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

elektor MAG

design › share › earn

596
SEPTEMBER/OKTOBER 2023
ELEKTORMAGAZINE.DE
* SEIT 1961 *

Raspberry Pi Pico als Spektrum-Analyser

FFTs auf preiswerter Hardware-Basis

IM FOKUS

Drahtlose Kommunikation

SDR-Zeitzeichen-Empfänger

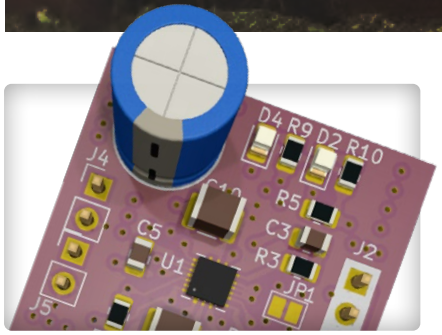
Am Puls der Zeit

Ist Mobilfunk die energiesparendste IoT-Option?

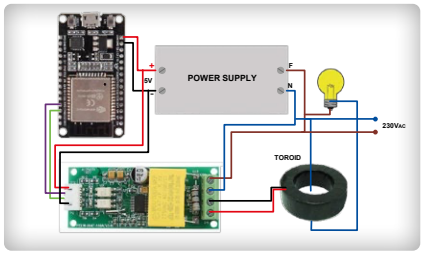
Energieanforderungen für LTE-M und NB-IoT

LoRa, ein Schweizer Taschenmesser

Das LoRa-Protokoll und seine Vorteile



Motortreiber-Breakout-Board
5-A-Treiber für DC-Motoren
S. 56



Cloud-basierter Energiezähler
Mit ESP32-Modul und PZEM-004T-Spannungs-/Stromsensor
S. 102

±40 V Linearer Spannungsregler
Stromversorgung für den Fortissimo... und mehr!
S. 18

300010
4 198630 314904

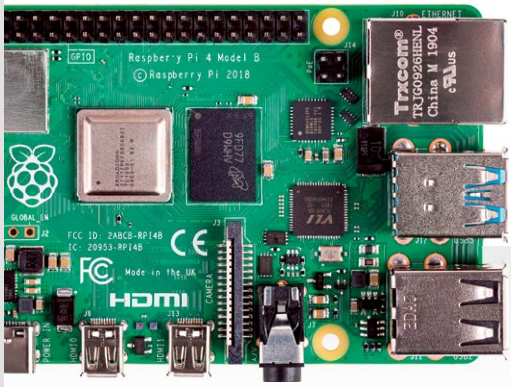
UNSER SORTIMENT VON TECHNIKERN FÜR TECHNIKER

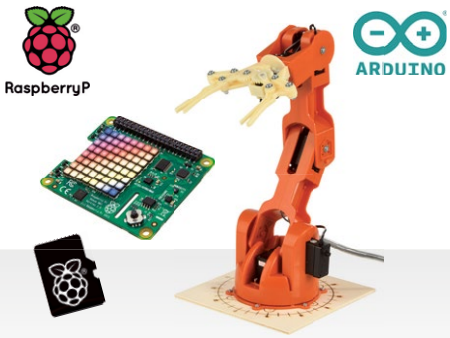
The best part of your project: www.reichelt.de

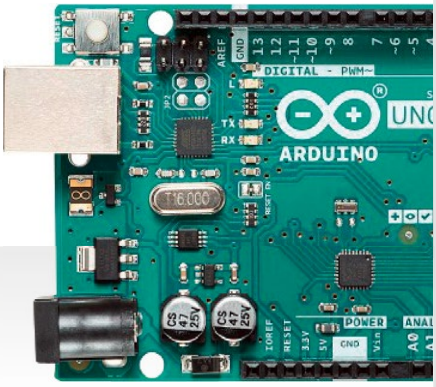
Nur das Beste für Sie – von über 1.500 Markenherstellern

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

Entwicklerboards – Kompakte Spezialisten für Elektronikprojekte







Die ideale Open-Source-Plattform für vielseitige Entwicklungen und zur einfachen Integration in bestehende Anwendungen

▶ https://rch.lt/rpi

Raspberry Pi APPROVED RESELLER

Die Open-Source-Elektronik-Prototyping-Plattform basierend auf flexibler und einfach bedienbarer Hard- und Software

▶ https://rch.lt/arduino

ARDUINO OFFICIAL RESELLER 2021

THE BEST PARTS OF YOUR PROJECTS

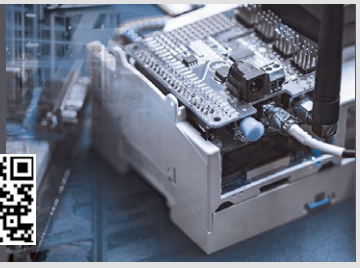
KI UND MACHINE LEARNING

Jetzt entdecken ▶
https://rch.lt/ki

reichelt elektronik MAGAZIN

Fachbeitrag – Entwicklerboards im industriellen Einsatz

Jetzt lesen ▶
https://rch.lt/MG784



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

www.reichelt.de

Bestellhotline: +49 (0)4422 955-333

reichelt elektronik – The best part of your project

IMPRESSUM

54. Jahrgang, Nr. 596
September/Oktober 2023
ISSN 0932-5468

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

Für alle Ihre Fragen
service@elektor.de

Mitglied werden
www.elektormagazine.de/abo

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

Cover-Bild
Midjourney, bearbeitet von Harmen Heida

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

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

PEFC zertifiziert
Dieses Produkt stammt aus nachhaltig bewirtschafteten Wäldern und kontrollierten Quellen
PEFC
PEFC/30-31-151
www.pefc.de

EDITORIAL

Jens Nickel

Chefredakteur ElektorMag

Von Einfach bis Modern

Unsere Zukunft wird weitgehend drahtlos sein. Zur Zeit forschen Unternehmen und Hochschulen schon an 6G, das Datenraten von bis zu 400 GBit/s erreichen wird und mit Frequenzen bis 3 THz arbeitet - losgehen soll das Ganze ab 2030. Das Spektrum von Drahtlos-Anwendungen ist allerdings schon heute riesig. Es reicht von der Positionsbestimmung über den stromsparenden Langstrecken-Datenfunk bis zum Einsatz eines Smartphones als Fernsteuerung. In jedem Fall hatten wir für dieses Heft so viele Artikel und Ideen, dass wir unmöglich alles in eine Ausgabe packen konnten. Die Vielfalt der Elektronikthemen, für die wir von unseren Lesern besonders geschätzt werden, wäre sonst gefährdet gewesen. Einige unserer Wireless-Artikel - darunter ein interessanter Bericht über die zentimetergenaue Positionsbestimmung zu maker-freundlichen Kosten - finden Sie daher in den kommenden Ausgaben.
Für all diejenigen, die besonders an bestimmten Bereichen wie „Leistungselektronik", „Wireless", „Messen und Testen" oder „Elektronik-Produktion" interessiert sind, gibt es in Kürze spezielle Themenseiten auf unser Website Elektormagazine.de. Sie werden dort nicht nur Artikel aus unseren Heften, sondern auch exklusive Online-Inhalte finden. Zu Hintergrundberichten und (Video-)Tutorials für Einsteiger gesellen sich dabei zusätzliche Projekte aus unserer weltweiten Community.
Vielleicht werde ich dort auch einmal mein ganz persönliches Wireless-Projekt vorstellen, ein drahtlos angesteuertes, von einzelnen Lithium-Akkus versorgtes Boxensystem für den Außeneinsatz. Es ist zwar alles (noch) aus fertig zugekauften Komponenten zusammengesetzt; dennoch ist bereits einige Arbeit hineingeflossen. Jetzt wollen Sie vielleicht wissen, welchen hypermodernen Wireless-Standard ich nutze? Nun, es ist guter alter FM-Funk auf 863..865 MHz, der auch bei „Silent Discos" zum Einsatz kommt. Wenn es auf eine kinderleichte Einrichtung und niedrigste Latenzen ankommt, ist Einfachheit eben immer noch Trumpf!

Veröffentlichen Sie bei Elektor!

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

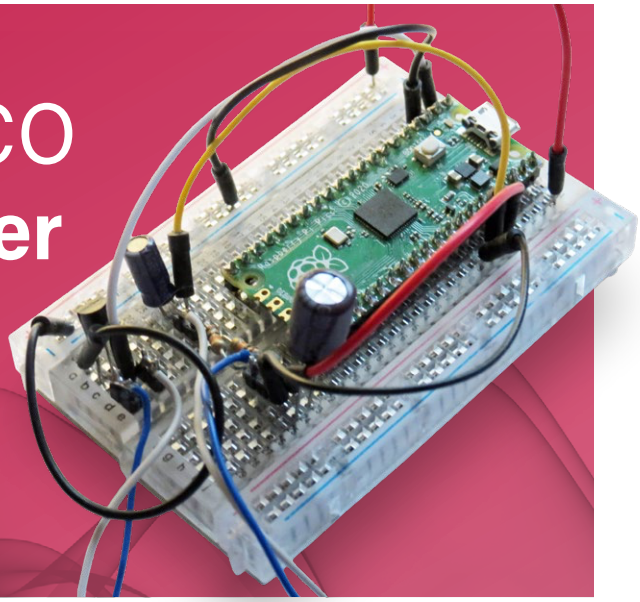
www.elektormagazine.com/submissions

Elektor-Labs: Ideen und Projekte

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

www.elektormagazine.de/labs

Raspberry Pi Pico
als Spektrum-Analyser
FFTs auf preiswerter Hardware-Basis



6

Rubriken

LoRa, ein Schweizer Taschenmesser
Teil 1: Das LoRa-Protokoll und seine Vorteile
36

Hintergrund

Industry

Projekte

Vorschau

Elektor November/Dezember 2023
Das nächste Heft ist wie immer randvoll gefüllt mit Schaltungsprojekten, Grundlagen sowie Tipps und Tricks für Elektroniker. Schwerpunkt wird die Elektronik-Entwicklung und -Produktion sein.

Aus dem Inhalt:
- PCB-Dienstleister mit und ohne Bestückung
- KI-Tools für Entwickler
- BLE in der Praxis
- Günstige GNSS-RTK-Systeme
- DIY-Heizplatte
- ChatGPT verbessert Firmware
- Maßgeschneidertes Gehäuse designen
- KiCad 7 - die neuen Features
- Solarbetriebene Lichterkette

Und vieles mehr!

Elektor November/Dezember 2023 erscheint am 15. November 2023. Änderungen vorbehalten!

Raspberry Pi Pico als Spektrum-Analyser

FFTs auf preiswerter Hardware-Basis

Von Prof. Dr. Martin Oßmann

Dieser Artikel zeigt, wie man mit einem simplen Raspberry Pi Pico einen softwarebasierten Spektrum-Analyser bauen kann. Eine Basisversion mit einer Auflösung von 12 Bit und einer Sampling-Rate von 500 kS/s eignet sich sehr gut für Messungen im Audiobereich. Mit einem externen ADC werden sogar 50 MS/s erreicht!

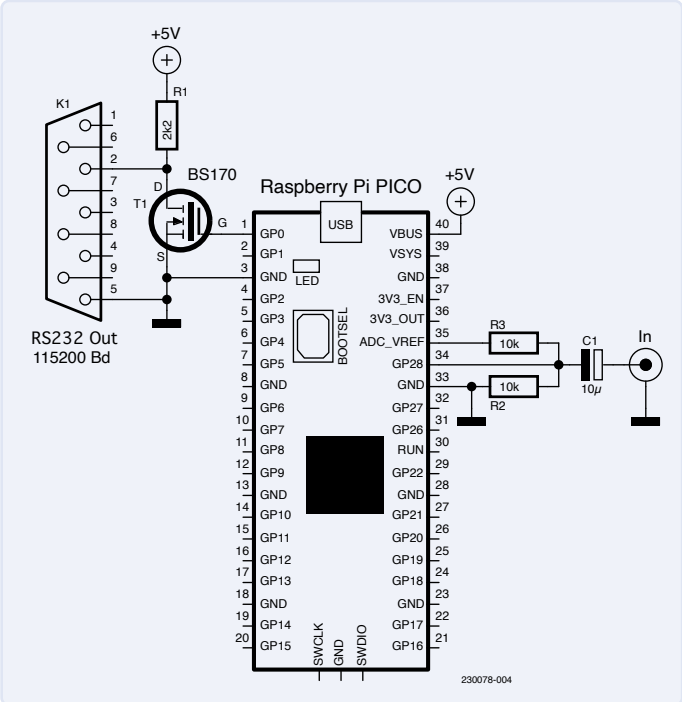


Bild 1. Minimale Verschaltung des Pico-Boards für Datentransfer zum PC.

Verwendet man den integrierten ADC des Mikrocontrollers RP2040, werden analoge Signale mit bis zu 500 kS/s abgetastet und mit 12 Bit aufgelöst. Im einfachsten Fall dient ein PC zur eigentlichen Frequenzanalyse und Anzeige der Resultate. Als erste Ausbaustufe wird ein LCD angeschlossen, und schon hat man einen Stand-Alone-Analyser. Mit einem externen ADC werden im nächsten Schritt bis zu 50 MS/s erreicht, wodurch sich der Analyser dann für Signale bis 25 MHz eignet.

500 kS/s und PC-Anschluss

Für erste Experimente wird der Raspberry Pi Pico wie in **Bild 1** beschaltet. R2 und R3 heben den ADC-Nullpunkt auf den halben Aussteuerungsbereich an. C1 entfernt Gleichspannungsanteile des Eingangssignals. Der verwendete Eingang ADC2 nutzt gemeinsam mit GPIO28 den Pin 34. T1 dient zusammen mit R1 der Herstellung eines TTL-Pegels für die Datenausgabe zum PC via serielle Schnittstelle mit 115.200 Bd. Da die Beschaltung recht einfach ist, kann man sie gut auf einem Steckbrett wie in **Bild 2** realisieren. Der ADC arbeitet zunächst mit der Default-Abtastrate von 500 kHz. Er übergibt seine 12-bit-Samples an eine FIFO-Einheit, von wo aus sie in einen Puffer gelangen, der *N* Werte speichern kann. *N* hat typischerweise einen Wert von 1.024. Immer wenn der Puffer gefüllt ist, werden die Werte seriell an den PC geschickt.

DFT und FFT

Zur Berechnung des Spektrums wird die DFT (**D**iskrete **F**ourier **T**ransformation) verwendet. Diese berechnet aus *N* Abtastwerten s_n (mit $n = 0$ bis $N-1$) des Eingangssignals dann *N* Spektrumswerte Z_k (mit $k = 0$ bis $N-1$). Die Transformationsvorschrift ist dabei:

$$Z_k = \sum_{n=0}^{N-1} s_n e^{-\left(\frac{2\pi i k n}{N}\right)} = \sum_{n=0}^{N-1} s_n \left[\cos\left(\frac{2\pi kn}{N}\right) - i \sin\left(\frac{2\pi kn}{N}\right)\right]$$

Die DFT besteht aus *N* komplexen Zahlen. Ist das Eingangssignal wie hier reell, ist die resultierende DFT symmetrisch und es gilt:

$$Z_{N-k} = \overline{Z}_k$$

Es sind also nur die ersten $N/2$ Werte relevant. Das entspricht dem Abtasttheorem, laut dem sich nur bis zur halben Abtastrate

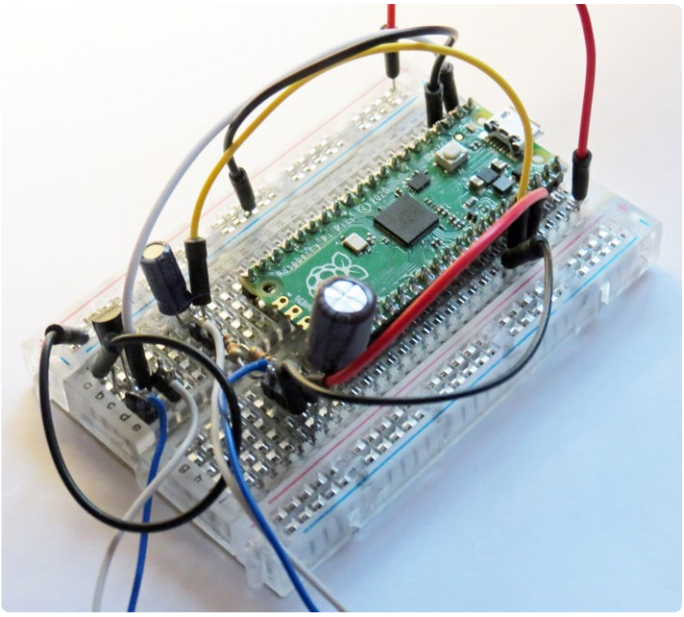


Bild 2. Mein Steckbrettaufbau der Schaltung von Bild 1.

eindeutige Werte ergeben. Folglich werden nachfolgend immer nur die ersten $N/2$ Spektralwerte genutzt. Das berechnete Spektrum entspricht dem des mit der Periode N fortgesetzten Eingangssignals S. Um zu verstehen, warum die DFT tatsächlich eine Spektralanalyse durchführt, betrachtet man zuerst, wie das Eingangssignal $s_n = cos(2\,\pi\,m\,n\,/\,N)$ mit zum Beispiel $m = 5$ und $N = 16$ transformiert wird. Dieses Eingangssignal ist eine Cosinus-Schwingung mit der Frequenz $f = f_s\,m\,/\,N$.
Als transformierte Werte ergeben sich $Z_m = Z_5 = N/2 = 8$, und die anderen Werte Z_k sind alle Null. Eine Schwingung der Frequenz m macht sich also genau im m-ten transformierten Wert bemerkbar. Wenn statt des Cosinus-Signals ein Sinus-Signal transformiert würde, wäre der Imaginärteil von Z_m ungleich Null. Doch was passiert, wenn man ein Signal transformiert, das sich aus Schwingungen verschiedener Frequenzen zusammensetzt? Auch das ist einfach: Die DFT ist nämlich eine lineare Transformation, weshalb die DFT einer Summe auch der Summe der DFTs der Einzelsignale entspricht. Als Formel bekommt man

$$DFT(u\,Un + v\,Vn) = u\,DFT(Un) + v\,DFT(Vn)$$

mit den Vorfaktoren u und v.
Wenn sich das Eingangssignal aus mehreren Schwingungen $s_n = cos(2\,\pi\,m\,n\,/\,N)$ mit verschiedenen Frequenzen m zusammensetzt, ergeben sich genau die Einzelschwingungen im Spektrum. Ein Signal $s_n = A\,cos(2\,\pi\,m\,n\,/\,N)$ mit der Amplitude A ergibt im Spektrum den Wert $Z_m = A\,N\,/\,2$. Dieser Wert ist von der Samplezahl abhängig. Um das zu beseitigen, skaliert man die DFT mit dem Faktor $2\,/\,N$. Das neue Spektrum hat dann die Werte $W_k = 2\,Z_k\,/\,N$. Ein Cosinus-Signal mit der Amplitude A ergibt dann genau einen Spektrumswert A. Der Amplitudenwert A ist dabei der Spitzenwert des Cosinus-Signals. Für den Effektivwert muss man noch durch $\sqrt{2}$ teilen.

Rechenzeit

Die Rechenzeit der DFT wächst quadratisch mit der Anzahl N der Samples. Sie kann daher bei größerem N schon recht lange dauern. Messungen mit dem Raspberry Pi Pico führten zur Näherung

Tabelle 1. Rechenzeit für verschiedene N.

N	DFT in s	FFT in s
256	0,590	0,008
1.024	9,437	0,041
4.096	150,995	0,197
16.384	2.415,919	0,918

$T_{DFT} \approx 9\,N^2\,\mu s$. Wenn N eine Zweierpotenz ist, kann man statt der DFT die FFT (**F**ast **F**ourier **T**ransform) verwenden, da sie die gleichen Werte wesentlich schneller berechnet. Auf dem Pico-Board entspricht die Rechenzeit $T_{FFT} \approx 4\,N\,log_2(N)$. In **Tabelle 1** finden sich die Laufzeiten für einige praktikable Werte von N.
Für eine einfache Spektrumanalyse wird häufig $N = 1.024$ verwendet. Die Rechenzeit der DFT wäre hier mit circa 9 s zwar noch erträglich, doch die FFT ist mit 40 ms sehr viel schneller. Gelegentlich kommt hier auch $N = 16.384$ vor, bei dem die DFT schon gut 40 Minuten dauern würde – inakzeptabel langsam. Die FFT benötigt hierfür weniger als eine Sekunde! N wird daher zugunsten der FFT immer als Zweierpotenz gewählt.

Effektivwert und dBm

In der Nachrichtentechnik werden Größen oft als Effektivwerte angegeben. Da man hierzu die zeitabhängige Größe quadriert, mittelt und dann die Wurzel zieht, hat sich hierfür der Terminus RMS (**R**oot **M**ean **S**quare) eingebürgert. Für ein diskretes periodisches Signal mit der Periodenlänge N wie bei den DFT-Signalen ergibt sich der Effektivwert folgendermaßen:

$$RMS_s = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1}|s_n|^2}$$

Man kann auch den Effektivwert der Spektralwerte bilden. Man erhält dann

$$RMS_Z = \sqrt{\frac{1}{N}\sum_{k=0}^{N-1}|Z_k|^2} \quad \text{und} \quad RMS_W = \sqrt{\frac{1}{N}\sum_{k=0}^{N-1}|W_k|^2}$$

Interessanterweise gilt nun

$$RMS_S = RMS_Z\,/\,\sqrt{N} \quad \text{beziehungsweise} \quad RMS_S = \sqrt{N}\cdot RMS_W\,/\,2$$

Man kann den Effektivwert also auch im Frequenzbereich statt im Zeitbereich bilden. Hierfür muss man nur den Vorfaktor berücksichtigen. Dieser Zusammenhang wird auch als Parsevalsche Gleichung [1] bezeichnet. Die Energie im Zeitbereich spiegelt sich also durch die Summe der Energien der Einzelschwingung im Frequenzbereich genau wider. Energie- oder Leistungsverhältnisse werden in der Nachrichtentechnik gerne in dB angegeben. Für das Leistungsverhältnis $v = P_1\,/\,P_0$ erhält man $v_{dB} = 10\,log_{10}(P_1\,/\,P_0)$. Bei absoluten Pegeln ist der Bezugspegel $P_0 = 1\,mW$ üblich. Man erhält dann $P_{dBm} = 10\,log_{10}(P\,/\,1\,mW)$. Will man Spannungen U

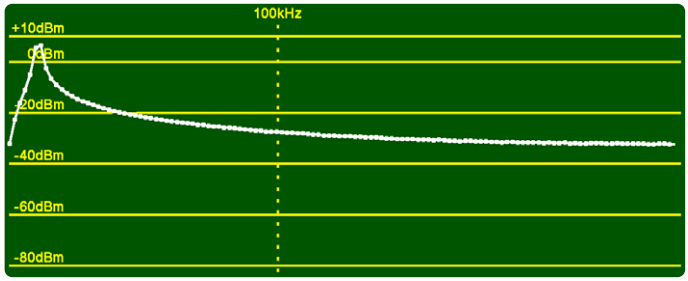


Bild 3. Schlechte Spektraldarstellung bei halbzahliger Frequenz.

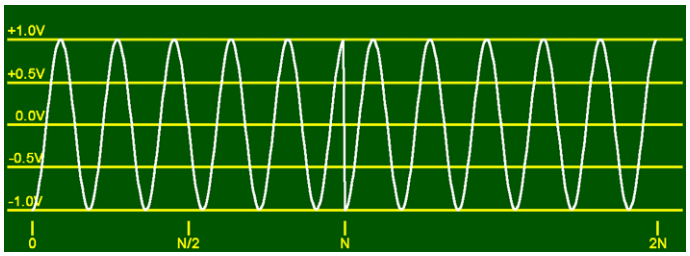


Bild 4. Bei Sample N ergibt sich ein Amplitudensprung bei periodischer Fortsetzung des Signals.

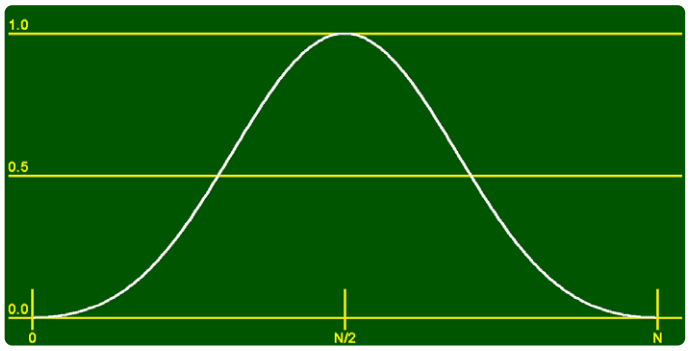


Bild 5. Blackman-Fensterfunktion.

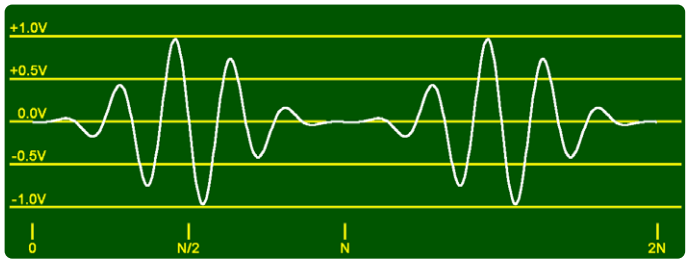


Bild 6. Das mit der Fensterfunktion multiplizierte Signal hat an den Rändern einen stetigen Verlauf.

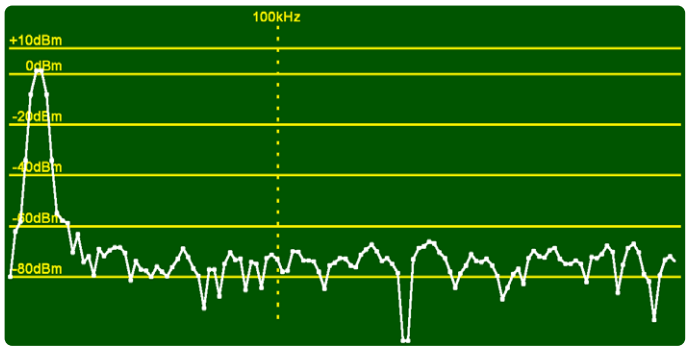


Bild 7. Spektrum nach Anwendung der Fensterfunktion.

in dBm angeben, nutzt man die Leistung P an einem Widerstand von $R = 50\ \Omega$, an dem die Spannung U_{RMS} anliegt. Da am Widerstand die Leistung $P = U^2 / R$ umgesetzt wird, erhält man $U_{dBm} = 10\ log_{10}(U^2_{RMS} / (R\ 1\ mW))$.
Beim hier beschriebenen Spektrum-Analyser wird die Y-Achse in dBm skaliert, der üblichen logarithmischen Darstellung bei Spektrum-Analysern. Doch Achtung: In der Audiotechnik bezieht sich dBm oft einen Widerstand von 600 Ω.

Signale und Fenster

Bisher wurden Eingangssignale betrachtet, die genau in ein Fenster von N Werten passen. Die Frequenz m des Signals $s_n = A\ cos(2\ \pi\ m\ n\ /\ N)$ war also ganzzahlig. **Bild 3** zeigt das Spektrum bei $m = 5{,}5$ und $N = 256$.
Eigentlich würde man ein linienartiges Spektrum erwarten, weil das Signal ja nur eine Frequenz enthält. Stattdessen erhält man ein Spektrum, das beidseitig der Maximalwerte nur langsam abfällt. Das ist für einen Spektrum-Analyser natürlich unbrauchbar. Die Ursache ist, dass das periodisch fortgesetzte Signal (Periodenlänge N) an den Rändern einen Sprung enthält. Das kann man gut am Amplituden/Zeit-Diagramm zweier Perioden in **Bild 4** erkennen: Bei Sample N springt das Signal von +1 V nach -1 V. Dieser Sprung generiert spektral hohe Frequenzkomponenten jenseits der Signalfrequenz. Um diese Auswirkungen zu vermeiden, wendet man die Fenstertechnik an, was im Prinzip einer überlagerten Hüllkurve entspricht. Dabei multipliziert man das Eingangssignal mit einer Fensterfunktion, wodurch es an den Fenstergrenzen sanft auf die Amplitude 0 reduziert wird, ansonsten das Signal aber nicht viel ändert. In **Bild 5** ist die verwendete Fensterfunktion abgebildet.
Bild 6 zeigt zwei Perioden des mit der Fensterfunktion multipliziertem Signals. Man kann schön erkennen, wie das Signal amplitudenmoduliert wird, sodass es an den Periodengrenzen (0, N, 2 N…) nicht springt, sondern stetig verläuft. Es sind verschiedene Fensterfunktionen gebräuchlich, die jeweils verschiedene Vor- und Nachteile haben. Wie das verwendete Blackman-Fenster (3-Term) berechnet wird, zeigt das folgende C-Listing:

```
double windowFun(int k) {
  double alpha=0.16;
  double a0=(1-alpha)/2;
  double a1=0.5;
  double a2=alpha/2;
  return a0-a1*cos(2*PI*k/(N-1))+a2*cos(4*PI*k/(N-1));
}
```

Die Fensterfunktion ist aus zwei Cosinus-Funktionen zusammengesetzt. Nach der Anwendung des Fensters ergibt die FFT das Spektrum von **Bild 7**. Das spektrale Maximum ist um $m = 5{,}5$ herum vier Bins breit und fällt dann deutlich ab. Bei größeren Werten für N kann man die spektrale Verbreiterung gut in Kauf nehmen und erhält gegebenenfalls gute, linienartige Spektren. Man muss allerdings berücksichtigen, dass die Fensterfunktion die Signalamplitude reduziert. Als Fensterfaktor kann man wie im folgenden Listing näherungsweise den arithmetischen Mittelwert der Fensterfunktionswerte nehmen.

```
float getWindowFactor() {
  float mean=0;
  for (k=0; k < N ; k++) {
    mean += abs(windowFun(k));
  }
 return mean/N ;
}
```

Zwecks Kompensation der Amplitudenreduktion dividiert man das Spektrum nach der FFT durch den Fensterfaktor. Bei Verwendung einer Fensterfunktion gilt die Parsevalsche Gleichung nicht mehr.

Analyser-Optionen

Der Spektrum-Analyser bietet mehrere konfigurierbare Möglichkeiten. Im einfachsten Fall tastet der µC das Signal ab, berechnet das Spektrum, gibt es aus und wiederholt dies endlos. Die Samplerate f_s entspricht der Abtastfrequenz. Sie legt gleichzeitig den Frequenzbereich des Spektrums fest, denn dieser reicht prinzipiell von 0 bis $f_s/2$. Bei 500 kS/s kann man also Frequenzanteile bis 250 kHz analysieren. Die Signal-Länge N legt fest, wie viele Samples für das Spektrum verwendet werden, und definiert damit die Auflösung der FFT. Zwei benachbarte (als Bins bezeichnete) Spektrallinien haben den Abstand $\Delta = f_s / N$. Bei 500 kS/s und $N = 1.024$ ergibt sich die Auflösung $\Delta = 500\ kHz / 1.024 \approx 488\ Hz$. Das ist ausreichend, um Rundfunk-HF-Signale aufzulösen.

Wenn man Signale mit Rauschanteilen analysiert, sind die einzelnen Spektren oft sehr zerklüftet. Das lässt sich verbessern, indem über aufeinanderfolgende Spektren gemittelt wird. Dieses Vorgehen wird durch die Option „MEAN“ der Analyser-Software aktiviert, die übrigens kostenlos von der Elektor-Webseite zu diesem Artikel unter [2] heruntergeladen werden kann. Am Spektrum des Zeitzeichensenders DCF77 kann man sehen, wie gut sich die Mittelung auswirkt. Es wird hierbei mit 250 kHz gesampelt, und N beträgt 16.384. **Bild 8** zeigt das resultierende Spektrum im Bereich von ±5 kHz um die Mittenfrequenz von 77,5 kHz.

Die weiße dünne Linie zeigt ein einzelnes Spektrum, das deutlich verrauscht ist. Durch die Mittelung (dicke grüne Kurve) kann man gut erkennen, dass DCF77 ein Rauschspektrum hat. Dies liegt daran, dass DCF77 die Daten neben der Amplitudenmodulation (sehr schmalbandig) mit pseudozufälligem Rauschen bei einer Bitrate von $77{,}5\ kHz / 120 \approx 645{,}8\ Hz$ sendet. Die Bandbreite von ±645,8 Hz wird durch die beiden senkrechten Linien dargestellt.

Als nächstes wird das Spektrum des Senders EFR (129,1 kHz) analysiert. EFR sendet RTTY-Daten mit einem Shift von ±170 Hz und einer Datenrate von 200 Bd. Im Spektrum sollten also zwei Spektrallinien bei 128,93 kHz und 129,27 kHz zu sehen sein. Weil aber immer nur kurze Nachrichten mit großen Abständen gesendet werden, kommt die Frequenz bei +170 Hz nur selten vor und ist daher schwer zu entdecken. Hierfür bietet die Software die Funktion *MAXHOLD*, bei der das Maximum der Spektren über viele Spektren gebildet wird. Auf diese Weise zeigen sich auch die kurz auftretenden Spektralanteile bei 129,27 kHz (siehe **Bild 9**).

Die dünne weiße Linie ist ein Einzelspektrum. Da gerade keine Nachricht gesendet wird, zeigt sich nur der Peak bei 128,93 kHz. Das MAXHOLD-Spektrum entspricht der dickeren grünen Kurve.

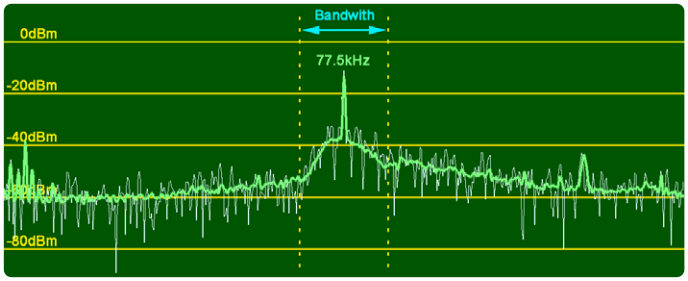


Bild 8. Das „normale“ Spektrum des DCF77 (weiß) und das gemittelte Spektrum (grün).

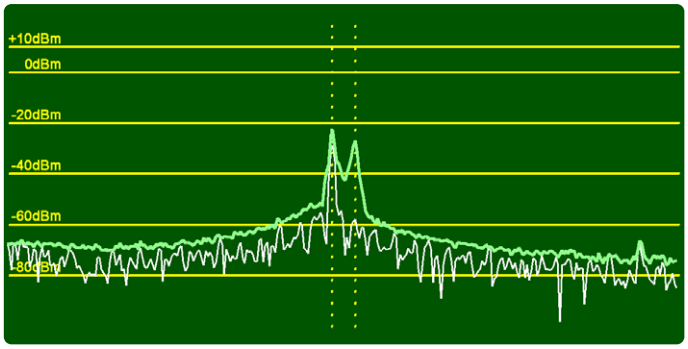


Bild 9. Spektrum des DCF49 (EFR bei 129,1 kHz) mit MAXHOLD-Funktion.

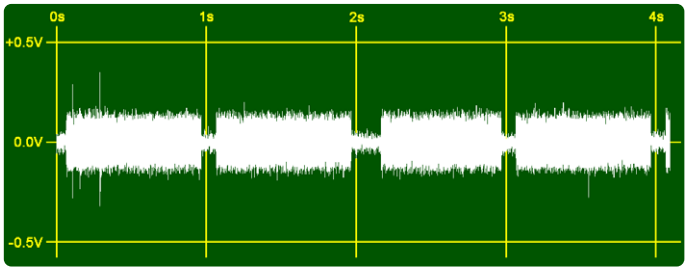


Bild 10. Amplitude über der Zeit des DCF77-Signals.

Die beiden Peaks bei 129,1 kHz ±170 Hz (bei den gepunkteten senkrechten Linien) sind deutlich zu sehen. Um ein so fein aufgelöstes Spektrum zu erhalten, sind mit $N = 16.384$ recht viele Samples erforderlich. Die Samplerate ist 500 kHz und die Auflösung entsprechend $\Delta = 500\ kHz / 16.384 \approx 30{,}5\ Hz$. Die beiden Spektrallinien sind also gerade mal 340 Hz / 30,5 Hz ≈ 11 Frequenz-Abtastpunkte voneinander entfernt.

Zeitbereich (Oszilloskop-Funktion)

Selbstverständlich ist auch das Signal im Zeitbereich informativ, sei es nur um zu kontrollieren, wie weit der ADC ausgesteuert wird. Auch das kann die Software. Es wird dabei einfach das gesampelte Signal angezeigt. **Bild 10** zeigt eine Art Oszillogramm des DCF77-Signals bei 77,5 kHz.

Da hier insbesondere die sekündlichen Trägerabsenkungen interessant sind, empfiehlt sich eine niedrigere Abtastrate und eine höhere Anzahl an Samples zu wählen. Für die Darstellung in Bild 10 wurde $f_s = 4\ kHz$ und $N = 16.384$ gewählt. Das Zeitfenster hat damit eine Größe von $T_s = N/f_s \approx 4\ s$, was zu vier Trägerabsenkungen in Bild 10 führt. Das 77,5-kHz-Signal wird dabei stark unterabgetastet, was aber kein Problem darstellt, da hier ja nur die Amplitude relevant ist.

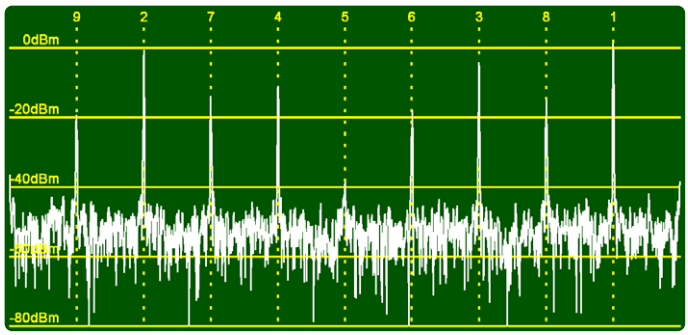


*Bild 11. Unterabtastung eines 225-kHz Rechtecksignals mit f_s = 500 kHz. Es zeigt eine Reihe von Spektrallinien bei f_k = 225 kHz * k mit k = 1, 2, 3…*

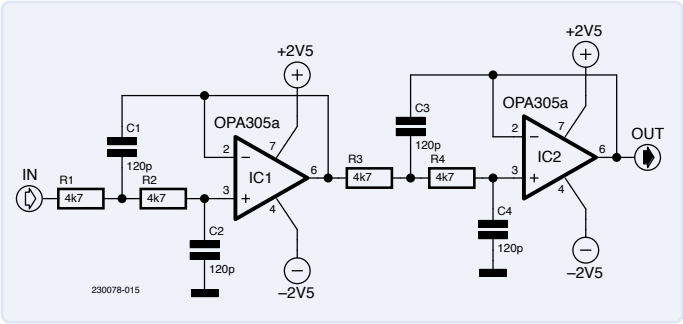


Bild 12. Anti-Aliasing Filter in Form von zwei Sallen-Key-Filtern 2. Ordnung mit einer Grenzfrequenz von 280 kHz.

Abtast-Theorem

Laut Abtast-Theorem dürfen die zu analysierenden Signale nur Frequenzen bis maximal der halben Abtastrate enthalten. Entweder respektiert man das, oder man baut ein entsprechendes Tiefpassfilter vor den ADC. Bis hierher wurde noch kein geeignetes Filter vorgeschlagen, da dies immer vom konkreten Einsatz abhängt. Außerdem ist es auch interessant, die Verhältnisse bei der Analyse höherer Frequenzen zu studieren. Wenn man Signale mit unendlich kurzen (Dirac-)Impulsen abtastet, entsteht ein periodisches Spektrum mit Periodenweite = Abtastrate. Die im Signal enthaltenen Spektrallinien werden also periodisch mit der Weite f_s wiederholt. Dementsprechend tauchen auch Frequenzen größer als f_s im Spektrum auf. Sie werden quasi ins Grundintervall heruntergefaltet. Ist die Abtastweite länger, werden höhere Frequenzen dabei stärker gedämpft. Die Zusammenhänge werden am folgenden Beispiel veranschaulicht: Die Samplerate f_s beträgt 500 kHz. Analysiert wird ein Rechtecksignal mit einer Frequenz von 225 kHz und einem Duty Cycle (Tastgrad) von 10 %.

Im Spektrum von **Bild 11** erscheint die Grundwelle bei 225 kHz am weitesten rechts und hat den höchsten Pegel. Die erste Oberwelle (k = 2) hat die Frequenz $2 \times 225\ kHz = 450\ kHz$, was bereits jenseits von $f_s / 2$ liegt. Aufgrund der Symmetrie erscheint sie daher bei $f = 500\ kHz - 450\ kHz = 50\ kHz$ im Spektrum unter der mit „2“ beschrifteten gepunkteten vertikalen Linie mit noch beträchtlicher Amplitude. Die Frequenz für q entspricht $225\ kHz \times 9 = 2.025\ kHz$. Da dies $4 \times 500\ kHz + 25\ kHz$ entspricht, erscheint dieser Peak bei 25 kHz ganz links im Spektrum unter der mit „9“ beschrifteten Cursorlinie. Die Cursorlinien-Indizes k = 1…9 entsprechen der theoretisch erwarteten Position der Oberwellen des analysierten Signals mit den Frequenzen $k \times 225\ kHz$ – die sogenannten Alias-Frequenzen. Obwohl die Frequenzen bis über 2 MHz gehen, erscheinen zehn Peaks im bis zu 250 kHz reichenden Fenster. Daraus folgt, dass der ADC durchaus höhere Frequenzen als $f_s / 2$ erfassen kann. Diesen Effekt nutzt man bei der sogenannten Unterabtastung. Man kann das Signal vor Abtastung durch einen Bandpass mit einem Durchlassbereich von $f_s / 2$ so filtern, dass nur die relevanten Bereiche des Spektrums erfasst werden. Hierzu würde man das Setting um eine schnelle Sample&Hold-Funktion erweitern, um die hohen Frequenzen noch mit dem ADC verarbeiten zu können. Würde dieser Effekt aber stören, filtert man das Signal mit einem Tiefpass (Anti-Aliasing-Filter), um höherfrequente Signalanteile vor der Abtastung zu unterdrücken.

Anti-Aliasing Filter

Nachfolgend wird die Wirkung eines Anti-Aliasing-Tiefpasses demonstriert. Dabei wird ein symmetrisches Rechtecksignal mit 225 kHz und einem Tastverhältnis von 50 % durch den Tiefpass von **Bild 12** vor dem ADC gefiltert. Das Filter wurde mit LTspice [3] simuliert und mit Opamps des Typs OPA2350 aufgebaut.

Man erkennt im Spektrum des gefilterten Signals von **Bild 13** deutlich, dass die Oberwellen stark gedämpft sind und nur die 225-kHz-Grundwelle deutlich über -40 dBm herausragt. Allerdings dämpft unser Filter schon bei 225 kHz ganz ordentlich; ein Filter mit steileren Flanken wäre aber besser.

Rauschen und ENOB

Bisher wurden nur Signale untersucht, die sich aus einzelnen Sinusschwingungen verschiedener Frequenzen und Amplituden zusammensetzten. In der Praxis gibt es aber auch Rauschsignale unterschiedlicher Bandbreite. Deren nachrichtentechnische Behandlung ist nicht gerade trivial, weswegen nur einfache Fälle beschrieben werden. Als Beispiel werden für binäres Rauschen mit der Rauschfrequenz f_n die Rauschwerte n_k mit der Amplitude A generiert. Dabei wird jeweils gewürfelt, ob für den nächsten Wert $s_k = +A$ oder $s_k = -A$ gelten soll. Zunächst gilt $f_n = 1\ MHz$, damit die Werte schneller generiert werden, als sie der ADC abtasten kann. Damit sind die Abtastwerte unabhängig voneinander und der Wert ist jeweils ±A. Das Rauschsignal ist folglich nahezu „weiß“, weshalb in seinem Spektrum alle Frequenzen gleich stark vorkommen. Das Signal wird durch Software auf dem Raspberry Pi Pico generiert. Damit diese Aufgabe parallel zur Spektrum-Analyser-Software laufen kann, nutzt sie den zweiten Core der RP2040-CPU. Sie ist mit Hilfe einer PIO-State-Machine implementiert, wodurch bis zu 125 Mb/s erzeugt werden können.

In der Regel arbeitet man mit dem Effektivwert der Rauschspannung. Da das Signal nicht periodisch ist, kann man formal nicht die Formel für *RMS* benutzen. Doch für große Samplezahlen M erhält man näherungsweise den Effektivwert mit:

$$RMS_n \approx \sqrt{\frac{1}{M}\sum_{k=0}^{M-1} n_k^2}$$

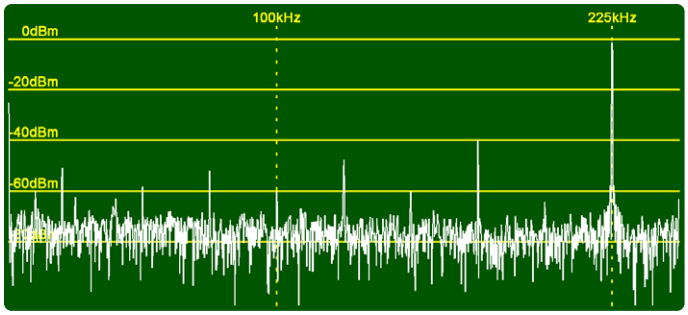


Bild 13. Spektrum des Rechtecksignals nach Anti-Aliasing-Filterung.

Für das binäre Rauschen lässt sich der Effektivwert also doch einfach berechnen. Da jedes Sample den Absolutbetrag A hat, ergibt sich folglich die simple Gleichung $RMS_n = A$. Im Testaufbau gilt $A = 750\ mV$. **Bild 14** zeigt ein Beispiel dieses Signals, und in **Bild 15** ist das zugehörige Spektrum dargestellt. Die rote dünne Linie ist ein einzelnes Spektrum. Sie ist sehr zerklüftet, weil sie nur eine einzelne Musterfunktion des Spektrums ist. Die dicke grüne Linie ist das Ergebnis einer Mittelwertbildung über viele Spektren (MEAN-Funktion) und macht eher deutlich, dass es sich hier um ein weißes Rauschen mit einem bestimmten Pegel handelt.
Wenn man sich Rauschspektren mit verschiedenen Signallängen N anschaut, erkennt man die Abhängigkeit des angezeigten Pegels auch von N. Es gibt den folgenden Zusammenhang zwischen Effektivwert, angezeigtem Pegel und N: Bei großem N gilt $RMS_s \approx RMS_n$. Aufgrund der Parsevalschen Gleichung (siehe oben) gilt außerdem

$$RMS_W = 2\ RMS_S\ / \sqrt{N}$$

Nun wird sich die Energie in

$$RMS_W = \sqrt{\frac{1}{N}\sum_{k=0}^{N-1}|W_k|^2}$$

gleichmäßig auf w_k verteilen. Dabei wird vereinfachend davon ausgegangen, dass alle w_k gleich groß und gleich w sind. Fügt man das alles zusammen, erhält man: $w^2 = 4\ RMS^2_n / N$. Dabei ist w genau die gesuchte Stärke des Rauschsignals im Spektrum. In Bild 15 ist dieser Pegel als dicke violette Linie zu sehen. Es ist tatsächlich eine recht gute Vorhersage des realen Rauschpegels. Es wird auch klar, warum dieser Pegel mit größerem N sinkt: Die Energie des Rauschen wird nämlich auf mehr Spektrallinien verteilt.
Im nächsten Beispiel wird die Bitrate der Rauschquelle auf $f_n = 25\ kHz$ reduziert. Daher sind immer 20 aufeinanderfolgende Abtastwerte gleich, was einer Art Tiefpassfilterung entspricht. Das resultierende Spektrum ist in **Bild 16** dargestellt.
Die dünne rote Linie zeigt wieder eine einzelne Musterfunktion. Das gemittelte Spektrum ist grün dargestellt. Die erste Nullstelle des Spektrums liegt bei 25 kHz. Das Spektrum ist proportional zu $H(f) = si(\pi\ f / f_N)$; dabei ist $si(x) = sin(x) / x$ die sogenannte Spaltfunktion. Vereinfachend kann man den Pegel um 0 herum wie folgt berechnen: Die Bandbreite ist circa $f_{BW} = 12{,}5\ kHz$, was $k_{BW} = N\ f_{BW} / f_s = 25$ Spektrallinien entspricht.
Wenn das Spektrum in der Bandbreite konstant w entspricht, ergibt sich die Gleichung $RMS^2_w = 2\ k_{BW}\ w^2 / N$ für die Effektivwerte, indem die Spektrallinien innerhalb der Bandbreite summiert werden. Der Faktor 2 ist notwendig, da die FFT ein symmetrisches Spektrum mit den Spektrallinien N - k_{BW} bis N liefert. Mit der Parsevalschen Gleichung ergibt sich damit die gesuchte spektrale Stärke w zu $w^2 = 2\ RMS^2_n / k_{BW}$. Sie ist in Bild 16 mit der dicken violetten Linie eingezeichnet und trifft das aktuelle Spektrum ganz gut.
Das nächste Beispiel ist etwas praxisnäher: Der ADC bekommt nun das Signal eines Sinusgenerators mit der Amplitude von 3 V_{SS} (etwa 14 dBm) angeboten. Die Signalfrequenz von $fs = 48{,}828\ kHz$ entspricht genau der 100. Spektrallinie. **Bild 17** zeigt das resultierende Spektrum.

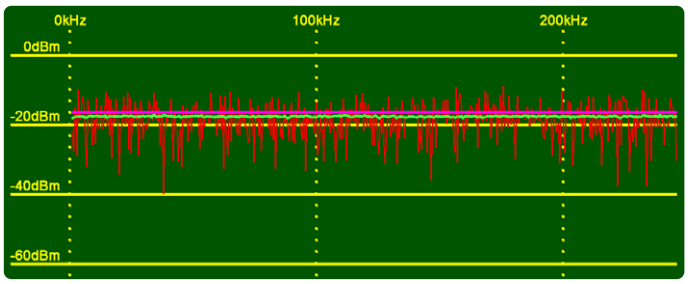


Bild 14. Binäres Rauschen als Zeitsignal mit einer Amplitude von ±750 mV.

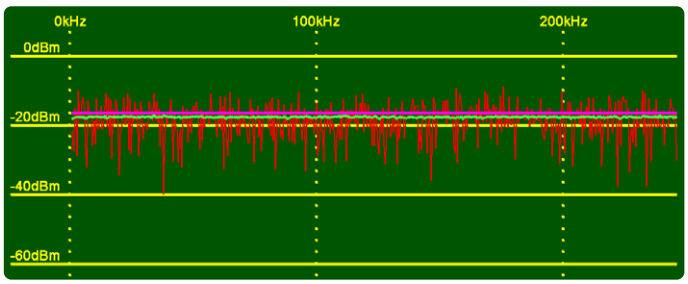


Bild 15. Spektrum des binären Rauschens von Bild 14.

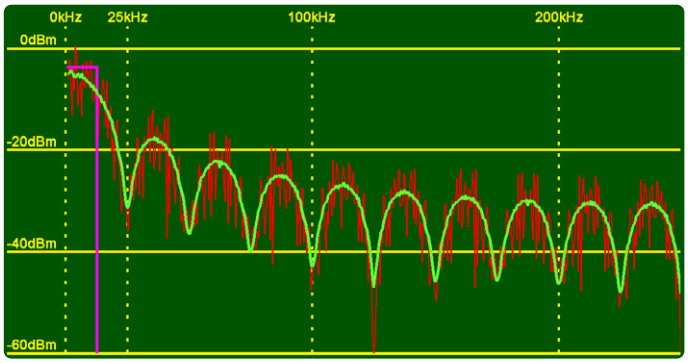


Bild 16. Spektrum des Rauschens bei einer Bitrate von 25 kHz.

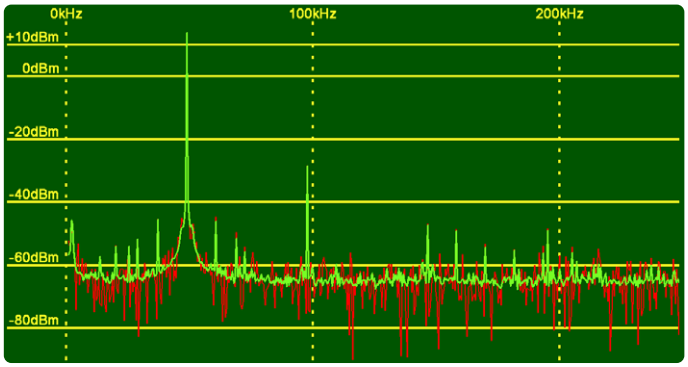


Bild 17. Spektrum eines Sinussignals mit 48,828 kHz bei einer Abtastrate von 500 kHz.

Die Spektrallinie bei f_s ist gut zu erkennen und hat auch die richtige Größe. Bei 2 f_s sieht man die erste, vom Signalgenerator verursachte Oberwelle mit einem Pegel von rund 45 dB unterhalb des Signals. Zu erkennen sind außerdem zahlreiche kleine Spektrallinien, da die Bauteile nicht ideal sind und zum Beispiel geringe Nichtlinearitäten aufweisen. Interessant ist auch der Rauschpegel. Am gemittelten Spektrum (grün) kann man gut erkennen, dass dieser bei etwa -65 dBm liegt. Mit der oben angegebenen Formel kann man den Pegel in die effektive Rauschspannung von etwa 2 mV_{RMS} (etwa 5 mV_{SS})

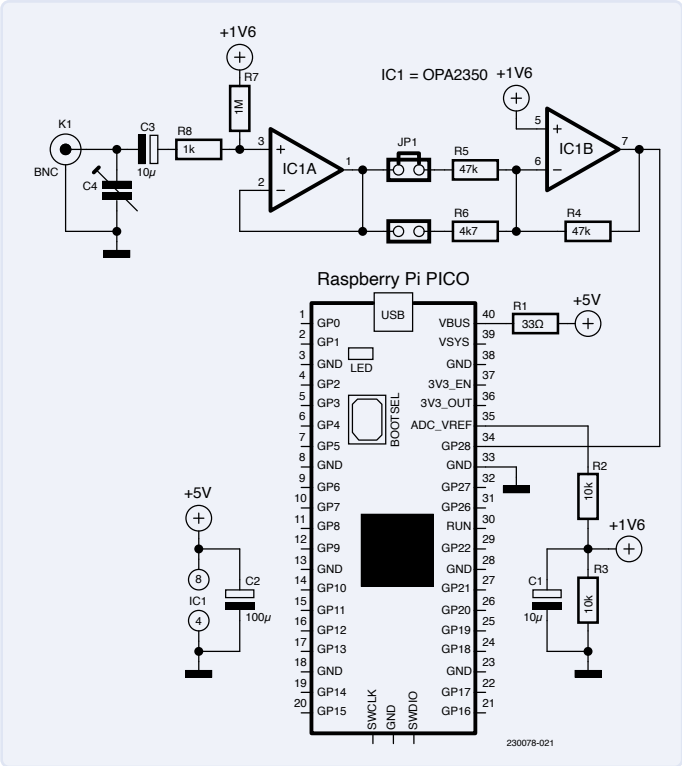


Bild 18. Vorverstärker mit einem Eingangswiderstand von 1 MΩ. Alternative Opamptypen sind AD823 und MCP602.

umrechnen. Die Auflösung des ADC beträgt *3,3 V / 4.096 = 0,8 mV*. Das Rauschen überdeckt also etwa sechs Quantisierungsstufen. Durch das Rauschen gehen so mehr als 2 Bit verloren. Dennoch heben sich spektrale Peaks mit -55 dBm noch gut vom Rauschen ab. Die Dynamik beträgt also mindestens *14 dBm - (-55 dBm) = 69 dB* – ein Faktor von ≈2.800. Die reale Auflösung, die ENOB (**E**ffective **N**umber **O**f **Bit**s) des ADC erreicht dank Mittelung mehr als den Wert 11, was gar nicht schlecht ist. Bei den Rauschexperimenten wurde übrigens ohne Fensterung gearbeitet, damit die Pegel genau eingehalten werden und die Parsevalsche Gleichung anwendbar ist.

Vorverstärker mit 1-MΩ-Eingang

Von Oszilloskopen ist man gewöhnt, dass sie einen Eingangswiderstand von 1 MΩ haben. Dadurch belasten sie die Messobjekte im Normalfall nicht und man kann die üblichen 1:1 und 10:1 Tastköpfe verwenden. Diesen Komfort sollte ein Spektrum-Analyser ebenfalls bieten, und daher wurde der Vorverstärker von **Bild 18** entworfen. Der Opamp IC1A fungiert als Impedanzwandler. Die Eingangsimpedanz wird durch R7 bestimmt. IC1B abeitet als Vorverstärker mit einer durch JP1 wählbaren Verstärkung von 1 oder von 10. IC1A muss einen niedrigen Bias-Strom haben, sonst würde dieser Strom an R7 einen zu großen Spannungsabfall hervorrufen und dadurch sich die Nulllinie des Signals aus der Mitte des ADC-Erfassungsbereichs verschieben. Der Typ AD8042 würde beispielsweise mit seinem Bias-Strom von I_B = *1,2 µA* eine Offset-Verschiebung um *1,2 µA × 1 MΩ = 1,2 V* verursachen. Geeignete Opamp-Typen sind in Bild 18 angegeben. Mit C4 kann man gegebenenfalls eine Anpassung an einen Tastkopf realisieren. In **Bild 19** sieht man den auf einer Lochrasterplatine aufgebauten Prototypen nebst Pico-Board und Display.

ADC-Takt und Samplerate

Laut Datenblatt muss der ADC-Takt 48 MHz betragen. Dieser Takt wird normalerweise von der USB-PLL generiert. Der 16-Bit-Vorteiler

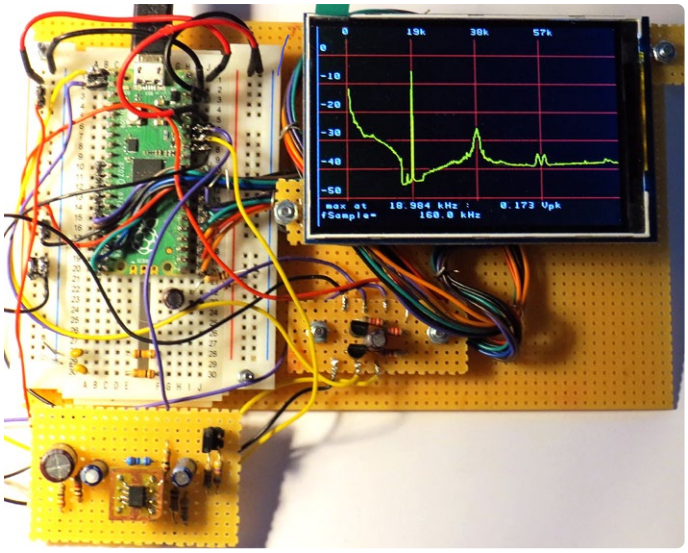


Bild 19. Analyser aus einem Pico-Board mit LCD und einem 1-MΩ-Vorverstärker zeigt ein UKW-Multiplex-Signal.

wird dabei nicht aktiviert (Teilung durch 1). Der ADC wird im *Free-Running Mode* betrieben, in dem er fortlaufend Werte erfasst und ausgibt. Das Timing wird durch einen ADC-Zähler erzeugt, der vom ADC-Takt angesteuert wird. Der ADC-Zähler läuft beim Wert *ADCclkDiv* über und triggert dann den ADC neu. Da eine Konversion 96 Takte benötigt, muss *ADCclkDiv* ≥ *96* sein. Man erreicht beim kleinsten Wert von *ADCclkDiv = 96* die höchste Samplefrequenz von f_s = *48 MHz / 96 = 500 kHz*. Der ADC-Zähler ist ein fraktioneller 16.8-Zähler: Der Integer-Teil ist 16 Bit breit und man kann fraktionell mit 8 Bit teilen, um krumme Sampleraten möglichst genau zu erzeugen. Die langsamste Samplerate ist f_s = *48 MHz / 65.536 = 732 Hz*.

Im nächsten Experiment wird der ADC für eine Samplerate von 1 MHz weit außerhalb seiner Spezifikation betrieben. Hierzu muss der ADC-Takt 96 MHz betragen. Der USB-PLL-Takt mit seinen 48 MHz ist nicht ausreichend. Der ADC wird daher von der PLL für den Systemtakt *SYS-PLL* getaktet. Der hiermit für den Prozessor erzeugte Takt liegt normalerweise bei 125 MHz. Wird er aber auf 96 MHz eingestellt, kann man damit direkt den ADC takten und eine Samplerate von 1 MHz ermöglichen. Leider läuft dann der Prozessor etwa 23 % langsamer. Das ist aber kein Problem, da die CPU durch die Software sowieso nicht ausgelastet wird.

Als Testsignal dient ein 400-kHz-Sinussignal mit einer Amplitude von 2 V_{SS}. **Bild 20** zeigt das resultierende Spektrum. Das Nutzsignal wird mit dem richtigen Pegel an der richtigen Stelle dargestellt. Die erste Oberwelle bei 800 kHz wird allerdings auf *1 MHz -800 kHz = 200 kHz* heruntergefaltet. Der Rauschpegel von -65 dBm ist nicht schlechter als bei einer Samplerate von 500 kHz. Es scheint also keine schwerwiegenden Einwände gegen die Taktung des ADC mit 96 MHz zu geben.

LC-Display

Da in Spektren oft auch feine Details interessieren, wurde dank seiner relativ hohen Auflösung von 320 × 480 Pixel das Arduino-Shield MAR3502 gewählt. Es wird, wie in **Bild 21** gezeigt, parallel an das Pico-Board angeschlossen. Damit lässt sich ein Spektrum-Analyser als eigenständiges Gerät aufbauen – allerdings fehlen dazu noch die Bedienelemente.

In Bild 20 sieht man das Spektrum eines UKW-Multiplex-Audiosignals auf dem Display. Man kann gut das Monosignal (L+R) im

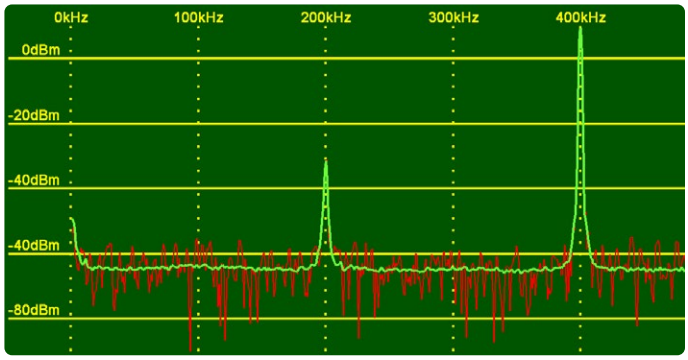


Bild 20. Spektrum eines 400-kHz-Sinussignals bei einer Abtastrate von 1 MHz.

Bereich unter 17 kHz erkennen. Bei 19 kHz erscheint die Spektrallinie das Stereo-Pilot-Tones. Bei 38 kHz zeigt sich das Spektrum des Stereo-Differenzsignals (L - R). Bei 57 kHz wird das Spektrum des RDS-Signals sichtbar, das aus zwei Seitenbändern um 57 kHz herum besteht.

12-bit-ADC mit bis zu 50 MS/s

Für niedrigere Frequenzen reicht die Abtastrate von 500 kHz des im RP2040 integrierten ADC zwar aus, doch gibt es auch im HF-Bereich interessante Spektren. Hierfür wird das Pico-Board mit dem Chip ADS807E um einen schnellen, externen ADC erweitert. Mit diesem 12-bit-ADC sind bis zu 53 MS/s möglich. Für die Spektren braucht es dann schon ordentlich Rechenleistung, und demnach einige Tricks, um diese hohe Samplerate auch praktisch zu erreichen. Glücklicherweise verfügt der ADS807E über eine integrierte Referenzspannung. Der ADC wird wie in **Bild 22** an das Pico Board angeschlossen.

Durch den parallelen Anschluss werden viele GPIO-Pins belegt, so dass man leider das LCD nicht gleichzeitig anschließen kann. Aus diesem Grund bekommt der ADS807 ein zweites Pico-Board spendiert. Bei den ersten Experimenten damit wurden die Daten wie Anfangs mit dem PC verarbeitet und anzeigt. An K2 können verschiedene analoge Frontends angeschlossen werden, die nachfolgend vorgestellt werden. **Bild 23** zeigt den Testaufbau eines Pico-Boards mit ADS807 und 50-Ω-Trafo-Frontend.

Übrigens: Wem der ADS807 zu teuer ist oder wer dessen 12-Bit-Auflösung nicht benötigt, der kann alternativ einen preiswerten 8-Bit-ADC wie etwa den ADS830E einsetzen. Er bietet Sampleraten bis zu 60 MS/s und benötigt lediglich eine angepasste Sample-Routine.

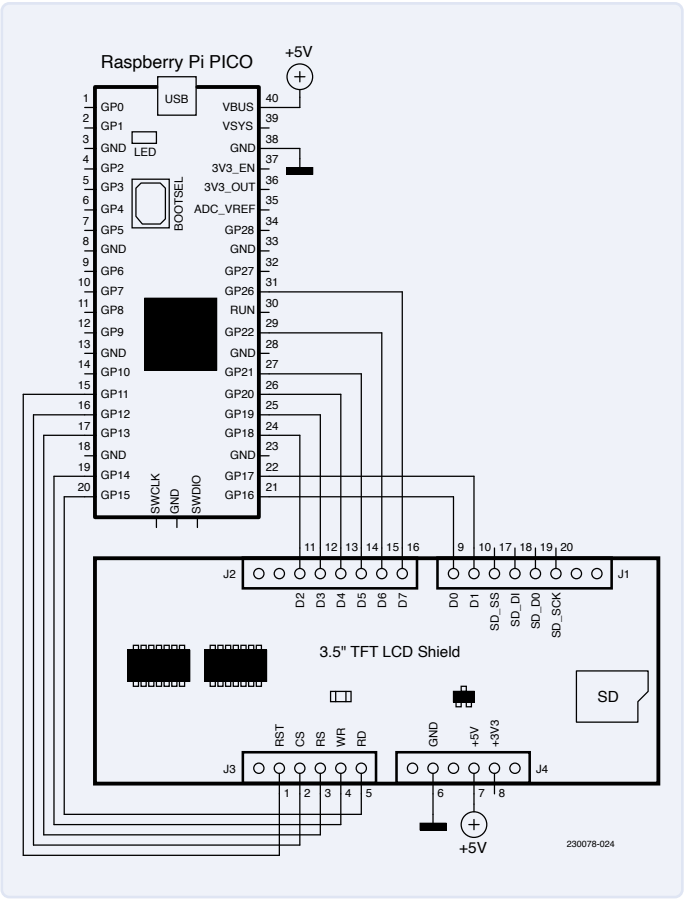


Bild 21. Paralleler Anschluss eines 3,5"-LCDs mit 480 x 320 Pixel an das Pico-Board.

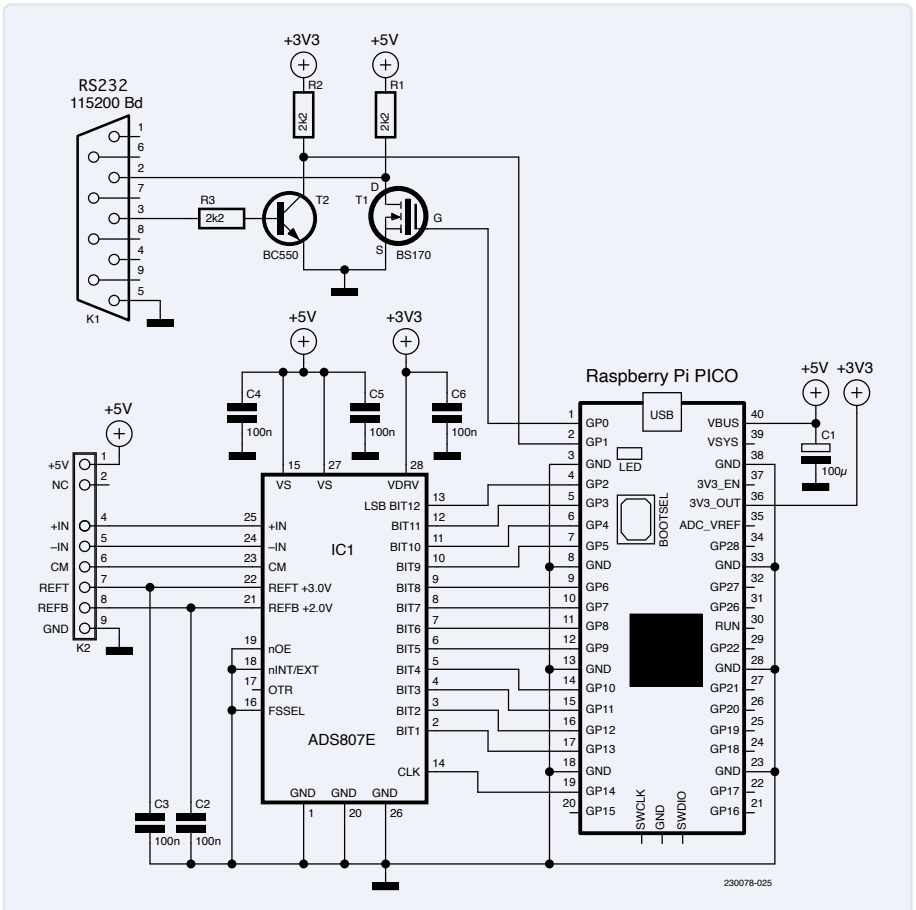


Bild 22. Anschluss des externen ADC ADS807 an das Pico-Board.

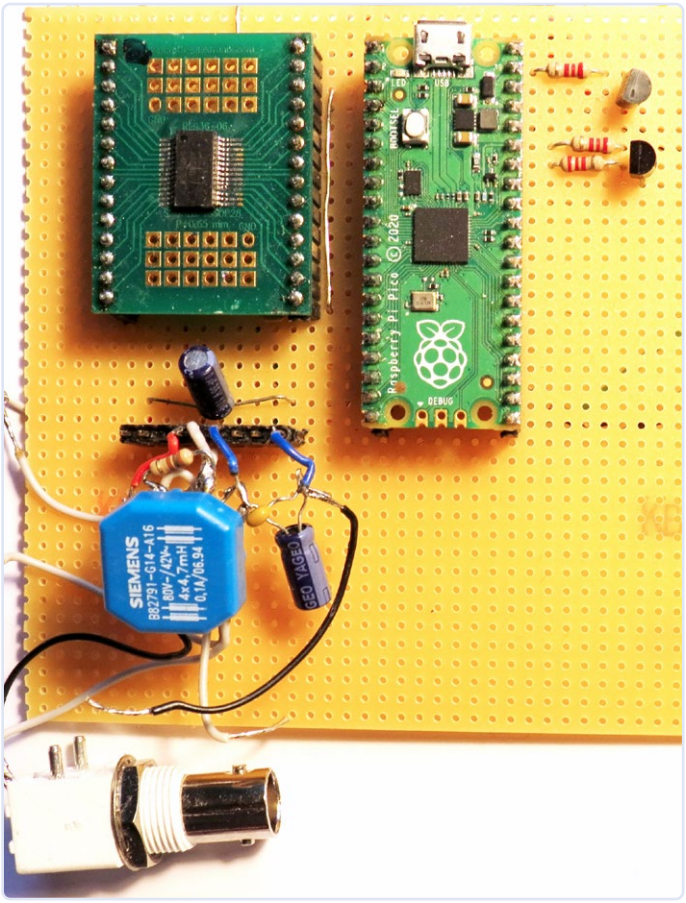


Bild 23. Zusätzliches Pico-Board mit ADS807 und 50-Ω-Trafo-Frontend.

Schnelles Abtasten per PIO, FIFO und DMA

Die RP2040-CPU wird normalerweise mit 125 MHz getaktet. Das folgende Listing zeigt die minimale Schleife, um damit *N* Datenwerte zu erfassen:

```
for (k = 0; k < N; k++) {
  sampleBuffer[k] = gpio_get_all();
  gpio_put(ADCclock,0);
  gpio_put(ADCclock,1);
}
```

Laut meiner Messung schafft die RP2040-CPU maximal 15 MS/s – noch ganz schön weit von den 53 MS/s eines ADS807 entfernt. Allerdings verfügt die CPU als besonderes Feature über eine PIO-Einheit (**P**eripheral **I**nput **O**utput), die aus acht programmierbaren State-Machines besteht. Diese Zustands-Automaten kann man mit dem CPU-Takt ansteuern und mit einfachen Befehlen programmieren. Nachfolgend werden einige Aspekte der Sampling-Anwendung beleuchtet. Die vollständigen Details sind ziemlich komplex, weshalb hierfür auf die RP2040-Dokumentation [4] verwiesen wird. Das PIO-Programm besteht nur aus zwei Befehlen:

```
.wrap_target
   in pins,12     side 0b0
   nop            side 0b1
.wrap
```

Die Schlüsselwörter `.wrap_target` und `.wrap` bewirken, dass die beiden Befehle zwischen ihnen endlos hintereinander ausgeführt werden. Dabei kostet die Schleife selbst keinen Overhead. Der Befehl

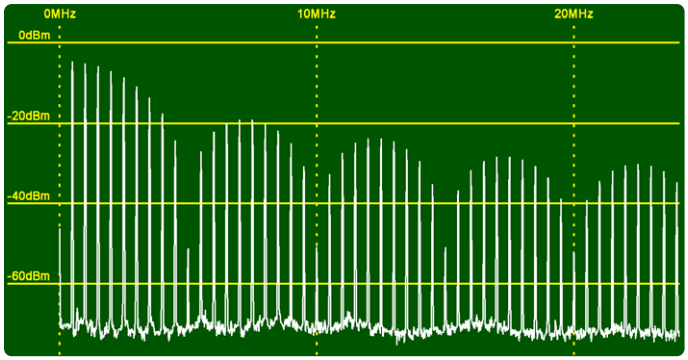


Bild 24. Spektrum eines 500–kHz-Rechtecksignals mit einem Tastverhältnis von 10 %.

`in` bewirkt, dass 12 Bit vom ADC in das ISR (**I**nput **S**hift **R**egister) bewegt werden. Der Befehl `nop` bewirkt bekanntlich nichts. Hinter den Befehlen wird die Option `side` benutzt. Damit kann man parallel zum eigentlichen PIO-Befehl auch noch GPIO-Pins beeinflussen. Auf diese Weise wird der ADC-Takt an GPIO14 erzeugt. Da jeder Befehl einen Zyklus benötigt, kann dieses Programm prinzipiell 125 / 2 MS/s verarbeiten. Das genügt für den ADS807. Neben dem eigentlichen Programm spielt die Konfiguration der State-Machine eine wichtige Rolle. In ihr wird zum Beispiel festgelegt, welche Pins per *side-set* beeinflusst werden. Außerdem wurde konfiguriert, dass nach je zwei Samples (wenn im ISR 24 bit angekommen sind) der ISR-Wert in ein Ausgabe-FIFO geschrieben wird. Die Daten werden via DMA (**D**irect **M**emory **A**ccess) schnell in den Speicher geschrieben. Das eigentliche Sampling läuft folglich vollkommen autark ab. Die CPU muss nach jedem DMA-Transfer lediglich die Werte aus dem Puffer holen.

Für den ADC ADS807 kann im Prinzip die gleiche Software verwendet werden wie beim integrierten ADC. Es muss lediglich die Sample-Routine angepasst werden. Um die maximale Samplerate von 53 MHz zu erreichen, kann man einen CPU-Takt von beispielsweise 106 MHz konfigurieren und den PIO-Vorteiler auf 1 stellen. Nun läuft das PIO-Programm mit 106 MHz. Da ein Sample jeweils zwei Takte benötigt, ergibt sich exakt die Samplerate von 53 MHz. Leider wird dabei die CPU gut 15 % langsamer getaktet als üblich. Für die hier vorliegenden Aufgaben reicht das aber problemlos aus. **Bild 24** zeigt das Spektrum eines Rechtecksignals mit einem Tastgrad von 10 % bei einer Abtastrate von 50 MHz. Die Oberwellen bis 25 MHz sind deutlich sichtbar. Aufgrund des Tastverhältnisses von 10 % fehlt jede 10. Oberwelle. Via Unterabtastung kann man mit dem ADS807 noch problemlos Signalanteile bis circa 200 MHz analysieren

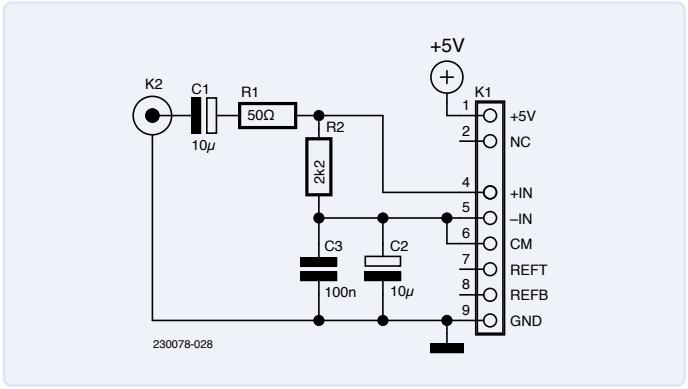


Bild 25. Single-ended Frontend.

Analoge Frontends

Die differentiellen Eingänge *+IN* und *-IN* des ADS807E haben auch bei höheren Frequenzen ein hohes Signal/Rauschverhältnis. Allerdings ist der Chip etwas schwieriger anzusteuern als ADCs mit unsymmetrischem Eingang. Nachfolgend werden mögliche Frontends beschrieben, die man an K2 in Bild 22 anschließen kann. Der Aussteuerungsbereich des ADS807 liegt bei 2 V_{SS}, woraus sich eine Auflösung von 2 V / 4.096 = 480 µV ergibt.

Frontend 1: Single-ended

Im einfachsten Fall ignoriert man den differentiellen Eingang, legt *-IN* auf CM (**C**ommon **M**ode) und speist das Signal einfach an *+IN* ein. Man arbeitet dann mit Wechselspannungskopplung, und der DC-Offset von 2,5 V wird von CM geliefert (**Bild 25**). Wer eine 50-Ω-Eingangsimpedanz haben möchte, kann einfach einen 50-Ω-Widerstand parallel zum Eingang K2 legen.

Frontend 2: Trafokopplung mit 50 Ω

Mit einem (HF-)Trafo mit drei eng gekoppelten Wicklungen mit gleicher Windungszahl lässt sich eine einfache Ansteuerung des differentiellen Eingangs erreichen. Beim Prototyp wurde eine 4,7-mH-Common-Mode-Drossel mit vier Wicklungen benutzt, von den nur drei verwendet werden (**Bild 26**). Die verwendete blaue Drossel von Siemens ist gut in Bild 23 zu sehen.
Je eine Wicklung speisen die Eingänge *+IN* und *-IN* mit entgegengesetzter Phase. Die Wicklungen besitzen jeweils einen Parallelwiderstand von 100 Ω. Die 1:1-Kopplung transformiert diese Widerstände in zwei parallelgeschaltete 100-Ω-Widerstände auf die Primärseite, wodurch sich eine Eingangsimpedanz von 50 Ω ergibt. Da der ADC dabei die doppelte Eingangsspannung sieht, ergibt sich eine Verstärkung mit dem Faktor 2. Der Eingang ist gleichzeitig galvanisch vom ADC getrennt, was Gleichtakt-Störungen vermeidet. Die Schaltung eignet sich gut für Messungen an 50-Ω-Systemen. Wenn man den Trafo L1 selbst wickeln will, sollte man die drei Wicklungen trifilar (quasi eine Wicklung mit drei Drähten) wickeln, um auch bei hohen Frequenzen eine gute Kopplung zu erreichen. Der Kern sollte eine hohe Permeabilität auch bei Frequenzen bis 25 MHz haben.

Frontend 3: 25-MHz-Vorverstärker mit 1 MΩ

Für bestimmte Messungen benötigt auch ein schneller Spektrum-Analyser einen hochohmigen Eingang. Damit lassen sich dann auch die üblichen 10:1-Oszilloskop-Tastköpfe verwenden. **Bild 27** zeigt die Schaltung eines passenden Vorverstärkers.
Der Opamp IC1A dient als Impedanzwandler mit dem durch R5 bedingten Eingangswiderstand von 1 MΩ. Der Typ OPA2350 ist nicht nur schnell, sondern bietet einen niedrigen Bias-Strom und geringes Eingangsrauschen. IC2A und IC2B dienen als bipolarer Treiber für den ADC. Da der unipolare Eingang in ein differentielles Signal gewandelt wird, entsteht eine zweifache Verstärkung. IC2 wurde bezüglich hoher Bandbreite selektiert, damit der Phasenunterschied zwischen *+IN* und *-IN* möglichst klein bleibt. R2 und R4 dienen der Stabilität dieser Opamps. IC1B puffert die Offset-Spannung von 2,5 V. **Bild 28** zeigt unter anderem den Aufbau des Frontends auf einer Lochrasterplatine.

