4 198630 314904 30004

MICROCHIP

CURIOSITY NANO

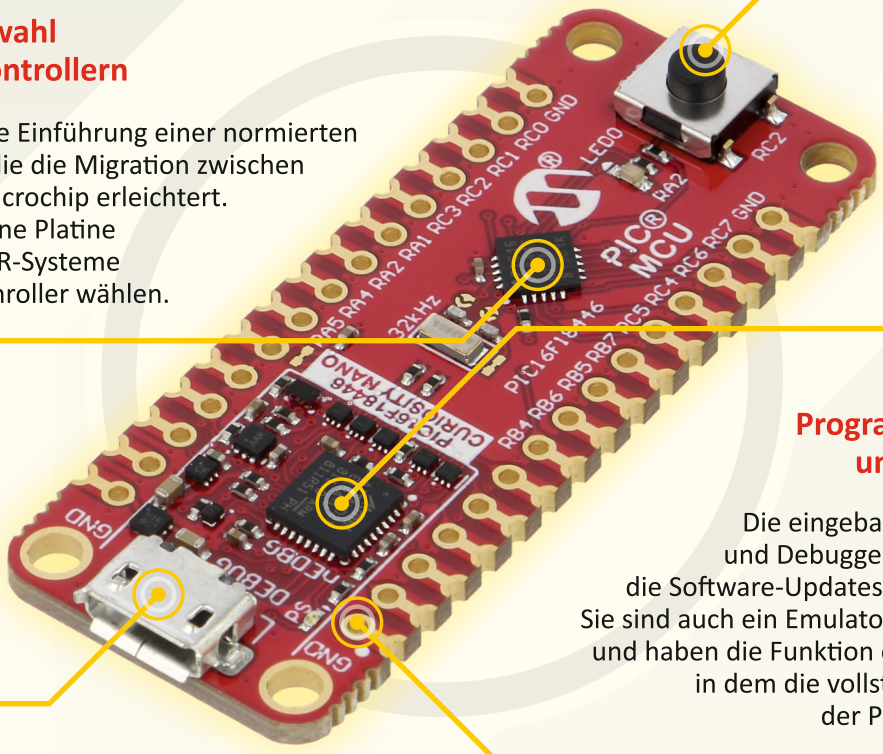
Prototypen-Platinen, die mit Erweiterungsmodulen zusammenarbeiten

Basiskomponenten

Die Platinen Curiosity Nano haben eine Taste, eine LED, einen Quarzresonator sowie einen programmierbaren Spannungsregler. Somit verlangen der grundlegende Prototypenbau und die Einarbeitung in die Plattform kein zusätzliches Zubehör.

Breite Auswahl an Mikrocontrollern

Die Serie wurde für die Einführung einer normierten Plattform konzipiert, die die Migration zwischen den Produkten von Microchip erleichtert. Die Kunden können eine Platine mit 8-Bit-PIC- und -AVR-Systeme sowie 32-Bit-Mikroconroller wählen.



Integrierte Programmiergerät und Debugger

Die eingebauten Programmiergerät und Debugger ermöglichen nicht nur die Software-Updates ohne zusätzliche Tools. Sie sind auch ein Emulator der COM-Schnittstelle und haben die Funktion eines Massenspeichers, in dem die vollständige Dokumentation der Platine aufbewahrt wird.

USB-Schnittstelle

Um den eigenen Code auf der Plattform MC Nano zu programmieren und laufen zu lassen, reicht es, die Platine an die USB-Schnittstelle des Computers anzuschließen und das .hex-File auf das Gerät zu kopieren.

Normierte Ausführungen

Unabhängig vom ausgewählten Mikroprozessor-Modell oder sogar seiner Architektur, befinden sich die Basis-Ausführungen auf der Platine immer an der gleichen Stelle. Die hohe Standardisierung reduziert die Kosten und verkürzt die Zeit des Prototypings.

TME Electronic Components

TME Germany GmbH
Dohnanyistraße 28-30, 04103 Leipzig
+49 341 21203429

tme@tme-germany.de

facebook.com/TME.eu
youtube.com/TMElectroniComponent
linkedin.com/company/1350565
instagram.com/tme.eu
twitter.com/tme_eu
tiktok.com/@tme.eu

tme.eu / tme.com

IMPRESSUM

55. Jahrgang, Nr. 592
März/April 2023
ISSN 0932-5468

Erscheinungsweise: 8x jährlich

Verlag
Elektor Verlag GmbH
Lukasstraße 1
52070 Aachen
Tel. 0241 95509190

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

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

Anzeigen
Margriet Debeij (Leitung)
Mobil: +31 6 380 780 29
E-Mail: margriet.debeij@elektor.com

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

Es gilt die Anzeigenpreisliste ab 01.01.2023.

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

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

EDITORIAL

Von Jens Nickel

Chefredakteur ElektorMag

Spannende Zeiten

Als wir dieses Heft geplant haben, konnten wir noch nicht wissen, dass KI einen solchen Hype erfahren würde. Das Zauberwort lautet hier natürlich ChatGPT, ein Chatbot, der nicht nur Fragen zu allen möglichen Wissensgebieten beantwortet, sondern auf Knopfdruck auch Geschichten, Songtexte und vieles mehr auswerfen kann. Nicht nur einer unser freien Autoren (siehe Seite 64), auch ich habe ChatGPT ausprobiert; mit Fragen zu solaren Inselanlagen und den passenden Stromspeichern. Das Ergebnis hat mich beeindruckt. Der Chatbot konnte nicht nur abschätzen, wie groß die Kapazität einer LiFePo4 für ein typisches 2-Personen-Wochenendhaus ausfallen müsste, er lieferte mir auch auf Nachfrage fünf Vorschläge für eine solche Batterie, mit Hersteller und Typ. Darüber hinaus waren ChatGPT auch die Grenzen seiner eigenen Aussagen „bewusst". Der Bot warnte mich, dass es bei Kapazitätsberechnungen auf den Energieverbrauch und die Größe der Module ankäme und empfahl mir, vor dem Kauf einen Fachmann zu Rate zu ziehen.
Doch natürlich hat ChatGPT kein Bewusstsein oder Gespür für die Dinge, der er/sie/es von sich gibt. Der Output ist eine Art gewichteter Durchschnitt von zig Millionen von möglichen Aussagen, die alle irgendwo in ähnlicher Form im Internet zu finden sind. Die sprachliche Fertigkeit überrascht, doch fast alles hat man so oder so ähnlich schon mal gehört. Diese „Durchschnittsbildung" aus einem riesigen Wissensfundus lässt die Antworten von ChatGPT oft etwas breiig erscheinen.
Dennoch sind wir bei Elektor fasziniert: Ich kann mir gut vorstellen, dass der Bot uns und Ihnen in Zukunft helfen könnte. Zum Beispiel bei der Suche nach einem passenden Elektor-Artikel in unserem Archiv, denn schließlich schlummern dort 62 Jahre geballtes Elektronikwissen. Vielleicht auch beim Schreiben von Boilerplate-Code, dem Anlegen von Stücklisten und dem Design von immer wiederkehrenden Schaltungsteilen?
Zumindest mittelfristig sehe ich es nicht als Gefahr, dass ChatGPT unsere Jobs oder auch Ihren Input, liebe Leser, zu unserer Zeitschrift überflüssig macht. Bisher bekommt KI keine Gänsehaut, wenn sie ein tolles Projekt oder Gerätchen entdeckt, dass man super für eigene Entwicklungen einsetzen könnte. KI hat keinen Einfall unter der Dusche und schreibt mir auch keine launigen E-Mails, dass früher alles besser war. Diese Emotionen sind es, die uns antreiben; im Keller-Labor und beim Texten von Artikeln.
Sehen Sie das anders? Welche Erfahrungen haben Sie schon mit KI gemacht? Und wo wird das alles hinführen? Schreiben Sie mir unter redaktion@elektor.de!

Elektor auf der embedded world
Vom 14. bis zum 16. März findet die embedded world in Nürnberg statt, ein Pflichttermin für alle Mikrocontroller- und Software-Profis. Wie in jedem Jahr ist auch Elektor vertreten - **besuchen Sie uns am Stand 4A-646!**

Unser Team

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

ESP32-basierter Wecker Cloc 2.0

Gesteuert durch Zeitserver, steuert Radio und TV

6

Rubriken

Hintergrund

Industry

Projekte

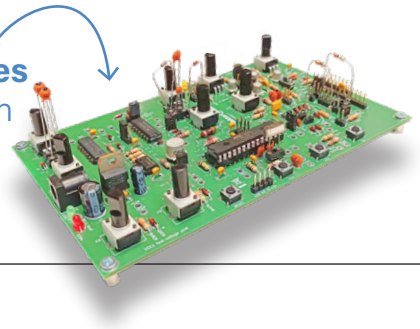
Vorschau

Elektor Mai/Juni 2023
Das nächste Heft ist wie immer randvoll gefüllt mit Schaltungsprojekten, Grundlagen sowie Tipps und Tricks für Elektroniker. Schwerpunkt der nächsten Ausgabe ist das Messen und Testen.

Aus dem Inhalt:
- Energielogger
- HF-Tastkopf für Oszilloskope
- Anpassbare Universal-Fernbedienung mit ESP32
- Praxisprojekt mit der Android WiFi-API
- BL808 & Co.: Neue RISC-V-MCUs für KI, Video und vieles mehr
- eCO_2-Telegram-Bot
- Grundlagen: Analoge Signale und Mikrocontroller
- Super-Servo-Tester
- Programmierung sprachgesteuerter IoT-Applikationen

Und vieles mehr!

Elektor **Mai/Juni 2023** erscheint am 11. Mai 2023.



Cloc 2.0:
Der Wecker, den Sie sich schon immer gewünscht haben

Von Yves Bourdon (Frankreich)

Uhren sind wahrscheinlich das beliebteste Thema in der DIY-Elektronik, zusammen mit Radioempfängern. Es würde mich nicht wundern, wenn jeder Leser dieses Artikels mindestens einmal eine Uhr gebaut hat. Ich habe in meinem Leben schon viele Uhren entworfen, und viele davon hatten Weckfunktionen. Daher weiß ich, was ich von einem Wecker erwarte. Da mein idealer Wecker nicht auf dem Markt erhältlich ist, habe ich ihn selbst gebaut. Cloc 2.0 ist das Ergebnis meiner Bemühungen und genau dies will ich hier vorstellen.

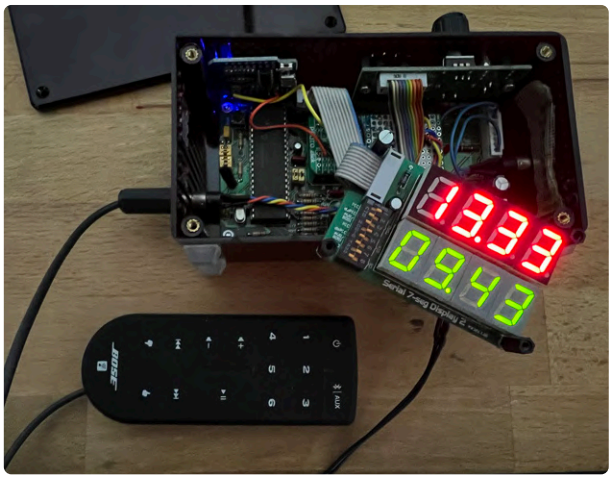


Bild 1. Cloc 1.0, der Vorgänger von Cloc 2.0. Die Uhr basiert auf einem PIC32 und verwendet eine echte Fernbedienung als IR-Ausgang.

Bild 2. Das Main-Panel zeigt neben der aktuellen Uhrzeit und dem Datum eine Übersicht über die wichtigsten Einstellungen.

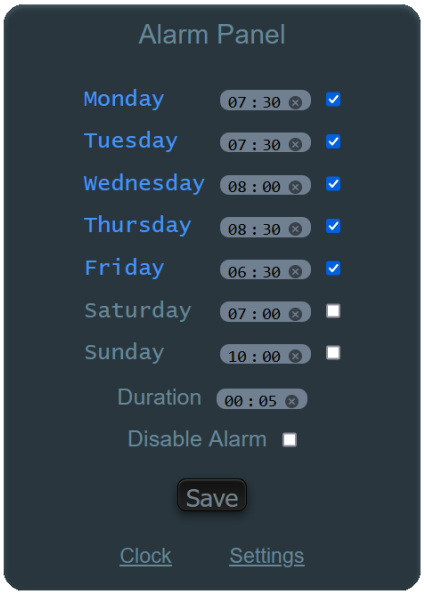


Bild 3. Im Alarm-Panel können Sie die Alarmzeit für jeden Wochentag einstellen.

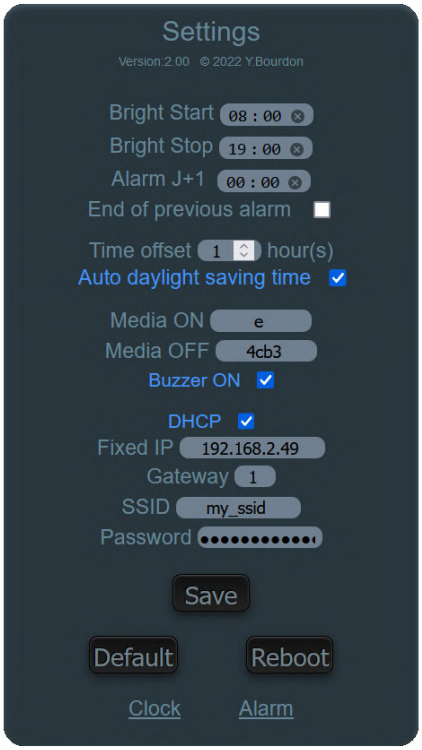


Bild 4. Im Settings-Panel lassen sich alle vom Benutzer konfigurierbaren Parameter einstellen.

Meine erste Uhr baute ich im Jahr 1971 mit Nixie-Röhren aus einem alten IBM-Computerdisplay, das ich geschenkt bekommen hatte. Die Platine enthielt eine beeindruckende Anzahl von Bauteilen, und der Fehler betrug etwa zehn Sekunden pro Tag. In den folgenden Jahren habe ich viele Uhren entworfen und gebaut, darunter auch die berühmtesten von Elektor.

Meine Nixie-Uhr habe ich auch als Wecker verwendet. Die Funktionen habe ich im Laufe der Jahre immer weiterentwickelt, insbesondere die Schnittstelle zwischen Mensch und Maschine so weit wie möglich vereinfacht. Mein letzter Wecker basierte auf einem PIC32-Controller mit zwei Displays (**Bild 1**). Er verfügte über ein ESP8266-Modul, um die Zeit automatisch von einem Online-Zeitserver über NTP einzustellen. Der PIC32 „drückte" Tasten auf einer echten Fernbedienung, um meinen Lieblingsradiosender einzuschalten oder Musik abzuspielen. Ein Drehencoder und zwei Drucktasten ermöglichten das Einstellen des Weckers. Dieser Wecker war die Grundlage für das neue Projekt Cloc 2.0.

Schön und leicht nachzubauen

Ich wollte einen Wecker bauen, der einfach zu bauen ist, ohne SMD-Bauteile und leicht in ein handelsübliches Gehäuse passend. Außerdem wollte ich den Retro-Touch der 7-Segment-Anzeigen beibehalten, deren variable Helligkeit den Wecker nachts sehr gut ablesbar macht, ohne dass er zu sehr stört, wenn man gerne im Dunkeln schläft.

Die Webschnittstelle

Cloc verbindet sich mit einem WLAN, so dass der Wecker auf einen beliebigen Zeitserver zugreifen kann. Die Verbindung kann über DHCP oder mit einer festen IP-Adresse hergestellt werden. Sobald die Verbindung steht, können Sie auf den Webserver zugreifen, um die Uhrzeit zu erfahren, die Alarme einzustellen und die Parameter der Uhr anzupassen. Die optimierte grafische HTML-Schnittstelle, die aus den drei Bereichen ESP32 CLOCK (das Hauptmenü), Alarm und Settings (Einstellungen) besteht, funktioniert auf den meisten Standardbrowsern und Handheld-Geräten.

ESP32 CLOCK

Das Hauptfenster (**Bild 2**) zeigt die aktuelle Uhrzeit (NTP Time) mit Datum sowie die Start- und Endzeit des bevorstehenden Alarms an. Es zeigt auch die SSID des WLAN-Netzwerks, mit dem es verbunden ist, sowie dessen Signalstärke RSSI an. Die Farbe der RSSI-Anzeige (grün, orange oder rot) hängt wie bei einer Ampel vom Feldstärkebereich ab. Das Hauptfenster wird etwa jede Sekunde aktualisiert.

Alarm Panel

Das Alarmfeld in **Bild 3** dient der Einstellung der Alarmzeiten für jeden Wochentag. Wenn Sie einen Wochentag abwählen, wird der Alarm für diesen Tag deaktiviert. Es ist auch möglich, alle Alarme mit einem einzigen Klick zu deaktivieren (Disable Alarm).
Auch die Dauer des Alarms (Duration) ist konfigurierbar. Zu Beginn eines Alarms wird der Befehl *Media ON* über die Infrarot-LED gesendet, am Ende des Alarms wird der IR-Befehl *Media OFF* gesendet. Zusätzlich kann der Alarm einen Summer auslösen (muss in Settings aktiviert werden).

Bei jeder Änderung eines Alarmparameters wird ein kurzer Signalton ausgegeben. Wenn sich der Wert gegenüber dem vorherigen Wert geändert hat, wird er im EEPROM gespeichert.

Settings

Die folgenden Parameter können im Einstellungs-Panel eingestellt werden (**Bild 4**):

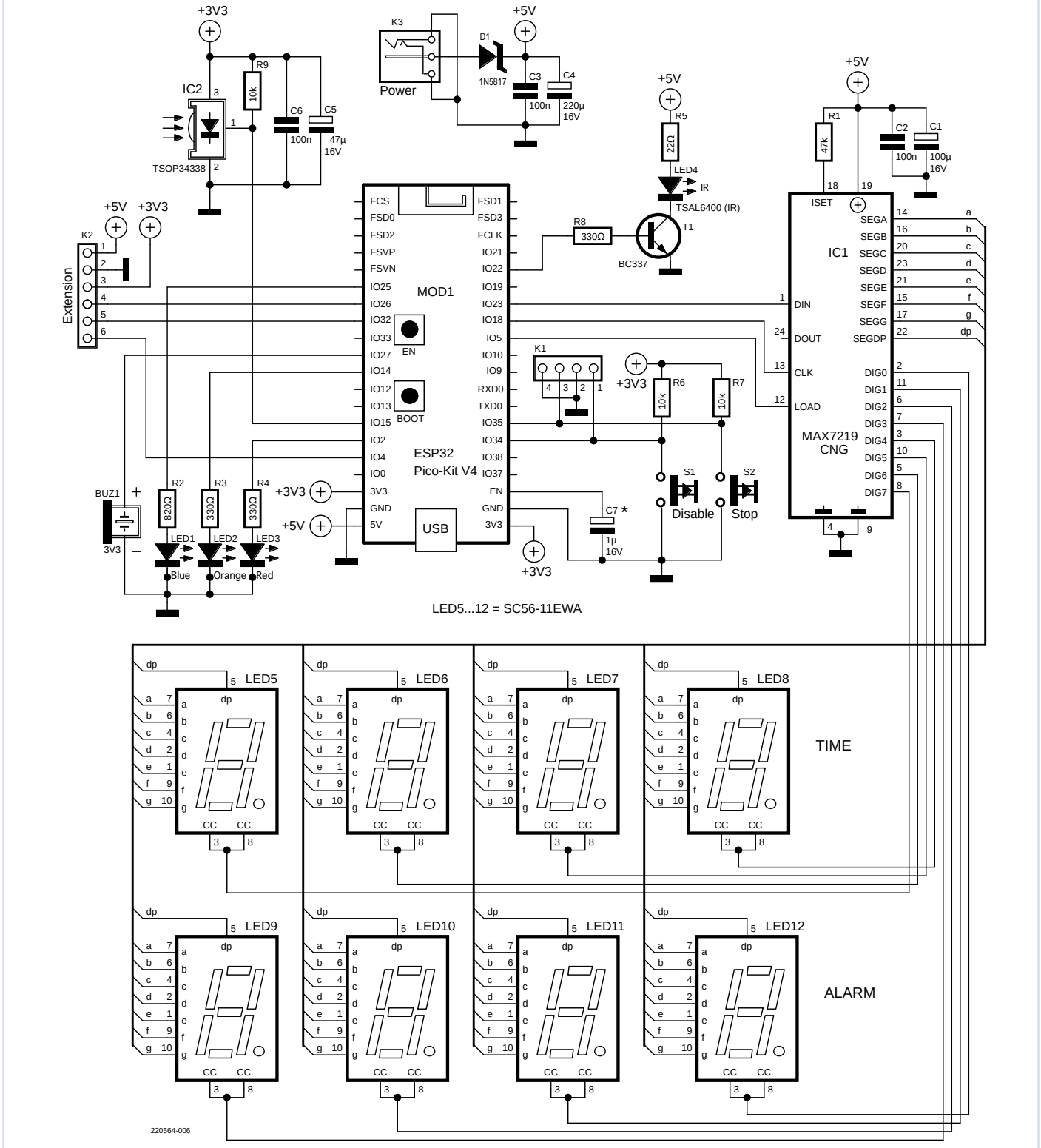


Bild 5. Der ESP32-PICO-Kit sorgt für die Internetverbindung und stellt die Uhrzeit bereit, während der MAX7219 2x4 7-Segment-Ziffern ansteuert. Beachten Sie die IR-Schnittstelle um IC2 (Eingang) und LED4 (Ausgang).

- Dauer der hell leuchtenden Anzeige (Bright Start und Bright Stop)
- Der Zeitpunkt, zu dem die nächste Weckzeit angezeigt wird (Alarm J+1), zum Beispiel abends beim Zubettgehen oder unmittelbar nach dem letzten Alarm (durch Markieren des Kästchens *End of previous alarm*).
- Sommerzeit, automatisch oder manuell
- Infrarot: *Media ON*- und *Media OFF*-Code zur Steuerung eines anderen Geräts (beispielsweise eines Radios)
- Aktivieren/Deaktivieren des Summers
- Anmeldeinformationen für das WLAN-Netzwerk mit DHCP oder einer festen IP-Adresse

Drucktasten

Cloc stellt dem Benutzer zwei Drucktasten zur Interaktion zur Verfügung: S1 (Disable) und S2 (Stop). Im normalen Betriebsmodus wird durch Drücken von S1 der Alarm aktiviert oder deaktiviert (die untere Anzeige schaltet sich ein oder aus). Sie können den Alarm auch über das Webinterface (de)aktivieren. Wenn S1 beim Einschalten von Cloc gedrückt wird, werden die Standardwerte wiederhergestellt, wie sie im Programm definiert sind (und die Sie vor dem Programmieren der Uhr an Ihre eigenen Bedürfnisse anpassen müssen!). Dies ist der richtige Weg, wenn Sie die Uhr zum ersten Mal einschalten.

Wenn im normalen Modus ein Alarm ausgelöst wurde, wird er durch Drücken von S2 sofort gestoppt und die untere Anzeige hört auf zu blinken. Wenn Sie S2 drücken, ohne dass ein Alarm ertönt, wird der IR-Code *Media ON/OFF* an ein Gerät Ihrer Wahl gesendet. So können Sie die Uhr als einfache Fernbedienung verwenden.

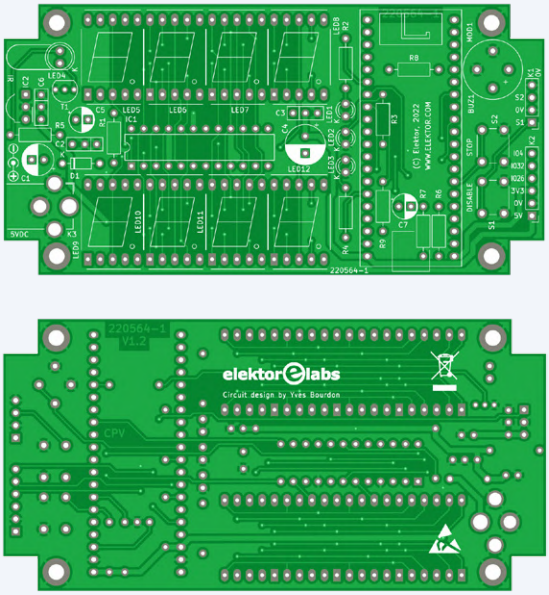
Wenn S2 während des Einschaltens gedrückt wird, geht die Uhr mit der Adresse 192.168.4.1 in den Access-Point-Modus. Sie können sich über ein Mobiltelefon, ein Tablet oder einen Computer mit der Uhr verbinden, um auf die Webschnittstelle zuzugreifen, über die sich alle frei programmierbaren Parameter des Geräts konfigurieren lassen.

Beschreibung der Schaltung

Der Schaltplan von Cloc 2.0 ist in **Bild 5** dargestellt. Fast zwei Drittel des Schaltplans werden von den beiden 7-Segment-Anzeigen und ihrem Treiber-IC1 eingenommen. Zur Ansteuerung der acht Ziffern, je vier für Zeit- und Alarm-Anzeige, habe ich den MAX7219 verwendet. Der DIN-Pin wird ignoriert und stattdessen der 2-Draht-SPI-Bus als Dateneingang verwendet. Dank Open-Source-Bibliotheken ist der Treiber leicht zu programmieren. Mit dem Widerstand R1 lässt sich die maximale Helligkeit der Anzeige einstellen (je höher der Wert, desto dunkler die Anzeige). Beachten Sie, dass High-Brightness-7-Segment-Ziffern sehr hell sind, viel zu hell für einen Wecker.

Das für Cloc gewählte ESP32-Modul ist das ESP32-PICO-Kit, das viele I/O-Ports bietet und die perfekten Abmessungen für dieses Projekt aufweist. Der Kondensator C7 wird nur benötigt, wenn das ESP32-Modul nach dem Einschalten im Reset-Modus verbleibt. Einige (ältere?) ESP32-Module sollen angeblich unter diesem Problem leiden, aber

STÜCKLISTE



Widerstände:
R1 = 47 k
R2 = 820 Ω
R3, R4, R8 = 330 Ω
R5 = 22 Ω
R6, R7, R9 = 10 k

Kondensatoren:
C1 = 100 µ, 16 V
C2, C3, C6 = 100 n
C4 = 220 µ, 16 V
C5 = 47 µ, 16 V
C7 = 1 µ, 16 V

Halbleiter:
D1 = 1N5817
IC1 = MAX7219
IC2 = TSOP34338
LED1 = blau, 3 mm
LED2 = orange, 3 mm
LED3 = rot, 3 mm
LED4 = TSAL6400, IR-LED, 5 mm
LED5...LED12 = 7-Segment-Ziffer 0,56“, CC (z. B. SC56-11EWA)
T1 = BC337

Außerdem:
BUZ1 = Piezo-Summer ohne Oszillator, 12 mm, 3,3 V
K1 = 1x4-polige Stiftleiste
K2 = 1x6-polige Stiftleiste
K3 = Klinkenbuchse
MOD1 = ESP32-PICO-Kit
S1, S2 = Taktiler Schalter 6x6 mm
Platine 220465-1
Gehäuse Hammond 1591CTDR (transluzent rot)
DS3231 Echtzeituhr-Modul (RTC)

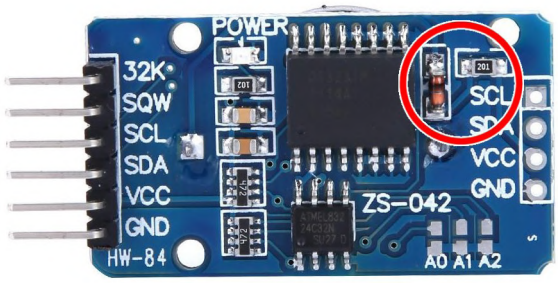


Bild 6. Wenn das (optionale) DS3231-RTC-Modul von einer nicht-wiederaufladbaren CR2032-Knopfzelle versorgt wird, entfernen Sie besser den Widerstand oder die Diode (oder beides) im roten Kreis, um eine Beschädigung der Batterie zu vermeiden.

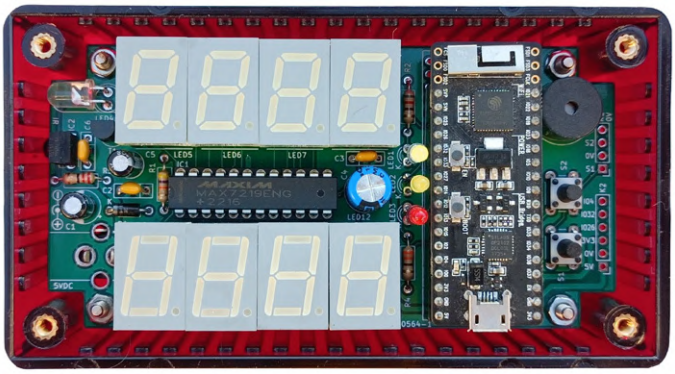


Bild 7. Die montierte Platine passt gut in das Gehäuse. Sie können die Platine aber auch gedreht auf dem Deckel montieren, um die vier Gehäuseschrauben zu verbergen. Der Stromversorgungsanschluss kann auf beiden Seiten der Platine angebracht werden.

ich habe es nie bemerkt. Wenn Sie sich entscheiden, das Modul zu befestigen, montieren Sie es liegend und von R6 und R7 weg zeigend. Der Alarmsummer BUZ1 (ohne integrierten Oszillator) wird von einem PWM-Signal an Port IO27 des ESP32 gesteuert. Die Ports IO34 und IO35 des ESP32 sind mit den Tastern S1 und S2 sowie dem Anschluss K1 verbunden. Bei einem typischen Aufbau würden S1 und S2 oben auf der Uhr statt auf der Platine montiert und an K1 angeschlossen werden. Obwohl der ESP32 integrierte Pull-ups von einigen hundert Kilo-Ohm hat, werden zusätzlich niederohmigere Pull-up-Widerstände R6 und R7 von 10 kΩ eingesetzt, um bei den auf diese Weise verdrahteten Taster keine Probleme mit Störungen zu bekommen.

Drei (Debug-)LEDs (LED1, LED2 und LED3) zeigen den Status der WLAN-Verbindung (blau), die Verbindung zum Webserver (orange) und den ein- und ausgehenden IR-Datenverkehr (rot) an.

IR-Schnittstelle

C5 und C6 entkoppeln den Infrarotempfänger IC2, um ein möglichst sauberes IR-Signal zu erhalten. Die Infrarot-Sende-LED (LED4) wird von Transistor T1 gesteuert. Der Strom durch sie wird durch R5 auf etwa 150 mA begrenzt. Der IR-Ausgang ist kräftig genug, um alle Geräte in der Nähe auf die gleiche Weise zu steuern wie Originalfernbedienungen.

Stromversorgung

Die externe 5-V_{DC}-Stromversorgung muss mindestens 500 mA liefern können. Die Diode D1 schützt die Schaltung vor einer verpolt angeschlossenen Spannung. Zwei Elektrolytkondensatoren (C1, C4) puffern Stromspitzen, kleine parallel geschaltete 100-nF-Kondensatoren (C2, C3) dämpfen höhere (unerwünschte) Frequenzen.

Erweiterungen

Der Anschluss K2 ermöglicht die Erweiterung der Uhr zum Beispiel mit einem Echtzeituhrmodul (siehe unten). Hierfür stehen die drei I/O-Ports IO4, IO26 und IO32 zur Verfügung.

Optionale Echtzeituhr (RTC)

Ein DS3231 RTC-Modul kann an den Erweiterungsbus K2 angeschlossen werden. Die RTC übernimmt den Betrieb, wenn der NTP-Server aus irgendeinem Grund nicht erreichbar ist (wenn beispielsweise die WLAN-Verbindung unterbrochen ist oder ein Problem beim Internet-Provider vorliegt). Das RTC-Modul wird von der Software automatisch erkannt und konfiguriert und einmal pro Tag (um 12:00 Uhr) aktualisiert. Der DS3231 ist ein sehr leistungsfähiger Chip, der dank eines integrierten Temperatursensors eine Quarzdriftkompensation bietet. Das RTC-Modul DS3231 verfügt über eine I²C-Schnittstelle, so dass der Anschluss an K2 nur vier Drähte erfordert: GND, 3,3 V, SDA und SCL. Wenn ein RTC-Modul erkannt wird, übernimmt Port IO26 das SDA-Signal und Port IO32 das SCL-Signal des I²C-Busses.

Gefahr!

Seien Sie vorsichtig, diese RTC-Module können gefährlich sein! Das Modul verfügt zwar über eine Batterieladeschaltung, aber oft ist auf der Rückseite des Moduls eine nicht wiederaufladbare CR2032-Batterie zu finden, die irgendwann explodieren kann. Es gibt zwei Lösungen für dieses Problem. Die einfache, aber teurere Lösung besteht darin, die Batterie durch eine wiederaufladbare LIR2032-Batterie zu ersetzen. Die zweite Lösung ist, die Diode oder den 200-Ω-Widerstand (oder beides) in der Nähe des SCL-Pins des 4-poligen Verbinders zu entfernen, siehe **Bild 6**.

Mechanische Details

Ein gutes Gehäuse für Cloc ist das 1591CTDR von Hammond, das 120 mm x 63 mm x 38 mm misst. Es ist durchscheinend rot (für IR) und perfekt für die 7-Segment-Anzeigen der Tageszeit und der Alarmzeit (rot und/oder orange). Auch die Übertragung und der Empfang von IR-Signalen profitieren von einem solchen optischen Filter. Beachten Sie aber, dass Licht mit geringem oder keinem Rotanteil (wie von der blauen LED) aufgrund dieser Filterung kaum sichtbar ist.

Und wenn wir schon beim Thema LEDs sind: Das ESP32-PICO-Kit hat eine (viel zu) helle rote Power-LED, die Sie wahrscheinlich verstecken oder entfernen möchten.

Die Platine für die Uhr wurde mit diesem Gehäuse im Hinterkopf entworfen. Das KiCad6-Projekt kann von [1] heruntergeladen werden. Die auf vier 16-mm-Bolzen montierte Platine passt perfekt in das Gehäuse, wie in **Bild 7** gezeigt. Zur Befestigung der Platine müssen lediglich vier 3,2-mm-Löcher in den Boden gebohrt werden, ein 8-mm-Loch wird an der Seite für den Stromanschluss benötigt, und zwei 12-mm-Löcher an der Oberseite sind zwei schönen Drucktasten vorbehalten.

Das optionale RTC-Modul kann unter der Hauptplatine mit der Bauteilseite nach oben an K2 angeschlossen werden, wobei die 4-polige unbestückte Stiftleiste verwendet wird (die 6-polige abgewinkelte Stiftleiste sollte entfernt werden, um Kurzschlüsse zu vermeiden). Sie können das RTC-Modul auch mit doppelseitigem Klebeband auf die Hauptplatine kleben (wie ich es getan habe).

Die Software-Entwicklungsumgebung

Ich habe die Arduino-IDE mit hinzugefügten ESP32-Board-Paket verwendet, um die Software für Cloc 2.0 zu entwickeln. Um Fehler bei der Kompilierung oder Speicherverwaltung zu vermeiden, müssen Sie das Board *ESP32 PICO-D4* mit einer CPU-Frequenz von 240 MHz auswählen. Für das *Partitions Scheme* wählen Sie *Default* (das heißt, 4 MB mit SPIFFS). Wenn Sie die richtige serielle Schnittstelle gewählt haben, sollte das Kompilieren und Hochladen des Programms reibungslos funktionieren!

Eigenschaften

- Basierend auf einem ESP32-Prozessor
- Zwei unabhängige 7-Segment-Anzeigen für Tageszeit und Alarmzeit
- Die Uhrzeit wird automatisch von einem Online-Zeitserver eingestellt
- Alarmzeit für jeden Tag der Woche
- Der Alarmausgang: Summer und Infrarotcode für beispielsweise Radio, HiFi-Anlage oder TV
- Zwei Drucktasten für die Interaktion mit dem Gerät
- Integrierter Webserver für die Fernkonfiguration über WLAN
- Alle Einstellungen werden im EEPROM gespeichert
- Over-the-Air-Modus ermöglicht die Aktualisierung der Firmware per Fernzugriff
- Optionales Echtzeituhr-Modul DS3231 mit Pufferbatterie.

Die folgenden externen Bibliotheken müssen in der IDE installiert werden, bevor das Programm kompiliert wird. Ich habe im Quellcode angegeben, wo man sie bekommt, aber die meisten können auch über den Bibliotheksmanager der IDE bezogen werden.

- AsyncElegantOTA
- AsyncTCP
- ESPAsyncWebServer
- IRremote (siehe Zeile 126 von `Alarm_Clock.ino`).
- LedControl
- RTClib

Wichtig! `LedControl.h` muss geändert werden. Die Zeile 30 (`#include <avr/pgmspace.h>`) muss auskommentiert werden, wie dies auch im Kommentar im Quellcode beschrieben ist.
Der Download [1] enthält einen Ordner mit dem Namen *libraries*, der alle verwendeten Bibliotheken enthält, darunter auch die bereits modifizierte `LedControl.h`. Kopieren Sie diese Bibliotheken in den Unterordner *libraries* des Arduino-Sketchbook-Ordners. Sie können sehen, wo er sich befindet, wenn Sie *Datei -> (Vor)Einstellungen* der IDE aufrufen.

Unter [2] erfahren Sie, wie Sie die Option *ESP32 Sketch Data Upload* zur IDE hinzufügen, damit Sie das Webinterface der Uhr in den Datenspeicher des ESP32-Moduls laden können. Vergessen Sie nicht, diese Daten hochzuladen, sonst wird der Webserver nicht funktionieren.

Um die Firmware des Weckers aus der Ferne zu aktualisieren, gehen Sie mit einem Browser auf die Adresse `IP_of_cloc/update`, wobei `IP_of_cloc` die IP-Adresse des Weckers darstellt. Sie können nun eine ausführbare Datei auf Ihrem Computer auswählen und sie auf die Uhr hochladen. Eine ausführliche Erläuterung der OTA-Schnittstelle finden Sie unter [3].

Programm-Interna

Das Programm ist recht gut dokumentiert, leider oft in französischer Sprache. Nachdem Sie das Folgende gelesen haben, sollten Sie sich aber leicht zurechtfinden. Es empfiehlt sich, die serielle Schnittstelle des ESP32 zu überwachen, da viele Debug-Informationen über diese Schnittstelle ausgegeben werden.

Nach dem Einbinden der Bibliotheken werden die Standardwerte definiert, die entweder beim ersten Booten verwendet werden (wenn Sie die Taste zum Deaktivieren des Alarms (S1) gedrückt halten) oder auf dem Panel Settings, nachdem Sie auf die Schaltfläche Default geklickt haben. Die Reihenfolge der Konstanten folgt der Reihenfolge der Parameter im Panel Settings (Zeilen 40...65 der Datei *Alarm_Clock.ino*). Tragen Sie insbesondere die feste IP-Adresse und das Gateway sowie die WLAN-Anmeldedaten ein. Im Alarm-Panel (Zeilen 67...84) sind die Dinge auf die gleiche Weise organisiert.

Beim Einschalten wird zunächst die Funktion `setup` ausgeführt. Es ist einfach, die Initialisierungsreihenfolge der Cloc-Hardware zu verfolgen.

Die Funktion `initRTC` prüft, ob ein DS3231-RTC-Modul angeschlossen ist. Wenn zu diesem Zeitpunkt die Taste S1 gedrückt gehalten wird, werden die Standardwerte geladen.

Wenn dies der erste Bootvorgang der Uhr ist oder wenn die Taste Stop (S2) während des Einschaltens gedrückt wurde, geht die Uhr in den Access-Point-Modus. In diesem Fall können Sie sich mit einem Browser mit der Web-Schnittstelle 192.168.4.1 verbinden, die von der Uhr angezeigt wird.

Nach dem Abrufen der IP-Adresse (DHCP oder fest) und der Verbindung mit dem WLAN wird die NTP-Zeit gesucht. Wenn sie gefunden wird und eine RTC vorhanden ist, wird letztere synchronisiert. Die IP-Adresse der Uhr wird kurz angezeigt, bevor auf die Zeitanzeige umgeschaltet wird. Wenn die Anzeige HH:HH auf der Zeitanzeige und AA:AA im Alarmdisplay blinkt, konnte der NTP-Server nicht erreicht werden und Sie müssen das Gerät neu starten (wenn keine RTC vorhanden ist; andernfalls zeigt es die RTC-Zeit an). Die Initialisierung endet mit dem Start des Webservers und des `AsyncOTA`-Dienstes, der eine Fernaktualisierung der Firmware und des SPIFFS ermöglicht.

Die Funktion `loop` führt den Rest des Programms in einer Endlosschleife aus. Wenn die RTC vorhanden ist (`flagRTC=1`), liest sie die RTC-Zeit aus. Ohne RTC versucht es die NTP-Zeit (`NTPflag=1`). Sobald die Uhr verbunden ist, liest das Programm die RTC-Zeit. Dabei enthält `strTime` die NTP-Zeit als „hh:mm:ss“, wenn sie gültig ist, ansonsten eine „0“.

Dann wird der Zustand der Tasten überprüft (`checkButtons`) und bestimmt, ob die Uhr Alarm geben muss und ob ein IR-Code empfangen wurde. Da der Server asynchron arbeitet, müssen wir Flags verwenden, um dem Hauptprogramm Parameteränderungen und Alarmauslösungen zu melden.

Der Bildschirm wird jede Sekunde aufgefrischt (`dispTime`) und die Alarme werden überprüft (`checkAlarm`). Außerdem prüft das Programm, ob es sich in der Hochhelligkeitsperiode befindet oder nicht (`checkBrightness`) und ob es die RTC mit der NTP-Zeit synchronisieren muss (`checkSyncRTC`). Dies geschieht einmal pro Tag um 12:00 Uhr (`syncRTC=12*60`). Die Funktion `checkTimer` prüft, ob ein Alarm aktiviert werden muss. Dazu werden die Tageszeit und die Alarmzeit in Minuten umgerechnet (`computeMinutes`) und anschließend verglichen (`testRange`).

Wie erwähnt, arbeitet der Server asynchron. Jedes Mal, wenn sich Settings oder Alarm Panel ändern, werden die anzuzeigenden Daten (`ParametresProcessor` und `TimerProcessor`) zurückgegeben. Zu diesem Zweck müssen die in der HTML-Seite enthaltenen `%value%`-Tags durch die tatsächlichen Werte ersetzt werden.

Für das Haupt-Panel ist es etwas anders, da die Uhrzeit, das Datum und die Start- und Endzeiten des Alarms fast in Echtzeit angezeigt werden müssen. Eine in JavaScript geschriebene Funktion fragt ungefähr jede Sekunde das Hauptprogramm ab, um die anzuzeigenden Werte zu erhalten (`onDataUpdate`). Bei jeder Abfrage blinkt LED2 kurz auf, um anzuzeigen, dass auf den Server zugegriffen wird (`flashRqstBeacon`). Um zu überprüfen, ob die Seite die Werte tatsächlich in Echtzeit anzeigt, drücken Sie S1 (Alarm Disable). Die Alarmzeit wird sowohl auf der 7-Segment-Anzeige als auch auf der HTML-Seite gleichzeitig angezeigt beziehungsweise ausgeschaltet.

Bezüglich der IRremote-Bibliothek

Die `IRremote`-Bibliothek funktioniert einwandfrei, erfordert jedoch einige Integrationsregeln. Verschieben Sie die Initialisierungssequenzen nicht, wenn Sie nicht wirklich wissen, was Sie tun.

Um statt mit einem gräßlichen Piepton aufzuwachen, lasse ich mich von wohlklingender Musik meines Bose-Radio wecken. Daher habe ich den Empfang und die Dekodierung auf Bose-Fernbedienungen beschränkt (siehe Zeile 125 von `Alarm_Clock.ino`). Sie können diese Zeile entfernen, um andere Fernbedienungen zu dekodieren. Ich empfehle, den Infrarotempfang auf die Marke Ihres Radios oder Ihrer HiFi-Anlage zu beschränken. Siehe [4] für mögliche Optionen.

Auch die IR-Übertragung ist auf das Bose-Format beschränkt. Daher ist es sehr wahrscheinlich, dass Sie den Code an die Marke der von Ihnen verwendeten Hi-Fi-Anlage anpassen müssen. Ersetzen Sie die Zeile `IrSender.sendBoseWave` in den Funktionen `startMedia` und `stopMedia` in `Alarm_Clock.ino` durch eine Funktion, die Ihrer Marke entspricht (siehe wiederum [4]).

Um herauszufinden, welche IR-Codes für Ihr (Audio-)System zu verwenden sind, verbinden Sie den USB-Anschluss des ESP32-Moduls mit einem Computer, auf dem ein serieller Monitor läuft, zum Beispiel der in der Arduino-IDE integrierte *Serielle Monitor*. Richten Sie Ihre Fernbedienung auf den IR-Empfänger der Uhr und drücken Sie eine Taste, um den entsprechenden Code in hexadezimaler Form auf dem Computer anzuzeigen. Geben Sie die Codes in hexadezimaler Form in die Felder *Media ON* und *Media OFF* des Einstellungsfeldes ein. IR-Aktivität wird durch kurzes Blinken der LED3 angezeigt. Folglich blinkt diese LED auch, wenn ein Alarm aktiv ist.

Eine DIY-Lösung

Ich benutze diese neue Version von Cloc 2.0 nun schon seit einigen Monaten und habe noch nie verschlafen, weil das Gerät nicht funktioniert hat. Die Möglichkeit, Weckzeiten für jeden Tag der Woche einzustellen, ist sehr praktisch.

Ich empfehle dringend, die RTC-Option hinzuzufügen, denn wenn der NTP-Server nicht antwortet (weil beispielsweise Ihr Modem aus irgendeinem Grund offline ist), verliert der Wecker die Zeit, wenn die nächste Neusynchronisierung der NTP-Zeit angesagt ist. Sie müssen dann den Wecker anhalten und neu starten, wenn die NTP-Zeit wieder verfügbar ist. Für noch mehr Zuverlässigkeit habe ich einen kleinen NTP-Server gebaut, der die Zeit von einem GPS empfängt [5]. So kann sich mein Wecker auch bei einem Abbruch der Internetverbindung automatisch einstellen.

Ich hoffe, dass viele von Ihnen diesen Wecker bauen werden. Bitte zögern Sie nicht, mich zu kontaktieren (siehe Kasten **Fragen oder Kommentare**), wenn Sie auf Schwierigkeiten stoßen oder Verbesserungen an meinem Lieblingswecker vornehmen möchten!

(220564-02)RG

Haben Sie Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Senden Sie eine E-Mail an den Autor unter yb.electronique@orange.fr oder kontaktieren Sie Elektor unter redaktion@elektor.de.

Passende Produkte

- **ESP32-PICO-Kit V4 (SKU 18423)**
  www.elektor.de/18423
- **Elektor Arduino Electronics Bundle (SKU 19440)**
  www.elektor.com/19440
- **SparkFun Real Time Clock Modul – RV-8803 (Qwiic) (SKU 19646)**
  www.elektor.de/19646

WEBLINKS

[1] Downloads für Cloc 2.0 bei Elektor Labs: https://www.elektormagazine.de/labs/cloc-le-reveil-20
[2] ESP32 SPIFFS Daten-Upload: https://randomnerdtutorials.com/install-esp32-filesystem-uploader-arduino-ide/
[3] ESP32 OTA-Updates: https://randomnerdtutorials.com/esp32-ota-over-the-air-arduino/
[4] IRremote-Bibliothek: https://github.com/Arduino-IRremote/Arduino-IRremote
[5] NTP-Server bei Elektor Labs: https://www.elektormagazine.de/labs/ntp-server

Raspberry Pi Pico: PIO in der Praxis

Experimente mit der Programmable I/O des RP2040

Von Prof. Dr. Martin Ossmann (Deutschland)

Das Raspberry Pi Pico Board ist mit nur etwa 4 € nicht nur unglaublich billig, sondern auch sehr leistungsfähig. Die verwendete MCU RP2040 verfügt zudem über ein einzigartiges Feature: Eine programmierbare Ein-/Ausgabe (Programmable I/O = PIO). In diesem Artikel wird das Ganze anhand von Anwendungsbeispielen vorgestellt.

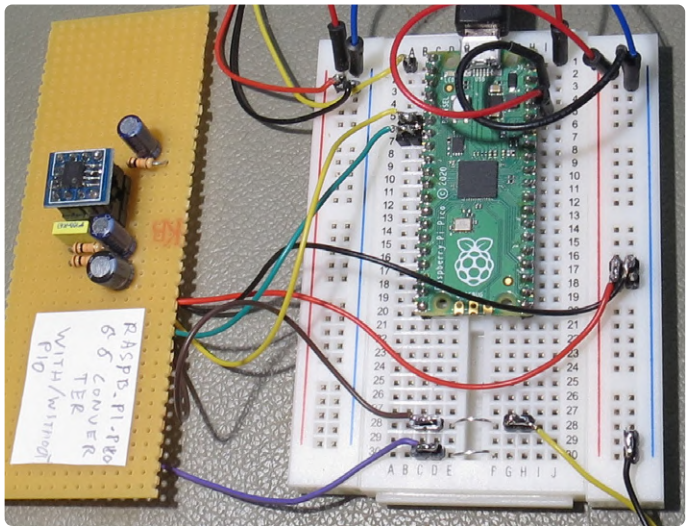


Bild 1. Das Raspberry Pi Pico Board mit der Delta-Sigma-Erweiterung.

Dass sich mit einer programmierbaren I/O-Einheit schöne Dinge anstellen lassen, ist unmittelbar einleuchtend, wenn man schon einmal Mikrocontroller programmiert hat. Die Frage ist, ob das kompliziert ist und ob sich der Aufwand lohnt. Die Antworten auf diese Fragen können Sie sich anhand der folgenden praxisnahen Beschreibung sicher selbst geben, denn die Beispiele machen klar, wie genau man die PIO beim RP2040 einsetzt. **Bild 1** zeigt meinen experimentellen Hardware-Aufbau, auf dem die besprochenen Beispiele laufen.

Der PIO-Aufbau

Die programmierbare Ein-/Ausgabe dient dazu, zusätzlich zu den vorhandenen Standards wie SPI und I²C weitere Protokolle mit den I/O-Pins zu realisieren. Sie besteht aus zwei PIO-Blöcken, die jeweils über vier I/O-Prozessoren verfügen. Ein einzelner I/O-Prozessor ist dabei wie in **Bild 2** gezeigt in die Hardware eingebunden.

Der eigentliche Kern verfügt über die 32-bit-Universalregister X und Y. Dazu kommen zur Ein/Ausgabe noch zwei 32-bit-Register, die auch als Schieberegister fungieren können. Zur Eingabe gibt es das ISR (**I**nput **S**hift **R**egister) und zur Ausgabe das OSR (**O**utput **S**hift **R**egister). Die PIO-Kerne sind über jeweils vier Einträge tiefe FIFOs mit der RP2040-CPU verbunden. Die TX-FIFO-Einheit dient naheliegenderweise zum Senden und die RX-FIFO-Einheit zum Empfangen. Die Einträge sind 32 bit breit. Beide FIFO-Einheiten können, wenn man nur eine Richtung benötigt, als acht Einträge tiefe FIFOs geschaltet werden. Jeder PIO-Prozessor verfügt zur Takterzeugung über einen

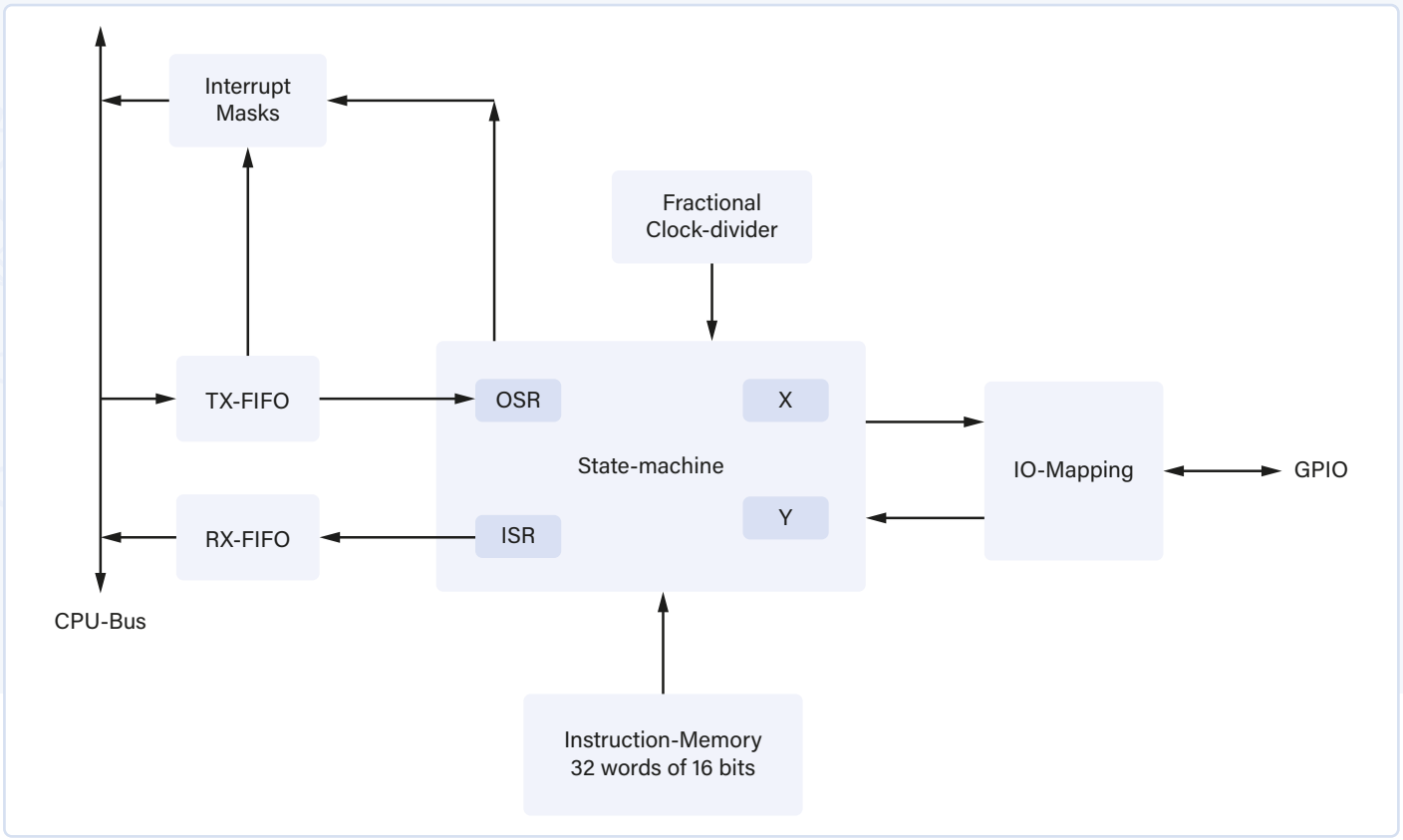


Bild 2. Blockschaltbild eines I/O-Prozessors (Quelle [4]).

16.8 bit Fractional Divider (siehe unten). Damit kann man die Taktrate jedes Prozessors sehr flexibel und genau zwischen 2 kHz und 125 MHz festlegen.

Die PIO-Prozessoren verstehen nur neun Befehle. Die PIO-Befehle sind 16 bit breit und kommen aus einem Speicher, der 32 Befehle umfasst. Diese 32 Befehle stehen allen vier PIOs eines PIO-Blockes gemeinsam zur Verfügung. Die einzelnen PIO-Programme sind kurz genug, um in diesen kleinen Speicher zu passen. Der Speicher wird von der RP2040-CPU gefüllt. Die zugehörige Programmiersprache heißt PIO-Assembler und ist flexibel genug, um damit kurze PIO-Programme schreiben zu können. Die Ausführung jedes Befehls dauert einen Taktzyklus.

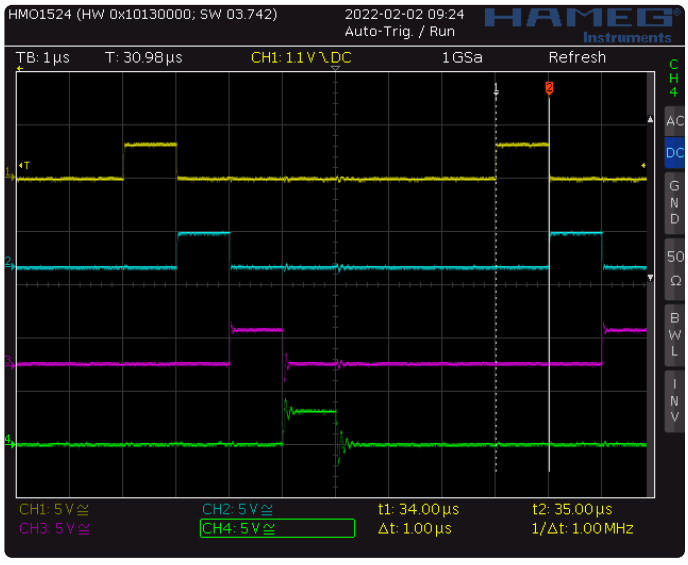


Bild 3. Signale des ersten Beispielprogramms.

Ein einzelner PIO-Prozessor kann nur wenige GPIO-Pins adressieren, und er verwendet dafür kurze Adressen. Welche Pins dann konkret angesprochen werden, legt das IO-Mapping fest. Dafür gibt es mehrere programmierbare Konstanten, die für den jeweiligen Bereich (Input, Output, Side-Set und Set) festlegen, welches der erste adressierte Pin ist. Dadurch lassen sich die IO-Pins eines PIO-Prozessors sehr flexibel festlegen. Mehrere Prozessoren können auch den gleichen Pin beeinflussen.

Tools

Als Entwicklungsumgebung wurde VS Code verwendet. Die Installation und die Benutzung dieses Tools sind in [1] beschrieben. Dort finden sich auch Hinweise zur Benutzung des Projekt-Generators. Mit diesem erzeugt man ein Raspberry-Pi-Projekt mit allen notwendigen Dateien (Quellcode, Konfigurationsdatei, Make-Setup etc.).

Dabei legt man auch fest, ob man die `printf`-Ausgabe per USB oder per UART machen möchte. Man konfiguriert hier ebenfalls, welche Libraries man benutzen will. In diesem Beitrag war das beispielsweise das PIO-Tool. Hilfestellung zur VS-Code-Installation erhält man zusätzlich in [2]. Dort wird auch detailliert beschrieben, welche zusätzlichen Tools (Make, Git und so weiter) man installieren muss.

Alle Softwarebeispiele sind unter [3] herunterzuladen.

Ein erstes Programm

Das erste Programm soll schlicht einen Impuls von 1 µs Länge erzeugen und diesen, wie in **Bild 3** gezeigt, nacheinander an vier Ausgänge legen. Zur Überprüfung des entstehenden Signals wird ein Oszilloskop

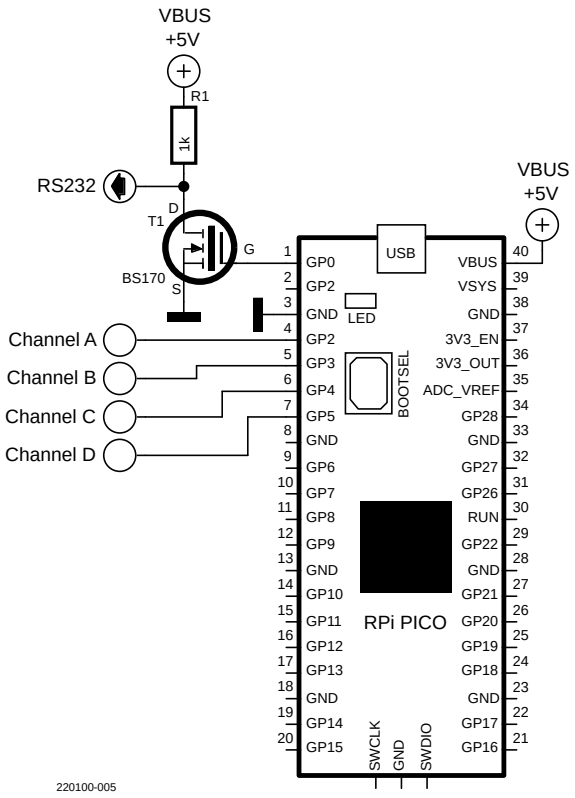


Bild 4. Anschluss eines Vierkanal-Oszilloskops an den Raspberry Pi Pico für das erste Beispiel.

angeschlossen. **Bild 4** zeigt, wie die einzelnen Kanäle eines Vierkanal-Oszilloskops mit den I/O-Pins des Raspberry Pi Pico verbunden sind. Das folgende Listing enthält das erste Beispiel-PIO-Programm. Nach drei Zeilen mit Assemblerdirektiven steht der erste ausführbare Befehl in Zeile 4. Es werden vier Impulse erzeugt.

Listing 1:

```
001 .program pioTest
002 .side_set 2 opt
003 .wrap_target
004   nop           side 0b01 // GP2=1, GP3=0
005   nop           side 0b10 // GP2=0, GP3=1
006   set pins,0b01 side 0b00 // GP2=0, GP3=0,
                                GP4=1, GP5=0
007   set pins,0b10       //    GP4=0,      GP5=1
008   set pins,0b00  [2]  //    GP4=0,      GP5=0
009 .wrap
```

Die Zeilen 4 und 5 enthalten den `nop`-Befehl, der einfach nichts tut. In Zeile 4 wird der Befehl durch den Text `side 0b01` erweitert. Dies bewirkt, dass als Seiteneffekt an Side-Pin 0 eine „1" ausgegeben wird und an Side-Pin 1 eine „0". Jeder PIO-Befehl kann durch einen solchen Seiteneffekt ergänzt werden. Dadurch kann man leicht mehrere Pins zusätzlich beeinflussen. Dazu muss man natürlich auch festlegen, welches die Side-Set-Pins sind, mit Konfigurationsbefehlen in der Programmiersprache C. Die Festlegung geschieht in nächsten Listing durch `sm_config_set_sideset_pins()` in Zeile 5. Dies bewirkt, dass die Side-Set-Pins bei GP2 beginnen.

Listing 2:

```
001 #define set_base 4   // pins start at GP4
002 #define set_count 2  // number of pins is 2
003 sm_config_set_set_pins (&c,set_base,set_count);
004
005 #define side_set_base 2 // side-pins start at GP2
006 sm_config_set_sideset_pins(&c, side_set_base);
007
008 float div = 125.0;
009 sm_config_set_clkdiv(&c, div);
```

Der `nop`-Befehl in Zeile 4 des ersten Listings erzeugt den Impuls an Side-Pin 1 (= GP3). In Zeile 6 steht ein `set`-Befehl, der die Pins 0 und 1 der Set-Pins setzt. Damit gelangt der Impuls an Pin GP4. Als Seiteneffekt werden die Side-Pins 0 (GP2) und 1 (GP3) zurückgesetzt. In der Konfigurations-Routine im zweiten Listing in Zeile 3 wird festgelegt, dass die Set-Pins ab Pin GP4 beginnen. Mit dem `set`-Befehl kann man Register oder IO-Pins mit konstanten Werten laden.
Die Ergänzung `[2]` hinter dem `set`-Befehl in Zeile 8 des ersten Listings bewirkt, dass auf den Befehl eine Verzögerung um zwei Takte folgt. Durch diese Angabe in eckigen Klammern kann man jeden Befehl um einen bis 31 Takte verzögern.
Jetzt muss nur noch durch einen geeigneten Takt dafür gesorgt werden, dass jeder Impuls genau 1 µs dauert. Dazu muss man den Systemtakt von 125 MHz durch 125 dividieren. Mit der resultierenden Taktrate von 1 MHz dauert jeder Takt die erwünschten 1 µs. Dies wird in der Konfigurations-Routine des zweiten Listings in den Zeilen 8 und 9 festgelegt. Die Assembler-Direktiven `.wrap_target` und `wrap` im ersten Listing bewirken, dass die Befehle zwischen ihnen zur Laufzeit in einer Schleife endlos wiederholt werden. Dabei kostet der Rücksprung an den Schleifenanfang (`.wrap_target`) keine Zeit. Man kann also „zero-cycle-overhead-Loops" programmieren. Bei unserem Programm dauert das einmalige Durchlaufen der Schleife also sieben Taktzyklen (fünf Befehle + zwei Zyklen Verzögerung).

Takterzeugung: Fractional Divider

Wie schon erwähnt wird der Takt für einen PIO-Prozessor mit Hilfe eines „Fractional Dividers" aus dem 125-MHz-Systemtakt erzeugt. Der 16-bit-Divider kann maximal bis 2^{16} = 65.536 zählen und hat acht binäre Nachkommastellen. Die Genauigkeit ist damit 1/256. Der „gebrochene Teiler" wird dabei so wie in DDS-Signalgeneratoren realisiert. Der Nachkommateil wird fortlaufend addiert und bei jedem Übertrag

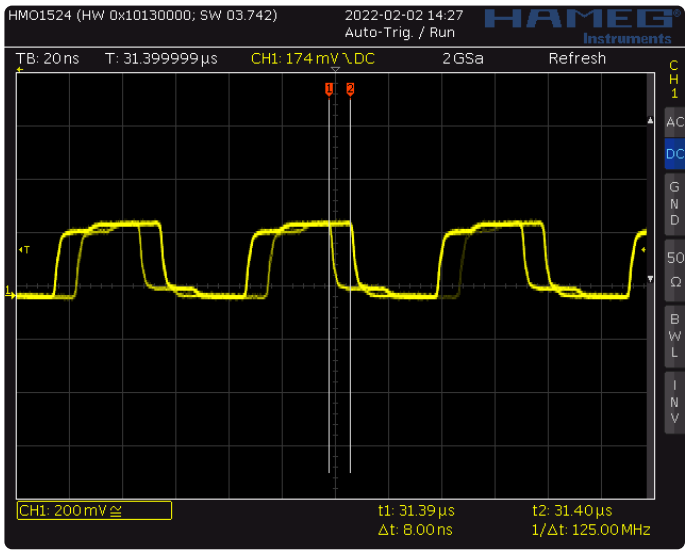


Bild 5. Die gebrochene Taktrate führt zu deutlichem Jitter beim generierten Takt.

wird eine Taktflanke erzeugt. Im folgenden Listing wird die Einstellung einer gebrochenen Taktrate demonstriert.

Listing 3:

```
001 float clockFrequency=28e6;
002 float div=125e6/clockFrequency;
003 sm_config_set_clkdiv(&c,div);
```

Durch die Art der Takterzeugung schwankt die Taktperiode um einen Systemtakt und es entsteht ein Jitter. Im Beispiel soll eine Taktrate von 28 MHz erzeugt werden. Da 28 kein Teiler von 125 ist, hat der Takt-Teiler logischerweise Nachkommastellen. Der Takt-Teiler ist 125/28 = 4,4642857...
Das dritte Listing zeigt, wie dieser Teiler zur Konfiguration genutzt wird. Zur Demonstration wird mit dem PIO-Programm im folgenden vierten Listing ein einfaches Rechtecksignal erzeugt.

Listing 4:

```
001 .program theProgram
002 .side_set 1 opt
003     nop    side 0
004     nop    side 1
```

Das Oszilloskop zeigt in **Bild 5** sehr deutlich, dass der Takt einen Jitter aufweist. Bei größeren Werten des Taktteilers fällt der Jitter natürlich viel weniger ins Gewicht, so dass man ihn oft vernachlässigen kann. Mit dem „Fractional Divider" kann man dann zum Beispiel Baudraten wie etwa die typischen 115.200 bit/s genau genug erzeugen.

Serielles Senden: TX-UART

Das nächste Beispiel ist etwas praxisnäher: Es geht um das serielle Senden von Daten über eine RS232-Schnittstelle. Die Schnittstelle wird dabei an Pin GP4 angeschlossen, damit GP0 am eingebauten UART für Debug-Zwecke freibleibt. Da das Interface invertiert, muss man ein invertiertes Signal erzeugen (idle high). Die Impulsform an Pin GP4 sieht beim Senden des ASCII-Zeichens dann so aus wie in **Bild 6**.
Zusätzlich zum TXD-Signal soll der UART während des Sendens der acht Datenbits eine 1 auf einer Busy-Leitung ausgeben. Als Busy-Leitung soll dabei GP5 (side-Pin 0) verwendet werden. Das zugehörige PIO-Programm ist im folgenden Listing zu sehen.

Listing 5:

```
001 .program theProgram
002 .side_set 1 opt
003     pull              [1]
004     set pins,0 side 1 [2] // startBit=0
005     set x,7               // loop for 7+1 times
006 bit:
007     out pins,1        [2] // LSB first
008     jmp x-- bit
009     set pins,1 side 0 [2] // stopBit=1
```

Der `pull`-Befehl in Zeile 3 entnimmt ein 32-bit-Wort aus der TX-FIFO-Einheit und speichert es im OSR. Wenn in der TX-FIFO-Einheit kein Wort gespeichert ist, wartet der Befehl solange, bis ein Wort in der FIFO-Einheit gespeichert wird und fügt dieses dann direkt in das OSR ein. Durch den set-Befehl in Zeile 4 wird der Ausgabe-Pin als Start-Bit auf 0 gesetzt.
Die acht niederstwertigen Bits des Wortes im OSR werden nun in einer Schleife ausgegeben. Dazu muss man den Schleifenzähler initialisieren. Als Schleifenzähler dient das X-Register. Der set-Befehl in Zeile 5 initialisiert es mit dem Wert 7.
Das Label `bit` in Zeile 6 markiert den Schleifen-Anfang. Der erste und einzige Befehl in der Schleife ist der `out`-Befehl in Zeile 7.

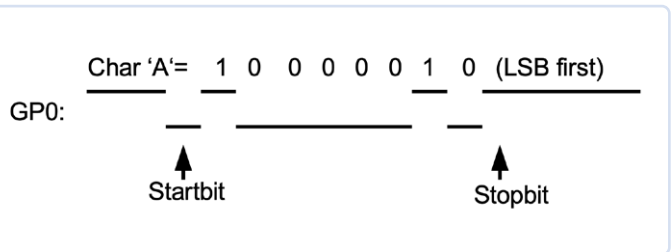


Bild 6. Impulsdiagram beim Senden des Zeichens „A".

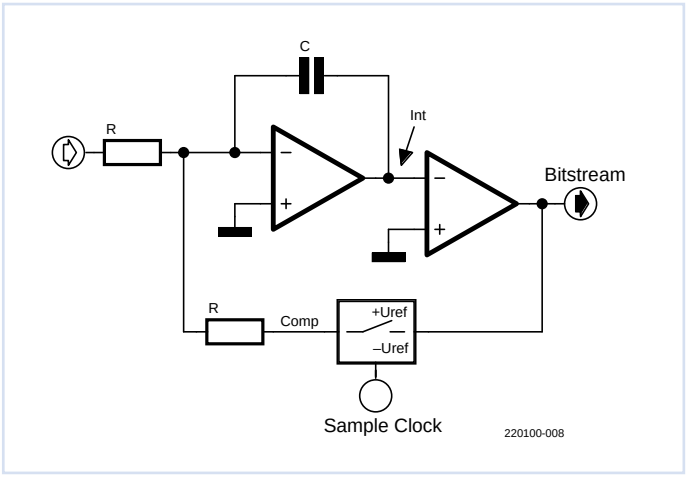


Bild 7. Prinzipschaltung eines Delta-Sigma-ADC.

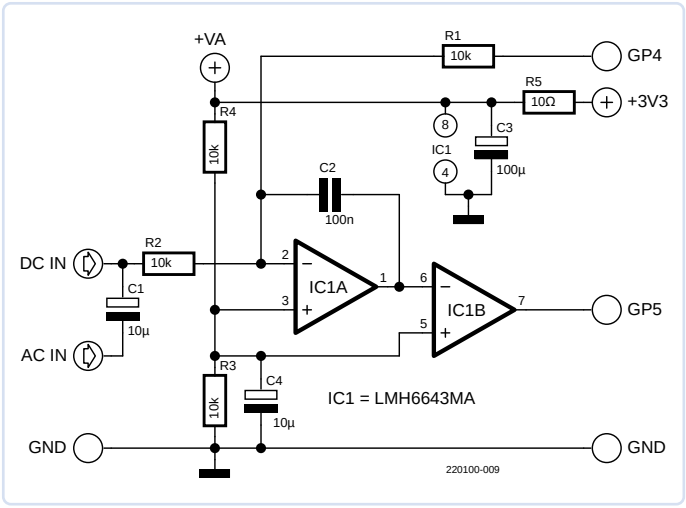


Bild 8. Konkrete Schaltung des Delta-Sigma-ADC nach Bild 7.

Dieser gibt das LSB aus dem OSR am Out-Pin 0 aus und schiebt die Bits im OSR um eine Position nach rechts. Die Out-Pins bilden neben den Set-Pins und den Side-Set-Pins das dritte Fenster der GPIO-Pins. Als Out-Pin wurde ebenfalls Pin GP4 eingestellt.

Der bedingte Sprung in Zeile 8 mit der Option `x--` verringert das X-Register um 1 und springt dann an die Position `bit` (Schleifenanfang), wenn `x` größer als 0 ist. Auf diese Weise wird die Schleife acht Mal durchlaufen. Der letzte Befehl der TX-Sende-Routine ist der `set`-Befehl in Zeile 9, der das Stopp-Bit erzeugt, indem er Set-Pin 0 auf 1 setzt. Durch die beiden Side-Erweiterungen wird das Busy-Signal am Side-Pin 0 (GP5) erzeugt.

Die Verzögerungs-Erweiterungen in den eckigen Klammern bewirken, das jedes Bit genau vier Takte lang ist. Die Baudrate entspricht also der Taktrate geteilt durch vier. Das Setzen von `clkDiv` auf 125.000.000 / (4 * 115.200) ergibt folglich die gewünschte Baudrate von 115.200 bit/s.

Zur Übergabe von Zeichen mit einem C-Programm an den PIO-UART dient die Routine `pio_sm_put_blocking(…)`, wie die in Routine `PIOprint()` im nächsten Listing zeigt. Sie übergibt hier ein Zeichen an die TX-FIFO-Einheit. Falls die Einheit voll ist, blockiert sie und wartet, bis wieder Platz vorhanden ist.

Listing 6:

```
001 void PIOprint(const char *s) {
002    while (*s)
003       pio_sm_put_blocking(pio, sm, *s++);
004    }
```

Die TX-UART-Routine zeigt schön, dass man mit nur sechs Befehlen sinnvolle Funktionen erreichen kann, wobei Side-Set- und Verzögerungs-Befehlsergänzungen eine wichtige Rolle spielen.

Delta-Sigma-ADC

Im folgenden Abschnitt soll ein Delta-Sigma-A/D-Wandler mit Hilfe der PIO realisiert werden. Das Prinzip eines solchen Wandlers ist in **Bild 7** zu sehen. Dabei wird versucht, die Eingangsspannung mit einer schaltbaren Spannungsquelle zu kompensieren. Ein Integrator am Eingang integriert das Fehlersignal und ein nachfolgender Komparator bestimmt das Vorzeichen der Kompensationsspannung im nächsten Zeitintervall. Am Ausgang entsteht so eine Bitfolge, die im Mittel der Eingangsspannung folgt. **Bild 8** zeigt eine so konzipierte konkrete Schaltung.

Diese Schaltung wird an GP4 und GP5 des Raspberry Pi Pico angeschlossen. Via PIO wird die Bitfolge ermittelt und jeweils acht Bit gesammelt in einem Byte in der RX-FIFO-Einheit abgelegt. Der Mikrocontroller entnimmt dann die Bytes aus dieser Einheit und verarbeitet sie weiter. Das zugehörige PIO-Programm ist im folgenden Listing zu sehen.

Listing 7:

```
001 .program hello
002 Loop1:
003    set   x,7
004 inLoop:
005    in    pins,1
006    mov   osr,~isr
007    out   pins, 1
008    jmp   x-- go1   [20]
009    push  noblock
010    jmp   Loop1     [23]
011 go1:
012    jmp   inLoop    [26]
```

Die äußere Schleife ab dem Label `Loop1` wird endlos wiederholt. Die innere Schleife macht immer acht Durchläufe. Jeder Durchlauf beginnt mit dem Befehl hinter dem Label `inLoop`. Der `in`-Befehl in Zeile 5 holt

ein Bit vom Input-Pin und schiebt es in das ISR. In Zeile 6 wird das ISR invertiert ins OSR kopiert. In Zeile 7 wird das niederstwertige Bit am Ausgabe-Pin ausgegeben, was den nächsten Kompensationsschritt im Integrator bewirkt.
Das eingelesene Bit ist das nächste Ausgabe-Bit des Delta-Sigma-Wandlers. Acht Bit werden im ISR gesammelt. Der `jmp`-Befehl in Zeile 8 kontrolliert die innere Schleife. Ist sie acht Mal durchlaufen, wird der `push`-Befehl in Zeile 9 ausgeführt. Dieser Befehl speichert das ISR in der RX-FIFO-Einheit, aus dem es die MCU entnehmen kann. Die Verzögerungs-Befehlserweiterungen in eckigen Klammern sorgen dafür, dass jeder Delta-Sigma-Schritt 50 Taktimpulse dauert. Der Taktteiler wurde auf 25 eingestellt. Die Abtastrate des Delta-Sigma-Converters ist daher 125 MHz / (25*50) = 100 kHz.
Zur Demonstration wurde ein Sinussignal mit 50 Hz durch den Delta-Sigma-Converter gewandelt. **Bild 9** zeigt das resultierende Spektrum. Der Abstand zwischen Nutzsignal und Rauschen liegt bei circa 60 dB. Das ist gar nicht schlecht für eine solch einfache Schaltung!

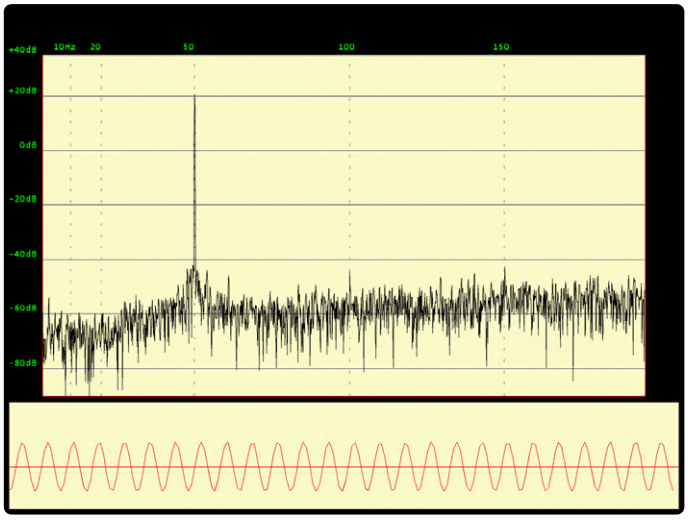


Bild 9. Spektrum eines 50-Hz-Sinussignals nach Delta-Sigma-Wandlung.

Signalgenerator

Als Nächstes wird ein nach **Bild 10** konzipierter R2R-DAC an das Board des Raspberry Pi Pico angeschlossen und damit ein Signalgenerator realisiert. Der Generator soll ein periodisches Signal erzeugen. Dazu müssen Daten periodisch aus einer Tabelle im Speicher zum D/A-Wandler transportiert werden. Das klappt besonders bequem per DMA (**D**irect **M**emory **A**ccess). Der DMA-Controller des Raspberry Pi Pico überträgt hierfür die Daten in die TX-FIFO-Einheit. Das PIO-Programm besteht aus einem einzigen ausführbaren Befehl, wie nachfolgend gezeigt.

