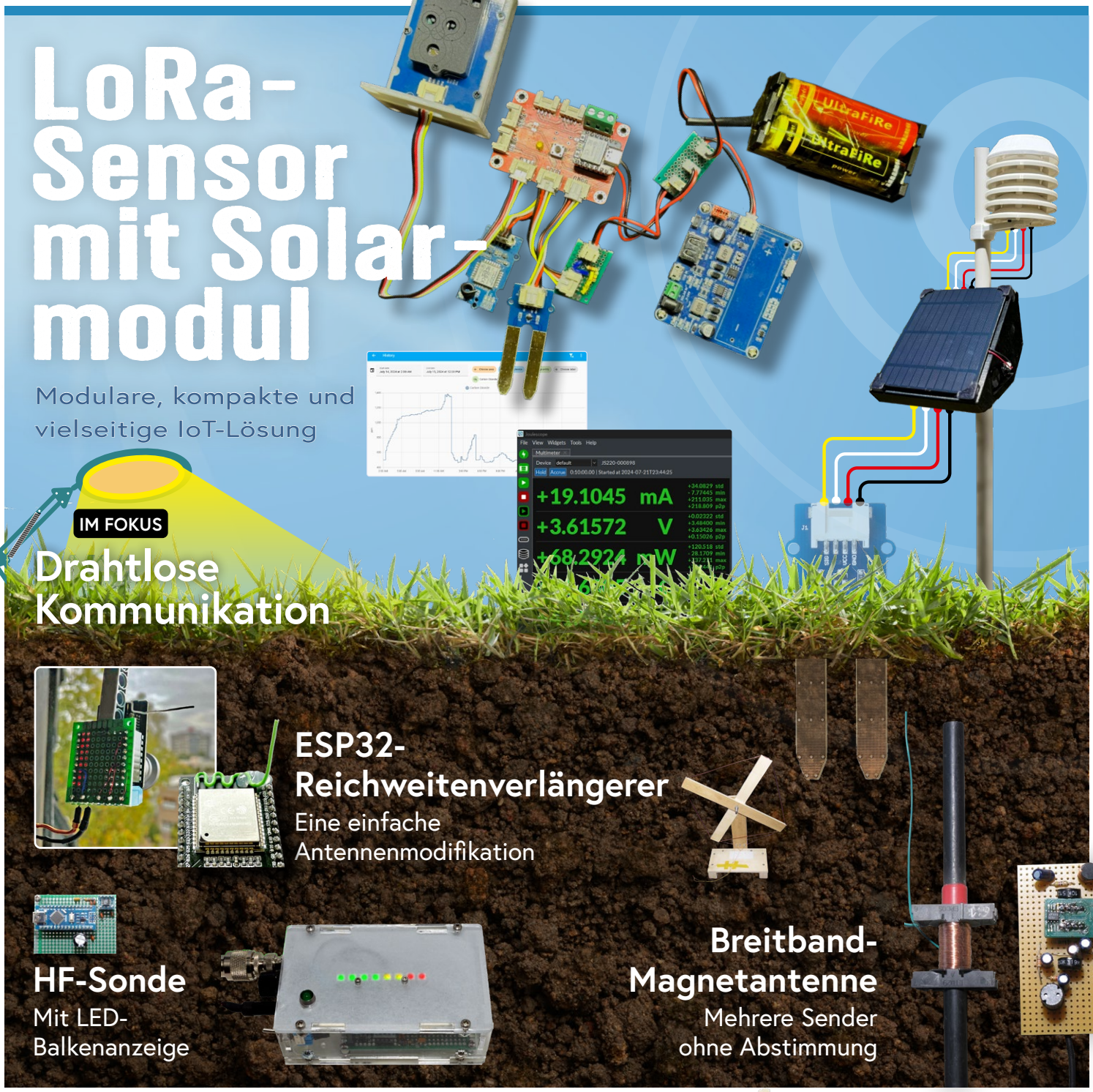
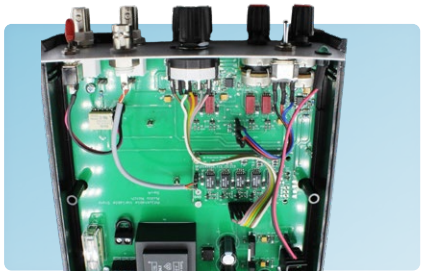
G3078E (D) € 14,90 | CHF 27,00 | (A, B, L) € 16,00

elektor MAG

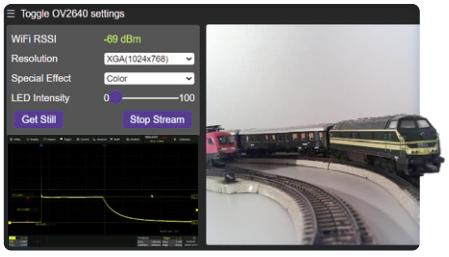
design > share > earn

604
* SEIT 1961 *
SEPTEMBER/OKTOBER 2024
ELEKTORMAGAZINE.DE

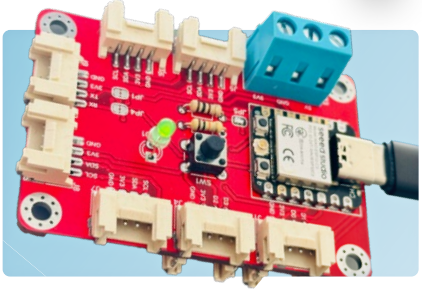




Audio-Notchfilter
Mit einstellbarer Frequenz



Modelleisenbahn mit Kamera
Einbau eines ESP32-CAM-Moduls



Elektor eXpansion Board
Für ESP32-S3 und andere XIAO-Boards

4 198630314904 40010

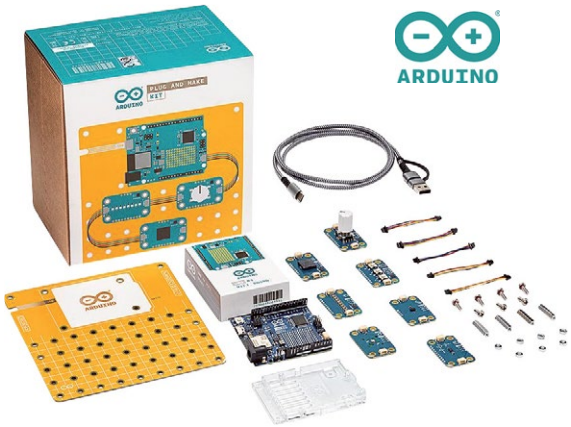
UNSER SORTIMENT VON MAKERN FÜR MAKER

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

ARDUINO OFFICIAL RESELLER 2023

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 und Entwicklung, Instandhaltung, IT-Infrastruktur und Kleinserienproduktion sowie auf Maker zugeschnitten.



ARDUINO

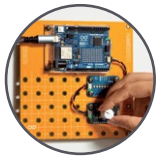
ARDUINO – Die vielseitige Open-Source-Plattform für Elektronik-Prototyping & interaktive Projekte

Arduino Plug and Make Kit

inklusive Arduino Uno R4 WiFi, Modulino® Sensoren & Aktoren, USB-C-Kabel uvm.

Entfalten Sie Ihre Kreativität mit dem Arduino Plug-and-Make-Kit. Es ist Ihr perfekter Ausgangspunkt und bietet alles, was Sie benötigen, um das Arduino-Ökosystem zu erkunden

- kein Breadboard und kein Löten notwendig



NEU

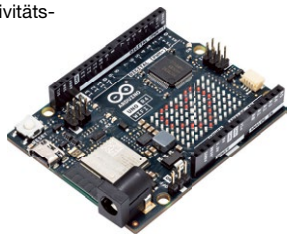
Bestell-Nr.:
ARD AKX00069

94,90 (79,75)

Arduino UNO R4 WiFi

Der Arduino UNO R4 WiFi kombiniert die Rechenleistung und die aufregenden neuen Peripherien des RA4M1-Mikrocontrollers von Renesas mit der drahtlosen Konnektivitätsleistung des ESP32-S3 von Espressif.

- Renesas RA4M1 (Cortex M4)
- WiFi/BT LE (ESP32-S3)



BESTSELLER

Bestell-Nr.:
ARD UNO R4 WIFI **24,80** (20,84)

Arduino Portenta HAT Carrier

Erstellen Sie Prototypen wie ein Profi, automatisieren Sie Ihr Projekt oder erhöhen Sie die Messlatte Bereich Robotik. Portenta Hat Carrier macht Hochleistung für Sie einfach!

- ergänzen Sie Ihre Portenta-Projekte mit Raspberry Pi® Hats
- 16-pol. Header, Gigabit Ethernet, microSD, USB-A

Bestell-Nr.:
ARD PORTENTA HAT **45,50** (38,24)

reichelt elektronik MAGAZIN

Arduino im reichelt Magazin:
Arduino Projekte, Ratgeber und How-tos - legen Sie los!

Jetzt lesen ▶
https://rch.lt/m-arduino

Arduino - Leistungsfähige Mikrocontroller für Schalt- und Steuerungsaufgaben

Ideal für alle, die interaktive Objekte oder Umgebungen programmieren wollen.

Gleich entdecken ▶
www.reichelt.de/arduino

ARDUINO

- Top Preis-Leistungs-Verhältnis
- über 150.000 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

Es gelten die gesetzlichen Widerrufsregelungen. Alle angegebenen Preise in € inklusive der gesetzlichen MwSt., zzgl. Versandspesen für den gesamten Warenkorb. Es gelten ausschließlich unsere AGB (unter www.reichelt.de/agb, im Katalog oder auf Anforderung). Abbildungen ähnlich. Druckfehler, Irrtümer und Preisänderungen vorbehalten. reichelt elektronik GmbH, Elektronikring 1, 26452 Sande, Tel.:+49 (0)4422 955-333

TAGESPREISE! Preisstand: 20. 8. 2024

55. Jahrgang, Nr. 604
September/Oktober 2024
ISSN 0932-5468

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

Für alle Ihre Fragen: service@elektor.de

Mitglied werden: www.elektormagazine.de/abo

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

Urheberrecht
© Elektor International Media b.v. 2024

Die in dieser Zeitschrift veröffentlichten Beiträge, insbesondere alle Aufsätze und Artikel sowie alle Entwürfe, Pläne, Zeichnungen einschließlich Platinen sind urheberrechtlich geschützt. Ihre auch teilweise Vervielfältigung und Verbreitung ist grundsätzlich nur mit vorheriger schriftlicher Zustimmung des Herausgebers gestattet. Die veröffentlichten Schaltungen können unter Patent- oder Gebrauchsmusterschutz stehen. Herstellen, Feilhalten, Inverkehrbringen und gewerblicher Gebrauch der Beiträge sind nur mit Zustimmung des Verlages und ggf. des Schutzrechtsinhabers zulässig. Nur der private Gebrauch ist frei. Bei den benutzten Warenbezeichnungen kann es sich um geschützte Warenzeichen handeln, die nur mit Zustimmung ihrer Inhaber warenzeichengemäß benutzt werden dürfen. Die geltenden gesetzlichen Bestimmungen hinsichtlich Bau, Erwerb und Betrieb von Sende- und Empfangseinrichtungen und der elektrischen Sicherheit sind unbedingt zu beachten. Eine Haftung des Herausgebers für die Richtigkeit und Brauchbarkeit der veröffentlichten Schaltungen und sonstigen Anordnungen sowie für die Richtigkeit des technischen Inhalts der veröffentlichten Aufsätze und sonstigen Beiträge ist ausgeschlossen.

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

DEUTSCHE FACHPRESSE

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

PEFC Certified
This product is from sustainably managed forests and controlled sources
PEFC
PEFC/30-31-151
www.pefc.org

Jens Nickel

Chefredakteur ElektorMag

Das drahtlose Jahrhundert

Sicher stimmen Sie mir zu: Das 20. Jahrhundert war ein Jahrhundert des elektrischen Kabels, und dieses Jahrhundert wird eines der Drahtlosigkeit sein. Die Vorteile der leitungslosen Kommunikation sind überwältigend. Sie erlaubt hochmobile Geräte und macht Anwendungen wie das IoT überhaupt erst möglich.
Vor allem weil eine zuverlässige und abhörsichere Übertragung sichergestellt werden muss, ist die drahtlose Kommunikation freilich komplexer und daher auch mit einer größeren Verzögerung behaftet. Alleine die Aufgabe, Teilnehmer zeitlich zu synchronisieren, ist zumindest bei einer digitalen Übertragung (Bluetooth, WiFi, ...) alles andere als trivial - bei einer Kabelverbindung wäre dies eine Aufgabe für Elektronikanfänger. Das macht Anwendungen, die minimale Verzögerungen oder zumindest eine abgestimmte Verzögerung auf verschiedenen Kanälen erfordern, per se schwierig. Noch immer wartet die Elektronikgemeinde auf den Markteintritt von Bluetooth LE Audio, das endlich eine synchrone Übertragung des linken und rechten Kanals eines Stereosignals erlaubt. Mit einem unserer regelmäßigen Autoren hatte ich kurz überlegt, ob wir einen Hands-on-Artikel zum Thema erstellen. Doch die ersten Entwicklerboards sind noch teuer und der Umgang mit der Software sperrig. Deswegen haben wir den Artikel noch einmal für eine spätere Ausgabe zurückgestellt.
In diesem Heft berichten wir aber über andere spannende Drahtlos-Technologien, die Sie sofort in eigenen Projekten einsetzen können. Zum Beispiel den stromsparenden Datenfunk LoRa. Auf den Seiten 6 und 94 finden Sie zwei Projekte, bei denen autarke Sensorknoten Messwerte aufnehmen und per LoRa übermitteln. Die Artikel ergänzen sich gut. Bei unserem Autor Claus Kühnel steht mehr die Stromversorgung und deren Messung im Vordergrund (Seite 94). Mein Kollege Saad geht in seinem LoRa-Artikel (Seite 6) mehr darauf ein, wie die Daten in die Cloud kommen, wo man sie visualisieren und abrufen kann. Mir gefällt bei beiden Projekten, wie fertige Module gewinnbringend eingesetzt werden, was viel Entwicklungszeit spart.
Wie immer gibt es zu unserem gedruckten Heft auch ein Bonusheft mit weiteren Projekten, abzurufen auf unserer Wireless-Themenseite (*www.elektormagazine.de/drahtlose-kommunikation*). Hier beschreibt unser Autor Sebastian Westerhold zum Beispiel, wie man ein günstiges LNB für den Empfang von TV-Sendern für Amateurfunkzwecke umrüsten kann. Unbedingt reinschauen!

Mehr über Drahtlos-Technologie

Wenn Sie ein besonderes Interesse an Wireless-Technologien und -Anwendungen haben, besuchen Sie unsere Wireless-Themenseite, auf der Sie das ganze Jahr über aktuelle Nachrichten und exklusive Artikel finden:
www.elektormagazine.de/drahtlose-kommunikation

Unser Team

Chefredakteur: Jens Nickel (v.i.S.d.P.) | **Redaktion:** Asma Adhimi, Roberto Armani, Eric Bogers, Jan Buiting, Stuart Cording, Rolf Gerstendorf (RG), Ton Giesberts, Ouafae Hassani, Hedwig Hennekens, Saad Imtiaz, Alina Neacsu, Dr. Thomas Scherer, Jean-Francois Simon, Clemens Valens, Brian Tristam Williams | **Regelmäßige Autoren:** David Ashton, Tam Hanna, Ilse Joostens, Prof. Dr. Martin Ossmann, Alfred Rosenkränzer | **Grafik & Layout:** Harmen Heida, Sylvia Sopamena, Patrick Wielders | **Herausgeber:** Erik Jansen | **Technische Fragen:** redaktion@elektor.de

INHALT

55. Jahrgang - Nr. 604
September/Oktober 2024

Rubriken

3 Impressum

48 Aller Anfang...
muss nicht schwer sein: mehr Verstärkerschaltungen

58 Aus dem Leben gegriffen
Über die Selbstverständlichkeit der Dinge

109 Projekt 2.0
Korrekturen, Updates und Leserbriefe

112 2024: Eine Odyssee in die KI
Beschleuniger: Desktop vs. Embedded, ein Blick auf einige Optionen

BONUS-INHALT

Lesen Sie die kostenlose Elektor-Bonus-Ausgabe *Drahtlose Kommunikation*!

- Projekt: Modifizierung von Satelliten-LNBs für 10-GHz-Experimente
- Review: Der ultra-portable Dekoder-Empfänger ATS25 max
- Projekt: ArtNet-zu-DMX-Konverter mit ESP32
- Infografik: Drahtlose Kommunikation

www.elektormagazine.de/drahtlose-kommunikation

Hintergrund

32 TensorFlow Lite auf kleinen Mikrocontrollern
Ein (sehr) einsteigerfreundlicher Blickwinkel

44 Elektor Bücher
HF-Sonde mit LED-Balkenanzeige

86 Das System LeoINAGPS
Nützliche Einblicke in Ihr Elektrofahrzeug

100 AWS für Arduino und Co.
Teil 2: Daten senden mit AWS-IoT-ExpressLink

Projekte

6 Ein autonomer Sensorknoten
LoRa-Datenübertragung und Solar-Stromversorgung

18 Elektor eXpansion Board v1.0
Für ESP32-S3 und andere XIAO-Controller-Boards

23 Modelleisenbahn mit Kamera
Einbau eines ESP32-CAM-Moduls

28 Breitband-Magnetantenne für Langwelle
Mehrere Sender ohne Abstimmung

38 Ein Hub für RS-422 und RS-485
Verdrahten Sie Ihre Geräte wie einen Stern

52 Open Vario
Ein Open-Source-Multifunktions-Variometer für Paraglider

60 KI-gestützter Wasserzähler
Teil 2: Firmware-Installation, Anwendung und Montage

81 Audio-Notchfilter mit einstellbarer Frequenz
Universelle Lösung zur Unterdrückung von Frequenzen im Audiobereich

94 Solarbetriebener LoRa-Knoten
Eine modulare, kompakte und vielseitige IoT-Lösung

118 ESP32-Reichweitenverlängerer
Eine einfache Antennenmodifikation

Tauchen Sie ein in die Drahtlose Kommunikation

Besuchen Sie die Elektor-Webseite *Drahtlose Kommunikation* für mehr Projekte, Videos und Tutorials!
www.elektormagazine.de/drahtlose-kommunikation

Industry

68 Intelligente Landwirtschaft
ML-basierte Schädlingserkennung mit IoT-Vernetzung

72 Warum Anybus CompactCom die ideale Wahl für die industrielle Embedded-Kommunikation ist

75 Kommunikationsstandard IQRF
Zuverlässigkeit für verlustbehaftete drahtlose Mesh-Netzwerke

78 Bau eines intelligenten Agrarroboters
Wesentliche technische Überlegungen und Herausforderungen

Vorschau

Elektor November/Dezember 2024
Das nächste Heft ist wie immer randvoll gefüllt mit Schaltungsprojekten, Grundlagen sowie Tipps und Tricks für Elektroniker. Schwerpunkt wird das Thema **Produktion und Bauteile** sein.

- Platinenlayout und Sicherheit
- Elektor Audio-DSP FX-Prozessor-Platine
- Stand-alone Quarz-Tester
- Multirechner für Elektronik
- KiCad 8: Die neuen Funktionen
- ESP32-Projekte mit PlatformIO
- Opamp-Tester für Audio- und andere Anwendungen
- 3D-Weihnachtsbaum-Platine
- LED-Anzeigen mit dem MAX7219

Elektor November/Dezember 2024
erscheint am 13. November 2024
Änderungen vorbehalten!

Ein autonomer Sensorknoten

LoRa-Datenübertragung und Solar-Stromversorgung

Von Saad Imtiaz (Elektor)

Dieser Sensorknoten basiert auf energieeffizienter LoRa-Übertragung über große Distanzen und kann für die Fernüberwachung von Umweltdaten eingesetzt werden. An dem Elektor eXpansion Board mit dem ESP32-C3 XIAO-Controller sind ein LoRa-WIO-E5-Modul, ein CO2-Sensor des Typs SCD30 und ein Bodenfeuchtigkeitssensor angeschlossen. All dies wird von einem Solarpanel äußerst effizient mit Energie versorgt. Die Daten werden an The Things Network (TTN) gesendet und können über Datacake oder Home Assistant abgerufen und für einen schnellen Überblick in Echtzeit und/oder für Automatisierungsaufgaben verwendet werden. Dieser Artikel beschreibt das gesamte System, von der Einrichtung Ihres LoRaWAN-Gateways bis hin zur nahtlosen Integration mit TTN, Datacake und Home Assistant.

Source: Adobe Stock

Bild 1. Der solarbetriebene LoRaWAN-Sensorknoten.

Die Fernüberwachung von Umweltdaten ist für die Bewältigung der Herausforderungen des Klimawandels und des Ressourcenmanagements von entscheidender Bedeutung. Viele der auf dem Markt erhältlichen LoRa-Sensorknoten sind jedoch teuer und nicht anpassbar, was sie für bestimmte Anwendungen und abgelegene Einsatzgebiete unpraktisch macht. Um diese Einschränkungen zu überwinden, wird in diesem Projekt ein vielseitiger, kostengünstiger LoRa-Sensorknoten vorgestellt. Dieses maßgeschneiderte System bietet eine effiziente Lösung für die Datenerfassung aus der Ferne, indem es LoRa-Kommunikation mit großer Reichweite und Solarenergie nutzt, um einen kontinuierlichen Betrieb zu gewährleisten. Durch die Aufnahme anpassbarer Sensoren ist das System flexibel und für verschiedene Anforderungen der Umweltüberwachung geeignet.

Eines der Merkmale dieses Projekts ist die Einbindung in die Open-Source-Automatisierungsplattform Home Assistant, die leistungsstarke Automatisierungsfunktionen ermöglicht. Durch Messung der Bodenfeuchtigkeit kann man damit beispielsweise die Gartenbewässerung automatisieren, was die Besitzer von der Last der täglichen manuellen Bewässerung befreien kann.

In diesem Artikel werden wir die Einrichtung dieses LoRaWAN-Sensorknotens wie in **Bild 1**, die Konfiguration eines LoRaWAN-Gateways und die Integration mit The Things Network (TTN) kennenlernen und uns dann auf eine spannende Reise begeben, auf der wir die Daten auf Plattformen wie Datacake und Home Assistant visualisieren. Dieses Projekt ist der „lebendige" Beweis dafür, dass erschwingliche und anpassungsfähige Technik umfassende Überwachungslösungen bieten kann, die die Unzulänglichkeiten der bestehenden kommerziellen Optionen ausbügeln und den Weg für vielfältige(re) Anwendungen ebnen.

System-Übersicht

Der Kern dieses Projekts ist das Mikrocontroller-Board XIAO ESP32-C3 von Seeed Studio. Sie können dieses Board und andere Module, die für dieses Projekt verwendet werden, im Elektor Store finden (siehe **Passende Produkte**.) Die Wahl des XIAO-Boards fiel aufgrund seiner kompakten Größe und der ausreichenden I/O-Funktionen, die für dieses Projekt erforderlich sind. Um die I/O-Optionen zu erweitern, wurde als Träger das Elektor eXpansion Board [1] verwendet, das die Funktionalität des Mikrocontrollers um sechs I²C-Anschlüsse erweitert.

Was sind LoRa, LoRaWAN, Gateway und TTN?

LoRa: Long Range ist eine Drahtlos-Kommunikationstechnik, die für die Übertragung von Daten über große Entfernungen bei minimaler Stromaufnahme entwickelt wurde. Sie arbeitet in Sub-Gigahertz-Frequenzbändern (868 MHz in Europa, 915 MHz in Nordamerika) und kann Daten bis zu 15 km in offenen ländlichen Gebieten und 5 km in städtischen Umgebungen übertragen, was sie ideal für IoT-Anwendungen wie Umweltüberwachung und intelligente Landwirtschaft macht.

LoRaWAN: Das Long Range Wide Area Network ist ein Netzwerkprotokoll, das auf der LoRa-Technologie basiert. Es verwaltet die Kommunikation zwischen LoRa-Geräten und Gateways und unterstützt großflächige IoT-Implementierungen mit sicherer, bidirektionaler Kommunikation, anpassbaren Datenraten und effizienter Netzwerknutzung.

Gateway: Ein LoRaWAN-Gateway verbindet LoRa-Geräte mit dem Internet. Es empfängt Daten von LoRa-Geräten und leitet diese über WLAN, Ethernet oder Mobilfunk an einen zentralen Server weiter. Gateways decken typischerweise mehrere Kilometer ab und können mehrere simultane Sendungen von Geräten verarbeiten.

TTN: The Things Network ist ein globales Open-Source-LoRaWAN-Netzwerk. Es bietet die Infrastruktur und die Werkzeuge zur Verbindung von LoRaWAN-Geräten, einschließlich Geräte-Registrierung, Datenweiterleitung und Einbindung in Plattformen wie Datacake und Home Assistant. TTN wird von einer globalen Gemeinschaft unterstützt und ist eine ausgezeichnete Ressource für die Bereitstellung von IoT-Lösungen mit LoRaWAN.

Obwohl in diesem Projekt nur einer dieser I²C-Anschlüsse benötigt wird, ermöglicht die anpassbare Natur dieses Sensorknotens den Einsatz zusätzlicher Sensoren. In **Bild 2** ist das Blockdiagramm des Projekts zu sehen.

Für die LoRa-Kommunikation wurde das einfach einzurichtende und preiswerte LoRa-Modul E5 WIO von Seeed Studio gewählt, das vom Host-Controller über eine UART-Verbindung und AT-Befehle zur Datenübertragung und Konfiguration gesteuert werden kann. Das Modul erlaubt eine zuverlässige Datenübertragung über lange Strecken. In meinen Tests konnte ich eine Reichweite von etwa 700 m in einer städtischen Umgebung mit einigen Bäumen und Häusern dazwischen erreichen. Da sich mein Gateway in einem Innenraum neben einem Fenster befand, ist dies eine gute Reichweite, da wie bei jedem anderen drahtlosen Kommunikationssystem die Sichtlinie eine große Rolle spielt. Die Reichweite kann erheblich verbessert werden, wenn das Gateway mit einer hohen Antenne ausgestattet wird. Auch eine bessere Antenne auf dem Modul selbst könnte die Effizienz weiter steigern.

Das Solarladesystem wird von einem 3-W-Solarmodul, ebenfalls von Seeed Studio gespeist, das an ein *Solar Power Management Modul*e von Waveshare angeschlossen ist, das im Grunde ein MPPT-Tracker mit einigen Schutzschaltungen ist. Dieses System lädt zwei 18650er Akkus parallel auf. Der MPPT-Tracker (Maximum Power Point Tracking) spielt eine entscheidende Rolle bei der Optimierung der Leistungsabgabe des Solarmoduls. Er passt den elektrischen Arbeitspunkt des Solarmoduls kontinuierlich an und sorgt dafür, dass es mit maximaler Effizienz arbeitet. Nach dem Test mehrerer MPPT-Tracker erwies sich das Waveshare-Modul als das effizienteste und zuverlässigste für diese Anwendung.

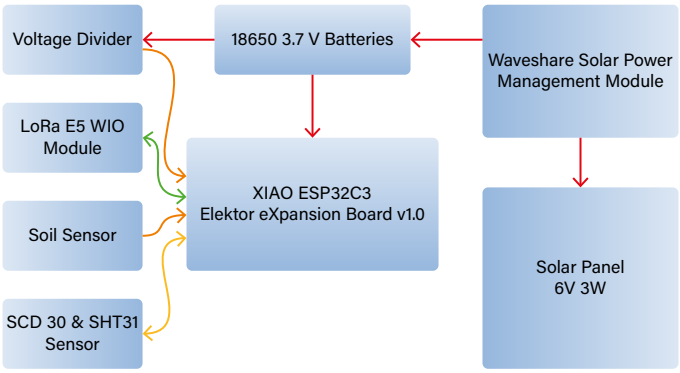


Bild 2. Blockschaltbild des Projekts.

Wegen seiner hohen Genauigkeit wurde der SCD30 von Seeed Studio als CO_2-Sensor ausgewählt. Da der SCD30 ein NDIR-Sensor ist, ein „echter" CO_2-Sensor, zeigt er die CO_2-Konzentration in der Umgebungsluft in ppm (Parts per Million) an. Ein weiterer Vorteil dieses Sensors ist, dass er auch einen Temperatur- und Feuchtigkeitssensor des Typs SHT31 enthält. Da das SCD30-Modul nicht für den Außeneinsatz konzipiert ist, wurde ein maßgeschneidertes 3D-gedrucktes Wetterhäuschen entworfen, in dessen Inneren der Sensor vor widrigem Wetter geschützt ist.

Der im Projekt verwendete Bodenfeuchtigkeitssensor stammt ebenfalls von Seeed Studio. Er ist zwar gut verfügbar, aber nicht wirklich für den Außeneinsatz geeignet, da die Anschlüsse und die Platine nicht wasserdicht sind, wie in **Bild 3** zu sehen. In einer zukünftigen Version könnte ein besser geeigneter Bodenfeuchtigkeitssensor in Betracht gezogen werden.

Bild 3. Da der Bodensensor offene Pads und einen JST-Anschluss besitzt, kann es bei hoher Feuchtigkeit zu einem Kurzschluss kommen.

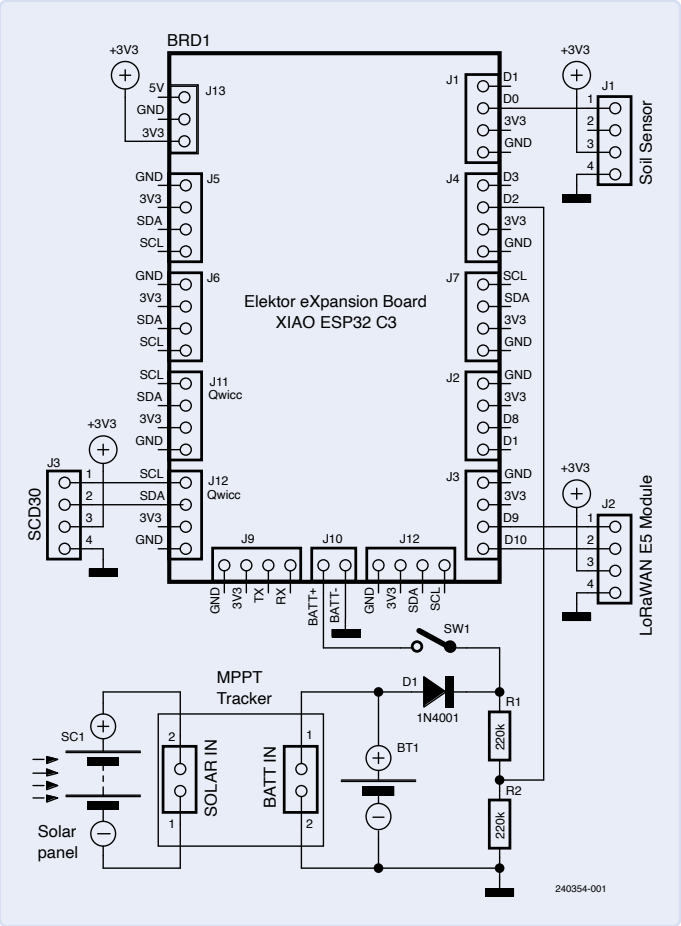


Bild 4. Das Schaltplan des Projekts mit dem zentralen Elektor eXpansion Board.

Der Mikrocontroller XIAO ESP32-C3 kommuniziert mit dem Sensor SCD30 über I²C und mit dem LoRa E5-Modul über den UART. Der Bodenfeuchtesensor wird an einen analogen Pin angeschlossen, und die Batteriespannung wird mit einer Spannungsteilerschaltung überwacht. Der Schaltplan des Projekts ist in **Bild 4** zu sehen.

Einrichten Ihres LoRaWAN-Gateways

Bevor wir uns den technischeren Teilen des Projekts zuwenden, beginnen wir mit der wichtigsten Aufgabe: dem Einrichten des LoRaWAN-Gateways. Da wir LoRaWAN zur Übertragung von Sensordaten verwenden, sollte ein LoRaWAN-Gateway in der Nähe die vom Sensorknoten gesammelten Daten ins Internet senden. Wenn Sie kein LoRaWAN-Gateway in Ihrer Nähe finden, müssen Sie Ihr eigenes einrichten.
Das Einrichten eines eigenen LoRaWAN-Gateways ist keine Raketentechnik. Zuerst müssen Sie die Gateway-Hardware erwerben; ich habe das LoRaWAN-Gateway LPS8v2 von Dragino gewählt. Um das Gateway zu nutzen, müssen Sie es mit einem LoRaWAN-Netzwerk verbinden. Ich habe mich für *The Things Network* (TTN) entschieden, aber Sie können auch andere Netzwerke wie Helium, Datacake oder andere verwenden oder sogar zwei Netzwerke gleichzeitig nutzen. Hier ist ein kurzer Überblick über den Einrichtungsprozess:

- **Verbindung mit WLAN**: Schalten Sie zunächst Ihr LoRaWAN-Gateway ein und verbinden Sie es mit Ihrem häuslichen WLAN. Dadurch kann das Gateway auf das Internet zugreifen.
- **Registrierung bei TTN**: Gehen Sie zur Konsole von *The Things*

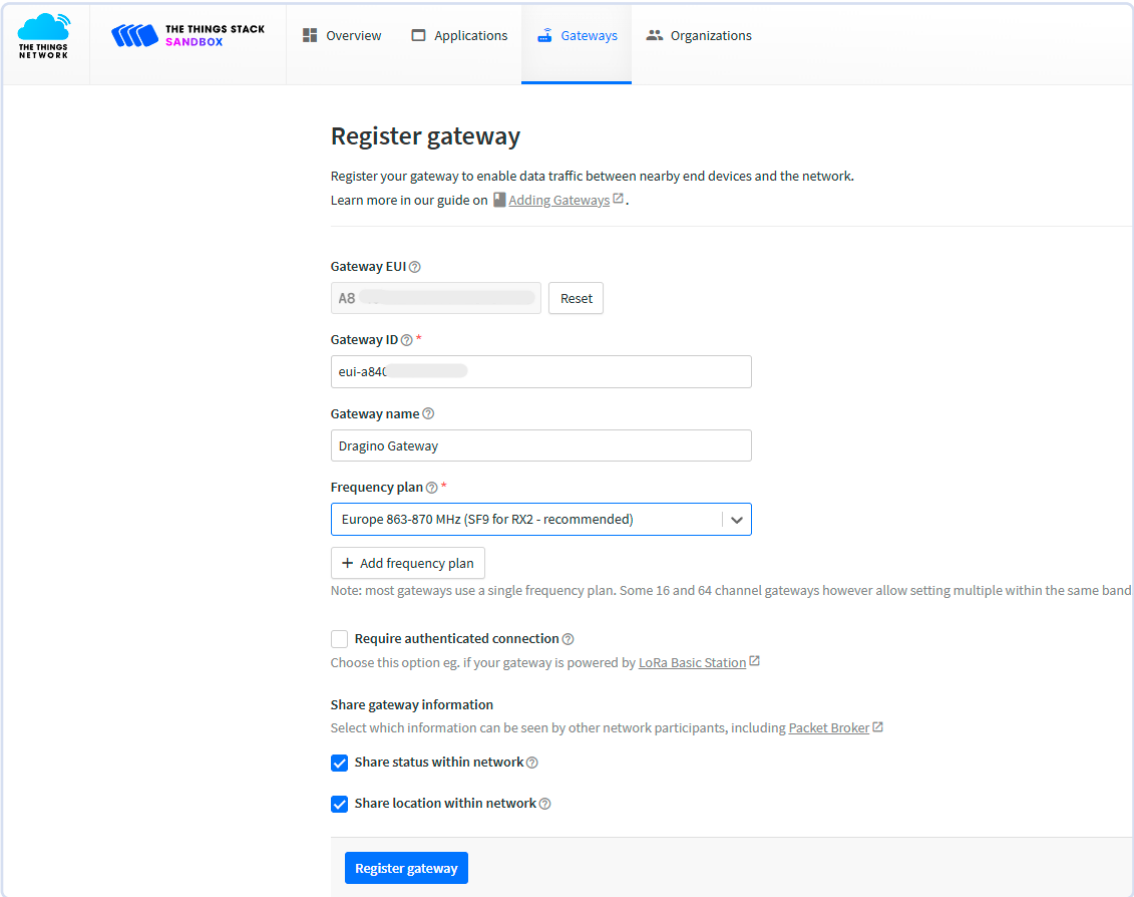


Bild 5. Hinzufügen Ihres Gateways zum The Things Network.

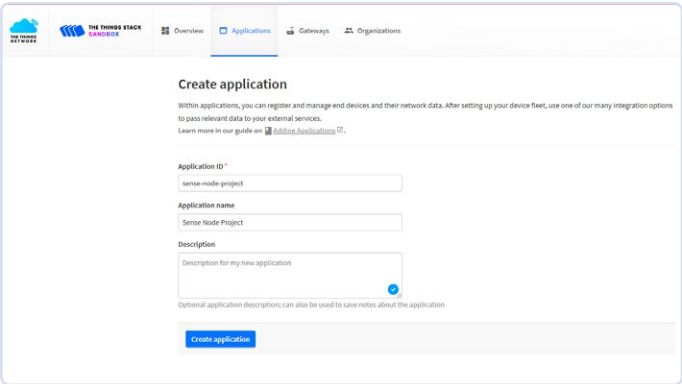


Bild 6. Erstellen der Anwendung der TTN-Konsole.

Network und erstellen Sie ein Konto, falls Sie noch keins haben. Sobald Sie eingeloggt sind, registrieren Sie Ihr Gateway, indem Sie die erforderlichen Details wie die EUI des Gateways (eine eindeutige Kennung für Ihr Gateway) angeben und Ihre Region auswählen, wie in **Bild 5** dargestellt.

- **Konfiguration des Gateways**: Folgen Sie den Anweisungen von TTN, um Ihr Gateway zu konfigurieren. Dies bedeutet in der Regel die Eingabe der Netzwerkeinstellungen, die Auswahl des geeigneten Frequenzplans und die Sicherstellung, dass das Gateway so eingestellt ist, dass es mit TTN-Servern kommuniziert.
- **Letzte Schritte**: Nach der Konfiguration sollte das Gateway eine Verbindung zu TTN herstellen und bereit sein, Daten von Ihren LoRa-Knoten zu empfangen. Sie können den Status Ihres Gateways auf der TTN-Konsole überwachen, um sicher zu sein, dass es korrekt funktioniert.

Es gibt viele detaillierte Schritt-für-Schritt-Anleitungen im Netz [2] [3] zur Einrichtung verschiedener Arten von LoRaWAN-Gateways. Wenn Sie diesen Anleitungen folgen, können Sie eventuelle Probleme beheben und Ihr Gateway korrekt konfigurieren und verbinden. Mit Ihrem eingerichteten LoRaWAN-Gateway registrieren wir nun im nächsten Abschnitt unseren Sensorknoten im The Things Network.

Projektintegration in The Things Network

Sobald Ihr Gateway eingerichtet ist, besteht der nächste Schritt darin, das LoRa-E5-Modul in *The Things Network* [4] zu integrieren. Beginnen Sie mit der Registrierung Ihres Geräts auf der TTN-Konsole. Zuerst müssen Sie eine neue Anwendung erstellen und dann Ihr Gerät hinzufügen. Gehen Sie zur TTN-Konsole und klicken Sie auf *Create application*. Geben Sie nun eine ID und einen Namen ein und klicken Sie dann auf *Create application*, wie in **Bild 6** dargestellt.
Jetzt müssen Sie Ihr Gerät in der Anwendung registrieren, die Sie gerade erstellt haben. Dazu gehen Sie zu Ihrer Anwendung und klicken auf *Register end device*. Während dieses Vorgangs erhalten Sie wichtige Anmeldeinformationen wie die *Device EUI*, *Application EUI* und den *App Key* (**Bild 7**). Diese Anmeldeinformationen sind unverzichtbar, um Ihr Gerät so zu konfigurieren, dass es mit TTN kommunizieren kann, und werden später im Arduino-Code verwendet. Sie sollten diese Informationen wenn schon nicht in Stein meißeln, so doch ganz altmodisch auf Papier aufschreiben!
Nach der Registrierung des Geräts auf der TTN-Konsole müssen Sie das Nutzdatenformat einrichten, damit das TTN die von Ihrem Gerät gesendeten Daten korrekt interpretiert. TTN ermöglicht es Ihnen, einen benutzerdefinierten Payload-Formatierer in JavaScript zu definieren, der die von Ihrem Gerät gesendeten Rohdaten dekodiert. So sieht das JavaScript dieses Payload-Formatierers aus:

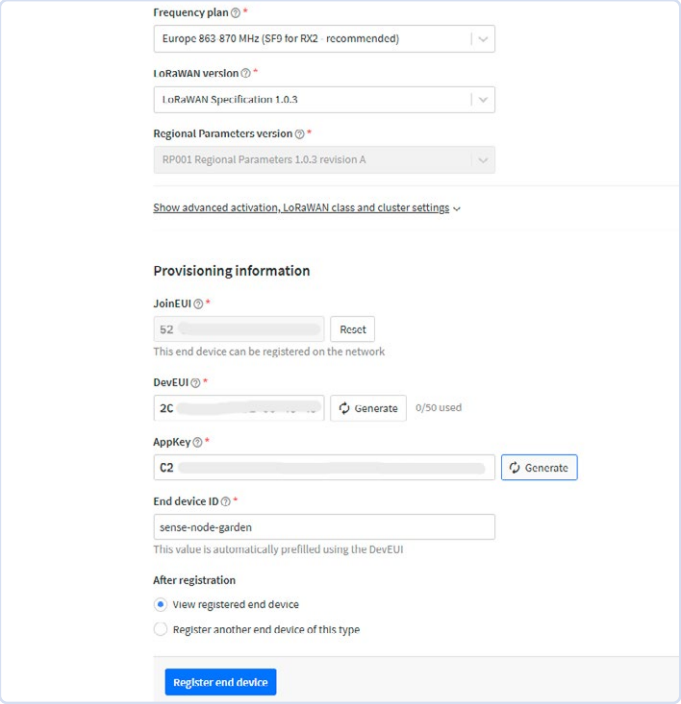


Bild 7. Registrierung des Endgeräts in der Anwendung.

```
function Decoder(bytes, port) {
  var decoded = {};
  if (port === 8) {
    decoded.soilMoisture = (bytes[0] << 8) | bytes[1];
    decoded.temp = ((bytes[2] << 8) | bytes[3]);
    decoded.humi = (bytes[4] << 8) | bytes[5];
    decoded.co2 = (bytes[6] << 8) | bytes[7];
    decoded.battery = (bytes[8] << 8) | bytes[9];
  }
  return decoded;}
```

Um dies zu implementieren,

- **navigieren Sie zu Payload-Formaten**: Gehen Sie in Ihrer TTN-Konsole zu Ihrer Anwendung und wählen Sie den Tab *Payload Formats*.
- **wählen Sie Dekoder-Funktion**: Wählen Sie den Funktionstyp *Decoder*.
- **fügen Sie den Code ein**: Kopieren Sie den bereitgestellten JavaScript-Code und fügen Sie ihn in den Dekoder-Funktionseditor ein.
- **speichern Sie die Änderungen**: Speichern Sie die Änderungen, um den Dekoder auf Ihre Anwendung anzuwenden.

Diese Dekoder-Funktion verarbeitet die von Ihrem Sensorknoten empfangenen Bytes und wandelt sie in lesbare Sensorwerte wie Bodenfeuchtigkeit, Temperatur, Feuchtigkeit, CO_2-Werte und Batteriespannung um.

Die Software

Die Software ist so gestaltet, dass sie die Sensordaten effizient erfasst und überträgt und gleichzeitig die Stromaufnahme durch den Tiefschlafmodus minimiert. Schauen wir uns die wichtigsten Teile des Codes und ihre Zusammenarbeit an.

Listing 1: Der Arduino-Sketch (Ausschnitt)

```
#include <Arduino.h>
#include <SCD30.h>
#include <HardwareSerial.h>
#include <config.ino>

// Defining Hardware the second internal UART -
// Serial2 for the LoRaWAN E5 Module - Pin 9 and 10
HardwareSerial Serial2(1);

...
//*** Initializing variables ***//

void setup() {

  ...
  //*** Initializing Sensors, Serial 1 & Serial 2 ***//

  // Check if the AT command returns OK
  if (at_send_check_response("+AT: OK", 100, "AT\r\n")) {
    // Set the flag to indicate the LoRa module exists
    is_exist = true;
    // Send AT command to get AppEUI and check the response
    at_send_check_response("+ID: AppEui", 1000, "AT+ID\r\n");
    // Set the LoRa module to LWOTAA mode
    at_send_check_response("+MODE: LWOTAA", 1000, "AT+MODE=LWOTAA\r\n");
    // Set the data rate to EU868
    at_send_check_response("+DR: EU868", 1000, "AT+DR=EU868\r\n");
    // Set the channel number range
    at_send_check_response("+CH: NUM", 1000, "AT+CH=NUM,0-2\r\n");
    // Set the APP Key for authentication, replace with your generated APP Key from TTN
    at_send_check_response("+KEY: APPKEY", 1000,
             "AT+KEY=APPKEY,\"C2XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX\"\r\n");
                        // Enter your generated APP Key here.
    // Set the LoRa module to Class A
    at_send_check_response("+CLASS: C", 1000, "AT+CLASS=A\r\n");
    // Set the port number to 8
    at_send_check_response("+PORT: 8", 1000, "AT+PORT=8\r\n");
    // Delay to ensure all commands are processed
    delay(200);
    // Print confirmation that the LoRaWAN setup is complete
    Serial.println("LoRaWAN");
    // Set the flag to indicate the module has joined the network
    is_join = true;
  }
  else {
    is_exist = false;
    Serial.print("No LoRa E5 module found.\r\n");
  }
}

void loop() {
  getSensors();
  sendData();
  // Configure the wake-up source and duration for deep sleep
  esp_sleep_enable_timer_wakeup(10 * 60 * 1000000);
                  // 10 minutes in microseconds
  // Enter deep sleep mode
  esp_deep_sleep_start();
}
```

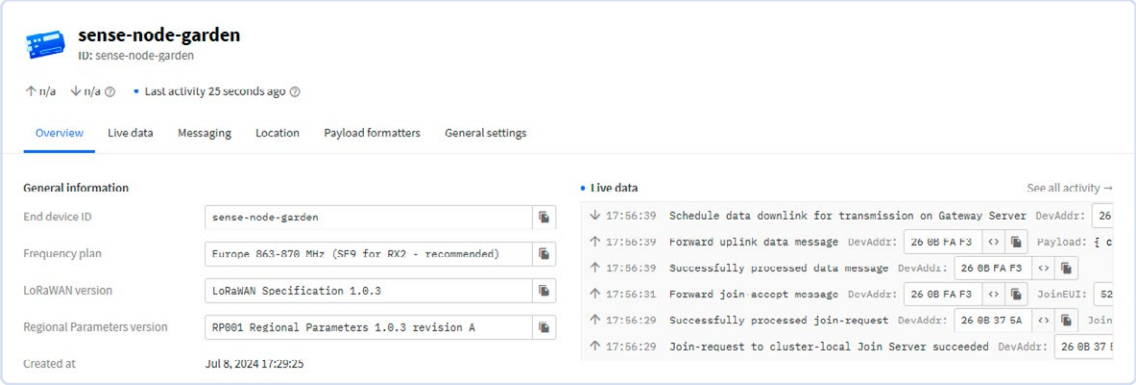


Bild 8. Daten, die vom Sensorknoten gesendet werden, angezeigt in der TTN-Geräteübersicht.

Der Code beginnt wie üblich mit der Einbindung der erforderlichen Bibliotheken und der Definition der Hardwarekonfiguration. In der `setup`-Funktion initialisieren wir die serielle Kommunikation, konfigurieren die Sensoren und richten das LoRa-Modul ein. Dies umfasst die Einstellung von Parametern für das LoRa-E5-Modul, einschließlich der Eingabe des *App Key*, der auf der TTN-Konsole generiert wurde (siehe unten). In **Listing 1** sehen Sie eine verkürzte Version des Codes; der vollständige Code und alle Hardware-Dateien ist auf dem GitHub-Repository dieses Projekts [5] verfügbar.

Die wichtigsten Vorgänge finden in der `loop`-Funktion statt. Sie sammelt Sensordaten, sendet sie an TTN und schaltet dann den Mikrocontroller aus, um Strom zu sparen und weckt ihn nur bei Bedarf. Im Tiefschlafmodus schaltet der XIAO ESP32-C3 fast ganz ab und benötigt statt 8 mA nur noch etwa 142 µA. Dadurch hält die Batterie länger. Nur 5 µA gehen auf die Kappe des ESP32C3 selbst; den Rest benötigt der integrierte Spannungsregler und das Batteriemanagement-IC. Der Mikrocontroller wacht regelmäßig auf, um Daten zu sammeln und zu übertragen, bevor er wieder in den Tiefschlaf fällt.

Die Funktion `getSensors()` liest Werte des Bodenfeuchtesensors und des CO_2-Sensors SCD30, berechnet (über einen Spannungsteiler) die Batteriespannung und bildet den Durchschnitt mehrerer Messwerte, um die stabile und präzise Messungen zu erzielen.

Die Funktion `getSensors()` formatiert die gesammelten Sensordaten in eine Nutzlast und sendet sie über das LoRa-E5-Modul an TTN. Dazu werden die Sensordaten in eine hexadezimale Zeichenkette umgewandelt, wie im folgenden Codeschnipsel zu sehen ist. Diese Funktion sorgt dafür, dass die Daten korrekt formatiert und über LoRa übertragen werden, indem sie sich zunächst mit dem nächstgelegenen Gateway verbindet und die Sensordaten sendet, was eine Echtzeitüberwachung ermöglicht. Nach dem Senden der Daten wird das LoRa-E5 Modul in einen Schlafmodus versetzt, um Strom zu sparen.

```
// Prepare and send the sensor data
// as a hexadecimal string via LoRa
char cmd[128];
sprintf(cmd, "AT+CMSGHEX=
   \"%04X%04X%04X%04X%04X\"\r\n",
   (int)soilMoisturePercent, (int)temp,
   (int)humi, (int)co2, (int)battery);
```

Der App Key

Bevor Sie den Code hochladen, müssen Sie den *App Key* hinzufügen, der generiert wurde, als Sie Ihr Gerät zur TNN-Anwendung hinzugefügt haben. Er ist wichtig, weil er Ihr Gerät bei TTN authentifiziert. Er wird in der `setup`-Funktion eingegeben und in der Funktion `at_send_check_response` verwendet.

```
at_send_check_response("+KEY: APPKEY", 1000,
"AT+KEY=APPKEY,\"XXXXXXXXXXXXXXXXX\"\r\n");
// Enter your generated APP Key here
```

Dieser Abschnitt des Codes sorgt dafür, dass Ihr Gerät registriert und authentifiziert wird, damit es Daten an das TTN senden kann. Wenn alles richtig gemacht wurde, können Sie die weitergeleiteten Uplink-Nachrichten von Ihrem LoRa-Sensorknoten im Live-Datenbereich Ihres TTN-Geräts sehen, wie in **Bild 8** dargestellt. Die gesammelten Dateien für das gesamte Projekt ist im Repository auf GitHub [5] zu finden.

Stromaufnahme

Bei jedem System, das mit Batterien betrieben wird, muss besonders auf den Strombedarf geachtet werden, denn niemand möchte es ständig alle paar Tage aufladen. Außerdem sollten Systeme für den Außenbereich nur eine minimale Wartung erfordern. Die Berechnung der Stromaufnahme der einzelnen Komponenten und die Überprüfung der durchschnittlichen Stromaufnahme des gesamten Systems, die anschließende Berechnung der Coulomb-Werte für die Abschätzung der Batterielebensdauer und vieles mehr kann ein wenig hektisch sein. Um diese Herausforderungen zu meistern, lassen sich Tools wie *Power Profiler* oder DC-Energieanalysatoren verwenden, die genaue Messwerte liefern und den Strombedarf jeder einzelnen Komponente des gesamten System analysieren.

Vor kurzem habe ich ein Joulescope JS200 [6] in die Finger bekommen, und ich muss sagen, dass es diesen Prozess erheblich vereinfacht. Dank seiner Kompaktheit und seiner benutzerfreundlichen Oberfläche ist das Gerät ideal für alle Arten von Stromverbrauchstests. Mein System

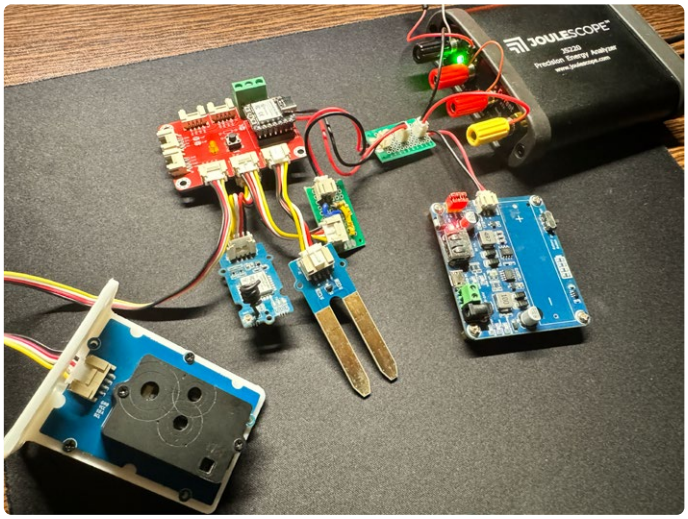


Bild 9. Die Stromaufnahme des Projekts wird von einem JouleScope ermittelt.

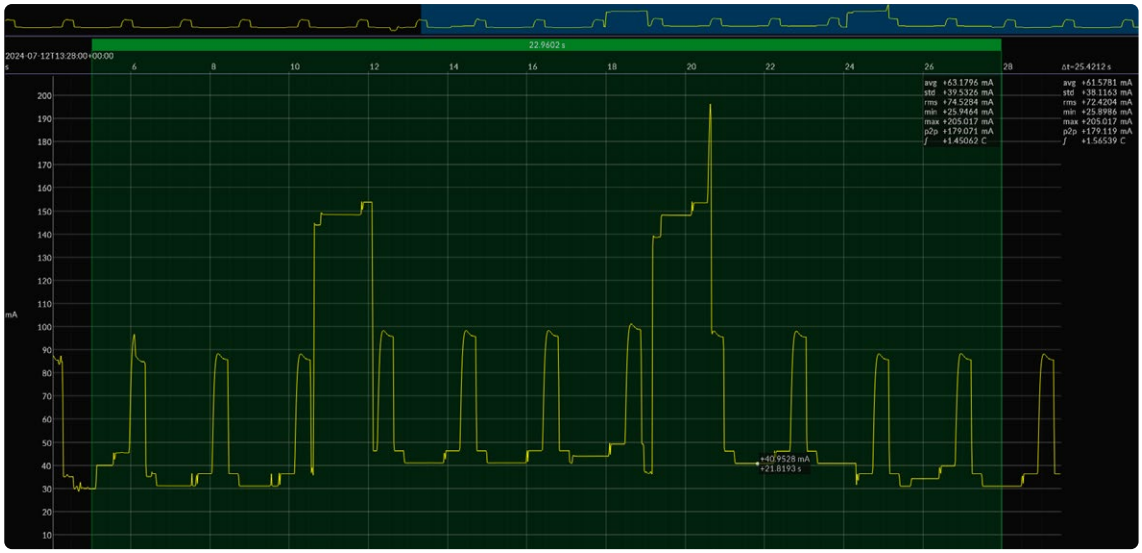


Bild 10. Der Stromaufnahme des Sensors SCD30 und des LoRa-Moduls.

hatte zunächst keine Optimierungen der Stromaufnahme zum Ziel; die Priorität lag darin, sicherzustellen, dass alles zusammen funktioniert. Die Optimierung des Codes und der Stromaufnahme war sekundär und kam als nächstes.

Nach dem ersten Aufbau des Systems habe ich es mit dem Joulescope verbunden, um zu sehen, was los war, wie in **Bild 9** zu sehen ist. Und das erste, was mir auffiel, war, dass die IR-LED für die CO_2-Messung im SCD30-Sensor alle zwei Sekunden fast 68 mA zieht, was über 30 Sekunden zu einem gemittelten Strom von 25,3 mA führt, wie in **Bild 10** zu sehen. Nach Konsultation des Datenblatts und der bereitgestellten Bibliothek stellte ich fest, dass das Messintervall standardmäßig auf zwei Sekunden eingestellt war. Dieser Makel war einfach zu beseitigen, da das Messintervall gesteuert werden kann, so dass IR-LED nur dann eingeschaltet wird, wenn eine Messung ansteht. Dadurch werden die Stromspitzen reduziert und der durchschnittliche Stromverbrauch gesenkt. Allerdings verbraucht der SCD30-Sensor selbst im Leerlauf immer noch etwa 4,5 mA, und ich konnte keine softwarebasierte Lösung finden, um dies zu reduzieren.

Als Nächstes habe ich mich auf das Modul LoRa-E5-WIO konzentriert, das der Batterie ebenfalls einen beträchtlichen Strom entzieht. Während des LoRa-Betriebs nimmt das Modul bis zu 110 mA auf, aber das ist akzeptabel, da er nur eine Sekunde andauert. Das Wichtigste ist meist, dass der Batterie so wenig Strom wie möglich entnommen wird, wenn sich das System im Leerlauf und der ESP32-C3 im Tiefschlafmodus befindet, da das Modul im Leerlauf bis zu 10 mA verbraucht. Das ist angesichts der langen Leerlaufphasen eine beträchtliche Menge an Leistung! Nachdem ich einige Online-Ressourcen durchsucht hatte, fand ich heraus, dass es einen AT-Befehl gibt, mit dem das Modul in den Tiefschlafmodus versetzt werden kann, wodurch die Stromaufnahme im Leerlauf auf 51,7 µA gesenkt wurde, was eine erhebliche Verbesserung darstellt. In **Bild 11** sehen Sie die Stromkurve des LoRa-E5-Moduls vor und nach der Aktivierung des Schlafmodus. Die Stromaufnahme des LoRa-Moduls kann sogar noch weiter auf 3 µA gesenkt werden, wenn der LDO auf dem Modul entfernt wird.

Bei weiteren Tests stellte ich fest, dass der Bodenfeuchtesensor die Stromaufnahme des Systems um bis zu 4 mA erhöht, wenn er eine gewisse Feuchtigkeit erkennt. Dieser Sensor misst die Feuchtigkeit über die Kapazität, was zu einer übermäßigen Stromaufnahme führt, wenn der Boden leitfähiger wird. Dies führt übrigens zu einem weiteren Problem: Die blanken Leiterbahnen des Sensors wirken wie ein Elektrolysegerät und verursachen Korrosion und Mineralablagerungen auf den Sonden. Um das Korrosionsproblem zu mildern, kann ein beschichteter Bodensensor verwendet werden. Um jedoch die übermäßige Stromaufnahme bei steigender Feuchtigkeit zu eliminieren, muss die Stromzufuhr zum Sensor unterbrochen werden, wenn keine Messung erfolgt, was eine eher hardwarebasierte Problemlösung erfordert. Bei der ersten Projektversion habe ich es einfach gehalten und mich auf softwarebasierte Optimierungen konzentriert, aber künftige Versionen werden mehr hardwarebasierte Lösungen bieten.

Akkulaufzeit

Dies führt uns zur Berechnung der Akkulaufzeit des Systems, die mit dem Joulescope erheblich erleichtert wurde. Nach allen softwarebasierten Leistungsoptimierungen habe ich das System mit dem Joulescope verbunden. In **Bild 12** ist die Stromaufnahme des gesamten Systems nach den Optimierungen zu sehen. Beachten Sie, dass der SCD30 jetzt die IR-LED nur einschaltet, wenn der ESP32-C3 wach ist, und dann die Messungen stoppt, wenn der ESP32-C3 in den Tiefschlafmodus wechselt.

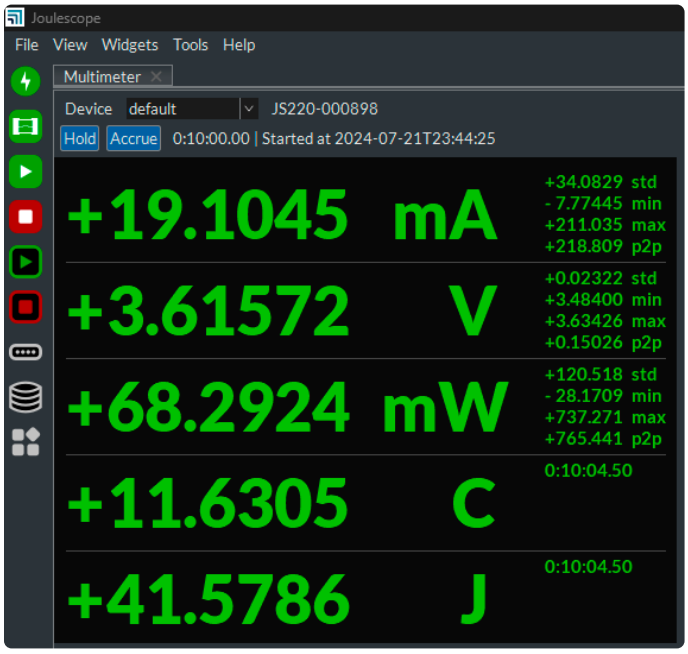


Bild 13. Screenshot des Multimeter-Modus des JouleScopes.

Bild 11. Stromaufnahme des LoRa-Moduls ohne und mit Schlafmodus.

Bild 12. Stromaufnahme des gesamten Systems nach allen Stromoptimierungen.

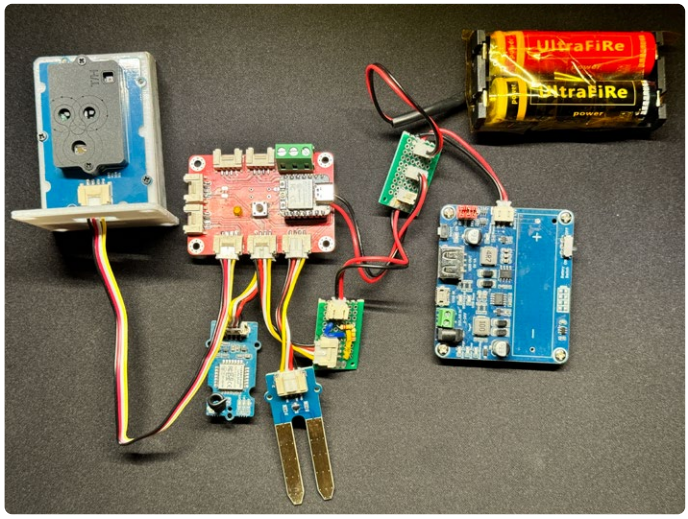


Bild 14. Alle Komponenten des Projekts sind zusammengebaut und bereit, in das Gehäuse eingebaut zu werden.

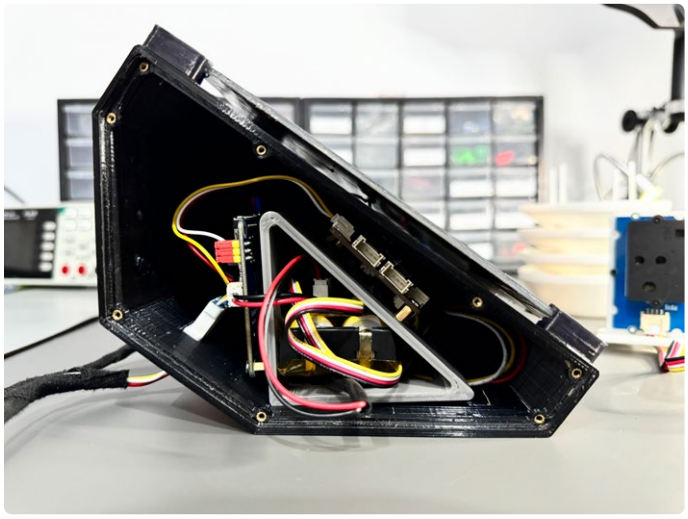


Bild 15. Alle Komponenten im Gehäuse mit oben montiertem Solarpanel.