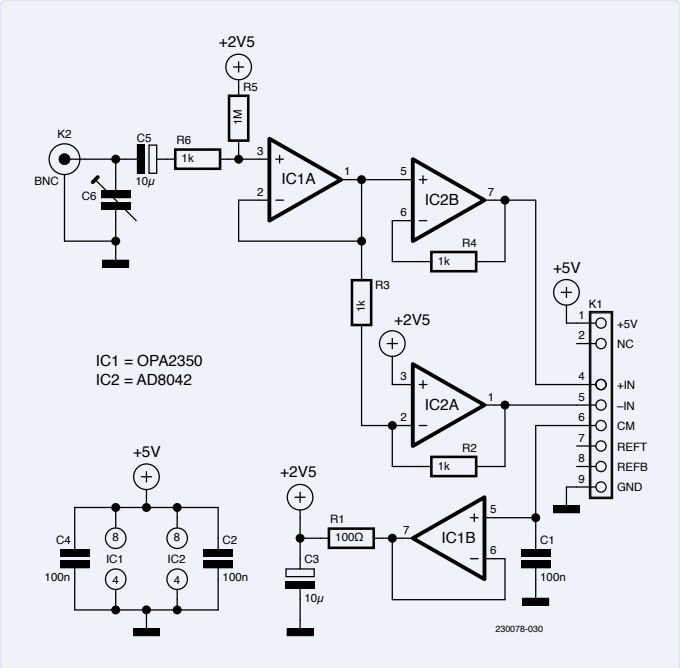


Bild 27. HF-Vorverstärker mit einer Eingangsimpedanz von 1 MΩ.

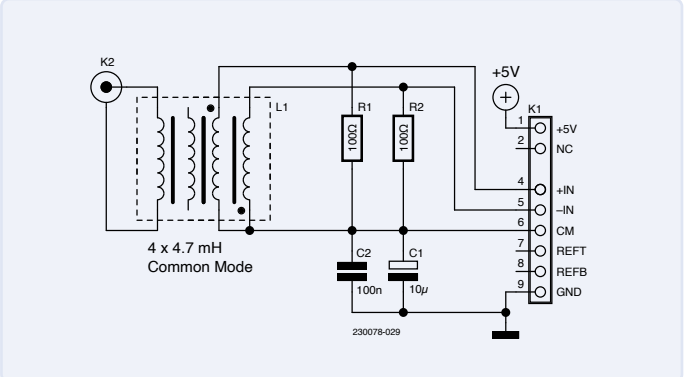


Bild 26. Trafogekoppeltes 50-Ω-Frontend.

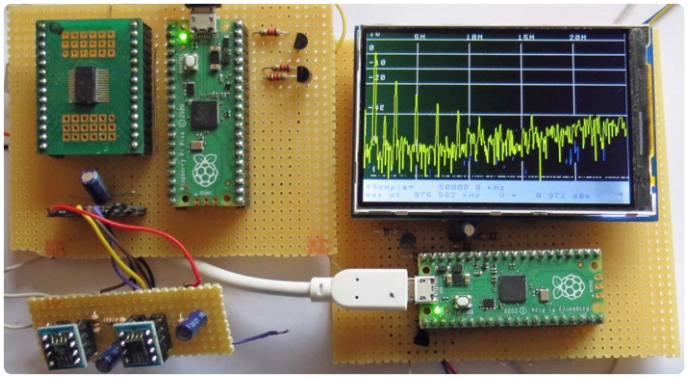


Bild 28. Stand-Alone-Spektrum-Analyser auf der Basis von zwei Pico-Boards.

Rauschen

Bestimmte Rauschquellen machen dem Schaltungsentwickler das Leben schwer. An R5 entsteht zum Beispiel eine thermische Rauschspannung von $U_{N,\,RMS} = \sqrt{(4\,k_B\,T\,B\,R)}$. Dabei ist k_B die Boltzmann-Konstante, T die absolute Temperatur (in Kelvin), B die Bandbreite (hier 25 MHz) und R der Wert von R5 (1 MΩ). Damit ergibt sich eine effektive Rauschspannung von 6,3 mV.
Das ist ein Vielfaches der ADC-Auflösung und dieser Rauschpegel macht sich kräftig bei breitbandigen, hochohmigen Messungen bemerkbar. Daraus ergibt sich auch, weshalb für HF ein niederohmiges System mit beispielsweise 50 Ω vorteilhafter ist. Der Eingangsstrom eines OPA2350 rauscht mit einer Rauschdichte von 4 fA/√Hz. Dadurch entsteht an R5 bei einer Bandbreite von 25 MHz eine vernachlässigbare effektive Rauschspannung von 20 µV. Für das Eingangsspannungsrauschen wird eine Dichte von 7 V/√Hz angegeben. Das ergibt bei 25 MHz eine ebenfalls vernachlässigbare Rauschspannung von 30 µV. Es wird aber deutlich, wie die Rauschparameter des Opamps das Design beeinflussen. Der AD8042 zum Beispiel weist ein Stromrauschen von 500 fA/√Hz auf und ist daher als Impedanzwandler am Eingang ungeeignet.

Grafisches Display

Wie schon erwähnt, kann man aufgrund zu weniger GPIO-Pins nicht gleichzeitig den externen ADC und das LCD anschließen. Für ein Stand-Alone-Gerät wird dieses Manko durch jeweils getrennte Pico-Boards für Datenerfassung mit dem externen ADC und der Anzeige per LCD kompensiert. Letzteres ergibt ein Grafik-Terminal, das verschiedene Grundfunktionen wie das Zeichnen von Linien und Rechtecken sowie die Textausgabe eigenständig ausführen kann. Auf dem Pico-Board mit dem ADS807 sind die Spektrum-Analyser-Funktionen realisiert. Zwecks Grafik-Ausgabe sendet dieser Teil entsprechende Befehle per serieller Schnittstelle an das Grafik-Terminal. Bei einem Preis von knapp 5 € für ein Pico-Board ist dieses Dual-Board-Design ökonomisch vertretbar. Bild 28 zeigt, wie diese Lösung mit zwei Pico-Boards aussehen kann. Das Display zeigt das Spektrum eines 1 MHz-Rechtecksignals bei einer Samplerate von 50 MS/s. Die Oberwellen werden mit steigender Ordnung schnell kleiner.

230078-02

Sie haben Fragen oder Kommentare?

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

Über den Autor

Martin Oßmann begann im Alter von zwölf Jahren Elektor zu lesen - und zu tüfteln, versteht sich. Nach dem Studium der Elektrotechnik und mehrjähriger Tätigkeit als Entwicklungsingenieur war er Professor am Fachbereich Elektrotechnik und Informationstechnik an der FH Aachen. Er ist nicht nur Autor wissenschaftlicher Publikationen, sondern veröffentlicht seit mehr als drei Jahrzehnten regelmäßig Schaltungen und Softwareprojekte mit viel technischem Know-how in Elektor.

Passende Produkte

- **Joy-IT JDS6600 Signalgenerator & Frequenzzähler**
  www.elektor.de/18714
- **OWON HDS242 2-Kanal-Oszilloskop (40 MHz) + Multimeter**
  www.elektor.de/20415
- **Raspberry Pi Pico RP2040**
  www.elektor.de/19562

WEBLINKS

[1] Parsevalsche Gleichung: https://de.wikipedia.org/wiki/Parsevalsche_Gleichung
[2] Webseite zu diesem Artikel: https://www.elektormagazine.de/230078-02
[3] LTspice: https://tinyurl.com/3zpwzk4y
[4] Datenblatt RP2040 : https://tinyurl.com/2sf4uvfm

Erfolg beginnt mit den richtigen **elektronischen Bauelementen**

Als autorisierter Distributor von Anbietern wie Molex, Omron oder Phoenix Contact bieten wir ein breites Sortiment an elektronischen Bauelementen – zu fairen Staffelpreisen.

conrad.de/elektronische-bauelemente

Alle Teile des Erfolgs

CONRAD

±40 V Linearer Spannungsregler

Eine alternative Stromversorgung für die Fortissimo-100-Endstufe... und andere!

Von Ton Giesberts (Elektor Labs)

Für Puristen, die jede Form von Schaltnetzteil für den High-End-Verstärker Fortissimo-100 ablehnen, bietet dieses Projekt einen linearen, symmetrischen Spannungsregler mit mehr als 500 VA, der sich durch eine niedrige Dropout-Spannung, einen hohen Ausgangsstrom und eine ausgezeichnete Stabilität auszeichnet – und vollständig mit diskreten Bauteilen aus einem Bausatz aufgebaut ist!

Da fast alle Hochleistungs-Audioverstärker von einer stabilisierten Versorgungsspannung profitieren, wurde dieses lineare Netzteil speziell für eine symmetrische Ausgangsspannung von ±40 V und Spitzenströme von 13 A (15 A Spitze erreichbar) entwickelt. Der durchschnittliche Strom beispielsweise einer Fortissmo-100-Endstufe, der eine 3-Ω-Last treibt, beträgt etwa 4 A pro Kanal.

Überlegungen zum Entwurf

Es hat sich gezeigt, dass der High-End-Audio-Verstärker Fortissimo-100 von Elektor [1] am besten mit einem geregelten ±40-V-Netzteil funktioniert, so dass ein „einfaches" Netzteil, bestehend aus Transformator, (Brücken-)Gleichrichter und einem Satz dicker Speicherkondensatoren nicht in Frage kommt. Ein Schaltnetzteil ist vielleicht auch nicht ganz passend, aber das ist eher eine Frage des persönlichen Geschmacks, denn der Schaltregler SMPS800RE leistet gute Arbeit. Dennoch kann es zwingende Gründe für diesen Linearregler geben, der wie der Verstärker selbst nur aus bedrahteten Bauteilen besteht.

Damit der Spannungsregler ohne Einbrüche der Ausgangsspannung arbeiten kann, muss die Eingangsspannung der Schaltung die Ausgangsspannung um mindestens 3 V übersteigen, bei stärkeren Schwankungen der Netzspannung sogar noch mehr. Im Vergleich zu den meisten Schaltnetzteilen, die einen weiten Eingangswechselspannungsbereich besitzen, ist ein Linearregler aber weniger effizient, und es wird ein großer Leistungstransformator mit einer höheren Nennleistung benötigt als ohne Linearregler.

Die meisten handelsüblichen Netztransformatoren haben heutzutage standardisierte Sekundärspannungen. Um direkt ±40 VDC

zu erzeugen, ist ein Transformator mit einer Nennspannung von 2×30 V vielleicht die beste Wahl. Die resultierende Leerlauf-Gleichspannung beträgt dann in der Regel etwa 42 VDC, was weitgehend vom internen Aufbau des Transformators und dem Spannungsabfall an den Gleichrichterdioden abhängt. In der Praxis ist die Leerlauf-Ausgangsspannung eines Leistungstransformators immer ein paar Prozent höher als unter Last. Die nächsthöhere Standard-Sekundärspannung beträgt 35 V, was bei niedriger Ausgangsleistung etwa 49...50 VDC ergibt - oder mehr, in meinem Labs-Testaufbau habe ich sogar fast 52 V gemessen.

Bei einer 8-Ω-Last an der Endstufe benötigt der Regler nur eine kleine Glättungskapazität. Der Vorteil der größeren Welligkeit ist eine etwas geringere Verlustleistung in dem/den Regler(n) im Netzteil. Bei niedrigeren Impedanzen sollte die Restwelligkeit jedoch nicht über die Dropout-Spannung (43 V bei 10 A) hinausgehen In einem Labortest erwies sich ein Ringkerntransformator mit 2×35 V und 300 VA und einer Glättungskapazität von 20.000 µF als robust genug, um den Regler zu versorgen. Die maximale Sinusleistung (kurz vor dem Clipping) bei 20 Hz und 0,1 % THD+N an einer 3-Ω-Last verursachte einen Dropout von nur 1,8 V_{Spitze} am Versorgungsausgang. Wohlgemerkt, die Dauerausgangsleistung beträgt dann 227 Watt an der 3-Ohm-Last, und der 300-VA-Transformator ist leicht überlastet, aber nicht genug, um die Schutzschaltung des Fortissimo-100 auszulösen.

Funktionsweise in der Theorie

Die Grundlage eines jeden Spannungsreglers besteht darin, die Ausgangsspannung zu messen, sie mit einem Referenzwert zu vergleichen und die Ausgangsstufe entsprechend zu steuern, um etwaigen Änderungen entgegenzuwirken. Obwohl die vorliegende Reglerschaltung diesem Konzept folgt, gibt es eine markante Abweichung: Die viel höhere sekundäre Referenzspannung von etwas über 33 V liegt relativ nahe an der Ziel-Ausgangsspannung von 40 V. Je höher die Referenzspannung - hier 33,6 V - desto mehr Spielraum bleibt einer (einfachen) Schaltung, um sowohl die Welligkeit der Eingangsspannung zu unterdrücken als auch die Regelung der Ausgangsspannung zu erhöhen.

Einfach ausgedrückt, besteht die Schaltung aus einer Referenzspannung, einem Differenzverstärker und einem Ausgangspuffer. Zusätzlich sind beide Regler mit einem SOA-Schutz (Safe Operating Area) ausgestattet. Schauen wir uns **Bild 1** an, um die Funktionsweise des **positiven** Reglers zu untersuchen.

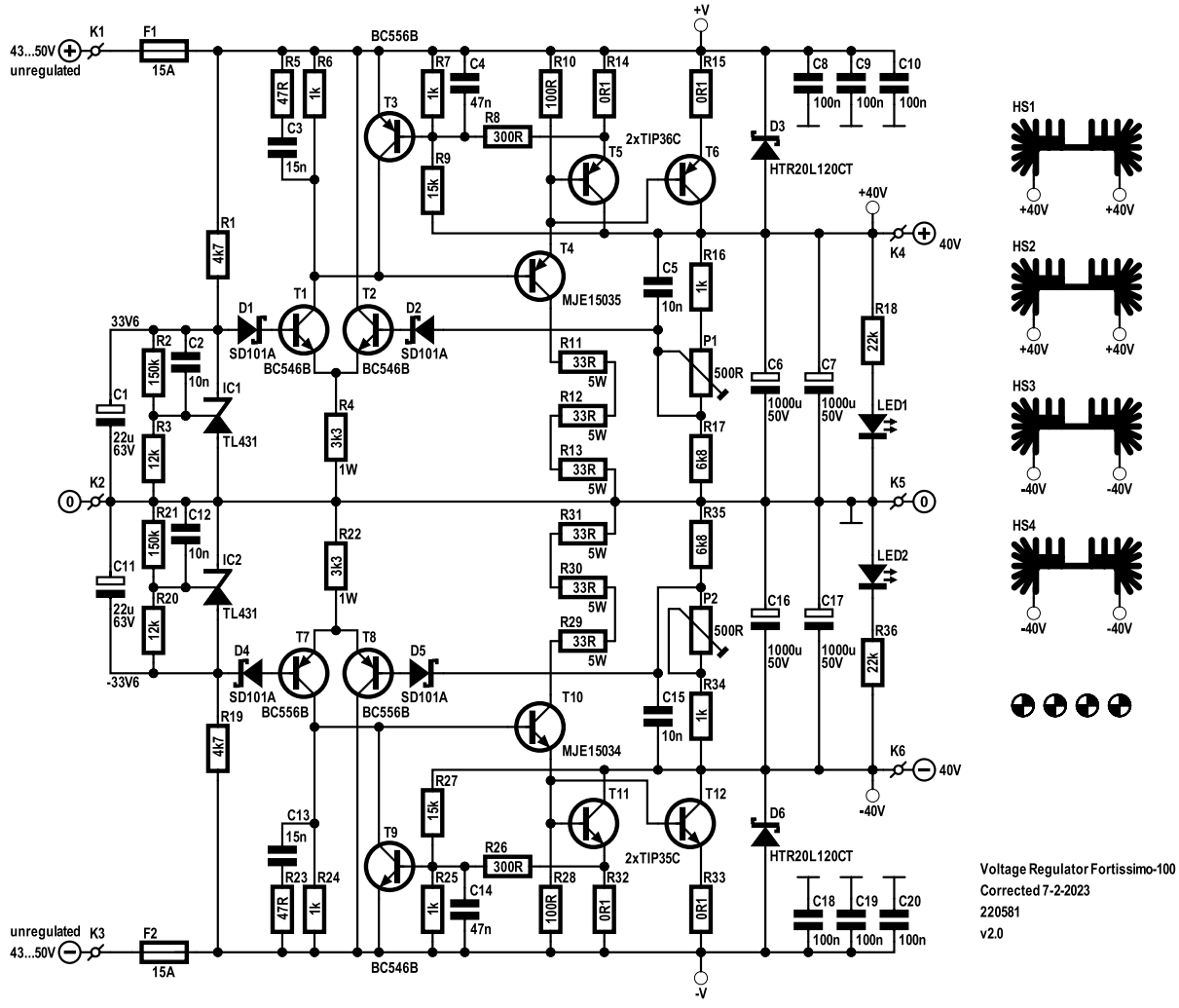


Bild 1. Schaltung des linearen ±40-V-Spannungsreglers, der hauptsächlich, aber nicht ausschließlich, für den High-End-Verstärker Fortissimo-100 von Elektor bestimmt ist.

Kurze Spezifikationen

Eingangsspannungsbereich	52 VDC (bei Low-Power) bis 43 VDC
Ausgangsspannungsbereich	ca. 38,9 VDC bis 41,4 VDC (theoretisch) 38,6 VDC bis 41,1 VDC (gemessen)
Spannungsabfall (Dropout) bei 6 A	42 V
Spannungsabfall (Dropout) bei9,5 A	43 V
Spannungsabfall (Dropout) bei 13,5 A	44 V
Maximaler Strom	15 A Spitze (halbe Sinuskurve) 4,8 A gemittelt
SOAR-Schutz	15 A bei Eingangsspannung 45 VDC
Unterdrückung der Restwelligkeit	>60 dB (bei 5 ADC Last)
Leerlaufstrom	27 mA (bei 52 VDC Eingangsspannung)
Zusammenbau	Elektor-Bausatz; Netztrafo, (Brücken-) Gleichrichter, Ladeelkos sind hinzuzufügen

Referenzspannung

Die Referenzspannung wird nicht durch eine Z-Diode erzeugt, da übliche Z-Dioden üblicherweise erhebliche Temperaturkoeffizienten besitzen. Spezielle temperaturkompensierte Versionen sind heutzutage nur noch schwer zu bekommen, insbesondere mit einer Z-Spannung von 33 V. Anstelle einer Z-Diode wird deshalb eine einstellbare Präzisions-Shunt-Spannungsreferenz namens TL431 mit einer maximalen Arbeitsspannung von 36 V verwendet. Ihre interne Referenzspannung (das heißt, die ursprüngliche Referenzspannung des 40-V-Reglers) beträgt typisch 2,495 V. Der Kathodenstrom durch die TL431 wird durch den Widerstand R1 eingestellt. Wenn die Eingangsspannung zwischen 43 V und 50 V liegt, wird der Strom zwischen 1,9 mA und 3,4 mA eingestellt, was sich als ausreichend erwiesen hat, um eine stabile Referenzspannung von 33,9 V zu erzeugen. Die 33,9 V werden durch die Widerstände R2 und R3 wie folgt eingestellt:

$$U_{KA} = 2{,}495 \times (1 + R2/R3) + I_{REF} \times R2$$

Der Einstellstrom I_{REF} des TL431 beträgt typisch 1,8 µA, so dass die Referenzspannung theoretisch 33,95 V beträgt. Dies ist jedoch für einen Kathodenstrom von 10 mA spezifiziert; beim Prototyp liegt dieser Strom etwas niedriger, so dass in der Praxis eine Spannung von 33,55 V gemessen wurde. Der TL431 wird durch C1 entkoppelt, während C2 die Gesamtstabilität verbessert.

Differenzverstärker

Der Differenzverstärker ist minimalistisch aufgebaut und besteht aus T1 und T2 mit R4 als Stromquelle. Die Spannung an der Basis von T1 ist ziemlich konstant. Das Gleiche gilt für die Spannung an R3, auch wenn die Spannung am Basis-Emitter-Übergang von T1 und an D1 leicht mit der Temperatur schwankt. Die Schottky-Dioden D1 und D2 begrenzen eine (gerade noch denkbare) Sperrspannung von T1 und die Basis-Emitter-Spannung von T2.
Um den Einfluss des Spannungsabfalls über den Dioden zu verringern und gleichzeitig die Eingangs-Offsetspannung des Differenzpaares aufgrund von Temperaturänderungen nicht zu stark werden zu lassen, ist das Transistorpaar auf der Platine nebeneinander angeordnet, so dass beide Diodenübergänge dieselbe Temperatur haben. Ein paar Millivolt Offset, die auch durch Unterschiede der beiden Transistoren T1 und T2 verursacht werden, haben keine nennenswerte Auswirkung auf die viel höhere Ausgangsspannung von 40 V. Selbst eine Offset-Änderung von 30 mV bedeutet weniger als 1 % Abweichung der Ausgangsspannung, was für den Betrieb des Leistungsverstärkers unerheblich ist.
Die Spannung am Kollektorwiderstand R6 wird zur Ansteuerung der Ausgangsstufe verwendet. R5 mit C3 sowie C4 und C5 bilden die Frequenzkompensation, um den Regler auch bei aktiver SOA-Schutzschaltung (T3/R7/R8/R9) stabil zu halten. Der Spannungsteiler R16-P1-R17 misst die Ausgangsspannung und sorgt für die Gegenkopplung des Differenzverstärkers. Um alle Toleranzen ausgleichen zu können, lässt sich an P1 ein Ausgangsspannungsbereich von etwa 38,6 V bis 41,1 V einstellen. Wenn sich der Schleifer des Trimmpotis in der Mitte befindet, liegt die Ausgangsspannung ziemlich genau bei 40 V.

Ausgangsstufe

Obwohl es Transistoren gibt, die den maximal erforderlichen Ausgangsstrom bewältigen könnten, wenn eine konstante Eingangsspannung von 50 V an den Regler angelegt wird, teilen sich hier die beiden Transistoren T5 und T6 brüderlich die Arbeit, um:

- die Verlustleistung pro Transistor auf ein sicheres Maß zu begrenzen
- den Überlastbereich zu vergrößern
- eine niedrigere Dropout-Spannung und einen größeren sicheren Betriebsbereich zu erreichen

Die Umsetzung dieser Kriterien verringert das Risiko, dass die Endstufe im Falle einer Überlast oder sogar eines Kurzschlusses beschädigt wird. Die „dicken" PNP-Leistungstransistoren TIP36C (und die NPN-Varianten TIP35C im Negativregler) sind von mehreren Herstellern leicht erhältlich. Im positiven Regler werden PNP-Transistoren verwendet, um den kleinstmöglichen Spannungsabfall der Ausgangsstufe so niedrig wie möglich zu halten, wobei die Basisströme gegen Masse fließen.
Die Dropout-Spannung ist die Summe aus der Sättigungsspannung der Transistoren und dem Spannungsabfall an den Emitterwiderständen. Ein niedrigerer Wert für die Emitterwiderstände würde die Dropout-Spannung zwar etwas verringern, aber die Ströme durch die beiden Transistoren können zu stark werden. Bei hohen Kollektorströmen ist die Verstärkung der Transistoren sehr gering, so dass ein zusätzlicher Transistor (T4) erforderlich ist, um den Ausgang der Differenzstufe zu puffern.
Um zu verhindern, dass die Sättigungsspannung von T4 die Dropout-Spannung der Ausgangsstufe erhöht, ist sein Kollektor über eine Reihenschaltung von Widerständen mit Masse verbunden. Dies begrenzt die Verlustleistung von T4 und den Bedarf an Kühlkörpern. Allerdings gibt es dabei einen Haken: Sollte - aus welchen Gründen auch immer - die Eingangsspannung unter die Dropout-Spannung fallen, leitet T4 dauerhaft und die Verlustleistung in seinem Kollektorwiderstand von insgesamt 100 Ω ist mit 16 W (bei 40 V Eingangsspannung) recht hoch. Dies sollte zwar in der Praxis nicht vorkommen, dennoch wurden drei 5-Watt-Widerstände

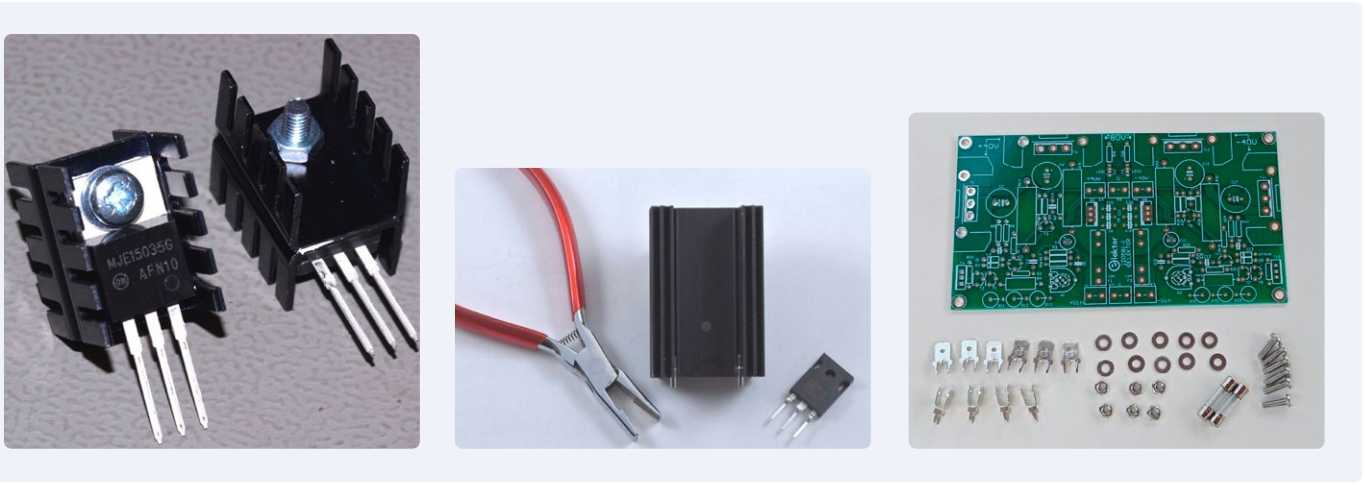


Bild 2. Einige Aufnahmen aus der Bauanleitung für den ±40-V-Linearspannungsregler. Die Anleitung ergänzt diesen Artikel und kann unter [2] kostenlos heruntergeladen werden.

verwendet, damit sie auch im Falle eines Falles nicht abbrennen.

Ein weiterer Vorteil dieses Kollektorwiderstands ist, dass er die Basisströme von T5 und T6 begrenzt und somit als einfache Strombegrenzung fungiert. Für den eigentlichen Schutz sorgt jedoch T3: Der Ausgangsstrom wird vom Spannungsteiler R7/R8 als Spannungsabfall über dem Emitterwiderstand von T5 gemessen, und dieser treibt die Basis von T3. Wenn zum Beispiel der Strom durch R14 etwa 7 A beträgt, liegt der gesamte Ausgangsstrom bei 14 A. Der höchste zu erwartende Ausgangsstrom beträgt aber etwas über 12 A_{Spitze} mit einer 3-Ω-Last am Verstärkerausgang. T3 fängt an zu leiten, und zwar - wegen R9 - sogar schon früher, je nach der Spannung an T5. Der genaue Pegel, bei dem T3 in Durchlassrichtung vorgespannt wird, ist temperaturabhängig und sinkt mit steigender Temperatur - ein zusätzlicher Schutz, und bei Musikwidergabe wird dies kein Problem sein.

D3 schützt die Ausgangsstufe, falls die Eingangsspannung plötzlich unterbrochen oder kurzgeschlossen wird. T5 und T6 sind mit einem Paar 1000-µF-Kondensatoren mit niedrigem ESR entkoppelt. LED1 zeigt, ob die Spannung von +40 V am Ausgang anliegt.

Obwohl es auf den Fotos den Anschein hat, dass D6 auf der Platine verkehrt herum eingebaut ist, können sowohl D6 als auch D3 in beiden Richtungen eingebaut werden und funktionieren trotzdem korrekt. Die Diode HTR20L120CT hat in ihrem 3-poligen TO220-Gehäuse nämlich zwei interne Dioden mit einer gemeinsamen Kathode, die mit dem mittleren Pin des Bauteils verbunden ist.

Der Reglereingang ist durch eine 15-A-Sicherung geschützt. Der maximale effektive Strom muss berücksichtigt werden, und bei maximalem Sinus-Halbwellenstrom beträgt der Effektivwert $I_{Spitze}/2$, also 6,5 A. Bei sehr niedrigen Audiofrequenzen von beispielsweise 16,4 Hz (wenn Sie Orgelmusik mögen) kann der Spitzenstrom jedoch einige Millisekunden andauern. Damit die Sicherung unter solchen Bedingungen nicht durchbrennt, wird hier ein 15-A-Typ verwendet, was nebenbei auch den Spannungsabfall reduziert. Wenn der Verstärker und/oder der Regler viel mehr Strom abnimmt, brennt die Primärsicherung am Netztrafo ohnehin durch, die 15-A-Sicherung kommt bei einem plötzlichen Kurzschluss „dahinter" zu ihrem zuverlässigen Einsatz.

Bausatz, Bauanleitung und Stückliste

Der Elektor-Shop bietet einen umfangreichen Bausatz für das Projekt Linearer Spannungsregler [2] an, der die Platine und alle in der

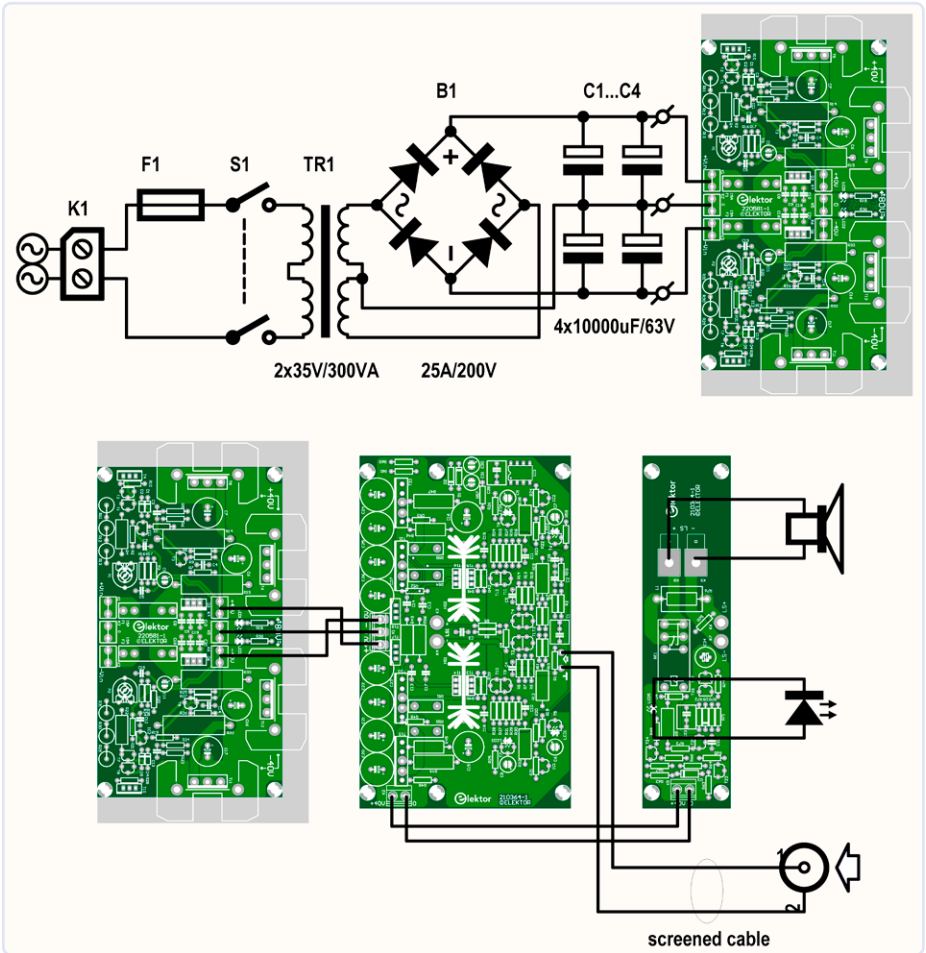
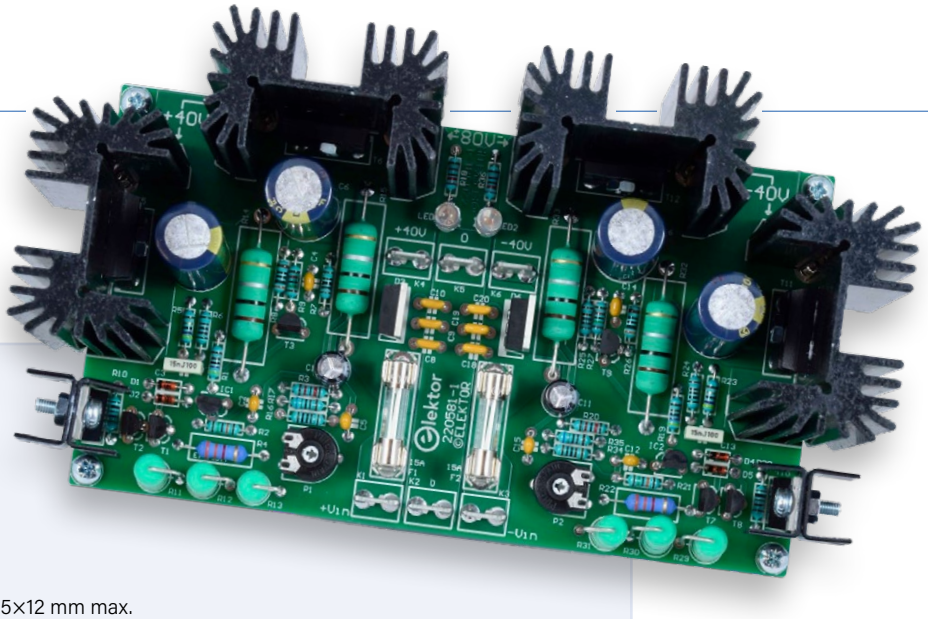


Bild 3. Vorgeschlagener Verdrahtungsplan für die ungeregelte Stromversorgung (oben) und Schaltplan der ±40-V-Kombi aus linearem Spannungsregler und Fortissimo-100 (unten).



Stückliste

(Elektor-Bausatzinhalt)

Widerstände:

R1, R19 = 4k7, 1%, 0,6 W
R2, R21 = 150 k, 1 %, 0,6 W
R3, R20 = 12 k, 1 %, 0,6 W
R4, R22 = 3k3, 5%, 1 W, Gehäusegröße 5×12 mm max.
R5, R23 = 47 Ω, 1%, 0,6 W
R6, R7, R16, R24, R25, R34 = 1 k, 1%, 0,6 W
R8, R26 = 300 Ω, 1 %, 0,6 W
R9, R27 = 15 k, 1 %, 0,6 W
R10, R28 = 100 Ω, 1 %, 0,6 W
R11, R12, R13, R29, R30, R31 = 33 Ω, 5%, 5 W, Gehäuse-ø 6,4 mm max. (axial, aufrecht montiert)
R14, R15, R32, R33 = 0,1 Ω, 10%, 5 W (Gehäuse 7,8×25 mm max.)
R17, R35 = 6,8 k, 1%, 0,6 W
R18, R36 = 22 k, 1 %, 0,6 W
P1, P2 = 500 Ω, 0,15 W, Trimmer, von oben einstellbar (Piher PT10LV10-501A2020-S)

Kondensatoren:

C1, C11 = 22 µ, 20%, 63 V, Raster 2,5 mm, ø 6,3 mm max.
C2, C5, C12, C15 = 10 n, 10%, 100 V, Keramik X7R, Raster 5 mm
C3, C13 = 15 n, 5%, 100 V, PET, Raster 5 mm
C4, C14 = 47 n, 10 %, 50 V, Keramik X7R, Raster 5 mm
C6, C7, C16, C17 = 1000 µ, 20%, 50 V, Raster 5 mm, ø 12,5 mm, 5000 h bei 105 °C (EEUFC1H102L, Panasonic)
C8, C9, C10, C18, C19, C20 = 100 n, 10%, 100 V, Raster 5 mm, Keramik X7R

Halbleiter:

D1, D2, D4, D5 = SD101A SB00018/D8, DO-35
D3, D6 = HTR20L120CT, TO-220
LED1, LED2 = LED, grün, 5 mm (T-1¾)
T1, T2, T9 = BC546B, TO-92
T3, T7, T8 = BC556B, TO-92
T4 = MJE15035, TO-220
T5, T6 = TIP36C, TO-247
T10 = MJE15034, TO-220
T11, T12 = TIP35C, TO-247
IC1, IC2 = TL431BCLPG, TO-92

Außerdem:

K1, K2, K3, K4, K5, K6 = Faston-PCB-Tab, zwei Stifte, Loch-ø 1,4 mm, 6,35×0,83 mm
F1, F2 = Sicherungsklammern, 20 A, Littelfuse 01000020Z, für 5×20 mm-Sicherung
F1, F2 = Schmelzsicherung, flink, 15 A, 5×20 mm
HS1, HS2, HS2, HS4 = Kühlkörper MC33271 (für T5/T6/T11/T12), 2,7 °C/W
4× Kühlkörper Typ FK231SA220, 24 K/W (für T4/T10, je zwei)
10× M3-Scheibe, glatt, Stahl
6× Schraube M3×10, Flachkopf
6× M3-Mutter

Stückliste aufgeführten Bauteile enthält. Dieser hervorragende Bausatz erspart Ihnen die Mühe, elektronische und mechanische Teile zu bestellen und Platinen auf Bestellung anfertigen zu lassen.
Dem Bausatz liegt eine zwölfseitige Bauanleitung bei, die Schritt für Schritt erklärt, wie man das Projekt zusammenbaut und hoffentlich ein perfektes Ergebnis erzielt. Die Anleitung ist reich an Zeichnungen und Fotos, von denen einige in **Bild 2** zu sehen sind. Sie enthält auch viele Tipps und Details zum präzisen Löten, zur Positionierung der Teile, zur Handhabung der Werkzeuge und zu den einfachen mechanischen Arbeiten, die für den Bau des Projekts erforderlich sind.
Da es sich bei dem Spannungsregler nicht um eine vollständige Stromversorgung handelt, müssen ein Leistungstransformator, ein Gleichrichter und Glättungskondensatoren hinzugefügt und wie im Verdrahtungsplan in **Bild 3** mit dem Spannungsregler und dem Fortissimo-100-Verstärker verbunden werden.

Sicherheitsaspekte

Obwohl der Aufbau des Projekts und seine praktische Anwendung in der Bauanleitung ausführlich beschrieben sind, fühlen wir uns doch verpflichtet, den folgenden Sicherheitshinweis auch in diesem Artikel zu drucken:
Die großen Kühlkörper sind mit der ±40-V-Ausgangsspannung verbunden, nicht mit Masse. Schalten Sie deshalb immer die Eingangsspannung ab, bevor Sie den Regler berühren oder daran arbeiten!

Erzielte Ergebnisse

Im Elektor-Labor wurde ein Testaufbau erstellt, um den die Leistungsfähigkeit des Fortissimo-100-Verstärkers in Kombination mit dem hier beschriebenen ±40-V-Linearspannungsregler zu verifizieren. Beide Geräte wurden aus den jeweiligen Elektor-Bausätzen gebaut. Im ungeregelten Schaltungsteil wurden folgende zusätzliche Bauteile verwendet:

- 1 Stk. TX-146-300-235, Leistungstransformator (300 VA, 2×35 VAC sekundär).
- 2 Stk. 10.000 µF-Elko pro Versorgungsspannungsschiene (das heißt 20 mF auf jeder Schiene).
- 1 Stk. SB352SBPC, Brückengleichrichter, 35 A/200 V (25 A/100 V funktioniert ebenfalls zufriedenstellend).

Bei niedrigen Ausgangspegeln des Fortissimo-100 zeigt das Frequenzspektrum, dass im Vergleich zum Schaltnetzteil SMPS800RE sehr geringe Verbesserungen erzielt werden können (**Bild 4**). Die Grafik zeigt das Frequenzspektrum bei 1 W an 8 Ω. Die Artefakte des SMPS800RE sind verschwunden, aber der Rest des Spektrums ist im Wesentlichen gleich. Die Gesamtleistung der Kombination ist beeindruckend, mit einer harmonischen Verzerrung und einem Rauschen, das so niedrig ist wie:
0,0007 % (1 kHz, 1 W, 8 Ω, B = 22 kHz)
0,0013 % (1 kHz, 1 W, 8 Ω, B = 80 kHz)

Der hier beschriebene ±40-V-Linearspannungsregler, der als Bausatz bei Elektor erhältlich ist, ist eine gute Alternative zu den besten „bezahlbaren" Schaltnetzteilen auf dem Markt. Er sollte diejenigen unter Ihnen zufriedenstellen, die das Konzept oder die Leistung von „diesen @#!%^-Schaltreglern" ablehnen, auch wenn sie so praktisch und so klein sind. Sie können sich gerne an den technischen Diskussionen über den linearen ±40-V-Spannungsregler auf der Elektor-Labs-Seite beteiligen, die für das Projekt eingerichtet wurde [3].

RG -- 220581-02

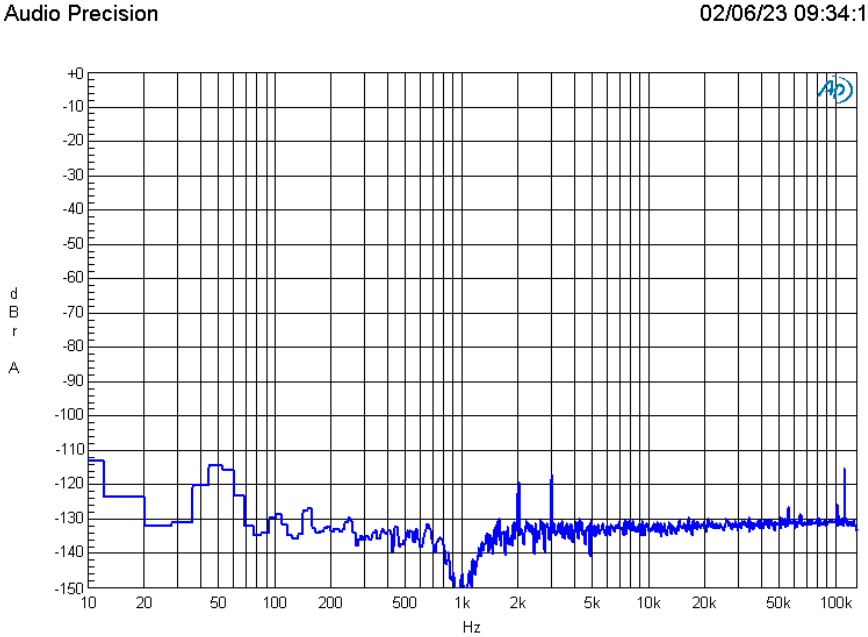


Bild 4. Hier sind die Messergebnisse! Frequenzspektrum eines Fortissimo-100-Ausgangssignals bei Betrieb mit 1 W an 8 Ω und Versorgung durch den ±40-V-Linear-Spannungsregler (Platine Nr. 220581-1).

Haben Sie Fragen oder Kommentare?

Wenn Sie technische Fragen haben, können Sie sich per E-Mail an die Elektor-Redaktion wenden: redaktion@elektor.de.

Passende Produkte

- **Bausatz ±40 V Linearer Spannungsregler**
  https://elektor.de/20439

WEBLINKS

[1] High-End-Verstärker Fortissimo-100: https://www.elektormagazine.de/magazine/elektor-281/61094
[2] Bausatz ±40 V Linearer Spannungsregler: https://elektor.de/elektor-40-v-linear-voltage-regulator-kit
[3] Projekt „±40 V Linearer Spannungsregler" auf Elektor Labs:
https://elektormagazine.de/labs/linear-voltage-regulator-for-fortissimo-100

Drahtlose MCU-Kommunikation flexibel gemacht

EEPROM eröffnet Netzwerk-Perspektiven für drahtlose MCUs

Von Gamal Labib (Ägypten)

Wenn Sie einen Mikrocontroller mit dem ESP8266 an ein WLAN anschließen, möchten Sie vielleicht einen flexibleren Ansatz als fest im Code verankerte WLAN-Zugangsdaten. Hier möchte ich Ihnen eine Lösung mit einer Reihe von bevorzugten AP-Netzwerken demonstrieren, aus denen man interaktiv auswählen kann. Zusätzlich wird die WPS-Funktion (Wi-Fi Protected Setup) genutzt, die von vielen APs und Routern unterstützt wird.

Der ESP8266 ist ein beliebtes Mikrocontroller-Modul, das drahtlose Kommunikation unterstützt und für verschiedene Anwendungen im Internet der Dinge (IoT) eingesetzt wird. Die Boards, die auf diesem Modul basieren, haben Access-Point-Zugangsdaten wie den Service Set Identifier (SSID) und das Passwort fest in ihren Arduino-Sketches kodiert. In einigen Fällen geben die Entwickler sogar feste IP-Einstellungen vor. Solche fest kodierten Einstellungen können jedoch zu Problemen führen, wenn sich die Topologie des drahtlosen lokalen Netzwerks (WLAN) ändert. Dann heißt es, zur Arduino-IDE oder einer ähnlichen Entwicklungsumgebung zurückzukehren, nur um die Sketches auf den eingesetzten MCU-Boards zu aktualisieren. Das ist nicht nur lästig, sondern kann zu einem Problem werden, da die Boards abgeschaltet und demontiert werden müssen, um die Zugangsdaten auf den neusten Stand zu bringen.

In diesem Artikel werde ich Ihnen eine flexible Lösung zur Verbindung eines MCU-Boards mit einem WLAN demonstrieren. Anstatt einen einzigen Satz von WLAN-Zugangsdaten in Projektsketches fest zu kodieren, warum nicht eine Reihe von bevorzugten AP-Netzwerken zur interaktiven Auswahl anbieten? Durch einen interaktiven Dialog zwischen User und MCU-Board über einen Touchscreen, eine Webseite oder ein ähnliches Mittel wird es möglich, neue WLAN-Einstellungen für das MCU-Board im laufenden Betrieb festzulegen. Außerdem können wir die Funktion *Wi-Fi Protected Setup* (WPS) nutzen, die von vielen APs und Routern unterstützt wird. Mit WPS kann ein MCU-Board nach Belieben dem bevorzugten Netzwerk des Users beitreten. Es stellt sich jedoch eine Frage: Müssen bei jedem Neustart der MCU die Verbindungsschritte des interaktiven Dialogs oder des WPS durchlaufen werden? Die Antwort ist nein, solange wir die neu festgelegten WLAN-Einstellungen in einem nichtflüchtigen Speicher aufbewahren, auf den die MCU zugreifen kann. Glücklicherweise verfügt der ESP8266 über ein internes EEPROM (Electrically Erasable Programmable Read-Only Memory), auf das der Code des Users Daten speichern und abrufen kann. Ich habe diese Funktion genutzt, um bis zu zehn WLAN-Zugangsdaten zu speichern, die mit einer der oben genannten Methoden festgelegt wurden. Dieser Ansatz bietet nicht nur Flexibilität bei der Vernetzung, sondern ermöglicht es dem MCU-Board auch, sich mit dem WLAN mit der besten Abdeckung unter den zehn gespeicherten zu verbinden, wenn der Entwickler dies wünscht.

Man sollte allerdings beachten, dass ein umfangreiches und häufiges Schreiben auf das interne EEPROM nicht empfohlen wird, da die erwartete Lebensdauer eines EEPROMs davon eingeschränkt wird. Um diese Einschränkung zu umgehen und die vorgesehenen Aufgaben der MCU zu erhalten, habe ich in meiner Lösung einen externen EEPROM-Chip eingebunden, um die gleiche Aufgabe mit einem externen EEPROM zu erledigen. Für unser WLAN-Konfigurationsbeispiel werden wir jedoch keine umfangreichen Schreibvorgänge durchführen, daher habe ich mich zunächst für die Verwendung des internen EEPROMs entschieden.

Hardware

Die Stückliste für dieses Projekt ist recht kurz. Ich habe das ESP8266-basierte Board WeMos D1 Mini verwendet, das für seinen geringen Platzbedarf und sein einfaches Handling in der Arduino-IDE bekannt ist. Ein 0,9"-OLED-Grafikdisplay-Modul mit einer Auflösung von 128×64 Pixeln zeigt Informationen und Debugging-Meldungen. Es ist mit dem I²C-Bus der MCU verbunden, zusammen mit einem externen 8-KB-EEPROM-Chip.

Um zwischen den acht Betriebsarten der MCU zu wählen (**Tabelle 1**), habe ich drei DIP-Schalter kombiniert, die mit den drei digitalen GPIOs D5…D7 des WeMos verbunden. Da die Pins der DIP-Schalter so winzig sind, dass sie nicht in ein

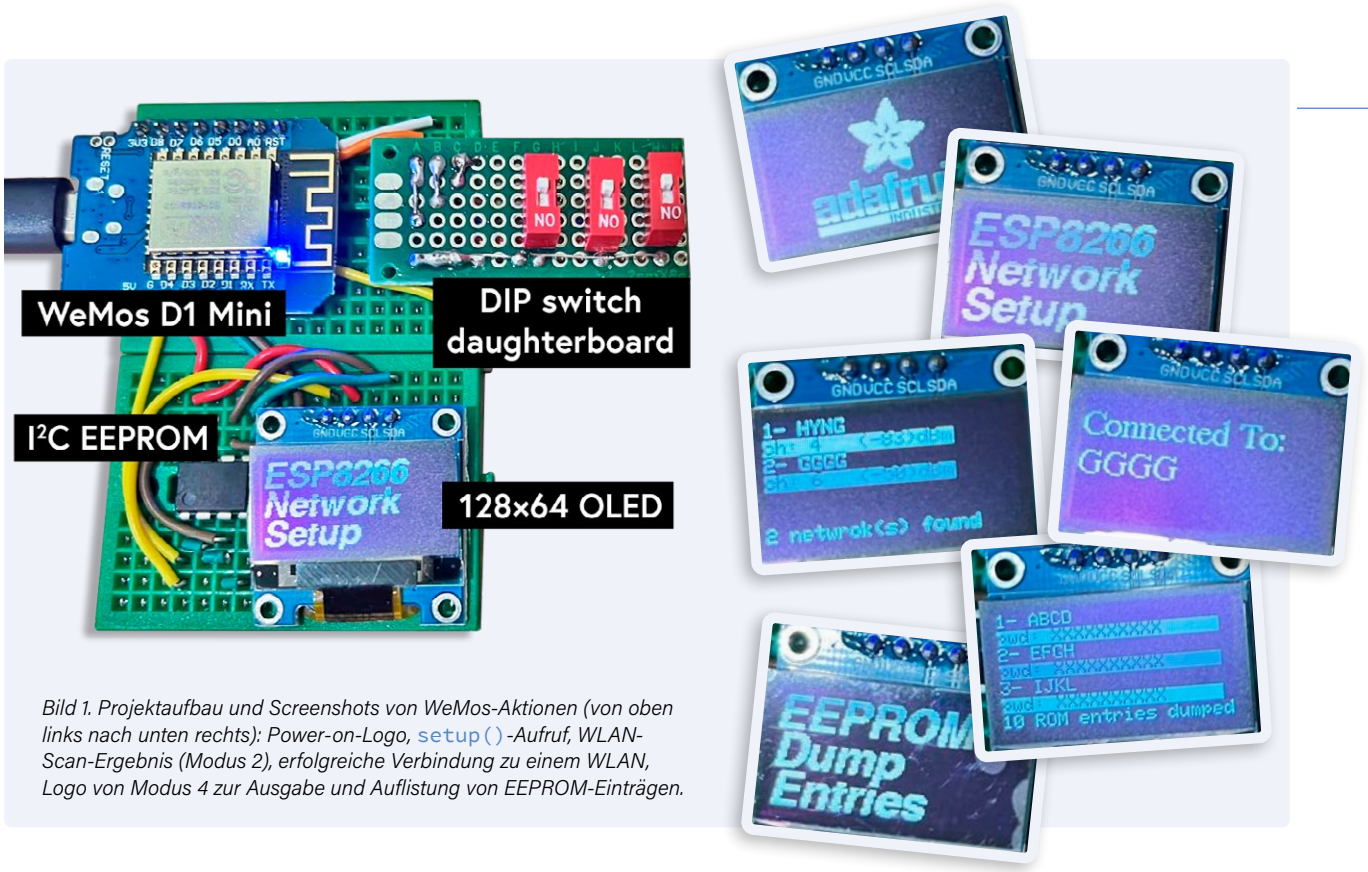


Bild 1. Projektaufbau und Screenshots von WeMos-Aktionen (von oben links nach unten rechts): Power-on-Logo, `setup()`-Aufruf, WLAN-Scan-Ergebnis (Modus 2), erfolgreiche Verbindung zu einem WLAN, Logo von Modus 4 zur Ausgabe und Auflistung von EEPROM-Einträgen.

Tabelle 1: DIP-Schalter-Konfiguration der relevanten WeMos-Modi.

Mode	DIP 1	DIP 2	DIP 3	WeMos-Aktion
1	0	0	0	Standard-WLAN gewählt
2	0	0	1	Verfügbare WLANs scannen
3	0	1	0	Internes EEPROM ausgeben
4	0	1	1	Internes EEPROM initialisieren
5	1	0	0	Externes EEPROM prüfen
6	1	0	1	Check external EEPROM
7	1	1	0	WPS
8	1	1	1	Interaktive WLAN-Einrichtung

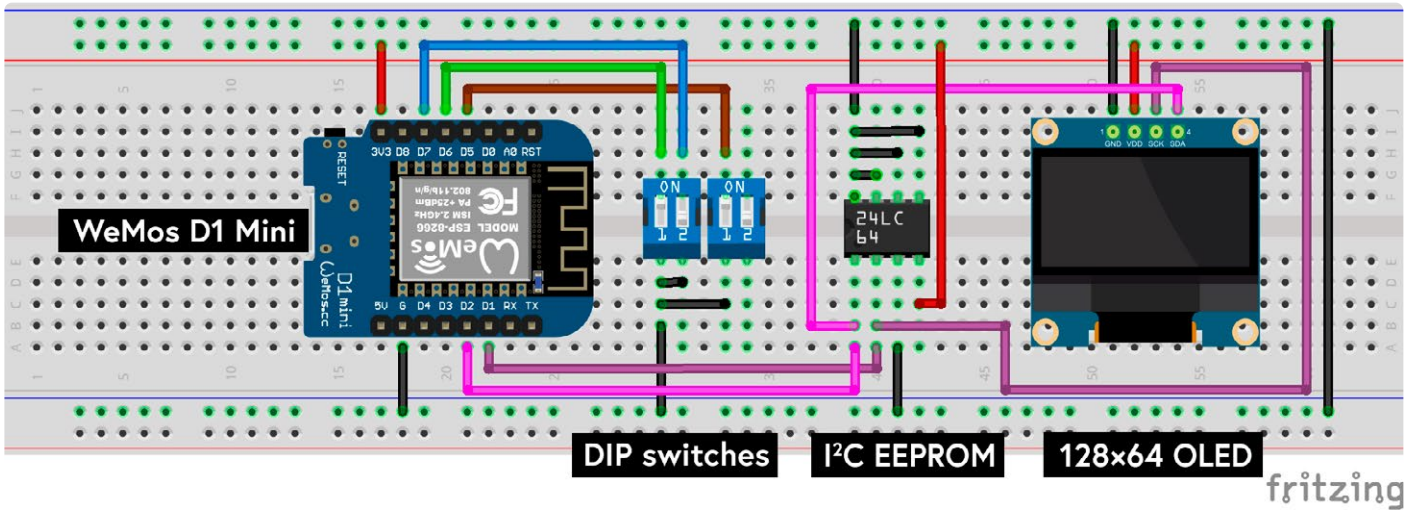


Bild 2. Fritzing-Verdrahtung des Projekts mit einem zusätzlichen vierten DIP-Schalter für erweiterte Netzwerkoptionen.

Breadboard gesteckt werden können, habe ich sie auf eine kleine Experimentierplatine gelötet. Der Breadboard-Aufbau mit der DIP-Schalterplatine und Screenshots des WeMos in Aktion sind in **Bild 1** dargestellt, ein Fritzing-Schaltplan mit den einzelnen Komponenten des Projekts ist in **Bild 2** zu sehen.

Software

Das Projekt legt Wert auf bewährte Praktiken der Softwareentwicklung, wie ein gut strukturierter und wiederverwendbarer Code. Dazu habe ich die etwa 800 Zeilen des Sketches in sechs separate Dateien unterteilt, von denen jede einen bestimmten Teil der Hardware oder eine bestimmte Funktion behandelt (**Tabelle 2**). Dieser Ansatz macht den Code überschaubarer und hilft dem Leser, den komplexen Code zu verstehen. Die Datei mit dem Hauptsketch konzentriert sich auf die WeMos-Netzwerkmodi innerhalb der `setup()`-Funktion, während die `loop()`-Funktion für den WeMos-Anwendungscode frei bleibt. Während der Entwicklungsphase habe ich `loop()` verwendet, um die Onboard-LED blinken zu lassen als Beweis, dass der WeMos korrekt funktioniert.

Nutzung des EEPROMs für WLAN-Netzwerkdaten

Das ESP8266-Modul verfügt über 4 MB Flash-Speicher, von denen 4 KB für die Emulation eines EEPROMs reserviert sind, der üblicherweise in Arduino-Controllern eingebaut ist. Unser Beispiel-Sketch speichert standardmäßig die letzten funktionierenden AP-Zugangsdaten im internen EEPROM. Jedes Mal nach dem Einschalten kann unser Sketch die AP-Zugangsdaten aus dem internen oder externen nichtflüchtigen Speicher abrufen. Das EEPROM ist aber auch über Funktionsaufrufe der EEPROM-Bibliothek des Arduino zugänglich, und ich werde auf diese Weise die Netzwerkdaten speichern, die nichtflüchtig sein müssen, wenn das Modul ausgeschaltet ist. Das Konzept besteht im Wesentlichen aus der Beibehaltung der in **Tabelle 3** vorgeschlagenen Struktur zur Speicherung von WLAN-AP-Zugangsdaten - so viele, wie das interne oder externe EEPROM aufnehmen kann. Jeder nachfolgende Eintrag in der Struktur enthält die SSID des AP und die Passphrase in einem Speicherplatz von 32 Byte beziehungsweise 64 Byte, also insgesamt 96 Bytes pro AP.

Immer wenn der Benutzer ein neues WLAN zur Liste der bevorzugten Netzwerke hinzufügt, fügt der Code dieser Struktur die Anmeldeinformationen (SSID und Passphrase) des neuen Netzwerks hinzu. Der Benutzer muss das EEPROM beim ersten Lauf des Projektcodes im Modus 3 löschen, wodurch das Ende der Liste durch leere Einträge markiert wird. Der Benutzer hat die Wahl, die fest codierten WLAN-Zugangsdaten anzuwenden (Modus 1) oder die Konnektivität anhand der Zugangsdaten im EEPROM zu überprüfen (Modus 8). Mein Projekt beschränkt sich auf diesen Mechanismus für bis zu zehn WLANs, begrenzt durch die definierte Konstante `MEMCNT` im Code, zusätzlich zum Löschen der Struktur und ihrer Initialisierung mit einer voreingestellten Liste von AP-Zugangsdaten nach Wahl des Benutzers. Der Leser kann die Größe von `MEMCNT` je nach Wunsch erhöhen oder verringern. Hintergrundwissen zum Zugriff auf das EEPROM finden Sie unter [2]. Im Folgenden werden die Betriebsmodi des WeMos erläutert, die gemäß Tabelle 1 an den DIP-Schaltern gewählt werden. Anschließend ist ein Neustart des WeMos erforderlich, um die neue Einstellung zu übernehmen.

Tabelle 2: Aufschlüsselung der Arduino-Code-Dateien und ihrer Funktionen.

.ino-Datei	Beschreibung	Zeilen
wemos-d1-mini-network	Globale Variablen, Einrichten des Netzwerkmodus	240
wlan-utils	Manipulation von WLAN-Verbindungen: WPS-Setup, WLANs scannen/auflisten, Verbindung prüfen	110
tft-utils	TFT-Display manipulieren	130
internal-eeprom-utils	EEPROM mit fünf voreingestellten WLAN-Zugangsdaten initialisieren, EEPROM zurücksetzen, Einträge lesen/schreiben	180
external-eeprom-utils	Größe des und Verbindung zum externen EEPROM prüfen	40
ap-web-setup	Interaktives WLAN-Setup: Webserver starten, erkannte WLAN-SSIDs anzeigen, ausgewähltes WLAN im EEPROM speichern	100

Tabelle 3: EEPROM-Housekeeping-Tabelle für multiple AP-Einstellungen.

Variable (Bytes)	Beschreibung	Typ
2	AP-Eintrag Nummer	Integer
32	AP1 SSID	String
64	AP1 Passwort	String
.	.	.
.	.	.
.	.	.
32	APn SSID	String
64	APn Passwort	String

Modus 1: Standard-WLAN-Einstellungen (hartkodiert)

Eine MCU benötigt WLAN-Zugangsdaten, die aus einer SSID und einem Passwort für den jeweiligen AP bestehen. Die Übergabe der WLAN-Zugangsdaten kann entweder fest im geflashten Sketch kodiert sein oder interaktiv an die MCU übergeben werden. Im zweiten Fall benötigt der User entweder eine Tastatur und ein Display oder den seriellen Monitor der IDE, die über die serielle Schnittstelle verbunden ist. In diesem Projekt verwende ich so die konventionelle Methode, die AP-Zugangsdaten direkt im Sketch festzulegen.

```
14:34:00.668 -> scan available wlans
14:34:05.788 -> Disconnecting previously connected WiFi
14:34:08.148 -> scan completed
14:34:08.148 -> 1 Networks found
14:34:08.148 -> 1: GGGG (6)  (-63)
14:34:08.268 ->
```

Bild 3. Meldungen des seriellen Monitors der Arduino-IDE - Scan des drahtlosen Netzwerks im Modus 2.

```
14:35:43.671 -> check external eeprom
14:35:48.751 -> Disconnecting previously connected WiFi
14:35:48.871 -> check external eeprom
14:35:48.871 -> External eeprom isConnected with Status:
14:35:48.871 ->
14:35:48.871 -> TEST: determine size of external eeprom
14:35:48.871 -> 80      FF
14:35:48.871 -> 100     0
14:35:48.871 -> 200     0
14:35:48.871 -> 400     0
14:35:48.871 -> 800     FF
14:35:48.911 -> 1000    AA
14:35:48.911 -> 2000    AA
14:35:48.911 -> external eeprom size: 8192 Bytes
```

Bild 4. Meldungen des seriellen Monitors - Modus 6: Überprüfung der Integrität des externen EEPROMs.

Modus 2: WLANs scannen

In diesem Modus trennt der WeMos die Verbindung zu einem verbundenen WLAN, um nach anderen zugänglichen Netzwerken zu suchen. Erkannte WLANs werden mit ihren SSIDs und Signalstärken in Dezibel (dB) aufgelistet. Dieser Modus gibt dem Leser einen Einblick in die Möglichkeiten der Vernetzung des WeMos-Boards und welchen Modus er für die Einbindung in ein WLAN wählen sollte. **Bild 3** zeigt das Ergebnis nach dem Bootens des WeMos in diesem Modus, bei dem nur ein WLAN erkannt (und später, wie in Bild 1d zu sehen, auch verbunden) wurde.

Modus 3: Internes EEPROM löschen

In diesem Modus löscht der WeMos sein internes EEPROM, um eine neue Liste der bevorzugten WLANs erstellen zu können. Alle Bytes im EEPROM werden in diesem Modus auf 0×00 zurückgesetzt.

Modus 4: Inhalt des internen EEPROMs ausgeben

Dieser Modus veranlasst den WeMos, die Struktur der WLAN-Zugangsdaten, wie in **Tabelle 3** erläutert, aus dem internen EEPROM auszulesen. Bild 1e und Bild 1f zeigen Screenshots der angezeigten EEPROM-Einträge, die derzeit gespeichert sind. In diesem Modus wird eine Reihe von formatierten SSIDs und Passphrasen für die gespeicherten Zugangsdaten der APs auf dem Display ausgegeben.

Modus 5: Internes EEPROM initialisieren

In diesem Modus kann der Leser die Zugangsdaten von zehn bevorzugten WLANs im internen EEPROM voreinstellen. Diese Aktion ist eine Weiterentwicklung der traditionellen Hartkodierung der Anmeldedaten eines einzigen WLANs. Nach dem Neustart im Modus 8 prüft der WeMos die Liste der Zugangsdaten für WLAN-Verbindungen. Wenn seine Versuche fehlschlagen, eine Verbindung zu einem der bevorzugten WLANs herzustellen, muss entweder auf den interaktiven oder den WPS-Modus zurückgegriffen werden.

Modus 6: Überprüfen des externen EEPROM-Status

In diesem Modus prüft der WeMos die Funktionsfähigkeit des externen EEPROMs auf der Platine, in diesem Fall ein 24LC64 mit 8 KB. **Bild 4** zeigt die Ausgabe dieser Überprüfung im Serial Monitor. Ich habe diesen externen Speicher nicht weiter in das Projekt einbezogen, da das interne EEPROM nur minimal genutzt wird. Um ein externes EEPROM zu nutzen, muss der I²C-EEPROM-Code [3] und die Bibliothek [4] in das Projekt integriert werden.

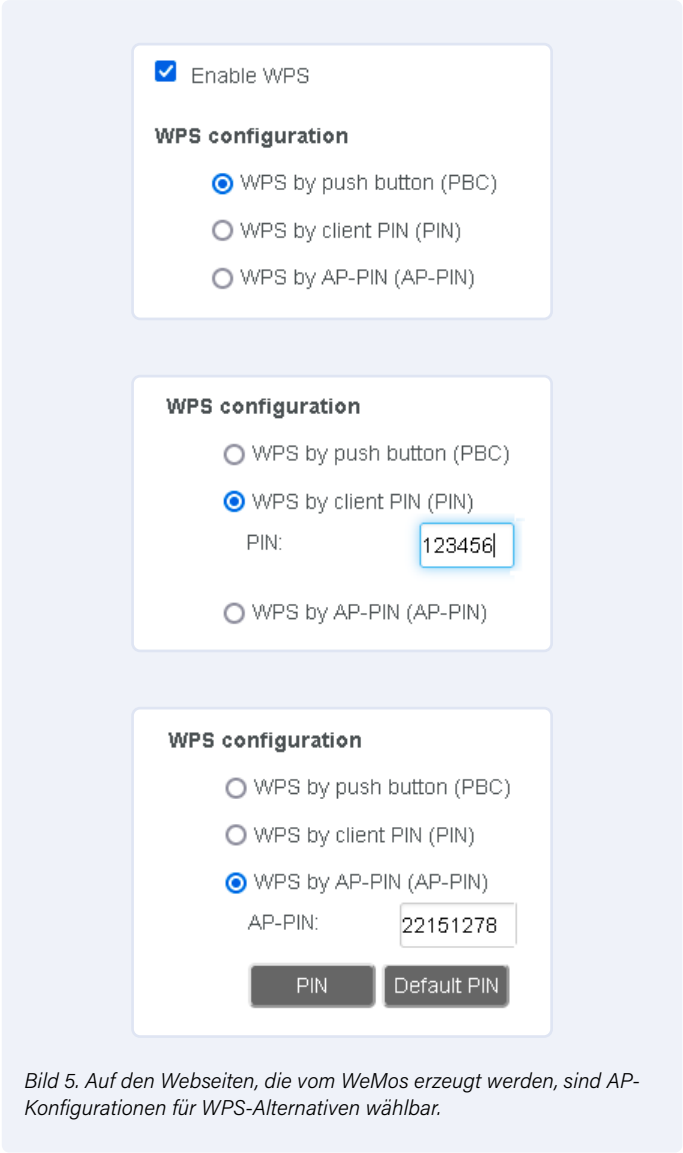


Bild 5. Auf den Webseiten, die vom WeMos erzeugt werden, sind AP-Konfigurationen für WPS-Alternativen wählbar.

Bild 6. WPS-Taste und Anzeige eines typischen APs.

```
14:37:03.552 -> Disconnecting previously connected WiFi
14:37:03.632 -> Try wps if necessary
14:37:03.632 ->
14:37:03.632 -> Could not connect to WiFi ... state= '7'
14:37:03.672 -> Please press WPS button on your router
14:37:03.672 -> WPS config start
```

Bild 7. Meldungen im seriellen Monitor – Modus 7: WPS-Setup.

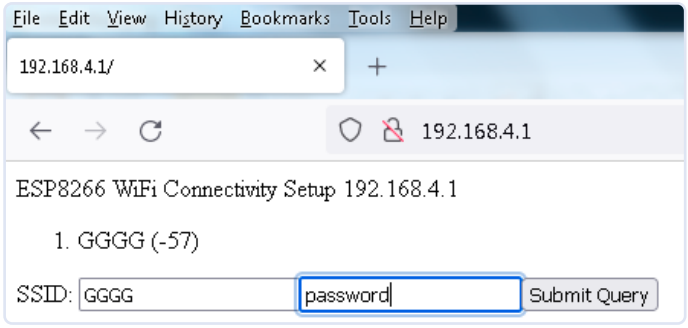


Bild 8. Modus 8, die interaktive WLAN-Auswahl.

Modus 7: Direkte Verbindung zum WLAN (WPS)

WPS (Wi-Fi Protected Setup) ist eine Funktion, die von modernen APs unterstützt wird und den Anmeldevorgang von drahtlosen Geräten mit einem WLAN zu vereinfachen. Anstatt einer MCU die AP-Anmeldedaten wie SSID und Passphrase/Pre-Shared-Key zur Verfügung zu stellen, kann die MCU die AP-Security-PIN oder ihre eigene voreingestellte PIN bereitstellen (siehe **Bild 5**). Alternativ kann ein WPS-Knopf am AP-Gerät gedrückt werden, um eine direkte Verbindung zur anfragenden MCU herzustellen (siehe **Bild 6**). **Bild 7** zeigt das Ergebnis des Verbindungsaufbaus mit WPS. In diesem Projekt beschränke ich die WPS-Verbindung auf die Alternative mit dem AP-Taster, da dies den Sketch einfacher und allgemeingültiger macht, als jeden AP in Reichweite unterzubringen:

```
bool wpsSuccess = WiFi.beginWPSConfig();
if (wpsSuccess) {
    String newSSID = WiFi.SSID();
    if (newSSID.length() == 0)  { wpsSuccess = false; }
}
```

Das WeMos-Board muss sich in der Nähe des gewünschten AP befinden, damit die AP-Parameter im EEPROM des Moduls registriert werden können. Um das Modul wieder mit demselben AP zu verbinden, muss die WPS-Taste später nicht mehr gedrückt werden, da das Modul die gespeicherten AP-Parameter zur Wiederverwendung abruft:

```
WiFi.mode(WIFI_STA);
WiFi.begin(WiFi.SSID().c_str(),WiFi.psk().c_str());
```

Beachten Sie, dass das ESP8266-Modul als Herzstück des WeMos-Boards automatisch die letzten erfolgreichen Netzwerk-Zugangsdaten an einer bestimmten Stelle im internen EEPROM speichert. Daher müssen wir unsere in Tabelle 3 dargestellte Housekeeping-Struktur an anderer Stelle im EEPROM platzieren, um Konflikte mit den anderen Mechanismen der MCU zu vermeiden.

Modus 8: Interaktive WLAN-Einrichtung

In diesem Modus prüft der WeMos zunächst die Korrektheit der Anmeldedaten im EEPROM. Gelingt es der MCU, eine Verbindung zu einem AP herzustellen, endet dieser Modus und die `loop()`-Funktion übernimmt. Wenn nicht, geht die MCU in den interaktiven Modus über, in dem sie die WLANs in ihrer Umgebung scannt und einen Webserver mit der IP-Adresse **192.168.4.1** startet, um die erkannten AP-SSIDs anzuzeigen. Der Benutzer muss den WLAN-Adapter seines Computers auf die festen IP-Einstellungen für das private Netzwerk des WeMos einstellen, da der WeMos keinen DHCP-Dienst anbietet. Auf einer Webseite ähnlich der in **Bild 8** werden die Scanergebnisse angezeigt, und der User kann das bevorzugte Netzwerk auswählen. Er klickt dann auf den Button *Submit Query*, um die WLAN-Zugangsdaten im EEPROM zu speichern und eine Verbindung mit dem ausgewählten WLAN herzustellen. EEPROMs können bei der Vernetzung drahtloser MCUs einen Unterschied ausmachen. Anstatt die WLAN-Zugangsdaten im MCU-Code fest zu kodieren, zeigt dieser Artikel den Zugriff auf ein externes oder das interne EEPROM der MCU, um die Zugangsdaten mehrerer APs für mögliche Verbindungen dynamisch zu speichern und zu aktualisieren. Der zum Artikel gehörende Quellcode ist modular und gut strukturiert, leicht verständlich und bietet wiederverwendbaren Code für den Umgang mit EEPROMs, WLANs und für die Kommunikation mit der MCU über das Web.

RG — 230268-02

Über den Autor

Gamal Labib beschäftigt sich seit zwei Jahrzehnten enthusiastisch mit eingebetteten Systemen und ist derzeit Mentor (bei codementor.io). Er hat einen MEng und einen Doktortitel in IT. Er schreibt nicht nur für Fachzeitschriften, sondern ist auch Gastprofessor an ägyptischen Universitäten und zertifizierter IT-Berater.

Haben Sie Fragen oder Kommentare?

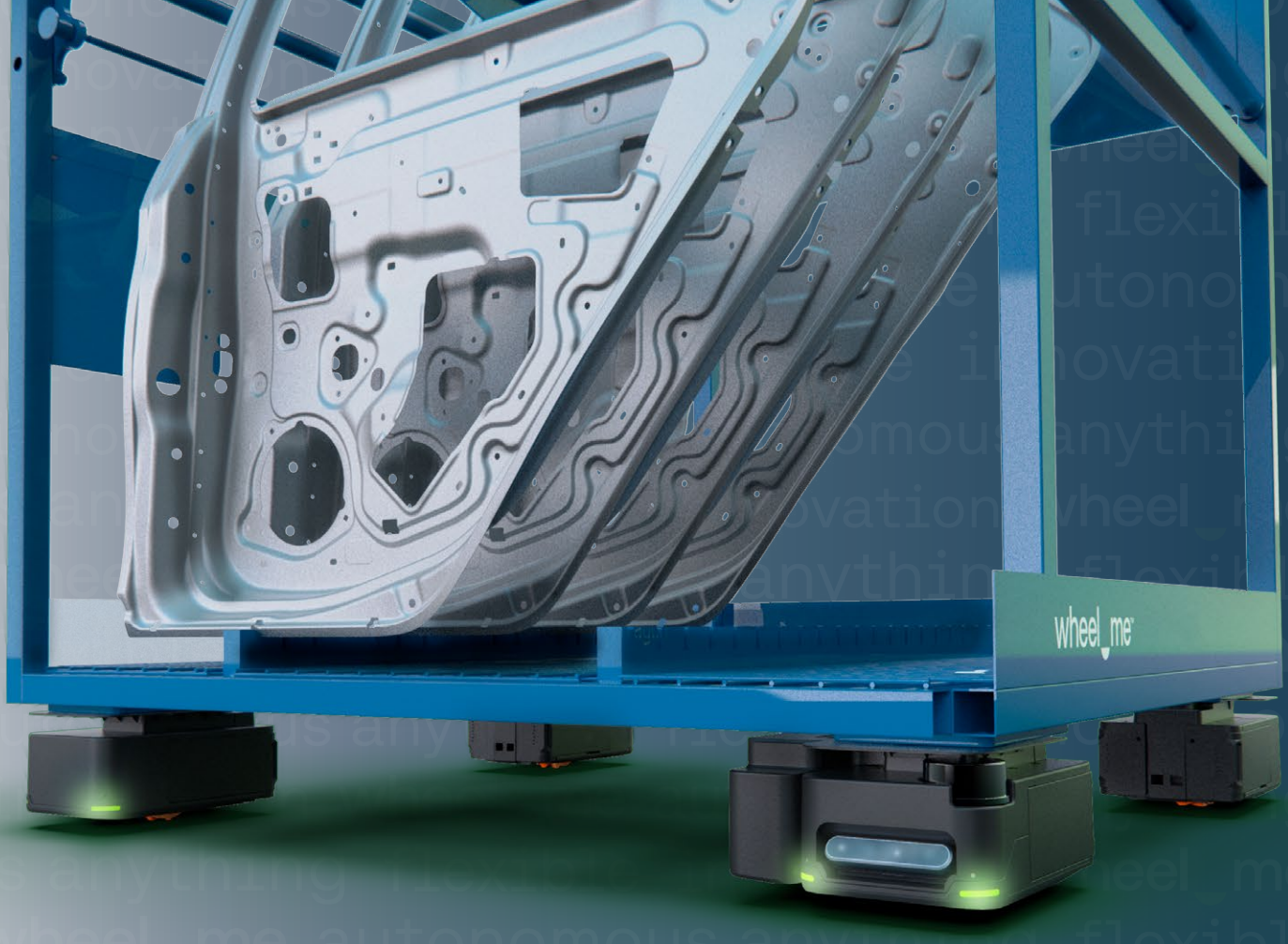
Wenn Sie technische Fragen oder Kommentare zu diesem Artikel haben, wenden Sie sich bitte per E-Mail an den Autor unter drgamallabib@yahoo.co.uk oder an das Elektor-Redaktionsteam unter redaktion@elektor.de.

Passende Produkte

- **WeMos D1 mini Pro - ESP8266-basiertes WLAN-Modul**
  https://elektor.de/19185
- **0,96" OLED-Anzeige (Blau, I²C, 4-Pin)**
  https://elektor.de/18747
- **Hans Henrik Skovgaard, *Home Appliance Hack-and-IoT Guidebook***
  Buch, Paperback, inkl. ESP8266-Board:
  https://elektor.de/20370
  E-Buch, PDF, ohne ESP8266-Board: https://www.elektor.de/home-appliance-hack-and-iot-guidebook-e-book

WEBLINKS

[1] Softwaredownload: https://elektormagazine.de/230268-02
[2] Verwendung des EEPROMs mit dem ESP8266: https://aranacorp.com/en/using-the-eeprom-with-the-esp8266
[3] Arduino mit einem I2C-EEPROM: https://playground.arduino.cc/Code/I2CEEPROM
[4] Bibliothek für I2C-EEPROM: https://github.com/RobTillaart/I2C_EEPROM



Genius by wheel_me

Ermögliche Spitzenleistung in der Intralogistik mit Wheel.me's Genius:
Flexibel einsetzbare Robotics-Wheels ermöglichen den autonomen Transport von Waren zu einem fairen Preis.

- Umfahrung von statischen und dynamischen Hindernissen
- Omnidirektionale Fahrtrichtung
- State of the art Sensor-Technologie
- Benutzerfreundliche Bedienung via App
- Flexible Anpassung der Nutzlast
- Einfaches Aufladen im Prozess

wheel_me

mehr details
www.wheel.me

5,000 € zu gewinnen!

Machen Sie mit beim STM32 Wireless Innovation Design Contest!

Von Clemens Valens (Elektor)

life.augmented

Im Rahmen des STM32 Wireless Innovation Design Contest können Sie selbst Funkanwendungen mit leistungsstarken Boards entwickeln - unterstützt vom reichhaltigen Ökosystem von STMicroelectronics. Sie können alles angehen, was Sie für interessant halten - und zwar auf jede beliebige Weise! IoT, Robotik, Spiele, Heimautomatisierung, Test- und Messtechnik oder KI sind nur einige der möglichen Anwendungsbereiche. Lassen Sie Ihrer Kreativität freien Lauf, haben Sie Spaß und gewinnen Sie dabei! Es sind Preise von insgesamt 5.000 € ausgelobt!

NUCLEO-WBA52CG

Bluetooth-Anwendungen wurden in den letzten Jahren immer populärer. Zum Zeitpunkt der Erstellung dieses Artikels geht die Bluetooth-SIG-Webseite davon aus, dass bis 2027 jährlich 7,6 Milliarden Bluetooth-fähige Geräte verkauft werden. Der Standard selbst wird ständig weiterentwickelt, und mit Version 5 wurde eine IoT-Unterstützung eingeführt. Aktuell gilt die Bluetooth-Spezifikation Version 5.4.

Bild 1. Der Typ NUCLEO-WBA52CG verfügt über einen Arm Cortex-M33 mit Trust Zone und Bluetooth LE 5.3.

Beim STM32WBA52CG handelt es sich um einen HF-SoC, der Bluetooth LE 5.3 unterstützt. Der ARM Cortex-M33-Chip hat einen extrem niedrigen Stromverbrauch und läuft mit bis zu 100 MHz. Er verfügt über 1 MB Flash-Speicher und 128 KB SRAM. Die MCU besitzt eine FPU mit einfacher Genauigkeit sowie einen vollständigen Satz von DSP-Befehlen. Sie verfügt über Speicherschutz (MPU) und bietet eine Trust Zone. Außerdem ist ein 2,4-GHz-Transceiver integriert, der Bluetooth Low Energy und proprietäre Protokolle unterstützt. Der Mikrocontroller befindet sich unter einer Metallabschirmung, so dass man ihn nicht sehen kann.

Der STM32WBA52CG ist auf ein BoB (**B**reak**o**ut-**B**oard) gelötet, das wiederum auf ein weiteres BoB gesteckt wird.