Listing 8:

```
001 .program pio_serialiser
002
003 .wrap_target
004     out pins,8 // use autopull
005 .wrap
```

Der `out`-Befehl bringt die acht niederstwertigen Bits aus dem OSR zu den acht out-Pins GP0...GP7. Danach wird das OSR automatisch aus der TX-FIFO-Einheit neu geladen, weil in der Konfigurationsroutine das PIO-Feature *auto-pull* aktiviert wurde. Mit dem nächsten Befehl werden dann die nächsten acht Bit ausgegeben. Die Ausgabeschleife ist also genau einen PIO-Taktimpuls lang. Die Daten müssen daher möglichst schnell aus dem Hauptspeicher zur TX-FIFO-Einheit gelangen. Dazu werden mit einer Daten- und einer Control-DMA zwei DMA-Controller des RP2040 hintereinandergeschaltet. Dabei sorgt der Control-DMA-Controller für den schnellen Neustart des Daten-DMA-Controllers. Mit dieser Technik kann man alle vier Systemtakte (125 MHz / 4) ein neues Byte ausgeben. Für eine derartig

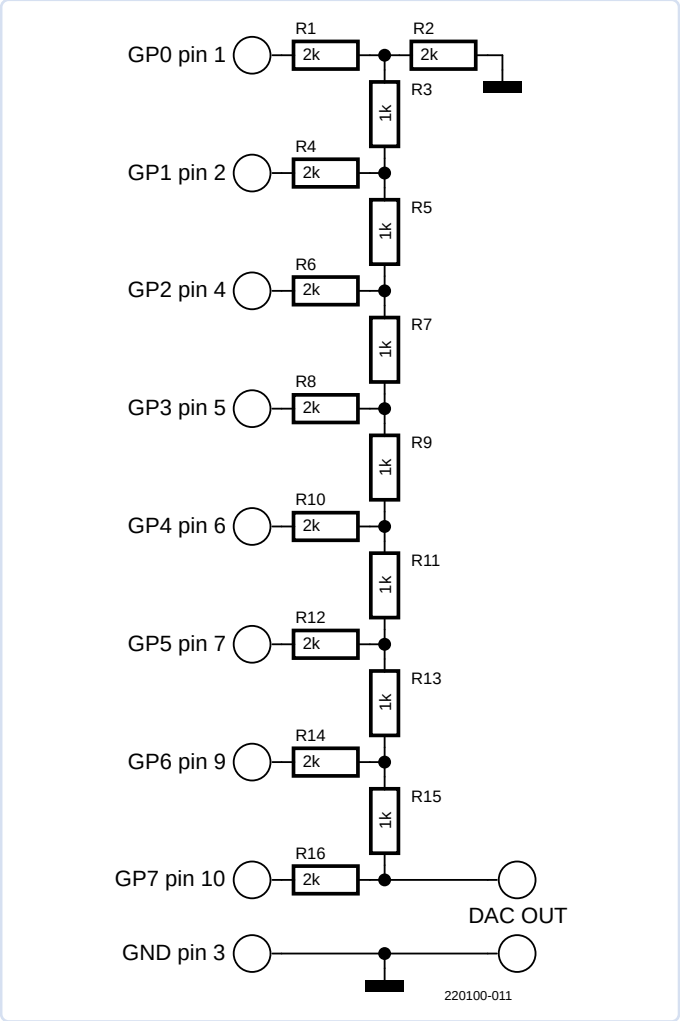


Bild 10. Schaltung des R2R-DAC.

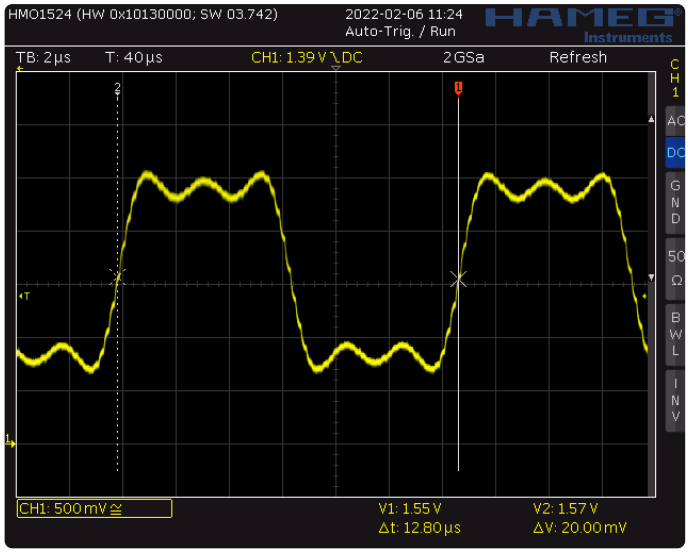


Bild 11. Grundwelle mit zwei Oberwellen eines Rechtecksignals.

hohe Ausgabegeschwindigkeit ist der einfache D/A-Wandler nach Bild 10 jedoch nicht geeignet.

Bei **Bild 11** wurde *clkDiv* auf 25 gesetzt. Der sich resultierende DAC-Takt ergibt sich somit zu 125 MHz / 25 = 5 MHz. Da eine Periode des ausgegebenen Signals 64 Samples lang ist, errechnet sich die erzeugte Frequenz zu 5 MHz / 64 = 78,125 kHz. Als Signalform wurde die Fourier-Approximation eines Rechtecks aus Grundwelle plus zwei Oberwellen genommen.

Register-Initialisierung

Mit dem `set`-Befehl kann man die Register X und Y mit festen Werten initialisieren. Da Befehle in 16 bit codiert werden, kann der set-Befehl nur Konstanten von 0 bis 31 verarbeiten. Um X oder Y mit größeren Konstanten zu initialisieren, kann man aber einen Trick anwenden. Indem man aus einem C-Programm die Routine `pio_sm_exec(pio, sm, instruction)` aufruft, unterbricht man die Befehlsfolge von PIO-Block `pio` sowie State-Machine `sm` und schiebt den Befehl `instruction` ein. Mit der folgenden Befehlsfolge kann man das Register X mit dem Wert 500.000 initialisieren.

Listing 9:

```
001 pio_sm_put_blocking(pio, sm, 500000);
002 pio_sm_exec(pio, sm,
      pio_encode_pull(false, false));
003 pio_sm_exec(pio, sm,
      pio_encode_mov(pio_x, pio_osr));
```

Zuerst bringt man die 500.000 in die TX-FIFO-Einheit, und dann führt man auf dem PIO-Controller den `pull`-Befehl aus, welcher die 500.000 von der FIFO-Einheit ins OSR bringt. Abschließend bringt der Befehl `mov X,OSR` den Wert wie gewünscht ins X-Register. Nun kann man die eigentliche PIO starten. Als Beispiel dient das Programm des nächsten Listings, welches eine LED blinken lässt.

Listing 10:

```
001 .program blink
002 .side_set 1 opt
003    mov y,x   side 1
004 yLoop1:
005    jmp y--   yLoop1
006    mov y,x   side 0
007 yLoop2:
008    jmp y--   yLoop2
```

MicroPython

Ein Raspberry Pi Pico lässt sich auch prima in MicroPython programmieren. Zur PIO-Programmierung gibt es ein entsprechendes Python-Interface. Dabei wird nicht etwa in PIO-Assembler, sondern durch entsprechende Python-Wörter programmiert, die aber stark an die Assembler-Programmierung erinnern. Die Benutzung von MicroPython hat den Vorteil, dass man komplette Projekte in einer Datei speichern kann. Das zugehörige Beispielprogramm ist im **Listing MicroPython** zu finden.

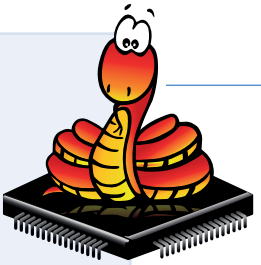
Das eigentliche PIO-Programm wird als Python-Routine `pio_prog` codiert. Sie besteht aus dem Code in der Schleife zwischen `wrap_target()` und `wrap()`. Das Programm toggelt den Side-Set-Pin 0 – hier der LED-Pin GP25. Die On/Off-Zeit wird jeweils durch eine Y-Warteschleife realisiert. Der Anfangswert des Y-Registers steht im X-Register. Der Wert im X-Register soll von außen über die TX-FIFO-Einheit gesteuert werden. Hierzu dient der Befehl `pull(noblock)`. Ist ein Wort in der TX-FIFO-Einheit, wird dieses in das OSR-Register gebracht. Die Option `noblock` bewirkt, dass bei leerer FIFO-Einheit der Inhalt des X-Registers ins OSR gebracht und der Befehlsablauf fortgesetzt wird.

Schlussbemerkungen

An diesem Punkt endet der Überblick über die PIO-Programmierung der MCU RP2040. Es wurde demonstriert, dass man mit wenigen Befehlen leistungsfähige Interfaces programmieren kann. Die PIO-Programmierung dieser MCUs und macht sie vielfältig einsetzbar, ohne dass man gleich FPGAs einsetzen müsste.

Dieser Artikel konnte nur einen groben Überblick über die PIO-Programmierung geben. Wer praktisch einsteigen will, sollte sich anhand der angegebenen Literatur [1][2][4][5][6][7] mit zahlreichen Beispielen einen genaueren Einblick verschaffen.

220100-02



Listing MicroPython

```
001 from machine import Pin
002 from rp2 import PIO, StateMachine, asm_pio
003 from time import sleep
004
005 @asm_pio( sideset_init=(PIO.OUT_LOW))
006 def pio_prog():
007      wrap_target()
008      pull(noblock)
009      out(x, 32)
010
011      mov(y, x) .side(1)
012      label("Loop1")
013      jmp(y_dec, "Loop1")
014
015      mov(y, x) .side(0)
016      label("Loop2")
017      jmp(y_dec, "Loop2")
018      wrap()
019
020 class PIOAPP:
021      def __init__(self, sm_id, pinA, periodLength, count_freq):
023         self._sm = StateMachine(sm_id, pio_prog, freq=count_freq, sideset_base=Pin(pinA)
025          self._sm.active(1) # start PIO execution
026
027      def set(self, value): self._sm.put(value) # put val into TX-FIFO
029
030 pio1 = PIOAPP(0, pinA=25, periodLength=1, count_freq=1_000_000)
032 print("ready1")
033 while True:
034      pio1.set(200000) # 200 ms On/Off time
035      sleep(1) # for one second
036      pio1.set(100000) # 100 ms On/Off time
037      sleep(1) # for one second
038      print("idle")
038      print("idle")
```

Sie haben Fragen oder Kommentare?

Gerne können Sie sich an Prof. Dr. Oßmann unter der E-Mail-Adresse ossmann@fh-aachen.de oder an die Elektor-Redaktion unter der E-Mail-Adresse redaktion@elektor.de wenden.

Passende Produkte

- **Raspberry Pi Pico RP2040 (SKU 19562)**
  www.elektor.de/19562
- **Dogan Ibrahim, Raspberry Pi Pico**
  Buch (deutsch), Paperback (SKU 19866): www.elektor.de/19866
  E-Buch (deutsch), PDF (SKU 19867): www.elektor.de/19867
- **Dogan Ibrahim, Raspberry Pi Pico W + GRATIS Pico W**
  Buch (englisch), Paperback (SKU 20335): https://elektor.de/20335
  E-Buch (englisch), PDF (SKU 20336): https://elektor.de/20336

WEBLINKS

[1] Einführung in Raspberry Pi Pico: https://datasheets.raspberrypi.com/pico/getting-started-with-pico.pdf
[2] Anleitung zum Aufsetzen der Toolchain, Shawn Hymel: https://tinyurl.com/yde5dd8a
[3] Software-Download: https://www.elektormagazine.de/220100-02
[4] Datenblatt RP2040: https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf
[5] Raspberry Pi Pico SDK: https://datasheets.raspberrypi.com/pico/raspberry-pi-pico-c-sdk.pdf
[6] SDK-Dokumentation: https://raspberrypi.github.io/pico-sdk-doxygen/index.html
[7] RP2040 MicroPython: https://docs.micropython.org/en/latest/library/rp2.html

ChipTweaker des armen Mannes

Billige Wege, sie zum Sprechen zu bringen

Von Luka Matic (Kroatien)

Hardware-Sicherheitsmodule (HSMs) sind elektronische Geräte, die verschiedene Hardware-Schutzsysteme gegen unbefugtes Lesen der internen geheimen Daten verwenden. Sie führen in der Regel Aktionen wie Verschlüsselung oder digitale Signierung durch, können aber auch andere Aufgaben übernehmen. Bankkarten oder Zugangskarten sind typische Beispiele für HSMs. Sie müssen PINs und private Schlüssel so gut wie möglich geheim halten, falls die Karte gestohlen wird oder verloren geht. Darüber hinaus verfügen viele Allzweck-MCUs über Speicherschutzfunktionen, die vor dem illegalen Kopieren von proprietärer Firmware schützen sollen. Der hier vorgestellte ChipTweaker ist ein Werkzeug, mit dem man die Grundlagen von nicht-invasiven Angriffen auf HSMs erlernen kann. Mit ihm kann man versuchen, einen speichergeschützten Mikrocontroller zu entsperren.

Wie in einem früheren Artikel [1] erläutert, können es sich unsere armen Low-Budget-Spione Alice und Bob nicht leisten, ihre eigenen HSMs zu bauen, da dies teure Spezialausrüstung und Kenntnisse erfordert, über die sie wahrscheinlich nicht verfügen. Für ihre kritischen Operationen müssen sie ihrer eigenen Hardware aus billigen Allzweckbauteilen vertrauen. Das funktioniert gut, solange sie die Strategien zum Schutz kritischer Daten (OpSec) befolgen. Andererseits kommen sie nicht umhin, HSMs wie Bankkarten zu verwenden, und stolpern vielleicht über ein HSM, dessen Geheimnisse sie ausspionieren wollen. Der ChipTweaker des armen Mannes ist daher sehr nützlich für sie, da er es ihnen ermöglicht, die Grundlagen nicht-invasiver Angriffe auf HSMs zu erlernen.

Hardware-Angriffe

Angriffe auf HSMs lassen sich in drei Haupttypen unterteilen:

1. **Nicht-invasiv**: Das HSM wird nicht geöffnet oder entkapselt. Der Angriff wird durch Analyse verschiedener elektrischer Signale an den regulären Kommunikationsanschlüssen und Stromversorgungspins des HSM durchgeführt. Auf solche (preiswerten) Angriffe ist der ChipTweaker ausgelegt.
2. **Semi-invasiv**: Die oberste Schicht des Siliziumchips des HSM wird entfernt, die elektrischen Kontakte auf dem Silizium-Die aber nicht freigelegt. Der Angriff wird durchgeführt, indem bestimmte Teile der HSM-Mikroschaltungen mit Lichtimpulsen beleuchtet werden. Ein mäßig teurer Angriff.
3. **Invasiv**: Der Siliziumchip ist vollständig freigelegt, so dass Mikrosonden an die Kontakte auf dem Die angeschlossen werden können. Der Angriff erfolgt durch Einspeisung elektrischer Signale in die HSM-Mikroschaltkreise. Dies ist ein teurer Angriff, der hochspezialisierte Ausrüstung erfordert.

Es gibt natürlich auch leistungsstarke digitale Geräte wie den ChipWhisperer [2]. Ich wollte jedoch etwas bauen, das auf älterer und anschaulicher Technologie basiert, ähnlich der Arbeit von Dr. Skorobogatov [3], bei der alle analogen und digitalen Signale vollständig zugänglich sind. Dies verspricht eine deutlich bessere Einsicht in die Prinzipien nicht-invasiver Angriffe.

Spezifikationen

Nicht-invasive Angriffe basieren auf verschiedenen Methoden, Fehler in das HSM einzuspeisen, wie etwa das Senden absichtlich schlecht formatierter Daten oder das Anlegen von Signalen mit falscher Amplitude und Frequenz an einen der HSM-Pins (einschließlich der Stromversorgungspins). Diese können das DUT (Device Under Test, oder besser gesagt, das Device Under Attack) dazu bringen, viele unkontrollierte Aktionen auszuführen und hoffentlich so seine Geheimnisse zu enthüllen.
Nicht-invasive Angriffe erfordern für jedes einzelne DUT eine Menge an Datenverarbeitung und Tweaking und sie gleichen einer

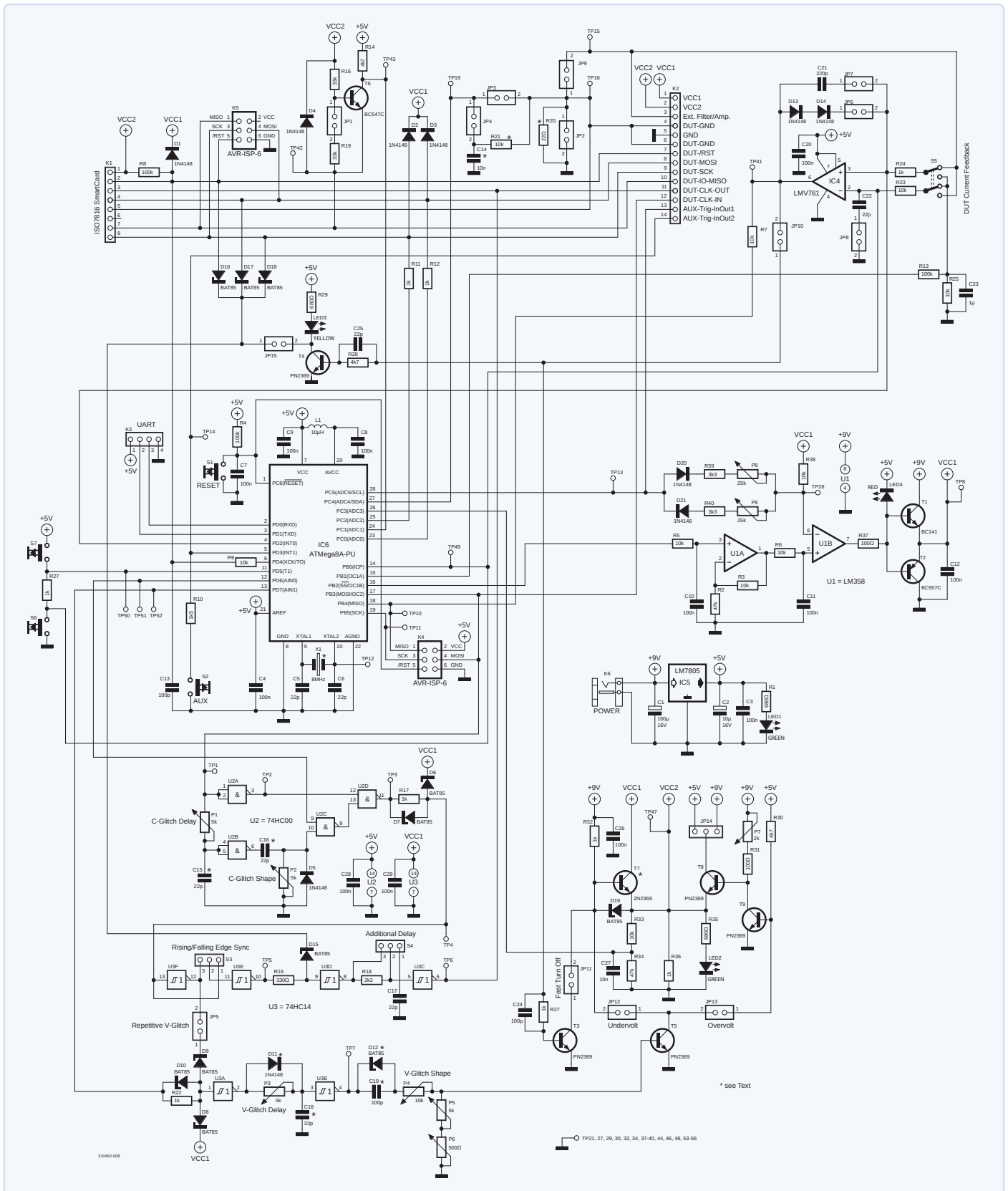


Bild 1. Der Schaltplan des ChipTweakers für Arme. Man beachte die Verwendung von schnellen Transistoren des Typs 2369. Aus Gründen der Wärmeableitung sollte T7 ein TO-18-Typ (Metallgehäuse, 2N2369) sein. Die Kunststofftypen PN2396, die sonst verwendet werden, sind möglicherweise billiger und/oder leichter zu finden. Andere Teile, die mit einem Stern gekennzeichnet sind, sollten in einer Fassung montiert werden, um ein einfaches Experimentieren mit verschiedenen Werten zu ermöglichen.

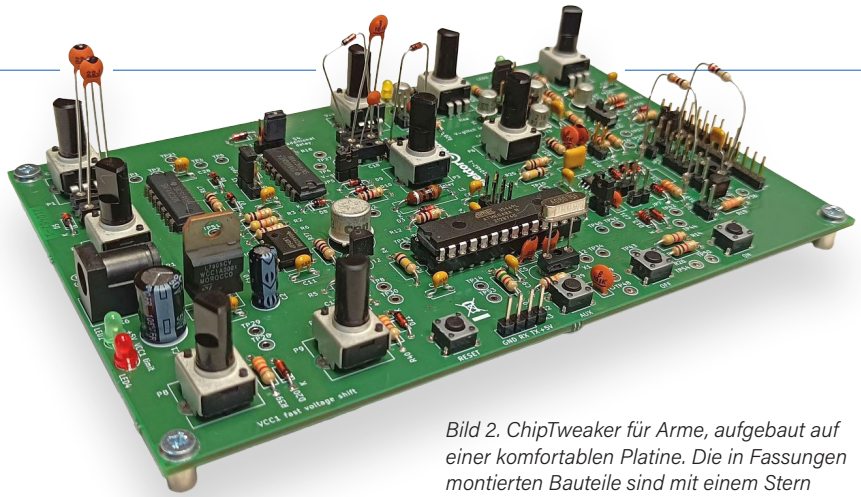


Bild 2. ChipTweaker für Arme, aufgebaut auf einer komfortablen Platine. Die in Fassungen montierten Bauteile sind mit einem Stern gekennzeichnet.

Jagd im Dunkeln, aber wenn es möglich ist, das HSM nicht-invasiv zu knacken, kann auch der billige und einfache ChipTweaker helfen. Er hat die folgenden Eigenschaften:

1. **Takt-Glitch-Generator**: Das Anlegen von zu schnellen (kurzen) Signalen an den Takteingang (CLK) des HSM kann dazu führen, dass die CPU einen Maschinenbefehl überspringt oder nicht korrekt ausführt und so zum Beispiel eine PIN-Prüfung überspringt.
2. **Spannungs-Glitch-Generator**: Eine kurzzeitige Änderung der Versorgungsspannung des HSM außerhalb des Nennbereichs kann ebenfalls einen unkontrollierten Betrieb verursachen. Der ChipTweaker kann langsame, mittlere und schnelle Spannungsspitzen erzeugen, sowohl Unter- als auch Überspannung.
3. **Variable Spannungsversorgung** des Prüflings von 1,0 V bis 6,0 V, gesteuert durch den Mikrocontroller des ChipTweakers.
4. **Schneller DUT-Abschaltkomparator**: Bei Über- oder Unterschreiten eines bestimmten Schwellwerts des Versorgungsstroms (zum Beispiel bei einem Sabotageangriff durch Beschreiben des EEPROMs), schärfen/entschärfen oder direkt von der Haupt-MCU ausgelöst.
5. **Bit-banging der SPI- und der UART-Schnittstelle mit einem bidirektionalen Pin** für die Kommunikation mit dem Prüfling (einschließlich aller Arten von SmartCards), mit bidirektionalen Spannungspegelwandlern, die den Bereich 1,5...6,0 V abdecken.

Neben den bereits erwähnten Angriffen kann der ChipTweaker zusammen mit einem digitalen Oszilloskop (etwa dem LabNation SmartScope), einem analogen 100...200-MHz-Speicheroszilloskop wie dem Tektronix 466 und einem PC mit einer Software wie MATLAB für die Offline-Datenanalyse alle Arten von Timing- und Power-Analysen durchführen.

Im Gegensatz zu den volldigitalen ChipWhisperern sind alle analogen und digitalen Signale für die Untersuchung und Analyse offen zugänglich, und das Glitch-Timing kann dank der gleitenden analogen Einstellung mit Potis ohne Quantisierungstufen noch genauer angepasst werden. Aus diesem Grund kann sogar ein langsamer ATmega8 die gesamte Attack-Prozedur koordinieren. Ein FPGA wäre da völlig überdimensioniert. Es können Glitches mit einer Dauer von weniger als 10 ns erzeugt werden, die in CPUs, die für einen Betrieb bis 50 MHz und höher ausgelegt sind, Fehler verursachen. Viele SmartCards und MCUs arbeiten ohnehin nur bis zu 20 MHz.

Beschreibung der Hardware

In der Mitte des Schaltplans (**Bild 1**) befindet sich der Mikrocontroller (IC6, ein ATmega8), der die Angriffe koordiniert. Die Schaltung um IC1 ist eine variable Spannungsquelle für den Prüfling, deren Ausgangsspannung VCC1 durch PWM aus dem Timer OC1B von IC6 gesteuert wird. Dies ermöglicht die Erzeugung von langsamen Spannungsspitzen. Der MCU-Pin PC5 kann die drei Zustände L, H und High-Z annehmen, wodurch er die Verstärkung von IC1B schnell ändern und somit drei verschiedene Werte von VCC1 erzeugen kann, die durch P8 und P9 eingestellt werden. Auf diese Weise wird ein Spannungssprung mittlerer Geschwindigkeit (im Mikrosekundenbereich) erzeugt. Die rote LED4 leuchtet zur Warnung auf und begrenzt VCC1 auf etwa 6 V, was für einen 5-V-Prüfling nicht zu hoch ist.

Erzeugung von Glitches (Spitzen)

Der Prüfling wird normalerweise von VCC2 über den Transistor T7 (ein 2N2369 in einem TO-18-Gehäuse) versorgt. T5 kann seine Basis schnell nach unten ziehen (Jumper JP12 setzen), was zu einer schnellen Unterspannungsspitze führt (in der Größenordnung von 10 ns). Wenn JP13 gesetzt ist, zieht T8 schnell VCC2 hoch, was zu einer schnellen Überspannungsspitze führt. Schnelle Spannungsglitches werden durch C19, P4, P5 und P6 beeinflusst. Die Verzögerung der Spannungsspitze in Bezug auf den CLK-Impuls wird mit P3 eingestellt. Wenn der MCU-Ausgang PD7 auf High geht, wird eine einzelne Spannungsspitze ausgelöst. Bleibt er auf High und ist JP5 kurzgeschlossen, werden die Spannungsspitzen bei jedem CLK-Impuls ausgelöst.

Mit IC2 wird eine Takt-Spitze erzeugt. Der PWM-Ausgang OC2 der MCU erzeugt den CLK-Impuls für den Prüfling. Wenn Port PD6 High ist, gibt IC2C bei jedem CLK-Puls einen kurzen Glitch an IC2D weiter. Die Polarität der Spitze kann mit einem Jumper oder Schalter an S3 gewählt werden.

IC4 ist ein schneller Komparator mit positiver Rückkopplung (verstärkt durch D13, D14 und C21), der den Prüfling innerhalb weniger Nanosekunden abschalten kann. Dies kann auch durch einen Befehl von der MCU (PD5 oder PD2 zum Einschalten, PB0 zum Ausschalten des Prüflings) oder durch Drücken der Tasten S6 und S7 erreicht werden.

Stromverbrauchsattacken

IC4 kann auch so konfiguriert werden, dass er den Prüfling ausschaltet, wenn die Stromaufnahme des Prüflings (gemessen am Widerstand R20) höher oder niedriger ist (ausgewählt durch S5) als ein Schwellwert (an C23, gesteuert durch PWM am OC1A-Pin). Dies verhindert normalerweise, dass der Prüfling in sein internes EEPROM schreibt (was zusätzlichen Strom verbraucht). Der Ausgang von IC4 aktiviert T3, wodurch T7 ausgeschaltet wird. Außerdem wird der Transistor T4 (Hilfsabschaltung) aktiviert, um alle Leitungen der SmartCard auf Low zu ziehen und eine mögliche Phantom-Rückversorgung zu verhindern.

Der Analogeingang ADC4 überwacht die Stromaufnahme des Prüflings. Da es sich um einen langsamen ADC handelt, kann das Signal mit R21/C14 gefiltert oder bei Bedarf durch einen schnelleren externen Filter/Verstärker (an Anschluss K2) zusätzlich verarbeitet werden. Für einen Timing- oder Power-Analyse-Angriff wird dieses Signal typischerweise mit einem Oszilloskop aufgezeichnet, um es offline mit einem Tool wie MATLAB zu verarbeiten und zu analysieren. Da die MCU mit 5 V betrieben wird, der

Prüfling aber mit der (in der Regel) niedrigeren Spannung VCC2, ist ein bidirektionaler Pegelwandler (um T6 herum) für den I/O-Pin 7 des Prüflings am Anschluss K1 erforderlich (ein bidirektionaler UART-Port, wie ihn die meisten SmartCards heutzutage verwenden). Pin 4 und Pin 8 werden für SmartCards mit SPI-Schnittstelle verwendet (zum Beispiel die FunCard). Da sie unidirektional sind, sind die Pegelwandler einfacher (Dioden D2 und D3).

Auf der Platine

Um den Bau dieses ChipTweakers zu erleichtern, wurde eine Platine im Europaformat entworfen (**Bild 2**). Das KiCad-Projekt und die Gerberdateien finden Sie unter [4]. Alle Bauteile sind bedrahtet, mit Ausnahme von IC4 im SOIC-8-Gehäuse. Der Aufbau der Platine sollte keine Probleme bereiten, aber es gibt einige Dinge zu beachten:

- An Stelle der 3-poligen Jumper „Sxx" sind vielleicht als SPDT-Schiebe- oder Kippschalter praktischer.
- Vielleicht möchten Sie lieber kleine Trimmpotis statt Potentiometer verwenden. Trimmpotis sind leichter zu beschaffen und bieten eine größere Auswahl an Werten.
- T7 muss aus Gründen der Wärmeableitung ein 2N2369 in einem TO-18-Metallgehäuse sein. Die TO-92-Transistoren PN2396 können durch 2N2369-Transistoren ersetzt werden, je nachdem, was einfacher zu beschaffen ist.
- Für die mit einem Sternchen bezeichneten Bauteilen (D11, C18, und so weiter, außer T7) sollten Fassungen montiert werden, um ein einfaches Experimentieren mit verschiedenen Werten und Typen zu erleichtern.

Die Platine hat viele Testpunkte, immer mit einem Masseanschluss in der Nähe. Versuchen Sie nicht, deren Nummerierung zu verstehen; es gibt da keine Logik.

Fertig zur Attacke!

Das Testobjekt für die Demonstrationsanwendungen ist eine Standard-FunCard mit einem AVR AT90S8515 von Microchip und einem AT24C-EEPROM. Ohne ernsthafte Schutzvorkehrungen ist sie relativ leicht angreifbar, so dass sie ein lohnendes Opfer für die ersten Schritte darstellt. Laden Sie die FunCard über den ISP-Anschluss K5 mit der Firmware [4]. Auf dieser Projektseite finden Sie auch die beiden ATmega8-Firmware-Varianten für den Angriff auf eine FunCard, von denen eine über den ISP-Anschluss K4 (und nicht K5) die MCU IC6 des ChipTweakers programmiert. Sie können nun einige grundlegende Aktionen ausprobieren.

Schließen Sie ein serielles Terminal an den UART-Port K3 an und konfigurieren Sie es für 9600-8-N-1 (9600n81). Nachdem Sie den Reset-Knopf S1 gedrückt haben, sollten Sie eine Terminalausgabe wie in **Bild 3** sehen. Option 0 und Option 1 dienen zum Senden einfacher Befehle an die MCU und die FunCard. Alle Befehle sind 4 Bytes lang, wobei das letzte Byte immer gleich 0x0d (CR) ist, wie auch in **Tabelle 1** und **Tabelle 2** angegeben.

```
> 0 : Send a single command to Smartcard
> 1 : Send a single command to MCU
> 2 : Find minimum Vcc2
> 3 : Adjust minimum Vt
> 5 : Funcard no glitch demo
> 6 : Funcard clock glitch demo
> 7 : Funcard volt glitch demo
> 8 : Volt glitch loop test
> 9 : Volt & Clock glitch loop test
> A : Fast Vcc2 test
> B : Brute-force PIN demo, Funcard MCU internal counter
> b : Brute-force PIN demo, Funcard AT24C external counter
> R : Reset MCU & Smartcard

Vcc2 RAW set to:0xDD    Vcc2 feedback: 5,092V
Vt RAW set to:0xFF    Vt scaled set to: 0,452V    Vi feedback: 0,166V
Smartcard fclk division factor set to RAW:0x00

> Select Option (0-X): 5

No glitch demo.

Answer To Reset: 0x45 0x67 0x78 0x9A 0xBC 0xDE 0xF0 0x12 0x34 0x56 0x78 0xDE 0xAD 0x01 0x23 0x00 0x00 0x00 0x00 0x00
```

Bild 3. Wählen Sie aus einem Menü die Attacke aus, die Sie ausführen wollen.

Befehl	Beschreibung
V2x	x ist eine 8-Bit-Zahl, die das PWM-Verhältnis für OC1B zur Einstellung der VCC2-Spannung definiert
Vtx	x ist eine 8-Bit-Zahl, die das PWM-Verhältnis für OC1A festlegt, um die Vt-Spannung einzustellen, einen Schwellenwert an C23 zur Aktivierung von IC4
Fxy	x ist ein roher Frequenzteilungsfaktor, um einen CLK-Impuls für den Prüfling an OC2 zu erzeugen. Wenn er auf 0 gesetzt wird, läuft OC2 mit der halben MCU-Quarzfrequenz; y wird ignoriert
GLx	Die untersten 3 Bits von x setzen die MCU-Pins PD7, PD6 und PC5; zum Testen
onx	Einschalten des Prüflings; x wird ignoriert
ofx	Ausschalten des Prüflings; x wird ignoriert
Sxy	Speichern der aktuellen Einstellungen im EEPROM der MCU; x und y werden ignoriert

Tabelle 1: Befehle an die MCU. Beenden Sie jeden Befehl mit 0x0d (CR).

Befehl	Beschreibung
Rxy	Lesen eines Bytes von Adresse x des internen EEPROMs des AT90S; y wird ignoriert
Wxy	Schreiben des Bytes y an die Adresse x des internen EEPROMs des AT90S
Pxy	Test einer PIN-Nummer xy (im BCD-Format). Die PIN ist im internen EEPROM des AT90S gespeichert
rxy	Lesen eines Bytes von Adresse x des AT24C EEPROMs; y wird ignoriert
wxy	Schreiben des Bytes y an die Adresse x des AT24C EEPROMs
pxy	Test einer PIN-Nummer xy (im BCD-Format). Die PIN wird im AT24C EEPROM gespeichert

Tabelle 2: Befehle an die SmartCard. Beenden Sie jeden Befehl mit 0x0d (CR).

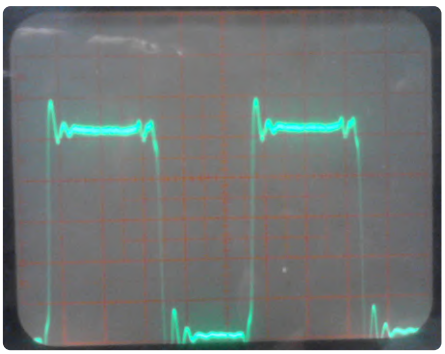


Bild 4. Ein sauberes Taktsignal ohne Glitches. Das Oszilloskop wurde auf 50 ns/div auf der horizontalen Achse und 1 V/div auf der vertikalen Achse eingestellt.

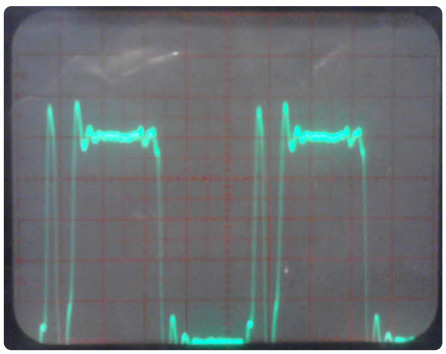


Bild 5. Ein gestörtes Taktsignal (50 ns/div horizontale, 1 V/div vertikale Achse). Stellen Sie die Potis P1 und P2 so ein, dass ein Glitch wie hier gezeigt entsteht.

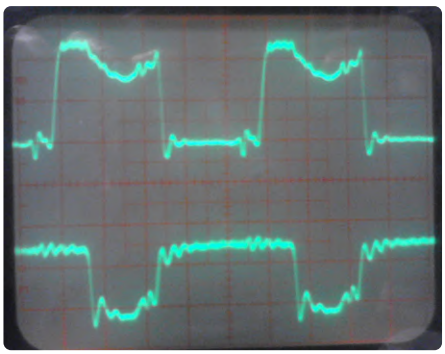


Bild 7. Die Spannungsstörung wurde 30…40 ns nach der steigenden Flanke des Taktsignals ausgelöst. Die obere Kurve ist das Taktsignal, die untere die Versorgungsspannung V_{CC2} (50 ns/Div horizontal; 1 V/Div vertikal).

Takt-Glitch-Attacke

Dies ist eine Demo für einen einfachen Takt-Glitch-Angriff: Die FunCard führt mehrere 16-Bit-Operationen an zwei Eingangsoperanden durch, die sich im internen EEPROM der MCU an den Adressen 0x00, 0x01, 0x02 und 0x03 im Big-Endian-Format befinden. Wenn man in verschiedenen Stadien der Berechnungen einen zu schnellen Takt sendet, kommt es zu unterschiedlichen Fehlern. Manchmal ist vielleicht das Ergebnis der Rechenoperation falsch, manchmal wird einfach ein Maschinen-Befehl übersprungen. Siehe **Bild 4**, **Bild 5** und **Bild 6** für die Signale und die Ergebnisse. Ein bei der FunCard wirkungsvoller Glitch ist ein aktiver Low-Puls mit einer Dauer zwischen 10 ns und 20 ns, der etwa 10 ns bis 20 ns nach der steigenden Flanke des CPU-Taktes gefeuert wird (siehe Bild 5).

```
> 0 : Send a single command to Smartcard
> 1 : Send a single command to MCU
> 2 : Find minimum Vcc2
> 3 : Adjust minimum Vt
> 5 : Funcard no glitch demo
> 6 : Funcard clock glitch demo
> 7 : Funcard volt glitch demo
> 8 : Volt glitch loop test
> 9 : Volt & Clock glitch loop test
> A : Fast Vcc2 test
> B : Brute-force PIN demo, Funcard MCU internal counter
> b : Brute-force PIN demo, Funcard AT24C external counter
> R : Reset MCU & Smartcard

Vcc2 RAW set to:0xDD    Vcc2 feedback: 5,092V
Vt RAW set to:0xFF    Vt scaled set to: 0,452V    Vi feedback: 0,166V
Smartcard fclk division factor set to RAW:0x00

> Select Option (0-X): 6

Clock glitch demo.

Answer To Reset: 0x45 0x67 0x78 0x9A 0xBC 0xDE 0xF0 0x12 0x34 0x56 0x78 0xDE 0xAD 0x01 0x23 0x00 0x00 0x00 0x00 0x00
0xF62F 0x01 0x25 0x25  OK!
0xF34B 0x02 0x3E 0x3E  OK!
0xF62F 0x03 0x25 0x25  OK!
0xF62F 0x04 0x25 0x25  OK!
0xF5E4 0x05 0xD9 0xD9  OK!
0xF62F 0x06 0x25 0x25  OK!
0xF62F 0x07 0x25 0x25  OK!
0xF576 0x08 0x6B 0x6B  OK!
0x48E8 0x09 0x30 0x30  OK!
0x47E8 0x0A 0x2F 0x2F  OK!
0xF676 0x0B 0x6C 0x6C  OK!
0x47E8 0x0C 0x2F 0x2F  OK!
0xF62F 0x0D 0x25 0x25  OK!
0x0000 0x0E 0x00 0x00  OK!
0x5555 0x0F 0x55 0xAA  Error!
0xF62F 0x10 0x25 0x25  OK!
0xF62F 0x11 0x25 0x25  OK!
0xF62F 0x12 0x25 0x25  OK!
0xF62F 0x13 0x25 0x25  OK!
```

*Bild 6. Terminalfenster mit den Ergebnissen der Clock-Glitch-Attacke. **Spalte 1** ist das 16-Bit-Ergebnis der mathematischen Operation, wie sie vom Prüfling ausgeführt wird. Sie wird falsch sein, wenn der Glitch erfolgreich war. **Spalte 2:** Glitch-Verzögerung. Glitching zu verschiedenen Zeitpunkten ergibt unterschiedliche Ergebnisse. **Spalte 3:** Prüfsumme als Summe der beiden Bytes des 16-Bit-Ergebnisses, wie von der FunCard berechnet. **Spalte 4:** von der MCU des PMCT berechnete Prüfsumme. **Spalte 5: Error!**, wenn die Spalten 3 und 4 nicht übereinstimmen.*

Spannungs-Glitch-Angriff

Dies ist eine Demo für einen einfachen Spannungs-Glitch-Angriff. Die FunCard führt dabei dieselben Operationen wie im vorherigen Beispiel durch. Ein effektiver Unterspannungs-Glitch mit einer Dauer von mindestens 50 ns liefert gute Ergebnisse mit dem in **Bild 7** gezeigten Timing. Hier wurde der Glitch 30…40 ns nach der steigenden Flanke des Taktsignals ausgelöst. Die Versorgungsspannung für die FunCard wurde auf 2,24 V eingestellt. Der Spannungs-Glitch fiel auf etwa 1,5 V ab.

Stellen Sie die Form des Spannungsabfalls mit P4, P5 und P6 ein, die Glitch-Verzögerung und die Synchronisation mit P3, S3 und S4. Die Ergebnisse werden auf die gleiche Weise wie im vorherigen Beispiel angezeigt (**Bild 8**).

Cracken des Speicherschutzes zum Extrahieren der Firmware

Jeder Typ von geschützter MCU oder Smart-Card erfordert ein anderes Angriffsverfahren. Es erfordert eine Menge „Trial and Error" und Fischerei im Trüben, um mögliche Schwachstellen zu finden. Ein Verfahren, das bei vielen Microchip-AVRs funktioniert, ist ein langsamer Unterspannungs-Glitch.

Es funktioniert folgendermaßen: Wenn die FunCard-MCU AT90S8515 geschützt ist, werden beide Speicher-Lock-Bits auf den Wert 0 programmiert. Der Flash-Speicher kann dann nicht mehr gelesen, sondern nur noch gelöscht werden. Im seriellen Programmiermodus (der Reset-Eingang ist auf 0 V gezogen) löscht der Chip-Erase-Befehl zuerst den Flash-Speicher (alle Flash-Bytes werden auf 0xFF zurückgesetzt) und dann

die Lock-Bits (sie werden deaktiviert, indem sie auf den Wert 1 zurückgesetzt werden).
Ein Konstruktionsfehler bei vielen AVR-MCUs (nicht bei allen) ermöglicht es uns, den Schutz auf folgende Weise zu knacken: Wenn die Versorgungsspannung unter das nominale Minimum von 2,7 V auf 1,6...1,7 V gesenkt wird, steht gerade noch ausreichend Strom zur Verfügung, um die Lock-Bits zu löschen, aber nicht mehr, um auch den Flash-Speicher zu löschen. Der Angriff wird bei 1,1 V gestartet und die Spannung wird schrittweise erhöht. Die Lock-Bits wurden erfolgreich bei 1,62 V entfernt, ohne den Flash-Speicher zu löschen! Wenn man versucht, die Firmware oder die Gerätesignatur eines geschützten AVR-Mikrocontrollers zu lesen, lautet die Antwort „0x00, 0x01, 0x02, 0x03“, und man weiß, dass der Speicher geschützt ist (**Bild 9**). Sobald man aber die korrekte Signatur erhält („0x1E, 0x93, 0x01“ für den AT90S8515), wurden die Speicher-Sperrbits entfernt und der Programmspeicher kann nun mit jedem ISP-Programmiergerät ausgelesen werden.

```
> 0 : Send a single command to Smartcard
> 1 : Send a single command to MCU
> 2 : Find minimum Vcc2
> 3 : Adjust minimum Vt
> 5 : Funcard no glitch demo
> 6 : Funcard clock glitch demo
> 7 : Funcard volt glitch demo
> 8 : Volt glitch loop test
> 9 : Volt & Clock glitch loop test
> A : Fast Vcc2 test
> B : Brute-force PIN demo, Funcard MCU internal counter
> b : Brute-force PIN demo, Funcard AT24C external counter
> R : Reset MCU & Smartcard

Vcc2 RAW set to:0x60    Vcc2 feedback: 2,238V
Vt RAW set to:0xFF    Vt scaled set to: 0,452V    Vi feedback: 0,053V
Smartcard fclk division factor set to RAW:0x00

> Select Option (0-X): 7

Volt glitch demo.

Answer To Reset: 0x45 0x67 0x78 0x9A 0xBC 0xDE 0xF0 0x12 0x34 0x56 0x78 0xDE 0xAD 0x01 0x23 0x00 0x00 0x00 0x00 0x00
0xF62F 0x01 0x25 0x25  OK!
0xF62F 0x02 0x25 0x25  OK!
0xF5E4 0x03 0xD9 0xD9  OK!
0xF62F 0x04 0x25 0x25  OK!
0x3F4B 0x05 0x8A 0x8A  OK!
0xF62F 0x06 0x25 0x25  OK!
0xF62F 0x07 0x25 0x25  OK!
0x482F 0x08 0x77 0x77  OK!
0x492F 0x09 0x78 0x78  OK!
0x482F 0x0A 0x77 0x77  OK!
0xF676 0x0B 0x6C 0x6C  OK!
0x482F 0x0C 0x77 0x77  OK!
0xF62F 0x0D 0x25 0x25  OK!
0x0000 0x0E 0x00 0x00  OK!
0xF642 0x0F 0x38 0x38  OK!
0x0000 0x10 0x00 0x00  OK!
0xF62F 0x11 0x2F 0x25  Error!
0xF62F 0x12 0x25 0x25  OK!
0xF62F 0x13 0x25 0x25  OK!
```

Bild 8. Terminalfenster mit den Ergebnissen des Spannungs-Glitch-Angriffs. Die Spalten sind in der gleichen Weise formatiert wie bei der Clock-Glitch-Attacke (siehe Bild 6).

Power-Analyse-Angriff auf eine Bankkarte

Bankkarten werden immer raffinierter und verwenden mehrere Methoden, um kritische Speicherbereiche physisch zu schützen. Außerdem versuchen sie, den Verlauf der Stromaufnahme zu verschleiern, so dass kritische Vorgänge nicht ohne weiteres erkannt werden können. Wenn bestimmte Vorgänge wie die Überprüfung der eingegebenen PIN zeitlich genau eingegrenzt werden können, dann wissen wir aber auch, wann Glitches eingespeist werden können, um die Schutzmaßnahmen zu umgehen. Einige der bekannten Methoden zum Schutz von Bankkarten sind:

1. Verwendung eines internen RC-Taktgebers für sicherheitskritische Operationen und Umschalten auf einen externen CLK nur für ein präzises Timing, etwa bei der Kommunikation über den UART. Dies verhindert Clock-Glitch-Angriffe.
2. Verwendung sehr schneller Ladungspumpen an kleinen internen Stromversorgungskondensatoren im Pikofarad-Bereich, um die Versorgungsspannung stabil aufrecht zu erhalten. Dies verhindert Spannungs-Glitch-Angriffe.
3. Änderung der Halbperioden des internen RC-Taktes nach dem Zufallsprinzip, basierend auf einem internen echten Zufallszahlengenerator. Auf diese Weise finden die

CLEAR | AutoScroll | Reset Cnt | 13 | Cnt = 1721 | HEX / ASCII | StartLog | StopLog | Req/

```
Stage:0x01 Vcc2 RAW:0x43 Init:0x53
    Reading firmware:0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09
Reading signature... 0x00 0x01 0x02 0x03
Y-erase the lockbits, N-read again, F-finish?
    Vcc2 feedback: 1,575V

Stage:0x01 Vcc2 RAW:0x44 Init:0x53
    Reading firmwar03 03 x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09
Reading signature... 0x00 0x01 0x02 0x
Y-erase the lockbits, N-read again, F-finish?
    Vcc2 feedback: 1,598V

Stage:0x01 Vcc2 RAW:0x45 Init:0x53
    Reading firmware:0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09
Reading signature... 0x00 0x01 0x02 0x03
Y-erase the lockbits, N-read again, F-finish?
    Vcc2 feedback: 1,622V

Stage:0x01 Vcc2 RAW:0x46 Init:0x53
    Reading firmware:0x0C 0x1C 0x1A 0x19 0x18 0x17 0x22 0x2C 0x14 0x13
Reading signature... 0x1E 0x93 0x01 0xFF
Y-erase the lockbits, N-read again, F-finish?

Stage:0x01 Vcc2 RAW:0x47 Init:0x53
    Reading firmware:0x0C 0x1C 0x1A 0x19 0x18 0x17 0x22 0x2C 0x14 0x13
Reading signature... 0x1E 0x93 0x01 0xFF
Y-erase the lockbits, N-read again, F-finish?
```

Transmit

Bild 9. Knacken des Flash-Speicherschutzes.

Bild 10. Versorgungsstrom einer Bankkarte bei der Überprüfung einer falschen PIN.

kritischen Operationen (wie die Überprüfung der PIN) nicht immer zur gleichen Zeit statt, was Angriffe erschwert.

4. Hinzufügen von Zufallsrauschen auf der Versorgungsspannungsleitung, um Stromanalyse-Angriffe zu erschweren (**Bild 10**).
5. Bei der Überprüfung einer PIN wird zuerst der PIN-Eingabeversuchszähler im EEPROM der SmartCard verringert, dann die PIN überprüft und der PIN-Versuchszähler wieder erhöht, wenn die PIN in Ordnung ist. Bei alten Karten wurde der Wert nur in das EEPROM geschrieben, um den Zähler bei einer falschen PIN-Eingabe zu verringern. Dies ermöglichte aber Brute-Force-Angriffe (Durchlaufen aller PINs), um die PIN zu ermitteln, indem die Stromzufuhr nach jedem falschen Versuch schnell unterbrochen wurde.
6. Verwendung sehr sorgfältig gestalteter Funktionen für sicherheitskritische Operationen (hier ist Assembler-Programmierung erforderlich!), die immer die gleiche Anzahl von CPU-Taktzyklen und (möglicherweise) die gleiche Menge an Strom verbrauchen, unabhängig von ihren Eingangsvariablen. Dies verhindert Angriffe durch Zeit- und Leistungsanalyse.
7. Verlängerung einer sicherheitskritischen Operation, die normalerweise weniger als 1 ms dauert, auf beispielsweise 200 ms. Ein relevantes Signal mit einer Länge von 1 ms ist dann in 199 ms Datenmüll versteckt.

Um eine moderne Bankkarte nicht-invasiv auszuschalten, ist (wenn überhaupt möglich) eine umfangreiche Analyse der aufgezeichneten Daten und eine lange und mühsame Arbeit erforderlich. Lesen Sie das (nicht ganz billige) Fachbuch *Power Analysis Attacks* [5] für weitere umfassende und detaillierte Informationen über Power-Analyse-Angriffe.

Testen und Lernen

Der ChipTweaker ist keine High-Tech-Anlage, aber er kann zum Testen und Erlernen nicht-invasiver Angriffe auf HSMs verwendet werden. Ich hoffe, Sie fanden diesen Artikel interessant, zumindest als guten Ausgangspunkt. Das Entwerfen und Angreifen von Ein-Chip-HSMs ist heutzutage für alle Beteiligten eine sehr harte Arbeit. Genau aus diesem Grund bleibt dieser Bereich für weitere Forschung offen.
Viel Spaß beim Spionieren!

(210462-02)RG

Haben Sie Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Senden Sie eine E-Mail an den Autor unter luka.matic@fer.hr oder schreiben Sie der Elektor-Redaktion unter redaktion@elektor.de.

Passende Produkte

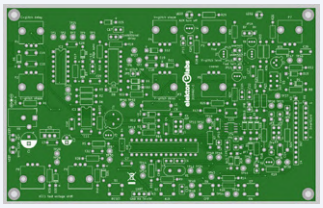
- **LabNation SmartScope USB-Oszilloskop SKU 17169:** www.elektor.de/17169
- **Luka Matic, Electronic Security and Espionage, Elektor 2021** Paperback, SKU 19903: www.elektor.de/19903 E-Buch, SKU 19904: www.elektor.de/19904

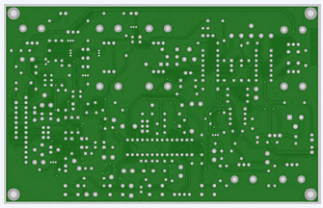
Funcard. (Quelle: www.cellularcenter.it)

WEBLINKS

[1] Luka Matic, „Manipulationssicheres Datenpaket", Elektor 5/ 2020: https://www.elektormagazine.de/magazine/elektor-147/58581
[2] ChipWhisperer von NewAE Technology Inc.: https://www.newae.com/chipwhisperer
[3] Sergei P. Skorobogatov, „Copy Protection in Modern Microcontrollers": https://www.cl.cam.ac.uk/~sps32/mcu_lock.html
[4] Downloads zu diesem Artikel bei Elektor Labs: https://www.elektormagazine.de/labs/poor-mans-chipwhisperer-or-a-smartcard-tweaker
[5] S. Mangard, E. Oswald, T. Popp, „Power Analysis Attacks: Revealing the Secrets of Smart Cards", Springer, 2007: https://amzn.to/3RWBtuy

STÜCKLISTE





Widerstände:
R1,R29,R35 = 680 Ω
R2,R34 = 47 k
R3,R5,R6,R7,R9,R23,R25,R33,R38 = 10 k
R4,R8,R13 = 100 k
R10 = 1k5
R11,R12,R17,R22,R24,R26,R27,R32,R36 = 1 k
R14,R28,R30 = 4k7
R15 = 330 Ω
R16,R19 = 33 k
R18 = 2k2
R20* = 22 Ω*
R21* = 10 k*
R31,R37 = 100 Ω
R39,R40 = 3k3
P1,P2,P3,P5 = Poti, 5 k, vertikal
P4 = Potentiometer, 10 k, vertikal
P6 = Potentiometer, 500 Ω, vertikal
P7 = Potentiometer, 2 k, vertikal
P8,P9 = Potentiometer, 25 k, vertikal

Kondensatoren:
C1 = 100 µ, 16V, RM3,5
C2 = 10 µ, 16V, RM2,5
C3,C4,C7,C8,C9,C10,C11,C12,C20,C26,C28,C29 = 100 n, RM5
C5,C6,C17,C22,C25 = 22 p, RM2,5
C13,C24 = 100 p, RM2,5
C14* = 10 n, RM5*
C27 = 10 n, RM5
C15*,C16* = 22 p, RM5*
C18* = 33 p, RM5*
C19* = 100 p, RM5*
C21 = 220 p, RM2,5
C23 = 1 µ, RM5

Induktivität:
L1 = 10 µ

Halbleiter:
D1,D2,D3,D4,D5,D13,D14,D20,D21 = 1N4148
D11* = 1N4148*
D6,D7,D8,D9,D10,D15,D16,D17,D18,D19 = BAT85
D12* = BAT85*
IC1 = LM358
IC2 = 74HC00
IC3 = 74HC14
IC4 = LMV761, SOIC8
IC5 = 7805, TO220
IC6* = ATmega8A-PU*
LED1,LED2 = LED, 3 mm, grün
LED3 = LED, 3 mm, gelb
LED4 = LED, 3 mm, rot
T1 = BC141, TO39
T2 = BC557C
T3,T4,T5,T8,T9 = PN2369, TO92
T6 = BC547C
T7 = 2N2369, BIS18

Außerdem:
JP1...JP13,JP15 = 1x2-polige Stiftleiste, Raster 2,54 mm
JP14,S3,S4 = 1x3-polige Stiftleiste, Raster 2,54 mm
K1 = 1x8-polige Stiftleiste, Raster 2,54 mm
K2 = 1x14-polige Stiftleiste, Raster 2,54 mm
K3 = 1x4-polige Stiftleiste, Raster 2,54 mm
K4,K5 = 2x3-polige Stiftleiste, Raster 2,54 mm
K6 = Niederspannungsbuchse
S5 = Schiebeschalter (C&K JS202011CQN)
X1* = Quarz 8 MHz*

* mit Fassung für einfaches Experimentieren

Echter **Zufallszahlen-Generator** mit USB

Zwei PICs zum Preis von einem AVR

Von Matthias Wolf (Germany)

Ein guter True Random Number Generator (TRNG) ist eine nützliche Sache. Sichere Datenverschlüsselung zum Beispiel ist ohne ihn fast unmöglich. Auch Spiel- und Glücksspielanwendungen benötigen TRNGs der Spitzenklasse. Elektor hat 2017 einen analogen TRNG veröffentlicht, in diesem Update ergänzen wir ihn um einen USB-Anschluss.

Der preiswerte Zufallszahlengenerator (True Random Number Generator, TRNG) von Luka Matic [1] nutzte eine SD-Karte, um die erzeugte Zufallsfolge zu speichern. Jetzt stellen wir eine Adaption dieses Artikels vor, bei der die Speicherkartenbuchse durch eine USB-Schnittstelle ersetzt wird.

Die neue Schaltung (**Bild 1**) enthält anstelle eines einzelnen ATtiny2313A zwei PIC-Mikrocontroller von Microchip Technology, einen für die analoge Signalverarbeitung (PIC16F19156) und einen für die USB-Schnittstelle (PIC18F25K50). Beide Controller kommunizieren miteinander, von Hochgeschwindigkeits-Optokopplern galvanisch getrennt, über einen SPI-Bus, um zu verhindern, dass digitale Störungen von der USB-Schaltung das Analogsignal beeinträchtigen.

Filter-Kalibrierung

Mit einer kleinen PC-Anwendung werden die analogen Filter in Echtzeit auf einfache Weise kalibriert (**Bild 2**).
Meine Filtereinstellungen für die analogen Filter sind wie folgt:

- S5: C37,C38,C39,C40,C41 EIN mit C39 bei 109 pF
- S6: C45,C46,C47 EIN mit C47 bei 25 pF
- S7: Switch1 EIN und P6 auf 409 Ω

Diese Einstellungen hängen von den Bauteilen ab, die für den TRNG verwendet wurden, Sie benötigen möglicherweise andere. Der Typ der verwendeten Z-Dioden ist ebenfalls wichtig. Ich habe zunächst die BZX384C12-E3-08 von Vishay verwendet, aber ihr Rauschen war zu hoch. Schließlich verwendet der PIC16 im Vergleich zum Original-TRNG von Luka eine etwas höhere Abtastrate (800 kHz, 1,25 µs pro Sample).

Holen Sie sich die Dateien!

Alle Projektdateien können von [2] heruntergeladen werden. Der Download umfasst den Schaltplan (drei Seiten), das Target 3001 CAD-Projekt, die Firmware für den PIC16 (analoges Sampling, MPLAB X-IDE mit PCM C-Compiler von CCS), die Firmware für den PIC18 (USB-Handling, MPLAB X-IDE mit XC8) und die Desktop-Anwendung (C#, Visual Studio 2017 commmunity edition).

(190235-02)RG

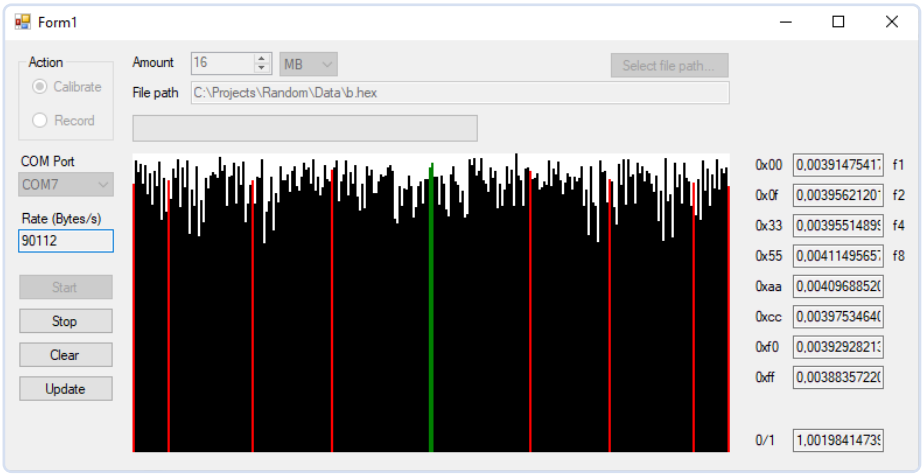


Bild 2. Screenshot der PC-Desktop-Anwendung zur Kalibrierung der analogen Filter und Aufzeichnung von Zufallsdaten.

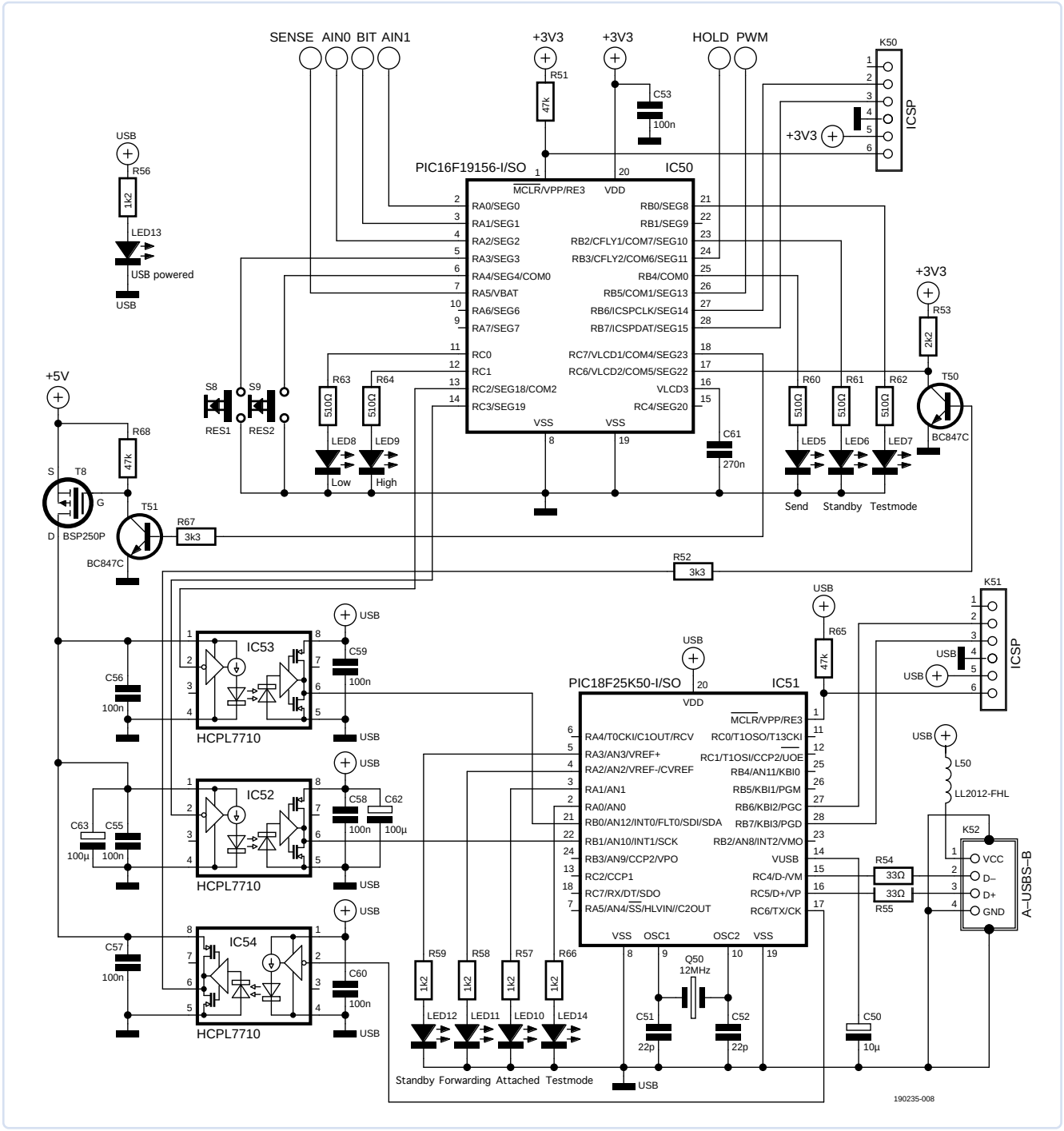


Bild 1. Der digitale Teil des ursprünglichen TRNG von 2017 mit seiner ATtiny2313A-MCU wurde durch zwei PIC-Mikrocontroller ersetzt. Der USB-Anschluss ersetzt den SD-Kartenslot.

Haben Sie Fragen oder Kommentare?

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

Passende Produkte

- **L. Matic, Electronic Security and Espionage (Elektor 2021)**
  Buch, SKU 19903: www.elektor.de/electronic-security-and-espionage
  E-Buch, SKU 19904: www.elektor.com/electronic-security-and-espionage-e-book

WEBLINKS

[1] L. Matic, „Zufallszahlen-Generator“, Elektor 3/2017: https://www.elektormagazine.de/magazine/elektor-201703/40169
[2] Dieses Projekt bei Elektor Labs: https://www.elektormagazine.de/labs/usb-random-number-generator

Pimp **my mike...**

Pegelanhebung selbst gebaut

Von **Dr. Thomas Scherer**

Die letzten Jahre werden nicht nur „wegen Corona“ unvergessen bleiben, sondern auch weil plötzlich sehr viele Menschen Erfahrungen mit Online-Konferenzen machen mussten – und nicht nur die, welche sich plötzlich im Home-Office wiederfanden. Dass die Bilder häufig rauschen – geschenkt. Doch wie steht es um die Tonqualität?

Nachdem *Skype* kein Fremdwort mehr ist und man auch mit den Begriffen *Zoom* und *Teams* jonglieren kann, haben viele Zeitgenossen in den letzten Jahren nicht nur befriedigende Erfahrungen mit Video-Telefonie und Online-Konferenzen gemacht. Neben Fehlbedienungen mancher Teilnehmer nerven vor allem die begrenzte Bandbreite der Internetverbindungen, die schlechte Qualität der Video-Bilder (aufgrund mieser Kameras und/oder schlechtem Licht) sowie last not least der schlechte Klang der Tonübertragung. Das gilt ganz besonders für die, welche mit einem Laptop unterwegs sind. Anders als bei einem PC liegt es nämlich bei schlechter Laptop-Hardware nicht so nahe, sich eine bessere Webcam zuzulegen.

Laptop-Frust

Aufgrund meiner über die ganze Welt verstreuten Freunde, Bekannten und Verwandten habe ich schon lange recht viel mit „Video-Telefonie“ zu tun gehabt. Da ich „für unterwegs“ auch tragbare Computer nutze, habe ich schon viel geflucht über die Qualität der damit möglichen Videoübertragung.

Bild 1. Der Lieferumfang umfasst das eigentliche Mikrofon, den Windschutz, eine Klemmhalterung samt Tripod und das USB-Kabel.

Den Vogel schoss dabei ausgerechnet ein etwas teureres Produkt des Herstellers mit dem angebissenen Apfel ab: Nachdem ich damals mein älteres MacBook Pro gegen ein neues MacBook Air ausgetauscht habe, konnte ich kaum glauben, welch unsäglich krisseliges und unterbelichtetes Bild sich Apple da für viel Geld zu verkaufen traute. Zwar hätte ich ja einfach eine extra Webcam kaufen und per USB anstecken können, doch genau dafür kauft man sich ja was Tragbares, damit man jede Menge extra Teile mit sich rumschleppen muss, nicht wahr? Hinzu kam, dass das MB Air nur über USB-C-Schnittstellen verfügt. Sehr fortschrittlich, aber Webcams mit USB-C sind Mangelware und die Notwendigkeit zum Mitschleppen eines zusätzlichen USB-C/USB-A-Konverters hat dann meine Leidensfähigkeit überschritten. So kam es, dass der neue Laptop verkauft wurde und ich reuig zum älteren Vorgängermodell zurückgekehrt bin.

Das Problem minderwertiger Kameras (und Mikrofone) in Klapprechnern ist übrigens nicht auf Apple beschränkt. Auch Windows-10-Nutzer kann es treffen. Ich kenne mindestens ein (teures) Samsung-Notebook, bei dem das Bildrauschen unübersehbar und der Klang fürchterlich ist. Es gibt aber durchaus portable Rechner mit mittelprächtiger Videoqualität. Schade, dass genau diese Aspekte in den Tests einschlägiger Zeitschriften zu kurz kommen. Zumindest seit Corona sollte die Relevanz deutlich geworden sein. Es ist übrigens auch nicht wirklich nachvollziehbar, warum jedes Billighandy eine deutlich bessere Front-Cam und ein halbwegs brauchbares Mikrofon verbaut hat – und nur Laptops nicht!

Abhilfe

Die einfachste Lösung zur Lösung aller bild- und tontechnischen Probleme ist der Kauf einer Webcam. Bei einem PC ist das eh obligatorisch, aber auch bei einem fürs Home-Office zur Verfügung gestellten Firmen-Laptop ist eine solche Anschaffung überlegenswert. Sicherlich wird man mit einer extrem preiswerten No-name-Webcam aus

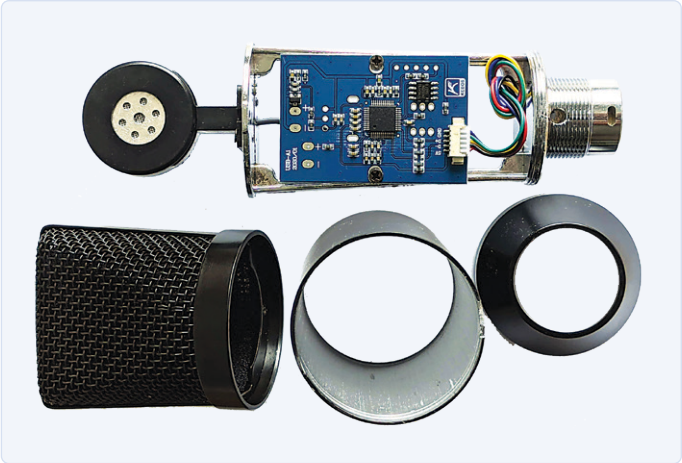


Bild 2. Das neue Mikrofon in seine Einzelteile zerlegt.

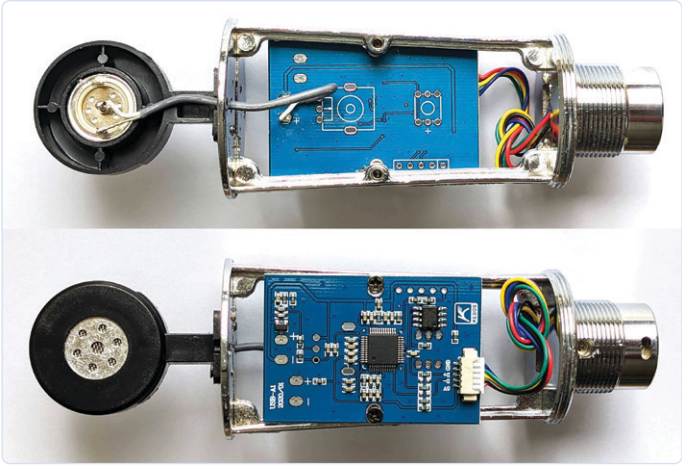


Bild 3. Das Innere des Mikrofons von vorn und von hinten.

China nur mit viel Glück zufrieden sein. Aber sowohl von Logitech als auch von Microsoft und anderen Markenherstellern gibt es brauchbare Webcams ab etwa 35 €. Die darin verbauten Mikrofone sind zwar nicht perfekt, aber durchaus gut genug und für Sprachübertragung meistens ausreichend.
Wem die Qualität der Kamera ausreicht, aber die des Tons nicht, der kann auch über den Erwerb eines externen Mikrofons nachdenken. Die Industrie hat diesen Markt erkannt und bietet etliche Modelle mit USB- statt Klinkenstecker an. Dank integriertem A/D-Wandler und der Integration der nötigen Treiber in alle gängigen Betriebssysteme ist der Anschluss an einen PC unter Windows, MacOS oder Linux völlig problemlos möglich. Natürlich könnte man auch „normale" Mikrofone mit analogem Ausgang verwenden – zumindest bei den meisten PCs. Bei modernen Laptops wird aber zunehmend der Mikrofoneingang eingespart. Als Ersatz täte es dann ein preiswerter USB-Konverter, der über einen solchen Eingang und meistens auch über einen Audio-Ausgang verfügt. Bloß hat man damit ein weiteres kleines Teil zum Verlieren und Suchen. Nichts für mich.

USB-Mikrofon

Nachdem ich die letzten Jahre mit verschiedenen preiswerten Mikrofonen nie wirklich zufrieden war, habe ich kürzlich ein sogenanntes „Großmembran-Mikrofon" mit USB-Stecker geordert. Diese Kondensator-Mikrofone werden ja auch in Tonstudios eingesetzt und sollten daher schon etwas taugen. Wikipedia widmet sich den Eigenschaften von Mikrofonen unter dem Stichwort „Mikrofonierung" [1].
Außerdem: Durch das enorme Wachstum der sozialen Medien produzieren viele (junge) Leute Podcasts oder betreiben aktiv einen eigenen Videokanal. Hierfür braucht es dann nicht nur eine gute Videokamera, sondern auch ein gutes Mikrofon. Genau für solche Zwecke werden heute „Podcast-Sets" bestehend aus Großmembran-Mikro und Standfuß, manchmal sogar mit Spinne, Galgenhalterung und Popschutz angeboten.
Also wollte ich mein Glück in diesem Sektor versuchen. Bei Audio-Technik zeigen sich zwei Phänomene besonders: Zum einen ist nicht jedes vertretene Argument zwingend rational und zum anderen wird Qualität schnell richtig teuer. Das Preis/Leistungsverhältnis ist leider keine sanft ansteigende Gerade, sondern hat einen eher exponentiellen Verlauf. Und da mir ein besserer Klang nur begrenzt viel Geld wert war, googelte ich nach preiswerten Großmembran-USB-Mikrofonen. Und wurde fündig.

CAD Audio U29

Dieses Mikrofon kommt mit einem Tripod und einem Windschutz und kostet mit 34 € erstaunlich wenig. Damit konnte ich nicht viel falsch machen, dachte ich. Also habe ich es bestellt und zwei Tage später bot sich mir nach dem Auspacken der Anblick von **Bild 1**. Optisch macht das Mikrofon einen wertigen Eindruck: Kein Plastik - alles Metall.
Ich habe es gleich an meinen PC angeschlossen, die Freeware Audacity gestartet und damit eine Aufzeichnung sowie zum Vergleich auch mit dem Mikrofon meiner Webcam (Logitech C525) gemacht. Ich war nicht enttäuscht. Ich konnte den klar besseren Klang hören und zwar so deutlich, dass sich eine Messung erübrigte. Nicht schlecht!

Gespannt war ich, ob auch meine Gesprächspartner bei Skype einen Unterschied bemerken würden. Und auch da wurde der Unterschied bemerkt. Zwei „Versuchspersonen" konnten recht klar hören, welche der beiden Mikrofone aktiv war, wenn ich umschaltete. Der Unterschied? Das Mikrofon von CAD war nicht ganz so basslastig und hatte deutlich klarere Höhen.

Pegel und Folgen

An sich hätte die Geschichte hier zu Ende sein können. Bei den Aufnahmen bemerkte ich aber, dass der Pegel des Mikrofons von CAD gegenüber der Webcam etwas mehr als 5 dB geringer, das Mikro also leiser war. Zwar bieten die Programme für Video-Conferencing eine Pegelanhebung, aber dann hätte der Regler „voll aufgedreht" sein müssen.
Niedrigen Pegeln bei Mikrofonen bin ich nicht zum ersten Mal begegnet. Doch als Elektroniker weiß man sich da zu helfen, weswegen ich auch schon Verstärker in Mikrofonen „nachgerüstet" oder durch geänderte Bauteile die Verstärkung vorhandener Elektronik erhöht habe. Das sollte auch bei diesem Mikrofon möglich sein, oder?

Kaum gedacht, war das Mikrofon auch schon zerlegt, wie **Bild 2** beweist. Oben sieht man die eigentliche Mikrofon-Kapsel, deren Membran mit etwa 1/2" Durchmesser für mein Verständnis des Begriffs „Großmembran" etwas klein ausfällt. **Bild 3** zeigt die zu klein geratene Kapsel auch von hinten. Sei's drum.

Interessanter ist die Platine: Der Chip in der Mitte ist ein SoC mit Audio-Eingang und USB-Schnittstelle. Im Gegensatz zum seriellen EEPROM rechts unten ist das SoC wohl aus Geheimniskrämerei nicht beschriftet. Schade! Aber das kleine schwarze SMD-Dreibein oben ist ja der eigentliche Vorverstärker.
Eine erste Messung an den beiden Pads „-" und „+" oben in der Mitte zeigte, dass an der Mikrofonmembran exakt 0,0 V anlagen. Also keine Vorspannung. Damit muss es sich um ein Elektret-Kondensator-Mikrofon handeln. Ich war trotz des guten Klangs etwas enttäuscht. Das SMD-Dreibein entpuppte sich als „J35", ein N-Kanal-JFET des Typs 2SK1109 mit einem Drain-Strom von etwa 200 µA bei kurzgeschlossenem Gate.

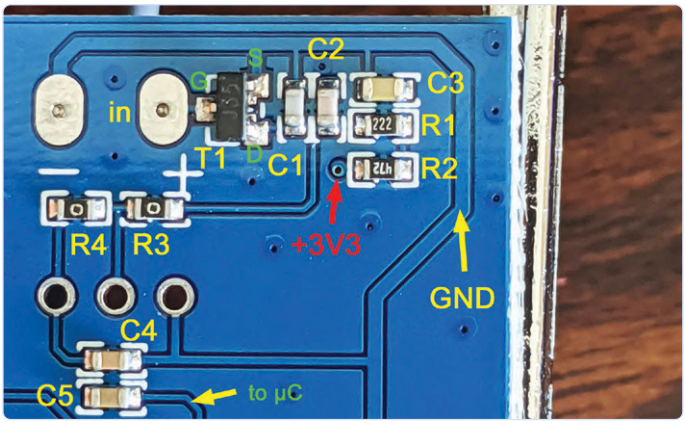


Bild 4. Beschriftete Detailaufnahme der Schaltung des Impedanzwandlers.

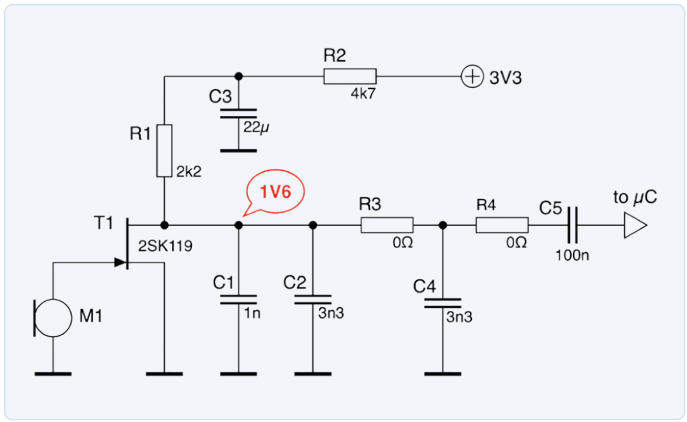


Bild 5. Die Schaltung des Impedanzwandlers.

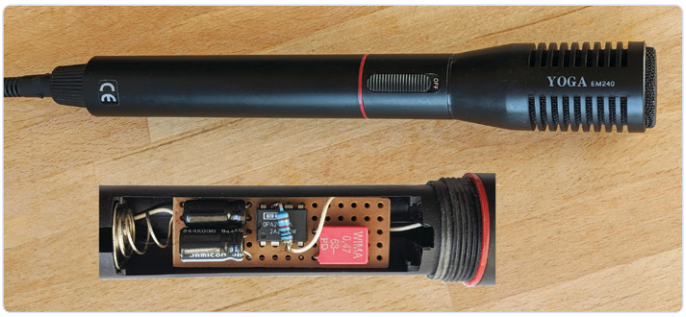


Bild 6. Das Billig-Mikrofon Yoga EM240 mit eingebautem Vorverstärker.