Mithilfe der Akkumulier-Funktion im Multimeter-Modus des Joulescope habe ich über zehn Minuten die durchschnittliche Stromaufnahme und die von dem System verbrauchte Ladung in Coulomb (Amperesekunden) gemessen, wobei jede Minute neue Sensordaten über LoRa gesendet wurden. Wie in **Bild 13** zu sehen ist, betrug der durchschnittliche Strom 19,10 mA, was für mein Ziel einer einwöchigen Betriebsdauer (ohne gleichzeitige Nachladung) zu hoch ist. Durch die Erhöhung des Intervalls der Datenermittlung auf zehn Minuten sank der durchschnittliche Strom auf 4,905 mA, was für diese Anwendung akzeptabel ist.
Mit dem Joulescope wurde dann über eine Stunde und fünfunddreißig Minuten (5700 s) die insgesamt verbrauchte Batteriekapazität von 28,1403 As gemessen. Um die Akkulaufzeit abzuschätzen, muss man zunächst die Angabe der Batteriekapazität von Milliamperestunden (mAh) in Amperesekunden (As) umrechnen. Ich verwendete zwei Li-Ion-Batterien des Typs 18650 parallel, von denen jede eine Batteriekapazität von 1800 mAh hat.

Batteriekapazität = 3600 mAh × 3600 = 12.960 As

Da die Abschaltspannung bei etwa 3,3 V liegt, was höher ist als die Entladeschlussspannung der Akkus von 3 V, müssen wir etwa 80 % bis 90 % der Nennkapazität als nutzbar annehmen. Unter Berücksichtigung der Selbstentladungsrate der Batterien nehmen wir als Annäherung 85 %:

Nutzbare Kapazität = 0,85 × 12.960 As = 11.016 As

Nun berechnen wir den durchschnittlichen Strom:

28,1403 As / 5700 s = 0,004936 A = 4,936 mA

Und schließlich die Akkulaufzeit, indem wir die nutzbare Batteriekapazität durch den Strom teilen, um die Akkulaufzeit in Stunden zu ermitteln:

(11.016 As / 0,004936 A) / 3600 = 619,94 h.

Das entspricht etwa 25 Tagen Akkulaufzeit mit einer einzigen Ladung. Ohne jegliche Leistungsoptimierungen würde das gesamte System nur sechs Tage durchhalten! Nachdem ich mit der Stromaufnahme nach der Entfernung einiger weiterer unerwünschter LEDs zufrieden war, bereitete ich das System für den Einsatz vor (**Bild 14**).

Bild 16. Seitenansicht der Installation des LoRa-Sensorknotens, die die wasserdichten Verbinder für USB von JST zeigt.

Komponenten und Gehäuse

Ein maßgeschneidertes Gehäuse [5] wurde im 3D-Druck entwickelt, das alle Komponenten des LoRa-Sensorknotens aufzunehmen. Das Hauptgehäuse wurde so konstruiert, dass es wetterbeständig ist und alle Komponenten umschließt, wie in **Bild 15** zu sehen. Beim Entwurf mussten mehrere Faktoren sorgfältig berücksichtigt werden. Das Gehäuse garantiert, dass alle elektrischen Verbindungen wasserdicht sind, wobei spezielle wasserdichte JST-Steckverbinder (Typ JWPF) verwendet werden, um externe Sensoren und die Solarpanel-Stromversorgung mit dem *Solar Power Management Module* zu verbinden. alle Lücken zwischen Steckverbindern und Gehäuse wurden mit Epoxidharz abgedichtet.
Ein wasserdichter Knopf wurde ebenfalls installiert, um das System neu starten oder ein- und ausschalten zu können, und eine wasserdichte USB-C-Buchse erlaubt es, neue Firmware hochzuladen oder Probleme zu debuggen, ohne das Gehäuse zu öffnen (**Bild 16**). Dieser Anschluss ermöglicht es auch, die Batterie „notzuladen", falls das Wetter über mehrere Tage schlecht ist und die Batterien aufgrund fehlenden Sonnenscheins entleert sind.
Da der SCD30-Sensor für Temperatur, Feuchtigkeit und CO_2 nicht für den Außeneinsatz vorgesehen ist, wurde ein maßgeschneidertes

Bild 17. LoRa-Sensorknoten und der Thermometerhütte.

Wetterhäuschen entworfen, um die Sensorik vor Regen zu schützen (**Bild 17**). Dieses Wetterhäuschen kann auch andere Sensoren aufnehmen, mit einem Bauteilhalter im Inneren, der auf dem Platinenlayout eines Grove-Moduls basiert und die Montage von bis zu vier kleinen Grove-Sensoren ermöglicht.
Nachdem das System wie in **Bild 18** installiert worden war, konnte es seine wasserdichten Fähigkeiten gleich am nächsten Tag beweisen:

Bild 18. Der LoRa-Sensorknoten im Einsatz.

Nach einer Woche mit starken Regenfällen und einem Sturm blieb das System im Inneren staubtrocken und alle Komponenten funktionierten weiterhin einwandfrei.

Integration in Home Assistant

Die Integration Ihres LoRa-Sensorknotens in Home Assistant [8] bietet eine ausgezeichnete Lösung für die langfristige Datenspeicherung und -analyse, ohne die durch Plattformen wie Datacake auferlegten Speicherbeschränkungen. Da Home Assistant auf Ihrer eigenen Hardware läuft, gibt es keine Speicherplatz- oder Datenpunktbeschränkungen, was des Assistant ideal für eine kontinuierliche Überwachung macht.
Damit The Things Network und Home Assistant zusammenarbeiten, beginnen Sie damit, in der TTN-Konsole einen *API Key* zu erstellen. Navigieren Sie zu Ihrer Anwendung, gehen Sie zum Abschnitt *API Keys* und erstellen Sie einen neuen Schlüssel mit den erforderlichen Berechtigungen. In *Integrations* im TNN-Applikations-Dashboard muss *Storage Integration* aktiviert werden, da ansonsten die Sensordaten nicht in Home Assistant angezeigt werden.

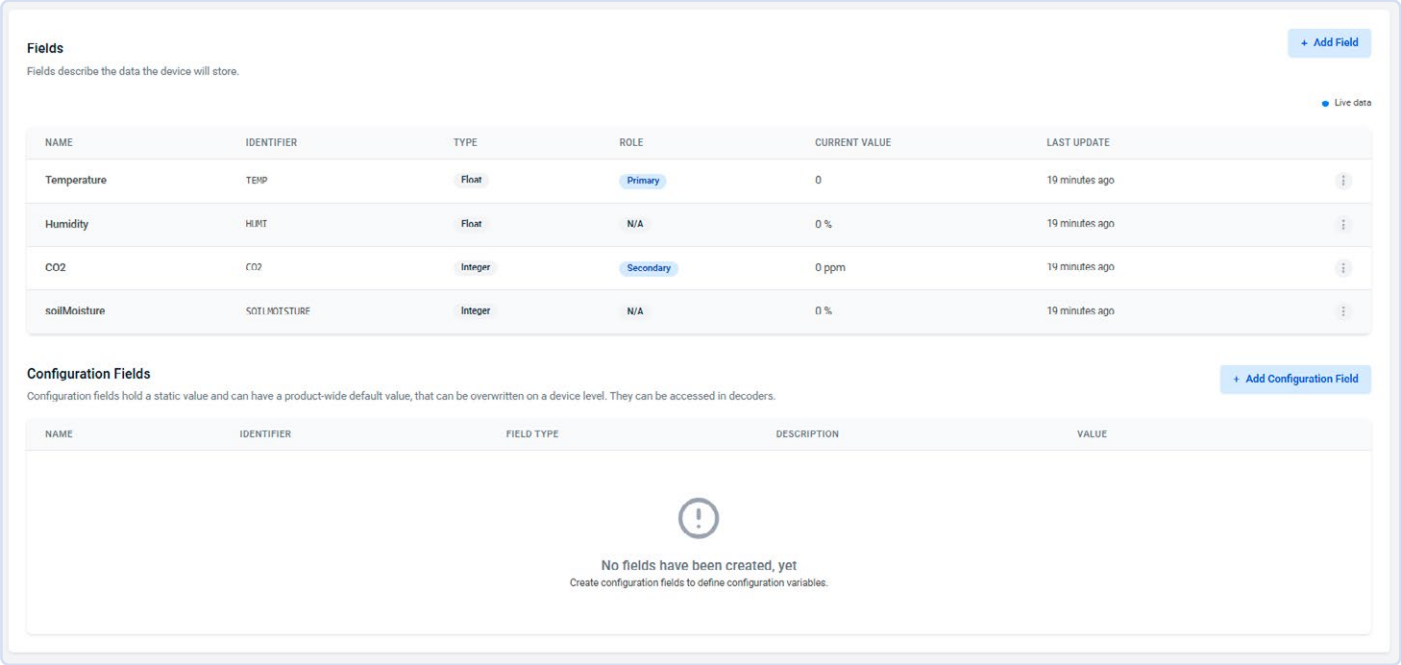


Bild 19. Konfiguration der Datacake-Felder von Sensordaten, die von TTN empfangen wurden.

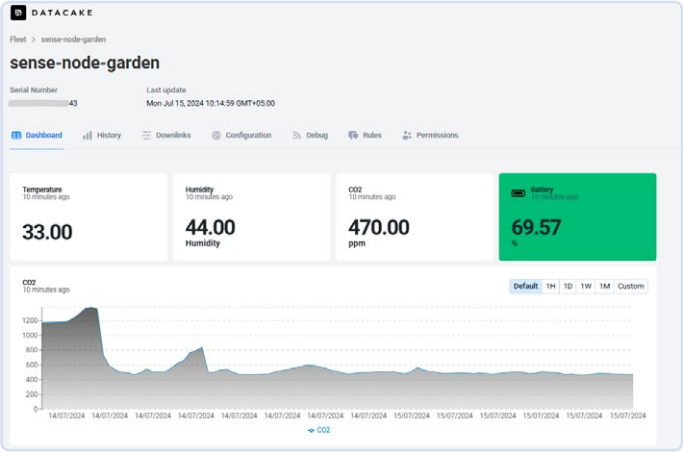


Bild 20. Alle gesammelten Sensordaten werden im Datacake-Dashboard gezeigt.

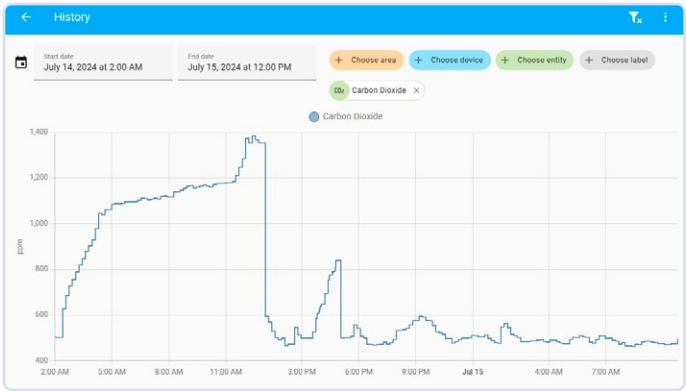


Bild 21. CO_2-Verlauf im Home Assistant.

Gehen Sie anschließend in Home Assistant zu *Devices & Services* und fügen Sie die Integration von *The Things Network* hinzu. Wenn Sie dazu aufgefordert werden, geben Sie den Namen Ihrer Anwendung in TTN und den von Ihnen generierten API-Schlüssel ein. Dadurch wird Ihre Home-Assistant-Instanz mit Ihrer TTN-Anwendung verknüpft, sodass dem Datenfluss von Ihrem LoRa-Sensorknoten zum Home Assistant nichts im Wege steht.
Um die Sensorwerte in einem Liniendiagramm wie in **Bild 21** anzuzeigen, müssen Sie die Maßeinheit für jede Sensorentität definieren, die vom The Things Network empfangen wird. Fügen Sie die folgenden Einträge zur Datei *configuration.yaml* im Home Assistant hinzu:

```
sensor.lora_sense_node_temperature:
  unit_of_measurement: "°C"
sensor.lora_sense_node_co2:
  unit_of_measurement: "ppm"
sensor.lora_sense_node_humidity:
  unit_of_measurement: "%"
sensor.lora_sense_node_soilmoisture:
  unit_of_measurement: "%"
```

In diesen Einträgen ist `sensor.lora_sense_node_temperature` der Entitätsname, unter dem Home Assistant die Temperaturwerte von TTN empfängt. Definieren Sie auf ähnliche Weise die Maßeinheit für CO_2, Feuchtigkeit und Bodenfeuchtigkeit. Mit dieser Konfiguration werden die Sensorwerte korrekt auf Ihrem Home-Assistant-Dashboard angezeigt.
Wenn Sie diese Schritte durchgeführt haben, können Sie eine detaillierte und interaktive Anzeige im Dashboard Ihres Home Assistant wie in **Bild 22** sehen, die alle Sensorwerte anzeigt, die der LoRa-Sensorknoten sendet. Diese Einrichtung bietet eine robuste und flexible Lösung zur Überwachung und Analyse von Umweltdaten, die die Speicherkapazitäten und die anpassbare Benutzeroberfläche von Home Assistant nutzt.

Potenzial für Verbesserungen und Anwendungen

Obwohl der LoRa-Sensorknoten schon jetzt sehr effektiv ist, gibt es erheblichen Raum für zukünftige Verbesserungen, insbesondere in der Energieeffizienz. Im Leerlauf zieht das System etwa 4,32 mA, was für batteriebetriebene Langzeitanwendungen ziemlich viel ist. Durch die Integration einer externen Echtzeituhr (RTC) mit einer Power-Latch-Schaltung (automatische Abschaltung) könnte man den Leerlaufstrom drastisch auf bis zu 50 nA reduzieren.

In diesem verbesserten Aufbau würde die externe RTC vom Mikrocontroller gesteuert und die RTC die Stromabschaltung verwalten, das heißt, würde die Stromzufuhr zum gesamten System unterbrechen, wenn es nicht aktiv Sensordaten sendet, und es in festgelegten Intervallen wieder einschalten. Dieser Ansatz würde die Batterielebensdauer erheblich verlängern und das System noch besser für entfernte und langfristige Einsätze geeignet machen. Diese Lösung könnte in der nächsten Version der eXpansion-Platine implementiert oder als separates Modul hinzugefügt werden, das in das bestehende Setup integriert (will heißen, in einen Steckverbinder des eXpasion Boards gestöpselt) werden kann. Halten Sie Ausschau nach einem entsprechenden Artikel!
Zusätzlich gibt es weitere Hardware-Optimierungen, die in Erwägung gezogen werden können. So könnten zum Beispiel Komponenten mit hohem Energiebedarf durch effizientere Alternativen ersetzt, die Firmware zur Minimierung der Energieaufnahme optimiert und die Effizienz des Solarladesystems weiter verbessert werden.
Die modulare Natur dieses Systems ermöglicht es, verschiedene Sensoren hinzuzufügen, um seine Funktionalität zu erweitern. Durch die Integration weiterer Sensoren kann der LoRa-Sensorknoten für eine Vielzahl von Anwendungen genutzt werden, einschließlich, aber nicht beschränkt auf:

- **Umweltüberwachung**: Durch Hinzufügen von Sensoren für Luftqualität, Lichtintensität und Geräuschpegel kann man dieses System zu einer umfassenden Umweltüberwachungsstation ausbauen.
- **Landwirtschaftliche Anwendungen**: Die Integration von Sensoren für Boden-pH, Nährstoffe und Wetterdaten kann wertvolle Informationen für die Präzisionslandwirtschaft liefern.
- **Industrielle Überwachung**: Das Hinzufügen von Sensoren für Gaslecks, Vibrationen und Druck kann bei der Überwachung industrieller Umgebungen zur Sicherheit und Effizienz beitragen.

Zusammenfassend bietet der aktuelle LoRa-Sensorknoten eine robuste und flexible Plattform für die Fernüberwachung der Umwelt. Mit zukünftigen Verbesserungen, die sich auf Energieeffizienz und zusätzliche Sensoren konzentrieren, hat dieses System das Potenzial für verschiedenste Anwendungen, was es zu einem unverzichtbaren Werkzeug für die Überwachung verschiedenster Parameter macht.

SE – 240354-02

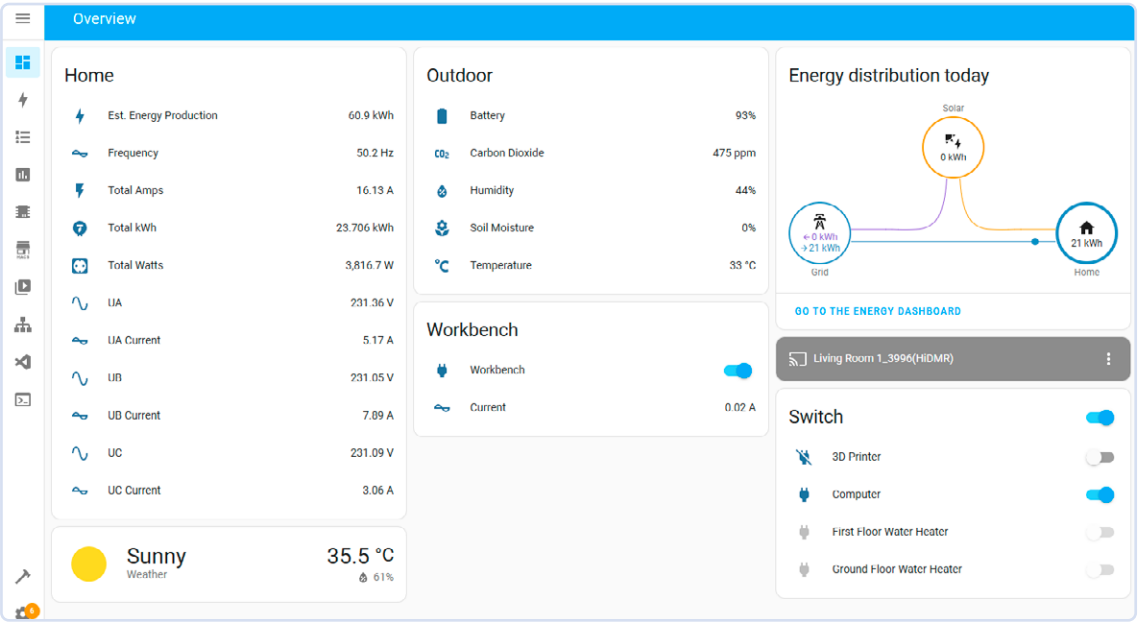


Bild 22. Numerische Übersicht der Daten des Sensorknotens im Dashboard von Home Assistant.

Sie haben Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Dann wenden Sie sich bitte an den Autor unter saad.imtiaz@elektor.com oder an die Elektor-Redaktion unter redaktion@elektor.de.

THEMENFOKUS
Besuchen Sie unsere Webseite **Drahtlose Kommunikation** für mehr Artikel, Projekte, Nachrichten und Videos.
www.elektormagazine.de/drahtlose-kommunikation

Über den Autor

Saad Imtiaz, Senior Engineer bei Elektor, ist Mechatronikingenieur mit umfangreicher Erfahrung in Embedded Systems und Produktentwicklung. In seiner Karriere hat er mit einer Vielzahl von Unternehmen zusammengearbeitet, von innovativen Startups bis hin zu etablierten globalen Unternehmen, und dabei zukunftsweisende Prototypen- und Entwicklungsprojekte vorangetrieben. Mit seinem reichen Hintergrund, der auch eine Zeit in der Luftfahrtindustrie und die Leitung eines Technologie-Startups umfasst, bringt Saad eine einzigartige Mischung aus technischem Fachwissen und unternehmerischem Geist in seine Aufgaben bei Elektor ein. Hier trägt er zur Projektentwicklung sowohl in der Software- als auch in der Hardware-Entwicklung bei.

Passende Produkte

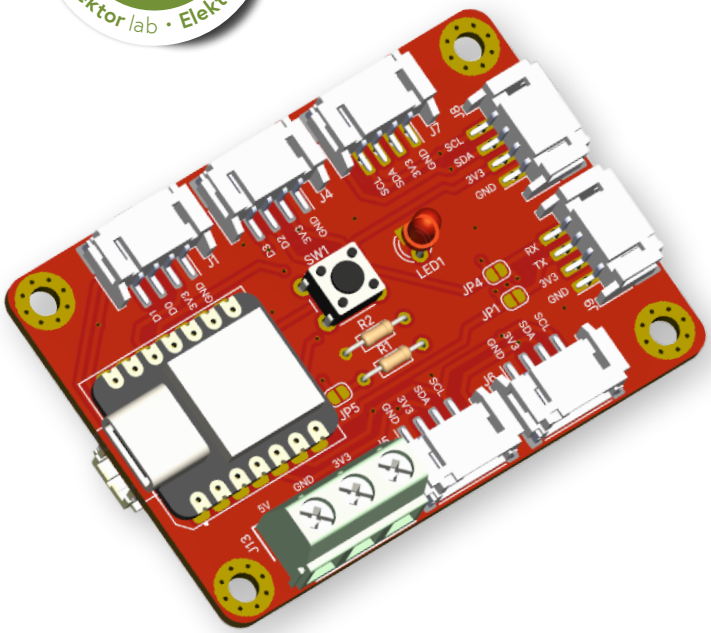
- **Seeed Studio XIAO ESP32C3**
  www.elektor.de/20265
- **Seeed Studio Grove SCD30 CO2**
  www.elektor.de/20012
- **Seeed Studio LoRa-Entwicklungskit E5 STM32WLE5JC**
  www.elektor.de/19956
- **Seeed Studio Solarpanel für Außenumgebungen (3 W)**
  www.elektor.de/19131
- **Waveshare Solarstrom-Managementmodul**
  www.elektor.de/20488
- **Dragino LoRa/LoRaWAN IoT Kit v3 (EU868)**
  www.elektor.de/20775

WEBLINKS

[1] Saad Imtiaz, „Elektor eXpansion Board v1.0“, Elektor 9-10/2024: https://elektormagazine.de/240250-02
[2] Einrichtung des Things Network V3 auf Dragino: http://wiki.dragino.com/xwiki/bin/view/Main/Notes%20for%20TTN/
[3] Dragino-LPS8N-Einrichtung mit dem Things Network: https://www.thethingsindustries.com/docs/gateways/models/dragino-lps8/
[4] The Things Network: https://www.thethingsnetwork.org/
[5] Github-Repository LoRa-Sensorknoten: https://github.com/ElektorLabs/lora-sensor-node
[6] Joulescope JS220: Präziser Energie-Analysator: https://www.joulescope.com/products/js220-joulescope-precision-energy-analyzer
[7] Datacake: https://datacake.co/
[8] Home Assistant: https://www.home-assistant.io/

Elektor eXpansion Board v1.0

Für ESP32-S3 und andere XIAO-Controller-Boards



Von Saad Imtiaz und Jens Nickel (Elektor)

Kompakt, preiswert und leistungsstark! Die XIAO-Familie umfasst sieben Breakout-Boards in Briefmarkengröße mit ESP32-Controllern sowie mit dem RP2040, dem SAMD21 und anderen. Doch für die Konnektivität in der Prototyping-Phase wird noch ein Erweiterungsboard benötigt. Das eXpansion Board von Elektor besitzt eine Vielzahl von Anschlüssen (nicht nur aus dem Seeed-Ökosystem) sowie eine User-Taste, LED plus einige Schutzfunktionen.

In der sich rasant entwickelnden Welt der Elektronik sind flexible und robuste Entwicklungswerkzeuge unverzichtbar. Viele Entwickler sind frustriert, wenn sie mit Erweiterungsplatinen arbeiten müssen, die in Bezug auf Konnektivität und Schutzfunktionen unzureichend sind. Dieses Erweiterungsboard für die winzigen XIAO-Mikrocontroller-Breakout-Boards von Seeed Studio [1] behebt diese Probleme und bietet Vielseitigkeit und robuste Funktionen.

Unsere Platine ist die perfekte Lösung für unser Projekt *Modulares DC-Leistungsmessgerät* [2], aber natürlich kann und soll die eXpansion-Platine in vielen verschiedenen Projekten eingesetzt werden, bei denen Sie einen ESP32-C3 oder -S3 (oder einen anderen Controller aus der XIAO-Familie – siehe **XIAO-Controllerboards**) und nicht zu viele Pins benötigen. Die XIAO-Boards bieten Ihnen insgesamt elf GPIO-Pins (D0…D10), die unter anderem als SPI, I²C und UART sowie als digitale und analoge Ein- und Ausgänge dienen. Die XIAO-Boards sind untereinander pinkompatibel. Die I²C-Pins finden Sie zum Beispiel immer an der Position D4 und D5. Das macht das Prototyping noch einfacher!

Dabei stehen D4 und D5 für die Position des Pins auf dem XIAO-Board, nicht für GPIO-Pin 4 und -Pin 5 der MCU. Wenn Sie in der Arduino-IDE programmieren, hängt die Pin-Nummer für Funktionen wie `digitalRead(...)` von der verwendeten XIAO-Platine ab - bitte beachten Sie **Tabelle 1** und die Dokumentation der XIAO-Platine.

Das Schaltbild

Die Erweiterungskarte besticht durch ihr klares, funktionales Design, das auf eine maximale Konnektivität und Funktionalität ausgelegt ist. Die Schaltung in **Bild 1** zeigt als Hauptmerkmal selbstverständlich die Erweiterungsverbinder, die die elf GPIO-Pins (D0…D10) eines XIAO-Boards Ihrer Wahl leicht zugänglich machen. Die XIAO-Boards sind Teil des Grove-Ökosystems [3] und bieten Ihnen eine große Auswahl an Grove-Peripheriemodulen. Deshalb finden Sie eine Vielzahl von Grove-Anschlüssen auf dem Board – und noch viel mehr. Dies macht unser Board einzigartig auf dem Markt.

XIAO-Controllerboards

Derzeit sind sieben Boards mit Controllern verschiedener Hersteller verfügbar. Erfahren Sie mehr über ihre Spezifikationen in [1].

- XIAO ESP32S3 Sense
- XIAO ESP32S3
- XIAO ESP32C3
- XIAO nRF52840 Sense
- XIAO nRF52840
- XIAO RP2040
- XIAO SAMD21

Tabelle 1: Pin-Verbindungen des XIAO-Boards mit den GPIO-Nummern des Controller-Boards

XIAO-Board Pin	ESP32-C3	SMD21	ESP32-C6	ESP32-S3
D0	GPIO2	D0	GPIO0	GPIO1
D1	GPIO3	D1	GPIO1	GPIO2
D2	GPIO4	D2	GPIO2	GPIO3
D3	GPIO5	D3	GPIO21	GPIO4
D4 (SDA)	GPIO6	D4	GPIO22	GPIO5
D5 (SCL)	GPIO7	D5	GPIO23	GPIO6
D6 (TX)	GPIO21	D6	GPIO16	GPIO43
D7 (RX)	GPIO20	D7	GPIO17	GPIO44
D8	GPIO8	D8	GPIO19	GPIO7
D9	GPIO9	D9	GPIO20	GPIO8
D10	GPIO10	D10	GPIO18	GPIO9

Wenn Sie in der Arduino-IDE programmieren, wählen Sie die richtige XIAO-Variante, und wenn Sie Pins definieren, verwenden Sie die entsprechenden GIPO-Nummern, um den richtigen Pin auf dem Board zu wählen.
Möchten Sie zum Beispiel, dass ein ESP32-C3 die LED1 (angeschlossen an D3) auf dem eXpansion-Board blinken lässt, müssen Sie die entsprechende GPIO-Nummer verwenden, 5 oder D3, wie im folgenden Codeschnipsel gezeigt, um LED1 als Ausgang zu definieren.

```
#define LED D3 oder #define LED 5
// As GPIO5 corresponds with D3

void setup() {
  pinMode(LED, OUTPUT);
}
void loop () {
  digitalWrite(LED, HIGH);
  delay(1000);
  digitalWrite(LED, LOW);
  delay(1000);
}
```

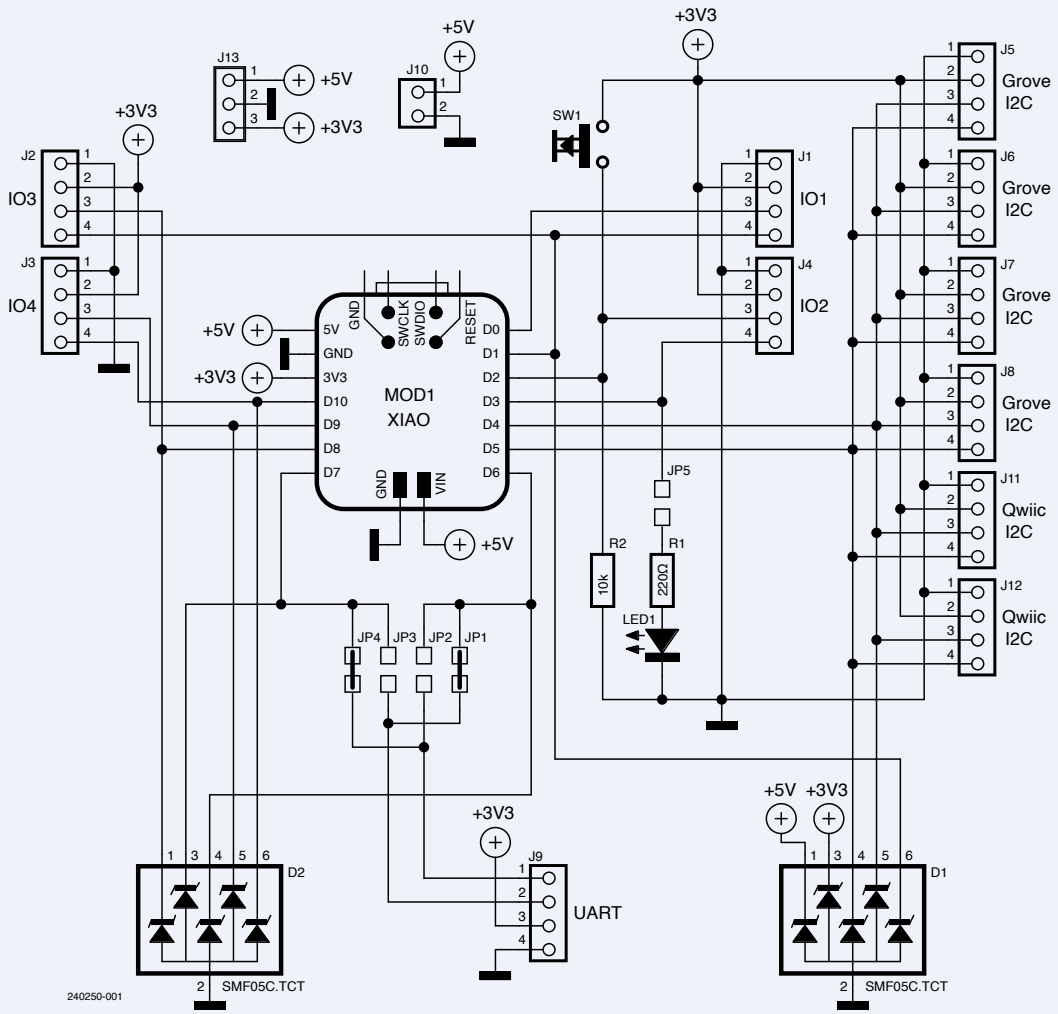


Bild 1. Schaltbild des eXpansion Board V1.0.

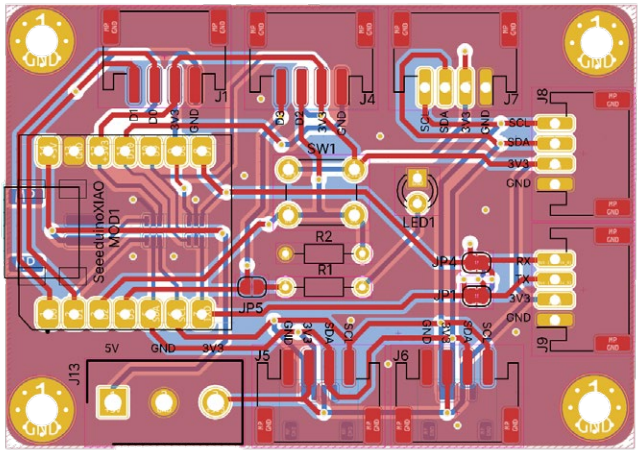


Bild 2. Die Lagen des Platinenlayouts.

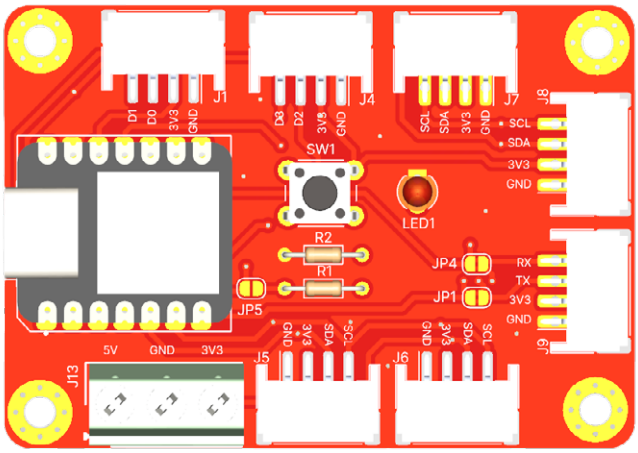


Bild 3. Das Layout der Ober- und Unterseite der Platine.

Es gibt eine unglaubliche Anzahl von I²C-Chips und -Modulen, von Sensoren über Aktoren bis hin zu Elementen mit Benutzerschnittstellen wie Displays. Wie Sie vielleicht wissen, können viele I²C-Module parallel an die I²C-Schnittstelle eines Controllers angeschlossen werden. Es ist lediglich darauf zu achten, dass sie unterschiedliche I²C-Adressen haben. Daher haben wir darauf geachtet, eine Vielzahl von I²C-Pins (D4 und D5) bereitzustellen. Wir haben uns für vier Grove I²C-Anschlüsse (J5…J8) [4] und zwei I²C-Qwiic-Anschlüsse (J11 und J12) [5] entschieden. Diese Steckverbinder ermöglichen Ihnen eine mühelose Integration verschiedener Grove- und Qwiic-I²C-Module (zum Beispiel [6]). Sie können selbstverständlich auch andere Steckverbinder einlöten, die in das 2-mm-Raster passen, zum Beispiel 4-polige PH2.0-Steckverbinder von JST.

Selbstverständlich sind nicht alle Peripheriemodule über den I²C-Bus verbunden. Sensoren werden beispielsweise oft über eine Art Ein-Draht-Schnittstelle angeschlossen. Daher haben wir auch die Pins D0…D3 und D8…D10 an einige Grove-Stecker angeschlossen. D8, D1, D9 und D10 bilden zudem eine vollständige SPI-Schnittstelle mit Chip Select (CS). Diese Pins sind an J2 und J3 verfügbar, und wenn Sie auf sie zugreifen wollen, müssen Sie beide Anschlüsse gleichzeitig verwenden.

J9 bildet zusammen mit D6 und D7 einen Grove-kompatiblen UART-Anschluss. Um ihn noch vielseitiger zu machen, sind die RX- und TX-Pins über Jumper (JP1…JP4) austauschbar. Dies erhöht die Flexibilität des Boards und ermöglicht eine robuste serielle Kommunikation nicht nur für „periphere" UART-Module wie zum Beispiel einen RS-232- oder RS-485-Adapter oder ein Grove GPS-Modul [7]. Mit vertauschten Pins können Sie hier auch ein anderes Controller-Board mit Grove-UART-Anschluss oder sogar ein zweites eXpansion-Board anschließen, um miteinander zu kommunizieren. Um die RX/TX-Anschlüsse zu vertauschen, trennen Sie einfach die Lötbrücken JP1 und JP4 auf und verbinden Sie die Brücken JP2 und JP3.

Die Platine verfügt zudem über eine LED1 und einen Taster SW1 für rudimentäre Ein- und Ausgangsfunktionen. Diese beiden Elemente sind mit den Anschlüssen D2 beziehungsweise D3 verbunden, und zwar auf folgende Weise: Die LED dient als Statusanzeige, während der Taster für Benutzereingaben verwendet wird. Die LED1 kann durch Auflösen des JP5-Jumpers entkoppelt werden. So kann der Benutzer die LED deaktivieren, wenn sie nicht benötigt wird, und hat freien Zugriff auf den entsprechenden GPIO-Pin.

Zusätzlich haben wir einen ESD-Schutz für die 3,3-V-, 5-V-, I2C- und SPI-Leitungen hinzugefügt – und zwar mit den TVS-Dioden SMF05C.TCT (D1 und D2). Mehr dazu weiter unten.

Die Platine besitzt auch eine Schraubklemme für 3,3 V, 5 V und Masse, die für die Stromversorgung anderer elektronischer Module und Geräte verwendet werden kann. Für den Anschluss einer Batterie ist ein 2-mm-Steckverbinder verfügbar, wodurch sich die Platine sowohl für stationäre als auch für portable Geräte eignet.

Platinenlayout

Das Leiterplattenlayout ist in **Bild 2** dargestellt. Um die Platine kompakt zu halten, ohne dabei Abstriche bei der Konnektivität zu machen, wurden I²C- und IO-Anschlüsse auch auf der Rückseite der Platine platziert (**Bild 3**). Dies wurde durch die Verwendung von SMD-Steck-

Stückliste

Widerstände:

R1 = 220 Ω

R2 = 10 k

Halbleiter:

D1, D2 = SMF05C.TCT

LED1 = 3 mm 3 V LED

MOD1 = Mikrocontrollerboard aus der XIAO-Reihe

Sonstige:

J1…J9 = 4-poliger Grove-Verbinder (Raster 2 mm)

J10 = 2-poliger JST-PH-Verbinder (Raster 2 mm)

J11, J12 = 4-poliger Qwiic-Verbinder (Raster 1 mm)

J13 = 3-polige Schraubklemmblock (Raster 5,08 mm)

SW1 = 6-mm-Drucktaster

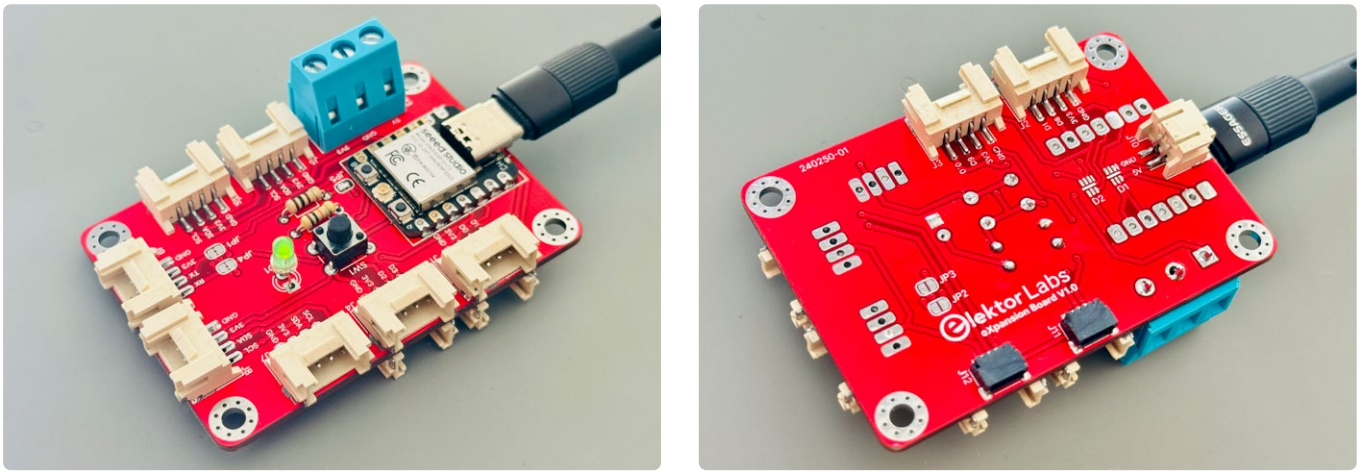


Bild 4: Die Elektor eXpansion Platine V1.0

verbindern anstelle von gewöhnlichen Verbindern mit Durchsteckkontakten (THT) möglich. Für den UART und zwei der I_2C-Anschlüsse wurden außerdem THT-Pads hinzugefügt, so dass THT-Steckverbinder verwendet werden können, wenn keine SMD-Steckverbinder mehr zur Verfügung stehen.
Die Anordnung der Bauteile und Steckverbinder sorgt dafür, dass die Platine kompakt und dennoch hoch funktional bleibt und eine Vielzahl von Projektanforderungen erfüllt, wie in **Bild 4** dargestellt. Für das eXpansion Board haben wir selbstverständlich auch ein 3D-gedrucktes Gehäuse entworfen, dass Sie in **Bild 5** sehen können. Das Gehäuse bietet selbstverständlich Zugang zu allen Anschlüssen und lässt sich daher ohne Probleme in jedes Projekt integrieren. In **Bild 6** sehen Sie das eXpansion Board mit einem LiPo-Akku in Aktion.

Die Bedeutung von ESD-Schutz

Elektronische Geräte sind oft verschiedenen Umweltbelastungen ausgesetzt. Eine davon ist die elektrostatische Entladung (ESD). ESD bedeutet, dass sich Elektronen plötzlich von einem Gegenstand zum anderen bewegen, was meistens passiert, wenn zwei Gegenstände miteinander in Kontakt kommen. Ohne Schutz können Mikrocontroller und andere empfindliche Teile beschädigt oder gar zerstört werden. Mikrocontroller sind empfindlich, weil sie kompliziert aufgebaut sind. Ein ESD-Ereignis kann die Spannungstoleranz der MCU überschreiten und zu Ausfällen führen, so dass Geräte nicht mehr richtig funktionieren, Daten falsch sind oder Geräte komplett ausfallen. Das kann in wichtigen Bereichen wie der industriellen Automatisierung, bei medizinischen Geräten oder Außensensoren große Probleme verursachen. Um diese Risiken zu vermindern, ist die Erweiterungsplatine mit einem ESD-Schutz für die 3-V-, 5-V-, I^2C- und SPI-Leitungen ausgestattet. Dieser Schutz sorgt dafür, dass elektrostatische Entladungen, die im Feld auftreten, sicher abgeleitet werden und die empfindliche MCU nicht erreichen und beschädigen können. Der ESD-Schutz ist entscheidend, um die Integrität und Zuverlässigkeit des Geräts in realen Anwendungen aufrecht zu erhalten.

Bild 5. 3D-gedrucktes Gehäuse.

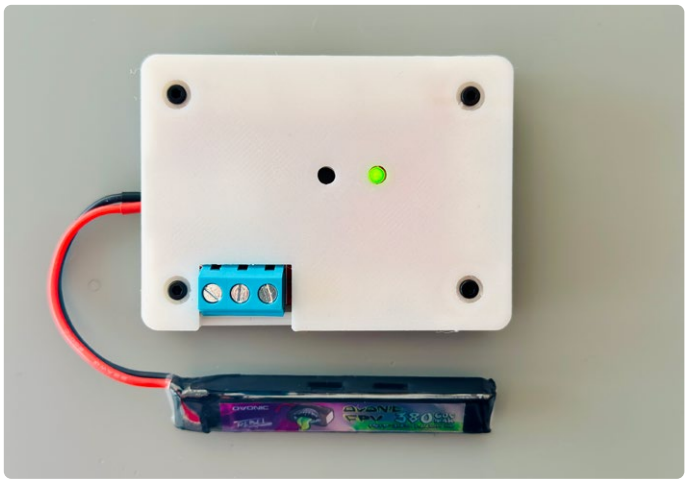


Bild 6. Das eXpansion Board wird von einem LiPo-Akku versorgt.

Zukünftige Erweiterungen und Anwendungen

Diese Erweiterungsplatine ist mit allen Mikrocontrollern der XIAO-Produktreihe kompatibel. Es ist ein Game-Changer für Entwickler, Pädagogen und Bastler gleichermaßen, insbesondere für schnelles Prototyping, Bildungsprojekte und IoT-Lösungen, da es umfangreiche Anschlussmöglichkeiten und robuste Stromversorgungsfunktionen bietet. Alle KiCad-Dateien und Produktionsdateien sind auf dem GitHub-Repository von Elektor Lab für dieses Projekt verfügbar [8].

In Zukunft planen wir, die Erweiterungsplatine für die Entwicklung verschiedener Projekte zu verwenden und dabei die zahlreichen I²C- und Qwiic-Anschlüsse zu nutzen, um verschiedene Sensoren und Module nahtlos zu integrieren. Da die XIAO-ESP32-Serie zudem WLAN-fähig ist, eignet sich das Board ideal für Fernsteuerungs- und IoT-Anwendungen. Es erfüllt aktuelle Anforderungen und ist eine gute Grundlage für neue Entwicklungen in der Elektronik und bei Mikrocontrollern.

SE – 240250-02

Haben Sie Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Dann wenden Sie sich bitte an den Autor unter saad.imtiaz@elektor.com oder an die Elektor-Redaktion unter redaktion@elektor.de.

Passende Produkte

- **Seeed Studio XIAO ESP32C3**
  www.elektor.de/20265
- **Seeed Studio XIAO ESP32S3 Sense**
  www.elektor.de/20705
- **Seeed Studio Arduino Sensor Kit (Basis)**
  www.elektor.de/20068

WEBLINKS

[1] Seeed Studio XIAO: https://wiki.seeedstudio.com/SeeedStudio_XIAO_Series_Introduction

[2] Saad Imtiaz, „Das modulare DC-Leistungsmessgerät AmpVolt, Teil 1", Elektor 5-6/2024: https://www.elektormagazine.de/magazine/elektor-342/62846

[3] Einführung in das Grove-Ökosystem: https://wiki.seeedstudio.com/Grove_System/

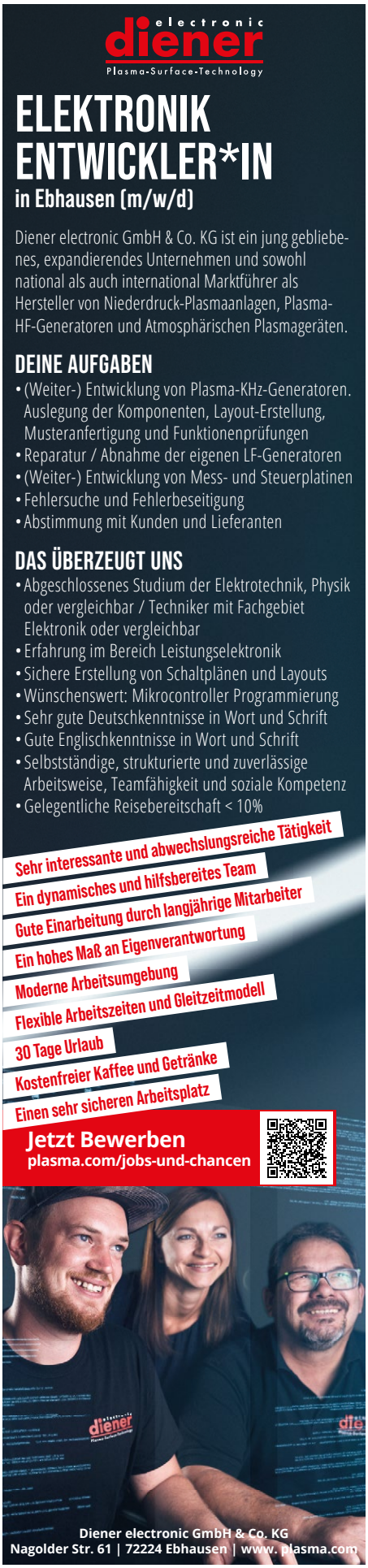
[4] 10 Stück Universeller vierpoliger Verbinder (Raster 2 mm) für Grove von Seeed Studio: https://elektor.de/18671

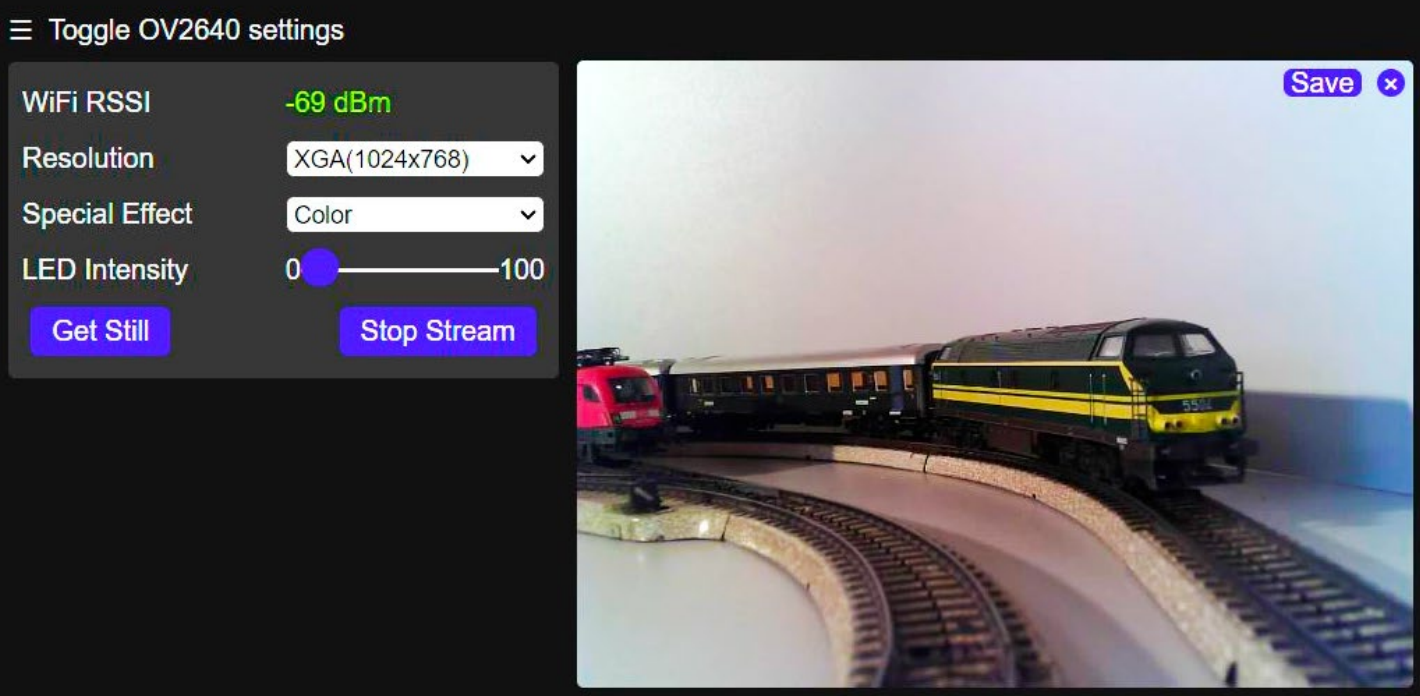
[5] Horizontaler 4-poliger JST-SMD-Steckverbinder für Qwiic: https://sparkfun.com/products/14417

[6] Umweltsensor-Breakout CCS811/BME280 für Qwiic von SparkFun: https://elektor.de/19580

[7] GPS-Modul für Grove von Seeed Studio: https://elektor.de/20022

[8] Elektor eXpansion Board v1.0, Github-Repo: https://github.com/ElektorLabs/eXpansion-board





Modelleisenbahn mit Kamera

Einbau eines ESP32-CAM-Moduls

Von Gilbert Ghyselbrecht (Belgien)

Auf die Frage, ob es möglich wäre, eine Kamera in die Lokomotive einer Modelleisenbahn einzubauen, dachte ich sofort an ein ESP32-CAM-Modul. Es ist überall sehr billig und mit unterschiedlichen Linsen/Kamera-Ausführungen zu bekommen, wobei die Kameras über verschieden lange Folienkabel zum Anschluss an das Modul verfügen. In diesem Artikel geht es um den Einbau, eine Basisplatine zur Stromversorgung, und die nötige Software.

Bild 1. Eine geeignete Lokomotive für das Kameramodul.

Zunächst musste ich eine geeignete Lokomotive suchen, die genügend Platz für das ESP32-CAM-Modul bietet. Die Lokomotive sollte vorzugsweise eine Kunststoffhaube haben, wenn man die WLAN-Antenne im Innenraum unterbringen möchte. In der HO-Serie von Märklin habe ich ein geeignetes Modell gefunden, nämlich die Elektrolokomotive vom Typ B103 (**Bild 1**). Nachdem ich einen anderen Ort zur Unterbringung des digitalen Zugdekoder-Moduls gefunden hatte, wurde genügend Platz für das Kameramodul und eine selbst entwickelte Basisplatine frei (**Bild 2**).

Stromversorgung

Eine zweite Herausforderung war es, eine zuverlässige 5-V-Stromversorgung für den ESP32 sicherzustellen. Wir gehen hier von digital gesteuerten Gleisen aus. Bei einem altmodischen analogen Gleis mit Transformator liegt auf den Gleisen keine Spannung, wenn der Zug steht. Bei einem digitalen System hingegen liegt auf den Gleisen immer eine Wechselspannung von etwa 18 V, die von der Steuereinheit kommt. Welches System oder welche Marke verwendet wird, spielt dabei keine Rolle.

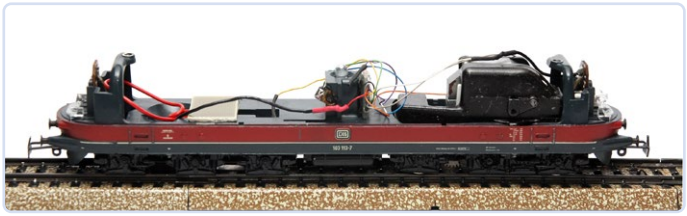


Bild 2. Die Lokomotive mit freiem Platz für das Kameramodul.

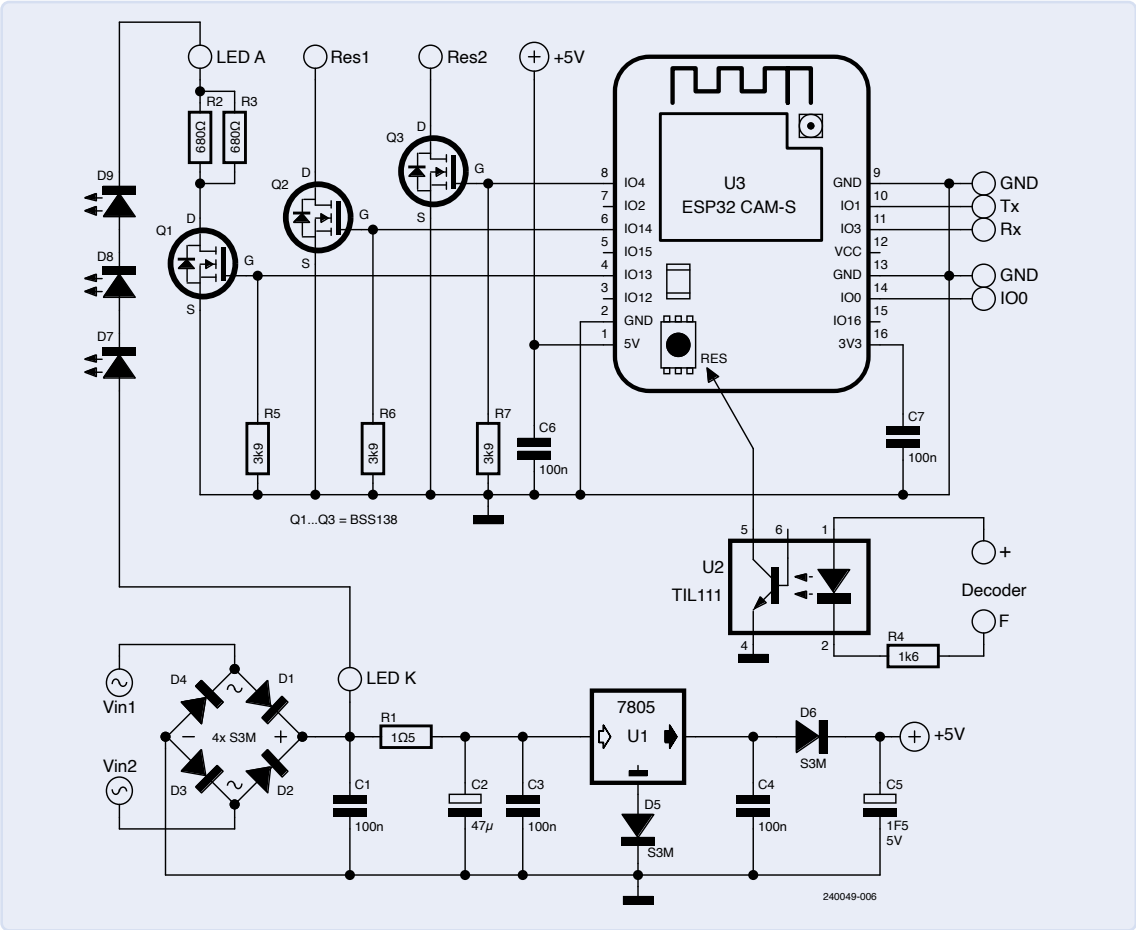


Bild 3. Schaltplan der Stromversorgungsplatine.

Die Stromaufnahme erfolgt über die Räder des Zuges (beziehungsweise beim Märklin-System über die Räder und einen Schleifkontakt an der Mittelschiene). Allerdings ist diese Stromversorgung bei einem fahrenden Zug nicht 100%ig stabil, da an Weichen oder bei Gleisverschmutzungen manchmal kurze Unterbrechungen auftreten können. Dies kann dazu führen, dass sich der ESP32-Prozessor zurücksetzt oder ein Brownout auftritt: Die WLAN-Verbindung wird unterbrochen und alles muss neu gestartet werden.

Die Idee, eine kleine Batterie als „Notstrom-Versorgung" zu verwenden, habe ich schnell verworfen, da dies auch eine Ladeschaltung erfordert, um die Batterie sicher aufzuladen. Dazu käme eine Schaltung, um die Stromversorgung im Laufe der Zeit abzuschalten, damit verhindert wird, dass sich die Batterie vollständig entlädt. Aufgrund des begrenzten Platzes in der Lokomotive ist dies alles zusammen nicht umsetzbar. Eine Lösung wäre, die Batteriestromversorgung in einem separaten Waggon unterzubringen und diesen hinter die Lok zu kuppeln, also quasi die Wiedergeburt des Schlepptenders.

Nach zahlreichen Versuchen entschied ich mich dann doch für einen 5-V-Superkondensator mit 1,5 F und den Abmessungen 25 × 17 × 9 mm [1]. Damit funktioniert die Kamera nach einer Unterbrechung der Stromversorgung etwa acht Sekunden lang normal weiter, was bei normaler Nutzung für einen zuverlässigen Betrieb ausreicht.

Die Schaltung

Bild 3 zeigt den vollständigen Schaltplan. Links unten wird die 18-Volt-Versorgung der Schienen an den Brückengleichrichter (D1…D4) angeschlossen. Der Rest der Schaltung darf in keiner Weise mit irgendwelchen Metallteilen der Lokomotive in Berührung kommen!

Der 1,5-Ω-Widerstand R1 begrenzt übermäßige Spitzenströme und dient gleichzeitig als Sicherung. Der Spannungsregler U1, ein gewöhnlicher 7805, ist über ein Isolierplättchen mit der Metalloberfläche der Lokomotive verbunden, die der Wärmeabfuhr dient. Die Diode D5 in der Masseleitung des 7805 kompensiert den Verlust in der Seriendiode D6 zum Superkondensator C5, der dadurch auf 5 V aufgeladen wird. Der Superkondensator wird sehr schnell, in etwa 3 s geladen, wobei der Kurzschlussschutz des 7805-Reglers den Strom auf etwa 1,5 A begrenzt (**Bild 4**).

Für die Stromversorgung wurde eine einseitige Platine entworfen, die den Platzverhältnissen in der Lok gerecht wird. Hauptsächlich sind THT-Bauteile vorgesehen, die aber auf der Oberfläche montiert

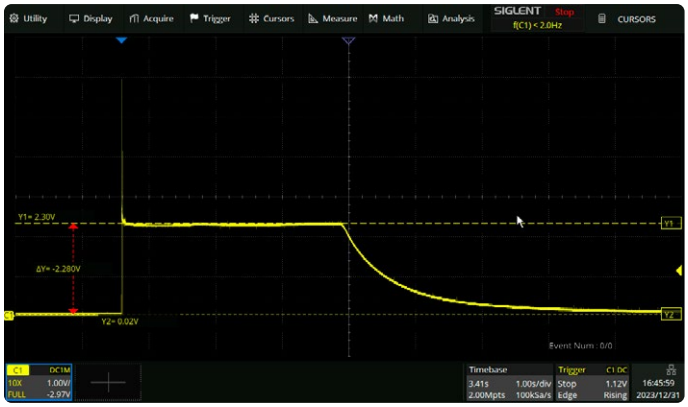


Bild 4. Aufladen des Superkondensators, gemessen an R1.

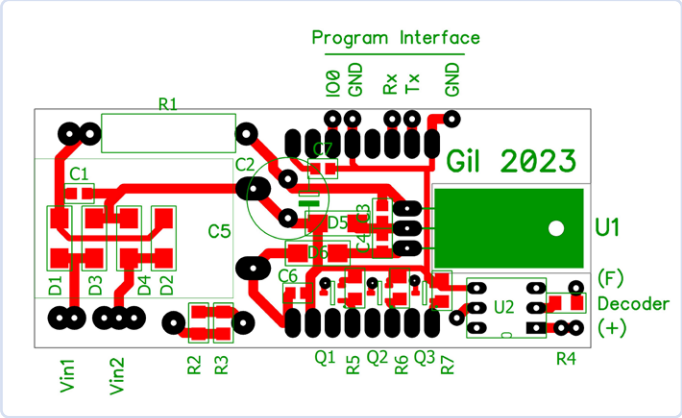


Bild 5. Platinenentwurf für die Stromversorgung.

Bild 7. Drei weiße LEDs an der Vorderseite der Lokomotive.

werden. Die Platine wird mit der Kupferseite nach oben auf das Chassis montiert und mit dem umgebogenen 7805-Regler fixiert (siehe **Bild 5** und **Bild 6**).
Von zwei achtpoligen Stiftleisten werden die Stifte umgebogen und auf der Platine verlötet, um das ESP32-CAM-Modul einzustecken. Von der Kamera OV2640 habe ich zwei Typen mit einem 75 mm langen Anschlusskabel bestellt, eine mit einem Objektiv mit 66° Aufnahmewinkel und die andere mit 120° Aufnahmewinkel. Je nach verfügbarem Platz wird die geeignete Kamera an der Vorderseite der Lokomotive hinter der Windschutzscheibe eingebaut. Das originale Kunststofffenster des Zuges vor der Kamera muss allerdings entfernt werden, da es nicht ausreichend transparent ist.

Ausgänge und mehr

An der Vorderseite der Lokomotive werden drei helle weiße LEDs (D7…D9) angebracht, um das Gleis in dunklen Abschnitten und Tunneln ausreichend zu beleuchten (**Bild 7**). Die drei LEDs sind in Reihe geschaltet und werden von einem PWM-Signal (Pin IO13 des ESP-Moduls) über den MOSFET-Puffer Q1 (ein BSS138) und einen Serienwiderstand von 340 Ω (zwei 680-Ω-SMDs parallel) angesteuert. Die Helligkeit der LEDs lässt sich vom ESP32 aus einstellen.
Es stehen zwei weitere Ausgänge mit BSS138-Puffern für andere Anwendungen zur Verfügung, die über IO14 und IO4 ansteuerbar sind.

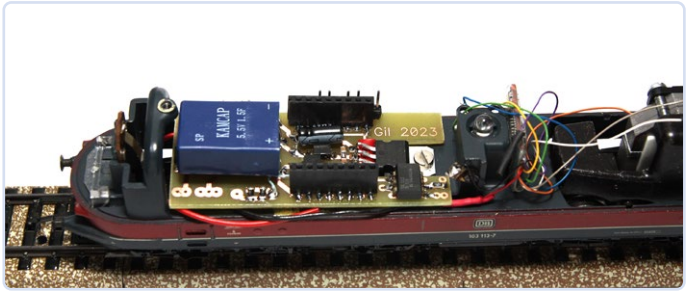


Bild 6. Die Stromversorgungsplatine [6] in der Lokomotive wird mit dem 7805 fixiert.

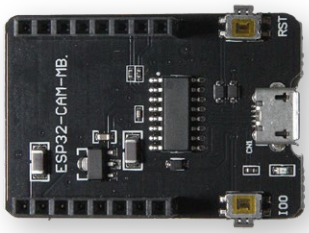


Bild 8. Programmiermodul für die ESP32-CAM.

Die Stromaufnahme des Moduls beträgt bei aktiver WLAN-Verbindung 250 mA und sinkt im Reset-Zustand auf 15 mA.