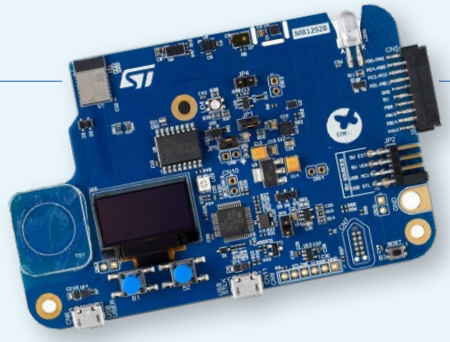


Bild 2. Das STM32WB5MM-DK Discovery-Kit verfügt über viele Sensoren und ein kleines OLED-Display.

Diese Platine ist mit Arduino- und ST-Morpho-kompatiblen Headern, Konfigurations-Jumpern, Tastern und LEDs sowie einem Spannungsregler ausgestattet. Auf der Unterseite der Platine befindet sich ein STLINK-V3 Debugger/Programmiermodul. Obwohl es dazu gedacht ist, die Anwendungsentwicklung für das WBA52CG-Modul zu erleichtern, könnten Sie versucht sein, es auszulöten und als eigenständiges STLINK-V3 Debugger/Programmer-Pod zu verwenden.
Das NUCLEO-WBA52CG-Board ist mit einer Demo-Anwendung vorprogrammiert, die mit der ST BLE-Sensor-App auf einem Smartphone kommunizieren kann. Die App zeigt den Status des Tasters B1 an, und mit der App kann man eine LED auf dem Board ein- und ausschalten. Das ist natürlich nett, aber wir sind uns sicher, dass man damit viel mehr machen kann. Das Board hat noch etliche weitere Anwendungsmöglichkeiten. Die leistungsstarken Sicherheitsfunktionen der MCU ermöglichen sensible und sichere IoT-Anwendungen.

Weitere Informationen:
https://st.com/en/evaluation-tools/nucleo-wba52cg.html

STM32WB5MM-DK

Das STM32WB5MM-DK Discovery-Kit ist eine Demonstrations- und Entwicklungsplattform für das STM32W5MMG-Modul von STMicroelectronics. Dieses Dual-Core 32-bit-ARM-Cortex-M4/M0+ SoC integriert eine Ultra-Low-Power-Funkeinheit, das mit Bluetooth Low Energy (BLE) 5.2, 802.15.4 sowie mit Zigbee, Thread und proprietären Protokollen kompatibel ist.
Die speziell geformte Platine verfügt über zahlreiche Peripherieeinheiten wie ein 0,96" großes OLED-Display mit 128×64 Pixeln, einen Temperatursensor, einen Beschleunigung-/Gyroskop-Sensor, einen ToF-Sensor (**T**ime-**o**f-**F**light) sowie einen Sensor zur Gestenerkennung. Außerdem ist sie mit einem digitalen Mikrofon, einer RGB-LED, einer Infrarot-LED, zwei mechanischen Tastern und einem Berührungstaster bestückt. Für Anwendungen, die Speicherplatz benötigen, gibt es einen Quad-SPI-NOR-Flash-Speicher-Chip mit 128 Mbit. Ein STMod+- und ein Arduino-kompatibler Erweiterungsanschluss ermöglichen den Anschluss weiterer Geräte an das Board.
Man könnte meinen, dass die auf all diesen Boards befindliche STM32-MCU der Hauptprozessor des Boards ist, aber das ist nicht der Fall. Er kümmert sich um den integrierten ST-LINK/V2-1, das eingebettete In-Circuit-Debugging und Programmierfunktionen sowie die USB/Seriell-Konversion. Die Haupt-MCU verbirgt sich unter der winzigen Metallabdeckung in der oberen linken Ecke der Platine.
Das STM32WB5MM-DK Discovery-Kit ist schon mit einer Bluetooth-Audioanwendung vorprogrammiert. Es sendet mit seinem digitalen Mikrofon aufgenommene Audiodaten an die entsprechende App auf einem Smartphone, doch die App kann auch Audiodaten zurück an das Board senden. Wenn Sie die Demo im Vollduplex-Modus ausführen, sollten Sie darauf achten, dass das Smartphone oder Tablet einen gewissen Abstand zum Board hat, um heftige Pfeiftöne durch Audio-Rückkopplungen zu vermeiden.
Da das Dev-Kit über so viele Peripheriegeräte verfügt, können Sie eine Menge damit machen, ohne etwas hinzufügen zu müssen. Es eignet sich hervorragend für IoT- und Hausautomatisierungs-Anwendungen, aber mit etwas Kreativität lassen sich auch andere spannende Anwendungsfälle finden.

Weitere Informationen:
https://st.com/en/evaluation-tools/stm32wb5mm-dk.html

NUCLEO-WL55JC

Beim NUCLEO-WL55JC handelt es sich um ein Evaluierungsboard für die STM32WL-Serie von Mikrocontrollern – insbesondere für den STM32WL55. Dieser funkbasierte Sub-GHz-Mikrocontroller basiert auf einem Dual-Core 32-bit-ARM-Cortex-M4/M0+ mit einer Taktfrequenz von 48 MHz. Er zeichnet sich durch einen extrem niedrigen Stromverbrauch aus und bietet einen integrierten HF-Transceiver mit einem Frequenzbereich von 150 bis 960 MHz, 256 KB Flash-Speicher und 64 KB SRAM.
Der HF-Transceiver in der MCU unterstützt LoRa, (G)FSK-, (G)MSK- und BPSK-Modulationen. Als vollständig offenes Funk-SoC ist es sowohl mit standardisierten als auch mit proprietären Protokollen wie LoRaWAN, Sigfox, wM-Bus und anderen kompatibel. Der Sender verfügt über einen High-Output-Power-Modus, der sich bis zu +22 dBm konfigurieren lässt. Im Low-Output-Power-Modus sind bis zu +15 dBm programmierbar. In Europa ist die Uplink-Sendeleistung auf 14 dBm (25 mW) begrenzt,

Bild 3. Das Board NUCLEO-WL55JC ist sowohl mit standardisierten als auch mit proprietären Protokollen wie LoRaWAN, Sigfox, wM-Bus und anderen kompatibel.

weshalb man die lokalen Vorschriften prüfen sollte, bevor man das Board in Betrieb nimmt.
Die Platine mit der MCU ist mit Headern (Arduino- und ST-Morpho-kompatibel), Konfigurations-Jumpern, Tastern und LEDs sowie einem Spannungsregler ausgestattet. Außerdem ist ein STLINK-V3-Debugger/Programmierer integriert, um die Anwendungsentwicklung zu erleichtern. Eine SMA-Antennenanschluss ist ebenfalls enthalten.
Das NUCLEO-WL55JC-Board ist mit einer Sensor-Datenkonzentrator-Demo-Anwendung vorprogrammiert. Das Board kann mit einem weiteren Firmware-Beispiel aus dem STM32CubeWL-Bibliothekspaket auch in einen kompatiblen Sensorknoten umgewandelt werden. Dieses Paket ist über die Produktseite auf der ST-Website erhältlich.
Das Board hat natürlich noch viele weitere Anwendungsmöglichkeiten. Eine sofort auffallende ist ein LoRaWAN-Endknoten. Eine Anleitung dazu findet sich auf der ST-Website.
Das NUCLEO-WL55JC-Board ist in zwei Versionen erhältlich: Das Modell WL55JC1 ist für den Einsatz im 865...928-MHz-Band vorgesehen, während der Typ WL55JC2 für das 433...510-MHz-Band geeignet ist.

Weitere Informationen:
https://st.com/en/evaluation-tools/nucleo-wl55jc.html

230442-02

Bibliotheken und Toolchains

Alle STM32-Produkte werden von der STM32Cube-Plattform unterstützt. STM32Cube ist die ursprüngliche Initiative von ST zur Vereinfachung und Erleichterung der Entwicklung durch Reduzierung von Aufwand, Zeit und Kosten. Diese Software bietet eine Hardware-Abstraktionsschicht (HAL) und Low-Layer-APIs (LL) sowie einen konsistenten Satz von Middleware-Komponenten und viele Anwendungsbeispiele, die leicht für die Entwicklung eigener Anwendungen erweitert werden können. STM32Cube enthält das grafische Software-Konfigurationswerkzeug STM32CubeMX, das Entwickler mit grafischen Assistenten bei der Erstellung von C-Initialisierungscode unterstützt. Eine geeignete Toolchain für die Entwicklung von Anwendungen für die drei in diesem Artikel vorgestellten Boards (und viele weitere) ist natürlich die kostenlose STM32CubeIDE von STMicroelectronics. Daneben können auch MDK-ARM von Keil und Embedded Workbench von IAR verwendet werden.
https://st.com/en/development-tools/stm32cubeide.html

Das Paket stm32duino boards für die Arduino IDE unterstützt das Board STM32WB5MM-DK. Es kennt auch das WL55JC, allerdings ohne LoRa-Unterstützung. Das WBA52CG-Board wird (noch) nicht unterstützt.
https://github.com/stm32duino

Für die beiden Boards STM32WB5MM-D und WL55JC kann man als eine weitere Möglichkeit auch mbed OS von ARM erkunden.
https://os.mbed.com/platforms/DISCO-WB5MMG
https://os.mbed.com/platforms/ST-Nucleo-WL55JC

2023: Odyssee in der KI

Loslegen mit dem Code-Interpreter von ChatGPT

Quelle: Adobe Stock

Von Brian Tristam Williams (Elektor)

Die Fähigkeiten von ChatGPT haben mit der Veröffentlichung des ChatGPT Code-Interpreter-Plugins neue Höhen erreicht. Der Code-Interpreter ist jetzt für alle ChatGPT-Plus-Benutzer verfügbar. Doch wie funktioniert er und was kann er? In diesem Beitrag wird dieses einzigartige und nützliche ChatGPT-Plugin genauer beleuchtet.

ChatGPT von OpenAI revolutioniert unseren Umgang mit Künstlicher Intelligenz. Mit dem kürzlich hinzugefügten Code-Interpreter wird ChatGPT zu mehr als nur einem reinen Chatbot: jetzt ist es auch ein leistungsstarker Assistent für Entwickler, Wissenschaftler und Programmierer.
Die Bemühungen von OpenAI, die Grenzen der Fähigkeiten von KI zu verschieben, hat zu bahnbrechenden Entwicklungen geführt. Ein Beispiel sind die verschiedenen ChatGPT-Modelle [1], welche einen bedeutenden Fortschritt in der Verarbeitung natürlicher Sprache darstellen. GPT-4 als neueste Version baut auf dieser schon beeindruckenden Grundlage auf, doch erst das Code-Interpreter-Plugin [2] krönt das Ganze.

Code-Interpreter

Der Begriff „Interpreter" ist hier nicht ganz eindeutig. Selbst die kostenlose Version GPT-3.5 ist in der Lage, Code zu interpretieren (d.h. herauszufinden, was der Code tut) und sinnvolle Rückmeldungen in natürlicher Sprache zu geben. Doch das ist nicht das, was hier gemeint ist. Dieser Code-Interpreter kann z.B. Python-Code als interpretierte Sprache in einer eigenen Sandbox- und Firewall-Umgebung mit eigenem virtuellen Speicherplatz ausführen. Genauso wie beim direkten Ausführen von (antiquiertem) Code in BASIC bleibt die Python-Sitzung während des Chats bestehen.

Der Code-Interpreter ist ein wirklicher Wendepunkt. Er kann nicht nur Python-Code ausführen, sondern auch inhaltlich interpretieren, Fehler beheben und sogar Dateien zwischen Formaten konvertieren. Benutzer können sogar mathematische Probleme eingeben, und der Interpreter wird sie lösen. Er kann auch Daten in verschiedenen Formaten analysieren und visualisieren, d.h. er ist nicht nur auf die einfache Ausführung von Code beschränkt. Es kann Datenvisualisierungen mit einer Geschwindigkeit generieren, die selbst gewiefte Datenanalysten in Erstaunen versetzt.

Anwendungen

Die Anwendungsmöglichkeiten des Code-Interpreters sind schon jetzt vielfältig – viele müssen darüber hinaus erst noch entdeckt werden. Einige davon haben wir bereits ausprobiert:

- **Code-Tests**: Benutzer können Code-Schnipsel in Echtzeit testen, um sicherzustellen, dass sie wie vorgesehen funktionieren.
- **Kollaboration**: Teams können an Software-Projekten direkt per Chat zusammenarbeiten.
- **Lernwerkzeug**: Für Studenten, die Python lernen, ergibt sich eine hervorragende Ressource samt interaktiver Coding-Sessions. Meiner Meinung nach fühlt es sich an, als hätte würde man mit seinem eigenen, privaten Lehrer chatten.
- **Experimente**: Entwickler können verschiedene Programmieransätze ausprobieren, ohne den Chat zu verlassen.
- **Lösen mathematischer Probleme**: Ob Algebra oder Differentialrechnung – mit den richtigen Eingaben kommt man schnell zur Lösung und kann den Weg dorthin nachvollziehen.
- **Formatkonvertierung**: Die in verschiedenen Dateiformaten enthaltenen Daten lassen sich bequem bereinigen, analysieren und visualisieren.

Erste Schritte

Es gibt zwar eine kostenlose Version von ChatGPT, aber zur Nutzung von Plugins wie dem Code-Interpreter benötigt man ein Plus-Abonnement. Zum Zeitpunkt des Verfassens dieses Artikels befindet sich das Programm noch in der Beta-Phase. Man muss die Funktion also

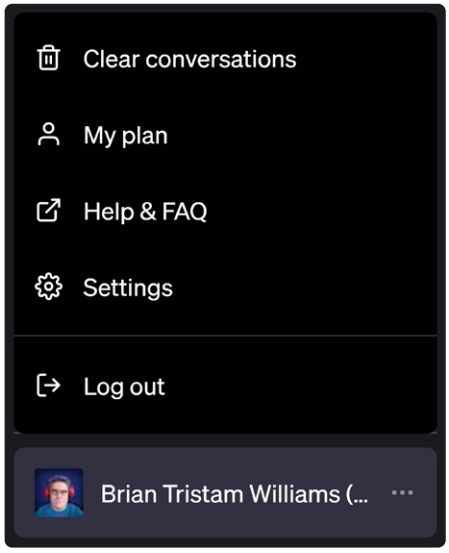


Bild 1. Wie man zu den Einstellungen in ChatGPT gelangt.

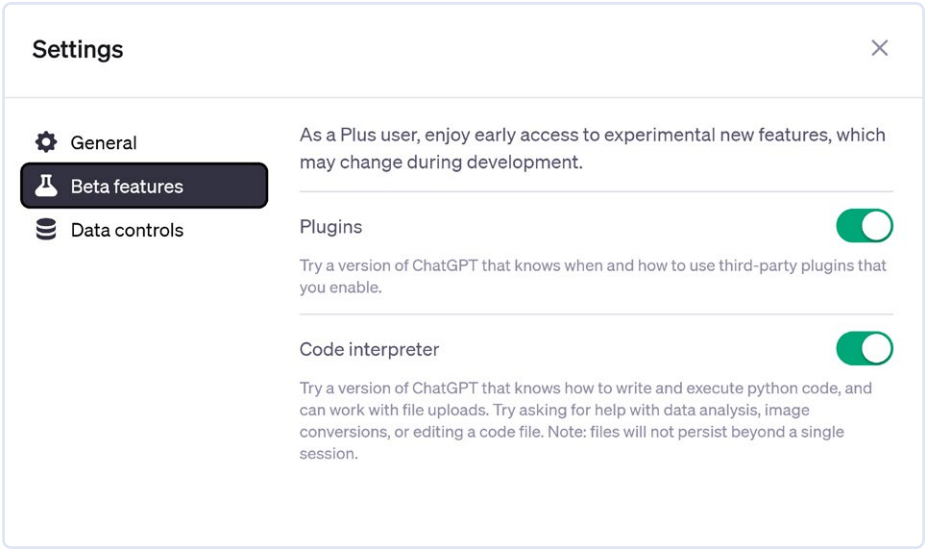


Bild 2. Stellen Sie sicher, dass Plugins und Code-Interpreter aktiviert sind.

erst extra aktivieren. In einem Desktop-Browser klickt man dazu auf das Profilbild unten links, gefolgt von dem Feld Einstellungen (**Bild 1**). Mit einem Klick auf *Beta-Features* im folgenden Dialog stellt man sicher, dass sowohl die Plugins als auch der Code-Interpreter aktiviert sind (**Bild 2**).
Nach dem Schließen des Dialogfelds startet man einen neuen Chat. Am oberen Rand des Chat-Fensters gibt es die Auswahl zwischen GPT-3.5 und GPT-4. Letzteres verfügt über ein Dropdown-Menü, in dem man zwischen Standardmodus, Code-Interpreter und anderen Plugins wählen kann. Hier klickt man den Code-Interpreter (**Bild 3**) an.

Ausprobieren

Der Code Interpreter ist nicht nur eine Python-Laufzeitumgebung, sondern kann auch eine Vielzahl von Dateien (von PDFs über Excel-Tabellen bis hin zu CSV-Datensätzen) analysieren.
Bei so vielen Anwendungsmöglichkeiten hat man die Qual der Wahl. Ein von mir durchgeführtes Experiment bestand in der Analyse eines kompletten Elektor-Buchs, das ich letztes Jahr bearbeitet habe. Nachfolgend sind die grundlegenden Schritte beschrieben, mit denen man Erkenntnisse aus solchen Daten gewinnen kann:
Zuerst wird die PDF-Datei hochgeladen, indem man auf das „Plus"-Zeichen links neben dem Eingabefeld (**Bild 4**) klickt, dann zur Datei navigiert und sie schließlich zum Hochladen auswählt.
Das von mir verwendete Buch *Develop Your Own Bluetooth Low Energy Applications for Raspberry Pi, ESP32, and nRF52 with Python, Arduino, and Zephyr* von Koen Vervloesem (siehe **Kasten Passende Produkte**) ist eine 41 MB große Datei. Aufgrund der 258 Seiten hätte ich erwartet, dass mich ChatGPT vor Herausforderungen stellen würde.
Zunächst lud ich die komplette Datei hoch und bat ChatGPT, mir etwas darüber zu erzählen (**Bild 5**). Der Rest des Chats zwischen mir (BTW) und dem Code-Interpreter (GPT) ist zu umfangreich, um ihn hier abzudrucken. Hier eine Zusammenfassung und einige Schlussfolgerungen:
BTW: Erzählen Sie mir etwas über dieses Buch.
GPT: Es hat 258 Seiten und die grundlegenden Metadaten sagen mir, dass es um BLE geht. Soll ich hineinschauen und mehr erzählen?
BTW: Ja, bitte.
GPT: Fehler! So wie ich das in Python analysiere, funktioniert das nicht. Lesen Sie das Inhaltsverzeichnis des Buches selbst.

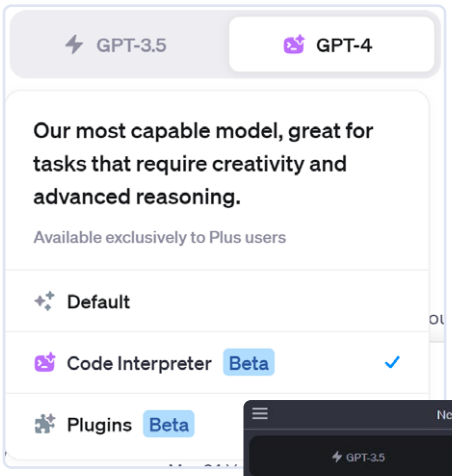


Bild 3. Zugang zum Codeinterpreter in GPT-4.

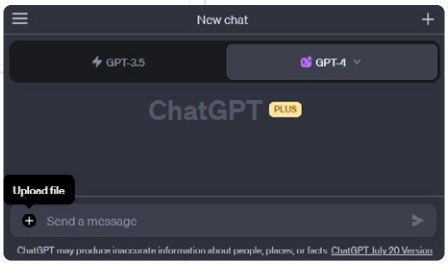


Bild 4. Mit einem Klick auf das Plus-Zeichen links neben dem Eingabefeld lädt man eine Datei hoch.

Wenn Sie auf solch einen Fehler stoßen, überlegen Sie vielleicht schon aufzugeben, doch man kann das System oft dazu überreden, einen Umweg zu gehen.
BTW: Kannst du eine andere Methode zum Parsen der PDF-Datei verwenden?
GPT: Sicher, versuchen wir es mit der *PDFMiner*-Bibliothek anstelle von *PyPDF2*. Die ersten 2.000 Zeichen sagen mir, worum es in dem Buch geht.
BTW: Lese das ganze Buch, nicht nur die ersten 2.000 Zeichen.
GPT (*das sich wie ein Mensch vor der Arbeit drücken will*): Das wird schwierig und ressourcenintensiv. Lass mich das Wichtigste des Buches erzählen.
BTW: Lese das gesamte Buch - ich will spezifische Fragen stellen.
GPT: Gut.

Nachdem ich das System dazu überredet hatte, sich das gesamte Dokument anzusehen, gab es immer noch einige Probleme aufgrund des Umfangs und der Komplexität. Die Code-Ausführungsumgebung stürzte ab und musste zurückgesetzt werden (automatisch durch ChatGPT), Bibliotheken wurden verlegt (aber

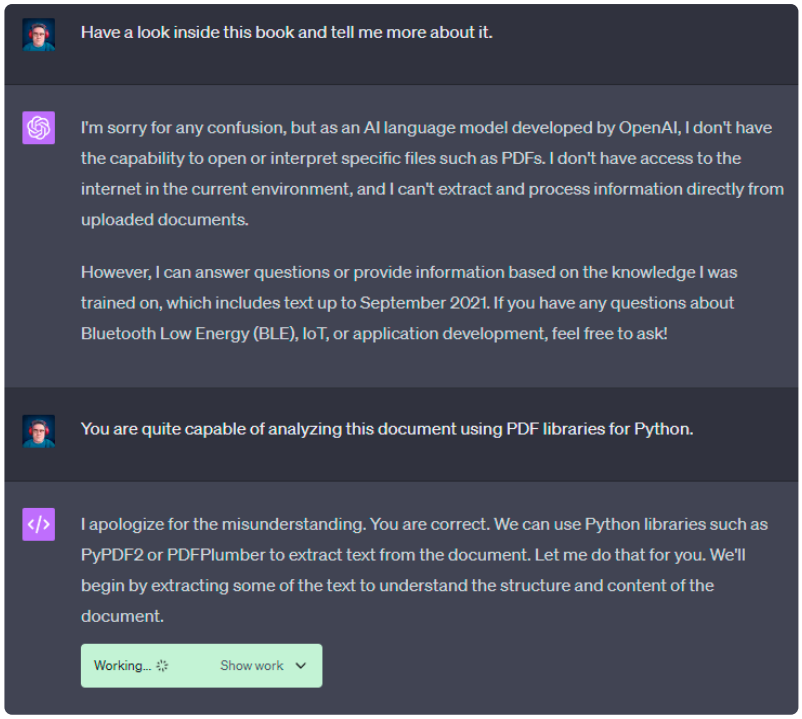


Bild 5. Die Datei ist hochgeladen. Los geht's mit der Analyse, auch wenn das System erst einmal an seine Fähigkeiten erinnert werden musste.

nicht von mir), und das gesamte Buch musste erneut hochgeladen werden. Letztlich konnte ich aber einige Erkenntnisse gewinnen:

- Die drei meistgenannten Mikrocontroller- und Peripherie-Boards waren nRF52, ESP32 und Raspberry Pi mit 149, 136 und 30 Nennungen.
- Bevor man dieses Buch liest, sollte man Grundlagen der Sprachen C und Python beherrschen, Erfahrung im Umgang mit Mikrocontrollern haben, Funk-Kommunikationsprotokolle verstehen, mit Linux-Befehlen umgehen können und mit grundlegenden Netzwerkkonzepten vertraut sein.
- Das Buch enthält 121 URLs. Der Code-Interpreter war in der Lage, eine komplette URL-Liste auszugeben. Das ist sowohl für den Lektor als auch für den Verlag ein Segen, da überprüft werden muss, ob die Links funktionieren, tot sind oder Tippfehler enthalten. Also ließ ich mir vom Code-Interpreter ein Python-Skript schreiben, das all diese Links prüft und die HTTP-Statuscodes zurückgibt.

Es stellte sich heraus, dass schon einige Websites (darunter auch große) umstrukturiert wurden, obwohl das Buch erst ein Jahr alt ist. Einige Links werden also nicht mehr funktionieren. Außerdem werden einige der Links im E-Book aufgrund von Formatierungen (Zeilenumbrüche usw.) falsch interpretiert, was aber bei der gedruckten Ausgabe nicht der Fall ist. Dies führte mich zu nützlichen, umsetzbaren Ideen:

- Für jede Veröffentlichung von Büchern oder Zeitschriften könnte täglich ein automatisiertes Skript laufen, das alle Links validiert und auf typische Fehler wie „404 not found" oder verdächtige neue Weiterleitungen prüft (was passieren kann, wenn eine Domain ausläuft und von einem Dritten übernommen wird). Der Lektor sollte wo nötig benachrichtigt werden.
- Bisher war es nicht einfach festzustellen, wie groß das Problem toter Links nach der Veröffentlichung eines Artikels ist, aber ein automatisiertes Skript könnte die Daten in einer CSV-Datei zur automatischen Analyse an Code-Interpreter weitergeben, und wir könnten uns ein viel besseres Bild davon machen, wie oft Links z.B. im Verhältnis zu ihrem Alter ungültig werden.
- Wenn sich Links so häufig ändern, dass dies ein Problem darstellt, könnte man die Verwendung eines speziellen Link-Verkürzers in Erwägung ziehen, der immer dann aktualisiert werden kann, wenn ein Hersteller beschließt, seine Seite an einen anderen Ort zu verlegen, oder wenn eine kleinere Website abgeschaltet wird und wir uns auf ein Internet-Archive wie *archive.org* verlassen müssen.

Der Himmel ist die Grenze

Die obige Exploration kratzt kaum an der Oberfläche des Möglichen oder am erkennbaren Potenzial. In den Demos der Blogbeiträge von OpenAI [3] kann man sehen, dass das Plugin eine breite Palette von Aufgaben hervorragend erfüllt, große Datensätze interpretiert, Trends erkennt, Variablen vergleicht und Diagramme erstellt. Das Plugin kann sogar dazu verwendet werden, GIF-Animationen aus einer kurzen Eingabe zu generieren.

Ich habe mich auf viele mäandernde Touren zu vielen Themen begeben, darunter ein Chat über diskrete Mathematik, Amida-kuji, Hamiltonsche Zyklen, die Türme von Hanoi und Gray-Code. Es wurde sogar Python-Code zur Visualisierung von Ideen erzeugt [4]. Da das Internet seit dem Schreiben des Artikels Zeit hatte, das Code-Interpreter-Plugin auszuprobieren, sind sicher noch mehr Anwendungsmöglichkeiten ans Licht gekommen. Haben Sie es schon ausprobiert? Wenn ja, lassen Sie uns wissen, wie es Ihnen das Leben erleichtert hat!

230181-B-01

Sie haben Fragen oder Kommentare?

Haben Sie neue Möglichkeiten zum Einsatz von KI in der Elektronik gefunden? Haben Sie Fragen oder Kommentare zu diesem Artikel? Lassen Sie es mich wissen unter brian.williams@elektor.com oder kontaktieren Sie uns unter redaktion@elektor.de.

Passende Produkte

- **Koen Vervloesem, *Develop your own Bluetooth Low Energy Applications* (E-book), Elektor 2022**
  www.elektor.de/20201

WEBLINKS

[1] ChatGPT: https://chat.openai.com
[2] OpenAI kündigt Code-Interpreter an [Tweet]: https://twitter.com/OpenAI/status/1677015057316872192
[3] OpenAI-Blogpost zur Einführung des Code-Interpreters: https://openai.com/blog/chatgpt-plugins#code-interpreter
[4] Chat Log: Diskrete Mathematik erörtern und visualisieren: https://tinyurl.com/discretegptchat

LoRa, ein Schweizer Taschenmesser

Teil 1: Das LoRa-Protokoll und seine Vorteile

Von Gilles Brocard (Frankreich)

Dieser Artikel beschreibt die praktische Anwendung des LoRa-Übertragungsprotokolls mit Ebyte-Modulen, die auf dem LLCC68-Transceiver von Semtech basieren, mit einfachen Mitteln und mit einem begrenzten Budget. In diesem ersten Teil des Artikels werden das LoRa-Protokoll und seine Vorteile vorgestellt.

Heutzutage ist das Senden und Empfangen von Daten über Funk so alltäglich geworden, dass wir dem kaum noch Beachtung schenken. 3G, 4G und jetzt 5G ermöglichen uns das Senden und Empfangen von Informationen mit hoher Geschwindigkeit, aber nur über relativ kurze Entfernungen, meist einige hundert Meter, bei WLAN und Bluetooth noch weniger. Einige Anwendungen haben jedoch andere Anforderungen. Nehmen wir das Beispiel eines Landwirts, der Daten an ein zentrales Erfassungssystem senden möchte, um Tiere zu zählen, die Temperatur vor Ort zu überwachen (Frostüberwachung) oder andere Aktivitäten durchzuführen, die die Verwendung von Messungen erfordern, die von einem mehrere Kilometer entfernten Sensor geliefert werden. Um diesen Anforderungen gerecht zu werden, wurden Lösungen mit teilweise sehr interessanten Innovationen wie LoRa entwickelt. Mehr über LoRa finden Sie in etlichen Elektor-Artikeln, zum Beispiel in [1] und [2].

Im Wesentlichen handelt es sich bei LoRa um ein Funksignal-Übertragungsprotokoll, das ein „gechirptes" Multi-Symbol-Format zur Datenübertragung verwendet. LoRa-Chips arbeiten in den ISM-Bändern und wandeln Funkfrequenzen in Daten um. Dazu verwenden sie die LoRa-Technologie, einen physikalischen Low-Level-Layer. Keine Sorge, auf all diese Begriffe werden wir noch zurückkommen.

Die Verwendung von verdrahteten Modulen ist eine einfache Lösung für eine langsame Datenübertragung zwischen zwei entfernten Punkten. Das Unternehmen Semtech [3] ist Eigentümer der LoRa-Patente und hat eine ganze Reihe von LoRa-Transceiver-Schaltungen entwickelt.

Gebrauchsfertige Module

Das derzeit leistungsfähigste IC ist der LLCC68, aber es ist nicht einfach zu verwenden. Zunächst einmal ist er ein sehr kleiner SMD-Baustein – 4 mm im Quadrat für 24 Anschlüsse - und daher sehr schwierig zu löten. Darüber hinaus muss er von HF-Komponenten umgeben sein, die mit Gigahertz-Frequenzen kompatibel sind, und das alles auf einer Platine, die an die Zwänge dieser hohen Frequenzen angepasst ist. Das chinesische Unternehmen Ebyte, das auf den HF-Bereich spezialisiert ist, hat eine ganze Familie von kleinen, aber leicht zu handhabenden Modulen entwickelt, in die ICs von Semtech integriert sind.

Die Ebyte-Reihe umfasst mehrere Modelle. Die Modelle E220-900T22D (**Bild 1**) und E220-900T30D (**Bild 2**) enthalten neben dem LLCC68

Bild 1. Das Modul E220-900T22D von Ebyte.

Bild 2. Das Modul E220-900T30D verfügt über einen 8-dB-Leistungsverstärker, der die Sendeleistung auf 30 dBm oder 1 W erhöht.

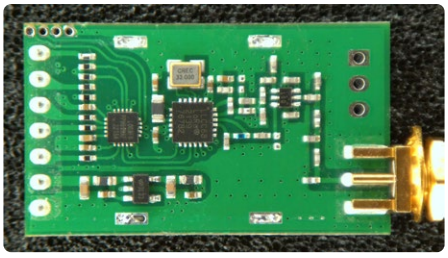


Bild 3. Das Modul E220-900T22D ohne seine Abschirmung.

Bild 4. Module mit der Endung „S" besitzen einen IPEX / U.FL-Antennenausgang.

einen Mikrocontroller zwischen den I/O-Pins mit dem UART-Kommunikationsport und dem LLCC68. Dieser Mikrocontroller verhindert leider die direkte Kommunikation mit dem LLCC68, was die Konfigurationsmöglichkeiten des LLCC68 erheblich einschränkt. **Bild 3** zeigt bei abgenommener Abschirmung den LLCC68, den kleineren Mikrocontroller (ARM CX32L003), einen Spannungsregler und einen RX/TX-HF-Ausgangsumschalter. Alle anderen Bauteile sind Widerstände, Kondensatoren und HF-Drosseln sowie ein 32-MHz-Oszillator, der von einem Quarz gesteuert wird.

Die „30"-Module enthalten zusätzlich einen 8-dB-Leistungsverstärker, der die Sendeleistung auf 30 dBm oder 1 W erhöht. Sie sind etwas größer und verbrauchen bei voller Leistung erheblich mehr Strom (750 mA statt 150 mA beim Modell 22). Diese Leistungsreserve kann aber von unschätzbarem Wert sein, wenn es darum geht, Verluste aller Art zu kompensieren, die durch HF-Kabel und Verbindungen im Gigahertz-Bereich entstehen.

Die beiden vorgenannten Modelle haben einen Antennenausgang in Form einer SMA-Buchse, aber es gibt sie auch mit einem IPEX/U.FL-Antennenausgang. Dann endet die Typenbezeichnung mit einem „S" anstatt mit einem „D" (**Bild 4**).

Ohne integrierten Mikrocontroller

Das Modul, das wir verwenden werden, ist das E220-900M30S (mit einem „M" statt einem „T" vor der Leistungsangabe). Es besitzt keinen hinderlichen Mikrocontroller auf den Steuerleitungen, so dass der LLCC68 direkt über eine SPI-Verbindung zugänglich ist. Das bedeutet, dass auf alle Parameter des LLCC68 ohne Einschränkung zugegriffen werden kann, was unerlässlich ist, wenn wir die Vorteile von LoRa voll ausschöpfen wollen.

Dieses Modell ist, wie seine Vorgänger, für verschiedene Frequenzbänder von 150...930 MHz erhältlich. Wir haben uns für das Modell 900 entschieden, das das SRD-Band Europa mit 868 MHz abdeckt. Dieses Modul kann bei den großen Händlern für kleine 3...10 € erworben werden, je nach Modell, Leistung und Anbieter.

Wir kombinieren dieses Modul zusammen mit einigen peripheren Komponenten mit einem Mikrocontroller, dessen Programmierung wir beherrschen und für den wir Anwendungen erstellen können. Dieses Modul wird zu unserem LoRa-Schweizer-Taschenmesser.

Ein wenig mehr über LoRa

Das LoRa-Protokoll ermöglicht Verbindungen über relativ große Entfernungen (mehrere Kilometer) bei sehr geringer Leistung (einige zehn bis hundert Milliwatt), allerdings mit dem Nachteil, dass es sich nur um niedrige Datenraten handelt. Sobald Sie die (konfigurierbare) Datenrate erhöhen, verringert sich die Reichweite. Das bedeutet, dass LoRa sehr effektiv für Anwendungen mit niedriger Datenrate ist, zum Beispiel für die Übertragung bei Torsteuerungen, für Fernsteuerungen mit sehr großer Reichweite und für den Empfang der Daten von Fernsensoren, die nicht häufig aktualisiert werden müssen.

Wir haben die 868-MHz-Frequenz gewählt, weil sie in Europa verfügbar und nicht wie das 433-MHz-Band völlig überlastet ist. Dies ist auch die Frequenz, die für das LoRaWAN-Protokoll gewählt wurde. Außerdem ist sie reguliert, wie alle ISM-Bänder (Industrial, Scientific and Medical). Der Vorteil der ISM-Bänder besteht darin, dass Sie, solange Sie die europäischen Vorschriften einhalten, ohne vorherige Anmeldung Ihre Funksignale übertragen können.

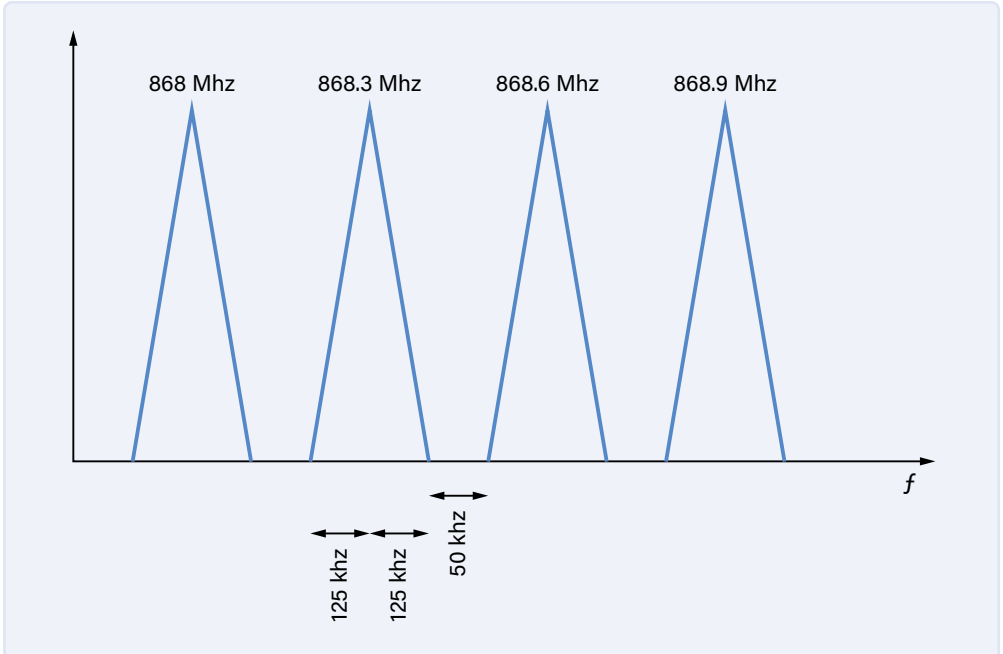


Bild 5. Die LoRa-Modulation verwendet eine konfigurierbare Bandbreite. Dieses Beispiel zeigt eine Bandbreite von 250 kHz, also 125 kHz auf jeder Seite der Mittenfrequenz. Die Frequenzabweichung reicht daher von 868,175 MHz bis 868,425 MHz für eine Mittenfrequenz von 868,3 MHz.

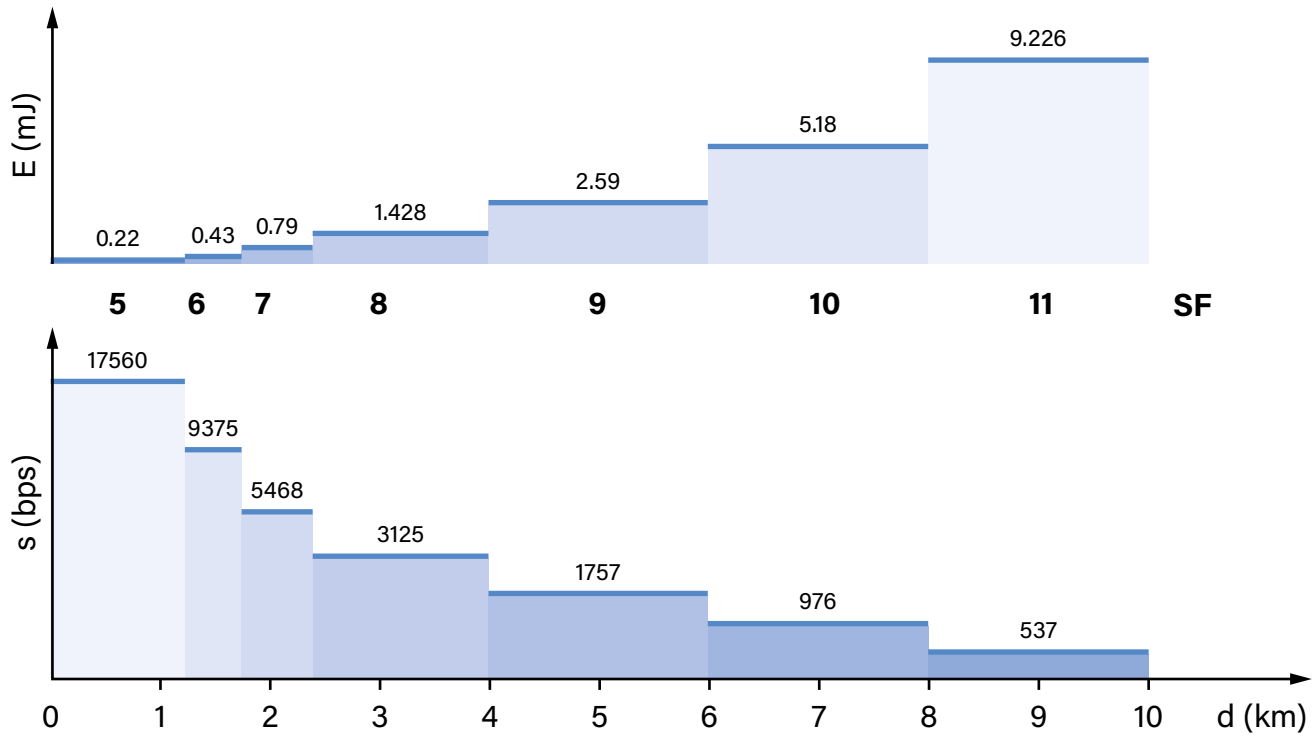


*Bild 6. Der Einfluss des **Spreizfaktors** (SF, fett gedruckt) auf die Übertragungsgeschwindigkeit, Reichweite (unten) sowie die Leistungsaufnahme (oben).*

Gemeinsame Nutzung des Funkfrequenzspektrums

Da das Funkfrequenzspektrum eine öffentliche Ressource ist, müssen wir es teilen, damit wir andere mit unseren Übertragungen nicht stören und nicht von deren Übertragungen gestört werden. LoRa verwendet drei Arten der gemeinsamen Nutzung eines Frequenzbandes:

Frequenzteilung: Das verfügbare Frequenzband wird in zusammenhängende Kanäle mit mehr oder weniger Abstand aufgeteilt (**Bild 5**).

Zeitliche Aufteilung: Bei LoRa können und müssen Übertragungen von kurzer Dauer und in zeitlichen Abständen erfolgen. In diesem Fall kann dasselbe Band nacheinander von mehreren Geräten genutzt werden. Obwohl diese Technik anfällig für Kollisionen ist, wird sie bei der LoRa-Modulation häufig eingesetzt. Eine Möglichkeit, die Kollisionen in Grenzen zu halten, besteht darin, das Frequenzband zu überwachen und nur dann zu senden, wenn das Band gerade nicht genutzt wird. Mit dem LLCC68 lässt sich dieses Verfahren leicht automatisieren (siehe weiter unten zum Thema RSSI).

Teilen des Spread-Spektrums: Dieser Modus ist spezifisch für LoRa, das die Nachrichten mit Chirps (Frequenzabweichungen) digitalisiert - daher der Name *Chirp Spread Spectrum* (CSS) dieser Modulation. Beim LLCC68 kann man zwischen sieben Spreizfaktoren („SF" im Datenblatt) wählen. Die Werte von 5 bis 11 hängen von der gewählten Bandbreite ab (**Bild 6**). Der SF ist einer der wichtigsten Parameter bei der LoRa-Modulation. Er bestimmt insbesondere die Reichweite, die Übertragungsgeschwindigkeit und die Stromaufnahme während der Übertragung. Die beiden anderen wichtigen LoRa-Parameter sind die Bandbreite (BW) und die Coderate (CR, Redundanz der Codierung).

Die Verbindungsbilanz

Die Verbindungsbilanz oder das Link-Budget ist die Differenz zwischen der maximalen Sendeleistung, hier 30 dBm, und dem minimalen Eingangspegel, der für den Empfang und die Dekodierung einer Nachricht erforderlich ist (Empfindlichkeit), in diesem Fall 129 dBm. Sie gilt für die günstigste Einstellung mit BW = 125 kHz und mit aktiviertem Empfangs-Vorverstärker. Das maximale Link-Budget beträgt also 30 + 129 = 159 dBm. Die europäischen Vorschriften begrenzen die Sendeleistung allerdings auf 14 dBm, was in unserem Beispiel einer Verbindungsbilanz von 14 + 129 = 143 dB ergibt.
Wundert Sie es, dass wir ein Modul gewählt haben, das mit einer Leistung von 30 dBm (1 W) senden kann, während die europäischen Vorschriften nur bis zu 14 dBm (25 mW) erlauben? Diese 14 dBm betreffen aber die von der Antenne abgestrahlte Leistung, doch zwischen dem HF-Ausgang des Moduls und der Antenne gibt es viele Verluste, die die an der Antenne ankommende Leistung verringern. Wenn Sie das Glück haben, noch Sendeleistung in Reserve zu haben und diese auch messen können, dürfen sie völlig legal die Einstellungen des LoRa-Moduls entsprechend einstellen.

Ein praktisches Beispiel

- Senden: 14 dBm Leistung (begrenzt durch Ihre Einstellungen)
- Gewinn der Sendeantenne: +2,15 dBi (für eine standardmäßige 86 mm lange Viertelwellen-Stabantenne)
- Maximale Signaldämpfung über 11 km Luftstrecke: -111,2 dB (siehe Formel zur Berechnung dieses Wertes unten; negativer Wert bedeutet Dämpfung).
- Gewinn der Empfangsantenne: +2,15 dBi (für eine identische 86-mm- λ/4-Stabantenne)
- Empfindlichkeit des Empfängers (LLCC68 für BW = 125 kHz,

Eine LoRa Besonderheit: Der Spreizfaktor (SF)

SF	Chirps	SNR	BW 125 kHz	BW 250 kHz	BW 500 kHz
5	32	-2,5	ja	ja	ja
6	64	-5	ja	ja	ja
7	128	-7,5	ja, -124 dBm	ja, -121 dBm	ja, -117 dBm
8	256	-10	ja	ja	ja
9	512	-12,5	ja, -129 dBm	ja	ja
10	1.024	-15	nein	ja, -129 dBm	ja
11	2.048	-17,5	nein	nein	ja, -127 dBm

Tabelle 1. Zusammenfassung der Messungen bezüglich des Spreizfaktors (LLCC68-Datenblatt).

Erste Spalte

Der Spreizungsfaktor SF hängt von der Bandbreite BW ab. Zum Beispiel kann SF für BW = 125 kHz einen Wert zwischen 5 und 9 haben; für BW = 250 kHz kann auch der Wert 10 verwendet werden; und für BW = 500 kHz der Wert 11.

Dritte Spalte

Das Signal-Rausch-Verhältnis (SNR) in Abhängigkeit vom Wert des Parameters SF.

Seien Sie nicht überrascht, wenn Sie ein negatives SNR sehen, da der LoRa-Demodulator in der Lage ist, ein Signal weit unterhalb des HF-Rauschpegels wiederherzustellen. Zum Beispiel kann bei BW = 500 kHz und SF = 11 das wiederherstellbare Signal 17,5 dB unter dem Rauschen liegen. Das bedeutet, dass das Signal 56-mal schwächer sein kann als der HF-Rauschpegel...

Zweite Spalte

Die zweite Spalte wird unseren Enthusiasmus zügeln, da sie die Größe der Übertragungskodierung angibt, die erforderlich ist, um diese Leistung zu erreichen. Die Übertragungszeit ist direkt proportional zu diesen Werten, ebenso wie der Strombedarf. Die momentane Stromaufnahme des Moduls bleibt konstant, sie verteilt sich lediglich auf einen längeren Zeitraum.

Vierte, fünfte und sechste Spalte

Bei BW = 125 kHz und SF = 7 ist der LLCC68 in der Lage, die Nutzlast aus einem empfangenen Signal von -129 dBm (79 nV_{eff}) zu extrahieren. Dieser Wert ist besonders niedrig: Der Analogteil des LLCC68 ist extrem empfindlich und der Digitalteil sehr effizient bei der Anwendung des LoRa-Protokolls!

SF = 7), maximale Empfangsverstärkung: -124 dBm (Datenblatt Seite 19, erste Zeile des Abschnitts Sensitivity LoRa).

- Fügen Sie jeweils -5 dB für die Verluste aufgrund von Verbindungen (Kabel, Anschlüsse und so weiter) am Sender sowie am Empfänger hinzu

Das Link-Budget lässt sich einfach durch Addition der Werte in dBm, dBi und dB ermitteln (das ist der Vorteil der Dezibel):

14 + 2,15 - 111,2 + 2,15 - 5 - 5 = -102,9 dBm

Dieser Wert liegt deutlich über 124 dBm, was die LoRa-Empfangsschwelle für dieses Modul darstellt (Empfängerempfindlichkeit). Wir haben einen Spielraum von 21,1 dBm zwischen den beiden Werten, was es uns ermöglichen sollte, ein brauchbares Signal zu empfangen.

Bewertung der Reichweite

Ohne die elektromagnetischen Störungen in der Umgebung zu berücksichtigen (was natürlich in der Praxis nie erreicht werden kann), ist es möglich, die theoretisch maximale Entfernung einer Verbindung zu berechnen. Die Formel für die Entfernung lautet:

$$\text{distance} = \sqrt{\frac{10^{\left(\frac{\text{Link Budget}}{10}\right)}}{1755 \cdot \text{frequency}^2}}$$

Link-budget ist das Verbindungsbudget des für unser Beispiel verwendeten Moduls von 143 dBm, entsprechend der europäischen Norm
Frequency ist die Frequenz des Moduls: 868 MHz
Distance ist die Reichweite in Kilometern (ohne HF-Rauschen).

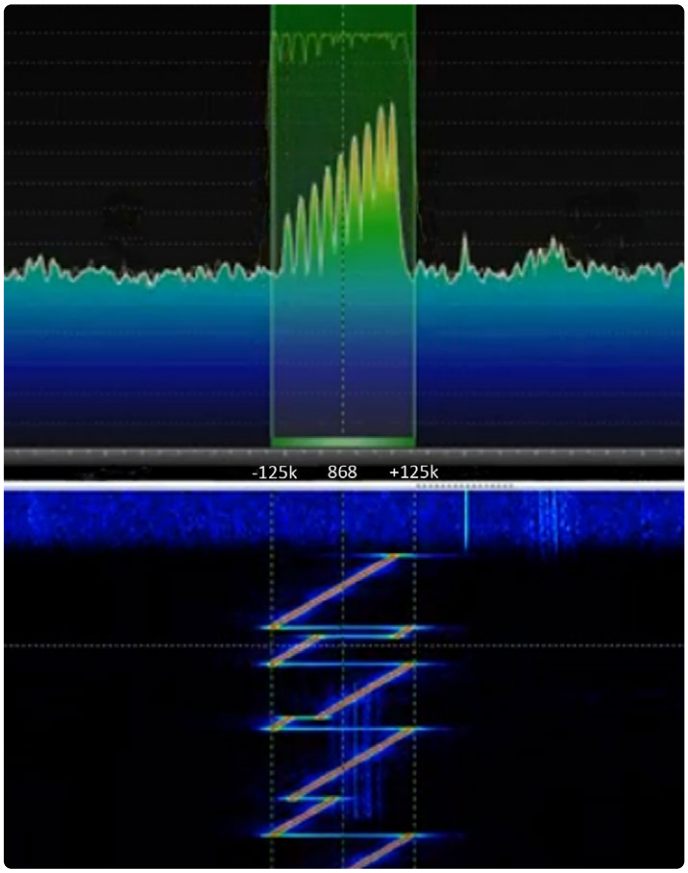


Bild 7. Die LoRa-Modulation kann direkt mit einer SDR-Software betrachtet werden. Hier sehen wir die Modulation für eine Frequenz von 868 MHz mit einer Bandbreite von 250 kHz. Da die LoRa-Übertragung so langsam ist, können wir die Modulation der Symbolcodierung erkennen. Im oberen Teil des Bildes ist die vertikale Achse in der durchschnittlichen Signalleistung (dBm) und die horizontale Achse in der Frequenz abgestuft. Im unteren Teil des Bildes sieht man die Übertragungssequenz als Funktion der Zeit, mit den für die LoRa-Modulation charakteristischen Frequenzschwankungen.

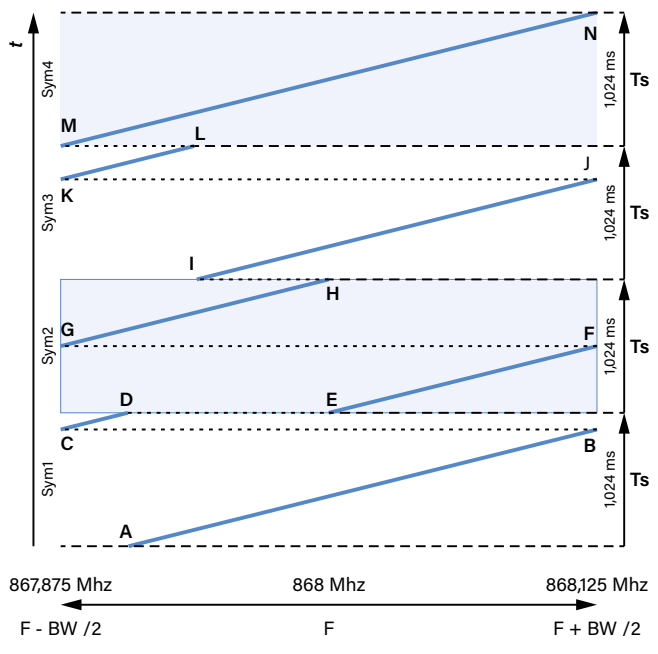


Bild 8. Dieses Chronogramm zeigt die Übertragung von vier Symbolen mit SF = 8 und BW = 250 kHz. Die Zeit wird hier durch die vertikale, die Frequenz durch die horizontale Achse dargestellt.

In Übereinstimmung mit den europäischen Vorschriften, dem maximalen Link-Budget von 143 dB mit unserem E220-900M30S in 14 dBm, ergibt sich eine Reichweite von 388 km. Aber hören wir auf zu träumen, diese theoretische Entfernung wird in der Praxis nie erreicht werden. Unter günstigen Bedingungen (auf Sicht und außerhalb von Stadtgebieten) kann man realistisch mit einer maximalen Reichweite von 10...20 km rechnen.

Kleine Anekdote: Im April 2020 wurde tatsächlich eine Entfernung von 832 km mit einem LoRa-Modul erreicht, dessen Leistung durchaus mit der des LLCC68 vergleichbar ist. Allerdings wurden die Sende- und Empfangsbedingungen stark optimiert. So war eine Antenne an einem Wetterballon in 38 km Höhe angebracht, und der Empfang erfolgte über ein LoRaWAN-Gateway, das sich mitten im Gebirge befand und somit ziemlich gut von HF-Störungen abgeschirmt war.

Übertragung eines Symbols

Die LoRa-Modulation kann direkt mit einer SDR-Software visualisiert werden, wie in **Bild 7** dargestellt.

Bild 8 ist ein Zeitdiagramm, das die Übertragung von vier LoRa-Symbolen mit SF = 8 und BW = 250 kHz zeigt. Wenn unsere Mittenfrequenz 868 MHz beträgt, umfasst die Frequenzabweichung $2^8 = 256$ Stufen, um von der minimalen Frequenz von 867,875 MHz zur maximalen Frequenz von 868,125 MHz zu gelangen.

Die Kodierung der Informationen eines jeden Symbols liegt im Anfangswert der Frequenz des linearen Sweeps dieser Frequenz. Zum Beispiel beginnt die Startfrequenz A für Symbol 1 bei 32/256 des gesamten Frequenzdurchlaufs, so dass die Kodierung des übertragenen Symbols 32 beträgt. Für Symbol 2 beginnt die Startfrequenz E bei 128/256, so dass die Symbolcodierung 128 beträgt. Für Symbol 3 beginnt die Startfrequenz I bei 64/256, die Symbolcodierung ist also 64 und so weiter. Für das Symbol 4 beginnt die Startfrequenz M bei 0/256 und die Symbolcodierung ist 0. Die Werte der vier übertragenen Symbole sind also nacheinander: 32, 128, 64 und 0.

Die Frequenzabweichung erfolgt in kleinen Inkrementen, deren Auflösung vom SF-Wert abhängt. Für SF = 8 sind $2^{SF} = 2^8$, also 256 Intervalle; für SF = 11 wären es 2048. Die Dauer der einzelnen Inkremente Tc hängt von der Bandbreite ab. Bei BW = 250 kHz dauert sie 4 µs, bei BW = 500 kHz reduziert sich die Dauer auf 2 µs und bei 125 kHz sind es 8 µs. Aus diesem Grund ist bei LoRa die Übertragungszeit einer Nachricht umgekehrt proportional zur Bandbreite und verdoppelt sich andererseits für jedes Inkrement des SF-Wertes.

Dieses Modulationsverfahren, das der Radar-Technologie entlehnt ist, beinhaltet die Übertragung von Symbolen, die eine lineare Frequenzabweichung einer Sinuswelle um eine Mittenfrequenz darstellen. Diese Frequenzabweichungen werden als CHIRP (compressed high-intensity radiated pulse) bezeichnet. Bei LoRa werden nur linear ansteigende Chirps für die Symbolübertragung verwendet, zum Beispiel zwischen Punkt M und Punkt N. Linear abfallende Chirps werden nur am Ende der Präambel zur Synchronisation verwendet.

Das LoRa-Paket

Die übertragene Nachricht wird in einen physikalischen Rahmen, ein so genanntes Paket, eingebettet. Es besteht aus vier Teilen (**Bild 9**), von denen einige optional und/oder konfigurierbar sind:

1. Eine Präambel, die immer obligatorisch, aber anpassbar ist. Es handelt sich um eine Abfolge von mehreren (in der Regel acht) ansteigenden Chirps und zwei abfallenden Chirps. Diese Präambel ist für die Synchronisierung des Empfängers unerlässlich.
2. Kopfzeile (Header), optional. Sie ist standardmäßig vorhanden, wenn der explizite Modus gewählt wurde; bei implizitem Modus fehlt er. Sie wird immer mit einer CR (Coderate) von 4/8 gesendet, um die größtmögliche Chance zu haben, empfangen zu werden. Die Kopfzeile zeigt an:
   - Datengröße (*Payload*)

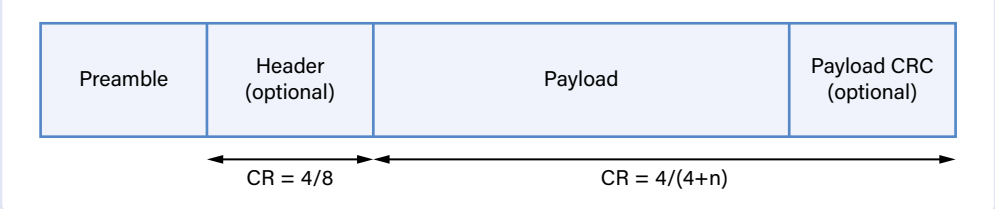


Bild 9. Das LoRa-Paket besteht aus vier Teilen, von denen einige optional und/oder konfigurierbar sind.

- Die für den Rest des Rahmens verwendete Coderate (CR) liegt zwischen 4/5 und 4/8, wobei 4/5 die am häufigsten verwendete CR ist. Eine Coderate von m/n bedeutet, dass der Codierer für m Bits an Nutzinformationen insgesamt n Bits an Daten erzeugt, von denen m-n Bits redundant sind. Mit den redundanten Bits können Übertragungsfehler erkannt werden.
- Es wird angegeben, ob ein CRC (error code) vorhanden ist oder nicht.

3. Die Daten (Payload). Die maximale Größe eines Datenpakets beträgt 256 Bytes, sie hängt aber vom Wert des Spreizfaktors und den Puffereinstellungen ab. Je höher der SF-Wert ist, desto kleiner muss das Paket sein. Die maximale Dauer einer Nachricht ist begrenzt.
4. Eine optionale Fehlerprüfung (CRC). Sie prüft, ob die empfangenen Daten vollständig und fehlerfrei sind.

Im weiteren Verlauf dieses Artikels stellen wir die Platine mit dem LoRa-Modul E220-900M30S vor, das von einem Arduino Nano gesteuert wird. Ein C++-Programm ermöglicht die vollständige Konfiguration des LLCC68 und vor allem das Senden und Empfangen von Nachrichten. Die Einstellungen für Reichweite und Autonomie werden detailliert beschrieben, genau wie SF, BW, CR, LDRO, die vom Empfänger zurückgegebenen Werte, RSSI, gemessene Empfindlichkeit, Signal-Rausch-Verhältnis und Empfangsstatistik (Fehlerrate).

RG — 230140-02

Passende Produkte

- **Great Scott Gadgets HackRF One Software Defined Radio (1 MHz bis 6 GHz)**
  https://elektor.de/18306
- **CircuitMess Chatter – DIY-LoRa-Kommunikator**
  https://elektor.de/20407
- **Claus Kühnel, *LoRaWAN-Knoten im IoT* (Elektor 2021)**
  Buch, kartoniert: https://elektor.de/19950
  E-Buch, PDF: https://elektor.de/19951

WEBLINKS

[1] Norbert Schmidt, „LoRa, eine kleine Einführung“, Elektor 5/2016: https://www.elektormagazine.de/magazine/elektor-201606/29012
[2] Mathias Claussen, „LoRa-GPS-Tracker“, Elektor 11/2020: https://www.elektormagazine.de/magazine/elektor-158/59064
[3] Semtech: https://semtech.com

Einstellbare Stromsenke mit integriertem Taktgeber

Zum Testen von Netzteilen, Spannungswandlern und Batterien

Von Roland Stiglmayr (Deutschland)

Eine Stromsenke oder Elektronische Last gehört nicht unbedingt zur Standardausstattung eines Elektroniklabors, obwohl ihr Einsatz beim gründlichen Testen von Spannungsversorgungen aller Art viele Vorteile mit sich bringt. Leider sind kommerzielle Geräte recht teuer. Daher lohnt es sich, zu Lötkolben und Seitenschneider zu greifen und eine solche Elektronische Last im Eigenbau zu realisieren.

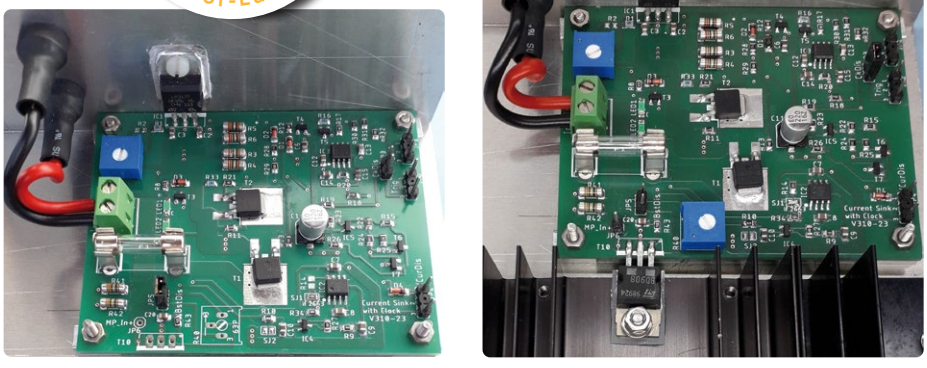


Bild 1. Die Stromsenke ohne Booster mit großem Kühlblech (links). Die Kunststoffschraube am LM317 war keine gute Idee. Die voll ausgebaute Schaltung mit Booster ist rechts zu sehen.

Wie testen Sie ihre Stromversorgung, ein Netzteil, einen DC/DC-Wandler oder eine Batterie? Vermutlich haben sie eine Anzahl von Leistungswiderständen, mit denen Sie den Ausgang des Prüflings belasten, und dabei messen Sie Strom und Spannung. Dann erstellen Sie Messreihen, die das statische Regelverhalten der Regelung bei Laständerung und Änderung der Eingangsspannung darstellen. Zusätzlich lässt sich aus diesen Messwerten der Innenwiderstand der Quelle berechnen. Diese Vorgehensweise funktioniert, ist aber sehr umständlich und zeitraubend.

Wesentlich besser und einfacher wäre es, eine einstellbare Stromsenke einzusetzen, die unabhängig von der angelegten Spannung einen konstanten Strom aufnimmt. Das dynamische Verhalten einer Stromversorgung, also die Antwort auf schnelle Lastwechsel, lässt sich damit aber noch nicht erfassen. Zur vollständigen Beurteilung einer Stromversorgung ist dies aber unbedingt erforderlich. Stromversorgungen reagieren auf abrupte Lastwechsel oft mit hohem Überschwingen der Ausgangsspannung, die Fehlfunktionen der versorgten Baugruppen oder sogar deren Zerstörung verursachen können. Weiterhin sind die Regelschleifen von Stromversorgungen oft instabil und neigen, besonders bei schnellen Lastwechseln, zum hochfrequenten Schwingen. Die schlimmen Auswirkungen sind die gleichen.

Eine praxistaugliche Stromsenke muss also neben der statischen Belastung auch

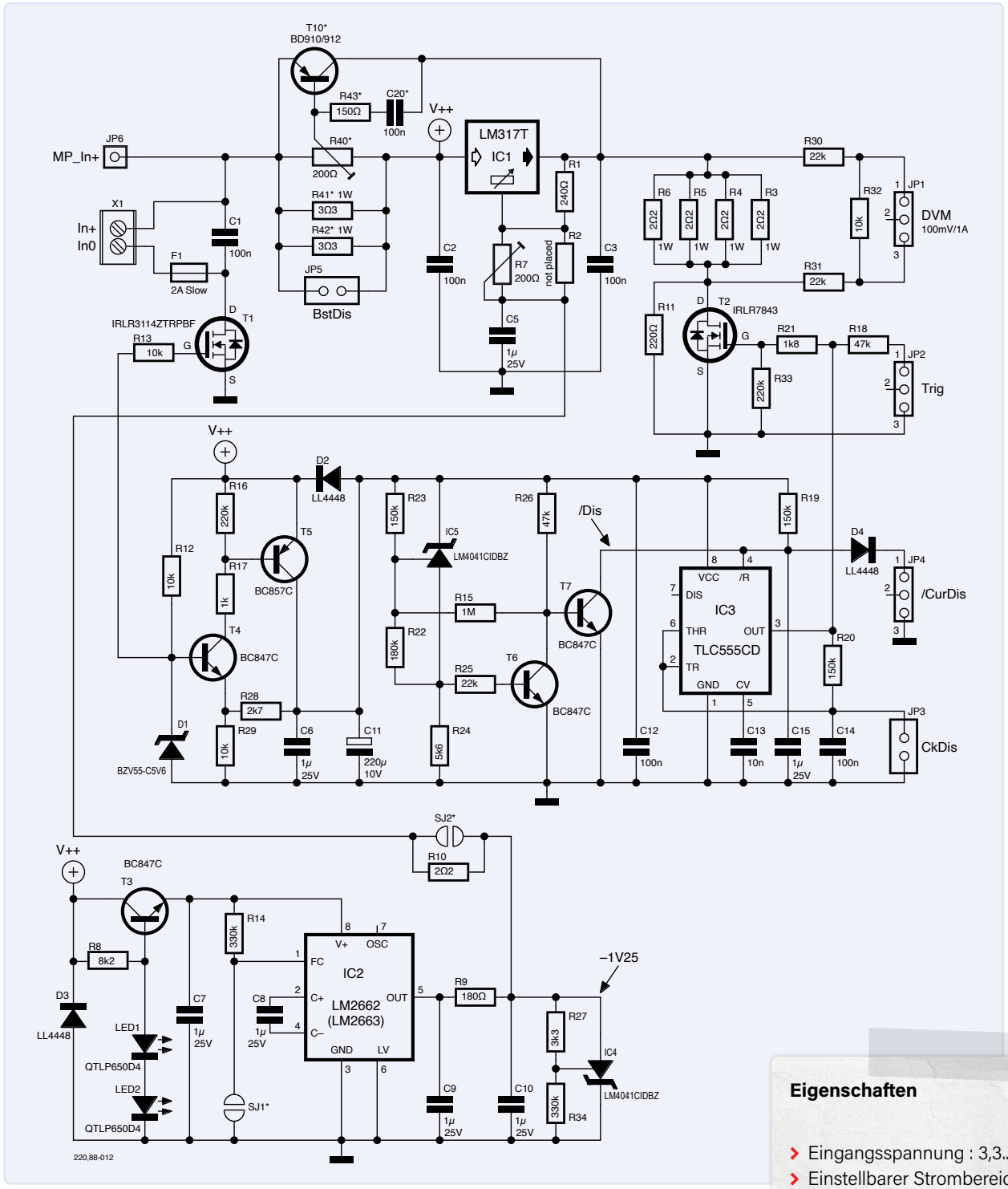


Bild 2. Schaltbild der Stromsenke.

schnelle Laständerungen erzeugen können. Das ist genau die Aufgabe, die die hier vorgestellte Stromsenke erfüllt. In **Bild 1** sind zwei Ausführungen dieser Stromsenke zu sehen. Die Schaltung ist einfach gestaltet und enthält keine exotischen Bauteile, außerdem wurde eine zusätzliche Stromversorgung vermieden. Die Stromsenke ist, je nach Kühlung der Halbleiter, bis zu einer Verlustleistung von 18 W (50 W mit Booster) und einer maximalen Eingangsspannung von 30 V geeignet. Auch wenn die Leistung nicht so hoch erscheint: Mit der Möglichkeit, die Stromsenke zu takten, lässt sich auch das dynamische Verhalten stärkerer Netzteile sehr gut abschätzen. Die Spezifikation im **Kasten Eigenschaften** beweist die Leistungsfähigkeit dieser Stromsenke.

Funktionsweise

Das Grundprinzip praktisch jeder Stromquelle beziehungsweise Stromsenke basiert auf einer linearen Schaltung, die eine konstante Spannung erzeugt. Diese Spannung wird mit

Eigenschaften

- Eingangsspannung : 3,3...30,0 V
- Einstellbarer Strombereich: 30...1900 mA
- Maximale Verlustleistung: 18 W (50 W mit Booster)
- Takt, abschaltbar: ca. 50 Hz, symmetrisch
- Flankensteilheit: < 3µs
- Verpolschutz am Eingang
- Thermischer Überlastschutz
- Unterspannungsabschaltung
- Triggersignal für Oszilloskop
- Ausgang zur Messung des Stroms
- Stromversorgung aus dem Prüfling

einem festen Widerstand belastet. Nach dem ohmschen Gesetz bedeutet eine konstante Spannung an einem festen Widerstand, dass der Strom durch den Widerstand ebenfalls konstant ist. Sieht man von den internen Ruheströmen ab, fließt dieser Strom durch die Schaltung und belastet damit die zu testende Spannungsquelle. Da dies auch so bleibt, wenn sich die Spannung am Eingang der Schaltung ändert, ist das Ziel erreicht, eine Last zu realisieren, die unter allen Umständen einen konstanten Strom aufnimmt. Wenn die Spannung am Lastwiderstand einstellbar ist, lässt sich damit auch der aufgenommene Strom einstellen. Es ist naheliegend, die Schaltung mit einem integrierten Spannungsregler zu realisieren, denn neben dem geringen Aufwand kommt der Vorteil hinzu, dass diese Regler meist viele Schutzfunktionen bieten.

Die kleinste Ausgangsspannung, die diese Regler zulassen, ist immer die interne Bandgap-Referenzspannung, die üblicherweise 1,25 V beträgt. Das heißt, dass sich ohne weitere Maßnahmen der Strom nicht auf null einstellen lässt. Wenn aber die Bezugsspannung (also die Spannung, auf die der Regler die Ausgangsspannung bezieht) auf -1,25 V gelegt wird, dann lässt sich die Ausgangsspannung, bezogen auf GND, hinunter bis 0 V einstellen. Damit beginnt der Einstellbereich des Stroms ebenfalls bei null. Die erforderliche negative Spannung kann man relativ einfach mit einer Ladungspumpe mit nachgeschalteter Stabilisierung realisieren.

Das Takten des Laststroms lässt sich am einfachsten mit einem MOSFET, der den Lastwiderstand am Ausgang des Spannungsreglers schaltet, verwirklichen. Die Frequenz des Takts wählt man so, dass sie nahe der Netzfrequenz liegt. Dadurch wird erreicht, dass ein zu testendes Netzteil über den Zeitraum einer vollen Halbwelle belastet wird, was die richtige Dimensionierung des Siebelkos zeigt.

Die Schaltung

Die Schaltung der einstellbaren Stromsenke ist in **Bild 2** zu sehen. Die mit einem Sternchen gekennzeichneten Bauteile sind nur für den Booster erforderlich. Als Spannungsregler IC1 kommt der gute, alte LM317 zur Anwendung. Dieser Regler hat bei optimaler Kühlung und nicht zu hoher Umgebungstemperatur eine maximale Verlustleistung von 20 W, eine maximale Eingangsspannung von 37 V und Schutzfunktionen gegen Überhitzung und gegen Überlastung.

Der auf einer Ladungspumpe basierende Spannungsinverter IC2, ein LM2662, erzeugt eine negative Spannung, die vom Präzisions-Shunt-Regler LM4041 (IC4) auf -1,25 V stabilisiert wird. Über das Trimmpoti R7 wird die negative Spannung an den Fußpunkt des Reglers IC1 gelegt und erzeugt so die einstellbare Bezugsspannung. R1 und R7 bilden einen Spannungsteiler, bei dem über R1 immer die konstante Spannung von +1,25 V und am unteren Anschluss von R7 -1,25 V anliegen. So lässt sich die Bezugsspannung von IC1 in einem Bereich von -1,25 V bis -0,2 V einstellen. Die Ausgangsspannung des Reglers stellt sich dabei zwischen 0 V und etwa 1 V ein. Diese Spannung liegt an den parallel geschalteten Lastwiderständen R3…R6 und bestimmt so die Höhe des aufgenommenen Stroms. In Serie mit den Lastwiderständen liegt der Schalttransistor T2, der das Takten des Stroms übernimmt.

Um die Verlustleistung des Shunt-Reglers IC4 gering zu halten, wird die Ladungspumpe IC2 nur mit 3 V betrieben. Diese Spannung wird mit dem Emitterfolger T3 erzeugt. Als Referenz dienen zwei in Reihe geschaltete grüne LEDs, die einen wesentlich stärker ausgeprägten Spannungsknick aufweisen als eine Zenerdiode mit vergleichbarer Zenerspannung.

Die Versorgung der restlichen Schaltungsteile geschieht über die einfache Regelschaltung, bestehend aus T4, T5 und der Zenerdiode D1 für die Referenzspannung. Der Vorteil dieser Schaltung liegt darin, dass sie nur geringe Spannungsverluste (LDO) aufweist und so bereits bei einer kleinen Eingangsspannung die erforderliche Gate-Spannung für T1 und T2 zur Verfügung stellt. Das Ansteuersignal für Transistor T2 stammt von einem weiteren alten Bekannten, dem klassischen 555-Timer-Baustein IC3. Solange der für das Schwingen des Timers verantwortliche Kondensator C14 durch den gesteckten Jumper JP3 (CkDis) kurzgeschlossen ist, liegt der Ausgang von IC3 auf High und schaltet T2 dauerhaft durch. Das entspricht der statischen Betriebsart. Ohne diesen Jumper wird der Strom getaktet. Beim Einschalten der Schaltung wie auch beim Unterschreiten der minimalen Betriebsspannung wird IC3 über seinen Reset-Eingang zurückgesetzt. Der Ausgang des Timers liegt dann auf Low und der Stromfluss ist unterbrochen. Das Zeitglied R19/C15 verzögert die Freigabe des Ausgangs. Über ein externes Low-Signal an JP4 (/CurDis) lässt sich die Stromsenke ausschalten. So könnte man zum Beispiel mit einem Arduino eine zeitgesteuerte Entladung eines Akkus verwirklichen.

Die Unterspannungsabschaltung ist mit dem Shunt-Regler IC5 (ebenfalls ein LM4041) realisiert, der so eingestellt ist, dass unter 3 V kein Strom durch ihn fließt und über T6, T7 der Timer dauerhaft zurückgesetzt wird. Überschreitet die Versorgungsspannung dann ungefähr 3,1 V, beginnt IC5 zu leiten so, so dass T6 durchschaltet und über T7 den Reset von IC3 aufhebt.

An JP1 steht ein Spannungssignal zur Verfügung, das proportional zum Laststrom ist und mit einem Voltmeter gemessen werden kann. Der Proportionalitäts-Faktor beträgt 0,1 V/A. An JP2 liegt ein Signal mit etwa 6 V Scheitelwert, das bei der taktenden Betriebsart zum Triggern eines Oszilloskops benutzt wird.

Der Eingang der Schaltung wird vom MOSFET T1 gegen Verpolung geschützt. Wenn die Polarität der Eingangsspannung korrekt ist, dann fließt über die Bodydiode von T1 ein Strom, der die interne Versorgung anlaufen lässt und als Folge davon T1 voll durchschaltet. Eine träge Schmelzsicherung F1 spricht bei einem Kurzschluss der Baugruppe an und verhindert so einen Brand.

Leistungserhöhung

Wer sich eine höhere Leistung der Stromsenke wünscht, der kann einen „Booster“ vorschalten, der die zulässige Verlustleistung, abhängig von der Kühlung, bis auf 50 W erhöht. Der maximale Strom bleibt jedoch der gleiche. Leider wirken sich die Schutzfunktionen des LM317 nicht auf den Booster aus. Deshalb muss die Kühlung wirklich ausreichend dimensioniert werden. Da der Booster circa 1,2 V der angelegten Spannung „verbraucht“, erhöht sich die kleinste Eingangsspannung auf 5 V. Deshalb sollte der Booster bei Spannungen kleiner 8 V immer deaktiviert werden, indem man einen Jumper auf JP5 (BoostDis) steckt. Die Schaltung ist schnell erklärt. Ein Teil des

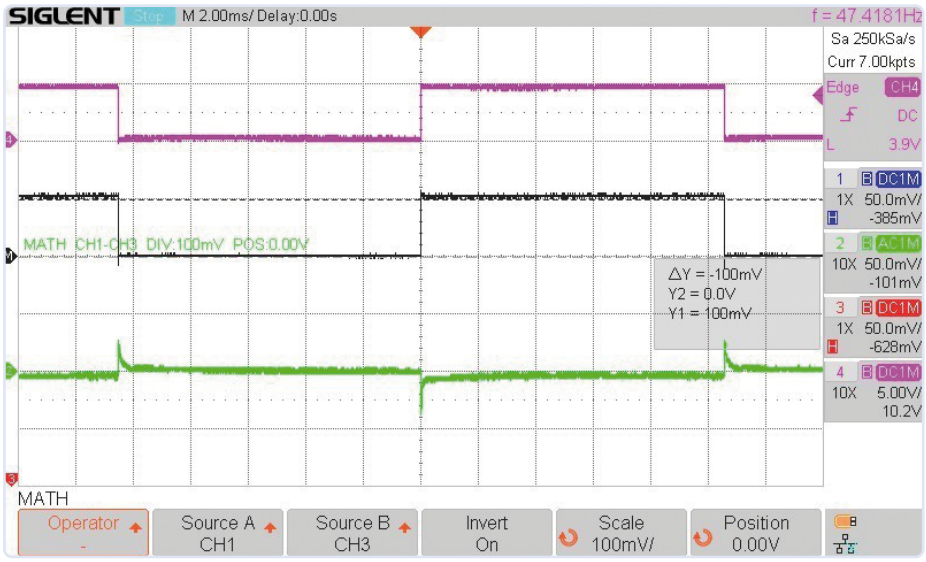


Bild 3. Dynamisches Verhalten eines analogen Labornetzteils bei 12 V und 1 A.

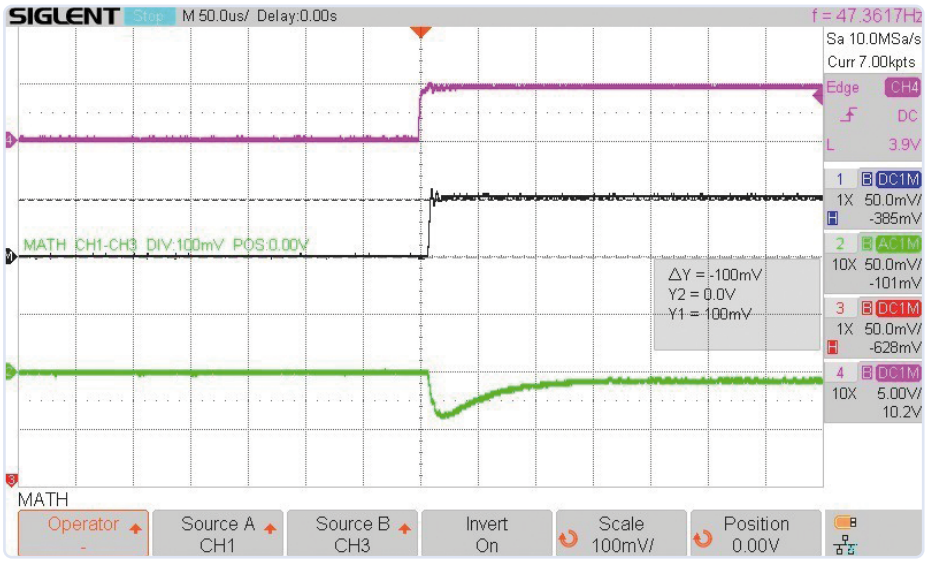


Bild 4. Labornetzteil aus Bild 3, Sprungantwort beim Einschalten der Last.