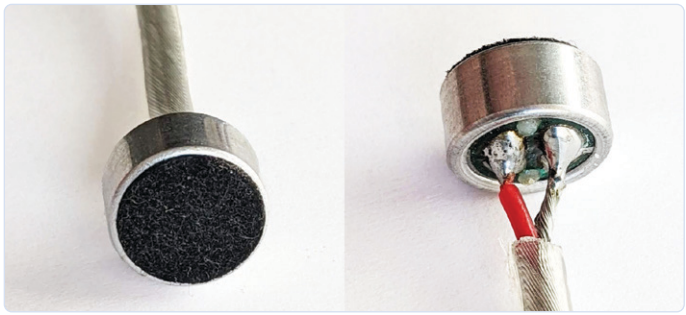


Bild 7. Eine kleine Elektret-Kapsel mit einer 5-mm-Membran.

Impedanzwandler

Bild 4 zeigt den Ausschnitt der Platine rund um diesen JFET. Eine Impedanzwandlung ist nötig, da die Kapazität der Mikrofonmembran ja sehr niedrig und ihre Impedanz für Audiosignale sehr hoch ist. Dank Source-Schaltung verstärkt T1 – der Ansatz für einen höheren Ausgangspegel. In **Bild 5** ist die Schaltung der Bauteile von Bild 4 zu sehen. R2 dient zusammen mit C3 der Siebung der Versorgungsspannung. An R1 und R2 fallen zusammen etwa 1,7 V ab; der Drain-Strom beträgt also rund 245 µA.

Was auffällt: Ohne die Gleichstromeinstellung zu gefährden kann man einfach R1 und R2 vertauschen und schon verdoppelt sich die Verstärkung (≈ +6 dB). Das macht keine Probleme, da der Tiefpass aus 2,2 kΩ und 22 µF immer noch bei guten 3,2 Hz liegt. Allerdings liegt dann der Tiefpass aus R1 = 4,7 kΩ und C1+C2+C4 bei unter 5 kHz. Das lässt sich ändern, indem man C2 oder C4 entfernt.

Resultate

Nach erfolgter Modifikation lag der Pegel jetzt sogar leicht über dem meiner Webcam und der Klang war unverändert. Der Eingriff ist also gut gegangen.

Nun interessierte mich, wie das geänderte CAD-Mikro im Vergleich mit anderen Mikrofonen so klingt. Also habe ich Aufzeichnungen davon, von meiner Webcam, von meinem MacBook Pro sowie von einem alten Billig-Mikrofon (**Bild 6**) und einer preiswerten, nackten Elektret-Mikrofonkapsel (**Bild 7**) gemacht. Nicht ganz überraschend klang das MacBook pro (subjektiv) am schlechtesten. Das Fehlen von Bässen machte aus meiner sonoren eine eher blecherne Stimme. Beim Mikrofon von Yoga hatte ich schon vor Jahren einen rauscharmen Vorverstärker eingebaut, da es im Originalzustand trotz niedrigem Pegel zu viel rauschte.

Damit Sie sich selbst einen Eindruck verschaffen können, habe ich kurze MP3-Dateien generiert, die Sie von der Elektor-Webseite zu diesem Artikel [3] herunterladen können. Zu hören ist jeweils ein kurzer Satz und etwa eine Sekunde Rauschen – alles auf gleichen Pegel gebracht, sodass sich lediglich der Klang unterscheidet. **Bild 8** zeigt die fünf verschiedenen Störpegel, jeweils um 30..40 dB verstärkt. Nur die Mikrofone von CAD, Logitech und Yoga zeigen „richtiges" Rauschen. Beim Logitech ist zudem ein überlagertes Sirren zu hören, beim MacBook Pro HF und bei der Elektret-Kapsel ein Netzbrummen.

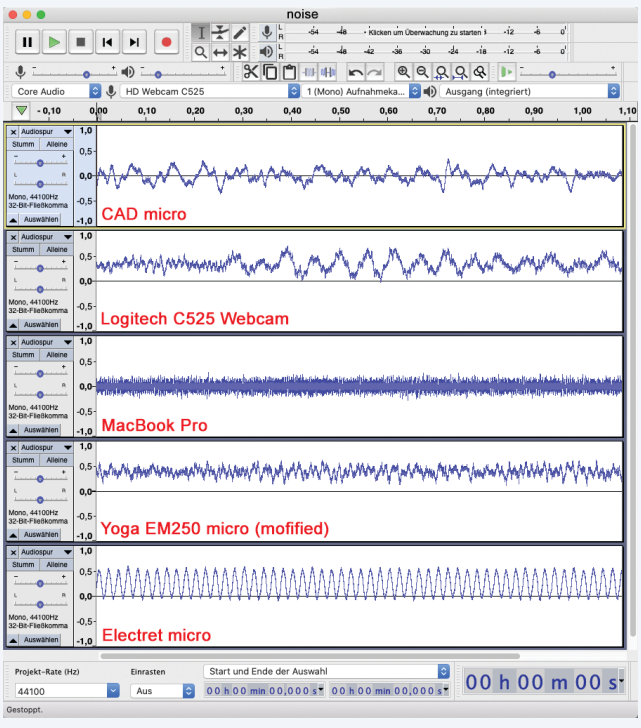


Bild 8. Unterschiedliche Störpegel der verschiedenen Mikrofone.

Bild 9. So ist das CAD-Mikrofon richtig positioniert: Rechts (oder links) neben dem Monitor.

Fazit

Ob ich mit dem Klang des CAD-Mikrofons zufrieden bin? Meine Antwort könnte von Radio Eriwan sein: Im Prinzip ja, aber...

Einerseits wurde mir eine kleine als Großmembran verkauft. Papier und HTML sind eben geduldig. Auf der anderen Seite war es preiswert und tatsächlich besser als das Mikro der Webcam, und das ist doch schon mal was.

Übrigens: So ein Mikrofon bespricht man nicht von vorne, sondern wegen seiner Richtcharakteristik von der Seite, wie in **Bild 9** gezeigt.

200434-02

Sie haben Fragen oder Kommentare?

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

Passende Produkte

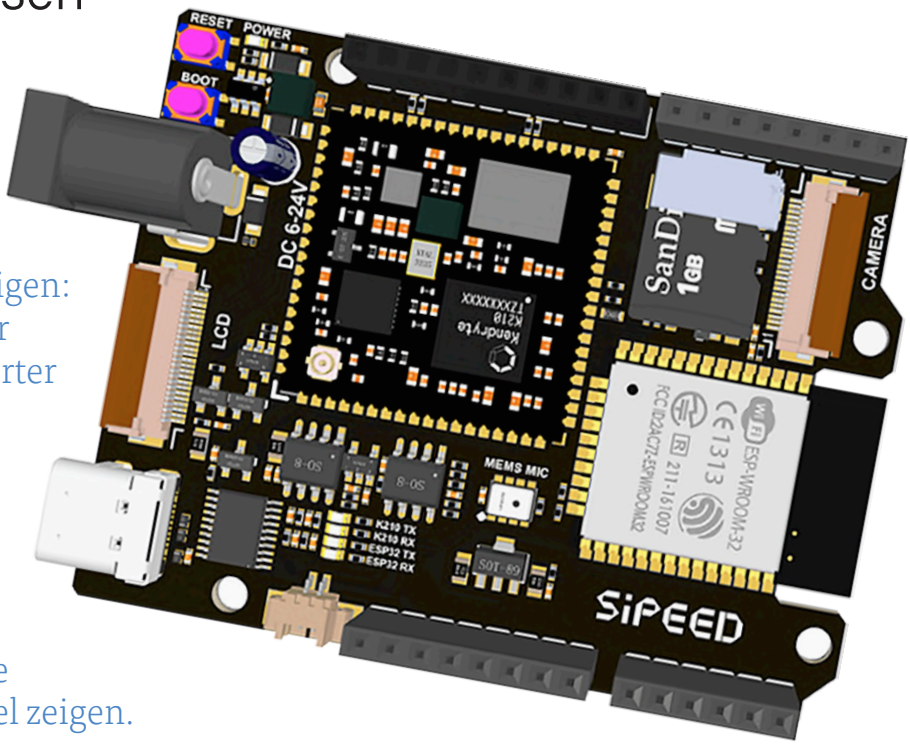
- **Mikrofon-Vorverstärker (Leerplatine) (SKU 18096)**
  www.elektor.de/18096
- **Robert Sontheimer, Audioschaltungstechnik**
  E-Buch (PDF): www.elektor.de/18458

WEBLINKS

[1] Wikipedia: Mikrofonierung: https://de.wikipedia.org/wiki/Mikrofonierung
[2] Audacity: www.audacity.de
[3] Audio-Dateien: www.elektormagazine.de/200434-1

FFT mit Maixduino

Frequenzspektren erfassen



Von **Tam Hanna**

Mit der Fast Fourier Transformation lassen sich nicht nur Aufgaben im Audiobereich erledigen: Wer die Geräusche einer Turbine oder die Spannungspegel schlecht gepufferter Fahrzeuge analysiert, kann in vielen Fällen Rückschlüsse auf Drehzahlen und Co. ziehen. Das bei Elektor erhältliche Maixduino-Board mit dem K210-SoC eignet sich nicht nur für erste Versuche mit Neuronalen Netzen, sondern auch für die schnelle Echtzeit-FFT, wie wir in diesem Artikel zeigen.

Die Weiterentwicklung sowohl von Software als auch von Hardware haben dazu geführt, dass einst nur für Privatmilizen und Regierungen mögliche Technologien auch für Otto Normalentwickler verfügbar sind. Das Halbleiterunternehmen Sipeed, bisher vor allem bekannt für RISC-V-Prozessor- und Cryptoexperimente, bietet mit dem Maixduino nun eine neue Entwicklungsplatine an. Das Board bringt neben einem für die Netzwerk-Kommunikation zuständigen ESP32 auch ein als *K210 Kendryte* bezeichnetes SoC mit, das von Seiten des Herstellers für die Verwendung in AI-Applikationen vorgesehen ist. Die Nennung des Namens Kendryte ist keine Schleichwerbung von Elektor, weil sich viele Informationen in Google nur durch den Suchbegriff „kendryte + Problem" finden lassen.

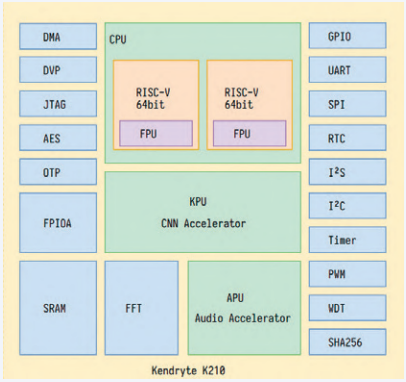


Bild 1. Der K210 bringt einige Erweiterungen mit. (Quelle: Canaan Inc.)

Wer mehr über das verwendete SoC erfahren möchte, sollte die stellenweise etwas langsam reagierende Webseite [1] laden. Im Datenblatt des K210 findet sich vor allem eine Beschreibung der Pins, interessant ist aber auch das Architekturdiagramm in **Bild 1**.

Der Hauptprozessor (CPU) basiert auf der quelloffenen RISC-V-Architektur (mit den Optionen I, M, A und C). Diese Kerne werden von zwei Helferprozessoren flankiert: Erstens gibt es die KPU, hinter der sich ein Neural Network Processor verbirgt (ohne weitere Informationen im Datenblatt), der ein in Hardware gehaltenes Neuronales Netzwerk realisiert. Analog zum einst von Danaher im TDS 5xxD und TDS 7xxD verwendeten DPO-Tristar-Prozessor haben wir es auch hier mit einem One-Trick-Pony zu tun.

Die andere Erweiterung ist der Audioprozessor mit der Bezeichnung APU. Er verarbeitet Audioinformationen mit bis zu 192 kHz, legt diese Informationen per DMA direkt im Arbeitsspeicher ab und kann zudem automatisch FFT-Prozesse anwenden. Wir werden diese APU in den folgenden Schritten zur Realisierung einer kleinen FFT nutzen.

Der Vollständigkeit halber sei auf die übliche Beigabe diverser Peripherie hingewiesen, mit der wir uns an dieser Stelle aber nicht aufhalten wollen.

Eine Frage des Werkzeugs

Interessanter für uns sind die Entwicklungstools. Auf dem untersten Level steht dabei das unter [2] zum Download bereitstehende

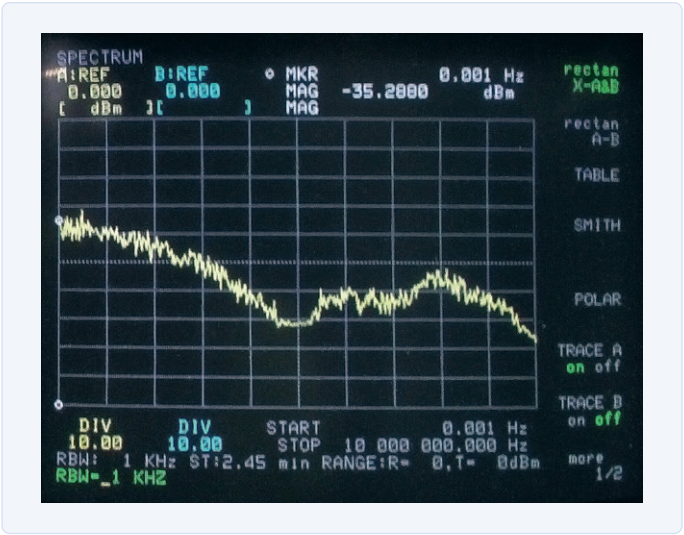


Bild 2. Schmale Frequenzbänder liefern eine hohe Auflösung, brauchen aber viel Sweep Time (ST)...

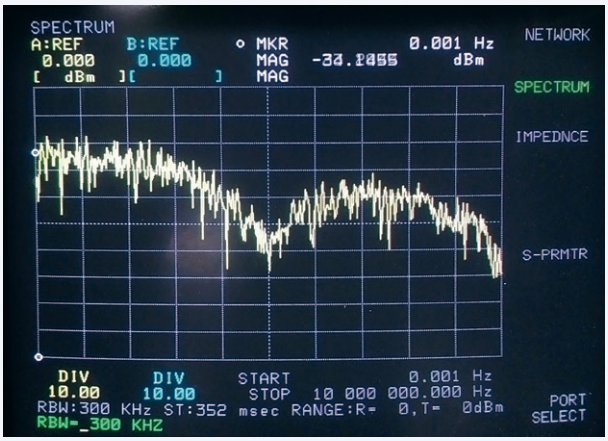


Bild 3. ...während schneller sweepbare, breitere Bänder weniger genaue Informationen liefern.

Stand-alone SDK - es ist eine (erfreulicherweise) in C gehaltene Bibliothek, die uns die direkte Interaktion mit Prozessorprimitiva ermöglicht. Eine Ebene darüber gibt es mit ArduinoCore [3] ein für die Arduino-IDE vorgesehenes Framework, das sich allerdings auch mit Platform.IO verwenden lässt. Zu guter Letzt gibt es mit MaixPy eine von MicroPython von abgeleitete Programmierumgebung, die sich an alle richtet, die mit Python sehr viel Erfahrung haben (bitte zuerst den Textkasten lesen!).

Zur Demonstration der Möglichkeiten der FFT wollen wir auf Audiodaten setzen - der Maixduino bringt ein MEMS-Mikrofon mit, das über den I2S-Bus mit dem Prozessrechner verbunden ist. Es handelt sich dabei um eine „niedrig hängende Frucht" - die unter [4] veröffentlichte Beispiel-Applikation dürfte eine der ersten Anwendungen gewesen sein. Beginnen wir mit der Einbindung einiger benötigter Headerdateien:

```
#include <Sipeed_ST7789.h>
#include "i2s.h"
#include "fft.h"
```

Im nächsten Schritt müssen wir uns Gedanken darüber machen, wie die zu verarbeitenden Puffer aufzubauen sind. Im in der Einleitung genannten Datenblatt stellen wir fest, dass der Kendryte nicht mit beliebigen Größen arbeiten kann - wer Hardware-Beschleunigung nutzen möchte, muss sich für die Puffergrößen 64, 128, 256 oder 512 entscheiden.
Auch wenn manche Mathematiker darüber jammern mögen, ist es vernünftig, bei der Arbeit mit der FFT immer einen klassischen Spektralanalysator im Hinterkopf (oder besser auf dem Labortisch) zu haben. In unserem Fall ist die Eimergröße insofern relevant, als sie die Feinheit der entstehenden Frequenzbänder beschreibt.
Bild 2 und **Bild 3** sind vom altehrwürdigen HP4195A des Autors abfotografiert und illustrieren prinzipiell die Vorgehensweise. Während unser analog arbeitender Spektrumanalysator aufgrund von Rauschen in beiden Fällen „unruhige" Bänder anzeigt, ist die tatsächliche Kurve bei der FFT gleichmäßiger. Die Frequenzauflösung folgt aus der Formel *Sampling Rate / FFT-Size*; bei der Erbeutung von 1024 Samples mit 8192 Hz beträgt die Auflösebandbreite RBW = 8 Hz.
Wie so oft gilt auch bei der FFT, dass Maximierung nicht alles ist. Der Zusammenhang zwischen der Anzahl der Frequenzbänder und der Anzahl der eingehenden Samples ist prinzipiell der Faktor 2; als kleine Erschwernis kommt dabei allerdings hinzu, dass FFT-Algorithmen mit wachsender Zahl von Samples sehr schnell sehr komplex werden. Dies führt zu einem Wachstumsverhalten wie in **Bild 4**, in dem die Genauigkeit unserer FFT-Operation durch Probleme aus der „realen" Welt beschränkt wird.

Im nächsten Schritt legen wir eine Gruppe von Konstanten fest. In der Praxis ist es empfehlenswert, insbesondere die FFT-Länge über ein `#define` festzulegen, um spätere Anpassungen zwischen Genauigkeit und Geschwindigkeit zu erleichtern:

```
uint32_t rx_buf[1024];
#define FFT_N                512U
#define FFT_FORWARD_SHIFT    0x0U
#define SAMPLE_RATE          38640
#define FRAME_LEN            512
```

Das eigentliche Vorhalten der per DMA angelieferten Informationen setzt dann noch eine Gruppe zusätzlicher Speicherfelder voraus:

```
int16_t  real[FFT_N];
int16_t  imag[FFT_N];
int    hard_power[FFT_N];
uint64_t fft_out_data[FFT_N / 2];
uint64_t buffer_input[FFT_N];
uint64_t buffer_output[FFT_N];
fft_data_t *output_data;
fft_data_t *input_data;
complex_hard_t data_hard[FFT_N] = {0};
```

Initialisierung der Hardware

Der Hersteller des Maixduino spendiert uns - wie in der Einleitung schon gesagt - ein MEMS-Mikrofon des Typs MSM261S4030H0. Da das Hantieren mit diesen kleinen Mikrofonen immer wieder interessant ist, sei hier auf **Bild 5** verwiesen. Anhand der Pinbelegung ist klar erkennbar, dass sich das Bauteil von sich aus um die Signaladaptierung kümmert und die Daten über einen seriellen Bus ausliefert. Als erstes Hindernis präsentiert sich, dass die Software die für die Kommunikation mit dem Mikrofon erforderlichen Datenleitungen nicht automatisch initialisiert. Dies lässt sich im Beginn von `setup` problemlos beheben:

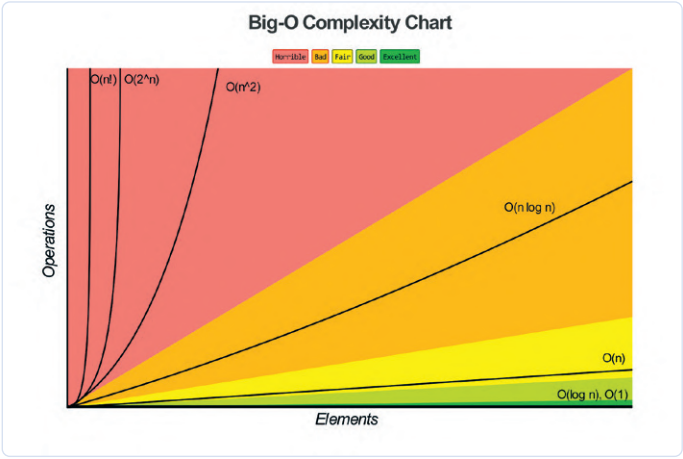
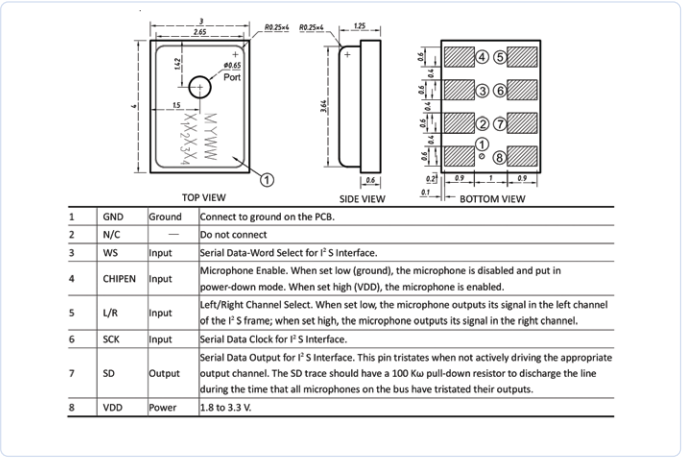


Bild 4. Wer unendlich kleine Frequenzbänder möchte, benötigt viel Rechenleistung. (Bild: https://www.bigocheatsheet.com/)



1	GND	Ground	Connect to ground on the PCB.
2	N/C	—	Do not connect
3	WS	Input	Serial Data-Word Select for I²S Interface.
4	CHIPEN	Input	Microphone Enable. When set low (ground), the microphone is disabled and put in power-down mode. When set high (VDD), the microphone is enabled.
5	L/R	Input	Left/Right Channel Select. When set low, the microphone outputs its signal in the left channel of the I²S frame; when set high, the microphone outputs its signal in the right channel.
6	SCK	Input	Serial Data Clock for I²S Interface.
7	SD	Output	Serial Data Output for I²S Interface. This pin tristates when not actively driving the appropriate output channel. The SD trace should have a 100 KΩ pull-down resistor to discharge the line during the time that all microphones on the bus have tristated their outputs.
8	VDD	Power	1.8 to 3.3 V.

Bild 5. MEMSensing erspart dem Sensor-Anwender viel Arbeit. (Bild: MEMSensing Microsystems)

```
void setup()
{
    lcd.begin(15000000, COLOR_RED);
    fpioa_set_function(20, FUNC_I2S0_IN_D0);
    fpioa_set_function(18, FUNC_I2S0_SCLK);
    fpioa_set_function(19, FUNC_I2S0_WS);
```

Im nächsten Schritt ist eine Konfiguration der I2S-Hardware erforderlich. Der Code ist im Allgemeinen durch die Hardware des Boards vorgegeben; als wichtige Einstellung bietet sich die mit der Methode `i2s_set_sample_rate` festzulegende Sample-Rate an. Da wir angesichts der Größe der FFT ja nur vergleichsweise wenige Optionen haben, ist die Sample-Rate eines der wenigen Mittel zur „Verarbeitung" der Frequenzband-Breite:

```
    i2s_init(I2S_DEVICE_0, I2S_RECEIVER, 0x3);
    i2s_set_sample_rate(I2S_DEVICE_0, SAMPLE_RATE );
    i2s_rx_channel_config(I2S_DEVICE_0, I2S_CHANNEL_0,
                  RESOLUTION_16_BIT, SCLK_CYCLES_32,
                    TRIGGER_LEVEL_4, STANDARD_MODE);
```

Danach fehlen nur noch einige Initialisierungen:

```
    dmac_init();
    sysctl_enable_irq();
}
```

Unsere nächste Aufgabe ist die Durchführung der eigentlichen FFT-Operation. Dabei müssen wir uns im ersten Schritt darum kümmern, dass die per I2S zur Verfügung gestellten Informationen im Speicher liegen:

```
void loop()
{
  int i, sampleID;

  i2s_receive_data_dma(I2S_DEVICE_0, &rx_buf[0],
                    FRAME_LEN * 2, DMAC_CHANNEL3);
  dmac_wait_idle(DMAC_CHANNEL3);
```

Durch den Aufruf von `dmac_wait_idle` befehlen wir dem Prozessor, bis zum Freiwerden des DMA-Kanals eine Pause einzulegen. Auf diese Art und Weise ist sichergestellt, dass jeder Durchlauf von `loop` mit einer konsistenten und neuen Datenbasis arbeitet. Entfernt man (versuchsweise) den Aufruf von `dmac_wait_idle`, erhält man seltsame Aliasing-Artefakte auf dem Bildschirm. Dies dürfte daran liegen, dass die Verwendung der FFT-Module laut Datenblatt ausschließlich über die DMA-Engine erfolgen kann. Kommt es hierbei zu Kollisionen, so droht Unheil!

Bei der Arbeit mit den FFT-Engines und ähnlichen festen Funktionseinheiten müssen Sie immer darauf achten, dass die Hardware die Lieferung der Informationen in einem beschriebenen Speicherformat erwartet. Im Fall von `tag_fft_data` handelt es sich um folgende Struktur, die im SDK an mehreren Stellen deklariert ist:

```
typedef struct tag_fft_data
{
    int16_t  I1;
    int16_t  R1;
    int16_t  I2;
    int16_t  R2;
} fft_data_t
```

Aufmerksame Leser fragen sich an dieser Stelle, warum zwei Komponenten namens R und I erwartet werden - zur Erklärung müssen wir im ersten Schritt festlegen, dass R für Real und I für Imaginär steht. Die Fouriertransformation ist im klassischen, also symbolischen mathematischen Bereich ja als Operation auf Basis komplexer Zahlen definiert. FFT-Algorithmen erwarten deshalb - zumindest in der generischen Version, die Hardware-Entwickler gerne im Interesse maximaler Flexibilität in Hardware gießen - die Lieferung von sowohl realen als auch imaginären Komponenten. Da unsere Toninformationen allerdings ausschließlich aus realen Elementen bestehen, setzen wir die Werte von I an dieser Stelle immer auf null:

```
    for ( i = 0; i < FFT_N / 2; ++i)
    {
        input_data = (fft_data_t *)&buffer_input[i];
        input_data->R1 = rx_buf[2*i];
        input_data->I1 = 0;
        input_data->R2 = rx_buf[2*i+1];
        input_data->I2 = 0;
    }
```

Hier bietet sich das „Recyclieren" des FFT-Prozesses an. Der antiquarisch manchmal günstig und online als PDF kostenlos erhältliche Klassiker „Numerical Recipies in C" [5] offeriert hierzu im zwölften Kapitel (der zweiten Ausgabe) einige interessante Formeln, deren Verständnis allerdings fundierte Mathematikkenntnisse voraussetzt. Lohn der Mühen wäre hier, dass ein FFT-Prozess zwei reale Funktionen gleich-

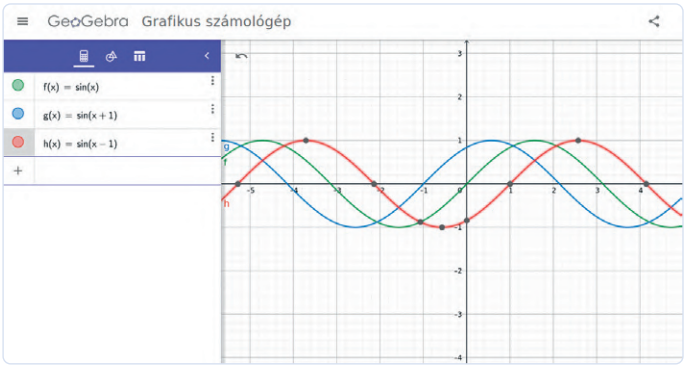


Bild 6. Diese Schwingungen sind trotz identischer Frequenz und Amplitude nicht identisch.

zeitig beackert – der Entwickler kann die ausgegebenen Informationen dann mit sehr geringem Aufwand separieren.
In der Praxis ist die FFT einer jener Bereiche, die man als Elektroniker ad infinitum spielen kann - diese Vorgehensweise ist in der Praxis sogar oft effektiver als ein immenser mathematischer Lernaufwand. Wir wollen als nächsten Schritt die FFT-Befehle ausführen, um danach die Ergebnisse zu gewinnen:

```
fft_complex_uint16_dma(DMAC_CHANNEL0, DMAC_CHANNEL1,
   FFT_FORWARD_SHIFT, FFT_DIR_FORWARD, buffer_input,
   FFT_N, buffer_output);
```

Beachten Sie hier die Übergabe der Operation `FFT_DIR_FORWARD`, die die Hardware-Engine darüber informiert, dass wir aus dem Zeit- in den Frequenzbereich überwechseln. Es gibt in der Praxis immer wieder Situationen, in denen man ein im Frequenzbereich vorliegendes Signal in den Zeitbereich zurück transferieren möchte - in diesem Fall würde hier die Konstante `FFT_DIR_BACKWARD` übergeben. Daraufhin werden jedenfalls die Daten geerntet:

```
for ( i = 0; i < FFT_N / 2; i++)
{
    output_data = (fft_data_t*)&buffer_output[i];
    data_hard[2 * i].imag = output_data->I1 ;
    data_hard[2 * i].real = output_data->R1 ;
    data_hard[2 * i + 1].imag = output_data->I2 ;
    data_hard[2 * i + 1].real = output_data->R2 ;
}
```

Beachten Sie, dass die zurückgegebenen Informationen einen Real-und einen Imaginärteil besitzen. Zum Verständnis dieses Problems wollen wir abermals, wenn auch kurz, in die Mathematik zurückkehren. Die Formel

$$S_f(t) = a_0/2 \, \Sigma \, [a_n \cos(n\omega t) + b_n \sin(n\omega t)]$$

zeigt eine von vielen Darstellungen der grundlegenden Fourier-Reihe. Die Verwendung von I beziehungsweise J in der Formel zeigt uns, dass wir mit komplexen Einheiten arbeiten – da die FFT im Grunde genommen nur eine numerische Repräsentation der klassischen Fouriertransformation darstellt, muss sie sich an die Regeln halten.

Wieso komplex?

Unsere soeben durchgeführte Sampling-Operation ist zwar praktisch real, macht die Mathematikerin aber nicht froh. Das liegt unter anderem daran, dass es keinen „definierten" Trigger gibt - aus Sicht der Mathematikerin sind die in **Bild 6** gezeigten Sinus-Signale phasenverschoben.

Die FFT-Operation bringt dies dadurch zum Ausdruck, dass die zurückgelieferten Werte komplexe Zahlen darstellen. Dadurch lässt sich sowohl die Mächtigkeit als auch die Phasenverschiebung der jeweiligen Komponente ausdrücken – die beiden Sinuswellen wären auch im Ergebnis unterschiedlich.
Der Elektroniker interessiert sich meist nicht für die komplexen Komponenten, weshalb wir sie durch Berechnung des Absolutbetrags zerstören:

```
float pmax=10;
for (i = 0; i < FFT_N; i++)
{
    hard_power[i] = sqrt(data_hard[i].real *
             data_hard[i].real + data_hard[i].imag *
             data_hard[i].imag);
```

Logarithmen und Quadratwurzeln laden zur Vereinfachung ein, so dass sich die Berechnungen mathematisch optimieren lassen. Einfacher bedeutet aber nicht, dass die Vorgehensweise leichter verständlich wären – im Gegenteil. Deshalb soll die Vereinfachung hier im Interesse besserer Übersichtlichkeit unterbleiben.
Wer die weiter oben genannte Zusammenfassung zweierlei Funktionen in einer FFT-Operation durchführt, müsste an dieser Stelle eine Extraktion durchführen und die beiden Ergebnisse (durch einen im Vergleich zur FFT einfachen Algorithmus) trennen. Wir führen im nächsten Schritt noch eine Logarithmierung durch, um dB-artige Werte zu erhalten:

```
     hard_power[i] = 20*log10(2*hard_power[i]/FFT_N);
     if( hard_power[i]>pmax && i>5)
         pmax = hard_power[i];
 }
```

Unsere nächste Aufgabe besteht jedenfalls darin, die Werte auf einem Bildschirm darzustellen. Als Haupt-Ärgernis erweist sich hier die Display-API des Arduinos, die beim Übergeben „ungültiger" Koordinatenwerte stark an Performance einbüßt:

```
    lcd.setRotation(0);
    lcd.fillScreen(COLOR_WHITE);
    for (int i=0;i<256;i++)
    {
      int dval = 240 - 240*(hard_power[i]/pmax);
      if (dval<2)dval=1;
      if (dval>238)dval=238;
```

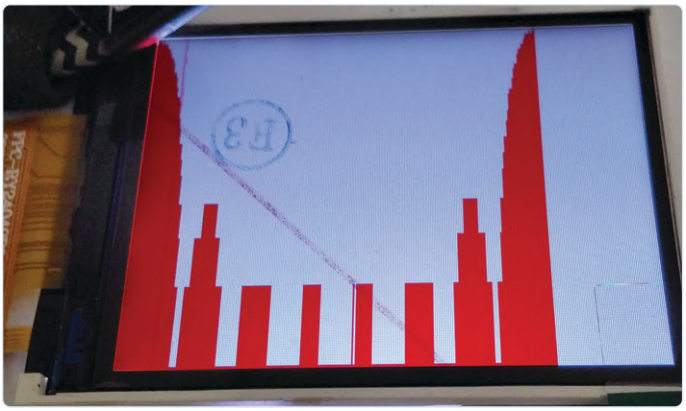


Bild 7. Diese „Symmetrie" ist verwirrend.

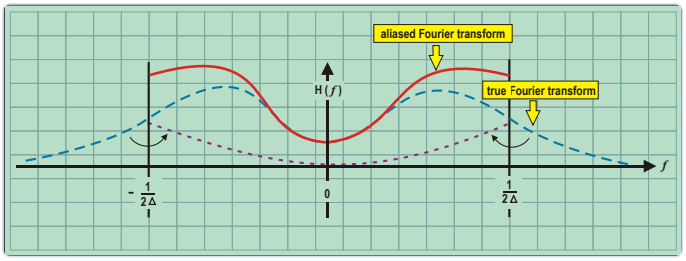


Bild 8. Vom Nyquist-Fenster „eliminierte" Spektralelemente tauchen als Artefakte auf.

```
      lcd.drawLine( i,  240, i, dval  , COLOR_RED);
    }
    delay(50);
}
```

Damit sind wir hier für einen ersten Testlauf bereit, der zum in **Bild 7** gezeigten Ergebnis führt. Wer möchte, kann an dieser Stelle noch Testtöne in das System einbringen - Lohn dieser Mühen ist, dass man den Testton beziehungsweise die ihn repräsentierende Zeile am Bildschirm verschieben kann.

Von der Spiegelung

Die FFT ist einer jener Themen, bei denen man als Elektroniker Tausende von Stunden verbringen kann. Aufmerksamen Beobachtern wird auffallen, dass ein Testton sowohl von der „linken" als auch von der „rechten" Seite über den Bildschirm wandert.
An dieser Stelle warten mehrere Hindernisse. So wiederholt sich unsere FFT schon beim 128. Feld, was gemäß der soeben festgestellten Theoreme erst beim 256. Feld passieren dürfte. Ursache dafür ist eine Problematik bei der Anlieferung der Samples über den I2S-Bus - ein

MEHRERE ARTEN VON PYTHON

Im Bereich Python für Mikrocontroller kämpfen zwei Konzepte um die Vorherrschaft. Das eine Konzept ist MicroPython, das eine vollwertige Runtime implementiert. Das Zielsystem enthält einen kompletten Python-Interpreter, der analog zu einer Workstation sogar Bibliotheken aus dem Internet lädt. Vorteil dieser Vorgehensweise ist eine höhere Flexibilität, nachteilig ist, dass es wegen der doch eng beschränkten Ressourcen (Stichwort: Arbeitsspeicher) zu Problemen kommen kann.

Die andere Vorgehensweise wird derzeit ausschließlich vom am Maixduino nicht verfügbaren Zerynth repräsentiert. Die Workstation erzeugt einen Monolithen, der auf das Zielsystem gebrannt wird. Der geringeren Flexibilität steht höhere Stabilität entgegen – befinden sich die Bibliotheken auf der Workstation, so lassen sich ressourcenintensive Prozesse abseits des Mikrocontrollers erledigen.

Thema, das jetzt aber nicht weiter vertieft werden soll. Unter [6] findet sich eine angepasste Version, die das Problem abmildert.
Die Betonung liegt hierbei auf dem Begriff abmildert, weil im Fall von FFTs gegen komplett reale Eingabewerte immer eine Spiegelung auftritt. Die Werte von null bis N/2 entsprechen dabei denen von N/2+1 bis N. Ursache dafür ist, dass sich die komplexen Anteile der FFT ja aufheben müssen.
Für weitere Spiele mit der FFT gibt es übrigens noch eine ganze Menge von Tätigkeitsbereichen, zum Beispiel die Einführung einer Fensterfunktion, um den Eingang zu „beschneiden". Dahinter steht der Gedanke, dass außerhalb des Nyquist-Fensters liegende Teile auf den beiden Seiten nach innen gespiegelt werden, wobei eine Fensterfunktion dies unterdrücken würde (siehe **Bild 8**). Ein empfehlenswertes Dokument hierfür ist der im Jahre 1981 von Albert H. Nuttall veröffentlichte Artikel „Some Windows with Very Good Sidelobe Behavior" [7].

Fazit

Die Hardware-FFT-Engine ermöglicht niederschwellige Experimente mit FFT-Algorithmen und Anwendungen. Es handelt sich dabei um eine Fähigkeit, die man sowohl im zivilen als auch im militärischen Bereich immer wieder benötigt. Wer eine solche Platine erwirbt und damit ein wenig spielt, dürfte den Zeiteinsatz und finanzielle Auslagen mit Sicherheit nicht bereuen.

200297-01

Passende Produkte

- **Sipeed MAix BiT Kit für RISC-V AI+IoT (SKU 19239)**
  www.elektor.de/19239

WEB LINKS

[1] Kendryte K210: https://bit.ly/3fptbHu
[2] Standalone-SDK: https://github.com/kendryte/kendryte-standalone-sdk
[3] ArduinoCore-k210: https://github.com/Seeed-Studio/ArduinoCore-k210
[4] fft_lcd: https://github.com/MrJBSwe/fft_lcd/blob/master/main.c
[5] Teukolsky, Press, Vetterling und Flannery: Numerical Recipes in C: www.numerical.recipes/
[6] stft_lcd: https://github.com/jpiat/stft_lcd
[7] Albert H. Nuttall: Some Windows with Very Good Sidelobe Behavior : https://bit.ly/3fTkCWb

LISTING 1. DAS VOLLSTÄNDIGE MAIXDUINO-LISTING.

```
#include <Sipeed_ST7789.h>
#include "i2s.h"
#include "fft.h"
SPIClass spi_(SPI0); // MUST be SPI0 for Maix series on board LCD
Sipeed_ST7789 lcd(320, 240, spi_);
uint32_t rx_buf[1024];
#define FFT_N               512U
#define FFT_FORWARD_SHIFT   0x0U
#define SAMPLE_RATE         38640
int16_t  real[FFT_N];
int16_t  imag[FFT_N];
int    hard_power[FFT_N];
uint64_t fft_out_data[FFT_N / 2];
uint64_t buffer_input[FFT_N];
uint64_t buffer_output[FFT_N];
fft_data_t *output_data;
fft_data_t *input_data;
complex_hard_t data_hard[FFT_N] = ;
#define FRAME_LEN           512

void setup()
 {
     lcd.begin(15000000, COLOR_RED);
```

Fortsetzung auf der nächsten Seite

```
    fpioa_set_function(20, FUNC_I2S0_IN_D0);
    fpioa_set_function(18, FUNC_I2S0_SCLK);
    fpioa_set_function(19, FUNC_I2S0_WS);

   i2s_init(I2S_DEVICE_0, I2S_RECEIVER, 0x3);
   i2s_set_sample_rate(I2S_DEVICE_0, SAMPLE_RATE );
   i2s_rx_channel_config(I2S_DEVICE_0, I2S_CHANNEL_0,
                         RESOLUTION_16_BIT, SCLK_CYCLES_32,
                         TRIGGER_LEVEL_4, STANDARD_MODE);

    dmac_init();
    sysctl_enable_irq();
}
void loop()
 {
   int i, sampleID;

   i2s_receive_data_dma(I2S_DEVICE_0, &rx_buf[0], FRAME_LEN * 2, DMAC_CHANNEL3);
    dmac_wait_idle(DMAC_CHANNEL3);//wait to finish recv

     for ( i = 0; i < FFT_N / 2; ++i)
     {
         input_data = (fft_data_t *)&buffer_input[i];
         input_data->R1 = rx_buf[2*i];   // data_hard[2 * i].real;
         input_data->I1 = 0;             // data_hard[2 * i].imag;
         input_data->R2 = rx_buf[2*i+1]; // data_hard[2 * i + 1].real;
         input_data->I2 = 0;             // data_hard[2 * i + 1].imag;
     }

    fft_complex_uint16_dma(DMAC_CHANNEL0, DMAC_CHANNEL1, FFT_FORWARD_SHIFT, FFT_DIR_FORWARD,
                                                          buffer_input, FFT_N, buffer_output);
    for ( i = 0; i < FFT_N / 2; i++)
     {
         output_data = (fft_data_t*)&buffer_output[i];
         data_hard[2 * i].imag = output_data->I1 ;
         data_hard[2 * i].real = output_data->R1 ;
         data_hard[2 * i + 1].imag = output_data->I2 ;
         data_hard[2 * i + 1].real = output_data->R2 ;
     }
    float pmax=10;
     for (i = 0; i < FFT_N; i++)
     {
         hard_power[i] = sqrt(data_hard[i].real * data_hard[i].real +
                                          data_hard[i].imag * data_hard[i].imag);
        //Convert to dBFS
         hard_power[i] = 20*log10(2*hard_power[i]/FFT_N);
        if( hard_power[i]>pmax && i>5)
             pmax = hard_power[i];
     }

     lcd.setRotation(0);
     lcd.fillScreen(COLOR_WHITE);
     for (int i=0;i<256;i++)
     {
       int dval = 240 - 240*(hard_power[i]/pmax);
       if (dval<2)dval=1;
       if (dval>238)dval=238;
       lcd.drawLine( i,  240, i, dval  , COLOR_RED);
     }
     delay(50);
}
```

Aus dem Leben gegriffen

Entwurfslogik (oder Un-Logik)

36-poliger Amphenol-Steckverbinder, ausgewählt 1970 für Drucker. Quelle: Shutterstock / KPixMining.

Von Ilse Joostens (Belgien)

Als ich ein Teenager war, baute ich wie eine Besessene eine ganze Menge elektronischer Schaltungen. Ich kümmerte mich wenig um die Details, wie eine Schaltung funktionierte. So schnell wie möglich ein Ergebnis zu bekommen war das, was mich am meisten antrieb. Auch die saubere Fertigstellung eines Projekts ließ oft zu wünschen übrig, denn ich war vollkommen zufrieden, wenn das, was ich gebaut hatte, mehr oder weniger funktionierte.

Im Laufe der Zeit bin ich dazu übergegangen, selbst Schaltungen zu entwerfen, zunächst nur als Hobby, aber schließlich auch als Gewerbe. Vor allem in letzterem Fall musste ich mich von der Idee lösen, immer die bestmögliche Schaltung zu entwerfen. Finanzielle Erwägungen, die Verfügbarkeit von Bauteilen in den benötigten Mengen und das, was man gerade zur Hand hat, sind auch bei kommerziellen Projekten wichtige Faktoren, und so trifft man manchmal Entwurfsentscheidungen, die auf den ersten Blick nicht logisch erscheinen.

Das Centronics-Fiasko

Irgendwann benötigte ich für eines meiner früheren Projekte eine große Anzahl heller RGB-LEDs und bestellte sie kostengünstig bei einem deutschen Anbieter. Die LEDs waren völlig in Ordnung, und der Lieferant war sogar so freundlich, einen Haufen Widerstände zusammen mit den LEDs zu liefern, mit der Idee, dass sie als Vorwiderstände verwendet werden könnten. Ich brauchte diese Widerstände eigentlich nicht; die meisten hatten einen Widerstand von 510 Ω, was nicht so üblich war. Also wollte ich sie loswerden, und schließlich habe ich sie einfach bei der Konstruktion meines Fluoreszenzanzeigen-Shields für Arduino verwendet (R1 im Schaltplan) [1]. Der Wert dieses Widerstands ist nicht besonders wichtig, aber ich wurde schnell ein Opfer meines eigenen Erfolgs. Die Bausätze gingen weg wie warme Semmeln, und ehe ich mich versah, war mein Vorrat an 510-Ω-Widerständen aufgebraucht und ich musste neue bestellen. Ich musste eine vertrauenswürdigere Quelle für diese Widerstände finden, aber das war noch nicht alles - damals waren sie viel teurer als zum Beispiel 470-Ω- oder 560-Ω-Widerstände. Das war ein Schuss ins eigene Knie! Der Bausatz erfreute sich so großer Beliebtheit, dass im Internet modifizierte Versionen auftauchten, darunter auch ein Entwurf mit sechs Röhren.

Guglielmo Marconi. Quelle: Shutterstock / Morphart Creation.

Es stellte sich heraus, dass jeder den 510-Ω-Widerstand wahllos kopiert hatte.

Jetzt erinnere ich mich reumütig an die Sache mit der Centronics-Schnittstelle [2]. Im Jahr 1970 entwickelte Centronics einen preiswerten Nadeldrucker (Modell 101) [3]. Zufälligerweise verfügte die damalige US-Muttergesellschaft Wang Laboratories über einen Überschuss an 20.000 36-poligen Amphenol-Flachkabelsteckern, die ursprünglich für einen ihrer frühen Rechner bestimmt waren. Es wurde schnell beschlossen, diese Steckverbinder für die parallele Schnittstelle des neuen Druckers zu verwenden. Dieser - meiner Meinung nach etwas klobige - Stecker wurde bald zum Industriestandard, und wer weiß, wie viele Centronics danach noch bestellen musste.

Kosteneffizienz und „Muntzing“

Vor einiger Zeit stieß ich in einem deutschen Online-Forum auf Kritik an meinem Design für die Fluoreszenzröhrenuhr mit einem ESP32 [4]. Es begann damit, dass jemand beim Bau des Projekts etwas falsch gemacht hatte und dann versuchte, mich als Entwickler zu beschuldigen. Dann bemerkte ein anderer Forumsteilnehmer, dass die Verwendung einer 47-µH-Induktivität im Aufwärtswandler zu hohen Spitzenströmen führte und dass das Ganze nicht sehr effizient war, wenn man bedenkt, dass das Gate des MOSFETs direkt vom Ausgang des NE555 angesteuert wurde. Diese Kritik war nicht ganz unbegründet, aber da ich wegen eines anderen Projekts mehr oder weniger in 47-µH-Drosseln ertrinke, bin ich geneigt, sie zu verwenden, wo immer es möglich ist. Insgesamt verbraucht die Uhr etwa 2,25 W, wovon der Anteil des Aufwärtswandlers bei etwas mehr als 550 mW liegt. Bei dieser Strommenge mache ich mir keine Sorgen um die Effizienz, und ich habe auch nicht vor, einen teuren, hochmodernen Wandler zu entwickeln. Wo der Wirkungsgrad wirklich wichtig ist, verwende ich ein LC-Display anstelle von Fluoreszenz-Röhren. Ein LCD hat aber leider nicht die gleiche Ausstrahlung. Ich mag manchmal ein bisschen übertreiben, aber das finanzielle Ergebnis muss stimmen. Ich will nicht so weit gehen wie der extravagante Geschäftsmann Earl „Madman“ Muntz [5][6], der angeblich immer einen Seitenschneider in der Brusttasche trug, um „unnötige“ Bauteile aus den Entwürfen seiner Ingenieure herauszuschneiden, um die Kosten niedrig zu halten. Trotz allem der Gier bezichtigt zu werden, mit einem abfälligen Verweis auf „gierige Investmentbanker“ im selben Forum, weil meine Bausätze angeblich etwas teuer sind, geht unter die Gürtellinie. Vielleicht haben diese Leute noch nie etwas von Steuern oder Arbeitskosten gehört.

Intuition versus Vernunft

In der Schule ging es vor allem um Rechnen, Argumentieren und - wo nötig - Beweisen, wobei die Theorie im Vordergrund stand, während die Praxis und außerdem empirische Methoden etwas vernachlässigt wurden. Ihr Chef wird es aber sicher nicht schätzen, wenn Sie für eine einfache Schaltung stundenlang Berechnungen anstellen, und für mich als Selbstständige gilt das Sprichwort „Zeit ist Geld“ nur allzu sehr. Natürlich greife ich manchmal auf meine Rechenkünste zurück, aber ich habe keine Angst davor, nach Intuition zu entwerfen, insbesondere bei relativ einfachen Dingen. Manche Schaltungen erfordern einfach einen empirischen Ansatz, vor allem, wenn man mit obskuren altmodischen Bauteilen arbeitet. Man sollte auch den Mut haben, bestehende Entwürfe wiederzuverwenden, anstatt das Rad immer neu erfinden zu wollen.

Bei der Entwicklung und Verfeinerung des magnetischen Detektors, der den Spitznamen „Maggie“ [7] trug, wählte Guglielmo Marconi ebenfalls einen empirischen Ansatz, der auf den Arbeiten von Ernest Rutherford und Harry Shoemaker beruhte. Der magnetische Detektor bewährte sich bis zum Aufkommen von Detektoren mit Vakuumröhren, und ein magnetischer Detektor befand sich auch an Bord der Titanic. Es dauerte zwanzig Jahre, bis die Wissenschaftler eine angemessene Erklärung für die Funktionsweise des Geräts liefern konnten. Das ist ein Zeugnis für Marconis Fähigkeiten als praktischer Ingenieur, wobei die Betonung auf „praktisch“ liegt.

(220671-02)SG

Haftungsausschluss/Disclaimer: Aller unnötiger Text wurde sorgfältig mit der Löschtaste, die ich immer in meiner Handtasche habe, aus dieser Kolumne entfernt.

WEBLINKS

[1] Ilse Joostens, „VFD Shield für Arduino“, Elektor 9/2015: https://www.elektormagazine.de/magazine/elektor-201509/28000
[2] Centronics-Schnittstelle: https://en.wikipedia.org/wiki/Parallel_port
[3] Benutzerhandbuch Centronics 101: https://manualslib.com/manual/1438231/Centronics-101.html
[4] „VFD-Röhren-Uhr mit ESP32“, Elektor 5/2018: https://www.elektormagazine.de/magazine/elektor-201805/41489
[5] Bob Pease - „What's All This Muntzing Stuff, Anyhow?“ - Electronic Design (1992): https://elektor.link/MuntzingStuff
[6] Dokumentation Madman Muntz: https://youtu.be/deFlB2G0mH8
[7] Magnetischer Detektor: https://de.wikipedia.org/wiki/Magnetischer_Detektor

Schrittmotor-Treiber UCN5804

Bemerkenswerte Bauteile

Von David Ashton (Australien)

Ein Schrittmotor-Treiber? Was ist daran bemerkenswert? Es gibt wahrscheinlich Hunderte, vielleicht Tausende davon! Nun, die Besonderheit dieses Bauteils liegt in seiner Einfachheit. Es ist nur ein einziges IC mit Schritt- und Richtungseingängen, das einen Schrittmotor direkt ansteuern kann – ohne weitere Logik oder ICs.

Der UCN5804 ist ein Schrittmotor-Treiber-IC von Allegro Microsystems. Er wurde in einem 16-poligen Gehäuse (DIP oder SOIC) angeboten, wobei die mittleren Pins auf jeder Seite (4/5 und 12/13) miteinander und mit Masse verbunden sind. Diese Anschlüsse sorgen zusätzlich zur Masseverbindung für eine gute Wärmeableitung. Es ist ein großartiges kleines IC, das vier Wicklungen eines unipolaren 4-Phasen-Schrittmotors mit Schritt- und Richtungseingängen ansteuern kann. Seine Ausgangsleistung (1,5 A bei bis zu 35 V) ist ebenfalls beeindruckend, so dass er die meisten kleinen Schrittmotoren direkt ansteuern kann. Es gibt auch einphasige und Halbschritt-Eingänge für etwas mehr Vielseitigkeit. Schön und einfach und leicht zu bedienen.

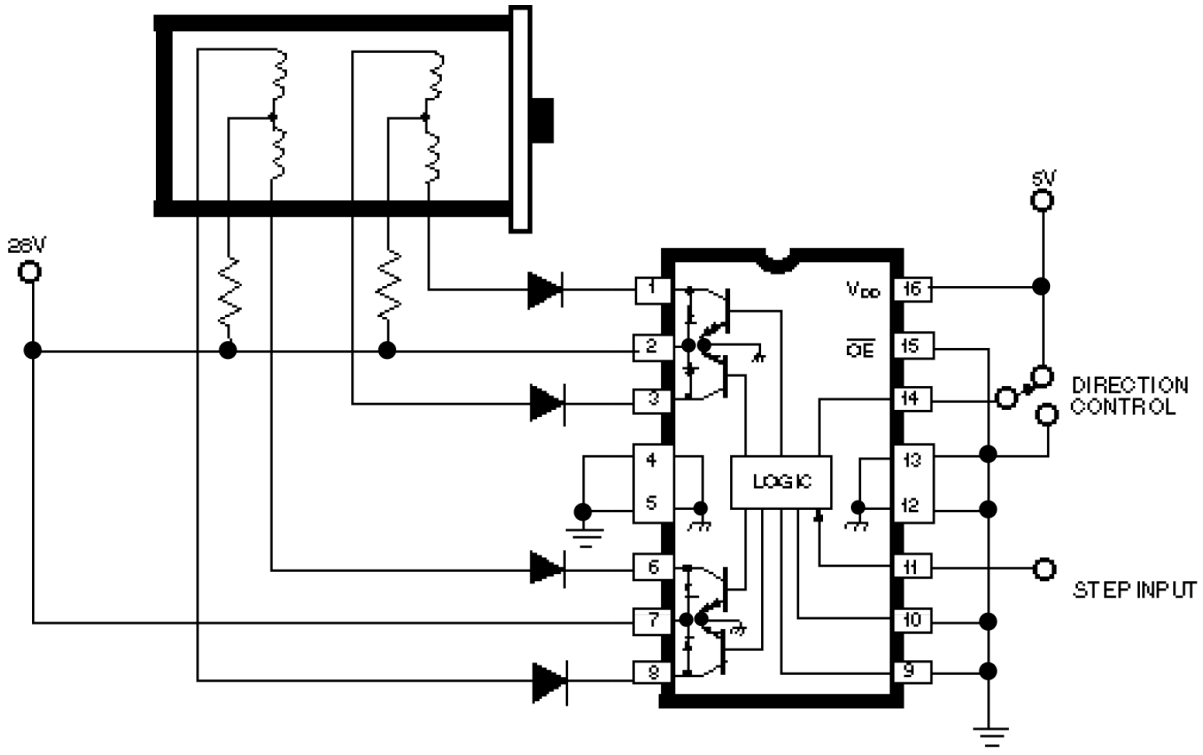


Bild 1. UCN5804-Applikationsschaltung. Quelle: Datenblatt Allegro Microsystems.

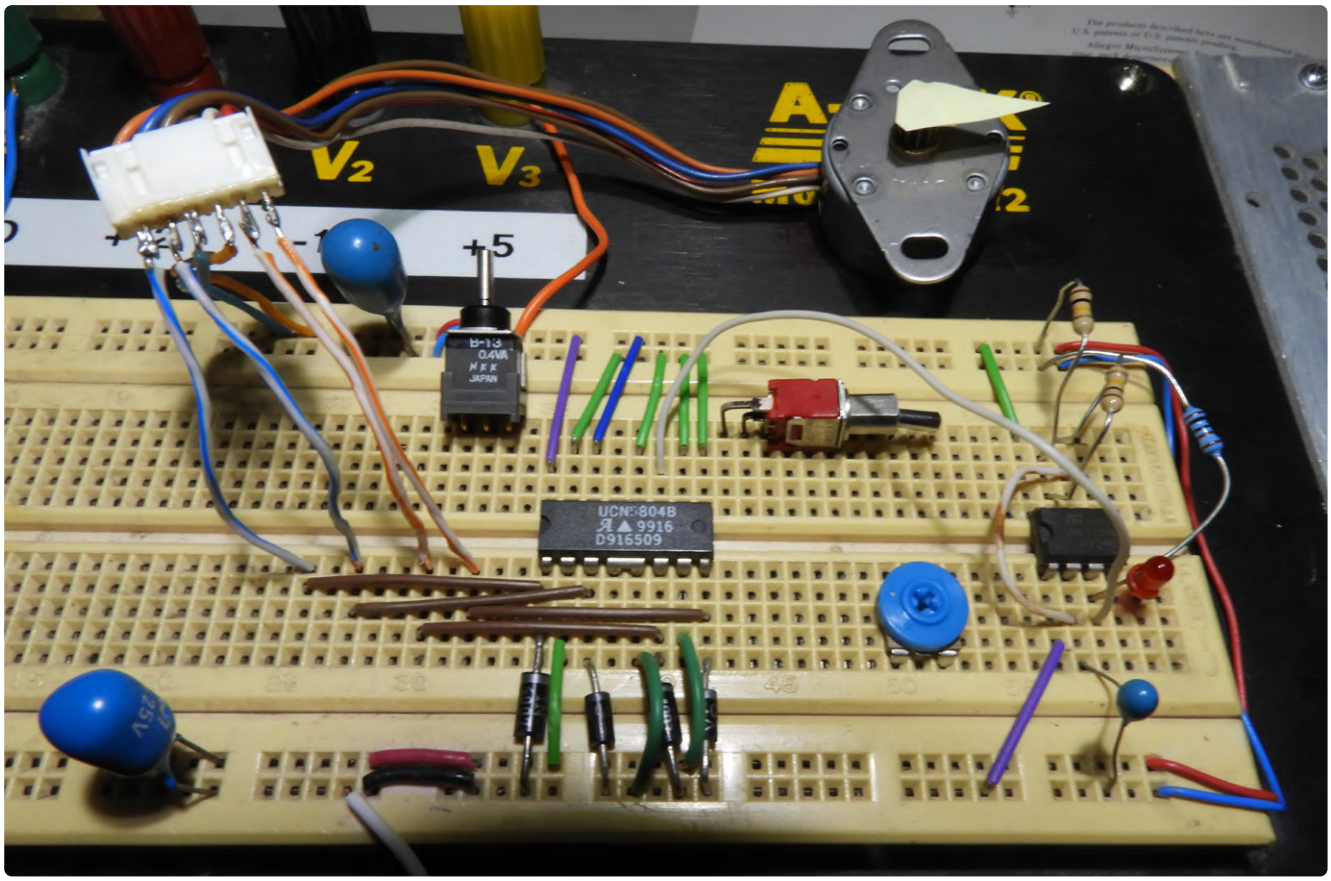


Bild 2. Eine Breadboard-Schaltung mit einem „geretteten" UCN5804, einem 555-Timer für die Schrittsteuerung und einem Schaltern für die Richtung.

Leider wurde die Produktion des Bauteils eingestellt, obwohl Allegro das Datenblatt noch auf seiner Website [1] zur Verfügung stellt, was lobenswert ist - viele Firmen ziehen Datenblätter von obsoleten Bauteilen zurück.

Eine typische Anwendungsschaltung ist in **Bild 1** zu sehen. Außer dem IC sind für den Antrieb eines Schrittmotors nur ein paar Dioden erforderlich. Die Richtungssteuerung wird hier mit einem Schalter realisiert, aber das könnte genauso gut ein Signal von einem Mikrocontroller oder einer anderen Logikschaltung sein.

Warum also hat Allegro einen so großartigen kleinen Schrittmotortreiber eingestellt? Wahrscheinlich, weil es sich um ein älteres bipolares IC handelt, nicht um einen MOS-Baustein, und weil er keine der Funktionen zur Senkung des Stromverbrauchs hat, die Hersteller heutzutage benötigen, um ihr System elektrisch effizient zu machen. Der Abkündigungshinweis auf dem Datenblatt empfiehlt die ICs A3967 und A3977 von Allegro, die überhaupt nicht dasselbe sind, da diese für die Ansteuerung von bipolaren Schrittmotoren (die heutzutage tatsächlich häufiger anzutreffen sind) mit H-Brücken-Treibern ausgelegt sind. Sie haben auch viel mehr Schnickschnack wie Strommessung, PWM-Ansteuerung und Halb-, Viertel- und Achtelschrittmodi zu bieten. Insbesondere für Bastler wie mich, die immer noch mit Lochraster- oder -streifenplatinen arbeiten, ist ein weiteres wichtiges Versäumnis, dass diese beiden ICs nur in 24- und 28-poligen SOIC-Gehäusen erhältlich sind.

Es gibt mehrere andere Schrittmotor-Treiberlösungen, aber sie alle bleiben in der einen oder anderen Hinsicht hinter dem UCN5804 zurück. Die bekannte L297/298-Kombination ist ziemlich vielseitig, aber es ist halt eine Zwei-IC-Lösung. Andere Bauteile, die sich als Schrittmotortreiber ausgeben, sind lediglich halbe H-Brücken und verfügen nicht über die schönen und einfachen Schritt- und Richtungseingänge. So bleibt es dann Ihnen, dem mutigen Entwickler, überlassen, die Steuerwellenformen mit einem anderen IC oder Ihrem Mikrocontroller zu erzeugen.

Es ist sehr schade, dass Allegro dieses IC nicht mehr anbietet, denn es ist ideal für Bastler (ja, ich weiß, Bastler sind ein kleiner Markt). In **Bild 2** sehen Sie meine Breadboard-Schaltung, die den UCN5804 zusammen mit einem 555-Timer und einigen Schaltern verwendet. Glücklicherweise habe ich noch ein paar davon, die ich aus ausrangierten Bandlaufwerken gerettet habe. Und wenn man sich umschaut, kann man echte wie gefälschte Exemplare im Internet für weniger als 10 € pro Stück erwerben.

(220530-02)RG

Haben Sie Fragen oder Kommentare?

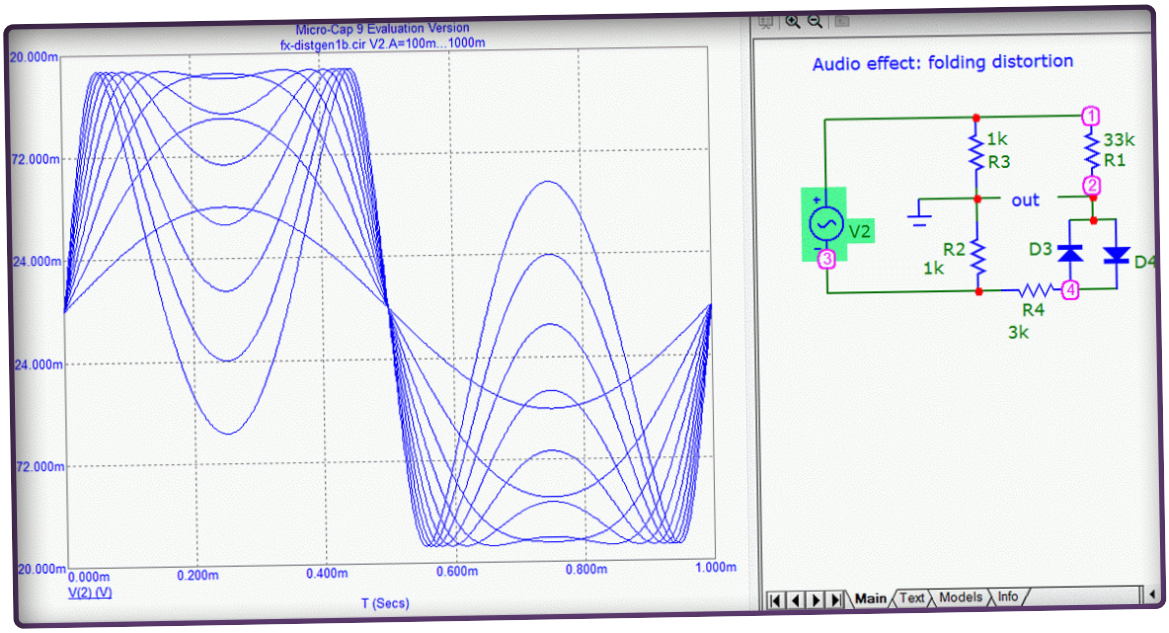
Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Schicken Sie eine E-Mail an Elektor unter redaktion@elektor.de.

WEBLINK

[1] Allegro Microsystems - UCN5804-Datenblatt: https://elektor.link/ucn5804

Schaltungssimulation mit Micro-Cap

Erste Schritte in eine komplizierte Welt



Von Raymond Schouten (Niederlande)

Ein Schaltungssimulator gehört in den Werkzeugkasten eines jeden Elektronikers, genau wie ein Netzteil, ein Multimeter und ein Oszilloskop. Und wie bei jedem anderen Werkzeug müssen Sie einige Zeit damit verbringen, den Umgang mit einem Schaltungssimulator zu erlernen, um ihn optimal nutzen zu können. In diesem Artikel zeigen wir Ihnen, wie Sie mit dem Freeware-Simulator Micro-Cap arbeiten können.

Mit einem Schaltungssimulator können Sie beim Testen von Ideen oder bei der Definition von Bauteilen nicht nur Zeit sparen, er ist auch ein gutes Lern- und Fehlerbehebungswerkzeug, das Ihnen zeigt, was in einer Schaltung vor sich geht.

Schaltungssimulationssoftware wie LTspice von Analog Devices, Tina von Texas Instruments und Micro-Cap verwenden den auf Texteingabe basierenden SPICE-Code, der an der University of California in Berkeley im Jahr 1973 entwickelt wurde. Moderne Software ergänzt diesen Code hauptsächlich durch eine grafische Benutzeroberfläche. Ein wesentlicher Teil dieses Artikels gilt daher für alle drei Simulatoren, insbesondere wenn es um Tipps zur Vermeidung von Simulationsfehlern geht.

Warum simulieren, wie simunieren?

Bei dem hier vorgestellten Simulator Micro-Cap [1] handelt es sich um einen Schaltungssimulator von professioneller Qualität (und Kosten: 4.500 $), der nach Einstellung der Produktentwicklung als Freeware veröffentlicht wurde. Im Vergleich zu anderen kostenlosen Simulatoren bietet er recht fortschrittliche Funktionen. Der Nachteil ist, dass er irgendwann veraltet sein könnte, weil nicht mehr daran gearbeitet wird. Auf jeden Fall können Sie diese kostenlose Software unter dem Link [1] finden.

Mit Micro-Cap können Sie Schaltungen zeichnen und die Signale darin untersuchen, indem Sie an den Parametern „drehen“. Die Software verfügt über eine große Bibliothek von Bauteilmodellen, die sogar Vakuumröhren enthält. Entwurfswerkzeuge für aktive und passive Filter sind integriert, und bei Audioanwendungen ist es sehr vorteilhaft, dass Sie sich den von Ihrer simulierten Schaltung erzeugten Sound auch anhören können.

Eine Schaltungssimulation ist wie der Bau einer Schaltung auf einem Steckbrett. Sie platzieren die Komponenten, verbinden sie zu einer Schaltung, fügen Versorgungsspannungen hinzu und schließen (optional) eine Signalquelle an. Anschließend messen Sie die

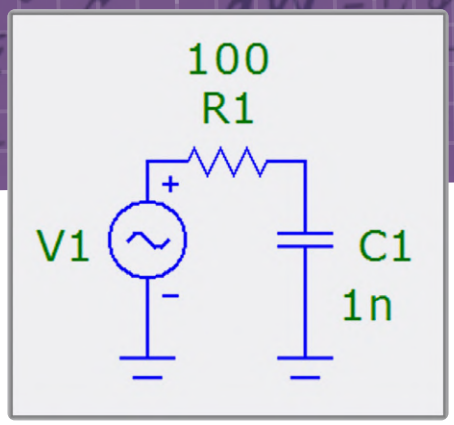


Bild 1. Meine erste Schaltung, ein einfaches RC-Tiefpassfilter.

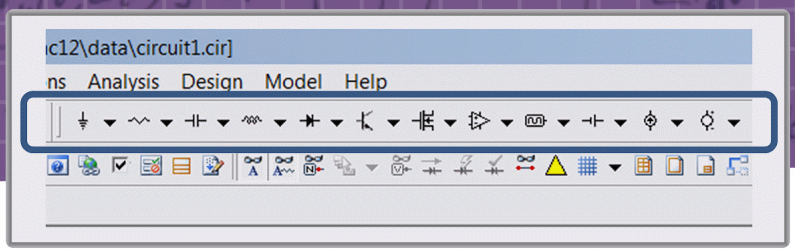


Bild 2. Micro-Cap besitzt eine Bauteil-Leiste.

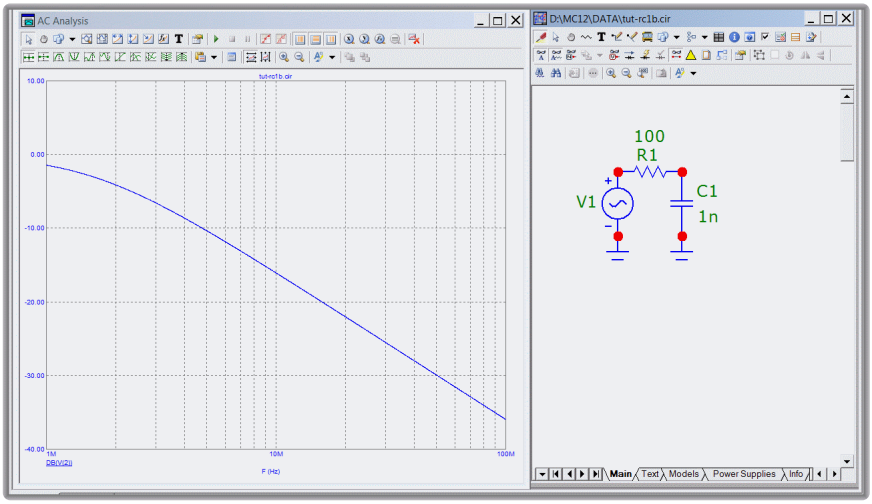


Bild 3. Durch Anklicken des RC-Knotens wird die Übertragungskennlinie des Filters angezeigt.

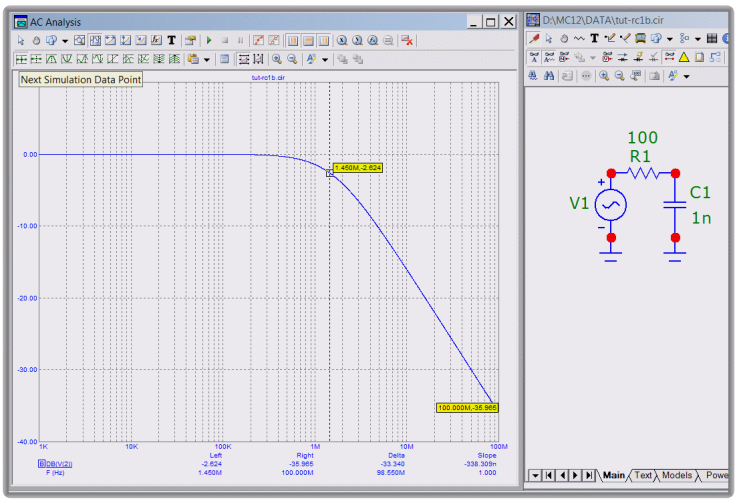


Bild 4. Stellen Sie die Grenzen des Diagramms so ein, dass Sie das sehen, was Sie sehen wollen.

Signale entweder mit einem Oszilloskop (im Zeitbereich) oder einem Spektrumanalysator (im Frequenzbereich). Weitere Einzelheiten dazu finden Sie im Kasten **Analysetypen**.

Zeichnen der ersten Schaltung

Nach dem Start von Micro-Cap beginnen wir mit der Erstellung eines einfachen RC-Tiefpassfilters und simulieren dessen Frequenzgang, wie in **Bild 1** dargestellt. Es gibt mehrere Möglichkeiten und Abkürzungen, um Bauteile zu einer Schaltung hinzuzufügen (siehe Kasten). Wir verwenden zunächst die Bauteilleiste am oberen Rand des Bildschirms (**Bild 2**). Klicken Sie kurz mit der linken Maustaste auf das Widerstandssymbol und bewegen Sie den Mauszeiger in Ihr leeres Arbeitsblatt (optional können Sie auf den kleinen Pfeil oder das Dreieck rechts neben der Symbolschaltfläche klicken, um eine andere Ausrichtung zu wählen). Es erscheint ein (amerikanisches) Widerstandssymbol. Klicken Sie erneut mit der linken Maustaste, um den Widerstand an der gewünschten Stelle im Arbeitsblatt zu platzieren. Geben Sie oben in dem Fenster, das sich nun öffnet, in das Feld *Value* den Wert *100* ein. Schließen Sie das Fenster mit der Schaltfläche *OK* oder durch Drücken der Eingabetaste.

Wiederholen Sie diesen Vorgang für einen Kondensator; setzen Sie seinen Wert auf *1n*. Wenn Sie den Kondensator so platzieren, dass einer seiner Drähte einen der Drähte des Widerstands berührt, werden die beiden Bauteile automatisch miteinander verbunden. Als Nächstes fügen wir einen Sinusgenerator oder eine Sinusquelle hinzu, indem wir vom Hauptmenü aus auf *Component* → *Analog Primitives* → *Waveform Sources* → *Sine Source* klicken. Wählen Sie *GENERAL* aus der Liste auf der rechten Seite des Fensters mit den Eigenschaften der Quelle aus, und klicken Sie dann auf *OK*.

Schließen Sie die Schaltung ab, indem Sie dem Kondensator und der Signalquelle die Massesymbole aus der Bauteilleiste hinzufügen.

Wenn Sie die Bauteile weit voneinander entfernt platziert haben, müssen Sie sie miteinander verdrahten. Dies können Sie über den Button *Wire mode* oder durch Drücken von Strg-W tun. Um eine Verbindung zu zeichnen, klicken Sie auf den Startpunkt des Drahtes und ziehen Sie den Zeiger bei gedrückter linker Maustaste zum Zielpunkt des Drahtes.

Simulieren!

Jetzt können wir schon eine erste Simulation durchführen. Wählen Sie im Menü *Analysis* die Option *Probe AC...*, um in den Analysemodus zu gelangen, in dem Sie „messen" können. In diesem Modus können Sie den Schaltungsaufbau nicht mehr ändern, sondern lediglich die Bauteilwerte. Die roten Punkte im Schaltbild sind die Knotenpunkte. Wenn Sie auf einen Punkt klicken, wird im Fenster *AC Analysis* ein Diagramm ähnlich dem in **Bild 3** erstellt.

Das Diagramm verwendet Standard-Frequenzeinstellungen, aber wir wollen, dass es bei einer niedrigeren Frequenz beginnt. Wählen Sie dazu im Menü *Probe* die Option *Limits...* aus. Ändern Sie den Frequenzbereich auf *100Meg, 1k* (die Angabe nennt zunächst den oberen, dann den unteren Wert) und klicken Sie auf *Close*. Das Diagramm ändert sich zu dem in **Bild 4** gezeigten. Wir können nun sowohl den Durchlassbereich des Filters (DC bis 1 MHz) als auch den Sperrbereich (alles über 1 MHz) sehen.

Sie können wiederum die Datenpunkte im Diagramm mit einem Cursor ablesen. Klicken Sie auf den Knopf *Next Simulation Data Point* oben links über der Grafik, wird ein Cursor eingeblendet, den Sie verschieben können.

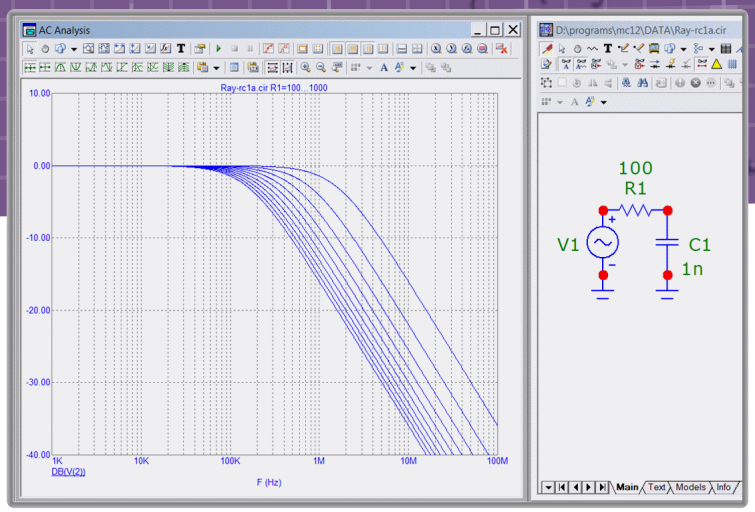


Bild 5. Die schrittweise Veränderung eines Parameters (hier der Wert von R1) zeigt seinen Einfluss auf die Übertragungsfunktion der Schaltung.

Simulation vs. Realität

Dämpft den Diagrammen in Bild 3 und Bild 4 zufolge unser Filter alles über 1 MHz? Das würde bedeuten, dass dieses Filter alle Frequenzen bis ins Unendliche unterdrückt. Wir wissen, dass dies in der Realität nicht der Fall ist. Warum ist es dann in der Simulation so? Der Grund ist, dass ein Simulator ideale Komponenten verwendet. Wenn Sie sich dem realen Verhalten einer Schaltung annähern wollen, müssen Sie die Unzulänglichkeiten sowohl Ihrer Bauteile als auch der realen Welt hinzufügen, indem Sie parasitäre Streuelemente hinzufügen. Weitere Informationen dazu, wie Sie Ihre Simulation einer realen Schaltung annähern können, finden Sie im Kasten **Beschränkungen des Komponentenmodells**.

Ändern eines Bauteilwerts im Analysemodus

Auch im Analysemodus können Sie Komponentenwerte ändern. Dazu müssen Sie zu *Select Mode* wechseln, indem Sie auf den Knopf mit dem Mauspfeilsymbol oben links im Schaltungsfenster klicken (oder Strg-E drücken). Nun können Sie auf den Wert eines Bauteils oder das Bauteil selbst doppelklicken und es auf dieselbe Weise wie zuvor beschrieben ändern. Die Simulationsergebnisse werden automatisch aktualisiert. Wenn Sie an einer anderen Stelle des Schaltkreises eine Probe durchführen möchten, müssen Sie zum Probe-Modus zurückkehren, indem Sie auf den *Probe*-Knopf klicken.

Wobbeln der Parameter

Eine leistungsstarke Funktion eines Simulators besteht darin, dass er den Wert eines oder mehrerer Parameter schrittweise ändern und die Ergebnisse als Kurvenschar anzeigen kann. Als Beispiel soll der Wert des Widerstands R1 wie bei einem Potentiometer verändert werden, um zu sehen, wie groß die Abstimmung bei einem bestimmten Potentiometerbereich ist. Wir werden die Grenzfrequenz des Filters ändern, indem wir R1 von 100 Ω bis 1000 Ω in Schritten von 100 Ω wobbeln. Wählen Sie im Menü *Probe* → *Stepping...* und im sich öffnenden Fenster unter *Step What* den Widerstand R1. In den Feldern *From, To* und *Step Value* geben Sie die entsprechenden Werte ein und vergessen Sie nicht, *Step It* auf *Yes* zu setzen. Klicken Sie auf *OK* und dann auf den *Run*-Knopf (mit dem grünen Pfeil oder Dreieck) oder F2 im Fenster *AC-Analysis*, um die Kurven in Abhängigkeit vom Widerstandswert anzuzeigen (**Bild 5**).

In diesem Beispiel waren die Schritte gleich groß (linear), aber Sie können auch exponentiell voranschreiten. Es ist möglich, mehrere Bauteilwerte gleichzeitig oder in verschachtelten Sweeps zu ändern. Sie können auch Bauteil-Modellwerte schrittweise verändern, zum Beispiel die Verstärkung eines Transistors.