Für die Programmierung über einen USB-FTDI-Adapter ist eine spezielle 3-polige Stiftleiste vorhanden. Verbinden Sie IO0 mit GND und drücken Sie den Reset-Taster auf dem Modul, um es in den Programmiermodus zu versetzen. Einfacher ist es, das Modul aus der Lokomotive herauszunehmen und in ein ESP32-CAM-Programmierboard (ESP32-CAM-MB) zu stecken, um es zu programmieren (**Bild 8**). Dieses spezielle Modul wird über USB mit der Arduino-IDE verbunden und enthält die notwendigen *BOOT*- und *RST*-Tasten für die ESP32-CAM.

Sobald das ESP32-CAM-Modul im Zug montiert ist, ist die Reset-Taste auf dem Modul selbst nicht mehr zugänglich. Daher wird ein zusätzliches Kabel an die Reset-Taste gelötet und mit Pin 5 des Optokopplers U2 auf der Basisplatine verbunden (**Bild 9**). Eine andere Möglichkeit ist, einen zusätzlichen Reset-Taster anzubringen, der von außen zugänglich ist. Vom Zugdekoder wird der AUX-Ausgang zur Ansteuerung von

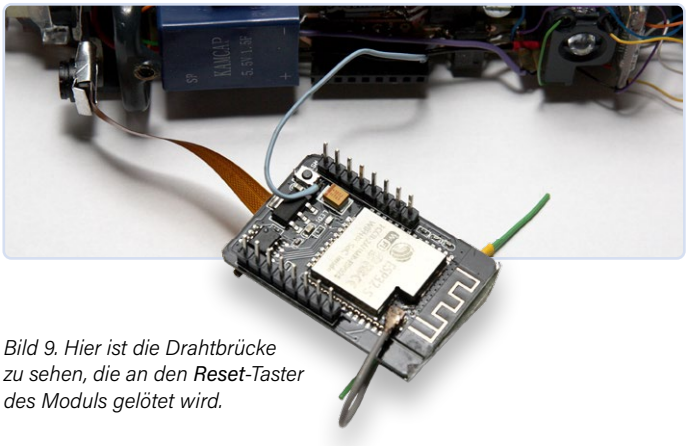


*Bild 9. Hier ist die Drahtbrücke zu sehen, die an den **Reset**-Taster des Moduls gelötet wird.*

U2 verwendet, der den Reset-Eingang des ESP-Moduls auf Low setzt, wodurch die Kamera ausgeschaltet oder neu gestartet wird.

Ich habe die WLAN-Antenne auf dem ESP-Modul durch eine kleine Dipolantenne aus einem Stück Draht ersetzt - angeschlossen über ein kurzes Koaxialkabel mit einem passenden klitzekleinen IPEX-Anschluss. Ein solches Kabel gehört bisweilen zum Lieferumfang des ESP-Moduls; man kann es (mit SMA-Verbinder auf der anderen Seite) aber auch leicht im Fachhandel erstehen. An der Modulplatine muss eine kleine Änderung vorgenommen werden, nämlich eine Lötbrücke neben dem Antennenstecker unterbrochen und eine andere gesetzt werden [2]. Die Wellenlänge bei 2,4 GHz beträgt 12,5 cm. Der Dipol besteht aus zwei 3 cm langen Drahtstücken, wobei ein Stück mit dem Innenleiter und das andere Stück mit der Abschirmung des Koaxkabels verbunden wird. Ein Schrumpfschlauch hält alles an seinem Platz. Die Antenne kann entweder innerhalb oder außerhalb der Lokomotive angebracht werden, jedoch besser nicht zu nahe am Motor oder anderen Metallteilen (**Bild 10**).

Software

Das Kameramodul wird mit der Arduino-IDE 1.8 oder 2.0 programmiert. Eine detaillierte Installationsanleitung für die ESP32-CAM finden Sie unter [3]. Nach der Installation öffnen Sie in der Arduino IDE *Datei → Beispiele* und wählen *ESP32 → Camera → CameraWebServer*. Nachdem man die Felder *SSID* und *PASSWORD* angepasst hat, wird der Code kompiliert und in das Kameramodul geladen.
Stellen Sie danach die Lokomotive auf die Schienen und schalten Ihre Modelleisenbahn-Steuerung ein. Über das Steuergerät aktiviert und deaktiviert man den AUX-Ausgang, um die ESP-CAM zurückzusetzen und zu starten. Das Modul verbindet sich mit dem heimischen WLAN.

Im Browser Ihres Computers geben Sie nun die IP-Adresse ein, die der Router für das Modul vergeben hat (normalerweise 192.168.1.xxx oder 198.168.0.xxx) und verbinden sich mit der Kamera. Die richtige Adresse wird im Seriellen Monitor der Arduino-IDE angezeigt, der auf 115200 Baud eingestellt ist. In der Beispiel-Testsoftware können Sie mit allen Einstellungen des Kameramoduls spielen, allerdings ist das Programm komplex und mitunter schwer zu verstehen. Daher habe ich ein einfaches Programm entwickelt [6], das nur die wichtigsten Einstellungen enthält und bei dem die Seiten der Webschnittstelle in separaten (lesbaren und zugänglichen) Header-Dateien ausgelagert sind.

Wenn die Verbindung zum WLAN fehlschlägt, schaltet das Modul in den AP-Modus unter der Adresse 192.168.4.1 mit der SSID *ESP32-CAM* (ohne Passwort) und ermöglicht so die Verbindung mit einem Browser ohne Router oder Internet.

Die Webseite wird mit Hilfe der drei Dateien erstellt, die in den Header-Dateien mitkompiliert wurden:

1. *index.html* in `#include "indexFile.h"`
2. *espcam.css* in `#include "cssFile.h"`
3. *espcam.js* in `#include "jsFile.h"`

Sobald das Modul gestartet ist und die WLAN-Verbindung steht, leuchten die weißen LEDs an der Vorderseite der Lok sehr schwach auf. Das Webinterface, das im Titelbild zu sehen ist, bietet die Möglichkeit, ein Bild aufzunehmen, es zu speichern oder einen Videostream zu starten. Die Bildauflösung kann eingestellt werden, wobei eine niedrigere Auflösung zu einer höheren Bildrate des Streams führt.

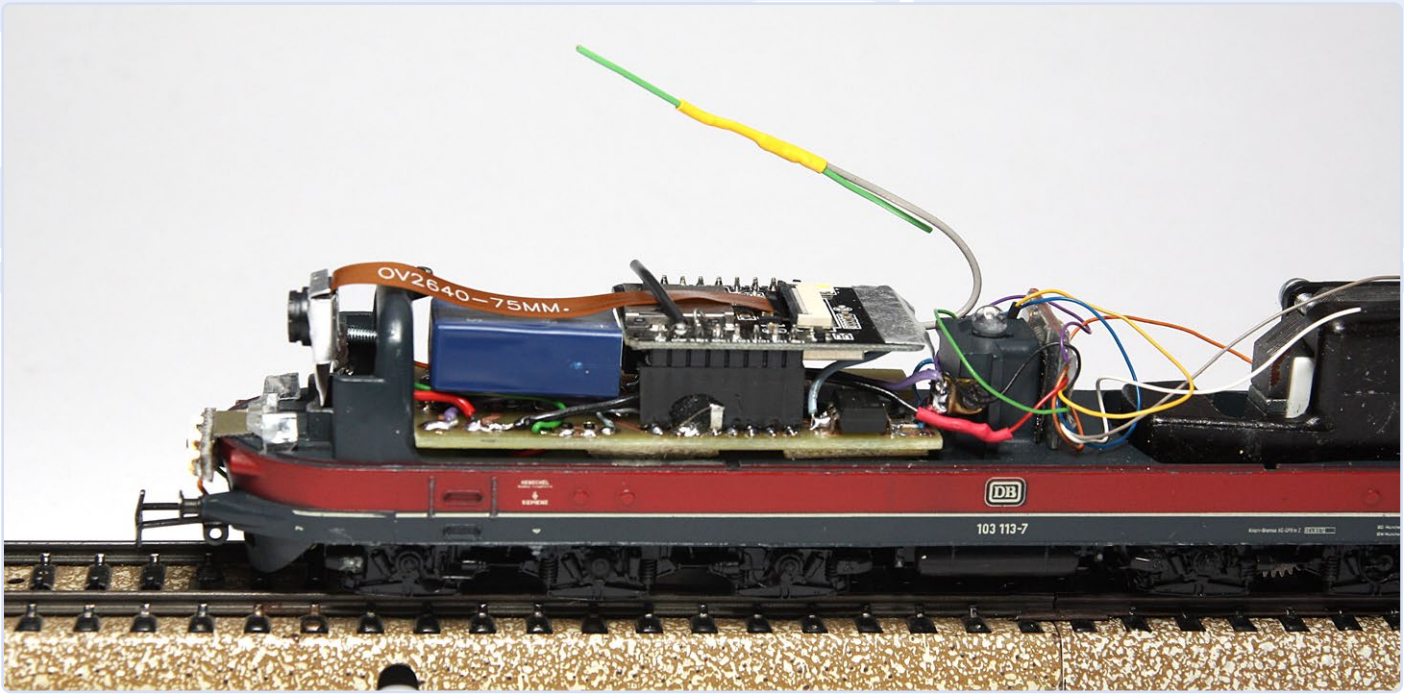


Bild 10. Fertiges Projekt mit Dipolantenne.

Videostream

Bei dem Videostream handelt es sich um einen MJPEG-Stream [4], der vom Client (dem Webbrowser) abhängt. Dieser hält die Verbindung offen und fordert Bild für Bild über JavaScript an. Die Geschwindigkeit des Browsers und die Qualität der WLAN-Verbindung bestimmen, wie viele Bilder pro Sekunde verarbeitet werden. Falls gewünscht, kann dies mit einem Oszilloskop an Pin IO12 überwacht werden.
Es gibt ein Menü zur Auswahl von Farb-, Schwarz-Weiß- oder Sepia-Bildern, und schließlich kann die Helligkeit der weißen LEDs an der Vorderseite der Lokomotive mit einem Schieberegler eingestellt werden. Beachten Sie, dass die LEDs bei maximaler Einstellung sehr grell leuchten.

Zum Schluss möchte ich auf das zusätzliche, noch mächtigere Programm *esp32-cam-webserver* auf Github [5] verweisen, das alle Funktionen der Kamera zur Verfügung stellt und es ermöglicht, das Programm Over The Air (OTA) direkt in die Lokkamera zu laden, ohne die Lok zu öffnen oder von den Gleisen nehmen zu müssen.

Dieses Projekt

Es handelt sich nicht um ein fertiges Produkt, da vieles vom spezifischen Lokmodell abhängt, in das Sie die Kamera integrieren möchten. Das Konzept der unterbrechungsfreien Stromversorgung (USV) mit einem Superkondensator könnte jedoch auch für andere Anwendungen interessant sein.

SG — 240049-02

Sie haben Fragen oder Kommentare?

Haben Sie technische Fragen oder Anmerkungen zu diesem Projekt? Dann zögern Sie bitte nicht, Kontakt mit dem Autor Gilbert Ghyselbrecht (gilbert.ghyselbrecht@skynet.be) oder der Elektor-Redaktion (redaktion@elektor.de) aufzunehmen.

Über den Autor

Gilbert arbeitete viele Jahre lang als Techniker beim maritimen Küstensender Oostenderadio. Während dieser Zeit war er für die Wartung der Sendeanlagen zuständig. Darüber hinaus konzentrierte er sich hauptsächlich auf die Entwicklung und den Bau von Fernsteuerungssystemen, die die Fernsteuerung der Geräte ermöglichten. Sein Fachwissen begann mit dem 8085-Mikroprozessor und der CP/M-Software und reicht nun bis zu fortgeschrittenen VoIP-Steuerungen.

Passende Produkte

- **ESP32-Cam-CH340 Entwicklungsboard**
  www.elektor.de/19333

  Bei dieser Version des ESP32-Cam-Moduls benötigt man keinen extra Programmieradapter, aber die Maße sind etwas größer. Bitte vorher checken!

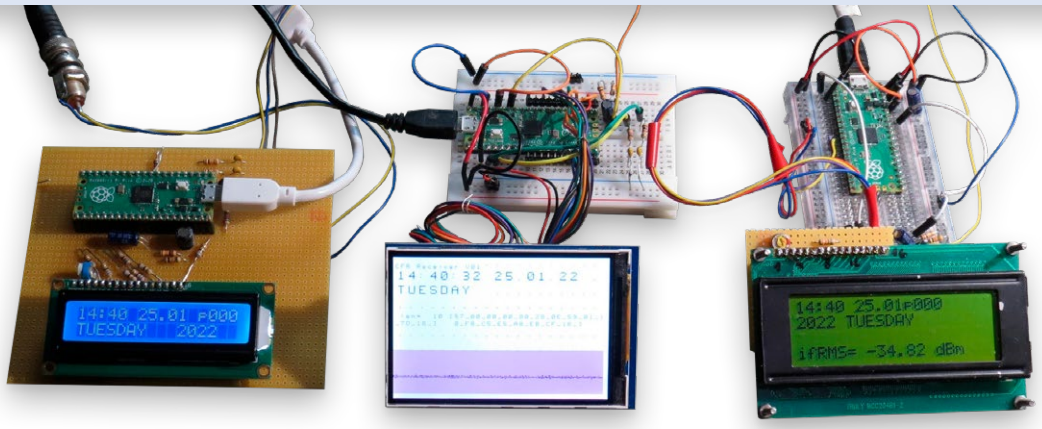
- **Jürgen Köhler, *Elektronik & Modellbahn*, Elektor (Band 1 bis 4)**
  E-Bücher, deutsch, PDF: www.elektor.de/search?q=modellbahn

WEBLINKS

[1] Kamcap: 5.5V Fully Sealed (Waterproof) Supercapacitor: https://www.kamcappower.com/products/5.5v-fully-sealed-waterproof-super-capacitor/
[2] Random Nerds Tutorials: ESP32-CAM Connect External Antenna: https://randomnerdtutorials.com/esp32-cam-connect-external-antenna/
[3] Random Nerds Tutorial: ESP32-CAM Video Streaming … (Einzelheiten zum ESP32-CAM-Modul): https://randomnerdtutorials.com/esp32-cam-video-streaming-face-recognition-arduino-ide/
[4] Wikipedia: Motion-JPEG: https://de.wikipedia.org/wiki/Motion_JPEG
[5] GitHub: ESP32 CAM Webserver: https://github.com/easytarget/esp32-cam-webserver
[6] Platinenlayout und Software: https://www.elektormagazine.de/240049-02

Breitband-Magnetantenne für Langwelle

Mehrere Sender ohne Abstimmung



Paralleler Zeitzeichenempfang auf verschiedenen Frequenzen mit drei Empfängern an einer Antenne.

Von Prof. Dr. Martin Ossmann (Deutschland)

Eine Magnetantenne verwendet, anders als etwa ein Dipol, primär die magnetische (und nicht die elektrische) Komponente des elektromagnetischen Felds zum Empfang der elektromagnetischen Wellen. Wichtigstes Merkmal einer Magnetantenne ist ihre Breitbandigkeit. Die vorgestellte Magnetantenne vereint deshalb eine hohe Empfindlichkeit mit ebenso hohem Komfort, da man sie nicht mühsam auf die Empfangsfrequenz abstimmen muss.

Will man mit einer Ferritantenne oder einer Rahmenspule Sendungen im Lang- oder Mittelwellenbereich empfangen, baut man üblicherweise einen L/C-Schwingkreis auf (zum Beispiel mit einem Drehkondensator) und schafft damit eine abgestimmte Antenne, gemäß der Prinzipschaltung in **Bild 1**.
Der Vorteil dabei ist die hohe Empfindlichkeit durch die Resonanzüberhöhung. Die andere Seite der Medaille ist allerdings, dass man bei jedem Senderwechsel den Schwingkreis neu abstimmen muss. Wir zeigen nun, wie man eine Magnetantenne bauen kann, die man nicht abstimmen muss, weil sie breitbandig arbeitet. Eine Breitbandantenne erhält man, wenn man die Empfangsspule mit einem Transimpedanzverstärker kombiniert, wie in **Bild 2** gezeigt.
Der Transimpedanzverstärker hat (im Idealfall) die Eingangsimpedanz von 0 und schließt damit die Empfangsspule L kurz. Damit wird auch der Effekt der parasitären Wicklungskapazität eliminiert. Das Empfangssignal ist der Strom in der quasi kurzgeschlossenen Wicklung.

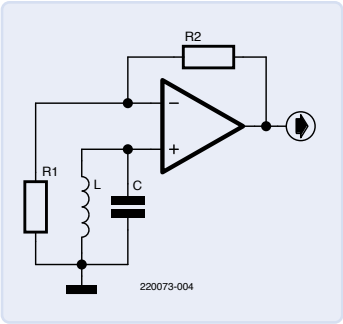


Bild 1.
Resonante Empfangsschaltung mit L/C-Schwingkreis und Opamp.

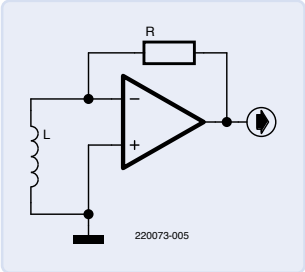


Bild 2.
Ein Transimpedanzverstärker schließt die Empfangsspule L effektiv kurz.

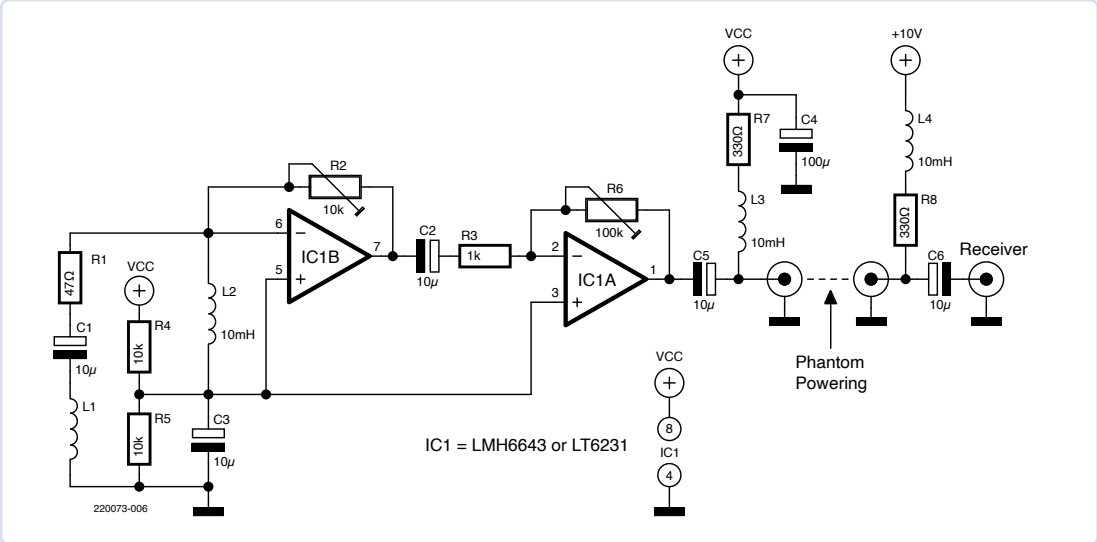


Bild 3. Vollständiger Schaltplan der realisierten Breitband-Antenne.

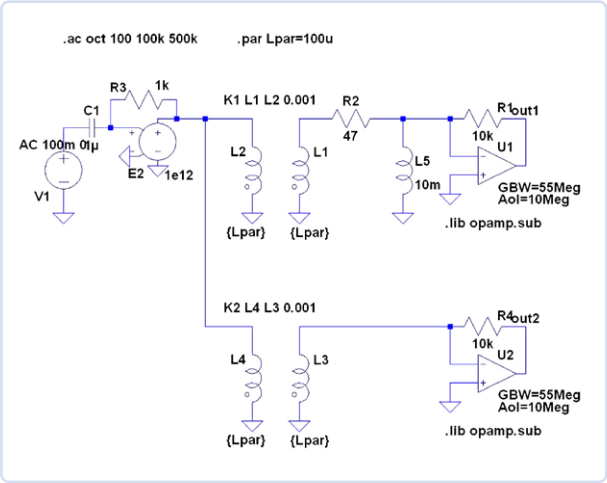


Bild 4. LTspice-Simulationsschaltung des Transimpedanzverstärkers.

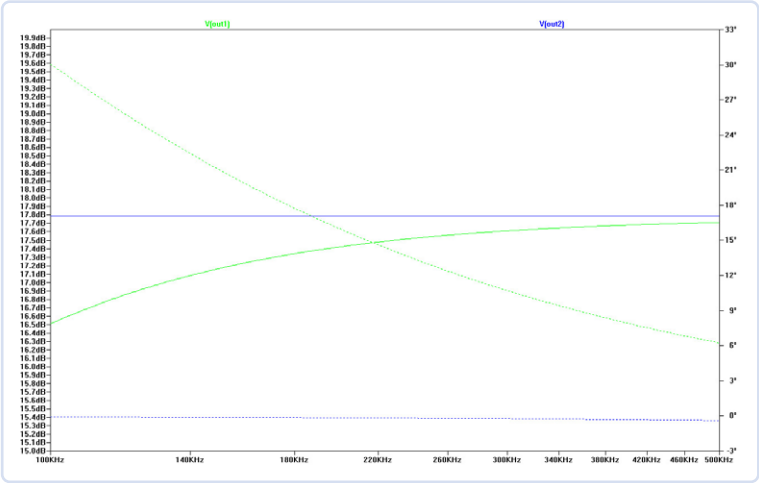


Bild 5. Frequenzgang der simulierten Schaltungen: grün ist die reale Schaltung, blau die ideale.

Schaltung mit Transimpedanzverstärker

Die praktische Schaltung mit zweistufiger Verstärkung und Phantomspeisung ist in **Bild 3** dargestellt. Verwendet man einen einfachen Transimpedanzverstärker wie in Bild 2, neigt die Schaltung zum Schwingen. Um dies zu unterdrücken, wurden die Komponenten R1 und L2 eingefügt, die die Schleifenverstärkung und damit die Schwingneigung verringern.

Dieses Konzept wurde auch in LTspice simuliert, um den Einfluss von R1 und L2 auf die Funktion der Schaltung zu untersuchen. Die Simulationsschaltung ist in **Bild 4** zu sehen (die Bauteilnummerierung entspricht *nicht* der aus Bild 3!). Der Hochpass aus C1 und R3 sowie E2 simuliert das Induktionsgesetz, nach dem die vom Empfangs-Magnetfeld induzierte Spannung linear mit der Frequenz zunimmt. Die Wirkung des Magnetfelds auf die Empfangsspule L1 wird durch die schwach gekoppelte Sendespule L2 simuliert. Das LTspice-Skript kann von der Projektseite [1] heruntergeladen werden.

In **Bild 5** sieht man die Abhängigkeit der Ausgangsspannung *V(out1)* von der Frequenz. Zum Vergleich ist auch die Ausgangsspannung der Originalschaltung *V(out2)* dargestellt. Man sieht, dass L5 und R2 nur ein schwaches Abfallen des Frequenzgangs für niedrige Frequenzen hervorrufen.

Rahmen oder Ferritstab?

In **Bild 6** sieht man die aufgebaute Schaltung zusammen mit der auf einen Ferritstab gewickelten Spule L1 (Ferritstab mit Durchmesser 10 cm und Länge 200 mm, darauf 20 Windungen Kupferlackdraht mit einem Durchmesser von 0,3 mm). **Bild 7** zeigt den Frequenzgang unserer Schaltung, gemessen mit einem Spektrumanalysator HP3585A. Der Trackinggenerator (Mitlaufgenerator), der ein kontinuierliches

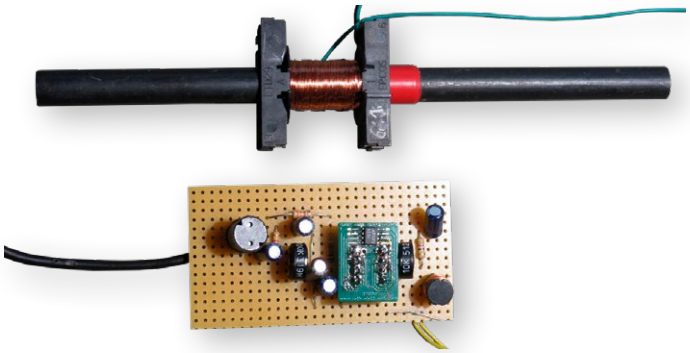


Bild 6. Ferritstab mit Wicklung und Transimpedanzverstärker als Lochrasteraufbau.

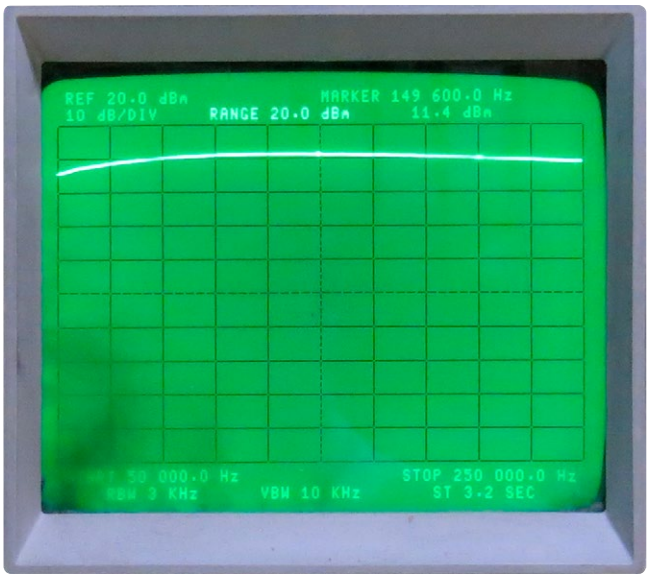


Bild 7. Frequenzgang der Breitbandantenne mit Ferritstab von 50 kHz bis 250 kHz.

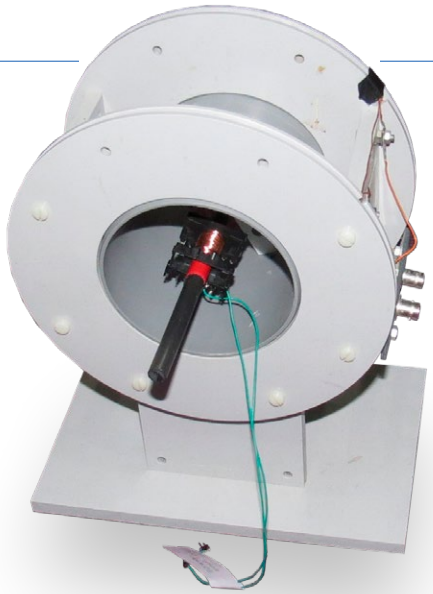


Bild 8. Messaufbau für die Frequenzgangmessung mit Helmholtzspule als Sendespule.

Sinussignal ausgibt und dessen Frequenz sich im Gleichtakt zum Empfängersweep ändert, speist dabei eine Helmholtz-Spule, in deren Mitte der Ferritstab angebracht ist, wie in **Bild 8** dargestellt. Im Langwellenbereich von 50 kHz bis 250 kHz ist der Frequenzgang relativ konstant.

Statt der Ferritantenne lässt sich natürlich auch eine Luft-Rahmen-Antenne zum Empfang verwenden. Dazu wurden 20 Windungen Kupferlackdraht mit einem Durchmesser von 0,3 mm wie in **Bild 9** auf ein Holzkreuz gewickelt, so dass sich ein Quadrat mit einer Seitenlänge von 30 cm × 30 cm ergibt.
Mit den Varianten Ferritantenne und Rahmenspule wurden Empfangsversuche im Frequenzbereich von 60 kHz bis 234 kHz unternommen. Die Ausgangsspannung für die verschiedenen empfangenen Sender ist in **Tabelle 1** dargestellt. Wie man sieht, kann man im glücklichen Aachen eine ganze Reihe europäischer Sender im Langwellenbereich einwandfrei empfangen.
Ein interessantes Feature der Breitband-Magnetantenne ist es, dass mit ihr der gleichzeitige Empfang mehrerer Sender möglich ist. Im Titelbild sieht man, wie an einer Antenne gleichzeitig die Sender DCF77 auf 77,5 kHz, DCF49 (EFR Dienst) auf 129,1 kHz sowie TDF auf 162 kHz mit Erfolg empfangen werden. Alle drei Sender senden die Zeitinformationen, die man auf den LCDs sehen kann. Es kommen dabei jeweils einfache SDR-Empfänger auf Basis eines Raspberry Pi Pico zum Einsatz [2].

Bild 9. Rahmenspule mit 30 cm × 30 cm-Wicklung mit 20 Windungen Kupferlackdraht.

Tabelle 1. Empfangsdaten diverser Zeitzeichensender.

Sender	Frequenz in kHz	Spannung für Ferritantenne in dBm	Spannung für Rahmenspule in dBm	Leistung des Senders in kW	Entfernung von Aachen in km
MSF	60	-52	-49	17	500
DCF77	77.5	-35	-25	50	220
DCF49	129,1	-35	-43	100	220
DWD	147,3	-44	-39	20	410
TDF	162	-30	-25	2.000	490
BBC	198	-35	-26	500	580
RTL	234	-30	-22	750/375	120

Damit ist die Vorstellung unserer Breitband-Magnetantenne abgeschlossen. Mit den entsprechenden Bauteilen kann man das Konzept sicher auch in den Mittel- und Kurzwellenbereich übertragen. Vielleicht erfahren Sie mehr dazu in einem zukünftigen Artikel?

RG — 220073-02

Sie haben Fragen oder Kommentare?

Wenn Sie technische Fragen oder Anmerkungen zu diesem Artikel haben, wenden Sie sich bitte an den Autor unter ossmann@fh-aachen.de oder die Elektor-Redaktion unter redaktion@elektor.de.

Passende Produkte

- **Raspberry Pi Pico RP2040 WH**
  www.elektor.de/20225
- **RTL-SDR V4 im Bundle mit Buch**
  Dogan und Ahmet Ibrahim, *Raspberry Pi 5 for Radio Amateurs* (Elektor 2020)
  www.elektor.de/20878

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.

WEBLINKS

[1] Artikel-Webseite mit LTspice-Skript: https://www.elektormagazine.de/220073-02

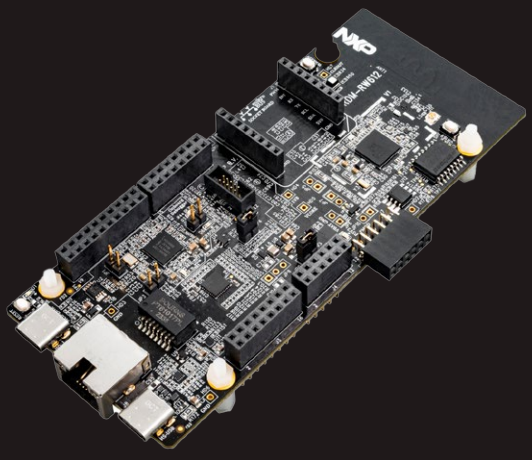
[2] Martin Ossmann, „Raspberry Pi Pico als Software Defined Radio", elektormagazine.de, April 2022: https://www.elektormagazine.de/articles/raspberry-pi-pico-makes-an-msf-sdr-de

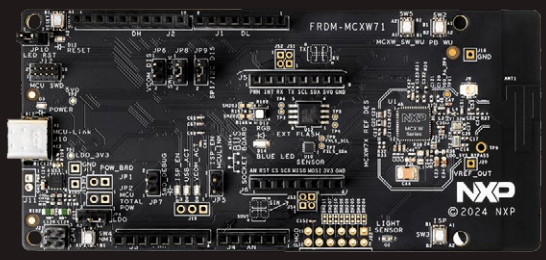
NXP Brighter Together

Drahtlose Innovation

Entdecken Sie kabellose Freiheit mit NXP's neuesten drahtlosen MCUs, die eine schnelle Entwicklung für Bluetooth, Matter, Wi-Fi, Thread und Zigbee ermöglichen – unterstützt durch FRDM-Boards und das MCUXpresso-Entwicklererlebnis

Weitere Informationen finden Sie unter **www.nxp.com/mcxw** und **www.nxp.com/rw612**





Wi Fi Bluetooth THREAD matter zigbee

TensorFlow Lite auf kleinen Mikrocontrollern

Ein (sehr) einsteigerfreundlicher Blickwinkel

Von Jean-François Simon (Elektor)

Mit TensorFlow Lite können Sie Machine-Learning-Modelle auf kleinen, ressourcenbeschränkten Mikrocontrollern ausführen. Lassen Sie uns einen genaueren Blick darauf werfen und es zusammen mit Edge Impulse für die Spracherkennung auf einem Arduino Nano 33 BLE Sense verwenden.

Künstliche Intelligenz (KI) und Maschinelles Lernen (ML) sind die neuen Schlagworte, und manchmal werden sie fälschlicherweise synonym verwendet. Facebook, Amazon, Google und viele andere setzen ML-Systeme ein, um Ihnen Inhalte zu präsentieren, die so genau wie möglich auf Ihre Vorlieben und Gewohnheiten zugeschnitten sind. ChatGPT ist ein weiteres Beispiel für einen spektakulären und beliebten Dienst, der Künstliche Intelligenz einsetzt. Diese Unternehmen haben eins gemeinsam: Sie verfügen über Server mit enormer Rechenleistung. Damit trainieren sie die Modelle und reagieren flüssig auf die Anfragen einer großen Zahl von Nutzern – und das alles durch die Verarbeitung riesiger Datenmengen.

Dies ändert sich jedoch mit dem Aufkommen von KI „on the edge“. Edge-KI bezieht sich auf den Einsatz von Algorithmen der Künstlichen Intelligenz und ihre Verarbeitung am Rande des Netzwerks. Das heißt, dass die Daten näher an der Datenquelle und weit entfernt von einem Server verarbeitet werden. Die Latenz ist geringer und die Bandbreite wird besser genutzt, so dass Daten in Echtzeit analysiert und Entscheidungen getroffen werden können. Der Begriff „Netzwerk“ wird dabei häufig verwendet, aber es funktioniert auch ganz ohne Netzwerk – zum Beispiel auf einem einfachen Mikrocontroller-Board, das gar nicht mit dem Internet verbunden ist.

TensorFlow Lite für Mikrocontroller

Vor einigen Jahren kam es zu einer interessanten Entwicklung in diesem Bereich, als TensorFlow Lite für Mikrocontroller (TFLite Micro, [1]) auf den Markt kam. Es handelt sich dabei um eine abgespeckte Version von TensorFlow, einem von Google entwickelten Open-Source-Framework für Maschinelles Lernen, das für die Ausführung von maschinellen Lernmodellen auf Mikrocontrollern entwickelt wurde und ML-Anwendungen auf kleinen, ressourcenbeschränkten Einheiten ermöglicht. Man kann TFlite Micro auf einem Arduino-Board betreiben.; nicht auf allen Arduinos, aber auf einigen. Es ist in C++ der Version 17 geschrieben und erfordert nur eine 32-Bit-Plattform sowie einige Kilobyte RAM. Es kann mit vielen ARM Cortex-M-Mikrocontrollern verwendet werden und ist auch mit ESP32 kompatibel. Die vollständige Liste der kompatiblen Plattformen finden Sie unter [1]. Während der ehrwürdige Arduino Uno der Aufgabe nicht gewachsen ist, kann der Arduino Nano 33 BLE Sense (**Bild 1**) verwendet werden. Dieses leistungsstarke Board ist perfekt für alle, die gerne mit TFlite Micro herumspielen möchten. das Board ist bereits mit Sensoren ausgestattet, darunter ein 9-achsiger Trägheitssensor, ein Feuchtigkeits-, Temperatur-, Lichtfarben- und Lichtintensitätssensor, ein Drucksensor und ein Mikrofon.

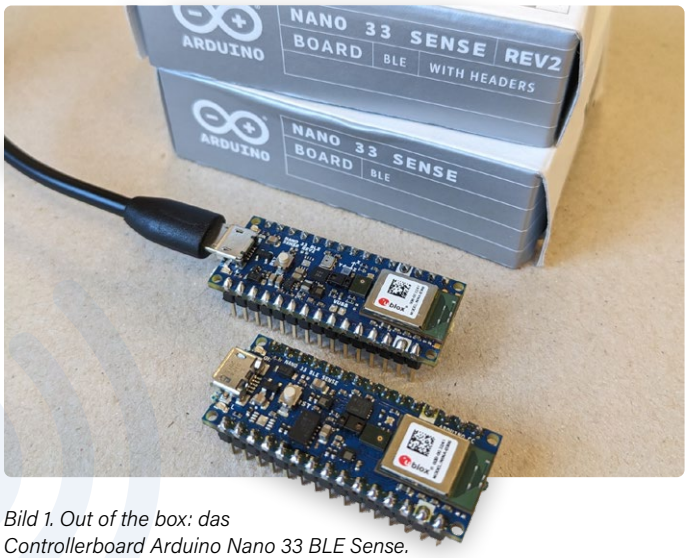


Bild 1. Out of the box: das Controllerboard Arduino Nano 33 BLE Sense.

Dieses Arduino-Board ist zwar leistungsstark, aber es reicht trotzdem nicht aus, um das Modell direkt auf dem Board zu trainieren. Die übliche Methode bei mikrocontrollerbasierten ML-Projekten ist, die Quelldaten vorzubereiten und ein Modell auf einem leistungsfähigen Rechner zu trainieren, zum Beispiel auf einem PC oder einem Remote-Server. So entsteht eine binäre Modelldatei, die anschließend in eine C-Header-Datei konvertiert werden muss. Schließlich

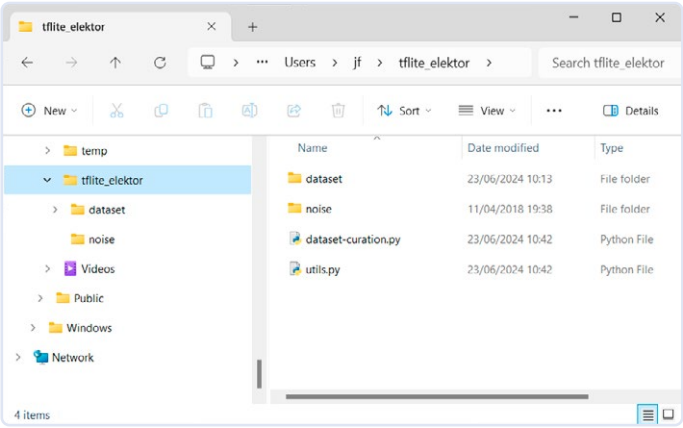


Bild 2. Die Dateien und Ordner vor Ausführung des Python-Skripts.

erstellen Sie mit den Funktionen der TFLite-Micro-Bibliothek ein Arduino-Programm und kompilieren es mit der Arduino-IDE.
Für alle, die gerne selbst anpacken und sich nicht scheuen, „die Hände schmutzig" zu machen, ist die Lektüre der offiziellen TensorFlow-Lite-Dokumentation [2] ein Muss. Ich habe auch interessante Artikel gefunden, die von DigiKey veröffentlicht wurden [3] und in denen empfohlen wird, einen Linux-PC mit Python zu verwenden und dann unter anderem TensorFlow, Keras, Anaconda, Jupyter Notebook und andere Hilfsmittel zu installieren. Die bessere Lösung ist die Ausführung von Python-Code in Google Colab [4], einer kostenlosen Cloud-basierten Plattform von Google, die es Nutzern ermöglicht, Python-Code in einer Online-Umgebung zu schreiben und auszuführen. Als Einsteiger in die Materie muss ich sagen, dass die TensorFlow-Dokumentation für mich schwer zu verstehen war. Es erfordert ein gutes Verständnis von neuronalen Netzen, um etwas funktionierendes Sinnvolles zu tun. Und wer damit noch nicht vertraut ist, wird schnell entmutigt und könnte vorschnell die Flinte ins Korn werfen.

Einfache Beispiele

Tutorials im Internet zeigen sehr oft sehr ähnliche Dinge, von denen einigen ein wenig der praktische Nutzen fehlt, um wirklich anregend zu sein. So wird zum Beispiel oft gezeigt, wie man ein Modell so trainiert, dass es einen Ausgabewert erzeugt, der einer Annäherung an den Sinus des Eingabewertes entspricht. Dabei werden natürlich die vorberechneten Werte der Sinusfunktion als Trainingsdatensatz verwendet. So kann das Modell, wenn es einmal richtig trainiert ist, bei einem Eingabewert x zwischen 0…2π und ohne Verwendung einer mathematisch implementierten Sinusfunktion eine Näherung von sin(x) als Ausgangswert liefern. Natürlich ist dies wahrscheinlich die absurdeste und unpraktischste Art, einen Sinus zu berechnen, insbesondere auf einem Mikrocontroller mit begrenzten Rechenressourcen.
Ein anderes, nützlicheres Beispiel ist die Spracherkennung. So kann der Mikrocontroller über ein Mikrofon hören, was in seiner Umgebung vor sich geht, einige Wörter erkennen (zum Beispiel *ja* und *nein* oder *Katze* und *Hund* und so weiter) und verschiedene Aktionen auslösen. Für diesen Artikel, der von einem Anfänger für Anfänger geschrieben wurde, möchte ich die Dinge einfach halten und deshalb die Verwendung der Spracherkennung auf einem Arduino Nano 33 BLE Sense demonstrieren.

Erkennung von Sprache

Hierfür werde ich das *Google Speech Command Dataset* verwenden. Er enthält 65.000 einsekündige Beispiele, wobei jeder Clip eines von 30 verschiedenen Wörtern enthält, die von Tausenden verschiedener Personen gesprochen wurden. Um das Modell zu trainieren, verwende ich Edge Impulse [5], eine Plattform, mit der Entwickler Machine-Learning-Modelle auf Edge-Geräten wie Mikrocontrollern erstellen, trainieren und einsetzen können. Es unterstützt intern TensorFlow Lite für Mikrocontroller und bietet eine einfache Möglichkeit, das Modell und die TFLite-Bibliothek auf dem Arduino-Board selbst einzusetzen, was sehr praktisch ist.
Um loszulegen, brauchen Sie einige Audiobeispiele. Erstellen Sie einen Ordner, der Ihr Arbeitsordner sein wird. Ich habe meinen *tflite_elektor* genannt. Laden Sie das *Google Speech Command Dataset* [6] herunter. Sorgen Sie für eine gute Internetverbindung; die Datei ist 2,3 GB groß. An der Extension *.tar.gz* sehen Sie, dass die Datei gleich doppelt komprimiert wurde. Verwenden Sie zum Entpacken der *.tar*-Datei 7-Zip oder eine vergleichbare Software; das in Windows eingebaute Tool ist für so große Dateien nicht zu empfehlen), und dekomprimieren Sie dann den Inhalt. Das Ergebnis ist der Ordner *speech_commands_v0.02*. Verschieben Sie diesen Ordner in Ihren Arbeitsordner. Sie können den Ordner einen einfacheren Namen geben, in meinem Fall schlicht *dataset*.

Vorbereiten der Daten

Als nächstes müssen Sie die Daten vorbereiten. Hierfür empfehle ich das ausgezeichnete Python-Skript von Shawn Hymel, das er großzügigerweise unter einer Open-Source-Lizenz zur Verfügung stellt. Laden Sie die Dateien *dataset-curation.py* und *utils.py* aus seinem GitHub-Repository [7] herunter und speichern Sie sie in Ihrem Arbeitsordner. Dieses Skript erfordert, dass der Ordner *_background_noise_* innerhalb von *dataset* von den Schlüsselwörtern getrennt wird. Ziehen Sie also diesen Ordner per Drag & Drop aus *dataset* heraus und legen Sie ihn in Ihrem Arbeitsordner ab. Sie können ihn auch beispielsweise in *noise* umbenennen. Ihr Arbeitsordner enthält jetzt die beiden Ordner *dataset* und *noise* sowie die beiden Python-Dateien (**Bild 2**).
Das Python-Skript erleichtert die Nutzung der riesigen Datenmenge, die im Google-Datensatz enthalten ist, erheblich. Außerdem ist es, wie Sie später sehen werden, sehr flexibel. Sie können es auch mit anderen Datensätzen und mit selbst aufgenommenen Audiodateien verwenden. Es wäre unpraktisch, mehrere Gigabyte an Daten auf die Server von Edge Impulse hochzuladen. Wählen Sie deshalb zunächst ein oder mehrere Schlüsselwörter als die Zielwörter aus, die der Arduino erkennen soll. Für dieses Beispiel habe ich das Wort *zero* gewählt. Das Skript erstellt für jedes Ziel-Schlüsselwort einen Ordner, in diesem Fall also einen einzigen Ordner mit dem Namen *zero*, sowie einen Ordner namens *_noise*, der zufällige Geräusche enthält, und einen Ordner namens *_unknown*, der zufällige Wörter enthält, die nicht die Ziel-Schlüsselwörter sind.
Das Skript mischt Hintergrundrauschen mit Schlüsselwort-Mustern, um die Robustheit des Modells zu erhöhen. Zunächst erstellt es die benötigten Ordner, dann extrahiert es kleinere Geräuschclips aus Hintergrundgeräuschsegmenten. Anschließend mischt es diese Geräuschclips mit Mustern von Ziel-Schlüsselwörtern und Nicht-Ziel-Schlüsselwörtern. Auf diese Weise wird die Widerstandsfähigkeit des Modells gegenüber Hintergrundgeräuschen verbessert und ein kuratierter Datensatz erstellt, der mit etwa 140 Megabyte viel kleiner ist und mit dem Edge Impulse problemlos arbeiten kann.

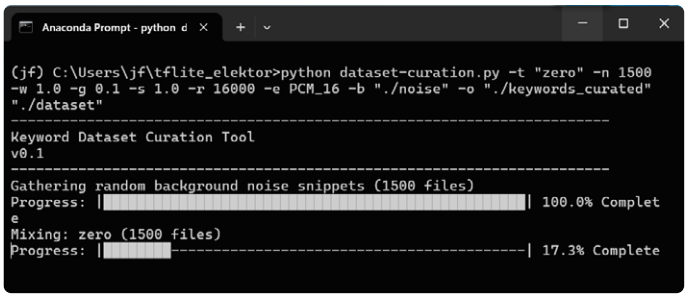


Bild 3. Das Python-Skript wird ausgeführt.

Arbeiten mit Python

Der Code wurde mit Python 3.7 getestet. Um mehrere verschiedene Python-Umgebungen mit unterschiedlichen Versionen und installierten Paketen zu verwalten, können Sie Anaconda [8] verwenden, mit dem Sie leicht eine saubere Installation der gewünschten Version erstellen. Hier erstelle ich eine neue Umgebung namens *jf*:

```
conda create -n jf python=3.7
```

Als nächstes müssen Sie die Pakete *librosa*, *numpy* und *soundfile* installieren:

```
python -m pip install librosa numpy soundfile
```

Das Paket *shutil* ist ebenfalls erforderlich, ist aber normalerweise in Python 3.7 schon vorhanden. Navigieren Sie von der Anaconda-Eingabeaufforderung oder vom Befehlszeilen-Interface Ihres Systems zu Ihrem Arbeitsverzeichnis und führen Sie das Skript mit dem Befehl

```
python dataset-curation.py -t "zero" -n 1500 -w 1.0 -g 0.1 -s 1.0 -r 16000 -e PCM_16 -b "./noise" -o "./keywords_curated" "./dataset"
```

Warten Sie ein paar Minuten, bis der Vorgang abgeschlossen ist (**Bild 3**). Nun ein kurzer Blick auf die Argumente, die das Skript benötigt:

`-t` steht für die Zielschlüsselwörter. Hier verwende ich `-t "zero"`.
`-n` ist die Anzahl der ausgegebenen Muster pro Kategorie. 1500 ist ein guter Ausgangspunkt.
`-w` und `-g` sind die Lautstärkepegel des gesprochenen Wortes beziehungsweise des Hintergrundgeräusches. `-w 1.0 -g 0.1` sind empfohlene Werte.
`-s` und `-r` sind die Längen der Samples (1 s) und die Abtastrate (16 kHz). Verwenden Sie `-s 1.0 -r 16000`.
`-e` ist die Bittiefe, hier verwenden wir 16-Bit-PCM.
`-b` ist der Ort des Hintergrundgeräusch-Ordners, `-o` ist der Ausgabe-Ordner und schließlich ist das letzte Argument ohne Label die Liste der Eingabe-Ordner. Hier ist es der Ordner *dataset*.

Wenn das Skript fertig ist, sollte es einen Ordner *keywords_curated* mit drei Ordnern erstellt haben: *_noise*, *_unknown* und *zero* (**Bild 4**).

Import in Edge Impulse

Der nächste Schritt besteht darin, diese Dateien in Edge Impulse zu importieren. Gehen Sie zur Website des Anbieters und erstellen Sie ein Konto, falls Sie noch keines haben. Nachdem Sie sich

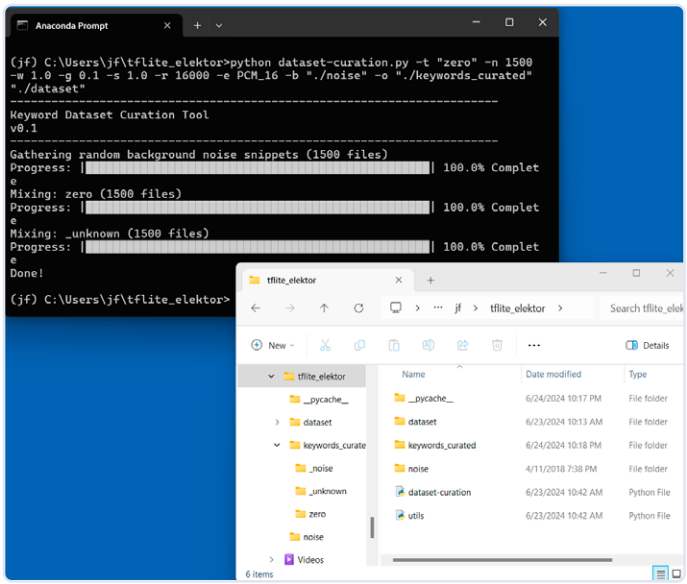


Bild 4. 100.0% Complete: Das Python-Skript hat seine Arbeit beendet.

angemeldet haben, erstellen Sie ein neues Projekt. Navigieren Sie im linken Menü zu Datenerfassung und klicken Sie dann auf *Add Data* und *Upload Data*. Klicken Sie auf *Select a folder* und wählen Sie den ersten Ordner, zum Beispiel *_noise*.
Vergewissern Sie sich, dass die Option *Automatically split between training and testing* aktiviert ist. Auf diese Weise verwendet Edge Impulse zunächst 80 % der hochgeladenen Proben zum Trainieren des Modells. Anschließend können wir die Leistung des trainierten Modells testen, indem wir es bitten, Daten zu verarbeiten, die es noch nicht gesehen hat; die restlichen 20 % sind für diesen Zweck reserviert.
Aktivieren Sie auch die Option *Label: infer from filename*, damit Edge Impulse anhand des Dateinamens erkennt, welche Proben das zu erkennende Wort oder die zu erkennenden Wörter enthalten und welche Proben Rauschen enthalten. Klicken Sie abschließend auf die Schaltfläche*Upload dat*" in der unteren rechten Ecke und warten Sie, bis die Übertragung abgeschlossen ist. Wiederholen Sie den Vorgang für die beiden verbleibenden Ordner *_unknown* und *zero*. Nach dem Upload gehen Sie zurück zu *Data Acquisition* sehen sich alle hochgeladenen Proben an. Kontrollieren Sie, ob etwa 20 % Ihrer Dateien im Testsatz und der Rest im Trainingssatz enthalten sind und auch, ob die Labels korrekt gelesen wurden (**Bild 5**).

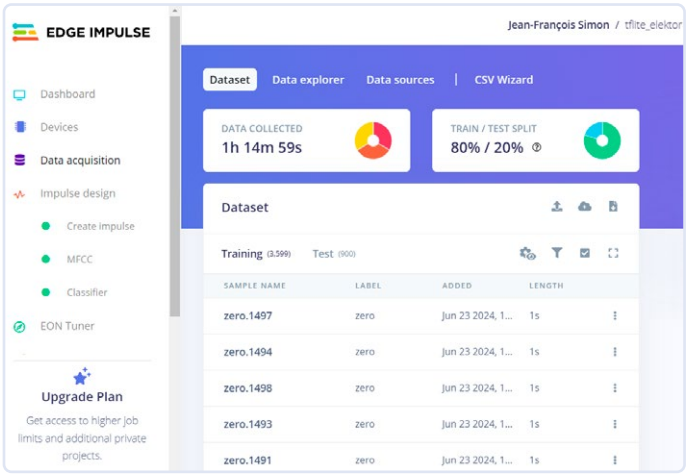


Bild 5. Die Audiosamples werden von Edge Impulse korrekt gespeichert.

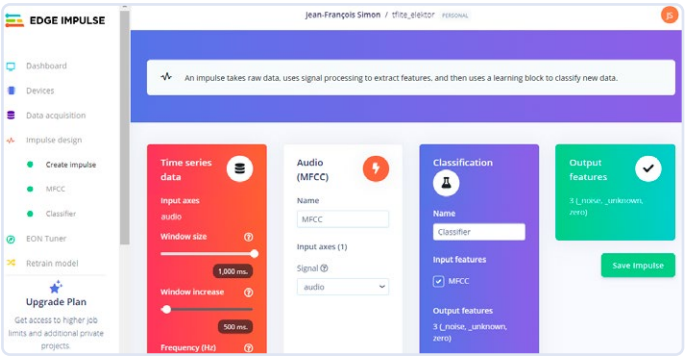


Bild 6. Einrichten eines „Impulses" oder Modells.

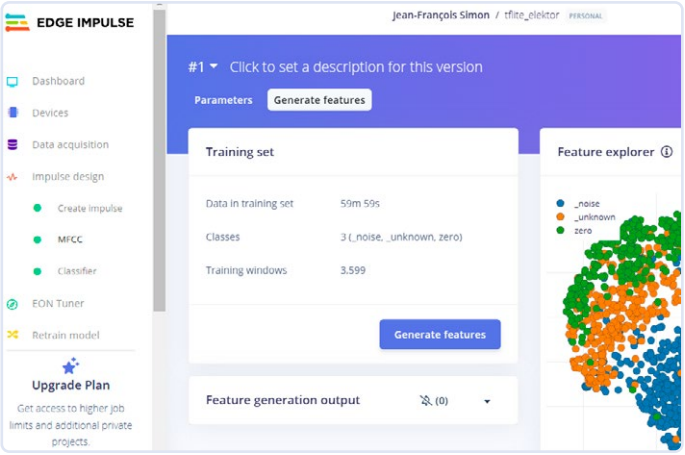


Bild 7 Der Abschnitt Generate Features*, in dem die Audiodaten verarbeitet werden.*

Als nächstes müssen Sie einen Verarbeitungsblock (Processing Block) hinzufügen. In Edge Impulse ist dies eine Komponente, die dazu dient, Sensor-Rohdaten in ein Format umzuwandeln, das für das Training und die Inferenz von Machine-Learning-Modellen geeignet ist. Er umfasst viele komplexe Dinge in einem einfachen Block, zum Beispiel die Vorverarbeitung der rohen Eingangsdaten, die Extraktion von Merkmalen (siehe unten), optionale Schritte wie Fourier-Transformationen und schließlich gibt er die Daten in einem Format aus, das mit den nächsten Schritten in der ML-Kette kompatibel ist.

Im allgemeinen Sprachgebrauch des Maschinellen Lernens sind Merkmale, Features eindeutige, quantifizierbare Attribute oder Eigenschaften der beobachteten Daten. In diesem Fall handelt es sich bei den zu extrahierenden Merkmalen um die *Mel-Frequenz-Cepstral*-Koeffizienten (MFCCs) [9], die in der Audiosignalverarbeitung und Spracherkennung häufig verwendet werden. Sie stellen das kurzfristige Leistungsspektrum eines Tonsignals auf einer nichtlinearen Mel-Skala der Frequenz dar.

Gehen Sie also zu *Impulse Design* und klicken Sie auf die Schaltfläche *Add a Processing Block*. Wählen Sie die erste Option *Audio (MFCC)*, indem Sie auf der rechten Seite auf *Add* klicken. Klicken Sie dann auf die Schaltfläche *Add a Learning Block* und wählen Sie die erste Option, die empfohlen wird, nämlich *Classification*. Klicken Sie schließlich rechts auf *Save Impulse* (**Bild 6**).

Training des Modells

Wählen Sie im linken Menü unter *Impulse Design* die Option *MFCC*. Navigieren Sie zum Tab *Generate Features* und klicken Sie auf *Generate Features* (**Bild 7**). Warten Sie, bis die Generierung der Merkmale abgeschlossen ist. Wechseln Sie anschließend zum Abschnitt *Classifier*, der sich direkt unter *MFCC* im linken Menü befindet. Klicken Sie in der oberen rechten Ecke auf *target* und wählen Sie *Arduino Nano 33 BLE Sense*. Sie können die Parameter des neuronalen Netzwerks anpassen, aber die Standardeinstellungen sind, wenig überraschend, besser als alles, was ich selbst hätte tun können. Sie können das neuronale Netzwerk mit dem grafischen Tool bearbeiten oder über das Popup-Menü in den Expertenmodus wechseln, wenn Sie mit Keras vertraut sind. In diesem Beispiel klicke ich einfach unten auf der Seite auf *Start Training*, um mit dem Training des auf den Daten beruhenden Modells zu beginnen. Wenn das Training beendet ist, können Sie die Ergebnisse im *Model*-Rahmen unten rechts überprüfen. Sie sehen dort eine allgemeine Bewertung der Genauigkeit, und 90 % gilt als recht gut; hier habe ich sogar 92,8 % erreicht (**Bild 8**).

Außerdem gibt es eine Matrix, die so genannte *confusion matrix*. Damit kann man überprüfen, wie gut das Modell funktioniert. Die Zeilen zeigen die tatsächlichen Bezeichnungen, die Spalten die vorhergesagten Bezeichnungen. Die Zahlen entlang der Diagonale, bei denen die vorhergesagte Bezeichnung mit der tatsächlichen Bezeichnung übereinstimmt, sollten viel höher sein als die anderen Werte. Hier zeigt die Diagonale 98,8 %, 87 % und 92,8 %, was gut genug sein sollte. Ein schwierigerer Test besteht darin, das Modell zu bewerten, indem man ihm Daten liefert, die es noch nicht gesehen hat. Gehen Sie dazu in den Abschnitt *Model testing* im linken Menü. Klicken Sie auf *Classify All* und lassen Sie den Test durchlaufen. Im unteren *Results*-Fenster ist die Punktzahl ein paar Prozent niedriger als zuvor, aber das war zu erwarten. Hier habe ich 90,56 % erreicht, was ein gutes Zeichen ist (**Bild 9**).

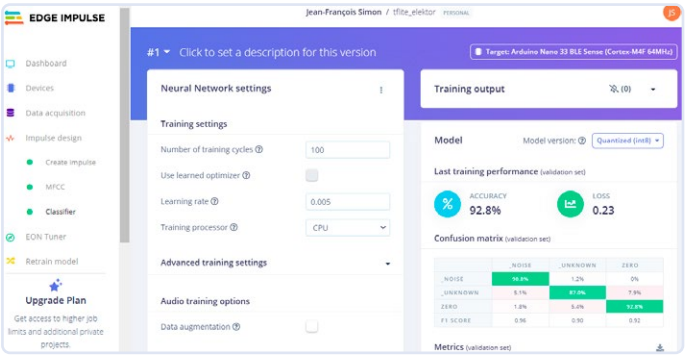


Bild 8. Das Modell hat das Training abgeschlossen.

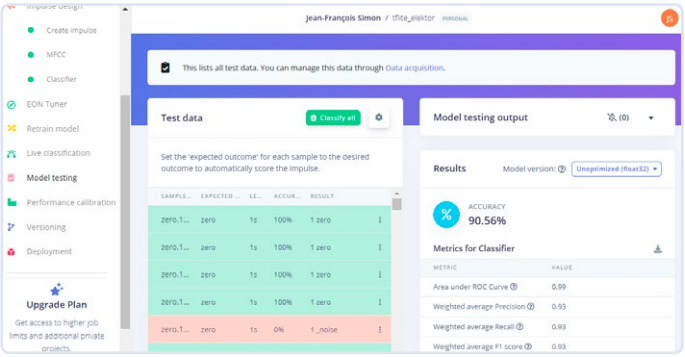


Bild 9. Testen des Modells mit unbekannten Daten.

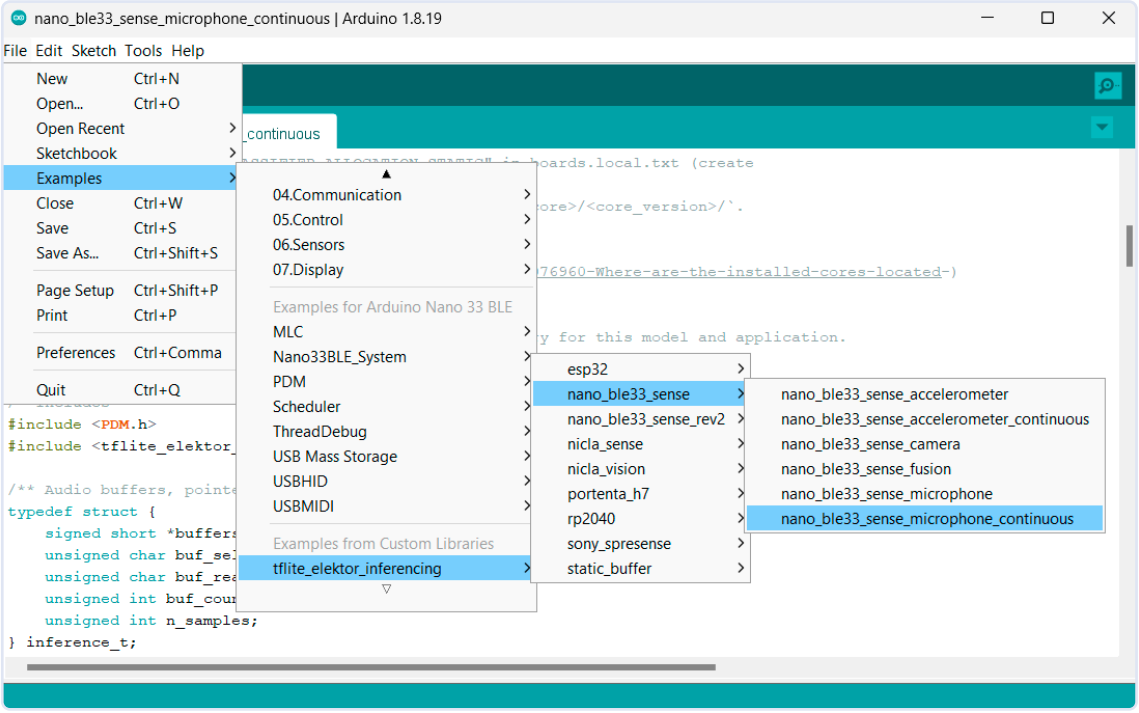


Bild 10. Verwendung des eingebauten Beispiels microphone_continuous.

Arbeit für den Arduino

Gehen wir nun auf die Seite *Deployment*. Edge Impulse bietet mehrere Optionen für die Paketierung des Modells: eine generische C++-Bibliothek für die allgemeine Verwendung bei Mikrocontrollern, Cube MX für STM32-CPUs, WebAssembly für JavaScript-Umgebungen und viele mehr. Klicken Sie auf *Search deployment options* und wählen Sie die *Arduino library* aus. Klicken Sie dann unten auf der Seite auf die Schaltfläche *Build*. Ein paar Sekunden später lädt Ihr Browser eine ZIP-Datei mit der Arduino-Bibliothek herunter.

Ich verwende die Arduino-IDE-Version 1.8.19. Öffnen Sie die Arduino-IDE und verbinden Sie Ihren *Arduino Nano 33 BLE Sense* mit Ihrem Computer. Wenn Sie dies zum ersten Mal tun, schlägt die IDE vor, das *Arduino Mbed OS Nano Boards package* herunterzuladen, was tatsächlich erforderlich ist. Sie können Sie die Bibliothek mit der üblichen Technik hinzufügen, indem Sie auf *Sketch -> Bibliothek einbinden -> .ZIP-Bibliothek hinzufügen...* klicken und die ZIP-Datei auswählen, die Sie gerade von Edge Impulse heruntergeladen haben. Gehen Sie dann zu *Datei -> Beispiele* und suchen

Ein komplexes Feld und Einfachere Optionen

Warum gibt es nicht mehr Projekte und Beispiele, die Maschinelles Lernen (ML) auf kleinen Mikrocontrollern verwenden? Meiner Meinung nach gibt es dafür mehrere Gründe.