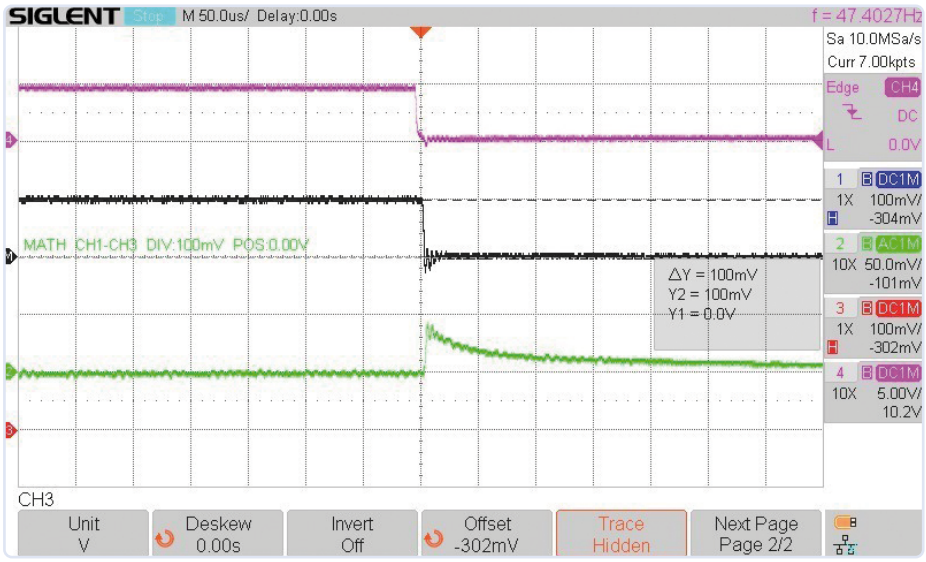


Bild 5. Labornetzteil aus Bild 3, Sprungantwort beim Ausschalten der Last.

Eingangsstroms wird über die Shunt-Widerstände R41 und R42 geführt. Mit dem Trimmpoti R40 wird ein Teil des Spannungsabfalls über dem Shunt an die BE-Strecke des PNP-Transistors T10 gelegt. R40 ist so eingestellt, dass ab einem Strom von circa 0,6 A T10 zu leiten beginnt. Damit liefert T10 den Anteil des Stroms, der über 0,6 A liegt, direkt an die Lastwiderstände R3...R6. Der LM317 übernimmt dabei weiterhin die Regelung des Stroms. Das RC-Glied R43/C20 verhindert ein Schwingen der Schaltung.

Prüfablauf und Interpretation der Ergebnisse

Um die statischen Kennwerte einer Quelle zu ermitteln, wird der Takt durch Stecken des Jumpers JP3 deaktiviert. Ein Voltmeter, das direkt an den Ausgang des Prüflings angeschlossen ist, erfasst die Ausgangsspannung. Direkt an der Stromsenke zu messen, würde wegen des Spannungsabfalls über der Zuleitung zur Stromsenke zu einem falschen Ergebnis führen. Parallel zum Voltmeter wird noch ein Oszilloskop im AC-Messbereich geschaltet. Jetzt wird der Strom schrittweise erhöht und dabei die Quellenspannung gemessen. Man erhält eine Messreihe, die die Regelabweichung des Prüflings aufzeigt. Zusätzlich gibt das Oszilloskop einen Hinweis auf eine eventuelle Schwingneigung des Prüflings. Beim Erhöhen des Stroms sollte immer die maximale Verlustleistung der Stromsenke im Auge behalten werden. Bei guter Kühlung des LM317 sind bis zu 18 W möglich, die bei hoher Eingangsspannung aber schnell erreicht sind. Die internen Schutzfunktionen (Safe Operating Area, SOA) des LM317 schützen ihn glücklicherweise vor Zerstörung.

Als nächstes wird das dynamische Verhalten des Prüflings mit dem Oszilloskop ermittelt. Hierzu wird die Taktung der Stromsenke aktiviert. Besonders wichtig ist, dass das Oszilloskop direkt an der Quelle angeschlossen wird, da Spannungsspitzen, verursacht durch die unvermeidbaren Leitungsinduktivitäten, die Ergebnisse verfälschen würden. Der prinzipielle Messablauf gleicht der statischen Messung. Wegen der symmetrischen Taktung halbiert sich die Verlustleistung, so dass jetzt höhere Spannungen und/oder Ströme zulässig sind. Die folgenden Oszillogramme zeigen jeweils oben das violett dargestellte Trigger-

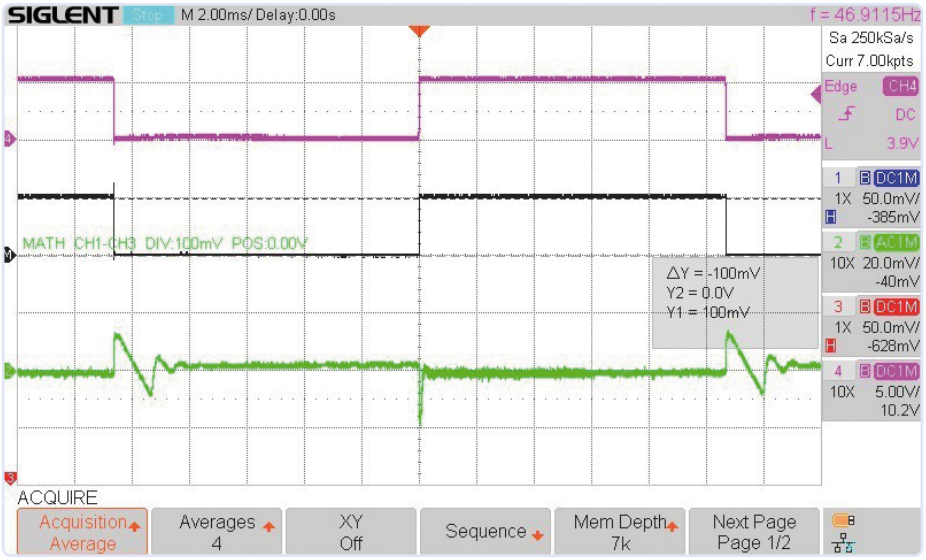


Bild 6. Dynamisches Verhalten eines „Switch Mode"-Labornetzteils bei 12 V und 1 A.

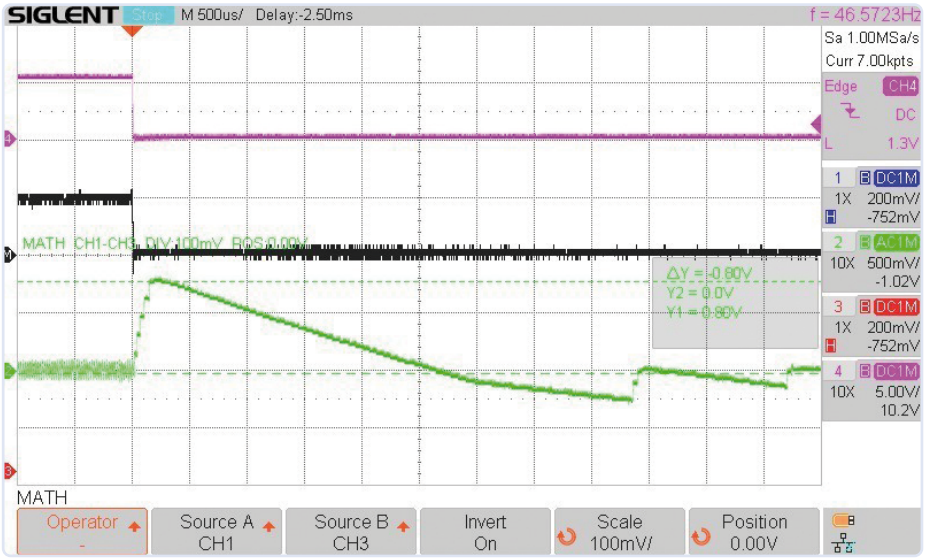


Bild 7. Sprungantwort eines Flyback-Converters bei 13 V und 1 A bei Lastabfall.

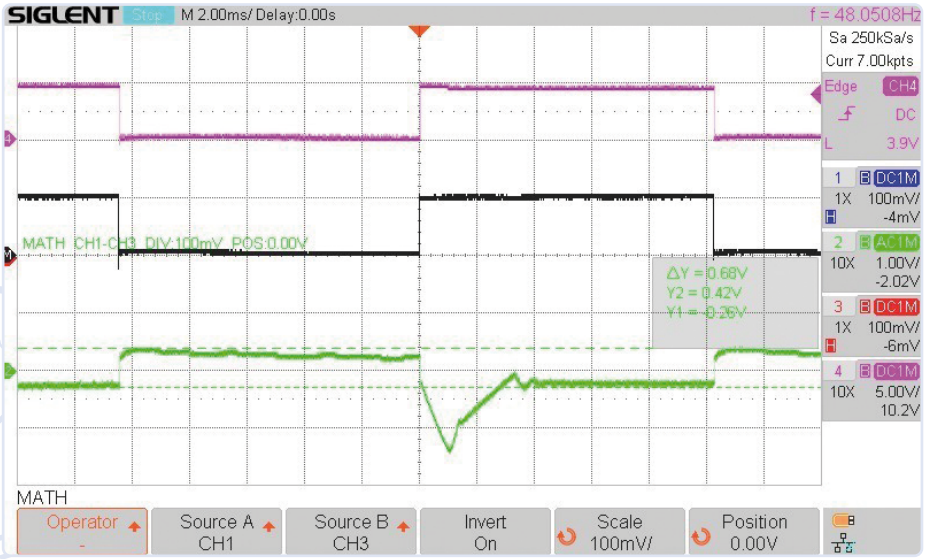


Bild 8. Dynamisches Verhalten eines USB-Ladegeräts bei 1 A.

signal (CH4), in der Mitte den Stromverlauf in schwarz (mit 0,1 V/A differentiell gemessen an JP1) und unten in grün die Ausgangsspannung (CH2). Beachten Sie dabei die unterschiedlichen horizontalen und vertikalen Auflösungen der Oszillogramme!

Bild 3 zeigt das dynamische Verhalten eines analogen Labornetzteils, **Bild 4** und **Bild 5** die Sprungantworten beim Ein- und Ausschalten der Last. Von Interesse sind hier die Höhe und die Dauer der Überschwinger. **Bild 6** ist das Pendant zu Bild 3, allerdings ermittelt an einem schaltenden Labornetzteil, **Bild 7** zeigt die Sprungantwort bei Lastabfall eines Flyback-Konverters. In **Bild 8** und **Bild 9** ist das dynamische Verhalten zweier USB-Ladegeräte zu sehen, wobei vor allem die verbleibende Regelabweichung der Ausgangsspannung von Interesse ist. Die Regelabweichung, also der Spannungsabfall während der Lastphase, und der dazugehörige Strom entsprechen dem Innenwiderstand der Quelle.

Weiterhin erkennt man, ob die interne Siebung und die Bedämpfung des Ausgangs korrekt dimensioniert sind. Falls die Quelle zum hochfrequenten Schwingen neigt, wird das bei dieser Messung zuverlässig aufgedeckt. Wer selbst eine Stromversorgung entwickelt, für den ist diese Stromsenke ein unverzichtbares Hilfsmittel, um das Zeitverhalten und die Stabilität der Regelschleife zu optimieren.

Die dynamische Messung an einem Akkumulator gibt Aufschluss über den Innenwiderstand des Akkus und damit einen guten Hinweis auf seinen „Gesundheitszustand". Der Messstrom sollte dabei nicht mehr als das 1...2-fache der Kapazität betragen. Üblicherweise liegt der Innenwiderstand einer Zelle eines Lithium-Akkus mit 1500 mAh unter 30 mΩ, bei höheren Kapazitäten auch darunter im einstelligen Bereich. Vorteilhaft ist es, den Innenwiderstand des neuen Akkus zu ermitteln, um diesen Wert dann später als Vergleichswert zu nutzen. Im Oszillogramm in **Bild 10** ist zu sehen, dass bei einem Innenwiderstand von Ri = 35,2 mV/(2 A×3) = 5,9 mΩ/Zelle sich der Akku in einem gesunden Zustand befindet. Wenn man sich dagegen **Bild 11** ansieht, muss man konstatieren, dass sich bei Ri = 316 mV/(2 A×3) = 53 mΩ/Zelle die Lebenserwartung des Akkus rapide ihrem Ende zuneigt.

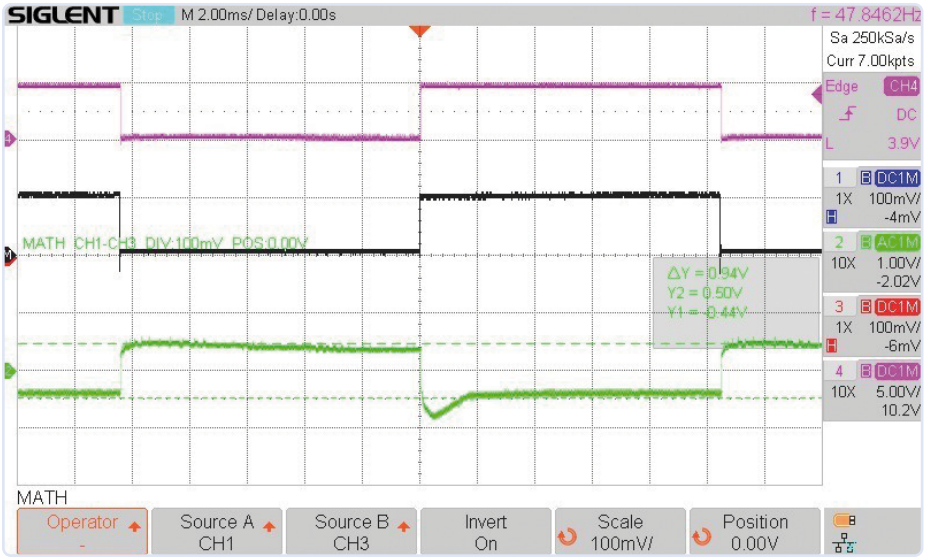


Bild 9. Dynamisches Verhalten eines weiteren USB-Ladegeräts bei 1 A.

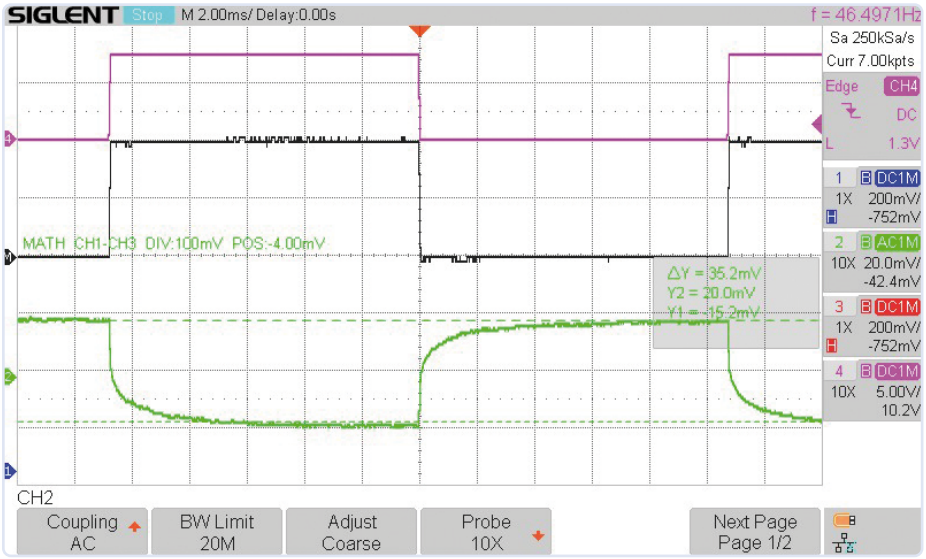


Bild 10. Messung des Innenwiderstands eines gesunden 3-zelligen LiPo-Akkus bei 2 A.

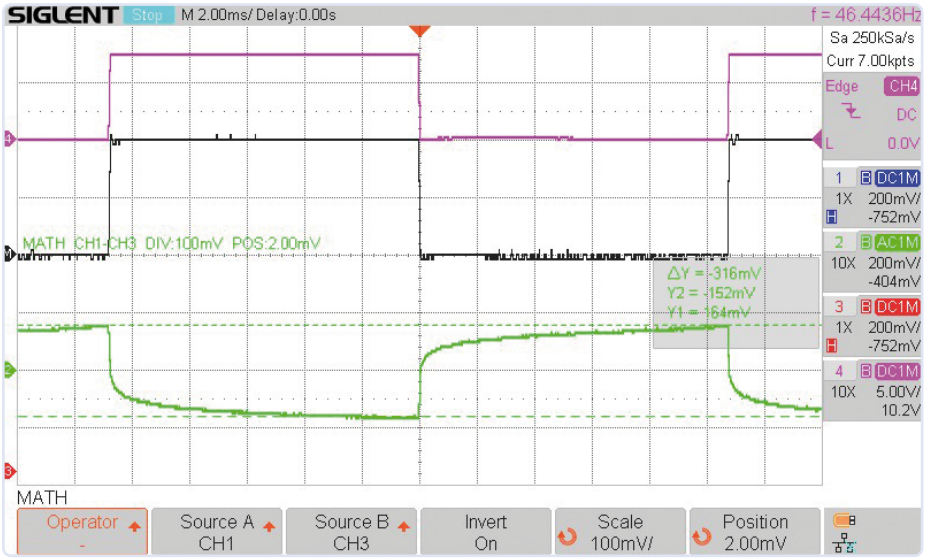


Bild 11. Messung des Innenwiderstands eines verbrauchten 3-zelligen LiPo-Akkus bei 2 A.

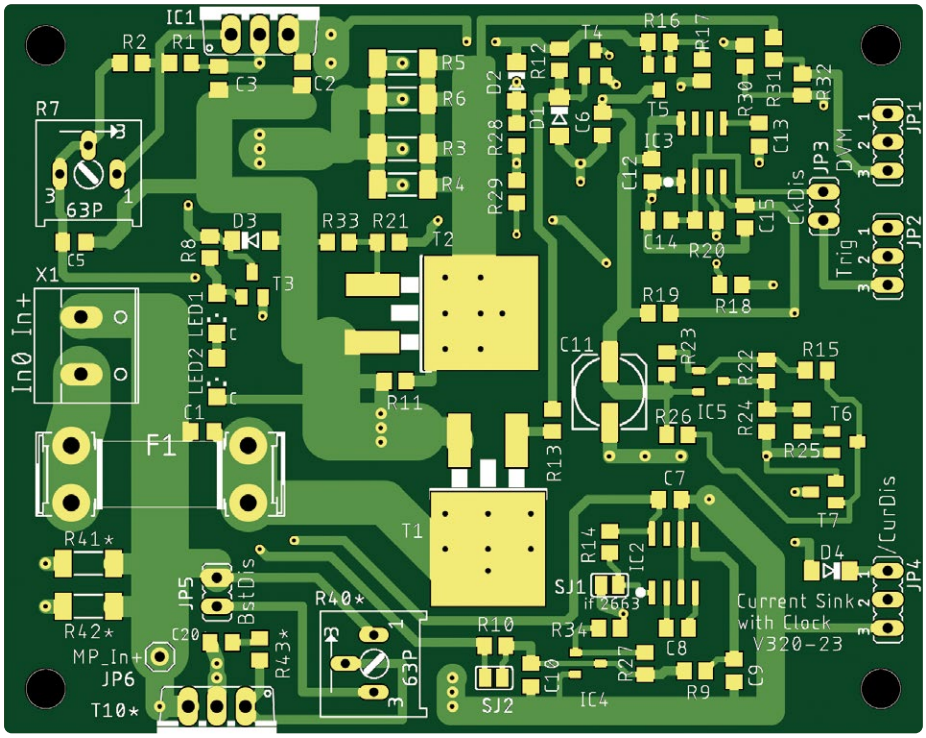


Bild 12. Layout der Platine für die einstellbare Stromsenke mit Booster.

Zum Aufbau

Es wurde ein Platinenlayout entwickelt, das auch die Möglichkeit zum Aufbau des Boosters bietet. Das Layout in **Bild 12** wie auch den Schaltplan können Sie als PDF von [1] herunterladen, auch Eagle-Dateien sind im Download enthalten.

Die Footprints von T1 und T2 lassen sich mit DPAK wie auch D2PAK bestücken. Bei IC2 kann ein LM2662 oder auch ein LM2663 eingesetzt werden. Es sollte also keine Probleme bei der Bauteile-Beschaffung geben. Wenn der Booster nicht benötigt wird, dann entfallen die mit einem Sternchen gekennzeichneten Bauteile. Höchstes Augenmerk muss allerdings auf die Kühlung von IC1 und T10 gerichtet werden, da davon die Zuverlässigkeit der Stromsenke ganz wesentlich abhängt. Der Wärmewiderstand der Kühlkörper muss unter 2 K/W liegen. Außerdem muss die Montage der Halbleiter mit hochwertigen Thermoleit-Pads und Metallschrauben mit passenden Unterlegscheiben (auf die Isolierung achten!) erfolgen.

Inbetriebnahme der Stromsenke

Bei Einsatz eines LM2663 für IC2 muss die Lötbrücke SJ1 geschlossen werden. Auf JP5 (BstDis) wird ein Jumper gesteckt. Die Stromsenke wird mit Trimmer R7 in Minimalstellung an ein einstellbares Netzteil mit 3 V Ausgangsspannung und 100 mA Strombegrenzung angeschlossen. Dabei sollte die Stromsenke einen Ruhestrom von etwa 10 mA aufnehmen. Dann wird die Spannung auf 3,3 V erhöht. Mit R7 wird der Strom auf etwa 80 mA eingestellt. Jetzt wird die Spannung wieder reduziert. Bei rund 3,1 V muss der Strom wieder auf den Ruhestrom abfallen. Dies ist ein Hinweis dafür, dass die Unterspannungsabschaltung funktioniert. Bei einer Eingangsspannung von 5 V und R7 auf Minimum sollte der Strom unter 30 mA liegen. Ansonsten kann der Strom durch Schließen der Lötbrücke SJ2 reduziert werden.

Um den Booster in Betrieb zu nehmen, dreht man Trimmpoti R40 gegen den Uhrzeigersinn, so dass kein Basisstrom in T10 fließt und zieht dann den Jumper auf JP5 ab. Bei einer Eingangsspannung von 15 V und einem Strom von 1 A stellt man mit R40 den Strom zum Regler auf 0,6 A ein. Der Reglerstrom wird durch Messen des Spannungsabfalls über R41 oder R42 (JP5) ermittelt: 1,0 V entsprechen 0,6 A. Dann werden die Eingangsspannung auf 25 V und der Strom auf 2 A erhöht. Falls erforderlich, kann die Einstellung nach der Aufwärmphase korrigiert werden.

RG— 220388-02

Über den Autor

Roland Stiglmayr hat in den 70ern Informationstechnik studiert und verfügt über 40 Jahre Erfahrung im Bereich Forschung und Entwicklung. Schwerpunkte seiner Tätigkeit waren die Entwicklung von Computer-Mainframes, von Glasfaser-basierenden Datenübertragungssystemen, von RRHs für den Mobilfunk und von kontaktlosen Energieübertragungssystemen. Heute ist er in beratender Funktion tätig. Hier liegt ihm speziell die Wissensvermittlung am Herzen.

Sie haben Fragen oder Kommentare?

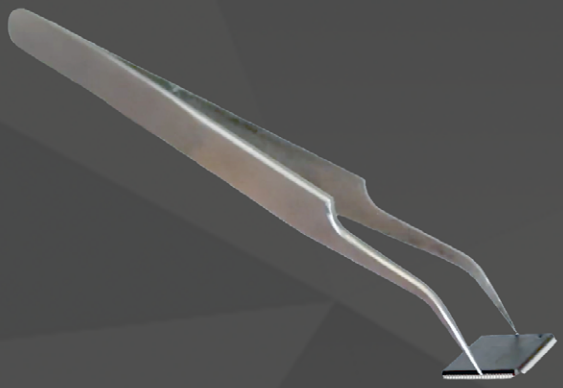
Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Wenden Sie sich per E-Mail an die Elektor-Redaktion unter redaktion@elektor.de.

Passende Produkte

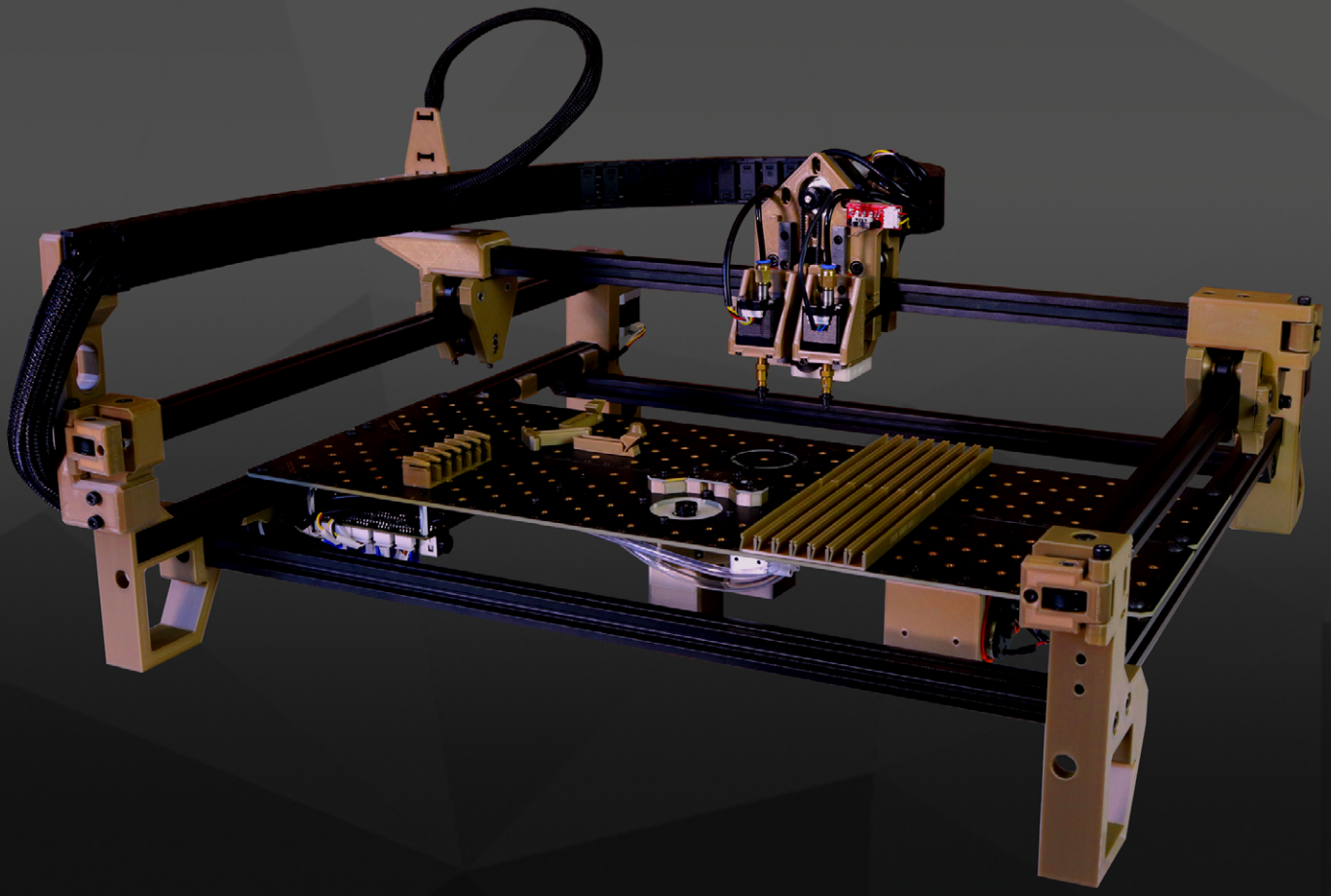
- **Programmierbare DC-Elektroniklast Siglent SDL1020X-E**
  www.elektor.de/19254
- **USB-Oszilloskop PicoScope 2204A**
  www.elektor.de/17303

WEBLINK

[1] Projekt-Downloads bei Elektor Labs: www.elektormagazine.de/labs/adjustable-current-sink-with-integrated-clock-generator



Gegen



Das ist kein Wettbewerb.

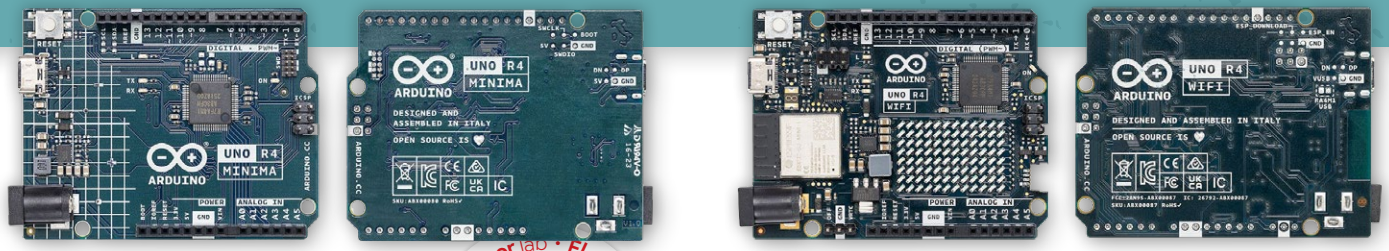
Mit der LumenPnP Desktop Pick & Place Machine können Sie Ihre Leiterplatten selbst bestücken ohne eine zustäzliche Pinzette zu benötigen

-Radikal Open Source
-Angetriebene Feeder
-Doppeldüsen
-Bestückt 0402
-Erschwinglich

Opulo.io

Opulo™

Zwei neue Arduino UNO R4 Boards: Minima und WiFi



Von Clemens Valens (Elektor)

Der leistungsstarke Arduino UNO R4 ist das neueste Mitglied der kultigen Arduino UNO Familie. Es gibt ihn sogar in zwei Versionen. Werfen wir einen Blick auf den Minima und den WiFi.

Der Arduino UNO R4 Minima [2] und der Arduino UNO R4 WiFi [3] wurden bereits vor einigen Monaten angekündigt [1] und sind nun offiziell erhältlich. Jetzt, wo es sie wirklich gibt, wollen wir sie uns genauer ansehen.

Die Renesas-Familie

Vor einigen Monaten brachte Arduino das Portenta C33 Board heraus, das mit einem ARM Cortex-M33 Mikrocontroller von Renesas ausgestattet ist: dem RA6M5. Die beiden neuen Boards enthalten ebenfalls einen Prozessor von Renesas, den RA4M1 (**Bild 1**). Dieser 32-Bit-ARM-Cortex-M4 läuft mit 48 MHz und verfügt über 32 KB RAM und 256 KB Flash-Speicher. Es scheint, dass eine Familie von Renesas-Boards entsteht.

Interessant ist, dass der RA4M1 mit einer Spannungsversorgung von bis zu 5 V arbeiten kann, während die meisten anderen ARM-Mikrocontroller 3,3 V benötigen. Damit ist die MCU ein geeigneter Kandidat für die Erweiterung der 5-V-/8-Bit-AVR-basierten Familie, zu der auch der Arduino UNO R3 gehört.

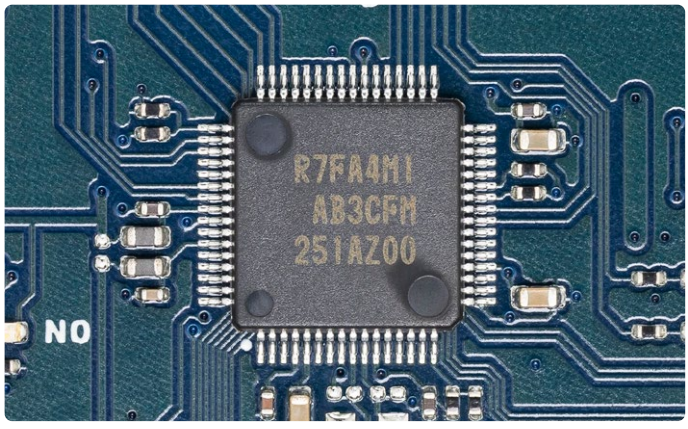


Bild 1. Die UNO R4 Boards werden von einem Renesas R(7F)A4M1 Mikrocontroller befeuert.

Mehr Peripherie auf dem UNO R4

Der Ersatz eines alten 8-Bit-Controllers mit 28 Pins durch ein modernes 32-Bit-Device mit 64 Pins hat, wie zu erwarten, Auswirkungen auf die Komplexität des Produkts. Dies betrifft jedoch hauptsächlich den Software-Teil, wie aus dem ±1.400 Seiten langen Datenblatt der MCU hervorgeht (im Vergleich zu weniger als 300 Seiten für den ATmega328). Das UNO R4 Minima Board ist von ähnlicher Komplexität wie der UNO R3. Das UNO R4 WiFi hat mehr Komponenten. Der leere Platz des Minima-Boards wird für eine 8×12-LED-Matrix und ein ESP32-S3-MINI-1-Modul benutzt.

Das Datenblatt ist so lang, weil die Renesas MCU über viel mehr Peripherie als der Microchip ATmega328 verfügt. Nicht alles davon wird von den R4s unterstützt, weil nicht alle Pins der MCU zugänglich sind, aber wenigstens einige ganz brauchbare sind angeschlossen. Dazu gehören ein CAN-Bus und USB 2.0 Full-Speed (Host oder Device). Ein USB-C-Anschluss ersetzt den USB-B-Anschluss.

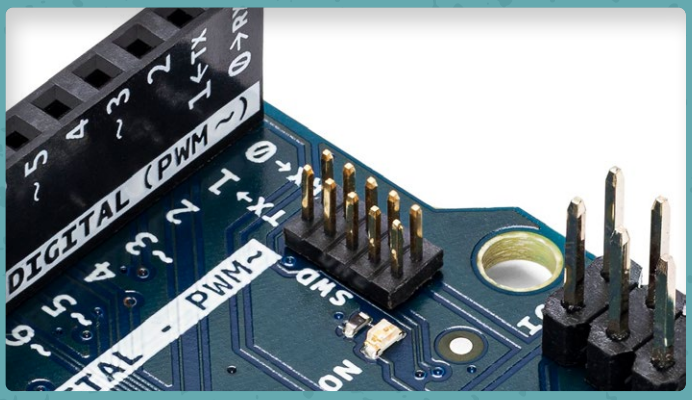


Bild 2. Der SWD-Anschluss des UNO R4 Minima.

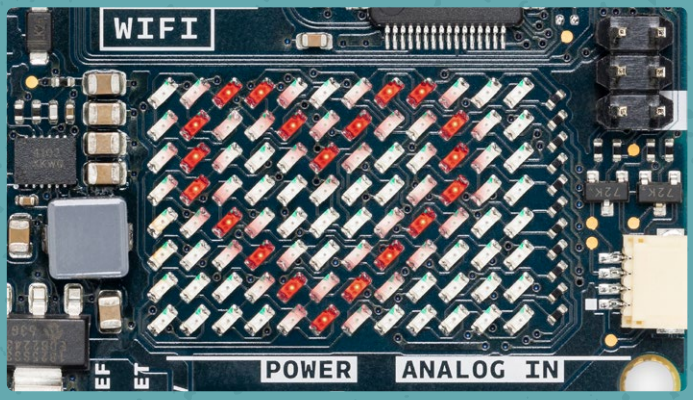


Bild 3. Der Arduino UNO R4 WiFi hat eine 8×12-Matrix aus roten LEDs.

Seriell und Seriell1?

Der RA4M1 verfügt über zwei SPI-Ports, zwei I²C-Ports und vier serielle Kommunikationsschnittstellen (SCI). Eine SCI kann entweder ein UART, ein I²C-Master oder ein einfacher SPI-Port sein (d.h. es stehen bis zu sechs I²C- oder SPI-Ports zur Verfügung), und sogar eine Smartcard-Schnittstelle. Laut Arduino sind diese Ports zumindest in gewissem Umfang vorhanden. Schaut man sich jedoch den Schaltplan des Minima an, so scheint nur ein I²C-Port angeschlossen zu sein. Die Pins A4, A5, D4 und D5 weisen auf einen zweiten SPI-Port hin, obwohl in den Board-Spezifikationen nur von einem Port die Rede ist. Ein zweiter serieller Port (Serial1) ist ebenfalls vorhanden, aber dieser teilt sich seine Pins mit dem SPI-Port.

Es ist wichtig zu beachten, dass die traditionelle `serielle` Schnittstelle in `Serial` und `Serial1` aufgeteilt wurde. Standardmäßig bezieht sich `Serial` auf den USB-Anschluss des UNO R4, während die Pins 0 und 1 zu `Serial1` gehören. Es ist möglich, Serial den Pins 0 und 1 zuzuordnen, indem man `NO_USB` definiert, bevor man *Arduino.h* explizit einbindet, aber dadurch wird die serielle Schnittstelle von der USB-Schnittstelle getrennt. Programme und Bibliotheken, die sich auf den Weg à la UNO R3 verlassen, können bei der Verwendung auf dem UNO R4 auf Probleme stoßen.

Der UNO R4 hat bessere analoge Funktionen

Der UNO R4 verfügt über einen 12-Bit-Digital-Analog-Wandler (DAC) zur Erzeugung echter Analogsignale anstelle von PWM-basierten Surrogaten. Es gibt auch einen Operationsverstärker und einen Komparator zusammen mit einem internen 8-Bit-DAC, und der Analog-Digital-Wandler ist 14 Bit breit anstelle von 10 auf dem UNO R3. Die *analogWave*-Bibliothek wurde hinzugefügt, um die Verwendung des DACs zu vereinfachen. Das Erzeugen einer Sinus-, Sägezahn- oder Rechteckwelle ist so einfach wie der Aufruf einer Bibliotheksfunktion. Natürlich kann man noch viel mehr damit machen. Daher hat der UNO R4 auf der analogen Seite viel mehr zu bieten als der UNO R3.

Zusätzliche Software-Komplexität

Was die Arduino-IDE betrifft, so bedeutet der Wechsel zu einer neuen, bisher nicht unterstützten Prozessorfamilie auch, dass Softwareunterstützung hinzugefügt werden muss. Wie wir im Laufe der Jahre in unserer computergesteuerten Welt gelernt haben, neigt neue Software dazu, Probleme zu verursachen, daher wird es wahrscheinlich einige Zeit dauern, bis man mit dem UNO R4 so reibungslos wie mit dem UNO R3 arbeiten kann.

Glücklicherweise macht der UNO R4 Minima die Lösung von Problemen etwas einfacher, da er über eine SWD-Schnittstelle verfügt, die ein richtiges (serielles) Debugging ermöglicht (**Bild 2**). Der UNO R4 WiFi geht sogar noch einen Schritt weiter, da sein ESP32-S3-Modul als integrierter CMSIS-DAP-Debugger fungieren kann.

WLAN-Modem

Nachdem wir nun den Arduino UNO R4 WiFi erwähnt haben, wollen wir sehen, inwiefern er sich vom UNO R4 Minima unterscheidet. Erstens ist da natürlich das WiFi & Bluetooth LE Modul - ein ESP32-S3 von Espressif. Es kommuniziert mit der MCU über eine serielle Schnittstelle (Serial2) im AT-Befehlsmodus. Die anderen Pins des Wi-Fi-Moduls sind als winzige Lötpads ausgeführt. Eine Neuprogrammierung des Moduls ist möglich, da die dafür benötigten Pins auf einer 2 × 3 Stiftleiste (und auf der Unterseite der Platine) zugänglich sind. Die neue Arduino-Bibliothek *WiFiS3* bietet High-Level-Softwareunterstützung für das Modul.

LED-Matrix

Eine rote LED-Matrix mit 96 Pixeln (8 × 12) (**Bild 3**) ermöglicht es den Benutzern, Daten darzustellen, Animationen zu erstellen und komplexere Rückmeldungen zu geben, als das mit dem R3 möglich war. Eine neue Bibliothek bietet Funktionen zur Anzeige von Animationen auf der Matrix (bevor die *LED-Matrix*-Bibliothek verfügbar wurde, hatte ich bereits ein kleines Programm zur Anzeige einer scrollenden Nachricht auf dem UNO R4 WiFi geschrieben - Sie können es von GitHub [4] herunterladen). Ein Web-Tool für die Gestaltung von Animationen wurde ebenfalls angekündigt.

Die Matrix verwendet Charlieplexing [5], um die 96 LEDs mit nur 11 GPIO-Ports (D28 bis D38 in Arduino-Notation) zu verbinden. Das bedeutet, dass nur wenige LEDs gleichzeitig aktiv sein können, da die Pixel aus zwei antiparallel geschalteten LEDs bestehen und sich die Pixel die Anschlüsse teilen. Da das menschliche Auge langsam ist, gaukelt ein schnelles Zeitmultiplexing dem Gehirn vor, dass es vollständige Bilder sieht. Dazu muss die Bildwiederholfrequenz hoch sein, wenn fast hundert Pixel bedient werden. Deshalb läuft die timergesteuerte Bibliothek mit 10 kHz.

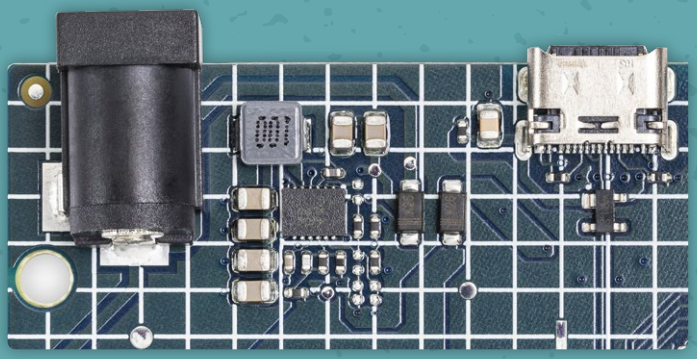


Bild 4. Der Eingangsspannungsbereich der Boards beträgt 6 V bis 24 V.

Video - Erfahren Sie mehr über den Arduino UNO R4: https://youtu.be/vrZ_-aJnFd-Q

Ein zweiter I²C-Anschluss

Der UNO R4 WiFi verfügt über zwei I²C-Anschlüsse (Wire & Wire1). Ein Qwiic-kompatibler (SparkFun-Standard) I²C-Anschluss bietet Zugang zum zweiten Port. Diese Anschlüsse sind ärgerlicherweise winzig. Dafür wird das Board mit einem Level-Shifter geliefert, da der Qwiic-Standard eine Versorgungsspannung von 3,3 V vorschreibt. Das bedeutet, dass das Board sowohl 5 V als auch 3,3 V I²C-Kommunikation verarbeiten kann.

Stromversorgung

Wenn der Arduino UNO R3 nicht über den USB-Anschluss mit Strom versorgt wird, läuft die Stromversorgung über einen einfachen linearen Spannungsregler. Bei den UNO R4-Boards wurde dieser durch einen Schaltregler ersetzt. Dieser ermöglicht einen viel größeren Eingangsspannungsbereich von 6 V bis 24 V (**Bild 4**). Der Regler, ebenfalls ein Produkt von Renesas, kann bis zu 1,2 A mit einem Wirkungsgrad von etwa 90 % liefern.
Beachten Sie, dass der UNO R4 WiFi auch einen Anschluss für die Stromversorgung der Echtzeituhr (RTC) hat.

Zusammenfassung

Die beiden Arduino UNO R4 Boards, der Minima und der WiFi, sind glaubwürdige Nachfolger des UNO R3. Sie haben den gleichen Umriss, Erweiterungsstecker und volle 5-Volt-E/A. Der UNO R4 WiFi ist ein bisschen ein UNO R4 Minima mit einem eingebauten Erweiterungsmodul. Mit seinem Wi-Fi-Modul, der LED-Matrix und dem Qwiic-Anschluss ist es einfach, IoT-Anwendungen zu entwickeln. Der UNO R4 ist viel leistungsfähiger als der UNO R3, mit mehr von allem, vom Speicher über die Peripherie bis zur Betriebsgeschwindigkeit. Ich musste ein spezielles Arduino UNO R4-Boards-Paket manuell in der 1.8.19 IDE installieren, bevor ich programmieren konnte. Die IDE in der Version 2 erkennt den UNO R4 und schlägt vor, die benötigten Softwarepakete zu installieren.
Für die meisten Benutzer wird der UNO R4 ein guter Ersatz für den UNO R3 sein!

Der UNO R4 ist ein großer Schritt nach vorn

Sehr interessant sind schließlich der Debugging-Anschluss des Minima und die CMSIS-DAP-Debugging-Fähigkeiten des ESP32-S3-Moduls am R4 WiFi, eine Funktion, auf die viele Entwickler seit den Anfängen von Arduino gewartet haben. Dies bringt den Arduino UNO endlich in die Arena der professionellen Entwicklung. Daher kann der Arduino UNO R4 als ein großer Schritt nach vorne betrachtet werden.

230443-02

Sie haben Fragen oder Kommentare?

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

Passende Produkte

- **Arduino UNO R4 Minima**
  https://elektor.de/20527
- **Arduino UNO R4 WiFi**
  https://elektor.de/20528

WEBLINKS

[1] Clemens Valens, „Der Arduino UNO R4 ist auf dem Weg zu uns!", elektormagazine.de: https://www.elektormagazine.de/news/der-arduino-uno-r4-ist-auf-dem-weg-zu-uns
[2] Arduino UNO R4 Minima: https://elektor.de/arduino-uno-r4-minima
[3] Arduino UNO R4 WiFi: https://elektor.de/arduino-uno-r4-wifi
[4] GitHub-Repository des Autors: https://github.com/ClemensAtElektor
[5] Charlieplexing: https://en.wikipedia.org/wiki/Charlieplexing

Logarithmische Potentiometer

Sie sind exponentiell!

Von Joseph Kreutz (Deutschland)

Wenn Sie einen Lautstärkesteller an einem Verstärker drehen oder an einem Mischpult verschieben, wünschen Sie sich im Idealfall eine lineare Beziehung zwischen dem Drehwinkel/Verschiebeweg des Potentiometers und der Stärke der Empfindung, die Ihr Ohr wahrnimmt. Was diesem Ideal nahe kommt, wird als „logarithmisches" Potentiometer bezeichnet, das aber eigentlich exponentiell ist. Schauen wir uns das etwas genauer an.

Die Natur hat uns so konzipiert, dass wir akustische Signale wahrnehmen, deren Amplitudenbereich von einem Teil bis zu 1.000.000 Teilen und mehr variiert, was einem Verhältnis von 120 dB oder mehr entspricht. Das sogenannte Weber-Fechner-Gesetz [1] besagt:

Empfindung = k × log(physikalische Anregung)

Dabei ist die Empfindung der Eindruck, den wir wahrnehmen, und der physikalische Reiz ist der Schalldruck an unseren Ohren [2]. Der Faktor k ist individuell verschieden, denn unsere Höreigenschaften sind genauso einzigartig wie die Haar- oder Augenfarbe. Ihre Hörempfindlichkeit hängt von der Frequenz, der effektiven Amplitude der akustischen Stimulation und von Ihrem Alter ab, da das Gehör mit dem Alter nachlässt.
Die untere Grenze des menschlichen Hörens wurde bei einem Schalldruckpegel von 20 µPa gemessen, der als Nullwert auf der Schalldruckskala, ausgedrückt in Dezibel, festgelegt wurde. Es ist auch nützlich zu wissen, dass ein Schallpegel von 130 dB beginnt, körperliche Schmerzen zu verursachen. Eine regelmäßige Belastung mit Schallpegeln über 100 dB kann zu dauerhaften Hörschäden führen.

Das ideale Lautstärkepotentiometer

Damit sich die Empfindung möglichst proportional zur Position (Drehung oder Verschiebung) des Steuergeräts verhält, sollte das von einem Potentiometer eingehaltene Gesetz einer logarithmischen Funktion entsprechen. Eine Exponentialfunktion entspricht dieser Anforderung am besten. Wir können einen Parameter definieren, der die Position eines Potentiometers darstellt:

$$\alpha = \frac{\text{Effektive Drehung}}{\text{Maximale Drehung}} = \frac{\text{Effektive Verschiebung}}{\text{Maximale Verschiebung}} \quad (I)$$

Da dieser Regler ein Minimum (im Prinzip 0) und ein Maximum (im Prinzip 1) hat, folgt daraus, dass $0 \leq \alpha \leq 1$. Das ideale Potentiometer sollte also eine Kennlinie haben, die der folgenden Funktion entspricht:

$$S = S_{ref} \times 10^{\alpha} \times F_p = 20 \times 10^{-6} \times 10^{5\alpha}$$

S_{ref} ist die untere Grenze der Empfindlichkeit des menschlichen Ohrs, also 20 µPa, und F_p ist ein Parameter, der für eine maximale Lautstärke von etwa 100 dB über der unteren Schwelle berechnet wird. Die Quelle [3] empfiehlt für einen Lautstärkesteller eine Schwankungsbreite von 90 dB, was diesem Wert sehr nahe kommt.

Potentiometer in der realen Welt

Die Konstruktion eines idealen Potentiometers, das exakt exponentiell ist, wäre zwar möglich, aber die technische Komplexität würde es zu einem teuren Bauteil machen. Die Hersteller haben diese Schwierigkeiten überwunden, indem sie Leiterbahnen mit zwei oder drei unterschiedlichen Widerstandsabschnitten hergestellt haben, um genügend lineare Segmente zur Annäherung an eine Exponentialfunktion zu schaffen. Dieser Kompromiss wird für die meisten Anwendungen als vollkommen zufriedenstellend angesehen.

Lineares Poti mit Widerstand

Eine weitere Möglichkeit der Annäherung an ein exponentielles Verhalten ist die Belastung eines linearen Potentiometers mit einem

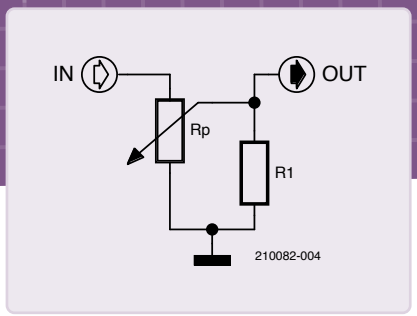


Bild 1. Durch einen zusätzlichen Widerstand an einem linearen Potentiometer ist es möglich, sich einem exponentiellen Verhalten anzunähern.

Widerstand, der zwischen seinem Schleifer und der Masse angeschlossen ist, wie in der Schaltung in **Bild 1** dargestellt. Die so erhaltene Reaktion ist durch die folgende Funktion gekennzeichnet:

$$U_{OUT} / U_{IN} = \alpha \times \gamma / [\alpha \times (1-\alpha)+\gamma] \quad (II)$$

Dabei steht α für die Drehung oder Verschiebung des durch (II) definierten Steuerelements und γ (Gamma) für das Verhältnis zwischen dem Lastwiderstand R1 und dem Gesamtwiderstand des Potentiometers Rp. Wenn zum Beispiel Rp = 1 MΩ und R1 = 39 kΩ, dann ist γ = R1 / Rp = 0,039.
Die Kurven in **Bild 2** stellen die ideale Exponentialfunktion (grün) und die mit einem Gammawert von 0,039 erhaltene Funktion (rot) dar. Diese Wahl von γ und die entsprechenden Widerstandswerte führen zu Ergebnissen, die der Exponentialfunktion am nächsten kommen. Es gibt jedoch eine Diskrepanz zwischen der idealen Kurve und dieser Annäherung, die bei sehr niedrigen Pegeln auftritt, und die Dämpfung zeigt einen kleinen, aber recht abrupten Sprung am Anfang der Kurve. Aus diesem Grund wird diese Lösung nicht für die Lautstärkeeinstellung in HiFi-Anlagen empfohlen [3].
Unter Berücksichtigung der in den Gleichungen I und II definierten Parameter ergibt sich für den Lastwiderstand $R_L(\alpha)$, wie er von der Schaltung vor der Baugruppe R1/Poti aus gesehen wird, folgender Wert

$$R_L(\alpha) = Rp \times \alpha \times (1-\alpha) + \gamma / (\alpha + \gamma)$$

Die Werte sind jeweils gleich Rp für α = 0 und Rp||R1 für α = 1. Diese Gleichung setzt voraus, dass die Impedanz der Stufe, die dem Lautstärkesteller folgt, ausreichend hoch ist, um vernachlässigbar zu sein. Die Stufe vor dem Lautstärkeregler muss dann nur in der Lage sein, den Strom durch den Lautstärkesteller zu liefern.
Beachten Sie dabei auch, dass der Gesamtwert des Widerstands

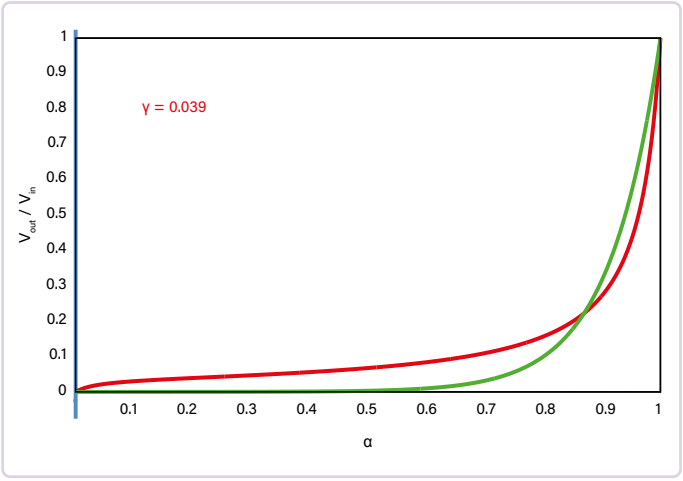


Bild 2. Die ideale Exponentialkurve (grün) und die tatsächliche Kurve mit einem γ-Wert von 0,039 (rot). Dieser Wert von γ kommt einer Exponentialfunktion am nächsten.

Rp eines Potentiometers in der Regel mit einer Toleranz von ±20 % angegeben wird und damit zu den Bauteilen mit den höchsten Toleranzen überhaupt gehört.

Thermisches Rauschen

Ein idealer Widerstand erzeugt eine thermische Rauschspannung, die gleich

$$e_n = \sqrt{(4 \times k \times T \times R \times B)} \quad (III)$$

ist, wobei k = 1,38 × 10^{-23} (Boltzmann-Konstante), T die Temperatur in Kelvin (hier 298,15°K für 25°C), R der Widerstandswert in Ohm und B die später bei der Messung verwendete Frequenzbandbreite (hier 20 kHz) ist.
Der von einem Potentiometer in der Schaltung von Bild 1 eingebrachte Serienwiderstand trägt höchstens zu einem Viertel des Gesamtwerts von α × (1-α) × Rp mit α = 0,5, also 250 kΩ bei, wenn Rp = 1 MΩ. Die entsprechende Rauschspannung beträgt:

$$e_n = \sqrt{(4 \times 1{,}38 \times 10^{-23} \times 298{,}15 \times 250.000 \times 2 \times 10^4)} \approx 9{,}07\ \mu V$$

Die maximale Rauschspannung wird durch den Teiler aus dem Serienwiderstand des Potentiometers und R1 (39 kΩ) abgeschwächt. Ihr Beitrag wird somit auf 1,22 µV reduziert.
Das durch den 39-kΩ-Widerstand erzeugte Rauschen muss ebenfalls berücksichtigt werden. Es beträgt bis zu 3,58 µV bei einer Temperatur von 25°C und einer Bandbreite von 20 kHz. Sein Beitrag zum Gesamtrauschen beträgt 3,1 µV. Damit beträgt die maximale thermische Rauschspannung

$$e_{n,max} = 3{,}1 \times 10^{-6} + 1{,}22 \times 10^{-6} = 4{,}32\ \mu V$$

Dies entspricht einer Verbesserung von 6,4 dB gegenüber einem Potentiometer allein.
Bei den meisten handelsüblichen Potentiometern besteht die Leiterbahn aus Kohlenstoff, was zu einem höheren Rauschpegel als dem nach Gleichung III berechneten führt. Einige Hersteller bieten Potentiometer mit einer Leiterbahn aus Kunststoff oder „Cermet", einer Kombination aus Metall und Keramik, an. Sie sind im Gebrauch weniger empfindlich und weisen im Allgemeinen eine geringere Rauschspannung auf als Kohlepotentiometer. Natürlich ist es besser, für R1 ein rauscharmes Bauteil zu wählen, zum Beispiel einen Metallfilmwiderstand.
Eine allgemein gültige Schlussfolgerung aus den obigen Ausführungen ist, dass in elektronischen Schaltungen zur Vermeidung von thermischem Rauschen Widerstände mit möglichst geringem Wert verwendet werden sollten.

Vergleich der Lösungen

Die Schwierigkeiten bei der Konstruktion eines Potentiometers mit exponentieller Kennlinie wurden mit den beiden oben genannten Methoden überwunden. Welche von ihnen ist besser?

Im Falle eines Lautstärkestellers für Consumer-Geräte ist die Annäherung durch lineare Segmente, wie sie von den Herstellern in ihren so genannten „logarithmischen" Potentiometern angeboten wird, eine durchaus annehmbare Lösung, da die Dämpfung bei niedrigen Pegeln einem idealen Exponentialwert recht nahe kommt und praktisch gegen Null tendiert. Beachten Sie hierzu die Kennlinien der Hersteller.
Für einen Lautstärkesteller in einem Mischpult oder am Eingang eines Verstärkers ist ein lineares Potentiometer, das mit einem Widerstand belastet ist, völlig ausreichend. Auch wenn das Stellglied bei niedrigen Pegeln einige Anomalien aufweist, ermöglicht die Verwendung eines Widerstandes mit einem linearen Potentiometer auch eine erhebliche Verringerung des thermischen Rauschens. Dies ist bei vielen Anwendungen, insbesondere bei professionellen Geräten, ein großer Vorteil.

RG — 210082-02

Passende Produkte

- **Douglas Self, *Small Signal Audio Design* (2nd Edition), Routledge, 2014**
  https://elektor.de/18046
- **Elektor Fortissimo-100 High-End-Verstärker-Kit**
  https://elektor.de/20273

WEBLINKS

[1] Weber-Fechner-Gesetz: https://de.wikipedia.org/wiki/Weber-Fechner-Gesetz
[2] Gehörrichtige Lautstärke: https://de.wikipedia.org/wiki/Geh%C3%B6rrichtige_Lautst%C3%A4rke
[3] D. Self, „Small Signal Audio Design (2nd Edition)", Routledge, 2014: https://elektor.de/18046

Motortreiber-Breakout-Board

Ein BoB für einen 5-A-Treiber für DC-Motoren mit einer Größe von 3×3 mm

Von Edwin van den Oetelaar (Niederlande)

Heutzutage haben es Maker und Studenten meist sehr eilig und sind nicht willens, elektronische Schaltungen zu entwerfen und zu löten. Sie stehen mehr auf fertige „Shields“, und der Umgang mit einem Lötkolben ist eine große Herausforderung. Deshalb habe ich ein Breakout-Board für den Motortreiber MP6619 von Monolithic Power Systems entwickelt, das sich gut für Projekte mit Motoren verwenden lässt.

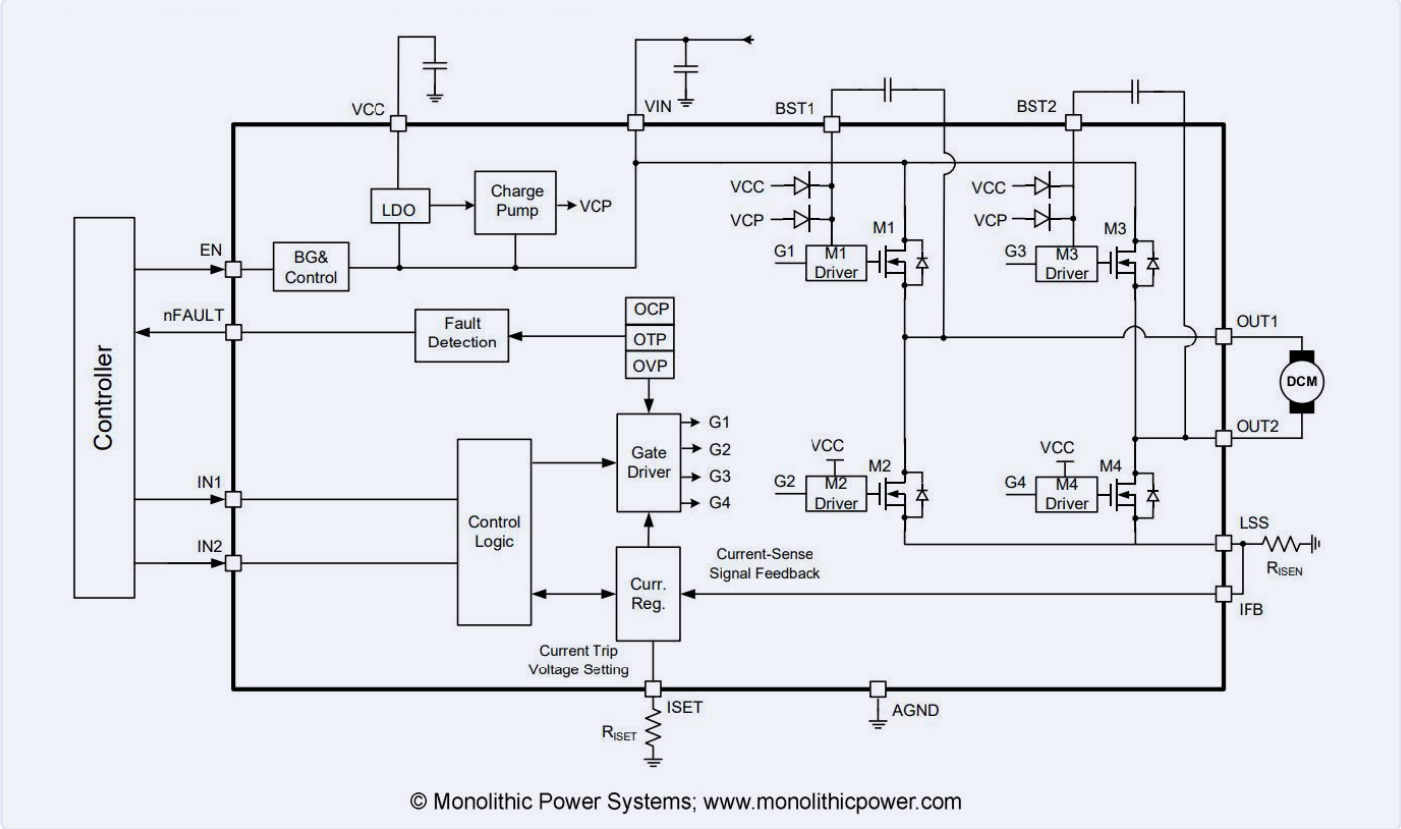


Bild 1. Blockdiagramm des MP6619 mit peripherer Beschaltung. (Quelle: Monolithic Power Systems [1])

Beim MP6619 handelt es sich um eine neue kompakte H-Brücke für die Steuerung von kostengünstigen Bürsten-Gleichstrommotoren und Magnetspulen. Diese DC-Motoren sind die zugänglichsten und erschwinglichsten für Bastler, die sich mit dem Bau von kleinen Robotern, Modellbau und kreativen Projekten beschäftigen. Dies ist ein Bericht über meine Entdeckungsreise von der Idee zu einem fertig entwickelten und getesteten Breakout-Board (BoB) für dieses winzige Motorsteuerungs-IC.

Das Abenteuer beginnt

Eines Tages erhielt ich eine E-Mail von einem Bauteile-Distributor über ein brandneues H-Brücken-IC namens MP6619. **Bild 1** zeigt das interne Blockdiagramm. Dieses winzige „Wunder" von nur 3×3 mm zeigt beeindruckende Eigenschaften wie 5 A bei 24 V, und auch anderen Aspekten erregten meine Aufmerksamkeit (siehe Kasten **Chip-Eigenschaften**). Wie kann das möglich sein? So viel Strom, kein Kühlkörper und alle Schutzfunktionen in einem solchen Miniaturgehäuse? Ich dachte, das ist sicher nur Marketing-Talk, denn Alternativen mit solchen Eigenschaften sind locker zehnmal größer.

Aber meine Neugier war geweckt, und ich wollte wissen, ob das stimmt und ob es für meine Projekte nützlich sein könnte. Also habe ich ein paar bestellt, und nach nur zwei Tagen lagen die Chips auf meinem Schreibtisch: Wow, sie waren wirklich winzig! Nur 3×3 mm mit 0,2 mm Abstand zwischen den Kontakten und ohne Beine. Worauf habe ich mich da nur eingelassen?

Ich stürze mich kopfüber in dieses Thema. Trotz des niedrigen Preises von etwa 2,50 €, der geringen Abmessungen und des ungewöhnlichen Platzbedarfs erwiesen sich die MP6619 als nicht direkt anwendbar für meine Projekte. Ich brauchte ein Breakout-Board, das den MP6619 für Tests und die spätere Integration handlicher machen würde.

Aber es gab keinen BoB für diesen winzigen *Quad, Flat, No-lead, 19-contact Footprint* (QFN-19), so dass ich selbst eine solche Platine entwickeln musste. Mein Abenteuer mit der MP6619-H-Brücke begann!

Entwickelt mit KiCad

Um eine MP6619-Breakout-Platine zu entwickeln, begann ich mit dem Studium des Datenblatts [1] und dem Zeichnen des BoB-Schaltplans in KiCad. Zuerst zeichnete ich das Symbol und den Footprint des ICs, das weder in der Bibliothek noch in SnapEDA verfügbar war, und dann das Layout.

Ich achtete besonders auf die Platzierung des $R_{(ISENSE)}$-Shunt-Widerstands und den Anschluss der Power-Pads. Dieser Shunt-Widerstand misst den Strom durch die MOSFET-Treiber und ist für das ordnungsgemäße Funktionieren des Breakout-Boards von entscheidender Bedeutung. Auch der Spannungsabfall über den Leiterbahnen sollte beim Leiterplattenlayout berücksichtigt werden!

Die OCP (Over-Current Protection) ist mit einem externen Widerstand einstellbar. Ohne diesen schaltet der Chip ab, wenn die Spannung über dem Shunt-Widerstand 200 mV erreicht. Im Datenblatt wird ein Shunt von 0,04 Ω vorgeschlagen, was den maximalen Strom auf 5 A begrenzt. Es sind auch andere Widerstandswerte möglich; ich habe für meinen ersten Test einen Shunt von 0,05 Ω gewählt, weil ich ihn gerade zur Hand hatte. Daraus ergibt sich eine Begrenzung des Ausgangsstroms auf 4 A.

Durch einen zusätzlichen $R_{(ISET)}$ von beispielsweise 80 kΩ wird der Grenzwert auf 100 mV gesetzt. So kann die Stromgrenze eingestellt werden, ohne einen anderen Shunt-Widerstand berechnen und bestücken zu müssen. Die Schwellspannung berechnet sich nach $U_{ITRIP} = 8/R_{ISET}$ (kΩ).

Die erste Version des BoB-Platinenlayouts habe ich schnell und ohne gründliches Nachdenken erstellt, da ich bald mit dem Testen beginnen wollte, aber ich habe mich zu früh gefreut. Das erste BoB enthielt zwei MP6619-Chips, LEDs, Widerstände, Kondensatoren und Anschlüsse. Damals dachte ich noch, dass Platinenanschlussklemmen nützlich sein könnten. Ich brauchte vier Kupferlagen, setzte viele Durchkontaktierungen, eine Massefläche und verwendete wegen der hohen Ströme möglichst breite Leiterbahnen. Das war die Arbeit eines Tages. Eine Woche, nachdem ich die Daten an einen chinesischen PCB-Hersteller geschickt hatte, hatte ich die Platinen dann auf meiner Werkbank. Das Löten des QFN war die größte

Chip-Eigenschaften

- Es ist nicht nur Marketinggerede: Der MP6619 kann tatsächlich 24 V und 5 A verarbeiten. Er ist gut gegen Überlast und Kurzschluss geschützt und erholt sich zuverlässig von Fehlersituationen.
- Das Hinzufügen einer Drosselspule oder eines Filters verbessert die Leistung erheblich. Ob dies erforderlich ist, hängt von Ihrer Anwendung und dem Motor ab.
- Die integrierten N-Kanal-MOSFETs haben einen niedrigen $R_{ds(on)}$ von 65 mΩ, was für die Begrenzung der Verlustleistung unerlässlich ist. Die für die Ansteuerung ihrer Gates erforderlichen Boost-Schaltungen sind ebenfalls integriert. Ihre Kondensatoren sind extern angeschlossen.
- Der MP6619-Chip wird ohne Bonddrähte hergestellt, um eine gute thermische Leistung in einem QFN-19-Gehäuse zu erreichen.
- Die Inseln des Footprints sind extra lang, um die Wärme vom Chip an die Platine abzuleiten.
- Die Platine muss sorgfältig entworfen werden und nicht nur den elektrischen, sondern auch den thermischen Anforderungen zu genügen.

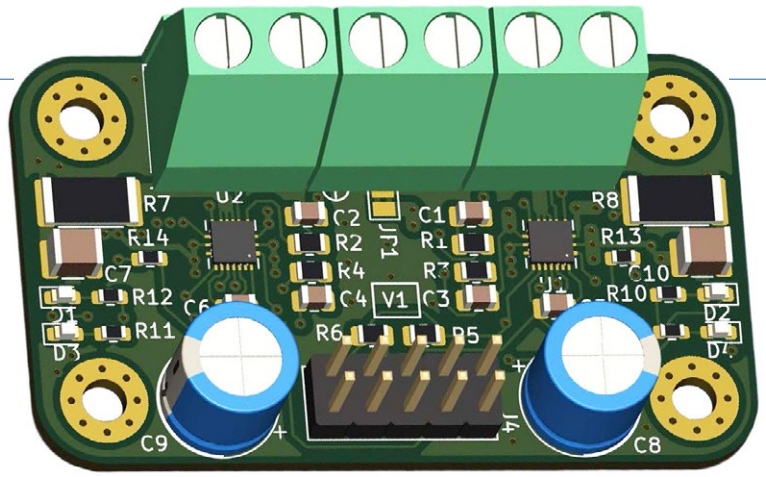


Bild 2. 3D-Modell der ersten Platine.

Herausforderung, da die Chips kleiner waren als alles, was ich bisher gewohnt war.

Um das QFN mit allen Anschlüssen korrekt zu platzieren, waren ein Stereomikroskop, ein guter temperaturgeregelter Lötkolben, ein Heißluftlötkolben, viel Flussmittel und noch mehr Geduld erforderlich. Sobald das QFN an seinem Platz und der Schweiß von der Stirn gewischt war, entpuppten sich die anderen „normalen" SMD-Bauteile als Kinderspiel; sie erschienen im Vergleich zum QFN gigantisch. Die Lötinsel für die Platinenanschlussklemmen ist größer als der gesamte QFN-Chip (**Bild 2**)! Ich habe diese erste Version der Platine auf der Elektor-Labs-Website veröffentlicht [2].

Erste Tests

Die ersten Tests wurden mit einem kleinen Gleichstrommotor [3] durchgeführt, der auf meiner Werkbank lag. Es war ein einfaches 12-V-Modell mit einem Blockierstrom von unter 5 A (**Bild 3**).

Der MP6619 besteht intern aus zwei Halbbrücken, wobei jede Halbbrücke einen Eingang hat (IN1, IN2). Der Chip selbst sorgt für das richtige Timing der High-Side- und Low-Side-MOSFETs, so dass kein Kurzschluss zwischen den Stromschienen entstehen kann.

Der Chip ist gegen alle typischen Probleme einer Motorsteuerung geschützt. Seine integrierten Funktionen UVLO (Under Voltage Lockout), OVP (Overvoltage Protection), OCP (Overcurrent Protection) und OTP (Over Temperature Protection) sollten fast alle Fehlersituationen sicher beherrschen (was wir noch sehen werden)!

Bild 3. Die ersten Tests wurden mit einem DC-Motor mit Planetengetriebe durchgeführt. (Quelle: E-S Motor / RobotShop [3])

Anschluss von IN1, IN2 und EN

Die Eingänge sind mit Pull-Down-Widerständen ausgestattet. Sobald der EN-Eingang einen High-Pegel erhält, wird der Chip aktiviert, der interne Spannungsregler beginnt zu arbeiten und der VCC-Ausgang des MP6619 liefert eine Spannung von 5 V. Ich habe einen kleinen 100-nF-Entkopplungskondensator hinzugefügt, um diese Spannung zu stabilisieren. Wenn die 5 V anliegen, werden auch die N-Kanal-MOSFETs aktiv. Je nach den Eingangspegeln an IN1 und IN2 werden entweder die High-Side- oder die Low-Side-MOSFETs geschaltet.

An den Eingängen dürfen Logikpegel von 3,3…5 V anliegen. Es ist also kein Problem, das BoB an einen ESP32 oder einen Arduino anzuschließen. Für meine ersten Tests habe ich Schalter verwendet, um die Eingänge mit 3,3 V zu versorgen, und den Motor an die Ausgänge des BoB angeschlossen.

Teilweiser Erfolg

Strom an! Doch mein Motor wollte sich nicht drehen, der nFault-Ausgang ging sofort auf Low (Fehleranzeige-LED leuchtete), aber ich hatte keinen Kurzschluss am Ausgang. Erst durch eine deutliche Verringerung der Eingangsspannung konnte ich den Motor schließlich langsam gegen und mit dem Uhrzeigersinn drehen. Es ist wichtig zu erwähnen, dass ich da noch keine PWM-Drehzahlregelung vorgesehen hatte. Ich experimentierte mit einer Verringerung des Shunt-Widerstands und der Erhöhung des R_{ISET}-Widerstands, um den Motor in Bewegung zu bringen, was letztendlich zu einem „menschlichen Fehler" führte – 12 V an IN1, ein verbranntes BoB als Ergebnis und ein tiefer Seufzer der Frustration. Die Platine ging in Rauch auf, und ich musste ein neues BoB mit Bauteilen bestücken.

Ein anderes BoB und weitere Tests

Schließlich fand ich heraus, dass der Motor mit Entstörkondensatoren ausgestattet ist, die parallel zum Kommutator geschaltet sind. Infolgedessen verhält sich der Motor weniger induktiv und mehr kapazitiv. Dies führt dazu, dass der Überstromschutz OCP sofort anspringt, wenn eine Stromspitze im Motor auftritt. Diese vorgebliche Überlast führt zu einem Timeout von 1 ms, bevor der MP6619 es erneut versucht. Dieser Vorgang wiederholte sich immer wieder, so dass der Motor nur leise vor sich hin quietschte.

Das Entfernen dieser Kondensatoren hilft teilweise, den Motor mit etwas höheren Spannungen steuern zu können. Nachdem ich das Datenblatt genauer studiert hatte (endlich!), hatte ich inzwischen auch Verbesserungspläne für mein BoB und beschloss, eine neue Version zu erstellen.

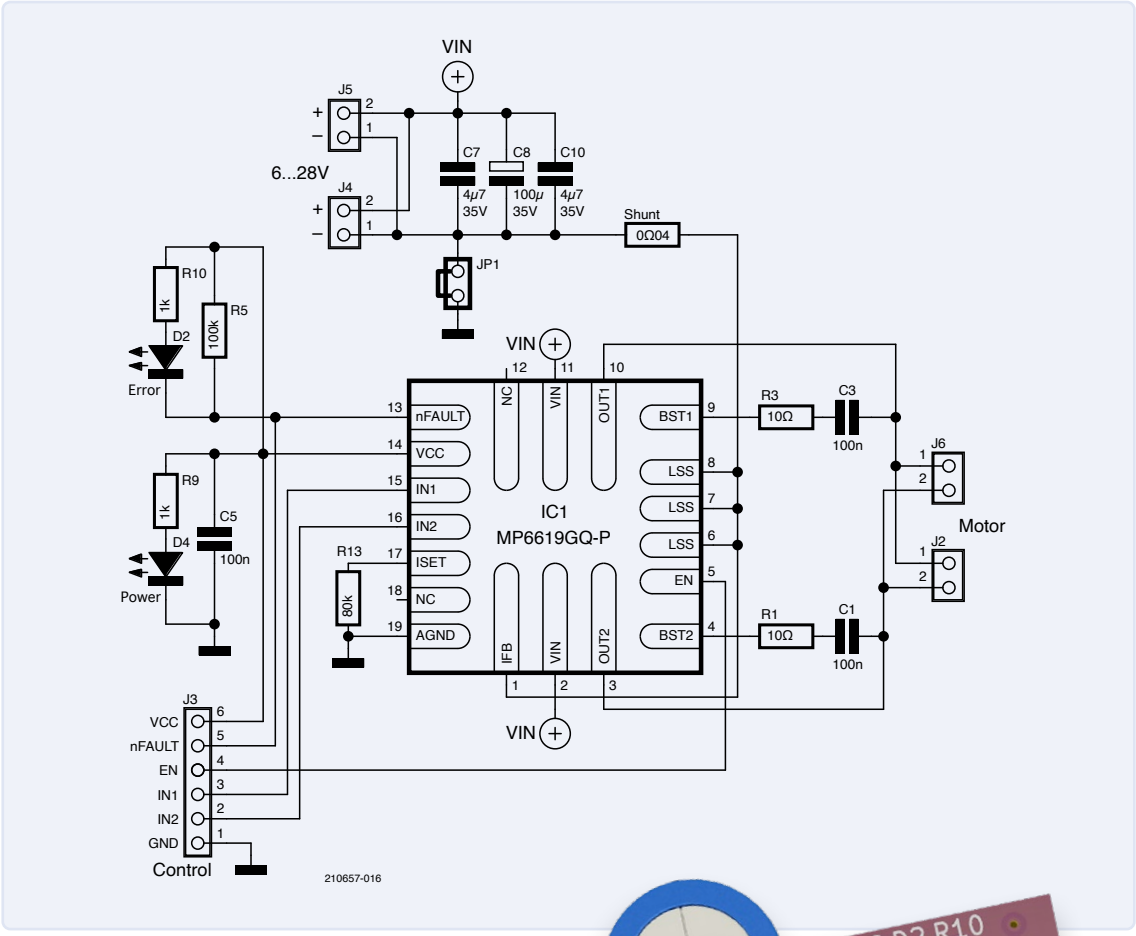


Bild 4. Der überarbeitete Schaltplan des Breakout-Boards.

Neuer Entwurf, neues Layout

Für meine zweite Version musste die Schaltung und das Layout angepasst werden. Ich habe die folgenden Änderungen vorgenommen:

- nur ein MP6619-Chip
- die Position des Shunts wurde überarbeitet
- eine 0,1"-Stiftleiste anstelle von Anschlussklemmen
- keine Befestigungslöcher
- klarer Bestückungsaufdruck
- minimale Anzahl von Bauteilen

Dies führte zu einem neuen Schaltplan (**Bild 4**) und in der Folge auch zu einer neuen Platine (**Bild 5** und **Bild 6**) mit (ebenfalls) vier Lagen und einer Grundfläche von nur etwa 25×25 mm, die in einem Nutzen mit ENIG-Finish (**E**lectroless **N**ickel **I**mmersion **G**old) hergestellt wurde. Nach der Bestückung des BoBs habe ich ein wirklich kompaktes Modul erhalten, das, wie ich glaube, von fast jedem Mikrocontroller gesteuert werden kann.

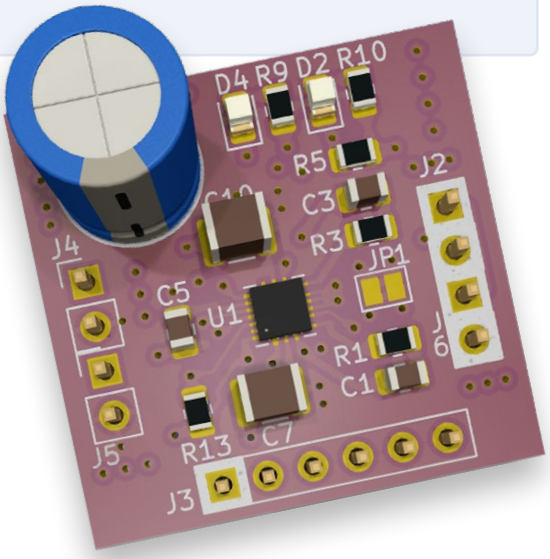


Bild 5. 3D-Modell des 25×25 mm großen BoB (Oberseite), bestehend aus vier Schichten und ENIG-Finish.

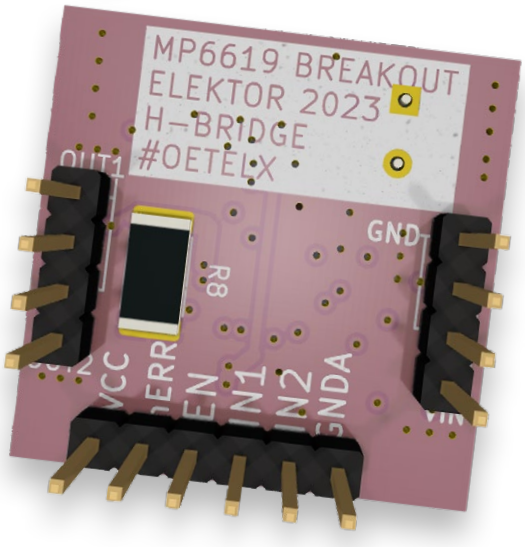


Bild 6. 3D-Modell des BoBs (Unterseite) mit Stiftleisten.