Transientenanalyse

Um nichtlineares Verhalten zu betrachten, müssen Sie den richtigen Analysemodus wählen (siehe Kasten **Analyse-Typen**). Um ein nichtlineares Verhalten zu erhalten, erstellen wir einen einfachen Audioverzerrungseffekt, einen Dioden-Clipper, wie in **Bild 6** gezeigt. Mit der Transientenanalyse können wir dann die Verzerrung wie mit einem Oszilloskop beobachten. Stellen Sie die Frequenz der Sinusquelle V1 auf 1 kHz ein (im Auswahlfeld F in der rechten unteren Ecke des Eigenschaftsfensters) und wählen Sie im Menü *Analysis* → *Probe* → *Transient...* aus. Wählen Sie im Menü *Probe* → *Limits...* und setzen Sie die *Maximum Run Time* auf *2m* und den *Maximum Time Step* auf *2u*. Es hat sich bewährt, den Zeitschritt auf 1/1000 des Laufzeitbereichs einzustellen. Prüfen Sie nun die Schaltung, sowohl am Eingang (mit der Sinuswelle) als auch am Ausgang (mit der geclippten Welle). Sie erhalten das Diagramm in **Bild 7**.

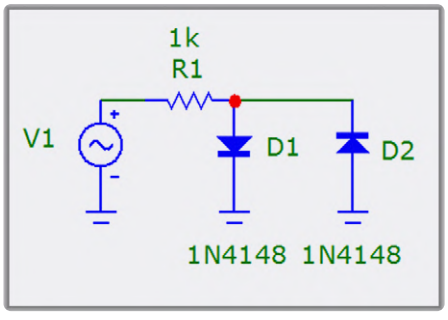


Bild 6. Diese einfache Schaltung zeigt ein nichtlineares Verhalten.

Wie viel ist M?

In der SPICE-Sprache steht M für Milli (10^{-3}). Wenn Sie also den Wert eines Widerstands auf 1 Mohm einstellen, entspricht dies 0,001 Ω! Sie sollten MEG für Mega oder 10^6 verwenden. Gewöhnungsbedürftig!

Analyse-Typen

Nachdem Sie die Schaltung gezeichnet haben, möchten Sie Messungen durchführen. Dazu müssen Sie einen Analysemodus auswählen. In der SPICE-Sprache wird die Messung mit einem Oszilloskop als *transient analysis* bezeichnet, während die Messung mit einem Spektrumanalysator *AC analysis* genannt wird. Die Verbindungspunkte zwischen den Bauteilen sind die *nodes*. Im Analysemodus können Sie die Spannungen an diesen Knoten und die Ströme, die durch sie fließen, untersuchen.

Um nichtlineares Verhalten (zum Beispiel Verzerrungen, Übersteuerungen) sichtbar zu machen, müssen Sie die Transientenanalyse verwenden. Die AC-Analyse geht immer von einem linearen Verhalten aus und liefert den Kleinsignalfrequenzgang von Schaltungen wie Filtern und Verstärkern. Bei beiden Analysetypen können Sie auch die Gleichspannungen und -ströme sehen. Es sind noch weitere Analysearten wie Rausch- und Impedanzanalyse möglich.

Wenn die Schaltung korrekt gezeichnet wurde (und beispielsweise keine schwebenden Knoten mehr vorliegen), wird ein Analysetyp ausgewählt. Dann wird eine Berechnung über einen benutzerdefinierten Zeitraum oder Frequenzbereich mit einer gewählten Auflösung durchgeführt. Die Einstellung dieser Auflösung ist ein Kompromiss zwischen der Glättung der Kurve und der Berechnungszeit. Sobald die Berechnung abgeschlossen ist, können Sie mit der Untersuchung der Schaltung beginnen.

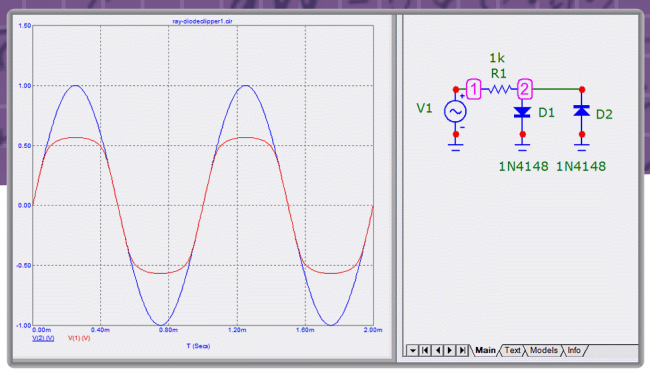


Bild 7. Eine Transientenanalyse der Schaltung aus Bild 6 ergibt diese Signale.

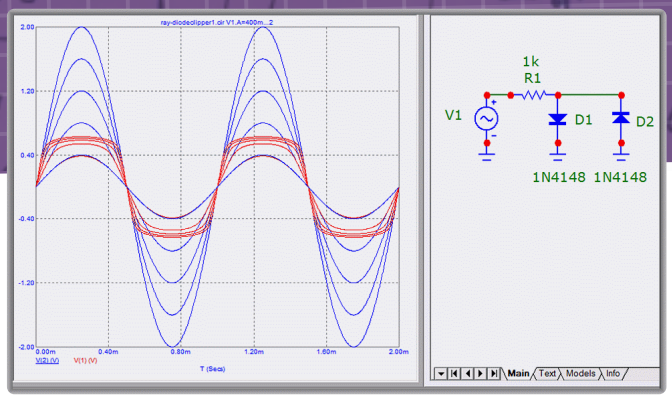


Bild 8. Die Ausgangssättigung nimmt allmählich zu, wenn wir die Amplitude des Eingangssignals schrittweise erhöhen.

Wobbeln der Amplitude

Sie können einen Bauteilwert oder einen anderen Modellparameter schrittweise verändern, aber Sie können auch die Amplitude der Sinusquelle variieren. Wählen Sie wiederum aus dem Menü *Probe → Stepping...* und unter *Step What* den Sinusgenerator V1 und rechts daneben die V1-Amplitude *A*. Darunter geben Sie von *400mV* bis *2V* in Schritten von *400mV* an. Vergessen Sie nicht, *Step It* auf *Yes* zu setzen, bevor Sie auf *OK* und im Fenster der *Transient Analysis* auf *Run* klicken. **Bild 8** zeigt die Ergebnisse.

Da es sich bei der Schaltung in Bild 6 um einen Audioeffekt handelt, wäre es nützlich zu hören, wie er sich anhört. Micro-Cap bietet die Möglichkeit, die Wellenform auf Ihrer Soundkarte abzuspielen oder sie als WAV- oder CSV-Datei zu speichern.

Die Dauer des Tons wird durch die *Probe Limits* festgelegt. Ändern Sie die *Maximum Run Time* auf *500m* und die *Maximum Time Step* auf *50u*, um eine halbe Sekunde Ton zu erhalten. Doppelklicken Sie anschließend auf das Messkurvenfenster, um dessen *Properties* zu öffnen, und wählen Sie den Tab *Save Curves*. Wählen Sie in der Liste *Curves* die Welle aus, die Sie hören möchten. Wenn Sie *stepping* aktiviert haben (zum Beispiel der Amplitude des Eingangssignals), können Sie jeden Schritt anhören, indem Sie ihn im Feld *What To Save* die Daten auswählen und dann auf *Play* drücken.

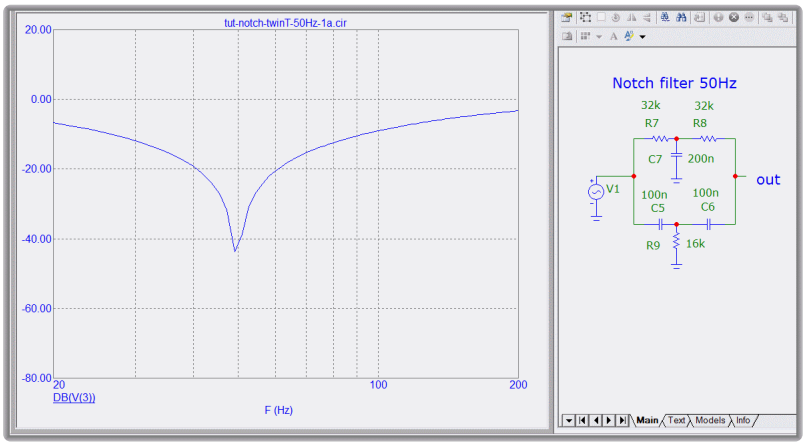


Bild 9. Die Auflösung der Simulation ist wichtig. Hier wurde Probe Limits *auf 100 Punkte eingestellt.*

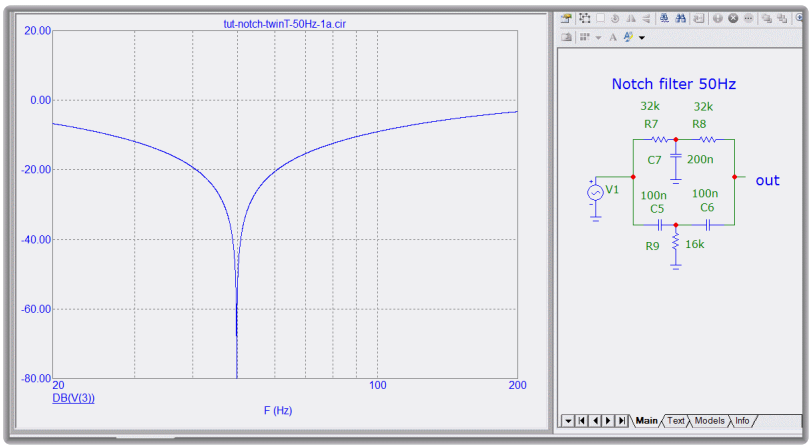


Bild 10. Dieselbe Simulation wie in Bild 9, aber jetzt mit Probe Limits *auf 10.000 Punkte eingestellt.*

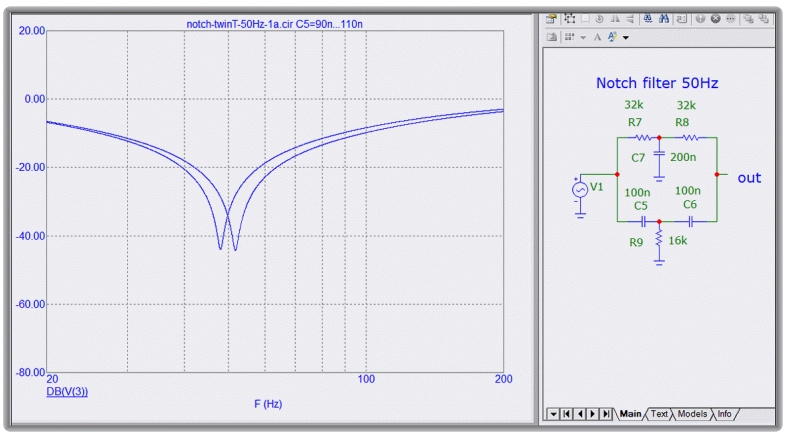


Bild 11. Dieselbe Schaltung wie in Bild 9, aber dieses Mal ist C5 nicht ganz mit C6 identisch.

Wo sind die Bauteile zu finden?

Alle Komponenten können über den Tab *Components* und das erweiterbare Bibliothekslistenfeld (das „Panel“) gefunden werden, aber es gibt auch einige Verknüpfungen. Es gibt die Komponentenleiste am oberen Rand des Bildschirms, aber noch schneller geht es mit Tasten wie *R* für einen Widerstand und *C* für einen Kondensator. Außerdem speichert das Programm die zuletzt verwendeten Bauteile im Startfenster des Tabs *Components*.

Auflösung der Simulation

Eine falsche Einstellung der Auflösung kann die Simulationsergebnisse beeinträchtigen oder sogar Details entfernen. Ein Notch-Filter hat zum Beispiel einen starken Einbruch bei der Frequenz, bei der es arbeitet. Eine unzureichende Frequenzauflösung lässt die Kerbe weniger tief erscheinen als sie ist. **Bild 9** und **Bild 10** zeigen die Ergebnisse desselben Doppel-T-Sperrfilters, das zweimal simuliert wurde, jedoch mit unterschiedlichen Einstellungen der Auflösung.

Ideale Bauteile

Dieses Filter hat mit den angegebenen Bauteilwerten eine unendlich tiefe Kerbe bei 50 Hz. Bevor Sie jedoch auf den realen Bau dieses „perfekten" Filters stürzen, sollten Sie sich ein paar Gedanken über die Bauteilwerte machen. In einem Simulator sind die beiden 100-nF-Kondensatoren identisch und genau 100 nF, und Sie erhalten so ein unendlich tiefes 50-Hz-Sperrfilter wie hier gezeigt. Eine Änderung des Wertes von beispielsweise C5 um plus oder minus 10 % hat einen großen Einfluss (**Bild 11**). Die Kerbe ist jetzt nur noch 40 dB tief und zudem verstimmt sich das Filter, was zu einer Unterdrückung von nur 35 dB bei 50 Hz führt. So zeigt Ihnen der Simulator Ihnen schon vor dem Bau einer Schaltung, wie sich die Toleranzen der realen Bauteile in der Schaltung auswirken. Für noch realistischere Simulationen müssen Sie auch andere parasitäre Parameter wie Leitungs- und Drahtlängen berücksichtigen (siehe Kasten Beschränkungen des Komponentenmodells).

Das ist noch nicht alles!

Dieser Artikel hat nur an der Oberfläche eines komplizierten Themas gekratzt. Wie zu Beginn dieses Artikels erwähnt, sollte ein Schaltungssimulator genauso zu Ihrem Werkzeugkasten gehören wie ein Multimeter und ein Oszilloskop. Bedenken Sie jedoch immer, dass ein Schaltungssimulator eine vereinfachte Version der Realität darstellt. Ob mit Micro-Cap oder einem anderen Simulator lassen sich nur korrekte Ergebnisse erzielen, wenn man ein gewisses Verständnis für die praktischen Aspekte des Aufbaus einer Schaltung mitbringt. Man muss wissen, wie man den Simulator einrichtet und wo seine Grenzen liegen. Es ist immer noch eine gute Idee, die Simulation mit dem Bau eines echten Prototyps zu kombinieren. Werfen Sie Ihr Breadboard mit seinen schlechten Kontakten (wie modelliert man diese?) also noch nicht weg!

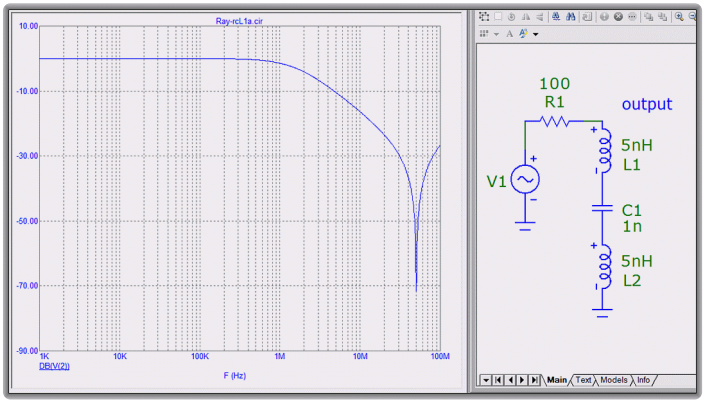
Haben Sie Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zu diesem Artikel?
Schicken Sie eine E-Mail an den Autor unter rs.elc.projects@gmail.com oder an Elektor unter redaktion@elektor.de.

(220336-02)RG

Beschränkungen der Komponentenmodelle

Eine Simulation ist nur so gut wie die Modelle, die sie verwendet. Alle Komponenten in unserem Filter sind ideal, was in der Praxis natürlich nicht der Fall ist. Bei unserem einfachen Tiefpassfilter ist die größte Einschränkung die Serieninduktivität der Kondensator-Anschlussdrähte. Jeder Millimeter Draht fügt etwa 1 nH an Induktivität hinzu, was die Impedanz des Kondensators oberhalb einer bestimmten Frequenz erhöht und die Filterwirkung verringert. Nehmen wir an, Ihr Kondensator hat 5 mm lange Anschlüsse, dann simulieren Sie dies, indem Sie zwei 5-nH-Induktivitäten in Reihe mit den Kondensatoranschlüssen hinzufügen (was dem Hinzufügen einer 10-nH-Drossel in Reihe entspricht).



Wir stellen fest, dass das Filter bei 50 MHz einen starken Einbruch aufweist und dann ab 5 MHz zu „lecken" beginnt, was heißt, dass die Amplitude wieder ansteigt. Dies wird durch den LC-Resonanzkreis verursacht. Wenn Sie HF-Störungen bis zu 1 GHz filtern wollen, ist dies ein Problem, aber in besonderen Fällen können Sie diesen Effekt nutzen. Wenn Sie zum Beispiel einen Abtasttakt bei 50 MHz filtern wollen, ist dies eine Verbesserung, aber Sie müssen den Induktivitätswert anpassen. Dies ist ein Beispiel dafür, wie ein Simulator Ihnen sogar helfen kann, eine Schaltung richtig aufzubauen.
Frage: Wenn Drähte Induktivität hinzufügen, ergibt sich dann bei Verwendung von SMD-Kondensatoren ein Filter ohne Induktivität? Nein, aber es kommt dem schon näher. Der Kondensator hat immer noch eine endliche Länge (2 mm sind 2 nH), und die Verbindung des Lötauges auf der Platine mit Masse fügt ebenfalls Induktivität hinzu. Durchkontaktierungen können mit 0,3 nH bis 1 nH zu Buche schlagen, je nachdem, wie Sie sie definieren und platzieren.

WEBLINK

[1] Webseite Micro-Cap: https://www.spectrum-soft.com/download/download.shtm

Der PAUL Award 2022

Junge Techniktalente und ihre kreativen Lösungen

Von Florian Schäffer, im Auftrag von Elektor

Die Förderung junger Elektroniktalente ist ein wichtiges Anliegen, so auch für den deutschen Fachverband Elektronikdesign und -fertigung (FED). Im Rahmen der diesjährigen FED-Konferenz in Potsdam wurde auch 2022 der mit 6.000 Euro dotierte PAUL Award verliehen. Jeder im Alter von 15 Jahren bis 25 Jahren konnte eine kreative Elektroniklösung als Beitrag einreichen; am Ende wurden drei Siegerteams gekürt, die wir hier vorstellen.

Nominierte und Gewinner des PAUL Award 2022 beim Gruppenfoto in Potsdam, (Copyright FED e.V.).

Steckdosenleiste mit Brandschutz

Das Siegerteam *Fisego* entwickelte eine Steckdosenleiste mit integriertem Temperaturschutz auf Basis eines Mikrocontrollers, um so Brände durch Überlast zu verhindern. Die Jury würdigte die Idee, Umsetzung und Dokumentation mit dem ersten Preis und 3.000 Euro Prämie. Sophia Reiter (22 Jahre und Studentin der Elektrotechnik) und Fabian Goedert (der 25-jährige studiert Bauingenieurwesen) konnten sich zudem über den von Elektor unabhängig ausgelobten Medienpreis freuen, der ihnen bei der Feierlichkeit in Potsdam überreicht wurde.

Das Jurymitglied Jürgen Deutschmann übergibt den ersten Preis für die brandsichere Mehrzwecksteckdose an Sophia Reiter und Johannes Steube (Copyright FED e.V.).

Im Prototyp der Steckdose aus dem 3D-Drucker überwacht eine Elektronik mit ATmega328, ACS712-Hallsensor und DS18B20-Temperatursensor die aktuelle Leistungsaufnahme und die Temperatur im Gehäuse. In einer dreistufigen Warnphase werden dem Anwender die Parameter auf einem OLED-Display angezeigt. Beim Überschreiten eines Grenzwertes erfolgt zuerst eine Warnung mit Handlungsanweisungen, bevor beim Erreichen des nächsten Grenzwerts die Steckdose automatisch abgeschaltet wird. Im Gegensatz zu Steckdosen mit Wärmesicherung oder Sicherungsautomat hat der Anwender hier die Möglichkeit, vor der totalen Abschaltung selbst einzugreifen und beispielsweise eine einzelne Last auszuschalten.

Selbstgestecktes Ziel der beiden Studenten ist nach Aussage von Sophia Reiter eine Mehrfachsteckdose für das Internet of Things (IoT) mit integriertem Brandschutzsystem, welches nicht nur im privaten Bereich eingesetzt werden soll, sondern auch die Industrie überzeugt und jede beliebige Art von Maschine oder Anlage überwachen kann. Eine weiterer Schritt zu diesem Ziel wurde durch die jüngst gegründete GmbH bereits durch die Prototypenfertigung der Steckdose durch Strangpressen begangen.

FED erkennt Nachwuchsprobleme und startet Ideen-Förderung

Der PAUL Award wurde im Jahr 2020 das erste Mal ausgetragen, musste dann aber bedingt durch Corona pausieren, so dass erst dieses Jahr am 28. September die zweite Vergabe stattfinden konnte. Wie Jürgen Braunsteiner, neu gewähltes Vorstandsmitglied des FED, in seiner Einleitungsrede vor etwa 100 Gästen betonte, steht die Branche vor einem Generationenwandel, bei dem vor allem kleinere und mittlere Unternehmen mit Nachwuchsproblemen kämpfen.

Die Förderung von jungen Menschen mit kreativen Ideen ist daher ein wichtiges Ziel: Sie sollen durch den PAUL Award motiviert werden, ihre Projekte zu entwickeln und eine Chance bekommen, diese zu präsentieren und Kontakte zur Industrie zu knüpfen. Als positives Beispiel verwies Braunsteiner auf die junge Australierin Cynthia Sin Nga Lam, die mit Ihrem Projekt *H2prO* Schmutzwasser per Photokatalyse reinigen und dabei auch noch Strom erzeugen will. Ihr Gerät mit dem Potential, die Welt zu verändern, sorgte im Jahr 2014 für viel mediales Aufsehen. Das Projekt zeigte, dass neue Ideen überall zu finden sind, wenn man sich nur traut, an der Realisierung selbst mit einfachen Mitteln zu arbeiten.

Energy-Harvesting mit Peltier-Element

Das kleine Teilnehmerfeld aus fünf Kandidaten musste sich zusätzlich der Herausforderung des Energy-Harvesting stellen:

Team Fisego mit dem Sonderpreis von Elektor (Copyright FED e.V.).

Der zweite Platz ging an Manuel Wilke für sein Batteriemanagementsystem (Copyright FED e.V.).

Junge Maker gesucht!

Beim PAUL Award können junge Maker mit einer innovativen Elektronik-Idee bis zu 3.000 Euro gewinnen. Jetzt Projektidee einreichen (Skizze genügt) und durchstarten!

Alle Infos: www.paul-award.de/mitmachen

Die Projekte sollten ohne externe Energiequelle auskommen. Eine Aufgabe, die auch vom Team *Drink Safe* gelöst wurde. Sie entwickelten einen Tassenuntersetzer, der zwar stark an ähnliche Gadgets erinnerte, die Tassen per USB warm halten, dem aber eine ganz andere Aufgabe zukommt: Der Untersatz ermittelt die Temperatur der Tasse und signalisiert über eine LED, wann das Getränk soweit abgekühlt ist, dass ein Genuss möglich ist, ohne sich dabei zu verbrühen . Angesichts von Warnhinweisen wie „hot drink" auf Einwegbechern keine allzu abwegige Idee.

Zur Funktion wird der Effekt eines Peltier-Elements ausgenutzt, das bei Temperaturdifferenz zwischen den beiden Seiten einen Stromfluss erzeugt. Weil die dabei erzeugte Spannung sehr gering ist, bedarf es einer integrierten Step-Up-Regelung, um die LED zu steuern. Dadurch wird keine zusätzlichen Stromquelle wie Batterie oder Akku benötigt. Die Idee und Umsetzung brachte dem Team den dritten Platz ein.

Wetterstation und ein Batteriemanagementsystem

Mit ihrer Idee, in einem Regenrinnen-Fallrohr ein Wasserrad zu installieren und damit eine autarke IoT-Wetterstation zu betreiben, um ohne weitere Spannungsversorgung auszukommen, konnte ein weiteres Team die Jury leider nicht überzeugen, was möglicherweise auch daran lag, dass das Projekt noch mit Kinderkrankheiten kämpfte. Ob die beiden Studenten sich davon nicht abhalten lassen und weiter tüfteln oder einer neuen Idee nachgehen, konnten sie im Anschluss an die Veranstaltung selbst noch nicht beantworten.

Der zweite Preis (2000 Euro) wurde an Manuel Wilke vergeben, der als Student der Berliner Hochschule für Technik (BHT) im fünften Semester das Batteriemanagementsystem (BMS) *nandomBMS* entwickelte. Der Entwurf basiert auf einem Cortex-M4 Mikrocontroller XC4000 und einem LTC6813 Load-Balancer mit BMS von Analog Devices und kann bis zu 18 in Reihe angeschlossene Zellen überwachen. Der Mikrocontroller kann mittels CAN mit der Außenwelt kommunizieren. Die Platine soll vor allem im hauseigenen E-Buggy genutzt werden und zeichnet sich durch eine galvanische Trennung zwischen der Hochvolt- und der Niedervoltseite aus.

2023 gibt es die nächste Runde des PAUL-Awards vom FED

Für das nächste Jahr wünschen sich die Veranstalter vor allem mehr Teilnehmende, was unter anderem durch mehr Öffentlichkeitsarbeit erreicht werden soll. Zudem gibt es keine technische Zusatzbedingung mehr. Lediglich die Kosten sollen sich im Taschengeldrahmen bewegen, wodurch sich der Fokus wohl eher auf Schülergruppen und die Maker-Bewegung richtet. Wer einzeln oder als Team teilnehmen will, kann ab sofort seine Bewerbung einreichen und dann bis zum 30.9.2023 daran arbeiten. Alle Informationen sind auf der Seite des PAUL-Award [1] zu finden.

(220599-02)RG

Die Gewinnerprojekte des PAUL Award wurden im Rahmen der FED-Konferenz ausgestellt: Ein neuer Prototyp der Mehrzwecksteckdose, die unbestückte Platine des Batteriemanagementsystems und Drink Safe.

Das für ihr Projekt Drink Safe ausgezeichnete Team BMK kam auf den dritten Platz (Copyright FED e.V.).

WEBLINKS

[1] PAUL Award: https://www.paul-award.de

Mein erstes Software-definiertes Radio

Gebaut in weniger als 15 Minuten

Von Clemens Valens (Elektor Labs)

Wussten Sie, dass Sie einen UKW-Radioempfänger bauen können, indem Sie einfach ein paar Blöcke und Linien auf eine virtuelle Leinwand zeichnen? Wenn Malen noch nicht Ihre Leidenschaft ist, sie wird es nach dem Lesen dieses Artikels sein!

Bild 1. Das in diesem Artikel verwendete SDR-Frontend ist HackRF One von Great Scott Gadgets.

Beim Software-Defined Radio, kurz SDR genannt, geht es um die Modulation und Demodulation von Funksignalen in der Software. Anstelle spezieller elektronischer Schaltungen wie Mischer oder Demodulatoren wird beim SDR Software verwendet. Natürlich ist etwas Hardware erforderlich, um analoge Funksignale in digitale umzuwandeln und umgekehrt, aber das ist alles, was Sie brauchen.
Der Einfachheit halber beschränken wir uns in diesem Artikel auf den Empfang von Funksignalen, aber das meiste, was hier beschrieben wird, gilt auch für den Sendebetrieb. SDR kann in beide Richtungen funktionieren.

SDR kann (fast) alles!

SDR bietet viele Vorteile gegenüber dem analogen Radio. Erstens beherrscht SDR jede erdenkliche Modulationstechnik und noch mehr. Das bedeutet, dass Sie mit SDR nicht nur AM- und FM-Radio hören können, sondern auch WLAN oder Bluetooth, digitales Fernsehen oder DAB+ dekodieren oder Amateurfunk betreiben und vieles mehr. SDR ist zu all dem in der Lage, weil es die Modulations- oder Demodulationstechnik in der Software implementiert, und Software ist leicht zu ändern. Da alles in Software implementiert ist, kann SDR sogar Dinge tun, die mit elektronischen Bauteilen allein sehr schwierig oder sogar unmöglich wären.
Ein zweiter Vorteil von SDR ist, dass man nur eine einzige Hardware benötigt, um all das zu tun. Man braucht keinen separaten UKW-Empfänger, keinen WLAN-Empfänger, keinen digitalen Fernsehempfänger und so weiter. SDR verwendet für alle Aufgaben dasselbe Front-End.
Dieses Front-End ist ein Analog-Digital-Wandler (ADC) mit einer Antenne, der analoge Funksignale in digitale Signale umwandelt, die von einem Computer verarbeitet werden können. Für die weitere Übertragung wird dann noch ein Digital-Analog-Wandler (DAC) benötigt. In Wirklichkeit ist diese Hardware natürlich ein bisschen mehr als nur ein ADC und ein DAC. Die Schaltung führt auch einige Mischungen durch, um das interessierende HF-Signal in den Bereich zu bringen, den sie abtasten kann (und umgekehrt beim Senden), aber die grundlegende Funktion bleibt dieselbe.
SDR-Frontends gibt es in vielen Varianten, und Sie müssen eines auswählen, das zu Ihrem Budget und Ihren Wünschen passt. Je teurer der Konverter ist, desto mehr kann man üblicherweise mit ihm machen. Hier verwenden wir den HackRF One von Great Scott Gadgets [1] (**Bild 1**). Es ist nicht das billigste SDR-Frontend, aber es unterstützt Empfang und Senden und funktioniert bis zu 6 GHz. Außerdem, und das ist nicht unwichtig, wird es von vielen SDR-Softwareplattformen wie GnuRadio [2], SDR Sharp (SDR#) [3] und SDR++ [4] unterstützt. Für das unten beschriebene Projekt dürften andere (billigere) SDR-Frontends wahrscheinlich genauso gut funktionieren, solange sie im UKW-Radio-Band (87,5...108 MHz) arbeiten können.

Offene Quellen, offene Hardware

Ein weiterer Vorteil von SDR ist die Tatsache, dass vieles davon Open-Source und offene Hardware ist, die von einer großen Gemeinschaft von Enthusiasten entwickelt wurde.

Sie entwirft und veröffentlicht Schaltpläne für Front-Ends und entwickelt Software-Tools und Algorithmen für SDR-Anwendungen. Auch HackRF One ist quelloffen, und die Designdateien sind auf GitHub [5] verfügbar.

Voraussetzungen

Wie bereits erwähnt, habe ich für dieses SDR-Projekt HackRF One als Front-End mit einer 88 cm langen Teleskopantenne verwendet. Wir werden GNU Radio für die Software verwenden, genauer gesagt *GNU Radio Companion* alias GRC als grafische Oberfläche von GNU Radio. GNU Radio ist für Linux, Windows und macOS verfügbar, so dass die Chancen gut stehen, dass es auch auf Ihrem Computer laufen kann. Ja, es läuft sogar auf einem Raspberry Pi.

Drag 'n Drop-Programmierung

GRC ist ein grafisches Programmierwerkzeug, mit dem Sie eine Radioanwendung erstellen können, ohne wirklich programmieren zu müssen. Sie ziehen einfach Blöcke auf eine Canvas (Leinwand) genannte Arbeitsfläche, die Sie mit virtuellen Drähten miteinander verbinden. Die Programmierung bleibt auf die Konfiguration der Blöcke beschränkt. GRC verfügt über eine große Bibliothek von Blöcken zur Erstellung aller Arten von Modulations- und Demodulationsschemata.

Wir beginnen mit einer leeren Leinwand. Sie ist nicht völlig leer, da sie bereits zwei Blöcke enthält: *Options* und *Variable*. Ein doppelter linker Mausklick auf einen Block öffnet diesen, so dass Sie seine Parameter ändern können. Im Block *Options* können Sie einige einfache Optionen definieren und einen Titel für die Leinwand festlegen (siehe **Bild 2**). Wir werden alles auf den Standardwerten belassen.

Globale Abtastrate

Der Block *Variable* definiert eine globale Abtastratenvariable namens `samp_rate`, die in anderen Blöcken verwendet wird. Streng genommen braucht man sie gar nicht, aber sie ist doch praktisch, und deshalb behalten wir sie bei. Für die Abtastung von FM-Signalen ist ihr voreingestellter Wert jedoch zu niedrig, weshalb wir ihn auf 10 Millionen erhöhen. Der Maximalwert beim HackRF One ist 20 Millionen, doch wenn man ihn zu hoch einstellt, kann es passieren, dass der Computer Schwierigkeiten hat, mitzuhalten.

Hinzufügen einer HF-Quelle

Jetzt brauchen wir ein HF-Eingabegerät, das in GRC *source* genannt wird. Es gibt keinen speziellen Block für eine HackRF-One-Quelle. Die Sektion *Soapy* enthält zwar eine Quelle namens HackRF, aber das hat bei mir nicht funktioniert. Die *osmocom*-Quelle dagegen schon. Das OsmoSDR-Frontend gibt es nicht mehr, aber der Block unterstützt offensichtlich auch andere SDR-Hardware, einschließlich HackRF One.

Ein Doppelklick auf den Block öffnet seine Eigenschaften, und wir sehen, dass die Abtastrate auf `samp_rate` eingestellt ist, den Namen der Variablen im GRC-Block.

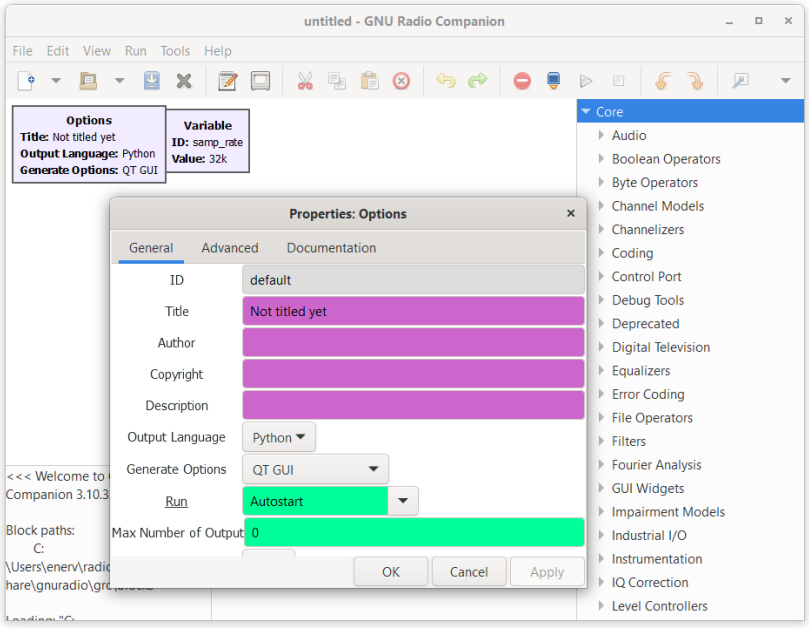
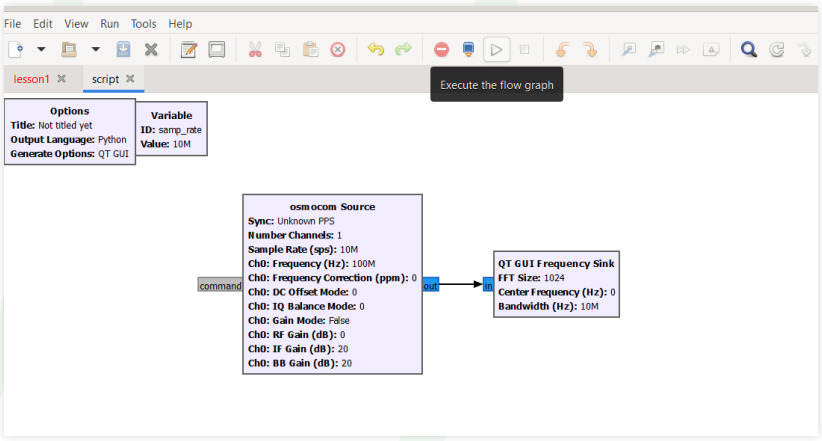


Bild 2. Ein neues Projekt in GNU Radio Companion (GRC) besteht aus zwei Blöcken. Ein Doppelklick auf einen Block öffnet dessen Eigenschaften.



▲

Bild 3. Klicken Sie auf den Button „Play", um Ihr erstes Flussdiagramm auszuführen.

Hier gibt es nur einen Parameter zu ändern, und zwar die HF-Verstärkung von Kanal 0, die für stärkere UKW-Funksignale zu hoch ist. Wir können ihn auf Null setzen.

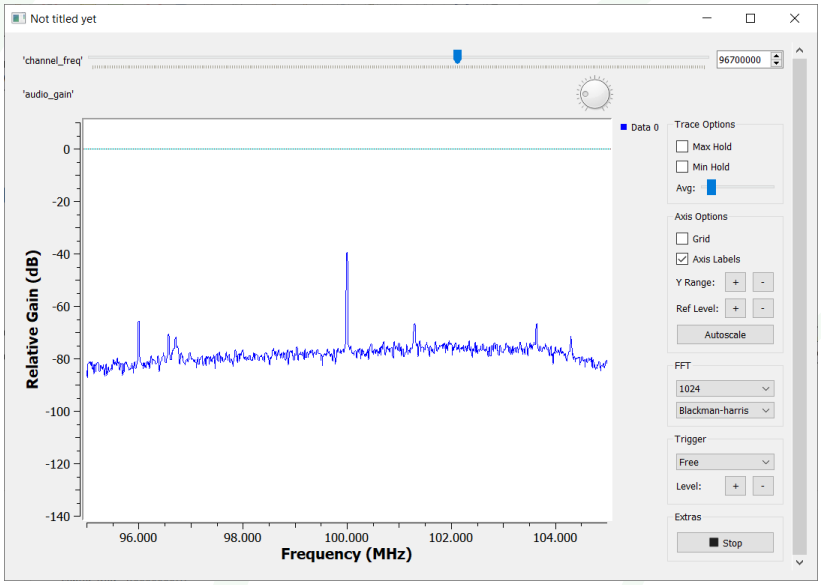
Spektrum-Analysator

Um schnell zu sehen, ob alles funktioniert, können wir der Leinwand einen Spektrum-Analysator-Block hinzufügen und ihn mit dem Ausgang des Blocks der HF-Quelle verbinden. Der Analysator wird *QT GUI Frequency Sink* genannt und befindet sich im Abschnitt *Instrumentation* von GRC. Die Art und Weise, wie diese Blöcke benannt sind, ist ein wenig verwirrend und macht es einem manchmal schwer, sie zu finden.

Um die Frequenzsenke mit der HF-Quelle zu verbinden, klicken Sie einfach auf ihren Eingang und dann auf den Ausgang der Quelle. Der umgekehrte Weg funktioniert auch, ebenso wie das Zeichnen mit gedrückter linker Maustaste. Beachten Sie, dass Sie nur Eingänge und Ausgänge der gleichen Farbe, in diesem Fall blau, verbinden können.

Ein erster Versuch

Sie können nun auf die Schaltfläche *Play* mit dem kleinen Dreieck klicken, um das Flussdiagramm auszuführen (**Bild 3**). Falls Sie dies noch nicht getan haben, müssen



▲

Bild 4. Die Aktivierung des Control-Panels auf der QT GUI Frequency Sink *ermöglicht den Zugriff auf verschiedene Steuerelemente.*

Sie Ihre Leinwand speichern, bevor Sie fortfahren können. Wenn alles in Ordnung ist, wird ein Frequenzdiagramm geöffnet. Wenn Sie an der Antenne herumfingern, sollten sich die Ergebnisse geringfügig ändern, aber das ist möglicherweise schwer zu erkennen. Um die Situation ein wenig zu verbessern, müssen wir die Frequenzdarstellung konfigurieren.

Klicken Sie dazu auf die Schaltfläche *Stop* mit dem kleinen Quadrat, um das Flussdiagramm zu beenden. Doppelklicken Sie auf den Block *Frequency sink*, um seine Eigenschaften zu öffnen und *averaging* auf *medium* einzustellen. Setzen Sie die *Center Frequency* auf denselben Wert wie die Frequenz von Kanal 0 der *Osmocom Source*, in unserem Fall also auf 100 MHz. Klicken Sie anschließend auf die Registerkarte *Config* und setzen Sie *Control Panel* auf *Yes*. Schließen Sie das Eigenschaftsfenster und klicken Sie erneut auf *Play*. Jetzt hat das Frequenzdiagramm ein Steuerfeld und Sie können das Kontrollkästchen *Max Hold* aktivieren (**Bild 4**). Wenn Sie nun an der Antenne herumfummeln, sollte dies besser sichtbar werden, und wenn dies der Fall ist, wissen Sie, dass Sie HF-Signale empfangen können.

Variablen sind praktisch

Eine schöne Sache ist, dass wir weiter nichts tun mussten, um das Flussdiagramm mit dem HackRF One zu verbinden. Es sind keine Treiber zu installieren, keine Ports auszuwählen, alles ist Plug and Play. Wir mussten nur die Mittenfrequenz der *Frequency Sink* auf den gleichen Wert wie die der HF-Quelle setzen, was bedeutet, dass sie an mindestens zwei Stellen benötigt wird. Wenn wir für die Mittenfrequenz eine globale Variable einsetzen, wie es GRC für die Abtastrate getan hat, können wir sie an nur einer Stelle festlegen.

Kopieren Sie dazu den Variablen-Block *sample rate*, öffnen Sie seine Eigenschaften und ändern Sie die ID in `center_freq`. Setzen Sie den Wert auf 100 MHz, die Frequenz von Kanal 0 in der *Osmocom Source* auf `center_freq` und setzen Sie dort den Wert auf `center_freq`. Die beiden Parameter sind nun miteinander verbunden. Wenn wir den Wert der Variablen `center_freq` ändern, ändert er sich auch in allen anderen Blöcken, die sie verwenden.

Mischen

Um mit SDR etwas zu erreichen, braucht man natürlich ein gewisses Wissen darüber, wie Radios funktionieren. Die Algorithmen in SDR tun dasselbe wie die elektronischen Schaltungen in einem analogen Radio. Um also einen UKW-Radiosender zu empfangen, müssen wir ihn abstimmen, demodulieren, filtern und hörbar machen.

Die Abstimmung kann mit einem Frequenzmultiplizierer, dem sogenannten Mischer erfolgen. Wenn dies mit Quadratur- oder komplexen Signalen geschieht, können die Frequenzen nach oben und unten verschoben werden. In GRC entsprechen die blauen Ein- und Ausgänge komplexen Gleitkommasignalen, die sich zum Mischen eignen.

Ziel ist es, die interessierende Frequenz in die Mitte des Frequenzdiagramms zu verschieben, aus dem sie leichter zu extrahieren ist. Für die Abstimmung auf einen Radiosender benötigen wir daher einen Mischer und ein weiteres Signal zur Steuerung der Frequenzverschiebung. Der *Multiply*-Block ist im Abschnitt *Math Operators* verfügbar. Für das Abstimmsignal können wir den Block *Signal Source* aus dem Bereich *Waveform Generators* verwenden.

Auf einen Sender abstimmen

UKW-Radiosender haben eine bekannte Frequenz, die wir als Kanalfrequenz bezeichnen. Da wir diese Frequenz leicht ändern können wollen, erstellen wir eine globale Variable mit dem Namen `channel_freq`. Wo ich wohne, gibt es einen Radiosender auf 96,7 MHz, also habe ich diesen Wert in `channel_freq` eingetragen.

Der Mischer muss die Kanalfrequenz auf die Mittenfrequenz verschieben, also muss die Frequenz der

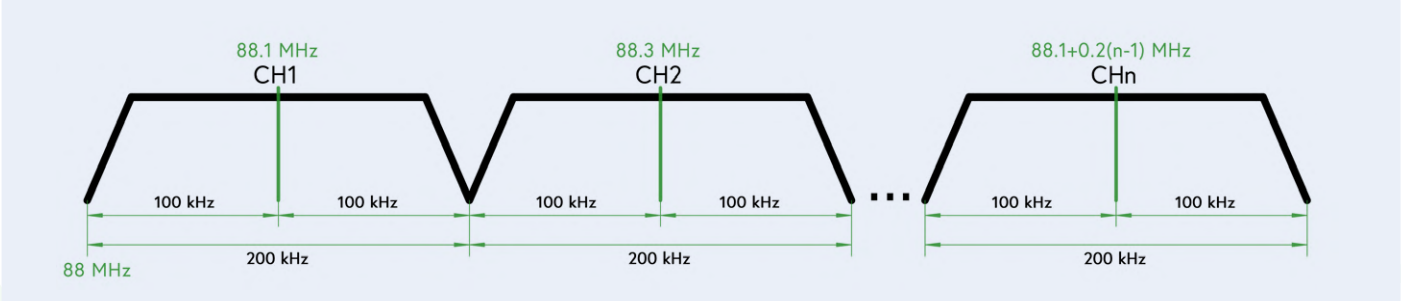
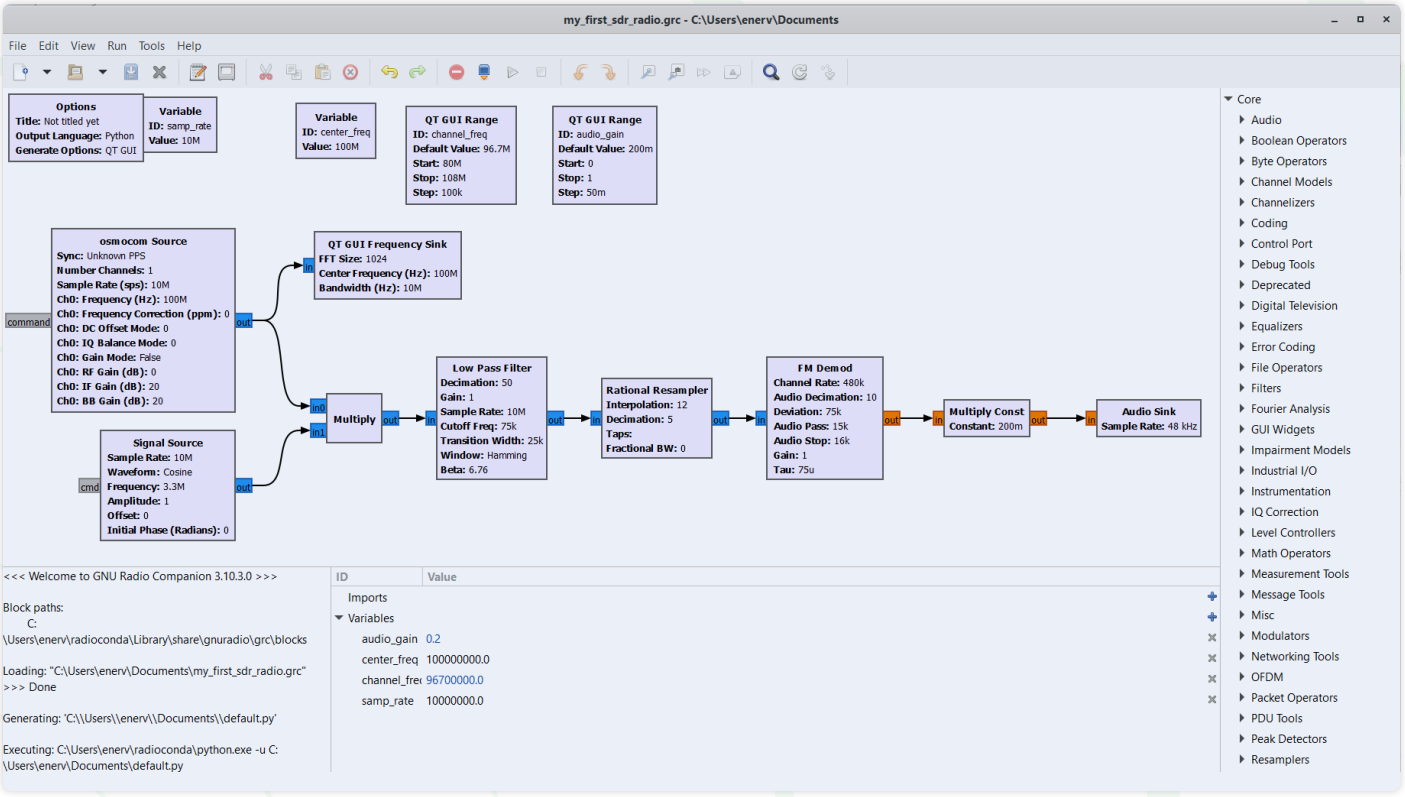


Bild 5. Das UKW-Frequenzband ist in 200 kHz breite Kanäle unterteilt.



Signalquelle die Differenz der beiden Werte sein. Die Differenz zwischen 100 MHz und 96,7 MHz beträgt 3,3 MHz, und das ist tatsächlich die im Block *Signal Source* angezeigte Frequenz.

Abwärtskonvertierung

Wenn der Sender auf der Mittenfrequenz liegt, müssen wir sein Signal extrahieren und abmischen, damit es demoduliert werden kann. Extrahieren bedeutet Filtern und Abmischen oder Herunterkonvertieren im SDR die Reduzierung oder Dezimierung der Abtastrate. Das *Low Pass Filter* aus dem Abschnitt *Filters* vereint beide Vorgänge in einem einzigen Block.

Das UKW-Band ist in Kanäle unterteilt, deren Breite von Land zu Land variiert, aber in der Regel bei 200 kHz (oder 100 kHz) liegt. Der Radiosender befindet sich in der Mitte eines Kanals, was bedeutet, dass er auf beiden Seiten 100 kHz Bandbreite hat (**Bild 5**). Die für ein 100-kHz-Signal erforderliche Mindestabtastrate beträgt gemäß dem Nyquist-Shannon-Theorem 200 kHz. Unsere Abtastrate beträgt 10 MHz, daher setzen wir den *decimation value* des Filterblocks auf 10 MHz / 200 kHz, also auf 50. Um das interessierende Signal zu extrahieren, brauchen wir kein supersteiles Filter; eine Grenzfrequenz von 75 kHz mit einer Übergangsbreite von 25 kHz ist für eine Gesamtbandbreite von 100 kHz ausreichend.

Demodulieren

Wir haben nun ein FM-moduliertes Quadratursignal mit einer Abtastrate von 200 kHz, das wir demodulieren wollen. Hierfür können wir den Block *FM Demod* aus dem Bereich *Modulators* verwenden. Am Ausgang des Demodulators liegt ein Audiosignal, das wir in ein Audiogerät einspeisen können, in unserem Fall in die Soundkarte des Computers.

Das Problem, auf das wir jetzt stoßen, ist die Anpassung der Samplerate. Die Audiosenke unterstützt verschiedene Abtastraten, die von der Soundkarte abhängen. Auf meinem Computer ist in allen Fällen das Verhältnis der möglichen Audio-Abtastraten zur Eingangs-Abtastrate von 200 kHz ein nicht ganzzahliger Wert. Aber der FM-Demodulator kann nur um einen ganzzahligen Wert dezimieren. Wie macht man es also richtig?

Die Lösung besteht darin, einen Sample-rate-Konverter (Resampler) einzufügen, der eine gebrochene Umwandlung durchführen kann. GRC bietet dafür zwei Optionen: einen fraktionalen und einen rationalen Resampler. Da unsere Eingangs- und Ausgangsraten ganzzahlige Werte sind, können wir den rationalen Resampler aus dem Abschnitt *Resamplers* verwenden. Ein fraktionaler Resampler kann ebenfalls verwendet werden, aber er verbraucht viel mehr der wertvollen Rechenressourcen. Um die beste Audioqualität zu erzielen, stellen wir die Abtastrate auf 48 kHz ein. Wenn wir dann den Demodulator mit der 10-fachen Abtastrate, also 480 kHz, arbeiten lassen, benötigen wir eine Abtastratenumsetzung von 200 : 480. Dieses Verhältnis lässt sich vereinfachen, indem man den größten gemeinsamen Teiler der beiden Werte ermittelt, der 40 beträgt. 200 : 40 = 5, und 480 : 40 = 12. Wenn wir also zuerst um 12 hochrechnen oder interpolieren und dann um 5 herunterrechnen oder dezimieren, wird aus einer Rate von 200 kHz eine Rate von 480 kHz. Diese Rate muss im *FM Demod*-Block zusammen mit einem Dezimierungsfaktor von zehn eingestellt werden, damit die Ausgangsrate mit der Eingangsrate der Audiosenke übereinstimmt.

Benutzersteuerung hinzufügen

Wenn Sie dieses Flussdiagramm ausführen und die Kanalfrequenz auf einen UKW-Radiosender in der Nähe einstellen, sollten Sie nun das Radioprogramm hören. Ist das nicht toll?

▲ *Bild 6. Das vollständige Flussdiagramm des UKW-Radioempfängers mit Abstimmungs- und Lautstärkeregler. Wenn Sie das Programm starten, wird es wie in Bild 4 aussehen.*

Wir können unser Radio noch ein wenig verbessern, indem wir eine Abstimmungssteuerung hinzufügen, mit dem Sie ganz einfach einen anderen Radiosender einstellen können. Dazu können wir einen *QT GUI Range*-Block aus dem Abschnitt *GUI Widgets* verwenden. Setzen Sie seine ID auf `channel_freq`, füllen Sie die Parameter aus und stellen Sie sicher, dass sie alle im Bereich liegen. Beachten Sie, dass es keinen Sinn hat, einen (sehr) kleinen Schrittwert einzustellen, da die meisten UKW-Sender auf MHz-Frequenzen mit nur einer Dezimalstelle arbeiten, so dass 100 kHz wahrscheinlich ausreichend ist.
Sie können unter etlichen Formen für die Steuerung wählen, aber Zähler plus Schieberegler sind am praktischsten, da Sie so die Frequenz als numerischen Wert sehen können. Sie können aber auch direkt eine Frequenz eingeben. Benennen Sie den alten Variablenblock `channel_freq` um oder löschen Sie ihn, da er nicht mehr benötigt wird und in Konflikt mit dem Schieberegler steht. Sie können das SDR jetzt wie jedes andere Radio einstellen.

Signalfarben

Eine weitere Verbesserung ist das Hinzufügen eines Lautstärkereglers. Dieser ist ähnlich wie der Abstimmungsregler, und Sie können ihn kopieren und seine Parameter anpassen. Benennen Sie die ID in `audio_gain` um. Wählen Sie eine *Multiply Const* aus dem Abschnitt *Math Operators* und fügen Sie sie zwischen dem *FM Demod*-Block und der Audiosenke ein. Beachten Sie, dass die Drähte jetzt rot sind. Das liegt daran, dass der Eingang und der Ausgang des Multiplizierers blau und nicht orange, also nicht vom gleichen Typ sind.
Öffnen Sie die Eigenschaften des *Multipliers* und setzen Sie den *IO-Type* auf Float. Setzen Sie außerdem das Feld mit der Bezeichnung *Constant* auf `audio_gain`, um es mit dem Schieberegler zu verbinden. Schließen Sie den Block und beachten Sie, dass die Drähte schwarz geworden sind, da alle Audioein- und -ausgänge jetzt orange sind.

Ausgabe-Optionen

Standardmäßig gibt GRC ein Python-Skript mit GUI aus, wie in den *Generate Options* im Block *Options* definiert. Es gibt aber auch andere Möglichkeiten wie C++. Wenn Sie alle GUI-Blöcke aus dem Flussdiagramm entfernen und *Generate Options* auf *No GUI* setzen, kann das Skript außerhalb von GRC ausgeführt werden. So können Sie eigenständige SDR-Anwendungen erstellen.

Das wars. Führen Sie das Flussdiagramm aus und genießen Sie Ihren softwaredefinierten UKW-Radioempfänger! Um tiefer in die faszinierende Welt von SDR einzutauchen, empfehle ich Ihnen den hervorragenden Videokurs von Michael Ossmann [6], der diesen Artikel inspiriert hat.

(220659-02)RG

Haben Sie Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Senden Sie eine E-Mail an den Autor unter clemens.valens@elektor.com oder kontaktieren die Elektor-Redaktion unter redaktion@elektor.de.

Passende Produkte

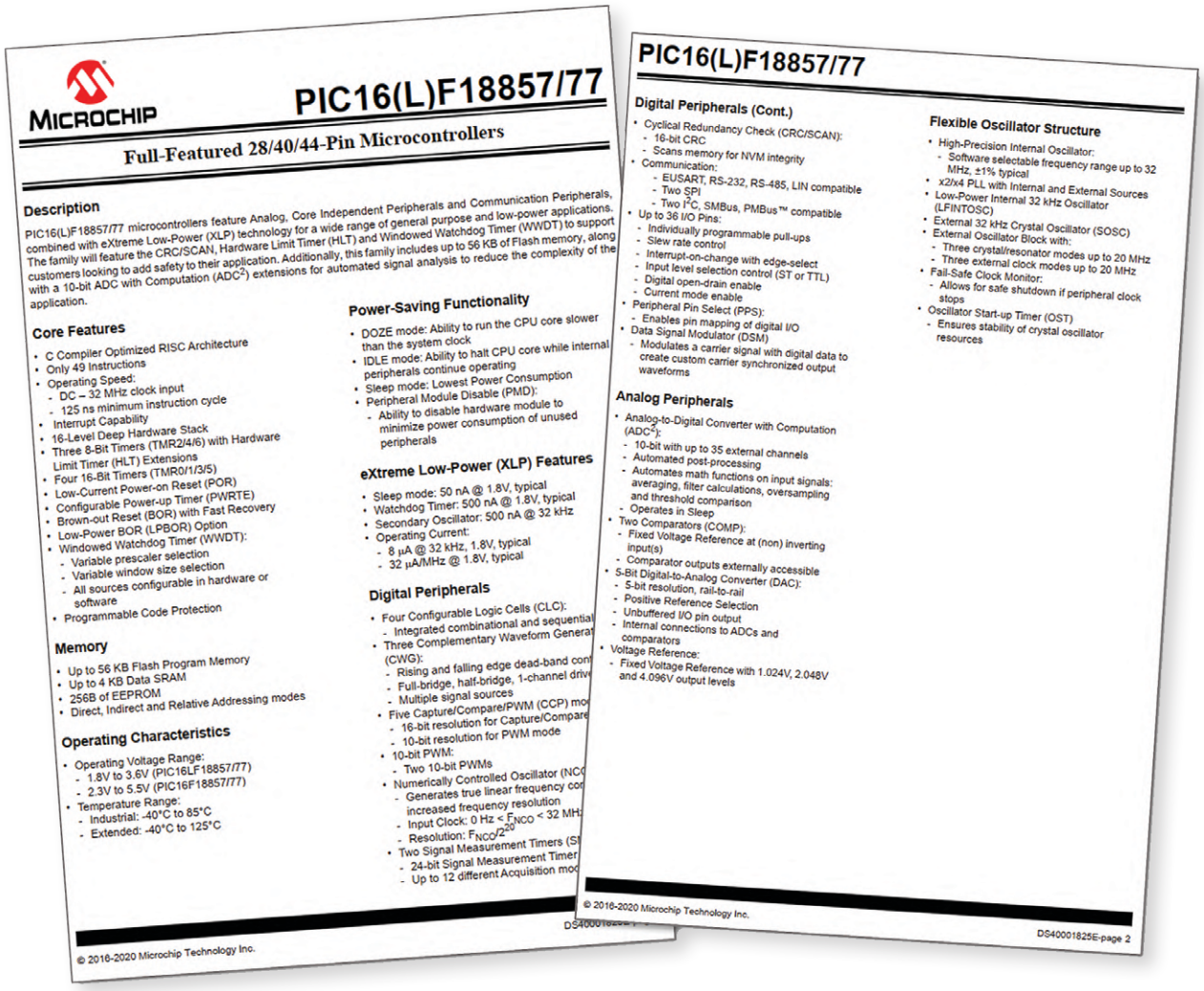
- **Great Scott Gadgets HackRF One Software Defined Radio (1 MHz bis 6 GHz)**
  SKU 18306: www.elektor.de/18306
- **Great Scott Gadgets ANT500 Teleskopantenne (75 MHz bis 1 GHz)**
  SKU 18481: www.elektor.de/18481
- **Elektor SDR-Hands-on Kit**
  SKU 19041: www.elektor.com/19041

WEBLINKS

[1] HackRF One bei Great Scott Gadgets: https://greatscottgadgets.com/hackrf/
[2] GNU Radio und GRC: https://www.gnuradio.org/
[3] SDR# a.k.a. SDR Sharp: https://airspy.com/download/
[4] SDR++: https://www.sdrpp.org/
[5] HackRF One auf GitHub: https://github.com/greatscottgadgets/hackrf
[6] Youtube-Videokurs zu SDR von Michael Ossmann: https://greatscottgadgets.com/sdr/

Mikrocontroller-Dokumentation verstehen

Teil 1: Aufbau einer Dokumentation



MICROCHIP

PIC16(L)F18857/77

Full-Featured 28/40/44-Pin Microcontrollers

Description

PIC16(L)F18857/77 microcontrollers feature Analog, Core Independent Peripherals and Communication Peripherals, combined with eXtreme Low-Power (XLP) technology for a wide range of general purpose and low-power applications. The family will feature the CRC/SCAN, Hardware Limit Timer (HLT) and Windowed Watchdog Timer (WWDT) to support customers looking to add safety to their application. Additionally, this family includes up to 56 KB of Flash memory, along with a 10-bit ADC with Computation (ADC^2) extensions for automated signal analysis to reduce the complexity of the application.

Core Features

- C Compiler Optimized RISC Architecture
- Only 49 Instructions
- Operating Speed:
  - DC – 32 MHz clock input
  - 125 ns minimum instruction cycle
- Interrupt Capability
- 16-Level Deep Hardware Stack
- Three 8-Bit Timers (TMR2/4/6) with Hardware Limit Timer (HLT) Extensions
- Four 16-Bit Timers (TMR0/1/3/5)
- Low-Current Power-on Reset (POR)
- Configurable Power-up Timer (PWRTE)
- Brown-out Reset (BOR) with Fast Recovery
- Low-Power BOR (LPBOR) Option
- Windowed Watchdog Timer (WWDT):
  - Variable prescaler selection
  - Variable window size selection
  - All sources configurable in hardware or software
- Programmable Code Protection

Memory

- Up to 56 KB Flash Program Memory
- Up to 4 KB Data SRAM
- 256B of EEPROM
- Direct, Indirect and Relative Addressing modes

Operating Characteristics

- Operating Voltage Range:
  - 1.8V to 3.6V (PIC16LF18857/77)
  - 2.3V to 5.5V (PIC16F18857/77)
- Temperature Range:
  - Industrial: -40°C to 85°C
  - Extended: -40°C to 125°C

Power-Saving Functionality

- DOZE mode: Ability to run the CPU core slower than the system clock
- IDLE mode: Ability to halt CPU core while internal peripherals continue operating
- Sleep mode: Lowest Power Consumption
- Peripheral Module Disable (PMD):
  - Ability to disable hardware module to minimize power consumption of unused peripherals

eXtreme Low-Power (XLP) Features

- Sleep mode: 50 nA @ 1.8V, typical
- Watchdog Timer: 500 nA @ 1.8V, typical
- Secondary Oscillator: 500 nA @ 32 kHz
- Operating Current:
  - 8 µA @ 32 kHz, 1.8V, typical
  - 32 µA/MHz @ 1.8V, typical

Digital Peripherals

- Four Configurable Logic Cells (CLC):
  - Integrated combinational and sequential
- Three Complementary Waveform Generat (CWG):
  - Rising and falling edge dead-band con
  - Full-bridge, half-bridge, 1-channel driv
  - Multiple signal sources
- Five Capture/Compare/PWM (CCP) mo
  - 16-bit resolution for Capture/Compare
  - 10-bit resolution for PWM mode
- 10-bit PWM:
  - Two 10-bit PWMs
- Numerically Controlled Oscillator (NC
  - Generates true linear frequency co increased frequency resolution
  - Input Clock: 0 Hz < F_{NCO} < 32 MH
  - Resolution: $F_{NCO}/2^{20}$
- Two Signal Measurement Timers (S
  - 24-bit Signal Measurement Timer
  - Up to 12 different Acquisition mo

DS40001825E-page 2

PIC16(L)F18857/77

Digital Peripherals (Cont.)

- Cyclical Redundancy Check (CRC/SCAN):
  - 16-bit CRC
  - Scans memory for NVM integrity
- Communication:
  - EUSART, RS-232, RS-485, LIN compatible
  - Two SPI
  - Two I^2C, SMBus, PMBus™ compatible
- Up to 36 I/O Pins:
  - Individually programmable pull-ups
  - Slew rate control
  - Interrupt-on-change with edge-select
  - Input level selection control (ST or TTL)
  - Digital open-drain enable
  - Current mode enable
- Peripheral Pin Select (PPS):
  - Enables pin mapping of digital I/O
- Data Signal Modulator (DSM)
  - Modulates a carrier signal with digital data to create custom carrier synchronized output waveforms

Analog Peripherals

- Analog-to-Digital Converter with Computation (ADC^2):
  - 10-bit with up to 35 external channels
  - Automated post-processing
  - Automates math functions on input signals: averaging, filter calculations, oversampling and threshold comparison
  - Operates in Sleep
- Two Comparators (COMP):
  - Fixed Voltage Reference at (non) inverting input(s)
  - Comparator outputs externally accessible
- 5-Bit Digital-to-Analog Converter (DAC):
  - 5-bit resolution, rail-to-rail
  - Positive Reference Selection
  - Unbuffered I/O pin output
  - Internal connections to ADCs and comparators
- Voltage Reference:
  - Fixed Voltage Reference with 1.024V, 2.048V and 4.096V output levels

Flexible Oscillator Structure

- High-Precision Internal Oscillator:
  - Software selectable frequency range up to 32 MHz, ±1% typical
- x2/x4 PLL with Internal and External Sources
- Low-Power Internal 32 kHz Oscillator (LFINTOSC)
- External 32 kHz Crystal Oscillator (SOSC)
- External Oscillator Block with:
  - Three crystal/resonator modes up to 20 MHz
  - Three external clock modes up to 20 MHz
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if peripheral clock stops
- Oscillator Start-up Timer (OST)
  - Ensures stability of crystal oscillator resources

Bild 1. Auf den ersten Seiten sind die im Mikrocontroller integrierten Funktionen aufgelistet. (Quelle: Microchip Technology)

Von **Stuart Cording** (Elektor)

Dokumentationen - egal, ob man sie liebt oder hasst, man muss sich damit auseinandersetzen. Bei Mikrocontrollern gibt es eine Menge Dokumentationen, weil sie im Vergleich zu anderen, einfacheren Halbleiterbauelementen wirklich komplexe Bauteile sind. In dieser Artikelserie werfen wir einen Blick darauf, wie Mikrocontroller dokumentiert sind, was im Datenblatt steht, und was nicht, und wo die fehlenden Details zu finden sind!

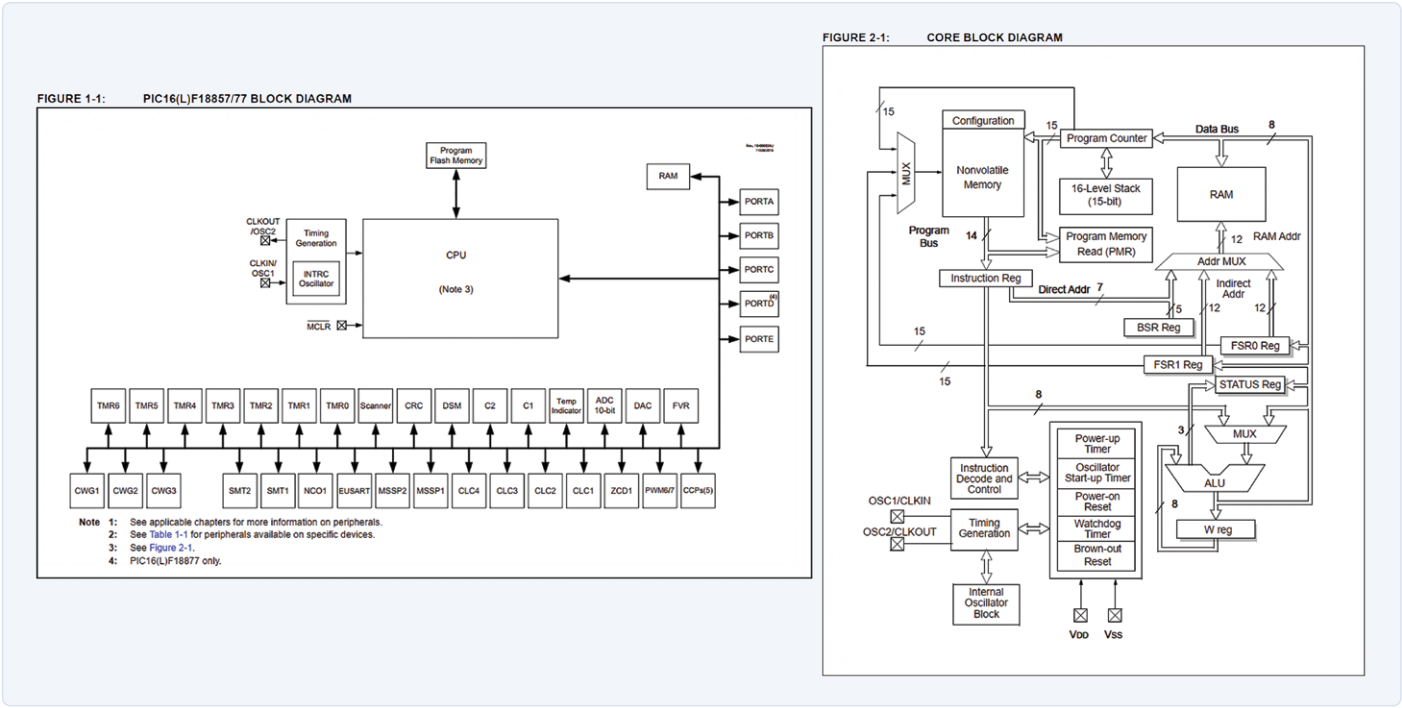


Bild 2. Das Blockdiagramm ist ein guter Ausgangspunkt, um die Gesamtfähigkeiten des Mikrocontrollers (links) und manchmal auch des Prozessorkerns (rechts) zu ermitteln. (Quelle: Microchip Technology)

Mikrocontroller-Datenblätter können heute leicht mehr als 600 Seiten umfassen. Glücklicherweise bietet Elektor eine Fülle von Artikeln und Bücher, die Ihnen den Einstieg erleichtern. Irgendwann werden Sie sich jedoch mit der echten Mikrocontroller-Dokumentation vertraut machen müssen.
Obwohl (oder weil?) eine Mikrocontroller-Dokumentation so viele Seiten umfasst, ist es unwahrscheinlich, dass Sie alle Informationen finden, nach denen Sie suchen. Zur Vervollständigung des Datenblattes müssen Sie auch die Dokumentation für die Tools finden, die den Quellcode in Firmware umwandeln, Tools für die Codeentwicklung und das Debugging sowie Tools für die Programmierung in der Massenproduktion. Wenn Sie neu in der Welt der Mikrocontroller sind, wird Ihnen dieser Leitfaden helfen, zu verstehen, wo welche Dinge dokumentiert sind, wie Sie die Hieroglyphen des Inhalts entziffern können und wo Sie herausfinden können, ob in dem, was Sie gelesen haben, mögliche Fehler enthalten sind.

Eine Dokumentation für den Microchip PIC16F18877

Zum Auftakt des ersten Teils dieser Serie von drei Artikeln konzentrieren wir uns auf einen einfachen 8-Bit-Mikrocontroller wie den PIC16F18877 und darauf, welche Dokumentationen angeboten werden. Öffnen Sie diesen Link und klicken Sie dann auf *Documents*. Sie werden sehen, dass uns das Datenblatt, Errata, einige *Supporting Collateral* zu einigen anwendungsspezifischen Punkten und Evaluierungsboards, die Programmierspezifikationen sowie eine lange Liste von Anwendungshinweisen angeboten werden. Weiter unten finden Sie den Quellcode zu einigen der Anwendungshinweise, einige Verkaufsbroschüren und ein Weißbuch über ADCs (Analog-Digital-Wandler).

Was steht im Datenblatt zum Mikrocontroller?

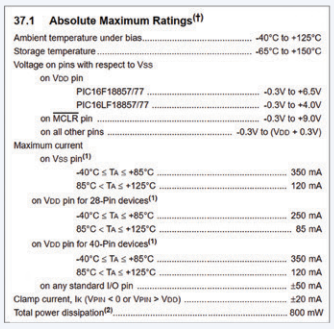
Wir beginnen mit dem Herunterladen des Datenblatts für den PIC16F18877 [2]. Datenblätter für Mikrocontroller können ziemlich entmutigend sein, und enthalten, vielleicht überraschend, nicht alles, was Sie wissen müssen. **Bild 1** zeigt, welche Informationen Sie aber mindestens finden sollten:

- **Detaillierte Informationen zu einem oder mehreren Mikrocontrollern** - Oft werden aus einem einzigen Silizium-Die mehrere verschiedene Mikrocontroller mit einer unterschiedlichen Anzahl von Pins und Gehäusen erstellt. Anstatt für jede Variante eine eigene Dokumentation zu erstellen und zu pflegen, werden Sie eher mehrere Bausteine in einem einzigen Datenblatt finden. Dies ist hier der Fall, wie auf Seite 1 erläutert, wobei zwei Varianten abgedeckt werden: der PIC16F18857 und der PIC16F18877.
- **Blockschaltbild des Mikrocontrollers** - Dies umfasst typischerweise den Prozessorkern (der Detaillierungsgrad variiert; je komplexer der Kern, desto einfacher die schematische Umsetzung), die Speicher, Busse und Peripherie. Daraus lassen sich schnell die grundlegenden Fähigkeiten des Mikrocontrollers ableiten. Das Blockschaltbild des Mikrocontrollers (**Bild 2**) finden Sie auf Seite 18, ein Blockschaltbild des Prozessorkerns auf Seite 33.
- **Gehäuseoptionen** - Diese reichen von Typen für Durchsteckmontage (falls noch verfügbar) bis hin zu einer Reihe von Optionen für die Oberflächenmontage. In diesem Beispiel beginnen sie auf Seite 4.
- **Speicher-Optionen** - Der Mikrocontroller wird möglicherweise mit unterschiedlichen Größen von RAM, Flash, EEPROM und anderen Speichertypen wie zum Beispiel Caches angeboten. In unserem Beispiel erhalten wir einen schnellen Überblick auf Seite 3. Die Tabelle listet die Speichergrößen auf und gibt an, wie viele der einzelnen Peripheriebausteine implementiert sind (**Bild 3**).
- **On-Chip-Peripherie-Blockdiagramme** - Die Funktionalität der On-Chip-Peripherie lässt sich leichter in Diagrammen als in Worten erklären. Blockdiagramme bieten eine wichtige Quelle für das Verständnis sowie Klarheit über Pin-Anschlüsse und Taktquellen, wenn der Mikrocontroller über mehrere Taktausgänge seines Oszillators und, falls vorhan-

PIC16(L)F188XX Family Types

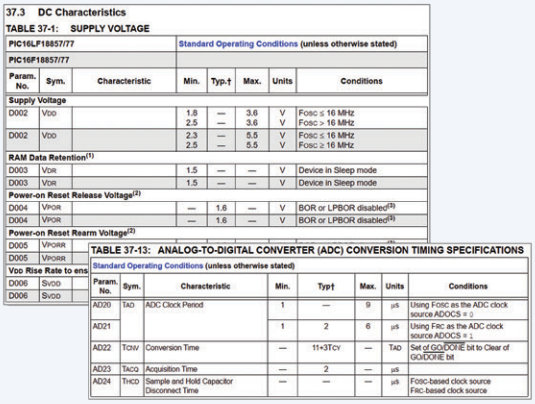
Device	Data Sheet Index	Program Flash Memory (Words)	Program Flash Memory (KB)	EEPROM (bytes)	Data SRAM (bytes)	I/O Pins(1)	10-Bit ADC2 (ch)	5-Bit DAC	Comparator	8-Bit (with HLT)/ 16-Bit Timers	SMT	Windowed Watchdog Timer	CRC and Memory Scan	CCP/10-Bit PWM	Zero-Cross Detect	CWG	NCO	CLC	DSM	EUSART/I2C/SPI	Peripheral Pin Select	Peripheral Module Disable
PIC16(L)F18854	(1)	4096	7	256	512	25	24	1	2	3/4	2	Y	Y	5/2	Y	3	1	4	1	1/2	Y	Y
PIC16(L)F18855	(2)	8192	14	256	1024	25	24	1	2	3/4	2	Y	Y	5/2	Y	3	1	4	1	1/2	Y	Y
PIC16(L)F18856	(3)	16384	28	256	2048	25	24	1	2	3/4	2	Y	Y	5/2	Y	3	1	4	1	1/2	Y	Y
PIC16(L)F18857	(4)	32768	56	256	4096	25	24	1	2	3/4	2	Y	Y	5/2	Y	3	1	4	1	1/2	Y	Y
PIC16(L)F18875	(2)	8192	14	256	1024	36	35	1	2	3/4	2	Y	Y	5/2	Y	3	1	4	1	1/2	Y	Y
PIC16(L)F18876	(3)	16384	28	256	2048	36	35	1	2	3/4	2	Y	Y	5/2	Y	3	1	4	1	1/2	Y	Y
PIC16(L)F18877	(4)	32768	56	256	4096	36	35	1	2	3/4	2	Y	Y	5/2	Y	3	1	4	1	1/2	Y	Y

Bild 3. Werfen Sie einen Blick auf die Speichermöglichkeiten der PIC16F18x7-Varianten sowie auf die implementierte Peripherie. (Quelle: Microchip Technology)



37.1 Absolute Maximum Ratings(†)

Ambient temperature under bias ..... -40°C to +125°C
Storage temperature ..... -65°C to +150°C
Voltage on pins with respect to Vss
on VDD pin
PIC16F18857/77 ..... -0.3V to +6.5V
PIC16LF18857/77 ..... -0.3V to +4.0V
on MCLR pin ..... -0.3V to +9.0V
on all other pins ..... -0.3V to (VDD + 0.3V)
Maximum current
on Vss pin(1)
-40°C ≤ TA ≤ +85°C ..... 350 mA
85°C < TA ≤ +125°C ..... 120 mA
on VDD pin for 28-Pin devices(1)
-40°C ≤ TA ≤ +85°C ..... 250 mA
85°C < TA ≤ +125°C ..... 85 mA
on VDD pin for 40-Pin devices(1)
-40°C ≤ TA ≤ +85°C ..... 350 mA
85°C < TA ≤ +125°C ..... 120 mA
on any standard I/O pin ..... ±50 mA
Clamp current, IK (VPIN < 0 or VPIN > VDD) ..... ±20 mA
Total power dissipation(2) ..... 800 mW



37.3 DC Characteristics

TABLE 37-1: SUPPLY VOLTAGE

Param. No.	Sym.	Characteristic	Min.	Typ.†	Max.	Units	Conditions
PIC16LF18857/77			Standard Operating Conditions (unless otherwise stated)				
PIC16F18857/77							
Supply Voltage							
D002	VDD		1.8	—	3.6	V	FOSC ≤ 16 MHz
			2.5	—	3.6	V	FOSC > 16 MHz
D002	VDD		2.3	—	5.5	V	FOSC ≤ 16 MHz
			2.5	—	5.5	V	FOSC ≥ 16 MHz
RAM Data Retention(1)							
D003	VDR		1.5	—	—	V	Device in Sleep mode
D003	VDR		1.5	—	—	V	Device in Sleep mode
Power-on Reset Release Voltage(2)							
D004	VPOR		—	1.6	—	V	BOR or LPBOR disabled(3)
D004	VPOR		—	1.6	—	V	BOR or LPBOR disabled(3)
Power-on Reset Rearm Voltage(2)							
D005	VPORR						
D005	VPORR						
VDD Rise Rate to ens							
D006	SVDD						
D006	SVDD						

TABLE 37-13: ANALOG-TO-DIGITAL CONVERTER (ADC) CONVERSION TIMING SPECIFICATIONS

Param. No.	Sym.	Characteristic	Min.	Typ†	Max.	Units	Conditions
Standard Operating Conditions (unless otherwise stated)							
AD20	TAD	ADC Clock Period	1	—	9	µs	Using FOSC as the ADC clock source ADOCS = 0
AD21			1	2	6	µs	Using FRC as the ADC clock source ADOCS = 1
AD22	TCNV	Conversion Time	—	11+3TCY	—	TAD	Set of GO/DONE bit to Clear of GO/DONE bit
AD23	TACQ	Acquisition Time	—	2	—	µs	
AD24	THCD	Sample and Hold Capacitor Disconnect Time	—	—	—	µs	FOSC-based clock source FRC-based clock source

Bild 4. Die elektrischen Spezifikationen für DC- und AC werden als absolute Grenzwerte sowie als Minimal/Typisch/Maximalwerte und als Timing-Grenzwerte angegeben. (Quelle: Microchip Technology)

den, über PLL (Phase-Locked Loop) verfügt. Einige Beispiele werden wir uns demnächst in Teil 2 der Artikelreihe ansehen.