- Um ein Modell zu trainieren, benötigt man eine große Menge vorab klassifizierter Daten. Bei der Sinusfunktion ist das einfach: Die erwarteten Ergebnisse sind im Voraus bekannt. Für englische Schlüsselwörter kann man auf den kostenlosen Google Speech Command-Datensatz zurückgreifen. Aber wenn man etwas anderes als Spracherkennung machen möchte, muss man eine große Menge an Daten sammeln, die speziell auf das Projekt zugeschnitten sind. Und das kann eine große und zeitraubende Herausforderung sein.
- Maschinelles Lernen ist ein komplexes Gebiet. Um über den Tellerrand hinauszublicken, muss man sich sehr gut mit Python und Linux auskennen und ein solides Verständnis der ML-Theorie und neuronaler Netze haben. Ich persönlich fand die Lernkurve sehr steil und habe eindeutig nicht die erforderlichen Qualifikationen oder Erfahrungen, um über dieses grundlegende Beispiel hinauszugehen.
- Die vorhandene Dokumentation für TensorFlow und TensorFlow Lite richtet sich eher an Software-Ingenieure, und es scheint, dass viele von ihnen mehr an leistungsfähigeren Plattformen wie dem Raspberry Pi 5 mit seinem externen KI-Beschleuniger oder dem Nvidias Jetson Nano interessiert sind. Trotzdem hat Pete Warden, ein wichtiger Mitwirkender bei TensorFlow und jetzt CEO von Useful Sensors, kürzlich ein großes Update für die TFLite-Micro-Portierung für den Raspberry Pi Pico veröffentlicht, die schon 2021 erstellt, aber seitdem wenig genutzt wurde. Hoffen wir, dass dies das Projekt wiederbelebt und Menschen dazu inspiriert, es zu nutzen!

Wenn Sie Ihrem Projekt nur ein paar einfache Sprachsteuerungsfunktionen hinzufügen möchten, ohne sich mit der in diesem Artikel beschriebenen Komplexität auseinanderzusetzen, sollten Sie vielleicht drei andere Optionen in Betracht ziehen: das Voice Recognition Module V3 von Elechouse, den Speech Recognizer von Grove und den Offline Language Learning Voice Recognition Sensor von DFRobot. Mit Preisen von 30 €, 20 € und 17 € sind diese Module etwas teurer, als Sie vielleicht erwarten, aber sie sind möglicherweise einfacher zu implementieren. Diese Module arbeiten nicht nach dem in diesem Artikel beschriebenen Prinzip. Man kann sie darauf trainieren, die eigene Stimme zu erkennen, indem man Befehle ein paar Mal wiederholt. Ich habe sie selbst noch nie benutzt, aber sie sehen ziemlich effektiv aus, und auf YouTube finden Sie zahlreiche Demonstrationsvideos.

Sie die soeben installierte Bibliothek. Möglicherweise müssen Sie die Arduino-IDE neu starten, damit sie erscheint. Der Name sollte mit dem Namen deines Edge-Impulse-Projekts übereinstimmen, bei mir ist es also *tflite_elektor_inferencing*.
Beachten Sie, dass es zwei separate Ordner gibt: *nano_ble33_sense* und *nano_ble33_sense_rev2* (**Bild 10**). Das hier verwendete Beispiel *microphone_continuous* erscheint nur im ersten Ordner, aber ich habe es mit beiden Versionen der Hardware mit Erfolg getestet. Andererseits müssen Sie wahrscheinlich die richtige Version auswählen, je nachdem, welches Board Sie haben, wenn Sie mit den anderen Beispielsketches spielen wollen, die den integrierten Beschleunigungssensor verwenden. Öffnen Sie das Beispiel *microphone_continuous*.
Sie können sich den Beispielsketch ansehen, um zu verstehen, wie alles aufgebaut ist. Der Mikrocontroller wartet in einer Endlosschleife, bis der Mikrofonpuffer voll ist. Dann ruft er die Funktion `run_classifier_continuous` auf, um die Inferenz mit dem neuronalen Netz auf den aufgezeichneten Audiodaten durchzuführen. Die Ergebnisse werden nur einmal pro Sekunde auf dem *Seriellen Monitor* ausgegeben. Der Code in der mitgelieferten Bibliothek ist manchmal nicht einfach zu verstehen, aber die Übung macht den Meister. Also auszuprobieren und sehen, was unter der Haube steckt!

Flashen des Boards

Stellen Sie im *Werkzeuge*-Menü der Arduino-IDE sicher, dass das richtige Board (Nano 33 BLE Sense) ausgewählt sowie der richtige COM-Port ausgewählt ist. Den zugewiesenen COM-Port finden Sie im Geräte-Manager, falls Sie Windows nutzen. Klicken Sie auf die Schaltfläche *Upload* und üben Sie sich in Geduld. Das Kompilieren des Projekts dauert eine Weile, da die Bibliothek, die die TensorFlow-Lite-Funktionen für die Inferenz sowie das auf Edge Impulse erstellte Modell im Binärformat enthält, recht umfangreich ist. Sobald der Vorgang abgeschlossen ist, werden etwa 171 kByte Flash-Speicher und ungefähr 47 kByte RAM für globale Variablen verwendet.

Es funktioniert!

Öffnen Sie nun den *Seriellen Monitor*, um die Ausgabe zu beobachten. Jede Sekunde gibt das System drei Zahlen aus, die die Wahrscheinlichkeiten angeben, dass ein bestimmtes Muster erkannt wurde, darunter zufälliges Rauschen, ein Wort, das nicht „zero" ist, und schließlich das Wort „zero". **Bild 11** zeigt ein Beispiel,

```
Predictions (DSP: 76 ms., Classification: 6 ms., Anomaly: 0 ms.):
    _noise: 0.15234
    _unknown: 0.84766
    zero: 0.00000
Predictions (DSP: 76 ms., Classification: 6 ms., Anomaly: 0 ms.):
    _noise: 0.98438
    _unknown: 0.01172
    zero: 0.00391
```

Bild 11. Der Arduino lauscht zufälligen Geräuschen im Raum.

```
Predictions (DSP: 76 ms., Classification: 6 ms., Anomaly: 0 ms.):
    _noise: 0.00000
    _unknown: 0.00391
    zero: 0.99609
```

Bild 12. 99,6%! Das Wort wurde mit ausgezeichneter Sicherheit erkannt.

in dem nichts Besonderes passiert. Wenn ich das Wort „zero" relativ nah am Arduino-Board sage, erreicht der dritte Wert einen sehr hohen Wert, fast 100 % (siehe **Bild 12**).
Nicht schlecht! Der nächste Schritt wäre, das Arduino-Board dazu zu bringen, mit diesen Informationen etwas Nützliches zu machen. Ich bin sicher, dass Sie interessante Anwendungen finden, um Sprachbefehle an Arduino-gesteuerte Geräte zu senden. Der oben beschriebene Prozess und das von Shawn Hymel entworfene Python-Skript können auch verwendet werden, um mehr als ein Wort zu erkennen. Die maximale Anzahl wird durch den Platz im Flash-Speicher des Arduino und die verfügbare Rechenleistung begrenzt. Im Code zeigt uns die Zeile `#define EI_CLASSIFIER_SLICES_PER_MODEL_WINDOW 4`, dass jedes Ein-Sekunden-Fenster in vier 250-ms-Scheiben unterteilt wird. Die Ausgabe im *Seriellen Monitor* zeigt, dass die vom Sketch verwendete Zeit 76 ms + 6 ms = 82 ms pro 250-ms-Slice beträgt, was ungefähr 33 % CPU-Auslastung entspricht. Es bleibt also noch Rechenleistung übrig, um ein eigenes Programm hinzuzufügen.

Weiterführende Schritte

Um es einfach zu halten, habe ich eines der bereits im Datensatz *Google Speech Command* verfügbaren Wörter verwendet. Wenn Sie ein Modell trainieren möchten, um ein Wort zu erkennen, das nicht Teil dieses Sets ist, muss eine große Anzahl von Audiodaten mit dem gesprochenen Wort aufgenommen werden, vorzugsweise von vielen verschiedenen Personen mit unterschiedlichen Stimmen, Altersgruppen und Intonationen gesprochen. Während der Google-Datensatz Tausende von Beispielen pro Wort enthält, könnten fünfzig bis hundert Aufnahmen eigene Wörter ein guter Anfang sein. Natürlich habe ich mit diesem einfachen Beispiel nur an der Oberfläche gekratzt. Es ist sehr zu empfehlen, tiefer zu gehen, wenn Sie daran interessiert sind! Haben Sie schon eine schöne Projektidee im Kopf, bei der Sie Maschinelles Lernen einsetzen möchten?

SE – 240357-02

WEBLINKS

[1] TensorFlow Lite for Microcontrollers: https://www.tensorflow.org/lite/microcontrollers
[2] TFLite Micro auf GitHub: https://github.com/tensorflow/tflite-micro/tree/main/tensorflow/lite/micro/examples
[3] Digikey-Tutorial, TensorFlow Lite, Teil 1: Wake Word Feature Extraction: https://tinyurl.com/bdf4dkfm
[4] Google Colab: https://colab.research.google.com/
[5] Edge Impulse: https://edgeimpulse.com/
[6] Google Sprachbefehle-Datensatz: http://download.tensorflow.org/data/speech_commands_v0.02.tar.gz
[7] Python-Skripte von Shawn Hymel: https://github.com/ShawnHymel/ei-keyword-spotting
[8] Anaconda: https://www.anaconda.com/download
[9] MFCCs in Wikipedia: https://de.wikipedia.org/wiki/Mel_Frequency_Cepstral_Coefficients

Ein **Hub** für **RS-422** und **RS-485**

Verdrahten Sie Ihre Geräte wie einen Stern

Von Sébastien Guerreiro de Brito (Frankreich)

Busse nach den Standards EIA-485 (RS-485) oder EIA-422 (RS-422) werden in der Industrie nach wie vor häufig für Zugangskontrollen, Videoüberwachung und viele andere Anwendungen eingesetzt. Der Standard erfordert jedoch eine Daisy-Chain-Verkabelung, was einschränkend sein kann. Hier erfahren Sie, wie Sie die Verdrahtung an die Sterntopologie anpassen können.

Der RS-422-Bus [1] ist in der Norm EIA-422-B-1994 als *Simplex-Bus* definiert, was bedeutet, dass es nur einen Sender gibt, der bis zu 10 Unit Loads (ULs) ansteuern kann. Die Differenz-Signale gewährleisten eine robuste Datenübertragung über große Entfernungen, auch in verrauschten Umgebungen. Die Norm spezifiziert insbesondere die elektrischen Eigenschaften der differentiellen verdrillten Zweidrahtleitung.

Da die Übertragungsleitung unidirektional ist, reicht ein einziges Adernpaar aus, wenn Sie nur entfernte Empfänger steuern wollen, ohne von ihnen eine Rückmeldung zu erhalten. Der häufigste Fall ist jedoch, dass die angeschlossenen Geräte in beide Richtungen kommunizieren, zum Beispiel wenn ein zentrales Steuergerät einen entfernten Sensor anspricht, um eine Messung anzufordern, woraufhin der Sensor seine Messung sendet. In diesem Fall sind zwei Leitungspaare erforderlich: Das erste (TX+/TX-) ist für Übertragungen vom Master-Gerät an die Slave-Geräte, das zweite (RX+/RX-) für Übertragungen von den Slave-Geräten an das Master-Gerät bestimmt. Da mehrere Slaves auf demselben Leitungspaar senden, müssen sie ihre Sender ein- und ausschalten können, damit immer nur ein Sender aktiv ist.

Baudrate vs. Reichweite

Der Sender ist in der Lage, Daten über eine Entfernung von circa 1.200 m zu übertragen, aber die Reichweite erhöht sich auf Kosten der Datenrate. Der Standard sieht Datenraten von bis zu 10 Mbit/s vor, allerdings nur über kurze Entfernungen, wie in **Bild 1** dargestellt. Aufgrund des Impedanzunterschieds zwischen dem Kabel und dem Empfänger kann es bei der Übertragung über große Entfernungen oder bei hohen Datenraten zu einer Reflexion der übertragenen Welle auf den Leitungen kommen. Zur Überwindung dieses Phänomens ist ein Abschlusswiderstand zur Anpassung der Last an den charakteristischen Widerstand des Kabels (typischerweise 120 Ω) erforderlich. Es gibt noch andere Abschlussmethoden, die jedoch weniger verbreitet sind und daher hier nicht behandelt werden.

Von RS-422 zu RS-485

Der RS-485-Bus [2] basiert auf dem RS-422-

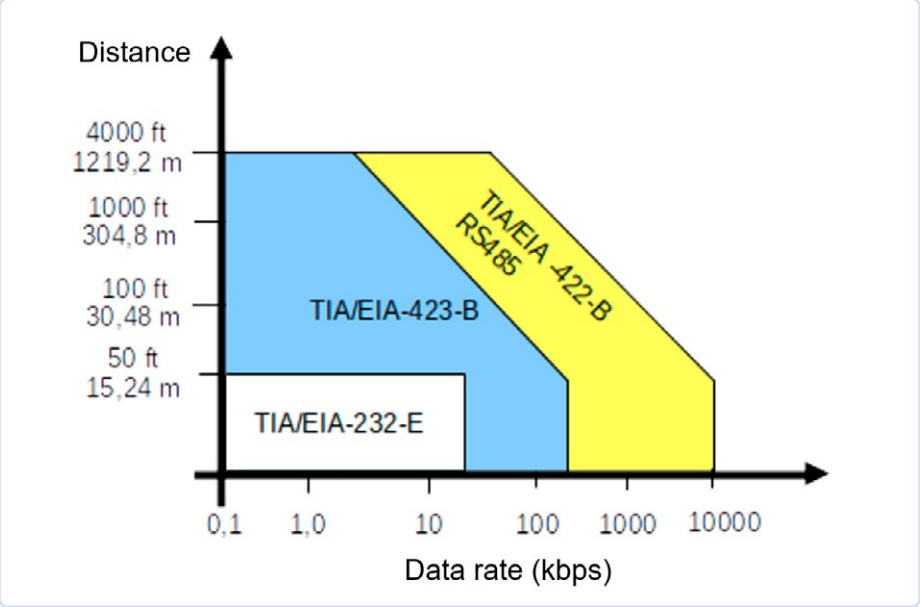


Bild 1. Datenrate vs. Leitungslänge.

Bus, weist aber einige zusätzliche Merkmalen auf, wie die Möglichkeit, die Anzahl der ULs auf maximal 32 zu erhöhen. Der RS-485-Bus kann eine Mehrpunktkonfiguration auf einem einzigen Adernpaar (oft als Data+/Data- oder A/B bezeichnet) für die Kommunikation zwischen einem Sender und mehreren Empfängern unterstützen (**Bild 2**). Diese als Halbduplex bekannte Konfiguration ist bei weitem die am häufigsten verwendete. Entgegen der landläufigen Meinung benötigt der RS-485-Bus für die Kommunikation auch eine Masseverbindung, so dass insgesamt drei Drähte vorhanden sind. Seltener kommen RS-485-Vollduplex-Verbindungen mit zwei verdrillten Adernpaaren wie bei RS-422 zum Einsatz, worauf ich hier nicht näher eingehen möchte. Für weitere Informationen sind zwei Dokumente von Texas Instruments, der *RS-485 Design Guide* [3] und die *RS-422 and RS-485 Standards Overview and System Configurations* [4] sowie der *RS-485/RS-422 Circuit Implementation Guide* von Analog Devices [5] sehr hilfreich.

Im Halbduplex-Modus sendet und empfängt jedes Gerät Daten auf demselben Adernpaar, was bedeutet, dass jedes Gerät seinen Sender nur dann aktivieren darf, wenn es senden muss. Dies bedeutet auch, dass die Empfänger in der Lage sein müssen zu erkennen, dass der übertragene Frame für sie bestimmt ist. Dazu ist ein Adressierungssystem erforderlich. *Broadcast*-Befehle können auch gesendet werden, indem ein Frame an einen virtuellen Empfänger mit einer reservierten Adresse (zum Beispiel 255) gesendet wird. Auf diese Weise wird das empfangene Telegramm von allen Empfängern berücksichtigt.

In der Praxis sendet bei dieser Art der Kommunikation der Master und die Empfänger hören zu. Nur wenn der Master eine bestimmte Information von einem der Empfänger angefordert hat, antwortet dieses Gerät. Wenn der Master *Broadcast*-Informationen an alle Empfänger gesendet hat, sollten/dürfen die Empfänger nicht antworten.

RS485- oder RS422-Treiber arbeiten alle nach einem ähnlichen Prinzip. Als Beispiel ist in **Bild 3** die Innenschaltung des ICs MAX489 von Maxim zu sehen, das in diesem Projekt verwendet wird. Das Datenblatt ist unter [6] verfügbar. Die über den Bus zu übertragenden Daten werden an *Driver Input* (DI) angeschlossen. Zur Steuerung der Kommunikationsrichtung (Empfangen oder Senden) werden

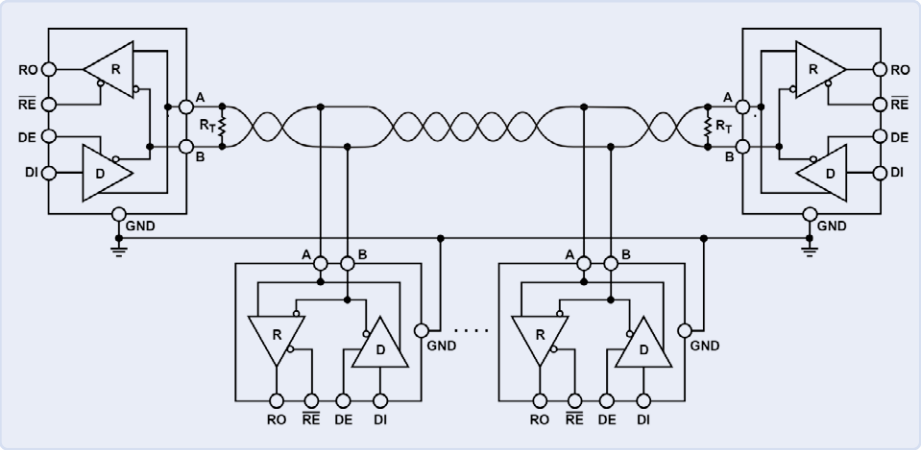


Bild 2. Typischer Halbduplex-RS-485-Aufbau. Quelle: Analog Devices

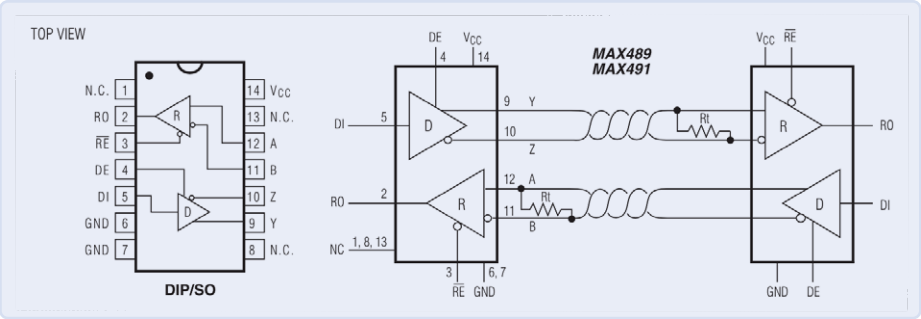


Bild 3. Pinbelegung des MAX489 und typische Betriebsschaltung. Quelle: Analog Devices [6]

Receiver Enable (/RE) und *Driver Enable* (DE) verwendet. Bei Halbduplex-Kommunikation können diese beiden Anschlüsse vom selben Signal angesteuert werden, bei Vollduplex-Kommunikation dagegen getrennt. Die Ausgangssignale Y (TX+), Z (TX-), B (RX-), A (RX+) sind mit dem Kommunikationsbus gekoppelt. Bei Halbduplex-Verbindungen, typischerweise in RS-485-Installationen, ist es üblich, Z mit B (Data-) und A mit Y (Data+) zu verbinden.

Einheitslasten

Die Norm für RS-485 legt fest, dass an einen bestimmten Sender höchstens 32 Sender/Empfänger gekoppelt werden dürfen, die jeweils eine Lasteinheit (Unit Load) darstellen. Dieser Begriff ist mit dem Eingangswiderstand der Empfänger verbunden (der in der Norm mit 12 kΩ angegeben wird). Da inzwischen einige RS-485-Geräte mit einem achtmal höheren Eingangswiderstand erhältlich sind und diese somit nur 1/8 UL darstellen, ist es theoretisch möglich, 8 × 32 = 256 Empfänger an denselben Bus anzuschließen.

Kabelverlegung

Die möglichen und verbotenen Netzwerkkonfigurationen sind in **Bild 4** dargestellt. Sternförmige Netze (a, c) sind ebenso verboten wie abgeschnittene (truncated) Verbindungen oder Ringkonfigurationen (b). Verkettete Systeme mit Stichleitungen (d) sind

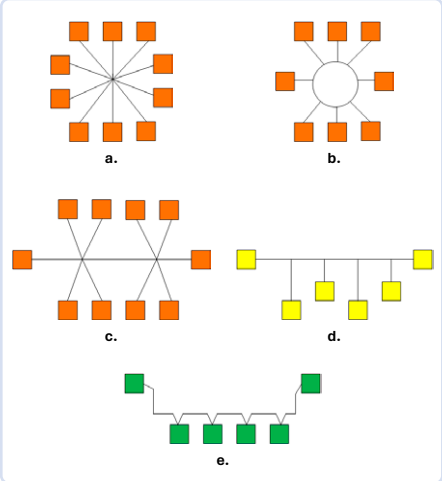


Bild 4. Viele Netzwerkkonfigurationen, von denen einige nicht zulässig sind.

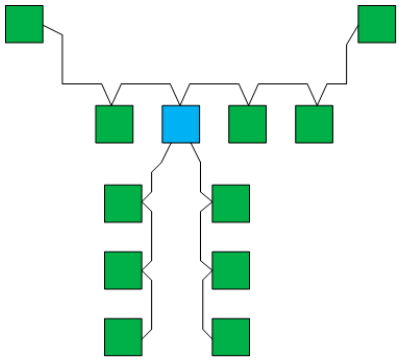


Bild 5. Beispiel für ein Netzwerk mit unserem RS-485/RS-422-Hub.

Stückliste

Widerstände:
(alle 0805, 0,1 W)
R1, R2, R7...R9, R31, R37...R40 = 4k7 , 1%
R3, R41...R45 = 1 k, 1%
R4, R10 = 22 k, 1%
R5, R6, R13, R20, R25, R30, R32 = 120 Ω, 1%
R11, R12, R14...R19, R21...R24, R26...R29, R33...R36 = 47 Ω, 1%
R46...R55 = 560 Ω, 1%

Kondensatoren:
C1 = 47 µ, 35V
C2 = 220 µ, 10V
C3...C5, C8, C12...C14, C24, C30 = 100 n, 0805
C6, C7 = 1 n, 0805
C20 = 10 n, 0805

Induktivität:
L1 = 100 µ

Halbleiter:
D1, D2, D4...D6, D28 = PML4148
D3 = 1N4004
D7 = MBRS130LT3
D8...D27 = PESD5V2S2UT
IC1, IC2 = 74HC14D
IC3, IC6...IC9 = MAX489CSD
LED1, LED3...LED8 = LED, 5 mm, grün
LED2 = LED, 5 mm, orange
U1 = LM2674-5

Außerdem:
JP1...JP5 = 2-polige Stiftleiste, Raster 2,54 mm
X1, X7 = 2-polige Leiterplatten-Grundleiste, Raster 5,08 mm, Phoenix Contact MSTBA 2,5/2-G-5,08 mit Leiterplattenstecker Phoenix Contact MSTB 2,5/2-ST-5,08
X2...X6 = 5-polige Leiterplatten-Grundleiste, Raster 3,81 mm, Phoenix Contact MC 1,5/5-G-3,81 mit Leiterplattenstecker Phoenix Contact MC 1,5/5-ST-3,81

nicht verboten, aber nicht empfehlenswert. Zusammenfassend kann gesagt werden, dass Netzwerke in einer Daisy-Chain-Konfiguration (e) zu verdrahten sind. Der Bus sollte durch jedes Gerät so nahe wie möglich an seinem Anschlusspunkt verlaufen, dann mit dem nächsten Gerät verbunden werden und so weiter bis zu einem Abschlusswiderstand.

Funktionsprinzip

Um dennoch eine Sternverkabelung zu ermöglichen, habe ich eine Schaltung entwickelt, die das Netzwerk in mehrere Busse aufteilt. Das Prinzip dieser Schaltung ist in **Bild 5** zu sehen. Das blaue Kästchen ist unsere Elektronikplatine, an die ein Daisy-Chain-Netzwerk am Eingang (oben im Bild) und mehrere Daisy-Chain-Netzwerke am Ausgang (unten) angeschlossen sind. Die Schaltung ist relativ einfach und benötigt weder einen Mikrocontroller noch ein FPGA. Ihre Funktionsweise ist in **Bild 6** dargestellt und folgt dem folgenden Prinzip.
Der Block auf der linken Seite des Bildes stellt die Eingangsseite dar, während die vier Blöcke auf der rechten Seite die Ausgänge 1...4 darstellen, jeder mit seinem eigenen RS-485-Treiber. Wie das Blockschaltbild zeigt, ist der Sender (TX, violett) auf der Eingangsseite mit allen Empfängern (RX) an den Ausgängen 1...4 verbunden, um Informationen an alle Empfänger zu übertragen, genau wie man es erwarten würde. Ebenso sind alle Senderausgänge auf den Bussen 1...4 (grün) mit dem Eingangsempfänger (RX) verbunden. Um Kollisionen zu vermeiden, werden logische Blöcke hinzugefügt. Wenn eine Aktivität auf der TX-Eingangsseite erkannt wird, werden alle TX-Ausgänge deaktiviert und umgekehrt.

Detaillierte Schaltpläne

Der detaillierte Schaltplan ist in **Bild 7** dargestellt. Wie oben angemerkt, besteht das Grundprinzip darin, das empfangene Signal zu übertragen. Alle von den Treibern eingehenden Signale (RX) werden durch die Dioden D4, D5, D6 und D28 oder-verknüpft und an den sendenden Treiber weitergeleitet. Für die ausgehenden Daten (TX) wird das Signal des Eingangstreibers durch die Schmitt-Trigger-Gatter IC1A und IC1B umgeformt und dann an die Ausgangstreiber gesendet.

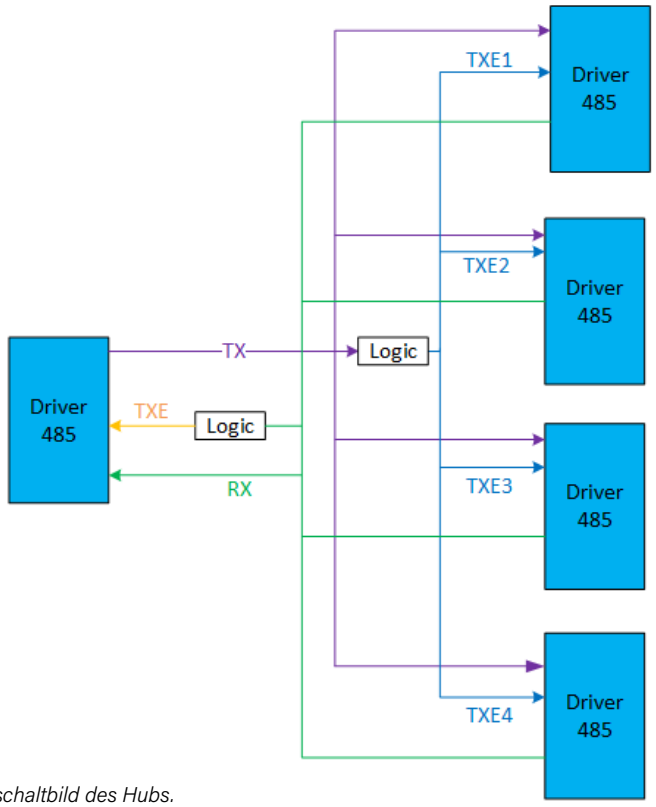


Bild 6. Blockschaltbild des Hubs.

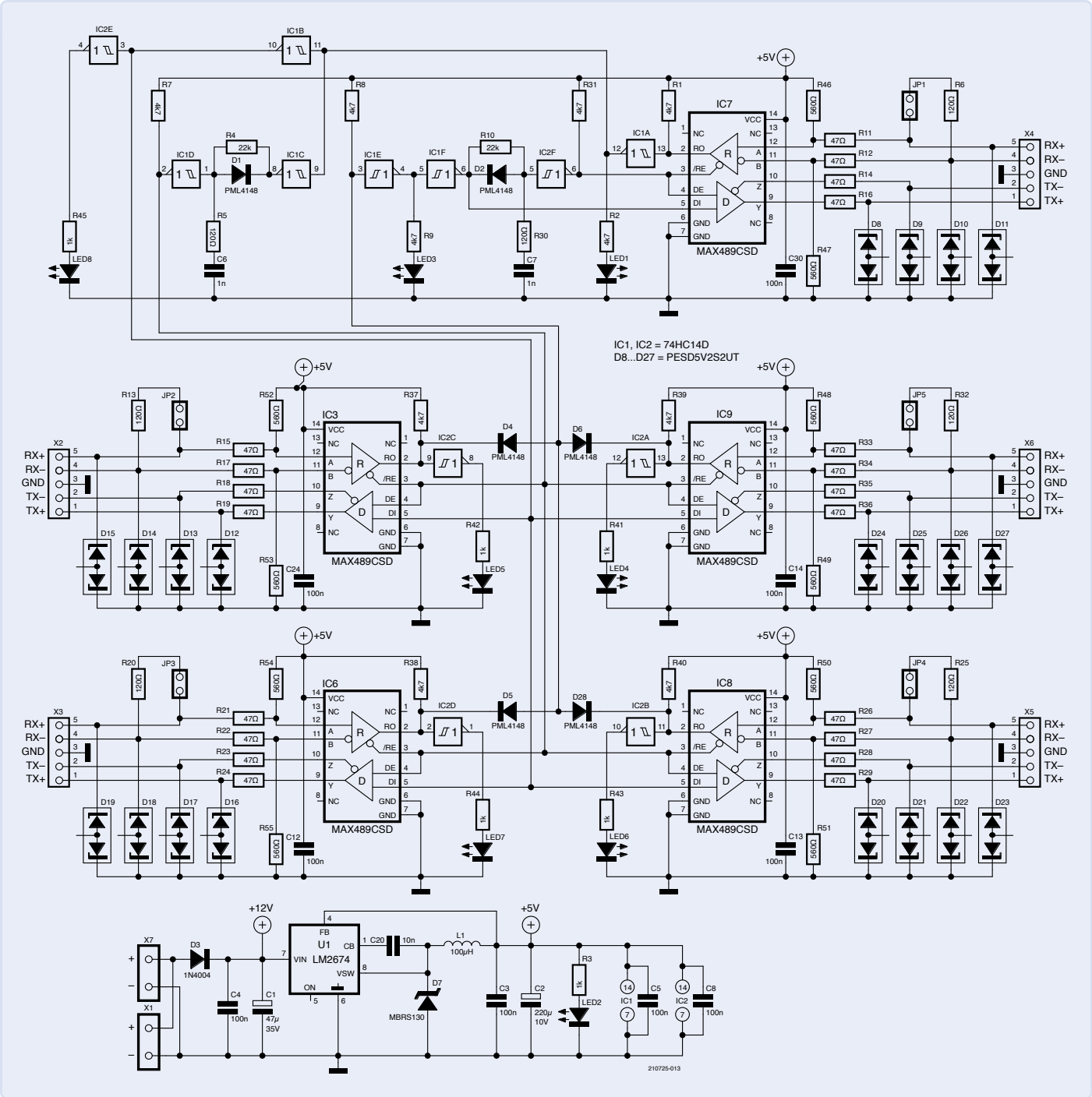


Bild 7. Der vollständige Schaltplan des Hubs.

Zum Schutz der Treiber vor elektrostatischen Entladungen (ESD) sind D8...D27 als unidirektionale Doppel-ESD-Schutzdioden ausgelegt. Obwohl die Ein- und Ausgänge des MAX489 von Haus aus robust sind, wurde jeder Leitung ein 47-Ω-Vorwiderstand hinzugefügt, um zusätzlichen Schutz vor externen Transienten zu bieten. Insgesamt gibt es zwanzig dieser Widerstände: R11, R12, R14...R19, R21...R24, R26...R29 und R33...R36. Mit den Jumpern JP2, JP3, JP4 und JP5 können bei Bedarf die Abschlusswiderstände (R6, R13, R20, R25, R32) an den Bus angeschlossen werden. Die Schaltung verfügt zudem

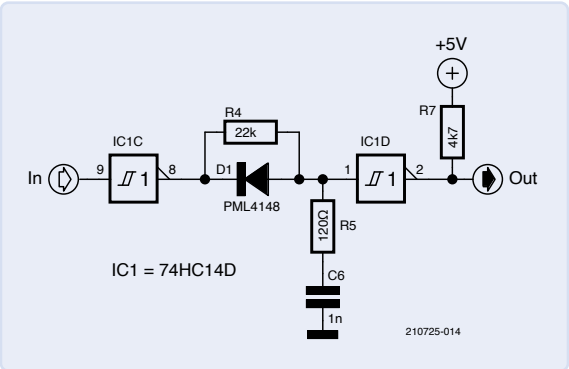


Bild 8. Schaltung zur Erkennung einer Übertragung.

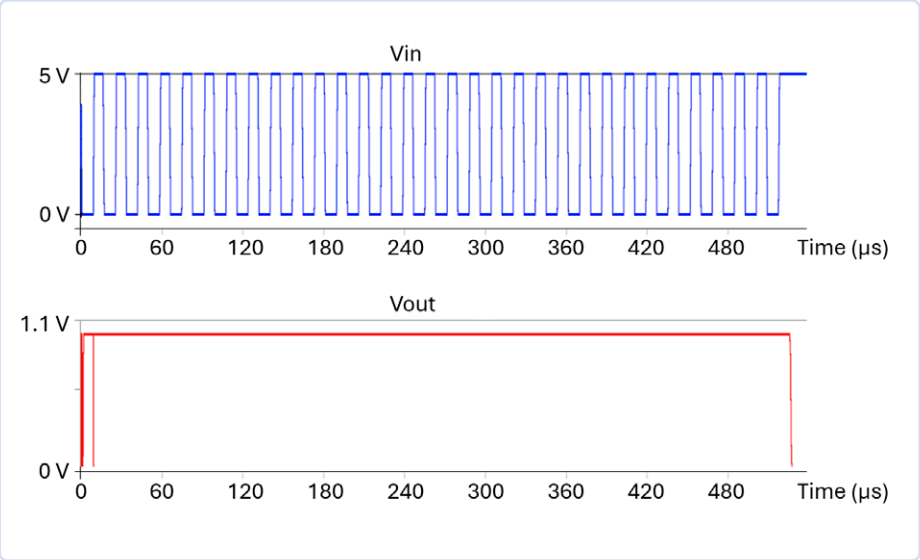


Bild 9. Simulation dieser Detektionsschaltung.

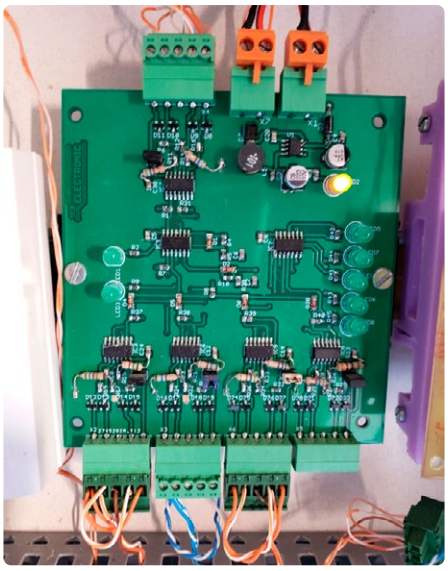


Bild 10. Schnappschuss meines Prototyps.

über mehrere Anzeige-LEDs: LED2 zeigt die erfolgreiche Stromversorgung an, während die anderen LEDs die Aktivität der Treiber beim Senden oder Empfangen von Daten signalisieren (siehe Verdrahtungsplan am Ende des Artikels).

Entscheidend für die Vermeidung von Kollisionen und zur automatischen Verwaltung der Signalrichtung (TXE) ist der Logikblock, der auf der Implementierung einer Verzögerung durch einen einfachen RC-Kreis und zwei Schmitt-Trigger-Inverter basiert. Diese Teilschaltung ist in **Bild 8** dargestellt und wird in doppelter Ausführung verwendet. Der erste Teil nutzt das Signal vom Ausgang von IC7, der zweite Teil ist mit dem Ausgang der Gruppe IC3/IC6/IC8/IC9 verbunden. Eine Simulation des Verhaltens zeigt **Bild 9**: Solange eine Übertragung aktiv ist (blau), bleibt der Ausgang high (rot).

Für ein korrektes Funktionieren des Systems müssen jedoch bestimmte Regeln beachtet werden: Erstens darf zu jedem Zeitpunkt nur ein Gerät Daten auf dem Bus senden, und zweitens muss der Datendurchsatz hoch genug sein, damit das RC-Filter seine Aufgabe erfüllen kann. Für die Spannungsversorgung (5 V) der Platine sorgt ein konventioneller DC/DC-Abwärtswandler LM2674-5 von Texas Instruments.

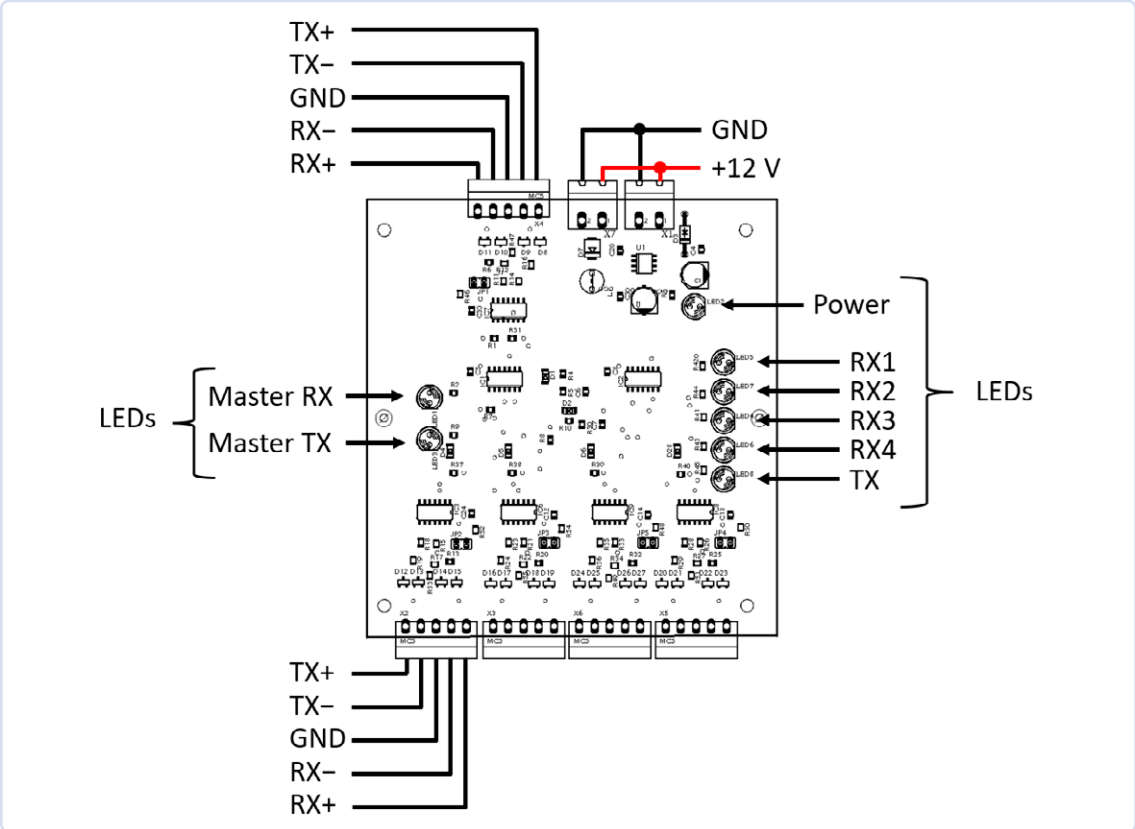


Bild 11. Verdrahtungsplan.

Praktische Umsetzung

Die hier beschriebene Schaltung wurde auf einer doppelseitigen Platine aufgebaut, wie in **Bild 10** dargestellt. Seit mittlerweile sieben Jahren läuft dieser Prototyp in meinem Haus einwandfrei und ermöglicht es meinem selbstgebauten Hausautomatisierungsserver, der mit Home Assistant kompatibel ist, mit mehreren Geräten in verschiedenen Gebäuden zu kommunizieren. Die Platine ist robust und zuverlässig und hat sich in der praktischen Anwendung bewährt. Diese Langzeitstabilität unterstreicht die Qualität und Effizienz der Konstruktion. Weitere Ansichten und Details zur Platine sind auf der Elektor-Labs-Projektseite [7] verfügbar. Der Schaltplan für die Anschlüsse auf der Platine ist in **Bild 11** dargestellt. Die Verdrahtung der drei verbleibenden Ausgänge im unteren Bereich erfolgt nach demselben Prinzip wie die des ganz linken Ausgangs, was eine konsistente und einfache Verdrahtung ermöglicht. SE – 210725-02

Über den Autor

Sébastien Guerreiro de Brito entdeckte die Elektronik bereits als Teenager, als er die Halbleiterhefte von Elektor kaufte. Nachdem er am Polytech Nantes ein Ingenieurdiplom in Elektronik und Industrieinformatik erworben hatte, entwickelte er verschiedene Schaltungen für den persönlichen Gebrauch. Um seine Leidenschaft zu teilen, veröffentlicht er regelmäßig einige dieser Schaltungen in Elektor.

Sie haben Fragen oder Anmerkungen?

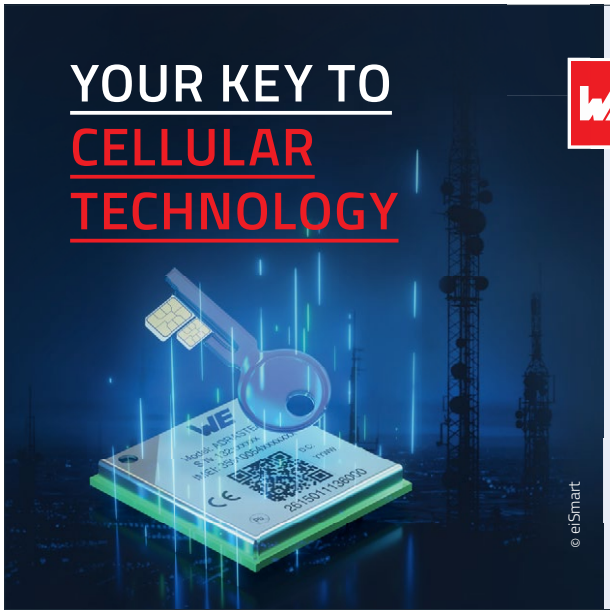
Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Dann wenden Sie sich bitte an die Elektor-Redaktion unter redaktion@elektor.de.

Passendes Produkt

Waveshare Industrial 6-Kanal-Relaismodul für Raspberry Pi Zero
www.elektor.de/20085

WEBLINKS

[1] RS-422 auf Wikipedia: https://de.wikipedia.org/wiki/RS-422
[2] RS-485 auf Wikipedia: https://de.wikipedia.org/wiki/RS-485
[3] Texas Instruments, RS-485 Design Guide: https://ti.com/lit/an/slla272d/slla272d.pdf
[4] Texas Instruments, RS-422 und RS-485 RS-422 and RS-485 Standards Overview and System Configurations: https://ti.com/lit/an/slla070d/slla070d.pdf
[5] Analog Devices, RS-485/RS-422 Circuit Implementation Guide: https://analog.com/en/resources/app-notes/an-960.html
[6] Datenblatt MAX489: https://analog.com/media/en/technical-documentation/data-sheets/MAX1487-MAX491.pdf
[7] Projektseite auf Elektor Labs: https://elektormagazine.de/labs/rs422485-hub



WÜRTH ELEKTRONIK
MORE THAN YOU EXPECT

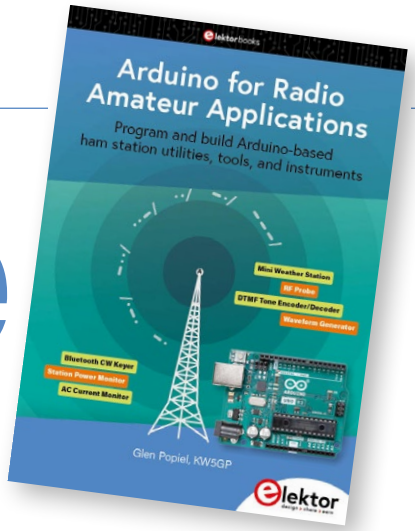
WE meet @ electronica
Halle A6-502

Adrastea-I ist ein Cellular-Modul mit hoher Leistung, extrem niedrigem Stromverbrauch, Multi-Band LTE-M und NB-IoT-Modul.

Trotz seiner kompakten Größe verfügt das Modul über integriertes GNSS, integrierten ARM Cortex M4 und 1 MB Flash-Speicher für die Entwicklung von Benutzeranwendungen. Das Modul basiert auf dem leistungsstarken Sony Altair ALT1250 Chipsatz. Das von Deutsche Telekom zertifizierte Adrastea-I-Modul ermöglicht eine schnelle Integration in Endprodukte ohne zusätzliche branchenspezifische Zertifizierung (GCF oder Betreiberzulassung, sofern eine Deutsche Telekom IoT-Konnektivität (SIM-Karte) verwendet wird. Für alle anderen Betreiber bietet das Modul bereits die branchenspezifische Zertifizierung (GCF) an.

www.we-online.com/gocellular

- Kompakte Größe
- Sicherheit und Verschlüsselung
- Lange Reichweite/weltweite Abdeckung
- Multiband Unterstützung



HF-Sonde

Mit LED-Balkenanzeige

Von Glen Popiel, KW5GP (USA)

In diesem Artikel können Funkamateure und Arduino-Enthusiasten aller Erfahrungsstufen einen Breitband-HF-Signalschnüffler bauen, der auf dem logarithmischen Verstärker AD8307 von Analog Devices, einem WS2812-RGB-LED-Streifen und dem Arduino Nano mit einem entsprechenden Sketch basiert. Glen, KW5GP, Sie sind dran!

Anmerkung der Redaktion: *Dieser Artikel ist ein Auszug aus dem Elektor-Buch* Arduino for Radio Amateur Application. *Er wurde formatiert und leicht bearbeitet, um den Konventionen und dem Seitenlayout von Elektor zu entsprechen. Der Autor und die Redaktion helfen Ihnen gerne bei Fragen weiter. Kontaktinformationen finden Sie unter* Fragen oder Kommentare.

Eines der coolen Dinge am Arduino ist, dass er als Werkzeug verwendet werden kann, um andere Werkzeuge zu bauen. Wenn Sie als Funkamateur wie die meisten sind, brauchen Sie nicht viel mehr als ein Multimeter und ein Leistungs-/Stehwellen-Messgerät, und wenn Sie etwas mit einem Gerät testen müssen, das Sie nicht haben, können Sie es sich normalerweise von einem anderen Funkamateur leihen. Der Hauptgrund, warum man keine eigene Testausrüstung besitzt, ist, dass viele Messgeräte ziemlich teuer sein können und/oder sich die Investition für etwas, das man nur ein- oder zweimal im Leben benutzt, einfach nicht rentiert.

In meinem Fall benötige ich nicht oft eine HF-Sonde (**Bild 1**), aber sie kann sehr nützlich sein, wenn man wissen will, ob der Sender tatsächlich sendet, und man kein Wattmeter oder SWR-Meter zur Hand hat, oder wenn man vielleicht eine HF-Störung im Shack aufspüren muss. Bei der Recherche für ein anderes Projekt stieß ich auf den Artikel *RF Driven On-Air Indicator* von Keith Austermiller, KB9STR, in der August-Ausgabe 2004 des *QST Magazine*, der seinerseits vom *No Fibbin' RF Field Strength Meter* von John D. Noakes, VE7NI in der August-Ausgabe 2002 des *QST* abgeleitet ist. Mit einigen geringfügigen Änderungen könnten diese Projekte in eine HF-Sonde umgewandelt werden, die es einem Arduino ermöglichen würde, einen adressierbaren LED-Stick vom Typ WS2812 anstelle eines Messgeräts anzusteuern, um die relative Stärke des HF-Signals anzuzeigen.

Bild 1. Die fertige HF-Sonde.

Designphilosophie und Schlüsselelemente

Auch wenn es sich um ein relativ einfaches und unkompliziertes Projekt zu handeln scheint, möchte ich dennoch die vollständige Dokumentation ausarbeiten. Diese Dokumentation kann sehr hilfreich sein, wenn ich das Projekt zu einem späteren Zeitpunkt wieder aufgreifen möchte, um es neu zu gestalten, Funktionen hinzuzufügen und so weiter.

Bild 2 zeigt das Blockdiagramm für die HF-Sonde. Zusätzlich zur HF-Eingangsschaltung wird der Arduino verwendet, um die Signalstärke auf einem 8-LED-RGB-LED-Stick vom Typ WS2812 anzuzeigen. Letzteres ist es, was dieses Projekt wirklich sehr vereinfacht. Anstatt diskrete RGB-LEDs anzusteuern, was jeweils drei digitale I/O-Pins und insgesamt 24 I/O-Pins für eine 8-LED-Anzeige erfordert, benötigen die adressierbaren RGB-LED-Sticks und -Streifen des Typs WS2812 nur einen einzigen digitalen I/O-Pin. Alternativ könnte man auch einen Bargraph-LED-Treiberchip wie den MAX7219 von Analog Devices verwenden, der ebenfalls nur einen digitalen I/O-Pin benötigt, aber warum sollte man die einen 24-Pin-Chip hinzufügen, wenn es nicht notwendig ist?

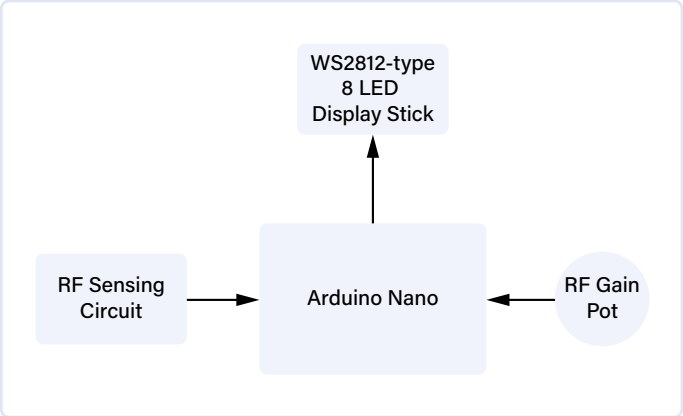


Bild 2. Das Blockdiagramm der HF-Sonde.

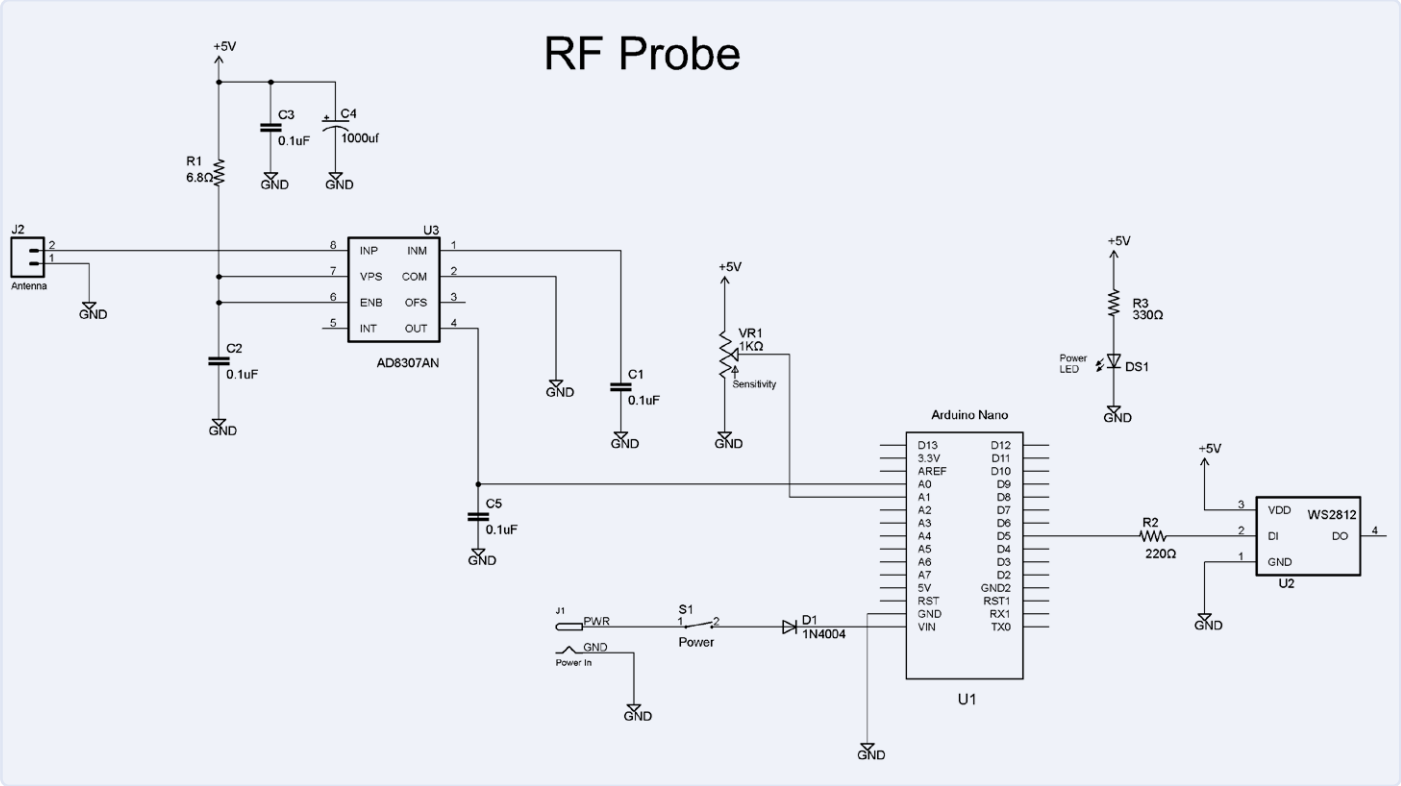


Bild 3. Schaltplan der HF-Sonde.

Schaltplan und Aufbau

Der Aufbau der HF-Sonde unterscheidet sich ein wenig von der üblichen Vorgehensweise in diesem Buch. Anstatt die HF-Sensoreinheit als Prototyp auf dem Breadboard aufzubauen und sich über die Auswirkungen von HF auf die Breadboard-Verdrahtung Gedanken zu machen, wird die HF-Sensorschaltung auf demselben Prototyping-Board wie der Arduino Nano aufgebaut. Anstelle des Operationsverstärkers aus dem Artikel *RF Driven On-Air Indicator* habe ich die Schaltung ein wenig modernisiert und stattdessen einen Logarithmischen Verstärker AD8307 von Analog Devices verwendet, wie im Schaltplan der HF-Sonde in **Bild 3** gezeigt. Der AD8307 kann HF bis zu 500 MHz erfassen, so dass das Projekt für alle Ihre Anforderungen an die HF-Detektion mit Ausnahme von Handy- und WLAN-Frequenzen verwendet werden kann. Da am Ausgang des AD8307 eine analoge Spannung liegt, die die Signalstärke darstellt, können wir den Ausgang des AD8307 direkt an den analogen Eingang A0 des Arduino Nano anschließen.

Der Antennenanschluss befindet sich an einer Seitenwand eines Solarbotics-Gehäuses und wird mit der Platine der HF-Sonde über gewöhnliche Header (2,54 mm) und DuPont-Buchsen (passende Buchsen in Steckergehäusen) verbunden, damit die Platine leicht entfernt werden kann, um eventuelle Verdrahtungsfehler zu korrigieren. Für den Logarithmischen Verstärker AD8307 wird eine Fassung verwendet, damit man den Chip ersetzen kann, wenn man ihn versehentlich mit 100 W füttert und in Rauch aufgehen lässt. Die HF-Sonde ist auf einer 60 mm × 80 mm großen Stück Prototyping-Platine aufgebaut (**Bild 4**) und in einem durchsichtigen Solarbotics-Gehäuse Mega SAFE montiert, einem Klarsicht-Gehäuse, das für den Einbau eines Arduino-MEGA vorbereitet ist. Eine SO-239-Koax-Einbaubuchse wurde an dem Gehäuse angebracht, so dass Sie verschiedene Drähte als HF-Antenne verwenden können. Ein kurzes Stück Massivdraht (25 cm, #14 AWG) wurde als Antenne an den Mittelleiter eines PL-259-Koaxialsteckers gelötet.

Eingang, Ausgang, Steuerelemente

In diesem Projekt werden wir den im Arduino Nano eingebauten 10-Bit-Analog-Digital-Wandler (ADC) nutzen, um die analoge Spannung des AD8307 in einen numerischen Wert umzuwandeln. Dieser Wert wird dann in die Anzahl der zu leuchtenden LEDs umgewandelt, die die Signalstärke darstellen. Anschließend werden entsprechend viele LEDs auf einem 8-LED-RGB-LED-Streifen vom Typ WS2812 zum Leuchten gebracht. Für die HF-Sonde benötigen Sie nicht unbedingt die Leistungsfähigkeit und die Grafik eines TFT-Displays, da Sie eigentlich nur an der Anzeige einer relativen HF-Feldstärke interessiert sind, wofür sich ein WS2812-LED-Streifen perfekt eignet. Ein 1-kΩ-Potentiometer wird verwendet, um die HF-Empfindlichkeit der Sonde einzustellen.

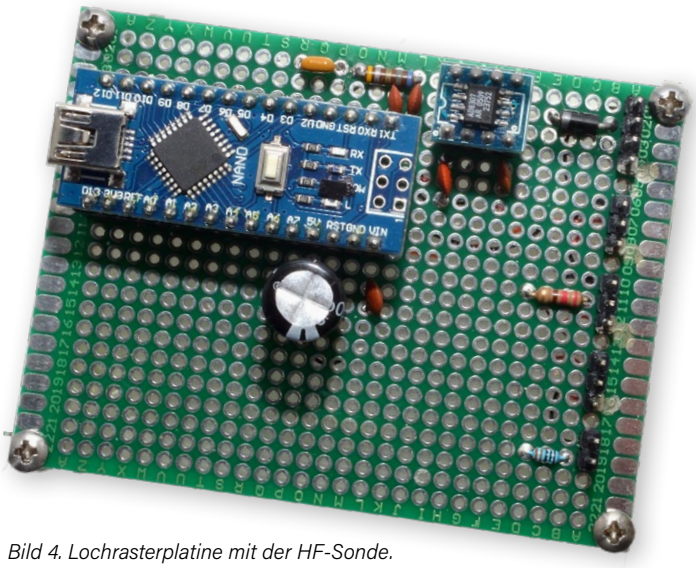


Bild 4. Lochrasterplatine mit der HF-Sonde.

Der Arduino-Sketch

Der Sketch für die HF-Sonde ist eigentlich recht einfach. Anhand des in **Bild 5** gezeigten Projektablaufplans ist der Sketch selbst recht klein, was zeigt, wie gut der Arduino für die einfachen I/O-Aufgaben dieses Projekts geeignet ist. Der Sketch muss lediglich die analoge Spannung vom AD8307 lesen und eine Balkendarstellung dieser Eingangsspannung auf dem LED-Stick ausgeben.
Der vollständige Sketch und die Bibliotheken für dieses Projekt sind online verfügbar [1]. Der Sketch verwendet vier grüne, zwei gelbe und zwei rote LEDs, um die Stärke des erkannten Signals anzuzeigen, wie in **Bild 6** dargestellt. Sie werden feststellen, dass ich ein kleines Stück Seidenpapier als Diffusor auf die Innenseite des durchsichtigen Gehäusedeckels geklebt habe; Sie können aber auch ein der halbtransparentes Gehäuse verwenden.
Zurück zur Software. Wir beginnen den Sketch für die HF-Sonde, indem wir die Adafruit_NeoPixel-Bibliothek einbeziehen, um den adressierbaren RGB-LED-Stick zu steuern. Es gibt mehrere Bibliotheken für die adressierbaren RGB-LEDs des Typs WS2812, aber ich bevorzuge alle Funktionen und Flexibilität in der Adafruit-Bibliothek.

```
#include <Adafruit_NeoPixel.h>
```

Als Nächstes definieren wir die Anzahl der LEDs auf dem adressierbaren LED-Stick und den I/O-Pin, an den wir sie anschließen werden. Das Tolle am WS2812 ist, dass er das PWM-Signal, das zur Helligkeitssteuerung der einzelnen LEDs des Sticks verwendet wird, intern erzeugt, so dass Sie die Datenleitung des LED-Sticks an jeden beliebigen digitalen I/O-Pin des Arduino anschließen können. Adressierbare RGB-LED-Sticks und -Streifen des Typs WS2812 benötigen nur einen digitalen I/O-Pin zum Betrieb. Wir werden `#define`-Direktiven verwenden, um die Einstellung der LED-Farben zu vereinfachen.

```
#define NUM_LEDS 8
#define LED_PIN 5
#define RF_input_pin A0
#define gain_pin A1
#define led_Red 255, 0, 0
#define led_Green 0, 255, 0
#define led_Yellow 255, 255, 0
#define led_brightness 50
#define update_delay 200
```

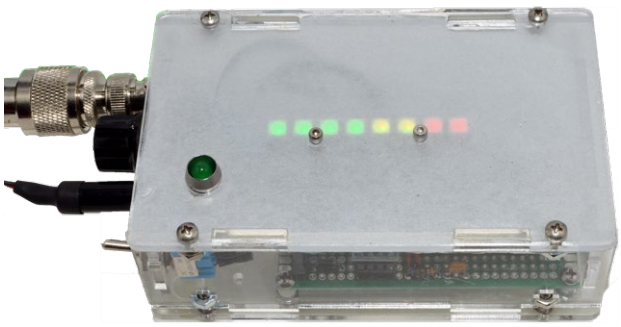


Bild 6. Die HF-Sonde im Gehäuse.

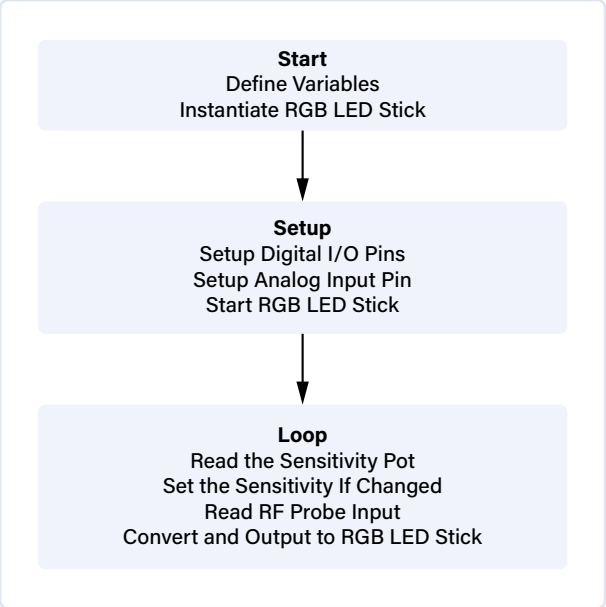


Bild 5. Das Flussdiagramm der HF-Sonde.