Weitere Tests und Experimente

Ich habe denselben Motor wie im ersten Test verwendet. OVP und ULVO verhielten sich gut. Jetzt habe ich es mit PWM-Steuerung versucht, um eine Drehzahlregelung zu realisieren. Die Tests mit Frequenzen von 1...50 kHz ergaben das erwartete Verhalten: Der Motor ließ sich gut regeln. Das Beispiel aus dem Datenblatt verwendet 20 kHz, und das hat bei niedrigen Spannungen gut funktioniert.

Bild 7. Eine Ringkerndrossel in Reihe mit dem Motor verbessert das Verhalten des BoBs.

Aber bei höheren Spannungen setzte wieder die OCP ein und der Motor blieb stehen. Durch Hinzufügen einer zusätzlichen 220-µH-Induktivität aus meiner Bauteilkiste (wie die in **Bild 7**) in Reihe mit dem Motor wurde das Regelverhalten deutlich verbessert und der Drehzahlbereich vergrößert.

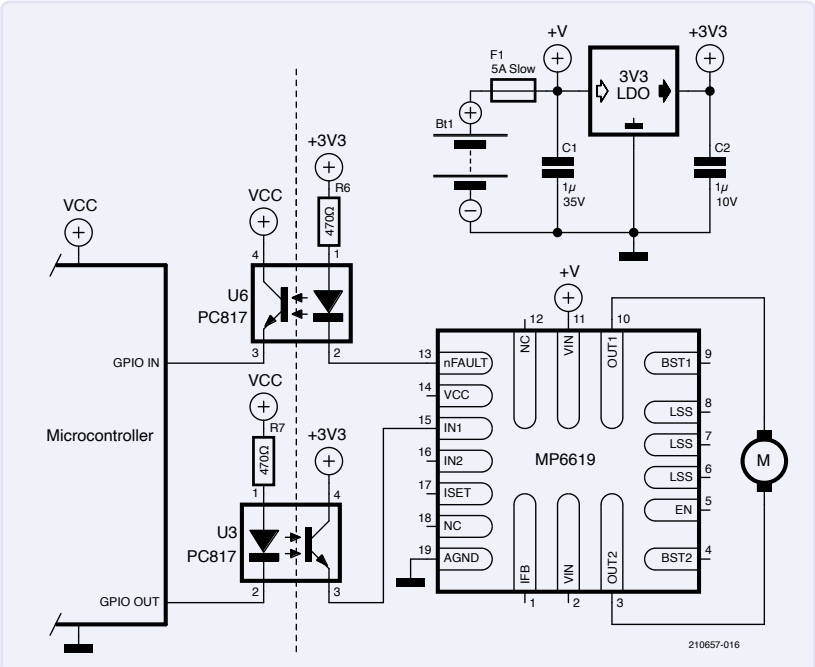
Infolgedessen lieferte der Motor mehr Leistung, und das OCP schaltete sich seltener ein. Solche zusätzlichen Induktivitäten scheinen also die Lösung für den Einsatz in Projekten zu sein.

Unter bestimmten Bedingungen gibt der Motor Energie in die H-Brücke zurück, da der Motor auch als Generator fungiert (EMK eines Gleichstrommotors). Diese Energie wird über die Dioden in den MOSFETs auf die Stromschienen abgeleitet. In meinen Tests habe ich einen Elektrolytkondensator C8 mit niedrigem ESR-Wert parallel zur Stromquelle verwendet. Mit dieser letzten Verbesserung war der BoB stabil.

Optokoppler

Bild 8 zeigt meine Testschaltung. Ich beschloss, Optokoppler zu verwenden, um eine galvanische Trennung zwischen der Steuerseite und den Motortreibern zu erreichen, da die Stromversorgung eines Motors in der Regel erhebliche Störpegel aufweist. Auch EMI kann Probleme verursachen. Die Isolation verringert das Risiko von Störungen der Niederspannungssignale und verhindert, dass starke Impulse, die vom Motor und seinem Treiber erzeugt werden, den empfindsamen Mikroprozessor und andere Elektronik erreichen.

Bild 8. Diese Schaltung mit Optokopplern zwischen Mikrocontroller und BoB wurde für die Tests verwendet.



Widerstände:
(SMD 0805)
R1, R3 = 10 Ω
R5 = 100 k
R8 = 0Ω04, 2 W, SMD 2512
R9, R10 = 1 k
R13 = 80 k

Kondensatoren:
C1, C3, C5 = 100 n, 50 V, SMD 0805
C7, C10 = 4µ7, 35 V, SMD 1210
C8 = 100 µ, 35 V, Elko, ø 8 mm

Halbleiter:
D2 = LED, rot, SMD 0806
D4 = LED, grün, SMD 0805
IC1 = MP6619GQ-P, QFN-19

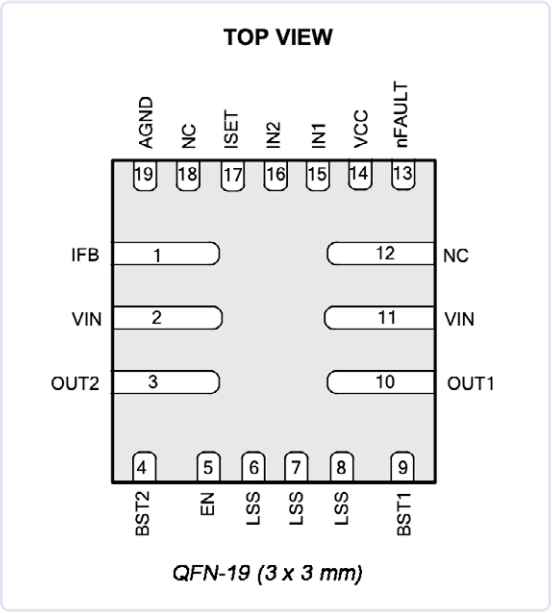
Außerdem:
J2, J4, J5, J6 = 1×2-polige Siftleiste, 2,54 mm
J3 = 1×6-polige Stiftleiste, 2,54 mm
BoB-Platine (siehe Text)

Die LEDs auf der Eingangsseite der Optokoppler können direkt von einem Mikroprozessor-Pin mit einem 470-Ω-Widerstand in Reihe betrieben werden. Auf der Ausgangsseite ist der Emitter des Fototransistors mit dem IN1 des BoB und damit mit dem Eingang des MP6619 verbunden, und sein Kollektor ist mit einer logischen Spannung gekoppelt, die von einem 3,3-V-LDO-Spannungsregler bereitgestellt wird, der wiederum von der Stromversorgung des Motors - in meinem Fall einer Batterie - gespeist wird. Dies ist ausreichend, um einen zuverlässigen und stabilen Betrieb zu gewährleisten.

Bild 8 zeigt auch, wie man das nFault-Signal der H-Brücke (Open-Drain-Ausgang) über einen Optokoppler an die MCU weiterleiten kann, und zwar auf die gleiche Weise wie beschrieben, nur umgekehrt. In einem solchen Fall benötigt der GPIO des angeschlossenen Mikrocontrollers einen Pull-Down-Widerstand.

QFN-Gehäuse

Die FCQFN-Gehäuse (**Bild 9**) können unregelmäßig geformte Pads haben, die oft in langen, schmalen Streifen angeordnet sind. Im Gegensatz zu normalen QFN-Gehäusen wird die Wärme über viele dieser Pads abgeleitet, anstatt über ein großes zentrales Pad. Dies stellt das Leiterplattendesign vor einige Herausforderungen, da es viele Pads mit unterschiedlichen Signalen gibt, die mit unterschiedlichen Kupferflächen verbunden werden müssen.



◀ *Bild 9. Pinbelegung des MP6619 im QFN-19-Gehäuse. (Quelle: Monolithic Power Systems [1])*

Kleine Durchkontaktierungen können innerhalb der Pad-Bereiche platziert werden (Via in PAD). Bei mehrlagigen Leiterplatten mit Stromversorgungs- und Masseflächen können Durchkontaktierungen diese Pads direkt mit den Flächen verbinden. In anderen Fällen muss Kupfer direkt an den Pads angebracht werden, um die Wärme vom IC an größere Kupferflächen abzuleiten. Eine Layout-Richtlinie finden Sie unter [4].
Das größte Hindernis, auf das ich bei diesem Projekt stieß, war aber das Löten des MP6619-Chips. Ich habe viel mit verschiedenen Techniken experimentiert und hatte schließlich Erfolg mit viel Flussmittel, verbleitem Lot und Heißluft, aber es blieb trotz allem eine Herausforderung, den Chip präzise anzubringen. Einmal platziert, sorgt dann die Oberflächenspannung des geschmolzenen Lots dafür, dass der Chip sich richtig an den Pads ausrichtet. Es stellte sich heraus, dass viel Übung nötig war, bis man die Chips befriedigend gut auf die Platine löten kann. Als Konsequenz habe ich mir eine Reflow-Hotplate angeschafft, um den Lötprozess in Zukunft zu vereinfachen. Keine schlechte Investition!

Epilog

In diesem Artikel habe ich Ihnen über meine Erfahrungen beim Entwurf und Bau eines Breakout-Boards für den H-Brücken-Motortreiber MP6619 berichtet. Ich habe die Bedeutung von Optokopplern für die elektrische Isolierung erörtert und mit Ihnen meine Erkenntnisse über die Arbeit mit Flip-Chip-QFN-Gehäusen und die Herausforderungen beim Löten solcher kleinen Bauteilen geteilt.

Die MP6619-Breakout-Platine bietet eine sehr kompakte Lösung für die Steuerung von preiswerten Gleichstrom-Bürstenmotoren, was sie zu einer guten Wahl für Bastler und Maker macht, die kleine Roboter und andere ähnlich kreative Projekte bauen. Alle Design- und Produktionsdateien sind Open-Source und können von meiner GitHub-Seite [5] heruntergeladen werden. Der MP6619 ist in kleinen Mengen bei Händlern wie Farnell, Mouser und direkt bei Monolithic Power Systems erhältlich. ⏮

RG — 210657-02

Über den Autor

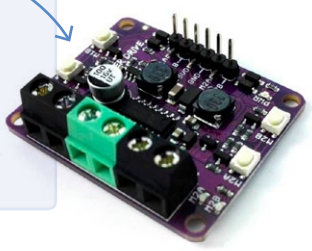
Edwin van den Oetelaar arbeitet als Ingenieur bei Fontys ICT in den Niederlanden. Er ist Mitglied der High Tech Embedded Software Research Group, die von Prof. Teade Punter geleitet wird. Als Spezialist für Hardware- und Softwareentwicklung genießt Edwin die Möglichkeit, verschiedenste technische Projekte in Angriff zu nehmen. Seine Fachkenntnisse erstrecken sich über ein breites Spektrum der Elektronik wie Streaming Data, intelligente Geräte, Gesundheitswesen, IoT, Bildverarbeitung und angewandte KI.

Haben Sie Fragen oder Kommentare?

Wenn Sie technische Fragen haben, können Sie die Elektor-Redaktion per E-Mail an redaktion@elektor.de kontaktieren.

Passende Produkte

- **Dogan Ibrahim, *Motorsteuerung mit Arduino und Raspberry Pi* (Elektor 2017)**
  Buch, kartoniert: https://elektor.de/18725
  E-Buch, PDF: https://elektor.de/18726
- **Cytron Maker Drive - H-Brücken-Motortreiber**
  https://elektor.de/18923
- **Peter Dalmaris, KiCad 6 Like A Pro – Fundamentals and Projects**
  E-Book, PDF, Elektor 2022:
  https://elektor.de/20159



WEBLINKS

[1] MP6619-Datenblatt bei Monolithic Power Systems: https://t1p.de/xab9q
[2] Erste Version bei Elektor Labs: https://t1p.de/9t1yj
[3] DC-Motor mit Planetengetriebe: https://t1p.de/mwxp1
[4] Motor Driver PCB Layout Guidelines — Part 2: https://tinyurl.com/2p8ry9hd
[5] GitHub des Autors: https://github.com/edwin-oetelaar

Quelle: Shutterstock / wk1003mike

Aus dem Leben gegriffen

Gefährliche Elektronik

Von Ilse Joostens (Belgien)

Wenn Sie regelmäßig Bauteile bei großen Elektronikherstellern in Übersee bestellen, ist die Wahrscheinlichkeit groß, dass Sie früher oder später auf eine Verpackung stoßen, die mit dem berüchtigten Warnhinweis „California Proposition 65“ versehen ist. Bei mir handelte es sich um Leiterplattenstecker, und „Big Brother“ wies mich freundlicherweise darauf hin, dass ich durch den Kontakt mit diesen Steckern Gefahr laufe, Krebs zu entwickeln oder Fruchtbarkeitsprobleme zu bekommen. Es sieht so aus, als ob Elektronik das neue Rauchen geworden ist. Ich nehme solche Warnungen nicht allzu ernst, aber wenn man mit Elektronik arbeitet, läuft man trotzdem Gefahr, gefährlichen Stoffen ausgesetzt zu sein - und ich spreche nicht von Bleilot, Lötdämpfen, Flussmitteln, Ätzmitteln, Lösungsmitteln, Lacken oder chemischen Produkten mit unklarer Zusammensetzung.

Eine kleine Zugabe

Wir alle haben schon gehört, dass integrierte Schaltkreise aus Sand hergestellt werden. Sand ist im Wesentlichen Siliziumdioxid, und Silizium ist heutzutage das am häufigsten verwendete Halbleitermaterial. In der Praxis jedoch steckt in ICs ein kleines bisschen mehr. Nicht nur ICs, sondern auch viele andere Bauteile - ob Halbleiter oder nicht - haben eine dunkle Seite und enthalten manchmal sehr exotische chemische Substanzen.

Arsen, das im viktorianischen Zeitalter für reizvolle grüne Tapeten verwendet wurde, wird heute in der Elektronikindustrie in großen Mengen zur Dotierung von Silizium und zur Herstellung von Galliumarsenid, einem weiteren wichtigen Halbleiter, eingesetzt. Die inzwischen berüchtigten PFAS und PFOS [1] werden in Prozessen zum Belichten und Ätzen von Siliziumwafern verwendet, und Rückstände dieser Stoffe verbleiben gelegentlich im Endprodukt. Bauteilverpackungen enthalten nicht nur Epoxid, sondern auch Flammschutzmittel in Form von organischen Brom- und Chlorverbindungen in Kombination mit Antimontrioxid [2]. Diese Stoffe sind zwar nicht hochgiftig, haben aber dennoch einen zweifelhaften Ruf, so dass man sich bemüht, ihre Verwendung einzuschränken.

Dann gibt es noch Beryllium - eine echte Horrorvorstellung, die Arsen harmlos aussehen lässt. Früher war im weißen Leuchtstoff von Leuchtstofflampen ein Zink-Beryllium-Silikat enthalten, bis viele Fabrikarbeiter an Berylliose erkrankten, einer schlimmen, chronischen Lungenkrankheit [3]. Heutzutage wird Berylliumoxid unter anderem als wärmeleitendes Keramikmaterial in Hochfrequenz-Leistungstransistoren verwendet. Bevor Sie eine Panikattacke bekommen: Kontakte aus Berylliumkupfer (die oft als Federkontakte verwendet werden) sind relativ harmlos, solange man sie nicht feilt, schleift oder schmirgelt.

Wenn Sie ein gewisses Alter überschritten haben, erinnern Sie sich wahrscheinlich noch an Selen-Gleichrichter (**Bild 1**) in Form eines Stapels dünner quadratischer Platten, die in der Mitte miteinander verbunden sind. Solange alles gut geht, ist das kein Problem, aber den Geruch eines überhitzten

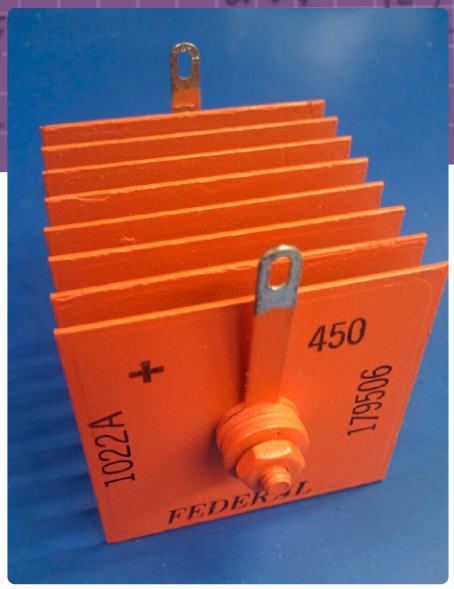


Bild 1. Ein Selen-Gleichrichter. Quelle: Wikimedia / Binarysequence / https://creativecommons.org/licenses/by-sa/3.0.

Selengleichrichters vergisst man nie. Der dabei freigesetzte Selenwasserstoff hat nicht nur einen unangenehmen Geruch, sondern ist auch hochgiftig [4]. Nicht, dass jemals jemand daran gestorben wäre - eine kleine Gnade.
Ein weiterer Stoff in der Galerie der Schurken ist Cadmium, am besten bekannt aus Nickel-Cadmium-Akkus, aber auch in LDRs, Cadmium-Tellurid-Solarzellen, Sensoren, transparenten Leitern, manchmal in SMD-Widerständen und sogar in PVC-Drahtisolierungen enthalten. Das bedeutet, dass es fast überall vorkommt.
Quecksilber, das früher in Batterien, Quecksilberschaltern, Nixie-Röhren, Energiesparlampen, Leuchtstoffröhren, Quecksilberlampen und Quecksilberdampfgleichrichtern verwendet wurde, ist ein bekannter Gefahrstoff. Leuchtstofflampen sind immer noch auf dem Markt, und Onkel Ali ist Ihr Freund, wenn Sie auf der Suche nach Quecksilberschaltern sind. Antike krakenartige Quecksilberdampfgleichrichter sind mit ihrem gespenstischen blauen Leuchten besonders faszinierend, wirken sogar ein wenig außerirdisch (**Bild 2**) [5]. Wenn Sie einen von ihnen kaputt machen, ist es an der Zeit, den Notarzt herbeizurufen.

Schließlich enthalten die Leuchtstoffe in den heutigen LED-Lampen eine ganze Reihe exotischer Elemente, darunter die Seltenerdmetalle Yttrium, Cerium und Europium. Obwohl diese laut ihren Datenblättern nicht sehr giftig aussehen, ist das Fehlen umfangreicher toxikologischer Daten ein Grund zur Sorge.

Patriotismus

Neben den bereits erwähnten giftigen Substanzen enthalten vor allem ältere Bauteile manchmal radioaktive Stoffe. So sind beispielsweise einige Röhrenwendeln mit Materialien wie Thoriumoxid beschichtet oder bestehen aus einer Thorium-Wolfram-Legierung - so auch die Wendel des Magnetrons in Ihrem Mikrowellenherd. Uran findet sich in Form von Urandioxid in den ehemaligen Urdox-PTC-Widerständen (**Bild 3**) und als Uranglas bei bestimmten Röhren. Und dann gibt es noch eine ganze Reihe von radioaktiven Vakuumröhren, Radarröhren, Überspannungsschutzgeräten und gasgefüllten Funkenstrecken [6]. So ziemlich alles, was Sie sich vorstellen können - Leuchtstofflampen, Leuchtstofflampenstarter, Metallhydridlampen und sogar Nixie-Röhren - kann radioaktives Material enthalten.

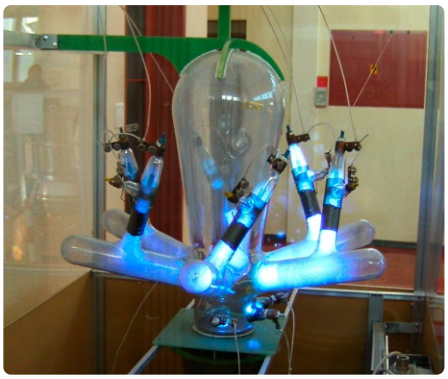


Bild 2. Ein Quecksilberdampf-Gleichrichter. Quelle: Wikipedia / Timberwind / Public Domain.

Bild 3. Ein Urdox-PTC-Widerstand. Foto: Ilse Joostens.

Auch industrielle Sensoren wie Dickenmessgeräte, Konzentrationsmessgeräte oder Sensoren zur Messung des Feuchtigkeitsgehalts von Materialien enthalten manchmal radioaktive Stoffe. Die Wahrscheinlichkeit, dass Sie mit diesen Geräten in Berührung kommen, ist nicht sehr groß, aber das gilt nicht für die (inzwischen verbotenen) Ionisationsrauchmelder. Früher gab es einige Typen mit Radium-226, aber neuere Geräte enthalten Americium-241. Das erinnert mich an den Mann, der versehentlich die radioaktive Quelle aus dieser Art von Rauchmelder in einem Cannabisverdampfer geraucht hat. Er wurde davon nicht high, aber dem Hörensagen nach hatte es einen seltsamen Geschmack [7].
Auch die ehemalige Sowjetunion hatte ihre Rauchmelder, aber aus patriotischen Gründen war Americium-241 tabu, also wurde stattdessen reaktorfähiges Plutonium verwendet, und zwar eine ganze Menge davon - bis zu 18,5 MBq im KI-1-Rauchmelder [8]. Damals waren die elektronischen Bauteile nicht sehr empfindlich, so dass halt eine starke Quelle erforderlich war. Dennoch ist es eine seltsame Erkenntnis, dass so etwas in vielen Regierungsgebäuden über dem Kopf schwebte.

Ich hoffe, dass Sie nach all dem noch gut schlafen können. In der Praxis sind die Dinge nicht so schlimm, solange Sie nichts Verrücktes damit tun und nicht zu viel magischen Rauch einatmen. Allerdings gibt es ein Problem für die Arbeiter, die Elektronikschrott recyceln. Dies soll also ein Plädoyer für nachhaltigere Elektronikprodukte sein.

RG — 230411-02

WEBLINKS

[1] PFAS: chips' poisonous ingredient that doesn't go away: https://euractiv.com/section/digital/news/pfas-chips-poisonous-ingredient-that-doesnt-go-away/
[2] Wikipedia: Bromierte Flammschutzmittel : https://de.wikipedia.org/wiki/Flammschutzmittel
[3] Environews Focus: „Beryllium: A Chronic Problem": https://ehp.niehs.nih.gov/doi/pdf/10.1289/ehp.94102526
[4] YouTube: BigCliveDotCom: Selenium rectifiers - the smelliest components ever: https://youtu.be/OOA1NaoKV6I
[5] YouTube: Kempton Steam Museum: The Mercury Arc Rectifiers: https://youtu.be/YhaQqgXrMMU
[6] YouTube: AE Laboratories: Radioactive Vacuum Tubes: https://youtu.be/m-6-aJv7GCI
[7] YouTube: That Chemist: Somebody Vaped a Smoke Detector: https://youtu.be/14UggnpN39w?t=627
[8] Special Nuclear Material: Analysis of Soviet smoke detector plutonium: https://carlwillis.wordpress.com/2017/02/07/analysis-of-soviet-smoke-detector-plutonium/

Ist Mobilfunk die **energiesparendste Option** für das IoT?

LTE-M und NB-IoT: Energieanforderungen in LPWAN-Implementierungen

Von Stuart Cording (Elektor)

Wenn Sie Ihr nächstes IoT-Projekt in Angriff nehmen, lohnt es sich, einen Blick auf das zellulare LPWAN-Angebot zu werfen. Während LoRaWAN auf dem Papier energiesparend zu sein scheint, haben Forscher bei realen Einsätzen starke Schwankungen in der Batterielebensdauer festgestellt. LTE-M und NB-IoT sind beide durchaus wettbewerbsfähig, wenn es um das Energiebudget geht, und bieten darüber hinaus eine Reihe weiterer Vorteile. Aber wie alle guten Dinge haben auch sie ihre eigenen Herausforderungen zu bewältigen.

Die Fähigkeit von Mobilfunknetzen, weltweiten Zugang zu Konnektivität zu bieten, wurde mir in den frühen 2000er Jahren klar. Ich genoss es, während einer Geschäftsreise durch die Schweizer Alpen einmal Passagier zu sein, als ein Kollege anrief und um eine Präsentation für ein Kundenmeeting bat. Mit meinem zuverlässigen Ericsson T68i auf dem Armaturenbrett, einem frühen Bluetooth-fähigen Mobiltelefon und dem Laptop zwischen den Knien machte ich mich daran, die Datei mit den blitzschnellen 115 kbps von GPRS zu versenden. Zugegeben, es brauchte mehrere Versuche, weil die Verbindung in den Tunneln abbrach, doch die prinzipielle Möglichkeit, mitten im Nirgendwo (die Schweizer mögen mir verzeihen) mit der Welt verbunden zu sein, zeigte mir, dass das Zeitalter der grenzenlosen drahtlosen Konnektivität angebrochen war.

Ein erster Schimmer des IoT

Dank Bluetooth konnte ich auf das Telefon auch über ein Terminalfenster zugreifen, wie bei den alten kabelgebundenen Modems, die durch die DSL-Technologie ersetzt worden waren. Das bedeutete, dass ein Bluetooth-fähiger Mikrocontroller und eine Software, die mit AT-Befehlen umgehen konnte, alles waren, was man brauchte, um sich mit der Welt zu verbinden. Wir waren schon dabei, das Internet der Dinge (IoT) zu implementieren, wir wussten es damals nur noch nicht. Jetzt, 20 Jahre später, ist das IoT etabliert, und wir haben noch bessere Mobilfunknetze. Aber wir würden uns wahrscheinlich schwer tun, ein Produkt oder eine Anwendung zu nennen, bei der es sich nicht um ein Smartphone oder ein Tablet handelt, das eine Mobilfunkverbindung für die Datenübertragung nutzt.

Eine der Herausforderungen in Bezug auf die Akzeptanz könnte die Verwirrung sein, die das gesamte IoT-Ökosystem mit seiner Nomenklatur stiftet, so dass es schwierig ist zu wissen, was man wählen soll und warum. Auf der obersten Ebene sind wir an den Übergang von 4G zu 5G gewöhnt, aber diese Marketingbegriffe sind für Verbraucher und Unternehmen nur für Smartphone- und Hochgeschwindigkeitsdatenverbindungen relevant. Für Anwendungen mit niedrigeren Datenraten, wie sie in der Maschinenkommunikation verwendet werden, gilt es separate Standards zu berücksichtigen.

Zellulares IoT mit höherer Datenrate

Der erste ist LTE-M, was für *Long-Term Evolution Machine Type Communication* steht. LTE-M verzweigt sich in zwei aktuelle Standards, *LTE Cat M1* und *LTE Cat M2*. Das *3rd-Generation Partnership Project* (3GPP) definiert diese Standards, wobei neue Funktionen in „Releases" ratifiziert werden. So war LTE Cat M1 Teil von Release 13 im Jahr 2015 und LTE Cat M2 Teil von Release 14 im Jahr 2017. Cat M1 bietet 1 Mbit/s Uplink und Downlink, während Cat M2 etwa 7 Mbit/s Uplink und 4 Mbit/s Downlink bietet. Beide unterstützen Voll- und Halbduplex (**Bild 1**). Zum Vergleich: 5G-Smartphone-Netzwerke bieten durchschnittlich etwa 100 Mbit/s [1].

Der Vorteil einer niedrigeren Datenrate liegt teilweise im geringeren Energiebedarf der LTE-M-Hardware. Gemäß der 3GPP-Spezifikation wurde eine Betriebsdauer von zehn Jahren mit einer 5-Wh-Batterie angestrebt. Auch wenn diese Lebensdauer erreichbar ist, stellt Google-Ingenieur Brian Ray [2] fest, dass das Erreichen einer Sendeleistung von 23 dBm während eines Uplinks, der höchsten unterstützten Stufe, zu Spitzenströmen von etwa 500 mA führt. Dies stellt eine nicht unbedeutende Herausforderung für die Entwicklung dar.

Die Abdeckung ist dank eines höheren maximalen Kopplungsverlusts (Maximum Coupling Loss, MCL) ebenfalls besser als bei Standard-LTE. Dieser Wert definiert den Punkt, an dem ein drahtloses System seine Fähigkeit verliert, seinen Dienst zu erbringen. In einer Studie von Sierra Wireless [2] wurde ein MCL von bis zu 164 dB für LTE Cat M1 ermittelt, was eine erhebliche Verbesserung gegenüber den 142 dB von herkömmlichem LTE darstellt und auch deutlich besser ist als das von 3GPP selbst gesetzte Ziel von 155,7 dB. In Bezug auf die Leistungsfähigkeit bedeutet dies eine bessere Konnektivität

	LTE-M		NB-IoT	
	LTE Cat M1	LTE Cat M2	LTE Cat NB1	LTE Cat NB2
3GPP Release	Release 13	Release 14	Release 13	Release 14
Peak Uplink Rate	1 Mb/s	7 Mb/s	66 kb/s	160 kb/s
Peak Downlink Rate	1 Mb/s	4 Mb/s	26 kb/s	127 kb/s
Voice over LTE	yes	yes	no	no
Duplex	full / half	full / half	half	half
Latency	<15 ms	<15 ms	<10 s	<10 s

Bild 1. LTE-M bietet mobiles IoT mit höheren Datenraten und VoLTE, während NB-IoT auf statische Anwendungen abzielt.

innerhalb von Gebäuden, was für intelligente Messanwendungen wichtig ist, bei denen das Gebäude einen Durchdringungsverlust von 50 dB verursachen kann, sowie natürlich auch eine bessere Reichweite im Freien.

Wie Smartphones unterstützt LTE-M auch mobile Daten, das heißt, Ihr Gerät verbindet sich ständig mit dem nächstgelegenen Mobilfunkmast, was es ideal für Sensoren macht, die zum Beispiel den Weg verderblicher Waren überwachen oder Fahrzeugflotten verfolgen. Außerdem ist Voice-over-LTE (VoLTE) enthalten, was nützlich ist, wenn Sie gelegentlich Sprache als Teil des Systems benötigen, zum Beispiel in einer Brandmeldezentrale oder einem Überwachungssystem für ältere Menschen. Und schließlich könnte LTE-M mit einer Latenzzeit von unter 15 ms eine Anwendung unterstützen, die unmittelbar auf Menschen zu reagieren scheint.

Zellularfunk für statische IoT-Knoten

Die alternative IoT-Technologie für Mobilfunk ist NB-IoT. Auch diese Technologie gibt es in zwei Varianten. LTE Cat NB1 wurde in Release 13 formalisiert, während LTE Cat NB2 seit Release 14 verfügbar ist. NB-IoT zielt auf Anwendungen ab, die sich nicht bewegen, zum Beispiel intelligente Zähler und Sensoren in der Landwirtschaft, Wetterstationen oder in Wasseraufbereitungsanlagen. Eine Übergabe von Sendemasten und VoLTE wird nicht unterstützt (**Bild 2**). Stattdessen kommt es nur zu einer Stromverbrauchsspitze, wenn sich das Gerät beim nächstgelegenen Sendemast anmeldet, danach kann das Funkmodul in einen Ruhemodus übergehen und weiß, dass es nach dem Aufwachen dort weitermachen kann, wo es aufgehört hat. Die Datenraten von NB-IoT sind viel niedriger als die von LTE-M. LTE Cat NB1 erreicht 26 kbit/s im Downlink und bis zu 66 kbit/s im Uplink, während LTE Cat NB2 127 kbit/s im Downlink und etwa 160 kbit/s im Uplink bietet. Im Gegensatz zu LTE-M unterstützt NB-IoT nur den Halbduplex-Modus. Auch die Latenzzeit ist viel höher: in der Regel werden 1,6 s bis 10 s angegeben. In Anbetracht der beabsichtigten Anwendungen dürfte es jedoch kein Problem sein, nach dem Empfang einiger Sensordaten mit einer neuen Antriebseinstellung für ein Gewächshausfenster oder eine Wasseraufbereitungsschleuse zu reagieren. Es ist auch anzumerken, dass dies eine viel bessere Latenzzeit ist als bei konkurrierenden Technologien des Low-Power-Wide-Area-Networking (LPWAN) wie LoRaWAN [4] und Sigfox [5].

IoT mit geringer Stromaufnahme über Mobilfunk erreichen

Die Stromaufnahme ist eine der wichtigsten Anforderungen für eine IoT-Anwendung, da sie in erster Linie entweder Batterien oder eine andere erneuerbare Energiequelle wie ein Solarpanel nutzt. Beide Drahtlostechnologien unterstützen verschiedene Stromsparmodi, die den Entwicklern Optionen zur Verlängerung der Batterielebensdauer bieten. Der erste dieser Modi ist der Energiesparmodus PSM, der es der Anwendung ermöglicht, das Mobilfunkmodul in einen tiefen Schlafzustand zu versetzen, was in den meisten Fällen zu einer Stromaufnahme von nur wenigen Mikroampere führt. Das Gerät benachrichtigt seinen Mobilfunkmast über seine Absicht und kann dann bis zu 413 Tage lang schlafen. Während dieser Zeit gibt es keine Möglichkeit, Daten an das Gerät zu übermitteln. Wenn es jedoch aufwacht, muss es sich nicht mehr beim Mobilfunkmast anmelden.

Der nächste Modus ist eDRX (extended Discontinuous Reception). Dieser leichtere Schlafmodus spart bis zu 40 Minuten Strom bei LTE-M oder bis zu drei Stunden bei NB-IoT. Auch das Aufwachen aus dem Ruhezustand vollzieht sich schneller als bei PSM. Diese Stromsparmöglichkeiten sind aber nicht immer und überall verfügbar. Ihre Konfiguration hängt von den Einrichtungen des Dienstanbieters ab [6], was bedeutet, dass die Akkulaufzeit in einem Land kürzer sein kann als in anderen, weil eine Einstellung nicht ausgehandelt werden kann. Die Nutzung eines Dienstanbieters, der eine Roaming-SIM anbietet, kann auch den Zugang zu diesen stromsparenden Funktionen einschränken [7].

Da die IoT-Implementierung in Mobilfunknetzen so unterschiedlich ist und von so vielen Faktoren abhängt, ist es kein Wunder, dass die Suche nach Hinweisen zur Stromaufnahme ein ziemlich fruchtloses Unterfangen ist. Meistens liefert eine Google-Suche Seiten mit der Aussage „10 Jahre Batterielebensdauer“, anscheinend sowohl für LTE-M als auch für NB-IoT und ohne Details zur Batteriekapazität zu nennen.

Zellulares IoT im Labor

Zum Glück haben sich Teams an verschiedenen Instituten die Zeit genommen, den Stromverbrauch zu untersuchen. Die Ergebnisse solcher Untersuchungen geben einige Hinweise darauf, was zu erwarten ist, und zeigen, wie das mobile IoT im Vergleich zu den

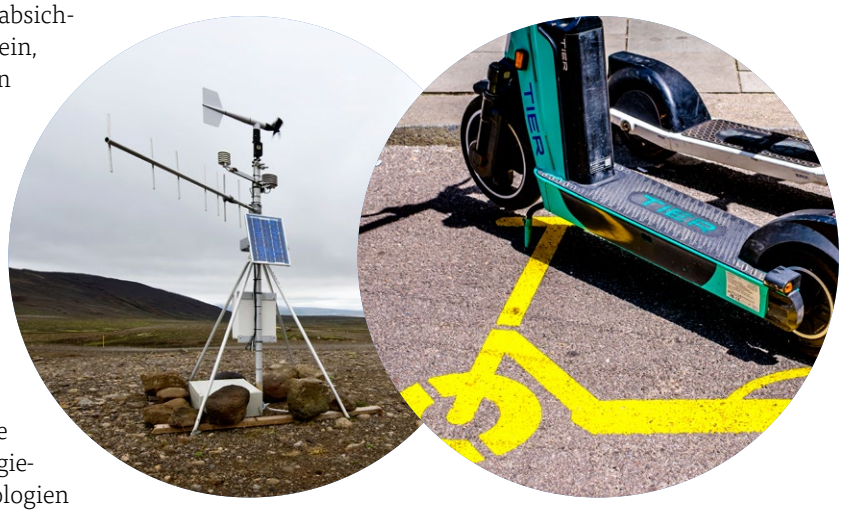


Bild 2. Mobile Anwendungen wie die gemeinsame Nutzung von E-Scootern sind mit LTE-M besser bedient. Eine statische Anwendung, zum Beispiel eine Wetterstation, profitiert von dem geringeren Energiebedarf von NB-IoT. (Quelle: Shutterstock)

Alternativen abschneidet. Tan [8] vergleicht zum Beispiel LoRa und NB-IoT. In dem Experiment werden MQTT-Pakete mit jeweils 50 Byte Daten gesendet. Bei einer optimalen Konfiguration und guten Verbindungsbedingungen benötigt NB-IoT etwa 200 mJ pro Transaktion. Bei LoRa hingegen hat der Entwickler mehr Kontrolle über die Übertragungskonfiguration, indem er einen Spreizfaktor (SF) einstellt. Bei SF7 ist die Bitrate höher, was kürzere Sendezeit bedeutet, während bei SF12, der niedrigsten Einstellung, die Sendezeit (bei gleicher Bitzahl) am längsten ist. Das Risiko besteht jedoch darin, dass SF7 die Reichweite für eine erfolgreiche Übertragung zu stark verringert, so dass der Datenaustausch wiederholt werden muss. Bei den Tests benötigte SF7 nur 100 mJ pro Transaktion, die Erhöhung auf SF12 jedoch 250 mJ.

Die Schlussfolgerung ist, dass eine 3.000-mAh-Batterie bei LoRa mit SF7 diese Konfiguration mehr als 32 Jahre lang mit Strom versorgen könnte, aber mit SF12 reduziert sich dies auf knapp 13 Jahre. Im Vergleich dazu könnte NB-IoT eine Betriebsdauer von knapp 20 Jahren ermöglichen. In Anbetracht der Tatsache, dass LoRa unter realen Bedingungen seine SF- und andere Transceiver-Konfiguration (Bandbreite) anpassen müsste, um erfolgreiche Datenübertragungen aufrechtzuerhalten, kann eine solche Variation der Batterielebensdauer als zu riskant angesehen werden.

Vergleich mit LoRaWAN

Dieses Risiko wird auch in der Forschung eines Teams der Universität Antwerpen [9] hervorgehoben. In ihrer Schlussfolgerung stellen sie fest, dass LoRaWAN unter kontrollierten Laborbedingungen zwar die geringste Stromaufnahme hatte, was eine jahrelange Anwendungsdauer verspricht, aber „beim Einsatz in Echtzeit (sic!) dieser Wert auf ein paar Monate schrumpft." Sie testeten auch NB-IoT im Vergleich zu Sigfox und DASH7 [10]. Während DASH7 unter den gleichen Bedingungen einen noch günstigeren Stromverbrauch als LoRaWAN aufwies, mutmaßte das Team, dass NB-IoT trotz der etwas höheren Stromaufnahme die bessere Option sein könnte. NB-IoT könnte einfach mehr Kriterien erfüllen, wenn man alles berücksichtigt, was eine IoT-Anwendung benötigt, zum Beispiel Verfügbarkeit, Latenzzeit, Abdeckung, Sicherheit, Robustheit und Durchsatz.

Die Drahtlostechnologie hat sich in den letzten zwei Jahrzehnten dank CMOS-Funktransceivern, hoch integrierten Funkmodulen und winzigen Antennen demokratisiert. Selbst die Software-Stacks sind oft frei verfügbar. Die Entwickler arbeiten jedoch oft nur am Endknoten und sind auf andere angewiesen, wenn es um die Infrastruktur geht, mit der sie sich verbinden. Obwohl LTE-Mobilfunknetze für Smartphone-Nutzer allgegenwärtig und einfach zu bedienen sind, gilt dies nicht für diejenigen, die sich für das Internet der Dinge darauf verlassen wollen.

Nichts für schwache Nerven

Für Uneingeweihte ist es schwierig, verlässliche Informationen darüber zu finden, was das mobile IoT kann und was nicht, ob wichtige Energiesparfunktionen weltweit verfügbar sind oder wie man ihre Verfügbarkeit feststellen kann. Das lässt das mobile IoT wie das arme Geschwisterchen der Smartphone-Industrie aussehen. Die jüngste Nachricht, dass Vodafone sein auf IoT-Dienste spezialisiertes Geschäft verkaufen will, trägt nicht dazu bei, dieses Bild zu verbessern. Obwohl Vodafone im vergangenen Jahr 150 Millionen IoT-SIM-Verbindungen verkauft hat [11], macht diese Sparte nur 2 % des Serviceumsatzes aus.

Obwohl Untersuchungen zeigen, dass das mobile IoT im Vergleich zu alternativen LPWAN-Lösungen in puncto Stromaufnahme und Batterielebensdauer wettbewerbsfähig ist, hat es immer noch seine Schwächen. Da die Netzinfrastruktur einiger Anbieter und in einigen Ländern Stromsparmaßnahmen nicht durchgängig unterstützen, dürften sich Techniker weiterhin schwer tun, konkrete Aussagen zur Batterielebensdauer zu machen, die für die Kunden attraktiv sind. Klar ist, dass jeder LPWAN-Ansatz Nachteile hat, und die Entwicklungsteams müssen diese je nach Anwendungsfall gegeneinander abwägen. Und es scheint, dass noch ein gerüttelt Maß an Forschungszeit erforderlich ist, um einen möglichst direkten Vergleich der am besten geeigneten LPWAN-Lösungen zu erhalten.

RG — 230376-02

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] „5G vs 4G: What's the difference?", EE Limited, September 2020: https://bit.ly/3MTuUYd
[2] B. Ray, „What is LTE-M?", Medium, Mai 2017: https://bit.ly/45QDq2U
[3] G. Vos et al., „Coverage Analysis of LTE-M Category-M1", Sierra Wireless, Januar 2017: https://bit.ly/3OYpoGG
[4] „What are LoRa and LoRaWAN?", The Things Network: https://bit.ly/43EgYc5
[5] Sigfox-Website: https://sigfox.com/
[6] „PSM and eDRX: Power saving in cellular LPWAN - possibilities and limitations", 1NCE GmbH: https://bit.ly/43p6a10
[7] P. Marshall, „Sleeping Battery: How eDRX and PSM Can Save Energy in LPWA IoT Edge Devices", Eseye, Mai 2021: https://bit.ly/3Ne1WUh
[8] L. Tan, „Comparison of LoRa and NBIoT in Terms of Power Consumption", KTH Royal Institute of Technology, Januar 2020: https://bit.ly/43mXrwl
[9] R. K. Singh et al., „Energy Consumption Analysis of LPWAN Technologies and Lifetime Estimation for IoT Application", Sensors (Basel, Schweiz), August 2020: https://bit.ly/3qvfmm7
[10] Website der DASH7-Allianz: https://www.dash7-alliance.org/
[11] M. Kleinman, „Vodafone dials up sale of stake in £1bn Internet of Things unit", Sky UK, Mai 2023: https://bit.ly/3MQZUbt

Kabellose Kommunikation in IoT-Systemen

mit MKR-Modulen von Arduino

Kommunikationsstandards der Arduino-Module für IoT

Von Transfer Multisort Elektronik Sp. z o.o.

In diesem Artikel geben wir einen kurzen Überblick über ausgewählte Arduino-Entwicklungskits aus der MKR-Familie für das Rapid Prototyping von IoT-Geräten, die drahtlose Kommunikationsstandards wie WiFi/ Bluetooth, LoRaWAN/Sigfox, GSM/3G oder NB-IoT nutzen.

Eines der größten Probleme, mit denen der Gerätemarkt des Internet der Dinge (IoT) [1] (IoT) derzeit konfrontiert ist, ist seine hohe Fragmentierung. Die Vielzahl von Geräten und Kommunikationsprotokollen macht es sehr schwierig, ein einheitliches und funktionales System aufzubauen, wenn wir uns für die Verwendung von Komponenten verschiedener Hersteller entscheiden. Es gibt viele Gründe für diese Aufspaltung und sie hängen nicht immer nur mit dem Wunsch der Designer zusammen, ihre eigenen lizenzierten Lösungen zu erzwingen. Der Begriff IoT umfasst viele Arten von Geräten. Dies können beispielsweise kleine Messsensoren sein, die mit alternativen Energiequellen betrieben werden und den Austausch einer kleinen Datenmenge über große Entfernungen erfordern, aber auch Fernkameras, die hochauflösende Bilder in Echtzeit übertragen. Die Spezifität des entworfenen Geräts zwingt daher die Entwickler, eine geeignete drahtlose Kommunikationstechnologie auszuwählen, die den Anforderungen des zu entwerfenden Geräts entspricht. Man muss unter anderem die Akkulaufzeit, die Kommunikationsreichweite oder zu übertragene Datenmenge berücksichtigen. Als Reaktion auf die Marktanforderungen stellten die Hersteller von Entwicklungskits, (einschließlich der Plattformen von Arduino [2]) sicher, dass ihr Portfolio so vollständig wie möglich der Bandbreite an Anforderungen entspricht. In diesem Artikel präsentieren wir eine kurze Beschreibung ausgewählter Arduino-Entwicklungskits aus der MKR-Familie, die für das Rapid Prototyping von IoT-Geräten mit drahtloser Kommunikation in Standards wie WiFi, Bluetooth, LoRaWAN/Sigfox, GSM/3G oder NB-IoT vorbereitet wurden.

Kommunikation über WLAN/ Bluetooth mit Arduino MKR 1000/1010

Die Kommunikation im ISM-2,4-GHz-Frequenzband unter Verwendung von WiFi- und Bluetooth-Standards ist seit mehreren Jahren auf dem IoT-Gerätemarkt präsent. Für die Anforderungen einer schnellen

Implementierung von Hardware-/Software-Prototypen mithilfe von WiFi-Kommunikation hat Arduino die Entwicklungskits MKR WLAN 1000 [3] und MKR WLAN 1010 [4] entwickelt. Das erste dieser Kits basiert auf dem Modul ATSAMW25 [5], das den SAMD21-Mikrocontroller, das Funkmodul WINC1500 [6] und Autorisierungsystem ECC508 [7] enthält. Das Set in der Version MKR 1010 ist mit einem NINA-W102-Funkmodul von u-blox [8] ausgestattet, das Bluetooth-/BLE-Kommunikation bietet.

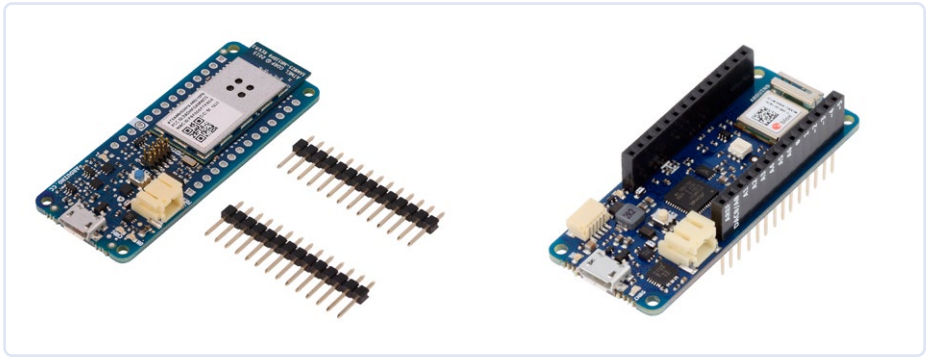
Von der Softwareseite bietet das Unternehmen Arduino die WiFi101-Bibliothek für Module MKR WiFi 1000 an, die die Verschlüsselung nach WEP und WPA2 Personal unterstützt. Für das Modul MKR WiFi 1010 (und andere Kits, die auf dem Modul u-blox NINA-W102 basieren, einschließlich Arduino NANO 33 IoT [9]) hat der Hersteller die WiFiNINA-Bibliothek sowie eine Reihe von Beispielanwendungen vorbereitet, die die Integration mit Android IoT Cloud und Azure, AWS IoT Core, Google Firebase oder Blynk präsentieren.

LoRaWAN- und Sigfox-Kommunikation – die Module Arduino MKR WAN 13x0 und FOX 1200

Die dynamische Entwicklung von IoT-Systemen hat zu einem erhöhten Interesse am Thema Smart Cities geführt. Leider ist die Kommunikation unter Verwendung der WLAN-/Bluetooth-/BLE-Standards lokal und erfüllt nicht alle Anforderungen, die für Projekte der Gruppe „Smart City" festgelegt wurden (zu denen unter anderem umfangreiche Netzwerke von Verschmutzungssensoren, Wasserstandüberwachung oder Parkplatzbelegung gehören). Die Lösung für die Probleme könnte die Verwendung eines der beiden beliebtesten Kommunikationsstandards im Bereich LPWAN (Low Power Wide Area Network) sein - LoRaWAN oder Sigfox, mit denen eine kleine Datenmenge über große Entfernungen übertragen werden kann. Für das Rapid Prototyping von Geräten mit LoRa/LoRaWAN-Kommunikation haben Arduino-Designer Entwicklungskits MKR WAN 1300 [10], und seinen Nachfolger MKR WAN 1310 [11] vorbereitet. Beide Module basieren auf dem Mikrocontroller Atmel SAMD21, der in anderen Modulen der Serie MKR von Arduino verwendet wird, sowie auf dem Funkmodul Murata CMWX1ZZABZ. Die neuere Version des Moduls wurde zusätzlich mit 2 MB Flash-Speicher, einem neuen Batterieladesystem und für geringen Stromverbrauch optimierten Stromversorgungssystemen ausgestattet.

Die Module MKR WAN 13x0 funktionieren mit der vom Hersteller bereitgestellten Arduino IoT Cloud. Die Ganzheitlichkeit der angebotenen Lösungen wird durch das für die Moduke MKR WAN 1310 optimierte Access Gate Arduino Pro Gateway LoRa Connectivity [12] ergänzt.

Eine interessante Alternative zur LoRa/LoRaWAN-Kommunikation ist der Sigfox-Standard, bei dem die Kommunikation von Knoten zum Access Gateway besonders im Vordergrund steht. Das Angebot von Arduino umfasst das Modul MKR FOX 1200 [13], das auf der Basis eines Mikrocontrollers Atmel SAMD21 aufgebaut wurde.



Bilder 1 and 2. Die Arduino-Module MKR WAN 1000 (links) und MKR WAN 1010 (rechts).

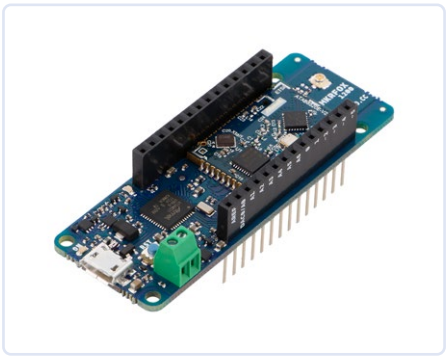
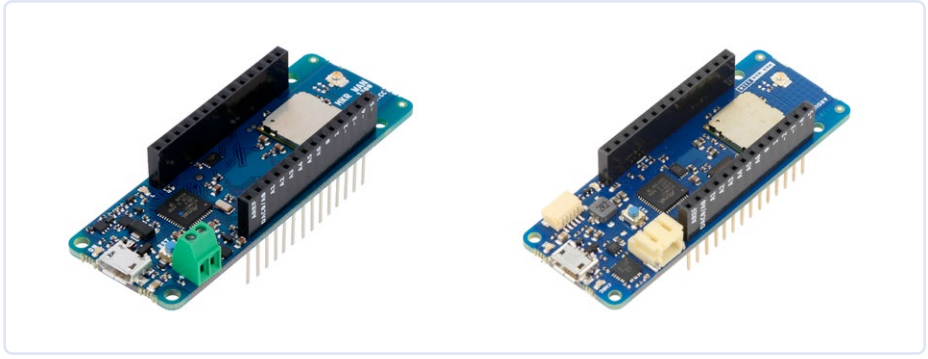


Bild 5. Modul MKR FOX 1200 von Arduino für die Kommunikation im Sigfox-Netzwerk.



Bilder 3 and 4. Die Arduino-Module MKR WAN 1300 (links) und MKR WAN 1310 (rechts).

Bild 6. Modul MKR GSM 1400 von Arduino für die Kommunikation im GSM/3G Netzwerk.

Das System Microchip Smart RF ATA8520 ist für die Funkkommunikation verantwortlich, deren Funkpfad auf die in Europa geltende ISM-Frequenz 868 MHz abgestimmt wurde.

GSM/3G-Kommunikation – das Modul MKR GSM 1400 von Arduino

Selbst ein umfangreiches Mesh-Netzwerk, das im LoRa/LoRaWAN-Standard arbeitet, kann derzeit keine globale Reichweite bieten. Bei IoT-Projekten, die ein nahezu unbegrenztes Kommunikationsgebiet erfordern, ist die Verwendung des GSM/3G-Standards die beste Lösung. Für die GSM/3G-Kommunikation hat Arduino das Modul MKR GSM 1400 [14] vorbereitet, ausgestattet mit einem Modem SARA-U210 der Marke u-blox und einem Autorisierungssystem Microchip ECC508 zum Implementieren von Kommunikationssicherheitsmechanismen. Das eingebaute GSM-Modem bietet eine Abdeckung der Kommunikation in den Bändern GSM 850 MHz, E-GSM 1900 MHz, DCS 1800 MHz und PCS 1900 MHz.
Um den Software-Vorbereitungsprozess zu verbessern, stellt der Hersteller die Bibliothek MKRGSM (die den Programmierer von der Bedienung des Moduls mit Low-Level-AT-Befehlen entbindet) zusammen mit einer umfassenden Reihe von Beispielen (u.a. GPRS-Verbindungen, Versenden/Empfangen von Textnachrichten, Bedienung von Sprachanrufen). Das Modul MKR GSM 1400 kann sowohl mit der Software Arduino IoT Cloud als auch mit alternativen Cloud-Lösungen arbeiten: Google IoT Cloud, Blynk oder SORACOM Air IoT, für die der Hersteller eine Reihe von Beispielimplementierungen vorbereitet hat.

Kommunikation im Narrowband-IoT: Arduino–Modul MKR NB 1500

Bei einer kurzen Beschreibung ausgewählter Kommunikationsstandards innerhalb der Internet of Things-Geräte können Lösungen, die auf dem Narrowband-IoT-Standard (NB-IoT) basieren und das lizenzierte LTE 800 MHz-Band für die Kommunikation verwenden, nicht ignoriert werden. Wie LoRaWAN- und Sigfox-Lösungen ist NB-IoT Teil der LPWAN-Netzwerkgruppe und sorgt so mit energiesparenden Funkmodulen für eine stabile Kommunikation über große Flächen und für eine langjährige Arbeit im Batteriebetrieb. Damit ist sie eine weitere Alternative zur LoRaWAN- und Sigfox-Kommunikation in Lösungen aus dem „Smart City"-Segment. Für das schnelle Prototyping von Endknoten, die im NB-IoT-Standard arbeiten, hat das Unternehmen Arduino einen Satz MKR NB 1500 [15], ausgestattet mit dem Modul u-blox SARA-R410M-02B [16], das die Konnektivität nach LTE/Cat M1/NB1 in den Bändern 1, 2, 3, 4, 5, 8, 12, 13, 18, 19, 20, 25, 26 und 28 ermöglicht. Zusätzlich ist das Set MKR NB 1500 mit dem Autorisierungssystem ECC508 von Microchip [17], einem MicroSIM-Kartenanschluss, einem Li-Po-Batterieladecontroller und einem externen Antennenanschluss ausgestattet.

Bild 7. Modul Arduino MKR NB 15 für die Kommunikation in Netzwerken des Narrowband IoT.

230468-02

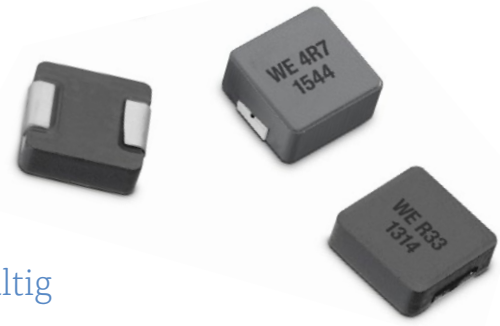
Transfer Multisort Elektronik Germany GmbH
Dohnanyistraße 28-30
04103 Leipzig
tme@tme-germany.de
www.tme.eu/de

WEBLINKS

[1] Internet der Dinge: https://www.tme.eu/de/katalog/embedded-systems-und-iot_113611/
[2] Arduino: https://tinyurl.com/27sp7jxp
[3] MKR1000 WIFI: https://tinyurl.com/2abbhmt3
[4] MKR WIFI 1010: https://tinyurl.com/2w8zeete
[5] Modul ATSAMW25: https://tinyurl.com/2f3kbdvm
[6] Funkmodul WINC1500: https://tinyurl.com/mrft25d8
[7] Autorisierungsystem ECC508: https://tinyurl.com/ycksykda
[8] u-blox: https://www.tme.eu/de/linecard/p,u-blox_1320/
[9] Arduino NANO 33 IoT: https://tinyurl.com/2p92e6zv
[10] MKR WAN 1300: https://tinyurl.com/28h3cx6z
[11] MKR WAN 1310: https://tinyurl.com/3bz2hzss
[12] Arduino Pro Gateway LoRa Connectivity: https://tinyurl.com/2bm47v9e
[13] MKR FOX 1200: https://tinyurl.com/3mfc5vw5
[14] MKR GSM 1400: https://tinyurl.com/bde6bjk3
[15] MKR NB 1500: https://tinyurl.com/3py6f7h6
[16] Modul SARA-R410M-02B: https://tinyurl.com/33mmphfm
[17] Microchip Technology: https://www.tme.eu/de/linecard/p,microchip-technology_632/

AC-Verluste in magnetischen Bauteilen

Vermeiden Sie heiße Induktivitäten!



Von George Slama (Würth Elektronik eiSos)

Eines der vielen Probleme, mit denen ein Entwickler von Stromversorgungen konfrontiert wird, ist die unerwartete Überhitzung eines sorgfältig ausgewählten magnetischen Bauteils. Dieses unerklärliche Verhalten kann Verwirrung und Frustration hervorrufen und den Druck erhöhen, das Projekt rechtzeitig und innerhalb des Budgets abzuschließen. Die genaue Abschätzung der AC-Verluste ist ein komplexes Thema und erfordert starke Nerven. Glücklicherweise gibt es eine einfachere Lösung.

Obwohl diese Baugruppen oberflächlich betrachtet oft einfach aussehen, ist die Entwicklung einer Stromversorgung ein komplexes Projekt mit vielen sich widersprechenden Anforderungen, die vom Endverbraucher und verschiedenen Aufsichtsbehörden gestellt werden. Der Endverbraucher möchte die kostengünstigste Lösung, oft in kompakter Bauweise und höchster Zuverlässigkeit. Aufsichtsbehörden verlangen Mindeststandards für Isolierungen und Kriechstrecken, was zu größeren Komponenten führt. Der Konstrukteur muss über ein breites Wissen in vielen Bereichen verfügen, darunter elektrische und mechanische Theorie und Steuerungstheorie. Zusätzlich gibt es das Mysterium der Elektromagnetik, bei dem unsichtbare Kräfte auf Materialien einwirken, um Energie zu speichern oder umzuwandeln, basierend auf uralten kryptischen Formeln aus der Vergangenheit.

Ein Abwärtsregler ist ein gutes Beispiel, da er die am weitesten verbreitete Topologie zur Spannungsreduzierung in nicht isolierten Schaltungen ist. Das Bestreben, die Größe zu reduzieren, wird durch eine Erhöhung der Schaltfrequenz erreicht. Dadurch können passive Komponenten wie Induktivitäten und Kondensatoren kleiner dimensioniert werden, da die Energiespeicherung pro Zyklus geringer ist. Die Größe dieser Komponenten ist proportional zu ihrer Energiespeicherkapazität. Durch die Verkleinerung der Größe verringert sich jedoch die Oberfläche, die für die Ableitung von Verlustwärme zur Verfügung steht, was die thermische Auslegung kritischer macht.

Bei elektromagnetischen Bauteilen gibt es zwei Verlustquellen: die Wicklungen und den Kern. Beide können weiter in Untertypen unterteilt werden (**Tabelle 1**).

Was sind Kernverluste?

Die Ursache für die Kernverluste durch Hysterese liegt in der Bewegung der magnetischen Dipole und der Verschiebung der Domänenwände bei hohen Sättigungsströmen. Wenn ein weichmagnetisches Material von einem äußeren Magnetfeld (Strom in einer Spule oder ein nahes Magnetfeld) beeinflusst wird, richten sich die magnetischen Dipole in den Domänen, die man sich als winzige Magnetbereiche mit Elementarmagneten im Material vorstellen kann, nach dem Feld aus. Dies erfordert Energie und Zeit. Beim Entfernen des äußeren Einflusses reorientieren sich die magnetischen Dipole, und die verschobenen Domänen springen zurück, aber nicht vollständig. Wenn sich die Stromrichtung ändert, kehren die Magnetpole in den Domänen ihre Richtung um, aber nicht vollständig. Die Energie der „Rückfederung" wird dem System zurückgegeben, während der Rest als Arbeit gegen die Verschiebung der Domänenzonen verbraucht und in Wärme umgewandelt wird. Je höher die Frequenz, desto mehr Elementarmagnete in den Domänen und ganze Domänenbereiche werden pro Sekunde verschoben, wodurch exponentiell mehr Energie benötigt wird. Die Bewegung der magnetischen Dipole ist auch proportional zum Flusswechsel. Je größer der Flusswechsel, desto mehr Bewegung und Energie ist erforderlich, die nicht vollständig zurückgewonnen wird. Die Fläche innerhalb der BH-Kurve stellt den Energieverlust pro Zyklus dar.

Wirbelstromverluste entstehen, wenn Wechselströme durch einen Leiter fließen und gemäß dem Faradayschen Gesetz eine Spannung in benachbarte Leiter induziert wird, die proportional zur Änderungsgeschwindigkeit des Magnetfelds ist. Der Kern selbst verhält sich wie eine Wicklung. Ferritmaterial besteht aus isolierten magnetischen Partikeln, die trotz des hohen spezifischen Widerstands bei hohen Frequenzen leitend sind. Dieser „dynamische Widerstand" sinkt mit steigender Temperatur exponentiell. Kürzere Anstiegszeiten führen zu größeren Spannungen. Impulse mit höherer Spannung verursachen exponentiell größere Verluste gemäß der Formel

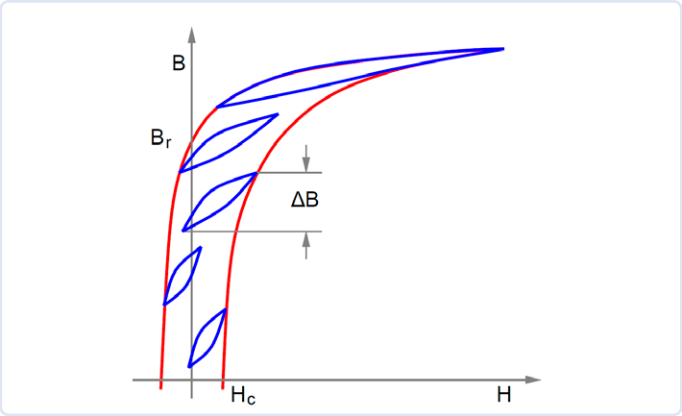


Bild 1. Nebenschleifen an verschiedenen Positionen entlang der Hauptschleife. Sie haben alle den gleichen Spitze-Spitze-Flusshub. Die Schleifenfläche entspricht dem Hysteresekernverlust pro Zyklus.

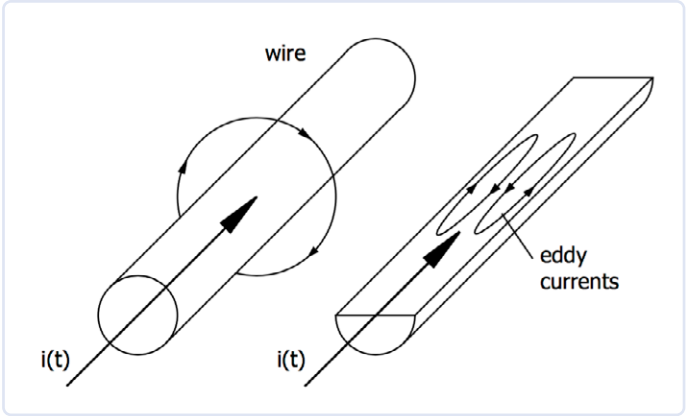


Bild 2. Wechselströme induzieren ein magnetisches Feld, das Wirbelströme in entgegengesetzter Richtung verursacht. Dies hebt den Stromfluss im Zentrum auf und verstärkt ihn in den äußeren Regionen.

$P = (V^2 \times D) / R$
(D = Tastverhältnis, R = Widerstand).

Je nach Topologie muss bei der Abschätzung der AC-Verluste auch der Effekt der Gleichstrom-„Vorspannung" berücksichtigt werden. Obwohl ein statischer Strom keine Bewegung der magnetischen Dipole in den Domänen verursacht, führt ein Wechselstrom mit einer Gleichstromvorspannung (DC Bias) dazu, dass sich die gesamte Hystereseschleife entlang der BH-Kurve verschiebt (**Bild 1**). Messungen zeigen, dass der Einfluss bei niedrigen Gleichströmen gering ist, aber bei höheren Werten eine signifikante Zunahme der Verluste zu verzeichnen ist. Wenn das Kernmaterial nahe der Sättigung ist und fast alle magnetischen Dipole ausgerichtet sind, wird mehr Energie benötigt, um die restlichen magnetischen Dipole der Domänen auszurichten. Herkömmliche Methoden zur Berechnung der AC-Kernverluste berücksichtigen dies nicht, was zu einer unerwarteten Überraschung führen kann. Die Differenz zwischen den berechneten Verlusten und den gemessenen Verlusten wird schließlich als Überschussverluste *) bezeichnet. Dieser entsteht unter anderem durch Relaxationseffekte, überschüssige Wirbelströme, Streuverluste und andere, weniger bekannte Phänomene.

Was sind Wicklungsverluste?

Der DC-Wicklungsverlust ergibt sich aus dem DC-Widerstand des verwendeten Leiters. Dies ist einfach der gemessene Gleichstromwiderstand multipliziert mit dem Gleichstromanteil der Stromkurvenform zum Quadrat, $P = I^2 \times R$. Die AC-Wicklungsverluste bestehen aus Skin- und überwiegend Proximity-Verlusten aus dem AC-Anteil der Stromkurvenform.

Tabelle 1. Verlustquellen, -arten und Einflussgrößen.

Verlustquelle	Stromart	Verlustart	Einflussgrößen:
Kern	AC	Hysterese	Kernmaterial, Temperatur, Signalform, Permeabilität
		Wirbelströme	Angelegte Spannung, Tastverhältnis (Duty Cycle), Widerstand, Permeabilität, Permittivität
	DC bias	(Hysterese)	Verschiebung der Hysteresekurve, Verzerrung kleiner Schleifen, Topologieabhängigkeit
		Überschuss	Unerklärliche Verluste *)
Wicklung	DC	Widerstand	Material, Temperatur
	AC	Skin-Effekt	Frequenz, Signalform (Harmonische), Temperatur, Position der Windung *)
		Proximity-Effekt	Frequenz, Signalform (Harmonische), Position der Windung, Anzahl der Schichten, Temperatur, Streufluss, Schichtung der Wicklungen (Verschachtelung),

** siehe Text*

Der Skin-Effekt entsteht dadurch, dass bei hohen Frequenzen die Stromdichte über den Leiterquerschnitt nicht mehr konstant ist, da der Strom zur Oberfläche des Leiters hin aufgrund der finiten Induktivitäten verdrängt wird. Der Leiter kann als Bündel unendlich vieler einzelner dünner Leiter betrachtet werden, deren Induktivitäten miteinander wechselwirken und dem Strom entgegenwirken (Lenzsche Regel). An der Außenfläche des Leiters ist die Summe dieser Wechselwirkungen aufgrund von Wirbelströmen (**Bild 2**) geringer als im Inneren des Leiters, wodurch die Stromdichte zum Leiterinneren exponentiell abnimmt.
Der Skin-Effekt, wie er üblicherweise definiert wird, gilt nur für einen einzelnen Leiter im freien Raum, weit entfernt von anderen Leitern. Bei Drosselspulen oder Transformatoren, bei denen normalerweise viele Windungen und Lagen eng gewickelter Draht vorhanden sind, wird der Skin-Effekt durch den Proximity-Effekt beeinflusst.
Der Proximity-Effekt beschreibt den Einfluss benachbarter Magnetfelder auf die Ströme in einem Leiter. Es gibt zwei Effekte: Benachbarte Drähte mit gleicher Stromrichtung ziehen sich gegenseitig an (die Felder zwischen ihnen heben sich auf), wodurch die Stromkonzentration an den Innenseiten verringert wird (**Bild 3 A**). Benachbarte Drähte mit entgegengesetzten Strömen stoßen sich gegenseitig ab (die Felder zwischen den Drähten addieren sich), wodurch die Stromkonzentration an den Innenseiten erhöht wird, während die gegenüberliegenden Seiten deutlich weniger Strom führen (siehe **Bild 3 B**).
Bei Übertragern fließt der Strom innerhalb einer Wicklung in die gleiche Richtung, zwischen Primär- und Sekundärwicklungen jedoch in entgegengesetzter Richtung. Bei Induktivitäten mit einer Wicklung fließt der Strom nur in eine Richtung. Die Anzahl der Lagen hat einen großen Einfluss auf die

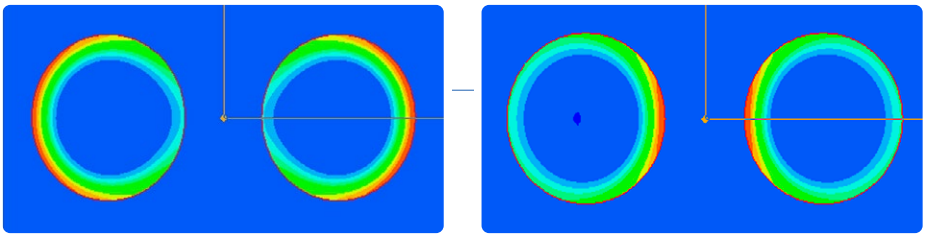


Bild 3. Ströme in benachbarten Drähten. Links - mit Strom in der gleichen Richtung. Rechts - mit Strom in entgegengesetzter Richtung. Rot steht für hohe Stromdichte, blau für niedrige Dichte.

Wechselstromwicklungsverluste, insbesondere bei Drosselspulen, da mit jeder Lage die MMF (Magnetomotorische Kraft) zunimmt und nicht durch eine Sekundärwicklung aufgehoben wird. Mit jeder zusätzlichen Lage steigen die Verluste exponentiell an, da die magnetomotorische Kraft das Produkt des magnetischen Flusses und des magnetischen Widerstandes (Reluktanz) ist.

Aus diesem Grund sind einlagige, kantengewickelte Flachdrahtinduktivitäten (auch bekannt als spiralförmig gewickelte Spulen) in Applikationen als Hochstrominduktivitäten so beliebt. Der Strom, egal ob Wechsel- oder Gleichstrom, fließt immer noch auf dem Weg des geringsten Widerstands, d.h. dem Innendurchmesser, aber die zusätzlichen Verluste durch die Nähe mehrerer Lagen entfallen.

Induktivitäten auswählen mit REDEXPERT

Im Laufe der Jahre hat Würth Elektronik eiSos Tausende von Messungen der Verluste an Induktivitäten aus seinem Portfolio unter realen Einsatzbedingungen durchgeführt - Rechtecksignalformen, Tastverhältnis, Gleichstromvorspannung, Ripplestrom und Temperatur. Dieser umfangreiche Datensatz von AC-Gesamtverlusten umfasst alle Auswirkungen von Konstruktionsmethoden, Drahttypen, Kernmaterial und Erregungssignalformen. Es ist nicht notwendig, langwierige und komplizierte Berechnungen durchzuführen, bei denen oft Informationen fehlen, oder zu versuchen, die Verluste mit Arrays von Widerständen, Induktivitäten und Kondensatoren zu modellieren, um Simulationen für jede mögliche Wahl durchzuführen.

Das Online-Tool zur Auswahl von Induktivitäten, „REDEXPERT" von Würth Elektronik eiSos (**Bild 4**) bietet einen bequemen Zugang zu diesen Daten und die Möglichkeit, verschiedene Induktivitäten unmittelbar zu vergleichen. Sie können den Wandlertyp auswählen und die grundlegenden Betriebsbedingungen eingeben, um eine Liste geeigneter Induktivitäten anzuzeigen. Sortieren und wählen Sie die Induktivitäten aus, die Ihren Anforderungen entsprechen, einschließlich Größe, Höhe und Form. Passen Sie die Betriebsbedingungen einfach an, um die Extremwerte Ihres Entwurfs zu prüfen. Die Diagramme zeigen sofort den gesamten Leistungsbereich einschließlich der Temperatureffekte. So können Sie als Anwender die Vorzüge der einzelnen Induktivitäten zu Ihrer eigenen Zufriedenheit vergleichen. Laden Sie dann nur die Datei mit der am besten geeigneten Induktivität herunter, und bestellen Sie gleichzeitig kostenlose Muster. REDEXPERT speichert Ihre Arbeit automatisch, indem Sie auf das Freigabesymbol in der oberen Menüleiste klicken, wo eine eindeutige URL angezeigt wird. Speichern Sie diese in Ihrem Designbuch, senden Sie sie per E-Mail an sich selbst oder an einen Kollegen, um sie zu teilen. Die genaue Anzeige, die Sie sehen, wird reproduziert, wenn Sie sie erneut brauchen.

230470-02

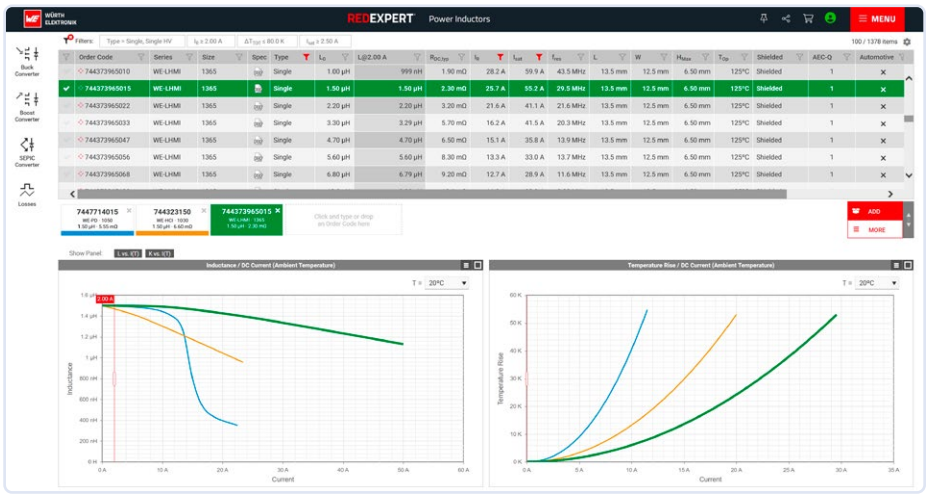


Bild 4. Die Tabellen und Diagramme in REDEXPERT ermöglichen einen schnellen und genauen Vergleich von Induktivitäten.

Über den Autor

George Slama hat während seiner gesamten über 40-jährigen Karriere Transformatoren entworfen und mit ihnen gearbeitet. Seine Konstruktionserfahrung reicht von Milliwatt-Audio- und Telekommunikationstransformatoren bis hin zu Ferroresonanz-, Radar-, Hochspannungs-, kleinen 3-Phasen-, Hochfrequenzschalt- und LTCC-Transformatoren und Induktivitäten. Seine Arbeit umfasste die Qualitätskontrolle, automatisierte Tests und Fertigungstechnik sowie alle Aspekte der Entwicklung und des Designs kundenspezifischer Schaltnetzteile. Slama hat auf Konferenzen in den USA und Europa zahlreiche Seminare über das Design von elektromagnetischen Bauteilen gehalten. Zurzeit arbeitet er als Senior Application and Content Engineer bei Würth Elektronik und entwickelt Anwendungshinweise und Tools, die den Entwicklern von Stromversorgungen bei der Lösung ihrer elektromagnetischen Probleme helfen.

WEBLINKS UND LITERATUR

[1] Baguley, C., Carsten, B. Madawala, „The Effect of DC Bias Conditions on Ferrite Core Losses", IEEE Transactions on Magnetics, Vol. 44, No. 2, February 2008:

[2] Barbisio, E., Fiorillo, F., Ragusa, C., „Predicting Loss on Magnetic Steels Under Arbitrary Induction Waveform and With Minor Hysteresis Loops", IEEE Transactions on Magnetics, Vol. 40, No. 4, July 2004:

[3] Online-Simulator REDEXPERT: https://redexpert.we-online.com/

Messungen für eine optimale Cloud-Implementierung

Bild 1. Das SNYPER-LTE Spectrum liefert umfangreiche Daten zur leichteren Auswahl der Antenne und des optimalen Standorts für Mobilfunkgeräte.

Von Stuart Cording für Mouser Electronics

Mit der Verbreitung des Internets der Dinge (IoT) und der Industrie 4.0 werden traditionelle Embedded-Systeme durch die Cloud-Technologie ergänzt. Real-Time-Computing, beispielsweise zur Steuerung eines Roboterarms oder eines Förderbands, wird mit Servern vernetzt, die Nutzungsdaten sammeln und auswerten. Mit praktisch unbegrenztem Speicherplatz und enormer Rechenleistung können präventive Wartungsmaßnahmen, Energieverbrauchsanalysen und andere umfangreiche Auswertungen vorgenommen werden. Das war früher nur mit großem Aufwand zu realisieren. Bei diesen Systemen kommt häufig eine Machine-Learning-Technologie zum Einsatz, um die Daten zu analysieren und Vorhersagen über Trends zu treffen. Vieles davon mag noch wie Science-Fiction erscheinen, aber es gibt bereits etliche Projekte, in denen die technischen Anforderungen untersucht werden. So hat beispielsweise der Automobilhersteller Audi gemeinsam mit Ericsson die Echtzeit-Steuerung von Roboterarmen unter Verwendung der URLLC-Fähigkeit (Ultra Reliable Low-Latency Communication) von 5G für die Fabrikautomatisierung demonstriert [1].

Standortanalysen für drahtlose Konnektivität

Die Verlegung von Kabeln für die Konnektivität ist kostenintensiv und schwierig und sorgt dafür, dass der Installationsort der Geräte nicht verändert werden kann. Dies steht im Widerspruch zum aktuellen Trend zur Flexibilität in der Fertigung, bei dem sich das Fabriklayout an neue Anforderungen anpasst, wofür wiederum drahtlose Konnektivität erforderlich ist.

Bei anderen Anwendungen, wie bei autonomen fahrerlosen Transportsystemen (FTS), ist die drahtlose Verbindung sogar die einzige Option. Vor der Einführung solcher Systeme sollte der Standort untersucht werden, um die Telekommunikationslandschaft genau zu erfassen. Die Netzbetreiber stellen zwar Karten der Netzabdeckung zur Verfügung, die Realität vor Ort kann jedoch erheblich abweichen, insbesondere innerhalb von Gebäuden. Mit Testgeräten wie dem Siretta SNYPER-LTE Spectrum (EU) [2] stehen den Nutzern leistungsfähige tragbare Spektrumanalysatoren zur Verfügung, die die Standortanalyse vereinfacht **(Bild 1)**. Das auf EU-Mobilfunknetze abgestimmte Gerät wird in einem praktischen Tragekoffer geliefert, in dem alles übersichtlich in Schaumstofffächern untergebracht ist. Neben dem Spektrumanalysator enthält das Kit ein multiregionales Netz- und Kfz-Ladegerät sowie USB- und HF-Kabel. Zwei kurze Antennen sind im Lieferumfang ebenfalls enthalten: eine für den universellen Einsatz in 4/3/2G und eine zweite für LTE bei 2.600 MHz. Mit einer dritten Richtantenne können gerichtete LiveSCAN-Messungen durchgeführt werden. Für Messungen benötigen die Nutzer eine aktivierte 4G-fähige SIM-Karte.

Der Signalanalysator SNYPER-LTE Spectrum kann Einzelpunktmessungen an einem Standort durchführen, um die Eigenschaften des Netzes zu erfassen, einschließlich Zellenbesitz und Signalstärke. Mit dem integrierten Akku kann das Gerät 48 Stunden lang Einzelmessungen durchführen (bei 20 Messungen pro Tag) oder bis zu 15 Stunden im Dauerbetrieb für eine liveSCAN-Messung. Außerdem können die Ergebnisse von 50 Messungen auf dem Gerät gespeichert werden.

Die liveSCAN-Messungen basieren auf Einzelmessungen und ermöglichen eine detailliertere Analyse. Mit der Standardantenne kann der Spektrumanalysator innerhalb des Gebäudes bewegt werden, um Hotspots mit optimaler Signalstärke zu ermitteln. Durch Umschalten auf die Richtantenne kann der Nutzer die Richtung der höchsten Signalstärke ermitteln. Anhand dieser Daten lassen sich dann die optimale Antennenwahl und der Standort für drahtlose Mobilfunkgeräte festlegen. Die Ergebnisse werden auf dem Bildschirm angezeigt oder können in kommagetrennten (CSV) und HTML-Formaten für Berichte und zur Dokumentation heruntergeladen werden.

Entwicklung von Testsystemen im L-Band

Die Entwicklung komplexer HF-Testsysteme wird durch Software-Defined-Radio-Ansätze (SDR) immer einfacher, denn sie besitzen hochwertige HF-Frontends und konfigurierbare

Bild 2. Die RFX-8440 ist eine hochgradig konfigurierbare L-Band-Datenerfassungskarte, die sich für HF-Prüf- und -Messanwendungen oder für den Masseneinsatz eignet.