› **Registerbeschreibungen** - Jeder Peripheriebaustein kann auf eine bestimmte Weise konfiguriert werden. Zum Beispiel kann ein UART (universelle asynchrone serielle Empfangs-/Sendeschnittstelle) oft für verschiedene Baudraten, Anzahl der Bits und so weiter eingestellt werden. In der Registerbeschreibung wird erklärt, wie das Peripheriegerät konfiguriert wird und wie man seinen Zustand feststellt, nachdem etwas passiert ist, zum Beispiel der Empfang eines Datenbytes. Wie man dies entschlüsselt, wird ebenfalls im zweiten Teil des Artikels behandelt.

› **Elektrische Spezifikationen** - Dies informiert den Benutzer über die Grenzwerte für Spannung und Strom, die an die Pins des Mikrocontrollers angelegt oder von ihnen gezogen werden können.Sie werden üblicherweise zweimal definiert, einmal als absolute Maximalwerte und einmal als minimale, typische und maximale Werte während des normalen Betriebs. Es gibt auch Timing-Charakteristiken, wie in **Bild 4** gezeigt, die ab Seite 592 in diesem Datenblatt zu finden.

› **Empfehlungen für Tools** - Sie benötigen Tools, um den Quellcode in Assembler zu konvertieren und Werkzeuge zum Debuggen der Ergebnisse. Die meisten Datenblätter enthalten eine Art Empfehlung der verfügbaren Tools. Hier werden sie kurz auf Seite 638 behandelt.

Haben Sie Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare? Schicken Sie ein E-Mail an den Autor unter stuart.cording@elektor.com.

Ausblick auf Teil 2

Nachdem wir die grundlegende Struktur eines Mikrocontroller-Datenblatts verstanden haben, werden wir im nächsten Teil des Artikels untersuchen, wie Register beschrieben werden und wie man die Blockdiagramme entschlüsselt. Außerdem werden wir uns die beiden wichtigsten Blöcke eines Mikrocontrollers - den Taktgeber und den Oszillator - sowie die Implementierung der Reset-Schaltung genauer ansehen.

200721-02

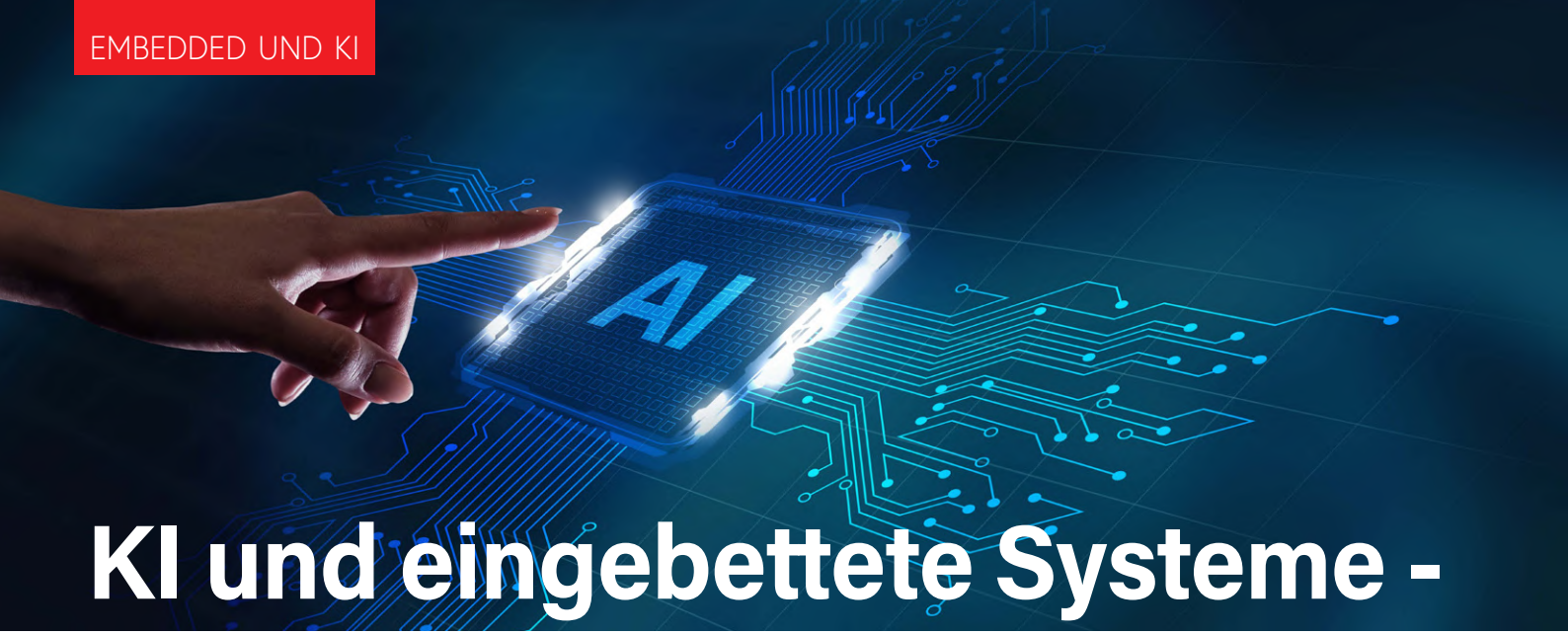
WEBLINKS

[1] Produktseite PIC16F18877: http://bit.ly/2KS1s8C
[2] Datenblatt PIC16F18877: https://bit.ly/3nNwPjn

Passende Produkte

› **T. Hanna, *Microcontroller-Basics mit PIC*, Elektor 2020**
www.elektor.de/mikrocontroller-basics-mit-pic

› **A. Pratt, *Programming the Finite State Machine*, Elektor 2020**
www.elektor.de/programming-the-finite-state-machine



KI und eingebettete Systeme - Was kommt als Nächstes?

Werkzeuge, Plattformen und das Ende der Textarbeiter

Von Stuart Cording (Elektor)

KI wird oft als das Schwert dargestellt, das jeden auch noch so kompliziert verwickelten Gordischen Knoten lösen kann. In anderen Fällen wird behauptet, dass KI das Ende der menschlichen Zivilisation einläutet. Als Ingenieure wissen wir, dass weder das eine noch das andere zutrifft. Aber wie können wir KI, oder besser gesagt maschinelles Lernen, in den von uns entwickelten Anwendungen besser nutzen? Und werden wir angesichts der jüngsten Fortschritte in der KI wirklich überflüssig?

Die Erwähnung der künstlichen Intelligenz (KI) ist ein sicherer Weg, um die Aufmerksamkeit der Mainstream-Presse auf sich zu ziehen. Dank des Internets und der Cloud-Dienste, gepaart mit manchmal fragwürdigen Datenquellen, scheinen überall neue KI-gesteuerte Dienste aufzutauchen. Mit ein paar Mausklicks können Ihre Worte in den Mund von Morgan Freeman gelegt oder Ihr Bild in ein neues Kunstwerk integriert werden. Das kann zwar amüsant sein und zum Nachdenken anregen, aber es ist nicht leicht, darin eine Verbindung zur Welt der eingebetteten Systeme zu finden. Aber genau die sind es, worum es in der Industrie geht.

Die meisten eingebetteten Systeme verwenden regelbasierte Programmieransätze, um ihre Funktionen zu implementieren. Bei einem Satz von Eingangsdaten wird durch eine Reihe von if/else- oder switch-Anweisungen festgelegt, wie das System reagiert. Dies funktioniert zwar gut bei einer begrenzten Anzahl von Eingaben, aber ab einem gewissen Punkt wird es aufgrund der Anzahl der Eingangsdaten oder der Feinheit ihrer Beziehungen untereinander schwierig, klare programmatische Regeln zu definieren.

Stellen Sie sich zum Beispiel einen Motor in einer Fabrik vor, der 24 Stunden am Tag, sieben Tage die Woche läuft. Die Erfahrung lehrt, dass mit der Zeit Verschleiß auftritt und das Lagerfett dickflüssiger wird. Mit der Zeit verändern sich dadurch die Anlaufzeit, die Geräuschentwicklung, die Vibrationen, die Betriebstemperatur und die Stromaufnahme des Motors. Diese Herausforderung wird heute mit regelmäßiger Wartung gemeistert, was aber zu festen Ausfallzeiten führt, die die Produktion stoppen. Und ob das System zu oft oder zu selten gewartet wird, lässt sich meist nur schwer abschätzen.

Außerdem ist es unwahrscheinlich, dass ein drohender Ausfall aufgrund eines Haarrisses in der Welle, den Lagern, dem Gehäuse oder den Einbauten rechtzeitig erkannt wird. Richtig trainiert, können KI-Ansätze anhand einer komplexen Mischung von Datenquellen drohende Ausfälle „vorhersagen“. Wenn eine solche Intelligenz in einem mikrocontrollerbasierten System eingesetzt wird, erhalten Sie ein preiswertes Überwachungssystem, das Zeit und finanzielle Ressourcen spart und die Verschwendung durch unnötige Schmierung und den Austausch von Teilen reduziert.

Intelligenz in Mikrocontroller einbauen

Auf der Ebene der Mikrocontroller sprechen wir eher von maschinellem Lernen (ML) als von KI. Dies bedeutet, dass eine Maschine so programmiert wird, dass sie anhand von Regeln, die durch die Analyse verfügbarer Daten entwickelt wurden, Entscheidungen trifft. Mikrocontroller sind zwar mehr als leistungsfähig genug, um solche ML-Algorithmen auszuführen, aber das Lernen aus Trainingsdaten übersteigt ihre Möglichkeiten und erfordert mindestens einen Desktop-

Computer, wenn nicht sogar einen Cloud-Server. Das im Jahr 2019 gegründete Unternehmen Edge Impulse hat eine Plattform für ML in eingebetteten Systemen entwickelt und ist erfolgreich Partnerschaften mit den weltweit führenden Halbleiterherstellern [1] eingegangen, um eine breite Unterstützung zu bieten.

Der Ausgangspunkt für jede ML-Anwendung sind Daten. Während einige Anwendungen wie autonomes Fahren Terabytes an Trainingsdaten benötigen, können einfache mikrocontrollerbasierte Systeme bereits aus wenigen Kilobytes an Daten lernen. Die erste Herausforderung besteht also darin, die Daten von der Platine auf die Edge-Impulse-Umgebung zu übertragen. Der nächstliegende Gedanke wäre, die Daten über die serielle Schnittstelle eines Arduinos an den PC zu senden und sie von dort aus als Textdatei in Edge Impulse hochzuladen. Die Plattform ist jedoch so eingerichtet, dass die Daten direkt eingelesen werden können.

Daten – das Futter für die KI

Die Kommandozeilen-Applikation (CLI) *Data Forwarder* [2] ist ein Tool, das Daten von einem Entwicklungsboard direkt an die Edge-Impulse-Umgebung sendet. Mit Ihrem Benutzernamen und Passwort wird eine Verbindung zwischen der seriellen Schnittstelle Ihres PCs und dem Server hergestellt. Auf der Seite des Mikrocontrollers genügt es, die Daten über die serielle Schnittstelle in einem Komma- oder Tabulator-getrennten Format auszugeben. Solange die Abtastrate relativ niedrig ist, ist dies ein idealer Weg, um repräsentative Daten direkt von den Sensoren zu sammeln (**Bild 1**). Leistungsfähigere eingebettete Systeme wie der Raspberry Pi oder der Jetson Nano von NVIDIA können das bereitgestellte Software-Development-Kit (SDK) [3] verwenden, das auch Sensoren wie Mikrofone und Kameras unterstützt, die größere Datenmengen erzeugen.

Nachdem die Daten hochgeladen wurden, besteht der nächste Schritt darin, einen „Impuls“ zu definieren. Diese bestehen aus zwei Blöcken: Im ersten werden die Daten in kleinere Teile zerlegt und mit Techniken der Signalverarbeitung relevante Merkmale extrahiert. Dadurch können die verfügbaren Sensordaten in konsistente Informationen für die zweite Verarbeitungsstufe umgewandelt werden. Im zweiten Block finden das Lernen und die Klassifizierung statt (**Bild 2**). In einem Beispielprojekt *Continuous Motion Recognition* wird gut erklärt, wie diese Blöcke konfiguriert sind, um Daten von Beschleunigungssensoren zu analysieren und diese Eingabe als eine von vier möglichen Gesten zu klassifizieren [4].

Dies ist der kritischste Schritt in jeder ML-Entwicklung, der oft unkonventionelles Denken und mehrere Iterationen erfordert, um den besten Ansatz herauszufinden. Manchmal ist es die beste Lösung, einige Sensoreingaben zu ignorieren, in anderen Fällen müssen mehr Daten erhoben werden. Möglicherweise stellen Sie sogar fest, dass Sie viel zu viel lernen oder dass das gewählte neuronale Netzwerkmodell für die angestrebte Klassifizierung nicht geeignet ist. Ein weiterer wichtiger Schritt ist die Klassifizierung von Anomalien. In dem Beispielprojekt gibt es vier definierte Gesten. Andere Bewegungen, die den erlernten Gesten ähnlich sind, müssen jedoch ausgeschlossen werden. Eine gute Erkennung von Anomalien liefert ein robusteres ML-Ergebnis.

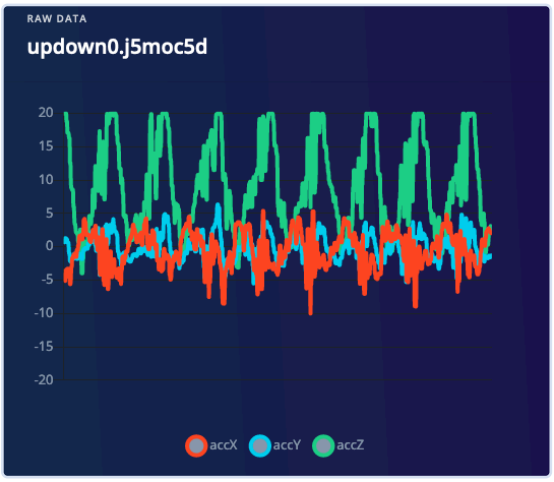


Bild 1. Ein Drei-Achsen-Beschleunigungsmesser liefert Bewegungsdaten für die Entwicklung einer ML-basierten Anwendung mit Gestenerkennung. (Quelle: Edge Impulse)

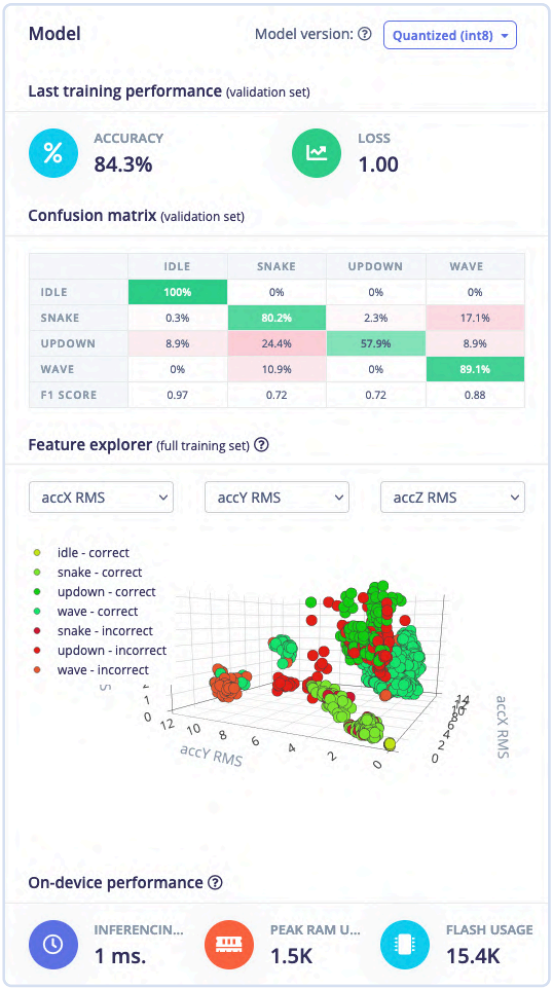


Bild 2. Die Edge-Impulse-Umgebung liefert nach einem ersten Trainingszyklus Rückmeldungen zu Genauigkeit, Geschwindigkeit und Speichernutzung. (Quelle: Edge Impulse)

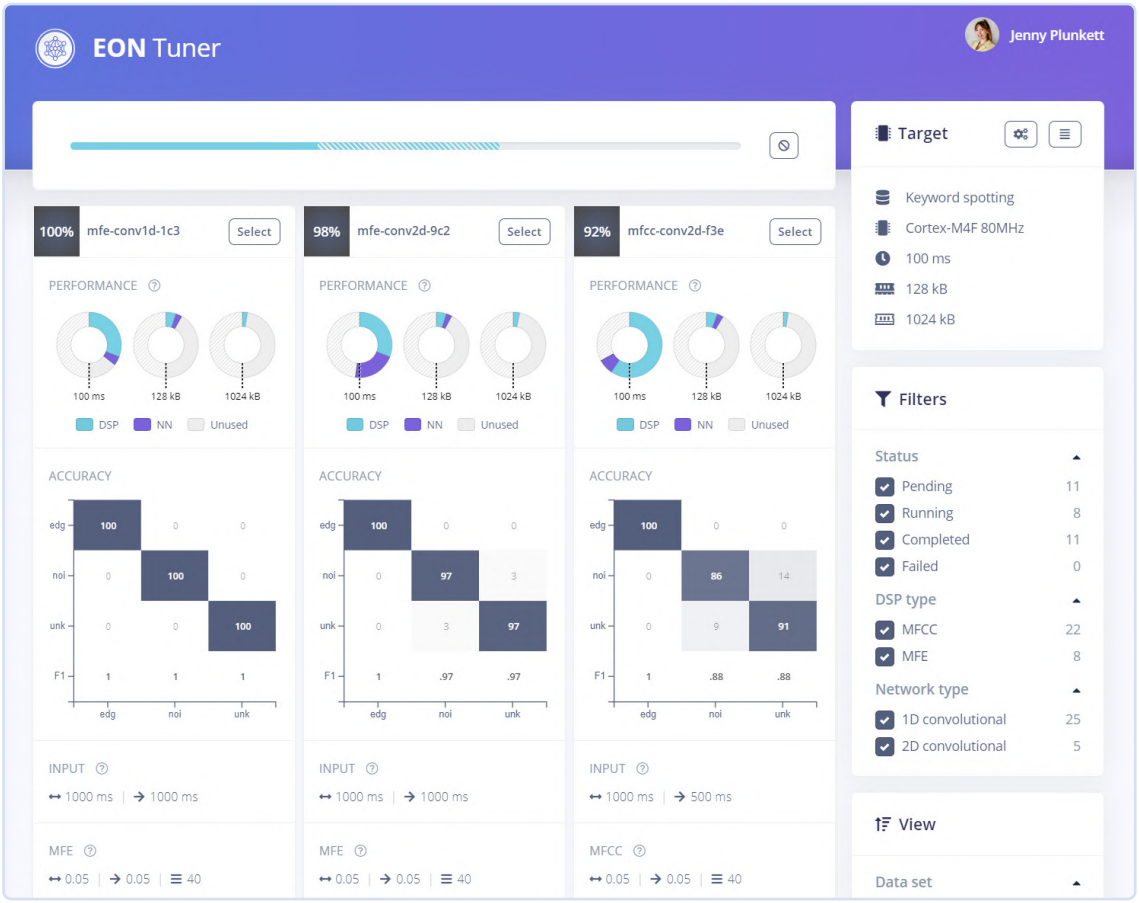


Bild 3. ML-Algorithmen werden mit dem EON-Tuner weiter für den Ziel-Mikrocontroller optimiert, wodurch die Inferenz-Reaktionszeiten und der Speicherbedarf reduziert werden können. (Quelle: Edge Impulse)

Der letzte Schritt ist die Bereitstellung. Über die Webschnittstelle wird die Firmware für den gewählten Controller/SBC heruntergeladen und in die Anwendung integriert. Für Arduino wird eine Bibliothek generiert, für andere Mikrocontroller eine C++-Datei erstellt. Natürlich ist die Leistung von Mikrocontrollern sehr unterschiedlich. Um das bestmögliche Ergebnis zu erzielen, bietet Edge Impulse seinen EON-Tuner [5] an. Mit diesem Tool kann die Erkennungsgenauigkeit weiter verbessert, die Inferenz beschleunigt und der Speicherbedarf verringert werden, indem Informationen über das Zielgerät, die Speichergröße und die Latenzzeit verwendet werden (**Bild 3**).

Bild 4. SlateSafety ermöglichte mit Edge Impulse die physiologische Überwachung mit Edge ML und verbesserte die Sicherheitsüberwachung von BAND V2. (Quelle: SafetySlate)

ML in realen Anwendungen

Reale Anwendungen nutzen diesen Ansatz, um ML-Funktionen einzubetten. BAND von SlateSafety [6] nutzt eine Reihe von biometrischen Sensoren zur Überwachung von Arbeitern, die unter schwierigen Bedingungen arbeiten (**Bild 4**), vom Ersthelfer bis zum Industriearbeiter, die wie Feuerwehrleute schwere persönliche Schutzausrüstung tragen. BAND lädt normalerweise Daten in die Cloud hoch, so dass Kollegen die Vitaldaten eines Benutzers überwachen können.

Besonders in Katastrophensituationen kann die Konnektivität jedoch lückenhaft oder gar nicht vorhanden sein. Das Entwicklungsteam setzte Edge Impulse ein, um Edge ML in das bestehende Produkt zu integrieren, das auf historischen biometrischen Daten trainiert wurde. Mit dem EON-Tuner wurde der Algorithmus für die Hardware optimiert und dann über ein Over-the-Air-Update bereitgestellt. Jetzt kann BAND auch ohne drahtlose Verbindung den Träger warnen, wenn das Risiko besteht, von der Hitze entkräftet zu werden.

Bessere Produktentwicklung mit KI

Natürlich kann man die KI nicht nur in das Produkt integrieren; man kann sie auch für dessen Entwicklung nutzen. Heutzutage werden viele komplexe Anwendungen mit einem modellbasierten Ansatz entwickelt, bei dem hauptsächlich eine Beschreibung der Funktionsweise der Anwendung mit Hilfe von Software und mathematisch-physikalischen Gleichungen erstellt wird. Aber auch dieser Ansatz hat seine Grenzen. Hier kommt Monolith und seine selbstlernende KI-Plattform [7] ins Spiel.

Die Plattform ist in der Lage, die physikalischen Eigenschaften komplexer Systeme auf der Grundlage bereits erfasster Daten zu erlernen. Beispielsweise durchlaufen Fahrzeuge auf einer Teststrecke eine Reihe von Tests, bei denen mehrere Sensoren Gier- und Wankbewegungen sowie Radgeschwindigkeit und Beschleunigung überwachen. Das Sammeln von Daten für verschiedene Steifigkeiten der Aufhängung vermittelt einen guten Überblick darüber, wie das Fahrzeug auf eine Reihe von Fahrsituationen reagiert. In der Regel werden die Daten analysiert, was zu neuen Einstellungen für die nächste Testfahrt führt. Monolith kann die Daten der ersten Testfahrten auswerten und das Ergebnis von Veränderungen der Aufhängung mit einem hohen Maß an Genauigkeit vorhersagen. Die Ergebnisse können verwendet werden, um die besten Einstellungen für die Aufhängung schneller zu finden, wodurch die Anzahl der zusätzlich erforderlichen physischen Testläufe reduziert wird.

Dieser Ansatz lässt sich auch auf die Messtechnik anwenden. Gaszähler beispielsweise müssen außerordentlich genau sein, um eine korrekte Abrechnung zu gewährleisten, doch das ist schwierig, wenn der Zähler verschiedene Gase messen muss. Bei einem Anwender stieß die Simulation von Ultraschallzählern an die Grenzen der rein mathematischen Analyse, so dass wiederholte Prüfverfahren die einzige Lösung für eine Kalibrierung waren, um die erforderliche Zertifizierung zu erreichen. Glücklicherweise führten all diese Tests zu einer umfangreichen Sammlung von Daten für die Analyse. Durch den Einsatz selbstlernender KI-Modelle konnte der Umfang der erforderlichen Tests um bis zu 70 % reduziert und die Entwicklung erheblich beschleunigt werden.

Ernsthafte Komprimierung der KI-Rechenleistung

Wettbewerbe wie die *DARPA Grand Challenge* [8], bei der Teams autonome Fahrzeuge konstruierten, die einen kurvenreichen Parcours durchfahren konnten, lösten eine Welle des Interesses an selbstfahrenden Autos aus. Heute, fast zwanzig Jahre später, sind zwar erhebliche Summen ausgegeben worden, aber es hat sich wenig berwegt. Tesla sorgt in diesem Bereich regelmäßig für Schlagzeilen, oft wenn übereifrige Besitzer zu viel Vertrauen in die Fähigkeiten ihres Fahrzeugs setzen [9]. Ansonsten scheint nur Waymo echte selbstfahrende Fahrzeuge für Fahrdienstleistungen anzubieten [10], aber diese operieren nur in Phoenix und San Francisco in den USA.

Eines der Probleme besteht darin, dass es äußerst schwierig ist, einen Computer dazu zu bringen, ein Auto zu fahren. Das Fahrzeug muss nicht nur ständig die Situation um sich herum einschätzen, sondern auch das Verhalten anderer Fahrer und Verkehrsteilnehmer wie Fußgänger und Radfahrer vorhersehen, die sich möglicherweise nicht an die Straßenverkehrsregeln halten.

▲ *Bild 5. Die Entwicklung von KI-Algorithmen wird durch den Einsatz von Hochleistungsrechnern im Fahrzeug wie in diesem flüssigkeitsgekühlten DYNACOR 40-36 beschleunigt. (Quelle: Eurotech)*

Die elektrische und elektronische Architektur von Fahrzeugen verändert sich, um den künftigen Anforderungen autonomer Fahrzeuge gerecht zu werden. Angesichts einer Vielzahl von Sensoren, die riesige Datenmengen liefern, geht die Branche zu Automotivem Ethernet über. Dieser Ansatz bildet derzeit die fortschrittlichen Fahrerassistenzsysteme (ADAS), die in die Steuerung von Bremsen, Beschleunigung und Lenkung eingreifen können, wenn wir Fahrer einen Fehler machen. Nach den Autonomiestufen der *Society of Automotive Engineers* (SAE) erfüllen entsprechende Premiumfahrzeuge derzeit die Stufe 2+, einige erreichen sogar die Stufe 3. Vollständige Autonomie zum „Zurücklehnen und Entspannen“ ist jedoch Stufe 5. Wir haben also noch einen weiten Weg vor uns!

Unternehmen wie Eurotech unterstützen die Industrie dabei, die Entwicklung der erforderlichen Algorithmen zu beschleunigen. Derzeit fallen bei einer achtstündigen Testfahrt 120 TB an Daten an, die zur Verarbeitung und Analyse ins Labor zurückgeschickt werden. Verbesserungen der KI-Algorithmen können im Labor anhand der gesammelten Daten getestet werden, aber bisher gab es nur wenige Möglichkeiten, Tests und die Entwicklung von Algorithmen in der Praxis zu unterstützen.

Eurotech nutzt seine Erfahrung im Bereich der Flüssigkeitskühlung und bietet eine Reihe von KI-Hardware an, die dieser Aufgabe gewachsen ist - im Wesentlichen kompakte Supercomputer, die in den Kofferraum eines Fahrzeugs passen. Ein Gerät wie der DYNACOR 40-36 ist für den Einsatz in On- und Offroad-Fahrzeugen geeignet [11]. Ausgestattet mit einer Intel-Xeon-CPU mit 16 Kernen und 64 GB RAM sowie bis zu zwei GV100-GPUs mit 32 GB RAM von NVIDIA bietet dieser lüfterlose Computer 237 TFLOPS für Deep-Learning-Anwendungen (**Bild 5**). Mehrere Gigabit-Ethernet-Schnittstellen unterstützen die Aufnahme der Masse an Sensordaten, von Radar und Kameras bis hin zu Lidar, die in 32 TB SSD-Speicher eingespeist werden. Durch mehr Inferenz- und Reinforcement-Tests während der Testfahrten wird der Weg zu Stufe 5 der Autonomie geebnet.

Bild 6. Ein von der OpenAI DALL-E 2 KI-generiertes Bild eines selbstfahrenden Fahrzeugs, angetrieben durch den bewährten Commodore 64. Da ist noch was zu verbessern!

Gefährdet KI meinen Job?

In den sozialen Medien wird immer wieder die Frage diskutiert, ob die Fortschritte der KI Arbeitsplätze in der Kreativbranche gefährden. Die von OpenAI vorgestellte Software DALL-E 2 [12] verwandelt Anfragen in natürlicher Sprache in Bilder (**Bild 6**). Noch beeindruckender ist jedoch die Fähigkeit, vorhandene Bilder realistisch zu bearbeiten. So kann es beispielsweise Objekte im Vorder- oder Hintergrund entfernen. Und bei einem Werk des niederländischen Malers Vermeer kann die KI das Gemälde „Mädchen mit dem Perlenohrring" so erweitern, dass es einen glaubhaften Eindruck von dem Raum vermittelt, in dem es saß, als der Künstler es gemalt hat.

Schriftsteller und Journalisten wie die Elektor-Redaktion und andere renommierte Presseorgane waren jedoch schockiert über die Einführung von ChatGPT [13]. Diese KI ist in der Lage, mit den Nutzern im Plauderton und in einer Vielzahl von Sprachen zu interagieren. Bisher haben Diskussionen selbst über Nischenthemen wie den Vorzügen von Siliziumkarbid-MOSFETs (SiC) und Galliumnitrid-Transistoren (GaN) und deren Vorteilen gegenüber Silizium-MOSFETs zu äußerst präzisen Antworten geführt.

Das Tool ist zwar außerordentlich clever, kennt aber nur Antworten zu Themen, die vor dem Training aufgetreten sind. Da es nicht ständig lernt, ist es nicht auf dem Laufenden, was das aktuelle Zeitgeschehen oder die neusten Dramen einer K-Pop-Band angeht (wie schade). Ein weiterer Kritikpunkt ist, dass die Antworten nach einer Weile ein wenig einstudiert und formelhaft wirken. Wer jedoch nach Inspiration oder einem Geburtstagsgedicht im Stil eines Prominenten sucht, wird nicht enttäuscht sein.

Um herauszufinden, ob die Zukunft von Elektor von rosafarbenen, fleischigen Dingern oder von Computern abhängt, biete ich Ihnen eine Zusammenfassung des Themas dieses Artikels von ChatGPT:

„Zusammenfassend lässt sich sagen, dass eingebettete Systeme und KI zwei Technologien sind, die zunehmend zusammen eingesetzt werden, um intelligente, autonome Geräte und Systeme zu schaffen. Eingebettete Systeme bieten die Hardware- und Softwareplattform für KI-Algorithmen, während KI-Algorithmen diese Systeme in die Lage versetzen, ihre Umgebung intelligenter und menschenähnlicher wahrzunehmen, zu analysieren und darauf zu reagieren. In dem Maße, wie sich die Fähigkeiten von eingebetteten Systemen und KI weiter verbessern, können wir mit einer breiten Palette aufregender neuer Anwendungen in Bereichen wie Robotik, Gesundheitswesen, Transportwesen und anderen rechnen."

Bis zum nächsten Mal ... oder vielleicht auch nicht!

(220673-02)RG

Haben Sie Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Schicken Sie eine E-Mail an den Autor unter stuart.cording@elektor.com oder kontaktieren Sie Elektor unter redaktion@elektor.de.

WEBLINKS

[1] Edge Impulse Partner: https://bit.ly/3vbqRhG
[2] Edge Impulse CLI Installation: https://bit.ly/3BQQt71
[3] Edge Impulse Ingestion SDK: https://bit.ly/3jplhp3
[4] Beispielprojekt zur kontinuierlichen Bewegungserkennung: https://bit.ly/3WAV9G5
[5] EON Tuner: https://bit.ly/3PQhFsr
[6] SlateSafety BAND: https://bit.ly/3YVpe5x
[7] Monolith: https://bit.ly/3BWZImh
[8] DARPA Grand Challenge, Wikipedia: https://bit.ly/2l6bTFA
[9] J. Stilgoe, „Tesla crash report blames human error - this is a missed opportunity", Guardian, Januar 2017 : https://bit.ly/3Vjp7gI
[10] DYNACOR 40-36: https://bit.ly/3GdmgBS
[11] Waymo One: https://bit.ly/3FNG8Ku
[12] DALL·E 2: https://bit.ly/3PNPWsD
[13] ChatGPT: https://bit.ly/3PLjZRo

Digitalisierung der vertikalen Landwirtschaft

Von Miroslav Adamov und Adithya Madanahalli, Würth Elektronik eiSos

Vertikale Landwirtschaft ist ein vielversprechender Ansatz für die Ernährung einer wachsenden Weltbevölkerung. Optimierte Beleuchtung, Bewässerung, Düngung und Klimatisierung sorgen für höchste Erträge und beste Qualität. Dieser Artikel stellt einen IoT-vernetzten Prototyp vor.

Laut der UN-Ernährungs- und Landwirtschaftsorganisation (FAO) wird eine prognostizierte Weltbevölkerung von fast 10 Milliarden Menschen bis 2050 eine Steigerung der weltweiten landwirtschaftlichen Produktion um 50 Prozent erfordern. Ackerland und andere landwirtschaftliche Aktivitäten sind jedoch begrenzt. Die landwirtschaftliche Nutzfläche ist in den letzten Jahrzehnten sogar zurückgegangen, von fast 40 % der weltweiten Landfläche im Jahr 1991 auf nur 37 % im Jahr 2018 [1].

Weltweit steht im Jahr 2020 eine landwirtschaftliche Nutzfläche von schätzungsweise 4,7 Milliarden Hektar zur Verfügung, was etwa einem Drittel der gesamten Landfläche entspricht. Von der landwirtschaftlichen Nutzfläche entfällt wiederum etwa ein Drittel auf Ackerland (etwa 1,6 Milliarden Hektar) und die anderen zwei Drittel auf dauerhafte Wiesen und Weiden (etwa 3,2 Milliarden Hektar) [2]. 1,6 Milliarden Hektar sind 12 % der weltweiten Landfläche. Diese globalen Anteile haben sich seit dem Jahr 2000 nicht wesentlich verändert. Andererseits ist die Weltbevölkerung von 6,15 Milliarden auf 7,91 Milliarden im Jahr 2021 gestiegen, was einem Anstieg von fast 30 % entspricht, und sie nimmt weiter zu.

Innovative landwirtschaftliche Techniken wie die Landwirtschaft in kontrollierter Umgebung (manchmal auch als vertikale Landwirtschaft bezeichnet) sind vielversprechend als Mittel zur Steigerung der landwirtschaftlichen Produktion, um den lokalen oder regionalen Nahrungsmittelbedarf zu decken und Milliarden von Menschen mit frischeren und nahrhafteren Lebensmitteln zu versorgen, während gleichzeitig die mit der konventionellen Landwirtschaft verbundenen Umweltauswirkungen verringert werden. Digitale Farmen können nur dann erfolgreich sein, wenn mehrere Schlüsseltechnologien wie modernste Beleuchtung, Steuerung und Überwachung sowie regionale Akzeptanz und entsprechende Vorschriften wirksam eingesetzt werden.

In diesem Artikel soll erörtert werden, wie wichtig künstliche Beleuchtung sowie Fernüberwachung und -steuerung für das Pflanzenwachstum in vertikalen Farmen sind und wie Sie Ihre eigene vertikale Farm mit Hilfe von Rapid-Prototyping-Tools schnell und kostengünstig bauen können.

Was ist digitale Landwirtschaft?

„Unter digitaler Landwirtschaft versteht man die nahtlose Integration digitaler Techniken in die Bewirtschaftung von Nutzpflanzen und -tieren sowie in andere Prozesse in der Landwirtschaft. Für Landwirte bietet die digitale Landwirtschaft die Möglichkeit, die Produktion zu steigern, langfristig Kosten zu sparen und Risiken zu eliminieren.“ [3]

Mit diesem Ansatz lässt sich nicht nur die Produktion auf den derzeit verfügbaren landwirtschaftlichen Flächen steigern, sondern er kann auch in dicht besiedelten städtischen Gebieten, in denen der Platz knapp ist, erfolgreich eingesetzt werden. Die Hauptvorteile des Indoor-Farming liegen darin, dass man seine eigenen Pflanzen anbauen, die Umweltfaktoren steuern und überwachen, hochwertige Produkte erzeugen und die Kohlenstoffemissionen minimieren kann. Der offensichtlichste Vorteil der vertikalen Indoor-Landwirtschaft ist aber der geringe Platzbedarf. Vertikale Farmen sind in der Lage, die Produktivität drastisch zu erhöhen, indem sie mehr Pflanzen auf der gleichen Fläche anbauen können.

Die digitale vertikale Landwirtschaft birgt mehrere Herausforderungen. Der größte Nachteil der vertikalen Landwirtschaft sind die hohen Stromkosten, die für den Betrieb einer großen Anzahl von LEDs erforderlich sind. Anstelle von LEDs mit weißem Vollspektrumlicht können solche Typen verwendet werden, die nur die benötigten Farben erzeugen. Es ist jedoch eine große Herausforderung, in großem Maßstab optimale Lichtbedingungen zu schaffen und dabei den Energieverbrauch im Auge zu behalten.

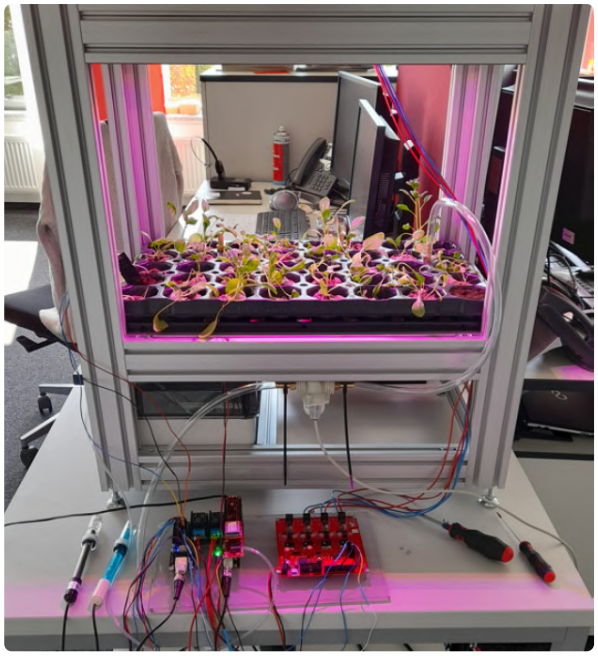


Bild 1. Prototyp einer Indoor-Farming-Box. (Quelle für alle Illustrationen: Würth Elektronik eiSos)

Außerdem ändern sich die für ein optimales Wachstum erforderlichen Umgebungsbedingungen je nach Pflanzenart und Wachstumsphase. Es ist äußerst schwierig, verschiedene Umweltfaktoren wie Temperatur, Luftfeuchtigkeit und Bodenfeuchte zu überwachen, zu steuern und sie dynamisch zu optimieren.

IoT zur Automatisierung der vertikalen Landwirtschaft

In diesem Artikel wird eine IoT-Lösung zur Bewältigung der oben genannten Herausforderungen und zur Erreichung einer vollständigen Automatisierung vorgeschlagen. Das Internet der Dinge (IoT) kann im weitesten Sinne als Oberbegriff für eine Reihe von Technologien definiert werden, die es Geräten ermöglicht, sich miteinander zu verbinden und zu interagieren. Vernetzte Geräte, die Daten erzeugen, ermöglichen eine Reihe neuer Anwendungen. Industrielle Automatisierung, Gesundheitswesen, intelligentes Zuhause, intelligente Städte, intelligente Netze und intelligente Landwirtschaft sind einige der Bereiche, in denen die IoT-Technologie erhebliche Vorteile bietet.

Das als „vierte industrielle Revolution“ oder „Industrie 4.0“ bezeichnete industrielle IoT (IIoT) betrifft die Digitalisierung von Anlagen und Prozessen, die Produkte, Maschinen, Dienstleistungen, Standorte mit Mitarbeitern, Managern, Lieferanten und Partnern verbindet. Die engere Vernetzung der digitalen mit der physischen Welt der Maschinen hilft, eine höhere Produktivität, mehr Sicherheit, Effizienz und Nachhaltigkeit zu erreichen.

Das IIoT schafft ein Universum von Sensoren, das ein beschleunigtes Deep Learning von bestehenden Abläufen ermöglicht, was eine schnelle Kontextualisierung, automatische Muster- und Trenderkennung erlaubt. Dies führt zu einer echten quantitativen Erfassung qualitativer Abläufe, was zu besserer Qualität, Effizienz, höherer Sicherheit, niedrigeren Kosten und vielen anderen Vorteilen führt.

Hier wird nicht nur eine IoT-basierte Lösung zur vollständigen Automatisierung und Steuerung von Beleuchtung, Bewässerung und anderen Umweltfaktoren einer vertikalen Farm vorgestellt, sondern dieses Konzept mit Rapid-Prototyping-Tools anhand eines vollautomatischen, vernetzten Systems mit Datenanalyse und Cloud-Konnektivität mit verschiedenen Konnektivitätslösungen für unterschiedliche Umgebungen in der Realität umgesetzt: **Bild 1** zeigt den Prototyp einer Indoor-Farming-Box.

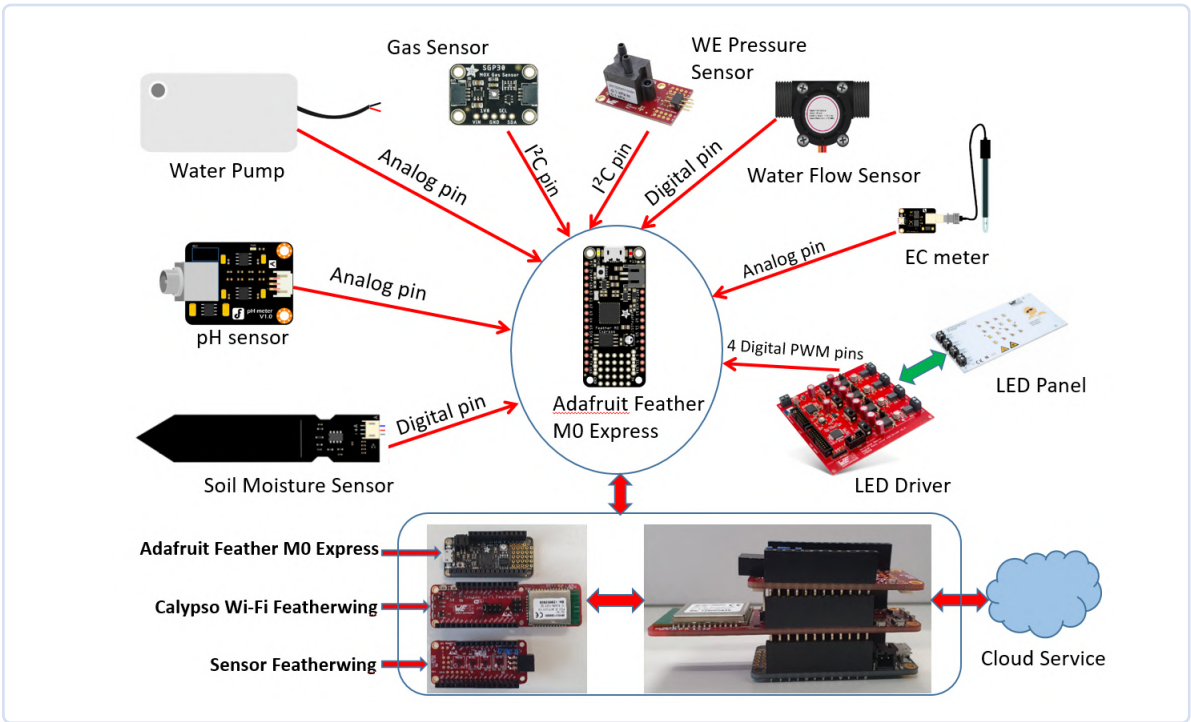


Bild 2. Prototyp des IoT-Systems auf Basis der Feather-Boards für vertikale Landwirtschaft.

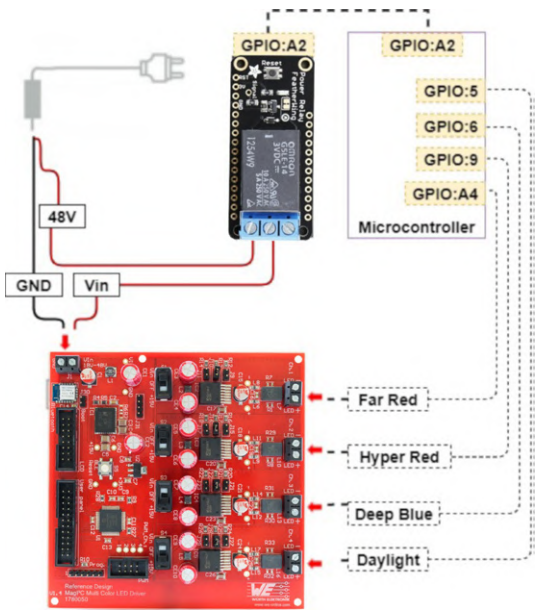


Bild 3. Das Beleuchtungssystem für Gartenbau besteht aus einem Array einfarbiger LEDs und einem LED-Treiber.

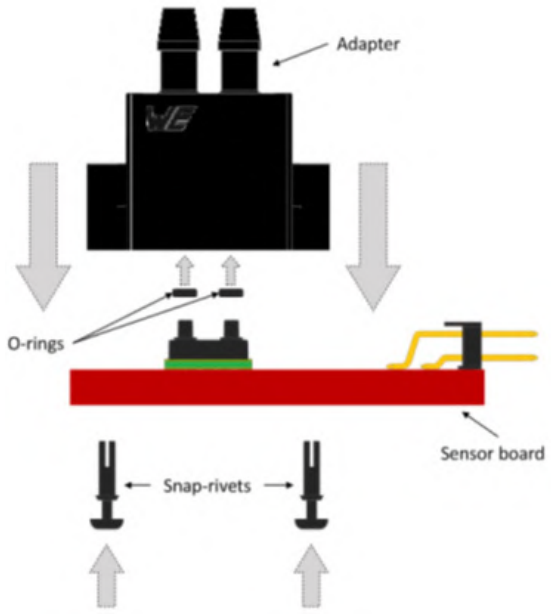


Bild 4. Der Differenzdrucksensor WSEN-PDUS ist ein sehr genauer, piezoresistiver Sensor auf MEMS-Basis.

„Feather"-basierte Implementierung

Die Kernaufgabe jeder IoT-Lösung besteht darin, Daten aus dem Feld in die Cloud zu übertragen, wo deren Analyse den gewünschten Mehrwert für die Anwendung generiert. Für die Realisierung einer solchen Anwendung wurde ein Open-Source-Hardware- und -Software-Ökosystem genutzt und eine komplette IoT-Lösung geschaffen. Würth Elektronik hat dazu das M0-Express-Feather-Board von Adafruit mit eigenen FeatherWings, Entwicklungsboards im Feather-Formfaktor mit Sensoren, Funk- und Power-Modulen, LEDs und LED-Treibern sowie diversen anderen Sensoren und Komponenten kombiniert (**Bild 2**) kombiniert.

Die Grundidee war, ein digitales vertikales Anbausystem zu schaffen, das einerseits das Wachstum, andererseits den Strom- und Wasserverbrauch zu optimiert.

Gesteuerte Bewässerung und Beleuchtung

Ein vertikales Anbausystem besteht aus einer Vertical-Farming-Konstruktion auf Basis von Erdsubstrat mit einem 4-Kanal-LED-Treiber und einem RGBW-LED-Design-Kit, mit dem auf einfache Weise RGBW-Farben für verschiedene Beleuchtungssituationen gemischt werden können, so dass das Wachstum der Pflanzen im Gartenbaupanel gefördert oder sogar eine Innenbeleuchtung auf Grundlage eines Human Centric Lighting-Konzepts (HCL) ermöglicht wird. Die Tröpfchen-Bewässerung und der Wassertank verwenden recyceltes Wasser, dessen pH- und EC-Werte gemessen, gesteuert und über verschiedene Konnektivitätsmodule von Würth Elektronik in die Cloud übertragen werden.

Ein Bodenfeuchtesensor überwacht die Bodenfeuchtigkeit und bei Bedarf wird Wasser mit einer kleinen Pumpe in das System gepumpt. Der Rest des Wassers wird aufgefangen, gefiltert und in den Tank zurückgeführt. Das Herzstück des Systems ist Adafruits Feather-M0-Express-Board mit dem Power-Relais. FeatherWings werden als Schalter verwendet. Die Daten werden an die Cloud gesendet, wo die Informationen verarbeitet, analysiert und zur Steuerung der Farm verwendet werden.

Adaptives Beleuchtungssystem

Das Beleuchtungssystem (**Bild 3**) besteht aus zwei verschiedenen Teilen, einem Gartenbau-LED-Panel mit vier separaten Kanälen mit speziellen einfarbigen Pflanzen-LEDs und dem MagI³C-Multicolor-LED-Treiber, beide Teil des WE-Beleuchtungsentwicklungskits [4]. Das Kit bietet eine einfache Lösung, um das Wachstum von Pflanzen zu verstärken. Mit dem LED-Treiber mit Step-Down-Leistungsmodul MagI³C ist es möglich, die Intensität und Farbe jedes der vier LED-Stränge individuell einzustellen, um den Anwendungsanforderungen gerecht zu werden und die LED-Werte auf das Pflanzenprofil einzustellen. Das Gartenbaupanel besteht aus sechs hyperroten, vier fernroten, zwei tiefblauen und vier weißen Keramik-LEDs. Das System kann über Bluetooth oder über die Cloud per WLAN oder Mobilfunkverbindung gesteuert werden. Diese Lösung stellt eine allgemeinere Cloud-Lösung dar.

Bewässerungsanlage

Der Füllstand des Wassertanks wird mit dem Differenzdrucksensor WSEN-PDUS gemessen. Dabei handelt es sich um einen sehr genauen piezoresistiven Sensor auf MEMS-Basis (**Bild 4**). Zusätzlich drückt eine 12-V-Pumpe das nährstoffreiche Wasser über einen Durchflusssensor in das Tropfbehältnis. Die Wasserqualität wird mit einem pH- und EC-Messgerät DFR-05874 beziehungsweise DFR0300 aus der Gravity-Reihe von DFRobot überwacht.

Die meisten natürlichen Gewässer weisen einen pH-Wert zwischen 5 und 8 auf. Der allgemein akzeptierte pH-Wert für Bewässerungswasser liegt zwischen 5,5 und 7,5, aber die besten Ergebnisse wurden mit pH-Werten zwischen 5,5 und 7 erzielt. Wasser in diesem pH-Bereich hält das Nährstoffgleichgewicht aufrecht, bietet eine wirksame chemische Desinfektion und verhindert Kalkablagerungen in Bewässerungsanlagen [5].

„Fertigation" ist ein Kofferwort aus den Begriffen Düngung (Fertilization) und Bewässerung (Irrigation) Aufgrund der geringen Substratoberfläche erfordert der Anbau in Substrat eine gute Bewässerung

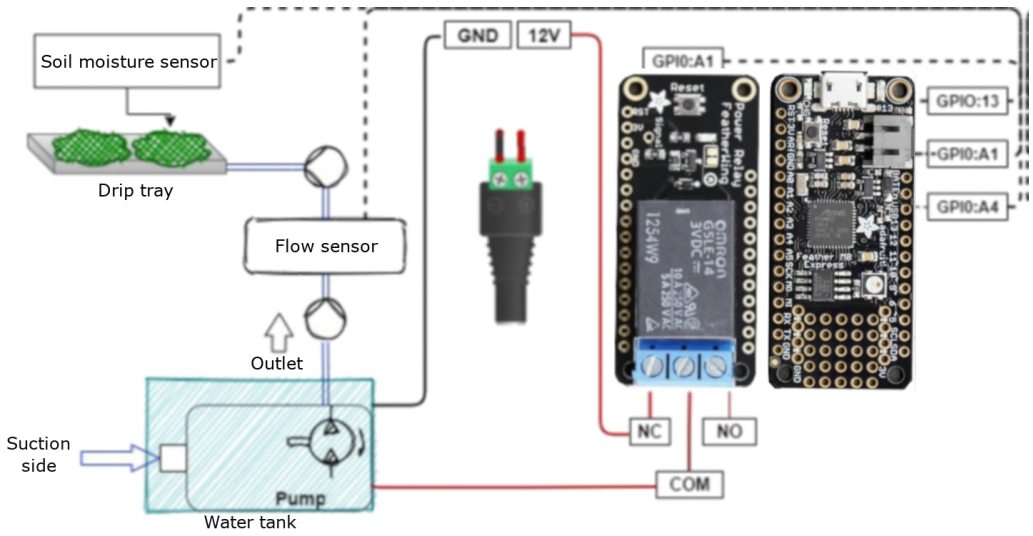


Bild 5. Bewässerungssystem mit Sensoren, Wassertank, Pumpe und Steuerplatinen.

und viel Dünger, der dem Bewässerungswasser zugesetzt wird. Mit einem EC-Messgerät wird eine Unter- oder Überversorgung mit Düngemitteln im Bewässerungswasser festgestellt. Die elektrische Leitfähigkeit (EC) ist ein Maß für die Konzentration von Ionen im Wasser und dient der Messung der Fähigkeit des Wassers, elektrischen Strom zu leiten. Je reiner das Wasser, desto geringer ist die Leitfähigkeit.

Für die Messung der Bodenfeuchtigkeit wurde der kapazitive STEMMA-Bodensensor von Adafruit verwendet. Bei kapazitiven Messungen wird nur eine Sonde verwendet, es gibt kein freiliegendes Metall, das oxidieren kann, und es werden keine Gleichströme in Pflanzen eingeleitet. Die Pumpe wird automatisch und direkt über die Cloud gesteuert. Ihre Betriebszeit wird auf der Grundlage der Anzahl der Pflanzen, der erforderlichen Bodenfeuchtigkeit und des Pumpendurchsatzes berechnet (**Bild 5**).

Erfassung der Umweltdaten

Sensoren messen den Zustand der Umwelt und interpretieren diesen als analoge oder digitale Daten. Auf der anderen Seite bewirken Aktoren eine physikalische Veränderung der gemessenen Umgebung. Die Fortschritte im Bereich der Elektronik im Allgemeinen und der Halbleiter im Besonderen haben dazu geführt, dass eine breite Palette von Sensoren und Aktoren zur Verfügung steht, die hocheffizient und dennoch sehr kompakt sind.

Kohlendioxid wird in großem Umfang in Anreicherungs- und Extraktionsprozessen eingesetzt. Dieser Prozess beschleunigt nicht nur das Wachstum von Pflanzen, sondern kann in den meisten Fällen auch als Begasungsmittel eingesetzt werden. Bei der Anreicherung wird in der Regel ein CO_2-Gehalt von 800...1500 ppm verwendet, um das Wachstum um 20...30 % zu beschleunigen. Die Erhöhung des Gesamtstoffwechsels hilft den Pflanzen, den Auswirkungen von Hitze zu widerstehen. Größere, gesündere und robustere Pflanzen vertragen extreme Umwelteinflüsse besser. Ein erhöhter Pflanzenstoffwechsel bedeutet jedoch zusätzliche Anforderungen an alle Beteiligten. Die Pflanzen benötigen nicht nur mehr Wasser und Nährstoffe, sondern auch eine zusätzliche Belüftung.

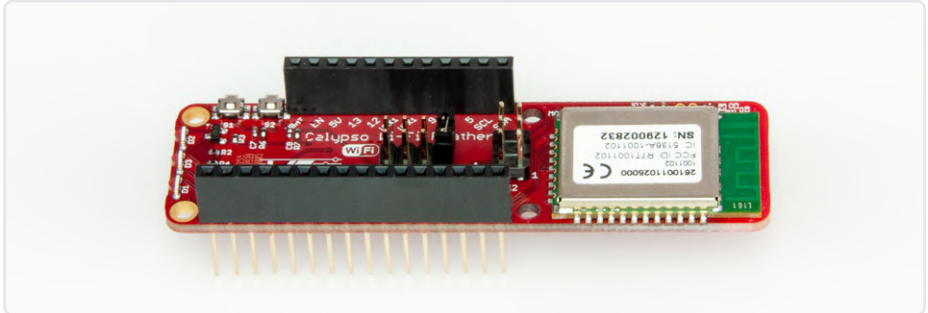


Bild 6. Der Calypso-FeatherWing bietet Konnektivität in Umgebungen, in denen ein WLAN verfügbar ist.

Der CO_2-Gehalt des Systems wurde mit dem Luftqualitätssensor SPG30 von Adafruit, der an den FeatherWing-Sensor von Würth Elektronik angeschlossen ist, gemessen und online überwacht. Es ist zu beachten, dass der Sensor den äquivalenten CO_2-Gehalt misst. Dieser eCO2-Wert wird auf Basis der H_2-Konzentration berechnet, es handelt sich also nicht um einen „echten" CO_2-Sensor für den Laborgebrauch. Der nächste Schritt wird die Anreicherung sein.

Es gibt mehrere Wachstumsstadien von Pflanzen, und in jedem Stadium benötigt eine Pflanze andere Licht- und Wärmebedingungen. Die meisten Pflanzen tolerieren normale Temperaturschwankungen, und die optimalen thermischen Bedingungen für das Pflanzenwachstum können nicht nur zwischen den einzelnen Phasen, sondern auch im Laufe des Tages variieren. Die Optimierung kann durch Messung der Temperatur und Optimierung der Tageszyklen erfolgen. Wenn das Licht ausgeschaltet ist, sollte die Temperatur um einige Grad niedriger sein.

Die Luftfeuchte wird als relative Luftfeuchtigkeit angegeben. Verschiedene Pflanzenstadien benötigen unterschiedliche Feuchtigkeitsniveaus. Eine zu feuchte Umgebung kann das Potenzial für die Ausbreitung von Krankheiten erhöhen. Sowohl die Luftfeuchtigkeit als auch die Temperatur werden mit dem WE-Sensor FeatherWing überwacht.

IoT-Konnektivität

Sensoren und Aktoren sind in der Regel in Geräten installiert, die nur begrenzt mit der digitalen Welt verbunden sind. Drahtlose Konnektivität bietet neben vielen anderen Vorteilen auch die für solche Anwendungen

notwendige Konnektivität. Heutzutage gibt es dazu eine Vielzahl von standardisierten und proprietären drahtlosen Lösungen, deren Auswahl von einer Reihe von Faktoren wie Reichweite, Durchsatz, Frequenzband, lokale gesetzliche Anforderungen und Energiebudget bestimmt wird.

Die Konnektivität wird über zwei verschiedene Ansätze realisiert, zum einen mit einem WLAN-FeatherWing Calypso (**Bild 6**) für Umgebungen, in denen ein WLAN verfügbar ist, zum anderen mit einem Adrastea-I-FeatherWing (**Bild 7**) für Umgebungen ohne WLAN. Beide Boards sind mit dem Rest des Systems über das M0-Express-Feather-Board von Adafruit verbunden.

Das Calypso-Board ist ein kompaktes WLAN-Funkmodul, das auf dem WiFi-Standard IEEE 802.11 b/g/n (2,4 GHz) basiert. Es verfügt über einen integrierten TCP/IP-Stack und ein sofort einsatzbereites MQTT-Protokoll. Das Adrastea-I-Modul ist ein kompaktes LTE-M/NB-IoT-Mobilfunkmodul mit integriertem GNSS und einem ARM-Cortex-M4-Prozessor, das für jede IoT-Anwendung geeignet ist.

Beide Module lassen sich einfach und sicher mit jeder Cloud verbinden. In diesem Fall fiel die Entscheidung auf einen externen M0-Prozessor und Microsoft IoT Central, eine Platform-as-a-Service, wegen ihrer Einfachheit und Benutzerfreundlichkeit. IoT Central verfügt über eine gebrauchsfertige Benutzeroberfläche und eine API für die Verbindung, Verwaltung und den Betrieb von IoT-Geräten. Mit seiner Telemetrie, seinen Eigenschaften und Befehlen wurde es zur Überwachung und Steuerung aller Aspekte des vertikalen Anbausystems verwendet.

KI-gestützter Ansatz

Darüber hinaus kann KI eingesetzt werden, um Muster zu erkennen und Pflanzenprofile in der Cloud zu erstellen, mit denen LED-Werte, Temperatur, Feuchtigkeit und andere Parameter automatisch eingestellt werden können, um optimale Bedingungen zu schaffen.

Vertikale Landwirtschaft allein wird zwar die Welt nicht vollständig ernähren können, aber sie wird einer größeren Anzahl von Menschen mehr frische Produkte liefern. Mit diesem Ansatz wurde eine Miniaturversion eines vertikalen Anbausystems demonstriert, das „normale" Menschen in ihre Küchen stellen können. Neu ist das nicht; früher nannte man solche Systeme Gewächshaus für Fensterbänke.

(230077-02)RG

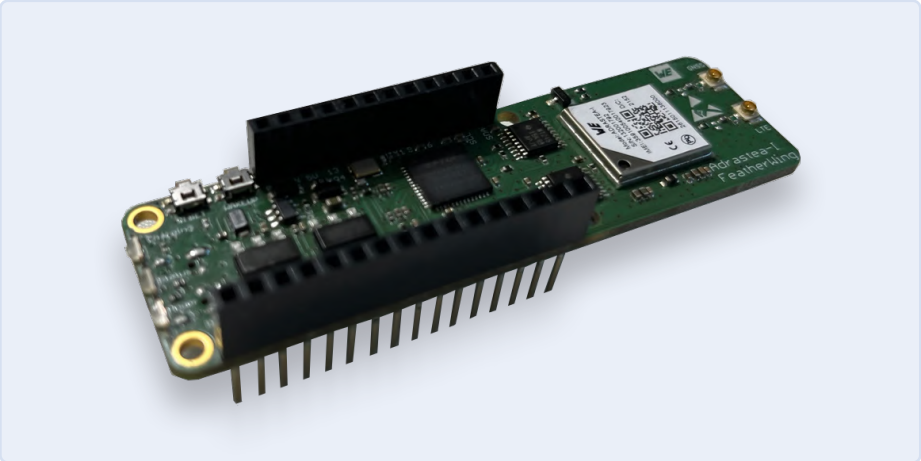


Bild 7. Das Adrastea-I-Modul ist ein kompaktes LTE-M/NB-IoT-Mobilfunkmodul mit integriertem GNSS und einem ARM-Cortex-M4-Prozessor.

Über die Autoren

Miroslav Adamov studierte Physik und Informatik an der Universität von Belgrad, Serbien. Danach setzte er seine wissenschaftliche Arbeit an der TU Berlin, dem WIAS Berlin, der FAU Erlangen/Nürnberg und dem Center of Private Equity Research in München fort. Nach einigen Jahren in der quantitativen Finanzwirtschaft kam er im Jahr 2015 als Senior Business Analyst zu Würth Elektronik. Seine Position als Senior IoT Solution Architect übernahm er im Jahr 2017 mit dem Fokus auf Konzeption und Implementierung von industriellen IoT Lösungen.

Adithya Madanahalli schloss sein Studium an der Technischen Universität München mit einem MSc. in Communication's Engineering ab. Anschließend arbeitete er mehrere Jahre als Software-Ingenieur im Bereich der drahtlosen Konnektivität und Sensoren. Seit 2022 ist Adithya Madanahalli als IoT-Ingenieur bei Würth Elektronik eiSos im Geschäftsbereich Wireless Connectivity and Sensors tätig. Dort ist er spezialisiert auf den Entwurf und die Entwicklung von IoT-Lösungen mit den Schwerpunkten Hardware, Embedded Software und End-to-End-Sicherheit.

WEBLINKS

[1] „The future of food and agriculture: Trends and challenges", ein Bericht der Ernährungs- und Landwirtschaftsorganisation der Vereinten Nationen (PDF), 2017: https://fao.org/3/a-i6583e.pdf

[2] „Land statistics and indicators – Global, regional and country trends, 2000-2020", Faostat Analytical Brief 48 (PDF): https://fao.org/3/cc0963en/cc0963en.pdf

[3] Ambrose, E.: „Digital agriculture: Why the future is now": https://ag.purdue.edu/stories/digital-agriculture-why-the-future-is-now/

[4] Würth Elektronik Lighting Development Kit: https://www.we-online.com/de/components/products/LIGHTING_DEVELOPMENT_KIT

[5] Brunton, V.: „Irrigation water quality" (PDF): https://dpi.nsw.gov.au/__data/assets/pdf_file/0005/433643/Irrigation-water-quality.pdf

Elektor Infografik

Embedded und KI, heute und morgen

Trotz der zahllosen Herausforderungen der COVID-19-Pandemie und der jüngsten geopolitischen Ereignisse gibt es für Führungskräfte, Unternehmer, professionelle Ingenieure und Studenten, die sich mit eingebetteten Technologien und Lösungen im Zusammenhang mit künstlicher Intelligenz befassen, ein breites Spektrum an interessanten Geschäftsmöglichkeiten. Werfen wir einen Blick auf einige Fakten und Zahlen.

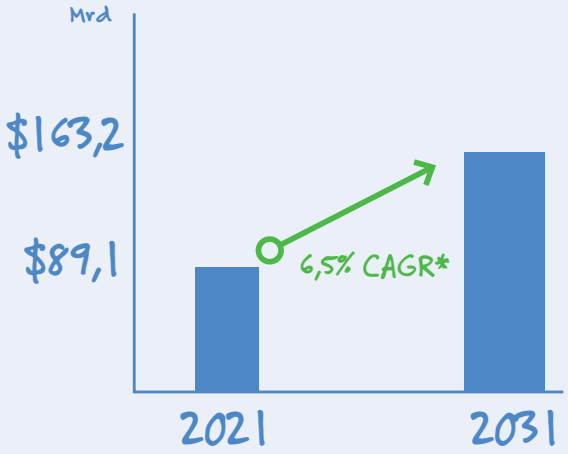
Eingebettete Systeme: Ein Blick in die Zukunft

Sie fragen sich, wie die Zukunft der Branche der eingebetteten Systeme aussehen wird? Wachstum scheint angesagt zu sein! Laut einem aktuellen Bericht von Allied Market Research ist die Branche auf dem besten Weg, bis in das Jahr 2031 ein Volumen von 163 Milliarden Dollar zu erreichen, wobei der asiatisch-pazifische Raum die Nase vorn haben wird. [1]

Unternehmen im Visier

- Advantech Corp.
- Analog Devices
- Infineon Technologies
- Intel Corp.
- Microchip Technology
- NXP Semiconductors
- Qualcomm
- Renesas Electronics
- STMicroelectronics
- Texas Instruments

(Quelle: Allied Market Research)



** Compound Annual Growth Rate = durchschnittliche jährliche Wachstumsrate*

Künstliche Intelligenz im Fokus

20%

Ungefähr einer von fünf Doktoranden der Informatik im Jahr 2020 spezialisierte sich auf KI/ML.

93,5 Mrd. $

Die privaten Investitionen in KI erreichten 2021 rund 93,5 Mrd. $ und waren damit mehr als doppelt so hoch wie im Jahr 2020.

Geht es nach dem globalen Marktforschungsunternehmen IDC, wird der weltweite Markt für künstliche Intelligenz - Software, Hardware und Dienstleistungen - im Zeitraum 2022 bis 2026 eine durchschnittliche jährliche Wachstumsrate von 18,6 % erzielen und im Jahr 2026 ein Volumen von 900 Milliarden US-Dollar erreichen. [2] Möglichkeiten für Firmen, professionelle Ingenieure, Unternehmer und sogar Maker gibt es im Überfluss. Sehen Sie sich diese Schlüsseldaten aus einem Bericht der Stanford University aus dem Jahr 2022 an. [3] Das Wachstum der KI ist ziemlich eindeutig.

Nordamerika

Von 2010 bis 2021 wurden 56,96 % der erteilten KI-Patente in Nordamerika angemeldet. 31,09 % kamen aus Asien und dem Pazifikraum, 11,27 % kamen aus Europa und Zentralasien.

2x

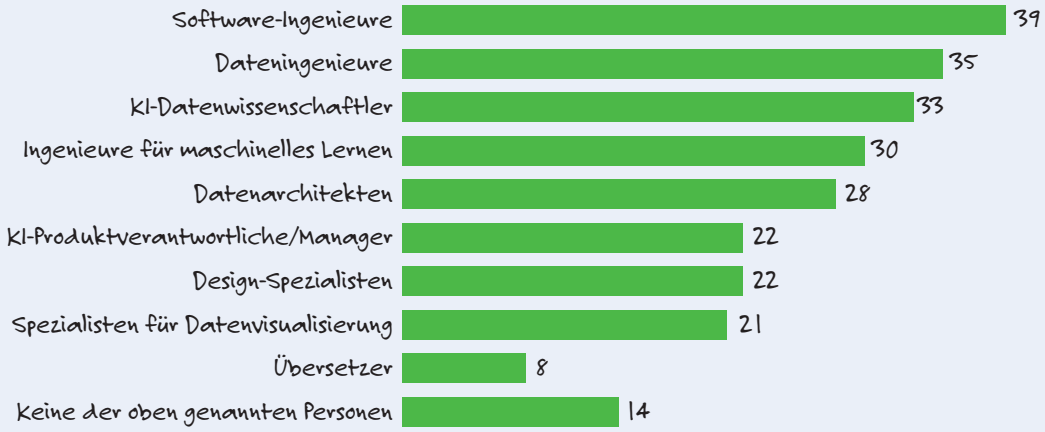
Die Zahl der (englischsprachigen) KI-bezogenen Veröffentlichungen in Zeitschriften, Büchern, Dissertationen, Konferenzen verdoppelte sich (weltweit) von 162.444 im Jahr 2010 auf 334.497 im Jahr 2021.

30x

Die Zahl der im Jahr 2021 angemeldeten KI-Patente war mehr als 30 Mal höher als im Jahr 2015. Das bedeutet eine jährliche Wachstumsrate von 76,9 %.

Wo sind die Jobs in der KI?

KI-bezogene Funktionen, die die befragten Unternehmen im vergangenen Jahr eingestellt haben

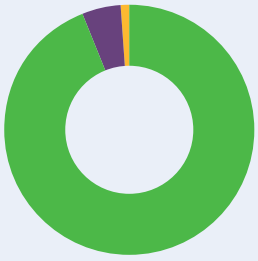


(Quelle: www.mckinsey.com)

Software-Ingenieure haben eine glänzende Zukunft! Laut dem aktuellen Bericht „State of AI" von McKinsey wurden Software-Ingenieure im Bereich KI in den letzten Monaten am meisten gesucht und eingestellt. Dateningenieure und KI-Datenwissenschaftler belegten den zweiten und dritten Platz. [4] Dies ist ein weiteres deutliches Zeichen dafür, dass viele Unternehmen nicht mehr nur mit KI experimentieren, sondern sie aktiv in Unternehmensanwendungen einbinden.

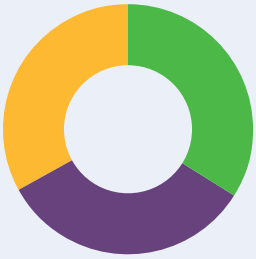
Was denken Wirtschaftskapitäne über KI?

Wir alle wissen inzwischen, dass künstliche Intelligenz ein großes Geschäft bedeutet. Aber möchten Sie auch wissen, wie Führungskräfte aus der Wirtschaft über KI denken? Auch wenn das Interesse je nach Branche variiert, sind KI-Lösungen auf jeden Fall ein Gesprächsthema unter Wirtschaftsführern auf Konferenzen, bei Vorstandssitzungen und auf Betriebsausflügen. Die folgenden Daten von Deloitte bieten einige Einblicke. [5]



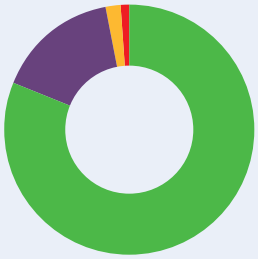
Bedeutung von KI für den Erfolg eines Unternehmens

- 94 % Sehr wichtig oder wichtig
- 5 % Eher wichtig
- 1 % Nicht wichtig



Was sind die drei größten Herausforderungen beim Start und der Durchführung von Projekten?

- 30 % Fehlende Finanzierung für KI-Technologien
- 29 % Auswahl der geeigneten KI-Technologien
- 29 % Unzureichende technische Fähigkeiten



Wird KI-Technologie Arbeitsleistung und Arbeitszufriedenheit verbessern?

- 82 % Zustimmung
- 16 % Weder Zustimmung noch Ablehnung
- 2 % Ablehnung
- 1 % Weiß nicht

WEBLINKS

[1]] CISION, „Embedded Systems Market to Reach $163.2 Billion by 2031, Globally, by 2031 at 6.5% CAGR", Allied Market Research, 2022: https://bit.ly/embedded-outlook-AMR
[2] IDC, „IDC Forecasts 18.6% Compound Annual Growth for the Artificial Intelligence Market in 2022-2026": https://bit.ly/IDC-AI-2022-2026
[3] D. Zhang, et al, „The AI Index 2022 Annual Report", HAI, Stanford University, März 2022: https://bit.ly/AI-index-rep-22
[4] McKinsey, „The State of AI in 2022 — And a Half Decade in Review", QuantumBlack, 6. Dez. 2022: https://bit.ly/mckinsey-AI-22
[5] Deloitte, „Fueling the AI transformation", Oktober 2022: https://bit.ly/deloitte-ai-state

Einführung in TinyML

Von Mark Patrick (Mouser Electronics)

In diesem Artikel werden Konzepte des maschinellen Lernens unter Verwendung von ressourcenbeschränkten Mikrocontrollern mit geringem Stromverbrauch untersucht, die unter dem Begriff „TinyML“ zusammengefasst werden. Das maschinelle Lernen prägt weiterhin viele Aspekte unseres täglichen Lebens, ob zu Hause, im Büro oder anderswo. Während viele ML-Applikationen eine beträchtliche Rechenleistung benötigen, um komplexe wissenschaftliche oder finanzielle Daten zu verarbeiten, bieten diejenigen, die für das Internet der Dinge (IoT) und andere Edge-basierte Applikationen entwickelt wurden, nur magere Rechen- und Konnektivitätskapazitäten.

KI und maschinelles Lernen: Bereits in unser Leben eingebettet

Mit unserer Armbanduhr oder einem Star-Trek-Gadget zu sprechen, war vor nicht allzu langer Zeit noch die Idee von Science-Fiction-Autoren – doch heute machen viele von uns genau das regelmäßig. Wir sprechen mit unseren Smartphone-Apps, mit dem Infotainment-System im Auto und mit unseren Smart Speakern zu Hause.

Künstliche Intelligenz (KI), die Wissenschaft von der Entwicklung von Computern, die denken, wahrnehmen, erkennen und Probleme lösen können, ist zur Grundlage der heutigen Computer- und Datenwissenschaft geworden. Das maschinelle Lernen (ML), ein Anwendungsbereich der KI, konzentriert sich auf die Verwendung von Algorithmen, die es Computern ermöglichen, zu lernen und zu verbessern, wie sie eine Aufgabe angehen, ohne dass sie explizit dafür programmiert wurden.

Viele Bereiche unserer Welt werden bereits heute durch maschinelles Lernen geprägt, von der Wettervorhersage über die Suche nach der besten Route für den täglichen Arbeitsweg bis hin zu den Werbebannern in Ihrer bevorzugten Social-Media-App. Auch die wissenschaftliche Forschung ist heute zunehmend auf maschinelles Lernen angewiesen, um Petabytes von Daten zu durchforsten und nach Trends zu suchen.

Wie maschinelles Lernen funktioniert

Bei den oben genannten Beispielen für maschinelles Lernen sehen wir in der Regel nur die Ergebnisse. Die Komplexität der Suche nach einem neuen schwarzen Loch oder die Anzahl der Permutationen vergangener Wetterereignisse, die zur Bestimmung der heutigen Wettervorhersage verwendet wird, bekommen wir normalerweise nicht zu sehen. Viele der komplexen Aufgaben erfordern große Datensätze, mehrere Algorithmuskandidaten und erhebliche Rechenleistung. Wenn Sie sich eingehender mit maschinellem Lernen befassen, werden Sie verschiedene Lernkategorien finden, die sich jeweils für bestimmte Aufgaben eignen. Wir werden hier nicht näher auf sie eingehen, aber es handelt sich um überwachtes, unbeaufsichtigtes, halbüberwachtes und verstärkendes Lernen.

Ein neuronales Netzwerk ist ein wesentlicher Bestandteil jeder ML-Applikation. Im Allgemeinen ist ein neuronales Netzwerk ein Versuch, die Funktion der Neuronen im menschlichen Gehirn in einem mathematischen Modell nachzubilden. Das Modell verwendet Algorithmen, um die Wahrscheinlichkeiten eines Ergebnisses abzuleiten; bei einer Bilderkennungsaufgabe beträgt beispielsweise die Wahrscheinlichkeit, dass es sich bei dem Bild um einen Hund handelt, 95%. Es gibt verschiedene Arten von neuronalen Netzwerken, und Sie werden Begriffen wie „Convolutional Neural Networks“ (CNN) und „Recurrent Neural Networks“ (RNN) begegnen. Jede Art von neuronalem Netzwerk verfügt über einen unterschiedlichen Satz miteinander verbundener Schichten, wodurch es sich besser für bestimmte Aufgaben eignet.

Beim überwachten Lernen werden dem neuronalen Netzwerk zahlreiche Trainingsdaten übermittelt, damit der Algorithmus ein Ergebnis ableiten kann. Für Bilderkennungsaufgaben eignet sich zum Beispiel

ein CNN am besten. Um verschiedene Obstsorten zu erkennen, muss man den Algorithmus des neuronalen Netzwerks mit Tausenden von gekennzeichneten Bildern verschiedener Obstsorten „füttern“, die aus verschiedenen Winkeln und mit unterschiedlichen Reifegraden aufgenommen wurden. Der Algorithmus sucht dann nach erkennbaren Merkmalen, die ihm helfen, eine Obstsorte von einer anderen zu unterscheiden. Die Trainingsphase ist iterativ und kann die Optimierung des Algorithmus beinhalten, um die höchsten Wahrscheinlichkeiten zu erreichen, wenn er mit einem Testdatensatz von Bildern konfrontiert wird. Sobald die Testdaten die beste Leistung des neuronalen Netzwerkalgorithmus erreichen, ist das Modell einsatzbereit. Bei der Anwendung, auch Inferenz genannt, leitet das Modell die Ergebnisse auf der Grundlage einer Wahrscheinlichkeitsbasis ab.

Für die Interaktion mit einem Smart Speaker ist üblicherweise ein Auslösewort oder ein Satz wie „Hey, Google“ nötig, sodass er aufwacht und beginnt, zuzuhören, was als Nächstes kommt. Ein Smart Speaker verfügt nicht über die Rechenkapazitäten eines Rechenzentrums, deshalb werden kurze Audiodateien aufgezeichnet und in die Cloud gestreamt, um die Art unserer Anfrage zu ermitteln. Die Erkennung des Auslösewortes oder Satzes ist ein hervorragendes Beispiel für einfaches maschinelles Lernen; willkommen bei TinyML!