Nachdem wir die wenigen Variablen für diesen Sketch deklariert haben, werden wir den LED-Stick instanziieren.

```
int signal_strength = 0;
int effective_strength = 0;
int max_strength = 1000;
int LEDsToLight = 0;
int rf_gain;

Adafruit_NeoPixel pixels
  (NUM_LEDS, LED_PIN, NEO_GRB + NEO_KHZ800);
```

Die `setup()`-Schleife für die HF-Sonde ist so kurz und so einfach wie nur möglich. Alles, was zu tun ist, ist den adressierbaren RGB-LED-Stick zu starten, alle LEDs auszuschalten und dann die `show()`-Methode der NeoPixel-Klasse aufzurufen, um die LED-Clear-Funktion auszuführen.

```
pixels.begin(); // INITIALIZE NeoPixel strip object

pixels.clear(); // Set all pixel colors to 'off'
pixels.show();
```

In der Hauptschleife `loop()` wird die analoge Spannung vom Schleifer des HF-Potis gelesen, der an den analogen Eingang Pin 1 des Arduino angeschlossen ist. Als Nächstes lesen wir die HF-Signalstärke am analogen Eingang Pin 0 und verwenden dann den Wert der Verstärkungseinstellung, um die Arduino-`map()`-Funktion anzupassen, um eine Art Empfindlichkeitseinstellung für die HF-Sonden-LEDs zu ermöglichen.
Jetzt wäre ein guter Zeitpunkt, um die Arduino-`map()`-Funktion vorzustellen, mit der sich die Skalierung Ihrer Daten hervorragend ändern lässt. In diesem Fall wird der 0...5-V-Analogeingang des AD8307, den wir zur Erkennung des HF-Signals verwenden, von einem Bereich möglicher digitaler Werte von 0...1.023 auf einen Wert von 0...8 abgebildet, was der Anzahl der LEDs auf dem LED-Stick entspricht. Der resultierende Wert gibt an, wie viele LEDs auf dem LED-Stick aufleuchten müssen, um die HF-Feldstärke anzuzeigen.

```
LEDsToLight = map(effective_strength, 0, max_strength,
 0, NUM_LEDS);

 if (LEDsToLight < 0)
 {
  LEDsToLight = 0;
 }

 if (LEDsToLight > NUM_LEDS)
 {
  LEDsToLight = NUM_LEDS;
 }
```

Da wir nun wissen, wie viele LEDs leuchten sollen, können wir bestimmen, welche Farbe die leuchtenden LEDs haben sollen. Wir haben ja festgelegt, dass die ersten vier LEDs immer grün, die nächsten zwei immer gelb und die letzten zwei immer rot leuchten sollen. Wir beginnen also damit, dass wir zunächst die vorherige LED-Einstellung löschen, um die LEDs für den aktuellen Wert der HF-Signalstärke einschalten zu können.

```
pixels.clear();
```

Als nächstes werden die Farben und die gewünschte Anzahl der zu leuchtenden LEDs festgelegt.

```
pixels.setBrightness(led_brightness);

for (int led = 0; led < LEDsToLight; led = led + 1)
{
 if (led <= 3)
 {
  pixels.setPixelColor(led, pixels.Color(led_Green));
 }
 if (led > 3 && led <= 6)
 {
  pixels.setPixelColor(led, pixels.Color(led_Yellow));
 }
 if (led > 5)
 {
  pixels.setPixelColor(led, pixels.Color(led_Red));
 }
}
```

Schließlich wird die Funktion `pixels.show()` verwendet, um die LED-Farben tatsächlich anzuzeigen, und die gewünschte Anzahl von LEDs leuchtet dann auf.

```
pixels.show();
```

WEBLINK

[1] Sketch und Bibliotheken (Download): https://tinyurl.com/kw5gpelektor

Stückliste

Widerstände:
R1 = 6Ω8, 1/8 W
R2 = 220 Ω, 1/8 W
R3 = 330 Ω, 1/8 W
VR1 = Poti 1 k

Kondensatoren:
C1, C2, C3, C5 = 0µ1, 25 V
C4 = 1000 µ, 25 V

Halbleiter:
D1 = 1N4004
DS1 = 5-mm-LED
U2 = WS2812 (8-Neopixel-Stick)
U3 = AD8307 (Logarithmischer Verstärker)

Außerdem:
J1 = Niederspannungs-Einbaubuchse
S1 = Schalter, 1×an (SPST)
U1 = Arduino Nano
Gehäuse: Solarbotics Mega S.A.F.E.

Der Wert `update_delay` wird zur Verzögerung zwischen den Durchläufen des Sketches verwendet. Ich habe ihn auf 200 ms festgelegt, was fünf Aktualisierungen pro Sekunde bedeutet. Sie können diesen Wert nach eigenem Gusto anpassen. Bei einem Wert von 0 werden die LEDs auf dem Stick so schnell aktualisiert, wie es dem Arduino möglich ist, und die Signalstärke wird nahezu in Echtzeit angezeigt. Diese Technik könnte auch verwendet werden, um eine dynamische Anzeige eines Audiopegels zu erstellen.

```
delay(update_delay);
```

Ideen für Erweiterungen

Dies wäre ein hervorragendes Projekt für eine Bonbon-Blechdose und einen Arduino Nano. Wenn man einen SMA-Stecker anstelle des SO-239 verwenden würde, könnte man alles in die Dose einbauen, mit einem kleinen Ausschnitt für den adressierbaren LED-Stick. Sie könnten den LED-Stick auch durch eine kleine OLED-Anzeige ersetzen und die HF-Sonde eine digitale Darstellung der relativen Feldstärke liefern lassen.

SE – 240326-02

Sie haben Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Schreiben Sie dem Autor eine E-Mail an glenpopiel@gmail.com oder Elektor an redaktion@elektor.de.

Passendes Produk

- **Glen Popiel, *Arduino for Radio Amateur Applications* (Elektor 2024)**
  Taschenbuch, englisch: www.elektor.de/20814
  E-Book, PDF, englisch: www.elektor.de/20815

Aller Anfang... muss nicht schwer sein

mehr Verstärkerschaltungen

Von Eric Bogers (Elektor)

Der Instrumentenverstärker war die Schaltung, die wir beim letzten Mal untersucht haben. Damit haben wir vorerst genug Theorie behandelt. Als Nächstes sind einige praktische Schaltungen dran.

Zunächst aber eine kleine Ergänzung zum letzten Teil. Unter der Überschrift „Symmetrische Verbindungen“ hatten wir das Folgende geschrieben:
„Für den Anschluss von Lautsprechern wird häufig ein (ungeschirmtes) zweiadriges Kabel mit verdrillter Ader verwendet, wobei eine Ader die Phase und die andere die Masse ist. Im Prinzip spielt es keine Rolle, welche Ader dafür verwendet wird; der Anschluss des Kabels an den Stecker ist genormt und eine der Adern ist gekennzeichnet.“
Wie mein Kollege Brian Tristam Williams richtig bemerkt hat, ist das ein wenig zu einfach gedacht. Es stimmt zwar, dass es elektrisch keinen Unterschied macht, wie die Lautsprecher mit den jeweiligen Ausgängen eines Verstärkers verbunden sind, aber für Ihre Ohren kann es (im Falle eines Stereoverstärkers) einen großen Unterschied machen! Für ein optimales Hörerlebnis sollten beide Lautsprecher (oder Lautsprecherboxen) gleichphasig angeschlossen werden, um akustische Kurzschlüsse zu vermeiden, die sich vor allem auf die unteren Frequenzen negativ auswirken. Bei den meisten Verstärkern und Lautsprecherboxen sind die Ausgänge und Eingänge gekennzeichnet, und so ist natürlich auch beim Lautsprecherkabel eine der Adern gekennzeichnet.

Operationsverstärker sind nicht nur in gewöhnlichen Verstärkerschaltungen nützlich; in der Niederfrequenzelektronik werden sie als „Universalhalbleiter“ für alles Mögliche eingesetzt.

Präzisions-Gleichrichter

Bei der Verwendung in Stromversorgungen spielt die Genauigkeit des Gleichrichters kaum eine Rolle, denn wir wollen lediglich eine Wechselspannung möglichst effizient in eine Gleichspannung umwandeln. Wenn wir jedoch eine einfache Diodenschaltung als Gleichrichter in Messgeräten verwenden, stehen wir vor dem Problem, dass die Ausgangsspannung immer niedriger ist als die Eingangsspannung - und zwar um so viel, wie die Durchlassspannung der Diode beträgt.
Dieses Problem kann umgangen werden, indem man die Gleichrichterdiode an den Ausgang eines Operationsverstärkers anschließt und die Rückkopplung hinter der Diode platziert (siehe **Bild 1**). Der Operationsverstärker verstärkt das Signal so stark, dass der Spannungsabfall an der Diode genau kompensiert wird.

Größerer Ausgangsstrom

Operationsverstärker können in der Regel nur einen Ausgangsstrom von einigen Milliampere liefern. Obwohl sie über einen Kurzschlussschutz verfügen (damit ein zu niedriger Lastwiderstand nicht zu einer Beschädigung des Bauteils führt), sind sie nicht für einen großen Ausgangsstrom ausgelegt. Aber es gibt natürlich eine Lösung für dieses Problem: die Kombination eines Operationsverstärkers mit einer symmetrischen Ausgangsstufe (**Bild 2**).
Auch hier ist die Rückkopplung hinter den Transistoren angeschlossen, so dass die Schaltung trotz der BE-Durchlassspannungen der Transistoren keine allzu großen Verzerrungen erzeugt.
Diese geringe Verzerrung verdanken wir jedoch ausschließlich der großen Leerlaufverstärkung, die, wie wir inzwischen wissen, mit steigender Frequenz schnell abnimmt. Die in Bild 2 gezeigte Schaltung eignet sich in der Regel sehr gut für Gleichstromanwendungen, aber Audiosignale werden eher schlecht verarbeitet.
In diesem Fall empfiehlt sich der Einsatz einer Klasse-B-Ausgangsstufe, wie in **Bild 3** dargestellt.
In diesem Fall liefert eine Diode (die über einen Widerstand mit Strom versorgt wird)

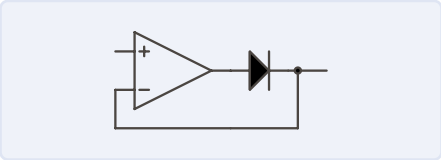


Bild 1. Präzisionsgleichrichter.

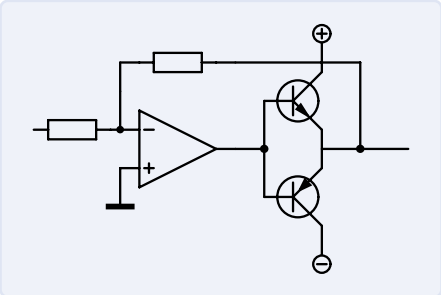


Bild 2. Erhöhen des Ausgangsstroms eines Operationsverstärkers.

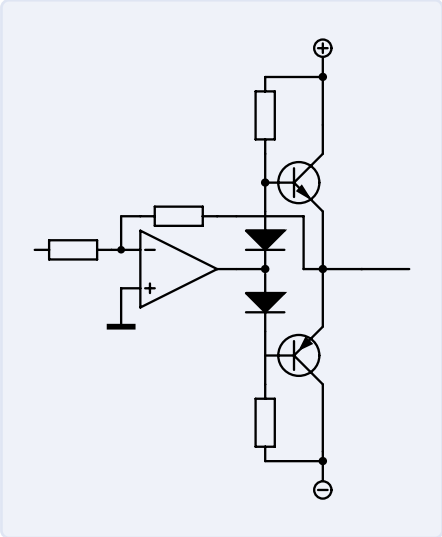


Bild 3. Geringere Verzerrung durch Einsatz einer Klasse-B-Ausgangsstufe.

die Vorspannung des jeweiligen Transistors. Wenn echte Spitzenqualität gefragt ist, dann kann eine Schaltung wie in **Bild 4** verwendet werden. Diese Schaltung aus dem CD-Spieler A725 von Studer treibt einen Ausgangsübertrager. Dieser professionelle CD-Spieler kam 1986 auf den Markt und wird immer noch für viel Geld auf Plattformen wie eBay angeboten.

Die Vorspannung für die Leistungstransistoren in dieser Schaltung wird nicht durch eine Diode, sondern durch einen nahezu identischen Transistor bereitgestellt. Dieser Transistor stammt aus demselben Wafer (also derselben Siliziumscheibe, aus der die einzelnen Transistoren herausgeschnitten werden) und ist zudem mit dem Ausgangstransistor thermisch gekoppelt - zu diesem Zweck werden die Transistoren zusammengeklemmt und mit einem Kupferring fixiert. Auf diese Weise hat die Vorspannung immer den optimalen Wert. Um mögliche Störungen zu unterdrücken, wird die Spannung von einem Elektrolytkondensator gepuffert.

Zwei weitere Details dieser Schaltung sind erwähnenswert: Erstens wird der Ausgangstransformator über eine Brückenschaltung gesteuert, so dass keine großen Ströme durch die Masseverbindung fließen und Störspannungen über den Verdrahtungs-/Leitungswiderstand verursachen können. Wenn Sie die bestmögliche Signalqualität wünschen, sollten Sie Ihre Masseleitung stets so weit wie möglich stromfrei halten! Zweitens wird die Versorgungsspannung für die Ausgangstransistoren über PTC-Widerstände angeschlossen. Daraus ergibt sich ein besonders wirksamer Überlastungsschutz, denn wenn die Lastimpedanz zu niedrig ist, wird die Versorgungsspannung einfach begrenzt.

Man denke an Situationen, in denen ein Lautsprecher zu Testzwecken einfach an die XLR-Ausgangsbuchse angeschlossen wird. Wäre der Überlastschutz mit Relais realisiert, die einfach die Last abschalten, dann wäre bald nichts mehr zu hören; und wenn eine größere Lastimpedanz angeschlossen würde, dann wäre die Audioqualität im Normalbetrieb bereits verschlechtert.

Wir bedanken uns bei der Firma Studer für die Erlaubnis, diese Schaltung hier zu veröffentlichen. Nicht nur das, sie geben - alte Schule - auch einen wertvollen Tipp:

Für Bastler, die diese Schaltung nachbauen wollen, sei angemerkt, dass die Schaltung nur dann einwandfrei funktioniert, wenn die Transistorpaare vom selben Wafer stammen (für die meisten Elektroniker wird es ein unüberwindbares Problem sein, solche Transistorpaare zu bekommen). Außerdem sind diese Transistoren paarweise thermisch gekoppelt. Wenn diese Anforderungen nicht erfüllt werden können, sollten die Widerstände von 3,3 Ω sicherheitshalber auf etwa 10 Ω erhöht werden.

Differenzierglied

Operationsverstärker können auch zur Durchführung komplexer mathematischer Operationen wie Differenzierung und Integration verwendet werden. **Bild 5** zeigt das Prinzipschaltbild eines Differenzierers. Die Ausgangsspannung kann mit der folgenden Formel berechnet werden:

$$U_{out} = -R \cdot C \cdot \frac{\Delta U_{in}}{\Delta t}$$

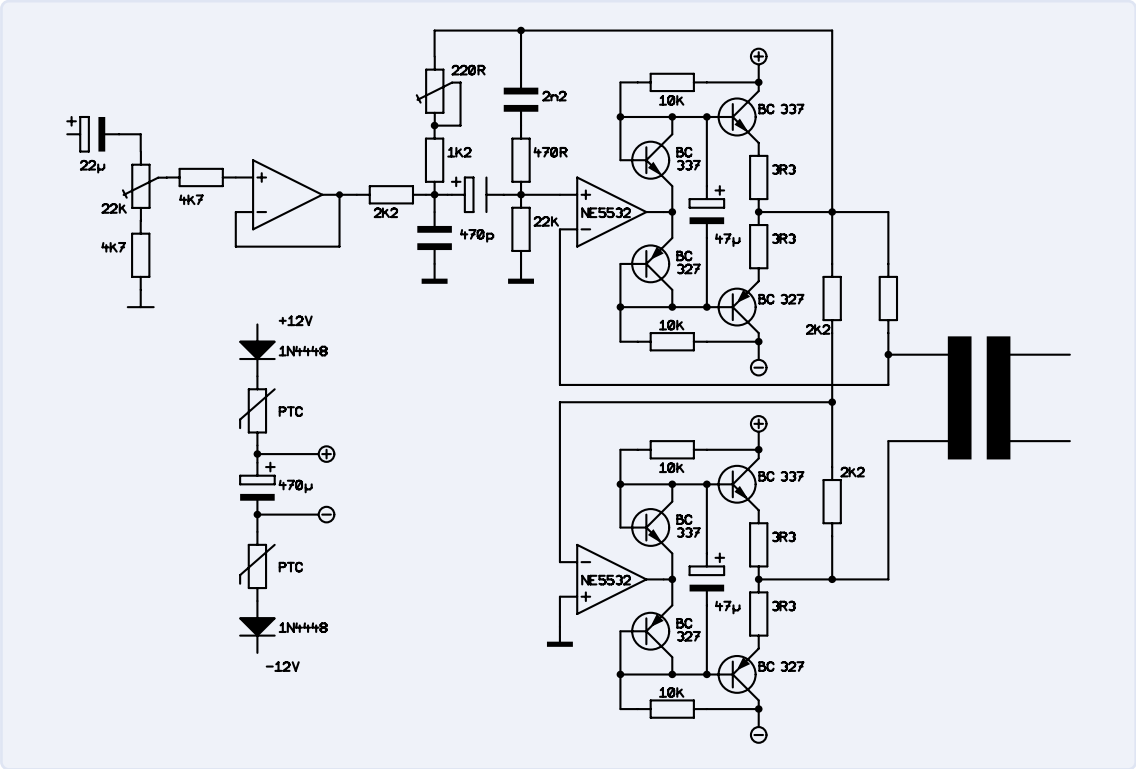


Bild 4. Ausgangsverstärker Studer A725.

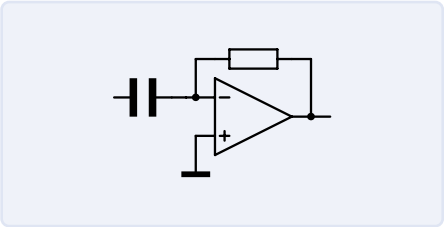


Bild 5. Differenzierglied.

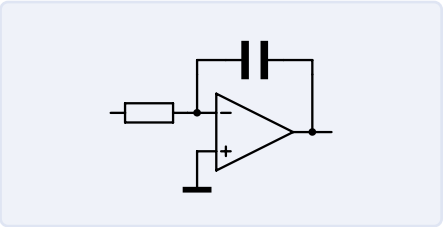


Bild 6. Integrator.

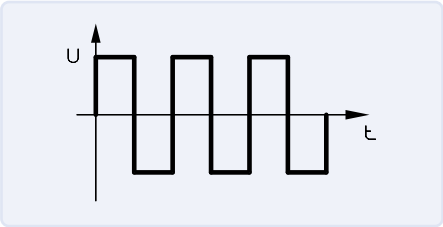


Bild 7. Rechtecksignal.

Integrator

Der Integrator in **Bild 6** ist das Gegenstück zum Differenzierer. Wie bei der Ausgangsspannung des Integrators ist:

$$U_{out} = \frac{1}{R \cdot C} \cdot \int U_{in} \cdot dt + C$$

Der Term „+ C“ bereitet dem experimentierfreudigen Elektroniker oft Kopfzerbrechen, da er dazu neigt, so groß zu werden, dass das Endergebnis den Ausgangsspannungsbereich des Operationsverstärkers überschreitet. Offsetspannungen und -ströme haben hier einen negativen Einfluss, da sie extrem schnell integriert werden und die Ausgangsspannung gegen die positive oder negative Versorgungsspannung abfallen lassen.

Rechteckgenerator

Der Rechteckgenerator ist ein Signalgenerator, der eine Rechteckwelle erzeugt (**Bild 7**) (aber das haben Sie wahrscheinlich schon erraten). In einer früheren Folge haben wir einen astabilen Multivibrator verwendet, um eine solche Rechteckwelle zu erzeugen. Die Opamp-Schaltung in **Bild 8** hat den Vorteil, dass sie nur einen Kondensator benötigt.

Wir nehmen an, dass der Ausgang gegen die positive Versorgungsspannung geschaltet ist. Der Spannungsteiler R2/R3 teilt die Ausgangsspannung auf (zum Beispiel) die Hälfte. C1 wird nun über R1 aufgeladen. Sobald die Spannung an C1 so groß wird wie die Spannung am nicht-invertierenden Eingang des Operationsverstärkers, kippt der Ausgang und wird negativ, und damit auch der nicht-invertierende Eingang. C1 wird nun über R1 entladen. Sobald die Spannung am Kondensator auf das Niveau des nicht-invertierenden Eingangs gesunken ist, kippt der Ausgang des Operationsverstärkers zurück und wird positiv. Und das geht so lange, bis jemand den Stecker zieht.

Dreieckgenerator

Die Schaltung in Bild 8 kann auch als Dreieckgenerator verwendet werden: Wenn die Spannung am nicht-invertierenden Eingang auf den kleinstmöglichen Wert heruntergeteilt wird, wird der Kondensator mit einem nahezu konstanten Strom geladen, so dass am invertierenden Eingang annähernde eine Dreiecksspannung erscheint.

Es ist jedoch auch möglich, eine Dreiecksspannung mit Hilfe eines Integrators zu erzeugen (**Bild 9**). Der linke Operationsverstärker ist ein Rechteckgenerator, wie wir ihn im vorherigen Abschnitt kennengelernt haben. Der rechte Operationsverstärker integriert die Rechteckspannung, so dass daraus ein dreieckförmiges Signal entsteht. Wenn wir keine Maßnahmen ergreifen würden, würde die Ausgangsspannung an der positiven oder negativen Versorgungsspannung „anschlagen“, was große Verzerrungen zur Folge haben würde. Deshalb wird der Kondensator C3 über R5 aufgeladen und die Spannung an diesem Kondensator über R6 an den invertierenden Eingang des Operationsverstärkers geleitet. Auf diese Weise können wir verhindern, dass die Ausgangsspannung gegen die Versorgungsspannung ansteigt, aber das bedeutet, dass das Ausgangssignal nicht mehr exakt symmetrisch ist. Bei Bedarf kann die Ausgangsspannung zum Beispiel durch einen Hochpassfilter geführt werden. Belassen wir es vorerst dabei. In der nächsten Folge werden wir mit einigen interessanten kleinen Operationsverstärkerschaltungen fortfahren. ◀

RG – 240310-02

Anmerkung der Redaktion: *Diese Artikelserie „Aller Anfang...“ basiert auf dem Buch „Basiskurs Elektronik“ von Michael Ebner, das in deutscher und niederländischer Sprache bei Elektor erschienen ist.*

Sie haben Fragen oder Anmerkungen?

Haben Sie Fragen oder Anregungen zu diesem Artikel? Wenden Sie sich bitte an die Elektor-Redaktion unter redaktion@elektor.de.

Passendes Produkt

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

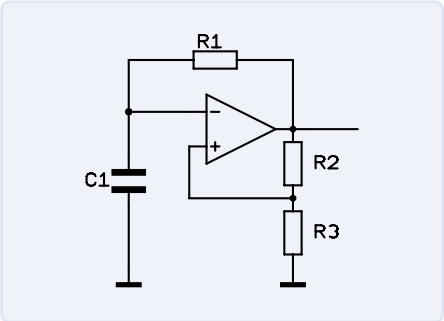


Bild 8. Rechteckgenerator.

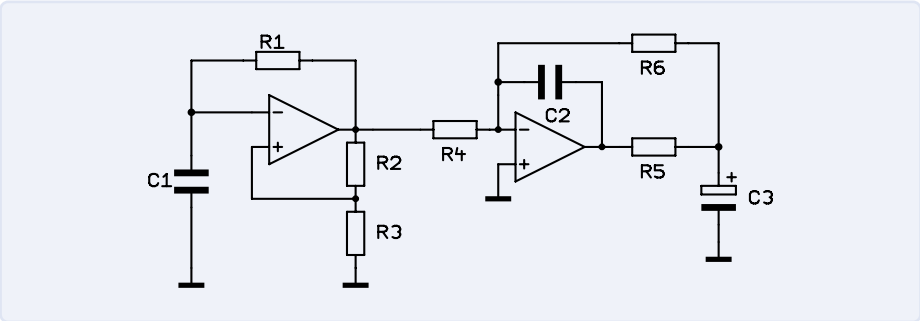
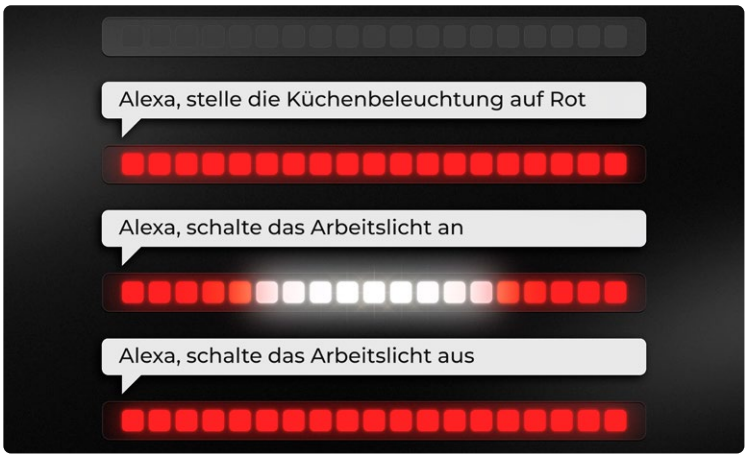


Bild 9. Dreieckgenerator.

Phoscon

MultiLayer-Segmentierung: Ein neuer Ansatz für adressierbare LED-Streifen

Ein Beitrag von dresden elektronik



Mit dem LED-Streifen kam Anfang der 2000er Jahre eine kostengünstige Lösung zur großflächigen Möbelbeleuchtung auf den Markt. Ob unter den Hängeschränken in der Küche, unter der Couch im Wohnzimmer oder im Gaming-PC des Nachwuchses – überall findet der LED-Streifen Anwendung. Durch seine Modularität lässt sich der Streifen in der Länge flexibel anpassen und erweitern, was zahlreiche Einsatzmöglichkeiten eröffnet.

Jedoch haben herkömmliche LED-Streifen ihre Einschränkungen. Da alle LEDs einer Farbe in einem Streifen parallelgeschaltet sind, leuchtet der gesamte Streifen immer in derselben Farbe. Dies führt oft zu einer statischen und einheitlichen Beleuchtung. Hier kommen adressierbare LED-Streifen ins Spiel. Diese verwenden einen Mikrocontroller für jedes LED-Modul (manchmal auch für mehrere Module, die ein Segment bilden). Man kann sich das wie ein Display mit einer Reihe von Pixeln vorstellen, wobei die einzelnen Module als Pixel fungieren. Diese Pixel sind in einer Kette verbunden, wobei der Datenausgang eines Pixels mit dem Eingang des nächsten Pixels verbunden ist. Dadurch entsteht ein großes Schieberegister, das über den Dateneingang des ersten Pixels angesteuert wird.

Vereinfacht gesagt hat jeder Pixel-Mikrocontroller die Aufgabe, die Farbe für sein Segment darzustellen und bei einem neuen Wert den alten Wert an das nächste Pixel weiterzugeben. So wird die Zielfarbe durch die gesamte Kette „geschoben".

Diese Technologie ermöglicht beeindruckende Lichteffekte und Animationen, die mit herkömmlichen LED-Streifen nicht realisierbar sind. Beispiele für solche Effekte sind „Rollender Regenbogen", „Sternschnuppe", „Knight Rider" oder „Sternenfunkeln". Diese Effekte zeigen die Möglichkeiten der adressierbaren LED-Streifen, bieten jedoch im Alltag jenseits der sporadischen Demonstration nur begrenzten Mehrwert.

MultiLayer-Segmentierung

Phoscon, eine Marke der dresden elektronik ingenieurtechnik GmbH, verfolgt daher mit dem neuen FLS-M (Matter-kompatibles Funk-Vorschaltgerät) einen innovativen Ansatz zur Nutzung von adressierbaren LED-Streifen: die MultiLayer-Segmentierung. Diese Technologie ermöglicht es, einen adressierbaren LED-Streifen in beliebig viele „virtuelle" Segmente zu unterteilen. Diese Segmente verhalten sich wie separate LED-Streifen und können verschiedenen Ebenen (engl.: Layers) zugeordnet werden. Segmente in höheren Ebenen sind im aktivierten Zustand dominant gegenüber Segmenten in niedrigeren Ebenen. Sind Segmente ausgeschaltet, wirken sie „transparent" und lassen die Segmente der unteren Ebenen „durchscheinen". Dieser Ansatz eröffnet neue Möglichkeiten zur flexiblen und vielseitigen Beleuchtungsgestaltung, indem er es erlaubt, denselben Bereich eines Streifens für mehrere Segmente zu nutzen.

Ein Beispiel

Stellen Sie sich vor, ein adressierbarer LED-Streifen wird in einer Küche installiert. Ein virtuelles Segment erstreckt sich über die gesamte Länge des Streifens und ist der Hintergrundebene zugeordnet. Ein weiteres Segment wird für den Bereich über der Spüle definiert und einer Vordergrundebene zugeordnet. Die Küche kann nun insgesamt in einer angenehmen Ambient-Farbe beleuchtet werden.

Sollte ein Arbeitslicht benötigt werden, kann das Vordergrund-Segment aktiviert werden, wodurch der Bereich über der Spüle in ein strahlendes Weiß wechselt. Wird das Arbeitslicht nicht mehr benötigt, kann das Segment abgeschaltet werden, sodass der Bereich über der Spüle wieder in der vorherigen Ambient-Farbe des Hintergrund-Segments leuchtet.

Mit der MultiLayer-Segmentierung können Anwender komplexe und mehrschichtige Lichteffekte realisieren, die weit über die Möglichkeiten herkömmlicher LED-Streifen hinausgehen. Dies eröffnet neue Perspektiven für kreative und funktionale Beleuchtungslösungen, die sowohl ästhetisch als auch praktisch überzeugen.

240414-02

Source: Adobe Stock

Open Vario

Open-Source Multifunktions-Variometer für Gleitschirmflieger

Von Cedric Jimenez (Frankreich)

Manchmal entstehen die besten Entwürfe aus einer Gelegenheit heraus! Cedric Jimenez, hörte von einem Freund vom STM32 Wireless Innovation Design Contest und gewann doch glatt den ersten Preis. Sein Beitrag zum Wettbewerb war ein innovatives Projekt zur variometrischen Messung der Flughöhe. Lesen Sie seine Geschichte!

Alles begann damit, dass mir ein Freund den Link zum *STM32 Wireless Innovation Design Contest* auf der Elektor-Webseite [1] weiterleitete: „Hey! Hast du diesen Wettbewerb gesehen? Er basiert auf einem Wireless-Chip von ST, und sie bieten ein kostenloses Entwicklungsboard an!" Er wusste, wie er mein Interesse wecken konnte, denn ich teste wirklich gerne neue Mikrocontroller, und ich bin ein großer Fan dieser Entwicklungsboards, mit denen man schnell Projekte einrichten und testen kann, um ihre Funktionen zu entdecken.

Auswahl der Hardware

Ich warf einen kurzen Blick auf die vorgeschlagenen Evaluierungsboards: zwei Nucleos (NUCLEO-WBA52CG, NUCLEO-WL55JC) und ein Discovery-Kit (STM32WB5MM-DK). Letzteres war meiner Meinung nach die attraktivste Basis für ein Projekt, weil es gut ausgestattet ist: Es enthält ein Display, Speicher, Sensoren, USB und Erweiterungsanschlüsse … Okay, ich hatte also ein Board [2], jetzt brauchte ich eine Projektidee.

Ich bin Gleitschirmpilot und habe schon lange mit dem Gedanken gespielt, eigene Fluginstrumente zu entwickeln. Vor etwa fünf Jahren habe ich sogar angefangen, über ein Hardware-Design nachzudenken, aber ich bin nicht sehr weit gekommen, hauptsächlich aus Zeitmangel. Deshalb habe ich diesen Wettbewerb zum Anlass genommen, diese Idee endlich in ein reales Gerät umzusetzen, das ich bei meinen Flügen verwenden kann. Beim Gleitschirmfliegen ist das Variometer, auch Steigmesser genannt, das wertvollste Gerät! Die Steigrate, ausgedrückt in Metern oder Fuß pro Sekunde, hilft bei der Suche nach den thermischen Strömungen, die es dem Gleitschirm ermöglicht, aufzusteigen und über mehrere Stunden in der Luft zu bleiben. Zusätzlich zu den Hauptfunktionen des Variometers verfügen kommerzielle Variometer oft über weitere Funktionen, beispielsweise Satelliten-Tracking (Global Navigation Satellite System, GNSS) und die Anzeige der Daten auf einem Smartphone.

Anforderungen an das Gerät

Ausgehend von den Funktionen, die das STM32WB5MM-DK-Board bietet, habe ich eine Liste der Funktionen erstellt, die ich implementieren wollte (**Bild 1**):

- Variometer
- GNSS-Positionsbestimmung
- Beschleunigungsmesser
- Thermometer
- Berechnung der Gleitzahl (horizontale Strecke, die bei einem Höhenverlust von 1 m zurückgelegt werden kann)

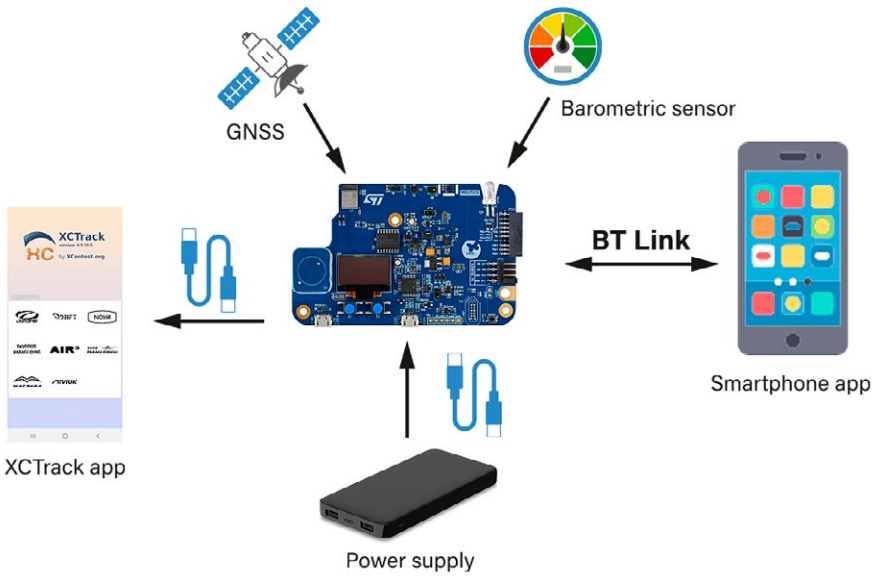


Bild 1. Das Entwicklungsboard STM32WB5MM-DK ist das Herzstück dieses Projekts.

- Flugdatenaufzeichnung (Höhe, Geschwindigkeit, Position…)
- Lokale Benutzerschnittstelle zur Anzeige von Sensor- und Flugdaten
- Bluetooth-LE-Verbindung zur Konfiguration von Parametern und zum Ablesen von Echtzeit-Flugdatenwerten
- USB-OTG-Anschluss zur Verbindung mit der bekannten Smartphone-App *XCTrack* (die als abgesetztes Display verwendet werden kann)
- USB-Verbindung für den Anschluss an selbst entwickelte PC-Python-Tools zum Exportieren von aufgezeichneten Flügen

Erforderliche zusätzliche Hardware

Obwohl das STM32WB5MM-DK-Board die meisten Anforderungen erfüllte, fehlten doch noch zwei Sensoren: GNSS für die Ermittlung der Geoposition des Geräts und ein Drucksensor zur Berechnung der Steigrate.

Die Berechnung der Steigrate erfolgt durch Messung der Differenz zwischen zwei gemessenen Höhen im Abstand von mehreren Sekunden. Ein GNSS berechnet die Höhe nicht präzise genug, so dass ein hochpräziser Barometer/Drucksensor erforderlich ist, um eine Genauigkeit von 15…30 cm bei der Höhenmessung zu erreichen.

Eine weitere Bedingung für die Auswahl zusätzlicher Sensoren war eine mechanisch robuste elektrischen Integration mit der STM32WB5MM-DK-Platine, die Vibrationen und Erschütterungen während der Starts, Flugturbulenzen und Landungen standhalten muss. Das gesamte Gerät muss außerdem so kompakt wie möglich sein, um leicht transportiert werden zu können. Deshalb entschied ich mich, den Arduino-UNO-kompatiblen Anschluss auf dem STM32-Board zu nutzen und auf ein spezielles Shield [3] gesteckte MikroE-Click-Boards zu verwenden. Damit wurden Lötarbeiten und Drahtbrücken vermieden. MikroElektronika bietet eine große Auswahl an gebrauchsfertigen Sensormodulen an (mehr als 1000!) und ich habe schon oft welche für schnelles Prototyping verwendet. Für das GNSS hatte ich bereits ein GNSS-4-Click-Board [4] von MikroE mit dem AM-M8Q-Modul von U-blox, das eine GNSS-Patch-Antenne mit omnidirektionalem Strahlungsdiagramm und eine UART-Schnittstelle enthält.

MikroE bietet drei hochpräzise Barometer/Drucksensoren an, die für mein Projekt verwendet werden könnten. Ich stöberte in Gleitschirmforen, sah mir die vorhandenen Dokumentationen an und entschied ich mich schließlich für das kleine Board Altitude-2-Clic [5] mit einem Drucksensor MS5607 von TE Connectivity, der auch in anderen Variometer-Projekten verwendet wird. Das Board ist einfach zu bedienen und gut dokumentiert und kann entweder über den I²C- oder den SPI-Bus angeschlossen werden.

Ich habe also alles zusammengesteckt, um zu sehen, wie es passt (ein Blockdiagramm des Systems sehen Sie in **Bild 2**), schaltete ein und … böse Überraschung: Weder der I²C- noch der SPI-Bus waren an den Pins verfügbar, die das barometrische Sensormodul doch dringend benötigt! Okay, das war nicht wirklich ein Hindernis, der I²C-Bus ist kein komplexes Protokoll, also setzte ich einen Bitbanging-I²C-Treiber als Workaround ein.

Software-Entwurf

Nun, da ich meinen Hardware-Entwurf zusammen hatte, konnte ich anfangen, über die Software nachzudenken. Ich mag es, meinen Quellcode mit anderen zu teilen, und ich mag es auch, die Arbeit anderer Leute zu verwenden. Deshalb gebe ich den Großteil meines Codes unter Open-Source-Lizenzen weiter, und deshalb ist auch dieses Projekt Open-Source [6]. Ich habe mich bei diesem nicht-kommerziellen Projekt für die MIT-Lizenz entschieden, um zukünftigen Mitwirkenden/Nutzern ein Maximum an Flexibilität bei der Verwendung zu ermöglichen.

Als Programmiersprache habe ich meine Lieblingssprache C++ verwendet. Sie bietet alle Low-Level-Fähigkeiten von C und darüber hinaus alle Merkmale einer modernen objektorientierten Sprache: Typensicherheit, Vererbung, Templates. Die meisten Leute denken, C++ bedeutet dynamische Speicherzuweisung, aber ist die Verwendung von `malloc()` und `free()` in einem C-Programm obligatorisch? Nein, ebenso wenig wie `new` und `delete` in C++. Es hängt nur von der verwendeten Untermenge der Sprache ab, und es gibt so viele Möglichkeiten ohne dynamische Zuweisungen.

ST bietet eine Menge Software-Tools für seine Mikrocontroller (STM32CubeIDE, STM32CubeMX und so weiter) und durch den Gerätetreiber HAL und die STM32Cube-MCU-Pakete viele Beispiele. Zwar schätze ich diese Tools sehr, da sie mir helfen, den Entwurf kleiner

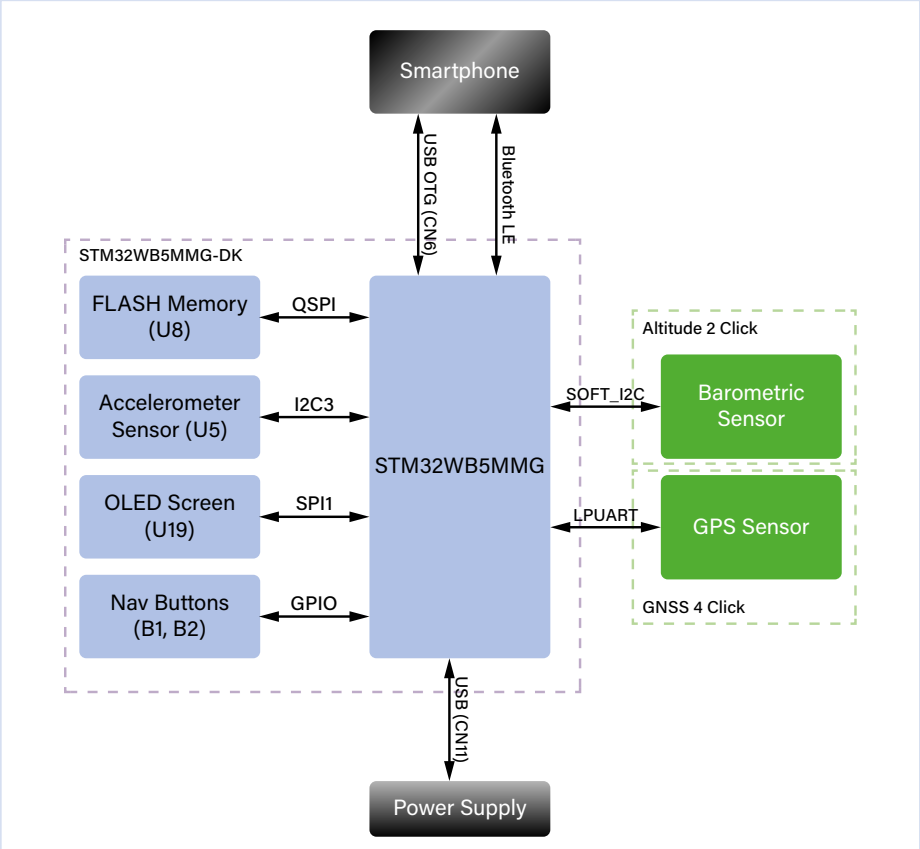


Bild 2. Das Blockschaltbild der Funktionen des Systems zeigt die physikalischen Schnittstellen zu den externen Modulen.

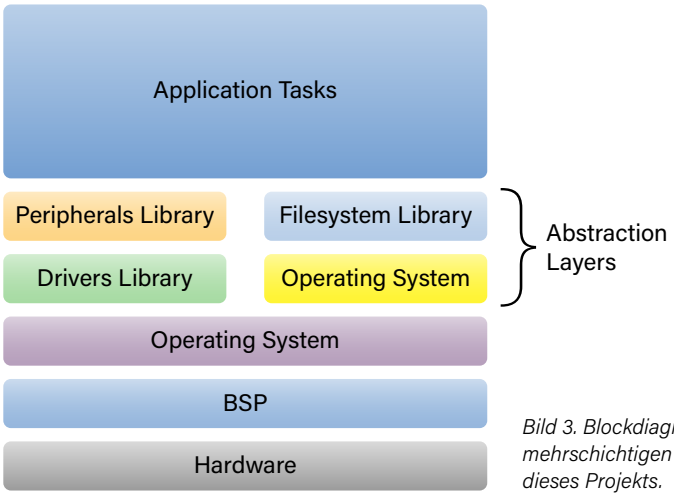


Bild 3. Blockdiagramm der mehrschichtigen Softwarearchitektur dieses Projekts.

Prototypen zu beschleunigen, indem sie einen Teil des Codes generieren und mich auf die richtigen Bibliotheken hinweisen, aber ich bevorzuge dennoch Visual Studio Code mit der C/C++-Erweiterung für Code-Highlighting und Browsing-Möglichkeiten sowie die Cortex-Debug-Erweiterung zum Debuggen von Embedded-Geräten mit ST-Links- oder J-Links-Debug-Sonden.
So hat man die gleichen Funktionen wie die STM32CubeIDE, ist aber durch Plugins und JSON-Konfigurationsdateien einfacher zu konfigurieren und anzupassen. Ich habe es außerdem vorgezogen, mein eigenes CMake/Make-Build-System zu schreiben, um völlig unabhängig von der verwendeten IDE zu sein (kein generierter Code durch ST-Tools) und so die volle Kontrolle über den Quellbaum und die Quelldateien zu haben.
Neben *STM32-HAL*, der *STM32-USB*-Gerätebibliothek und natürlich dem *STM32-WPAN*-Stack für Bluetooth habe ich mich für die folgenden Bibliotheken von Drittanbietern entschieden (alle auf Github verfügbar):

- FreeRTOS 10.6.2: Echtzeit-Betriebssystem (ich finde es, wenn die Ressourcen nicht zu knapp sind, viel einfacher, ein solches anstelle eines Entwurfs mit einer einzigen Loop zu verwenden)
- Little-FS 2.81: Einfach zu verwendendes, leichtgewichtiges, gegen Stromausfall geschütztes Dateisystem für NOR-Speicher
- YACSGL/YACSWL 0.0.1: Grafische Bibliothek mit geringem Platzbedarf, zugeschnitten auf schwarz-weiße OLED- und/oder ePaper-Displays

Zunächst habe ich für all diese Bibliotheken einen C++-Wrapper erstellt, um sie leichter in C++-Code zu integrieren. Und für einige von ihnen, wie die `mutex`-Klasse, habe ich das Standard-Verhalten `std::mutex` von C++ nachgeahmt, so dass der Code leicht geschrieben werden kann, wenn man in C++ firm ist. Dann beschloss ich, eine „klassische", aber dennoch effiziente Software-Schichtarchitektur zu implementieren (siehe **Bild 3**). Darüber hinaus habe ich ein von mir so genanntes „Board Abstraction Pattern" verwendet, das eine Schnittstelle definiert, die es mir ermöglicht, alle meine Objekte, die mit Peripheriegeräten und Gerätetreibern umgehen, über eine abstrakte Schnittstelle zu erreichen. Ok, das war jetzt eine Menge Theorie, also schauen wir uns jetzt den Code in **Listing 1** an, der auch auf GitHub [6] verfügbar ist.
Mit dieser Schnittstelle konnte ich in meinen Anwendungen problemlos auf die Hardware-Peripheriegeräte und -Treiber zugreifen, ohne mich um deren Implementierung kümmern zu müssen. Auf diese Weise konnte ich zum Beispiel ganz einfach Sensor-Ausgangssignale simulieren, um die Entwicklung der Algorithmen in der Anwendung zu beschleunigen. Bei einem anderen Projekt habe ich dasselbe Muster verwendet, um die Hardware zu abstrahieren und die gleiche Software auf verschiedenen Boards laufen zu lassen.

Testen

Der letzte Schritt bestand nun darin, mein Gerät unter realen Flugbedingungen zu testen. Von den simulierten Eingaben wusste ich, dass meine Algorithmen gültige Werte berechneten und sich korrekt verhielten, wenn ich die Parameter ihrer Filter änderte. Aber beim Gleitschirmfliegen dreht sich alles um Gefühle; daher musste ich diese Parameter während echter Flüge anpassen, damit sich das Gerät meiner Art zu fliegen entsprechend verhält. Und die Parameter sollten auch leicht zu konfigurieren sein, damit ein anderer Pilot sie anpassen kann.

Ich konnte das Gerät nicht der „freien Luft" aussetzen, da die direkte Sonneneinstrahlung (und der relative Luftstrom) den sehr empfindlichen Drucksensor beeinträchtigen würden. Stattdessen musste ich die Daten auf meinem Smartphone visualisieren und auf diesem Weg auch die Parameter steuern. Es gibt zwei Möglichkeiten, das STM32-Board mit einem Smartphone zu verbinden, nämlich Bluetooth und USB-OTG, und ich habe beide genutzt. Die Bluetooth-Verbindung stellt alle Parameter in verschiedenen Diensten und Eigenschaften zur Verfügung, was man auch dem Dokument, das ich für dieses Projekt auf die Elektor-Labs-Webseite gestellt habe [7], entnehmen kann. Leider hatte ich keine Gelegenheit, eine spezielle Smartphone-App mit einer schicken Benutzeroberfläche zu entwickeln, aber die Parameter lassen sich mit kostenlosen generischen BLE-Apps wie *BLE Scanner* (siehe **Bild 4**) leicht anzeigen und anpassen. Eine weitere Möglichkeit, den Smartphone-Bildschirm beim Fliegen zu nutzen, ist die unter Gleitschirmfliegern weit verbreitete XCTrack-App für Smartphones, die den OTG-Link des Discovery Kits nutzt. Damit

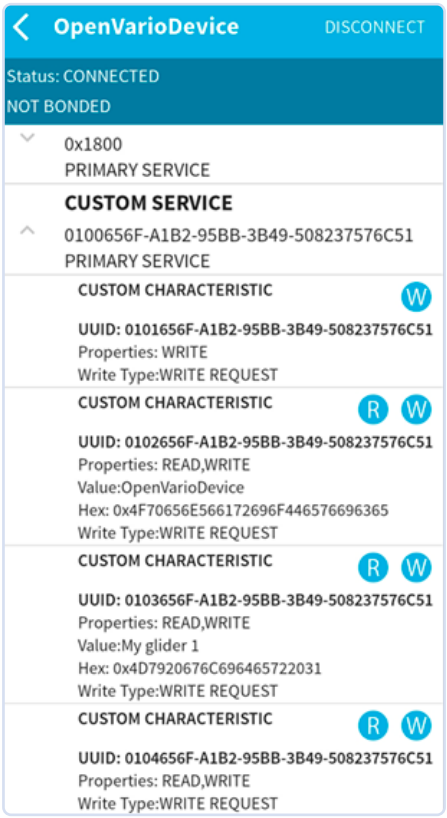


Bild 4. Screenshot der BLE-Scanner-App, mit der die laufenden Dienste angepasst werden können.

Bild 5. Mit einem kommerziellen Variometer habe ich die Messungen an meinem Open Vario verglichen.

konnte ich die von meinem Gerät berechneten Flugdaten auf meinem Smartphone visualisieren. Ich fügte auch das „kommerzielle" Variometer hinzu, das ich normalerweise verwende, um die berechneten Werte zu vergleichen, wie in **Bild 5** gezeigt.

Während der Flüge zeichne ich in regelmäßigen Abständen (Standardeinstellung ist 1 s) alle Ausgaben meiner Sensoren und Algorithmen zusammen mit meiner GNSS-Position auf. Jeder Flug wird in einer separaten Datei auf dem QSPI-NOR-Flash gespeichert. Mit den Standardeinstellungen kann ich bis zu fast 24 Stunden Flug aufzeichnen, was völlig ausreichend ist. Ich habe Python-Skripte entwickelt, um diese Dateien über den USB-Geräteanschluss des Discovery Kits auf einem PC zu überspielen und dort als CSV-Datei zu speichern, so dass sie leicht mit Excel analysiert werden können (**Bild 6**) und auch als KML-Datei in der Google Earth visualisiert werden können, um meinen Flug in 3D zu betrachten, wie in **Bild 7** dargestellt.

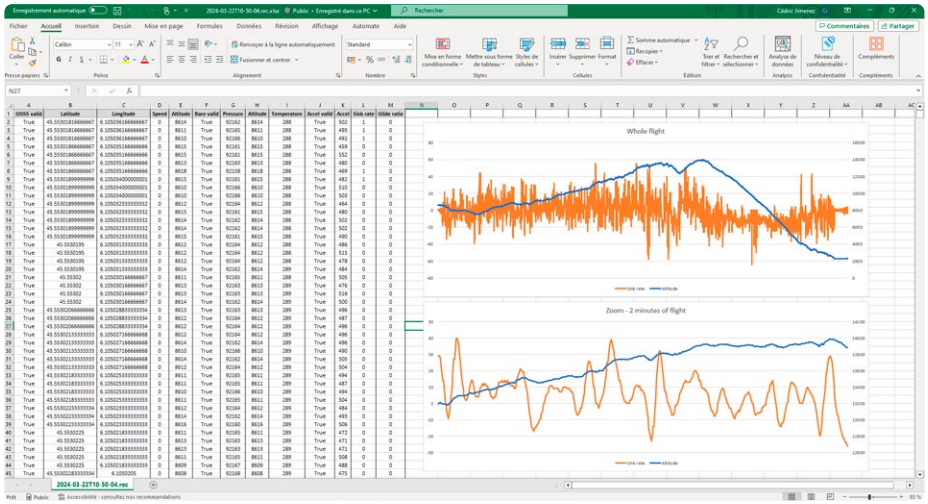


Bild 6. Die erfassten Daten, gespeichert in einer .csv-Datei, werden mit Excel analysiert und angezeigt.

Jede Menge Möglichkeiten

Ich denke, Sie haben bereits gemerkt, dass ich die Teilnahme an diesem Wettbewerb sehr genossen habe, auch wenn sie etwas zeitaufwändig war (5...10 Stunden pro Woche)! Er hat mir viele Möglichkeiten geboten: einen neuen Mikrocontroller zu entdecken, ein Projekt zu entwickeln, für das ich mir sonst nie Zeit genommen hätte, meine beiden Lieblingshobbys zu kombinieren und andere spannende Projekte zu entdecken, die die gleiche Hardware verwenden.

Auch der begrenzte Zeitrahmen für den Wettbewerb (danke übrigens für die Fristverlängerung) war eine große Herausforderung: Jedes Projekt bringt eine Menge techni-

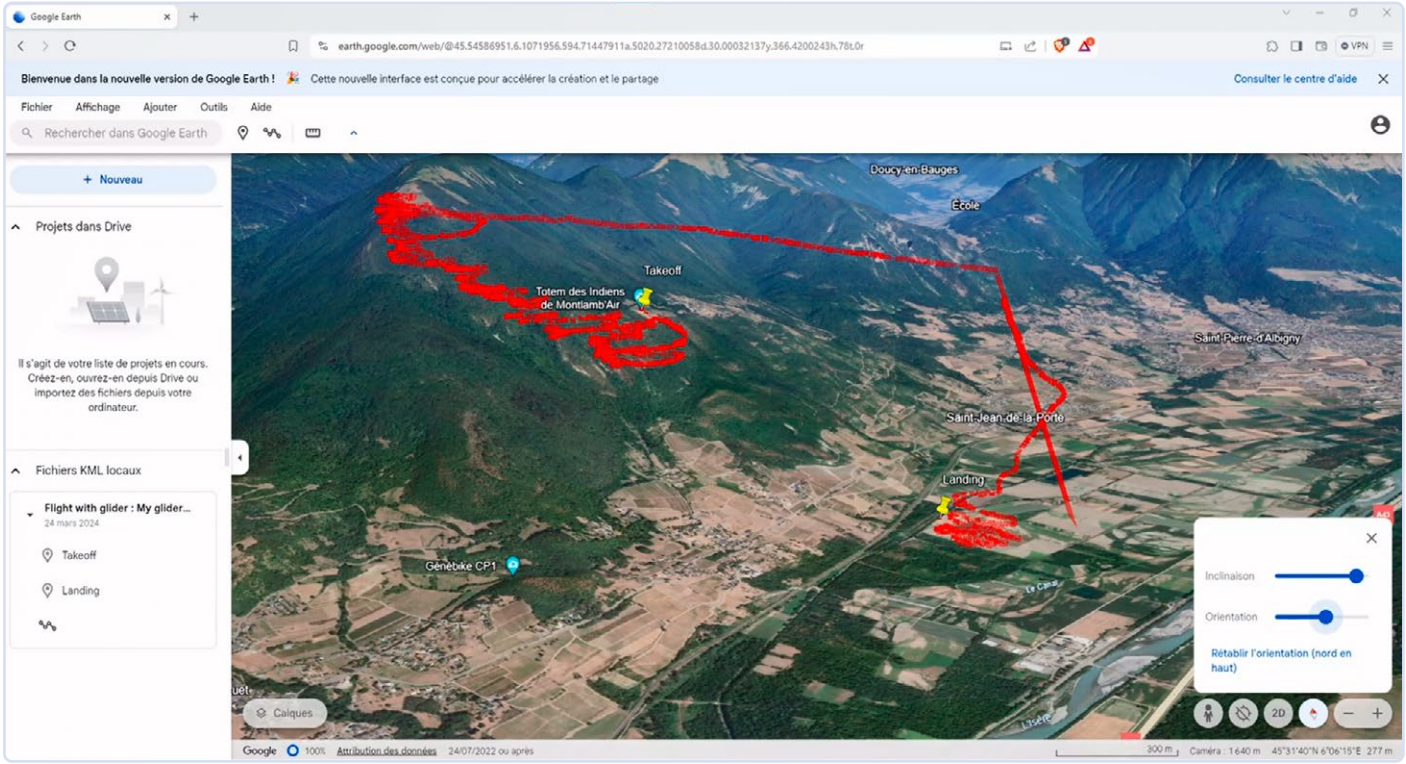


Bild 7. 3-D-Visualisierung der Flugdaten (als KML-Datei) in Google Earth.

scher Probleme mit sich, die es zu bewältigen gilt, und am Ende sollte das Projekt eine attraktive Anzahl von Funktionen bieten und funktional sein. Ich denke, dass mir mein Job als Embedded Software Engineer dabei sehr geholfen hat, da ich regelmäßig mit dieser Art von Einschränkungen konfrontiert werde.

Vielen Dank, Elektor und ST Microelectronics, für die Organisation dieses Wettbewerbs: Ich hoffe, dass Sie in Zukunft weitere Wettbewerbe organisieren werden!

SG – 240325-01

Über den Autor

Cedric Jimenez ist 40 Jahre alt und glücklicher Vater einer achtjährigen Tochter. Er lebt in den französischen Alpen, wo er viele Bergsteigeraktivitäten (Klettern, Paragliding) betreibt. Cedric ist ein großer Fan der Softwareentwicklung für Eingebettete Systeme. Er begann im Alter von neun Jahren mit der Programmierung in BASIC auf einem Thomson TO7/70 und wechselte einige Jahre später zu C++.
Obwohl er viele Programmiersprachen gelernt hat (VB, C#, C, Pascal, FORTRAN, Python, Ruby...), ist C++ immer noch seine Lieblingssprache.
Er hat einen Master-Abschluss in Informatik mit einer Spezialisierung auf Eingebettete Echtzeitsysteme. Er arbeitet seit mehr als 15 Jahren als Subunternehmer in verschiedenen Bereichen (Eisenbahn/U-Bahn, Kräne, medizinische Geräte...) und ist derzeit als Architekt für Embedded Software bei Schneider Electric tätig.

Sie haben Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Dann schreiben Sie bitte an den Autor unter cjz.73fr@gmail.com oder an die Elektor-Redaktion unter redaktion@elektor.de.

Listing 1: Eine Board-Abstraktionsklasse

```
/*
 * Copyright (c) 2023 open-vario
 * SPDX-License-Identifier: MIT
 */
#ifndef OV_I_BOARD_H
#define OV_I_BOARD_H
#include "i_accelerometer_sensor.h"
#include "i_barometric_altimeter.h"
#include "i_ble_stack.h"
#include "i_button.h"
#include "i_display.h"
#include "i_gnss.h"
#include "i_serial.h"
#include "i_storage_memory.h"
#include "i_usb_cdc.h"

namespace ov
{

/** @brief Interface for boards implementations */
class i_board
{

  public:
    /** @brief Destructor */
    virtual ~i_board() { }

    /** @brief Reset the board */
    virtual void reset() = 0;

    /** @brief Get the debug serial port */
    virtual i_serial& get_debug_port() = 0;

    /** @brief Get the USB CDC port */
    virtual i_usb_cdc& get_usb_cdc() = 0;

    /** @brief Get the storage memory */
    virtual i_storage_memory& get_storage_memory() = 0;

    /** @brief Get the display */
    virtual i_display& get_display() = 0;

    /** @brief Get the 'Previous' button */
    virtual i_button& get_previous_button() = 0;

    /** @brief Get the 'Next' button */
    virtual i_button& get_next_button() = 0;

    /** @brief Get the 'Select' button */
    virtual i_button& get_select_button() = 0;

    /** @brief Get the BLE stack */
    virtual i_ble_stack& get_ble_stack() = 0;

    /** @brief Get the GNSS  */
    virtual i_gnss& get_gnss() = 0;
```

```
    /** @brief Get the barometric altimeter */
    virtual i_barometric_altimeter& get_altimeter() = 0;

    /** @brief Get the accelerometer */
    virtual i_accelerometer_sensor& get_accelerometer() = 0;
};

} // namespace ov

#endif // OV_I_BOARD_H
```

Passendes Produkt

- **Majid Pakdel, *Advanced Programming with STM32 Microcontrollers*, Elektor, 2020**
  E-Buch, PDF, englisch:
  www.elektor.de/19527 7

WEBLINKS

[1] Webseite STM32 Contest: https://tinyurl.com/bdenyy24
[2] Webseite STM32WB5MM-DK Evaluation Tool: https://tinyurl.com/4dp83f8h
[3] MikroElektronika: Arduino Uno Click Shield: https://www.mikroe.com/arduino-uno-click-shield
[4] MikroElektronika: GNSS 4 Click Board: https://www.mikroe.com/gnss-4-click
[5] MikroElektronika: Altitude 2 Click Board: https://www.mikroe.com/altitude-2-click
[6] Github-Seite des Autors für Vario: https://github.com/open-vario/open-vario
[7] Elektor-Labs-Webseite für dieses Projekt: https://tinyurl.com/4rmn5j28

Aus dem Leben gegriffen

Über die Selbstverständlichkeit der Dinge

Bild 1. Frustration hinter dem Lenkrad (Quelle: Adobe Stock / EdNurg).

Von Ilse Joostens (Belgien)

Ab und zu brauche ich ein gutes Gespräch über Technik und Wissenschaft oder einen Abend, an dem ich mit Gleichgesinnten über neue Ideen und Projekte diskutiere. Im Alltag jedoch drehen sich die meisten Gespräche - leider - um schreckliche Krankheiten und die dazugehörigen medizinischen Behandlungen, das Wetter, zwischenmenschliche Probleme, Sport, Fernsehsendungen, Festivals und heutzutage sogar immer mehr um schwammige Dinge wie Astrologie, Schamanismus oder die sogenannten Gefahren der 5G-Strahlung. Dennoch nutzen diese Menschen täglich fortschrittliche Elektronik, ohne darüber nachzudenken, egal wie untechnisch und feenhaft sie auch sein mögen. Viele technologische (R)Evolutionen sind so selbstverständlich geworden, dass wir nicht mehr darüber nachdenken, wie das Leben früher war, als es diese Technologie noch nicht gab.

Landkarten lesen

Viele von uns benutzen ein Satellitennavigationssystem, umgangssprachlich „Navi". Früher waren das noch recht klobige Geräte mit einer auffälligen Antenne [1]. Heutzutage sind solche Systeme in modernen Autos eingebaut, und man kann auch kostenlose Apps auf dem Smartphone verwenden. Äußerst nützlich sind auch die Echtzeitinformationen über Baustellen, Unfälle, Staus, Geschwindigkeitsbegrenzungen, Wetterbedingungen, Streckensperrungen, Geschwindigkeitskontrollen und so weiter. Meine bessere Hälfte war schon immer ein großer Liebhaber von Navigationssystemen, und ich habe inzwischen aufgehört zu zählen, wie oft wir in die falsche Richtung gefahren sind, was zu viel verlorener Zeit und Stress geführt hat, aber auf diese Weise behalte ich die Erinnerung daran, wie es früher einmal war. Damals gab es diese unhandlich großen, faltbaren Straßenkarten, mit denen man am Esstisch die Route planen konnte, die aber hinter dem Steuer des Autos sehr frustrierend waren (**Bild 1**). Es gab auch diese Atlanten in handlicherem Format. Dann blätterte man hin und her, und jene wichtige Ausfahrt, Abzweigung oder sogar das Ziel befanden sich immer am Rande einer solchen Seite beim Übergang zu einer anderen Seite - 30 Seiten weiter im Buch.

Ebenso sind auch Google Maps und Google Street View [2] zu einem nicht mehr wegzudenkenden Bestandteil unseres Lebens geworden, und ich mag Street View besonders, um zur Abwechslung einmal verschiedene Orte auf der Welt zu erkunden, ohne dorthin reisen zu müssen. Das macht definitiv mehr Spaß als diese endlosen Dia-Abende mit Urlaubsfotos und dazugehörigen Kommentaren, wie es sie früher gab. Ich verwende Google Maps auch, um die Adressen von Kunden zu überprüfen, und dann habe ich mir aus Neugierde auch schon mal Street View angesehen.

Der Butler war's

Die Nutzung des Internets war nicht so bequem, als man noch über eines dieser quietschenden und pfeifenden Modems eine Verbindung herstellen musste, aber heutzutage sind wir immer und überall online, ohne überhaupt darüber nachzudenken. Das Versenden von Nachrichten, Fotos und Videos und sogar Videotelefonie - all das ist so normal geworden. Vor nicht allzu langer Zeit, im Prä-SMS-Zeitalter, gab es für kurze Nachrichten nur Postkarten, Telegramme [3] und später Pager, wenn es eilig war.