Hardware, wie z. B. FPGAs. Die Datenerfassungskarte RFX-8440 [3] von BittWare ist ein Funkfrequenzsystem auf einem Chip (RFSoC), das für Anwendungen im L-Band (1 bis 2 GHz) konzipiert ist und in so unterschiedlichen Bereichen wie GPS, Telekommunikation, Luftfahrt und Astronomie eingesetzt wird. Die Datenerfassungskarte **(Bild 2)** kombiniert ein Xilinx Zynq UltraScale+ ZU43 RFSoC mit einem analogen Frontend, das eine rauscharme Signalkonditionierung mit variabler Verstärkung (-40 bis 0 dBm) vor den Digitalisierern bietet. Es sind auch alternative Frontend-Konfigurationen erhältlich, mit denen sich der Eingangsbereich auf 4 GHz erweitern lässt. Insgesamt gibt es vier 14-Bit-ADCs mit 5 GSPS und vier 14-Bit-DACs mit 10 GSPS, die durch Takt-, Referenz- und Triggersignale ergänzt werden. Neben der programmierbaren Logik verfügt das RFSoC über einen Quad-Core Arm® Cortex®-A53 und einen Dual-Core Arm Cortex-R5. Die Karte besitzt einen PCIe-Anschluss, kann aber dank der Ethernet-, USB- und Display-Link-Schnittstellen auch eigenständig verwendet werden. Zusätzliche Erweiterungen sind über den achtkanaligen OCulink-Port möglich, der PCIe Gen4 x8, eine Schnittstelle für NVMe-Speicher oder ein duales 100-Gbit-Netzwerk bietet.

Bei Testsystemen sind in der Regel programmierbare Schalter erforderlich, damit die HF-Signale an die richtigen Ports geleitet werden können. Der programmierbare HF-Multiplexer T3SP-D4MX-BUNDLE [4] von Teledyne ergänzt die BittWare-Datenerfassungskarte mit einer Bandbreite von DC bis 10 GHz und Phasenanpassung. Das Gerät verfügt über ein Paar phasenangepasste (±2 ps) 20-cm-Kabel für die Eingänge und vier Paar phasenangepasste, farbcodierte 60-cm-Kabel für die Ausgänge. Die beiden Kanäle haben jeweils einen 1:4-Schalter sowie eine nicht angeschlossene Option, die beim Einschalten als Standardauswahl fungiert. Der Multiplexer kann über eine USB-2.0-Schnittstelle mit einer DLL in C/C++, C# und Python angesteuert oder in Entwicklungsumgebungen wie LabVIEW und MatLab integriert werden. Für die manuelle Steuerung steht ebenfalls eine einfache Software zur Verfügung **(Bild 3)**. Durch den Einsatz von HF-MEMS-Schaltern bietet jeder Kanal 1 Milliarde Schaltzyklen.

Bereit für die Cloud

Natürlich sind immer noch viele nicht vernetzte Maschinen im Einsatz, die nicht in die Cloud integriert sind. Da eine Datenanalyse jedoch eine bessere Entscheidungsfindung ermöglicht, möchten viele Hersteller Lösungen für die Datenerfassung nachträglich integrieren. Der DFRobot DAM-3918 [5] verfügt über acht analoge Eingangskanäle und kann analoge Sensordaten über seine optisch isolierte RS485-Schnittstelle unter Verwendung des Modbus-RTU-Protokolls übertragen. Er hat eine Genauigkeit von ±1 % und eignet sich für den Einsatz mit 2-, 3- und 4-Draht-Transmittern und -Wandlern (z. B. 0 bis 5 V, 4 bis 20 mA). Die 12-Bit-Eingangskanäle sind einzeln konfigurierbar, wobei 50 SPS für einen einzelnen Kanal und insgesamt 400 SPS zur Verfügung stehen. Die Stromversorgung muss über 18 bis 30 VDC erfolgen.

Oszilloskope und Remote-Datenerfassung

Leistungsstarke Laptops sind inzwischen weit verbreitet und werden von Entwicklungsingenieuren häufig verwendet. Daher haben die Hersteller von Test- und Messgeräten zahlreiche kompakte Desktop-Oszilloskope auf den Markt gebracht, bei denen die Messwerte auf dem Computerbildschirm angezeigt werden. Entwickler erhalten dadurch leistungsstarke analoge Frontends und können mit geringem Kostenaufwand eine Reihe von hochmodernen Funktionen zur Analyse und Protokolldekodierung nutzen. Das Analog Discovery Pro 3000 [6] von Digilent ist ein solches Messwerkzeug. Es ist als 4-kanalige (ADP3450) und 2-kanalige (ADP3250) Variante erhältlich.

Neben den 14-Bit-Analogeingängen mit 0,5 GSPS verfügt das Gerät über 16 digitale I/Os und ein programmierbares digitales Netzgerät. In Verbindung mit der Software *WaveForms* [7] von Digilent kann das Discovery Pro 3000 unter anderem auch Wellenformen erzeugen, Spektralanalysen durchführen und als Netzwerkanalysator eingesetzt werden. Das Gerät lässt sich auch über JavaScript oder über das WaveForms Software Development Kit (SDK) in C/C++, C#, Visual Basic oder Python steuern.

Der Linux-Modus, mit dem das Gerät eigenständig als leistungsstarke Hardware-Testlösung arbeitet, ist ein herausragendes Merkmal. In Verbindung mit dem WaveForms SDK lassen

Bild 3. Der T3SP-D4MX von Teledyne ist ein zweikanaliger 1:4-Multiplexer für DC bis 10 GHz, der über eine DLL oder die grafische Benutzeroberfläche WinD4MX.exe gesteuert werden kann.

Bild 4. Die Smart Bench Essentials sind eine Produktfamilie von Prüf- und Messgeräten von Keysight, die für Remote-Betrieb und -Interaktion entwickelt wurde und den konventionellen Betrieb mit Cloud-Konnektivität verbindet.

sich in diesem Terminal-basierten Modus eigene Tests oder Anwendungen erstellen und programmieren. Die Ergebnisse können über die Ethernet-Schnittstelle gestreamt oder in den lokalen Erfassungspuffern gespeichert werden, wo Platz für Millionen von Datenpunkten vorhanden ist.

Der „smartere“ Prüfstand

Prüfgeräte, die mit einem Laptop verbunden sind, bieten zwar viele Möglichkeiten in einem kleinen Paket, aber sie überladen den Prüfstand. Außerdem wird es dadurch schwieriger, den Computer für E-Mails, das Schreiben von Berichten und die unvermeidlichen Videoanrufe zu nutzen. Die Smart Bench Essentials [8] von Keysight Technologies sind eine Produktfamilie konventioneller, stapelbarer Prüf- und Messgeräte mit industrietauglicher Konnektivität, gut dimensionierten 7-Zoll-Displays (18 cm) und Funktionen zur gemeinsamen Nutzung von Messungen.

Die Produktfamilie umfasst zwei- und vierkanalige Oszilloskope, ein 5½-stelliges Multimeter, ein- und zweikanalige Arbiträrsignal-Generatoren und ein programmierbares Netzgerät mit drei Ausgängen **(Bild 4)**. Durch die Verwendung des Instrumenten-Stacking-Kits kann der Platzbedarf am Prüfstand minimiert werden. Mit der BenchVue-Software lassen sich Daten protokollieren und analysieren. Die interessanteste Lösung ist jedoch die PathWave Lab Management Software. Administratoren und Ausbilder können über kabelgebundene Ethernet- oder WLAN-Verbindungen den Überblick über die Geräte behalten, Firmware-Updates einspielen oder die Messtools per Fernzugriff über die Cloud verfügbar machen.

Tests und Messungen für die Cloud-Implementierung

In den Märkten, in denen Entwickler tätig sind, nimmt die Cloud-Konnektivität weiter zu. Dies setzt voraus, dass sie die Wireless-Landschaft vor Ort genau kennen, fortschrittliche und konfigurierbare HF-Testverfahren einsetzen oder Lösungen verwenden, mit denen ältere, aber noch brauchbare Fertigungsanlagen in die Cloud integriert werden können. Aufgrund ihrer Programmierbarkeit können Oszilloskope ohne Display als sehr vielseitige, ferngesteuerte Test- und Datenprotokollierungstools genutzt werden. Diese Geräte sind ebenfalls in der Lage, ihre Messungen zur weiteren Analyse an Cloud-Plattformen zu streamen. Selbst herkömmliche Prüfgeräte mit nur einem Gerät pro Box bieten eine fortschrittliche und robuste Konnektivität für effizientes Labormanagement, die gemeinsame Nutzung von Messungen und eine cloudbasierte Steuerung. Für nahezu alle Bereiche der Prüf- und Messtechnik gibt es eine Cloud-Lösung.

230458-02

Mouser Electronics
Elsenheimerstr. 11
80687 München
+49 (0)89 520 462 110
muenchen@mouser.com

Über den Autor

Stuart Cording ist ein freiberuflicher Journalist, der für Mouser Electronics schreibt. Er ist auf Videoinhalte spezialisiert und fokussiert auf technische Vertiefungen und Einblicke. Daher interessiert er sich besonders für die Technologie selbst, ihre Einbindung in Endanwendungen und Vorhersagen über zukünftige Entwicklungen. Mouser Electronics ist ein autorisierter Distributor für Halbleiter und elektronische Bauteile, der sich auf die Einführung neuer Produkte seiner führenden Herstellerpartner konzentriert.

WEBLINKS

[1] 5G für die Fabrikautomatisierung: https://www.ericsson.com/en/news/2020/2/5g-for-factory-automation
[2] Siretta SNYPER-LTE Spectrum: https://bit.ly/46DLXqf
[3] BittWare RFX-8440: https://eu.mouser.com/new/test-measurement/bittware-rfx-8440
[4] T3SP-D4MX-BUNDLE: https://bit.ly/3D0q476
[5] DFRobot DAM-3918: https://bit.ly/3D3gPTC
[6] Digilent Analog Discovery Pro 3000: https://bit.ly/3PR8XMR
[7] Digilent WaveForms : https://digilent.com/shop/software/digilent-waveforms/
[8] Keysight Smart Bench Essentials: https://bit.ly/3JRekaP

Matter-Implementierung: **Was braucht es, um** Matter-Geräte **einzusetzen?**

Von Sujata Neidig, NXP Semiconductors

Matter gehört zur Connectivity Standards Alliance (CSA) und ist ein universelles und offenes IoT-Protokoll, eine gemeinsame Sprache, die es Smart-Home-Geräten ermöglicht, marken- und plattformübergreifend miteinander zu kommunizieren, z. B. mit Amazon, Apple, Google und Samsung SmartThings. Matter beseitigt geschlossene Plattformen und sorgt für Interoperabilität, sodass die Verbraucher Flexibilität und Auswahl bei den Geräten haben, die sie kaufen, und darauf vertrauen können, dass diese nahtlos zusammenarbeiten. Einen tieferen Einblick in Matter erhalten Sie in der Publikation *Matter - Making Smart Homes Smarter* [1].

Matter legt auch die Messlatte für die Sicherheit höher. Jedes Matter-Gerät muss seine Identität nachweisen und authentifizieren, dass es ein Matter-zertifiziertes Gerät ist, bevor es dem Matter-Netzwerk beitreten darf. Sobald sich das Gerät im Netzwerk befindet, wird die gesamte Kommunikation verschlüsselt. Einen tieferen Einblick in die Matter-Sicherheit erhalten Sie in der Publikation *Matter - Making Smart Homes More Secure* [2].

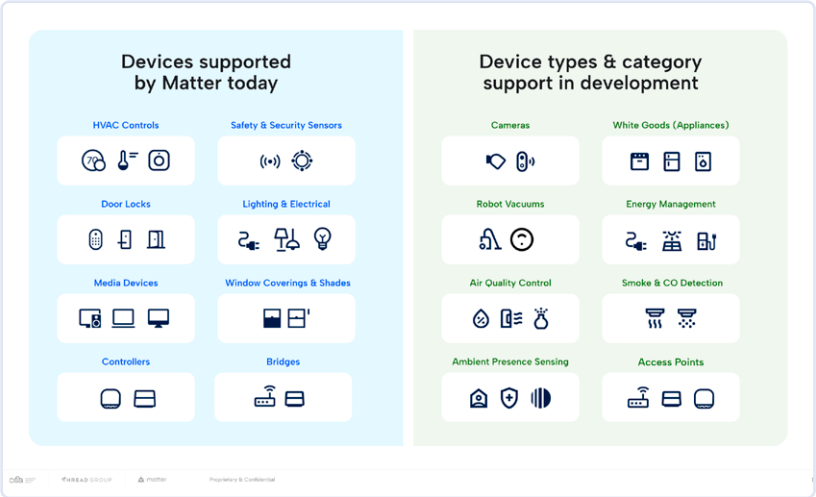


Bild 1. Matter erweitert laufend die Gerätetypen (Quelle: Connectivity Standards Alliance - CSA).

Matter-Gerätetypen

Matter ist die Art und Weise, wie Geräte miteinander kommunizieren, es definiert die Eigenschaften und Fähigkeiten von Geräten auf der Anwendungsebene. Bei der Einführung unterstützte Matter sieben Gerätekategorien und wird, wie in **Bild 1** zu sehen, auf viele weitere erweitert. CSA-Mitglieder treiben die Erweiterung von Matter auf weitere Gerätetypen voran.

Überlegungen zum Design

Matter stellt die Entwickler vor mehrere Entscheidungen, bei der Entwicklung ihres Systems, der Auswahl der Komponenten und der Planung der Implementierung von Sicherheitsmaßnahmen.

Zunächst muss ein Entwickler entscheiden, welche Funktionen das Gerät benötigt, basierend auf dem Matter-Gerätetyp, den zugehörigen Anwendungsclustern und den über Matter hinaus benötigten Funktionen. Ein Beispiel sind die Benutzerschnittstelle und die Zielvorgaben für den Energiebedarf (z. B. Netz- oder Batteriebetrieb, Batteriegröße und -lebensdauer, usw.).

Als Nächstes sollte der Entwickler die Anforderungen an die Konnektivität festlegen. Da es sich bei Matter um eine IP-basierte Technologie handelt, werden derzeit Wi-Fi, Thread und Ethernet unterstützt; die Geräte können eine oder mehrere dieser Optionen nutzen. Wi-Fi eignet sich ideal für Anwendungsfälle mit hoher Bandbreite, wie z. B. Audio- oder Videostreaming, während Thread ideal für Kontrollanwendungen mit geringer Bandbreite geeignet ist, bei denen Zuverlässigkeit und geringer Energieverbrauch eine Priorität sind. Neben der Konnektivität muss der Entwickler auch festlegen, welche Funktion(en) das Gerät unterstützen soll, z. B. Thread Border Router, Matter Commissioner, Matter Controller, Matter Bridge, usw. Der letzte Schritt besteht darin, die Sicherheitsanforderungen der Anwendung zu bewerten, die über die von Matter definierten Anforderungen hinausgehen. Zum Beispiel könnte ein intelligentes Türschloss einen Schutz gegen physische Angriffe implementieren.

Sobald diese Anforderungen festgelegt sind, ist es an

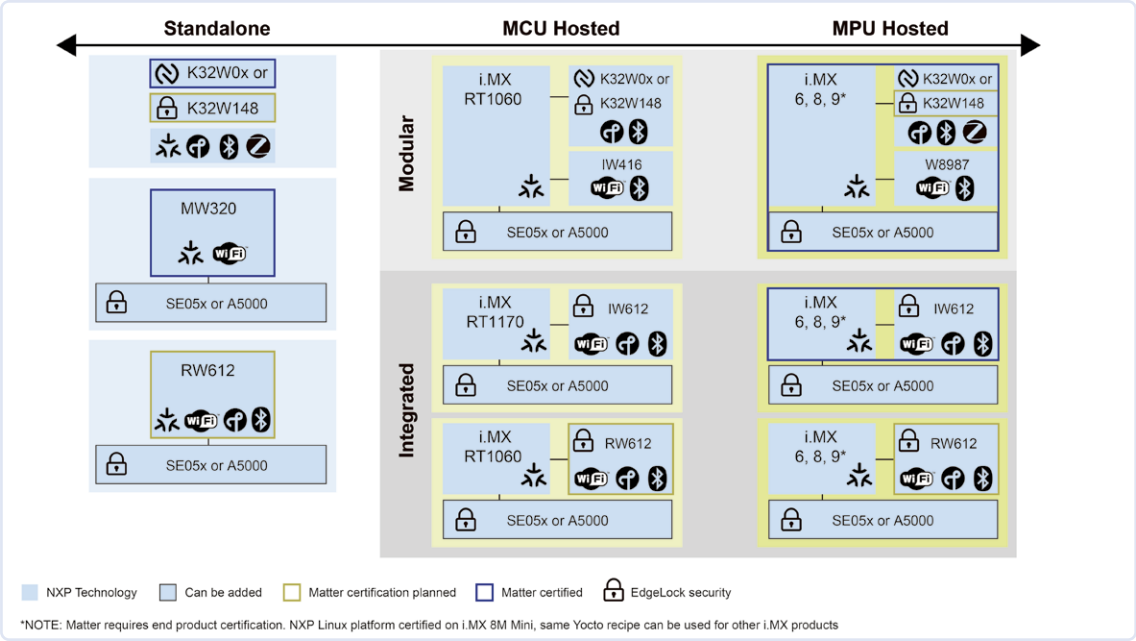


Bild 2. Systemarchitektur-Optionen für Matter-Geräte: Standalone und mit Host (Quelle: NXP Semiconductors).

der Zeit, die Systemarchitektur zu bestimmen (**Bild 2**):

- **Standalone**: Eine einzige MCU wird zur Implementierung der Anwendung und der drahtlosen Konnektivität verwendet. Dies ist eine ideale Architektur für einfachere Gerätetypen, die kosten-, größen- und stromsparend sind.
- **Mit Host:** Eine Host-MPU oder -MCU wird verwendet, um die Anwendung mit einer separaten drahtlosen MCU oder einem Transceiver zu implementieren, welcher das Drahtlos-Interface bereitstellt. Diese Architektur wird für komplexere Gerätetypen verwendet, die umfangreichere Benutzeroberflächen haben und/oder mehrere Netzwerke und Funktionen verwalten.

NXP bietet ein Portfolio von Matter-Entwicklungsplattformen an, die eine Vielzahl von Gerätetypen und Anwendungsfällen abdecken. Diese Plattformen umfassen die für das Gesamtsystem benötigten Schlüsselkomponenten: Verarbeitung, Konnektivität und Sicherheit. Weitere Informationen finden Sie unter [3].

Implementierung Ihres Matter-Geräts

Die Zertifizierung ist der nächste Schritt nach dem Entwurf des Geräts, der für die Herstellung der Interoperabilität unerlässlich ist. Darüber hinaus bietet die Zertifizierung die Lizenz zur Nutzung der Technologie und die Rechte zur Verwendung der Technologie-Logos. Die CSA bietet ein Zertifizierungsprogramm für Matter an, das Testskripte, Tools und Dienstleistungen (über autorisierte Testlabors) zur Unterstützung des Prozesses umfasst, einschließlich der Anforderung der abhängigen Zertifizierung, die die Zertifizierung der zugrunde liegenden Technologien - Thread, Wi-Fi und Bluetooth - validiert. Hinweis: Diese Technologien sind Eigentum anderer Normungsgremien, die jeweils Mitgliedschaftsoptionen anbieten.

Da Matter auf der Anwendungsebene angesiedelt ist, muss jedes Matter-Gerät eine Matter-Zertifizierung durchlaufen. Vorabtests können vom Entwickler mit den von der CSA bereitgestellten Tools durchgeführt werden. Sobald er fertig ist, bringt der Entwickler das Gerät zu einem ATL und beantragt die Zertifizierung bei der CSA. Die CSA bestätigt die Matter-Zertifizierung und die abhängigen Zertifizierungen und stellt die Zertifizierungs-ID aus. Das Gerät wird dann in die CSA-Liste der zertifizierten Produkte und in den Distributed Compliance Ledger aufgenommen.

Da Thread eine Netzwerkschicht ist, die in der Regel von der Anwendung nicht verändert wird, unterstützt das Zertifizierungsprogramm der Thread Group die Zertifizierung durch Ähnlichkeit. Wenn das Gerät eine Thread-zertifizierte Komponente (vom gewählten Siliziumlieferanten) verwendet und keine Änderungen an der Firmware vornimmt, kann die Zertifizierung durch ein schriftliches Verfahren ohne Verwendung einer ATL erteilt werden.

Der letzte Schritt ist die Einrichtung eines Produktionsablaufs für den Erhalt und die Einfügung eines Device Attestation Certificate (DAC) in jedes Gerät. Der EdgeLock 2GO-Service von NXP ist eine von der CSA zugelassene Matter Product Attestation Authority, die es NXP ermöglicht, Matter-DACs über die Cloud an die Fertigungsstandorte der Kunden zu liefern.

Das Produkt kann dann auf den Markt gebracht werden und dazu beitragen, den Verbrauchern ein autonomes Zuhause zu bieten! Außerdem lassen sich Verbesserungen und neue Patches für Sicherheitslücken nahtlos durch Over-the-Air-Updates an die Nutzer weitergeben.

230488-02

WEBLINKS

[1] Matter – Making Smart Homes Smarter: https://www.nxp.com/webapp/sps/download/preDownload.jsp
[2] https://www.nxp.com/webapp/Download?colCode=MATTERSMRTHOMEWP
[3] Matter – NXP Semiconductors: http://www.nxp.com/matter

Neue 2,4 GHz-Funkeinheiten **von Circuit Design**

Prädestiniert für Fernsteuerung und Überwachung

Von Christian Reimesch
(Reimesch Kommunikationssysteme GmbH)
für Circuit Design

Der Fernsteuersender KST2.4S besitzt sechs Eingänge, deren logische Zustände sich auf den Empfänger KSR2.4 per Funk auf dessen sechs Ausgänge übertragen lassen. Damit eröffnet sich ein breites Anwendungsfeld zur Steuerung von Baumaschinen, Forstmaschinen, landwirtschaftlichen Geräten, Garagentoren und mehr. Circuit Design hat großen Wert auf eine sichere Übertragung der Daten sowie eine hohe Verfügbarkeit der Funkstrecke gelegt.

Bei den Fernsteuereinheiten KST2.4S [1] und KSR2.4 [1] (**Bild 1**) handelt es sich um ein neues Produkt des japanischen Herstellers Circuit Design Inc. Der Fernsteuersender KST2.4S besitzt sechs Eingänge, deren logische Zustände sich auf den Empfänger KSR2.4 per Funk auf dessen sechs Ausgänge übertragen lassen.

Bild 1. Sender KST2.4S und Empfänger KSR2.4

Bild 2. Fernsteuermodul NK2.4Y.

Beide Einheiten basieren auf dem Fernsteuermodul NK2.4Y [2] aus gleichem Hause (**Bild 2**). Die komplette Funktechnik wurde beim NK2.4Y auf einem sehr kompakten Footprint realisiert. Das Modul erfüllt sowohl die europäische Normen der EU (CE) als auch der USA (FCC) und Kanada (IC) und eignet sich somit hervorragend für eigene Entwicklungen, die gänzlich mit den eigenständigen Einheiten KST2.4S und dem KSR2.4 kompatibel sind.
Damit eröffnet sich ein breites Feld von Anwendungen zur Steuerung von Bau- und Forstmaschinen, landwirtschaftlichen Geräten und Garagentoren. Auch Sicherheitssysteme wie drahtlose Notaus-Systeme sind als Anwendung denkbar. Hierbei müssen Sender und Empfänger je nach Sicherheitsanforderung mit einer erweiterten Steuerung sowie Zulassungen aus dem Sicherheitsbereich ergänzt werden.

Bidirektionale Kommunikation

KST2.4S und KSR2.4 müssen vor ihrer ersten Inbetriebnahme „gepaart" werden und sind ausschließlich für eine 1:1-Kommunikation zwischen Sender und Empfänger vorgesehen. Die Kommunikation der beiden Einheiten ist allerdings bidirektional und geht nicht nur in eine Richtung, wie die Begriffe „Sender" und „Empfänger" es vermuten lassen.
Zur Übertragung der sechs digitalen Eingänge, die übrigens potentialfrei über Optokoppler für einen Spannungsbereich von 6...35 V ausgelegt wurden, wird das 2,4 GHz-ISM-Band genutzt. Dieses von unterschiedlichen Funkanwendungen (Wi-Fi, Bluetooth, ZigBee und so weiter) genutzte Band stellt besondere Ansprüche an die Qualität der Funktechnik. Circuit Design hat daher großen Wert auf eine sichere Übertragung der Daten sowie eine hohe Verfügbarkeit der Funkstrecke gelegt. Die Daten werden im Frequenzsprungverfahren (Frequency Hopping) auf insgesamt 20 Kanälen zwischen 2403 MHz und 2479 MHz mit einer Funkdatenrate von 250 kbps in der Modulationsart GFSK übertragen. Ein Datenpaket des Senders KST2.4S ist circa 700 µs lang und wird innerhalb von 100 µs durch ein gleich langes Datenpaket des „Empfängers" KSR2.4 bestätigt. Die Aussendung des nächsten Datenpakets erfolgt dann im zeitlichen Abstand von 10 ms auf einem anderen Kanal. Hierbei sichert Circuit Design sowohl die Nutzdaten (Payload) als auch das gesamte Datenpaket über einen

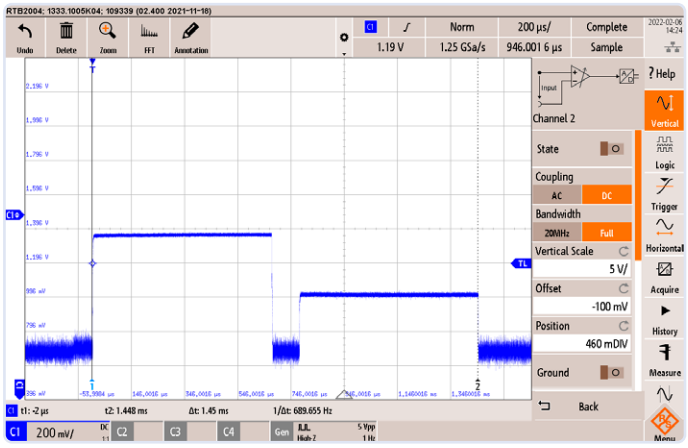


Bild 3. KST2.4S-Datenpaket und KSR2.4-Antwort innerhalb 1,45 ms.

eigenen 16-Bit CRC mit einer Hamming-Distanz von HD = 6 ab. Die Ausgabe auf der Empfängerseite erfolgt ebenfalls potentialfrei über PhotoMOS-Relais. Der Laststrom darf nach Herstellerangaben 200 mA pro Ausgang bei einer maximalen Spannung von 35 V betragen. Damit sind die meisten Relais direkt ansteuerbar, es ist lediglich die übliche Freilaufdiode parallel zur Relaisspule vorzusehen, um eine Schädigung des PhotoMOS-Relais zu verhindern.

Datenübertragung zwischen KST2.4S und KSR2.4

Wie bereits erwähnt, erfolgt die Datenübertragung im Frequenzsprungverfahren über ein abgesichertes, bidirektionales Protokoll. Im Diagramm (**Bild 3**) ist je ein Datenpaket des Senders KST2.4S und nach einer kurzen Pause das Antwortpaket des Empfängers KSR2.4 zu sehen. Insgesamt dauert die gesamte Kommunikation nur 1,45 ms. Die kurze Kanalnutzung in Verbindung mit dem Frequenzsprungverfahren vermindert das Risiko einer gegenseitigen beziehungsweise längerfristigen Beeinflussung mit anderen Funkanwendungen.

Ausgangsleistung

Die Ausgangsleistung des KST/KSR2.4Y beträgt 0 dBm (1 mW) auf dem ersten Kanal (2,403 GHz) und fällt auf dem höchsten Kanal von 2,479 GHz leicht um 1,15 dB ab. Im Diagramm **(Bild 4)** sind alle 20 Hopping-Kanäle abgebildet.

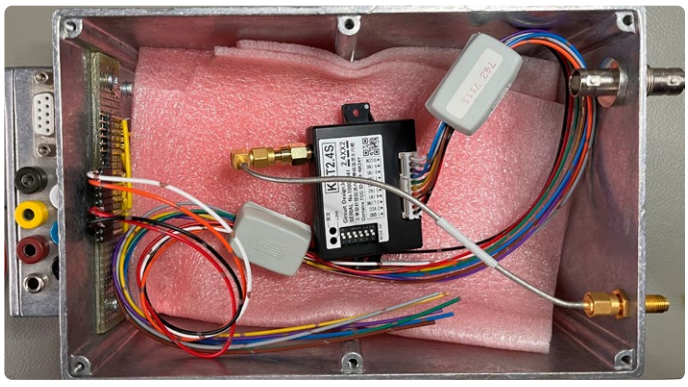


Bild 5. KST2.4S im abgeschirmten Gehäuse.

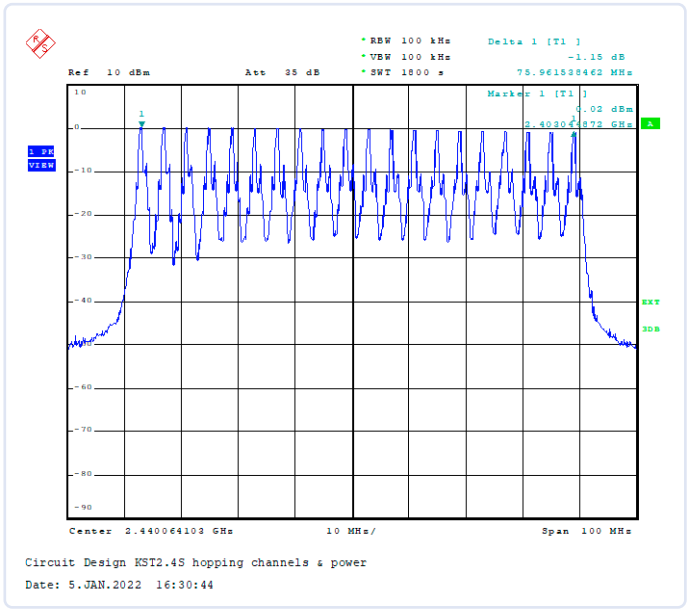


Bild 4. Darstellung der 20 Hopping-Kanäle.

Empfängerempfindlichkeit

Für diese Messung wurde ein KST2.4S in ein abgeschirmtes Gehäuse (**Bild 5**) eingebaut, das über Filter für die Eingänge und für die Stromversorgung verfügt. Nur so lässt sich eine Einkopplung des Sendesignals über andere Wege als die Antennenbuchse verhindern. Über eine Eichleitung (DPSP von Rohde&Schwarz) wurde eine Dämpfung von bis zu 95 dB zwischen Sender und Empfänger KSR2.4Y eingefügt. Bei diesen Empfangsbedingungen ist eine Datenübertragung zwischen KST2.4S und KSR2.4 gerade noch möglich. Über diese Dämpfung und die ermittelte Ausgangsleistung von 0 dBm (1 mW) ergibt sich eine Empfängerempfindlichkeit von etwa -95 dBm. Setzt man eine der zugelassenen Antennen mit ungefähr 2 dBi Gewinn von Circuit Design ein, ergibt sich eine theoretische Reichweite (siehe **Bild 6**) von 900 m

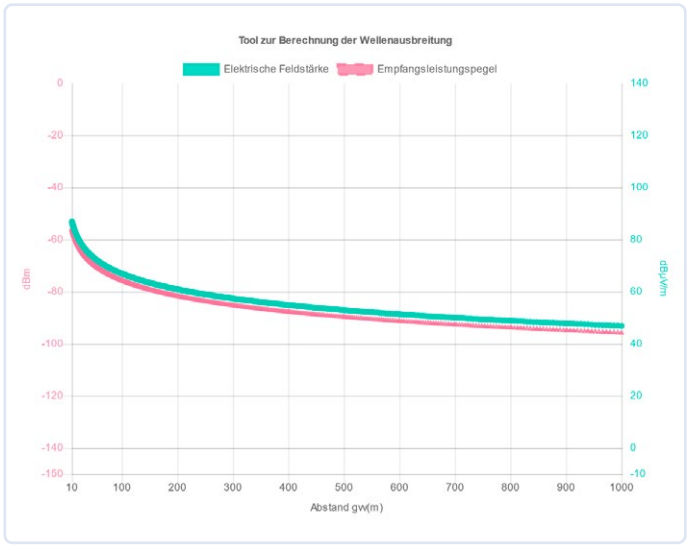


Bild 6. Berechnete Streckendämpfung im 2,4 GHz Band.

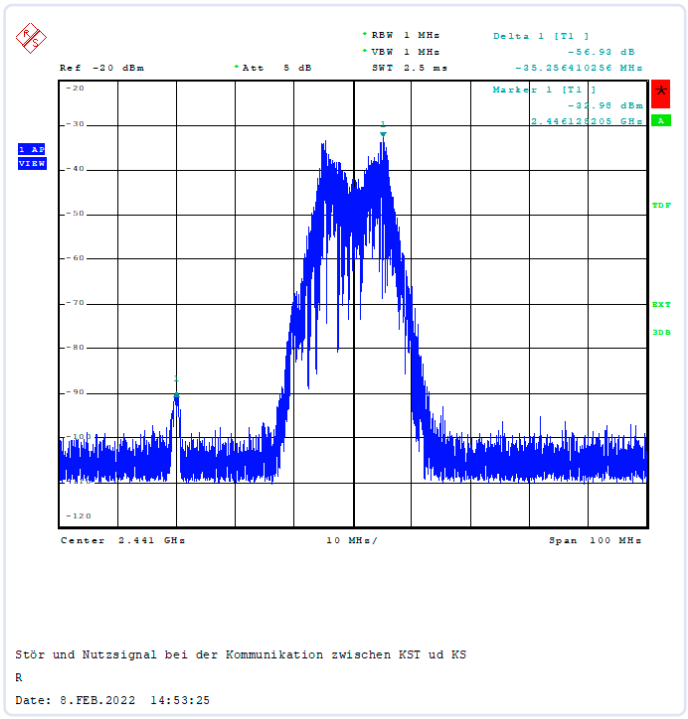


Bild 7. Nutzsignal (links) mit 60 dB stärkeren Störsignal.

bei optischer Sichtverbindung. Circuit Design gibt eine recht konservative Reichweite von 100 m an, die für den praktischen Betrieb eine Signalreserve von etwa 20 dB beinhaltet und der Zuverlässigkeit der Funkverbindung zugutekommt.
Das Diagramm zeigt die zu erwartende Empfängereingangsleistung in dBm in Abhängigkeit der Entfernung zur Gegenstation. Die Berechnung basiert auf der Annahme, dass sich beide Antennen 2,5 m über dem Boden befinden und einen Antennengewinn von 2,14 dBi aufweisen.

Störfestigkeit

Um die Störfestigkeit zu überprüfen, wurde dem Empfänger KSR2.4 ein Nutzsignal mit einem Pegel von -92 dBm (3 dB oberhalb der Empfängerempfindlichkeit) sowie ein Störsignal aus einem Signalgenerator SMIQ03B von Rohde&Schwarz über einen Combiner zugeführt. Das Störsignal bei 2,441 GHz hatte folgende Modulationsparameter:
Modulation 2FSK, Gaußfilter mit BT = 0,5, Frequenzhub = 5 MHz, Symbolrate 5 Mbit/s
Insgesamt wurde der Pegel des Störsignals so weit erhöht, dass noch eine sichere Kommunikation zwischen KST2.4S und KSR2.4 möglich war. Der Leistungspegel des Störsignals betrug dabei -31 dBm. Der relative Abstand $P_{nutz}/P_{stör}$ lag somit bei 60 dB.
Die Grafik in **Bild 7** beschreibt das Störszenario mit einem schnellen Digitalsignal in Bandmitte (2,441 GHz) mit einer Spitzenleistung von -31 dBm, wobei die Nutzspitzenleistung (kleiner Träger links bei -30 MHz) einen Pegel von -91 dBm hat.

Übertragungs- und Zeitverhalten

Momentary Mode

Grundsätzlich kann nur eine Übertragung vom KST2.4S zum KSR2.4 stattfinden, wenn die gegenseitigen Nutzsignalpegel auf der Funkseite

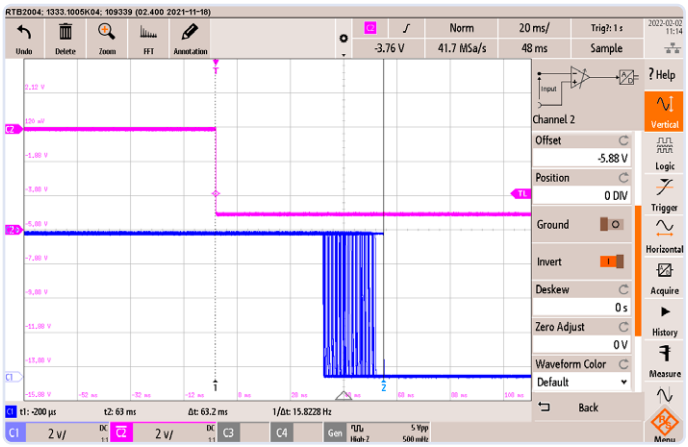


Bild 8. Momentary Mode: Zeitverhalten und Jitter des Ausgangssignals.

dies zulassen, das heißt, sobald die Empfänger mindestens einen Nutzsignalpegel von -95 dBm auf einem ungestörten Kanal empfangen. Hierbei ist zu beachten, dass keine Aussendung gestartet wird, wenn während des Einschaltens (Power ON) des KST2.4S ein Eingangssignal auf ON gesetzt ist.
Im Momentary Mode (Mode Schalter SW2 = OFF und SW3 = OFF) wird das Signal am Eingang des Senders KST2.4S eins-zu-eins auf den Ausgang des KSR2.4 übertragen. Dennoch gibt es ein gewisses Zeitverhalten. Im Diagramm in **Bild 8** ist das Ausgangssignal mit aktivierter Persistenz dargestellt, um den Jitter der Verzögerung sichtbar zu machen. In diesem Fall beträgt die Verzögerung maximal 60 ms; allerdings ist diese auch abhängig von der Qualität der Funkstrecke beziehungsweise von möglichen Funkstörungen. Kanal 2 (rosa) zeigt das Eingangssignal, Kanal 1 (blau) das Ausgangssignal.

One-Shot Mode

Im One-Shot Mode (SW2=ON, SW3=ON) wird abhängig von SW4 ein entweder 500 ms langer Puls (SW4 = OFF) beziehungsweise ein 200 ms langer Puls (SW4 = ON) übertragen.
Im **Bild 9** zeigt Kanal 2 (rosa) das Ausgangssignal des KSR2.4 im

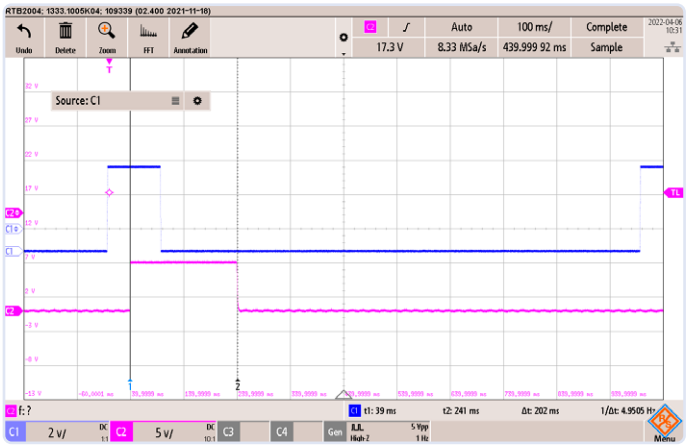


Bild 9. One Shot Puls: 200 ms.

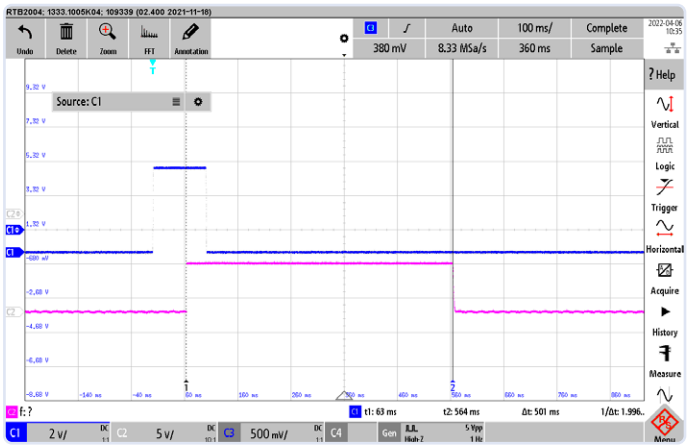


Bild 10. One Shot Puls: 500 ms.

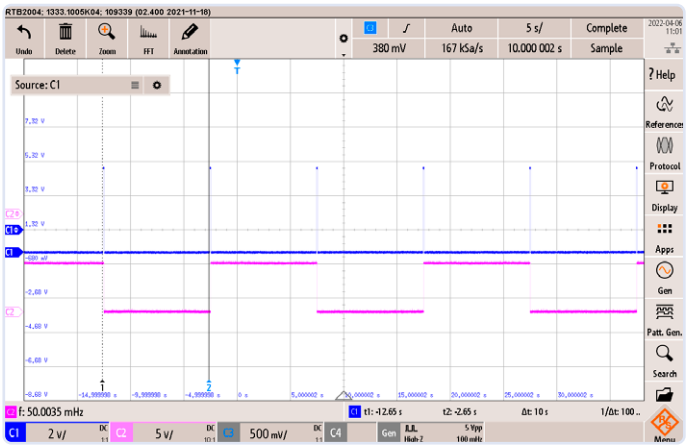


Bild 11. Toggle-Modus mit 100 ms-Triggerpuls am Eingang.

One-Shot Mode. Die Pulslänge entspricht der an SW4 (ON) eingestellten Dauer von 200 ms.
Nach Umschalten von SW4 auf OFF wird auch die Pulslänge von 500 ms sehr genau reproduziert (**Bild 10**). Für beide Messungen (200 ms / 500 ms) wurde auf der Senderseite beim KST2.4S ein 100 ms langer Puls verwendet.

Toggle Mode

Im Toggle Mode des KSR2.4 (SW2 = ON, SW3 = OFF) wird der Ausgang selbst bei einem kurzen Impuls am Eingang des KST2.4S umgeschaltet. Die Messung in **Bild 11** zeigt das Verhalten.
Alle 10 s erfolgt eine Umschaltung des Ausgangs des KSR2.4 (C2, rosa). Das Eingangssignal (blau) besteht aus 100 ms langen Pulsen, wie sie beispielsweise durch die kurze Betätigung eines Tasters entstehen. Der Toggle Mode scheint ideal zum Beispiel für die Initiierung eines Schaltvorgangs zu sein, also etwa Pumpe an, Pumpe aus, Winde an, Winde aus,...

Fazit

Das Fernsteuersystem KST2.4Y/KSR2.4 ermöglicht eine einfache und störfeste Steuerung für eine große Anzahl unterschiedlicher Applikationen. Durch die Überwachung der Funkverbindung über die LINK-Ausgänge beider Module sind auch Anwendungen im Bereich drahtloser Notaus-Systeme in Verbindung mit einer übergeordneten Steuerung denkbar. Die unterschiedlichen Ausgabemodi bieten eine hohe Flexibilität bei der Ansteuerung unterschiedlicher Anwendungen. Bei einfachen Applikationen, die keinen hohen Sicherheitsanforderungen unterliegen, kann auf externe Steuerungen verzichtet werden. Die Nutzung eines Frequenzsprungverfahrens garantiert eine hohe Verfügbarkeit der Funkstrecke, zudem bekommt jedes KST/KSR-Paar bei der Paarung eine eigene Hoppingsequenz zugewiesen.

RG — 230354-02

CIRCUIT DESIGN GmbH
Schleißheimer Str. 263
80809 München
Deutschland
www.circuitdesign.de

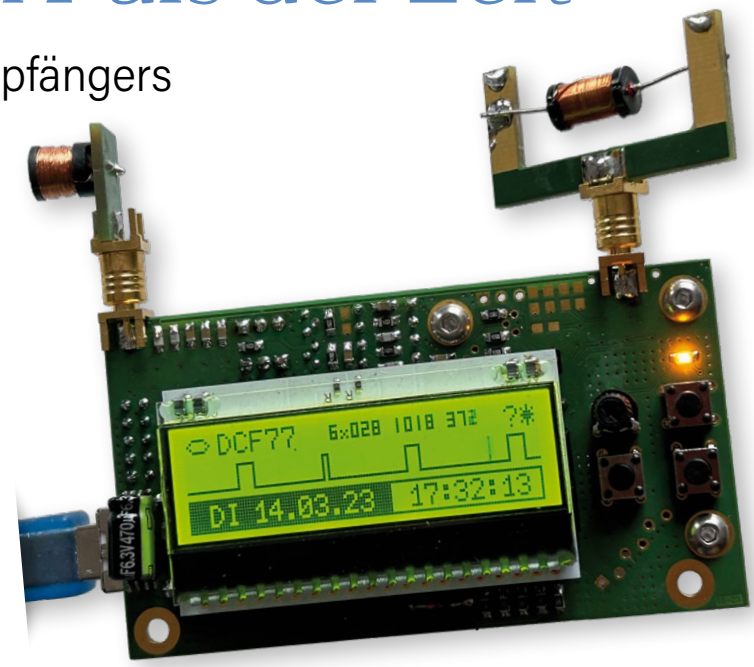
WEBLINKS

[1] Technische Unterlagen KST/KSR: https://circuitdesign.de/produkte/kst24s-ksr24/?=page03
[2] Technische Unterlagen NK2.4Y: https://circuitdesign.de/produkte/nk-2-4y/?=page03

PIC O'Clock

am Puls der Zeit

Design eines SDR-Zeitzeichen-Empfängers



Von Marco Rohleder (Deutschland)

Bei der Entwicklung eines Zeitzeichen-Empfängers kommt man an dem Prinzip eines Software-definierten Radios (SDR) nicht mehr vorbei. Die Software zur Dekodierung dieser Signale war schon öfter Thema in Elektor; in diesem Artikel soll es um die erforderliche Hardware gehen. Dabei war das Ziel, mit einem Minimum an Rechenleistung auszukommen, indem die peripheren, unabhängig von der CPU laufenden Funktionen eines gut ausgestatteten, aber dennoch einfachen 8-Bit-Mikrocontrollers genutzt wurden. Am Ende der Entwicklung, die hier beschrieben wird, konnte der verwendete PIC sogar zwei verschiedene Zeitzeichensignale gleichzeitig empfangen und dekodieren!

Wie kann man mit heute verfügbaren Mitteln einen Zeitsignalempfänger bauen? Da man letztendlich einen Mikrocontroller (MCU) benötigt, um die codierte Zeit zu extrahieren, lag der Ansatz nahe, möglichst viele der im Mikrocontroller vorhandenen Ressourcen zu nutzen, um damit auch den Empfänger zu realisieren. Es sollte also so viel wie möglich in Software und so wenig wie möglich mit zusätzlicher Hardware verwirklicht werden. Außerdem sollten keine exotischen und nur schwer beschaffbaren Bauteile verwendet werden, sondern, soweit möglich, nur Standard-Bauteile zum Einsatz kommen. Dies gilt insbesondere für SMD-Bauteile: Kondensatoren und Widerstände mit 0805-Footprint kann man noch gut händisch löten und die Schaltung dennoch platzsparend gestalten. Bei einem vielpoligen IC dagegen mit womöglich sehr kleinem Abstand der Beinchen hört der Spaß aber endgültig auf.

Neben dem DCF77-Signal auf 77,5 kHz aus Mainflingen in der Nähe von Frankfurt gibt es noch eine Vielzahl von Sendern in Europa und weltweit, die die aktuelle Uhrzeit ausstrahlen: MSF60 auf 60 kHz in Großbritannien, TDF162 auf 162 kHz in Frankreich oder auch WWVB auf 60 kHz in den USA. Doch nicht nur dedizierte Zeitzeichen-, sondern auch Rundsteuersender wie der DCF49 auf 129,1 kHz (ebenfalls aus Mainflingen) oder DCF39 auf 139 kHz in Burg bei Magdeburg senden die aktuelle Uhrzeit. Dabei haben DCF39 und DCF49 erfreulicherweise eine Sendeleistung von 100 kW, also deutlich mehr als die 50 kW beim DCF77. Auch diverse Meteo-Dienste wie der Seewetterbericht sind nach wie vor auf diesen Frequenzen aktiv. Wäre es nicht schön, alle diese Stationen (und noch mehr) im LW-Bereich nach eigener Wahl empfangen zu können?

Aller Anfang ist schwer

Erste Versuche, das DCF77-Signal an einer auf 77,5 kHz abgestimmten Ferritantenne zu verstärken und auf dem Oszilloskop einen 77,5-kHz Träger zu sehen, dessen Amplitude sich im Sekundentakt ändert, wurden zunächst bitter enttäuscht. Leider sind auf Langwelle sehr viele lokale Störquellen

vorhanden, Schaltnetzteile und LED-Beleuchtung in allen Formen und Farben und vor allem fernöstlicher Herkunft sorgen für einen Störpegel, der teils erheblich höher ist als das Nutzsignal. Schaut man sich dieses Kauderwelsch aus allen möglichen Signalen an, fällt es schwer zu glauben, dass sich hieraus das DCF77-Signal rekonstruieren lässt.

Die kommerziellen Module sind anscheinend als einfacher Geradeaus-Empfänger konzipiert und verwenden einen 77,5-kHz-Quarz als sehr schmalbandiges Bandpass-Filter. Das ist einfach und preiswert, aber man empfängt eben auch nur 77,5 kHz ±10 Hz und nichts anderes. Spätestens hier wird klar: Ohne massive (und wie wir später sehen werden, digitale) Filterei wird das nichts mit dem ungetrübten Zeitzeichen-Empfang.

Werfen wir zunächst einen Blick auf die Blockschaltung des voll ausgebauten Zeitzeichenempfängers (**Bild 1**). Gehen wir diese Blöcke der Reihe nach durch und füllen sie mit Inhalt!

Preselektor mit MOSFET-Kapazitätsdiode

Auch noch so effiziente und steilflankige, in Software realisierte Filter und der beste und schnellste A/D-Wandler nützen nichts, wenn dieser mit Störsignalen übersteuert ist und das Nutzsignal im Rauschen untergeht. An einem Preselektor, der erst einmal das Gros der Störungen beseitigt, führt kein Weg vorbei. Als Preselektor kommt üblicherweise ein LC Parallel-Resonanzkreis mit abstimmbaren Kondensator zum Einsatz (**Bild 2**). Dier Resonanzfrequenz des Schwingkreises ergibt sich zu:

$$f_0 = \frac{1}{2\pi\sqrt{LC}}$$

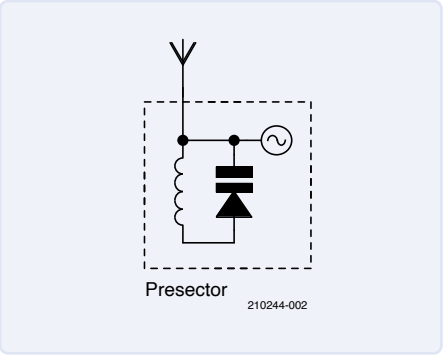


Bild 2. Prinzip eines Preselectors.

Verwendet man die zum Empfang im LW-Bereich übliche Ferrit-Antenne mit einer Induktivität von typisch 1500 µH, so ergibt sich eine notwendige abstimmbare Kapazität zwischen 400 pF und 6,7 nF, wenn man einen Frequenzbereich von 50…200 kHz abdecken möchte. Leider sind in dieser Größenordnung keine Kapazitätsdioden verfügbar.

Ich erinnerte mich, dass jeder Halbleiter eine ungewollte Sperrschichtkapazität besitzt, welche generell mit der Struktur-Größe ansteigt und spannungsabhängig ist. Erste Messungen an einem Leistungs-MOSFET sahen erfolgsversprechend aus, und aus einem Sortiment unterschiedlicher Typen lieferte ein IRF640 die besten Ergebnisse mit 500 pF bei 20 V_{UDS} und 3 nF bei 1 V_{UDS}. Um den Bereich noch besser durchstimmen zu können, wurden weitere Kapazitäten (C30…C34) hinzugefügt, welche mit bipolaren Kleinsignal-Transistoren (T30…T34) bei Bedarf zum Power-MOSFET (T35 und T36) parallelgeschaltet werden können. Der Preselektor in **Bild 3** lässt sich dann bei einer 1500-µH-Induktivität im Bereich von etwa 30…300 kHz abstimmen.

Um den Abstimmbereich des MOSFETs möglichst gut nutzen zu können, muss dieser mit einer höheren Spannung angesteuert werden, als der DAC im Mikrocontroller zu liefern in der Lage ist. Daher wird das Signal von einem Operationsverstärker LM358 von etwa 0…4,2 V auf den gewünschten Spannungsbereich von 0…30 V angehoben.

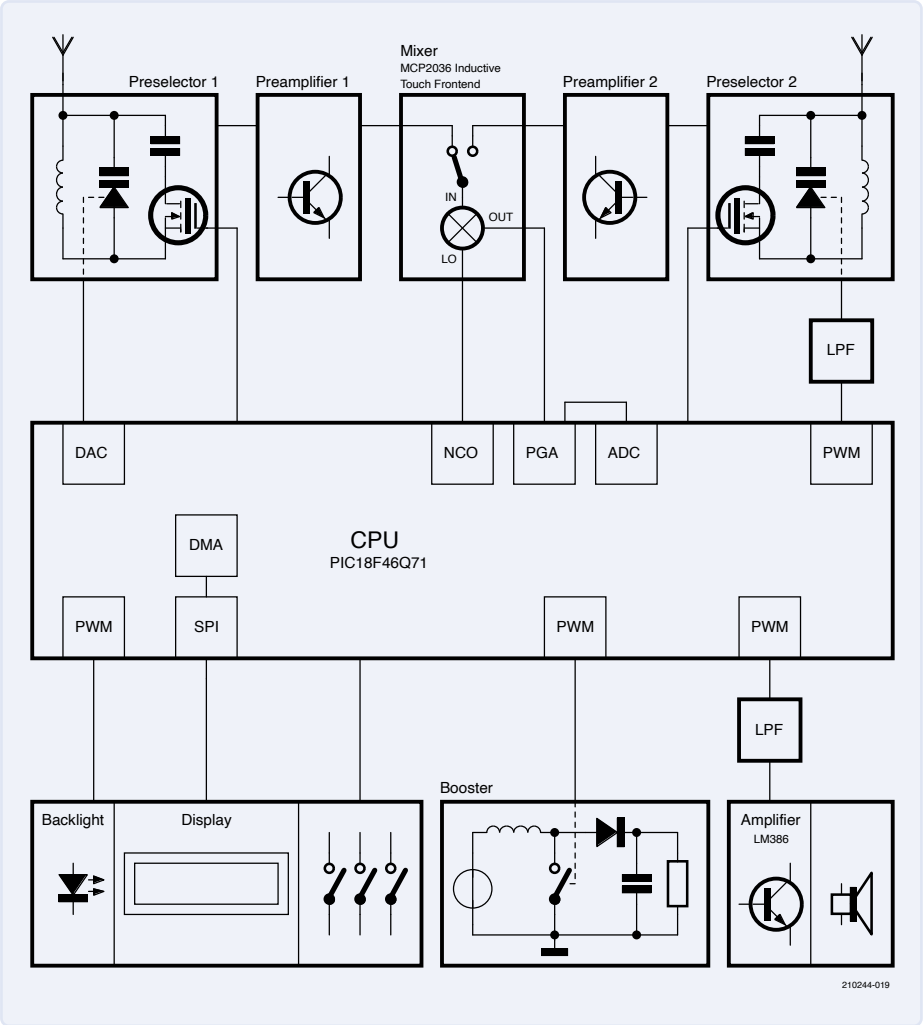


Bild 1. Blockschaltbild des Zeitzeichen-Empfängers.

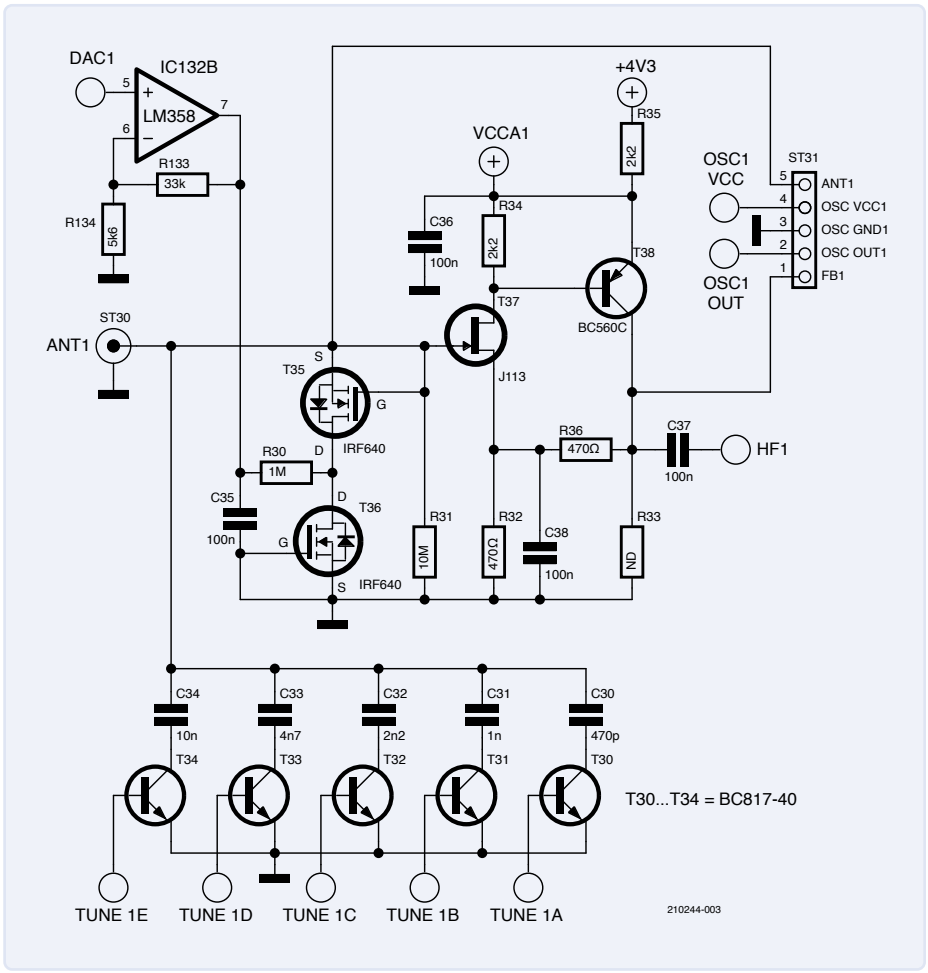


Bild 3. Der Preselector mit angeschlossenem Verstärker für das Antennensignal.

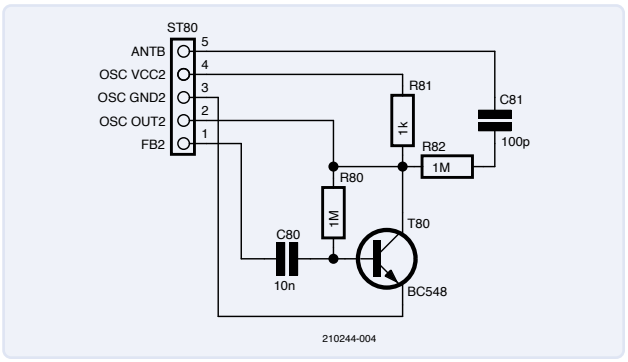


Bild 4. Oszillator, realisiert mit Transistor-Rückkopplung.

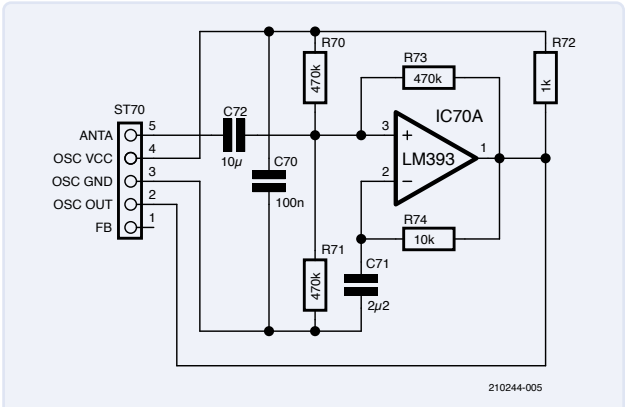


Bild 5. Ein Oszillator, der auf einem Opamp basiert.

Schwingkreis-Alternativen

Woher weiss man aber, auf welcher Frequenz der Preselektor bei gegebener Spannung abgestimmt ist? Hierfür koppelt man das verstärkte Signal auf den Antenneiungang zurück und es entsteht ein Schwingkreis. .Um verschiedene Konzepte ausprobieren zu können, kann dieser Teil der Schaltung am Steckverbinder ST31 (**Bild 3**) ausgetauscht werden.

Eine einfache Möglichkeit, die Rückkopplung zu realisieren, zeigt **Bild 4**. Hier wird die benötigte Phasenverschiebung und weitere Verstärkung mit einem zusätzlichen Transistor (T80) realisiert. Diese Schaltung hat den Vorteil, dass sie den Schwingkreis nur sehr gering bedämpft, jedoch stellte es sich als schwierig heraus, diese über einen weiten Bereich zuverlässig anschwingen zu lassen. Auf dieses Weise wird aus dem Preselektor ein Schwingkreis, dessen Schwingfrequenz in etwa der Resonanzfrequenz entspricht, jedenfalls im LW-Bereich. Ein kleiner Haken: Mit der anvisierten Empfangsfrequenz ändert sich leider auch die Phasenverschiebung vom Eingang bis zur Rückkopplung, so dass der Oszillator nicht immer mit der erwarteten Resonanzfrequenz schwingt.

Im zweiten Anlauf habe ich eine alte Schaltung nachempfunden, bei der mit Hilfe eines Komparators ein LC-Messgerät realisiert worden ist (**Bild 5**). Sie schwingt über einen weiten Induktivitäts- und Kapazitätsbereich an und trifft sehr gut die erwartete Resonanzfrequenz. Wie so oft in der Elektronik geht aber eine Verbesserung an einer Stelle mit einer Verschlechterung an einer anderen einher. Zwar wurde die Klemmung und Rückkopplung schon sehr hochohmig ausgeführt, dennoch ist der Eingangswiderstand deutlich größer als bei der erstgezeigten Lösung. Drei aus HF-Sicht parallel geschaltete 470-kΩ-Widerstände (R70, R71 und R73), also in etwa 150 kΩ, dämpfen den Schwingkreis wieder stärker. In der Praxis hatte dies auf die Empfangsqualität kaum Einfluss, außer dass die Gesamtverstärkung etwas erhöht werden muss.

Bei dieser Variante hätte ich gerne einen der in der MCU integrierten Komparatoren benutzt. Zwar bietet der in der MCU integrierte Pin-Multiplexer viele Möglichkeiten, welches Signal an welchem I/O-Pin herausgeführt werden soll. Bei massivem Einsatz der Analog-Peripherie stößt man hier jedoch auch an Grenzen, jedenfalls wenn das empfindliche Antennensignal nicht kreuz und quer

über das Platinen-Layout geführt werden soll. Von daher habe ich mich für einen externen Komparator (IC70) entschieden, um die Leitungswege möglichst kurz zu halten.
Am Ende bin ich bei der Schaltung in **Bild 6** als Kompromiss gelandet. Der Schwingkreis wird hier mit einem kurzen Impuls von 200 ns an ST31/90 (Pin 2) über R90/D90 zuverlässig zum Schwingen angeregt. Die Schwingung hält jedoch ohne Rückkopplung nur wenige Perioden an, was wiederum zu kurz ist, um sie mit den integrierten Timern genau genug zu messen. Daher wird das verstärkte Signal einem der in der MCU integrierten Komparatoren über ST31 (Pin 4) zugeführt, der über seinen Ausgang an ST31 (Pin 2) wieder Energie zuführt und das System so am Schwingen hält (siehe hierzu Bild 2). Die Vergleichsspannung am Komparator, die sich über einen DAC einstellen lässt, bestimmt hierbei, wie gut man die erwartete Resonanzfrequenz trifft. Vielleicht haben Sie, liebe Leser, ja eine Idee, wie man diese im laufenden Betrieb automatisch anpassen kann, so dass dies auch bei unterschiedlichen Zielfrequenzen zuverlässig funktioniert. Welche dieser drei Optionen man auch immer wählt, am Ende erhält man ein gut ausgesteuertes Digitalsignal, das einem der Zähler im Controller zugeführt wird. Da die angestrebte Frequenzauflösung nicht besser als 100 Hz sein muss, reicht es aus, die Frequenz mit einem Zeitfenster von 1/100 s = 10 ms zu erfassen.
Die Software misst beim Start zunächst die Frequenzbereiche aus, welche durch Zuschalten von C30...C34 und bei jeweils minimaler und maximaler Abstimmspannung erreicht werden können. Der am besten passende Bereich wird dann für den endgültigen Abstimmvorgang ausgewählt und die Abstimmspannung, beginnend mit halber Aussteuerung, treppenförmig nach oben oder unten angepasst. Die benötigte Zeit, um auf eine vorgegebene Empfangsfrequenz zu tunen, beträgt insgesamt weniger als 100 ms.

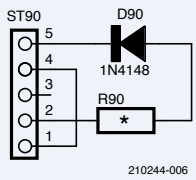


Bild 6. Ein Widerstand und eine Diode regen den Oszillator an.

Alternativ könnte man den Schwingkreis auf einer bekannten Frequenz anregen und die Kapazität so lange nachregeln, bis man die geringste Dämpfung und damit höchste Amplitude erhält. Hierzu kann man einerseits einen sehr hochohmigen Widerstand am Pfostenverbinder einfügen (ST31, Pin 2, Pin 5) und mit einer hochgenau am I/O-Pin erzeugten Referenzfrequenz (dazu später mehr) den Schwingkreis anregen.
Zusätzlich habe ich um das gesamte Platinenlayout einen Leiterbahnzug gelegt, welcher ebenfalls ausreichend ist, um die Antenne auf der gewünschten Abstimmfrequenz anzuregen. Dies funktioniert sehr gut, dauert aber im Vergleich zu den zuvor genannten Lösungen lange, da sehr viele Iterationsschritte notwendig sind. Am sinnvollsten erscheint es mir deshalb, beide Lösungen zu kombinieren. Man nähert sich mit dem Oszillator grob der Zielfrequenz und macht die Feinabstimmung dann durch Erregung und Abstimmung auf maximale Resonanz. Der (mühsam abgetippte) Screenshot in **Bild 7** zeigt einen typischen Kalibrier- und Abstimm-Vorgang auf 77,5 kHz. Leider ist, wie **Bild 8** zeigt, die Kapazitäts-/Spannungskennlinie der Sperrschichtkapazität nicht linear. Mit anderen Worten, bei kleinen Spannungswerten sind die Kapazitätsänderungen sehr groß, während sie mit höheren Spannungswerten immer kleiner werden.
Um dem abzuhelfen, können wir uns eine spezielle Eigenschaft des DAC im Controller zunutze machen, nämlich dass dieser mit einer internen, variablen Referenzspannung versorgt werden kann. Bei kleinen Ausgangsspannungen wählen wir 1,024 V als Referenzspannung und erhalten einen entsprechend kleinen Spannungshub. Bei höheren Abstimmspannungen wird dann schrittweise die Referenzspannung bis auf 4,096 V erhöht.
Das vorselektierte Signal steht jetzt zur weiteren Verarbeitung zur Verfügung. Der im Microcontroller verbaute A/D-Wandler hat jedoch leider eine maximale Wandlungsrate von 300 kHz.

Mischer und mehr

Da für die softwareseitige Implementierung des Empfängers die Abtastrate mindestens doppelt, besser viermal so viel wie die höchsten Empfangsfrequenz betragen sollte, muss das Empfangssignal zunächst auf eine tiefere Frequenz heruntergemischt werden, so dass es im weiteren Verlauf vom Mikrocontroller verarbeitet werden kann.

CHECKING TUNER ...		
Cx	F-LO	F-HI
0	127,000	227,200
1	106,300	147,500
2	91,400	113,900
3	82,700	98,200
4	74,400	84,900
5	69,400	77,700
6	64,800	71,400
7	61,300	66,900
8	56,000	60,000
9	53,800	57,300
10	51,400	54,600
11	49,700	52,600
12	47,700	50,200
13	46,300	48,700
14	44,800	46,900
15	43,700	45,600
16	39,800	41,200
17	39,000	40,400
18	38,100	39,400
19	37,400	38,600
20	36,500	37,600
21	35,900	36,900
22	35,200	36,100
23	34,600	35,600
24	33,500	34,400
25	33,000	33,800
26	32,500	33,200
27	32,100	32,800
28	31,500	32,200
29	31,100	31,700
30	30,600	31,300
31	30,300	30,800
TUNING ... (5)		
[512]	77,760	>
[256]	76,570	<
[384]	77,280	<
[448]	77,540	>
[416]	77,420	<
[432]	77,480	<
[440]	77,510	>
[436]	77,490	<
[438]	77,500	

Bild 7. Abstimmung des Tuners.

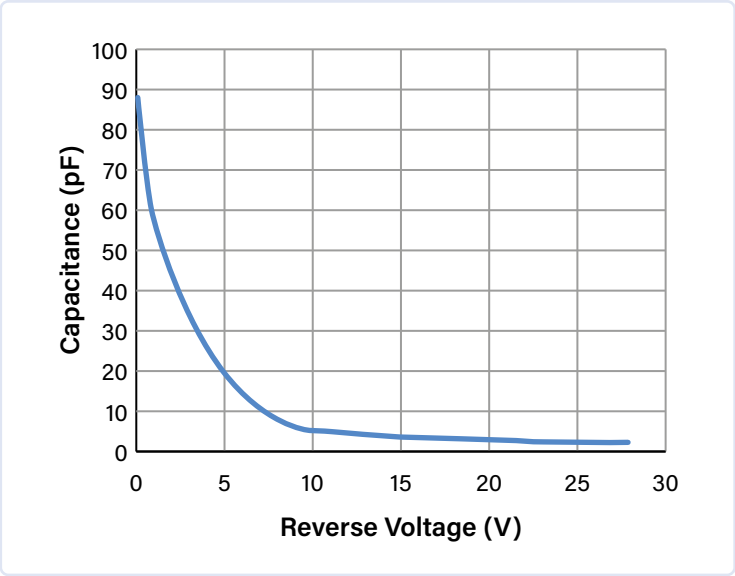


Bild 8. Nichtlinearer Verlauf der Sperrschichtkapazität. (Quelle: Digikey, https://tinyurl.com/3js37yjn)

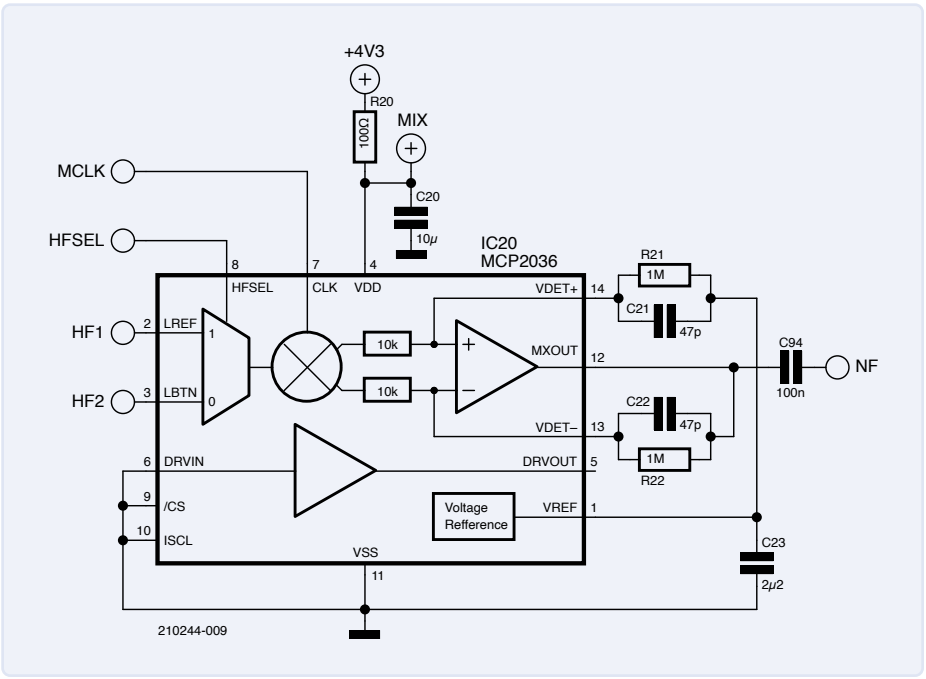


Bild 9. Blick ins Innere des MCP2036. (Quelle: Microchip)

Ich bin mehr durch Zufall auf einen interessanten Baustein gestoßen, den MCP2036 von Microchip, der eigentlich als Frontend für induktive Touch-Sensoren gedacht ist, aber auch einen Mischer enthält (**Bild 9**).

Über die beiden Eingänge LREF und LBTN wird das zu mischende Signal zugeführt. Am CLK-Eingang wird die Mischfrequenz (Local Oscillator) als Digitalsignal eingespeist, so dass sich am Ausgang eine Zwischenfrequenz im Bereich von einigen Kilohertz ergibt. Praktischerweise enthält der Chip auch noch eine Referenzspannungsquelle, einen Treiber, der hier nicht benutzt wird, sowie einen Operationsverstärker, der das Signal ein weiteres Mal verstärken und bei der Gelegenheit auch als Tiefpass filtern kann, um das ungewünschte Mischprodukt zu entfernen.

Das nun endgültig aufbereitete und heruntergemischte Signal wird dem A/D-Wandler zugefügt. Hier wird das Signal mit der vierfachen Zwischenfrequenz abgetastet, um später die I/Q-Komponenten zu erhalten und damit das empfangene Signal digital zu demodulieren.

Um aber AM-modulierte Stationen wie den Zeitzeichen-Sender DCF77 empfangen zu können, muss die Verstärkung der Eingangsstufe so geregelt werden, dass der A/D-Wandler „angemessen" angesteuert wird, also nicht zu hoch und nicht zu niedrig. Dazu habe ich mich der im Mikrocontroller PIC18F46Q71 integrierten Operationsverstärker bedient, die als invertierende Verstärker konfiguriert und kaskadiert sind. Erfreulicherweise gehört auch ein Widerstandsnetzwerk zur Einstellung der Verstärkung dazu, das in dieser Konfiguration eine bis zu 15-fache Verstärkung pro Operationsverstärker erlaubt. Auch die Referenzspannung des ADC lässt sich (wie beim DAC) zwischen 1,024 V und 4,096 V auswählen, entsprechend einer weiteren vierfachen Verstärkung. Und benutzt man dann nur acht der verfügbaren zwölf Bit des A/D-Wandlers, erhalten wir eine weitere einstellbare 16-fache Verstärkung. Insgesamt lässt sich also die Verstärkung im Bereich 1:(15×15×16×4) = 1:14400 einstellen (**Bild 10**).

Da sich, wie sich noch zeigen wird, bereits sechs Bits ausreichend sind, um das Signal zuverlässig zu dekodieren, vergrößert sich der Dynamikbereich zu 1:14400x4 = 1:57600, also über 95 dB. Das sollte reichen, egal ob man 2000 km vom Sender entfernt ist oder einige Meter daneben steht.

Nun muss noch die Mischfrequenz für den Mischer variabel erzeugt werden. Auch hierfür nutzen wir die Peripherie des Mikrocontrollers. Üblicherweise enthalten Mikrocontroller einen Timer/Counter, mit dem ein Taktsignal um 1/n geteilt werden kann. Dies ist für viele Anwendungen ausreichend, aber die Auflösung ist sehr schlecht: Angenommen, die Referenzfrequenz beträgt 16 MHz, dann ergibt sich bei n = 80 eine Ausgangsfrequenz von 200 kHz, mit dem nächstgrößeren Teiler n = 81 jedoch 197,5 kHz, also ein Sprung von 2,5 kHz.

Der hier verwendete PIC-Microcontroller hat neben diesen Standard-Timern noch einen weiteren, den so genannten **Numeric Controlled Oscillator** (NCO). Dies ist, wie **Bild 11** zeigt, ein Addierer, der zu seinem letzten Ergebnis jeweils einen vorher festgelegten Wert addiert, welcher dann das neue Ergebnis repräsentiert. Dies geschieht zudem in sehr kurzen Intervallen, hier mit der Taktfrequenz des Controllers von 64 MHz.

Jedes Mal, wenn der Addierer einen Überlauf erzeugt, wird entweder ein Ereignis generiert oder ein Ausgang getoggelt. Hiermit lassen sich dann bei niedrigen Frequenzen Auflösungen bis hinunter zu einigen Zehntel Hertz realisieren. Zugegeben, das Ausgangsignal weist einen Jitter auf, den wir hier jedoch vernachlässigen können.

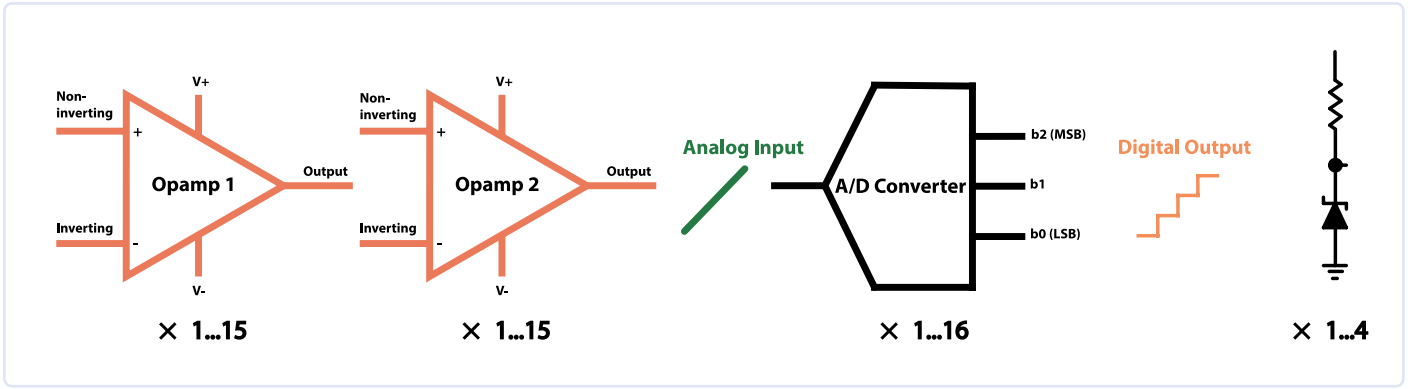


Bild 10. Einstellbare Gesamtverstärkung.

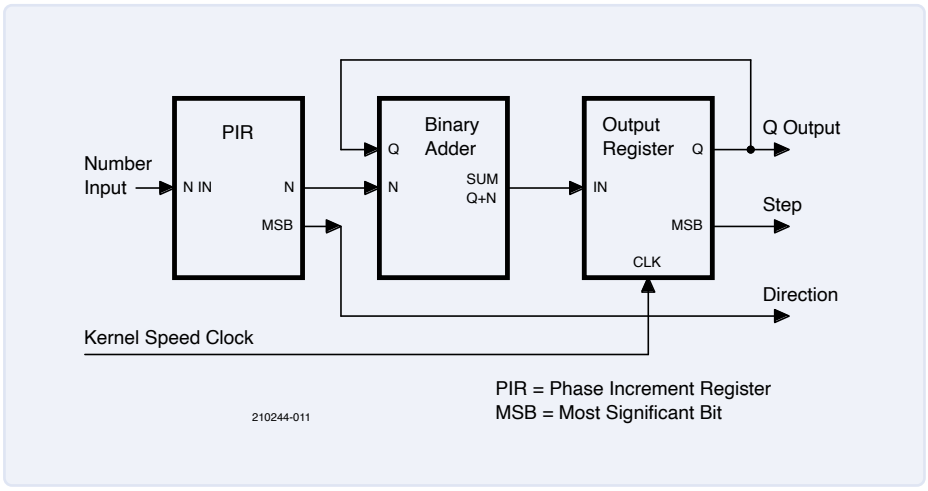


Bild 11. Numerisch gesteuerter Oszillator.