Maschinelles Lernen und das industrielle IoT

Da maschinelles Lernen allgegenwärtig wird, wächst die Liste der möglichen Applikationen exponentiell. Bei vielen industriellen Applikationen geht es beispielsweise nicht um „Big Data“, sondern darum, wie Produktionslinien effizienter gestaltet werden können. Die Kosten eines unerwarteten Ausfalls in der Produktion können hoch sein. Fällt etwa der Motor beim Mischen von gekühlten Lebensmitteln aus, kann es zum Stillstand der Produktion und zum Verlust von Rohstoffen kommen. Viele Unternehmen setzen daher jetzt auf eine vorausschauende Wartung, um geplante Ausfallzeiten zu terminieren und solche Ausfälle zu vermeiden. Die zustandsbasierte Überwachung von Produktionsanlagen wie Motoren und Aktuatoren hilft bei der Vorhersage, wann zum Beispiel ein Motorlager Anzeichen von übermäßigem Verschleiß zeigt. Algorithmen des maschinellen Lernens, die in IIoT-Edge-Sensorgeräten eingesetzt werden, können erkennen, wenn die Schwingungssignatur eines Motors von der Norm abweicht, und so eine rechtzeitige Warnung ausgeben.

Allein im industriellen Bereich sind die Anwendungsmöglichkeiten für maschinelles Lernen riesig, aber damit sind auch einige technische Herausforderungen verbunden. Im Gegensatz zu den „Big Data“-Applikationen sind die Rechen- und Speicherressourcen eines einfachen IIoT-Edge-Sensors nur ein Bruchteil dessen, was in einem Rechenzentrum verfügbar ist. Eine einzige Fabrik kann Hunderte von Sensoren haben, sodass die Größenvorteile ein Faktor sind, ebenso wie die physische Größe, die Verfügbarkeit einer geeigneten Stromversorgung und die drahtgebundene oder drahtlose Konnektivität. In den meisten Fällen ist der Algorithmus des neuronalen Netzwerks eines Vibrationssensors jedoch weniger anspruchsvoll als die Deep Space-Forschung, sodass Embedded-Entwickler nach Möglichkeiten suchen, um Netzmodelle auf batteriebetriebenen Mikrocontrollern mit geringem Stromverbrauch auszuführen.

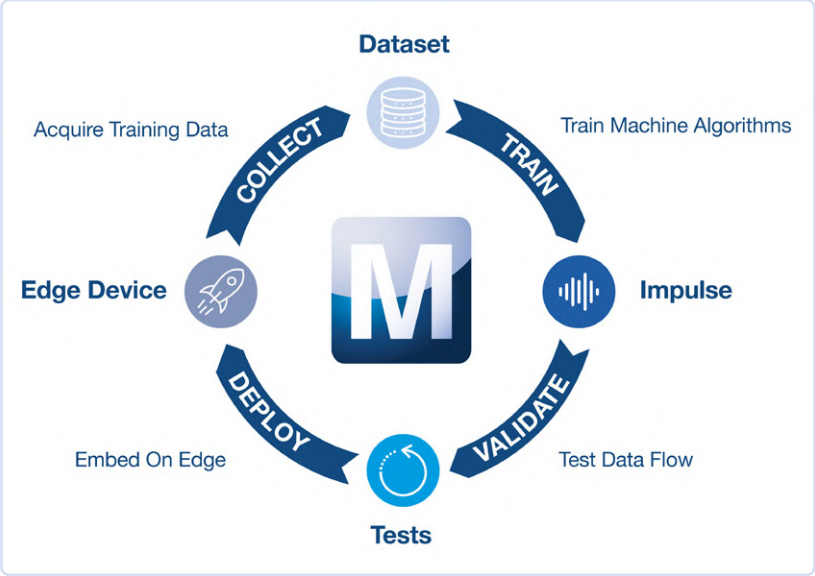


Bild 1: Trainings- und Inferenzschritte.

TinyML: Maschinelles Lernen an der Edge

Die zum Trainieren eines neuronalen Netzwerkmodells verwendete Computerplattform muss nicht dieselbe sein wie die Einsatzplattform, wodurch einer der ressourcenintensivsten Prozesse entfällt. Allerdings gibt es immer noch technische Herausforderungen. Kann der Algorithmus zeitnah ausgeführt werden? Wie wird das Gerät mit einem Hostsystem kommunizieren, um das Betriebspersonal zu benachrichtigen? Embedded-Entwickler sind ebenfalls mit Herausforderungen konfrontiert. Die meisten sind keine Datenwissenschaftler, weshalb das Erlernen der Konzepte des maschinellen Lernens und der Arbeit mit neuronalen Netzwerken eine steile Lernkurve darstellt.

230062-02

Über den Autor

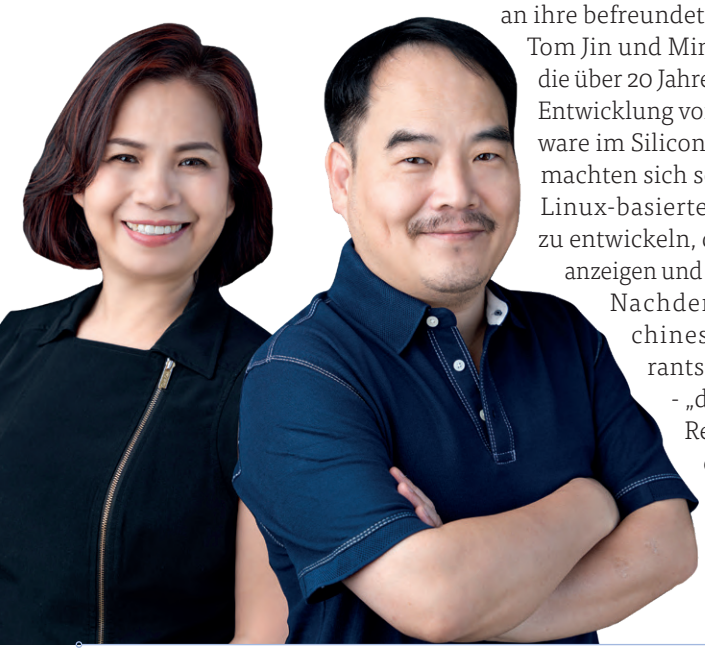
Als Technical Marketing Manager für EMEA bei Mouser Electronics ist Mark Patrick für die Erstellung und Verbreitung von technischen Inhalten in der Region verantwortlich - Inhalte, die für Mousers Strategie zur Unterstützung, Information und Inspiration der Elektronik-Branche von zentraler Bedeutung sind.

Bevor er das Technische Marketing-Team leitete, war Patrick Teil des EMEA-Lieferanten-Marketing-Teams und spielte eine wichtige Rolle beim Aufbau und der Entwicklung von Beziehungen zu wichtigen Produktionspartnern. Zusätzlich zu einer Vielzahl von technischen und Marketing-Positionen war Patrick acht Jahre lang bei Texas Instruments in den Bereichen Anwendungsunterstützung und technischer Vertrieb tätig.

Kassenknüller

Zufriedene Kunden: Das Raspberry Pi-basierte Kassensystem KwickPOS ist ein Hit in Restaurants und Einzelhandel.

Das in Houston ansässige Unternehmen KwickPOS hatte schon seinen ersten Kunden noch vor der Aufnahme seiner Geschäftstätigkeit im Jahr 2003: Die Besitzer eines chinesischen Restaurants, die auf der Suche nach einer eleganten Methode zur Abwicklung von Kundenbestellungen waren, wandten sich an ihre befreundeten IT-Entwickler Tom Jin und Ming Ye. Jin und Ye, die über 20 Jahre Erfahrung in der Entwicklung von Restaurantsoftware im Silicon Valley verfügen, machten sich schnell daran, ein Linux-basiertes Kassensystem zu entwickeln, das Bestellungen anzeigen und verarbeiten kann. Nachdem Ming Ye mit chinesischen Restaurants begonnen hatte - „die schwierigsten Restaurants, wenn es um die Speisekarte und das organisatorische Feedback im Restaurant geht“ -, sprach sich das Konzept bald bei anderen Gastronomen herum. Die Nachfrage war groß genug, um aus dem Konzept ein Geschäft zu machen.



Die Herausforderung

Die Umwandlung eines erfolgreichen und gefragten Kassensystems in ein vollwertiges Unternehmen war für Jin und Ye angesichts ihrer bisherigen Karrieren im Silicon Valley ein großer Sprung ins Ungewisse. Trotzdem beschlossen sie 2015, nachdem sie eine Reihe von Systemen gebaut hatten, KwickPOS in ein „echtes Unternehmen“ zu verwandeln und POS-Systeme von ihrem Hauptsitz aus in Houston, Texas, zu entwickeln und zu verkaufen. Nachdem sie das POS-Produkt um Backoffice-Support und Servicestrategien erweitert hatten, begannen sie, ihr Geschäft über Vertriebskanäle, Distributoren, Händlerservices und andere Organisationen, die Verkaufsabwicklungsfunktionen benötigen, auszubauen.

Teure Windows-basierte Server und Kioske hatten jedoch ihre Grenzen, nicht zuletzt wegen der Kosten und des Werts der Hardware für Diebe. Außerdem nehmen sie viel Platz in Anspruch. „Manche Restaurant-Kassensysteme haben drei oder vier Terminals,

> *Das Unternehmen hat jetzt über 2000 Kunden mit KwickPOS Systemen in ihren Restaurants*

von denen eines der eigentliche Server ist“, erklärt Ye. In mehr als einem Fall ist der Laptop oder Desktop-Computer einem Gelegenheitsdieb zum Opfer gefallen, sodass das betroffene Unternehmen keine Möglichkeit hatte, Bestellungen entgegenzunehmen und auszuführen.

Ein weiteres Problem für den Einzelhandel und die Gastronomie ist die Frage, was passiert, wenn die Internetverbindung ausfällt - eine berechtigte Sorge in Teilen der USA, wo Stromausfälle keine Seltenheit sind. Da immer mehr Bestellungen per E-Mail oder online eingehen, sind Geschäftskontinuität und Betriebszeit wichtiger denn je, ebenso wie die Fähigkeit, Zahlungen zu verarbeiten.

Die Lösung

Mit der Einführung des Raspberry Pi sah Jin sofort das Potenzial, KwickPOS auf dieser Plattform laufen zu lassen und es als Server sowie als diskretes, aber leistungsfähiges Terminal in oft beengten Kiosken und Restaurants einzusetzen, wo der Platz zum Essen und die Anzahl der Restaurantbesucher maximiert werden muss. Da die bestehende Backend-Software des Unternehmens seit 2013 auf Linux basierte, war der Umstieg auf den Raspberry Pi wirtschaftlich sinnvoll, zumal das Compute Module nur ein Zehntel so viel kostet wie eine Windows-Umgebung. KwickPOS konnte den Raspberry Pi für seinen Cloud-basierten Server voll ausnutzen.

Im Gegensatz zu den meisten Kassensystemen ist KwickPOS browserbasiert, mit einem Server vor Ort - was durch die kompakte Größe des Compute Modules möglich ist - und die auf den Terminals angezeigte POS-Anwendung wird in die Cloud kopiert. Ein Raspberry Pi Server im Inneren jedes Terminals verarbeitet die Zahlungsdaten. Dieser Aufbau hat einen entscheidenden Vorteil: den Offline-Modus. „Wenn das Internet ausfällt, ist das kein Problem“, sagt Ye. „Der Manager kann sich vom Terminal aus einloggen, es in den Offline-Modus versetzen und das Restaurant weiter betreiben.“ Wenn das Internet wieder verfügbar ist, wird das Terminal in den Online-Modus versetzt und alle bereits verarbeiteten Transaktionen werden kopiert und für das Restaurant fertiggestellt. „Die Betriebsbereitschaft ist ein sehr starkes Wettbewerbsargument.“

Warum Raspberry Pi?

Jin schätzt die Tatsache, dass das Raspberry Pi Compute Module so stabil ist. Er ist frustriert über das anhaltende Glauben, dass die Verwendung von Windows für Unternehmen „wie ein Gesetz“ ist. Viele Kunden verwenden es immer noch, obwohl sich ihr Server regelmäßig als Schwachpunkt in ihrem Setup erweist. Die KwickPOS-Dienste nutzen sowohl Compute Module 3 als auch 4.

Auch aus Sicht der Kunden ist der Raspberry Pi eine gute Wahl. „Die Kunden sind sehr zufrieden mit diesem Gerät“, sagt Ye. Zunächst einmal nimmt es nicht viel Platz in einem zwangsläufig beengten Servicebereich eines Restaurants ein und ist diskret genug, um nicht die Aufmerksamkeit von Gelegenheitsdieben zu erregen. Im seltenen Fall, dass ein Gerät gestohlen wird, kann der Betrieb weiterlaufen, da der Raspberry Pi-Webserver übernimmt. Das System stellt nicht nur sicher, dass das Unternehmen aufgrund fehlender Konnektivität keine finanziellen Einbußen erleidet, sondern funktioniert auch so, als wäre es angeschlossen, und der Kunde kann ein gleichwertiges Ersatzgerät ohne Datenverlust einsetzen.

KwickPOS hat seinen Kundenstamm auch auf Einzelhandelsgeschäfte erweitert. Wie bei unabhängigen Restaurants ist die Notwendigkeit, Bestellungen schnell und effizient aufzunehmen und zu verarbeiten, von größter Bedeutung. Wichtig ist, dass KwickPOS plattformunabhängig ist. Da Handheld-Zahlungsterminals mittlerweile weit verbreitet sind, kann KwickPOS in die bestehende Einrichtung eines Geschäfts integriert werden.

Die Ergebnisse

Seit der Umstellung auf Raspberry Pi im Jahr 2018 ist KwickPOS zu einer Organisation gewachsen, die Kunden in 45 US-Bundesstaaten sowie in Kanada und Mexiko hat. Es gibt sogar ein chinesisches Restaurant in London, Großbritannien, das das System nutzt. Das Unternehmen hat jetzt über 2000 Kunden mit KwickPOS-Systemen in ihren Restaurants, von unabhängigen Geschäften bis hin zu Einzelhandelsketten.

WEBLINKS

Erfolgsgeschichte: magpi.cc/success

High-Performance in jeder Klasse

Computer-on-Module-Standards

▲ *Bild 1. Die unterschiedlichen Modulgrößen der herstellerunabhängigen Computer-on-Module-Standards.*

Ein Beitrag von Congatec

Aktuell tut sich viel im Markt der Computer-on-Modules. Mit COM Express 3.1 gibt es eine neue Spezifikationsversion für den erfolgreichsten Computer-on-Module Standard. Auch wirft der High-Performance Embedded Computing Standard COM-HPC viele neue Fragen auf. Was müssen OEM und Systemdesigner deshalb heute wissen?

Computer-on-Modules sind laut Marktzahlen von IHS Markit das am weitest verbreitete Embedded-Design-Prinzip – sogar noch vor klassischen Embedded-Boards wie Mini-ITX oder 3,5-Inch Single-Board-Computern. Begründet ist die hohe Beliebtheit solcher Embedded-System-Designs in der gelungenen Kombination aus der Flexibilität kundenspezifischer Designs über ein einfach zu entwickelndes Carrierboard und ein einfach einzusetzendes, fertig entwickeltes Modul, das auch alle benötigten Treiber und Firmware direkt „ready-to-use“ mitbringt. Als Super-Komponenten bieten sie nämlich alle kritischen Bausteine wie CPU, Arbeitsspeicher, High-Speed-Interfaces und häufig auch die Grafikeinheit in einem funktionsvalidiertes Gesamtpaket mit sich. Vorteilhaft ist auch die Tatsache, dass Computer-on-Modules eines Standards über Prozessorgenerationen und Herstellergrenzen hinweg flexibel austauschbar sind. So können OEM ihre Lösungen flexibel skalieren und auch nach etlichen Jahren noch mit aktueller Prozessortechnik aufrüsten. Zudem lassen sich auch sehr einfach Multivendor-Strategien umsetzen, was Vorteile bei der Preisstellung sowie vor allem bei der Sicherung der Verfügbarkeit bietet. Für die Standardisierung sorgen zwei unabhängige Standardisierungsgremien: die in Amerika gehostete PICMG sowie die deutsche SGET, die aktuell in der Summe vier Computer-on-Module-Standards pflegen, die zumeist auch zahlreiche Varianten ausgebildet haben. Im High-End-Segment sind dies die Standards COM-HPC und COM Express. Im Low-Power-Bereich SMARC und Qseven.

COM-HPC, der neue High-Performance Maßstab

COM-HPC ist der neuste Standard der PICMG für Computer-on-Modules, der sich an High-Performance-Embedded-Computer-Designs richtet, die mit vorherigen Standards nicht erreicht werden konnten. Er ordnet sich mit seiner Performance oberhalb des weltweit führenden COM-Express-Standards ein. COM-HPC zielt auf die kommenden High-Speed Schnittstellen wie PCI Express 4.0 und 5.0 sowie 25-Gbit-Ethernet ab und bietet hierfür zwei unterschiedliche Modulvarianten an, die von der COM-HPC-Workinggroup unter ihrem Chairman Christian Eder von congatec entwickelt wurden: COM-HPC Server und COM-HPC Client. Technisch unterscheiden sich diese vor allem durch einen unterschiedlichen Footprint, die Anzahl und Art der ausgeführten Schnittstellen sowie die Speicherausstattung. Brandneu ist die COM-HPC-Mini-Spezifikation, die die Vorteile nun auch auf einem kreditkartengroßen Footprint bringt.

COM-HPC Server Module

COM-HPC Server definiert das Ultra-High-End des Embedded Computings und adressiert die neuen Edge- und Fog-Server im rauen Umfeld, die zunehmend massive Workloads verarbeiten müssen. Dafür spezifiziert COM-HPC Server zwei unterschiedlich große Footprints mit bis zu 64 PCIe Lanes und bis zu 256 Gigabyte/s sowie bis zu 8x Ethernet mit jeweils 25 Gbit/s. COM-HPC Server ist jedoch nicht auf x86-Technologie beschränkt, sondern sieht auch den Einsatz von RISC-Prozessoren, FPGAs und GPGUs als Recheneinheiten vor, was eine weitere Modularisierungsperspektive bietet. Um den Anforderungen an Serverapplikationen zu entsprechen, bieten die Module auch Master-Slave-Modi sowie ein Remote-Management, das auf den Befehlssatz des mächtigen IPMI-Standards zurückgreift und so etablierte Server-Technologie auch für Server-on-Module verfügbar macht. COM-HPC-Server-Module bieten zudem ein Leistungsbudget von bis zu 300 Watt, sodass sich mit diesem Standard auch besonders leistungsfähige Embedded-Edge- und -Fogserver entwickeln lassen. Zum Vergleich: Das aktuell leistungsstärkste Server-on-Module nach Standard COM Express Type 7 erlaubt maximal 100 Watt. Auch die Anzahl der ausgeführten Signal-Pins unterscheidet sich deutlich. Der Steckverbinder von COM Express hat 440 Pins, während COM-HPC mit 800 Pins fast doppelt so viele bietet.

COM-HPC-Client-Module

COM-HPC-Client-Module sind auf High-Performance-Embedded-Systeme mit integrierter Grafik ausgelegt. Sie bieten mit drei Digital-Display-Interfaces (DDI) und einem Embedded-DisplayPort (eDP) vier Grafikausgänge. Sie hosten bis zu vier SO-DIMM Sockel, in die man aktuell bis zu 128 GByte Arbeitsspeicher stecken kann. Für Peripherieanbindung stehen 48 PCIe-Lanes sowie 2x USB 4.0 zur Verfügung. Über zwei MIPI-CSI-Interfaces können zudem auch eingebettete Kameramodule direkt angeschlossen werden. COM-HPC-Client-Module wird es in drei unterschiedlichen Größen geben: 120 mm x 160 mm (Size C), 120 mm x 120 mm (Size B) und 120 mm x 95 mm (Size A).

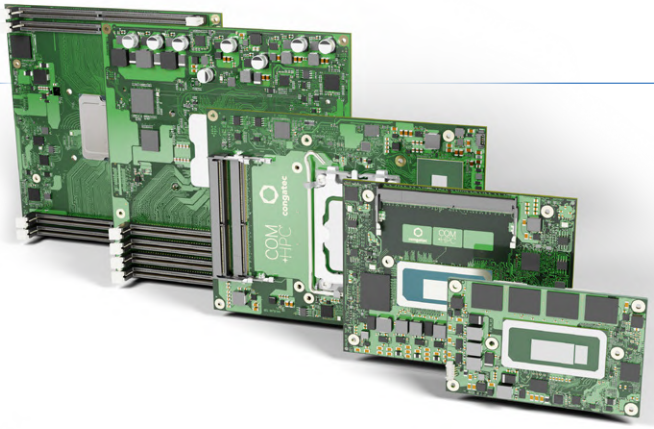


Bild 2. congatec verfügt aktuell über das umfangreichste COM-HPC-Portfolio. Von COM-HPC-Server über COM-HPC-Client bis zu COM-HPC-Mini.

COM-HPC-Mini-Module

COM-HPC Mini ist ein kommender Computer-on-Module-Standard, der derzeit von der PICMG entwickelt wird. Derzeit hat das technische Subkomitee das Pinout und den Footprint der neuen scheckkartengroßen (95 x 60 mm) High-Performance-Computer-on-Module genehmigt. Mit 400 Pins im Vergleich zu den 220 Pins von COM-Express-Mini ist der neue COM-HPC-Mini-Standard dafür ausgelegt, den steigenden Schnittstellenbedarf heterogener und multifunktionaler Edge-Computer zu erfüllen. Zu den Erweiterungen gehören bis zu vier USB 4.0 mit voller Funktionalität einschließlich Thunderbolt und DisplayPort-alternate-Mode, PCIe Gen 4/5 mit bis zu 16 Lanes, zwei Ethernet Ports mit 10 Gbit/s und vieles mehr. Zusammen mit der Tatsache, dass der COM-HPC-Mini-Steckverbinder für Bandbreiten von mehr als 32 Gbit/s qualifiziert ist – genug um PCIe Gen 5 oder sogar PCIe Gen 6 zu unterstützen – wird deutlich, dass sein Potenzial weit über die aller anderen kreditkartengroßen Modulstandards hinausgehen wird.

Der weltweit erfolgreichste Modulstandard: COM Express

Damit ist also der kleinste Footprint von COM-HPC Client fast genauso groß wie COM Express Basic mit 125 mm x 95 mm. Dadurch lässt sich erkennen, dass

Bild 3. Neueste Prozessortechnologie wie die High-End-Core-Prozessoren der 13. Generation in BGA-Bestückung von Intel liefert congatec auf COM-HPC- und COM-Express-Modulen.

sich COM-HPC Client deutlich oberhalb von COM Express verortet und damit Applikationen adressiert, die sich mit COM Express nicht erreichen lassen. COM Express wurde 2005 eingeführt und ist damit der bislang am längsten verfügbare unter den hier vorgestellten Computer-on-Module-Standards. Die Spezifikation definiert eine Familie unterschiedlicher Modulgrößen und Pinbelegungstypen. Anders als COM-HPC und die Small-Formfaktor-Modul-Spezifikationen Qseven und SMARC konzentriert sich COM Express allein auf die x86-Prozessortechnologie. Mit der Ratifizierung der 3.1-Spezifikation unterstützt COM Express nun auch neueste Hochgeschwindigkeitsschnittstellen wie PCIe 4.0 und USB 4. Trotz der Verbesserungen sind Module nach COM Express 3.1 Typ 6 vollständig abwärtskompatibel zu 3.0-Modulen und Carrier Boards, so dass auch ältere Designs mit den neuesten Prozessortechnologien ausgestattet werden können.

COM Express Type 7 Server-on-Module

Analog zu COM-HPC gibt es auch bei COM Express sowohl Server- als auch Client-Module, die vor allem mit den bekannten Pinouts Type 6 (Client) und Type 7 (Server) verfügbar sind. Wie bei COM-HPC ist das Server-Pinout Type 7 ein Headless-Server-on-Module ohne Grafikausgänge und ebenfalls für Embedded-Edge- und -Fog-Server konzipiert. Sehr interessant ist die Unterstützung von bis zu vier 10 GbE- und bis zu 32 High-Speed-PCIe-Lanes der Gen 3.0 für Schnittstellen und Speichermedien. Verfügbar sind diese Server-on-Modules beispielsweise mit Xeon-D-Prozessoren von Intel® oder EPYC-Embedded-3000-Prozessoren von AMD. Für sie bietet congatec auch ein 100-Watt-Ökosystem mit applikationsfertigen Kühllösungen an, um das Design-In der leistungsstärksten COM-Express-Server-on-Module zu vereinfachen.

COM Express Type 6 Computer-on-Modules

Für klassische Embedded-Applikationen mit Grafik bietet sich die Spezifikation PICMG COM Express Type 6 an. Diese Module sind mit Embedded-Prozessoren verfügbar, die von Intel® Core™, Pentium® und Celeron® bis zur AMD Embedded R-Serie reichen. Sie messen 95 mm x 125 mm (Basic) oder 95 mm x 95 mm (Compact) und bieten 440 Pins zum Carrierboard mit einer breiten Palette an aktuellen Computerschnittstellen. Sie unterstützen bis zu vier unabhängige Displays, 24 PCIe Lanes, USB 2.0 und USB 3.0 sowie Ethernet, CAN-Bus und serielle Schnittstellen. Sie bieten damit alles, was zum Aufbau leistungsstarker SPSen, HMIs, Shop-Floor-Systemen oder SCADA-Arbeitsplätzen in Leitwarten benötigt wird. Weitere Anwendungsgebiete sind High-End-Digital-Signage-Systeme und leistungsstarke Medizingeräte für die bildgebende Diagnostik. Mit der 3.1-Spezifikation können Module nach COM Express Type 6-nun auch optional MIPI-CSI-Anschlüssen auf dem Modul ausführen.

Bild 4. SMARC-2.1-Carrierboard und skalierbarer 3,5-Inch-Singleboard-Computer: Das conga-SMC1 schlägt den Bogen von modul-basierten Designs hin zu hoch skalierbaren Commercial-off-the-Shelf verfügbaren Standard-Boards.

COM Express Type 10 Mini-Module

Für den kleinsten Formfaktor COM Express Mini im Format 55 mm x 84 mm spezifiziert die PICMG das Type-10-Pinout und rundet damit den Satz der COM-Express-Spezifikationen in Richtung der Designs im Small Form Faktor (SFF) ab. Diese Module sind für Low-Power Atom™- und Celeron®- Prozessoren von Intel konzipiert. Auch hier können 3.1-konforme Module zwei MIPI-CSI-Anschlüsse ausführen. Dank der einheitlichen Konnektortechnik und Designguides, die im gesamten Ökosystem PICMG COM Express eingesetzt werden, können Entwickler besonders viele Funktionen wiederverwenden, was der Hauptvorteil dieser Mini-Spezifikation ist. Etablierter und weiter verbreitet sind jedoch die SMARC- und Qseven-Standards der SGET. Beide Standards unterstützen sowohl x86- als auch ARM-Applikationsprozessoren.

SMARC für Embedded Vision

SMARC adressiert das High-End der SFF-Applikationen. Dieser Standard hat jüngst mit der Revision 2.1 ein wichtiges Update erfahren. Die neue Revision bringt zahlreiche neue Features mit sich, wie SerDes-Support für erweiterte Edge-Konnektivität sowie zwei zusätzliche Interfaces auf dem Modul für nun insgesamt vier MIPI-CSI Kameraschnittstellen, um dem zunehmenden Bedarf nach der Fusion von Embedded-Computing und Embedded-Vision gerecht zu werden. Die neuen Funktionen sind rückwärtskompatibel zur Rev. 2.0. Alle Erweiterungen zur Rev. 2.0 sind zudem optional, sodass alle SMARC 2.0 Module von congatec automatisch auch kompatibel mit SMARC 2.1 sind.

Neben den zwei MIPI-Interfaces auf dem Konnektor ist die Unterstützung drahtloser Interfaces wie WLAN und Bluetooth direkt auf dem Modul ein weiteres Unterscheidungsmerkmal zu Qseven. Ideale Prozessoren für SMARC-Module sind die 5. Generation der Intel-Atom-Prozessoren sowie die gesamte Bandbreite der neuen i.MX-8-Applikationsprozessoren, die congatec in zwölf verschiedenen Geschmacksrichtungen auf seinen SMARC-Computer-on-Modules anbietet.

230071-02

Aller Anfang...

muss nicht schwer sein: Werden wir aktiv!

Von Eric Bogers (Elektor)

Wie bereits erwähnt, werden alle Bauteile, die einen Halbleiterchip enthalten (zum Beispiel Dioden) als aktive Bauteile bezeichnet. Eigentlich ist das nicht richtig, wenn man „aktiv“ sehr streng definiert, sind nur Bauteile, die einen Strom oder eine Spannung verstärken, wirklich aktiv. Und jetzt - mit den Transistoren - betreten wir endgültig die Welt der aktiven Bauelemente.

Das Wort Transistor ist ein Kunstwort, das sich aus den Wörtern „Transfer“ und „Resistor“ zusammensetzt. Man könnte einen Transistor also scherzhaft als „Übertragungswiderstand“ bezeichnen, aber zum Glück tut das niemand. Tatsächlich kann man aber den Widerstand dieses Bauteils durch Anlegen einer Steuerspannung verändern. Transistoren haben das heutige Elektronikzeitalter erst möglich gemacht, weil sie viel kleiner und weniger empfindlich sind als Elektronenröhren und weil sie deutlich weniger Energie verheizen. Transistoren haben letztlich auch integrierte Schaltkreise (ICs) möglich gemacht. ICs enthalten nämlich eine sehr große Anzahl von (aktiven und passiven) Bauteilen auf einem Chip. Bei modernen Mikroprozessoren handelt es sich um viele Millionen Transistoren in einem einzigen Gehäuse.

Aber beginnen wir mit Transistoren als individuelle Bauteile. Mit integrierten Schaltkreisen befassen wir uns in einer späteren Folge. Heutzutage gibt es für viele Anwendungen fertige Lösungen in Form von ICs, aber oft muss man in der Bauteilschublade nach „losen“ Transistoren wühlen - zum Beispiel, wenn man Ausgangstransistoren für einen Leistungsverstärker braucht.

In **Bild 1** sind einige Transistoren zu sehen. Als Faustregel kann man sagen, dass die Leistungsfähigkeit eines Transistors davon abhängt, wie groß er ist und ob er ein Metallgehäuse hat.

Bild 1. Mehrere Transistoren (Quelle: Shutterstock / Edkida).

Ein Bipolartransistor als Schalter

Wenn wir von einem Transistor sprechen, meinen wir normalerweise einen Bipolartransistor. Es gibt zwei Typen: npn und pnp, wobei die Buchstaben für die Reihenfolge der n-dotierten und p-dotierten Siliziumschichten der Transistorstruktur stehen.

Grundsätzlich (aber sehr stark vereinfacht) gesagt können Transistoren als Schalter oder als Verstärker verwendet werden. Zunächst werden wir uns mit der Verwendung eines Transistors als Schalter befassen, da es so einfacher ist, die Funktionsweise zu verstehen. Danach werden wir uns mit den Eigenschaften von Transistoren als Verstärker beschäftigen.

Bild 2 zeigt einen npn-Transistor auf der linken Seite und einen pnp-Transistor auf der rechten Seite, jeweils dargestellt durch eine Kombination von zwei Dioden. In der Praxis wird ein Transistor nie als einfacher Ersatz für zwei Dioden verwendet, aber wenn man mit einem Durchgangsprüfer testen will, ob ein Transistor ein npn- oder ein pnp-Typ ist (oder ob er noch funktioniert oder intern kurzgeschlossen ist), ist es praktisch, diese Ersatzschaltungen im Kopf zu haben.

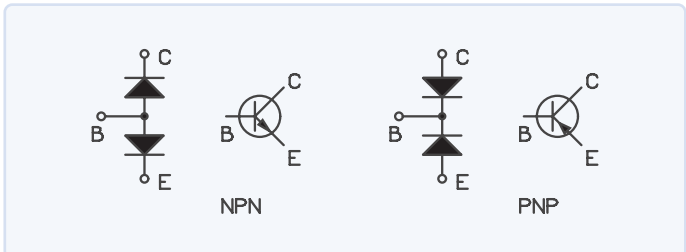


Bild 2. Äquivalenzschaltungen von Transistoren: C = Kollektor, B = Basis, E = Emitter.

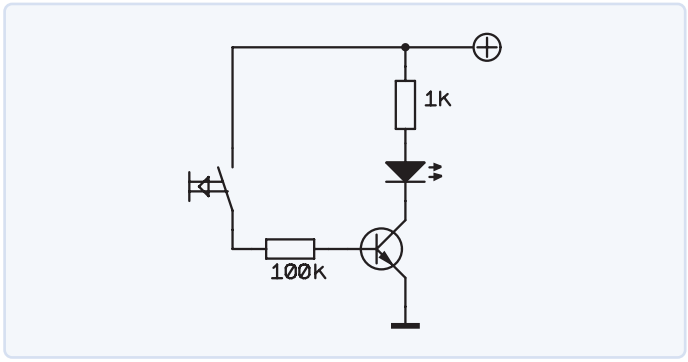


Bild 3. Ein Transistor, der als Schalter verwendet wird.

Bild 3 zeigt einen Transistor, der als Schalter verwendet wird. Übrigens benutzen wir in den Beispielen in dieser Folge immer npn-Transistoren, weil sie leichter zu verstehen sind, wenn man die Richtung des elektrischen Stroms (von positiv nach negativ) im Kopf hat. Jedes dieser Beispiele kann jedoch leicht für einen pnp-Typ umgewandelt werden, indem man einfach die Polarität aller gepolten Komponenten (Dioden, Elektrolytkondensatoren und natürlich die Versorgungsspannung) umdreht.

Gehen wir zurück zu Bild 3. Hier gibt es zwei Schaltkreise: einen für den Steuerstrom und einen für die Last. Der Steuerstrom fließt von Plus durch den Schalter, den Vorwiderstand und die Basis-Emitter-Diode des Transistors nach Masse. Dieser Basis-Emitter-Übergang ist für die Steuerung des Transistors verantwortlich. Wenn ein Strom durch diesen Übergang fließt, leitet der Transistor (man sagt auch: Er ist durchgeschaltet), und wenn kein Strom durch diesen Übergang fließt, ist der Transistor abgeschaltet (er sperrt). Der Basis-Emitter-Übergang verhält sich wie eine gewöhnliche Diode, was bedeutet, dass ein Strom erst dann zu fließen beginnt, wenn die Spannung an diesem Übergang einen Wert von etwa 0,7 V gegen den Emitter beziehungsweise Masse erreicht.

Der andere Stromkreis besteht aus dem Vorwiderstand der LED, der LED selbst und dem Kollektor-Emitter-Übergang des Transistors. Man könnte meinen, dass dort kein Strom fließt, weil die Kollektor-Basis-Diode in Sperrichtung vorgespannt ist. In der Praxis fließt der Strom aber nicht durch die Kollektor-Basis-Diode, sondern direkt vom Kollektor zum Emitter, so dass am Kollektor-Emitter-Übergang keine Spannung von 0,7 V anliegt.

Dies funktioniert aufgrund der besonderen Anordnung der verschiedenen Halbleiterschichten, es ist schließlich nicht möglich, einen Transistor aus einem Paar Dioden herzustellen.

Sie fragen Sie sich vielleicht (zu Recht), warum wir in der Schaltung von Bild 3 einen Transistor und einen zusätzlichen Widerstand verwenden, anstatt die LED einfach direkt mit dem Schalter zu betreiben. Da haben Sie völlig recht, das ist in diesem speziellen Beispiel völlig unnötig.

Der Strom durch den Schalter in dieser Schaltung ist um den Faktor 100 geringer als der Strom durch die LED, obwohl es sehr schwierig wäre, einen Schalter zu finden, der den Strom durch die LED nicht bewältigen könnte. Die hier gezeigte Schaltung eignet sich aber auch zum Schalten von viel höheren Strömen (und Spannungen), da es keinen Grund gibt, warum der Steuerkreis und der Lastkreis aus derselben Spannungsquelle gespeist werden müssen. Einfach ausgedrückt: Der Transistor ermöglicht es, mit einer Steuerspannung von wenigen Volt eine viel höhere Spannung zu schalten, vorausgesetzt, man verwendet einen geeigneten Transistor.

Der bistabile Multivibrator (Flip-Flop)

Betrachten wir nun einige Beispiele für Multivibratoren. Das sind Schaltungen, die zwischen zwei definierten Zuständen umschalten. **Bild 4** zeigt einen bistabilen Multivibrator, der allgemein als Flip-Flop bekannt ist.

Diese Schaltung wird als bistabil bezeichnet, weil sie zwei stabile Zustände hat - entweder leuchtet die LED auf der linken Seite oder die LED auf der rechten Seite. Die Schaltung wechselt ihren Zustand nur, wenn Sie einen der beiden Druckknöpfe drücken.

Nehmen wir an, der linke Transistor ist leitend, was bedeutet, dass die linke LED leuchtet. Die Spannung zwischen Kollektor und Emitter dieses Transistors ist also niedrig (etwa 0,1 V), was nicht ausreicht, um den rechten Transistor einzuschalten (dazu wären ja mindestens 0,7 V erforderlich). Der rechte Transistor ist also abgeschaltet, und seine Kollektor-Emitter-Spannung ist fast gleich der Versorgungsspannung U_V. Das bedeutet, dass der linke Transistor immer durch den Strom durch den 100-kΩ-Widerstand auf der rechten Seite eingeschaltet bleibt. Dies ist also ein stabiler Zustand.

Drückt man jedoch den linken Taster, fällt die Basis-Emitter-Spannung des linken Transistors auf null. Infolgedessen schaltet dieser Transistor ab und seine Kollektor-Emitter-Spannung steigt auf etwa U_V. Die linke LED wird dunkel, und der rechte Transistor wird durch den Strom durch den 100-kΩ-Widerstand auf der rechten Seite in den leitenden Zustand versetzt. Seine Kollektor-Emitter-Spannung fällt auf nahezu 0 V und die rechte LED leuchtet auf. Nun kann der linke Transistor nicht mehr eingeschaltet werden, auch nicht, wenn Sie den Taster loslassen. Dies ist also wiederum ein stabiler Zustand, aber das Gegenteil des Ausgangszustandes.

Die Schaltung verbleibt in diesem stabilen Zustand, bis jemand den rechten Taster drückt, wodurch die Schaltung wieder in den anderen stabilen Zustand übergeht.

Jetzt muss nur noch die Frage beantwortet werden, welcher Transistor nach dem Einschalten der Versorgungsspannung zuerst schaltet. Das lässt sich anhand des Schaltplans nicht vorhersagen. Würden beide Hälften der Schaltung mit identischen Bauteilen aufgebaut, wäre das reine Glückssache, oder anders gesagt, es würde vom unvorhersehbaren Bauteilrauschen abhängen. In der realen Welt gibt es jedoch keine exakt identischen Bauteile. Die Widerstände

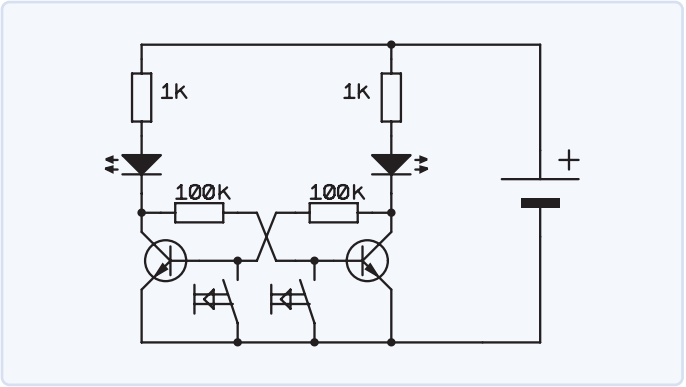


Bild 4. Ein bistabiler Multivibrator.

haben Fertigungstoleranzen, und die Stromverstärkungen der Transistoren sind nicht genau gleich (und außerdem hängt die Stromverstärkung von der Temperatur ab, also davon, welcher Transistor gerade leitend war und wahrscheinlich wärmer ist als der andere Transistor).
Es ist besser, die Frage umzudrehen: Wie kann man sicherstellen, dass immer derselbe Transistor zuerst durchschaltet, wenn die Versorgungsspannung eingeschaltet wird? Die Antwort ist einfach: Verringern Sie den Wert des Basiswiderstands des betreffenden Transistors auf die Hälfte seines Gegenspielers auf der anderen Seite.
Das war's für diese Folge. In der nächsten Folge werden wir zwei weitere Multivibrator-Schaltungen untersuchen und uns dann mit der Verwendung von Transistoren als Verstärker beschäftigen.

Kurzer Rückblick auf den Diac

In der November/Dezember-Ausgabe des letzten Jahres [1] erwähnten wir kurz den Diac als eine Art von Diode (die inzwischen mehr oder weniger veraltet ist). In jenem Artikel schrieben wir:
„In Ihrem bevorzugten Elektronikversandhandel (leider gibt es den „Elektronikladen um die Ecke" kaum mehr) können Sie ein Bauteil mit dem einfachen, aber exotischen Namen „Diac" kaufen, das zwei auf diese Weise verbundene Z-Dioden enthält."
Zwei Elektor-Leser (Pablo Medina und Maurizio Spagni) haben uns unabhängig voneinander darauf hingewiesen, dass dies nicht stimmt. Und beide haben absolut recht: Ein Diac ist ein ziemlich komplexes Bauteil mit einer negativen Widerstandscharakteristik, wodurch er sich zur Ansteuerung von Thyristoren und Triacs eignet [2].
Ein Diac mit zwei gegeneinander geschalteten Z-Dioden zu vergleichen, ist eine grobe Vereinfachung. Dennoch sind wir der Meinung, dass Elektronikanfänger die Funktionsweise eines Diacs besser verstehen können, wenn sie ihn als zwei Zenerdioden und nicht als negativen Widerstand betrachten. Bis sie verstanden haben, was mit einer negativen Widerstandscharakteristik gemeint ist und wie ein Halbleiterbauelement diese Art von Charakteristik haben kann, werden sie wahrscheinlich ihren Lötkolben vor Frustration aus dem Fenster geworfen haben.
Aber wie bereits erwähnt, haben beide Briefschreiber absolut recht, und wir werden in Zukunft unsere Worte sorgfältiger wählen.

(220670-02)RG

Die Artikelreihe „Aller Anfang ..." gründet auf dem Buch „Basiskurs Elektronik" von Michael Ebner, erschienen im Elektor-Verlag.

Haben Sie Fragen oder Kommentare?

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

Passende Produkte

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

WEBLINKS

[1] „Aller Anfang ... Es geht weiter mit den Z-Dioden", ElektorMag 11-12/2022: https://www.elektormagazine.de/220384-02
[2] Diac-Eintrag in Wikipedia: https://de.wikipedia.org/wiki/Diac

I²C-Kommunikation mit Node.js und einem Raspberry Pi

Sehen Sie Ihre Sensordaten in einem Browser

Von Franck Bigrat (Frankreich)

Den I²C-Bus gibt es seit den 1980er Jahren, und er wird von den heute gängigen Plattformen einschließlich der Raspberry-Pi-Serie gut unterstützt. Mit Node.js ist es möglich, einen Webserver in JavaScript zu programmieren, der Daten in einem lokalen Netzwerk oder im Internet bereitstellt. Dieser Artikel zeigt Ihnen, wie einfach es ist, einen I²C-Sensor an einen Raspberry Pi anzuschließen und die Ergebnisse in einem Webbrowser anzuzeigen.

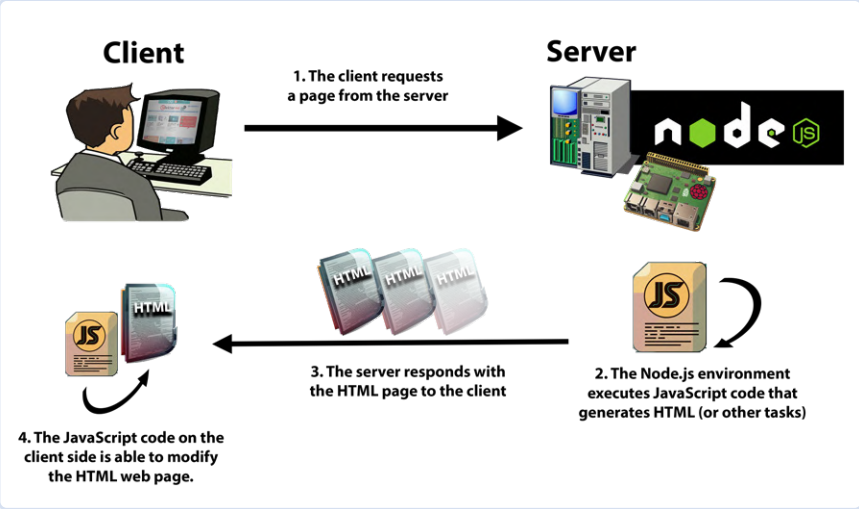


Bild 1. Kommunikationsfluss eines Node.js-Servers. (Quelle: sdz.tdct.org)

In diesem Projekt verwenden wir einen einfachen Raspberry Pi Zero und den I²C-Temperatursensor DS1621. Das Messergebnis wird in das lokale WLAN übergeben und auf einer Webseite auf einem Tablet oder Smartphone angezeigt, das mit demselben WLAN verbunden ist. Node.js verfügt über Module wie *http* und *express*, die es dem Programmierer ermöglichen, sehr einfach Webserver zu erstellen, so dass wir nicht den „schweren" Apache-Server benötigen. Obwohl Node.js auch ein Modul zur Verwaltung des DS1621-Sensors kennt, ist es interessanter, das Modul *i2c-bus* zu verwenden, so dass wir das Projekt für eine Vielzahl anderer I²C-Geräte erweitern können.

Was ist Node.js?

Node.js in wenigen Worten zu erklären, ist keine leichte Aufgabe, und es gibt viele Bücher und Websites, die sich mit Node.js befassen. Machen Sie sich einfach klar, dass *es sich bei Node.js um JavaScript handelt, das von einem Webserver ausgeführt wird, also nicht von einem Webbrowser beim Client* (allerdings kann der Browser natürlich auch JavaScript-Code ausführen, wenn es nötig ist). Es basiert auf der JavaScript-Engine von Google Chrome (V8). Einfach ausgedrückt: Der Programmierer schreibt JavaScript-Anwendungen, die jedoch auf dem Server ausgeführt werden. Die Übersicht ist in **Bild 1** dargestellt. Um (fast) alles über Node.js zu erfahren, empfehle ich dem Leser dringend den Besuch der Websites [1] und [2]. Der Leser kann auch auf diese Bücher zurückgreifen:

- *Node.js: Das umfassende Handbuch* von Sebastian Springer (Rheinwerk)
- *Node.js – Rezepte und Lösungen* von Philip Ackermann (Rheinwerk)
- *Node.js & Co.* von Golo Roden (dpunkt.verlag)
- *The Node Beginner Book* und *The Node Craftsman Book* von Manuel Kießling (Learnpub)

I²C und SMBus

In diesem Abschnitt gehe ich davon aus, dass der Leser mit dem I²C-Protokoll (Inter-Integrated Circuit) einigermaßen vertraut ist. Das Wichtigste ist, dass I²C in den 1980er Jahren von Philips entwickelt wurde, um die Anzahl der Verbindungen zwischen einem Mikroprozessor und mehreren anderen integrierten Schaltungen zu minimieren. Es handelt sich um ein serielles, synchrones Datenüber-

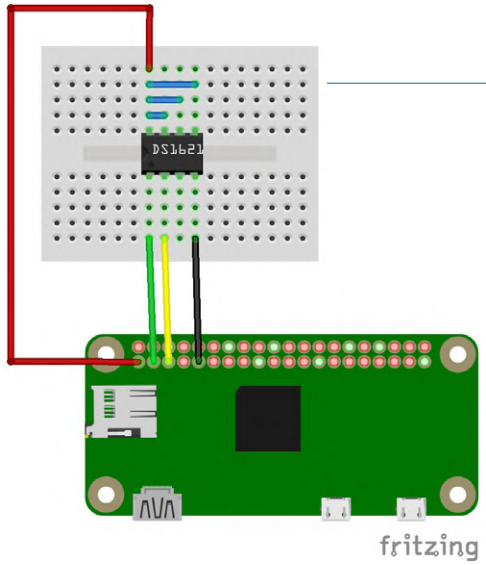


Bild 2. Anschlussdiagramm Raspberry Pi Zero W an den Sensor DS1621.

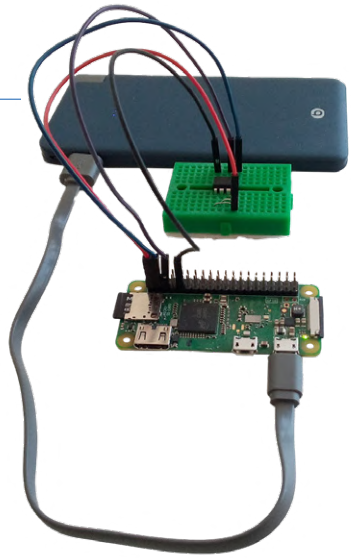


Bild 3. Die Schaltung ist verdrahtet und wird von einer Power-Bank versorgt.

tragungsprotokoll über zwei Leitungen (Daten und Takt). Es können mehrere Schaltungen/ICs gleichzeitig an dieselben Daten- und Taktleitungen angeschlossen werden.

Einzelheiten zu den Funktionen des Node.js-Moduls *i2c-bus* finden Sie unter [3]. Wenn Sie genau hinsehen, können Sie sehen, dass einige Funktionen ein *smbus*-Präfix haben oder im SMBus-Abschnitt enthalten sind. SMBus ermöglicht die Kommunikation zwischen dem Prozessor eines Computers und verschiedenartiger Peripherie wie Energieverwaltung-ICs, Temperatursensoren oder Lüftern.

Der Raspberry Pi unterstützt dieses Protokoll in der Hardware: Der *System Management Bus* ist vom I²C-Bus abgeleitet, und die beiden Protokolle sind recht kompatibel. Es gibt jedoch einige Unterschiede zwischen SMBus und I²C, die im Dokument [4] von Texas Instruments erläutert werden.

Konfigurieren des Raspberry Pi

Nun ist es an der Zeit, den Raspberry Pi mit dem DS1621-Temperatursensor gemäß dem Layout in **Bild 2** zu verbinden. Vergessen Sie nicht, die Pins 5...7 (A0, A1, A2) des DS1621 mit V_{CC} zu verbinden, um die I²C-Adresse des Geräts auf 4F (hexadezimal) einzustellen. Die verdrahtete Schaltung ist in **Bild 3** zu sehen.

Bevor wir weitermachen können, müssen wir natürlich zunächst den Raspberry Pi mit einem Betriebssystem und weiteren für uns wichtigen Softwarekomponenten ausstatten.

1. Laden Sie den Raspberry Pi Imager von *https://raspberrypi.com/software* herunter und installieren Sie ihn auf Ihrem PC.
2. Starten Sie den Raspberry Pi Imager (**Bild 4a**), klicken Sie auf *OS WÄHLEN* und dann auf *Raspberry Pi OS (Other)* aus der Liste, die sich öffnet (**Bild 4b**), und dann auf *Raspberry Pi OS Lite (32-bit)* aus dem Untermenü.
3. Klicken Sie auf *SD-KARTE WÄHLEN* und wählen Sie dann Ihre microSD-Karte aus (**Bild 4c**).
4. Klicken Sie auf das Einstellungen-Symbol (Zahnrad) auf der rechten Seite des Installers und legen Sie die Optionen für die Erstkonfiguration des Raspberry Pi fest:
   - Markieren Sie *Hostname* und geben Sie dann den gewünschten Hostnamen ein (oder lassen Sie *raspberry* stehen).
   - Aktivieren Sie *SSH aktivieren* und wählen Sie das Optionsfeld *Password zur Authentifizierung verwenden.*

Bild 4a. Der Raspberry-Pi-Imager.

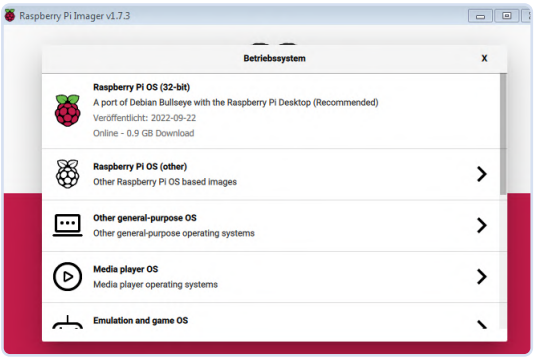


Bild 4b. Auswählen des Betriebssystems.

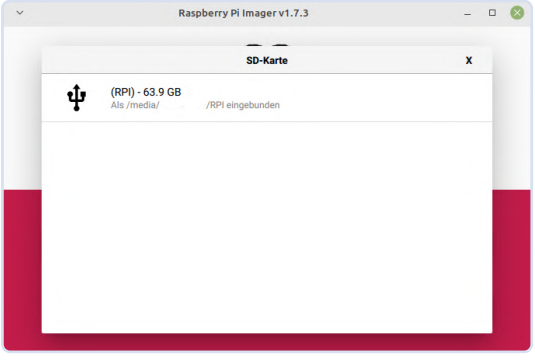


Bild 4c. Auswählen des Speichermediums.

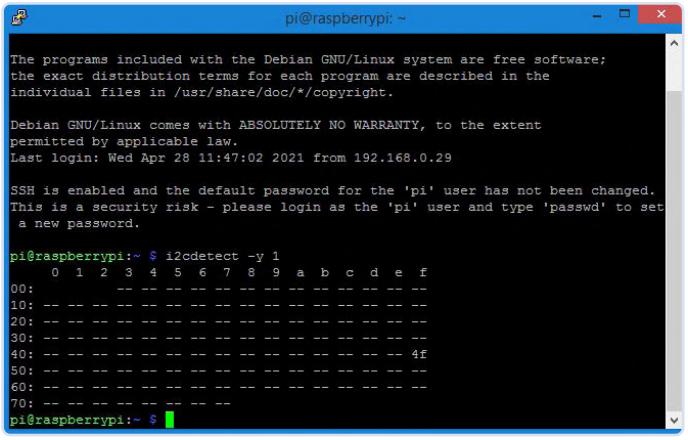


Bild 5. Ausgabe des Programms i2cdetect.

- Aktivieren Sie, wenn noch nicht geschehen, das Kontrollkästchen *Benutzername und Passwort setzen* und geben Sie den *Benutzername* an (pi ist schön) und ein *Passwort* ein (wählen Sie ein leicht zu merkendes wie *raspberry*)
- Aktivieren Sie *WiFi einrichten* und geben Sie die SSID und das Passwort Ihres WLANs ein.
- Aktivieren Sie *Spracheinstellungen festlegen* und wählen Sie Ihre *Zeitzone* und Ihr *Tastaturlayout* nach Belieben aus oder lassen Sie die Standardeinstellungen unverändert.
- Klicken Sie auf die Schaltfläche *SPEICHERN*.

5. Installieren Sie das Betriebssystem, indem Sie auf *SCHREIBEN* klicken und dann mit *JA* bestätigen. Wenn man die Installation unter Windows vornimmt, kann es notwendig sein, eine unter Linux gebrauchte SD-Karte zuvor in der Computerverwaltung neu zu formatieren.
6. Wenn der Schreibvorgang abgeschlossen ist, entfernen Sie die microSD-Karte,
7. stecken sie in den microSD-Slot des Raspberry Pi und schalten den Pi ein.
8. Ermitteln Sie die IP-Adresse des Raspberry Pi im Netzwerk entweder mit einem Tool wie dem *Advanced IP Scanner* oder noch besser in der Funknetz-Übersicht Ihres WLAN-Routers.
9. Verwenden Sie einen SSH-Client wie Putty, um sich mit dem Raspberry Pi zu verbinden. Die IP-Adresse ist die gerade ermittelte und als Benutzername/Passwort verwenden Sie (in unserem Beispiel) *pi*/*raspberry*.
10. Erweitern Sie das Dateisystem, um den vollen Speicherplatz der microSD-Karte zu nutzen:
    - Öffnen Sie das Raspi-Konfigurationstool: `sudo raspi-config`
    - Wählen Sie *Advanced Options* und dann die Option *Expand Filesystem*.
    - Bestätigen Sie mit der Eingabetaste und mit <*OK*>.
    - Gehen Sie zu <*Finish*> und bestätigen Sie mit Enter, dann bestätigen Sie den Neustart <*Yes*> und drücken Sie Enter.
    - 11. So aktivieren Sie den I²C-Bus:
    - Starten Sie PuTTY und Raspi-Konfigurationstool: `sudo raspi-config` erneut. Wählen Sie *Interface Options* und dann *I2C (Enable/disable automatic loading of I2C kernel module)*.
    - Bestätigen Sie mit der Eingabetaste. Bestätigen Sie erneut, indem Sie mit der Tabulatortaste zur Schaltfläche <*YES*> wechseln und erneut die Eingabetaste drücken.
    - Schließen Sie mit <*OK*> und dann <*Finish*> ab.
12. Verbinden Sie den Raspberry Pi mit dem DS1621 gemäß unserem Anschlussplan in Bild 2.
13. Prüfen Sie die I²C-Kommunikation:
    - Führen Sie zunächst ein globales Update durch: `sudo apt update`
    - Installieren Sie die I²C-Diagnosetools: `sudo apt install i2c-tools` (eventuell müssen Sie den Raspberry Pi zuerst neu starten).
    - Führen Sie den folgenden Befehl aus: `i2cdetect -y 1`
    - In der erscheinenden Liste sollte der Sensor, den wir fest verdrahtet haben (**Bild 5**), mit der Adresse *4f* erscheinen.
14. Installieren Sie nun Node.js und NPM (Node Package Manager): `sudo apt install -y nodejs npm` (dies kann eine Weile dauern).
15. Überprüfen Sie die Versionen mit `node -v` gefolgt von `npm -v`
16. Installieren Sie die Module *express*, *i2c-bus* und *sleep*:
    `npm install express`
    `npm install i2c-bus`
    `npm install sleep` (könnte Warnungen erzeugen)
17. Erstellen Sie ein Verzeichnis namens *Documents* und darin ein Unterverzeichnis namens *NodeJS*.
18. Kopieren Sie mit einem FTP-Client wie FileZilla die Dateien *DS1621_V3.js* und *DS1621_V4.html* in den Ordner *NodeJS*.

Probieren wir es aus!

Sobald alle Elemente angeschlossen, der Raspberry Pi konfiguriert, Node.js und die verschiedenen Module installiert und die *.js*- und *.html*-Dateien auf den Raspberry Pi kopiert sind, öffnen Sie das Verzeichnis *Documents/NodeJS* und führen Sie das Skript mit `node DS1621_V3.js` aus.
Verbinden Sie einfach einen PC, ein Tablet oder ein Smartphone mit Ihrem WLAN und öffnen Sie Ihren Lieblingsbrowser mit der Adresse:

[Ihre_Raspberry_Pi_IP_Adresse]:8080

Wenn die vorangegangenen Schritte korrekt ausgeführt wurden, sollte es wie in **Bild 6** aussehen. Nach ein paar Sekunden wird die Temperatur angezeigt (**Bild 7**). Von da an wird die Temperatur alle fünf Sekunden aktualisiert.

Erläuterung der Node.js- und HTML-Skripte

Was bei Node.js wichtig, aber schwierig ist, ist das Verständnis der Ausführung des Skripts. In Node.js ist die Ausführung eines Skripts nicht sequentiell (**Bild 8**): Die Anweisungen in der Funktion `Server.get('/',...)` werden nicht vor den Anweisungen in der Funktion `Server.get('/temp',...)` ausgeführt, die wiederum nicht vor den Anweisungen am unteren Ende ausgeführt werden.

Bild 6. Wir warten auf Daten.

Bild 7. Gemessene und angezeigte Temperatur.

Die Anweisungen in der Funktion `Server.get('/',...)` werden nur ausgeführt, wenn der Server die Anforderung *[Rpi_IP_Address]:8080* vom Client erhält und die Anweisungen in der Funktion `Server.get('/temp',...)` nur, wenn der Server die Anforderung *[Rpi_IP_Address]:8080/temp* vom Client erhält. Dies wird als *Ereignisgesteuerte Programmierung* bezeichnet.
In den ersten fünf Zeilen des Node.js-Skripts (linke Seite von Bild 8) importieren wir die erforderlichen Node.js-Module und erstellen verschiedene Objekte:

- Wir erstellen ein `I2C`-Objekt mit Hilfe des `i2c-bus`-Moduls:

```
const I2C = require('i2c-bus');
```

- Wir erstellen mit Hilfe des `express`-Moduls ein `express`-Objekt. Dieses Modul ist ein Framework für die Erstellung von Webanwendungen mit Node.js:

```
var express = require('express');
```

- Wir erstellen ein `fs`-Objekt mit Hilfe des `fs`-Moduls (Dateisystem). Damit können wir Dateien auf der microSD-Karte lesen und schreiben:

```
var fs = require('fs');
```

- Wir erstellen ein `sleep`-Objekt mit Hilfe des `sleep`-Moduls. Mit diesem Modul können wir Verzögerungen erzeugen:

```
var sleep = require('sleep');
```

- Schließlich erstellen wir ein `Server`-Objekt:

```
var Server = express();
```

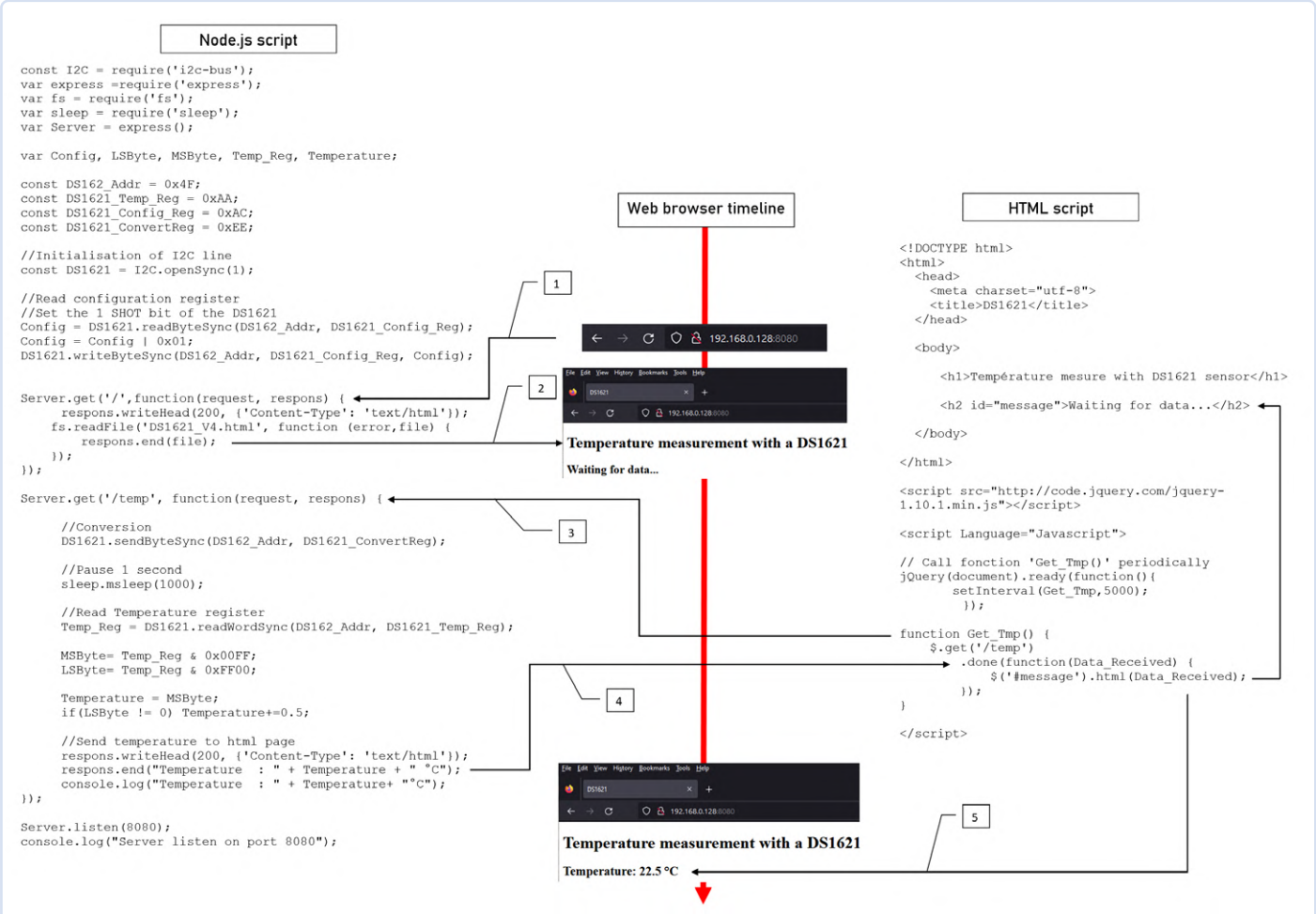


Bild 8. Zeitleiste der Ereignisse.

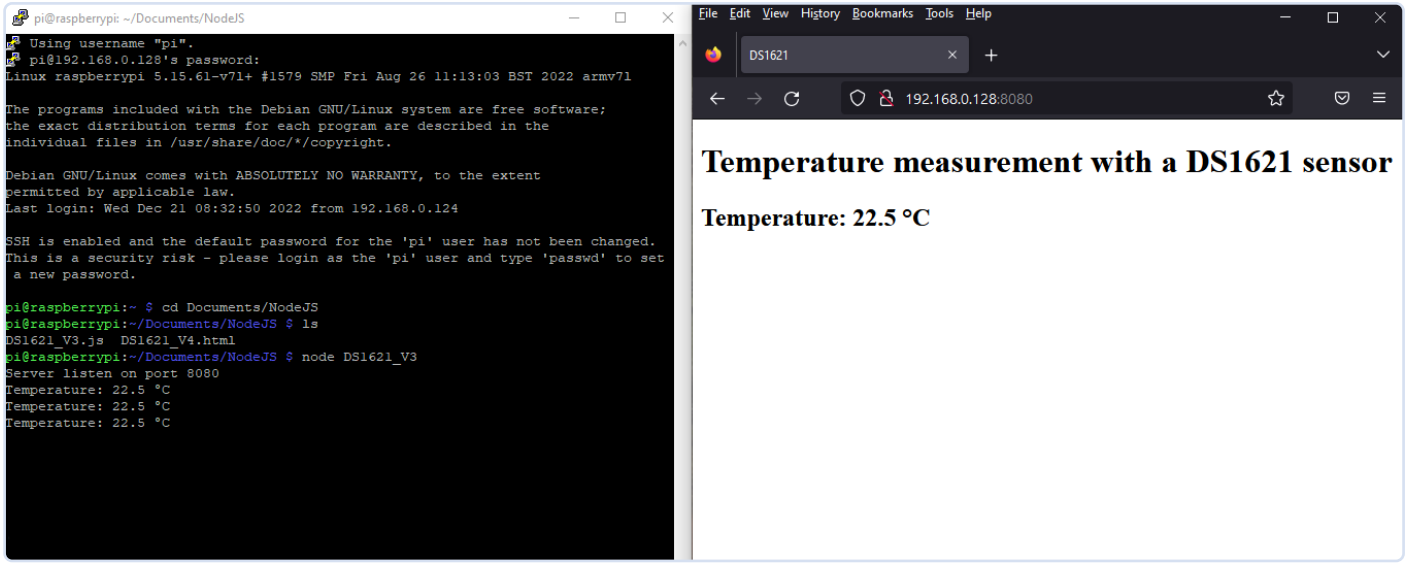


Bild 9. Das Linux-Terminal und der Webbrowser.

In den nächsten fünf Zeilen definieren wir fünf Variablen und vier Konstanten. Diese Konstanten enthalten die Register respektive ihre Adressen des DS1621:

```
var Config, LSByte, MSByte, Temp_Reg, Temperature;

const DS162_Addr = 0x4F;
const DS1621_Temp_Reg = 0xAA;
const DS1621_Config_Reg = 0xAC;
const DS1621_ConvertReg = 0xEE;
```

Als nächstes öffnen wir den I²C-Bus und erstellen ein Objekt namens `DS1621`. Dieses Objekt wird für die Kommunikation mit dem Sensor über I²C verwendet.

```
const DS1621 = I2C.openSync(1);
```

Die nächsten drei Zeilen setzen das *1SHOT*-Bit des DS1621. So wird die Temperatur nur einmal und nur dann gemessen und umgewandelt, wenn der Raspberry Pi als I²C-Master eine Umwandlung anfordert.

```
Config = DS1621.readByteSync(DS162_Addr,
         DS1621_Config_Reg);
Config = Config | 0x01;
DS1621.writeByteSync(DS162_Addr,
         DS1621_Config_Reg, Config);
```

Schauen wir uns nun an, was passiert, wenn der Browser eine Anfrage an den Server stellt (**Bild 9**).
Wenn das Node.js-Skript ausgeführt wird, erscheint im Linux-Terminal nach ein paar Sekunden die Meldung *Server listen on port 8080*. Öffnen Sie einen Webbrowser auf Ihrem PC oder Tablet, der mit dem gleichen WLAN verbunden ist. Geben Sie in der Adresszeile *http://* und die IP-Adresse des Raspberry Pi sowie den Port ein, auf dem der Server lauscht, zum Beispiel *http://192.168.0.128:8080*. Der Port 8080 wurde gewählt, um sich von der Standardeinstellung des Apache-Webservers (Port 80) zu unterscheiden.

1. Wenn Sie die IP-Adresse eingeben, sendet der Browser eine *Get*-HTTP-Anfrage an den Server. Was geschieht nun?
   a. Der Server erkennt die Anfrage mit `Server.get('/'...)` und führt die anonyme Callback-Funktion `function(request, respons)` aus.
   b. Das `request`-Objekt enthält die vom Browser gestellte Anfrage.
   c. Im `respons`-Objekt senden wir zwei Header als Parameter:
      i. Einen *200*-http-Code, was bedeutet, dass die Anfrage erfolgreich war
      ii. Die Zeichenkette *Content-Type': 'text/html'*, was bedeutet, dass der Browser bereit sein muss, Textdaten vom Typ HTML zu empfangen.
   d. Der Aufruf der Funktion `respons.writeHead()` gibt die Parameterdaten an den Browser zurück.

2. Die Funktion `fs.readFile()` liest die Datei *DS1621_V4.html*. Die anonyme Callback-Funktion `function(error, file)` wird ausgeführt, und wenn sie erfolgreich ist (wovon wir ausgehen), wird die Datei gelesen und im `file`-Objekt abgelegt. Der Aufruf der Funktion `respons.end(file)` sendet den HTML-Code an den Browser, der die Webseite anzeigt. Im Fehlerfall wird ein Fehlercode in das `error`-Objekt eingetragen.

Der Browser hat nun also die Webseite im HTML-Format. Wie geht es weiter? Werfen wir einen Blick auf das HTML-Skript (rechts in Bild 8). Das erste, was wir sehen, ist die Zeile `<h2 id="message">Waiting for data...</h2>`. Dies ist ein Platzhalter für die Zeile, in die die Temperatur geschrieben wird. Wir werden später sehen, wie.
Das zweite, was wir sehen, ist der darauf folgende Block mit dem JavaScript, das jedoch clientseitig vom Browser ausgeführt wird:

```
<script src="http://code.jquery.com/jquery-1.10.1.min.
js"></script>

<script Language="Javascript">
  // Call function Get_Tmp() periodically:
  jQuery(document).ready(function() {
         setInterval(Get_Tmp, 5000);
          });

  function Get_Tmp() {
      $.get('/temp')
```

```
        .done(function(Data_Received) {
            $('#message').html(Data_Received);
        });
  }
</script>
```

Zunächst importieren wir aus dem Web die jQuery-Bibliothek, die uns einige interessante Funktionen zur Verfügung stellt. Mit jQuery können wir einen Timer konfigurieren, der alle fünf Sekunden eine Funktion namens `Get_Temp()` aufruft:

```
jQuery(document).ready(function(){
    setInterval(Get_Tmp, 5000);
    });
```

Schließlich haben wir noch die Funktion `Get_Tmp()`:

```
function Get_Tmp() {
$.get('/temp')
    .done(function(Data_Received) {
        $('#message').html(Data_Received);
    });
}
```

Und das ist der interessante Teil des Skripts!
Schauen wir uns an, wie das Skript mit dem Node.js-Skript auf der Serverseite interagiert (siehe die Zeitleiste in der Mitte von Bild 8):

1. Wenn die Funktion `Get_Tmp()` aufgerufen wird, stellt sie eine *Get*-Anfrage an den Server. Was geschieht dann?
   a. Der Server erkennt die Anfrage mit `Server.get('/temp')` und führt die anonyme Callback-Funktion `function(request, respons)` aus.
   b. Der I²C-Master, also der Raspberry Pi, kommuniziert mit dem DS1621 über den I²C-Bus, fragt nach einer Datenumwandlung, liest die Daten und wandelt sie in Grad Celsius um.
   c. Das `request`-Objekt enthält die vom Browser gestellte Anfrage.
   d. Im `respons`-Objekt speichern wir wieder unsere beiden Parameter:
      i. Der Wert *200*, ein Code, der bedeutet, dass die Anfrage erfolgreich war.
      ii. Die Zeichenfolge *'Content-Type': 'text/html'*, was bedeutet, dass der Browser einen Text erhält, der als HTML interpretiert wird.
   e. Die Funktion `respons.writeHead()` gibt diese Daten an den Browser zurück.

2. Der Funktionsaufruf `respons.end("Temperature: " + Temperature + " °C")` sendet die Nachricht *Temperatur: XX °C* (eine ASCII-Zeichenkette) an den Browser. XX ist die vom DS1621 gemessene Temperatur.

3. Diese Zeichenkette wird in das Objekt `Data_Received` gestellt.

4. Erinnern Sie sich an den Header-Abschnitt `<h2 id="message">Warten auf Daten...</h2>`? Wir sehen, dass dieser Header-Tag die ID `"message"` besitzt. Wenn also die Zeichenfolge empfangen wird, wird dank der Zeile `$('#message').html(Data_Received);` die vom Server gesendete Zeichenfolge in den Header-Abschnitt eingefügt und die Webseite zeigt die Temperatur an.

Zum Schluss noch ein paar Worte zu den beiden letzten Zeilen des Node.js-Skripts:

```
Server.listen(8080);
console.log("Server listen on port 8080");
```

Die Zeile `Server.listen(8080);` startet den Server, und die Zeile `console.log("Server listen on port 8080");` schreibt die Meldung *Server listen on port 8080* in die Linux-Konsole (Shell). Schauen Sie sich Bild 8 ruhig etwas genauer an. Ich denke, dass es anhand dieser Zeitleiste besser zu verstehen ist.

(210367-02)RG

Haben Sie Fragen oder Kommentare?

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

Passende Produkte

- **Vincent Himpe, *Mastering the I²C Bus***
  Buch (Paperback), SKU 15385: www.elektor.de/15385
  E-Buch (PDF), SKU 16193: www.elektor.de/16193

- **Raspberry Pi Zero W Starter Kit**
  SKU 18328: www.elektor.de/18328

WEBLINKS

[1] Introduction to Node.js: A Beginner's Guide to Node.js and NPM: https://elektor.link/udaynode
[2] What is Node.js: A Comprehensive Guide: https://simplilearn.com/tutorials/nodejs-tutorial/what-is-nodejs
[3] Modul i2c-bus: https://npmjs.com/package/i2c-bus
[4] SMBus Compatibility With an I²C Device: https://ti.com/lit/an/sloa132/sloa132.pdf

Videoausgabe mit Mikrocontrollern

Teil 2: VGA- und DVI-Ausgabe

Von Mathias Claußen (Elektor-Labor)

Mikrocontroller können auch per VGA-Schnittstelle oder DVI Pixel an einen Monitor ausgeben. Ob beim ESP32 oder dem RP2040 des Raspberry Pi Pico – es sind deutlich mehr als 256 Farben möglich. Dank Sprites, Tiles und ein paar Tricks zaubern Mikrocontroller heute mindestens so anspruchsvolle Grafiken wie klassische Konsolen der 90er Jahre auf den Bildschirm.

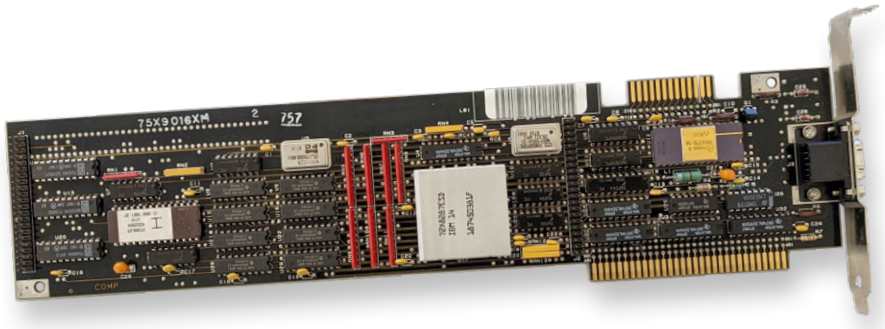


Bild 1. IBM VGA-Karte (Bild: http://nerdlypleasures.blogspot.com).

Im ersten Teil dieses Artikels ging es um die Ausgabe von Composite-Videosignalen im PAL- (**P**hase **A**lternating **L**ine) oder NTSC-Format (**N**ational **T**elevision **S**ystems **C**ommittee). Während die Ausgabe monochromer Bilder auch bei einem Arduino Uno möglich ist, stellen Bilder in Farbe schon höhere Anforderungen. Mit ein paar Tricks lässt sich aber auch ein ESP32 oder gar ein ATmega zur Farbausgabe überreden. Prinzipiell aber ist die Bildqualität bei Composite-Video bescheiden. Es fehlt vor allem Schärfe, und Text ist daher nicht gut lesbar. Die ersten Heimcomputer wie der Sinclair ZX81 oder der Commodore C64 gaben nur Composite-Video aus. Für professionelle Anwendungen wurde aber eine möglichst scharfe Textdarstellung, in manchen Fällen sogar Farbgrafik benötigt. Der IBM PC aus dem Jahr 1987 bot mit seiner VGA-Schnittstelle (**V**ideo **G**raphics **A**rray) daher etwas Neues und Besseres. Auch Mikrocontroller sind in der Lage, VGA-Signale zu erzeugen und so ein scharfes und farbiges Bild zu generieren. Einige Mikrocontroller können das Bild sogar komplett digital per DVI (**D**igital **V**isual **I**nterface) ausgeben. In diesem zweiten Teil geht es daher um VGA und DVI sowie die Tricks, die für die Grafikausgabe und Grafikeffekte eingesetzt werden.

VGA: Video für IBM-PC

VGA war die 1987 von IBM vorgestellte Grafikkarte (**Bild 1**) für PS/2-PCs. Diese Grafikkarte machte die damals neue dreireihige Sub-D-15-Buchse (**Bild 2**) populär. VGA definiert aber nicht nur die physikalischen Steckverbindung, sondern auch das Signaltiming und die nötigen Signale.

Die Farbsignale sind bei VGA analog und werden auf drei Signalleitungen übertragen (Rot, Grün und Blau). Die horizontale und vertikale Synchronisation operiert digital mit TTL-Pegeln. Die erste VGA-Karte von IBM konnte schon 256 Farben aus einer RGB-Palette von 3 x 6 bit (= 262.144 Farben) bei einer Auflösung von 320 x 200 Pixel ausgeben und schaffte bei 16 Farben immerhin 640 x 480 Pixel. Die ersten hierfür passenden IBM-Monitore konnten nur 640 x 480 oder 640 x 400 Pixel bei einer Bildwiederholrate von 60 Hz oder 70 Hz verarbeiten. IBMs VGA-Karten wurden sehr schnell von vielen anderen Herstellern geklont, wodurch VGA

Bild 2. VGA-Buchse an einem PC (Bild: Dr. Thomas Scherer).

Bild 3. Geöffneter Fernseher mit Kathodenstrahlröhre (Bild: Shutterstock / Sergio Sergo).