Ich bin das, was man einen „Kabelschneider" nennt, sowohl im wörtlichen Sinne des Wortes, wenn ich mal wieder mit dem Seitenschneider beschäftigt bin, als auch im übertragenen Sinne des Wortes (im Niederländischen jemand, der nicht mehr das allgegenwärtige Kabelfernsehen nutzt, sondern Streaming-Diensten den Vorzug gibt). Lineares Fernsehen ist sowieso nicht mehr mein Ding, und trotz Dutzenden von Kanälen gibt es im Fernsehen meist nichts, was mich fesseln kann, ganz zu schweigen von den lästigen Werbeunterbrechungen. Gebt mir Streaming-Dienste, damit kann man sich Filme und Serien bequem vom Sessel aus ansehen. Früher musste man bei Wind und Wetter in die Videothek gehen und riskierte

Bild 2. Produkt Generativer KI (Quelle: Ilse Joostens).

ein Bußgeld, wenn man seine Videokassette zu spät zurückbrachte oder vergaß, sie zurückzuspulen. Und es hatte eine gewisse Atmosphäre, wenn man in dieser geheimnisvollen und verbotenen Ecke hinter dem Vorhang schnüffeln durfte.

In einer früheren Kolumne habe ich mich mit der künstlichen Intelligenz befasst, die sich seither rasant entwickelt hat. Inzwischen habe auch ich begonnen, die generative KI intensiver zu nutzen (**Bild 2**), einschließlich ChatGPT, für das ich jetzt mehr Anwendungsbereiche habe.

Ich schätze, dass KI in naher Zukunft auch in der Rolle von James, Ihrem persönlichen digitalen Assistenten, Berater und Butler, alltäglich werden wird. Das ist nützlich, um sich aus der Verantwortung zu stehlen, wenn mal etwas schief läuft. Allerdings befürchte ich immer noch den Missbrauch von KI durch Regierungen, denn in Schweden und Deutschland will die Polizei Kameras mit Gesichtserkennung einsetzen [4].

Verwöhnte Elektroniker

Auch in der Elektronik hat sich in den letzten Jahrzehnten vieles zum Besseren gewendet. Das Entwerfen von Platinen war noch nie so einfach wie heute, mit vernünftigen, erschwinglichen und manchmal sogar kostenlosen Softwarepaketen. Ich arbeite schon seit Jahren mit KiCad [5] zu meiner vollsten Zufriedenheit. Selbst die Herstellung von Platinen

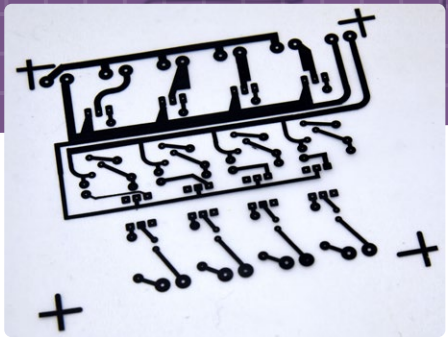


Bild 3. Abrubbel-Symbole auf Pauspapier (Quelle: Ilse Joostens).

Bild 4. Dicke Datenbücher mit Datenblättern (Quelle: Ilse Joostens).

ist ein Kinderspiel, da es überall auf der Welt erschwingliche PCB-Services gibt. Früher musste man sich mit Abrubbel-Symbolen auf Pauspapier herumschlagen (**Bild 3**) oder, wenn man Glück hatte, mit einem unbezahlbaren Softwarepaket mit steiler Lernkurve. Dann hatte man immer noch keine Platine und war meist darauf angewiesen, selbst zu ätzen. Es wird niemanden überraschen, wenn ich berichte, dass meine Eltern vor mehr als 30 Jahren nicht sehr erfreut waren, als meine selbstgebaute Ätzschale plötzlich undicht wurde und ein brauner Strahl Eisen-III-Chlorid gegen die Tapete spritzte.

Das war auch die Zeit der dicken Datenbücher (**Bild 4**) und Vergleichstabellen, soweit man sie finden konnte. Heutzutage ist es eine Selbstverständlichkeit, Datenblätter aus dem Internet zu holen. Fortgeschrittene Entwicklungswerkzeuge und Evaluierungsboards mit einer Vielzahl von leistungsstarken Prozessoren, Sensoren, drahtlosen Modulen und anderen lustigen Dingen sind jetzt für jedermann zugänglich geworden. Was mich betrifft, müssen die Tage der Logikgatter und des Brennens von EPROMs unter der UV-Lampe nicht zurückkehren.

SG – 240319-02

WEBLINKS

[2] Eine kurze Geschichte der GPS-Navigation im Auto: https://t1p.de/3lnc7
[3] Verrückte Bilder auf Google Street View (CNET): https://t1p.de/uau4k
[4] Belgien nimmt definitiv Abschied vom Telegramm (VRT nws): https://t1p.de/ktvbs
[5] Swedish Government to Move Forward with Police FRT Amid Gang Violence: https://t1p.de/cujse
[6] KiCad EDA - Eine plattformübergreifende und quelloffene Elektronik-Design-Automatisierungssuite: https://kicad.org

KI-gestützter Wasserzähler

Teil 2: Firmware-Installation, Anwendung und Montage

Von Daniel Scaini (Italien)

Elettronica In
WWW.ELETTRONICAIN.IT

In diesem zweiten und letzten Teil des Artikels wird gezeigt, wie man die Firmware installiert, das Kameraobjektiv optimal fokussiert und auf dem Wasserzähler positioniert. Und natürlich befassen wir uns mit dem gesamten KI-basierten Prozess für die korrekte Erkennung und Auslesung der Elemente des Zählers.

Firmware-Installation

Zunächst müssen Sie die vorkompilierte Firmware „AI on the Edge Device" aus dem GitHub-Projekt herunterladen. An diesem recht umfangreichen, leistungsfähigen und gut dokumentierten Projekt arbeiten viele Entwickler mit. Auf der Seite *Releases* [1] sind mehrere Firmware-Versionen verfügbar, die jeweils von einem Changelog begleitet werden, einer Textdatei, in der Änderungen, neue Implementierungen oder Fehlerbehebungen aufgelistet sind. Anschließend laden wir die *.zip*-Datei mit der vorkompilierten, zu installierenden Firmware herunter, in der sich drei Dateien mit der Erweiterung *.bin* befinden.
Außerdem gibt es eine weitere *.zip*-Datei, die das Material enthält, das auf der SD-Karte gespeichert werden soll. Die SD-Karte sollte eine Kapazität von 16 GB nicht überschreiten, weil bei Karten mit höherer Kapazität, zum Beispiel 32 GB oder 64 GB, Lesefehler auftreten können. Auf Software-Ebene darf die SD-Karte keine Partitionstabelle haben und muss im FAT32-Format formatiert sein. Wenn unser ESP32-CAM-Modul beim Start Probleme haben oder Fehlfunktionen aufweisen sollte, wäre es ratsam, die SD-Karte als Hauptverdächtige sofort auszutauschen, bevor man sich auf eine weitere Fehlersuche begibt.
Nachdem die heruntergeladene Datei entpackt ist, sieht man mehrere Ordner, von denen der Ordner *config* von besonderem Interesse für uns ist, weil er die TensorFlow-Dateien und die Konfigurationsdatei *config.ini* enthält. In der Datei *config.ini* sind alle Starteinstellungen zu finden, die für diejenigen, die die Firmware nicht direkt kompiliert haben, schwierig zu lesen oder zu interpretieren sein könnten.

In der umfangreichen Liste der Parameter ist der Eintrag `SetupMode = true` von besonderem Interesse. Wir ändern den Wert dieses Parameters von `true` auf `false`, um einen initialen Setup-Modus zu deaktivieren. Dieser Parameter ist dafür verantwortlich, was wir in unserem Browser sehen, wenn wir das Gerät zum ersten Mal aufrufen. Würden wir den Wert `true` beibehalten, erscheint ein Konfigurationsassistent, der möglicherweise unnötig ist und Probleme beim Schreiben auf die SD-Karte verursachen kann. Wenn Sie den Wert des Parameters auf `false` ändern, erhalten Sie direkten Zugriff auf das Hauptfenster, mit dem Sie sich vertraut machen müssen.
Es gibt eine zweite Datei, die wir bearbeiten müssen, um den ersten Start zu erleichtern, nämlich *wlan.ini*. Darin muss die SSID und das Passwort für Ihr WLAN eingegeben werden. Ohne diese Informationen würde der Chip im AP-Modus (als Access Point) starten und den Clients sein Netzwerk anzeigen. Ist dies geschehen, können wir mit der Installation der eigentlichen Firmware über das *Flash Tool* von Espressif oder über die Konsole mit *esptool* fortfahren.
Um mit dem ersten und anschaulicheren Verfahren zu beginnen, müssen zunächst *Flash Download Tools* unter [2] heruntergeladen werden. Wir entpacken die Software in einen Ordner und führen das darin enthaltene *.exe*-Programm aus. Wir wählen im Programm *ESP32* und *Develop* (**Bild 1**), um die *.bin*-Datei zu programmieren. Dabei wählen wir die Dateien in der unten beschriebenen Reihenfolge aus und geben daneben die jeweiligen Indizes ein:

- *bootloader.bin* ⇒ 0x1000
- *partitions.bin* ⇒ 0x8000
- *firmware.bin* ⇒ 0x10000

Die SPI-Geschwindigkeit (*SPEED*) sollte auf 40 MHz eingestellt werden, der Modus (*MODE*) auf *DIO* (**Bild 2**). Wir können nun das Programmiergerät anschließen und *START* drücken. Die Firmware wird nach einigen Augenblicken korrekt geladen.

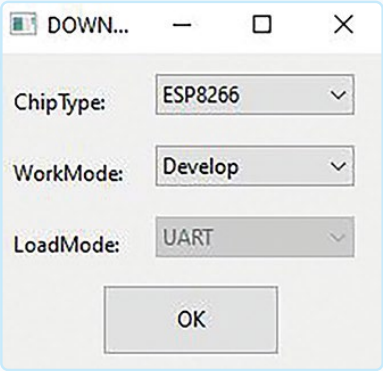


Bild 1. Auswahl von ESP32 und Develop zum Laden der .bin-Datei in den ESP32-CAM.

Die zweite Variante, die Firmware zu laden, erfordert ein Terminal und eine Python-Umgebung. Man könnte zum Beispiel Anaconda [3] verwenden, aber wir empfehlen dieses Verfahren vor allem erfahrenen Ubuntu-Nutzern. Als ersten Schritt installieren wir das *esptool*. Dazu müssen wir den folgenden Befehl in die Kommandozeile eingeben:

```
pip install esptool
```

Falls Sie Python 3 verwenden, lautet der Befehl:

```
pip3 install esptool
```

Jetzt schließen Sie den Programmer an und geben ein:

```
esptool erase_flash
esptool write_flash 0x01000 bootloader.bin 0x08000 parti-
tions.bin 0x10000 firmware.bin
```

Als erfahrener Benutzer werden Sie natürlich merken, ob der COM automatisch vergeben wird oder nicht. Wenn nicht, kann es nämlich schon beim ersten Befehl zu Problemen kommen. Nachdem die Firmware erfolgreich auf den ESP32 geladen wurde, kann das USB-Kabel vom PC abgezogen und alle zuvor vorgenommenen Verkabelungen entfernt werden, so dass nur noch die Spannungsversorgung 5 V und GND an unserem ESP32-CAM übrig bleiben.
Nach dem Neustart des Moduls oder dem Anschluss an eine externe Stromversorgung muss man beim ersten Booten der Firmware prüfen, ob die Datei *wlan.ini* verfügbar ist und gültige Anmeldeinformationen enthält. Wenn diese Informationen fehlen oder ungültig sind, startet das Modul (wie gesagt) im AP-Modus, ohne Passwort oder Schutz für das WLAN: Ein neues WLAN namens *AI-on-the-Edge* wird erstellt.
Es braucht nur wenige Schritte, um die Datei direkt über die Webseite des Moduls hochzuladen. Aber gehen wir der Reihe nach vor: Der erste Schritt besteht darin, die Remote-Setup-Datei herunterzuladen, die immer in den Projektversionen hier [1] verfügbar ist. Wenn wir uns dann mit dem WLAN *AI-on-the-Edge* verbinden, wird eine Seite angezeigt, die uns auffordert, eine Datei hochzuladen.
Wir laden die soeben heruntergeladene Datei hoch und warten, wie auf der Seite angegeben. Nach einem Upload, der bis zu 60 Sekunden dauern kann, erscheint der Screen zur Eingabe der Zugangsdaten für das heimische WLAN. Nachdem SSID und Passwort eingegeben wurden, klickt man auf *Write wlan.ini*. Wenn alles gut gegangen ist, wird zum Neustart aufgefordert.
An Bord des ESP32 befindet sich eine rote LED, deren Blinken mehrere Bedeutungen hat:

- 5× schnelles Blinken (< 1 Sekunde): Verbindung wird aufgebaut
- 3× langsames Blinken (1 Sekunde an/aus): WLAN-Verbindung hergestellt

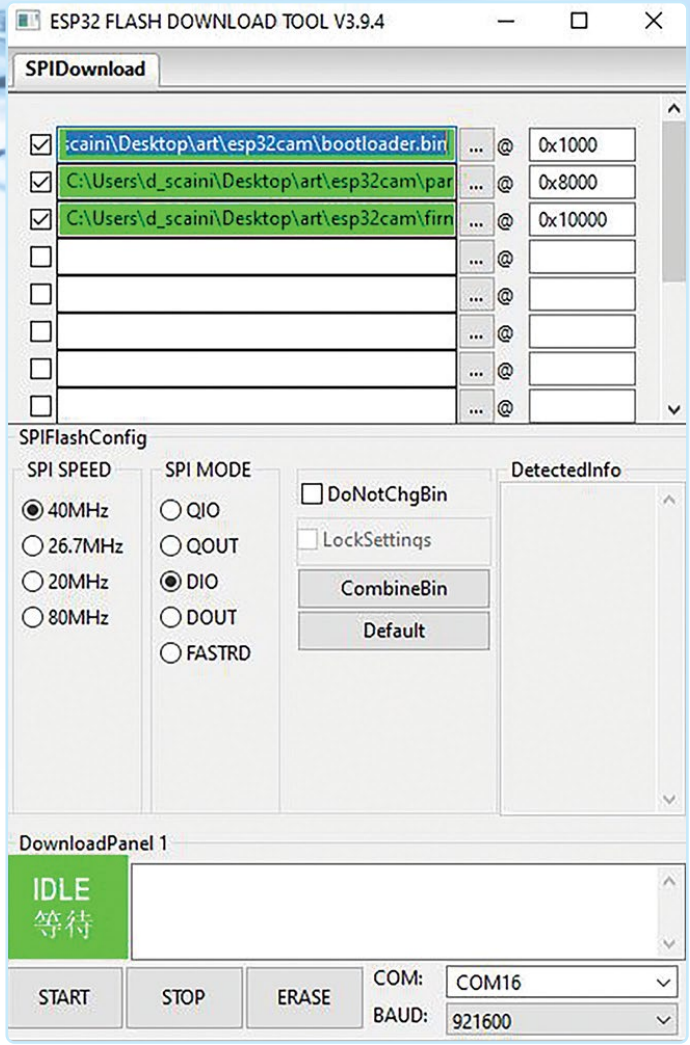


Bild 2. Im Bereich SPI SPEED wird die Geschwindigkeit auf 40 MHz, der SP MODE auf DIO gesetzt.

Einstellen der Kamera-Optik

Ein wichtiger Schritt nach dem Laden der Software in das Modul ist die Einstellung der Kameraoptik auf dem ESP32-CAM-Modul. Standardmäßig ist der Fokus für sehr große und nicht für kurze Entfernungen eingestellt, wie es in unserem Fall nötig wäre. Bei diesem Schritt ist es wichtig, sehr behutsam vorzugehen, wenn man versucht, die Außenseite der Linse mit einer Pinzette zu drehen und, falls nötig, den Kleber am Fuß der Linse mit einem Cutter zu entfernen. Bei diesen Anpassungsarbeiten (**Bild 3**) ist große Vorsicht geboten!

Positionierung des Geräts

Nach der Installation der Firmware und der Einstellung des Kamerafokus können wir unser Gerät auf dem zu überwachenden Wasserzähler anbringen. Die Kamera soll mechanisch fest sitzen und einen ausreichenden Abstand wahren, damit die Daten korrekt gelesen werden. Um dies zu ermöglichen, stehen mehrere Lösungen offen. Die beste wäre zweifellos der 3D-Druck einer Kunststoffhalterung für unser Gerät, die es geschützt und unbeweglich in einem kleinen Gehäuse hält, das fest mit dem Wasserzähler verankert ist. Ein Beispiel aus Thingiverse kann unter [4] heruntergeladen werden. Auf dieser Seite finden Sie die STL-Datei für das Gehäuse des Kammermoduls (**Bild 4**) und den Abstandshalter, mit dem Sie die Elektronik sicher am Messgerät befestigen können, wie in **Bild 5** dargestellt.

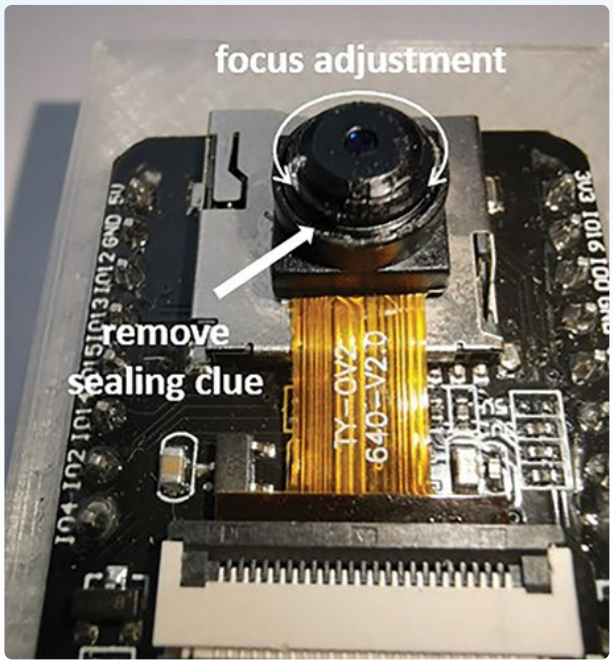


Bild 3. Einstellen der Optik für Nahaufnahmen.

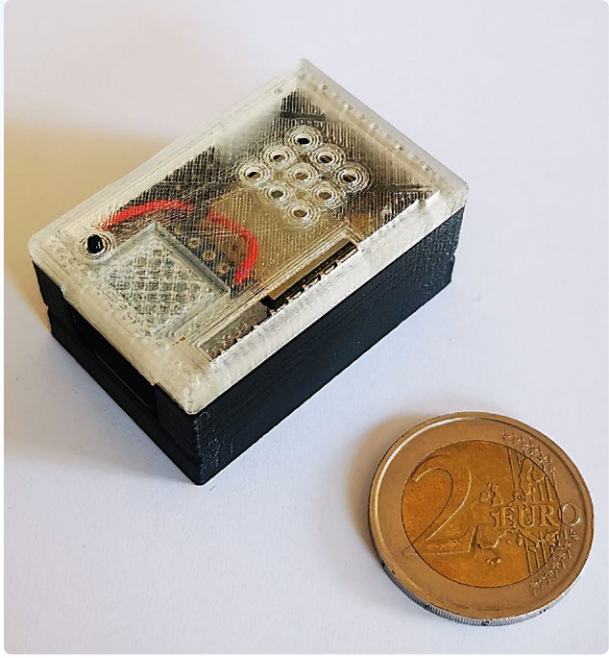


Bild 4. Das ESP32-CAM-Modul in seinem 3D-gedruckten Gehäuse. Die 2-Euro-Münze ist nicht im Lieferumfang enthalten!

Eine einfachere und schnellere Alternative ist es, einen Polyamid- oder Polystyrol-Würfel (Styropor) zu verwenden, wie er üblicherweise für Verpackungen verwendet wird. Wir platzieren das Modul auf der Außenseite des Würfels, mit dem Blick nach unten, und schneiden ein Loch, durch das die Kameralinse und der Blitz hindurchpassen. Das Loch in der Mitte hat dann genau die richtige Größe für die Anzeige des Wasserzählers. Wenn dieser wie auf dem Foto eine Abdeckung hat, kann man die Halterung weiter ausschneiden, damit sie perfekt passt.

Bild 5. Ein 3D-gedruckter Adapter verbindet den ESP32-CAM mit dem Zähler.

Erste Einstellungen und Erkennung

Bevor wir unser Gerät einschalten, nehmen wir die SD-Karte heraus und stecken sie in den PC. Um zu überprüfen, ob alle Schritte sorgfältig befolgt wurden und ob die Firmware-Installation erfolgreich war, sehen wir uns auf der SD-Karte die Datei *config.ini* im Ordner *config* an. Am Ende dieser Datei findet sich die Einstellung `SetupMode`, die auf `false` gesetzt werden muss. Dieser Schritt ermöglicht uns den direkten Zugriff auf den Haupt-Screen und nicht auf den des Einrichtungsassistenten. Die SD-Karte wird wieder in das Modul gesteckt und dieses dann eingeschaltet. Im Browser des PCs verbinden wir uns mit der IP-Adresse, die der Router dem ESP32 zugewiesen hat, in diesem Fall *192.168.43.102*. Es erscheint das in **Bild 6** gezeigte Bild. Die Messungen und die Identifizierung von Rechtecken oder Fotos lassen wir zunächst links liegen und konzentrieren uns auf den Menüteil in der schwarzen Leiste am oberen Rand. Unter *Configuration* öffnet sich ein Dropdown-Menü mit vier Punkten: *Set Previous Value*, *Configuration*, *Alignment*, *Region of Interest*.

Zunächst zum dritten Punkt: Hier finden wir die beiden Unterpunkte *Reference Image* und *Alignment Marker*. Klicken wir auf *Reference Image:* In diesem Abschnitt kann das erste Bild des Zählers erfasst und als Referenz für alle nachfolgenden Checks verwendet werden. Dazu klicken man auf *Create New Reference Image* und dann auf *Take Reference*, wodurch ein Befehl zur Aufnahme des Bildes gesendet wird. Oben in diesem Menü kann man die LED-Intensität, den Drehwinkel des Bildes, den Kontrast, die Sättigung und viele andere Werte einstellen, um die Anzeige des ausgewählten Bildausschnitts zu optimieren. Sobald die erste Aufnahme abgeschlossen ist, speichert man sie durch *Save* und dann, nach der Meldung einer erfolgreichen Speicherung, auf *Restart*, um die gespeicherte Referenz zu aktivieren. In diesem Stadium ist es wichtig, dass die Zählernummern innerhalb des Bildes korrekt zueinander ausgerichtet sind.

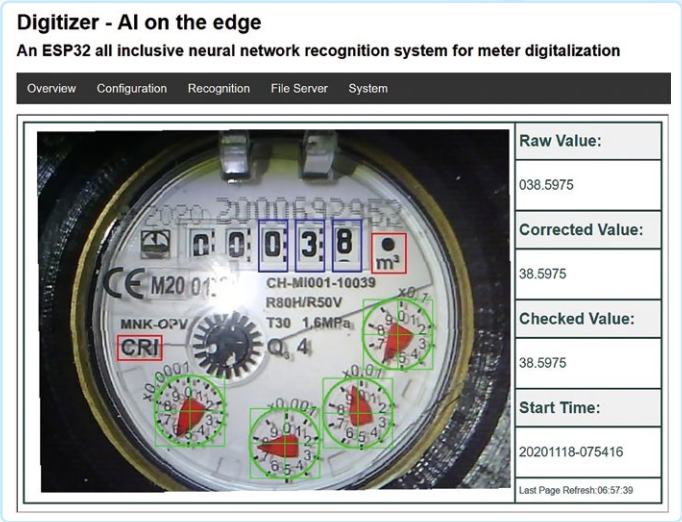


Bild 6. Die interne Webseite, über die das System konfiguriert werden kann.

Nach dem Neustart gehen wir zum nächsten Abschnitt, ebenfalls unter *Settings* ⇒ *Alignment* ⇒ *Alignment Marks* (**Bild 7**). Hier müssen wir dafür sorgen, dass das System die genaue Position von zwei Referenzmarken erkennt, die aus der Dropdown-Liste *Select Reference* ausgewählt werden. Die „0" enthält die CRI-Anzeige, die nicht bei allen Messgeräten vorhanden ist (**Bild 8**). Die „1" ist die Maßeinheit Kubikmeter. Nachdem Sie das Feld richtig verschoben haben, um die Markierungen zu zentrieren, klicken Sie auf *Update Reference Image* und dann auf *Save*, ohne das Gerät neu zu starten, denn zunächst müssen alle Referenzen festgelegt werden.
Aus dem Dropdown-Menü wählt man dazu *Settings* und dann *Region Of Interest*, woraufhin zwei weitere Unterrubriken erscheinen: *Digital ROIs* und *Analog ROIs*. Die erste führt zu der Anzeige in **Bild 9**, in der alle Ziffern der Anzeige mit den Werkzeugen am unteren Rand ausgerichtet werden. Einige der Ziffernbezeichnungen sind bereits

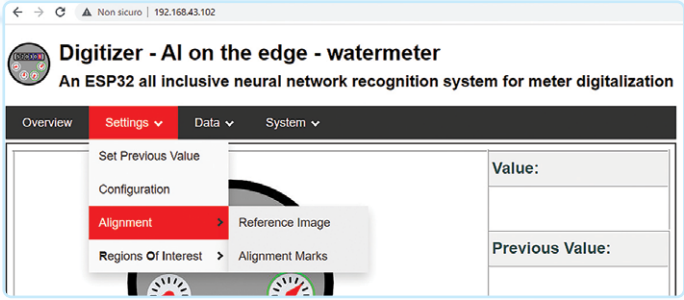


Bild 7. Im Menü „Alignment Marks" (Ausrichtungsmarkierungen) können die Referenzmarkierungen ausgewählt werden.

im Dropdown-Menü unter *New ROI* festgelegt, aber Sie müssen für jede Ziffer noch einen X- und Y-Wert eingeben. Ändern Sie nicht die Checkboxen an der Seite, da - wenn das Foto korrekt aufgenommen wurde - die ROIs alle gleich ausgerichtet und gleich groß sind.
Bei jeder ROI gibt es zwei rote Rahmen: Der innere Rahmen muss die gleiche Größe wie die Ziffer haben. Sobald dieser Schritt abgeschlossen ist, klicken Sie auf *Save* und fahren mit dem letzten Schritt fort. Im Abschnitt *Analog ROIs* mit den analogen Zeigeranzeigen müssen Sie ähnlich wie bei den digitalen ROIs vorgehen (**Bild 10**). Der runde, rote Rand der Identifikationsmaske muss genau mit dem runden Rand der Zeigeranzeige übereinstimmen. Auch hier können die verschiedenen Checkboxen so belassen werden, wenn das Foto gut ist, da alle Zeigeranzeigen die gleiche Größe haben. Danach klickt man erneut auf *Save* und schließlich auf *Reboot*, um die Änderungen zu aktivieren.

REST-Konfiguration, MQTT und Home Assistant

Nun, da das ESP32-CAM-Modul den Zähler lesen kann, brauchen wir noch eine bequeme Möglichkeit zur Anzeige der Daten. Glücklicherweise ermöglicht der Code den Zugriff auf die Daten auf mehrere Arten. In der vorkompilierten Firmware befindet sich ein http-Server, der bereits bei der Anpassung der ersten Einstellungen genutzt wurde.

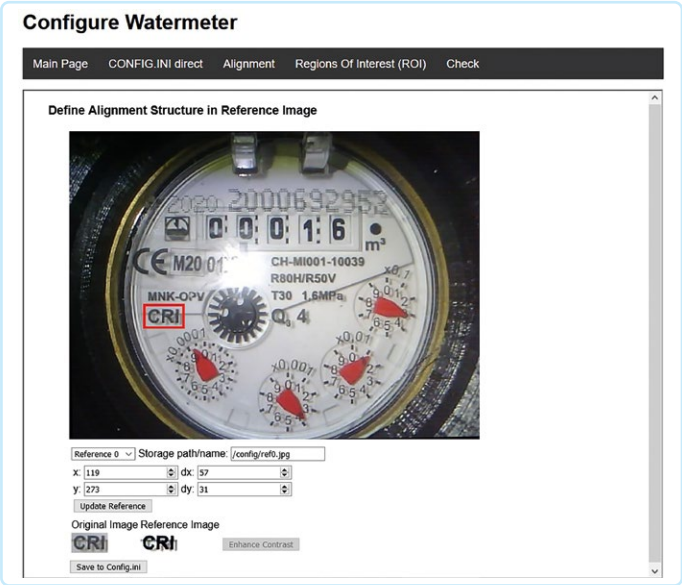


Bild 8. Die CRI-Marke ist nicht auf allen den Wasserzählern vorhanden.

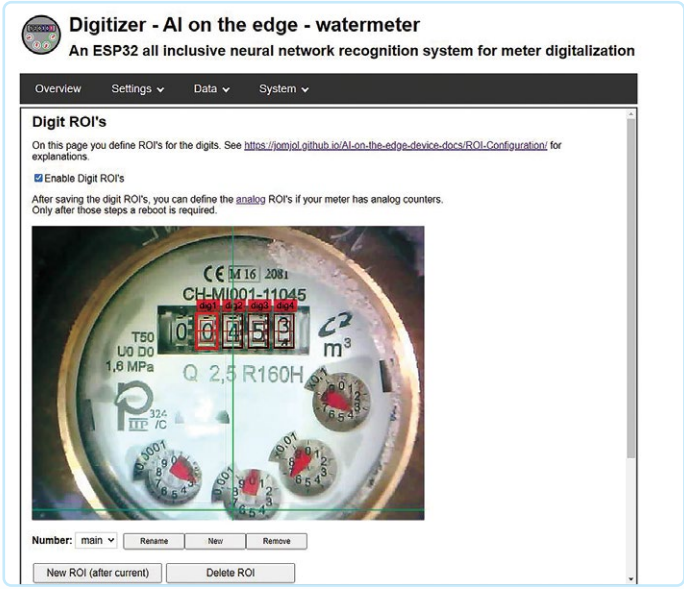


Bild 9. Markierung der Ziffernanzeigen.

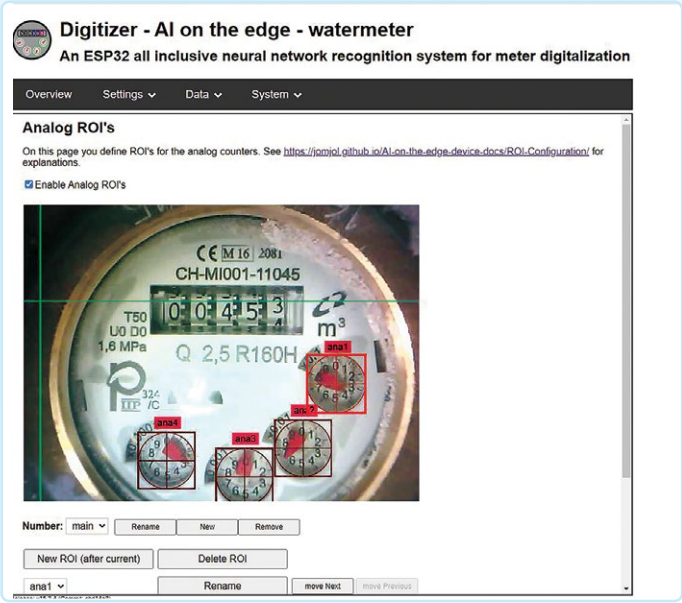


Bild 10. Kennzeichnung der analogen Zeigeranzeigen.

Dieser Server stellt uns nicht nur Webseiten, sondern auch REST-APIs zur Verfügung, die vollständig über die GET-Methode genutzt werden können. Die Besonderheit der GET-Methode liegt in ihrer Einfachheit: Alle Informationen sind in der URL und im Klartext enthalten. Der feste Bestandteil dieser Aufrufe ist der Einstiegspunkt *http://IP_DEVICE*, zu dem je nach Verwendungszweck einer der variablen Bestandteile hinzugefügt werden muss.

In **Tabelle 1** finden Sie die möglichen Befehle und ihre Bedeutung. All diese Befehle sind im selben Netzwerk wirklich sehr praktisch, aber was passiert, wenn wir uns außerhalb unseres Netzwerks befinden? Die Integration des MQTT-Dienstes beantwortet diese Frage. **M**essage **Q**ueue **T**elemetry **T**ransport ist ein sehr leichtgewichtiges Publish-Subscribe-Nachrichtenprotokoll und wird dort eingesetzt, wo die Bandbreite begrenzt ist oder wo man nur geringe Auswirkungen auf die Netzwerkebene haben möchte. Ein Blick in die Literatur zeigt, dass dieses im Jahr 1999 von Andy Stanford-Clark und Arlen Nipper entwickelte Protokoll in erster Linie für die Überwachung von Ölpipelines entwickelt wurde.

Ziel war es, ein wirklich leichtgewichtiges, bandbreiteneffizientes Protokoll zu schaffen, das wenig (Batterie-) Energie verbraucht, ein damals wie heute heikles Thema. Das Protokoll wurde 2013 in der Version 3.1 von IBM wiederbelebt und begann sich zu etablieren, auch dank eines starken Marktimpulses des Internets der Dinge. Eines der bekanntesten Beispiele für den Einsatz von MQTT ist sicherlich der Facebook-Messenger: Facebook hat dieses Protokoll nach eigenen Angaben eingeführt, um die Nachrichtenübermittlung zu verbessern und den Netzwerk- und Energiebedarf zu verringern.

Der Client ist ein beliebiges Gerät, von einem einfachen Mikrocontroller bis hin zu einem vollständigen Server, auf dem eine MQTT-Bibliothek läuft und der mit einem Makler (*broker*) verbunden ist. Dieser Client sendet Informationen, so genannte Nutzlasten (*payloads*), die nach Themen (*topics*) klassifiziert sind, an den Broker. Wenn ein Client eine neue Nutzlast an einen Broker sendet, leitet dieser die Information an alle mit ihm verbundenen Clients zu einem bestimmten Thema weiter.

Tabelle 1: Befehle, die über den Einstiegspunkt http://IP_DEVICE erteilt werden können

Pfad	Funktion	Erweiterung/Antwort

PATH_FISSO/GPIO	Bietet eine Schnittstelle zum Setzen oder Lesen des Status eines PINs des ESP	Status setzen
		`?GPIO=&Status=high or low`
		
		Status lesen
		`?GPIO=`

PATH_FISSO/reboot	Neustrat des Geräts	

PATH_FISSO/json	Um die Ergebnisse des letzten Lesevorgangs im JSON-Format zu erhalten	Beispielantwort:
		`{`
		`"main":`
		`{`
		`"value": "521.17108",`
		`"raw": "521.17108",`
		`"pre": "521.17108",`
		`"error": "no error",`
		`"rate": "0.023780",`
		`“timestamp”: “2023-01-13T16:00:42+0100”`
		`}`
		`}`

Pfad	Funktion	Erweiterung/Antwort

PATH_FISSO/value	Um die Ergebnisse als einen einzelnen Wert und nicht im JSON-Format zu erhalten	Um einen anderen Wert als den Standardwert zu extrahieren, können wir die folgenden GET-Parameter verwenden
		`?all=true&type=value`
		`?all=true&type=raw`
		`?all=true&type=error`

PATH_FISSO/img_tmp/raw.jpg	So erhalten Sie ein Rohbild des Zählers	
PATH_FISSO/img_tmp/alg_roi.jpg	Zur Gewinnung eines Rohbildes mit den verschiedenen Regions Of Interest (ROI) im Overlay	
PATH_FISSO/statusflow	Zeigt genau, welcher aktive Prozess auf dem Gerät momentan aktiv ist, einschließlich eines Zeitstempels	Ein Beispiel für eine Antwort könnte sein: `Take Image (15:56:34)`
PATH_FISSO/rssi	Zeigt die Stärke des empfangenen WLAN-Signals an (in dBm)	Eine Antwort könnte beispielsweise lauten: `-51`
PATH_FISSO/cpu_temperature	Zeigt die Temperatur der CPU (in °C)	
PATH_FISSO/sysinfo	Zeigt alle Informationen des Systems im JSON-Format	Ein Beispiel für eine Antwort könnte sein: `[{"firmware": "", "buildtime": "2023-01-25 12:41", "gitbranch": "HEAD", "gittag": "", "gitrevision": "af13c68+", "html": "development-branch: HEAD (Commit: af13c68+)", "cputemp": "64", "hostname": "WaterMeterTest", "IPv4": “192.168.43.102”,”freeHeapMem”: “2818330”}]`
PATH_FISSO/startime	Zeigt das „Inbetriebnahmedatum“ an	Ein Beispiel für eine Antwort könnte sein: `20230113-154634`
PATH_FISSO/uptime	Zeigt an, wie lange das System bereits aktiv ist	Eine Beispielantwort könnte lauten: `0d 00h 15m 50s`
PATH_FISSO/lighton	Ändert den Blitzstatus der Kamera auf AN	
PATH_FISSO/lightoff	Ändert den Blitzstatus der Kamera auf AUS	
PATH_FISSO/capture	Nimmt ein neues Bild auf (ohne Blitzlicht)	
PATH_FISSO/capture_with_flashlight	Nimmt ein neues Bild auf (mit Blitzlicht)	
PATH_FISSO/save	Speichert ein neues Bild auf der SD-Karte	Außerdem gibt es folgende Parameter: ?filename=test. `?filename=test.jpg&delay=1000`
PATH_FISSO/log	Zeigt nur den letzten Teil des Logs des Tages (nur die letzten 80 kBytes)	
PATH_FISSO/logfileact	Zeigt alle Logs des Tages an	

Kein Client hat eine Ahnung von der Anzahl oder den Positionen der Abonnenten (*subscribers*), die mit diesem *topic* verbunden sind. In unserem Fall ist der Client der Chip selbst, während anstelle des Brokers eine Hass.io/Home Assistant-Installation verwendet werden könnte. Um diesen Dienst zu aktivieren und zu integrieren, geht man zu *Settings* ⇒ *Configuration*. Unten auf der Seite findet sich der Abschnitt zur Mqtt-Konfiguration, den man wie in **Bild 11** bearbeiten kann. Unter URIs muss *mqtt://_BROKER ADDRESS:PORT* (normalerweise 1883) und, falls erforderlich, Benutzername und Passwort eingetragen werden. Danach gehen wir zum Dashboard des Home Assistant und fügen dort unter *Settings* ⇒ *People* einen neuen Nutzer hinzu, der sich mit denselben Anmeldedaten anmelden kann. Im Home-Assistant-Dashboard geht man zu *File Editor*, einem sehr empfehlenswerten Plugin, das die Bearbeitung von Konfigurationsdateien direkt von der Weboberfläche aus ermöglicht. Hier findet sich die Datei */config/configuration.yaml*, zu der folgenden Zeilen hinzugefügt werden:

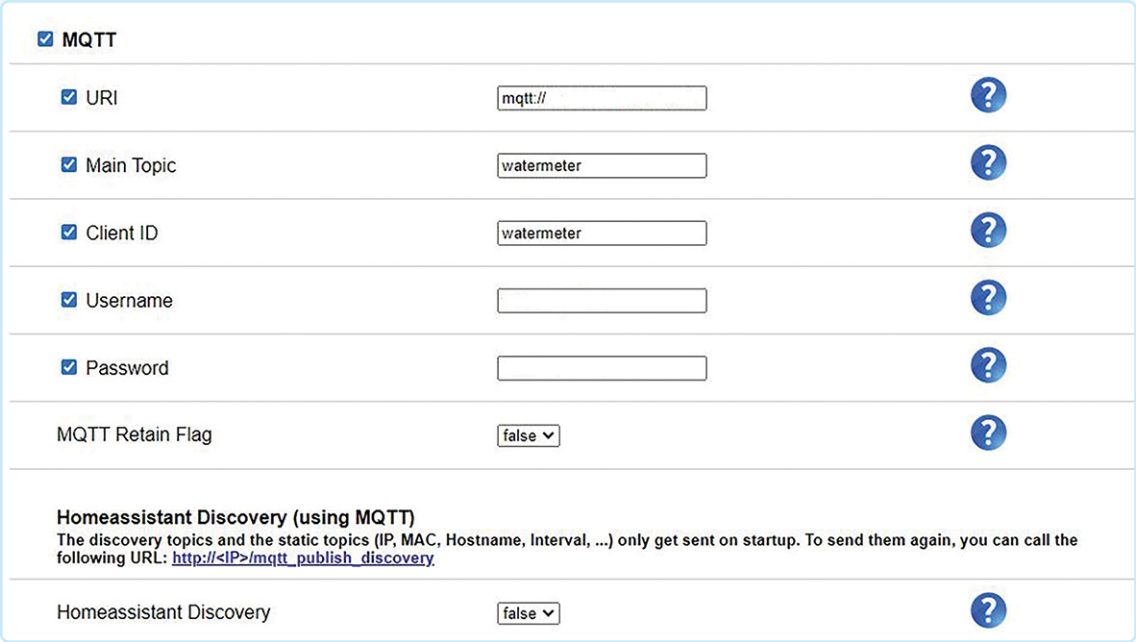


Bild 11. MQTT-Konfigurationsmenü.

```
sensor:
  - platform: mqtt
    name: "waterMeterMQTT"
    state_topic: "watermeter/main/raw"
    unique_id: watermeter_value
    unit_of_measurement: 'm³'
    state_class: total_increasing
    device_class: water # Needs HA 2022.11!
    icon: 'mdi:water-pump'
    availability_topic: watermeter/connection
    payload_available: connected
    payload_not_available: connection lost
```

Bei einer älteren Version des Home Assistant kann es sein, dass die Entität „water" fehlt und Sie stattdessen einfach „power" verwenden müssen/können. Anstelle von „raw" lässt sich auch „value" einsetzen, was die Informationen bereits verarbeitet und nicht in roher Form liefert. Nun müssen Sie den Server neu starten. Gehen Sie zu *Settings* ⇒ *Server Controls* ⇒ *restart*, um die neue Entität anzuzeigen. Sobald die Verbindung mit dem Server wiederhergestellt ist, klicken Sie auf *Edit dashboard* und fügen unseren Sensor wie in **Bild 12** dargestellt hinzu. In **Tabelle 2** finden Sie eine Liste aller zu verwendenden Topics (wobei davon auszugehen ist, dass bei der Einrichtung das Haupttopic als *watermeter* benannt wurde).

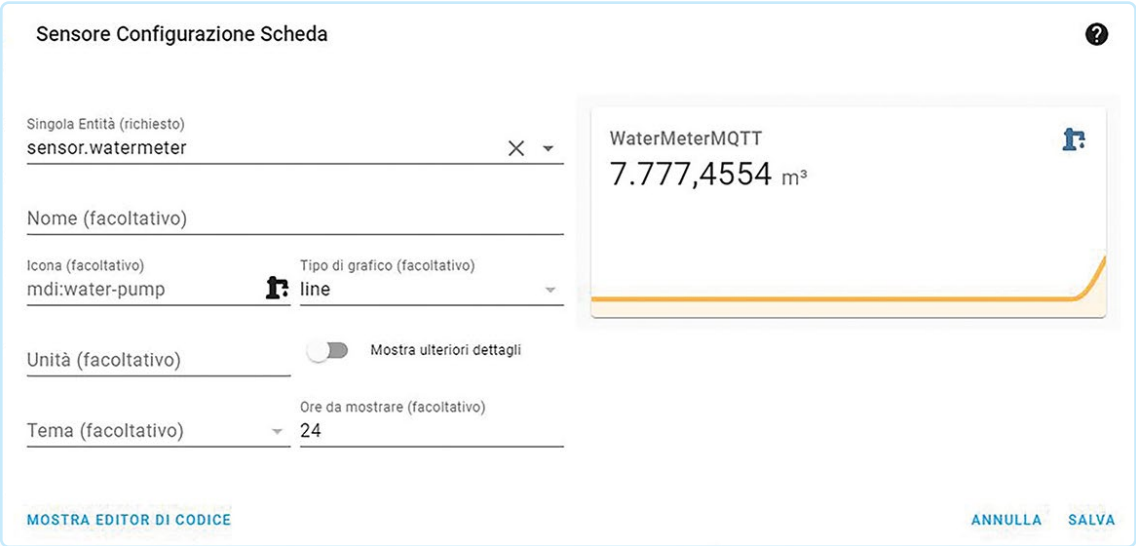


Bild 12. Sensor-Konfigurationsfenster.

Tabelle 2: Topics zur Verwendung mit MQTT

Topic	Beschreibung
watermeter/MAC	MAC-Adresse des ESP abrufen
watermeter/IP	IP-Adresse des ESP abrufen
watermeter/Hostname	Hostnamen abrufen
watermeter/Interval	Zeitintervall abrufen, in dem die Bilder gemacht werden
watermeter/Connection	Status der Verbindung abrufen
watermeter/Uptime	Aktivität des Chips seit dem letzten Neustart
watermeter/FreeMem	Freier Speicherplatz (in KB)
watermeter/wifiRSSI	Signalqualität
watermeter/CPUTemp	CPU-Temperatur
watermeter/main/error	Status von Stream-Fehlern. Wenn nicht vorhanden, ist die Antwort "No error" (main hängt davon ab, was gesetzt wurde).
watermeter/main/raw	Liefert den Wert des Messwerts vor der Nachbearbeitung
watermeter/main/rate	Wert des Wasserverbrauchs pro Minute
watermeter/main/rate_per_time_unit	Wert des Wasserverbrauchs pro Stunde
watermeter/main/changeabsolut	Differenz zwischen vorherigem und aktuellem Messwert
watermeter/main/timestamp	Zeitmarke des ESP
watermeter/main/Status	Status der Plattform
watermeter/main/json	Gibt eine Reihe von Werten im json-Format aus, einschließlich value, raw, Fehler, Rate, Zeitstempel, etc.
watermeter/GPIO/GPIO	Status eines GPIO bestätigen und lesen
watermeter/main/value	Messwerts nach dem Erfassungsprozess

Entwicklung und Modifikationen

Sie können die Quellen auch modifizieren, um zum Beispiel das Verhalten des Webservers zu ändern. Als ersten Schritt müssen Sie dazu einen GIT-Client installieren. Für diejenigen, die nicht wissen, wovon wir sprechen: GIT ist ein sehr weit verbreitetes Versionsverwaltungssystem. So klonen Sie das Projekt aus der entfernten Quelle:

```
git clone https://github.com/jomjol/AI-on-the-edge-
    device.git
cd AI-on-the-edge-device
git checkout rolling
git submodule update --init
```

Nachdem Sie den Client zum Extrahieren des Repositoriums installiert haben, empfehlen wir auch die Installation einer integrierten Entwicklungsumgebung (IDE), um einen vollständigen Überblick über das Projekt zu erhalten. Als IDE bietet sich Visual Studio Code an, das ein wichtiges Plugin namens *PlatformIO* bietet. Zunächst lädt und installiert man dazu Visual Studio Code von der entsprechenden Website [5].

Anschließend öffnet man die Erweiterungen (*Extensions*) in Visual Studio Code und sucht nach *platformio*. Das Plugin wird durch ein Klick auf den dazugehörenden Button automatisch installiert.
In unserer Entwicklungsumgebung öffnet man den Quellcode im Ordner *AI-on-the-edge-device/code*, öffnet dann durch einen Klick auf das Erweiterungssymbol und *New terminal* ein PlatformIO-Terminal und gibt zum Kompilieren des Codes `platformio run --environment esp32cam` ein. Im Ordner *AI-on-the-edge-device/code/.pio/build/esp32cam/* finden man dann die drei kompilierten Bineries *bootloader.bin*, *partitions.bin* und *firmware.bin*. Der gesamte Code ist in der Sprache C++ geschrieben.

Zukünftige Entwicklungen

Der Code ist gut dokumentiert und strukturiert und wird von den Entwicklern ständig gewartet, ergänzt, korrigiert und verbessert. Eine mögliche zukünftige Entwicklung könnte darin bestehen, den abgelesenen Wert am Ende eines jeden Monats an das eigene Smartphone zu senden, zum Beispiel per E-Mail oder über soziale Plattformen. Eine weitere Idee zur Verbesserung des Projekts könnte die Integration von BLE (Bluetooth Low Energy) sein. Dies würde es ermöglichen, eine Plattform zu schaffen, auf der das Ablesen über das Smartphone ohne zusätzliche externe Geräte erfolgen kann.

RG – 240213-B-02

Sie haben Fragen oder Kommentare?

Haben Sie technische Fragen oder Anmerkungen zu diesem Artikel? Schreiben Sie bitte an die Elektor-Redaktion unter redaktion@elektor.de.

Passende Produkte

- **ESP32-Cam-CH340 Entwicklungsboard**
  www.elektor.de/19333
- **FTDI Serielles TTL-RS232-USB-Kabel**
  www.elektor.de/20173

WEBLINKS

[1] GitHub-Projekt, „AI on the Edge Device", Release-Seite: https://tinyurl.com/2n8da7jj
[2] Espressif Flash-Tools: https://espressif.com/en/support/download/other-tools
[3] Anaconda Downloadseite: https://anaconda.com/download
[4] 3D-druckbare Gehäuse bei Thingiverse: https://thingiverse.com/thing:4573481
[5] Visual Studio Code Download-Webseite: https://code.visualstudio.com/Download
[6] Erster Teil dieses Artikels: https://elektormagazine.de/240213-02

Intelligente Landwirtschaft

ML-basierte Schädlingserkennung mit IoT-Vernetzung

Von Rafik Mitry (Mouser) und Shashank Hedge (Würth Elektronik eiSos)

Die Kombination von ML-basierten Bildverarbeitungssystemen am „Edge“ mit drahtloser Internetanbindung bietet völlig neue Perspektiven für die Landwirtschaft. Dieser Artikel zeigt, wie das IoT zur Implementierung eines Schädlingserkennungssystems genutzt werden kann.

Bild 1. Das Entwicklungsboard Thyone-I-Wireless-FeatherWing von Würth Elektronik verfügt über ein proprietäres 2,4-GHz-HF-Modul. (Quelle: Würth Elektronik eiSos)

Im Bereich der intelligenten Landwirtschaft (Smart Agriculture) stützte sich die Schädlingserkennung traditionell auf arbeitsintensive Methoden, was dazu führte, dass Ernten anfällig für Schäden waren und Landwirte mit unvorhersehbaren Erträgen konfrontiert wurden. Der zunehmende Einsatz von Mobilfunk-gestützter IoT-Vernetzung und Edge-Computing treibt die Entwicklung voran. Ähnlich wie bei der vorausschauenden Wartung in der Industrie ermöglicht das mobile Internet der Dinge (IoT) die Echtzeitüberwachung des Zustands von Nutzpflanzen mithilfe von Sensorsystemen. Darüber hinaus ist es mit maschineller Bildverarbeitung (Machine Learning, ML) auf Edge-Geräten nun möglich, Daten lokal zu analysieren, was batteriebetriebene Geräte mit Echtzeit-Überwachungsfunktionen ermöglicht. Dies ermöglicht ein rechtzeitiges Eingreifen, um Schädlingsbefall zu verhindern, die landwirtschaftliche Produktivität zu maximieren und Verluste zu minimieren. In diesem Artikel wird untersucht, wie das mobile Internet der Dinge und Edge-Computing die Schädlingserkennung von der Datenerfassung bis zur prädiktiven Analyse verändern und damit eine neue Ära der Präzisionslandwirtschaft einläuten.

Prototyping-Elemente

Die folgenden Elemente werden zur Realisierung eines Prototyps für die Schädlingserkennung verwendet:

- Arduino Nicla Vision [1]
- Würth Elektronik Thyone-I Feather-Wing [2]
- Würth Elektronik Adrastea-I Feather-Wing [3]
- Adafruit Feather M0 Express [4]

Software und Tools:

- Arduino-IDE
- OpenMV [5]
- Edge Impulse [6]
- Visual Studio Code mit PlatformIO-Erweiterung [7]

Der Thyone-I Wireless FeatherWing von Würth Elektronik ist mit einem proprietären 2,4-GHz-HF-Modul ausgestattet und

Bild 2. Der Adrastea-I-FeatherWing von Würth Elektronik bietet die Modulfunkstandards LTE-M / NB-IoT, externes LTE und Satellitennavigation (GNSS). (Quelle: Würth Elektronik eiSos)

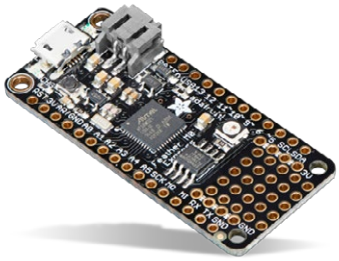


Bild 3. Das MCU-Board Adafruit-Feather-M0-Express basiert auf dem ATSAMD21G18 Arm Cortex-M0 Prozessor und ist über Mouser erhältlich. (Quelle: Adafruit)

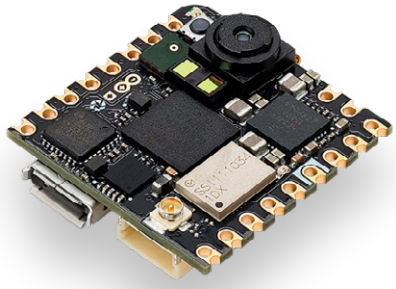


Bild 4. Das Bildverarbeitungssystem Arduino Nicla Vision ist ausgestattet mit einem robusten STM32H747AII6 Dual Arm Cortex M7/M4-Prozessor sowie einer 2-MP-Farbkamera, die TinyML unterstützt. (Quelle: Mouser Electronics)

bietet eine nahtlose drahtlose Verbindung mit einer beeindruckenden Reichweite von bis zu 300 Metern (**Bild 1**). Die intergierte AES128-Verschlüsselung und die Mesh-Netzwerkfähigkeit sorgen für robuste Sicherheit und zuverlässige Kommunikation. Darüber hinaus ermöglicht Thyone-I in Verbindung mit einem anderen Thyone-I-Gerät einen nahtlosen und sicheren Kabelersatz, wenn es im transparenten Modus konfiguriert ist.

Das FeatherWing-Board Adrastea-I verfügt über LTE-M/NB-IoT-Fähigkeit, externe LTE- und GNSS-Antennen und ein USB-Kabel (**Bild 2**). Zu den vielseitigen Funktionen gehören die flexible Auswahl der Betriebsart zwischen LTE-M und NB-IoT, integriertes GNSS mit GPS- und GLONASS-Unterstützung, eine ARM Cortex-M4-MCU und Kompatibilität mit verschiedenen Protokollen und Bändern, alles im praktischen Adafruit-Feather-Format, das eine einfache Erweiterung mit bestehenden Feather-Boards gewährleistet.

Das Adafruit Feather M0 Express ist ein Feather-MCU-Board, das auf dem Arm Cortex-M0-Prozessor ATSAMD21G18 basiert, der mit 48 MHz getaktet wird und mit 3,3 V arbeitet, genauso wie ein Arduino Zero (**Bild 3**). Dieser Chip verfügt über satte 256K Flash-Speicher (achtmal mehr als der Atmega328 oder 32u4) und 32K RAM (16 × soviel). Dieser Chip bietet einen eingebauten USB-Anschluss und somit USB-zu-Seriell-Programmier- und Debug-Fähigkeit.

Der Arduino Nicla Vision ist mit einem robusten STM32H747AII6 Dual Arm Cortex M7/M4-Prozessor und einer 2-MP-Farbkamera ausgestattet, was die Unterstützung von TinyML ermöglicht (**Bild 4**). Außerdem verfügt er über einen intelligenten 6-Achsen-Bewegungssensor, ein integriertes Mikrofon und einen Abstandssensor. Das Board wurde für eine nahtlose Kompatibilität mit allen Arduino-Portenta- und MKR-Produkten entwickelt, ist vollständig in OpenMV integriert, unterstützt MicroPython und bietet WiFi- und Bluetooth-Low-Energy-Kommunikation.

Edge Impulse ist eine Entwicklungsumgebung für maschinelles Lernen auf Embedded-Geräten. Sie ermöglicht die Erfassung von Sensordaten aus der realen Welt, die Entwicklung eines Algorithmus für maschinelles Lernen und den einfachen Einsatz eines Modells auf einem beliebigen Edge-Gerät.

Prototyping-Konfiguration

Bild 5 zeigt die Architektur des Prototyps. Das Nicla Vision Board dient als Bildverarbeitungskomponente des Schädlingserkennungssystems und hat die Aufgabe, das ML-Modell auszuführen. Es wird an ein Thyone-Board angeschlossen, das die Übertragung von Daten an den Hauptknoten (Gateway) ermöglicht. Dieser Hauptknoten besteht aus einem Thyone-Board, einem Adafruit-M0-Express und einem Adrastea-I-Board, das für die Übertragung der gesammelten Daten an die Cloud über eine IoT-Mobilfunkverbindung zuständig ist.

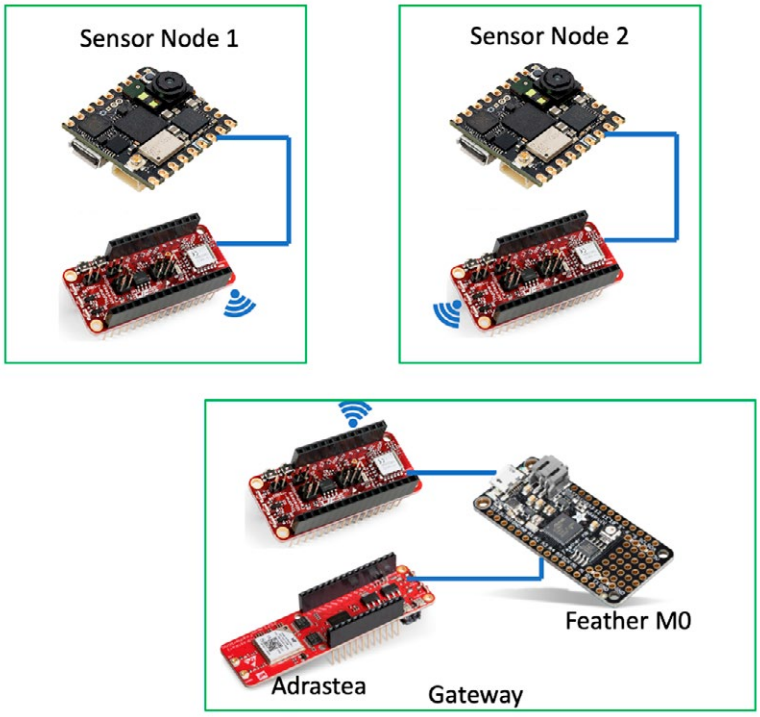


Bild 5. Die Architektur des Prototyps zur Schädlingserkennung mit mehreren Sensorknoten und einem Gateway. (Quelle: Mouser Electronics)

An den Sensorknoten müssen die Pins, wie in **Tabelle 1** dargestellt, angeschlossen werden. Um den FeatherWing Thyone-I in den transparenten Modus zu schalten, muss Pin 3 am Jumper JP1 mit 3V3 verbunden werden. Im Gateway wird der Thyone-I-FeatherWing im transparenten Modus einfach mit dem Adrastea-I-FeatherWing und dem Adafruit-Feather-M0-Express zusammengeschaltet.

Tabelle 1: Anschlüsse von Nicla Vision an FeatherWing Thyone-I.

Nicla Vision	FeatherWing Thyone-I
GND	GND
3V3	3V3
TX	RX
RX	TX

Um die Hardware in Edge Impulse zu konfigurieren, müssen die folgenden Softwarekomponenten installiert werden:
1. Edge Impulse CLI installieren.
2. Arduino CLI installieren.

Folgende Schritte sind erforderlich, um den Hauptknoten zu programmieren:
1. Installation von Visual Studio Code und dem Platform-IO-Plugin.
2. Klonen/Herunterladen des FeatherWings-Repositorys.
3. Öffnen des Beispiels *Thyone-Adrastea-Bridge*. Code erstellen und diesen auf das Adafruit-Feather-M0-Express laden.

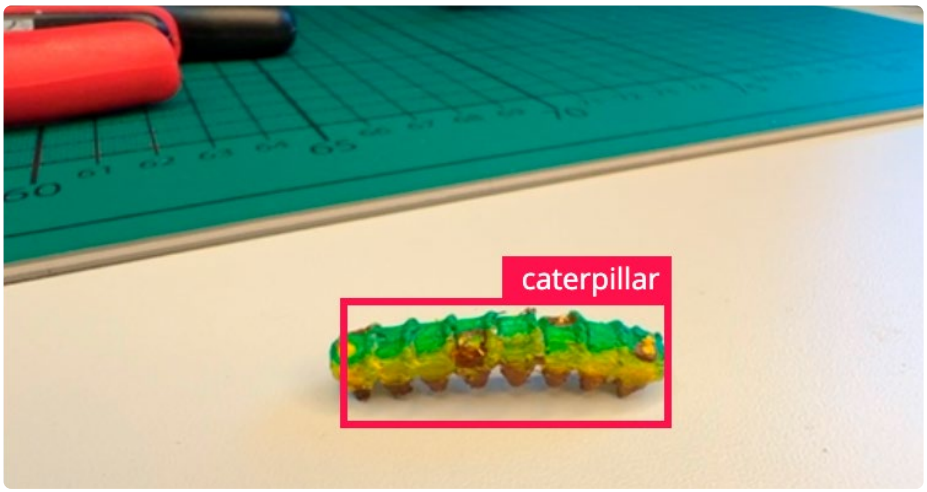


Bild 6. Kennzeichnung (Labeling) der Schädlingsbeispiele. (Quelle: Mouser Electronics)

Training des Bildverarbeitungsmodells

Edge Impulse macht die Erstellung eines Bildverarbeitungsmodells sehr einfach. Man muss nur ein paar Bilder aufnehmen und das Modell auf der Edge-Impulse-Plattform trainieren.

1. Erstellen Sie ein Konto auf der Edge-Impulse-Website.
2. Erstellen Sie ein neues Projekt.
3. Klicken Sie auf *Datenerfassung* und dann auf *connect a new device*.
4. Sie können mit Ihrem Mobiltelefon Fotos von dem Schädling aufnehmen, und diese werden automatisch mit Ihrem Edge-Impulse-Konto synchronisiert. Dazu müssen Sie den auf der Seite angezeigten QR-Code scannen.
5. Nachdem der QR-Code gescannt wurde, werden Fotos aus verschiedenen Winkeln und mit unterschiedlichen Lichteinstellungen aufgenommen.
6. Nachdem Sie die Bilder gesammelt haben, müssen Sie die Schädlinge etikettieren. Klicken Sie auf *Labeling queue*.
7. Beschriften Sie den Schädling mit der Maus, indem Sie einen Kasten um ihn herum erstellen (**Bild 6**). Damit ist das Sammeln der Daten abgeschlossen, und man kann zum nächsten Schritt übergehen, dem Training des Modells.

Erstellen des Impulses

Zur Erstellung des Impulses werden Rohdaten als Eingabe verwendet, dann durch die Signalverarbeitung Merkmale extrahiert, und schließlich wird ein Lernblock verwendet, um neue Daten zu klassifizieren (**Bild 7**).

1. Je nach Anwendung werden die Bilddaten angepasst.

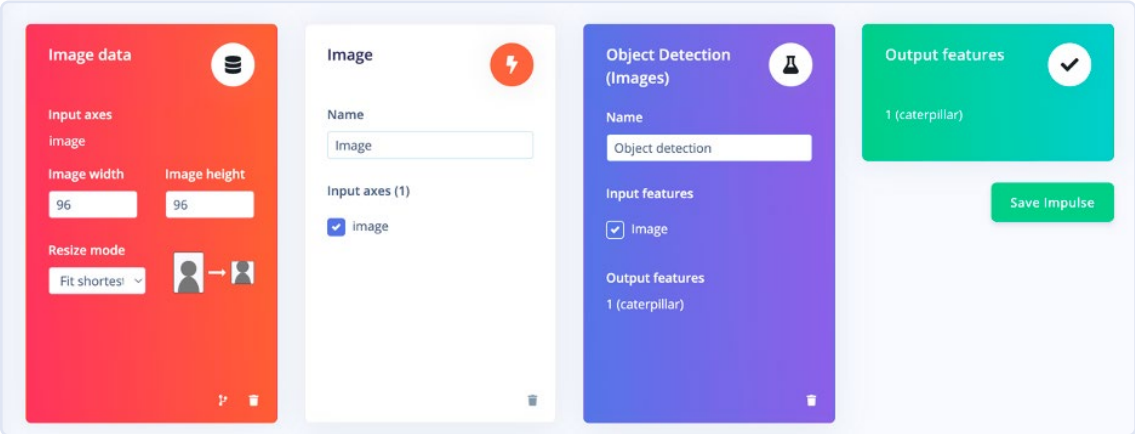


Bild 7. Erstellungsprozess eines Impulses. (Quelle: Mouser Electronics)

2. Für den Verarbeitungsblock wählen Sie das Bild.
3. Für den Lernblock wählen Sie Objekterkennung.
4. Speichern Sie den Impuls.