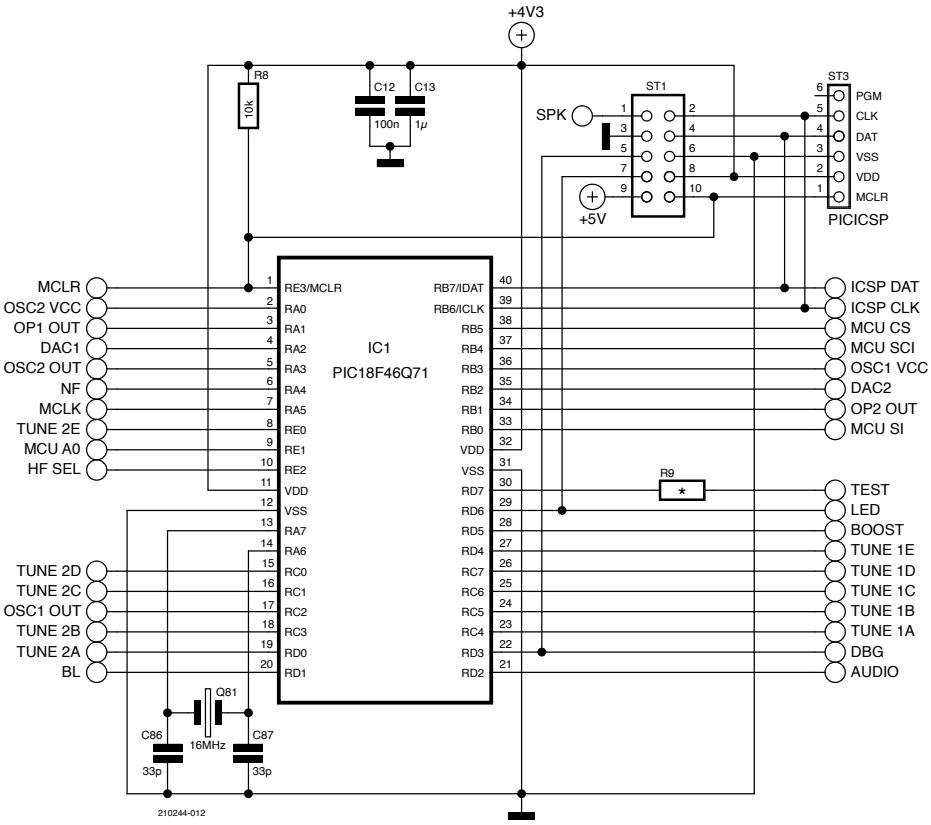


Bild 12. Das Hirn des Zeitzeichenempfängers, der PIC18F46.

Der Mikrocontroller

Wir haben schon sehr viel über den Mikrocontroller gesprochen. Bei den vielen Funktionen und Aufgaben, die diese MCU erfüllen muss, ist es nahezu ein Wunder, dass statt eines 32-Bit-Hochleistungscontrollers „nur" ein moderner 8-Bit Controller PIC18F46Q71 gewählt wurde [1], der sogar in einem makerfreundlichen DIP-Plastikgehäuse erhältlich ist. Er verfügt über reichhaltige Peripheriehardware, die wir hier ausgiebig nutzen. Außergewöhnlich in dieser Klasse sind der 12-bit-ADC mit 300 kHz Samplerate, zwei integrierte Operationsverstärker mit einem Verstärkungs-Bandbreitenprodukt von jeweils 5,5 MHz, ein mehrkanaliger DMA-Controller, mehrere schnelle PWM-Kanäle mit Dead-Time Generator, zwei UARTs sowie das Herz der Schaltung, den Numeric Controlled Oscillator. Er enthält außerdem noch andere interessante Peripherie wie programmierbare Logik-Zellen (mit denen sich beispielsweise auch ein 90° Grad phasenverschobenes Signal erzeugen lässt), zwei schnelle Komparatoren oder eine Hardware-CRC-Einheit, die jedoch in diesem Projekt nicht zum Einsatz kommen.

Da der Controller außerdem über einen Hardware-Multiplier verfügt, bringt er die entsprechende Rechenleistung mit, um auch komplexe Berechnungen wie beispielsweise zur Ausführung eines Hilbert-Filters zur Unterdrückung der Spiegelfrequenz ausführen zu können.

Bild 12 zeigt den Controller mit all seinen Ein- und Ausgangssignalen. ST3 ist der ICSP-Programmierport, außerdem wurden mehrere I/Os zum Debuggen auf den Steckverbinder ST1 geführt. Hiermit lassen sich serielle Daten mit Hilfe des UARTs übertragen, um beispielsweise den Arbeitsbereich des Preselektors anzuzeigen oder weitere hochauflösende PWMs konfigurieren, um Analogwerte auszugeben, zum Beispiel zum Debuggen der einzelnen Stufen des SDR-Demodulators.

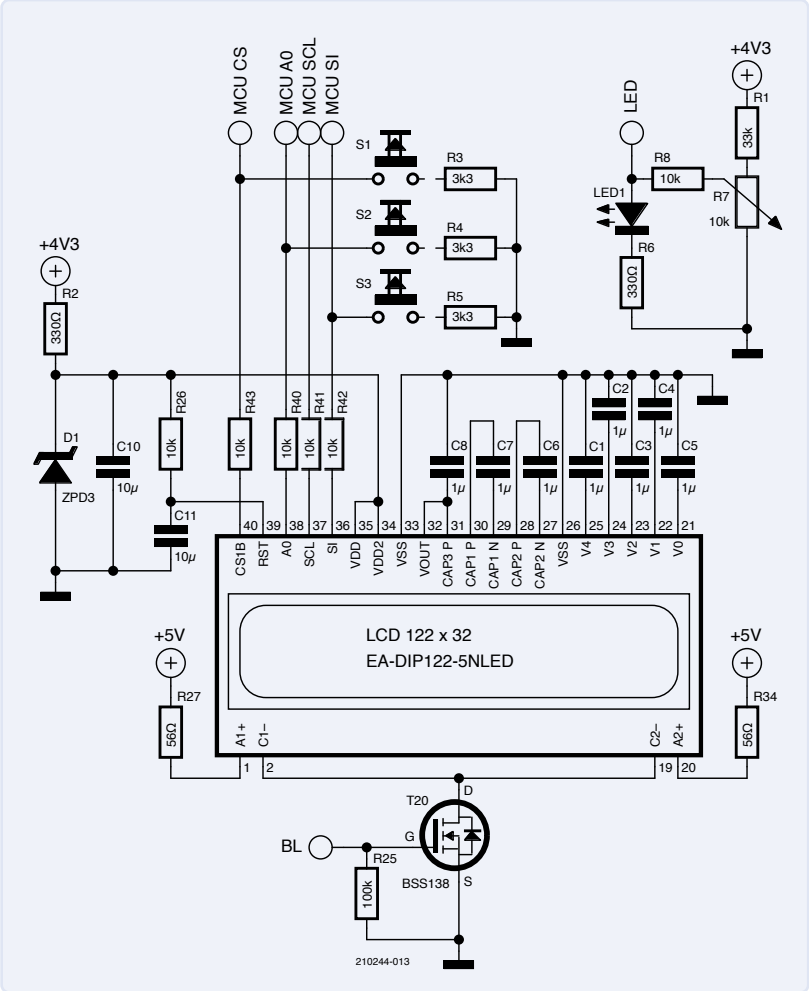


Bild 13. Das Benutzerinterface besteht aus einem Display, einer LED und einigen Tastern.

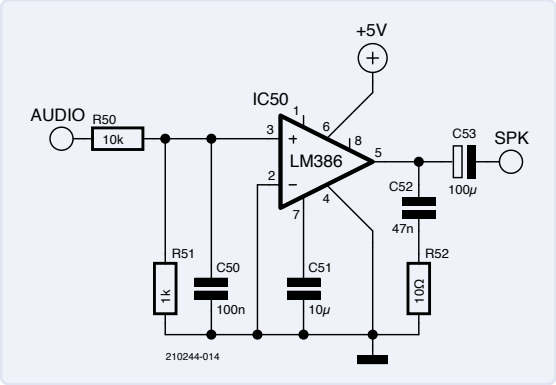


Bild 14. Audioausgabe mit dem altbewährten LM386.

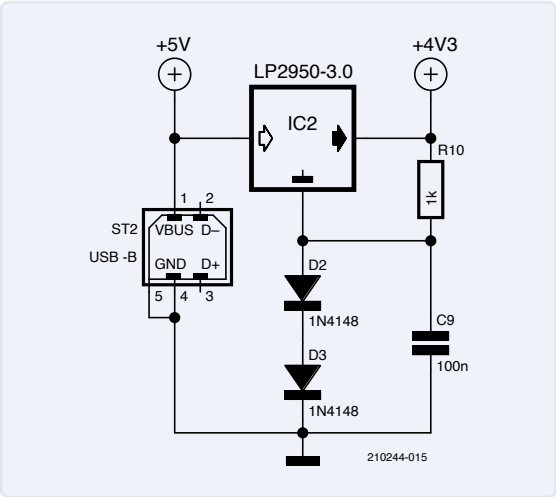


Bild 15. Die 5 V des USB-Steckernetzteils werden für den Digitalteil auf 4,4 V verringert …

Ein GUI gehört dazu

Natürlich sollte der Empfänger auch eine Benutzerschnittstelle in Form eines kleinen Grafik-Displays und Bedienknöpfen sowie einem Potentiometer erhalten (**Bild 13**). Dabei fiel die Wahl auf eines der bekannten DOG-Displays (Display-On-Glass), das über eine SPI-Schnittstelle an den Host-Controller gebunden wird. Ärgerlicherweise ist die SPI-Schnittstelle beim Display nur unidirektional ausgeführt, was heißt, dass man Daten nur zum Display schreiben, nicht aber wieder zurücklesen kann.

Die anzuzeigende Grafik muss daher zunächst im RAM des Microcontrollers erzeugt und dann periodisch an das Display übertragen werden. Hier greifen wir erneut auf die Peripherie des Controllers zurück, in dem Fall auf den DMA-Controller. Dieser überträgt zyklisch den Inhalt des „Display-RAMs" per SPI-Schnittstelle an das Display, was vollkommen im Hintergrund abläuft und keinerlei Rechenleistung beansprucht.

Da das Display insgesamt nur wenige hundert Mikroampere aufnimmt, wurde die zusätzliche Spannungsversorgung von 3,3 V mit einer Z-Diode realisiert. Die Pegelanpassung erfolgt über Serienwiderstände und den parasitär an den Displayeingängen vorhanden Bulk-Dioden.

Sound

Da auch einige Radiostationen im Empfangsbereich liegen (zum Beispiel BBC World auf 198 kHz), wurde außerdem ein Lautsprecher samt Verstärker in das Projekt aufgenommen. Dabei sollte wieder möglichst viel der im Controller vorhandenen Peripherie zur Implementierung des Verstärkers genutzt werden. Die erste Wahl fiel auf ein Klasse-D-Design, das heißt, die Endstufen-Transistoren werden direkt vom Controller entweder digital voll aufgesteuert oder vollständig gesperrt. Durch Wahl einer geeigneten Taktfrequenz, welche tunlichst keine Harmonischen im Bereich der Empfangsfrequenz enthalten sollte, könnte eine Störung des Empfangs vermieden werden. Doch was bei der Implementierung des Spannungsboosters (siehe folgenden Abschnitt) so gut funktioniert hat, ist hier nicht möglich, sei es wegen der doch deutlich höheren Leistungen oder der wesentlich längeren Kabelwege zu den Lautsprechern oder beidem.

So habe ich einen konventionellen analogen Verstärker rund um den Klassiker LM386 verwendet. Das eigentliche Audiosignal wird von der MCU zunächst digital verarbeitet, weil es ja nach der A/D-Wandlung ohnehin digital als I/Q-Komponenten vorliegt. Erst am Ausgang werden die errechneten Samples einer hoch taktenden PWM-Einheit zugeführt, passieren danach ein Tiefpassfilter und werden anschließend dem Eingang des Verstärker-ICs zugeführt (**Bild 14**).

Da die verwendete PWM-Einheit direkt aus dem Systemtakt von 64 MHz gespeist werden kann, ergibt sich bei einer Auflösung von 12 Bit eine PWM-Frequenz von 64 MHz / 2^{12} = 15,625 kHz. Dies eröffnet die Möglichkeit, neben der eigentlichen Audio-Ausgabe auch noch die Lautstärke einzustellen. Als Lautsprecher kommt ein Typ mit einer Größe von 40x40 mm zum Einsatz, der gleichzeitig als mechanische Stütze dient.

Spannungsversorgung

Ein paar kurze Worte zur Spannungsversorgung (**Bild 15**). Der gesamte Empfänger soll ja an einem USB-Port betrieben werden. Da die am USB-Port vorhandene Spannung weitestgehend unstabilisert ist, erzeugt

der 3V-LDO-Spannungsregler IC2 aus der USB-Spannung zunächst eine saubere Betriebsspannung für die Signalverarbeitung. Der Fußpunkt des Reglers ist über zwei Silizium-Dioden angehoben, so dass eine Ausgangsspannung von VDD = 4,3 V entsteht. Dies ist niedrig genug, um Spannungsschwankungen am USB-Port auszugleichen und hoch genug für die im Controller benutzten Referenzspannungsquellen.

Kehren wir in Gedanken an den Anfang zurück: Für die Abstimmspannung muss die USB-Spannung von 5 V ja kräftig angehoben werden. Dies geschieht mit einem erfrischend einfach aufgebauten ungeregelten Aufwärts-Wandler (**Bild 16**). Das PWM-Signal zur Ansteuerung von T70 wird mit entsprechendem Tastverhältnis vom Mikrocontroller erzeugt. Die Z-Diode begrenzt die Ausgangsspannung auf maximal 33 V. Die Effizienz dieser Schaltung ist zwar schlecht, aber da die Stromentnahme sehr klein ist und im Prinzip nur durch den Ruhestrom des Operationsverstärkers bestimmt wird, ist dies in diesem Fall vernachlässigbar.

Bild 17. Selbstgewickelte Rahmenantenne.

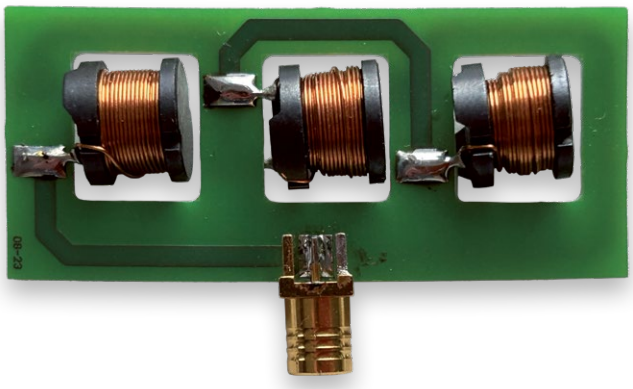


Bild 18. Mit missbrauchten Festinduktivitäten kann man einen guten Empfang erreichen.

Antennen

Die Antenne wird über ein SMB-Buchen-/Steckerpaar mit dem Empfänger verbunden. Sie ist wegen der Steckverbindung drehbar und kann bestmöglich in Richtung des Senders ausgerichtet werden.

Für die ersten Empfangsversuche habe ich eine Ferritantenne aus einem alten Funkwecker ausgebaut und auf einen Platinenabschnitt geklebt. Der Abstimm-Kondensator musste vorher natürlich entfernt werden, da dieser ja nun im Preselektor nachgebildet wird. Da fertig gewickelte und bisweilen sogar leere Ferritstäbe mittlerweile schwer zu beschaffen sind (und das Wickeln auch nicht jedermanns Sache ist), habe ich nach einer alternativen Antenne gesucht. Auf dem Restetisch eines Elektronikversenders habe ich dann eine Rahmenantenne gefunden und diese ebenfalls auf einen Platinenabschnitt geklebt. Die Empfangsergebnisse waren vergleichbar mit der Ferritantenne.

Daraufhin habe ich eine Platine mit Einkerbungen als Wickelhilfe entworfen (**Bild 17**) und diese Antenne selbst hergestellt. Ich musste sehr dünnen Kupferlackdraht verwenden, um die erforderliche Anzahl Windungen aufzubringen. Insgesamt ein sehr mühseliger Vorgang, aber prinzipiell umsetzbar.

Beim Kramen in meinen Bauteile-Schubladen fiel mir dann eine große Tüte Fest-Induktivitäten in die Hand, die ich vor kurzem erworben hatte. Lässt sich so etwas vielleicht als Antenne missbrauchen? Wenn man hiervon einige in Serie schaltet, entspricht das ja von der Anordnung her einer Ferritantenne, nur dass sie in mehrere Stücke geteilt ist (**Bild 18**). Tatsächlich konnte ich mit dieser Anordnung vergleichbare Empfangsergebnisse erzielen, zwar mit deutlich geringerer Signalamplitude, aber erstaunlich guter Empfangsqualität: DCF77 ließ sich problemlos aus 800 km Entfernung empfangen! Ich habe die Induktivitäten sowohl axial als auch radial montiert – und beides funktionierte. Kombiniert man beide Varianten, also abwechselnd radiale und axiale Ausrichtung, sollte sich eine Antenne realisieren lassen, die weitgehend richtungsunabhängig ist. Wenn beispielweise die axial angeordneten Induktivitäten gerade von einem Maximum an Feldlinien durchströmt werden, sehen die radial angeordneten Induktivitäten keine wirksamen Feldlinien mehr und umgekehrt.

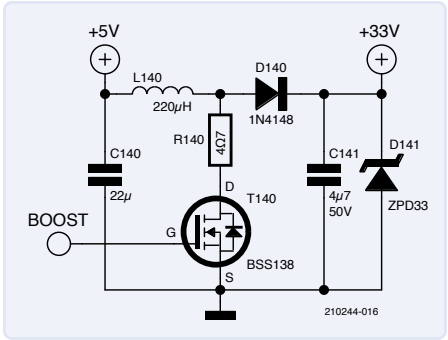


Bild 16. ...und für die Abstimmung auf 33 V erhöht.

Software

Grundsätzlich bedient sich die Software der Prinzipien eines klassischen SDR, sprich Demodulation des Signals anhand der I/Q-Komponenten. Wie dies grundsätzlich funktioniert, wurde schon in vielen Elektor-Artikeln [2][3] beschrieben, weswegen ich es hier nicht in allen Details wiedergeben muss. Einzig der Softwareschnipsel im **Kasten Zwei Empfänger...** ist wirklich neu und deshalb erwähnenswert.

Da diese Schaltung viel von der integrierten Peripherie Gebrauch macht, zum Beispiel bei der Erzeugung der Mischfrequenz und der Verwendung des DMA-Controllers, wird der Controller stark entlastet. So verbleibt auch diesem 8-bit-Controller genügend Rechenleistung, um die Station zu demodulieren.

Zwei Empfänger sind besser als einer

Zwar war die Schaltung ursprünglich nur dafür ausgelegt, Daten- und Zeitzeichendienste im LW-Bereich zu empfangen, da der Mischer aber über einen zweiten, umschaltbaren Eingang verfügt, habe ich kurzerhand einen weiteren Preselektor mit einer weiteren Antenne hinzugefügt. Dieser kann wahlweise für LW- oder MW-Empfang bestückt werden. Wie sich im Laufe der Software-Entwicklung gezeigt hat, ist damit sogar ein zeitgleicher Empfang auf zwei unterschiedlichen Frequenzen (!) möglich.

Leider verfügt der verwendete Controller nur über einen NCO, weswegen man zur Erzeugung der zweiten Mischfrequenz erneut in die Trickkiste greifen muss.

Zur Verwendung der niedrigeren Mischfrequenz wird einer der beiden 16-bit-Universaltimer verwendet. Da diese mit dem Systemtakt von 64 MHz getaktet werden, kommt man dennoch recht nah an die gewünschte Mischfrequenz heran. Bei 79,5 kHz (DCF77) beträgt der Fehler beispielsweise nur 200 Hz. Verwendet man den zweiten Universaltimer zur Erzeugung der Abtastrate für die Zwischenfrequenz, kann diese auf rund 1 Hz genau von 2000 Hz auf 2200 Hz angepasst werden und so der zuvor beschriebene Fehler wieder ausgeglichen werden. Der NCO des anderen Empfangskreises wird entsprechend auf die sich nun ergebende neue Zwischenfrequenz angepasst.

Natürlich hat auch der verwendete Quarz nur eine endliche Genauigkeit. Wenn diese mit 50 ppm angenommen wird, beträgt der Fehler bei 77,5 kHz bereits 4 Hz, zu viel für die bei der Dekodierung eingesetzten sehr schmalbandigen Filter. Der Fehler ist jedoch per Software bestimmbar. Wenn der gewünschte Sender mit ausreichender Signalstärke empfangen wird, kann die Frequenzkorrektur auch automatisch vorgenommen werden. Hierzu wird über viele Perioden ermittelt, ob sich der I/Q-Vektor mit oder gegen den Uhrzeigersinn dreht. Macht man dies über einen definierten Zeitraum, hat man einen Indikator, wie weit die tatsächliche Mischfrequenz von der gewünschten abweicht. Ein übergeordneter Regler kann dann die Zwischenfrequenz verkleinern oder vergrößern, um den Wert nahe Null zu bringen. Wie dies in C implementiert werden kann, zeigt folgender Code:

```
#define FILTERDEPTH 25
typedef struct {
  int16_t Ihistory[FILTERDEPTH];
  int16_t Qhistory[FILTERDEPTH];
  int32_t Iaverage;
  int32_t Qaverage;
  uint8_t index;
  uint8_t lastQuadrant;
  int8_t phaseChange;
} IQ;
IQ *iq;
void IQcallback( int16_t I, int16_t Q ) {

  uint8_t quadrant =0;

  // moving average lowpass filter I
  iq->Iaverage -= iq->Ihistory[iq->index];
  iq->Ihistory[iq->index] = I;
  iq->Iaverage += I;

  // moving average lowpass filter Q
  iq->Qaverage -= iq->Qhistory[iq->index];
  iq->Qhistory[iq ->index] = Q;
  iq->Qaverage += Q;

  iq->index++;
  if( iq->index >= FILTERDPTH )  {
    iq->index =0;
  }
  // integrate phase shift > phase error >
     estimate frequency offset of crystal
  if( iq->Qaverage<0 ) quadrant =1;
  if( iq->Iaverage<0 ) quadrant ^= 3;
  if( iq->lastQuadrant == ( (quadrant+1) & 3 ) )
    iq->phaseChange--;
  if( iq->lastQuadrant == ( (quadrant-1) & 3 ) )
    iq->phaseChange++;
  if( iq->phaseChange < -90 )
    iq->phaseChange = 90;
  if( iq->phaseChange > 90 )
    iq->phaseChange = -90;
  iq->lastQuadrant = quadrant;
}
```

Das letzte Glied in dieser Kette bildet das GUI, das auf neu berechnete Werte wartet und das Display bei Bedarf aktualisiert.

Ausblick

Wer kennt das nicht – nach dem Aufbau des ersten Prototyps kommen auf einmal viele neue Ideen auf, wie man es in einem weiteren Anlauf besser machen könnte. Dennoch bin ich dem ersten Entwurf erst einmal treu geblieben und habe ihn bis zu Ende ausgeführt. In einer Neuauflage des Entwurfs würde ich:

- die Benutzerschnittstelle um einen Dreh-Encoder erweitern, mit dem sich die Frequenz komfortabler einstellen lässt
- als Controller statt eines PIC18F46Q71 den kleineren Bruder PIC18F26Q71 ausprobieren, und zwar in doppelter Ausführung. Die erforderliche Platinenfläche bliebe fast gleich, jedoch erhielte man zwei NCOs, vier Operationsverstärker und vier Komparatoren, die sich HF-technisch gesehen besser routen ließen. Nachteilig wäre die dann notwendige Kommunikation zwischen den beiden Controllern, die sich aber gut in Grenzen halten ließe: Ein Kern könnte die Digitalisierung und Verarbeitung der Empfangsdaten übernehmen, der andere Kern das GUI. Da beide Controller aus der gleichen Taktquelle betrieben würden, wäre auch eine 1-Draht-UART-Schnittstelle im Megabit-Bereich ohne weiteres möglich.
- die verwendeten SMB-Buchsen, die fast unbezahlbar teuer geworden sind, durch die guten alten Klinkenbuchsen ersetzen. Eine 2,5-mm-Ausführung sollte hier gut funktionieren und die Antenne bliebe wie gehabt drehbar. Da diese Stecker/Buchsen auch als 3- und 4-polige Varianten erhältlich sind, könnte man sogar einen Teil der Schaltung, beispielsweise die Abstimm-Kondensatoren, auf die Antenne auslagern.

- mit entsprechender Anpassung der Preselektoren den Empfänger für den Empfang auf Mittelwelle tauglich machen oder für Amateurfunk-Zwecke gebrauchen. Praktische Versuche haben ergeben, dass der Mischer bis in den oberen einstelligen Megahertz-Bereich betrieben werden kann.

Wie man sieht, muss SDR nicht zwangsweise mit hoch getakteten 32-Bit-Controllern einhergehen, die oft in manuell schlecht lötbaren Gehäusen daherkommen. Der clevere Einsatz der Peripherie eines 8-Bit-Controllers aber kann die eine und andere Funktion ersetzen und letztendlich in einem einfach umzusetzenden Design münden.
Ein solches Projekt lebt und stirbt wie immer mit der Anzahl der Miteifernden, die daran beteiligt sind. Gerade die Umsetzung der Software benötigt sehr viel Zeit. Doch vielleicht findet sich der ein oder andere, der Spaß und Lust hat, diese mit weiterzuentwickeln. Lassen Sie es uns wissen!

RG — 210244-02

Sie haben Fragen oder Kommentare?

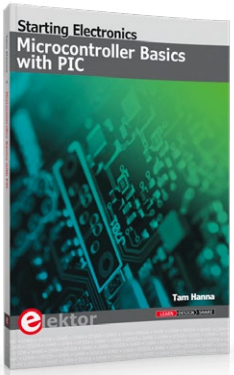
Haben Sie technische Fragen oder Kommentare zu diesem Projekt? Dann wenden Sie sich bitte an den Autor unter marco@happy-electrons.com oder an die Elektor-Redaktion unter redaktion@elektor.de.

Über den Autor

Marco ist Jahrgang 1970 und hat die Elektronik irgendwie mit in die Wiege gelegt bekommen. Er beschäftigt sich seit dem Teenie-Alter mit Elektronik, hat dann irgendwann Elektrotechnik studiert und ist seit etwa 35 Jahren im R&D-Bereich tätig. Zur Zeit arbeitet er für einen amerikanischen Halbleiter-Hersteller.
Amateurfunker ist Marco auch und hört auf das Rufzeichen DD9DD.

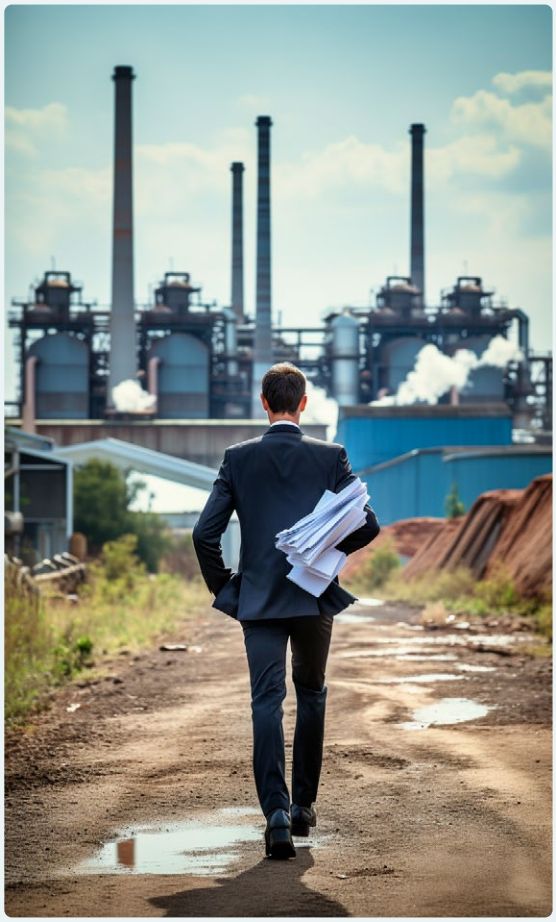
Passende Produkte

- **Bert van Dam, *PIC-Mikrocontroller*, Elektor**
  E-Buch, PDF: www.elektor.de/17423
- **Tam Hanna, *Mikrocontroller-Basics mit PIC*, Elektor**
  Buch, kartoniert: www.elektor.de/18945
  E-Buch, PDF: www.elektor.de/18946



WEBLINKS

[1] MCU PIC18F46: https://www.microchip.com/en-us/product/PIC18F46Q71
[2] Martin Oßmann, „SDR mit AVR", sechsteilige Artikelreihe Elektor 3/2012 bis 10/2012: https://www.elektormagazine.de/magazine-archive/2012
[3] Martin Oßmann, „SDR-Funkuhren", Elektor 1/2023: https://www.elektormagazine.de/magazine/elektor-289/61373



Source: Midjourney AI-generated image.
Prompt: "/imagine prompt: Back of person with lots of papers under his arm moving toward factory --ar 3:5"

▶

Due Diligence Directive

Business as Usual Will Not Do

By Priscilla Haring-Kuipers (The Netherlands)

Currently rolling through the EU is the Corporate Sustainability Due Diligence Directive or CSDDD [1]. The EU Parliament just voted generally to accept these new rules, and will continue to negotiate some of the finer points for another year. Things are about to change.

Our Sustainable Development Goals include the promotion of sustained, inclusive and sustainable economic growth. In the Paris Agreement, we promised to keep global warming within 1.5°C, and the private sector, with all their investments, must be in line with this if we are to keep our promises. Under the European Climate Law, we are committed to being climate-neutral by 2050 and to reduce our emissions by at least 55% in 2030. All of this means that there must be a change in the way companies produce and procure. Business as usual will not do.

Many issues in our global supply chains would be hard to tackle by any one company or even one country. These new due diligence rules intend to even the playing field within the EU market and prevent fragmentation in law. Germany already has a Supply Chain Act or "Lieferkettensorgfaltspflichtengesetz" [2], while other EU countries have limited requirements. By connecting all the big companies of many countries under common EU legislation, the power to effectively and sufficiently address environmental concerns and social inequality can be combined. Moving together is better.

The Ask

This directive concerns big EU companies with hundreds of employees and millions in turnover, as well as non-EU companies that make millions in the EU market. At its core, the CSDDD [3] is a way to force these big companies to take responsibility "by carrying out the following actions:

(a) integrating due diligence into their policies
(b) identifying actual or potential adverse impacts
(c) preventing and mitigating potential adverse impacts, and bringing actual adverse impacts to an end and minimizing their extent;
(d) establishing and maintaining a complaint procedure;
(e) monitoring the effectiveness of their due diligence policy and measures;
(f) publicly communicating on due diligence"

Companies will need to investigate themselves and their entire supply chain. When uncovering 'adverse impacts,' they must create corrective action plans with clear timelines and qualitative as well as quantitative metrics for improvement. Workers, unions and civil society organizations must be able to make human

rights violations or negative environmental impacts known. Companies have to provide a complaint procedure to support this, and inform relevant parties, as well as provide appropriate followup. Companies need to check on the implementation and effectiveness of their policies and on those of their business partners continuously. Such check-ins must be done at least once a year, or when new issues arise in the supply chain. All of this must be done transparently and communicated publicly.

Sanctions are mentioned in the directive, but it is left to every EU country to enforce them in line with national laws and proportional to the company's revenue. In the EU, they will set up a European Network of Supervisory Authorities to help and oversee countries do this.

A director of such a big company has a duty to oversee the corporate due diligence, to establish a code of conduct and to integrate this into the corporate strategy. The CEO must ensure that sustainability, human rights, climate change, and environmental consequences are properly considered in the short, medium, and long run. Making a profit is not a valid counterargument.

Adjust to Size

Big companies will be liable if they fail in their due diligence and preventable bad things happen or continue to happen. This does not mean that companies have to guarantee that bad things will never happen. Companies must take measures appropriate to their level of power in the chain and respond to human and environmental rights violations when they find them (and they are obligated to look).

The CSDD protects human and environmental rights throughout the lifecycle of a product and throughout the entire chain. This should mean that big digital companies are going to have to include the manufacturing of their hardware as part of their responsibility.

Small and medium-sized enterprises (SMEs), which make up 99% of all companies in the EU, are not included in the CSDDD. However, big companies will have demands of any SME they work with in order to be able to fulfil their own due diligence duties. EU countries are expected to support their SMEs by setting up dedicated websites, portals, and platforms — and perhaps provide financial support to build capacity. Big companies are expected to invest in the SMEs with whom they work, in order to help them to comply.

More Moral

I think this directive is a good example of how law slowly becomes a formal version of the values we hold as a society. Of course, you don't have to wait around for laws to be established, and it makes good business and moral sense to be ahead of the curve. One example of this is RS Group, providers of industrial and electronics products to engineers. They have already started to build an ethical supply chain seriously [4].

Many SMEs will soon discover what power (if any) they have in their supply chain, and, perhaps for the first time, undertake an ethical examination of their companies' actions. This forced self-reflection will likely come with compulsory forms to fill, new reporting practices and self-certifying quality control marks, along with big companies usurping more parts of the supply chain in order to maintain control and fulfil their legal responsibilities. If you are part of an SME that works with bigger companies, yesterday would be a good time to start investigating and reporting on your due diligence. Many will already have made many moral business choices. Now, these need just be formalized and communicated.

230428-01

Source: Midjourney AI-generated image. Prompt: "/imagine prompt: EU Parliament with EU flags --ar 5:3"

WEBLINKS

[1] EU CSDDD portal: https://bit.ly/43MAiUu
[2] Supply Chain Act (Wikipedia): https://en.wikipedia.org/wiki/Supply_Chain_Act
[3] Actual CSDDD: https://bit.ly/43Pg1Oa
[4] WEEF 2022 - Building an Ethical and Sustainable Supply Chain with Andrea Barrett: https://youtu.be/tuu88ePQwYQ

Aller Anfang...

Ist gar nicht schwer: Spannungsverstärkung

Von Eric Bogers (Elektor)

Bis jetzt haben wir den Transistor als Schalter und als Stromverstärker benutzt. Beides ist natürlich sehr wichtig in der Elektronik, aber richtig Spaß macht es erst, wenn wir Spannungen verstärken können. Und genau das werden wir in dieser Folge tun.

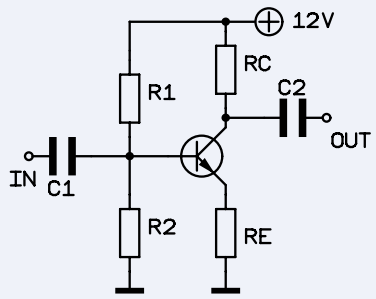


Bild 1. Die Emitterschaltung.

Bevor wir uns in die Emitterschaltung stürzen, ist ein Wort der Warnung angebracht. Es geht um den Artikel in der Ausgabe Mai/Juni 2023 und insbesondere um den astabilen Multivibrator, der in Bild 3 auf Seite 40 dargestellt ist. Wie uns Elektor-Leser Ruedi Schwarzenbach aus der Schweiz schrieb, birgt diese Schaltung einige potentielle Gefahren, die dem Elektronik-Neuling den Spaß ganz schön verderben könnten.

Darum geht es: Obwohl wir im Schaltplan und im Text keinen bestimmten Transistortyp erwähnt haben, wäre es eigentlich naheliegend, einen Allzwecktransistor wie den BC547 für diesen Zweck zu verwenden. Und da die Berechnungen im Text von einer Versorgungsspannung von 12 V ausgehen, könnte dabei etwas schief gehen. Denn laut Datenblatt verträgt der BC547 keine hohe Basis-Emitter-Sperrspannung, 5...6 V sind so ziemlich das Maximum. Das bedeutet, dass die Transistoren in der oben erwähnten Schaltung in Bild 3 bereits bei einer Versorgungsspannung von mehr als 5 V beschädigt werden können.

Obwohl wir erwähnt haben, dass „die Basis des linken Transistors also ein negatives Potential führt und der Transistor sperrt", sollte man bedenken, dass bei einem negativen Potential von 5...6 V der Basis-Emitter-Pfad eine Art Zener-Effekt aufweist. Es fließt ein Strom (der Kondensator wird über die beiden Transistoren kurzgeschlossen), der möglicherweise zum Ausfall des Transistors führen kann. Um diese Art von Problemen zu vermeiden, ist es besser, eine Spannungsquelle von 4,5 V für die Stromversorgung zu verwenden, zum Beispiel eine 4,5-V-Flachbatterie, und alle Berechnungen mit dieser Spannung durchzuführen.

Die Emitterschaltung

Bei der Schaltung in **Bild 1** handelt es sich um eine Emitterschaltung (nicht zu verwechseln mit einem Emitterfolger!), bei der der Emitter die gemeinsame Referenz für das Eingangs- und das Ausgangssignal darstellt. Die Emitterschaltung ist der Kollektorschaltung aus dem vorherigen Teil sehr ähnlich, nur dass jetzt ein Kollektorwiderstand vorhanden ist und die Ausgangsspannung vom Kollektor abgenommen wird. Die Hauptaufgabe des Emitterwiderstands besteht nur darin, ein Abdriften des Arbeitspunktes zu verhindern; sein Wert ist nun viel kleiner. Geht man erneut von einem Ruhestrom von 1 mA aus, so ergibt sich für den Emitterwiderstand ein Wert von 1 kΩ.

Die Ausgangsspannung kann im Bereich von 1...12 V variieren, also um 11 V. Wir wählen für die Leerlaufspannung die goldene Mitte, so dass am Kollektor eine Spannung von 6,5 V anliegt, während am Kollektorwiderstand eine Spannung von 5,5 V abfällt. Wir können nun RC berechnen:

$$R_C = \frac{U}{I} = \frac{5,5\ \text{V}}{1\ \text{mA}} = 5,5\ \text{k}\Omega$$

Wir nehmen hier einen Standardwert von 5,6 kΩ für RC. Für R2 wählen wir (willkürlich) einen Wert von 100 kΩ und berechnen R1 wie folgt:

$$R1 = \frac{U_0 \cdot R2}{U_B} - R2 = \frac{12\ \text{V} \cdot 100\ \text{k}\Omega}{1,7\ \text{V}} - 100\ \text{k}\Omega = 605,9\ \text{k}\Omega$$

Wir verwenden für R1 einen Standardwert von 560 kΩ.
Zur Verdeutlichung: Die 1,7 V ergeben sich aus der Spannung am Emitterwiderstand von 1 kΩ bei einem Ruhestrom von 1 mA plus der Spannung von 0,7 V, die am Basis-Emitter-Übergang anliegt.
Die Frage ist nun natürlich, woher die Spannungsverstärkung kommt und wie groß sie ist. Wenn wir davon ausgehen, dass, wenn sich durch eine angelegte Wechselspannung die Spannung an der Basis des Transistors um 0,1 V erhöht, auch die Emitterspannung um 0,1 V erhöht, was bedeutet, dass der Strom um 100 µA ansteigt. Infolge dieses größeren Stroms fällt eine höhere Spannung über dem Kollektorwiderstand ab (0,56 V mehr, um genau zu sein). Mit anderen Worten: Die Spannung am Kollektor des Transistors ist jetzt 0,56 V niedriger.
Eine Spannungs*erhöhung* an der Basis führt zu einer entsprechenden Spannungs*verringerung* am Kollektor, und nicht nur das - die Spannungsänderung am Kollektor ist größer als die an der Basis. Die Spannungsänderung an der Basis wird also verstärkt!
Auch in diesem Zahlenbeispiel haben wir die Differenz zwischen I_C und I_E vernachlässigt. Wenn wir uns diese bescheidene Freiheit erlauben, können wir für die Spannungsverstärkung der Emitterschaltung folgendes schreiben:

$$V_{OUT} = \frac{\Delta U_C}{\Delta U_B} \approx -\frac{R_C}{R_E}$$

In diesem Beispiel beträgt die Verstärkung also -5,6×. Das Minuszeichen bedeutet einfach, dass Eingangs- und Ausgangssignal gegenphasig sind. Oder anders ausgedrückt, wir haben es hier mit einem invertierenden Verstärker zu tun.
Mit dieser Schaltung kann ein Signal viel stärker verstärkt werden, aber in diesem Fall sollte ein kleinerer Emitterwiderstand und/oder eine höhere Versorgungsspannung gewählt werden. Ein „Design“ wie in **Bild 2** sollte unter allen Umständen vermieden werden. In dieser Schaltung hat der „Konstrukteur“ der Bequemlichkeit halber den Widerstand R2 und den Emitterwiderstand RE weggelassen. Wegen des weggelassenen Emitterwiderstandes liefert die Schaltung zwar eine sehr große Verstärkung, aber die Temperaturstabilität

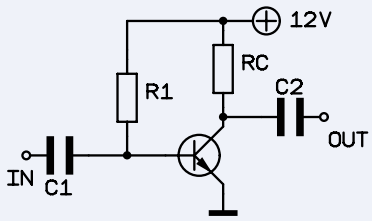


Bild 2. Eine katastrophale Verstärkerschaltung - so sollte man es nicht machen!

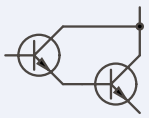


Bild 3. Die Darlington-Schaltung.

der Gesamtschaltung ist miserabel: Ändert sich die Temperatur nur um wenige Grad, steigt (oder sinkt) der Basisstrom - und damit auch der Kollektorstrom - und der Arbeitspunkt driftet. Außerdem ist die Stromverstärkung eines Transistors keine verlässlich fixe Größe, da sie eine sehr starke exemplarische Streuung aufweist - der Wert von R1 könnte nicht berechnet, sondern nur experimentell ermittelt werden.
Durch den weggelassenen Emitterwiderstand ergibt sich zudem eine sehr niedrige Eingangsimpedanz, so dass der Wert von C1 deutlich größer gewählt werden muss, damit die untere Grenzfrequenz nicht zu hoch wird.
Um es kurz zu machen: Eine Schaltung wie in Bild 2 ist nicht sinnvoll. Wer eine sehr große Verstärkung benötigt, sollte deshalb besser mehrere Verstärkerstufen in Reihe schalten (oder einen Operationsverstärker vorschalten).

Die Darlington-Schaltung

Für viele Anwendungen wird eine größere Stromverstärkung benötigt, als sie ein einzelner Transistor liefern kann. In solchen Fällen bietet sich die Darlington-Schaltung (**Bild 3**) an. Bei dieser Schaltung werden zwei Transistoren so geschaltet, dass der Emitter des ersten Transistors direkt mit der Basis des zweiten Transistors verbunden ist. Hier kann man (in guter Näherung) die einzelnen Stromverstärkungsfaktoren miteinander multiplizieren, so dass mit Kleinsignaltransistoren Werte von 10.000 und mehr erreicht werden können. Dabei ist zu beachten, dass sich die Basis-/Emitter-Spannungen der Transistoren addieren, so dass eine Darlington-Schaltung eine Vorspannung (Bias) von mindestens etwa 1,4 V benötigt.
Bei Leistungstransistoren mit eher bescheidener Stromverstärkung sollte die Darlington-Schaltung verwendet werden. In vielen Fällen sind die beiden Transistoren sogar bereits vom Hersteller in einem Gehäuse integriert – und dann spricht man von einem Darlington-Transistor.

Die Konstantstromquelle

Wenn wir eine LED verwenden, dann haben wir in den meisten Fällen bereits eine konstante Versorgungsspannung zur Verfügung, und es ist einfach, den erforderlichen Serienwiderstand entsprechend zu berechnen. Außerdem ist es kaum ein Problem, wenn die Versorgungsspannung (in gewissen Grenzen) schwankt: Es ist kein großer Unterschied, ob der LED-Strom 5 mA oder 20 mA beträgt (obwohl aus wirtschaftlichen Gründen ein niedriger Strom immer besser ist). Auch eine Versorgungsspannung, die sogar um den Faktor vier schwankt, ist noch mehr oder weniger akzeptabel; weist die Versorgungsspannung jedoch noch größere Schwankungen auf, empfiehlt es sich, eine Konstantstromquelle zu verwenden.

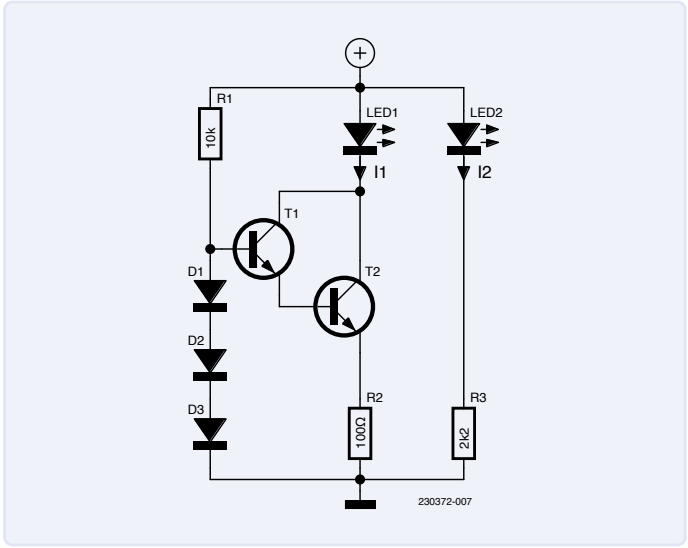


Bild 4. Eine (mehr oder weniger) konstante Stromquelle.

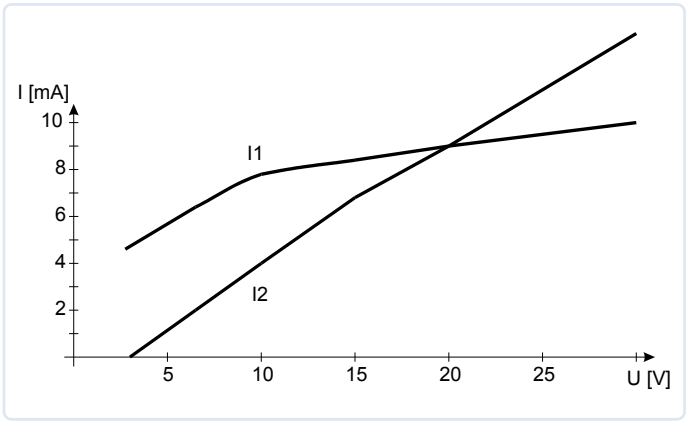


Bild 5. Stromfluss mit (I1) und ohne (I2) Konstantstromquelle.

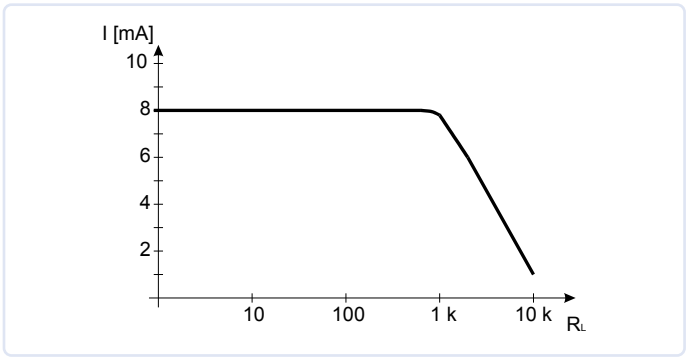


Bild 6. Eine Konstantstromquelle mit variierendem Lastwiderstand.

Die Bezeichnung „Konstantstromquelle“ für die Schaltung in **Bild 4** ist eigentlich etwas optimistisch, wie aus dem Diagramm in **Bild 5** hervorgeht. Andererseits ermöglicht diese Schaltung den Einsatz einer LED unter Bedingungen, bei denen die Versorgungsspannung um den Faktor 40 schwanken kann, und das ist auch gut so. Drei in Reihe geschaltete Dioden liefern eine mehr oder weniger konstante Spannung von etwa 2 V. Mit dieser Spannung wird ein Darlington-Transistor angesteuert, der einen Emitterwiderstand von 100 Ω hat. Die Spannung über diesem Widerstand beträgt etwa 0,7 V, was zu einem Strom von etwa 0,7 mA führt (was durchaus prasixnah ist). Nebenbei bemerkt: Einen Darlington-Transistor würde man in diesem Beispiel nicht wirklich benötigen. Wir haben den Strom durch die LED gemessen und in Bild 5 aufgetragen, und zum Vergleich den Strom durch eine LED nur mit Vorwiderstand. Bei der Konstantstromquelle steigt der Strom deutlich weniger steil an (die Kurve ist flacher), was bedeutet, dass die Schaltung auch bei deutlich höheren Versorgungsspannungen noch verwendbar wäre. Die Tatsache, dass der Strom nicht vollkommen konstant ist, also die Kurve nicht horizontal verläuft, ist auf den Spannungsanstieg über den Dioden zurückzuführen - eine Z-Diode würde ein wesentlich besseres Ergebnis liefern, und eine „zweistufige“ Stabilisierung wäre noch besser gewesen.

Eine Konstantstromquelle schneidet übrigens wesentlich besser ab, wenn die Versorgungsspannung konstant ist (12 V in **Bild 6**) und der Lastwiderstand (mit der LED) variiert.

Bei höheren Lastwiderständen sinkt der Strom, da sonst das Produkt aus Strom und Widerstand größer als die Versorgungsspannung wäre - was unmöglich ist. Solange jedoch die Spannung am Lastwiderstand deutlich unter der Versorgungsspannung liegt, ist der Strom durch die Last ziemlich konstant.

Natürlich werden Sie sich fragen, warum wir einen konstanten Strom bei variierendem Lastwiderstand benötigen. Nun, ohne eine solche Stromquelle wäre ein Differenzverstärker nicht möglich - und der Differenzverstärker - vielleicht die wichtigste Schaltung in der Elektronik - wird im nächsten Teil dieser Serie behandelt.

RG — 230372-02

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 an die Elektor-Redaktion unter redaktion@elektor.de.

Passende Produkte

- **B. Kainka, *Elektronik-Grundlagen und Einsteiger-Projekte* (Elektor, 2020)**
  Buch (kartoniert): https://elektor.de/19035
  E-Buch: https://elektor.de/19036

Infraschall-Rekorder mit dem Arduino Pro Mini

Ein Beispielprojekt aus dem Elektor-Buch „Arduino & Co.“

Von Robert Sontheimer (Deutschland)

Lasst uns etwas Verrücktes machen! Kombinieren wir einen Arduino Pro Mini mit einem Luftdrucksensor BMP180 und einem SD-Kartenmodul und erforschen wir, wie man Langzeitaufzeichnung von Infraschallsignalen erstellen kann.

Anmerkung der Redaktion: *Dieser Artikel ist ein Auszug aus dem 332-seitigen, frisch überarbeiteten Elektor-Buch „Arduino & Co - Messen, Schalten und Tüfteln“. Der Auszug wurde formatiert und leicht bearbeitet, um den Konventionen und dem Seitenlayout der Zeitschrift Elektor zu entsprechen. Da es sich um einen Auszug aus einer größeren Publikation handelt, kann dieser Artikel auf Diskussionen an anderer Stelle im Buch verweisen. Der Autor und die Redaktion haben ihr Bestes getan, um solche Fälle auszuschließen, und sind gerne bereit, bei Fragen zu helfen. Kontaktinformationen finden Sie im Kasten* **Fragen oder Kommentare?**

Der Luftdruck-Sensor liefert uns maximal etwa 100 Temperatur- und Luftdruck-Werte pro Sekunde. Da könnten wir doch aus diesen Daten über Minuten, Stunden oder Tage hinweg eine Stereo-Audiodatei (im PCM-WAVE-Format mit der Dateiendung „.wav“) generieren, die auf dem linken Kanal den Luftdruck aufzeichnet, und auf dem rechten Kanal die Temperatur. Die reguläre Abspiel-Frequenz (Samplingrate) setzen wir zum Beispiel auf 44100 Hz. Das ist ein gängiger Wert, der auch für Audio-CDs und vieles andere verwendet wird.

So wird die Datei später (z.B. am Laptop oder mit einem Audio-Player) in 441-facher Geschwindigkeit abgespielt, und wir können Infraschall hören. Mit der ursprünglichen Abtastrate von 100 Hz bei der Aufnahme können wir alle Frequenzen unterhalb 50 Hz aufzeichnen. Aus einem Infraschall-Ton mit 10 Hz werden beim Abspielen 4,4 kHz. Aus 1 Hz wird 441 Hz usw. Nach unten gibt es theoretisch keine Grenzen.

OK, zugegeben – als hochempfindliches Mess-Mikrofon taugt der BMP180 nicht wirklich. Obwohl er schon Druckschwankungen bei geringen Höhenunterschieden erkennt, muss Infraschall doch recht laut sein, um ihn später aus dem Rauschen rauszuhören.

Es ist aber auch viel interessanter, sich mit einem Audio-Bearbeitungsprogramm später die Luftdruck- und Temperatur-Messkurven anzuschauen – insbesondere, wenn die Aufnahme über mehrere Tage lief. Der linke Kanal zeigt uns dann den Luftdruck-Verlauf, der rechte Kanal den Temperatur-Verlauf.

Hier habe ich einfach mal über knapp 3 Tage hinweg aufgezeichnet, die fertige WAV-Datei mit einem Audio-Bearbeitungsprogramm geöffnet, und dann beide Kanäle getrennt normalisiert – also soweit verstärkt, dass der volle Bereich genutzt wird (**Bild 1**). Die obere Kurve mit dem Druckverlauf sieht hier noch etwas dick und ausgefranst aus, was daran liegt, das jeder Pixel der X-Achse aus zigtausenden Messungen besteht, die auch etwas Rauschen enthalten und somit in einem gewissen Bereich schwanken.

Bild 2 zeigt die Ergebnisse nach der Verarbeitung. Hier wurde nun die Aufzeichnung neu gesampelt und dabei jeweils 4000 Werte zu einem einzigen zusammengefasst – ähnlich wie wir ja auch sonst oft aus vielen Messungen den Durchschnitt bilden. Jetzt ist das Rauschen weg, und man erkennt die Verläufe von Druck und Temperatur ganz exakt.

Aus den Kurven können wir nun einiges herauslesen: Am Druckluft-Verlauf in der oberen Kurve sieht man, dass es zeitlich gesehen in der zweiten Hälfte etwas windiger wurde. Die vielen kleinen Ausschläge nehmen dort zu. Bei einem richtigen Sturm wären die Schwankungen noch extremer.

An der unteren Kurve mit dem Temperaturverlauf erkennt man deutlich den Tag-Nacht-Rhythmus. Die fast durchgehend

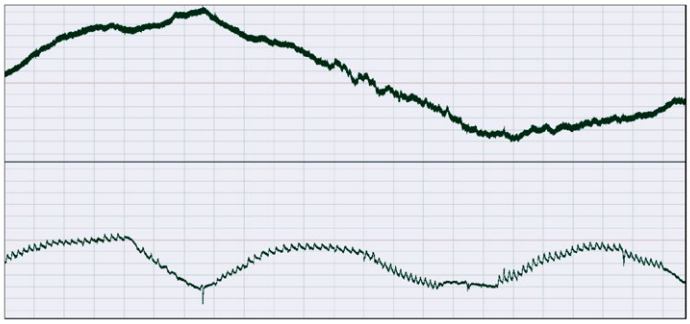


Bild 1. Verlauf des Luftdrucks (oben) und der Temperatur (unten), im Wesentlichen unbearbeitet.

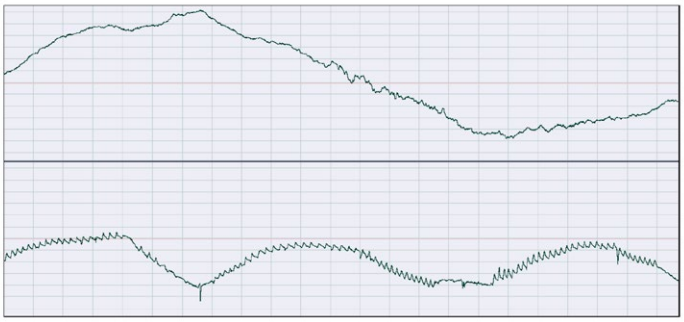


Bild 2. Graphischer Verlauf von Luftdruck (oben) und Temperatur (unten), bearbeitet.

gleichmäßigen kleinen Zacken sind die Schalt-Zyklen vom Thermostat der Heizung. Man könnte abzählen, wie oft der Thermostat in den 3 Tagen ein- und ausgeschaltet hat. Weiter erkennt man noch zwei kurze Ausschläge nach unten. Da war das Fenster kurz geöffnet.

Aufbau

Um das Projekt zu realisieren, benötigen wir:

- 1 Arduino-Board (16 MHz ATmega328)
- 1 BMP180-Sensormodul
- 1 (Micro-)SD-Kartenmodul
- 1 Taster
- Litze oder Jumper-Kabel
- 1 (Micro-)SD-Karte, max. 32 GB

Der Aufbau ist recht einfach. Zusätzlich zum BMP180-Sensor müssen wir das SD-Karten-Modul und den Taster zum Starten und Stoppen der Aufnahme anschließen. Am besten sind immer gelötete Leitungen. Wir können aber auch alle Verbindungen mit Jumper-Kabel stecken. Dann benötigen wir allerdings zwei VCC-Leitungen, aber der „Pro Mini" hat nur einen VCC-Pin.
Hierfür gibt es jedoch eine ganz einfache Lösung. Da der BMP180 kaum Strom verbraucht, können wir ihn auch mit einem Ausgang versorgen (z.B. mit Pin 9 statt VCC). Den Pin müssen wir dann nur im `setup()` als Ausgang definieren und auf HIGH schalten.

Sketch: Infraschall-Recorder

Ein guter Teil des Sketches mit dem Namen 13.9.ino, *der für dieses Projekt geschrieben wurde, wird in Teil-Listings mit den Nummern Listing 1a bis Listing 1h zusammen mit den folgenden Diskussionen vorgestellt. Das vollständige Programm ist viel länger, als es hier mit den Teil-Listings gezeigt werden kann. Das Programm und die Unterprogramme, die im Folgenden erwähnt werden, sind in der Software-Begleitdatei enthalten, die für das Buch veröffentlicht wurde und von der Elektor Books Support-Website [1] kostenlos heruntergeladen werden kann. Gehen Sie auf der Webseite auf die Registerkarte* Downloads. *Um diesem Auszug folgen zu können, halten Sie bitte* 13.9.ino *als Referenz bereit.*

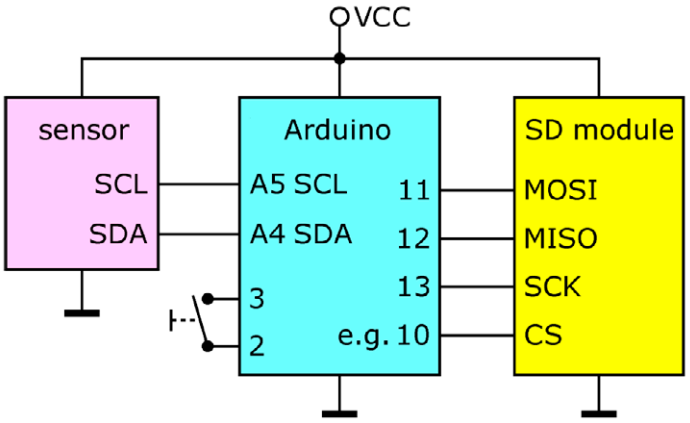


Bild 3. Verdrahtungsplan für BMP180-Sensor, SD-Kartenmodul, Schalter und Arduino.

Listing 1a:
Da wir hier den Luftdruck-Sensor mit dem SD-Karten-Modul kombinieren, müssen wir gleich drei Libraries einbinden, die I²C-Schnittstelle für den Luftdruck-Sensor, die SPI-Schnittstelle fürs SD-Karten-Modul und die speziellen SD-Karten-Funktionen.
Dann werden noch zwei Pins definiert, an denen wir einen Taster anbringen können, mit dem man die Aufnahme startet und stoppt. Pin 3 wird dabei auf LOW geschaltet, und dient uns so quasi als Masse-Pin für den Taster. Das ist besonders praktisch, weil so die beiden Taster-Pins nebeneinander liegen. Natürlich könnten wir statt Pin 3 auch die echte Masse (GND) verwenden.
Die weiteren Definitionen und Variablen, sowie alle Programmteile, die im Buch besprochen wurden, sind hier nicht abgedruckt. Uns interessiert jetzt nur, was speziell den Rekorder betrifft.

Listing 1a

```
#include <Wire.h>     // library for I2C interface
#include <SPI.h>      // library for SPI interface
#include <SD.h>       // library for SD card
#define record_pin 2 // switch pin to ground for
                      //   recording
#define low_pin 3     // Set output pin to LOW
                      //   (used as GND for button)
…
```

Listing 1b: Hier haben wir nun die Rekorder-Einstellungen. Die Variable `oversampling` bestimmt auch hier die Zahl der Druck-Messungen. Um 100 Werte pro Sekunde aufzuzeichnen, muss diese Einstellung auf 0 sein. Die `audio_rate` gibt dagegen an, wie viele Werte pro Sekunde wiedergegeben werden, wenn wir die Aufzeichnung später als Audio-File abspielen. `max_file_size` bestimmt die maximale Aufnahme-Länge. Die 10 Minuten beziehen sich allerdings auf die Abspielzeit. Die entsprechende Aufnahmezeit beträgt mehrere Tage. Wir können diesen Wert aber fast beliebig anpassen. Ist die maximale Dateigröße erreicht, wird die Datei geschlossen und eine neue Aufnahme gestartet.
Die Variablen `duration` bestimmen hier (wie beim Höhenmesser) die Verzögerungen beim Messen. Allerdings sind die Zeiten hier um 500 µs länger, denn wir arbeiten hier nicht mir Delays,

sondern mit einem festen Takt, innerhalb dessen auch die ganzen `Wire`-Anweisungen zum Temperatur- und Druckmessen ablaufen.

Listing 1b

```
// 0 -> 1x, 1 -> 2x, 2 -> 4x, 3->8x
char oversampling = 0;
// sampling rate of the output audio file
long audio_rate = 44100;
// 10 minutes' playback time (at 44,100 Hz)
unsigned long max_file_size = 105840044;
// time (µs) according to oversampling
int duration[4] = ;
// time for temperature measurement
int t_duration = 5000;
…
```

Listing 1c: Nach den speziellen Variablen zur Temperatur- und Druck-Berechnung, die wir hier wieder übersprungen haben, werden diese Variablen später für den Rekorder benötigt.

Listing 1c

```
// system time at which the data are available
unsigned int next_time;

long zero_value;  // pressure zero line
long record_value;// pressure value to save
long zero_temp;   // temperature zero line
long temp_value;  // temperature value to save

// indicates whether recording is activated
boolean must_record = false;
// indicates if recording is currently running
boolean is_recording = false;
// indicates whether pushbutton is pressed
boolean pressed = false;
// actual file size
unsigned long file_size;
// actual file number
unsigned int file_number = 0;
// counts how long button is not pressed anymore
byte state_counter = 0;
File wave_file;   // recording file
```

Listing 1d: Das `setup()` beginnt mit der Definition der Pins für den Taster. Durch den aktivierten internen Pull-up-Widerstand benötigen wir zum Taster keine weiteren Bauteile.

Listing 1d

```
void setup() {
  // switch pin on input with pull-up
  pinMode(record_pin, INPUT_PULLUP);
  // low pin on output ...
  pinMode(low_pin, OUTPUT);
  // ... and LOW [...]
  digitalWrite(low_pin, LOW);
  …
}
```

Listing 1e: Im `loop()` haben wir jetzt nur noch den abwechselnden Aufruf der Temperatur- und Druck-Messung, denn um keine Zeit zu verlieren, warten wir da nicht jeweils mit einem Delay auf die Daten, sondern nutzen die Wartezeit für alle weiteren Aufgaben. Bei der Temperatur-Messung wird die Funktion `calculate()` ausgeführt, während wir aufs Ergebnis warten. Bei der Druck-Messung führen wir derweil die Funktion `recording()` aus.

Listing 1e

```
void loop() {
  get_t(); // read temperature
  get_p(); // read pressure
}
```

Listing 1f: Das ist nun die `calculate()`-Funktion. Da werden zunächst aus den Messwerten die tatsächliche Temperatur und der tatsächliche Druck ermittelt, sofern der Druck schon einmal gemessen wurde. Das habe ich jetzt wieder mit „…“ abgekürzt, denn diese Berechnungen (mit den vielen Formel-Zeilen aus dem Datenblatt des BMP180) will gewiss keiner nochmal lesen.
Dann wird überprüft, ob die Null-Linien für Druck und Temperatur schon definiert sind. Ist das nicht der Fall, wird die erste Messung jetzt als Nullwert festgelegt. Wurde hingegen noch nicht einmal gemessen, verlassen wir die Funktion komplett.
Ansonsten werden die Werte jetzt zur Aufzeichnung vorbereitet. Hierzu ziehen wir den Druck-Nullwert vom Druck ab. Ist der Wert übersteuert, wird er begrenzt auf den zulässigen Bereich von -32768 bis 32767. Genau dasselbe machen wir anschließend auch mit der Temperatur. Somit stehen nun beide Werte für die Aufzeichnung bereit.

Listing 1f

```
// calculate temperature (in tenth degree) and
// pressure (in pascals)
void calculate() {

  // if pressure has already been measured before
  if (p) {
    …
  }

  // if zero value has not been determined yet
  if (!zero_value) {
    // cancel if no measurement has been taken yet
    if (!pressure) return;
    // actual value as zero line
    zero_value = pressure;
    // actual value as zero line
    zero_temp = b5;
  }
```

```
  // pressure value for recording
  record_value = pressure - zero_value;

  // max value if clipping
  if (record_value > 32767)
    record_value = 32767;
  // negative clipping
  else if (record_value < -32768)
    record_value = -32768;
  // temperature value for recording
  temp_value = b5 - zero_temp;
  // max value if clipping
  if (temp_value > 32767) temp_value = 32767;
  // negative clipping
  else if (temp_value < -32768) temp_
  value = -32768;
}
```

Listing 1g: In der Funktion `recording()` werden zunächst zwei Bedingungen überprüft, zum einen, ob die Aufnahme laufen soll (weil sie eingeschaltet ist und auch die maximale Dateigröße noch nicht erreicht hat), und ob die Aufnahme tatsächlich läuft. Durch diese beiden Unterscheidungen ergeben sich nun 4 Möglichkeiten. Im ersten Fall soll die Aufnahme laufen, aber sie läuft momentan noch nicht.

Listing 1g

```
void recording()  { // recording function
  // if recording should run
  if (must_record && file_size < max_file_size) {
    // if recording is not running yet
    if (!is_recording) {
      …
    }
  }
}
```

Listung 1h: Hier wird eine neue Aufnahme gestartet. Dazu wird in der `Do-While`-Schleife die File-Nummer solange erhöht, bis diese (zusammen mit dem weiteren Text) einen Dateinamen ergibt, der noch nicht existiert. Mit dem entsprechenden Dateinamen wird dann eine neue Datei geöffnet, und auch seriell eine entsprechende Meldung ausgegeben. Im Fehlerfall wird eine weitere Meldung ausgegeben und der Ablauf mittels Endlosschleife gestoppt.

Listing 1h

```
// search for the first not yet used
// recording number
do {
  file_number++; // next number (starting with 1)
} // repeat with next number while
   //file already exists
while (SD.exists("FILE_" + String(file_number)
       + ".wav"));
 Serial.println("Recording no."
                + String(file_number)
                + " is started!");
 wave_file = SD.open("FILE_"
                     + String(file_number)
                     + ".wav", FILE_WRITE);

if (!wave_file) { // if file could not be opened
   Serial.println
      ("Error: File could not be opened!");
// do not continue (infinite loop)
    while (true);
}
```

Damit ist der Platz für die Besprechung des Programms in diesem Artikel der Zeitschrift erschöpft. Im Buch ist der Rest des Programms zum Betrieb des Infraschallrekorders glücklicherweise ebenso gut dokumentiert wie die hier gezeigten Listings.

Verwendung

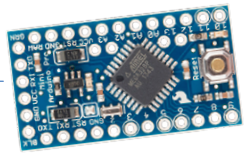
Am Anfang des Sketches können wir drei Variablen festlegen.

- `oversampling` sollten wir auf 0 setzen, um die schnellste Aufnahme-Frequenz von 100 Hz zu verwenden.
- `audio_rate` gibt die spätere Abspiel-Frequenz an. Da können wir auch einen geringeren Wert als 44100 einsetzen, damit der Zeitfaktor (44100 / 100 = 441) nicht ganz so hoch ist. Gängige Werte sind 8000, 11025, 16000, 22050, 32000, 44100 und 48000. Andere Frequenzen kann nicht jeder Player abspielen.
- `max_file_size` legt die maximale Dateigröße fest. 105840044 entspricht dabei einer Abspieldauer von zehn Minuten bei 44100 Hz, aber einer Aufnahmezeit von über drei Tagen.

Die fertige Aufnahme sollten wir dann „normalisieren", das heißt die Lautstärke des Signals soweit erhöhen, dass der volle Bereich genutzt wird, aber ohne zu übersteuern. Sonst ist das Signal zu leise. Natürlich könnten wir da auch schon beim Aufnehmen einen Verstärkungsfaktor einbauen, aber wir wissen ja vorher nicht, wie weit sich Temperatur und Luftdruck während der Aufnahme ändern. Die Normalisierung kann man mit einem Audio-Bearbeitungsprogramm wie Audacity mit wenigen Klicks vornehmen. Sie sollte für beide Kanäle getrennt erfolgen. Mit so einem Programm können wir uns dann auch die Aufzeichnung als Messkurven ansehen. Der linke Kanal zeigt den Verlauf des Luftdrucks während der Aufnahme. Der rechte Kanal zeigt den Temperaturverlauf.

Wetter-Rekorder

Wenn wir den `oversampling`-Wert auf 3 ändern, werden nur noch gut 32 (statt 100) Werte pro Sekunde aufgenommen, was für Wetter-Aufzeichnungen immer noch weit mehr als genug ist. Wir könnten die SD-Karten-Funktionen daher auch in den vorherigen Höhenmesser (im Buch Kapitel 13.8) einbauen – damit eine



Textdatei öffnen und bei jeder seriellen Ausgabe auch eine entsprechende Textzeile in die Datei schreiben. So bekommen wir die Wetter-Aufzeichnung als Textdatei.
Wir sehen also: Es gibt unzählige Möglichkeiten, was man so alles machen kann, und die einzelnen Kapitel bieten ja – neben den konkreten Projekten – auch zahlreiche Infos und Anregungen, die dabei helfen sollen, möglichst viele eigene Ideen umzusetzen. Leider sind wir hier schon am Ende des Artikels angelangt, und ich wünsche allen Lesern weiterhin viel Spaß beim Tüfteln, Basteln und Programmieren!

RG — 230393-02

Haben Sie Fragen oder Kommentare?

Haben Sie Fragen oder Kommentare zu diesem Artikel? Senden Sie eine E-Mail an den Autor unter r.sont@freenet.de oder an Elektor unter redaktion@elektor.de.

Über den Autor

Robert Sontheimer war sofort dabei, als vor rund 40 Jahren mit dem ZX81 und dem C64 die ersten Heimcomputer in unseren Wohnzimmern Einzug hielten. Damals baute er neben anderen skurrilen und originellen Ideen einen Plotter in einen Scanner um. Heute steuert er mit Arduino Pro Minis ganze CNC-Lasermaschinen und hat sogar ein passendes Absaugsystem erfunden: seinen „selbstwechselnden Toilettenpapierfilter“. In seinem Büro hat er einen Magneten, der schon seit Jahren schwebt - natürlich gesteuert von einem Pro Mini.

Passende Produkte

- **Robert Sontheimer, *Arduino & Co - Messen, Schalten und Tüfteln Überarbeitete Neuauflage*, Elektor 2022**
  Buch, kartoniert: https://elektor.de/19975
  E-Buch, PDF: https://elektor.de/19976

WEBLINK

[1] Software-Archiv zum Buch: https://elektor.de/19975

Quelle: Shutterstock

Cloud-basierter Energiezähler

Mit ESP32-Modul und PZEM-004T-Spannungs-/ Stromsensor

Von unserer Partnerzeitschrift Elettronica In
https://elettronicain.it

Von Emanuele Signoretta (Italien)

Mit den ständig steigenden Strompreisen sind eine weise Nutzung und Einsparung zur Notwendigkeit geworden. Mit einem ESP32-Modul und ein paar anderen Hardware-Komponenten ist es möglich, einen Energiezähler zu erstellen, der unsere Energiedaten über eine Verbindung im WLAN an die Cloud-Plattform InfluxDb sendet.

Italien ist, wie viele andere Länder auch, in hohem Maße von ausländischen Energiequellen abhängig, da die eigene Produktion erneuerbarer Energien nicht ausreicht, um den nationalen Bedarf zu decken. Dies wird zu einem größeren Problem, wenn externe Faktoren die Verfügbarkeit von fossilen Brennstoffen einschränken, was wiederum zu höheren Preisen führt. Infolgedessen müssen die Menschen in Italien, genau wie wir, ihren Energieverbrauch einschränken, auf bestimmte Annehmlichkeiten verzichten und ihren Energieverbrauch genau überwachen, um unangenehme Überraschungen bei ihren Strom- und Gasrechnungen zu vermeiden. Der Hauptgrund für die Reduzierung des Energieverbrauchs ist die Angst vor zu hohen Ausgaben und nicht die Sorge um den Klimawandel.
Um denjenigen zu helfen, die sich ihres Stromverbrauchs bewusst sind, wird in diesem Artikel ein Energieverbrauchsmesser vorgestellt, der ein ESP32-Modul von Espressif und einen Spannungs-/Stromsensor PZEM-004T verwendet. Der Zähler sammelt Daten und sendet sie an die Cloud, insbesondere an die Server des Online-Dienstes InfluxDB.

Die Hardware

Für die Hardware-Komponente des Energiezählers benötigen wir ein WLAN-Modul ESP32 (**Bild 1**) und einen Sensor PZEM-004T (in **Bild 2** in seinem transparenten Kunststoffgehäuse eingekapselt zu sehen). Außerdem benötigen wir ein Schaltnetzteil mit einem 5-V-Ausgang (wenn Sie ein Netzteil mit einem Micro-USB-Ausgang wählen, können Sie es für die Stromversorgung der ESP-Platine verwenden

Bild 1. Die ESP32-Platine.

Bild 2. Der Sensor PZEM-004T mit Transformator.

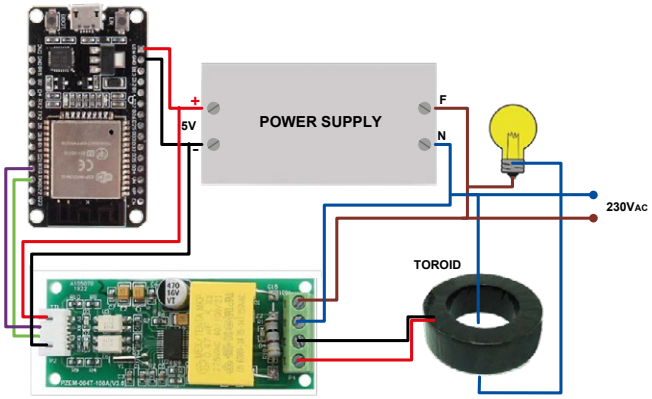


Bild 3. Verdrahtungsplan der Schaltung.

und den PZME-004T über V_{IN} mit Strom versorgen), Jumper weiblich auf weiblich für den Arduino, Klemmen, einige Elektrokabel und eine Kunststoffbox [2]. Das Datenblatt des Sensors finden Sie unter [3], während **Bild 3** die Verdrahtung der gesamten Schaltung zeigt – wir werden unten im Abschnitt **Einbau** näher darauf eingehen.

Das Modul besteht aus einer Schaltung zur Messung von Strom und Spannung, die in einem Kunststoffgehäuse mit einem offenen Ringkern untergebracht ist. Es enthält eine minimale Elektronik, die durch Optokoppler isoliert ist, und verfügt über eine serielle Schnittstelle. Das Modul kann Spannungen im Bereich von 80 V bis 260 V mit einer Auflösung von 0,1 V und einer Genauigkeit von 0,5 % messen. Ebenso können Ströme von 0 A bis 100 A mit einer Genauigkeit von 0,5 % und einer Auflösung von 0,001 A gemessen werden. Darüber hinaus kann der Sensor den Phasenwinkel (Leistungsfaktor) zwischen Spannungs- und Stromvektoren mit einer Genauigkeit von 1 % bestimmen und liefert Messwerte für die Wirk- und Blindleistung. Diese Fähigkeit ermöglicht es uns, den elektrischen Wirkungsgrad des zu messenden Geräts zu bewerten. Die Auswahl der Schaltungskomponenten beruhte in Anbetracht der aktuellen Halbleiterkrise auf Kosteneffizienz und den beeindruckenden Spezifikationen des ESP32.

InfluxDB-Konfiguration

Für den Empfang und die Analyse von Daten, die von IoT-Geräten übertragen werden, sind mehrere Softwarepakete verfügbar. Zu den namhaften Optionen zählen Home Assistant, Grafana, Blynk, ThingSpeak und andere. In diesem Zusammenhang werden wir den Fokus auf InfluxDB legen, dessen Logo in **Bild 4** dargestellt ist.

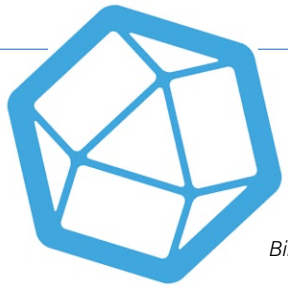


Bild 4. InfluxDB-Logo.

InfluxDB ist eine vielseitige Software, die Funktionen wie das Erstellen von Dashboards, das Ausführen von Abfragen und das Senden von Warnmeldungen ermöglicht. Sie bietet mehrere Anwendungsmöglichkeiten, darunter ausführbare Versionen für verschiedene Architekturen, Docker-Container und cloudbasierte Lösungen. Obwohl wir

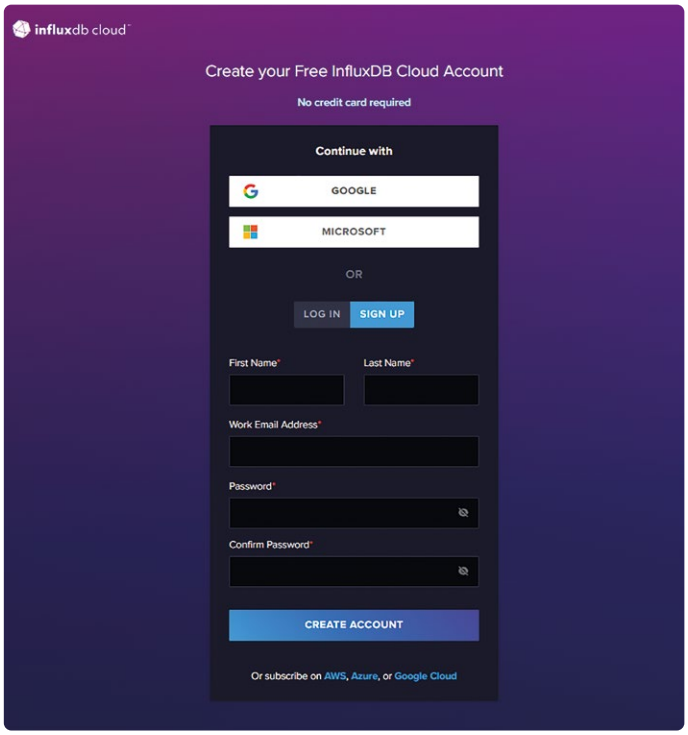


Bild 5. InfluxDB-Registrierungsseite.

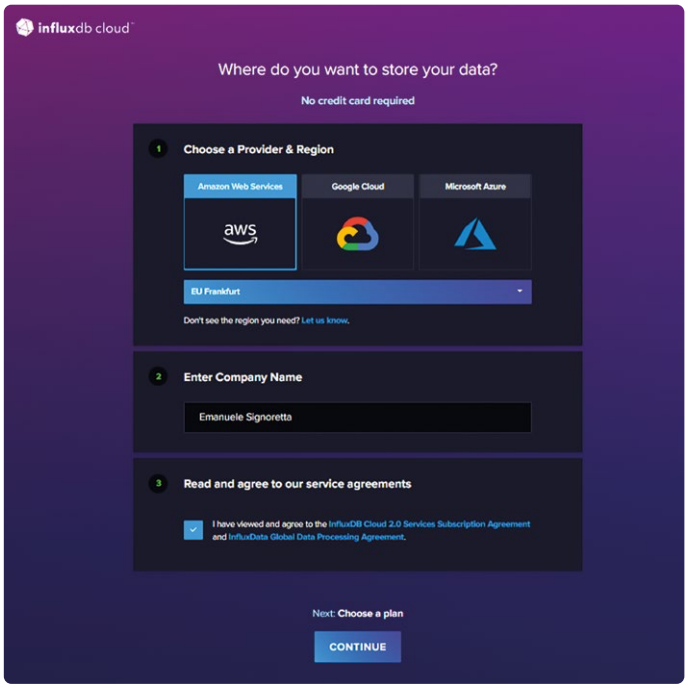


Bild 6. Auswahl des Anbieters und Akzeptanz der Nutzungsbedingungen.

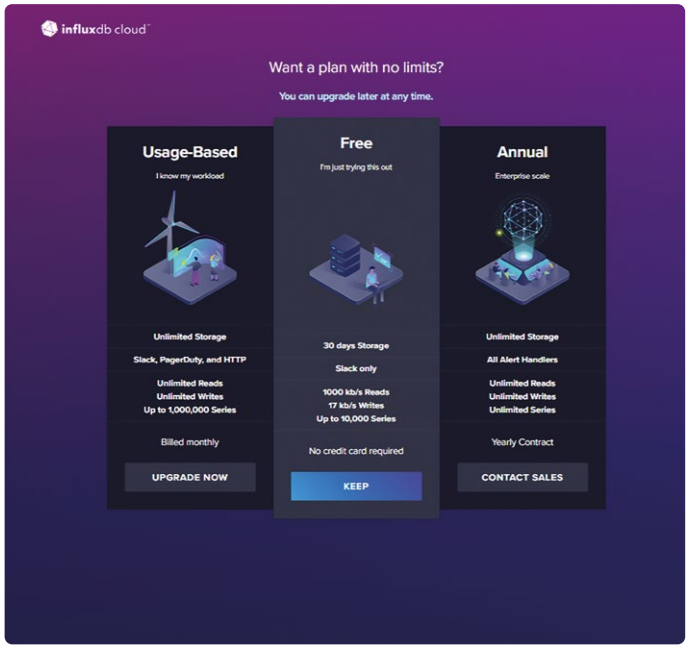


Bild 7. Auswahl des Abonnementplans.