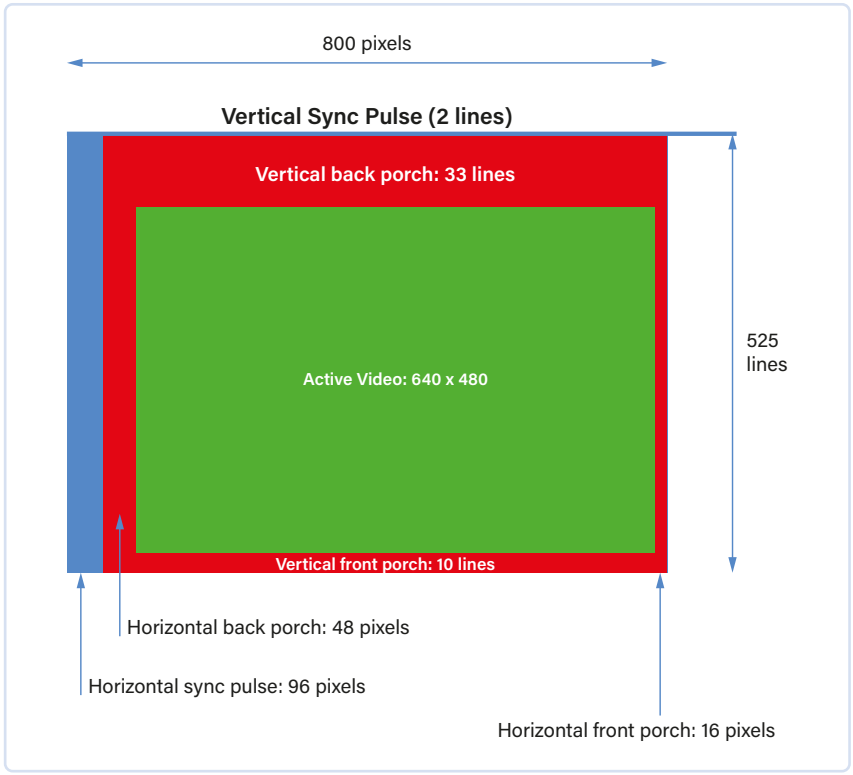


Bild 4. Timing eines VGA-Signals mit 640x480 sichtbaren Pixel (Quelle: https://tinyurl.com/bdf9xuf7).

zu einem De-facto-Standard für die Videoausgabe sogenannter IBM-kompatibler PCs wurde.

Sichtbare und unsichtbare Pixel

Ein Videosignal mit 640 x 480 sichtbaren Pixeln bei einer Bildwiederholrate von 60 Hz operiert mit einer Pixelfrequenz von 25,175 MHz. Die Multiplikation der Pixel eines Bildes mit der Bildfrequenz ergibt aber lediglich 18,432 MHz. Was ist mit der Differenz? Wie Composite-Signale nach PAL oder NTSC gibt es auch bei VGA nicht sichtbare Bereiche, denn auch dieser Standard wurde für die Anzeige mit Monitoren konzipiert, die auf Kathodenstrahlröhren (**Bild 3**) basieren. Grundsätzlich gelten hier ähnliche Timing-Anforderungen wie bei der Ansteuerung der Bildröhre eines Fernsehers.

Bild 4 zeigt das Timing für den Modus mit 640 x 480 Pixel und einem Pixeltakt von 25,175 MHz. Ein Bild beginnt mit einem vertikalen Synchronisationsimpuls von 63,5551142 µs (= zwei Bildzeilen mit 800 Pixel). Die drei Elektronenstrahlen sind in dieser Zeit ausgetastet, da sie sich während dieser Zeit vom Ende des letzten Bildes rechts unten zum Anfang des neuen Bildes links oben bewegen. Die Zeilen beginnen mit einem horizontalen Synchronisationsimpuls von 69 Pixel (≈ 3,81 µs). Bei den nächsten 33 Zeilen (Zeilennummern 3...35) kommt nach dem horizontalen Synchronisationsimpuls der sogenannte „vertical back porch" von hier 704 Pixel (≈ 27,96 µs). Er soll den Elektronenstrahlen beziehungsweise der magnetischen Ablenkung genug Zeit geben, wieder in die Ausgangsposition für das neue Bild zu gelangen. Anschließend folgen 480 Zeilen mit der eigentlichen Bildinformation, beginnend mit dem horizontalen Sync-Impuls von 96 Pixel und dann dem „horizontal back porch" von 48 Pixel (≈ 1,9 µs), plus die 640 Bild-Pixel einer Zeile (≈ 25,42 µs) und dem abschließenden „horizontal front porch" von 16 Pixel (≈ 0,64 µs). **Bild 5** zeigt den Aufbau einer solchen Bildzeile. Zum Schluss kommen noch zehn Zeilen mit je einem horizontalen Sync-Impuls gefolgt vom „vertical front porch" (wieder aus je 704 Pixel). Das übertragene Bild besteht also bei effektiven 640 x 480 Pixel bezüglich Timing aus 800 x 525 Pixel.
Die Elektronenstrahlen sind nur während des eigentlichen Bildes aktiv und bleiben sonst ausgetastet. Auch wenn sich das sprachlich komplex anhört, ist es doch eines der einfachsten Bildsignale, die durch einen

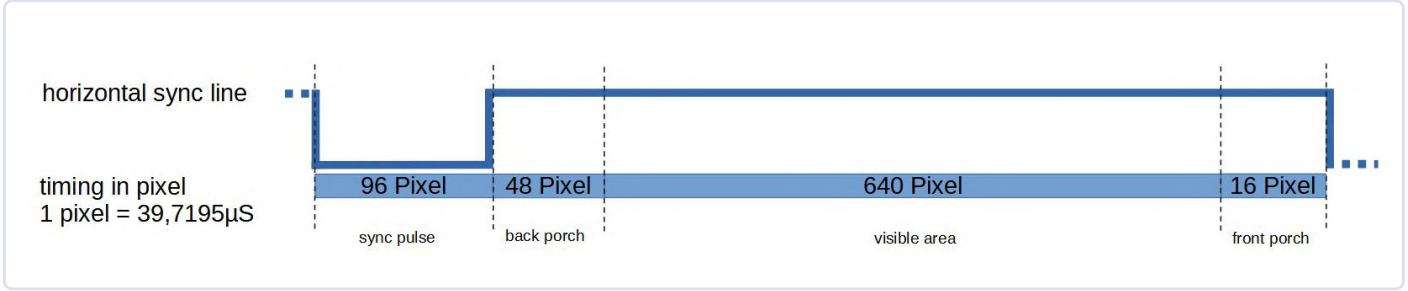


Bild 5. Timing einer Zeile in VGA.

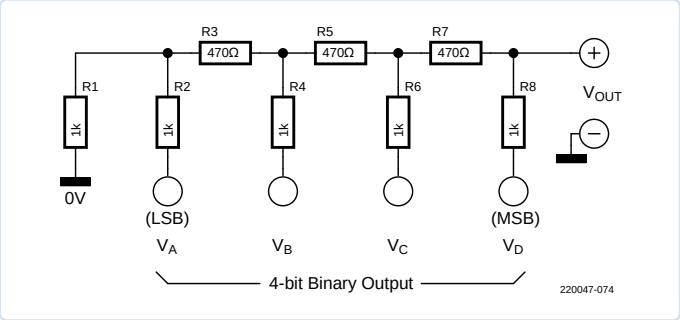


Bild 6. DAC mit R2R-Netzwerk.

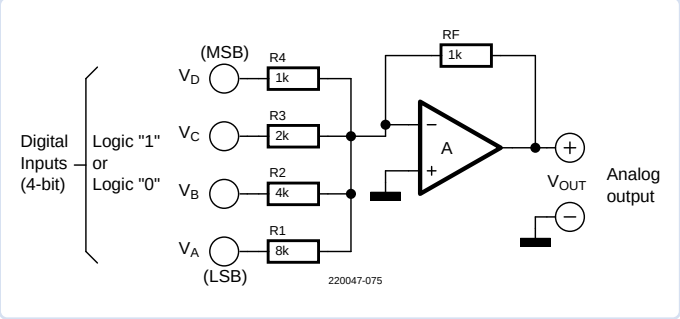


Bild 7. Binär gewichteter DAC mit Opamp (Quelle: https://tinyurl.com/waam99c4).

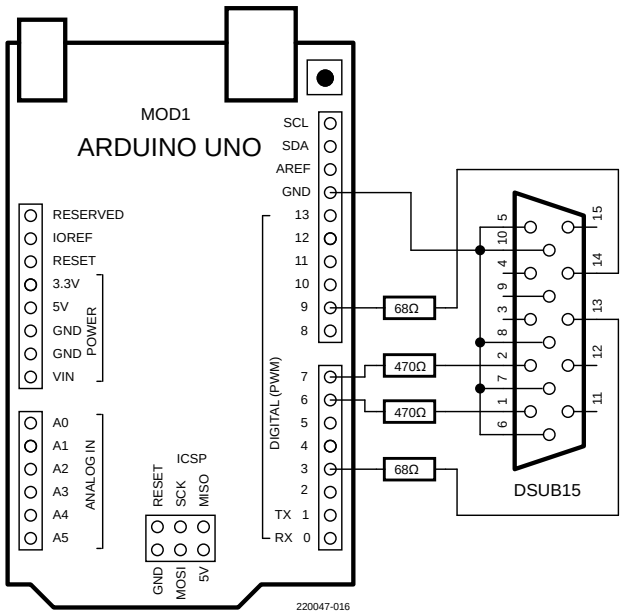


Bild 8. Beschaltung mit einem VGA-Ausgang beim Arduino Uno.

Mikrocontroller generiert werden können. Damit Sie sich nicht wundern: Im Internet finden sich auch leicht abweichende Timing-Angaben für den VGA-Bildaufbau. Und wenn man den Pixeltakt von 25,175 MHz durch die 800 x 600 Pixel teilt, kommen dabei nicht genau 60 Hz für die Bildfrequenz heraus, sondern ≈ 59,94 Hz. Doch die analoge Elektronik der zugehörigen Monitore nimmt das nicht so genau und würde auch noch bei größeren Abweichungen von 60 Hz sauber synchronisieren.

VGA-Signalpegel

Laut VGA-Spezifikation beträgt der Spannungsbereich der analogen RGB-Bildsignale 0...0,7 V, wobei 0,7 V der maximalen Helligkeit entspricht. Die horizontale und vertikalen Synchronisationssignale haben TTL-Pegel (0 bis 5 V).

Mikrocontroller mit Betriebsspannungen von 2,5 V bis 5 V können die Signale für die vertikale und horizontale Synchronisation direkt durch ihre I/O-Pins ausgeben, da 2,5 V zuverlässig als TTL-high-Pegel interpretiert werden. Für die Ausgabe der Farbinformationen muss man dann Spannungsteiler einsetzen.

Da die meisten Mikrocontroller intern nicht genügend schnelle DACs haben, werden in der Regel die GPIO-Pins mit Hilfe von Widerständen als einfache DACs für die Generierung der drei analogen Signale genutzt. Bei der Berechnung der nötigen Widerstände muss beachtet werden, dass die analogen RGB-Eingänge des Monitors mit 75 Ω terminiert sind. Als DAC kommen hier meistens R2R-Netzwerke (**Bild 6**) oder binär gewichtete Widerstände (**Bild 7**) zum Einsatz.

VGA mit ATmega

VGA-Signale lassen sich auch mit ATmega-Mikrocontrollern erzeugen. Von daher kann auch ein Arduino Uno mit VGA-Grafik ausgestattet werden. Mit vier Widerständen und einer 15-poligen SUB-D-Buchse kann ein Arduino Uno immerhin 120 x 60 Pixel in vier Farben darstellen. Die zugehörige Arduino-Bibliothek ist *VGAX* [1] und kann auf der GitHub-Seite des Projekts heruntergeladen werden.

Wird ein Arduino Uno wie in **Bild 8** beschaltet, so können sogar Grafiken in vier Farben wie in **Bild 9** ausgegeben werden. Limitierend ist hierbei die Menge an Speicher, die ein Arduino Uno bieten kann. Da die Farbsignale durch die GPIO-Pins generiert werden, müssen diese auch schnell genug mit neuen Werten zu versorgen sein.

Das Timing zur Signalerzeugung ist hier durch Interrupts gelöst. Ein Nebeneffekt davon ist,

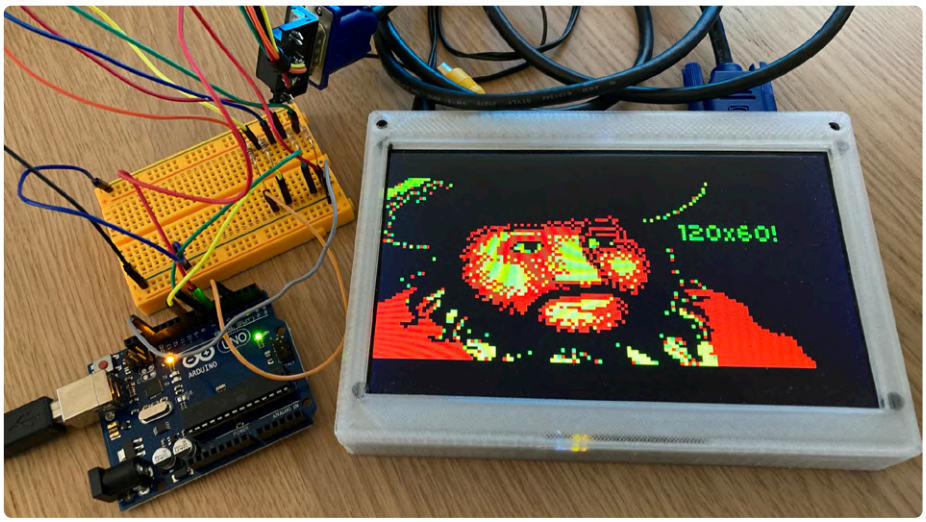


Bild 9. VGA-Ausgabe auf einem Bildschirm mit dem Arduino Uno.

dass Interrupts anderer Peripherie eventuell mit denen für die Signalerzeugung kollidieren könnten. Timer1 und Timer2 generieren hier die horizontalen und vertikalen Synchronisationsimpulse. Timer0 wird verwendet, um den Jitter bei den Interrupt-Aufrufen zu umgehen. Die Bibliothek *VGAXUA* [2] für Arduino Uno und Arduino Mega kann monochrom von 192 x 80 bis 200 x 240 Pixel ausgeben. Um Rechenzeit zu sparen, wird das Bild hier mit Hilfe des USART generiert. Jedes Byte im (Bild-)Speicher entspricht dann jeweils acht Pixel. Ältere Lösungen zur VGA-Ausgabe mit AVR-Chips haben entweder den Mikrocontroller bis zu 32 MHz übertaktet [3] oder aber externen Speicher für die Videoausgabe verwendet [4].

VGA mit ESP32

Auch ein ESP32 kann VGA. Ideale Voraussetzungen bieten sein Takt von 240 MHz und die flexible I/O-Matrix, die eine Ausgabe digitaler Signale mit bis zu 80 MHz erlaubt. Der interne Speicher des EPS32 mit 520 KB reicht für eine Ausgabe von 640 x 480 Pixel bei 256 Farben (= 307.200 Byte). Dabei bleibt genug RAM für andere Software übrig.

Die Timing-Angaben für ein VGA-Signal mit 640 x 480 Pixel und 60 Hz finden sich unter [5]. Das Projekt „ESP32 VGA" [6] zeigt einen interessanten Weg, ein VGA-Signal mit dem internen I²S-Controller (**I**nter-**IC** **S**ound) [6] zu erzeugen. Der I²S-Controller des ESP32 kann mehr als nur die Ausgabe von Audiodaten [7]: Er bietet zusätzliche Betriebsmodi als Kamera- und LCD-Interface (**Bild 10** zeigt den entsprechenden Teil der Blockschaltung). Das LCD-Interface des I²S-Controllers kann bis zu 24 bit breite Busse bereitstellen. Das würde sogar für TrueColor (16.777.216 Farben) reichen, wenn die kompletten 24 Bit für die Ausgabe von Farbe genutzt werden könnten. Ein VGA-Signal benötigt aber neben den RGB-Signalen noch die horizontale und vertikale Synchronisation, sodass nur 22 bit für Farbe zur Verfügung stehen (**Bild 11**).

Doch es gilt noch zwei Hürden zu überwinden: Ein Bild mit 640 x 480 Pixel belegt bei 24 bit Farbe 921.600 Byte an RAM – das ist mehr als ein ESP32 mitbringt. Außerdem muss ein Pixeltakt von 25,175 MHz erreicht werden. Für die Ausgabe von 24 Bit muss der I²S-Controller immer 32 Bit aus dem Speicher lesen. Es scheint, dass der I²S-Controller im ESP32 die 32 bit breiten Worte mit maximal

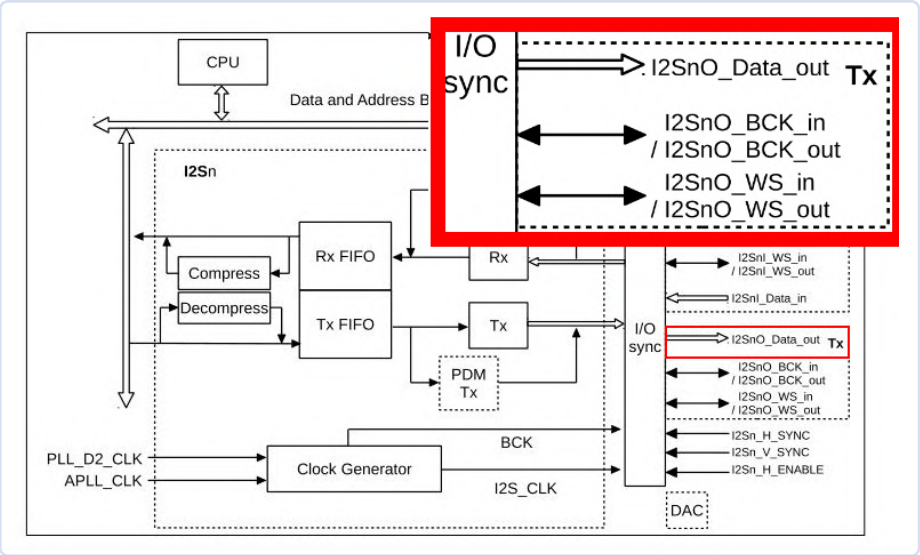


Bild 10. I²S-Controller des ESP32 (Quelle: Espressif / https://tinyurl.com/yrrbnjak)

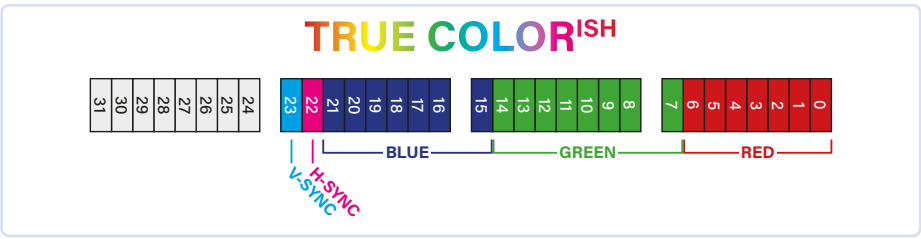


Bild 11. 22-bit-Videoausgabe mit H- und V-Sync beim ESP32 (Quelle: https://tinyurl.com/2xjv9tux).

10 MHz ausgeben kann, was nicht ausreicht. Die serielle Ausgabe der Pixel ist daher auf 10 MHz begrenzt. Die maximale Auflösung ist hingegen durch den internen Speicher begrenzt.

Unter [8] findet sich eine Auflistung der Pixel-Taktraten in Abhängigkeit von der VGA-Auflösung. Beim Modus mit 800 x 600 Pixel bei 60 Hz beträgt der Pixeltakt mit 40 MHz genau das Vierfache dessen, was ein ESP32 ausgeben kann. Es wären also rechnerisch 200 x 600 Pixel bei 60 Hz und 22-bit Farbe bezüglich Timing und RAM möglich. Allerdings wären diese Pixel nicht mehr rechteckig, sondern horizontal gestreckt. Für rechteckige Pixel muss die vertikale Auflösung geviertelt und jede Bildzeile vier Mal ausgegeben werden. Das Resultat sind 200 x 150 Pixel in TrueColor. Ein Bild mit theoretischen 800 x 600 Pixel (**Bild 12a**) wird so ziemlich grob, aber schön farbig (**Bild 12b**) angezeigt.

Bild 12. Aus 800 x 600 (a) werden 200 x 150 (b) bzw. 400 x 360 Pixel (c). Die Bilder b und c sind zur besseren Vergleichbarkeit auf die gleiche Größe wie Bild a skaliert.

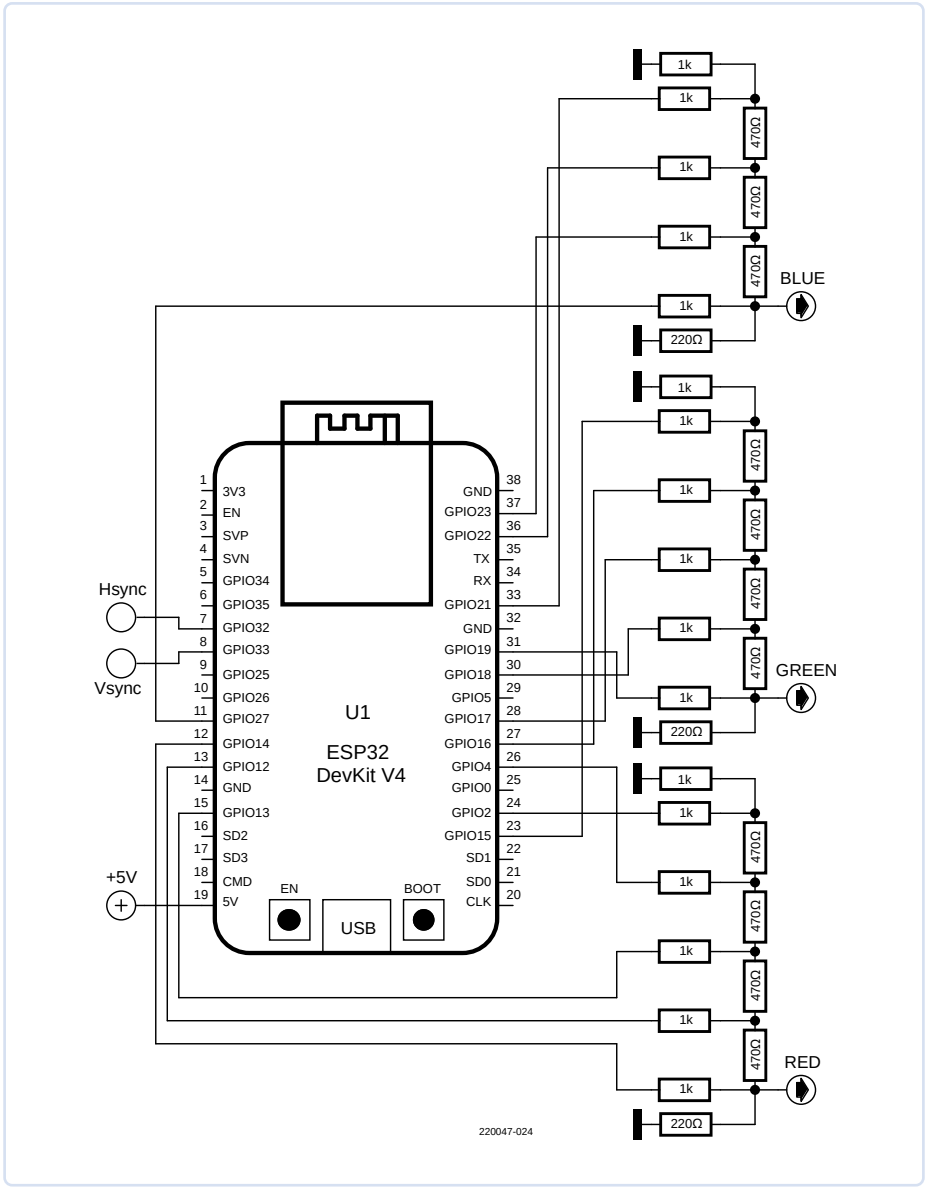


Bild 13. VGA-Signalerzeugung per R2R-DAC beim ESP32.

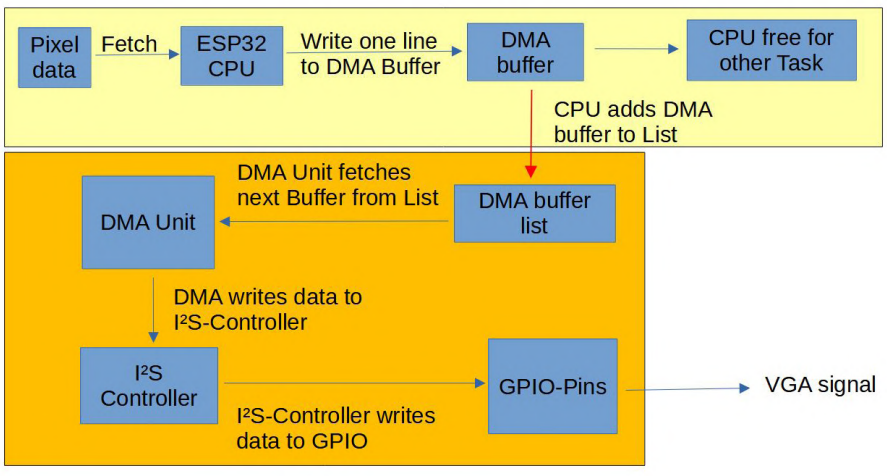


Bild 14. Pixelausgabe beim ESP32.

Es gibt aber einen brauchbaren Kompromiss, denn der I²S-Controller kann auch 16-bit-Worte verarbeiten. Von den 16 bit stehen dann 14 bit für die Bildinformation und 2 bit für die Synchronisationssignale zur Verfügung, was 16.384 darstellbare Farben ermöglicht. Für ein stabiles VGA-Signal sollte die APLL des ESP32 genutzt werden. Dies erlaubt eine stabile Zeitbasis für das VGA-Timing. Auf diese Weise sind Bilder in 14 bit Farbe mit Auflösungen von 320 x 240, 360 x 400 und 480 x 400 Pixel realisierbar.

Zur Ausgabe der RGB-Signale müssen die 14 bit Farbinformation D/A-gewandelt werden. Ein R2R-DAC ist eine einfache und effektive Lösung für diesen Zweck. Die Schaltung von **Bild 13** zeigt, wie man solch einen Ansatz umsetzen kann. Die Grafik aus Bild 12a sieht mit einer Auflösung von 360 x 400 Pixel (**Bild 12c**) deutlich besser aus.

Um die Daten aus dem RAM zu holen und anzuzeigen gibt es noch ein paar Tricks: Zum Transport der Daten zum I²S-Controller wird der DMA-Controller (**D**irect **M**emory **A**ccess) des EPS32 verwendet, um die CPU zu entlasten. Er kann Daten CPU-unabhängig von einer Stelle zu einer anderen transportieren. Als Angaben genügen hierfür Quelle, Ziel, Datenmenge, Transferart und die Transfer-Startbedingung.

Der ESP32 bereitet die Daten für jede Zeile vor und legt diese in einen Zeilenpuffer ab. Neben den sichtbaren Pixeln werden auch die Informationen für die horizontale und vertikale Synchronisation eingefügt. Ist eine Zeile fertig aufbereitet, transferiert der DMA-Controller die Daten vom Puffer an den I²S-Controller. Die CPUs des ESP32 können in dieser Zeit andere Aufgaben übernehmen. Ein grober Ablauf des Vorgangs ist in **Bild 14** dargestellt. Dieser Ablauf ermöglicht auch das Ausgeben von Grafiken ohne Vorhaltung des kompletten Bildes im Speicher. Dazu muss jedoch die CPU des ESP32 während der Ausgabezeit einer Bildzeile die Pixel für die nächste Zeile berechnen können.

Die VGA-Bibliothek von bitluni für das ESP32-Arduino-Framework steht auf GitHub [9] bereit. Die Library *ESP32Lib* kann aber deutlich mehr als nur die VGA-Ausgabe. Auch Funktionen für Sprites [10] und ein 3D-Renderer sind darin enthalten. Außerdem ermöglicht die Library *FabGL* [11] nicht nur eine VGA-Ausgabe mit dem ESP32, sondern stellt Zutaten für ein komplettes System aus Grafik

und Ton bereit. Es gibt sogar Video-Tutorials für *FabGL* auf YouTube [12].

VGA mit Raspberry Pi Pico

Sein SoC RP2040 macht den Raspberry Pi Pico nicht nur klein, sondern sehr vielseitig. Mit zwei CPU-Kernen, internen 264 KB RAM und externem Flash ist dieses Board preiswert und gleichzeitig für viele Projekte geeignet. Schon beim Erscheinen des Raspberry Pi Pico wurde die Erzeugung von VGA-Signalen mit einem passenden Demo-Board (**Bild 15**) vorgestellt. Inzwischen wurde das Board aktualisiert, so dass man auch einen Raspberry Pi Pico W verwenden kann. Die nötigen Files für das Board [13] sind als KiCad-Projekt vorhanden, sodass einem Nachbau nichts im Weg steht.

Der RP2040 bietet genug Platz, um ein 320 x 240 Pixel großes Bild bei 16 bit (= 65.536 Farben) komplett im RAM vorzuhalten. Die PIO-Statemachines des RP2040 sind ideal für diesen Zweck.

Für das VGA-Signal sind Werte für die RGB-Pegel sowie für die horizontale und vertikale Synchronisation notwendig. Für die Erzeugung dieser Signale mit den PIO-Statemachines des RP2040 ist nur etwas Software nötig. Der RP2040 bringt aber leider keine D/A-Wandler mit, sodass eine externe Beschaltung erforderlich ist.

Hierzu eignet sich entweder ein einfacher R2R-DAC oder ein binär gewichteter DAC. Bei einem R2R-DAC werden nur zwei Widerstandswerte benötigt, was die Sache sehr vereinfacht. Für einen binär gewichteten DAC wird für jedes Bit ein anderer Widerstandswert benötigt. Die Schaltung für das Demoboard mit VGA-Ausgang von **Bild 16** stammt aus einer PDF-Datei mit dem Titel „Hardware design with Rp2040" [14] der Raspberry Pi Ltd. Für Farben nach RGB565 werden jeweils fünf GPIO-Pins für die Farben Rot und Blau verwendet, Grün benötigt sechs Pins für die resultierenden 65.536 Farben. Die Werte der Widerstände ergeben sich aus dem Aufbau des DAC, der nötigen Ausgangsspannung und den Abschlusswiderständen des Monitors.

Die Werte der analogen RGB-Signale können zwischen 0 und 0,7 V liegen. Die RGB-Eingänge eines Monitors sind mit 75 Ω nach Masse terminiert. Daher müssen die Spannungsteiler aus den 3,3 V der GPIOs maximal 0,7 V für die drei Farben erzeugen.

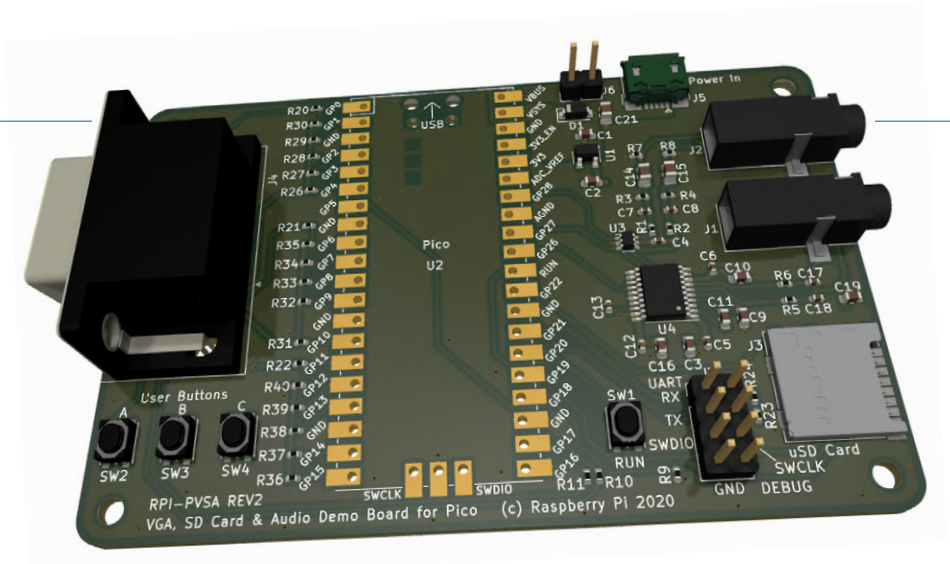


Bild 15. VGA-Base-Board für Raspberry Pi Pico.

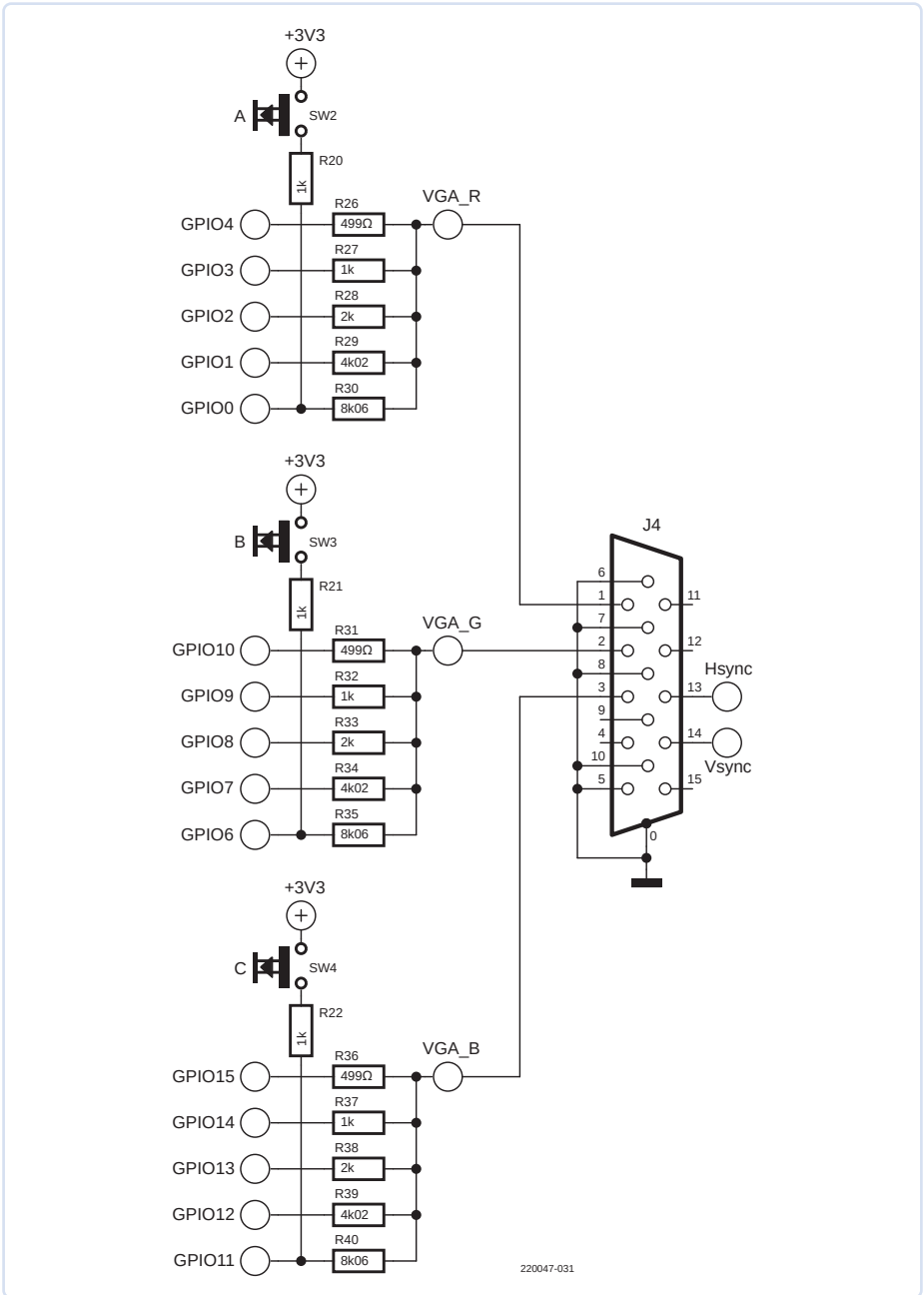


Bild 16. Ausschnitt aus der Schaltung der VGA-Base für den Raspberry Pi Pico.

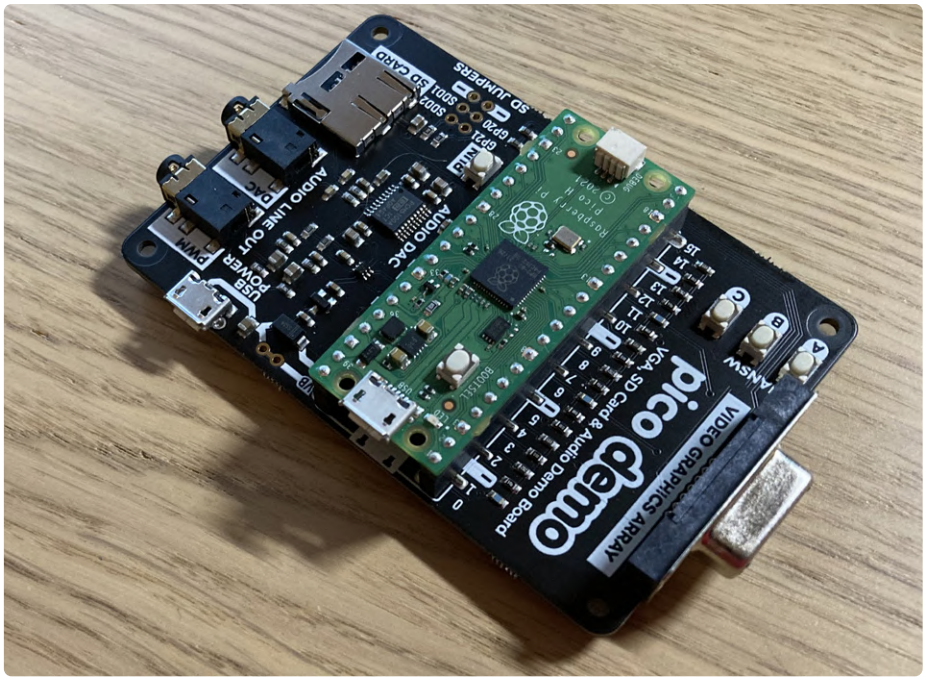


Bild 17. Pimoroni VGA-Base für den Raspberry Pi Pico.

Daraus ergibt sich das Verhältnis der Widerstände zu 1:2:4:8:16. Um von 3,3 V auf 0,7 V zu kommen, ist ein Spannungsteiler mit einem Verhältnis von 3,7:1 erforderlich. Es werden fünf oder sechs Widerstände pro Farbe parallelgeschaltet. Die Parallelschaltungen müssen 278 Ω ergeben, um bei einer Versorgung mit 3,3 V am 75-Ω-Abschlusswiderstand auf maximal 0,7 V zu kommen.

Mit Widerständen aus der E-96-Reihe sind Werte von 499 Ω, 1000 Ω, 2000 Ω, 4020 Ω und 8060 Ω mit 1 % Toleranz erhältlich. Parallelgeschaltet ergeben sich so 258 Ω, was zu ausreichend genauen 0,74 V am Abschlusswiderstand führt.

Wer ein fertig bestücktes Board haben möchte, kann auch zum Pimoroni Pico VGA Demo Base greifen (**Bild 17**, siehe auch den Kasten **Passende Produkte**). Neben dem VGA-Ausgang sind hier ein Audio-Codec sowie ein SD-Karten-Slot vorhanden.

Pixelausgabe mit RP2040

Damit ein VGA-Signal ausgegeben werden kann, müssen die Bilddaten generiert und gespeichert werden. Das RAM im RP2040 reicht aus, um ein Bild mit 320 x 240 Pixel bei 16-bit-Farbe vorzuhalten. Im GitHub Repository *pico-playground* [16] finden sich passende Demos.

Für eine einfache Ausgabe werden 150 KB Speicher reserviert, in dem die Bilddaten als RGB565-Werte gespeichert werden. Dabei wird das Bild Zeile für Zeile aus dem Speicher gelesen und dann für jede Zeile das passende Timing für die Bildausgabe erzeugt. **Bild 18** zeigt einen vereinfachten Ablauf.

Diese Art Bildausgabe aus dem RAM sorgt dafür, dass beide CPUs für andere Aufgaben frei bleiben. Es muss nur eine neue Bildzeile mit passendem Timing bereitgestellt werden. Die hierfür zuständige Bibliothek *pico_scanvideo* ist in [17] zu finden. Sie erlaubt es, mehr als eine Bildzeile im Voraus bereitzustellen. Die Anzahl an Bildzeilen ist wie die Skalierung der Bildschirmauflösung konfigurierbar. Damit kann man zum Beispiel 320 x 200 Pixel in einer Auflösung von 640 x 480 Pixel ausgeben. Mit dem Puffern der Bildzeilen lassen sich auch andere Tricks wie etwa das Generieren von Bildern „on the fly" erreichen. Dabei handelt es sich quasi um ein Rennen gegen die Zeit beziehungsweise gegen den Elektronenstrahl.

Raspberry Pi Pico als Zeichenkünstler

Der Raspberry Pi Pico und andere, auf dem RP2040 basierende Boards können wahre Zeichenkünstler sein. In **Bild 19** bewegen sich mehrere Sprites über den Bildschirm. Die Ausgabe erfolgt mit 640 x 480 Pixel bei 16 bit Farbe. Das ist mehr als eigentlich in den RAM des Raspberry Pi Pico passen würde.

Bild 19. Bewegte Sprites.

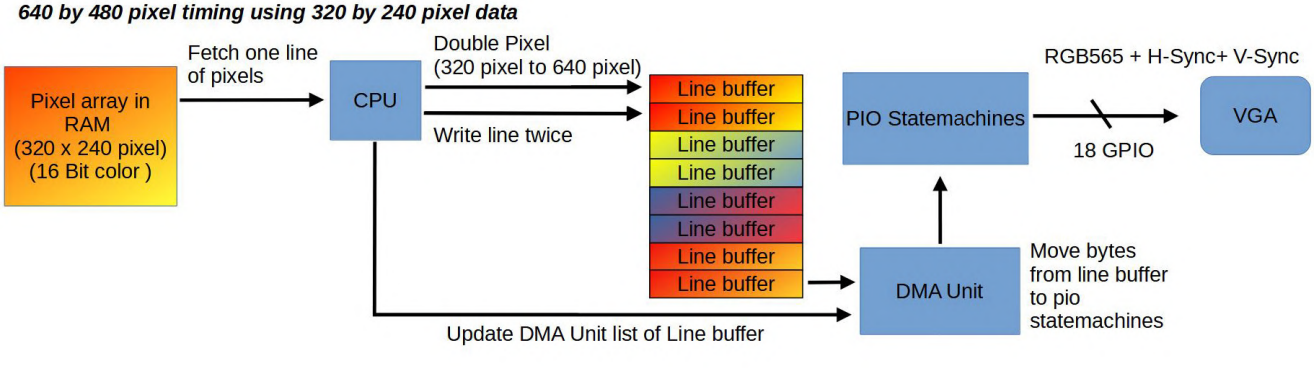


Bild 18. Ablauf der VGA-Ausgabe auf dem Raspberry Pi RP2040.

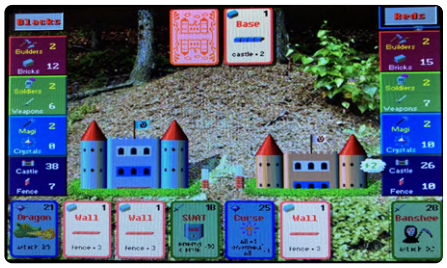


Bild 20. Ants auf dem Raspberry Pi Pico.

Bild 21. Pac-Man auf dem Raspberry Pi Pico.

Bild 22. Pseudo-3D mit dem Raspberry Pi Pico.

Dazu werden alle Pixel bei jedem neu auszugebenden Bild erneut gezeichnet.
Während diese Demo grundlegender Funktionen 2D-Grafik auf den Bildschirm zaubern kann, geht die PicoVGA-Bibliothek von Miroslav Nemecek noch ein paar Schritte weiter. Die Grafikausgabe wird auf 8 bit beschränkt, was mehr freie IO-Pins erlaubt sowie mit 75 KB statt 150 KB weniger RAM für ein komplettes Bild benötigt. Im GitHub-Repository der Bibliothek finden sich etliche Demos und kleine Spiele [18]. **Bild 20** zeigt das auf dem Raspberry Pi Pico laufende Spiel Ants. In **Bild 21** ist der Klassiker Pac-Man zu bewundern. Auf der deutschen Mag-Pi-Seite [19] findet sich eine kurze Beschreibung der Bibliothek.
Im Raspberry Pi Pico stecken Funktionseinheiten, die sogar Pseudo-3D-Effekte (**Bild 22**) erlauben. Diese Darstellungsart wurde einst durch SNES (**S**uper **N**intendo **E**ntertainment **S**ystem) mit Spielen wie Super Mario Kart (**Bild 23**) oder F-Zero als Mode-7-Grafik [20] bekannt. Die zugehörige Bibliothek gibt es auf GitHub unter [21].

DVI

DVI ist eine 1999 vorgestellte Schnittstelle [22], die Bilddaten ohne Qualitätsverluste digital an einen Monitor liefert. Eine typische weiße Buchse für zweikanaliges DVI-D ist in **Bild 24** zu sehen. Die genauen DVI-Spezifikationen finden sich in einer PDF-Datei unter [23].
Als die ersten LCD-Monitore erschwinglich wurden, gaben noch viele Grafikkarten die Bildinformationen über die VGA-Schnittstelle aus. Doch bei Full HD (1920 x 1080 Pixel) wurde das Bild dank analoger Signale nicht richtig scharf. Mit DVI gab es bald eine verbreitete digitale Schnittstelle, die einen LCD-Monitor direkt, ohne Wandlungsverluste mit hochauflösenden Bildern in 24 bit Farbe versorgen konnte.

8b/10b, TMDS und 1:10 Takt

Während bei VGA noch drei der fünf Signale analog waren, ist bei DVI alles digital. Vier differentielle Leitungspaare (±Data 0, ±Data 1, ±Data 2 und ±Clock) stellen bei DVI das Minimum einer einkanaligen Bildübertragung dar.
Die RGB-Bilddaten werden logischerweise via Data 0, Data 1 und Data 2 übertragen. Für ein Bild in 24 bit Farbe wird jede Farbe mit 8 bit übertragen. Hinzu kommt ein Taktsignal, das hilft die Farben auf der Empfängerseite zu kompletten Pixeln kombinieren zu können.
Die digitale Übertragung erfolgt im TMDS-Verfahren (**T**ransition-**M**inimized **D**ifferential **S**ignaling) [24], das durch seine Datencodierung für einen robusten Signaltransport und geringe elektromagnetische Abstrahlung sorgt.
Dabei wird die sogenannte 8b/10b-Codierung [25] angewendet, bei der von 1.024 möglichen Kombinationen nur 460 genutzt werden, um die 8-bit-Informationen für die drei Farben Rot, Grün und Blau zu kodieren. Etliche 8-bit-Werte können so durch zwei Kombinationen dargestellt werden. Vier Kombinationen werden für die Signalisierung von C0 und C1 zur horizontalen und vertikalen Synchronisation verwendet. Bei der Ausgabe der Pixel für eine Zeile werden die Anzahl von Nullen und Einsen im Datenstrom ausgeglichen, um gleichspannungsneutrale Datenströme zu erhalten, was durch die doppelte Repräsentation einiger 8-bit-Werte möglich wird.
Auch wenn die Bildinformationen komplett digital übertragen werden, sind noch horizontale und vertikale Synchronisationssignale vorhanden. Doch sind das keine expliziten Datenleitungen mehr, sondern sie werden mit Hilfe von C0 und C1 im Datenstrom von Data 0 codiert. Der DVI-Datenstrom ist so konzipiert, dass er einfach in ein analoges VGA-Signal gewandelt werden kann.
Das Taktsignal bei DVI entspricht nicht der Bit-Rate für Rot, Grün und Blau, sondern erfolgt im Verhältnis 10:1 und entspricht durch die 8b/10b-Codierung dem Pixeltakt. Für die mit 640 x 480 Pixel bei 60 Hz kleinste DVI-Auflösung ist wie bei VGA ein Pixeltakt 25,175 MHz bzw. eine Bitrate von 251,75 MHz für die Farbinformationen nötig. Die VGA-Ursprünge wirken sich auch noch bei DVI aus.

Bild 23. Super Mario Kart mit Pseudo-3D-Effekt.

Bild 24. DVI-D-Buchse an einem PC (Bild: Dr. Thomas Scherer).

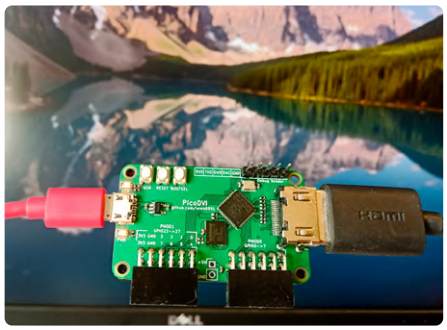


Bild 25. RP2040 mit DVI-Videoausgabe (Bild: Wren6991 / https://tinyurl.com/32t5rh9r).

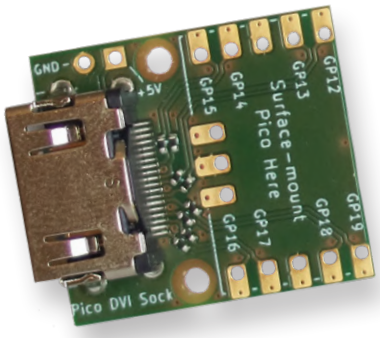


Bild 26. Pico DVI Sock für Raspberry Pi Pico.

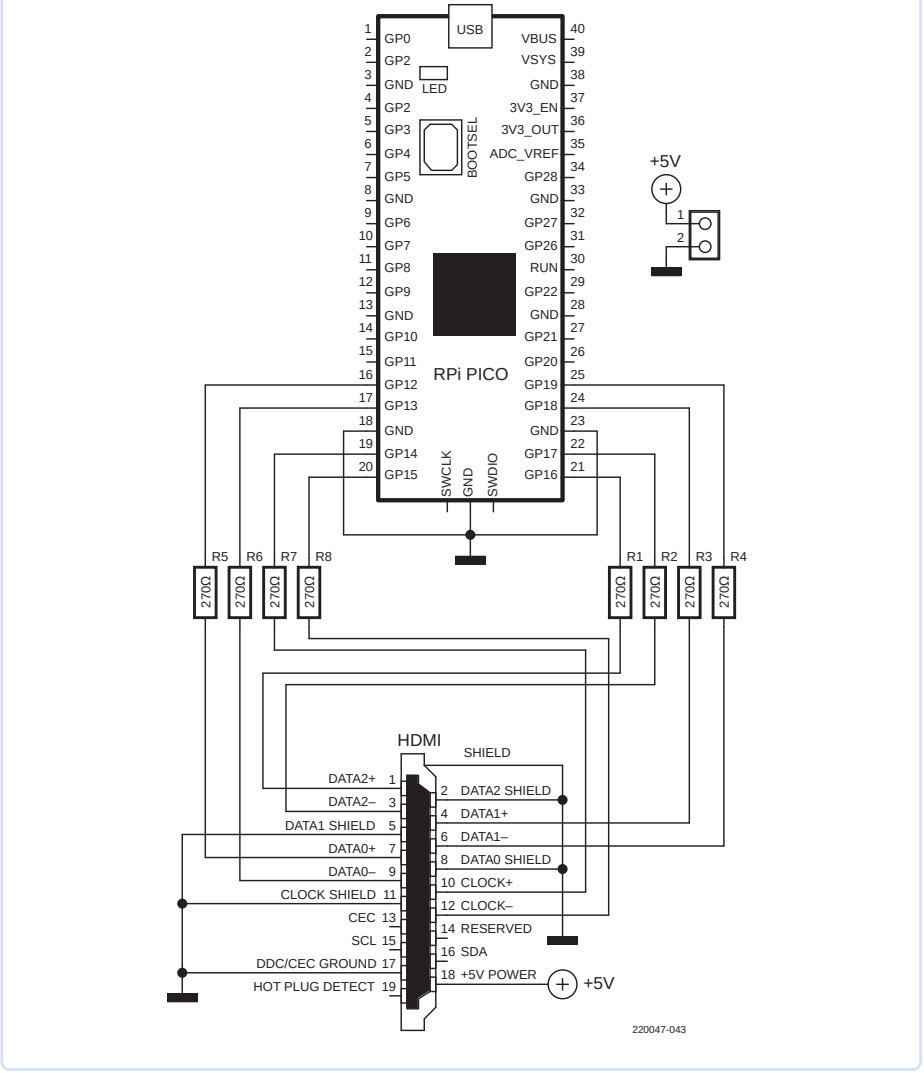


Bild 27. Schaltung des Pico DVI Sock.

DVI mit Raspberry Pi Pico

Geht man von der zuvor angegebenen, kleinstmöglichen Auflösung aus, muss ein Raspberry Pi Pico drei Datenströme mit 251,75 Mbit/s ausgegeben. Das ist mit seinem Maximaltakt von 133 MHz schlicht nicht möglich, denn pro Taktschritt müssten ja fast 2 bit für jeden der drei Farb-Datenströme ausgeben werden.

Schon an dieser Hürde könnte der Raspberry Pi Pico also scheitern. Doch der RP2040 ist extrem übertaktungsfreudig. Ein Takt von 251,75 MHz für beide Kerne ist ohne erkennbare Funktionsstörungen bei Raumtemperatur möglich. Allerdings müssen auch die GPIO-Pins des RP2040 dieses enorme Tempo mitmachen. Hinzu kommt, dass die Bilddaten noch als TMDS-Stream codiert sein und die Daten daher passend aufbereitet werden müssen. Luke Wren konnte zeigen [26], dass ein RP2040 dazu in der Lage ist (siehe **Bild 25**); lesen Sie hierzu auch das Interview unter [27].

Einer der beiden Kerne des RP2040 ist zu etwa 60 % mit TMDS-Encoding beschäftigt. Außerdem werden drei der acht PIO-Statemachines zur Ausgabe der Daten mit den nötigen 251,75 MHz eingesetzt. Ein paar Funktionen des DMA-Controllers bringen den CPU-Kernen Entlastung beim Datentransport. Auf diese Weise bleibt ein Kern komplett frei und steht voll für eigene Software zur Verfügung.

Eine DVI- beziehungsweise HDMI-Buchse (**Bild 26**) plus acht Widerstände reichen aus, um den Raspberry Pi Pico elektrisch mit dem DVI- oder HDMI-Eingang eines Monitors zu verbinden. Die zugehörige Beschaltung (**Bild 27**) ist recht übersichtlich.

Luke Wren hat im GitHub-Repository [26] des Projekts PicoDVI beschrieben, wie der Interpolator im RP2040 für die Beschleunigung des TMDS-Encodings genutzt werden kann, was Rechenzeit spart. Der Interpolator war eigentlich zur Erzeugung von Pseudo-3D-Grafiken wie bei Super Mario Kart oder Pilotwings (**Bild 28**) gedacht. Der Interpolator des RP2040 ist aber flexibel genug, um damit die CPU-Kerne bei der Erzeugung der drei TMDS-Datenströme aus den RGB-Daten zu entlasten.

Wie kommen nun die Pixel aus dem RAM zum Monitor? Für die Ausgabe mit einer Auflösung von 640 x 480 Pixel wird ein Bild mit 320 x 240 Pixel hochskaliert. Daher müssen nur noch 240 Zeilen 60 mal pro Sekunde TMDS-encodiert werden. Jede Zeile wird

Bild 28. Pseudo-3D im Spiel Pilotwings.

Bild 29. Walker Demo für den Raspberry Pi Pico.

Bild 30. Tiles in der Walker Demo.

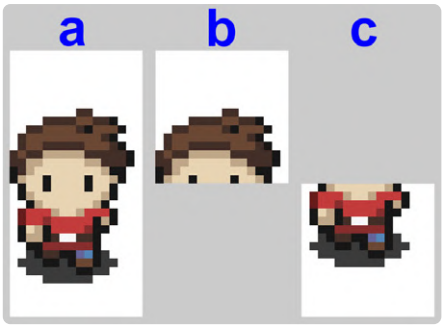


Bild 31. Spielfigur mit transparentem Hintergrund (a) sowie oberes (b) und unteres (c) Sprite der Spielfigur.

doppelt ausgegeben und passend dazu werden auch in der Vertikalen die Pixel doppelt so breit.

Um das zu erreichen, werden wie bei der Ausgabe eines VGA-Signals die Rohdaten zeilenweise bereitgestellt. Die Zeilen können aus einem Frame Buffer im Raspberry Pi Pico kommen oder aber „on-the-fly" bereitgestellt werden. Wie beim VGA- sind auch mit DVI-Ausgang interessante Projekte möglich. Dabei ist nur einer beiden CPU-Kerne samt seinem Interpolator mit der Erzeugung des DVI-Signals beschäftigt. Die Walker Demo (**Bild 29**) zeigt eindrucksvoll, dass ein Raspberry Pi Pico noch genug Reserven hat, um Pixel auf den Bildschirm zu zaubern.

Sprites, Kacheln, Tile-Maps beim RP2040

Sowohl für die VGA- als auch für die DVI-Ausgabe gibt es eine Grafikbibliothek, die Sprites erlaubt. Bei der Walker Demo wurde das fertige Bild aus vielen kleinen Elementen, den Kacheln (Tiles) zusammengesetzt (siehe **Bild 30**). Jede Kachel hat eine Größe von 16 x 16 Pixel. Diese Tiles sind als Hintergrundgrafik gedacht und haben in der Regel keine Transparenzinformationen – ganz im Gegensatz zur Spielerfigur (**Bild 31a**), die aus zwei Sprites (**Bilder 31b** und **31c**) zusammengesetzt ist. Sowohl der Hintergrund als auch die Sprites lassen sich individuell platzieren. Mit der passenden Auswahl lassen sich schöne Hintergründe wie in **Bild 32** zusammensetzen. Die Basis für die Grafikausgabe sind hier Tile-Maps (**Bild 33**) aus denen die 16 x 16 Pixel großen Teile herausgenommen und vom Raspberry Pi Pico dann passend zu einem Bild komponiert werden. Doch warum sollte man so etwas machen? Das Schöne an Sprites, Tiles und Tile-Maps ist der vergleichsweise geringe Speicherverbrauch und die Flexibilität ihres Einsatzes.

Neben der Tile-Map muss dann nur im Speicher stehen, an welchen Stellen die Tiles oder Sprites gezeichnet werden sollen.

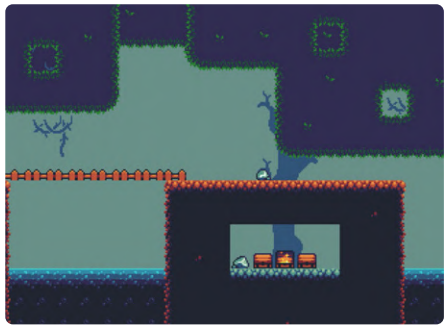


Bild 32. Demo einer 2D-Plattform.

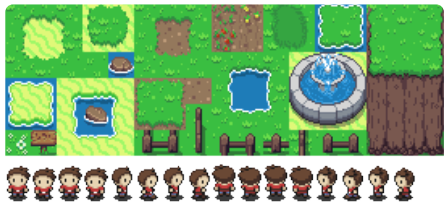


Bild 33. Tile-Maps (Quelle: ArMM1998 / https://tinyurl.com/5n976tcc).

Auch Animationen oder Scrolling sind so durch vergleichsweise einfache Updates von Sprite-Positionen möglich.
Für den Quelltext der Demos sei auf das entsprechende GitHub-Repository [28] von Luke Wren verwiesen. Die hier gezeigten Bilder stammen aus den Apps *tiles_and_sprites*, *tiles_parallax* und *tiles*. Wer einen passenden Raspberry Pi Pico mit DVI-Ausgang hat, kann die Apps selbst ausprobieren und mit dem Quelltext herumspielen.

Mikrocontroller als Zeichenkünstler?

Bislang wurden lediglich Demos aufgeführt und keine fertigen Spiele oder Anwendungen. Die Demos bieten aber genug Informationen, um eigene Projekte mit Grafik auszustatten. Sowohl ESP32 als auch der Raspberry Pi Pico sind preiswerte Boards, die eine erstaunlich gute Grafikausgabe erlauben. Ob Pac-Man oder Tetris per DVI, VGA oder Composite-Video – mit jeder Generation steigern sich die Fähigkeiten und die rohe Leistung von MCUs. Ihre Hardware erlaubt tolle Effekte wie 3D-Rendering oder bewegte Farbbilder mit Sprites. Wie damals bei den Heimcomputern der 80er Jahre können mit viel Kreativität und ein paar Tricks interessante Bilder aus dieser beschränkten Hardware herausgeholt werden. Die Mikrocontroller müssen beim Zeichnen nur passend orchestriert werden, denn alle Zutaten sind in diesem Artikel beschrieben. Es liegt nun an Ihnen, damit spannende Projekte zu realisieren. Stellen Sie Ihre Kreationen auf Elektor Labs [29] ein, dann können Sie technische Probleme und Features mit anderen Elektor-Mitgliedern diskutieren und hilfreiches Feedback erhalten.

220614-02

Sie haben Fragen oder Kommentare?

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

Passende Produkte

- **Pimoroni Raspberry Pi Pico VGA Entwicklungsboard (SKU 19769)**
  www.elektor.de/19769
- **Raspberry Pi Pico (SKU 19562)**
  www.elektor.de/19562
- **DVI-Aufsetzplatine für den Raspberry Pi Pico (SKU 19925)**
  www.elektor.de/19925
- **ESP32-PICO-Kit V4 (SKU 18423)**
  www.elektor.de/18423

WEBLINKS

[1] Bibliothek VGAX: https://github.com/smaffer/vgax
[2] Bibliothek VGAXUA: https://github.com/smaffer/vgaxua
[3] VGA-Pacman: https://niklas-rother.de/artikel/vga-pacman
[4] VGA mit ATmega und externem RAM: http://tinyvga.com/avr-sdram-vga
[5] VGA-Timing: http://tinyvga.com/vga-timing/640x480@60Hz
[6] ESP32 VGA: https://bitluni.net/esp32-vga
[7] ESP32 I²S-Controller Reference Manual: https://tinyurl.com/2ekfhshf
[8] VGA-Pixeltakt: http://tinyvga.com/vga-timing
[9] ESP32Lib von bitluni: https://github.com/bitluni/ESP32Lib
[10] Wikipedia: Sprites: https://de.wikipedia.org/wiki/Sprite_(Computergrafik)
[11] Bibliothek FabGL: http://www.fabglib.org
[12] Video-Tutorials für FabGL: https://www.youtube.com/user/fdivitto/videos
[13] RP2040-VGA als KiCad Projekt: https://tinyurl.com/ytcb3ysd
[14] Hardware-Design mit RP2040: https://tinyurl.com/23uj7sye
[15] Pimoroni Pico VGA Demo Base: https://tinyurl.com/5663v4cx
[16] GitHub: pico-playground: https://github.com/raspberrypi/pico-playground
[17] Bibliothek pico_scanvideo: https://tinyurl.com/yr8zk6pp
[18] Bibliothek PicoVGA: https://github.com/Panda381/PicoVGA
[19] Mag Pi: VGA-Grafikbibliothek für den Raspberry Pi Pico: https://tinyurl.com/38hjcmd9
[20] Wikipedia: SNES Mode 7: https://de.wikipedia.org/wiki/Mode_7
[21] Interpolator-Demo: https://tinyurl.com/26waz54x
[22] Wikipedia: DVI: https://tinyurl.com/r5wpkwnp
[23] Spezifikationen DVI: https://tinyurl.com/5wat6f9t
[24] Wikipedia: TMDS: https://tinyurl.com/2p8y8kyr
[25] Wikipedia: 8b/10b-Code: https://de.wikipedia.org/wiki/8b10b-Code
[26] GitHub: PicoDVI: https://github.com/Wren6991/PicoDVI
[27] Mathias Claußen interviewt Luke Wren, „DVI mit dem RP2040," ElektorMag 3-4/2023: https://www.elektormagazine.de/220575-02
[28] GitHub: Walkerdemos mit dem Rasperry Pi Pico: https://tinyurl.com/h2b4rf9f
[29] Elektor Labs: https://www.elektormagazine.de/labs

Das Echtzeit-Betriebssystem Metronom

Ein RTOS für AVR-Prozessoren

Von Dieter Profos, Dr. sc. techn., ETH Zürich (Schweiz)

Für vielerlei Aufgaben - wie zum Beispiel die Verarbeitung kontinuierlicher Signale - müssen Mikrocontroller Aufgaben in exakten Zeitabständen erledigen. Das hier vorgestellte Echtzeitbetriebssystem eignet sich (auch) für AVR-Controller mit wenig Speicher. Dafür muss man mit Einschränkungen wie einer reinen Assemblerprogrammierung leben, was bei Projekten, bei denen es hauptsächlich auf Geschwindigkeit und Echtzeitfähigkeit ankommt, aber immer noch einen guten Kompromiss darstellt.

Warum noch ein Betriebssystem mehr?

Mit dem Erscheinen von kleinen und kleinsten Prozessoren oder Controllern wurden Abläufe automatisierbar, die früher den Einsatz eines „richtigen" Computers niemals gerechtfertigt hätten. Bei diesen Kleinstcomputern sind keinerlei Peripheriegeräte (Tastatur, Maus, Bildschirm, Disk oder Ähnliches) zu steuern, die Betriebssysteme können damit auf das Allernotwendigste reduziert werden, nämlich das Organisieren der Verarbeitung von Nutzer-Programmen.

Die meisten Betriebssysteme sind darauf getrimmt, möglichst viele Programme möglichst effizient und (von außen gesehen) gleichzeitig auszuführen. Anders sieht die Sache jedoch aus, wenn mit den Prozessoren kontinuierliche, zeitgebundene Signale verarbeitet werden sollen: Hier braucht es Prozesse, welche in exakten Intervallen ablaufen. Die Funktion `delay()` in Arduino zum Beispiel ist dann nicht mehr genügend genau, da sie nur Wartezeiten erzeugt, nicht aber die für die Verarbeitung benötigten Laufzeiten mitberücksichtigt. Und diese fallen bei Abtastzeiten von 1 ms oder noch kürzer deutlich ins Gewicht. Es gilt also folgende zwei Probleme zu lösen:

- Gewisse Tasks sollen exakt zu vorgegebenen Zeiten ablaufen, andere dagegen nur, wenn gerade übrig bleibende Zeit dafür eingesetzt werden kann.
- Jeder unterbrechbare Task benötigt einen eigenen Stack zum Zwischenspeichern der Registerinhalte. Bei kleinen Controllern ist jedoch der Speicherplatz recht beschränkt: zum Beispiel 750 Bytes beim ATtiny25 oder 1 k beim ATmega8.

Das hier vorgestellte Betriebssystem *Metronom* kann als Open-Source-Software unter der BSD-2-Lizenz von der Elektor-Website heruntergeladen werden [1]; vorgesehen ist in den kommenden Monaten auch eine Veröffentlichung via GitHub.

Zyklische Tasks

Metronom ist genau für die Aufgabenstellung konzipiert, sogenannte *zyklische Tasks* zu exakt vorgegebenen zeitlichen Abständen auszuführen (mit bis zu acht unterschiedlichen Zykluszeiten). Pro Zykluszeit gibt es genau einen Task; sind mehrere, inhaltlich voneinander unabhängige Aktivitäten im gleichen Zyklus auszuführen, so sind diese im selben Task zusammenzufassen.

Die Zykluszeiten werden folgendermaßen erzeugt:

- Die Basis-Zykluszeit (beispielsweise 1 ms) wird durch die Hardwaremittel des Prozessors (Quarz- oder interner RC-Oszillator, hardware- und interruptgesteuerter Softwarezähler) hergestellt, und
- die weiteren Zykluszeiten werden durch eine Kette von Zählern

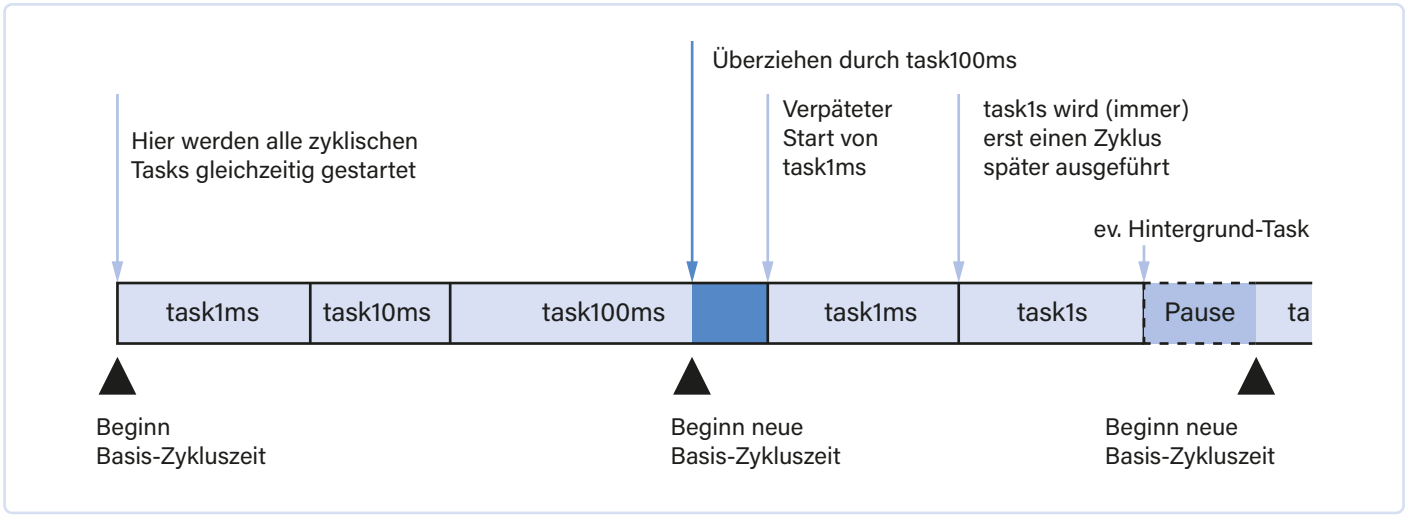


Bild 1. Ablauf mehrerer zyklischer Tasks (Extremfall).

erzeugt, sodass jede Zykluszeit ein Vielfaches der vorhergehenden ist (die Standard-Einstellung ist beispielsweise 1 ms 10 ms 100 ms 1 s).

Eine wichtige Eigenschaft des hier vorgestellten Betriebssystems ist, dass zyklische Tasks sich gegenseitig nicht unterbrechen können (*non preemptive*). Damit laufen einerseits diese Tasks möglichst zeitgenau ab, anderseits verliert der Prozessor keine „unproduktiven Zeiten" für Taskwechsel. Und da jeder zyklische Task – einmal gestartet – ohne Unterbrechung (ausgenommen durch Interrupts) zu Ende läuft, bevor der nächste zyklische Task gestartet wird, können alle zyklischen Tasks gemeinsam denselben Stack benützen.
Was geschieht jedoch, wenn ein Task länger als die Basis-Zykluszeit läuft (was eigentlich durch den Programmierer vermieden werden sollte) oder aber, wenn im selben Basiszyklus mehrere zyklische Tasks gestartet wurden, wobei die Summe der Laufzeiten die Basis-Zykluszeit übersteigt (was durchaus legitim ist)? Hier tritt eine weitere Eigenschaft von Metronom in Kraft: Die zyklischen Tasks besitzen Prioritäten: Der Task mit der kürzesten Zykluszeit hat höchste Priorität, der „zweitschnellste" Task die nächstniedrigere und so weiter. Können nicht alle gleichzeitig gestarteten zyklischen Tasks innerhalb der Basis-Zykluszeit abgearbeitet werden, so wird der gerade laufende Task nach Ablauf der Basis-Zykluszeit bis zu seinem Ende fortgesetzt – danach aber wird zuerst der höchstpriorisierte „schnellste" Task wieder ausgeführt und erst danach weitere, noch von vorher gestartete zyklische Tasks.
Ein Beispiel: Die Implementation der Zykluszeiten führt dazu, dass jede Sekunde einmal alle zyklischen Tasks gleichzeitig gestartet werden. Was dann geschieht, zeigt **Bild 1**.
Daraus ist abzuleiten, dass als absolut oberste Grenze jeder zyklische Task nicht länger als die kürzeste Zykluszeit dauern darf - also in unserem Beispiel 1 ms. Das entspricht bei einem ATtiny mit 8 MHz ungefähr 6000, bei einem ATmega mit 16 MHz ungefähr 14000 Instruktionen (der Rest der Instruktionen wird im Mittel durch das Betriebssystem selbst sowie die Behandlung von Interrupts gebraucht).

Hintergrund-Tasks

Nun gibt es jedoch gewisse Operationen, welche von Natur aus länger dauern:

- Ein Schreibzugriff auf das EEPROM dauert beispielsweise einige Millisekunden (typisch ungefähr 3,3 ms), also untragbar lange bei einem Basiszyklus von 1 ms.
- Das Übertragen eines Texts mit 9600 Bd ist mit einem Basiszyklus von 1 ms nicht machbar, da bereits die Übertragungszeit eines einzelnen Zeichens länger als 1 ms dauert.
- Sollen längere Berechnungen (zum Beispiel mit emulierten Arithmetik-Operationen!) durchgeführt oder Zeichenketten verarbeitet werden, so dauern diese innerhalb eines zyklischen Tasks oft zu lange und behindern damit die zeitgebundenen Abläufe.

Die Folgerung heißt demnach, dass trotzdem eine Möglichkeit erforderlich ist, solche Aufgaben an irgendeine unterbrechbare Art von Tasks zu delegieren.Dafür werden zwei Methoden in Kombination verwendet:

- *Verwendung von Interrupts statt aktives Warten:* Damit kann man das Warten auf das Ende eines Vorgangs (zum Beispiel das Übertragen eines Zeichens) an die Hardware „delegieren"; dieser Weg wird für interruptgesteuerte Vorgänge benutzt. Dies löst die Probleme für eine einzelne Zeichen-Übertragung oder für das Schreiben eines einzelnen Werts ins EEPROM, nicht aber das Warten auf das Ende der Gesamt-Operation, etwa die Übertragung eines ganzen Texts.
- *Einrichten von Hintergrund-Tasks:* Ein Hintergrund-Task läuft nur in denjenigen Zeiten, welche nicht durch zyklische Tasks beansprucht werden. Zudem ist er gegenüber den zyklischen Tasks jederzeit unterbrechbar, behindert also deren zeitgerechtes Ablaufen nicht.

Durch andere Hintergrund-Tasks jedoch kann ein einmal laufender Hintergrund-Task nicht unterbrochen werden. Es wird somit immer nur ein Hintergrund-Task abgearbeitet, und wenn dieser warten muss, wartet die ganze Verarbeitung von Hintergrund-Tasks. Die Abarbeitung der Hintergrund-Tasks wird damit zwar langsamer; dafür wird hiermit ermöglicht, dass für die Gesamtheit aller Hintergrund-Tasks ebenfalls nur ein einziger Stack-Bereich freigehalten werden muss.
Hintergrund-Tasks zeichnen sich durch folgende Eigenschaften aus:

- ein Hintergrund-Task ist jederzeit unterbrechbar zugunsten von zyklischen Tasks, nicht aber zugunsten eines anderen Hintergrund-Tasks.
- Das Ablaufen eines Hintergrund-Tasks wird durch einen Start-Aufruf an den Dispatcher ausgelöst.
- Hintergrund-Tasks werden in der Reihenfolge ihres Starts nacheinander ausgeführt.

- Hintergrund-Tasks können auf Ereignisse (`WAIT_EVENT`) warten, welche zum Beispiel durch interruptgesteuerte Abläufe ausgelöst werden.
- Hintergrund-Tasks können auch vorgebbare Zeiten abwarten (`DELAY`).
- Jedem Hintergrund-Task kann eine Start-Nachricht von drei 16-bit-Wörtern mitgegeben werden, mit welchen sein spezifischer Auftrag definiert werden kann (ein viertes Wort ist für die Startadresse des Task reserviert).
- Innerhalb des Benutzerprogramms kann eine beliebige Anzahl von Hintergrund-Task-Programmen vorhanden sein; es können jedoch maximal acht gleichzeitig gestartet sein.

Die Koordination der Tasks untereinander („wann darf welcher Task ablaufen") ist Aufgabe des sogenannten *Dispatchers*. Dieser führt alle „Verwaltungs-Abläufe" aus, vom Start von Tasks über die Sicherung/Wiederherstellung der Prozessorregister bis hin zur Sperrung/Freigabe von Interrupts.

Exceptions

Da Kleinstprozessoren meist keine textorientierten Peripheriegeräte besitzen, gestaltet sich die Fehlersuche insbesonders bei zeitabhängigen Funktionen sehr schwierig, da Haltepunkte oder ähnliches das Zeitverhalten völlig außer Takt bringen. Deshalb stellt der Betriebssystem-Kern einen vereinfachten Mechanismus zur Exception-Behandlung zur Verfügung, welcher in zwei Stufen aufgeteilt ist:

- Ein globaler `try-catch`-Bereich fängt alle im Kernel und in den Arithmetik-Emulationen auftretenden *Exceptions* (Ausnahmen/Fehler) ab. Die exception-spezifischen Daten können im EEPROM gespeichert und/oder via USART ausgegeben werden; danach führt das Betriebssystem einen Gesamtsystem-RESET (einschliesslich `user-reset`) aus. Dieser Exception-Bereich ist immer aktiv.
- Zusätzlich kann ein anwendungsorientierter `try-catch`-Bereich benutzt werden, der nur das eigentliche Anwender-Programm abdeckt. Die Behandlung solcher Exceptions geschieht anfänglich gleich wie oben: Die Exception-Daten werden gespeichert und/oder via USART ausgegeben; danach wird ein durch den Anwender zu definierender „Anwendungs-Restart" durchgeführt (Subroutine `user_restart`).

Interruptbehandlung

Interrupts werden auf vier unterschiedliche Arten behandelt:

- Der Reset-Interrupt wird durch das Betriebssystem genutzt und ist dem Benutzer nicht direkt zugänglich. Da jedoch der User diesen Interrupt auch zur Initialisierung der eigenen Abläufe benötigt, ruft das Betriebssystem nach der eigenen Initialisierung die Subroutine `user_init` auf, welche der Anwender mit seinem anwendungsspezifischen Initialisierungs-Code füllen kann.
- Der Timer/Counter0 wird für die Erzeugung des Basistakts für alle zyklischen Prozesse benutzt; er ist dem Anwender deshalb nicht zugänglich.
- Für den Einsatz des EEPROMs und des USARTs bietet das Betriebssystem fertige Treiber-Bausteine an, welche anlässlich der Betriebssystem-Generierung eingebunden werden können (siehe unten). Der Anwender kann aber auch eigene Service-Routinen an diese Interrupts koppeln oder sie bei Nichtgebrauch einfach offen lassen.
- Alle übrigen Interrupts stehen dem User direkt offen. Pro Interrupt ist jeweils eine sogenannte Interrupt-Service-Routine sowie eine Interrupt-Initialisierungs-Routine zu definieren. Gehören zu einem Device mehr als nur ein Interrupt (wie zum Beispiel bei Timern oder USART), so genügt eine gemeinsame Initialisierungs-Routine. Der Anwender aktiviert hierzu die entsprechenden Parameter in der Generierungs-Datei und fügt die Inhalte der betreffenden Initialisierungs- und Serviceroutinen in seinem Anwenderprogramm ein.
- Nicht genutzte Interrupts werden durch das Betriebssystem automatisch „abgefangen".