Klassifizierung über ein neuronales Netz

Der „Classifier" des neuronalen Netzes nimmt einige Eingabedaten und gibt einen Wahrscheinlichkeitswert aus, der angibt, wie wahrscheinlich es ist, dass die Eingabedaten zu einer bestimmten Klasse gehören. Für dieses Projekt wurden die Trainingszyklen auf 60 festgelegt.

Exportieren Sie den Impuls als OpenMV-Firmware und navigieren Sie zu OpenMV IDE. Sie können das von Mouser trainierte Modell verwenden, indem Sie die Dateien von der GitHub-Seite [8] herunterladen.

Schließen Sie Ihr Gerät an Ihren Computer an und klicken Sie auf *Connect* in der OpenMV IDE. Das Gerät sollte automatisch erkannt werden.

Drahtlose Vernetzung

Sensor zum Gateway: Der Thyone-I-Wireless-FeatherWing, der im transparenten Modus arbeitet, fungiert als kabelersetzende 2,4-GHz-Funkverbindung. In dieser Betriebsart werden alle Daten, die vom Nicla Vision über die UART-Schnittstelle eingehen, drahtlos übertragen. Ebenso werden die über die drahtlose Verbindung empfangenen Informationen an die UART-Schnittstelle weitergeleitet.

Gateway zur Cloud: Das Gateway besteht aus dem Peer Thyone-I-FeatherWing, um Nachrichten von Knotenpunkten zu empfangen. Die Daten aus dem Funk werden transparent über UART an den Feather-M0-Express übertragen. Die M0-MCU konfiguriert beim Hochfahren den Adrastea-I-FeatherWing für die Verbindung mit dem MQTT-Broker der Cloud. Beim Empfang von Nachrichten von den Sensorknoten veröffentlicht die M0-MCU die Daten direkt in der Cloud. Der MQTT-Broker der Cloud leitet die Nachrichten an alle abonnierten Clients weiter. Der Benutzer empfängt die Nachrichten mit einer beliebigen offenen MQTT-Mobiltelefonanwendung. Der Code für diese Anwendung steht auf der GitHub-Seite von Würth Elektronik zur Verfügung [9].

Leistungsstarke Kombination von Technologien

Fortschritte auf dem Gebiet der Elektronik und des ML ermöglichen es, komplexe Aufgaben wie die Objekterkennung mit begrenzten Hardwareressourcen durchzuführen. In Verbindung mit drahtloser Datenübertragung entsteht so ein sehr robuster Mechanismus zur Fernüberwachung und Fernsteuerung der Umgebung. In diesem Beispiel werden diese Technologien genutzt, um einen Prototyp eines Schädlingserkennungssystems zu entwickeln, das eine intelligente und vernetzte Landwirtschaft ermöglicht

240389-02

Über Rafik Mitry

Rafik Mitry kam 2019 zu Mouser Electronics, nachdem er seinen Master in Elektrotechnik an der Technischen Universität München abgeschlossen hatte, wo er auch drei Jahre lang in der Forschung auf dem Gebiet des Energy Harvesting tätig war. Als Technical Marketing Engineer bei Mouser erstellt Rafik einzigartige technische Inhalte, die aktuelle und zukünftige Technologietrends in der Elektronikbranche widerspiegeln. Rafik hält sich nicht nur über die neuesten Technologietrends auf dem Laufenden, sondern ist auch ein begeisterter Liebhaber von Luftfahrt und Tennis.

Über Shashank Hedge

Shashank Hedge ist Masterstudent an der Technischen Universität München. Er arbeitet seit 2022 bei Würth Elektronik eiSos als Praktikant/Werkstudent in der Abteilung Wireless Connectivity und Sensoren. Seine Forschungsinteressen umfassen Embedded-Systeme, Internet der Dinge und Fahrzeugkommunikation.

WEBLINKS

[1] Arduino Nicla Vision: https://tinyurl.com/arduino-nicla-vision
[2] Würth Elektronik Thyone-I FeatherWing: https://tinyurl.com/thyone-i
[3] Würth Elektronik Adrastea-I FeatherWing: https://tinyurl.com/adrastea-i
[4] Adafruit Feather M0 Express: https://tinyurl.com/mo-express
[5] OpenMV: https://openmv.io/pages/download
[6] Edge Impulse: https://edgeimpulse.com
[7] Visual Studio Code mit PlatformIO-Erweiterung: https://platformio.org/install/ide?install=vscode
[8] GitHub-Seite von Mouser: https://github.com/rmitry/pest-detection
[9] GitHub-Seite von Würth Elektronik eiSos: https://github.com/WurthElektronik/FeatherWings

Warum Anybus CompactCom die ideale Wahl für die industrielle Embedded-Kommunikation ist

Ein Beitrag von HMS Networks

Die Flexibilität, Verbindungen zu allen wichtigen Industrieprotokollen anbieten zu können, wird zu einem immer wichtigeren Merkmal von intelligenten Geräten. Die Entwicklung der erforderlichen Hard- und Software ist jedoch ein komplexer, zeitaufwändiger Prozess. Glücklicherweise gibt es eine einfachere Alternative. In diesem Artikel erklären wir, wie die Implementierung von Anybus CompactCom von HMS Networks in industrielle Geräte die Konnektivität erheblich vereinfacht.

Kein einzelnes vorherrschendes Industrieprotokoll

Im Laufe der Zeit haben unterschiedliche Protokolle in verschiedenen Regionen und Branchen an Popularität gewonnen. So ist beispielsweise PROFINET in Europa weit verbreitet, EtherNet/IP hat einen bedeutenden Marktanteil in Nordamerika, und EtherCAT und CC-Link sind in Asien, insbesondere in Japan, beliebt. Um den Erfolg Ihres Geräts auf einem globalen Markt zu gewährleisten, muss es in der Lage sein, sich mit mehreren Industrieprotokollen zu verbinden.

HMS Networks führt jedes Jahr eine umfassende Analyse des industriellen Netzwerkmarktes über die Verteilung der neuen verbundenen Knoten nach Typ und Protokoll innerhalb der Fabrikautomation durch. Die vollständige Studie mit der Analyse ist auf der Website von HMS Networks zu finden.

Die Herausforderung der Multi-Protokoll-Entwicklung

Die Unterstützung mehrerer Protokolle erfordert ein tiefes Verständnis sowohl der Hardware- als auch der Software-Architekturen. Diese sind von Natur aus komplex und für jedes Protokoll unterschiedlich, auch wenn sie auf den ersten Blick gleich erscheinen.

Anybus CompactCom stellt sich diesen Herausforderungen, indem es eine umfassende Kommunikationslösung bietet. Im Gegensatz zu den meisten Alternativen integriert Anybus CompactCom sowohl Hardware als auch Software in einem einzigen Modul.

Wesentliche Funktionen und Vorteile

Vollständiges Kommunikationsmodul: Das Anybus CompactCom unterstützt alle wichtigen Industrie- und Cloud-Protokolle. Mit nur einem einzigen Entwicklungszyklus können Sie Ihre Produkte einfach mit allen von Ihnen gewählten Protokollen verbinden.

Lebenszyklusmanagement: HMS pflegt sowohl die Hardware als auch die Software über den gesamten Lebenszyklus des Produkts, einschließlich regelmäßiger Updates der Netzwerkspezifikationen.

Verfügbar in zwei Formfaktoren:

- **Modul**: Enthält einen integrierten Anschluss, um Flexibilität und Benutzerfreundlichkeit zu gewährleisten, ideal für Nachrüstungsoptionen.
- **Brick**: Unterstützt kundenspezifische Anschlüsse, geeignet für Anwendungen, die einen besonderen Schutz vor Umwelteinflüssen erfordern, zum Beispiel Wasserdichtheit.

Vorzertifiziert für einfache Integration: Vereinfacht den Übergang zur vollständigen Zertifizierung und gewährleistet die Einhaltung von Industriestandards.

Flexible Hardwareschnittstelle: Unterstützt parallele, SPI- und serielle Kommunikation und bietet vielseitige Integration mit verschiedenen Host-Mikrocontrollern.

Leistungsstarke Softwareschnittstelle: Ein kostenloser und betriebssystemunabhängiger Treiber in der Programmiersprache C, der mit Ihrer Anwendungssoftware kompiliert werden kann.

Wie es funktioniert - Host- und Master-Architektur

Das Konstruktionsprinzip von Anybus CompactCom ist der Schlüssel zum Verständnis der Vorteile des Systems. Anybus CompactCom arbeitet in Automatisierungsgeräten als Kommunikationsknoten, der das Host-Gerät mit einer SPS verbindet und Prozessvariablen und Informationen austauscht (**Bild 1**). Diese Architektur ist wichtig, weil sie die Besonderheiten der Protokollverwaltung

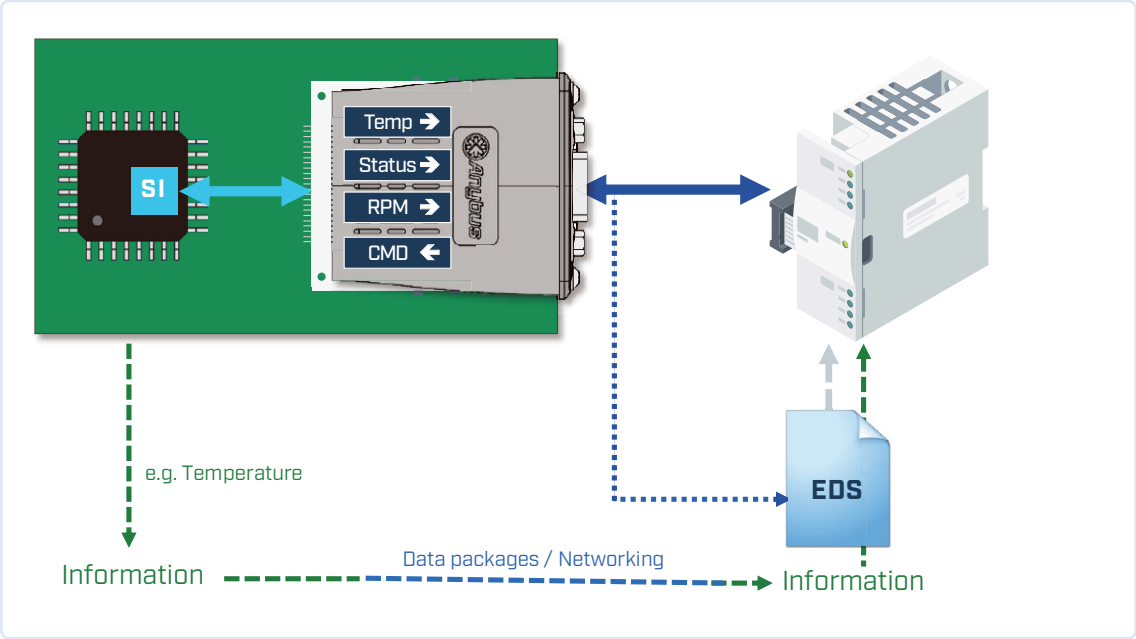


Bild 1. Anybus CompactCom verbindet den Host und den Master.

von der Kernanwendungslogik entkoppelt und so eine nahtlose Integration über verschiedene Protokolle hinweg gewährleistet. Für die Hersteller von Automatisierungsgeräten vereinfacht dieses Setup die Kommunikation zwischen ihren Mikrocontroller-Anwendungen und den Steuergeräten, so dass sich die Entwickler auf ihre Anwendung konzentrieren können, anstatt sich mit den Feinheiten der einzelnen Kommunikationsprotokolle auseinandersetzen zu müssen.

Integrieren Sie die Hardware

Das Herzstück von Anybus CompactCom ist der NP40, ein speziell entwickelter Chip, der programmierbare Hardware und Software kombiniert (**Bild 2**):

Mikrocontroller und FPGA: Der FPGA übernimmt die protokollspezifische Logik (zum Beispiel Echtzeit-Switches für PROFINET oder Standard-EtherCAT-Schnittstellen), während auf dem Mikrocontroller ein Echtzeit-Betriebssystem zur Verwaltung der Protokollstacks läuft.

Standard-API: Die mitgelieferte API bietet eine konsistente Softwareschnittstelle zwischen dem NP40 und Ihrem Host-Mikrocontroller, die trotz interner Protokollunterschiede eine nahtlose Softwareschnittstelle gewährleistet.

Hardware- und Anwendungsschnittstellen

Anybus CompactCom bietet Hardware- und Anwendungsschnittstellenoptionen, die auf Ihre Bedürfnisse zugeschnitten werden können (**Bild 3**). Die wichtigsten Komponenten und Überlegungen:

Datenmodi:

- **Parallel**: Verwendet 8 Bit oder 16 Bit parallel für den Hochgeschwindigkeitsdatenaustausch.
- **SPI (Serielle Peripherieschnittstelle)**: Ein schneller Datenaustausch mit bis zu 20 MHz, der komplexe Designs vereinfacht.
- **Seriell**: Langsamere Datenrate, unterstützt ältere Systeme.

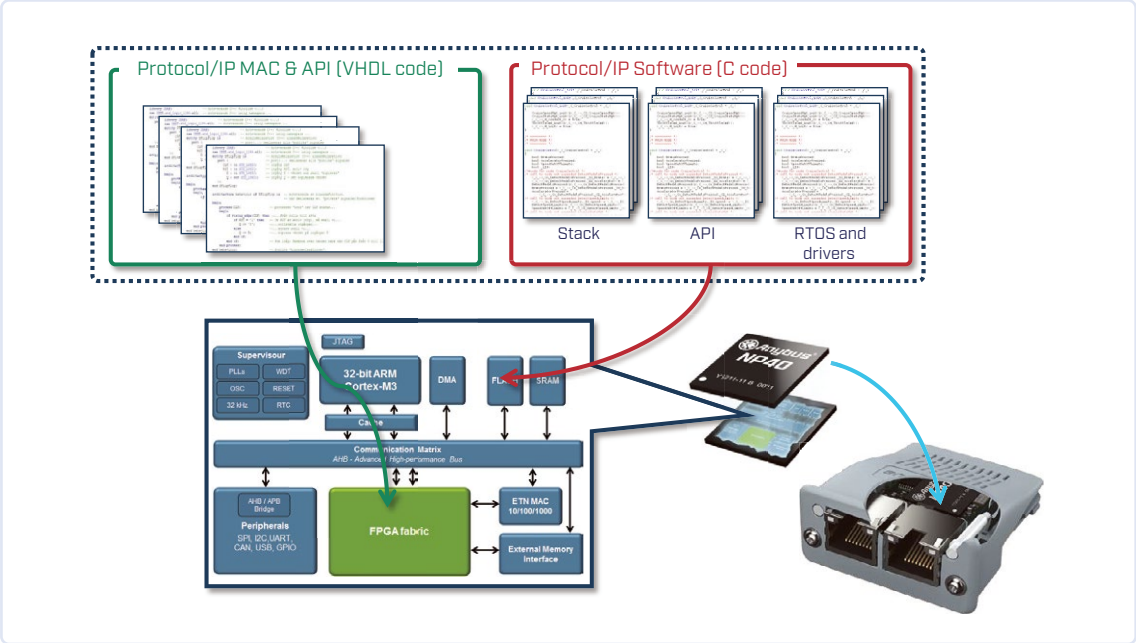


Bild 2. Anybus CompactCom bietet programmierbare Hardware und Software.

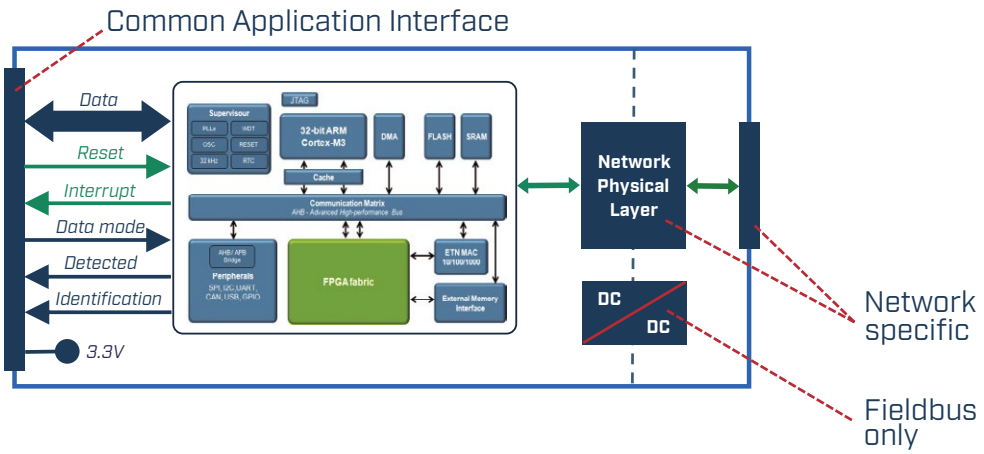


Bild 3. Die Hardware- und Anwendungsschnittstelle von Anybus CompactCom.

Flusskontrolle und Reset:
- **Reset-Leitung**: Anschluss an einen Ausgangsport des Mikrocontrollers, über den der NP40 per Software zurückgesetzt werden kann.
- **Interrupt-Leitung**: Programmablauf für bestimmte Ereignisse umschalten.

Erkennung und Identifizierung der Hardware:
- Die Anwendung kann testen, ob ein Anybus-CompactCom-Modul montiert ist und um welches Modell es sich handelt, indem sie den Startvorgang steuert.

Spannungsversorgung:
- Benötigt für den Betrieb eine 3,3 V-Spannungsversorgung.

Integrieren Sie die Software

Die API des Anybus CompactCom wird durch die Software-Schnittstelle gesteuert und bietet mehrere Optionen, die auf Ihre Bedürfnisse zugeschnitten werden können. Schlüsselkomponenten und Überlegungen:

Variablenverwaltung:
- **Anwendungsdaten-Instanzen (ADIs)**: Definieren Sie Netzwerkvariablen (ADIs) in Ihrer Software und geben Sie Namen, Datentypen, Elementanzahl und Wertebereiche an. Verwendet Zeiger für ein nahtloses Datenmanagement zwischen Anybus CompactCom und Ihrer Anwendung. Kann statisch oder dynamisch implementiert werden.
- **Azyklische und Echtzeit-Datenverarbeitung**: Verwalten Sie azyklische, nicht zeitkritische Anfragen und den Datenaustausch in Echtzeit effizient über die bereitgestellten Softwareschichten.

Netzwerkverwaltung:
- **Netzwerkobjekte**: Richten Sie netzspezifische Verwaltungsaufgaben ein, wie Hersteller-IDs, um einen reibungslosen Betrieb über verschiedene Protokolle hinweg zu gewährleisten.

Funktionsaufrufe:
- **'Keep alive'** und **'interrupt'**: Grundlegende Aufrufe zur Erleichterung der Bedienung der Schnittstellensoftware.

Zusätzliche Funktionen:
- **Ethernet-Goodies**: Aufrufe und Objekte zur Verwaltung des Webservers, E-Mail-Versand, FTP-Austausch, JSON und vieles mehr.

IIoT-Kommunikation

HMS bietet mit Anybus CompactCom IIoT Secure eine Variante, die eine sichere Kommunikation mit IIoT-Anwendungen über die Datenaustauschmethoden OPC UA oder MQTT ermöglicht. Um eine sichere Kommunikation zu gewährleisten, enthält Anybus CompactCom IIoT Secure Folgendes:

Zertifikatsverwaltung: Gewährleistet, dass die gesamte Kommunikation authentifiziert und sicher ist.

Verschlüsselung: Schützt Daten während der Übertragung, um unbefugten Zugriff zu verhindern.

Sicheres Gerät: Garantiert, dass das Gerät selbst vor Manipulationen und Cyber-Bedrohungen geschützt ist.

Secure Boot: Stellt sicher, dass das Gerät nur authentische Software ausführt und schützt so vor schädlichem Code.

Bei der Integration der Software können Sie festlegen, welche Variablen mit der lokalen Steuerung ausgetauscht werden müssen und welche an die digitalen Anwendungen gesendet werden sollen. Einmal konfiguriert, gewährleistet Anybus CompactCom einen nahtlosen und sicheren Datenaustausch.

Fazit

Anybus CompactCom ist eine umfassende Lösung für die industrielle Kommunikation, die sich durch eine beispiellose Flexibilität und einfache Integration auszeichnet. Durch die Kombination von robuster Hardware und hochentwickelter Software-Unterstützung entfallen die Komplexität und die versteckten Kosten, die mit der Entwicklung von Multiprotokoll-Lösungen verbunden sind. Dies macht es zur idealen Wahl für Hersteller, die auf einem globalen Markt wettbewerbsfähig bleiben wollen.

Weitere Informationen über Anybus CompactCom finden Sie unter https://www.hms-networks.com/de/network-interfaces-integration.

240392-02

IQRF Kommunikations-standard

Zuverlässigkeit für verlustbehaftete drahtlose Mesh-Netzwerke mit niedriger Datenrate

Ein Beitrag von IQRF Alliance

Laut Gartner Hype Cycles sind drahtlose Mesh-Netzwerke auch mehr als zwanzig Jahre nach den ersten Anzeichen eines Markterfolgs immer noch ein heißes Thema. Bisher gibt es bei allen weltweit unterstützten drahtlosen Netzwerktechnologien zu viele Kompromisse und Einschränkungen in Bezug auf Verzögerung, Hop-Anzahl-Grenzwerte, Determinismus und Zuverlässigkeit. Der Grund ist einfach: Technische Konzepte waren in erster Linie auf Routing-Optimierung und nicht auf Zuverlässigkeit ausgerichtet. Im Gegensatz zu diesem Ansatz konzentrierte sich IQRF vor allem auf die zuverlässige Nachrichtenübermittlung. Mit mehr als 250 Routing-Hops und zuverlässiger Nachrichtenzustellung zeichnet sich IQRF heute insbesondere bei Straßenbeleuchtungsanwendungen aus, bei denen große Reichweiten erreicht werden sollen und eine hohe Zuverlässigkeit gewährleistet sein muss.

Nach zwei Jahrzehnten auf dem Markt ist IQRF standardisiert und ermöglicht es jedem, alle technischen Funktionen und zuverlässigen Protokolle, die durch Dutzende Patente geschützt sind, unter einer „Royalty-free"-Lizenz zu nutzen und zu implementieren.

IQRF: Hauptmerkmale und Vorteile

IQRF ist eine komplette Technologie, ein Standard und ein Ökosystem bestehend aus Hardware (Transceiver, Gateways, Repeater, komplette Elektronik, Zubehör, Entwicklungstools), Software, Protokollen, Support, Tools und Inbetriebnahmediensten.
Seit 2004 haben MICRORISC und IQRF Tech IQRF entwickelt, eine zuverlässige bidirektionale drahtlose Netzwerkkommunikationstechnologie mit geringem Stromverbrauch. IQRF mit dem IQMESH-Protokoll ermöglicht eine äußerst robuste und ausgefeilte Mesh-Topologie. Tausende drahtlose Systeme auf der ganzen Welt sind der Beweis dafür.

Eigenschaften

Industrielle Zuverlässigkeit

Mehr Netzwerkknoten bedeuten dank des IQMESH-Protokolls eine höhere Zuverlässigkeit.
Es ist deterministisch und zuverlässig, selbst in anspruchsvollen Umgebungen.
Es verleiht dem drahtlosen Internet der Dinge (IoT) Robustheit und industrielle Zuverlässigkeit.

In einem IQRF-Netzwerk, das das IQMESH-Routing-Protokoll verwendet, kann das FRC-Protokoll (Fast Response Commands) veingesetzt werden, um eine schnelle Datenaggregation und Bestätigung von Broadcasts oder Multicasts zu ermöglichen.
Mit mehr als 200 Repeater-Nodes in einem einzigen Netzwerk, die in einer Mesh-Topologie zusammenarbeiten, ist es möglich, riesige Anlagen, Industriehallen und Straßen abzudecken und trotz schlechter Funkbedingungen in der Umgebung zuverlässig eine Nachricht zuzustellen.

Einfache Integration

Eine modulare Architektur, ein leichtes und dennoch leistungsstarkes DPA-Protokoll, Hardware- und Software-Entwicklungstools, zahlreiche Anleitungen, Beispiele und Bildungsunterlagen sowie exzellenter technischer Support machen die Entwicklung einfach.

Höchste Sicherheit

Der umfassende Ansatz von der Gerätezuordnung zum Netzwerk bis hin zur automatisch verschlüsselten Kommunikation basiert auf Sicherheitsstandards. Die Sicherheitsdienstschicht ist für die Gewährleistung der folgenden Sicherheitsziele verantwortlich: Frame-Integrität, Netzwerk-Frame-Authentizität, Vertraulichkeit von Fußzeilen und Nutzinhalten, Schutz gegen erneutes Versenden.

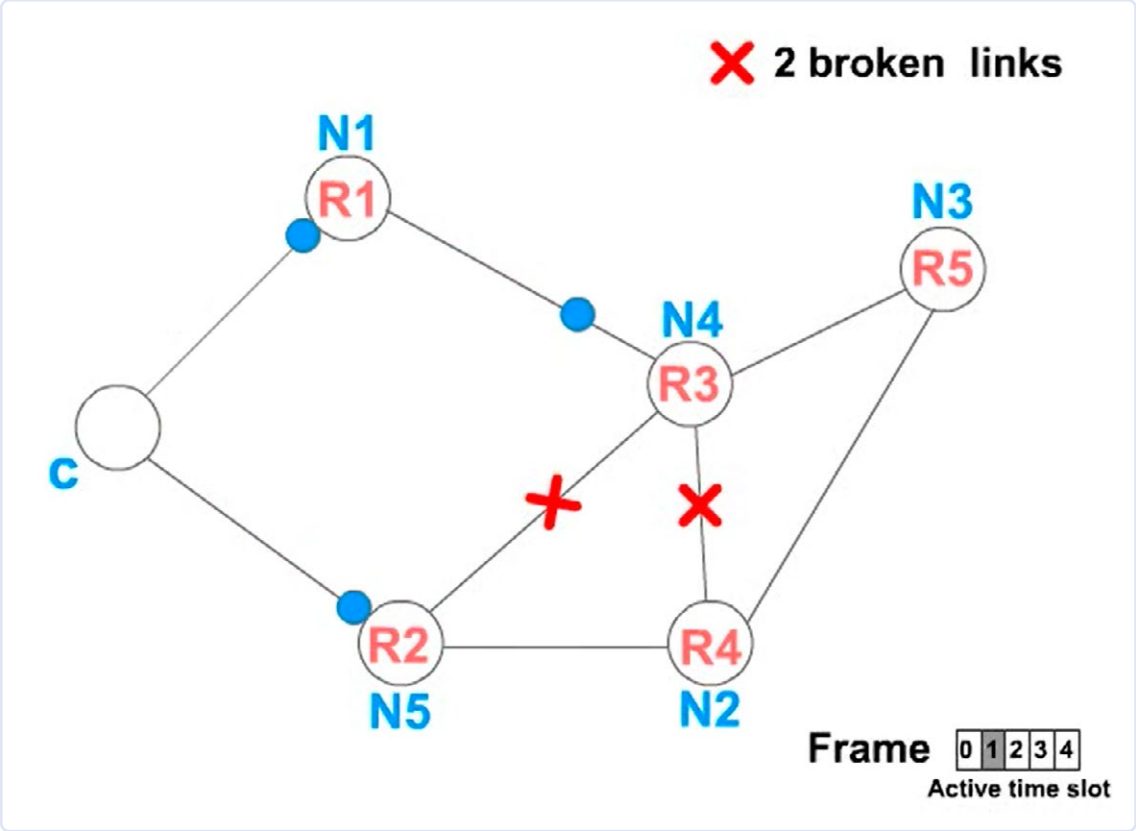


Bild 1. Routing IQRF.

Interoperabilität und ein umfassendes Ökosystem
Die Interoperabilität von Geräten verschiedener Hersteller basiert auf IQRF-Interoperabilitätsstandards. Für die IQRF-Interoperabilität zertifizierte Produkte sind auf dem Marketplace der IQRF Alliance gelistet und können problemlos in komplexen IoT-Projekten eingesetzt werden.

Verbrauch IQRF True Low Power
IQRF ist für einen effizienten Betrieb bei minimalem Stromverbrauch konzipiert und eignet sich daher für batteriebetriebene Geräte und Anwendungen, bei denen die Energieeffizienz von entscheidender Bedeutung ist.
Der geringe Verbrauch der drahtlosen Kommunikation bringt neue Standards für den Batteriebetrieb, der nur durch die Selbstentladung der Batterie definiert werden.

IQRF-Netzwerk mit IQMESH-Protokoll

IQRF-Netzwerke werden verwaltet und organisiert [1]. Das steuernde Element für andere Netzwerkgeräte, sogenannte *Knoten/Nodes*, ist der *Koordinator*. Nodes mit Routing-Fähigkeiten werden *Router* genannt. Die Netzwerkkommunikation erfolgt stets verschlüsselt und authentifiziert nach den neuesten Sicherheitsstandards.

Verknüpfungsprozess
Nodes werden vom Koordinator mit dem Netzwerk verknüpft. Während der Verknüpfung übermittelt der Koordinator die Verknüpfungsdaten auf sichere Weise über eine verschlüsselte Nachricht an die Nodes. Der IQRF-MAC ist die eindeutige Kennung des IQRF-Geräts, das zur Authentifizierung verwendet wird. Der Koordinator gibt Informationen wie Kommunikationsschlüssel und Netzwerkeinstellungen sicher weiter und weist jedem Nodes eine eindeutige logische Adresse zu, die zur Adressierung verwendet wird.

Topologie-Erkennungsprozess
Während dieses Prozesses ermittelt der Koordinator die Topologie der Routing-Nodes (Router) und weist ihnen eine virtuelle Routing-Nummer (VRN) zu, eine eindeutige Nummer, die die Entfernung vom Koordinator in Hops widerspiegelt und den Zeitraum für das Routing eines bestimmten Nodes definiert. TDMA-basiertes Directional Flooding garantiert eine deterministische und kollisionsfreie Routenführung.
Weitere Informationen zum Senden einer Nachricht in einem IQRF-Netzwerk mithilfe des IQMESH-Routingprotokolls finden Sie unter [2] (**Bild 1**).

FRC-Protokoll (Fast Response Commands)

Mit dem FRC-Protokoll können Sie schnell einen Befehl an mehrere Nodes im Netzwerk senden und kleine Daten von ihnen sammeln. Dies ist eine wesentlich schnellere Methode als die Abfrage einzelner Nodes einzeln mithilfe von Unicasts.

FRC ermöglicht in seiner aktuellen Implementierung das Senden von bis zu 30 Bytes an Benutzerdaten an alle oder ausgewählte Nodes und das Sammeln von

- 2 bits von bis zu 239 Nodes
- 1 Bytes von bis zu 63 Nodes
- 2 Bytes von bis zu 31 Nodes
- 4 Bytes von bis zu 15 Nodes

Die Befehlsausführungszeit ist streng deterministisch und unabhängig von der Topologie. Eine Animation des gesamten Prozesses finden Sie unter [3] (**Bild 2**).

Zertifizierung der IQRF-Interoperabilität

Unternehmen, die Produkte für das IQRF-Ökosystem herstellen, schließen sich in der IQRF Alliance [4] zusammen. Die IQRF Alliance testet Produkte auf Interoperabilität auf der Anwendungsebene.

Der Zertifizierungsprozess der IQRF-Interoperabilität umfasst die Beschaffung der HWPID für das zertifizierte Produkt, die Überprüfung der erforderlichen Dokumente des Produkts, die Überprüfung seiner Konformität mit dem IQRF-Interoperabilitätsstandard, die Lösung etwaiger Probleme und schließlich die Aunahme des Produkts in der Produktliste (Repository) und auf der Marketplace-Website.

Wenn Hersteller die IQRF-Technologie direkt in ihr Produkt implementieren oder ihre eigenen Transceiver bauen möchten, müssen sie die Richtlinien der IQRF Communication Standard Specification befolgen, die von der IQRF Standards Association [5] definiert werden.

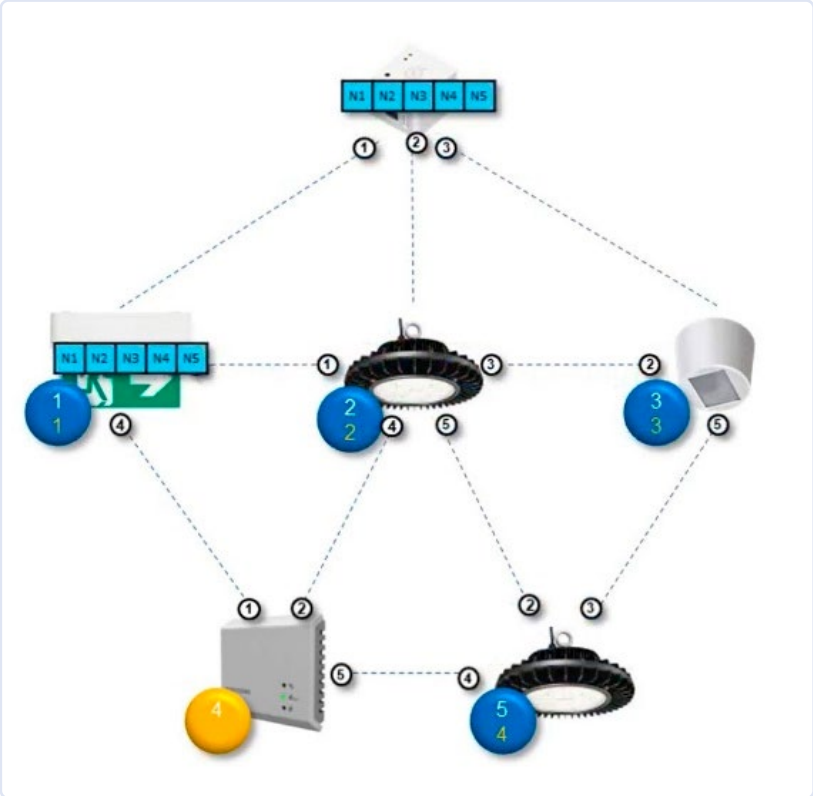
Die IQRF Standards Association ist eine gemeinnützige Organisation, die für die Spezifikation des IQRF-Kommunikationsstandards, die „Royalty-free"-Lizenzierung des offenen IQRF-Kommunikationsstandards und die Verwaltung von IQRF-MAC-Adressen verantwortlich ist. Hersteller können den IQRF-Kommunikationsstandard im Rahmen einer „Royalty-free" Lizenz in ihre Geräte implementieren. Die IQRF Standards Association zertifiziert die Interoperabilität auf der Basiskommunikationsebene.

Abschluss

Dank Routing kann IQRF zuverlässig über kurze Distanzen von bis zu mehreren Kilometern kommunizieren. Damit eignet es sich hervorragend für viele Low-Energy-Anwendungen aus der Überwachung, Automatisierung oder Steuerung. Es ist perfekt für Anwendungen von Sicherheit über Gebäudeautomation bis hin zu Smart City. Der Vorteil der IQRF-Technologie liegt in ihrer Kombination aus stromsparendem Betrieb, der Fähigkeit, in einer Netzwerktopologie mit hoher Zuverlässigkeit und Reichweite zu arbeiten, Interoperabilität von IQRF-Ökosystemprodukten, Sicherheitsfunktionen und einfacher Integration, was sie zu einer universellen Lösung für eine Vielzahl von IoT-Anwendungen mit drahtloser Übertragung macht.

240418-02

Bild 2. FRC-Protokoll.



WEBLINKS

[1] IQRF - Technology for wireless (2024): https://www.iqrf.org/
[2] IQRF-Netzwerk mithilfe des IQMESH-Routingprotokolls: http://www.iqrf.org/technology/iqmesh/unicast
[3] Animation des gesamten Prozesses: https://www.iqrf.org/technology/iqmesh/frc
[4] IQRF Alliance (2024): https://www.iqrfalliance.org/
[5] IQRF Standards Association (2024): https://standard.iqrf.org/

Bau eines **intelligenten Agrarroboters**

Wesentlichen technischen Überlegungen und Herausforderungen

Nahaufnahme eines automatisierten Ernteroboters. (Quelle: Sunday Cat Studio/stock.adobe.com)

Von Mark Patrick (Mouser Electronics)

Dieser Beitrag befasst sich mit den wichtigsten technischen Überlegungen und Herausforderungen bei der Entwicklung von Agrarrobotern, um den wirtschaftlichen Erfolg in rauen und schwierigen Umgebungen zu gewährleisten. Diese Roboter müssen einen Ausgleich zwischen Langlebigkeit und Präzision schaffen, effizient mit begrenzten Energieressourcen umgehen und robuste Bauteile enthalten, die Staub, Feuchtigkeit und extremen Wetterbedingungen standhalten. Echtzeitverarbeitung und Onboard-Intelligenz sind entscheidend, um die Abhängigkeit von unzuverlässigen Netzwerken zu minimieren. Gleichzeitig muss die Integration von Sensoren im Hinblick auf Kosten, Energie und Betriebsverhalten optimiert werden. Durch den Einsatz stromsparender Technologien und fortschrittlicher KI/ML-Lösungen im Edge-Bereich lässt sich der Funktionsumfang der Roboter weiter verbessern, sodass sie für eine nachhaltige Lebensmittelproduktion geeignet und effizient sind.

Die konventionellen landwirtschaftlichen Bewirtschaftungsverfahren stehen vor der Herausforderung, dass die Bevölkerung wächst, die Anbauflächen schrumpfen und die Ressourcen begrenzt sind. Durch den Einsatz intelligenter Lösungen, die modernste Technologien wie die Agrarrobotik nutzen, ist es jedoch möglich, diese anhaltenden Probleme direkt anzugehen und eine nachhaltige Lebensmittelproduktion zu gewährleisten.

Allerdings steht die Agrarrobotik, ähnlich wie die Industrie, die sie unterstützen soll, vor zahlreichen Herausforderungen. Landwirtschaftsbetriebe können wegen Faktoren wie Staub, Feuchtigkeit und Regen eine schwierige Umgebung für empfindliche elektronische Geräte darstellen. Außerdem sind Ressourcen wie Strom und Finanzen oft begrenzt. In diesem Beitrag befassen wir uns mit den Herausforderungen bei der Entwicklung von Agrarrobotern und erläutern die wichtigsten technischen Überlegungen und Herausforderungen, die Entwickler bewältigen müssen, um erfolgreiche Lösungen zu finden.

Wichtige Designüberlegungen

Im Gegensatz zu Robotern im Bau- und Fertigungsbereich, bei denen die Langlebigkeit im Vordergrund steht, oder zu medizinischen Robotern, bei denen die Präzision entscheidend ist, müssen Roboter in der Landwirtschaft einen optimalen Ausgleich zwischen diesen beiden Anforderungen finden, damit sie die Produkte verarbeiten können, ohne Schaden anzurichten. Außerdem müssen sie intelligent und sicher sein, um komplexe Aufgaben in der Nähe von Menschen und Tieren zu erledigen, und sie müssen für eine optimale Energieeffizienz entwickelt werden.

Wirkungsgrad

Der Wirkungsgrad ist für Agrarroboter von entscheidender Bedeutung, da sie über lange Zeiträume autonom betrieben werden müssen und oft nur über begrenzte Batterien oder Treibstoffvorräte verfügen. Damit landwirtschaftliche Roboter wie beispielsweise Erntemaschinen (**Bild 1**) erfolgreich eingesetzt werden können, müssen sie diese Anforderung erfüllen und wirtschaftlich betrieben werden können, insbesondere angesichts der finanziellen Engpässe vieler landwirtschaftlicher Betriebe.

Bei der Entwicklung von Agrarrobotern ist Energieeffizienz ein entscheidender Faktor. Durch Merkmale wie Schlafmodi, die den Roboter bei Bewegung intelligent aufwecken, oder Modi mit geringer Leistung, die den Funktionsumfang einschränken, um den Stromverbrauch zu senken, kann die Lebensdauer der Batterie verlängert oder der Kraftstoffverbrauch reduziert werden. Damit

der Wirkungsgrad von Robotern gewährleistet ist, müssen Entwickler entweder solche Strategien vorrangig entwickeln oder Bauteile verwenden, die diese Funktionen bereits enthalten.

Langlebigkeit

Um die Robotertechnik für die rauen Umgebungen in der Landwirtschaft auszulegen, sind robuste Bauteile und Materialien erforderlich. Steckverbinder müssen wetterfest sein und über Dichtungen verfügen, die das Eindringen von Feuchtigkeit verhindern. Gehäuse müssen aus haltbaren, UV-beständigen Materialien bestehen, die Schutz vor Stößen und rauen Wetterbedingungen bieten. Antennen müssen für hohe Zuverlässigkeit und optimales Betriebsverhalten bei rauen Wetterbedingungen entworfen werden.

Um die Zuverlässigkeit zu gewährleisten, können Entwickler von Haus aus robuste Komponenten mit hoher Schutzart verwenden oder sich für Standardkomponenten entscheiden, die in einem Schutzgehäuse untergebracht sind, um die Haltbarkeit zu gewährleisten. Platinen können beispielsweise mit Schutzlacken versehen werden, um sie vor Feuchtigkeit und Staub zu schützen. Darüber hinaus können Kühlkörper eingesetzt werden, um überschüssige Wärme von elektronischen Bauteilen oder Umgebungsbedingungen wie der Sonne abzuleiten. Zudem können Gehäuse so konstruiert werden, dass sie sowohl gegen Hitze als auch gegen das Eindringen von Wasser geschützt sind. Durch ein Design, das auf eine robuste Bauweise ausgerichtet ist, können Agrarroboter die Zuverlässigkeit und Langlebigkeit erreichen, die für Applikationen im Outdoor-Bereich erforderlich sind.

Echtzeit-Verarbeitung und Onboard-Intelligenz

In der Agrarrobotik kann der Einsatz von Cloud-basierter Intelligenz oder Verarbeitung aufgrund möglicher Kommunikationsausfälle problematisch sein. Die schlechte Mobilfunkabdeckung in ländlichen Gebieten stellt eine Herausforderung dar, und die Kosten für die Einrichtung lokaler WiFi-Netzwerke und Cloud-Dienste sind beträchtlich. Zudem führt die Cloud-basierte Kommunikation zu einem erheblichen Stromverbrauch der Batterien von stromsparenden Robotern wie Ernterobotern (**Bild 2**) und Unkrautbekämpfungsmaschinen.

Edge-basierte KI oder maschinelles Lernen (ML), bei der die Daten direkt auf dem Gerät und nicht in der Cloud analysiert werden, lösen diese Probleme effektiv. Da die Intelligenz an Bord bleibt, können Agrarroboter Analysen in Echtzeit durchführen und unmittelbar Entscheidungen treffen, ohne von externen Netzwerken abhängig zu sein. Dies reduziert den Energieverbrauch und verringert die Abhängigkeit von Netzwerken. Die Edge-Datenverarbeitung verbessert den Wirkungsgrad und die Sicherheit bei Aufgaben wie der Erkennung von Hindernissen, der Präzisionspflanzung und der Echtzeitüberwachung des Pflanzenzustands erheblich und arbeitet im Vergleich zu cloudbasierten Lösungen mit minimalen Verzögerungen.

Spezifikation der richtigen Sensoren

Die Integration von Sensoren ist für die Agrarrobotik unerlässlich. Sensoren wie LiDAR, Kameras und Bodenfeuchtedetektoren verbessern die Effizienz und Leistung, haben aber ihren Preis. Entwickler müssen daher Technologien auswählen, die einen Ausgleich zwischen finanziellen, energie- und platztechnischen Beschränkungen schaffen, damit ihr Design den Markterwartungen entspricht.

Entwicklung von zuverlässigen und hochleistungsfähigen Agrarrobotern

Der Erfolg von Agrarrobotern hängt entscheidend von der Leistungsfähigkeit der Bauteile ab.

Stromsparende Technologien

Ruhe- oder Stromsparmodi in Bauteilen wie Sensoren, Steuerungselektronik und Kommunikationsmodulen können dazu beitragen, die Batterielebensdauer oder den Kraftstoffverbrauch eines Roboters zu optimieren. Bildsensoren wie der AR0830 Hyperlux LP von onsemi [1] sind wesentliche Bestandteile von Agrarrobotern. Der AR0830 wurde speziell für hohe Leistung bei minimalem Strombedarf entwickelt. Mit einer Bildwiederholrate von 60 Bildern pro Sekunde eignet er sich ideal für Applikationen der maschinellen Bildverarbeitung in der Landwirtschaft, zum Beispiel die Erkennung von reifem Obst oder von Schädlingen und Unkraut. Neben dem Stromsparmodus bietet der Sensor auch Funktionen wie Wake-on-Motion- und Subsampling-Modi, mit denen sich die für die Datenübertragung erforderliche Bandbreite weiter reduzieren lässt, was die Energieeffizienz weiter verbessert. Mit diesen Modi lässt sich ein optimales Betriebsverhalten für verschiedene landwirtschaftliche Applikationen durch einen Ausgleich zwischen Auflösung, Empfindlichkeit, Bildrate und Stromverbrauch erzielen.

Bild 1. Eine automatisierte Erntemaschine im Einsatz mit einem Sammelwagen. (Quelle: kinwun/stock.adobe.com)

Bild 2. Tomatenpflückroboter verwenden häufig ML-basierte maschinelle Bildverarbeitung, um zu bestimmen, ob das Gemüse reif ist. (Quelle: kritsana/stock.adobe.com)

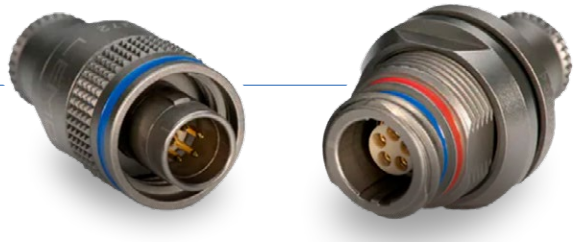


Bild 3. Die Steckverbinder der Baureihe M von Lemo mit Ratschenkupplung sind so konzipiert, dass sie rauen Umgebungsbedingungen standhalten. (Quelle: Mouser Electronics)

Die Steuerungselektronik spielt bei der Entwicklung von Agrarrobotern eine entscheidende Rolle, da sie wesentliche Betriebsabläufe überwacht und den Funktionsumfang und Wirkungsgrad der Roboter bestimmt. Die extrem stromsparenden Mikrocontroller (MCUs) der Baureihe STM32U5 von STMicroelectronics [2] basieren auf der Plattform Arm Cortex -M33 und eignen sich ideal für Subsysteme in der Agrarrobotik wie Mensch-Maschine-Schnittstellen (HMI) und Sensorsteuerung.
Die Baureihe STM32U5 ist für Umgebungstemperaturen von bis zu 125 °C geeignet und bietet wichtige Features wie Hardware-Sicherheitsfunktionen, Kommunikationsperipherie und Flash-Speicher. Mit dem Low Power Background Autonomous Mode (LPBAM) kann die Peripherie aktiv bleiben, während sich das Gerät im Stopp-Modus befindet, was erheblich Strom spart. Der große Funktionsumfang und die intelligenten Modi sorgen im Vergleich zu konventionellen Lösungen für einen deutlich höheren Wirkungsgrad.

Robuste Bauteile

Agrarroboter müssen den rauen Bedingungen in der Landwirtschaft standhalten können und benötigen robuste Bauteile. Lösungen wie die Außenantennen der IPW-Baureihe von TE Connectivity/Linx Technologies [3] bieten einen Frequenzbereich von 617 MHz bis 7,1 GHz, der sich für die Kommunikation im Mobilfunk-, WiFi- und LPWA/ISM-Bereich eignet. Die Antennen haben einen Gewinn von bis zu 8,7 dBi und sind nach IP67 zertifiziert, was eine starke Verbindung in weiten ländlichen Gebieten und einen dauerhaften Schutz gegen das Eindringen von Staub und Wasser gewährleistet. In der Agrarrobotik können diese Antennen mit LoRaWAN-Kommunikationsmodulen wie den LoRa-Lösungen von Microchip Technology [4] kombiniert werden. Im Vergleich zu anderen Wireless-Protokollen bieten diese LoRaWAN-Lösungen eine stromsparende Konnektivität durch Verwendung von Funkfrequenzbändern mit großer Reichweite im Sub-Gigahertz-Bereich, die eine Kommunikation über Entfernungen von mehr als 15 Kilometern in ländlichen und vorstädtischen Gebieten ermöglicht.
Die Zuverlässigkeit von Steckverbindern ist ein kritischer Punkt in der Agrarrobotik, da sie potenzielle Schwachstellen für das Eindringen von Wasser und Schmutz darstellen. Die Wahl der richtigen Steckverbinder für die interne Verdrahtung, die externe Kommunikation und die elektrischen Ladungen ist von entscheidender Bedeutung, um Ausfälle in der Praxis zu vermeiden. Die Baureihe M von Lemo [5] ist bekannt für ihre Langlebigkeit in der Industrierobotik und im Automotive-Bereich und eignet sich sehr gut für den Einsatz in der Landwirtschaft.
Diese Steckverbinder besitzen 2...114 Kontakte, bieten eine hervorragende Schaltleistung, sind für IP68 ausgelegt und können bis zu 15 Stunden dem Wasserdruck in 2 Metern Tiefe standhalten.

Machine Learning an der Edge nutzen

Präzision sowie eine schnelle und zuverlässige Verarbeitung sind in der Agrarrobotik von entscheidender Bedeutung. Die Integration moderner KI- oder ML-Lösungen ist unerlässlich geworden, um das Funktionsspektrum zu verbessern, die Leistung zu optimieren und die Anpassungsfähigkeit an unterschiedliche Aufgaben und Umgebungen zu gewährleisten.
Edge Impulse [6] ist eine vielseitige Plattform für die Entwicklung und den Einsatz von Edge-KI-Lösungen, die sich ideal für die Agrarrobotik eignet. Sie minimiert die Abhängigkeit von ständiger Cloud-Kommunikation, schont die Batterien und ermöglicht Echtzeit-Entscheidungen in Umgebungen, die im Hinblick auf die Konnektivität und Sicherheit schwierig sind.
Edge Impulse ist eine hardware-unabhängige Plattform und erlaubt die Nutzung eines nahezu unbegrenzten Hardware-Spektrums. Die Plattform unterstützt auch das Train Adapt Optimize Toolkit (TAO) von NVIDIA. NVIDIA TAO vereinfacht die Erstellung benutzerdefinierter KI-Modelle durch Transfer-Learning und ermöglicht es Entwicklern, mit minimalen Eingangsdaten vortrainierte Modelle für spezifische Aufgaben optimal abzustimmen. Dadurch eignet es sich ideal für Applikationen in der Landwirtschaft, wo KI-Modelle für Aufgaben wie die Obsternte, die Bodenüberwachung oder die Erkennung von Schädlingen präzise angepasst werden können und dazu beitragen, leistungsfähigere und effizientere Lösungen zu entwickeln.

Fazit

Agrarroboter haben zwar viele Eigenschaften und Anforderungskriterien mit ihren Pendants aus der Industrie gemeinsam, doch die Auswahl der Bauteile ist entscheidend, um sicherzustellen, dass sie den harten Anforderungen im landwirtschaftlichen Bereich gerecht werden. Die Einbindung robuster, effizienter und innovativer Technologien kann dazu beitragen, Produkte zu entwickeln, mit denen sich die Produktivität steigern, die Nachhaltigkeit gewährleisten und die Anforderungen in der Landwirtschaft erfüllen lassen, so dass die Zukunft der Lebensmittelproduktion in einer Welt mit begrenzten Ressourcen gesichert ist.

240380-02

WEBLINKS

[1] AR0830 Hyperlux LP von onsemi: https://eu.mouser.com/new/onsemi/onsemi-ar0830-image-sensors
[2] Baureihe STM32U5 von STMicroelectronics: https://eu.mouser.com/new/stmicroelectronics/stmicro-stm32u5
[3] Außenantennen der IPW-Baureihe von TE Connectivity/Linx Technologies: https://eu.mouser.com/new/te-connectivity/linx-outdoor-antennas
[4] LoRa-Lösungen von Microchip Technology: https://eu.mouser.com/new/microchip/microchip-lora-solutions
[5] Baureihe M von Lemo: https://eu.mouser.com/new/lemo/lemo-robotics-industrial-control
[6] Edge Impulse: https://mouser.co.uk/manufacturer/edge-impulse

Audio-Notchfilter mit einstellbarer Frequenz

Universelle Lösung zur Unterdrückung von Audiofrequenzen

Von Alfred Rosenkränzer (Deutschland)

Bei der Messung von Verzerrungen bei Audiogeräten ist es sinnvoll, das steuernde Sinussignal vor dem Spektrumanalysator auszufiltern. Mit diesem Notchfilter geht das besonders einfach, da die Notchfrequenz einstellbar ist. Außerdem ist man so nicht auf die typischen 1 kHz beschränkt, sondern kann auch im Bass- und Hochtonbereich messen.

Wozu das Einschleifen eines Notchfilters bei Messungen von Verzerrungen wie den THD (**T**otal **H**armonic **D**istortions) im Audiobereich gut ist, wurde bereits ausführlich in meinem Artikel „Fliege-Bandsperre für Audio-Messungen" [1] besprochen. Hier noch einmal das Wichtigste in Kürze.

Mit Notchfilter messen

Beim Messen der Verzerrungen des Testsignals am Ausgang einer Schaltung kann man nicht so einfach unterscheiden, wie viel davon bereits im speisenden Testsignal enthalten ist, wie viel davon vom zu beurteilenden Gerät stammt oder welche Teile durch das Messgerät selbst am Schluss beigetragen werden. Folgendes Vorgehen hilft dabei, trotzdem eine vernünftige Messung zu machen.

Zunächst misst man den Pegel des Testsignals ohne Notchfilter am Eingang des Messgeräts und notiert diesen. Statt nun den Ausgang (nach Pegelanpassung) einfach einem Spektrumanalysator zuzuführen, schaltet man zwischen Geräteausgang und Messgeräteeingang ein Notchfilter, das genau auf die Frequenz des als Testsignal dienenden Sinussignals abgestimmt ist. Das Testsignal wird somit stark abgeschwächt und kann daher keine Verzerrungen in der Eingangsstufe des Messgeräts beziehungsweise im Digitizer eines Spektrumanalysators erzeugen. Zumeist ist eine Abschwächung von 50 dB ausreichend. Die nun gemessenen Verzerrungen sind mit hoher Sicherheit bereits im zu messenden Signal enthalten und stammen daher entweder vom Testsignal selbst oder fast vollständig vom zu beurteilenden Audiogerät, wenn das Testsignal hochwertig ist.

Selbstverständlich muss für diese Anwendung das Notchfilter selbst qualitativ gut sein. Es darf möglichst wenig Verzerrungen selbst erzeugen, vor allen Dingen darf seine Eingangsstufe nicht übersteuert werden. Außerdem müssen die Frequenzen von Testsignal und Notchfilter sehr genau übereinstimmen. Hierzu muss vor allem bei einfacheren Notchfiltern ein Feinabgleich der Testfrequenz vorgenommen werden, da die hier verwendeten Notchfilter sehr schmalbandig sind und viele derartige Filter (so das unter [1] beschriebene) nicht in der Frequenz verstellbar ist. Ein Nachteil solcher Lösungen ist, dass man für jede Frequenz des Testsignals ein passendes Notchfilter benötigt, wenn man über die traditionelle Messung mit 1 kHz hinausgehen will.

Einstellbares Notchfilter

Vor allem die Einschränkungen bei den Testfrequenzen haben mich dazu veranlasst, eine neue Filterschaltung zu entwickeln, deren Frequenz bei hoher Qualität einstellbar ist. Damit wird der größte Teil des Audiospektrums abgedeckt. Was das Filter kann, ist unter **Technische Eigenschaften** aufgeführt.

Zunächst zur Blockschaltung von **Bild 1**. Das Eingangssignal wird durch einen vorgeschalteten Abschwächer mit schaltbarer Dämpfung von 0, 6, 12, 18 oder 24 dB auf den für das Notchfilter optimalen Bereich von etwa -10 dB (circa 0,9 V_{SS}) gebracht. Dieser Pegel ist ein optimaler Kompromiss zwischen Rauschen

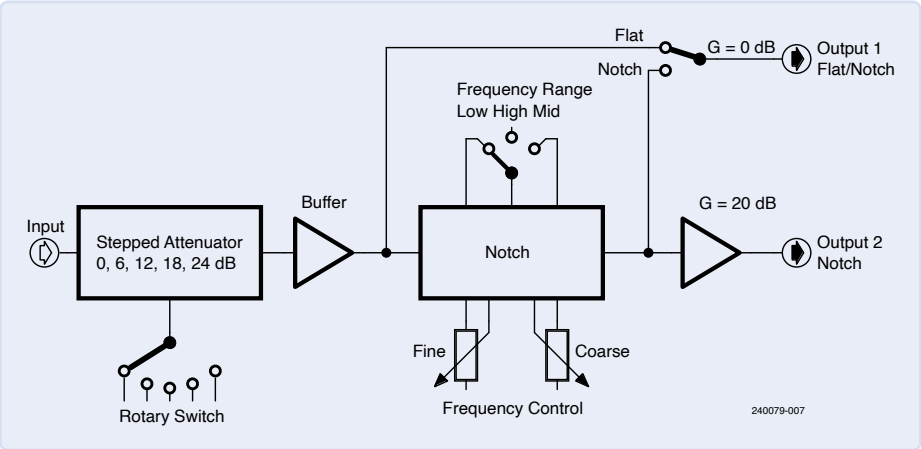


Bild 1. Blockschaltbild des Notchfilters.

Technische Eigenschaften

- Das Eingangssignal kann mit einem schaltbaren Abschwächer um 0, 6, 12, 18 oder 24 dB gedämpft werden.
- Die Notchfrequenz ist mit zwei Potis (*fine* und *coarse*) im Verhältnis 1:7 einstellbar.
- Es gibt drei Frequenzbereiche (*low*, *mid* und *high*), zwischen denen per Schalter gewählt werden kann. Damit wird der größte Teil des Audiospektrums abgedeckt.
- Das Notchfilter lässt sich überbrücken, indem per Schalter zwischen gefiltertem und dem Eingangssignal umgeschaltet werden kann.
- Ein zweiter Ausgang stellt ein um den Faktor 10 verstärktes, gefiltertes Signal zur Verfügung, um auch kleine Testsignale vernünftig messen zu können und/oder den Messbereich für Verzerrungen zu erweitern, indem diese um +20 dB angehoben werden.

Stückliste

Widerstände:
Wenn nicht anders angegeben:
MELF 0204 oder Dünnfilm SMD 1206, 1%
R1, R4, R13, R16, R24 = 100 Ω
R2 = 100 k
R3, R5, R6, R9 = 2k8
R7, R20 = nicht bestückt
R8, R21 = 1k8
R10 = 1k5
R11 = 0 Ω
R12 = 4k22, SMD 0603
R14 = 2k15
R15, R17 = 8k2
R18, R19, R25, R26 = 4k7, SMD 0603
R22, R23, R27, R28 = 10 k, SMD 0603
R29, R30 = 1 k, SMD 0603
R31, R32 = 47 Ω, SMD 1206
R33 = 3k3, SMD 0603
R36 = 1k1
R37 = 10 k
R38 = 220 Ω
R15, R17 = 8k2
P1 = 10 kΩ lin., Stereo-Poti, 4 mm
P2 = 1 kΩ lin., Stereo-Poti, 4 mm

Kondensatoren:
C1, C7 = 82 n, C0G, SMD 0805
C2, C8 = 100 n, C0G, SMD 1206
C3, C9 = 5n6, C0G, SMD 0603
C4, C10 = 22 n, C0G, SMD 0805
C5, C11 = 1n5, C0G, SMD 0603
C6, C12 = 3n9, C0G, SMD 0603
C13, C14 = 2.200 µ / 16 V, Elko, RM 2/10"
C15...C18, C22...C28 = 100 n, X7R, SMD 0603
C19, C20, C29...C36 = 22 µ / 20 V, Tantal, SMC-B
C21 = 220 µ / 16 V, Elko, RM 2/10"

Halbleiter:
D1...D5 = 1N4148, DO214AC
D6...D8 = SK34, DO214AC
B1 = B40C1000, Brückengleichrichter, rund
LED1, LED2 = LED, SMD 0805
T1...T4 = 2N3904, SOT23
IC1...IC3 = OPA2210, SOIC8
IC4 = 7812, TO220
IC5 = 7912, TO220
IC6 = 7805, TO220

Außerdem:
X1, X4 = BNC-Buchse, AMP 227161
X2 = 2-pol. Schraubklemme, RM 5 mm, AK300/2
X3 = 3-pol. Schraubklemme, RM 5 mm, AK300/3
K1...K3, K5, K6, K8 = 2-pol. Stiftleiste, RM 1/10"
K4 = 1×4-pol. Stiftleiste, RM 1/10"
K7 = 3-pol. Stiftleiste, RM 1/10"
K1 = SMD-Relais, 2 x um, TQ2-SA-5V (Form C)
K2...K5 = SMD-Relais, 5 V, 9802-05-00
F1 = Sicherung, 250 mA, Halter 22,5 mm
TR1 = Trafo, 3,6 VA, 2×12 V sek., Gerth 387.24.2
Attentuator = bestückte Platine (siehe Text)
Stufenschalter 1×5 (oder 2×5)
LED 5 mm mit Fassung
Schalter an K4, 1× um, mit Mittelstellung (siehe Text)
Schalter an K2
3 BNC-Buchsen, isoliert, Frontplattenmontage
2 × Drehknopf für Poti mit 4-mm-Achse
1 × Drehknopf für Stufenschalter 6-mm-Achse

und Verzerrungen, die im Notchfilter entstehen. Über einen Buffer gelangt das Signal an den Eingang des Notch und ungefiltert über den Umschalter an den Ausgang 1 (*Flat*/*Notch*). Die Messung des Pegels kann an diesem Ausgang in Schalterstellung *Flat* mit einem angeschlossenen Audio-Analyser, einem geeigneten Digitalvoltmeter oder einem Oszilloskop erfolgen.
Sofern möglich, gibt man den eingestellten Dämpfungswert in den Audio-Analyser ein, um so korrigierte und damit korrekte Pegel zu erhalten. Beim Audio-Analyser QA403 von QuantAsylum ist eine solche Eingabe möglich; bei weniger komfortablen Messgeräten muss man letztlich die gemessenen Pegel noch selber korrigieren.
Der Bereichsumschalter mit seinen drei Stellungen erlaubt die grobe Vorauswahl des Bereichs der Notchfrequenz. Mit der angegebenen Dimensionierung sind folgende Frequenzen einstellbar: *low* = 80...530 Hz, *mid* = 485...3.280 Hz und *high* = 2,99...20 kHz. Mit den beiden frequenzbestimmenden Potis (*fine* und *coarse*) lässt sich die Frequenz in diesen Bereichen im Verhältnis 1:7 variieren. Diese Bereiche kann man mit anderen Bauteilewerten eigenen Vorstellungen anpassen, was allerdings etwas Rechenaufwand erfordert.
Das Ausgangssignal des Notches gelangt entweder über den Umschalter direkt an Ausgang 1 oder aber noch einmal um 20 dB verstärkt an Ausgang 2. Letzteres ist für die Messung kleinerer Signale oder aber zur Erhöhung des Messbereichs bei niedrigen Verzerrungspegeln hilfreich, falls der Audio-Analyser keine kalibrierte Amplitudeneinstellung besitzt.

Die Schaltung

Bild 2 zeigt die detaillierte Schaltung und **Bild 3** das Layout der von mir entwickelten Platine. Der Eingangsabschwächer ist nicht in Bild 2 enthalten. Es handelt sich dabei um eine eigenständige Platine, die huckepack auf die Platine des Notches montiert wird. Die Beschreibung erschien unter dem Namen „Abschwächer für Audiosignale (2)" [2] in Elektor. Davon genügt normalerweise die Single-Ended-Version, wenn man keinen differentiellen Eingang benötigt. Die unterschiedlichen Abschwächungen werden mit einem Drehschalter ausgewählt.