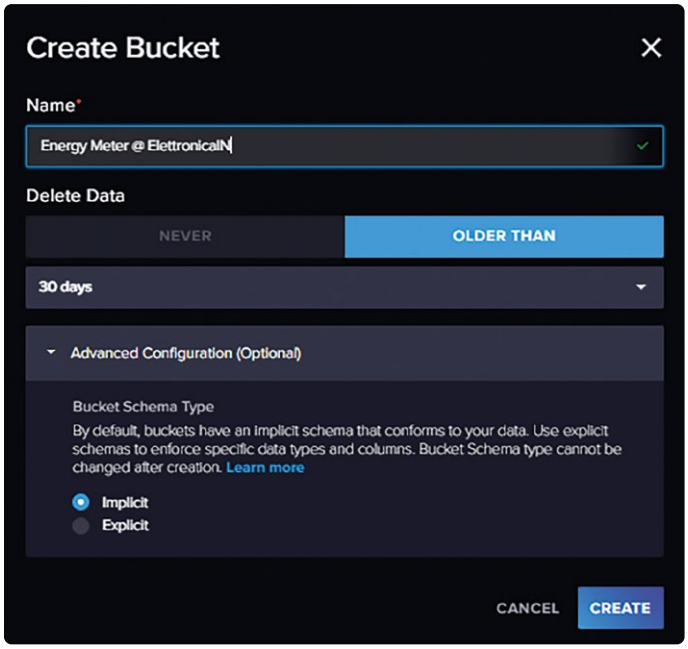


Bild 8. Erstellen eines neuen Buckets.

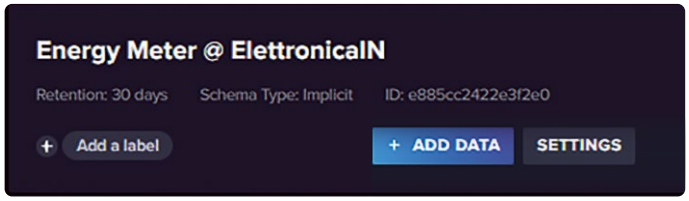


Bild 9. Ein neu erstellter Bucket.

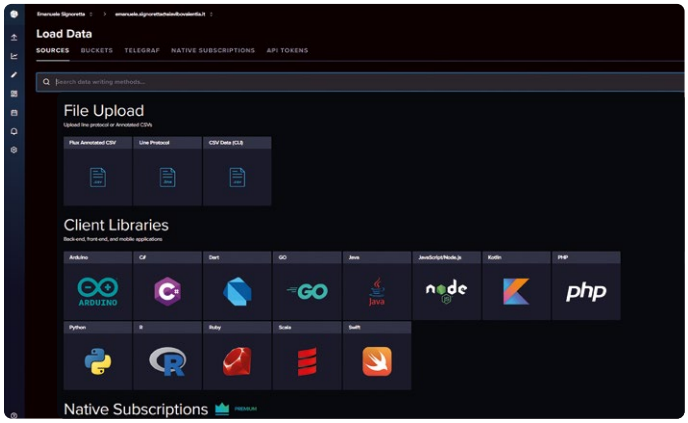


Bild 10. Auswahl der Daten-Upload-Quelle.

Bild 11. Ausschnitt aus einem Demo-Sketch.

ursprünglich vorhatten, aus Datenschutzgründen eine Instanz auf einem Raspberry Pi zu installieren, sahen wir uns aufgrund der Pi-Knappheit gezwungen, stattdessen den Dienst InfluxDb Cloud v2 zu nutzen. Um loszulegen, besuchen Sie bitte den Link [4] und registrieren Sie sich. Sie können sich über Ihr Microsoft- oder Google-Konto anmelden oder die erforderlichen Felder manuell ausfüllen, wie in **Bild 5** gezeigt. Nach Abschluss des Registrierungsvorgangs wird ein Bildschirm ähnlich wie **Bild 6** gezeigt, in dem Sie aufgefordert werden, verschiedene Details anzugeben, darunter den Namen des Dienstanbieters. Für unser Projekt wählten wir Amazon Web Services (AWS) als Anbieter. Auf dem folgenden Bildschirm (siehe **Bild** 7) müssen Sie einen Nutzungsplan auswählen. In unserem Fall haben wir uns für den kostenlosen Plan entschieden, der eine Datenspeicherung für eine Dauer von 30 Tagen und die Möglichkeit bietet, Benachrichtigungen über Slack zu erhalten..

Nachdem wir unsere Wahl bestätigt haben, wird das persönliche Dashboard angezeigt. Um einen Bereich zu erstellen, navigieren Sie zu *Load Data -> Buckets -> Create new bucket*. Daraufhin öffnet sich ein Bildschirm ähnlich dem in **Bild 8**. Füllen Sie das Feld *Name* aus und klicken Sie auf die Schaltfläche *Create*. Sobald der Bucket erfolgreich erstellt wurde, wird er unter den verfügbaren Datenquellen angezeigt, wie in **Bild 9** dargestellt.

Klicken Sie anschließend auf *Add data -> Client library*. Es erscheint ein neuer Bildschirm, der verschiedene Optionen zum Hochladen von Daten anbietet. Wählen Sie aus diesen Alternativen *Arduino* aus (**Bild 10**). Im folgenden Fenster (**Bild 11**) sehen Sie eine Reihe von Codeschnipseln, die einen Demonstrations-Sketch darstellen. Kopieren Sie die Daten für `INFLUXDB_URL`, `INFLUXDB_ORG` und `INFLUXDB_BUCKET` aus dem ersten Codeschnipsel.

Um die Einrichtung abzuschließen, müssen wir ein Zugriffstoken für den Bucket generieren. Öffnen Sie dazu die linke Seitenleiste und navigieren Sie zu *Load Data -> API Tokens*.

Klicken Sie in dem neu geöffneten Fenster auf *Generate API Token*. Nachdem Sie den Bucket ausgewählt haben, aktivieren Sie sowohl Lese- als auch Schreibrechte, wie in **Bild 12** dargestellt. Sobald das Token generiert ist, kopieren Sie es und bewahren Sie es sicher auf, da wir es in den nächsten Schritten benötigen werden.

Der Sketch

Der Sketch für unser Projekt ist eine Kombination mehrerer Sketche aus den von uns verwendeten Bibliotheken. Sie können die Sketch-Dateien aus dem GitHub-Repository [5] herunterladen. Analysieren wir nun unseren Code, der in drei Abschnitte unterteilt ist, die wir mit entsprechenden Listings vorstellen werden. Lassen Sie uns zunächst

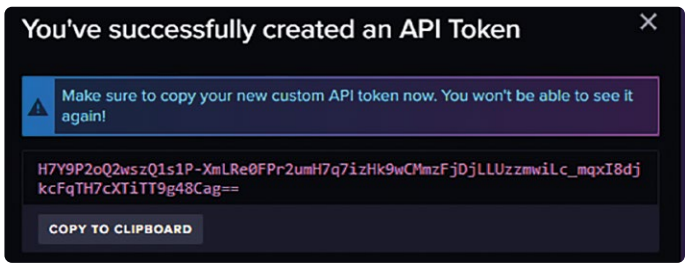


Bild 12. Erstellen eines API-Schlüssels.

Listing 1: Einbindung der Bibliotheken

```
#include <WiFiMulti.h>
#include <ESPmDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
#include <InfluxDbClient.h>
#include <InfluxDbCloud.h>
#include <PZEM004Tv30.h>
#include <Every.h>
```

über die Einbindung der erforderlichen Bibliotheken und Abhängigkeiten sprechen, wie in **Listing 1** gezeigt.

Die *WiFiMulti*-Bibliothek wird für die Verwaltung der WLAN-Kommunikation zwischen dem ESP32 und dem Access Point verwendet, die Bibliotheken *ESPmDNS*, *WiFiUDP* und *ArduinoOTA* [6] für das Hochladen von Sketches OTA (over-the-air) und die Verwaltung von Host-Namen. Um diese Funktionen nutzen zu können, muss die Python-Version 2.7.x auf Ihrem PC installiert sein. Die Bibliotheken *InfluxDbClient* und *InfluxDbCloud* sind für den Daten-Upload in die InfluxDB Cloud verantwortlich. Die Bibliothek *PZEM004Tv30* [7] ermöglicht die serielle Kommunikation mit dem Sensor. Schließlich sorgt die *Every*-Bibliothek dafür, dass Codeblöcke in regelmäßigen Abständen ausgeführt werden, ohne auf delay() zurückgreifen zu müssen.

Betrachten wir nun die nachfolgenden Codeabschnitte, beginnend mit **Listing 2**, das den Präprozessorabschnitt enthält. Mit `#define REFRESH_TIME 5000` legen wir das Intervall (in Millisekunden) für das Hochladen von Daten in die Cloud fest. Anschließend definieren wir den Gerätenamen, die Pins für die serielle Kommunikation und den gewünschten seriellen Port. Objekte mit Bezug zum WLAN und zum Sensor werden instanziiert. IP-Adressen werden definiert, und wenn Sie eine Verbindung zum Access Point mit einer statischen IP konfigurieren möchten, können Sie die Parameter entsprechend dem Subnetz Ihres Netzwerks anpassen. Schließlich werden die Parameter für die WLAN-Verbindung und den Zugriff auf die InfluxDB-Cloud definiert. Ersetzen Sie diese Codezeilen durch den kopierten Codeschnipsel und tragen Sie die fehlenden Daten ein.

Mit `#define WIFI_SSID` und `#define WIFI_PASSWORD` geben wir die SSID und die Passphrase des WLANs an.

Als Nächstes legen wir die Parameter für die Verbindung zur InfluxDB Cloud fest. Der einzige fehlende Parameter ist INFLUXDB_TOKEN, welcher aus dem zuvor erstellen API-Schlüssel abgeleitet werden kann. Fügen Sie den API-Schlüssel zwischen den Anführungszeichen ein. Mit `#define TZ_INFO "CET-1CEST,M3.5.0,M10.5.0/3"` spezifizieren wir die mitteleuropäische Zeitzone für Timestamps.

Um das Client-Objekt für die Verbindung zum InfluxDB-Server zu erstellen, verwenden Sie den folgenden Code:

```
InfluxDBClient client(INFLUXDB_URL, INFLUXDB_ORG,
                      INFLUXDB_BUCKET, INFLUXDB_TOKEN,
InfluxDbCloud2CACert);
```

Listing 2: Definitionen

```
#define REFRESH_TIME 5000 // Delay interval for data upload
#define DEVICE "ESP32"
#define PZEM_RX_PIN 27
#define PZEM_TX_PIN 26
#define PZEM_SERIAL Serial2

/*************************
ESP32 initialization
----------------------

   The ESP32 HW Serial interface can be routed to any GPIO pin
   Here we initialize the PZEM on Serial2 with RX/TX pins 26 and 27
*/
PZEM004Tv30 pzem(PZEM_SERIAL, PZEM_RX_PIN, PZEM_TX_PIN);
WiFiMulti wifiMulti;
IPAddress local_IP(192, 168, 178, 154);
IPAddress gateway(192, 168, 178, 1);
IPAddress subnet(255, 255, 255, 0);
IPAddress primaryDNS(192, 168, 178, 1); //optional
IPAddress secondaryDNS(1, 1, 1, 1); //optional
// WiFi AP SSID
#define WIFI_SSID ""
// WiFi password
#define WIFI_PASSWORD ""
// InfluxDB v2 server url, e.g. https://eu-central-1-1.aws.cloud2.influxdata.com
 // (Use: InfluxDB UI -> Load Data -> Client Libraries)
#define INFLUXDB_URL "https://eu-central-1-1.aws.cloud2.influxdata.com"
// InfluxDB v2 server or cloud API token
 //   (Use: InfluxDB UI -> Data -> API Tokens -> Generate API Token)
#define INFLUXDB_TOKEN ""
// InfluxDB v2 organization id (Use: InfluxDB UI -> User -> About -> Common Ids )
#define INFLUXDB_ORG ""
// InfluxDB v2 bucket name (Use: InfluxDB UI ->  Data -> Buckets)
#define INFLUXDB_BUCKET "Energy Meter @ ElettronicaIN"
// Set timezone string according to
//    https://www.gnu.org/software/libc/manual/html_node/TZ-Variable.html
#define TZ_INFO "CET-1CEST,M3.5.0,M10.5.0/3"
// InfluxDB client instance with preconfigured InfluxCloud certificate
InfluxDBClient client(INFLUXDB_URL, INFLUXDB_ORG,
    INFLUXDB_BUCKET, INFLUXDB_TOKEN, InfluxDbCloud2CACert);
// Data point
Point sensor("EnergyMeter");
```

Listing 3: setup()

```
void setup() {
  Serial.begin(115200);
  // Uncomment in order to reset the internal energy counter
  // pzem.resetEnergy()
  // Setup wifi
  WiFi.mode(WIFI_STA);
  //Comment to use DHCP instead of static IP
   if (!WiFi.config(local_IP, gateway, subnet, primaryDNS, secondaryDNS)) {
    Serial.println("STA Failed to configure");
  }
```

```
  wifiMulti.addAP(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to wifi");
  while (wifiMulti.run() != WL_CONNECTED) {
    //Serial.print(".");
    //delay(100);
    Serial.println("Connection Failed! Rebooting...");
    delay(5000);
    ESP.restart();
  }
  Serial.println();
  // Port defaults to 3232
  // ArduinoOTA.setPort(3232);
  // Hostname defaults to esp3232-[MAC]
  ArduinoOTA.setHostname("ESP32-Energy Meter");
  // No authentication by default
  // ArduinoOTA.setPassword("admin");
  // Password can be set with it's md5 value as well
  // MD5(admin) = 21232f297a57a5a743894a0e4a801fc3
  // ArduinoOTA.setPasswordHash("21232f297a57a5a743894a0e4a801fc3");
  ArduinoOTA
  .onStart([]() {
    String type;
    if (ArduinoOTA.getCommand() == U_FLASH)
      type = "sketch";
    else // U_SPIFFS
      type = "filesystem";
    // NOTE: if updating SPIFFS this would be the place to unmount SPIFFS using SPIFFS.end()
    Serial.println("Start updating " + type);
  })
  .onEnd([]() {
    Serial.println("\nEnd");
  })
  .onProgress([](unsigned int progress, unsigned int total) {
    Serial.printf("Progress: %u%%\r", (progress / (total / 100)));
  })
  .onError([](ota_error_t error) {
    Serial.printf("Error[%u]: ", error);
    if (error == OTA_AUTH_ERROR) Serial.println("Auth Failed");
    else if (error == OTA_BEGIN_ERROR) Serial.println("Begin Failed");
    else if (error == OTA_CONNECT_ERROR) Serial.println("Connect Failed");
    else if (error == OTA_RECEIVE_ERROR) Serial.println("Receive Failed");
    else if (error == OTA_END_ERROR) Serial.println("End Failed");
  });
  ArduinoOTA.begin();
  // Add tags
  sensor.addTag("Dispositivo", DEVICE);
  // Accurate time is necessary for certificate validation and writing in batches
  // For the fastest time sync find NTP servers in your area: https://www.pool.ntp.org/zone/
  // Syncing progress and the time will be printed to Serial.
  timeSync(TZ_INFO, "pool.ntp.org", "time.nis.gov");\
  // Check server connection
  if (client.validateConnection()) {
    Serial.print("Connected to InfluxDB: ");
    Serial.println(client.getServerUrl());
  } else {
    Serial.print("InfluxDB connection failed: ");
    Serial.println(client.getLastErrorMessage());
  }
}
```

Schließlich wird das Sensorobjekt mit `Point sensor("EnergyMeter")` erstellt. Dieses Objekt mit dem Namen `EnergyMeter` wird mit allen vom Sensor erfassten Daten verknüpft. Der Code in **Listing 3** stellt die Einrichtung der Karte dar und initialisiert die serielle Schnittstelle mit einer Baudrate von 115 200.
Der Codeblock beginnt mit `WiFi.mode(WIFI_STA)`; damit wird WLAN für die Verbindung mit einem Access Point im Stationsmodus initialisiert. Der nächste Codeblock:

```
if (!WiFi.config(local_IP, gateway, subnet,
  primaryDNS, secondaryDNS)) {
 Serial.println(“STA failed to configure”);
}
```

legt eine statische IP-Adresse fest, anstatt DHCP zu verwenden. Wenn ein Fehler auftritt, wird eine Warnmeldung an der seriellen Schnittstelle ausgegeben. Wenn Sie DHCP verwenden möchten, können Sie diesen Teil des Codes auskommentieren.
Der Funktionsaufruf `wifiMulti.addAP(WIFI_SSID, WIFI_PASSWORD)` versucht, mit der angegebenen SSID und der Passphrase eine Verbindung zum Access Point herzustellen. Schlägt dies fehl, wird zunächst eine Fehlermeldung an der seriellen Schnittstelle ausgegeben, und das Board wird nach einer Wartezeit von fünf Sekunden neu gestartet. Der Hostname wird mit `ArduinoOTA.setHostname("ESP32-EnergyMeter")` festgelegt, der über mDNS übertragen und in der Arduino-IDE zum Laden des OTA-Sketches angezeigt wird. Es gibt kommentierten Code, um das Laden des OTA-Sketches auf Benutzer mit einem Benutzernamen und Passwort zu beschränken. Kommentieren Sie diese Codezeilen aus, um diese Sicherheitsfunktion hinzuzufügen.
Die Einstellungen für den OTA-Modus werden mit dem folgenden Codeblock verwaltet:

```
ArduinoOTA.onStart([]() {
  String type;
  // …
  if (error == OTA_AUTH_ERROR) Serial.println("Auth
Failed");
  else if (error == OTA_BEGIN_ERROR) Serial.println("Begin
Failed");
   else if (error == OTA_CONNECT_ERROR) Serial.
println("Connect Failed");
   else if (error == OTA_RECEIVE_ERROR) Serial.
println("Receive Failed");
  else if (error == OTA_END_ERROR) Serial.println("End
Failed");
});
ArduinoOTA.begin();
```

Der Code-Block bearbeitet (in dieser Reihenfolge): OTA-Start, OTA-Ende, Sketch-Laden und Fehler. Mit `ArduinoOTA.begin()` wird das Laden des OTA-Sketches gestartet. Mit `sensor.addTag("Device", DEVICE);` wird ein Tag gesetzt, das in diesem Fall dem Gerätenamen entspricht. Diese Funktion könnte nützlich sein, um einen oder mehrere Sensoren innerhalb einer Gruppe zu unterscheiden.
Mit `timeSync(TZ_INFO, "pool.ntp.org", "time.nis.gov");` stellt der ESP eine Verbindung zu den beiden NTP-Servern (Network Time Protocol) *pool.ntp.org* und *time.nis.gov* her, um Datum und Uhrzeit zu ermitteln.
Schließlich versucht der Codeblock, eine Verbindung zu den InfluxDB-Servern herzustellen:

```
if (client.validateConnection()) {
  Serial.print(“Connected to InfluxDB: ”);
  Serial.println(client.getServerUrl());
}
else {
  Serial.print(“InfluxDB connection failed: ”);
  Serial.println(client.getLastErrorMessage());
}
```

Wenn die Verbindung erfolgreich ist, wird die Server-URL an der seriellen Schnittstelle ausgegeben, wenn nicht, die entsprechende Fehlermeldung.
Gehen wir nun zu **Listing 4** über, das den `loop()`-Code für unseren Sketch enthält. Mit `ArduinoOTA.handle()` wartet der ESP32 darauf, dass OTA kompilierte Sketche empfängt.
Der Code innerhalb des Blocks `EVERY(REFRESH_TIME) { ... }` wird in den zuvor definierten regelmäßigen Abständen und ohne Verzögerungen ausgeführt. Innerhalb dieses Blocks wird die Adresse des PZEM gelesen und an der seriellen Schnittstelle ausgegeben.
Als Nächstes werden Sensorvariablen erstellt und mit Funktionen aus der Sensorbibliothek initialisiert. Diese Variablen umfassen Spannung, Strom, Wirkleistung (in Wh), Wirkenergie (in kWh), Frequenz und Leistungsfaktor (zwischen -1 und 1, was das Verhältnis zwischen Wirkleistung und Scheinleistung darstellt). Wenn eine der Variablen keine Zahl ist, wird ein Lesefehler, andernfalls werden die Sensormesswerte an der seriellen Schnittstelle ausgegeben. Danach werden die initialisierten Felder und Timestamps gelöscht und die Variablen werden für das Hochladen in die Cloud vorbereitet. Schließlich werden die WLAN-Verbindung und der erfolgreiche Daten-Upload überprüft. Schlägt der Upload fehl, wird der letzte Fehlercode über die serielle Schnittstelle ausgegeben.

Einbau und Messung

Wir hatten geplant, die Stromversorgung, den ESP32 und das Hauptsensormodul in einer schicken Plastikbox wie im Titelbild unterzubringen, aus der nur die Kabel für die Stromversorgung und für den Stromsensor-Ringkernwandler herausschauen. Leider ist ein solches Gehäuse denkbar ungeeignet, da die 230-V-Schraubklemmen nicht berührsicher abgeschirmt sind. Das Datenblatt gibt auch keine Auskunft darüber, ob die Zuleitungen für den Ringkern regelgerecht galvanisch von der „Hochspannung“ getrennt sind; verlassen sollte man sich nicht darauf. Achten Sie also darauf, dass das Gehäuse mit den Zuleitungen den Vorschriften in puncto Sicherheit entspricht! Wenn man das Gerät dafür einsetzen möchte, den Energiehunger eines einzelnen Gerätes zu ermitteln, empfiehlt sich der berührungssichere Einbau der gesamten Elektronik inklusive aller Kabel in ein Steckergehäuse mit Schuko-Buchse. In diesem Falle wäre noch nicht einmal eine galvanisch getrennte 5-V-Spannungsversorgung erforderlich!
Orientieren Sie sich einfach an der Verdrahtung, die bereits am

Listing 4: loop()

```
void loop() {

  ArduinoOTA.handle();

  EVERY(REFRESH_TIME) {
    // Print the custom address of the PZEM
    Serial.print("Custom Address:");
    Serial.println(pzem.readAddress(), HEX);
    // Read the data from the sensor
    float voltage = pzem.voltage();
    float current = pzem.current();
    float power = pzem.power();
    float energy = pzem.energy();
    float frequency = pzem.frequency();
    float pf = pzem.pf();
    // Check if the data is valid
    if (isnan(voltage)) {
      Serial.println("Error reading voltage");
    } else if (isnan(current)) {
      Serial.println("Error reading current");
    } else if (isnan(power)) {
      Serial.println("Error reading power");
    } else if (isnan(energy)) {
      Serial.println("Error reading energy");
    } else if (isnan(frequency)) {
      Serial.println("Error reading frequency");
    } else if (isnan(pf)) {
      Serial.println("Error reading power factor");
    } else {
      // Print the values to the Serial console
      Serial.print("Voltage: ");      Serial.print(voltage);      Serial.println("V");
      Serial.print("Current: ");      Serial.print(current);      Serial.println("A");
      Serial.print("Power: ");        Serial.print(power);        Serial.println("W");
      Serial.print("Energy: ");       Serial.print(energy, 3);     Serial.println("kWh");
      Serial.print("Frequency: ");    Serial.print(frequency, 1); Serial.println("Hz");
      Serial.print("PF: ");           Serial.println(pf);
      ////UPLOAD DATA
      // Clear fields for reusing the point. Tags will remain untouched
      sensor.clearFields();
      // Store measured value into point
      sensor.addField("Tensione", voltage);
      sensor.addField("Corrente", current);
      sensor.addField("Potenza", power);
      sensor.addField("Energia", energy);
      sensor.addField("Frequenza", frequency);
      sensor.addField("Fattore di potenza", pf);
      // Print what are we exactly writing
      Serial.print("Writing: ");
      Serial.println(sensor.toLineProtocol());
      // Check WiFi connection and reconnect if needed
      if (wifiMulti.run() != WL_CONNECTED) {
        Serial.println("Wifi connection lost");
      }
      // Write point
      if (!client.writePoint(sensor)) {
        Serial.print("InfluxDB write failed: ");
        Serial.println(client.getLastErrorMessage());
      }
    }
    Serial.println();
  }
}
```

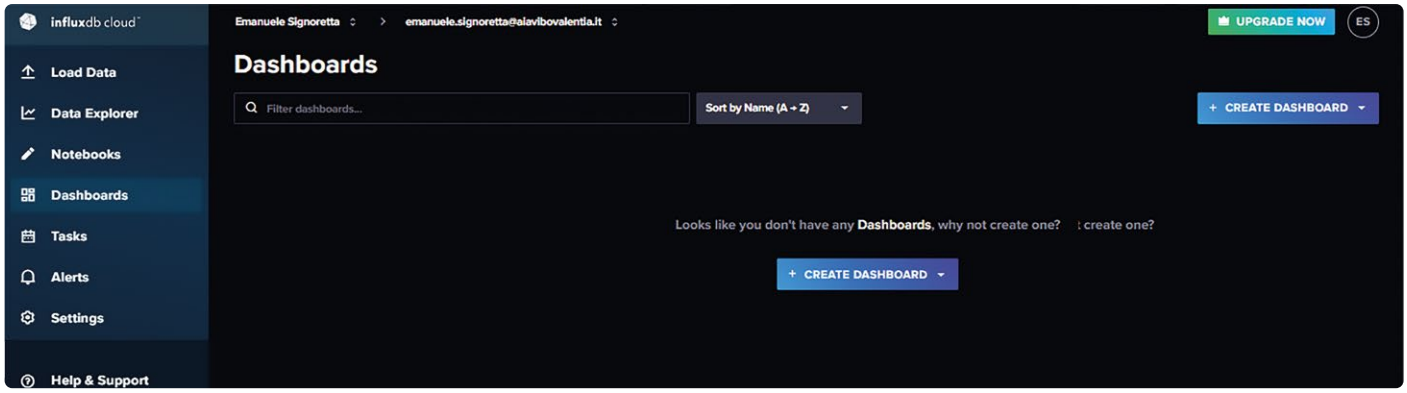


Bild 13. Erstellen des Dashboards.

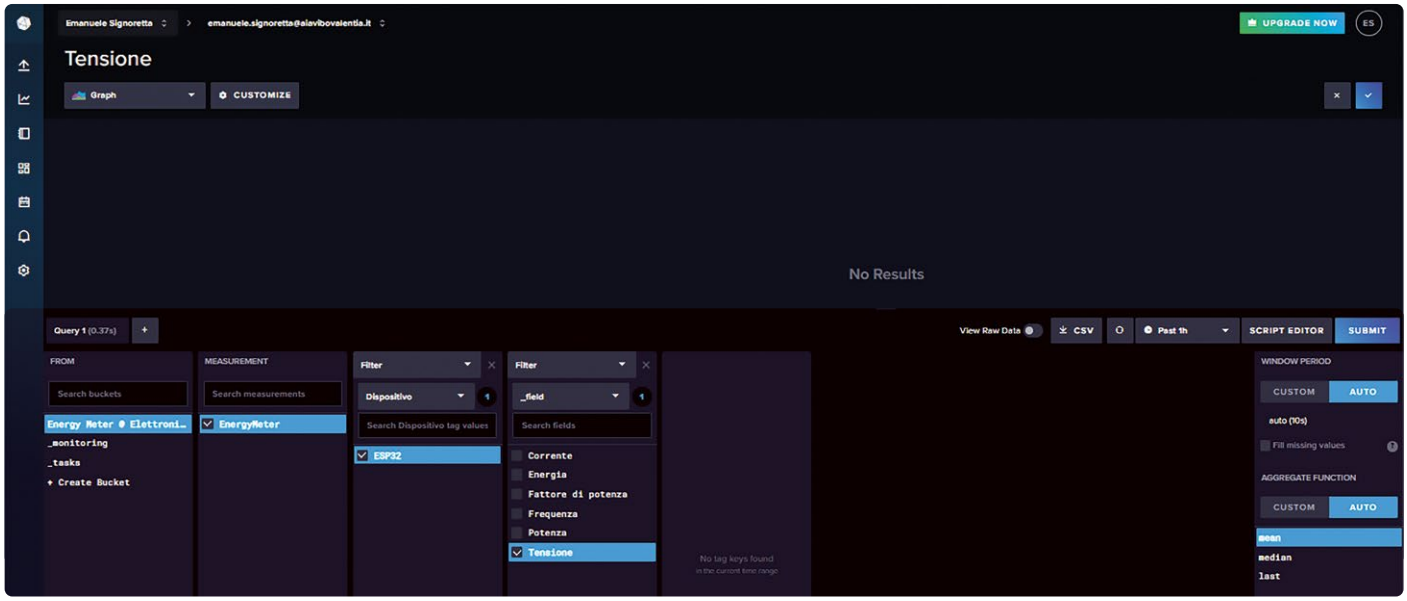


Bild 14. Hinzufügen der Dashboard-Grafiken.

Anfang dieses Artikels in Bild 3 gezeigt wurde, links die 5-V-Litzen und rechts in blau und braun die 230-V-Verkabelung, parallel zum Eingang des Netzteils und zu den leider nicht deutlich gekennzeichneten Klemmen 1/2 der PZEM-004T-Platine. Der Ringkern wird an die Klemmen 3/4 angeschlossen. Im Steckergehäuse wird ein Leiter (L oder N) sowie der Schutzleiter PE durchgeschleift und der andere durch den Ringkern geführt.

Bevor Sie die Einzelteile des Energiemeters sicher im Gehäuse fixieren, sollten Sie den Sketch via Micro-USB (über den das Modul auch mit Spannung versorgt wird) auf das ESP-Board laden. Dann können Sie alle Bestandteile wie beschrieben verkabeln und im Gehäuse fixieren. Nach der Kontrolle der Kabelverbindungen (siehe auch **Tabelle 1**) kann das Gehäuse verschlossen und das System gestartet werden. Bei dieser Vorgehensweise kommen Sie zu keinem Zeitpunkt der gefährlichen Netzspannung zu nahe!

Starten und Dashboard einrichten

Nach dem Start des Energiemeters kehren wir auf die InfluxDB-Website zurück und gehen von unserem Dashboard aus auf *Dashboards* -> *Create new dashboard*. Daraufhin erscheint ein Bildschirm wie in **Bild 13**, in dem wir unserem Dashboard einen Namen geben und auf *Add Cell* klicken. In dem neu erscheinenden Bildschirm (**Bild 14**) wählen wir für jedes der Diagramme, aus denen das Dashboard besteht, aus, welche Parameter aufgenommen werden sollen. Wir wählen in dieser Reihenfolge den Bucket, die Messungen, das Gerät und schließlich die anzuzeigenden Werte aus. Sie können wählen, welche Grafiken für die Werte verwendet werden sollen: Heatmap, Tachometer, einfaches Diagramm, Tabelle und so weiter, und auch welche Skalierungswerte verwendet werden sollen. Wir passen die Zellen nach unseren Wünschen an und wiederholen den Vorgang für jeden Wert, den wir anzeigen möchten, bis wir ein Ergebnis wie in **Bild 15** erhalten, das unser vollständiges Dashboard zeigt.

Schlussfolgerung

Damit ist die Beschreibung unseres einfachen, aber mächtigen Energiezählers abgeschlossen. Die Flexibilität des ESP32-Boards und die vielen Funktionen der unterstützten Netzwerkprotokolle und -plattformen

Tabelle 1: Niederspannungsverbindungen zwischen PZEM-004 und ESP32.

Pin PZEM-004	Pin ESP32
VCC	V5
GND	GND
RX	26
TX	27

Bild 15. Das fertige Dashboard.

ermöglichen es uns, die Anwendungsbereiche nach unseren Bedürfnissen zu erweitern. Schon mal über den Einbau in ein Hutschienengehäuse nachgedacht, das fest im Zählerkasten montiert wird?

Übersetzung von Matze Schrumpf — 230279-02

Haben Sie Fragen oder Kommentare?

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

Über den Autor

Emanuele Signoretta wurde im Jahr 2000 in Vibo Valentia, einer kleinen Stadt im Süden Italiens, geboren. Er interessiert sich leidenschaftlich für alles, was mit Informations- und Kommunikationstechnik zu tun hat. Während er die Oberschule besuchte, entdeckte er Arduino und die Einfachheit der Programmierung. Von da an schrieb er Code für Arduino- und Fishino-Boards. Jetzt hat er mit STM32- und ESP32-basierten Boards angefangen. Er glaubt an das Open-Source-Ethos und verwendet Linux-basierte Distros. Emanuele schließt derzeit seinen Bachelor-Abschluss in Elektronik und Kommunikationstechnik am Politecnico di Torino ab. Außerdem arbeitet er für die Rai, den nationalen italienischen Fernsehsender.

Passende Produkte

- **ESP32-C3-DevKitM-1**
  https://elektor.de/20324
- **Koen Vervloesem, *Getting Started with ESPHome*, Elektor 2021**
  Buch, Paperback: https://elektor.de/19738
  E-Buch, PDF: https://elektor.de/19739
- **Bundle: *Getting Started with ESPHome* + LILYGO TTGO T-Display ESP32 (16 MB)**
  https://elektor.de/19896

WEBLINKS

[1] ESP32-Board: https://futuranet.it/prodotto/esp32-scheda-di-sviluppo-32-gpio-con-wifi-e-bluetooth
[2] Gehäuse: https://www.strapubox.de/steckergehaeuse/schukosteckergehaeuse/
[3] PZEM-004 Datenblatt auf Github: https://bit.ly/3qTZdHf
[4] InfluxDB-Account: https://cloud2.influxdata.com/signup
[5] GitHub-Repository des Projekts: http://github.com/signorettae/ESP32-EnergyMeter
[6] ArduinoOTA-Bibliothek: https://github.com/jandrassy/ArduinoOTA
[7] Arduino-Bibliothek für den aktualisierten PZEM-004T: http://github.com/mandulaj/PZEM-004T-v30

Eine Anleitung zur Bare-Metal-Programmierung

Teil 2: Exaktes Timing, UART und Debugging

Von Sergey Lyubka (Irland)

Im ersten Teil der Anleitung haben wir gelernt, wie man auf Mikrocontroller-Register zugreift, um Pins zu steuern. Außerdem haben wir eine minimale Firmware und unsere erste Blinkende-LED-Demo mit Hilfe eines Linker-Skripts und eines Makefiles erstellt. In diesem zweiten Teil befassen wir uns mit dem genauen Timing über Systemtakte, dem UART und der Fehlersuche.

Anmerkung des Herausgebers: Diese Anleitung ist ein lebendes und wachsendes Dokument auf GitHub [1]. Daher haben wir uns entschlossen, diesen Artikel um einen weiteren Teil zu erweitern, den Sie in der nächsten Elektor-Ausgabe (11-12/2023) finden werden.

Blinky mit Systick-Interrupt

Für unsere erste LED-Demo „Blinky“ im ersten Teil der Anleitung [2] haben wir eine Verzögerungsfunktion namens `spin()` verwendet, die lediglich eine bestimmte Anzahl von NOP-Anweisungen ausgeführt hat. Für eine viel genauere Zeitmessung sollten wir den SysTick-Interrupt von ARM aktivieren. SysTick ist ein 24-Bit-Hardwarezähler, Teil des ARM-Kerns, und daher im Arm-v7-M Architecture Reference Manual [3] dokumentiert ist. Ein Blick in dieses Handbuch zeigt, dass SysTick vier Register hat:

- CTRL - zum Aktivieren/Deaktivieren von SysTick
- LOAD - ein anfänglicher Zählerwert
- VAL - ein aktueller Zählerwert, der bei jedem Taktzyklus dekrementiert wird
- CALIB - Kalibrierungsregister

Jedes Mal, wenn VAL auf Null fällt, wird ein SysTick-Interrupt erzeugt. Der SysTick-Interrupt-Index in der Vektortabelle ist 15, also müssen wir ihn setzen. Beim Booten läuft unser Board Nucleo-F429ZI von STMicroelectronics mit 16 MHz. Wir können den SysTick-Zähler so konfigurieren, dass er jede Millisekunde ein Interrupt auslöst.
Definieren wir zunächst ein SysTick-Peripheral. Wir kennen vier Register, und aus dem Arm-Referenzhandbuch geht hervor, dass die SysTick-Adresse `0xe000e010` ist. Also

```
struct systick {
  volatile uint32_t CTRL, LOAD, VAL, CALIB;
};
#define SYSTICK ((struct systick *) 0xe000e010)
```

Als nächstes fügen wir eine API-Funktion hinzu, die SysTick konfiguriert. Wir müssen den SysTick im `SYSTICK->CTRL`-Register aktivieren und ihn über `RCC->APB2ENR` takten, wie in Abschnitt 7.3.14 des Manuals [4] beschrieben:

```
#define BIT(x) (1UL << (x))
static inline void systick_init(uint32_t ticks) {
  // SysTick timer is 24 bits
  if ((ticks - 1) > 0xffffff) return;
  SYSTICK->LOAD = ticks - 1;
  SYSTICK->VAL = 0;
  // Enable systick
  SYSTICK->CTRL = BIT(0) | BIT(1) | BIT(2);
  RCC->APB2ENR |= BIT(14); // Enable SYSCFG
}
```

Wie bereits erwähnt, läuft das Nucleo-F429ZI-Board mit 16 MHz. Wenn wir `systick_init(16000000 / 1000)` aufrufen, wird also jede Millisekunde ein SysTick-Interrupt erzeugt. Wir sollten eine Interrupt-Handler-Funktion definieren wie folgende, die einfach einen 32-Bit-Millisekunden-Zähler inkrementiert:

```
// "volatile" is important!!
static volatile uint32_t s_ticks;
void SysTick_Handler(void) {
  s_ticks++;
}
```

Bei einem 16-MHz-Takt wird der SysTick-Zähler so initialisiert, dass alle 16.000 Zyklen ein Interrupt ausgelöst wird: Der anfängliche Wert von `SYSTICK->VAL` ist 15.999, dann wird er mit jedem Zyklus

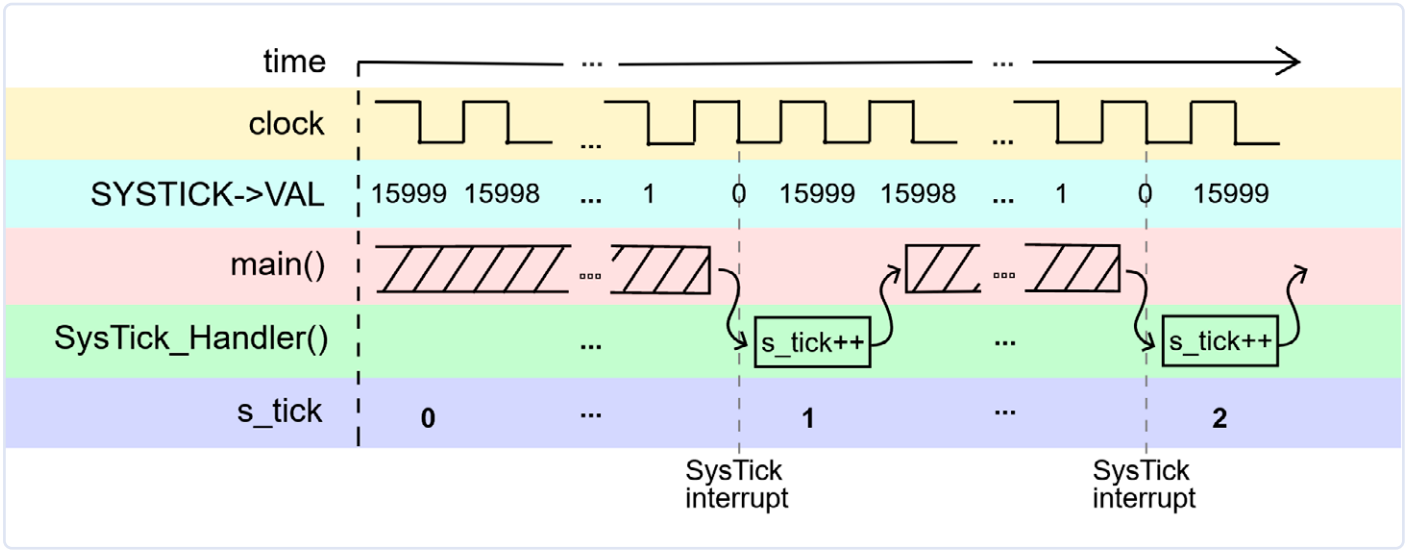


Bild 1. Zeitskalendarstellung der unterbrochenen Firmware-Ausführung mit der Funktion SysTick_Handler().

dekrementiert, bis er 0 erreicht und ein Interrupt ausgelöst wird. Die Ausführung des Firmware-Codes wird unterbrochen und die Funktion `SysTick_Handler()` aufgerufen, um die Variable `s_tick` zu erhöhen. **Bild 1** zeigt, wie dies auf einer Zeitachse aussieht.
Der Qualifier `volatile` ist hier erforderlich, weil `s_ticks` vom Interrupt-Handler geändert wird. `volatile` verhindert, dass der Compiler den `s_ticks`-Wert in einem CPU-Register optimiert/zwischenspeichert; stattdessen greift der generierte Code immer auf den Speicher zu. Aus diesem Grund ist der Qualifier `volatile` auch in den peripheren `struct`-Definitionen vorhanden. Es ist wichtig, dies zu verstehen, weshalb wir es anhand der einfachen Arduino-Funktion `delay()` demonstrieren wollen. Verwenden wir unsere Variable `s_ticks`:

```
// This function waits "ms" milliseconds
void delay(unsigned ms) {
  // Time in the future when we need to stop
  uint32_t until = s_ticks + ms; /
  while (s_ticks < until) (void) 0; // Loop until then
}
```

Kompilieren wir nun diesen Code sowohl mit als auch ohne den `volatile`-Qualifier für `s_ticks` und vergleichen wir den kompilierten Assemblercode:

```
// NO VOLATILE: uint32_t s_ticks;
ldr r3, [pc, #8] // cache s_ticks
ldr r3, [r3, #0] // in r3
adds r0, r3, r0 // r0 = r3 + ms
cmp r3, r0 // ALWAYS FALSE
bcc.n 200000d2
bx lr

// VOLATILE: volatile uint32_t s_ticks;
ldr r2, [pc, #12]
ldr r3, [r2, #0] // r3 = s_ticks
adds r3, r3, r0 // r3 = r3 + ms
ldr r1, [r2, #0] // RELOAD: r1 = s_ticks
cmp r1, r3 // compare
bcc.n 200000d2
bx lr
```

Wenn es kein `volatile` gibt, läuft die Funktion `delay()` in einer Endlosschleife und kehrt nie zurück. Das liegt daran, dass sie den Wert von `s_ticks` in einem Register zwischenspeichert (optimiert) und nie aktualisiert. Ein Compiler macht das, weil er nicht weiß, dass `s_ticks` an anderer Stelle durch den Interrupt-Handler aktualisiert wird! Der mit `volatile` kompilierte Code hingegen lädt den `s_ticks`-Wert bei jeder Iteration. Die Faustregel lautet also: **Werte im Speicher, die von Interrupt-Handlern oder von der Hardware aktualisiert werden, müssen als** `volatile` **deklariert werden!**
Jetzt sollten wir den Interrupt-Handler `SysTick_Handler()` zur Vektortabelle hinzufügen:

```
__attribute__((section(".vectors")))
    void (*tab[16 + 91])(void) = {
    _estack, _reset, 0, 0, 0, 0, 0, 0, 0, 0,
                     0, 0, 0, 0,  SysTick_Handler
};
```

Und damit haben wir schon eine präzise Millisekundenuhr! Lassen Sie uns eine Hilfsfunktion für beliebige periodische Zeitgeber erstellen:

```
// t: expiration time, prd: period,
//    now: current time. Return true if expired
bool timer_expired(uint32_t *t, uint32_t prd,
                   uint32_t now) {
  if (now + prd < *t) *t = 0;
    // Time wrapped? Reset timer
  if (*t == 0) *t = now + prd;
    // First poll? Set expiration
  if (*t > now) return false;
    // Not expired yet,return
  *t = (now - *t) > prd ? now + prd : *t + prd;
    // Next expiration time
  return true; // Expired, return true
}
```

Jetzt können wir unsere Hauptschleife aktualisieren und einen präzisen Timer für das Blinken der LEDs verwenden. In folgendem Beispiel wird Blinkintervall von 250 ms eingestellt:

```
// Declare timer and 500ms period
uint32_t timer, period = 500;
for (;;) {
  if (timer_expired(&timer, period, s_ticks)) {
    static bool on;       // This block is executed
    gpio_write(led, on); // Every "period" milliseconds
    on = !on; // Toggle LED state
  }
  // Here we could perform other activities!
}
```

Beachten Sie, dass wir durch die Verwendung von SysTick mit einer Hilfsfunktion `timer_expired()` unsere Main-Schleife (auch „Superschleife“ genannt) nicht blockierend gemacht haben. Das bedeutet, dass wir innerhalb dieser Schleife viele Aktionen durchführen können - zum Beispiel verschiedene Zeitgeber mit unterschiedlichen Zeiträumen, die alle rechtzeitig ausgelöst werden.
Den vollständigen Quellcode des Projekts finden Sie im Ordner *step-2-systick* [5].

UART-Debug-Ausgabe hinzufügen

Nun ist es an der Zeit, unserer Firmware eine für Menschen lesbare Diagnosemöglichkeit hinzuzufügen. Eine der MCU-Peripherien ist eine serielle UART-Schnittstelle. Ein Blick auf die Memory Map in Abschnitt 2.3 des Mikrocontroller-Handbuchs zeigt, dass es mehrere UART/USART-Controller gibt, also Schaltungsteile innerhalb der MCU, die bei entsprechender Konfiguration über bestimmte Pins Daten austauschen können. Eine minimale UART-Konfiguration verwendet die beiden Pins RX (Empfang) und TX (Senden).
In Abschnitt 6.9 des Nucleo-Board-Handbuchs [6] sehen wir, dass der Controller USART3 die Pins PD8 (TX) und PD9 (RX) verwendet und mit dem Debug-Anschluss ST-LINK auf dem Board verbunden ist. Das heißt, wenn wir USART3 konfigurieren und Daten über den PD9-Pin ausgeben, können wir sie über die ST-LINK-USB-Verbindung auf unsere Workstation übertragen.
Lassen Sie uns also eine praktische API für den UART erstellen, so wie wir es für GPIO getan haben. Abschnitt 30.6 [4] fasst die UART-Register zusammen, und hier ist unsere entsprechende UART-`struct`:

```
struct uart {
  volatile uint32_t SR, DR, BRR, CR1, CR2, CR3, GTPR;
};
#define UART1 ((struct uart *) 0x40011000)
#define UART2 ((struct uart *) 0x40004400)
#define UART3 ((struct uart *) 0x40004800)
```

Um einen UART zu konfigurieren, müssen wir:

- den UART-Takt aktivieren, indem das entsprechende Bit in `RCC->APB2ENR` gesetzt wird
- den alternativen Funktions-Pin-Modus für die RX- und TX-Pins einstellen. Je nach verwendetem Peripheriegerät kann es mehrere alternative Funktionen (AF) für einen bestimmten Pin geben. Die AF-Liste finden Sie im STM32F429ZI-Datenblatt, Tabelle 12 [7].
- die Baudrate (Empfangs-/Sendetaktfrequenz) über das BRR-Register einstellen
- den Empfang und das Senden der Peripherie über das CR1-Register aktivieren

Wir wissen bereits, wie man einen GPIO-Pin auf einen bestimmten Modus einstellt. Wenn sich ein Pin im AF-Modus (alternative Funktion) befindet, müssen wir auch die „Funktionsnummer“ angeben, also einstellen, welche Peripherie genau die Kontrolle übernimmt. Dies kann über das *Alternate Function Register* `AFR` der GPIO-Peripherie erfolgen. Wenn wir die Beschreibung des AFR-Register im Referenzhandbuch lesen, erfahren wir, dass die AF-Nummer vier Bits belegt, so dass die gesamte Einrichtung für 16 Pins zwei Register belegt.

```
static inline void gpio_set_af(uint16_t pin,
                               uint8_t af_num) {
  struct gpio *gpio = GPIO(PINBANK(pin)); // GPIO bank
  int n = PINNO(pin); // Pin number
  gpio->AFR[n >> 3] &= ~(15UL << ((n & 7) * 4));
  gpio->AFR[n >> 3] |= ((uint32_t) af_num)
                       << ((n & 7) * 4);
}
```

Um den registerspezifischen Code vollständig aus der GPIO-API auszublenden, verschieben wir die GPIO-Takt-Initialisierung in die Funktion `gpio_set_mode()`:

```
static inline void
  gpio_set_mode(uint16_t pin, uint8_t mode) {
    struct gpio *gpio = GPIO(PINBANK(pin)); // GPIO bank
    int n = PINNO(pin); // Pin number
// Enable GPIO clock
    RCC->AHB1ENR |= BIT(PINBANK(pin));
    ...
```

Nun können wir eine API-Funktion zur UART-Initialisierung erstellen (**Listing 1**).
Schließlich brauchen wir noch Funktionen zum Lesen und Schreiben auf dem UART. Im Referenzhandbuch [4], Abschnitt 30.6.1, erfahren wir, dass das Statusregister SR anzeigt, wenn Daten bereit stehen:

```
static inline int uart_read_ready(struct uart *uart) {
  // If RXNE bit is set, data is ready
  return uart->SR & BIT(5);
}
```

Das Datenbyte selbst kann aus dem Datenregister DR geholt werden:

```
static inline uint8_t uart_read_byte(struct uart *uart) {
  return (uint8_t) (uart->DR & 255);
}
```

Die Übertragung eines einzelnen Bytes kann auch über das Datenregister erfolgen. Nachdem ein Byte zum Schreiben gesetzt wurde,

Listing 1: API-Funktion zur UART-Initialisierung.

```
#define FREQ 16000000  // CPU frequency, 16 Mhz
 static inline void uart_init(struct uart *uart, unsigned long baud) {
   // https://www.st.com/resource/en/datasheet/stm32f429zi.pdf
   uint8_t af = 7;           // Alternate function
   uint16_t rx = 0, tx = 0;  // pins

   if (uart == UART1) RCC->APB2ENR |= BIT(4);
   if (uart == UART2) RCC->APB1ENR |= BIT(17);
   if (uart == UART3) RCC->APB1ENR |= BIT(18);
   if (uart == UART1) tx = PIN('A', 9), rx = PIN('A', 10);
   if (uart == UART2) tx = PIN('A', 2), rx = PIN('A', 3);
   if (uart == UART3) tx = PIN('D', 8), rx = PIN('D', 9);

  gpio_set_mode(tx, GPIO_MODE_AF);
   gpio_set_af(tx, af);
   gpio_set_mode(rx, GPIO_MODE_AF);
   gpio_set_af(rx, af);
   uart->CR1 = 0;                          // Disable this UART
   uart->BRR = FREQ / baud;                // FREQ is a UART bus frequency
   uart->CR1 |= BIT(13) | BIT(2) | BIT(3);  // Set UE, RE, TE
 }
```

muss das Ende der Übertragung abgewartet werden, was über Bit 7 im Statusregister angezeigt wird:

```
static inline void uart_write_byte(struct uart *uart,
                                   uint8_t byte) {
  uart->DR = byte;
  while ((uart->SR & BIT(7)) == 0) spin(1);
}
```

Und das Schreiben eines Puffers:

```
static inline void
  uart_write_buf(struct uart *uart,
                 char *buf, size_t len) {
    while (len-- > 0)
      uart_write_byte(uart, *(uint8_t *) buf++);
}
```

Jetzt initialisieren wir den UART in unserer `main()`-Funktion

```
…
uart_init(UART3, 115200); // Initialize UART
```

und können jedes Mal, wenn die LED blinkt, die Meldung „hi\r\n“ ausgeben!

```
if (timer_expired(&timer, period, s_ticks)) {
  …
  uart_write_buf(UART3, “hi\r\n”, 4); // Write message
}
```

Erstellen Sie den neuen Build, flashen Sie ihn und schließen Sie ein Terminalprogramm an ST-LINK an. Auf meiner Mac-Workstation verwende ich das Programm *cu* (call unix), das natürlich auch unter Linux funktioniert. Für Windows bietet sich das serielle Dienstprogramm PuTTY [8] an. Starten Sie ein Terminal und beobachten Sie die Meldungen:

```
$ cu -l /dev/cu.YOUR_SERIAL_PORT -s 115200
hi
hi
```

Der vollständige Quellcode des Projekts befindet sich im Ordner *step-3-uart* [9].

Umleitung von printf() auf UART

Nun ersetzen wir den `uart_write_buf()`- durch einen `printf()`-Aufruf, der uns die Möglichkeit gibt, Ausgaben zu formatierten, was das Drucken von Diagnoseinformationen flexibler macht. Dazu implementieren wir das sogenannte „printf()-style debugging“.
Die von uns verwendete GNU-ARM-Toolchain enthält nicht nur einen GCC-Compiler und andere Tools, sondern auch eine C-Bibliothek namens *newlib* [10], die von RedHat für eingebettete Systeme entwickelt wurde.
Wenn unsere Firmware eine Standard-C-Bibliotheksfunktion aufruft, zum Beispiel `strcmp()`, dann wird ein *newlib*-Code vom GCC-Linker in unsere Firmware eingefügt.
Einige der Standard-C-Funktionen, insbesondere Operationen für den Datei-Input/Output (IO), werden von *newlib* auf eine besondere Weise implementiert: Diese Funktionen rufen schließlich eine Reihe von Low-Level-IO-Funktionen auf, die als *syscalls* bezeichnet werden. Zum Beispiel:

- `fopen()` ruft schließlich `_open()` auf
- `fread()` ruft schließlich eine untergeordnete Funktion `_read()` auf

Listing 2. Die Funktion main() function wird schön kompakt.

```
#include "hal.h"

static volatile uint32_t s_ticks;
 void SysTick_Handler(void) {
   s_ticks++;
 }

int main(void) {
   uint16_t led = PIN('B', 7);            // Blue LED
   systick_init(16000000 / 1000);         // Tick every 1 ms
   gpio_set_mode(led, GPIO_MODE_OUTPUT);  // Set blue LED to output mode
   uart_init(UART3, 115200);              // Initialise UART
   uint32_t timer = 0, period = 500;      // Declare timer and 500ms period
   for (;;) {
     if (timer_expired(&timer, period, s_ticks)) {
       static bool on;                      // This block is executed
       gpio_write(led, on);                 // Every 'period' milliseconds
       on = !on;                            // Toggle LED state
       uart_write_buf(UART3, "hi\r\n", 4);  // Write message
     }
     // Here we could perform other activities!
   }
   return 0;
 }
```

- `fwrite()`, `fprintf()`, `printf()` rufen schließlich ein Low-Level-`_write()` auf
- `malloc()` ruft schließlich `_sbrk()` auf, und so weiter.

Indem wir einen `_write()`-Syscall modifizieren, können wir `printf()` auf beliebige Ziele umlenken. Dieser Mechanismus wird „IO-Retargeting“ genannt.
Hinweis: Auch die STM32-Cube-IDE verwendet ARM GCC mit *newlib*, weshalb Cube-Projekte typischerweise eine *syscalls.c*-Datei enthalten. Andere Toolchains, zum Beispiel CSS von TI und CC von Keil nutzen möglicherweise eine andere C-Bibliothek mit einem etwas anderen Retargeting-Mechanismus. Wir verwenden *newlib*, also modifizieren wir den `_write()`-Syscall, um auf UART3 zu drucken. Zuvor müssen wir aber unseren Quellcode wie folgt organisieren:

- Verschieben Sie alle API-Definitionen nach *mcu.h*
- Verschieben Sie den Startup-Code nach *startup.c*
- Erstellen Sie eine leere Datei *syscalls.c* für *newlib*-Syscalls
- Ändern Sie das Makefile, um *syscalls.c* und *startup.c* in den Build aufzunehmen.

Nachdem wir alle API-Definitionen nach *mcu.h* verschoben haben, wird unsere Datei *main.c* recht kompakt. Beachten Sie, dass dabei Low-Level-Register nicht auftauchen, sondern nur High-Level-API-Funktionen, die einfach zu verstehen sind - siehe **Listing 2**. Großartig, jetzt wollen wir `printf()` auf UART3 umleiten. Kopieren Sie in die leere Datei *syscalls.c* den folgenden Code und fügen Sie ihn ein:

```
#include "mcu.h"
int _write(int fd, char *ptr, int len) {
  (void) fd, (void) ptr, (void) len;
  if (fd == 1) uart_write_buf(UART3, ptr, (size_t) len);
  return -1;
}
```

Damit sagen wir: Wenn der Dateideskriptor `fd`, in den wir schreiben, 1 ist (was ein Standardausgabedeskriptor ist), dann schreibe den Puffer in UART3. Andernfalls wird er ignoriert. Das ist die Essenz des Retargeting! Das Rebuilden dieser Firmware führt zu einer Reihe von Linker-Fehlern, wie in **Listing 3** gezeigt. Da wir eine Funktion *newlib stdio* verwendet haben, müssen wir `newlib` mit dem Rest der Syscalls versorgen. Fügen wir nur einen einfachen Stub hinzu, der nichts tut (**Listing 4**).
Jetzt ergibt ein Rebuild keine Fehlermeldung mehr. Im letzten Schritt ersetzen Sie den Aufruf `uart_write_buf()` in der `main()`-Funktion durch einen `printf()`-Aufruf, der etwas Nützliches ausgibt, zum Beispiel einen LED-Status und den aktuellen Wert von Systick:

```
// Write message
printf("LED: %d, tick: %lu\r\n", on, s_ticks);
```

Die serielle Ausgabe sieht wie folgt aus:

```
LED: 1, tick: 250
LED: 0, tick: 500
LED: 1, tick: 750
LED: 0, tick: 1000
```

Herzlichen Glückwunsch! Wir haben erlernt, wie IO-Retargeting funktioniert, und können nun unsere Firmware mit `printf()`

Listing 3. Viele Linker-Fehler.

```
../../arm-none-eabi/lib/thumb/v7e-m+fp/hard/libc_nano.a(lib_a-sbrkr.o): in function `_sbrk_r':
 sbrkr.c:(.text._sbrk_r+0xc): undefined reference to `_sbrk'
 closer.c:(.text._close_r+0xc): undefined reference to `_close'
 lseekr.c:(.text._lseek_r+0x10): undefined reference to `_lseek'
 readr.c:(.text._read_r+0x10): undefined reference to `_read'
 fstatr.c:(.text._fstat_r+0xe): undefined reference to `_fstat'
 isattyr.c:(.text._isatty_r+0xc): undefined reference to `_isatty'
```

Listing 4. Hinzufügen einfacher Stubs.

```
int _fstat(int fd, struct stat *st) {
   (void) fd, (void) st;
   return -1;
 }

void *_sbrk(int incr) {
   (void) incr;
   return NULL;
 }

int _close(int fd) {
   (void) fd;
   return -1;
 }

int _isatty(int fd) {
   (void) fd;
   return 1;
 }

int _read(int fd, char *ptr, int len) {
   (void) fd, (void) ptr, (void) len;
   return -1;
 }

int _lseek(int fd, int ptr, int dir) {
   (void) fd, (void) ptr, (void) dir;
   return 0;
 }
```

debuggen. Den kompletten Quellcode des Projekts finden Sie im Ordner *step-4-printf* [11].

Debuggen mit Segger Ozone

Was ist, wenn unsere Firmware irgendwo feststeckt und `printf()`-Debuggen nicht funktioniert? Was ist, wenn sogar der Startup-Code nicht funktioniert? Dann brauchen wir einen Debugger. Es gibt viele Möglichkeiten, aber ich würde den Debugger Ozone von Segger empfehlen. Warum? Weil er eigenständig ist und keine eingerichtete IDE benötigt. Wir können Ozone direkt mit unserer Datei *firmware.elf* füttern und Ozone wird unsere Quelldateien einsammeln.

Laden Sie also Ozone von der Segger-Website herunter [12]. Bevor wir es mit unserem Nucleo-Board verwenden können, müssen wir nur noch die ST-LINK-Firmware auf dem Onboard-Debugger in die *jlink*-Firmware konvertieren, die Ozone versteht. Folgen Sie dann den Anweisungen auf der Segger-Website [13].

- Starten Sie Ozone. Wählen Sie unser *Device* im Assistenten (**Bild 2**).
- Wählen Sie einen Debugger, den wir verwenden wollen - das sollte ein ST-LINK sein (**Bild 3**).
- Geben Sie den Pfad zu unserer Datei *firmware.elf* an (**Bild 4**).
- Belassen Sie die Standardeinstellungen auf dem nächsten Screen und klicken Sie auf *Finish*. Schon ist unser Debugger geladen (beachten Sie, dass der *mcu.h*-Quellcode übernommen wird), siehe **Bild 5**.
- Klicken Sie auf den grünen Button zum Herunterladen und führen Sie die Firmware aus. Damit sind wir hier fertig (**Bild 6**).

Bild 2. Wählen Sie das Gerät im Assistenten aus.

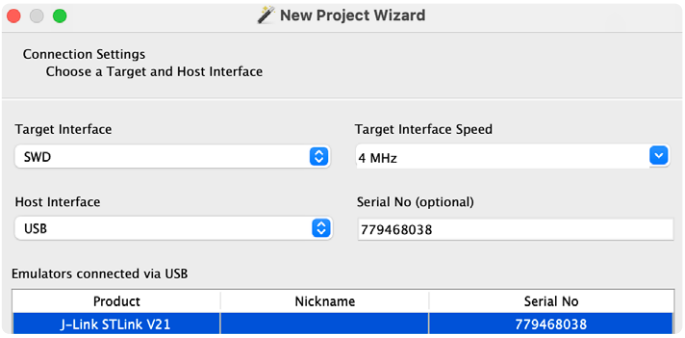


Bild 3. Wählen Sie STLink als Debugger.

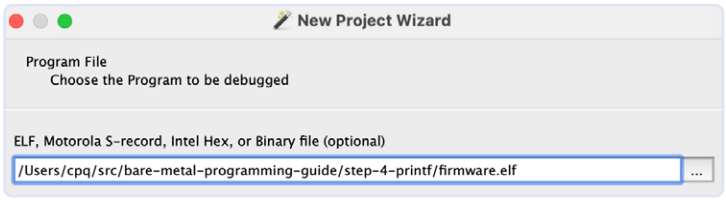


Bild 4. Das zu debuggende Programm ist unsere Datei firmware.elf.

Jetzt können wir in Einzelschritten durch den Code gehen, Haltepunkte setzen und die üblichen Debugging-Maßnahmen durchführen. Besonders hervorzuheben ist die praktische Ozone-Peripherie-Ansicht (**Bild 7**). Mit ihr können wir den Zustand der Peripherieeinheiten direkt untersuchen oder einstellen. Als Beispiel schalten wir die grüne On-Board-LED an PB0 ein:

1. Wir müssen zuerst GPIOB takten. Suchen Sie *Peripherals|RCC|AHB1ENR* und setzen Sie das *GPIOBEN*-Bit (**Bild 8**).
2. Finden Sie *Peripherals|GPIO|GPIOB|MODER* und setzen Sie *MODER0* auf „01“ (Ausgang) (**Bild 9**).
3. Suchen Sie *Peripherals|GPIO|GPIOB|ODR* und setzen Sie *ODR0* auf „1“ (ein) (**Bild 10**).

Jetzt sollte eine grüne LED leuchten! Viel Spaß beim Debuggen!

Im dritten Teil dieser Artikelreihe werden wir einen Webserver implementieren. Außerdem werden wir zeigen, wie ein Programm automatisch getestet werden kann, und vieles mehr. Bleiben Sie dran!

RG — 220665-B-02

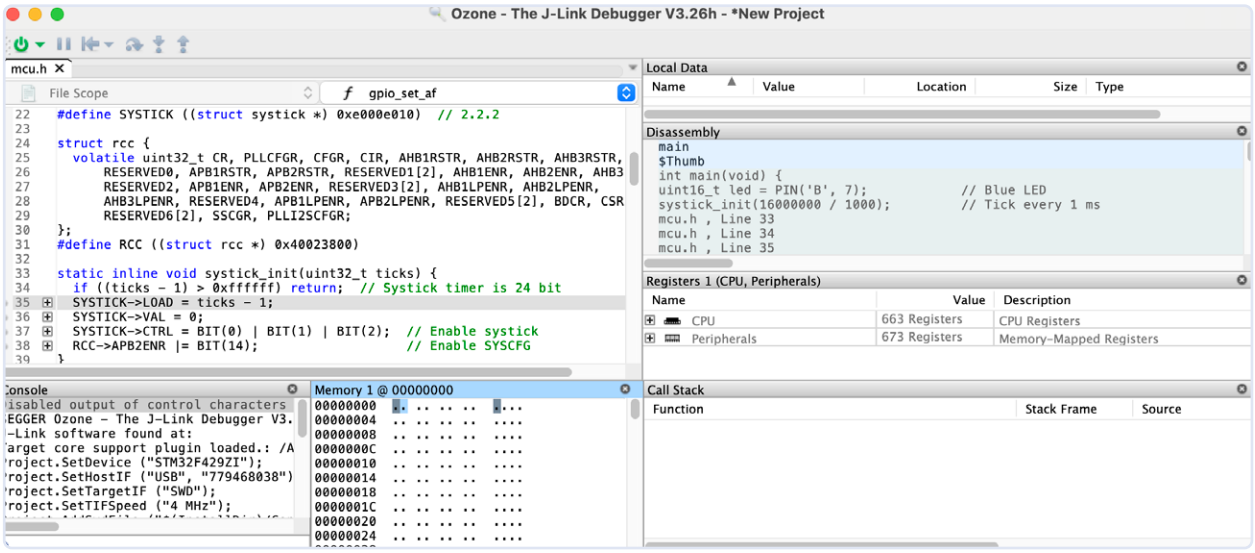


Bild 5. Der Debugger wird geladen, und bald erscheint mcu.h.

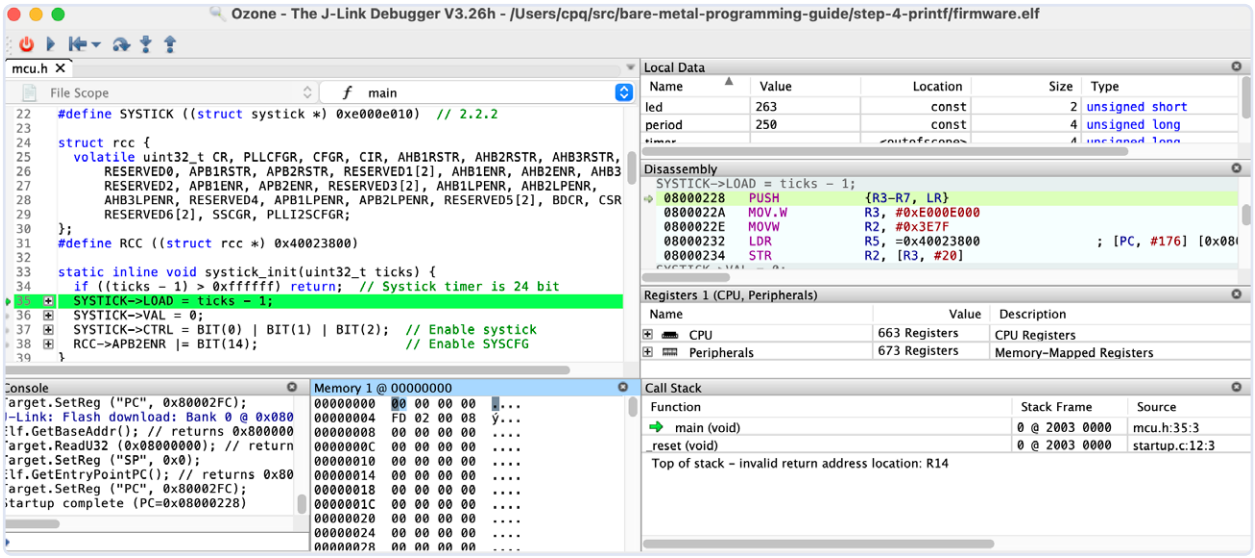


Bild 6. Nachdem wir die Firmware gestartet haben, wird sie in der Zeile SYSTICK->LOAD = ticks - 1; angehalten.

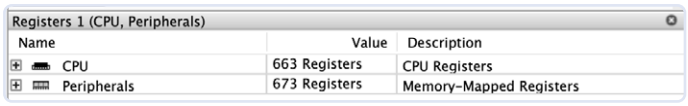


Bild 7. Praktische Peripherals-Ansicht in Ozone zur einfachen Prüfung und Konfiguration der Peripherals.

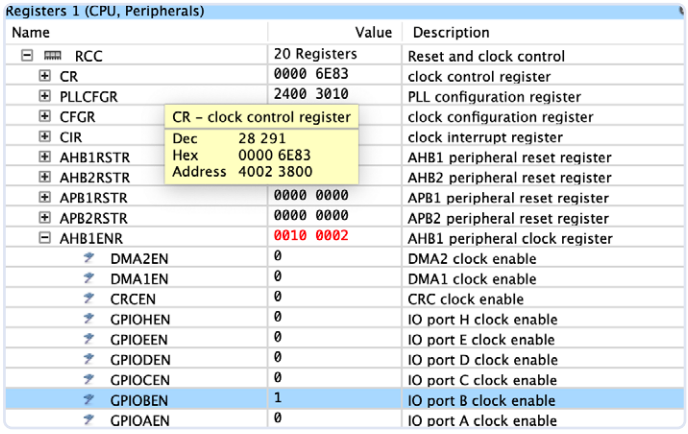


Bild 8. Aktivieren des Takts an Port B durch Setzen des Wertes von GPIOBEN auf 1.

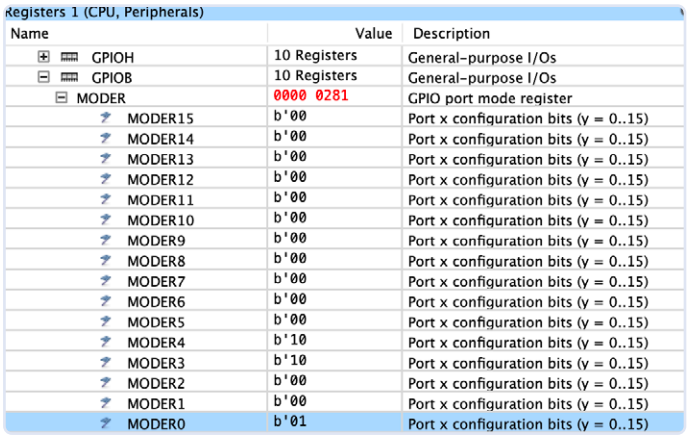


Bild 9. Setzen von MODER0 auf 1 (und damit Auswahl des Ausgangs) in den GPIO-Peripherals.

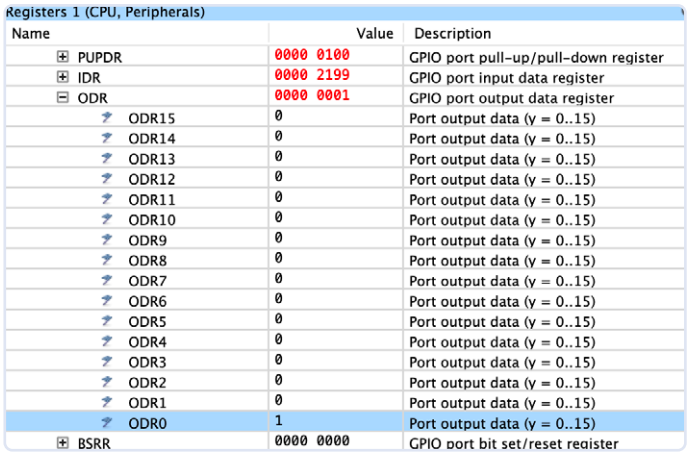


Bild 10. Einschalten von ODR0 durch Auswahl des Werts 1 in ODR (GPIO).

Haben Sie Fragen oder Kommentare?

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

Über den Autor

Sergey Lyubka ist ein Ingenieur und Unternehmer. Er hat einen MSc in Physik von der Staatlichen Universität Kyjiw, Ukraine. Sergey ist Direktor und Mitbegründer von Cesanta, einem Technologieunternehmen mit Sitz in Dublin, Irland (Embedded Web Server for electronic devices: https://mongoose.ws). Seine Leidenschaft ist die Programmierung von eingebetteten Bare-Metal-Netzwerken.

Passende Produkte

- **Dogan Ibrahim, *Nucleo Boards Programming with the STM32CubeIDE*, Elektor**
  https://elektor.de/19530
- **Dogan Ibrahim, *Programming with STM32 Nucleo Boards*, Elektor**
  https://elektor.de/18585

WEBLINKS

[1] GitHub-Repository für diesen Artikel: https://github.com/cpq/bare-metal-programming-guide
[2] Sergey Lyubka, „Anleitung zur Bare-Metal-Programmierung (Teil 1)“, Elektor 7-8/2023: https://elektormagazine.de/220665-02
[3] Arm v7-M Architecture Reference Manual: https://developer.arm.com/documentation/ddi0403/ee
[4] Reference Manual RM0090 für STM32F429: https://bit.ly/3neE7S7
[5] Ordner Step 2 SysTick: https://github.com/cpq/bare-metal-programming-guide/tree/main/steps/step-2-systick
[6] Benutzerhandbuch Nucleo-144 Board (UM1974): https://bit.ly/3oIBXKZ
[7] Datenblatt STM32F429ZI: https://st.com/resource/en/datasheet/stm32f429zi.pdf
[8] PuTTY: https://putty.org
[9] Ordner Step 3 UART: https://github.com/cpq/bare-metal-programming-guide/tree/main/steps/step-3-uart
[10] C-Bibliothek newlib: https://sourceware.org/newlib
[11] Ordner Step 4 printf: https://github.com/cpq/bare-metal-programming-guide/tree/main/steps/step-4-printf
[12] Ozone — The J-Link Debugger and Performance Analyzer: https://segger.com/products/development-tools/ozone-j-link-debugger
[13] ST-LINK On-Board in J-Link verwandeln: https://segger.com/products/debug-probes/j-link/models/other-j-links/st-link-on-board

elektorstore

www.elektor.de

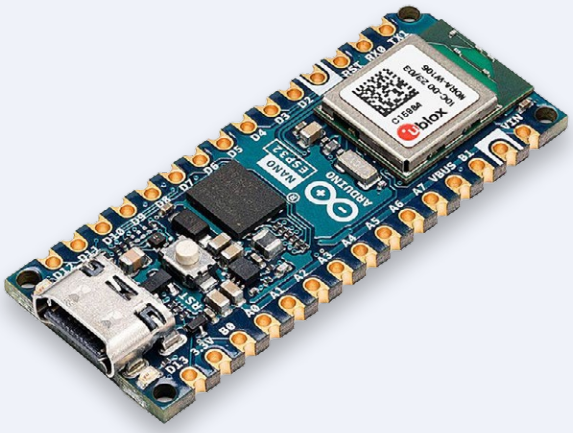
Der Elektor Store

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).

Arduino Nano ESP32



Der Arduino Nano ESP32 (mit und ohne Header) ist ein Nano-Formfaktor-Board, das auf dem ESP32-S3 (eingebettet im NORA-W106-10B von u-blox) basiert. Es ist das erste Arduino-Board, das vollständig auf einem ESP32 basiert. Es bietet Wi-Fi, Bluetooth LE, Debugging über natives USB in der Arduino-IDE sowie einen geringen Stromverbrauch.

Preis: 23,95 €

Mitgliederpreis: 21,56 €

www.elektor.de/20562

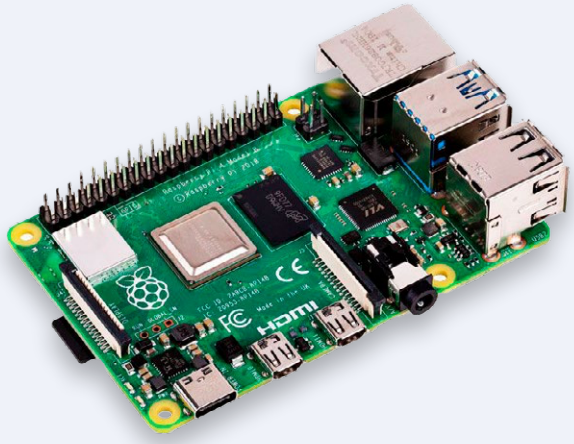
Logic Analyzer im Einsatz

Schritt-für-Schritt Anleitungen führen Sie in die Analyse moderner Protokolle von I²C, SPI, UART, RS-232, NeoPixel, WS28xx, HD44780 und 1-Wire Protokollen ein. Anhand von zahlreichen Experimentierschaltungen mit dem Raspberry Pi Pico, Arduino Uno und dem Bus Pirate üben Sie die praxisnahe Anwendung gängiger USB-Logikanalysatoren ein.

Preis: 32,80 €

www.elektor.de/20548

Raspberry Pi 4 (4 GB) Offizielles Starterkit



Preis: 104,95 €

www.elektor.de/20556

QuantAsylum QA403 24-bit Audio Analyzer

Preis: 799,00 €

Mitgliederpreis: 719,10 €

www.elektor.de/20530

FNIRSI DSO-TC3 (3-in-1) Oszilloskop, Komponententester & Signalgenerator

Preis: ~~74,95 €~~

Sonderpreis: 59,95 €

www.elektor.de/20520

ESP-Terminal (ESP32-S3 basiertes Dev-Board mit 3,5“ Display)

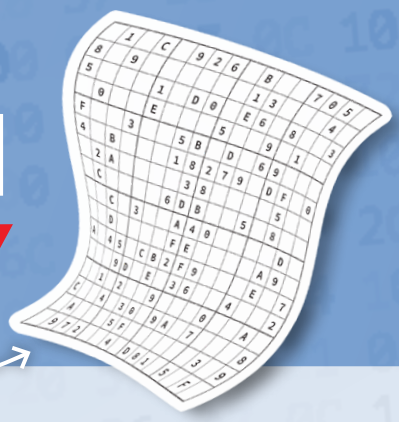
Preis: 44,95 €

Mitgliederpreis: 40,46 €

www.elektor.de/20526

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. Oktober 2023.

DIE GEWINNER DES HEXADOKUS AUS DER AUSGABE JULI/AUGUST STEHEN FEST!

Die richtige Lösung ist: **23BDF**.

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!

		E	3	F	B		5	8		2	6	4	0		
B					E	A			5	9					3
4					2	0	9	E	7	B					D
	7													5	
9		B	E	7	4					8	1	3	2		C
		4	6	9		C	A	0	B		7	D	F		
7		A	D	8		B			4		2	5	6		1
5	8			D	0	2			3	6	A			B	4
6	3			B	5	7			E	A	C			8	0
8		0	1	E		3			6		5	9	B		A
		D	B	4		6	2	3	F		9	C	7		
C		2	7	0	A					4	D	F	3		5
	9													7	
E					9	4	3	5	2	C					6
0					7	F			A	D					9
		8	4	5	C		B	7		F	0	A	D		

D	8	1	6	0	E	2	7	4	A	5	9	3	F	C	B
E	3	4	C	F	A	B	9	0	1	8	2	7	6	D	5
5	2	F	9	1	C	6	4	3	B	7	D	8	E	0	A
A	0	B	7	D	3	5	8	E	6	C	F	1	2	4	9
3	B	7	D	4	6	8	1	9	C	F	5	E	0	A	2
F	4	E	1	2	B	A	5	6	D	0	7	9	C	8	3
6	C	2	5	3	7	9	0	A	8	4	E	B	D	F	1
0	9	A	8	C	F	D	E	B	2	1	3	4	5	6	7
7	D	6	A	8	4	F	C	5	E	3	1	2	B	9	0
2	E	8	B	5	9	0	D	7	4	A	C	6	1	3	F
9	F	C	3	E	1	7	6	8	0	2	B	A	4	5	D
1	5	0	4	A	2	3	B	D	F	9	6	C	7	E	8
B	6	D	F	9	8	1	3	C	7	E	0	5	A	2	4
C	1	3	0	B	5	4	2	F	9	6	A	D	8	7	E
4	7	5	2	6	0	E	A	1	3	D	8	F	9	B	C
8	A	9	E	7	D	C	F	2	5	B	4	0	3	1	6

TME
Electronic Components

TRANSFER
MULTISORT
ELEKTRONIK

GLOBAL DISTRIBUTOR OF ELECTRONIC COMPONENTS

ECAD-Modell bei TME verwenden

- PCB-Symbole und -Muster
- 3D-Modelle
- Schneller Zugriff

EINE PERFEKTE LÖSUNG FÜR DESIGNER

TME Germany GmbH
Leipzig, tme@tme-germany.de

Schließen Sie sich uns an:

tme.eu

YOU NEED IT, WE HAVE IT!

tme.com

Stellen Sie sich vor, die Brüder Wright hätten Mouser als Partner gehabt

Mit der größten Auswahl der neuesten Produkte™
sind innovativen Designs keine Grenzen gesetzt.

mouser.de/new
+49 (0) 89 520 462 110

MOUSER ELECTRONICS