Programmierumgebung

Metronom ist aus Effizienzgründen in AVR-Assembler (Atmel/Microchip) geschrieben und setzt somit voraus, dass die Benutzerprogramme ebenfalls in Assembler geschrieben sind; eine Schnittstelle zu C wurde nicht implementiert. Stattdessen gibt es jedoch eine Library mit vielen Subroutinen für beispielsweise 8-Bit-Arithmetik sowie 16-Bit-Arithmetik (vier Grundrechenarten); eine 16-Bit-Fractional-Bibliothek ist in Vorbereitung.
Zur Erleichterung der Programmierarbeit liegen alle Betriebssystem-Aufrufe als Makros vor. Zur Vermeidung von Namenskollisionen gilt folgende Namenskonvention: Alle Variablen und Sprungziele innerhalb des Betriebssystems und der Bibliotheken beginnen mit Unterstrich („_") – daher sollten alle Namen im Benutzerprogramm ausschließlich mit Buchstaben beginnen. Andere Zeichen als Buchstaben und Zahlen und der Unterstrich sind nicht zulässig.
Die Gesamtstruktur von Metronom und dem jeweiligen Benutzerprogramm sieht man in **Bild 2**.

Betriebssystem-Aufrufe

Für die vollständige Liste der Betriebssystem-Aufrufe wird auf die Referenzen am Schluss des Artikels verwiesen; hier nur eine grobe Übersicht:

Makros für Exception-Handling:

- `KKTHROW` wirft eine systemweite Exception, was bedeutet, dass nach Speichern/Ausgeben der Exception-Information das gesamte System neu gestartet wird.
- `KTHROW` wirft eine auf das Benutzerprogramm begrenzte Exception, was bedeutet, dass nach Speichern/Ausgeben der Exception-Information nur die Benutzer-Subroutine `user_restart` ausgeführt wird; danach werden die zyklischen Tasks neu gestartet.

Makros für das Benutzen von Hintergrund-Tasks:

- `_KSTART_BTASK` startet einen Hintergrund-Task.
- `_KDELAY` versetzt den aufrufenden Hintergrund-Task für n (0 bis 65535) ms in den Schlafzustand.
- `_KWAIT` versetzt den aufrufenden Hintergrund-Task in einen Schlafzustand, aus dem er mit
- `_KCONTINUE` wieder geweckt wird.

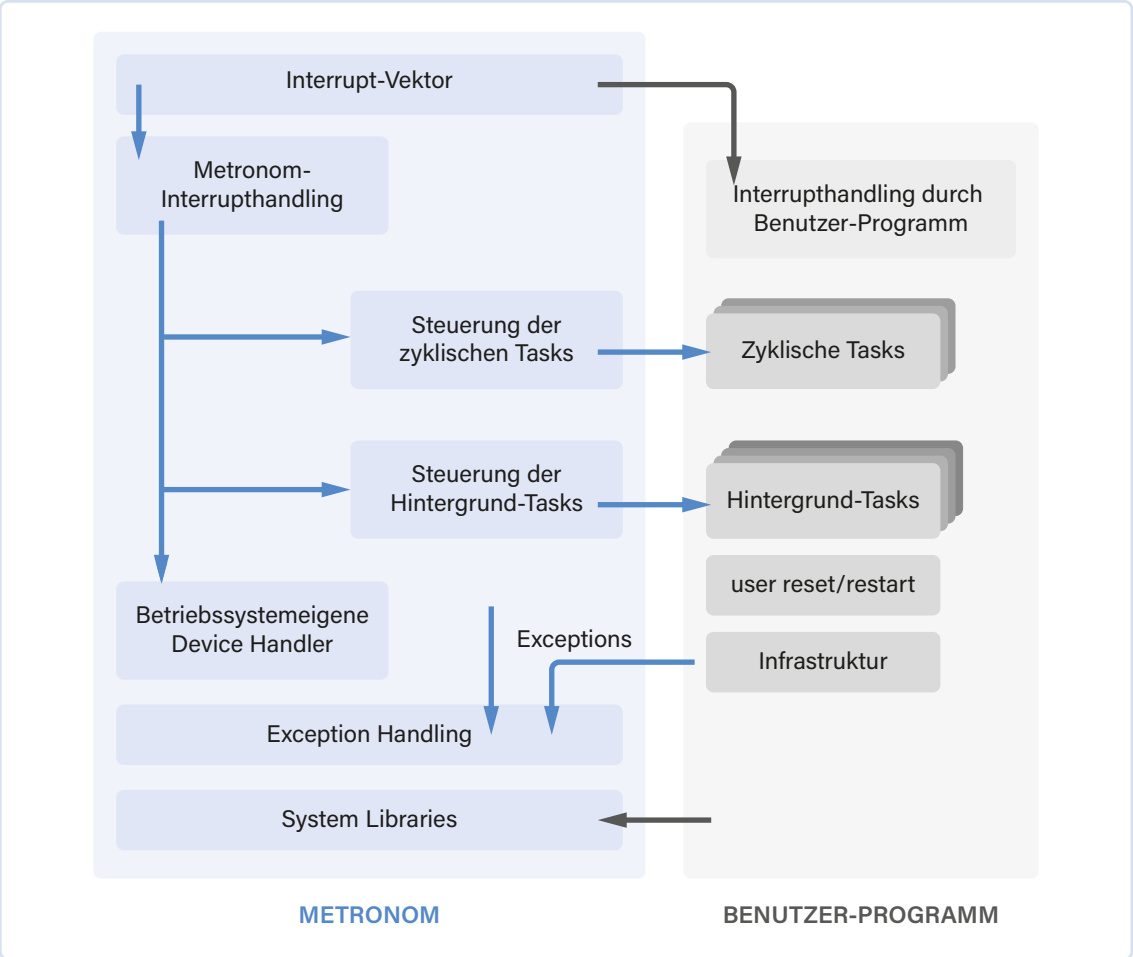


Bild 2. Gesamtstruktur von Metronom und dem Benutzer-Programm.

Makros für 8- und 16-Bit-Arithmetik

Generell werden für Arithmetik-Operationen aller Art die Register r25:r24 als Akkumulator und r23:r22 als Speicher für den Zweit-Operanden (falls benötigt) verwendet.

Hierfür gibt es über 20 unterschiedliche Funktionen wie `_mul8u` für eine 8x8bit-Multiplikation oder `_abs16` für einen 16Bit-Absolutwert. Ferner gibt es viele Lade- und Speicher-Pseudocodes wie etwa `_ld16` (Lade 16-Bit-Zahl in den Akkumulator).

Makros für die EEPROM-Nutzung

- `_KWRITE_TO_EEPROM` zum Schreiben ins EEPROM,
- `_KREAD_FROM_EEPROM` zum Lesen vom EEPROM.

Makros für USART-Nutzung

- `_KWRITE_TO_LCD` ist ein spezifischer USART-Treiber, welcher dem darzustellenden Text gleich auch die notwendigen Steuerzeichen für ein 2x16 LCD-Display beifügt
- `_KREAD_FROM_USART` (bisher nicht implementiert).

Systemgenerator SysGen

Für das Generieren eines Systems (also des vollständigen Codes) wird der eigene Systemgenerator SysGen verwendet, der ebenfalls Bestandteil des Gesamtpakets ist. SysGen ist nicht auf Metronom beschränkt, sondern kann auch für allgemeine Generieraufgaben verwendet werden.

Manche Leser mögen sich fragen, warum ein eigener System-Generator entwickelt wurde, nachdem es ja eine Vielzahl von Preprozessoren und Makrogeneratoren gibt. Doch für das Generieren des Betriebssystems Metronom genügen die Funktionsumfänge der Preprozessoren im Atmel-Studio sowie von Standard-C nicht. Insbesondere da der Preprozessor keine String-„Arithmetik" erlaubt, kann man kein „Standard-Directory" oder „Library-Directory" vorgeben und von diesem aus darin enthaltene Dateien anwählen. Eine Recherche bei *stackoverflow* hat gezeigt, dass andere Leute dasselbe Problem haben wie ich, aber keiner der heutigen Preprozessoren damit umgehen kann.

Der Preprozessor von Atmel-Studio (und auch der GNU-Preprozessor) bieten folgende Funktionen für das Zusammensetzen der beteiligten Dateien:

- `define / set <name> = <numerischer_ausdruck>`
- `if … elif … else … endif`, auch geschachtelt
- `ifdef, ifndef`
- `include <pfad> | exit`

Es fehlen folgende Funktionalitäten:

- `<pfad>` kann nur als fixer String vorgegeben werden, nötig wäre aber ein String-Ausdruck von (beliebig vielen) Teil-Strings, sowohl Stringvariablen als auch String-Konstanten.
- `define` und `set` können nur numerische Werte zuweisen, keine Strings, kein Aneinanderhängen von Strings, und auch keine logischen Ausdrücke.
- Für das (einmalige) Einbinden von Library-Programmen genügen die in AVRASM oder C-Preprozessor gegebenen Makro-Möglichkeiten nicht; da Makros in AVRASM keine Include-Anweisungen enthalten dürfen, ist zum Beispiel das automatische Einbinden von Emulations-Routinen nicht möglich.

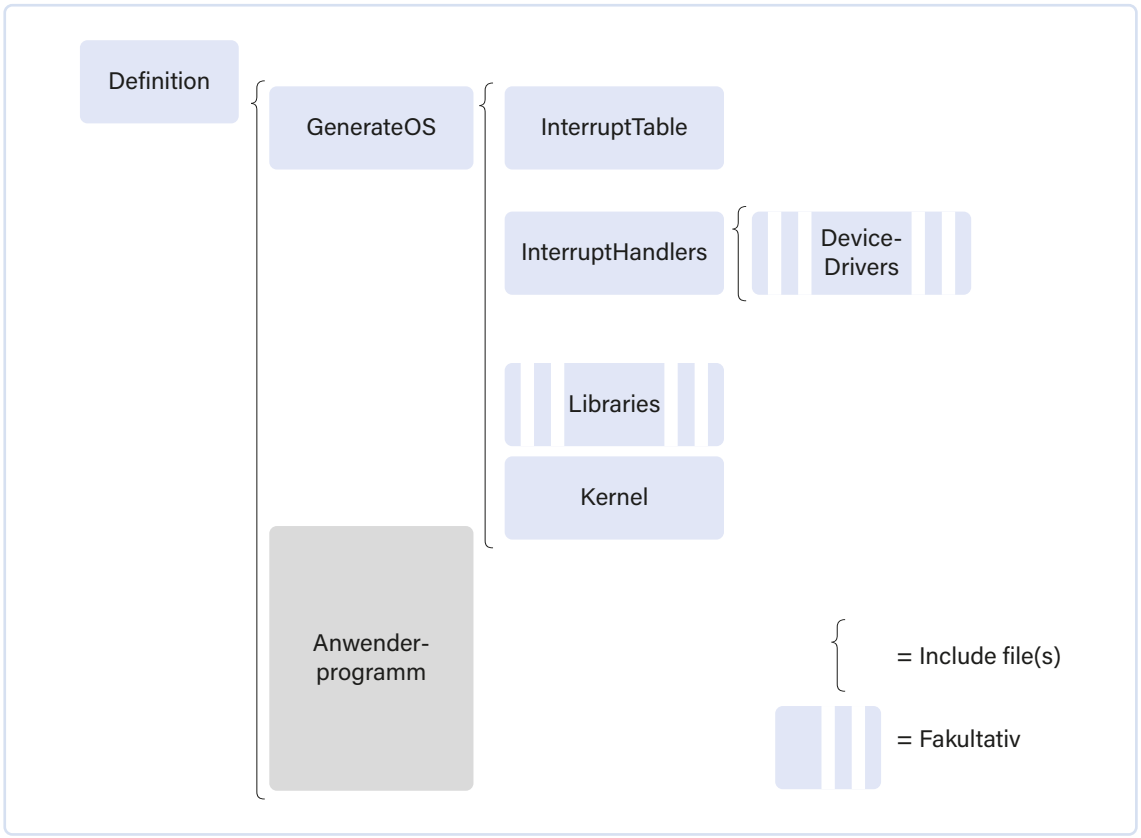


Bild 3. Generierstruktur von Metronom-Systemen.

Das führt zu folgendem Funktionsumfang:

- `define / set <name> = <numerischer_ausdruck> | <string ausdruck> | <boole'scher ausdruck>`
- `if … elif … else … endif`, auch geschachtelt
- `ifdef, ifndef` wird umgewandelt in `if isdef(..)` bzw. `! isdef(..)` und ist damit auch innerhalb von Boole-Ausdrücken verwendbar
- `include <string ausdruck> | exit`
- `message | error`
- `code` (zum Erzeugen von Code-Zeilen mit generierbarem Inhalt)
- `macro/endmacro` mit passender Parameter-Kennzeichnung
- Eine zusätzliche Bedingung ist, dass Anweisungen bestehender Preprozessoren mit denjenigen von SysGen mischbar sind, ohne dass sie sich gegenseitig stören.

Das Programm SysGen kann ebenfalls von der angegebenen Website [1] heruntergeladen werden. SysGen ist in Java (Version 12) geschrieben und benötigt für seinen Ablauf eine entsprechende Java-Installation.

Programmieren mit Metronom

Um dem Anwender das mühsame Durcharbeiten des Betriebssystem-Quellcodes zu ersparen, ist das gesamte Betriebssystem in generierbarer Form aufgebaut. Das bedeutet, dass der Anwender nur die Definitions-Datei sowie – falls erforderlich – die selber programmierten Interrupt-Routinen ausfüllen muss. Wo diese hingehören und wie sie verbunden werden, wird durch den Generiervorgang automatisch erledigt.
In seiner Grundform setzt sich ein Anwendersystem aus den Teilen wie in **Bild 3** zusammen.
Die Interrupt-Tabelle und der Kernel werden stets als Ganzes in das entstehende Gesamt-Programm eingebaut; von den Device-Handlern und den Bibliotheken dagegen werden nur die wirklich benötigten Teile übernommen.
Um den Ablauf einer Generierung zu veranschaulichen, wird hier das Generier-Skript eines meiner eigenen Projekte in **Listing 1** wiedergegeben.

210719-02

Sie haben Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Schicken Sie eine E-Mail an den Autor unter profos@rspd.ch oder kontaktieren Sie Elektor unter redaktion@elektor.de.

Passende Produkte

- **Allan He, Lingyuan He, *Embedded Operating System***
  Buch (Paperback), SKU 19228: www.elektor.de/19228
  E-Buch (PDF), SKU 19214: www.elektor.de/19214

WEBLINKS

[1] Generierdateien für Metronom, Generierprogramm SysGen, Dokumentation: https://www.elektormagazine.de/labs/metronom-a-real-time-operating-system-for-avr-processors

Listing 1.

```
; ********************************************************
; Master Definition
; ********************************************************
; Stand: 03.05.2022
;
; This file contains all informations required to generate your user system for
; AVR processors.
; It consists of three parts:
;
; 1. Definitions
;    A bunch of variable definitions defining which functionalities to include.
;    This part must be edited by the user.
;
; 2. An $include statement for the actual generation of the operating system.
;    DO NOT MODIFY THIS STATEMENT!
;
; 3. The $include statement(s) adding the user program(s).
;    This part must be edited by the user.
;

; ********************************************************
; PART 1: DEFINITIONS
;
; This script is valid for ATmega8, ATmega328/P and ATtiny25/45/85 processors.
; If you want to use it for any other processors feel free to adapt it accordingly.

$define processor = "ATmega8"

; Remove the ; in front of the $set directive if you want to use the EEPROM
; $set _GEEPROM=1
; if you want to write your own routines to write to the EEPROM use the following
; definition:
; $set _GEEPROM=2
; Enabling this definition will insert an appropriate JMP instruction to your
; interrupt service routine e_rdy_isr in the InterruptHandlers.asm file
; Remove the ; in front of the $set directive if
; ... you want to output serial data via the USART, or
; ... you want exception messages to be sent outside via the USART
; $set _GUSART=1
; if you want to write your own routines to use the USART
; use the following definition instead
; $set _GUSART=2
; Enabling this definition will enable the interrupt service routines usart_udre_isr,
; usart_rxc_isr and usart_txc_isr in the InterruptHandlers.asm file.

; ------------------------------------------------------------
; Define the division ratios of the time intervals for cyclic tasks
; The definition shown here is the standard preset for 1 : 10 : 100 : 1000 ms
; The first ratio being 0 ends the divider chain.
.equ _KRATIO1 = 10◦◦◦◦◦◦◦◦◦◦◦ ; 1 -> 10ms
.equ _KRATIO2 = 10◦◦◦◦◦◦◦◦◦◦◦ ; 10 -> 100ms
.equ _KRATIO3 = 10◦◦◦◦◦◦◦◦◦◦◦ ; 100ms -> 1s
.equ _KRATIO4 = 0◦◦◦◦◦◦◦◦◦◦◦◦ ; end of divider chain
.equ _KRATIO5 = 0
.equ _KRATIO6 = 0
.equ _KRATIO7 = 0
; NOTE: Do not remove "superfluous" .EQU statements but set them to 0 if not used!

; ------------------------------------------------------------
; Define the constants used for generation of the 1ms timer interrupt
; IMPORTANT: The following definitions depend on the processor being used
; and the frequency of the master clock
```

continued overleaf

```
$if (processor == "ATmega8")
; The definitions below are based on a system frequency of 12.288 MHz (crystal)
; This frequency has been chosen in order to use the crystal also for USART@9600 Bd
;
; set prescaler for counter0 to divide by 256, yields 48kHz counting freq for Counter0
.equ _KTCCR0B_SETUP = 4
; Counter0 should divide by 48 in order to produce interrupts every 1ms;
; since counter0 produces an interrupt only at overflow we must preset
; with (256-48) - 1 = 207.
$code ".equ _KTCNT0_SETUP = " + (256 - 48) - 1

$elif ... similar for other processors
;
$endif
;
; ------------------------------------------------------------
; Define the characteristics of USART transmission
; (if you don't use the USART just neglect these definitions):
$set fOSC = 12288000
$set baud_rate = 9600
$code ".equ _KUBRR_SETUP = " + (fOSC / (16 *baudrate) - 1)

; parity: 0 = Disabled,
; (1 = Reserved), 2 = Enable Even, 3 = Enable Odd
.equ _KPARITY = 0

; stop bits: 0 = 1 stop bit, 1 = 2 stop bits
.equ _KSTOP_BITS = 1

; data bits transferred: 0 = 5-bits, 1 = 6-bits, 2 = 7-bits, 3 = 8-bits, 7 = 9-bits
.equ _KDATA_BITS = 3
;
; ------------------------------------------------------------
; Connect a user defined interrupt handler (except RESET and Timer0)
; by removing the ; in front of the appropriate $set directive;
; don't change any names but just let the $set statement as is

; Interrupts for ATmega8
; $set _kext_int0 = 1    ; IRQ0 handler
$set _kext_int1 = 1      ; IRQ1 handler/initializer is supplied by user
; $set _ktim2_cmp = 1    ; Timer 2 Compare Handler
; $set _ktim2_ovf = 1    ; Timer 2 Overflow Handler
; $set _ktim1_capt = 1   ; Timer 1 Capture Handler
;
; etc. etc. etc.

;
; *********************************************************
; PART 2: GENERATING THE OPERATING SYSTEM
;
.LISTMAC
;
$include lib_path + "\GenerateOS.asm"
;
;
; *********************************************************
; PART 3: ADD THE USER PROGRAM
;
$include user_path + "\MyApplication.asm"
;
$exit
```

DVI auf dem RP2040

Ein Interview mit Luke Wren, Chip-Entwickler bei Raspberry Pi

Von Mathias Claußen (Elektor)

Luke Wren, einer der Ingenieure des RP2040-Mikrocontrollers von Raspberry Pi, hat mit diesem kleinen Chip Erstaunliches geleistet und ihn an seine Grenzen gebracht. Elektor-Ingenieur Mathias Claussen wollte mehr wissen, vor allem über die Tricks und Hacks, die nötig sind, um die Videoausgabe aus einem so winzigen Controller herauszuholen.

Der Raspberry Pi RP2040 ist eine MCU für etwa 1 Euro mit erstaunlichen Fähigkeiten. Einer seiner Ingenieure ist Luke Wren, der nicht nur am Raspberry Pi RP2040 gearbeitet hat, sondern auch gezeigt hat, dass der kleine RP2040 DVI-Ausgabe beherrscht, wie dies der Artikel „Videoausgabe mit Mikrocontrollern (2)" [1] zeigt. Wenn er nicht gerade an geheimen Projekten für Raspberry Pi arbeitet, teilt er einige seiner Freizeitprojekte mit der Community auf Twitter (@wren6991) und den Code auf GitHub [2].

Mathias Claussen: Kannst du uns ein wenig über dich erzählen?
Luke Wren: Fangen wir mit der schwierigsten Frage an! Ich bin Ingenieur bei Raspberry Pi, und wenn ich das nicht tue, arbeite ich normalerweise an meinen Hobbyprojekten, spiele schlecht Gitarre, oder, seit kurzem, lerne ich Sprachen. Ich lebe in Cambridge, Großbritannien, nicht weil es eine besonders aufregende Stadt ist, sondern eher aus Trägheit nach dem Abschluss meines Studiums. Ich habe in Deutschland gelebt, als ich jünger war, aber mein Deutsch ist inzwischen ziemlich eingerostet, deshalb bin ich froh, dass wir das hier auf Englisch machen.

Mathias: Wie lange arbeitest du schon mit dem Raspberry Pi?
Luke: Ich bin im September 2018 als Mitarbeiter eingestiegen, habe aber vorher schon ein Praktikum bei Raspberry Pi gemacht.

Mathias: Was war deine Rolle bei der Entwicklung des RP2040?
Luke: Ich habe an einem Teil des digitalen Designs gearbeitet - hauptsächlich PIO, DMA, XIP-Cache, Bus-Strukturen und PWM. Außerdem habe ich am Boot-ROM und am SDK gearbeitet, und natürlich musste ich auch bei der Dokumentation mit anpacken.

Mathias: Video von einer MCU/CPU zu bekommen ist etwas, was der Sinclair ZX81 konnte, aber DVI ist etwas Neues für eine MCU für einen Euro. Die VGA-Ausgabe wird von den meisten MCUs problemlos erledigt, aber was ist die Herausforderung bei DVI?
Luke: Es gibt zwei Dinge, die DVI-D schwieriger machen als VGA. Der erste ist die Serialisierung der Daten: Der minimale Pixeltakt für DVI-D beträgt 25 MHz, und der Bittakt ist zehnmal so hoch, so dass man mindestens drei differenzielle serielle Leitungen (rot/grün/blau) mit 250 Mbps ansteuern muss.
Zweitens sendet DVI-D nicht einfach nur rohe Pixeldaten, sondern kodiert sie zunächst. Die Kodierung ist in der Hardware einfach, aber in der Software etwas fummelig. Vor allem, wenn die Software mit der Rohgeschwindigkeit der seriellen Ausgabe mithalten muss.
Alles andere ist ähnlich. Es ist wirklich nur DPI durch eine schnellere Leitung. (*Anmerkung der Redaktion: Display Pixel Interface (DPI) ist eine parallele RGB-Pixelschnittstelle - einschließlich eines Pixeltakts - zur Übertragung von Pixeldaten an ein Display*).

Mathias: Was war deine Motivation, DVI am RP2040 auszuprobieren?
Luke: Nachdem der Stress der Fertigstellung des Siliziumchips vorbei war, wollten einige von uns sehen, wie hoch wir die Systemtaktfrequenz treiben könnten. In der Praxis gibt es einen gewissen Spielraum gegenüber der Nennfrequenz von 133 MHz. Ich hatte im Rahmen meines RISCBoy-Projekts [3] mit DVI-D auf einem FPGA gespielt, und als ich bemerkte, dass es eine Überschneidung zwischen den niedrigsten DVI-Bit-Taktfrequenzen und den höchsten Systemtaktfrequenzen auf dem RP2040 gab, ging ein Licht in meinem Kopf an. Die Motivation war: „Ich frage mich, ob das möglich ist."

Mathias: Was war die größte Herausforderung, um einen DVI-Ausgang (**Bild 1**) auf dem RP2040 zu realisieren?
Luke: Die TMDS-Kodierung. Wenn man den Algorithmus in der DVI-Spezifikation befolgt, gibt es keine Chance, ihn auf zwei Cortex-M0+-Kernen, die mit der Bittaktfrequenz laufen, schnell genug zu bekommen. Aber es gibt einige Tricks und Abkürzungen, um es doch möglich zu machen, und dann etwas sorgfältig handgeschriebenen Code, um es brauchbar schnell zu machen. Der RP2040 hat zwar viel RAM, aber nicht genug, um die Pixel eines TMDS-kodierten Bildes zu speichern, so dass man während der Kodierung ein „Wettlauf gegen den Elektronenstrahl" (Anmerkung: Im Original „Racing the Beam" ist auch ein Videospiel) veranstalten muss.

Bild 1. Raspberry Pi Pico mit dem Pico-DVI-Sock (Quelle: Luke Wren).

Mathias: Du musstest den RP2040 leicht übertakten (von normalerweise 133 MHz auf 252 MHz). Gibt es einen kritischen Signalpfad im Chip für die DVI-Signale (man muss die I/O-Pins mit Geschwindigkeiten ansteuern, die auch schneller sind als die Standardgeschwindigkeit)?
Luke: Die erste Einschränkung, auf die man beim RP2040 stößt, ist, dass der Systemtakt 1:1 mit dem Bittakt übereinstimmen muss. Wenn man also versucht, zu Modi mit höherer Auflösung zu wechseln, stürzen die Prozessoren einfach ab. Der kritische Setup-Pfad für den Bereich des Systemtakts des RP2040 sind die Adressphasen-Signale des Prozessors zu den SRAMs.
Davon abgesehen sind wir auch ziemlich nah an den Grenzen dessen, was man über diese gewöhnlichen 3V3-Pads treiben kann; wenn man sich das Augendiagramm (**Bild 2**) für 720p30 (372 Mbps) auf meinem GitHub-Account [2] ansieht. Es funktioniert, aber nur gerade so. Ich bezweifle, dass man 1080p30 ohne spezielle Hardware sehen würde.

Mathias: Abgesehen von der Geschwindigkeitssteigerung, wie wichtig sind die PIOs und der Interpolator im RP2040, um DVI zum Laufen zu bringen?
Luke: Es ist eine feste Anforderung, drei serielle Datenbits plus ihre differenziellen Komplemente auf GPIOs mit mindestens 250 Mbps darstellen zu müssen. Um eine Konvertierung von Single-Ended zu Pseudo-Differential in der PIO durchzuführen, halbiert sich die DMA-Bandbreite, und die Aufteilung der TMDS-Verbindungen in drei FIFOs ist nützlich, wenn man die Kodierung in der Software durchführt, da man so seinen Code für die Kodierung der Rot/Grün/Blau-Komponenten spezialisieren kann. So etwas wie PIO ist entscheidend, wenn man keine dedizierte Hardware hat.
Die Interpolatoren helfen, die nötige Leistung der Adressgenerierung für TMDS-Kodierung bereitzustellen, was sicherlich der Schlüssel zu einigen der Demos ist, die du gesehen hast, aber mein Trick mit der verdoppelten TMDS-Kodierung würde auch ohne die Interpolatoren auf einen einzelnen Cortex-M0+ Kern passen.

Mathias: Dein Pico DVI Sock für den RP2040 verwendet eine physische HDMI-Verbindung (**Bild 3**), die eindeutig als DVI-only gekennzeichnet ist, so dass wir keinen Audio erhalten. Ist das „nur" ein Lizenzproblem mit dem HDMI-Konsortium?
Luke: Es gibt nichts, was einen daran hindert, HDMI-Dateninseln hinzuzufügen und eine Audioausgabe zu realisieren. Tatsächlich hat das schon jemand mit einem NES-Emulator-Port gemacht [4][5]! Für die Audiosignale sind keine zusätzlichen physischen Verbindungen erforderlich, obwohl man streng genommen keine HDMI-Funktionen verwenden darf,

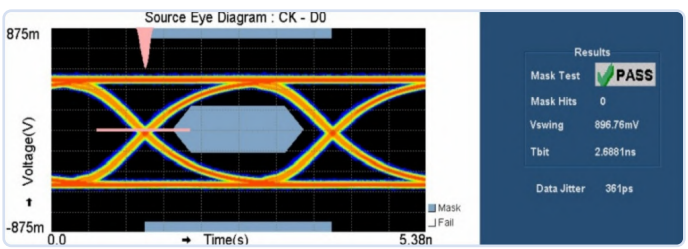


Bild 2. Augendiagramm für RP2040-DVI bei 720p30 (Quelle: Luke Wren).

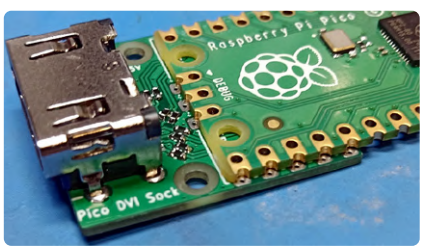


Bild 3. Pico-DVI-Sock für den Raspberry Pi Pico (Quelle: Luke Wren).

Bild 4. Hochauflösende Bilder mit einigen Tricks (Quelle: Luke Wren).

bevor man den Display-Datenkanal abgefragt hat, der bei meiner Pico DVI Sock nicht angeschlossen ist. Die HDMI-Lizenzierung ist sicherlich ein kompliziertes Problem, das ich nicht angehen möchte. Deshalb habe ich das Repository auch nur „PicoDVI" genannt, den Rest überlasse ich der Community.

Mathias: Wenn man den DVI-Ausgang verwendet, wie viele Ressourcen des RP2040 sind dann an diese Aufgabe gebunden? Bleibt da überhaupt noch Zeit, um anderen Code auf der MCU auszuführen?
Luke: Das kommt auf den Videomodus an. Bei der pixelverdoppelten RGB565-Ausgabe verbraucht man etwa 65 % der Kapazität eines Kerns für die TMDS-Kodierung und DMA-Interrupts, und der andere Kern steht dann vollständig für die Erzeugung des Videos und die Ausführung des Hauptprogramms zur Verfügung.
*Anmerkung der Redaktion: Neben der reinen Videogenerierung wurden einige Anwendungen für den Pico DVI Sock hinzugefügt. Eine davon ist das Bewegen mehrerer Portraits von Eben Upton auf dem Bildschirm. Ein 640×480-Pixel-Bild, das als Vollbild mit 8-Bit-Auflösung gespeichert wird, würde etwa 308 KB RAM benötigen (mehr als der RP2040 hat), also beschränken wir es auf ein Maximum von 320×240 mit 16-Bit-Farbe (154 KB) im RAM. Die Demo (***Bild 4***) ist nicht so pixelig, so dass ein Software-Trick im Spiel zu sein scheint.*

Mathias: Mit der Software zur Erzeugung eines DVI-Signals wird eine Bibliothek geliefert, die auch Sprites verarbeitet. Kannst du mehr darüber erzählen?
Luke: Klar! Wenn man eine solche Videoausgabe schreibt, ist das nächste Problem, auf das man stößt, dass man ein Video braucht, das man tatsächlich ausgeben kann. Die ARMv6-M-Sprite-Bibliothek ist etwas, das ich während der Arbeit an PicoDVI für genau diesen Zweck zusammengeschustert habe.
Das entscheidende Merkmal dieser Bibliothek ist, dass sie keinen Frame-Buffer zum Rendern benötigt, sondern nur einen Scanline-Buffer. Das Rendering erfolgt direkt vor der TMDS-Kodierung. Das bedeutet, dass Videoauflösungen unterstützt werden können, die mit einem einfachen Frame-Buffer nicht in den Speicher passen würden, und dass der meiste Speicherplatz für die eigentlichen Grafikkomponenten frei bleibt.
Es gibt einige recht schnelle Blit- und Fill-Routinen, einige Tiling-Routinen und einige affin transformierte Sprite-Routinen, mit denen man skalierte/gedrehte/geschnittene Sprites erstellen kann, genug für Spiele auf dem Niveau eines Game Boy Advance oder so. (*Anmerkung der Redaktion: Ein Beispiel ist in* ***Bild 5*** *zu sehen*).

Mathias: Woher hattest du die Inspiration für die Bibliothek - und die Zeit, sie zu schreiben?
Luke: Ich habe einige Zeit an scanline-basierter Grafikhardware für RISCBoy gearbeitet, und da ich das in Hardware gemacht habe, war es ziemlich einfach, es in Software nachzubilden. Alles im PicoDVI-Repository habe ich in meiner Freizeit auf meinem Laptop gemacht, mit Ausnahme der Augendiagramme, für die ich ein Oszilloskop auf der Arbeit benutzt habe.

Mathias: Es gibt eine vom Videospiel Zelda inspirierte Sprite-Demo für das RP2040 (**Bild 6**). Kannst du uns etwas über die Idee dahinter erzählen? Bei einem NES

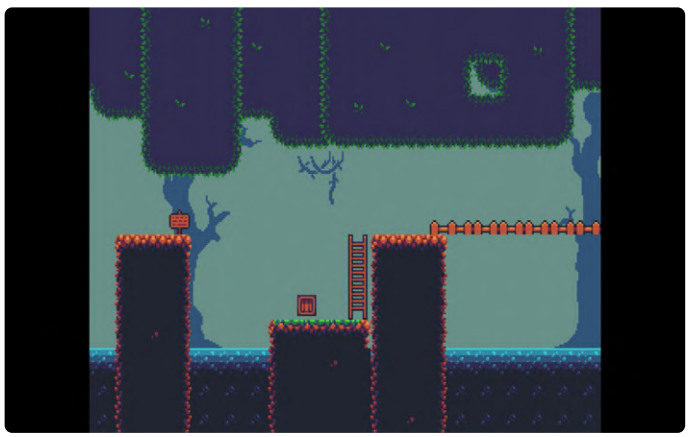


Bild 5. Sprite-Demo für den Raspberry Pi RP2040.

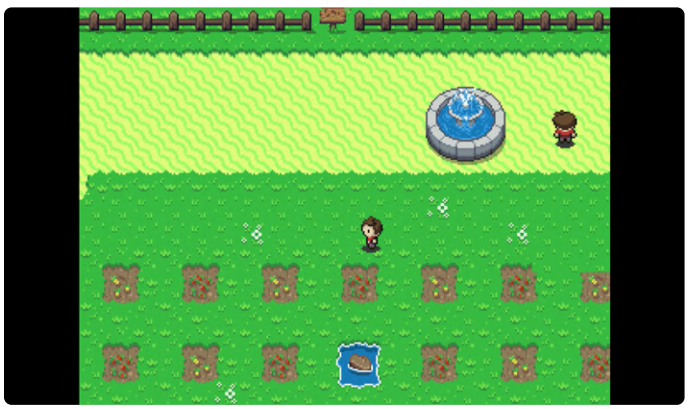


Bild 6. Walker-Demo mit DVI-Ausgang.

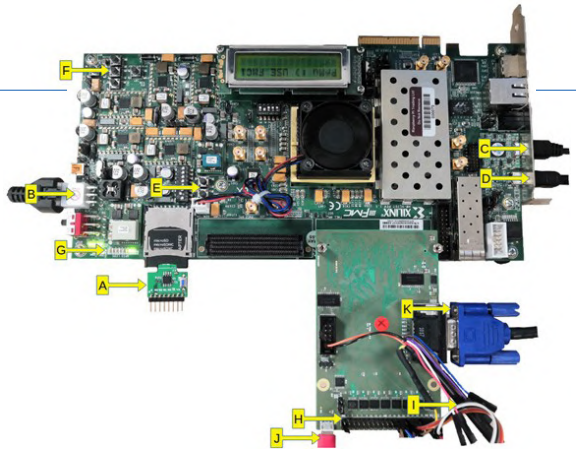
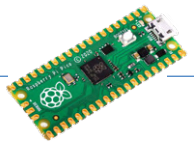


Bild 7. Prototyp Raspberry Pi RP2040-FPGA (Quelle: Luke Wren).

oder SNES würde man spezielle Hardware verwenden, um solche Bilder zu komponieren, aber hier haben wir nur zwei CPUs, die die Pixel bewegen.
Luke: Es ist eigentlich eine Portierung einer der RISCBoy-Demos. Wie du richtig sagst, gibt es eine Menge Overhead, wenn man das alles in Software macht, und der RISCBoy, der mit 36 MHz läuft, kann genauso viele Sprites auf den Bildschirm bringen wie der RP2040 mit 252 MHz. [6]

Mathias: In den Dokumenten für die erwähnte Bibliothek gibt es die Idee eines Mario-Kart-Klons für den RP2040. War das nur eine Idee, oder hat man schon damit begonnen, daran zu basteln? Es wird auch erwähnt, dass der Interpolator dafür nützlich sein könnte.
Luke: An dieser Stelle muss ich gestehen, dass der ursprüngliche Grund für den Interpolator im Chip der Wunsch war, Textur- und Kachelmapping im Stil von Mode 7 der SNES zu erzeugen, obwohl wir zwischenzeitlich einige Zeit damit verbracht haben, ihn zu einem allgemein nützlichen und fähigen Stück Hardware zu machen.
Wir haben nie einen echten MK-Klon gebaut, obwohl man im Internet viele Beispiele von Leuten sehen kann, die ähnliche Techniken verwenden. Wir hatten einen 3D-Würfel mit Textur-Mapping und Ebens Gesicht darauf, der auf einem FPGA lief.

Mathias: In der Dokumentation zu eurem Pico DVI Sock für den RP2040 Pico erwähnt ihr den Prototyp eines 48-MHz-FPGAs. Kannst du uns etwas über diesen Prototyp erzählen?
Luke: Wir hatten einen nachts laufenden Task, um ein FPGA-Image aus dem neuesten RP2040-Quellcode zu erstellen, damit wir es am nächsten Tag für die Softwareentwicklung verwenden konnten.
Wir haben einfach ein handelsübliches Virtex-7-Entwicklungsboard verwendet, mit einer Tochterplatine für die Pegelwandlung der FPGA-IOs auf 3,3 V (**Bild 7**) und einer weiteren kleinen Platine, die einen QSPI-Flash in die SD-Fassung steckt, der mit der XIP-Schnittstelle des RP2040 verbunden ist [7].
Der FPGA-Aufbau ist so ziemlich ein vollwertiger RP2040 [8] - die Takt-/Reset-Schaltung ist vereinfacht, und der ADC wurde weggelassen, aber ansonsten ist alles vorhanden und korrekt. Das macht ihn zu einer idealen Plattform für die Softwareentwicklung, auch wenn die eigentliche Verifizierung des Chips mit konventionellen Simulationen und formaler Modellüberprüfung erfolgte.



Mathias: Neben dem DVI-Ausgang arbeitest du noch an einigen anderen Projekten. Eines davon ist die PicoStation 3D. Kannst du uns ein wenig darüber erzählen? Wenn du alle Teile beschaffen könntest, wäre es dann heute noch dasselbe Design?
Luke: Also, die PicoStation 3D (**Bild 8**) ist eine der vielen Hobby-Platinen, für die ich vor der Markteinführung des RP2040 noch Luft hatte. Es ist ein Board mit einem RP2040, einer iCE40UP5K-FPGA, microSD, Audioausgang, DVI-D-Ausgang über HDMI-Buchse und zwei SNES-Controller-Fassungen. Ich habe damals viel über 3D-Grafikhardware gelesen und wollte eine Plattform, um damit zu spielen, und zwar im Rahmen einer kleinen Spielkonsole.
Es schmerzt mich, dass ich dieses Projekt so lange auf Eis legen musste, aber abgesehen von den Problemen mit den Bauteilen habe ich auch einfach zu viele andere Projekte am Laufen. Es ist alles Open-Source, also würde ich mich freuen, wenn jemand anderes die Idee aufgreift und sie weiterführt.
Ich denke, die Wahl des FPGAs ist genau richtig - er ist klein und langsam genug, damit man für seine Demos hart arbeiten muss, gerade noch DVI-D-fähig, und er hat einen großzügigen Onboard-Speicher und eine Handvoll 16-Bit-DSP-Einheiten, so dass er eine brillante Plattform zum Spielen mit Spielzeug-Grafikhardware darstellt. Außerdem lässt er sich gut mit dem RP2040 kombinieren. Worüber ich gerne noch nachdenken würde, ist das IO. Wie wäre es zum Beispiel, wenn man den DVI-Anschluss

Bild 8. PicoStation3D-Prototyp (Quelle: Luke Wren).

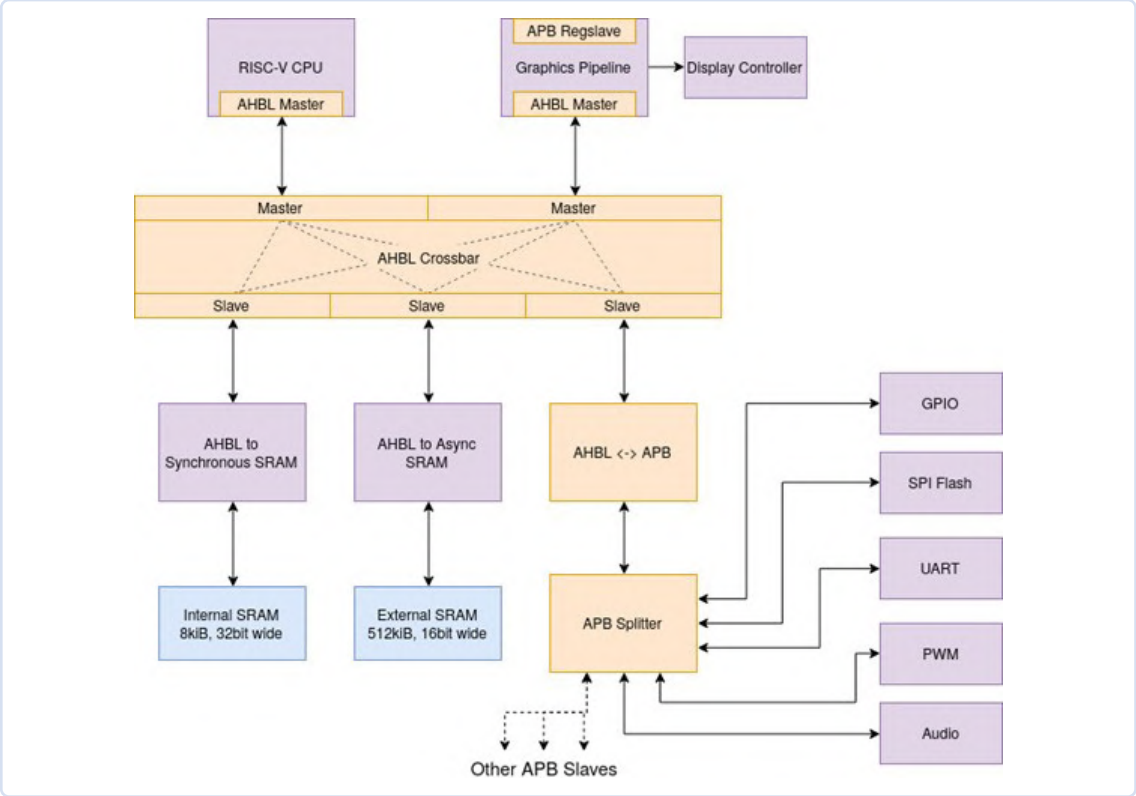


Bild 9. RISCBoy-Architektur (Quelle: Luke Wren).

zum Mikrocontroller und die SNES-Controller zum FPGA verlagern würde, und wie wäre es, wenn man die Audio-Schaltung noch ein bisschen besser gestalten würde, solche Dinge.
Ich denke, der physikalische Formfaktor ist genau richtig, da er durch die beiden SNES-Controller-Anschlüsse definiert ist.

Mathias: Neben den Raspberry-Pi-basierten Geräten hast du auch einen RISCBoy entwickelt, der von einem selbst entwickelten RISC-V-Kern und anderen Peripheriegeräten gesteuert wird, wie zum Beispiel einer Grafik-Engine (2D-Sprite-basiert?). Kannst du ein paar Worte zu dieser Entwicklung sagen?
Luke: Der RISCBoy ist mein etwas verspäteter Konkurrent des Game Boy Advance. Die vollwertige Hardware passt in einen iCE40 HX8K mit etwas externem parallelem SRAM (**Bild 9**). Irgendwann wird es eine physische Version geben, aber im Moment ist es immer noch ein HX8K-Entwicklungsboard mit etwas SRAM und Tasten und einem SPI-Display, das daran hängt

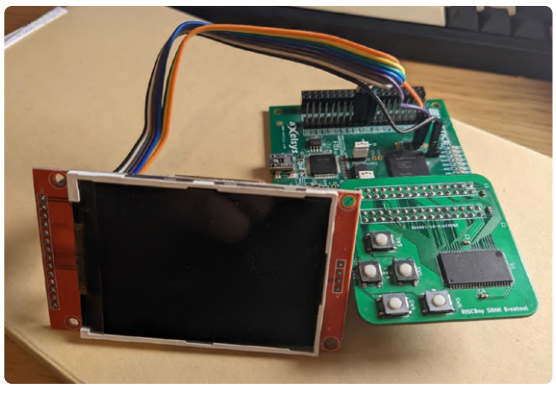
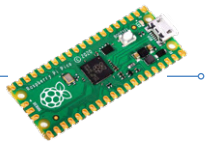


Bild 10. RISCBoy-Prototyp (Quelle: Luke Wren).

(**Bild 10**). Ich habe mit der Arbeit daran ungefähr zu der Zeit begonnen, als ich die Universität verließ. Alles ist von Grund auf neu geschrieben. Das ist zwar keine gute Art des Entwickelns, aber eine großartige Art zu lernen, und es macht mehr Spaß, Fehler zu beheben, wenn es keinen einzigen Teil des Hardware/Software-Stacks gibt, dem man vertrauen kann. Es gibt einen 32-Bit-Prozessor (Hazard5), eine programmierbare 2D-Grafikhardware und die gesamte Infrastruktur, um alles miteinander zu verbinden.
Die Grafikhardware erstellt die üblichen Sprites, Kacheln, affin-transformierte Sprites/Kacheln und so weiter, aber sie tut dies, indem sie vom Speicher aus Befehlslisten ausführt, die begrenzte Unterstützung für den Ablaufsteuerung und die Verzweigung zu Unterprogrammen bieten. Während jedes Bildes gibt der Prozessor die Befehlsliste für das nächste Bild aus. Es gibt zwei Scanline-Puffer in der Hardware, einen, in den gerendert wird, und einen, der zum Display gesendet wird, so dass die Kosten für Bandbreite, Latenz und Speicherplatzbedarf beim Rendern in einen Frame-Puffer vermieden werden.
Im Moment ist das Projekt auf Eis gelegt, aber ich habe definitiv vor, es eines Tages zu beenden.

Mathias: Dein RISC-V Core wird von Tag zu Tag reifer. Kannst du uns etwas über seine Ursprünge erzählen?
Luke: Ich habe im Moment ein paar Prozessoren in verschiedenen Stadien der Entwicklung, aber ich nehme an, du fragst nach Hazard3. Es ist ein dreistufiger In-Order-Skalarprozessor, der ursprünglich aus dem Hazard5-Quellcode (dem RISCBoy-Prozessor) abgeleitet wurde. Hazard3 ist für mich wirklich eine Lernplattform - es ist schön und gut, RISC-V-Spezifikationen zu lesen, aber das ist nicht dasselbe,



wie sie tatsächlich zu implementieren, und die Verwendung eines einfachen dreistufigen Prozessors als Basis ermöglicht es mir, mich auf die Peripherie zu konzentrieren, da die eigentliche Kern-Pipeline einfach funktioniert.
Dennoch ist die Leistung heutzutage ziemlich konkurrenzfähig (etwa 3,2 CoreMark/MHz), und ich habe viel Zeit in die Verifizierung und Dokumentation gesteckt. Die Einführung von Hardware-Debugging und die Durchführung von End-to-End-Tests mit GDB, OpenOCD und meinem eigenen JTAG-Transportmodul war wahrscheinlich der bisherige Höhepunkt des Projekts, aber ich habe während der ganzen Zeit viel gelernt.
Ich denke, dass Hazard3 in Zukunft eine Goldgrube an Komponenten für alle zukünftigen 32-Bit-Prozessoren der Embedded-Klasse sein wird, an denen ich arbeite. Ich habe bereits gesehen, wie jemand auf Twitter mein Debug-Modul verwendet hat, um Debug-Unterstützung für den Kern eines anderen Entwicklers hinzuzufügen. Der Quellcode hat keine externen Abhängigkeiten und sollte ziemlich portabel sein. Außerdem steht er unter der Apache-2.0-Lizenz.

Mathias: Derzeit ist das Design ein 32-Bit-RISC-V, der vollständig Open-Source entwickelt wird. Hast du irgendwann Ressourcen für die Entwicklung vom Raspberry Pi erhalten (Zeit, Hardware, Software-Tools)?
Luke: Alle meine Heimprojekte werden in meiner eigenen Zeit und Hardware und mit Open-Source-Tools durchgeführt. Im Moment habe ich einen Ryzen-7-5800X Rechner zu Hause, der mir bei der Ausführung von Batch-Jobs hilft. Ich verwende Yosys für die FPGA-Synthese, nextpnr für Place-and-Route und CXXRTL für die Simulation.
Ich verwende einen iCEBreaker (iCE40UP5K) und einen ULX3S (ECP5 85F) als Referenzplattformen für Hazard3, obwohl ich eine ziemlich enorme Anzahl anderer FPGA-Entwicklungsplatinen besitze, die ich eigentlich öfter verwenden sollte. Wenn ich etwas auf einem FPGA debuggen muss, komme ich gut mit meinem Saleae Logic 8 zurecht, den ich mir als Student gekauft habe, aber die meiste Zeit kann ich mich auf Simulationen verlassen.

Mathias: Du bist gerade dabei, mit deinem Kern von 32 Bit auf 64 Bit umzusteigen. Was war das schwierigste Hindernis während der bisherigen Entwicklung?
Luke: Ich habe bisher nur ein Wochenende damit verbracht, mit RV64 herumzuspielen, und das hat ausgereicht, um die RV64IC-Compliance und die Debug-Compliance-Tests zu bestehen, es gibt also keine steile Lernkurve - die Dinge werden nur größer und langsamer.
Ich habe der Einfachheit halber auf der Hazard3 Codebasis geschrieben, aber wenn ich mich ernsthaft mit einer RV64-Implementierung befassen wollte, dann wären einige Änderungen an der Mikroarchitektur erforderlich. Es gibt einen Grund, warum man nicht viele dreistufige 64-Bit-Prozessoren sieht, und schon gar nicht solche mit der Sprung-Entscheidung in Stufe 2. Hazard3 wird ein 32-Bit-Prozessor der Embedded-Klasse bleiben.

Mathias: Gibt es irgendwelche Pläne, den Kern irgendwann in der Zukunft auf echtem Chip zu testen? Oder sogar den Kern mit einer 2D-Engine zu kombinieren, um einen RISCBoy64 zu erhalten?
Luke: Ich würde gerne irgendwann mal ein SkyWater PDK-Tape ausprobieren, aber im Moment konzentriere ich mich auf andere Projekte. Ich möchte etwas Interessanteres bauen als „nur ein weiteres RISC-V-System“ und ich bin mir noch nicht ganz sicher, was das sein wird. Für etwas wie RISCBoy gibt es keinen großen Vorteil, einen 64-Bit-Prozessor zu verwenden.

Mathias: Vielen Dank für deine Zeit, Luke.

(220575-02)RG

Passende Produkte

- **Raspberry Pi Pico (SKU 19562)**
  www.elektor.de/19562
- **DVI Sock für Raspberry Pi Pico (SKU 19925)**
  www.elektor.de/19925

WEBLINKS

[1] Mathias Claußen, „Videoausgabe mit Mikrocontrollern (2)“, ElektorMag 3-4/2023: https://www.elektormagazine.de/220614-02
[2] Luke Wrens GitHub-Repository: https://github.com/Wren6991
[3] RISCBoy auf GitHub: https://github.com/Wren6991/RISCBoy
[4] pico-infones auf GitHub: https://github.com/shuichitakano/pico-infones
[5] Video bei Twitter: https://twitter.com/shuichi_takano/status/1477702448907419649
[6] Sprite-Demo, die auf der RISCBoy-Hardware mit 36 MHz läuft: https://twitter.com/wren6991/status/1333708886956707840
[7] Foto QSPI-SD-Board: https://twitter.com/wren6991/status/1134719550027632640
[8] Microcontroller RP2040 : https://raspberrypi.com/products/rp2040/

Display HAT Mini

Zeigt die Wettervorhersage auf dem Raspberry Pi

Von Clemens Valens (Elektor)

Der Display HAT Mini von Pimoroni ist mit einem rechteckigen 2,0"-LCD mit 320 x 240 Pixeln, vier taktilen Tasten und einer RGB-LED ausgestattet. Er ist für den Raspberry Pi Zero und Zero 2 W gedacht und eignet sich für viele IoT- und Heimautomatisierungsanwendungen.

Bild 1. Das Display HAT Mini eignet sich hervorragend als IoT- oder Hausautomatisierungs-Display.

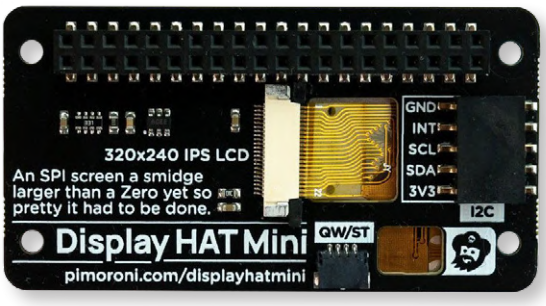


Bild 2. I²C-Verlängerungsstecker sind auf der Rückseite verfügbar.

Der *Display HAT Mini* von Pimoroni ist mit einem rechteckigen IPS-LCD (In-Plane Switching) mit 2,0" Durchmesser und SPI-Schnittstelle ausgestattet. Er ist für den Raspberry Pi Zero und Zero 2 W gedacht, aber da er den Standard-40-Pin HAT-Anschluss hat, kann er auf jeden Raspberry Pi gesteckt werden, der mit einem solchen Anschluss ausgestattet ist, wenn man vorsichtig ist, da der I²C-Anschluss des HAT (Breakout Garden Header) mit dem Display-Anschluss des Raspberry Pi kollidiert.

Spezifikationen des Display HAT

Die Auflösung des Displays beträgt 320 x 240 Pixel (3:2), was etwa 200 PPI (Pixel pro Zoll) entspricht. Es hat eine Farbtiefe von 65K. Montiert auf einem Raspberry Pi Zero ohne Abstandshalter beträgt die Gesamthöhe circa 15 mm. Das Display wird mit zwei 10 mm hohen Abstandshaltern und vier kleinen Schrauben (in meinem Fall fünf) geliefert. Für mich sind diese Abstandshalter zu hoch, 8,5 mm wären viel besser gewesen (aber nicht Standard).

Neben dem Display verfügt der HAT auch über vier taktile Tasten und eine RGB-LED. Die Drucktasten sind sehr nahe am Display platziert, was ihre Bedienung etwas erschwert. Ich vermute, dass sie so angeordnet wurden, um eine zusätzliche mechanische Unterstützung für das Display zu bieten.

Obwohl der HAT den Zugriff auf den 40-poligen HAT-Anschluss blockiert, sind Erweiterungen dank des Qw/ST-Anschlusses (Qwiic/STEMMA QT) und des so genannten Breakout Garden Headers möglich. Beide Header ermöglichen den Zugang zum I²C-Bus.

Unterstützt von Python-Bibliotheken

Um den *Display HAT Mini* auf einem Raspberry Pi zu verwenden, müssen Sie eine Bibliothek installieren. Eine ausführliche Anleitung dazu finden Sie in der GitHub-Ecke. Beispiele für die Verwendung des Displays mit Pygame und PIL sind ebenfalls vorhanden. Ich habe den *Display Mini HAT* an einen Raspberry Pi Zero 2 W angeschlossen, auf dem Buster läuft, und ihn über SSH gesteuert. Nachdem ich die Bibliotheken installiert hatte, funktionierten alle Beispiele problemlos.

Bauen Sie eine Wettervorhersage-Anzeige

Nachdem ich die Anzeige zum Laufen gebracht hatte, wollte ich etwas mit ihr machen. Im Internet hatte ich eine Sammlung hübscher Wettersymbole gefunden, und so beschloss ich, eine Anzeige für die Wettervorhersage in Python3 zu erstellen, die das entsprechende Wettersymbol zusammen mit Temperatur, Luftdruck, Luftfeuchtigkeit sowie Windrichtung und -geschwindigkeit anzeigt.

Diese Daten können von einem Online-Wettervorhersageserver bezogen werden, von denen es viele gibt, die kostenlosen Zugang bieten.

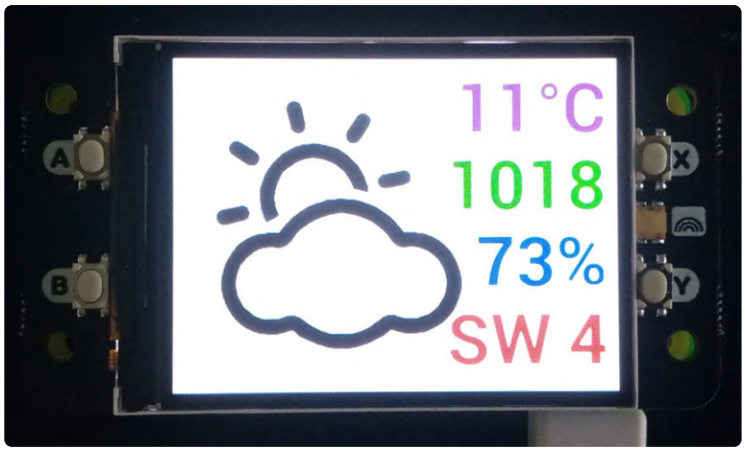
Ich habe die Tasten „A“ und „B“ verwendet, um die Helligkeit der Hintergrundbeleuchtung des Displays zu steuern. Mit der Taste „X“ kann man zwischen Grad Celsius und Grad Fahrenheit wählen, während man mit der Taste „Y“ zwischen der Windrichtung in Grad oder als Himmelsrichtung (zum Beispiel „SW“ oder „N“) umschalten kann.

Eine helle LED

Die RGB-LED liefert einige Statusinformationen. Ursprünglich sollte grün den ordnungsgemäßen Betrieb anzeigen, aber ich fand die Helligkeit selbst bei sehr niedrigen Werten zu hoch, also habe ich sie abgeschaltet. Jetzt leuchtet die LED nur noch im Falle eines Problems, nämlich rot, wenn die Wetterdaten nicht abgeholt werden konnten, und orange, wenn die Wetterdaten ungültig sind.

Beachten Sie, dass der Bildschirm standardmäßig auf dem Kopf steht, was den Tastendruck auf dem HAT betrifft. Aus diesem Grund dreht das Programm den Anzeigepuffer um 180°, bevor es ihn auf den Bildschirm kopiert.

Der Stromverbrauch meines Systems betrug bei maximaler Beleuchtungsintensität etwa 200 mA.



▲ *Bild 3. Der Display HAT Mini verfügt außerdem über vier taktile Tasten und eine RGB-LED.*

Mein Code (einschließlich der kompletten Icon-Sammlung) kann unter ClemensAtElektor auf GitHub gefunden werden.

Zusammenfassend

Zusammenfassend lässt sich sagen, dass der *Display HAT Mini* ein cooles Add-on für einen Raspberry Pi Zero (2 W) ist. Die Bildqualität ist großartig, und es ist einfach, ihn in Ihren Anwendungen zu implementieren. Die unterstützende Bibliothek bietet auch Zugriff auf die Drucktasten (für meinen Geschmack etwas zu nah am Display platziert), die RGB-LED (etwas zu hell) und die Hintergrundbeleuchtung.

Beachten Sie, dass der HAT wegen des Displays etwas größer ist als ein Raspberry Pi Zero (2 W), 35 mm statt 30 mm, also wählen Sie Ihr Gehäuse mit Bedacht.

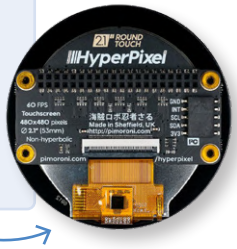
220126-02

Haben Sie Fragen oder Anmerkungen

Haben Sie technische Fragen oder Anmerkungen zu diesem Artikel? Dann wenden Sie sich bitte an den Autor unter clemens.valens@elektor.com oder die Elektor-Readaktion unter redaktion@elektor.de.

Passende Produkte

- **Pimoroni Display HAT Mini für Raspberry Pi Zero (SKU 19990)**
  www.elektor.de/19990
- **HyperPixel 2.1 Round – Hi-Res Display für Raspberry Pi (SKU 19870)**
  www.elektor.de/19870



WEBLINKS

[1] Display HAT Mini von Pimoroni: https://www.elektor.de/hyperpixel-2-1-round-hi-res-display-for-raspberry-pi
[2] Pimoroni auf GitHub: https://github.com/pimoroni/displayhatmini-python
[3] ClemensAtElektor auf GitHub: https://github.com/ClemensAtElektor/rpi_weather_display

WEEF 2022 Awards: Feiern Sie das Gute!

Von Priscilla Haring-Kuipers (Niederlande)

Am 15. November 2022 fand auf der electronica in München das zweite jährliche World Ethical Electronics Forum (WEEF) mit ausführlichen Vorträgen und Debatten zu einer Reihe von ethischen Themen rund um die Elektronik statt. Feiern Sie mit uns die vier WEEF-Preisträger.

Auf unserem zweiten World Ethical Electronics Forum (WEEF) in München am 15. November 2022 haben wir vier Auszeichnungen an sehr beeindruckende Persönlichkeiten verliehen. Aus unserem WEEF-Index wählte unsere Jury vier Gewinner anhand ihres Beitrags zur ethischen Elektronik in drei Dimensionen aus:

1) Ihr Maß an Einfluss
2) Ihr Maß an Innovation
3) Ihre Bereitschaft zu teilen

Unser World Ethical Electronics Forum Index besteht aus Personen, von denen wir bei WEEF wissen, dass sie ihr Bestes für die Ethik in der Elektronik tun, und aus Personen, die von unseren Mitgliedern und Fachleuten aus der Branche nominiert wurden. Wenn Sie jemanden kennen, der in diese Sammlung guter Menschen aufgenommen werden (und für die nächsten WEEF-Auszeichnungen kandidieren) sollte, können Sie ihn nominieren, indem Sie dieses Antragsformular auf der WEEF-Website ausfüllen.[1]

WEEF-Gewinner Sven Krumpel

Sven Krumpel [2] ist Eigentümer von CODICO, einem Unternehmen, das großen Wert auf die Work-Life-Balance seiner Mitarbeiter legt. Das Unternehmen bietet nahezu vollkommene Flexibilität bei Arbeitsplatz und Arbeitszeit und ist ein gutes Beispiel dafür, wie man wirklich in seine Mitarbeiter investiert.

„Danke, dass Sie mich nominiert haben. Ich wusste gar nicht, dass ich eine ethische Person bin. Wir sind ein Familienunternehmen, das Elektronik entwickelt und vorantreibt. Wir sehen, wie wichtig es ist, Projekte zur Energieeffizienz zu entwickeln. Ich denke, Ethik ist eine Angelegenheit der Menschen; Sie als Person sind ein Beispiel. Als Firmeninhaber hat man nicht nur Angestellte, sondern auch Menschen mit den gleichen Werten. Unsere Mitarbeiter sind ein Teil der Familie. Wir haben gerade einen 12.000 m² großen Park in der Nähe unserer Büros eröffnet, der nicht nur für die Arbeit im Freien, sondern auch für die Freizeit gedacht ist. Unsere Mitarbeiter treiben Sport im Park, laden ihre Familien ein und veranstalten Geburtstagsfeiern. Wir bauen Gemüse und Kräuter an. Wir haben Bienen. Ich denke, wir haben ein Leben im Überfluss geführt, in dem wir zu viele Ressourcen verbraucht und auch die anderer genutzt haben, und wir müssen etwas zurückgeben. Das ist es, was wir tun, und es ist nichts Besonderes."

WEEF-Gewinner Gopal Kumar Mohoto

Gopal Kumar Mohoto [3] ist ein Ingenieur, der an der Elektrifizierung des Verkehrs in Bangladesch arbeitet. Er wurde beim ClimateLaunchpad 2020 zum globalen Unternehmer Nr. 1 gewählt und war Klimabotschafter beim Global Youth Climate Network. Er hat an Batteriewechsel-Stationen für elektrische Tuk-Tuks (Auto-Rikschas) gearbeitet und steht kurz davor, die erste Charge von umgerüsteten Benzin-zu-Elektroautos an Kunden in Bangladesch zu liefern. Ein Interview mit ihm können Sie hier [4] lesen.

„Ich fühle mich sehr geehrt, nominiert worden zu sein, obwohl ich nicht weiß, von wem. Die im Westen verfügbaren Optionen für elektrisches Fahren sind im globalen Süden zu teuer. Deshalb arbeite ich an einer nachhaltigen und bezahlbaren Möglichkeit, den Verkehr hier zu elektrifizieren. Wir müssen mit den Dingen, die wir bereits haben, besser umgehen und sie nutzen, da sie wertvoll sind. Wir müssen uns um all die Materialien kümmern, die wir abgebaut haben. Wir müssen uns um Mutter Erde kümmern."

WEEF-Gewinner Frank Stührenberg

Frank Stührenberg [5] ist der Geschäftsführer von Phoenix Contact und Vorstandsmitglied der Stiftung KlimaWirtschaft. Er setzt sich maßgeblich für die All Electric Society ein, die ein Zukunftsbild einer Welt entwirft, in der regenerativ erzeugte elektrische Energie weltweit als Primärenergie in ausreichender Menge für jedermann zur Verfügung steht.

„Mir ist nicht bekannt, wer uns nominiert hat. Die Auszeichnungen, die unser Unternehmen sonst erhält, sind in der Regel technischer Natur, daher hat mich dieser Preis überrascht. Das Flair von WEEF passt zu unserem Unternehmen. Wir sind ein Familienunternehmen und haben inzwischen mehr als 20.000 Familienmitglieder. Wir müssen daran arbeiten, unseren moralischen Kompass nicht zu verlieren, und wir arbeiten aktiv daran. Wir haben unsere langfristige Strategie darauf ausgerichtet, zu 100 % auf erneuerbare elektrische Energie zu setzen. Wir hatten das Gefühl, dass wir das tun sollten, also haben wir angefangen. Wir brauchen Energie, um die Welt zu entwickeln, und was nützt es, ein 10-Milliar-

Milda Pladaitė (links) erhält eine Auszeichnung von Beatriz Souza von Elektor. (Foto Messe München)

Sven Krumpel (rechts) erhält eine Auszeichnung von Johann Wiesböck von ELEKTRONIKPRAXIS (Foto Messe München)

Gopal Kumar Mohoto.

Frank Stührenberg (rechts, mit dem Geschäftsführer der Messe München Reinhard Pfeiffer, Foto Messe München).

den-Euro-Unternehmen zu sein, wenn wir keine Welt haben, in der wir leben wollen? Ich danke Ihnen, dass Sie diese Entwicklung und diese Initiative unterstützen."

WEEF-Gewinnerin Milda Pladaitė

Milda Pladaitė [6] ist eine Bauingenieurin aus Litauen und Mitglied des Young Engineers Future Leaders Committee des World Federation of Engineering Organizations (WFEO). Sie hat bei der WFEO eine Arbeitsgruppe zum Ziel 13 für nachhaltige Entwicklung gegründet, die einen Beitrag zur nachhaltigen Entwicklung der Länder leisten soll, indem sie die Arbeit der jungen Ingenieure der WFEO für den Klimaschutz einsetzt. Derzeit arbeitet sie an einer sozialen Initiative, die Ingenieuren hilft, unternehmerisch tätig zu werden, indem sie sie mit der Industrie, sozialen Investoren und akademischen Organisationen zusammenbringt.

„Ich habe viele Menschen für Preise nominiert, aber ich weiß nicht, wer mich für diesen Preis nominiert hat. Eine Freundin sagte, als sie hörte, dass ich für die WEEF-Auszeichnung nominiert wurde: ‚Ich wusste gar nicht, dass du ein Spezialist für Ethik bist!' Das bedeutet natürlich, dass sich jeder in der Ethik einsetzen kann. Dieses Forum ist eine großartige Initiative, und ich bin sicher, dass es in Zukunft weiter wachsen wird. Dies ist eine einzigartige Gelegenheit, über Ethik in der Elektronik zu diskutieren."

An der Überraschung unserer WEEF-Preisträger können wir sehen, dass es nicht immer offensichtlich ist, was Ethik in der Elektronik ist. Vielleicht zögern wir als Branche, über Ethik zu sprechen, obwohl wir das gar nicht müssen. Ziel unseres World Ethical Electronics Forum ist es, eine Plattform zu bieten, um zu diskutieren, was wir tun und warum wir es tun, um Geschichten und Praktiken miteinander zu teilen und um das Gute gemeinsam zu feiern.

(220650-02)SG

Über die Autorin

Priscilla Haring-Kuipers schreibt über Technologie aus einer sozialwissenschaftlichen Perspektive. Sie interessiert sich besonders für Technologien, die das Gute im Menschen unterstützen, und glaubt fest an die Wirkungsforschung. Sie hat einen MSc in Medienpsychologie und lässt This Is Not Rocket Science möglich werden.

Das World Ethical Electronics Forum

Das World Ethical Electronics Forum (WEEF) inspiriert globale Innovatoren mit offenen Diskussionen und Veröffentlichungen über Ethik und nachhaltige Entwicklungsziele. Besuchen Sie worldethicalelectronicsforum.com, um sich inspirieren zu lassen und sich zu beteiligen.

WEBLINKS

[1] WEEF Index, Nominierungsformular: https://worldethicalelectronicsforum.com/nomination-form
[2] WEEF, Sven Krumpel: https://worldethicalelectronicsforum.com/index/184374/Sven__Krumpel
[3] WEEF, Gopal Kumar Mohoto: https://worldethicalelectronicsforum.com/index/182949/Gopal_Kumar_Mohoto
[4] P. Haring-Kuipers, „Retrofitting and Upcycling: Interview with Gopal Kumar Mohoto", ElektorMagazine.com, 2022: https://www.elektormagazine.com/gopal-interview-for-next-eia
[5] WEEF, Frank Stührenberg: https://worldethicalelectronicsforum.com/index/184378/Frank__St%C3%BChrenberg
[6] WEEF, Milda Pladaitė: https://worldethicalelectronicsforum.com/index/182955/Milda__Pladait%C4%97

elektorstore

www.elektor.de

Der Elektor Store

Nie teuer, immer überraschend!

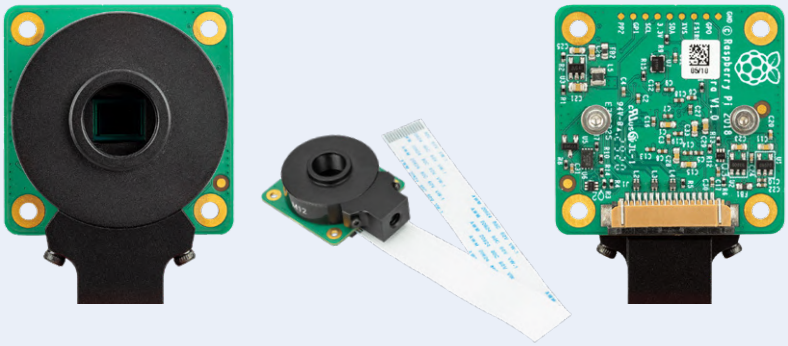
Der Elektor Store hat sich vom Community-Store für Elektor-eigene Produkte wie Bücher, Zeitschriften, Bausätze und Module zu einem umfassenden Webshop entwickelt, der einen großen Wert auf überraschende Elektronik legt.

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). Unsere Bedingungen:
Nie teuer, immer überraschend!

Raspberry Pi High Quality Camera Module (M12 Mount)

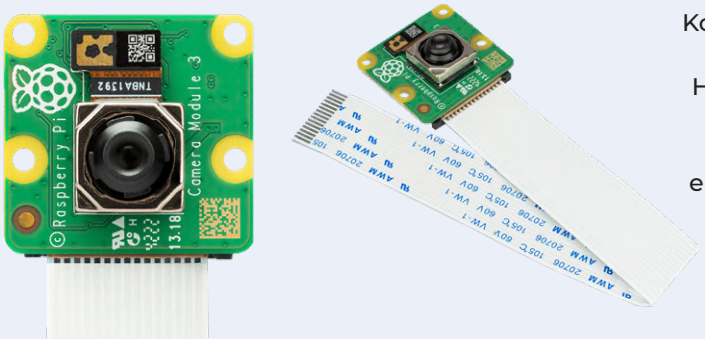
Die Raspberry Pi High Quality Camera ist eine günstige, hochwertige Kamera von Raspberry Pi. Es bietet eine Auflösung von 12 Megapixel und einen Sensor mit 7,9 mm Diagonale für eine beeindruckende Leistung bei schlechten Lichtverhältnissen.

Preis: 59,95 €



www.elektor.de/20366

Raspberry Pi Camera Module 3



Raspberry Pi Camera Module 3 ist eine Kompaktkamera von Raspberry Pi. Es bietet einen IMX708 12-Megapixel-Sensor mit HDR und verfügt über einen Autofokus mit Phasenerkennung. Das Camera Module 3 ist in Standard- und Weitwinkelvarianten erhältlich, die beide mit oder ohne Infrarot-Sperrfilter erhältlich sind.

Preis: 32,95 €

www.elektor.de/20362

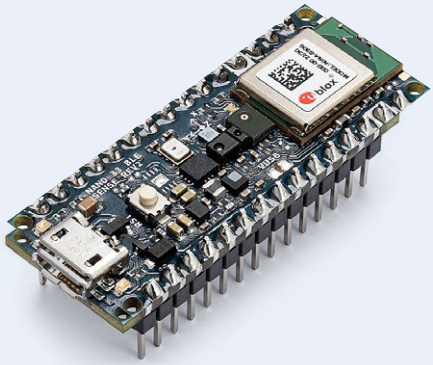
YDLIDAR OS30A 3D-Tiefenkamera

Preis: 139,95 €

Mitgliederpreis: 125,96 €

www.elektor.de/20350

Arduino Nano 33 BLE Sense Rev2 mit Header



Preis: 54,95 €

Mitgliederpreis: 49,46 €

www.elektor.de/20404

Arduino Student Kit

Preis: 74,95 €

Mitgliederpreis: 67,46 €

www.elektor.de/20329

Home Appliance Hack-and-IoT Guidebook

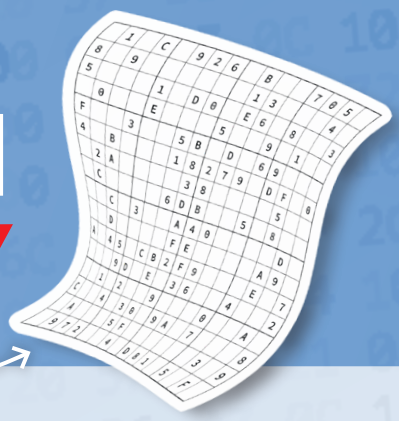
Preis: ~~54,90 €~~

Sonderpreis: 39,95 €

www.elektor.de/20370

Hexadoku

Sudoku für Elektroniker



Wie in jeder Ausgabe finden Sie auch in diesem Heft unser ganz spezielles Sudoku. PC, Oszilloskop und Lötkolben können sich erholen, während Ihre kleinen grauen Zellen auf Hochtouren arbeiten. Wenn Sie alle Hex-Ziffern in den grauen Kästchen herausgefunden haben, sollten Sie uns diese gleich zumailen – denn hier warten fünf Elektor-Gutscheine!

Die Regeln dieses Rätsels sind ganz einfach zu verstehen: Bei einem Hexadoku werden die Hexadezimalzahlen 0 bis F verwendet, was für Elektroniker und Programmierer ja durchaus passend ist. Füllen Sie das Diagramm mit seinen 16 x 16 Kästchen so aus, dass alle Hexadezimalzahlen von 0 bis F (also 0 bis 9 und A bis F) in jeder Reihe, jeder Spalte und in jedem Fach mit 4 x 4 Kästchen (markiert durch die dickeren schwarzen Linien) **genau einmal** vorkommen. Einige Zahlen sind bereits eingetragen, was die Ausgangssituation des Rätsels bestimmt. Wer das Rätsel löst – sprich die Zahlen in den grauen Kästchen herausfindet – kann einen von fünf Gutscheinen im Wert von 50 Euro gewinnen!

EINSENDEN

Schicken Sie die Lösung (die Zahlen in den grauen Kästchen) per E-Mail oder Post an:

Elektor Redaktion
Lukasstraße 1
52070 Aachen

E-Mail: hexadoku@elektor.de

Als Betreff bitte nur die Ziffern der Lösung angeben!

Einsendeschluss ist der 15. April 2023.

DIE GEWINNER DES HEXADOKUS AUS DER AUSGABE NOVEMBER/DEZEMBER STEHEN FEST!

Die richtige Lösung ist: **AEF1C**

Aus allen Einsendungen mit der richtigen Lösung haben wir die fünf Gewinner eines Elektor-Wertgutscheins über je 50 € gezogen.

Die Namen der Gewinner werden unter www.elektormagazine.de/hexadoku bekannt gegeben.

Herzlichen Glückwunsch!

	1	3	F									E	2	7	
0	6	5											8	C	D
	8				2					E				6	
9	B	D	E	4		7			C		2	F	3	A	5
F	7	E	8			D			B			A	6	5	1
	9	0		6	1					3	F		4	B	
	D	4			F					9			7	8	
1		6	3	2			B	D			8	9	E		F
			6		8	0			D	2		5			
	E	A			D	9	2	8	5	F			0	1	
5				1			E	4			9				8
	0		9		5		4	A		1		C		F	
D						1			8						E
		9		8	7					D	A		5		
	A		1		9	3			2	7		8		D	
7	5					4	F	E	3					9	2

E	5	2	A	4	B	8	C	6	3	F	0	D	7	9	1
0	3	B	4	6	F	2	7	D	9	1	A	E	5	8	C
C	F	1	7	D	E	9	A	B	4	5	8	2	0	3	6
6	D	8	9	0	1	3	5	C	7	E	2	B	A	F	4
1	8	F	3	5	7	4	0	E	B	2	6	9	D	C	A
B	4	C	D	3	2	6	9	8	0	A	5	F	E	1	7
5	A	E	2	8	C	D	1	7	F	9	4	6	B	0	3
7	0	9	6	B	A	E	F	1	C	3	D	4	2	5	8
F	9	7	B	E	6	5	4	0	8	D	C	1	3	A	2
3	E	5	C	7	8	A	2	F	1	B	9	0	4	6	D
D	2	0	8	9	3	1	B	4	A	6	E	C	F	7	5
4	6	A	1	C	0	F	D	5	2	7	3	8	9	B	E
8	C	3	F	1	5	B	E	9	D	4	7	A	6	2	0
9	1	4	0	2	D	7	6	A	5	C	B	3	8	E	F
A	B	6	5	F	4	0	3	2	E	8	1	7	C	D	9
2	7	D	E	A	9	C	8	3	6	0	F	5	1	4	B

Treten Sie jetzt der Elektor Community bei!

Jetzt elektor GOLD membership Mitglied werden!

- Komplettes Webarchiv ab 1970
- 8x Elektor Doppelheft (Print)
- 8x Digital (PDF)
- 10% Rabatt im Online-Shop und exklusive Angebote
- Zugriff auf über 5.000 Gerber Dateien aus Elektor Labs

Auch erhältlich

Die digitale elektor GREEN membership Mitgliedschaft!

- Zugang zu unserem Webarchiv
- 10% Rabatt in unserem Online-Shop
- 8x Elektor Doppelheft (PDF)
- Zugriff auf über 5.000 Gerber Dateien aus Elektor Labs

www.elektormagazine.de/Abonnement

elektor design > share > earn

Mehr lieferbar

Die größte Auswahl an Halbleitern und elektronischen Bauelementen auf Lager und versandfertig™

mouser.de

MOUSER ELECTRONICS