Der Ausgang des Abschwächers ist mit K1 verbunden und gelangt so zum Buffer IC1A. Dessen Ausgangssignal wird über R4 und die Kontakte des Relais Rel1 in Stellung Flat an Ausgang 1 (X1, K3) geführt. Über R5 wird das Notchfilter angesteuert. Dabei handelt es sich um ein rund um IC1B, IC2A, IC2B und IC3A aufgebautes „State Variable Filter". Wichtig ist dabei, dass die frequenzbestimmenden Bauteile bei IC2A und IC3A gleich sind. Das betrifft einerseits die Kondensatoren in den Gegenkopplungszweigen von IC2A und IC3A als auch die Widerstände vor den invertierenden Eingängen dieser Opamps, also die Serienschaltungen aus den 1-kΩ- und 10-kΩ-Potis sowie R8 beziehungsweise R21 (da R7 und R20 nicht bestückt werden).
Bei den Potis muss es sich um Stereoausführungen handeln, da die Integratoren um

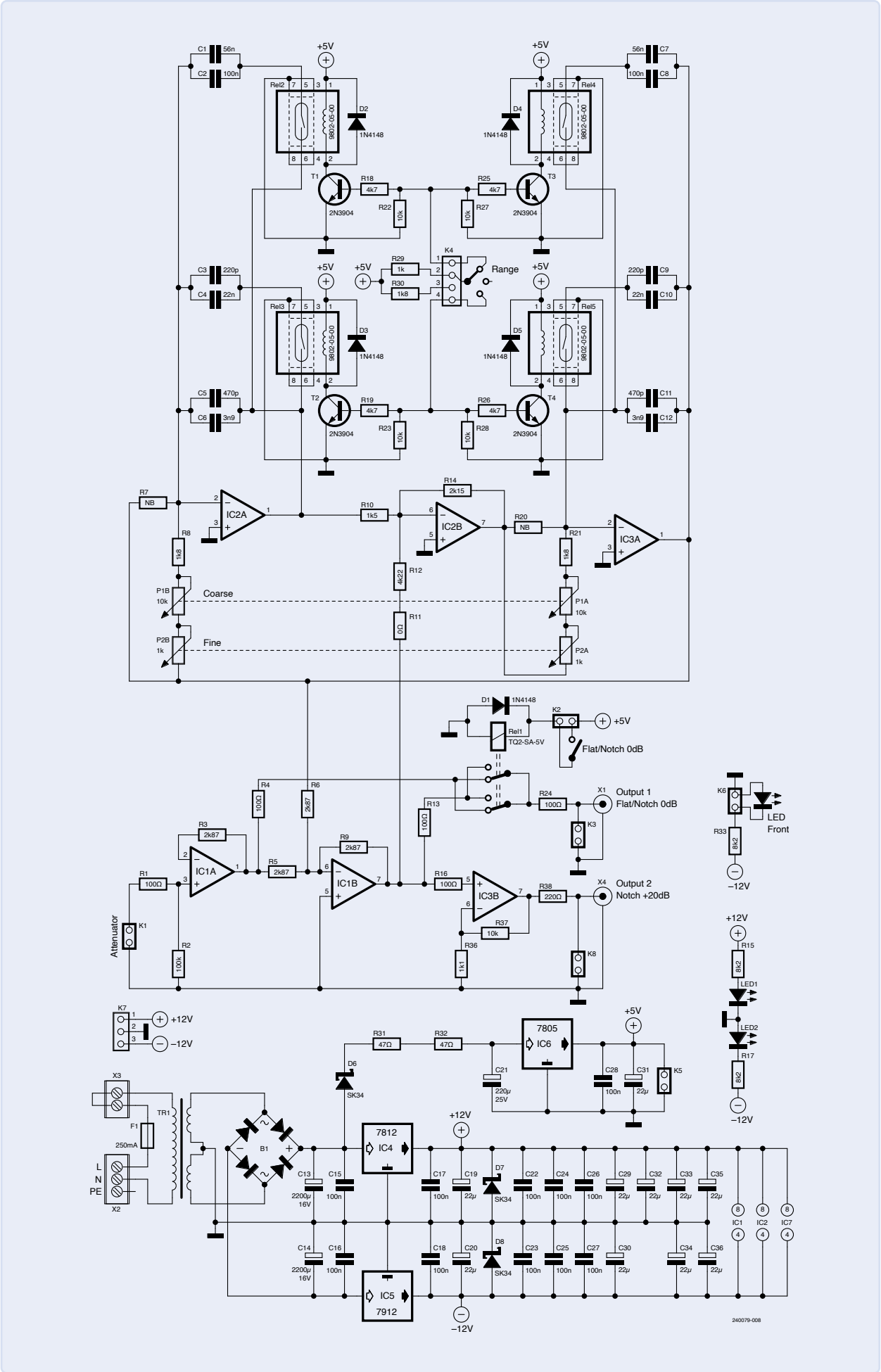


Bild 2. In der Schaltung des Notchfilters ist der Eingangsabschwächer nicht enthalten. Er wird an K1 angeschlossen.

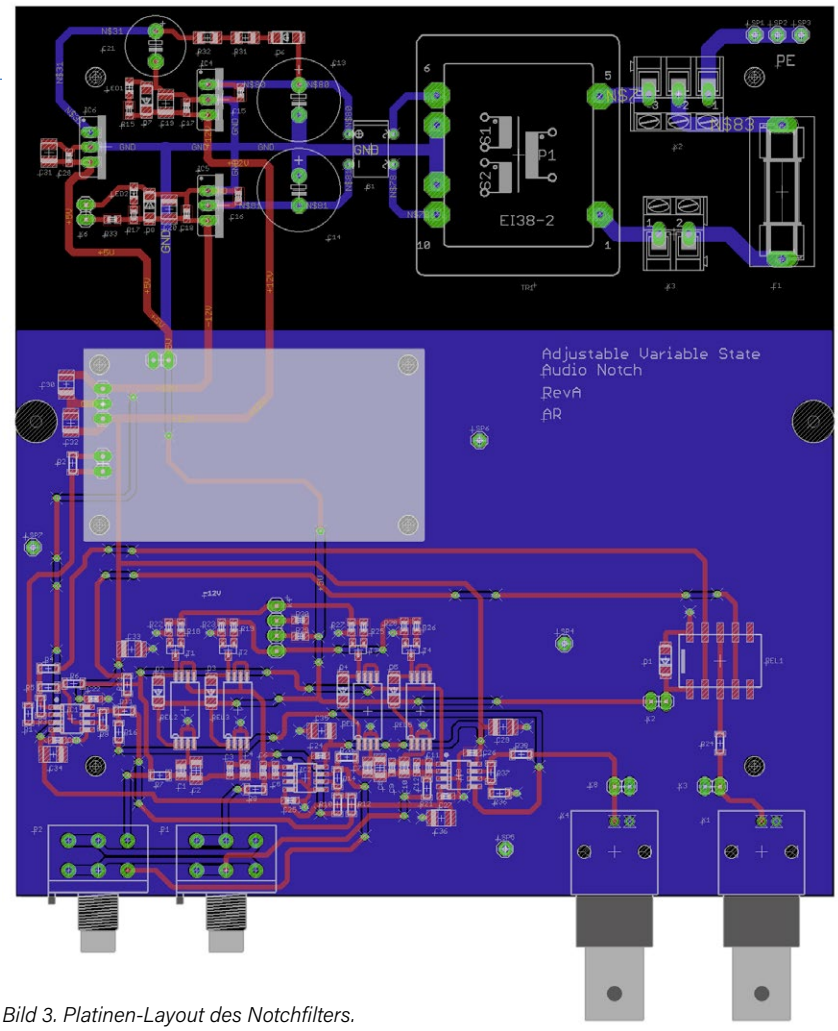


Bild 3. Platinen-Layout des Notchfilters.

IC2A beziehungsweise IC3A immer gleichartig verstellt werden müssen. Wenn man R7 und R20 ignoriert (da nicht bestückt), entspricht bei beiden Integratoren mit auf null gedrehten Potis der minimale Widerstand einfach R8 beziehungsweise R21 = 1,8 kΩ. Sind die Potis voll aufgedreht, addieren sich jeweils noch 11 kΩ dazu. Die Variationsbreite beträgt daher 2,8 kΩ / 1,8 kΩ ≈ 7,11

C5 und C6 beziehungsweise C11 und C12 sind die fest verbauten Kondensatoren bei IC2A beziehungsweise IC3A und bestimmen den höchsten Frequenzbereich. Über die Relais Rel3 beziehungsweise Rel5 werden für den mittleren Frequenzbereich jeweils die parallel geschalteten C3 und C4 (beziehungsweise C9 und C10) parallel zu C5 und C6 (beziehungsweise C11 und C12) geschaltet. Auf gleiche Weise schalten die Relais Rel2 (Rel4) für den unteren Frequenzbereich jeweils die parallel geschalteten C1 und C2 (C7 und C8) parallel zu C5 und C6 (C11 und C12).

Alle Relais sind 5-V-Versionen. Die Relais zur Bereichsumschaltung werden von T1 bis T4 angesteuert. An K4 kann zwecks Einstellung ein Umschalter mit Mittelstellung (wie bei meinem Prototypen) oder ein Drehschalter mit drei Stellungen angeschlossen werden. Rel1, das zum Umschalten zwischen dem ((abgeschwächten und gepufferten) Eingangssignal) und dem Ausgang des Notches dient, wird direkt mit einem an K2 angeschlossen Schalter aktiviert.

Für den zweiten Ausgang mit höherem Pegel wird das Signal vom Notchfilter mit IC3B um den Faktor 10 = 20 dB verstärkt. Andere Verstärkungsfaktoren sind durch Ändern von R36 und/oder R37 möglich. Falls man keine BNC-Buchsen einbauen möchte, kann man an K3 und K8 zum Beispiel auch RCA-Buchsen (Cinch) anschließen.

Die Stromversorgung von ±12 V für die Opamps wird aus dem Trafo TR1, gefolgt von einem Brückengleichrichter, Siebelkos und Spannungsreglern gebildet. LED1 und LED2 zeigen das Vorhandensein der symmetrischen Versorgungsspannungen an. An K6 kann eine zusätzliche LED als Betriebsanzeige für die Frontplatte angeschlossen werden. Über K7 wird auch die Platine des schaltbaren Abschwächers mit ±12 V versorgt.

An den Stromversorgungspins der Opamps sind direkt 100-nF-Abblockkondensatoren platziert. Dazwischen finden sich 22-µF-Tantalkondensatoren, die für saubere Versorgungsspannungen sorgen sollen. Für die 5-V-Versorgung der Relais ist mit IC6 ein eigener Spannungsregler zuständig. Über K5 werden damit auch die Relais des Abschwächers versorgt.

Bau, Bauteile und Qualität

Eine ausgeklügelte Schaltung alleine reicht nicht aus für ein gutes Filter. Die Gesamtqualität hängt entscheidend von der Qualität der Bauteile im Signalweg ab. Die relevanten Widerstände sind daher vom Typ Mini MELF 204, die qualitativ mit bedrahteten Metallfilm-Widerständen mithalten können. Es lassen sich natürlich auch konventionelle und damit preiswertere Dünnfilm-Widerstände im SMD-Format 1206 verbauen. Die Kondensatoren sollten aber zwingend COG-Typen sein, denn ein einziger X7R-Kondensator im Signalweg verdirbt die ganze Schaltung. Als Operationsverstärker benutze ich den Typ OPA2210.

Bild 4 zeigt die bestückte Platine eingebaut in ein Teko-Gehäuse und **Bild 5** die noch nicht perfekte Frontplattenbeschriftung. Sie wurde mit einem Zeichenprogramm entworfen und per Laserdrucker auf eine selbstklebende Folie gedruckt. Die Löcher wurden mit Locheisen ausgestanzt.

Bild 4. Bestückte Platine des Prototyps, eingebaut im Gehäuse.

Bild 5. Die Frontplatte des Notchfilters.

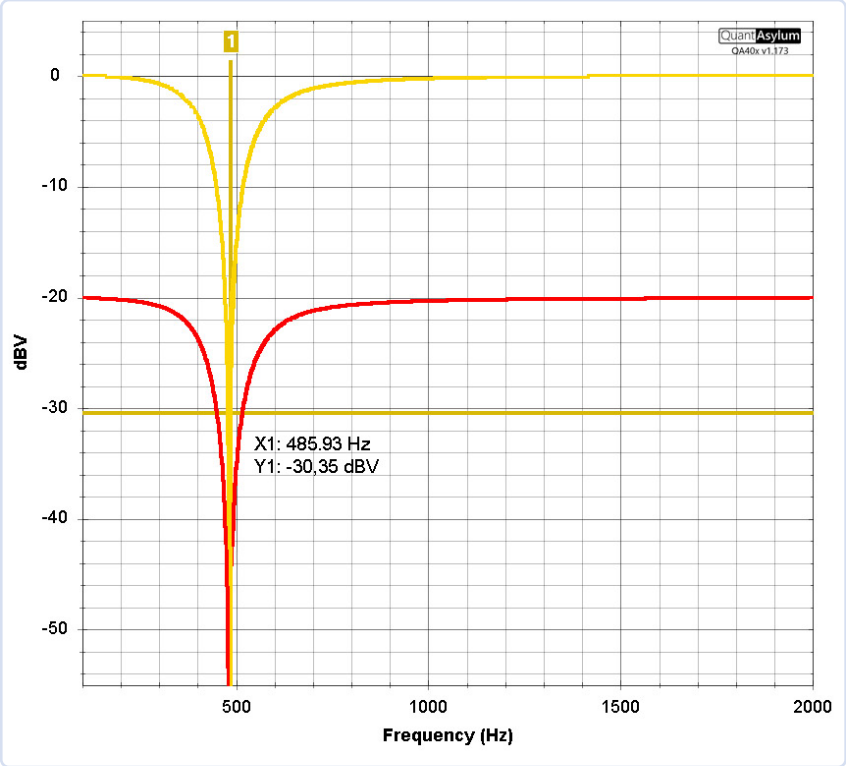


Bild 6. Plot des Frequenzgangs von Ausgang 1 (rot) und Ausgang 2 (gelb).

Bild 6 zeigt den Frequenzgang des Ausgangs 1 (Flat/Notch) in Stellung Notch in rot und des um 20 dB verstärkten Ausgangs 2 in gelb, erfasst mit dem Audio-Analyzer QA403 von QuantAsylum. Das Filter ist wirklich sehr schmalbandig: Bei der doppelten Notch-Frequenz ist die Dämpfung schon nahezu null.

240079-02

Sie haben technische Fragen oder Kommentare?

Schicken Sie bitte eine E-Mail an den Autor unter alfred_rosenkraenzer@gmx.de oder kontaktieren Sie Elektor unter redaktion@elektor.de.

Über den Autor

Alfred Rosenkränzer arbeitete viele Jahre als Entwicklungsingenieur, zu Anfang im Bereich der professionellen Fernsehtechnik. Seit Ende der 1990er Jahre entwickelt er digitale High-Speed- und Analogschaltungen für IC-Tester. Das Thema Audio ist sein privates Steckenpferd.

Passende Produkte

- **QuantAsylum QA403 24-Bit-Audioanalysator**
  www.elektor.de/20530
- **JOY-iT JDS6600 Signalgenerator & Frequenzzähler**
  www.elektor.de/18714

WEBLINKS

[1] Alfred Rosenkränzer, „Fliege-Bandsperre für Audio-Messungen", Elektor 9-10/2022: https://www.elektormagazine.de/magazine/elektor-273/60954
[2] Alfred Rosenkränzer, „Abschwächer für Audiosignale (2)", Circuit Special 2024: https://www.elektormagazine.de/240176-B-02

Das System LeoINAGPS

Nützliche Einblicke in Ihr Elektrofahrzeug

Von Giovanni Carrera (Italien)

Es ist wichtig, den Stromverbrauch eines Fahrzeugs im Laufe der Zeit zu verfolgen, aber es ist noch viel interessanter, dies im Zusammenhang mit der Art der zurückgelegten Strecke zu tun. Dieses Projekt ermöglicht es Ihnen, mit Hilfe eines GPS-Empfängers und Google Maps Informationen über die Streckenführung, den An- und Abstiege und die Höhe zu sammeln und so die grundlegenden elektrischen Parameter des Fahrmotors zu ermitteln.

Ein behinderter Freund von mir benutzt einen Elektroroller, um sich autonom bewegen zu können. Er lebt viele Monate im Jahr in einem kleinen Dorf im ligurischen Apennin, nur wenige Schritte von meinem Haus entfernt. Wir befinden uns in einer Gebirgsgegend und es gibt viele Auf- und Abstiege, auch mit steilen Hängen. Aus diesem Grund müssen die beiden für die elektrische Traktion energiespendenden Blei-Säure-Batterien mit großer Kapazität recht häufig gewechselt werden.

Ich habe mich gefragt, wie viel der Roller unter verschiedenen Bedingungen verbraucht, und sogar darüber nachgedacht, auf leistungsfähigere Batterien wie Lithium-Ionen-Batterien umzusteigen. In diesen Rollern werden schwere Gleichstrom-Permanentmagnetmotoren und Blei-Säure-Batterien verwendet, im Gegensatz zu E-Bikes und Scootern, die kompakte, leichte bürstenlose Motoren und Lithium-Batterien besitzen. Der Roller, mit dem die Tests durchgeführt wurden, ist mit zwei Batterien (12 V/36 Ah) ausgestattet, die kürzlich durch Batterien mit größerer Kapazität (45 Ah) ersetzt wurden.

Zuvor hatte ich bereits ein System zur Messung der Stromaufnahme von Gleichstromgeräten entwickelt und zu diesem Projekt einen GPS-Empfänger hinzugefügt, der die Position und Geschwindigkeit des Fahrzeugs auf einer microSD-Karte speichert. Dieses System ist natürlich auf alle Elektrofahrzeuge anwendbar, sofern der Shunt und der Spannungsadapter entsprechend angepasst werden.

Für das GPS habe ich eigene Funktionen zur Auslesung von Geschwindigkeit, Position und UTC-Zeit entwickelt. Die GPS-Höhenmessung habe ich verworfen, da sie nicht genau genug ist, um Steigungen präzise zu bestimmen. Während der ersten Programmkopilierungen stellte ich fest, dass der Arbeitsspeicher (2 kB) der MCU ATMega328 des Arduino Nano für einen stabilen Betrieb nicht ausreichte. Daher erwog ich, sie durch die MCU ATmega32U4 mit 2,5 kB RAM und einem freien UART zu ersetzen, wie sie im Arduino Leonardo verwendet wird. Der Name dieses Projekts leitet sich deshalb teils vom Namen „Leonardo" (da die Platine Arduino-Leonardo-kompatibel ist), vom Namen des Moduls INA226, das Strom und Spannung an der Last misst, sowie vom GPS-Empfänger ab, der das System vervollständigt. Durch diese Tests konnte ich den Energiebedarfs des Rollers meines Freundes unter verschiedenen Bedingungen auf einer auf Google Maps leicht erkennbaren Strecke ermitteln. Leider konnte die Steigung nicht aus den vertikalen GPS-Koordinaten berechnet werden, da deren Genauigkeit und Auflösung nicht ausreichten. Es gibt zwar Drucksensoren, die die Höhe mit einer Auflösung von wenigen Zentimetern messen können, aber diese wurden in den Tests nicht verwendet.

Beschreibung des Systems

Das Funktionsschema der verschiedenen Bauteile des Systems ist in **Bild 1** dargestellt.

Das Modul INA226

Strommessungen mit einem Shunt sind auf zwei Weisen möglich:

- In Richtung Masse (Low-Side): Der Shunt wird zwischen der Last und der Masse angeschlossen.
- Zur Quelle hin (High-Side): Der Shunt wird zwischen dem Pluspol der Stromversorgung und der Last angeschlossen.

Die integrierte digitale Schaltung INA226 von Texas Instruments kann den Strom über einen High-Side- oder Low-Side-Shunt messen. Zusätzlich ermitttelt sie die Spannung, berechnet die Leistung und gibt einen multifunktionalen Alarm aus. Die Funktion des INA226 ist in **Bild 2** dargestellt.

Der interne Analog-Digital-Wandler (ADC) basiert auf einem 16-Bit-Delta-Sigma-Wandler (ΔΣ) mit einer typischen Wandlungszeit von wenigen Millisekunden, wodurch er auch für schnell wechselnde

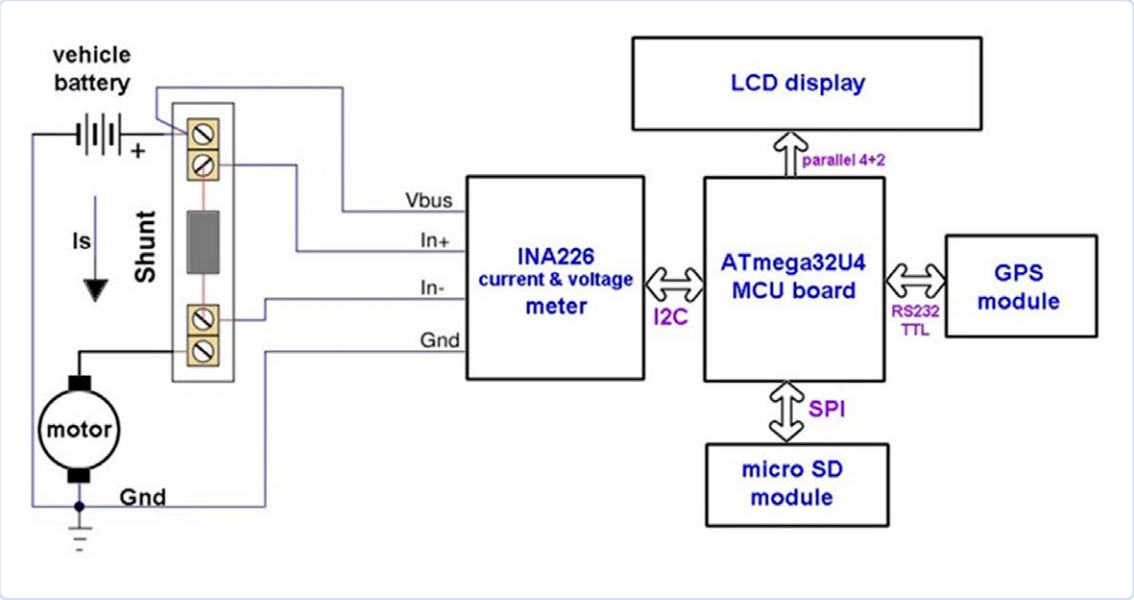


Bild 1. Das Funktionsschema des Systems.

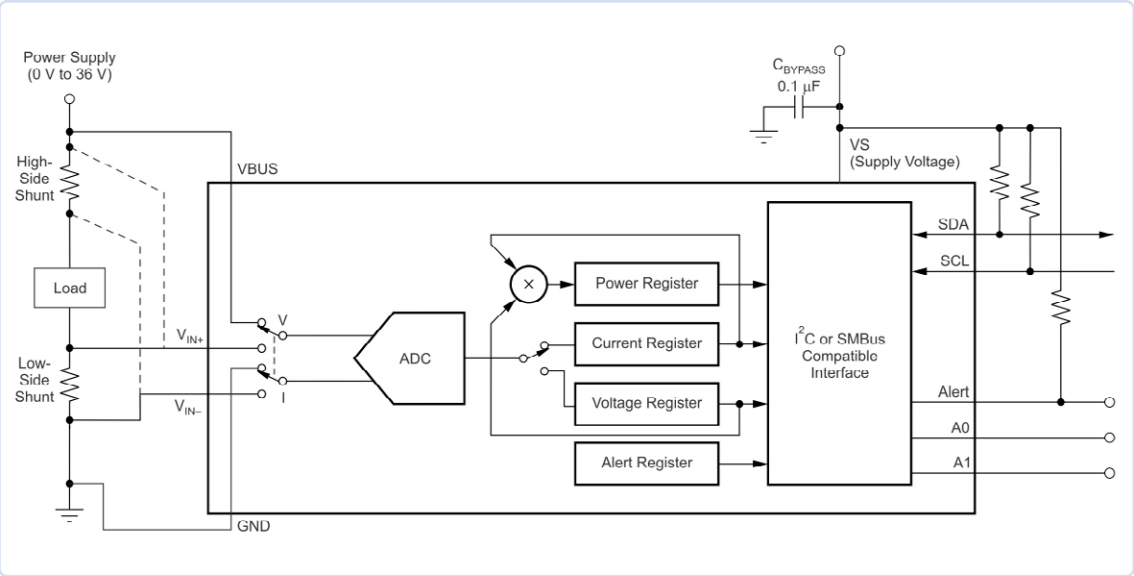


Bild 2. Innenschaltung des INA226. (Quelle: Texas Instruments [1])

Ströme geeignet ist. Darüber hinaus kann er auch negative Ströme messen, wobei negative Werte im Zweierkomplementformat dargestellt werden.

Die Auflösung der Shunt-Spannung beträgt 2,5 µV mit einem Skalenendwert von 32.768 × 2,5 µV = 81,92 mV. Für die Spannung VBUS beträgt die Auflösung 1,25 mV bei einem theoretischen Skalenendwert von 40,96 V, wobei in der Praxis 36 V nicht überschritten werden darf. Die Auflösung der Leistung ist 25-mal so hoch wie die des Stroms, wobei der Skalenendwert vom verwendeten Shunt abhängt. Alles in allem bietet das System also eine bemerkenswerte Messgenauigkeit. Der Chip ist über den I²C-Bus mit einem Mikrocontroller verbunden. Auf dem Markt sind kleine Module auf Breakout-Boards erhältlich, wie das von mir verwendete in **Bild 3**. Der INA226 besitzt zwei Adresspins, A0 und A1, deren Anschluss an V_{CC}, GND, SDA oder SCL die I²C-Adresse des Moduls bestimmt. Bei dem von mir verwendeten Modul sind zwei Pull-Down-Widerstände eingebaut, wodurch die Adresse auf 0x40 eingestellt ist. Auf der Rückseite des Moduls befinden sich Pads, mit denen man die Adresse ändern kann: Durch Verbindung mit V_{CC}, SDA oder SCL können bis zu 16 verschiedene Adressen eingestellt werden, was besonders nützlich für die Überwachung einer Batterie mit vielen Zellen ist.

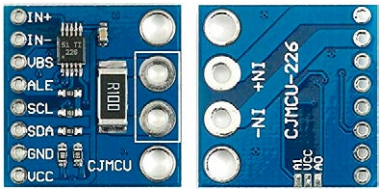


Bild 3. Das INA226-Modul.

Das Modul ist standardmäßig mit einem 0,1-Ω-Shunt (R100) ausgestattet, der die Messung eines maximalen Stroms von 0,8192 A ermöglicht. Ich habe natürlich den Shunt entfernt, an seiner Stelle einen 1-µF-Kondensator als Rauschfilter eingelötet und meinen eigenen Shunt verwendet (mehr dazu unten).

Das Modul ATmega32U4 Pro Micro

Das verwendete Pro-Micro-Modul besitzt eine MCU des Typs ATmega32U4 und ist Arduino-Leonardo-kompatibel. Es kann mit der Arduino-IDE programmiert werden, die auf das Leonardo-Board eingestellt ist. Das Ganze ist mit etwa 33 mm × 18 mm winzig, noch kleiner als der Arduino Nano, was in etwa der Größe eines alten 24-poligen DIL-EPROMs entspricht. Wenn Sie keine Platzprobleme haben, können Sie stattdessen auch ein „richtiges" Arduino-Leonardo-Board verwenden.

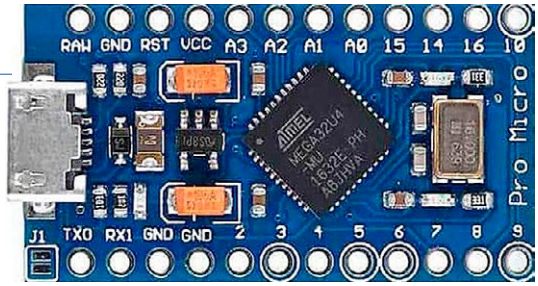


Bild 4. Der ATmega32U4-auf dem Pro-Micro-Modul.

Bild 6. Das LCD.

Bild 4 zeigt die für dieses Projekt eingesetzte Platine, die über einen 5-V-Spannungsregler verfügt und deshalb nicht nur über USB, sondern auch über den RAW-Eingang (6...12 V) mit Energie versorgt werden. Wird der RAW-Pin mit einer Mindestspannung von 6 V versorgt, liefert der Regler eine V_{CC} von +5 V. Bei der Stromversorgung über USB stehen am RAW-Pin etwa +4,8 V zur Verfügung, was durch den Spannungsabfall an einer Schottky-Diode, die zusammen mit einer Sicherung als Schutz dient, verursacht wird. Möchte man den internen Regler verwenden, der keinen Kühlkörper besitzt, sollte man die Eingangsspannung auf 6...7,2 V begrenzen und Lasten über 100...150 mA vermeiden. Wenn wie in unserem Fall ein externer Spannungsregler zur Verfügung steht, kann die Platine auch direkt über den VCC-Pin mit +5 V betrieben werden, wodurch der interne Regler umgangen wird.

Das MicroSD-Modul

Ich habe ein SD-Modul ausgewählt, das speziell für Arduino entwickelt wurde und mit 5-V-Logikpegeln sowie einer 5-V-Stromversorgung arbeitet. Das Modul ist in **Bild 5** zu sehen. Es ist mit einem 3,3-V-Spannungsregler und einem Pegelwandler ausgestattet. Module ohne Pegelwandler und 3,3-V-Spannungsregler sind für dieses Projekt nicht geeignet. Für die Verwaltung der SD-Karte habe ich die SD-Bibliothek von Arduino verwendet.

Das LCD

Ich habe ein gängiges zweizeiliges 16-Zeichen-LCD mit einem Hitachi HD44780-kompatiblen Controller und Hintergrundbeleuchtung aus hocheffizienten LEDs verwendet. Diese LEDs zogen ursprünglich etwa 20 mA, was ich durch einen externen Widerstand (in Reihe mit dem internen Widerstand) auf 10 mA reduzieren konnte. In **Bild 6** ist das Display zu sehen, das über eine parallele Schnittstelle mit vier Daten- und zwei Steuerbits an die MCU angeschlossen ist.

Das GPS-Modul

Auf dem Markt gibt es zahlreiche preiswerte GPS/GNSS-Satellitenempfänger. Wichtig ist, dass sie mit 5 V betrieben werden können und die Baudrate der seriellen Schnittstelle bekannt ist, die normalerweise 4.800 bit/s oder 9.600 bit/s beträgt. Einige Module besitzen eine eingebaute Antenne, während andere eine externe Antenne verwenden. Für dieses Projekt habe ich einen älteren Empfänger mit einer Baudrate von 4.800 bit/s, einer integrierten Antenne, einem Magnetfuß und einem 4-poligen JST-Steckverbinder zur Verbindung mit dem System verwendet.

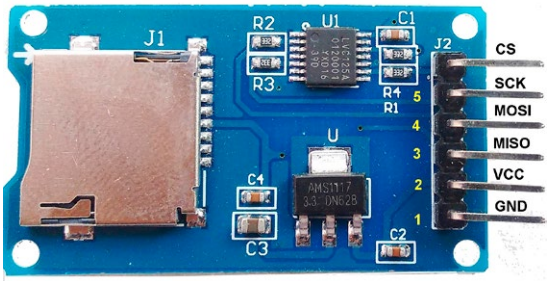


Bild 5. Das microSD-Modul.

Schaltplan mit Modulen

Bild 7 zeigt den Schaltplan des Systems. Neben dem Anschluss für die serielle Schnittstelle des GPS-Moduls habe ich auch einen Schalter eingebaut, mit dem man eine Leistungsmessung mit oder ohne GPS auswählen kann. Die Taste *SS* startet und stoppt die Datenerfassung auf der microSD-Karte. Der Spannungsregler LM7805 könnte theoretisch entfallen, wenn wir den auf der Platine vorhandenen Regler verwenden. Der externe Regler ist jedoch zuverlässiger und benötigt eine Mindestspannung von 7 V und kann - mit einem Kühlkörper ausgestattet - Spannungen bis zu 24 V verarbeiten. Die Stromaufnahme des Geräts (inklusive Hintergrundbeleuchtung) beträgt etwa 47...50 mA. Die Schottky-Diode D1 verhindert einen Konflikt, der entstehen würde, wenn sowohl die externe Stromversorgung (5Ve) als auch die USB-Versorgung gleichzeitig angeschlossen sind. Die Widerstände R1 und R2 bilden zusammen mit dem Kondensator C5 ein Tiefpassfilter erster Ordnung zur Rauschunterdrückung.

Shunt-Berechnung

Bei hohen Strömen ist es ratsam, den Shunt nicht selbst zu bauen: Im Handel sind Hochstrom-Shunts (mit Kelvin-Schaltung) leicht erhältlich; sie sind auch sehr genau, typischerweise im Bereich von 0,5 %, und sehen ähnlich aus wie in **Bild 8**. Ein recht gängiger Spannungswert an den Ausgangsklemmen des Shunts beträgt 75 mV für den gesamten Messbereich. Diese Spannung liegt sehr nahe an der maximalen Eingangsspannung des INA226 von 81,92 mV. In meinem Fall wollte ich Ströme bis zu 50 A messen, dessen Widerstand sich für die maximalen 75 mV wie folgt berechnen lässt:

R_{se} = 75 mV / 50 A = 1,5 mΩ

Der Vollausschlag des Stroms ist dann:

I_{fs} = 81,92 mV / R_{se} = 54,613 A

Die Auflösung des Stroms ist:

54,613 / 32.768 = 1,666 mA

Die maximale Leistung, die vom Shunt abgeleitet wird, beträgt:

$P = U \times I = U^2 / R = 4{,}474$ W

Bild 8. Der 50-A-Shunt.

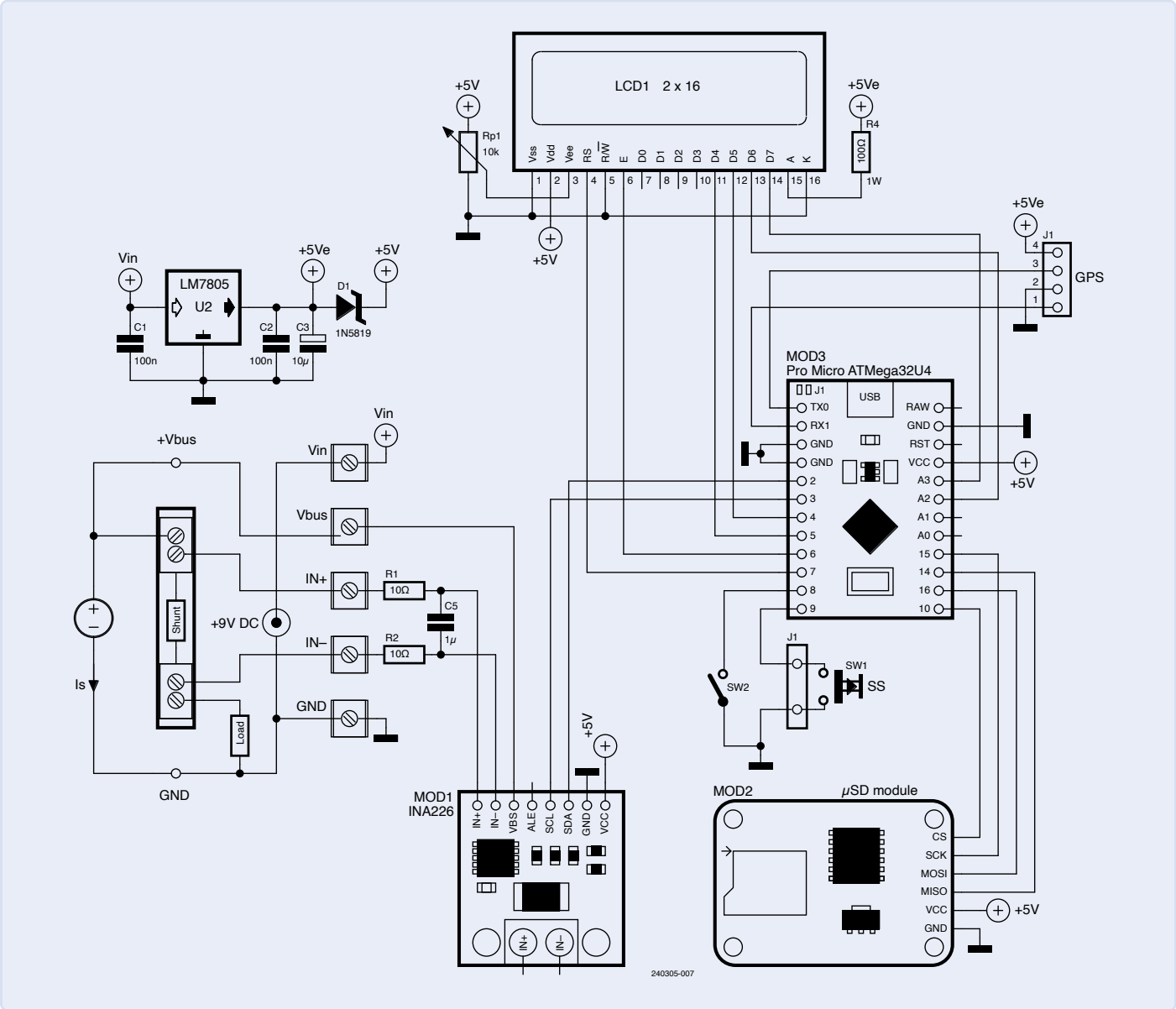


Bild 7. Das Schaltbild des gesamten Systems.

GPS-Daten

Für die Interpretation der NMEA-Meldungen, die vom GPS-Empfänger gesendet werden, habe ich selbst geschriebene Funktionen verwendet. Das Programm liest die NMEA-RMC-Sätze (Recommended Minimum Specific Data), da sie neben den Positionskoordinaten auch die Geschwindigkeit in Knoten sowie Datum und Uhrzeit enthalten. Ein Beispiel für einen solchen Satz wäre:

```
$GPRMC,141507.870,A,4429.2796,N
,00907.4338,E,0.00,,180522,,*1B
```

Das bedeutet UTC-Zeit 14:15:07,870, gültige Daten, Breitengrad 44°29,2796′ N, Längengrad 9°07,4338′ E, Geschwindigkeit 0 Knoten, kein Kurs, Datum 18/05/2022, keine Modusanzeige. Die Parameter sind in **Tabelle 1**

Tabelle 1: Die NMEA-RMC-Felder

i	Feld	Name	Beschreibung
1	hhmmss.sss	UTC-Zeit	im Format hhmmss.sss format (000000.000 ~ 235959.999)
2	x	Status	"V" = Navigations-Empfänger-Warnung "A" = Gültige Daten
3	1111.1111	geogr. Breite	Im Format ddmm.mmmmmm format, einschließlich führender Nullen
4	A	N/S-Indikator	"N"=Nord, "S"=Süd
5	yyyyy.yyyy	geogr. Länge	Im Format ddmm.mmmmmm, einschließlich führender Nullen
6	A	E/W-Indikator	"E"=Ost, "W"=West
7	x.x	Geschwindigkeit auf Erde	Ausgedrückt in Knoten (000.0 ~ 999.9)
8	u.u	Bewegungsrichtung auf Erde	Ausgedrückt in Grad (000.0 ~ 359.9)
9	xxxxxx	UTC-Daten	Im Format ddmmyy
10	v	Modus-Indikator	"N" = Ungültige Daten "A" = Autonomer Modus "D" = Differentieller Modus
11	hh	Checksumme	Fehlerprüfung, eingeleitet mit Asterix (*)

aufgeführt. Die Koordinaten werden von Grad und Minuten in Dezimalgrade umgewandelt. Die auf der microSD-Karte gespeicherten Daten sind wie folgt aufgebaut:

```
,time,latitude,longitude,speed
```

Und im Falle des vorherigen Beispiels:

```
,141507,4429.2796,00907.4338,0.00
```

INA226 Verwaltungssoftware

Ich habe eine modifizierte Version der Bibliothek *INA226.h* von Korneliusz Jarzebski verwendet, da das Original einige Fehler enthielt. Ich habe die Funktionen `calibrate(float rShunt, float iMaxExpected)` und `setShuntVoltageLimit(float voltage)` geändert. Vielleicht hat der Autor die Bibliothek inzwischen aktualisiert, aber ich empfehle, die Version zu verwenden, die auf der Elektor Labs-Webseite für diesen Artikel [2] zum Download bereitsteht, da sie mit diesem System getestet wurde.
Die Funktion `calibrate()` hat als Eingabeparameter den Shunt-Widerstand und den maximalen Strom, in unserem Fall:

`rShunt = 0.0015 Ω` und `iMaxExpected = 54.613 A`

Es werden mehrere Variablen berechnet:

`currentLSB = iMaxExpected / 32768` = 54,613 / 32.768 = 1,666 [mA]
`calibrationValue = 0.00512 / currentLSB / rShunt`
= 0,00512 / 0,00166 / 0,0015 = 2.048
`powerLSB = currentLSB * 25 = 0.000625 * 25` = 15.625 [mW]

Die Datenformate von Display und microSD

Unter Berücksichtigung der Auflösung der Variablen, die auf dem Display angezeigt werden sollen, sind dies ihre Formate:

Bus-Spannung:	`V=xx.xx`	(sieben Zeichen)
Shunt-Strom:	`I=xx.xxx`	(acht Zeichen)
Bus-Leistung:	`W=xxx.xx`	(acht Zeichen gedruckt)

In der ersten Zeile werden daher Spannung und Strom angezeigt, getrennt durch ein Leerzeichen. In der zweiten Zeile folgt die Leistung, etwa so:

```
V=xx.xx I=xxx.xxx
W=xx.xx
```

Falls GPS verwendet wird, werden in der zweiten Zeile die Geschwindigkeit in km/h und die UTC-Zeit angezeigt.

Messungen der Energieaufnahme

Der String `dataString`, der auf SD gespeichert wird, ist wie folgt aufgebaut:

```
Vbus,Ishu,power,Vshu
```

Dabei steht `Vbus` für die Batteriespannung, `Ishu` für den durch den Shunt fließenden Strom, `power` für deren Produkt und `Vshu` für die Spannung

Bild 9. Der Elektroroller mit der Messausrüstung und mein Freund Roberto, der die Änderungen an seinem Fahrzeug sichtbar besorgt zur Kenntnis nimmt.

an den Enden des Shunts. Wenn kein GPS an Bord ist, sind dies die einzigen Messungen für jede Probenahme; wenn GPS vorhanden ist, werden auch Zeit-, Positions- und Geschwindigkeitsdaten hinzugefügt.

LeoINAGPS-Systemtests

Für diese Tests habe ich den oben beschriebenen 50-A-Shunt verwendet, der der Klasse 0,5 % entspricht. Diese Tests wurden in einem kleinen Dorf im ligurischen Apennin durchgeführt, und die Messgeräte wurden auf einem Elektromobil montiert, das in **Bild 9** zu sehen ist und von seinem Besitzer, meinem Freund Roberto, gefahren wird. Um Platz für die Messgeräte zu schaffen, wurde die Verkleidung unter dem Fahrersitz entfernt.
Die Teststrecke begann im Garten des Hauses meines Freundes. Der erste Teil der Strecke war durch mehrere mehr oder weniger steile Abfahrten gekennzeichnet, während der Rückweg über eine andere Straße mit ziemlich steilen Anstiegen führte. Es gab jedoch auch mehrere Abschnitte, die in beide Richtungen befahren wurden. Die Strecke war relativ kurz, wies aber deutliche Steigungen auf, die den Motor dieser meist in China gefertigten Motorroller stark beanspruchten. Die Roller sind mit einem konventionellen Gleichstrommotor von etwa 1.000 W ausgestattet, der von zwei in Reihe geschalteten 12-V-Blei-Säure-Batterien gespeist wird, die für den elektrischen Antrieb geeignet sind. Der Schutz erfolgt über eine 40-A-Sicherung und einen 30-A-Thermoschalter. Der eingebaute elektronische MOSFET-Controller steuert den Motor im PWM-Modus. Die ursprünglichen Batterien hatten eine Kapazität von 35 Ah, wurden aber durch Batterien mit 45 Ah ersetzt, um die Reichweite auf unebenem Gelände wie im Testgebiet zu erhöhen. Diese Batterien haben in solch bergigem Gelände eine relativ kurze Lebensdauer und müssen fast jedes Jahr ausgetauscht werden. Die angegebene Kapazität von 45 Ah wurde auf der Grundlage einer 20-stündigen Entladung (I = 2,25 A) berechnet, während sie sich bei einer 5-stündigen Entladung, die den realen Bedingungen näher kommt, auf 35 Ah (I = 7 A) verringerte.
Mit Ausnahme des GPS wurden die Instrumente unter dem Sitz des Rollers und über den Batterien angebracht, wie in Bild 8 und **Bild 10** zu sehen ist. Das LeoINAGPS-System wurde von zwei in Reihe geschalteten Lithiumbatterien (7,2 V) versorgt.

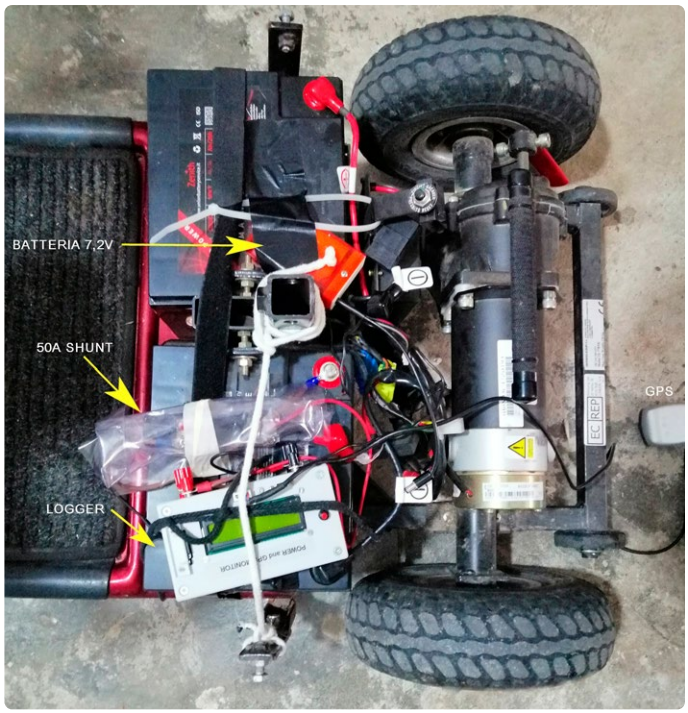


Bild 10. Platzierung der Messgeräte an Bord.

Der GPS-Empfänger wurde samt Antenne auf einem speziellen Ständer montiert, damit er den Fahrer nicht behindert (siehe Bild 9). GPS-Empfänger liefern in der Regel jede Sekunde eine Positionsmessung, sodass das System mit dem Empfang von NMEA-Nachrichten synchronisiert werden muss. Jede Messung, die auf der microSD-Karte gespeichert wird, besteht aus acht Daten: Batteriespannung [V], gelieferter Strom [A], Leistung [W], Shunt-Spannung [V], UTC-Zeit, Breitengrad, Längengrad und Geschwindigkeit [km/h]. Die Koordinaten werden in Grad und Zehntelgrad angegeben.
Bevor die Erfassung durch Drücken der Taste auf der Vorderseite des Loggers gestartet wird, muss man warten, bis das GPS das Signal von mindestens drei Satelliten empfangen und verarbeitet hat, was ein paar Minuten dauern kann. Wenn das GPS in Betrieb ist, beginnt die LED zu blinken, und die UTC-Zeit erscheint auf dem Display.
Die italienische Ortszeit liegt während der Sommerzeit (MESZ) eine Stunde und während der Normalzeit (MEZ) zwei Stunden vor der UTC-Zeit. Nach Abschluss der Tests muss die Taste erneut gedrückt werden, um die Erfassung zu beenden. Der Test verlief sehr gut und dauerte 963 Sekunden, also etwa 16 Minuten.

Messungen

Bild 11 zeigt die Zeitverlaufsdiagramme der Batteriespannung, des gelieferten Stroms, der Leistung und der Geschwindigkeit des Fahrzeugs als Funktion der Zeit, ausgedrückt in Sekunden. Der Abschnitt, in dem die Geschwindigkeit gleich Null ist, ist darauf zurückzuführen, dass das Fahrzeug angehalten hat, um das Foto in Bild 9 zu machen. Bei Abfahrten wirkt der Bürstenmotor wie ein Dynamo, und in diesen Abschnitten kehrte sich der Strom um und die Spannung überschritt einige Male 32 V, was für die Batterien nicht gerade ideal ist.
Wichtige Informationen erhält man aus dem Zeit-Strom-Diagramm, in dem man leicht die verschiedenen Abfahrten erkennen kann, die durch die negativen Stromwerte gekennzeichnet sind: die erste Abfahrt (Punkte 89...112), dann eine kleine Bodenwelle und der Beginn der zweiten Abfahrt (Punkt 136), gefolgt vom Stopp (Punkte 148...186) und der Fortsetzung mit verschiedenen Abfahrten (Punkte 189...300), die längste, die dritte (Punkte 660...683), und die vierte, intensivste Downhill-Strecke (Punkte 759...786).
Wie zu erwarten war, trat der höchste Verbrauch an den Steigungen auf, mit Spitzen von mehr als 800 W. Man kann die Stromspitzen beim Anlassen des Motors oder beim Anhalten des Rollers (Bremsen) sehen. Die typische Geschwindigkeit während des Tests betrug etwa 4...5 km/h mit einigen Spitzen von 6 km/h, was praktisch Schritttempo entspricht. Dies ist darauf zurückzuführen, dass es keine langen flachen Abschnitte gab. Die Hinfahrt endete am Punkt 372; die Rückfahrt erfolgte auf einer weitgehend anderen Strecke und war zudem länger. Der Zeitpunkt der Umkehr ist im Geschwindigkeitsdiagramm als Spitze von 5,2 km/h auf 2 km/h zu erkennen. Dieses Manöver lässt sich noch besser aus dem Längengraddiagramm in **Bild 12** ablesen.

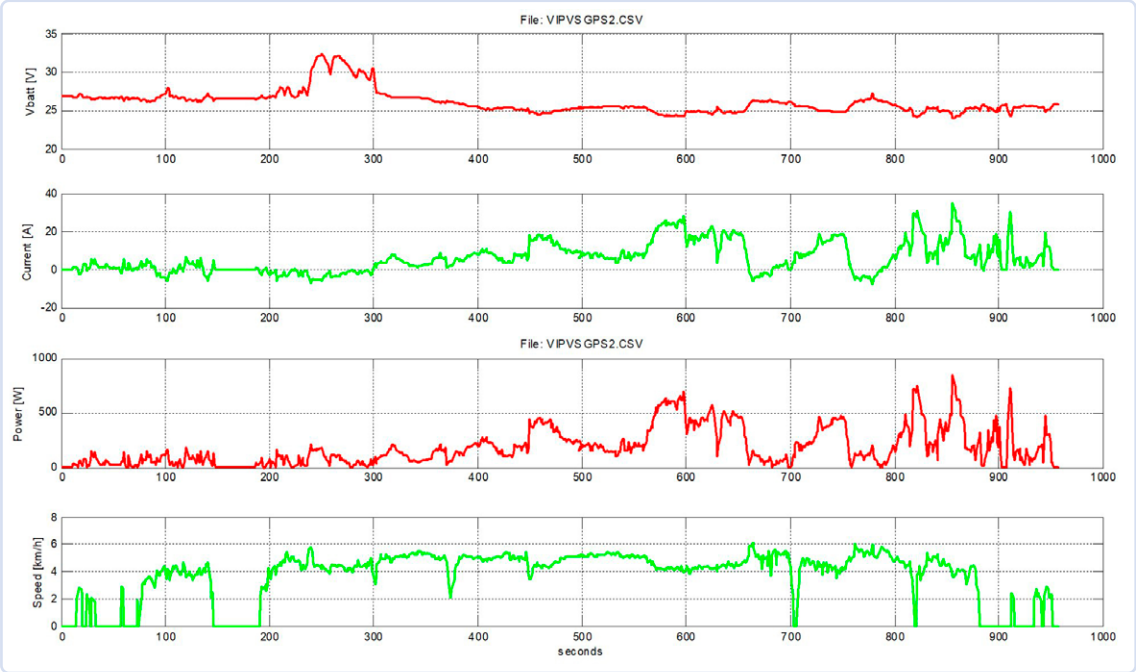


Bild 11. Zeitverlauf-Diagramm der Hauptparameter.

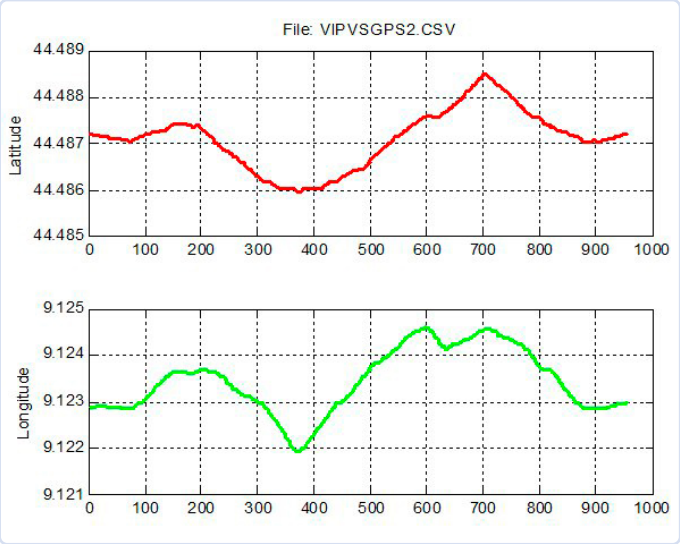


Bild 12. Diagramm der X/Y-Koordinaten.

Nach einer ziemlich steilen Steigung bog der Roller in eine kleine Straße ein, die er hundert Meter lang befuhr und dann wieder zurückkehrte. Diese Umkehr, die am Punkt 704 stattfand, ist im Breitengraddiagramm zu sehen. Aus diesen Kurven geht hervor, dass die Koordinaten des Endpunktes dem Startpunkt ähneln.

Import der Route in My Maps

Um die Route über den Satellitenfotos zu sehen, können Sie My Maps [3] und Google Maps verwenden, die online und kostenlos verfügbar sind. Sie benötigen ein Google-Konto und müssen die folgenden einfachen Schritte ausführen:

- Melden Sie sich mit Ihrem Computer bei *My Maps* an.
- Klicken Sie auf *+EINE NEUE KARTE ERSTELLEN*.
- Klicken Sie in der Kartenlegende auf *Ebene hinzufügen*.
- Geben Sie der neuen Ebene einen Namen, wenn Sie möchten.
- Klicken Sie in der neuen Ebene auf *Importieren*.
- Wählen Sie die Datei *VIPVSGPS.csv* aus oder laden Sie sie hoch und klicken Sie auf *Auswählen (Select)*
- Wählen Sie die Spalten mit den Breitengraden (erster Wert), den Längengraden (zweiter Wert) und den Bezeichnungen für die einzelnen Punkte (zum Beispiel Leistung) aus.

Wenn Sie die Route mit Google Maps anzeigen und das Satellitenfoto überlagern möchten, importieren Sie mit diesen einfachen Schritten diese Ebene. Das Ergebnis ist in **Bild 13** zu sehen, und **Bild 14** zeigt den Ausschnitt, an dem die Route beginnt und endet. Mit diesem Programm können Sie neben jedem Punkt einen bestimmten Wert eingeben. In diesem Fall habe ich die zum jeweiligen Zeitpunkt gemessene Leistung des Motors angegeben.

Energieberechnung

Um eine Vorstellung von der Reichweite zu bekommen, ist es wichtig zu wissen, wie viel Energie aus den Batterien entnommen wurde. Die Menge an elektrischer Energie ΔE, die von den Batterien an den Motor geliefert wird, hängt von der abgegebenen Leistung und der Zeit $\Delta t = t_2 - t_1$ ab. Wenn die Leistung P über die Zeit konstant wäre, ließe sich die Energie leicht berechnen:

$$\Delta E = P \times \Delta t \text{ [J]} \quad (1)$$

Im internationalen System wird die Energie in Joule ausgedrückt (1 J = 1 W/s), doch für elektrische Systeme verwenden wir lieber

Bild 13. Jetzt geht's rund! (Quelle: Google Maps)

Bild 14. Detail der Start- und Endpunkte des Tests. (Quelle: Google Maps)

Wattstunden (Wh), die durch 3.600 geteilt werden. In unserem Fall ist die Leistung aber keineswegs konstant, sondern schwankt, so dass wir von einer diskreten zu einer kontinuierlichen Formel (1) übergehen müssen:

$$E = \int_{t_1}^{t_2} P(t)dt$$

Eine grundlegende numerische Methode zur Durchführung einer Integration ist die der Trapezoide:

$$E = \int_{t_1}^{t_2} P(t)dt \approx (t_2 - t_1)\frac{P(t_1) + P(t_2)}{2}$$

Dieses System ist zwar nur näherungsweise genau, aber es kann die Fläche unter einer Funktion P(t), die das definite Integral darstellt, durch eine Vielzahl von Trapezen ersetzen, deren Flächeninhalte sich leicht berechnen lassen, wie es in **Bild 15** zu sehen ist. Diese Methode ist umso genauer, je kleiner das Δt ist. In unserem Fall beträgt die Abtastperiode Δt eine Sekunde, also:

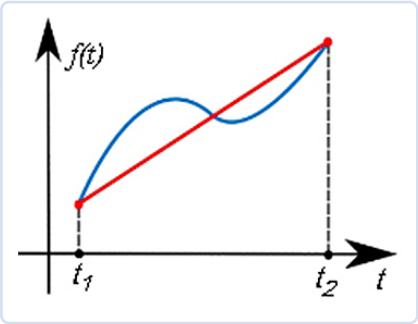


Bild 15. Integration durch Trapeze.

$E_2 = (P_1 + P_2) \times 1s / 2 + E_1$ [J]

Mit einer Tabellenkalkulation ist es einfach, die Energie in Joule und, durch Division mit 3600, in Wattstunden zu berechnen. Um die ungefähre Reichweite des Rollers abzuschätzen, müssen wir die von den Akkus entnommene Ladung *C*, ausgedrückt in Amperestunden (Ah), berechnen, indem wir den Strom integrieren, ähnlich wie bei der Energie:

$C_2 = (I_1 + I_2) \times 1s / 2 + C_1$ [Ah]

Die Ladung steht in Zusammenhang mit der Energie durch die Spannung. Die Energie in Wh erhält man durch Multiplikation der Ladung mit der durchschnittlichen Spannung. Die Ergebnisse der Testdaten, die in Matlab verarbeitet wurden, sind:

Verarbeitung der Messungen des LeoINAGPS-Systems:
- Name der verarbeiteten Datei = VIPVSGPS2.CSV
- Anzahl der erfassten Proben = 963
- Abtastperiode = 1,000 [s]
- Dem Motor zugeführte Energie = 48,943 [Wh]
- „Verlorene" Kapazität (P und N) = 1,650 [Ah]
- „Verlorene" Kapazität (P) = 1,793 [Ah]
- Durchschnittliche Batteriespannung = 26,103 [V]
- Durchschnittlicher Entladestrom = 6,206 [A]
- Kapazität (E/U durchschnittlich) = 1,875 [Ah]

In diesem Fall wird die verlorene Kapazität (positiv und negativ) unterschätzt, zum einen aufgrund des zu hohen Δt und zum anderen wegen der Abschnitte, in denen der Strom negativ ist (Steigungen), die nicht effektiv zum Laden der Batterie beitragen und letztendlich eher schädlich als nützlich sind. Zusätzlich wurden die Kapazität, die nur mit positivem Strom (P) berechnet wurde, die durchschnittliche Batteriespannung und der durchschnittliche Strom ermittelt.

Das Programm

Beim Start initialisiert das Programm die Ein- und Ausgänge, das LCD, das INA226-Modul, überprüft das Vorhandensein der microSD-Karte und liest die Position des GPS-Schalters aus. In setup()wird die serielle Schnittstelle mit einer Baudrate von 4.800 kbit/s initialisiert:

```
Serial1.begin(4800); // initialize UART with GPS baud rate
```

Falls Ihr GPS-Modul moderner ist und eine Baudrate von 9600 kbit/s verwendet, müssen Sie diesen Funktionsparameter entsprechend ändern.
Wenn der Schalter auf GPS gestellt ist, sendet das Programm die Meldung „*Wait for GPS ..*" an das Display und wartet auf ein gültiges Signal vom GPS-Empfänger. Die Funktion `GetData()` extrahiert die Zeit, die Position und die Geschwindigkeit aus den NMEA-RMC-Datensätzen.
Wenn die Daten gültig sind (*Status = „A"*), werden die Geschwindigkeit und die Zeit (hhmmss) in der zweiten Zeile des Displays angezeigt. Diese Funktion konvertiert zudem die Koordinaten, die ursprünglich in Grad und Minuten angegeben sind, in Dezimalgradnotation.

SE – 240305-02

Über den Autor

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

Sie haben Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Dann wenden Sie sich bitte an den Autor unter giov.carrera@gmail.com oder an die Elektor-Redaktion unter redaktion@elektor.de.

Passende Produkte

- **OPEN-SMART GPS - Serielles GPS-Modul für Arduino**
  www.elektor.de/18733
- **PeakTech 4350 Stromzange**
  www.elektor.de/18161

WEBLINKS

[1] INA226 High-Side or Low-Side Measurement, TI, SBOS547A, August 2015: https://www.ti.com/lit/ds/symlink/ina226.pdf
[2] Elektor-Labs-Seite für diesen Artikel: http://www.elektor.de/labs/the-Leo-INA226GPS-system
[3] Webseite Google My Maps: https://mymaps.google.com/

Solarbetriebener LoRa-Knoten

Eine modulare, kompakte und vielseitige IoT-Lösung

Von Dr. Claus Kühnel (Schweiz)

Autonom arbeitende Sensorknoten benötigen neben der drahtlosen Kommunikation auch eine autonome Energiequelle. In diesem Artikel stellen wir Ihnen einen batteriebetriebenen, durch eine Solarzelle gepufferten IoT-Knoten mit LoRaWAN-Konnektivität vor. Wir konzentrieren uns dabei auf die Stromversorgung und die Fernüberwachung der verschiedenen Spannungen über Bluetooth und LoRa. Da aber viele verschiedene Sensormodule verfügbar sind und es auch eine gute Softwareunterstützung gibt, können Sie das Projekt leicht an Ihre spezifischen Bedürfnisse anpassen.

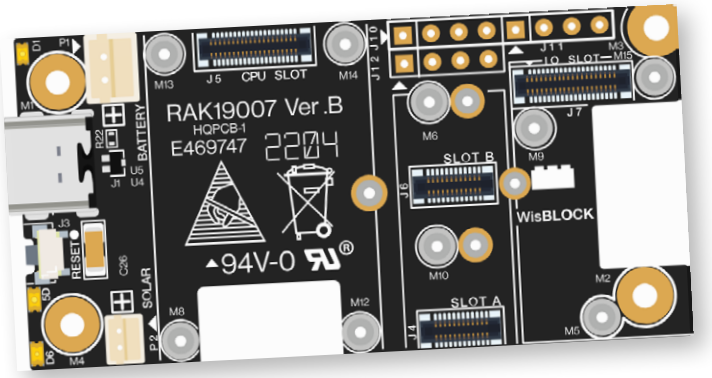


Bild 1. RAK19007-WisBlock-Basis-Board der zweiten Generation. (Quelle: RAKwireless)

LoRa und LoRaWAN

LoRa, kurz für Long Range, ist die ideale drahtlose Kommunikationstechnologie für IoT-Anwendungen. Sie ermöglicht eine Datenübertragung über große Entfernungen bei geringem Strombedarf. LoRaWAN ist ein Protokoll, das auf LoRa aufbaut und die perfekte, standardisierte und skalierbare Methode zur Verwaltung der Konnektivität von LoRa-Geräten über große Entfernungen bietet. LoRa und LoRaWAN ermöglichen gemeinsam stromsparende, weiträumige Netzwerklösungen für verschiedenste Anwendungen. Ich werde euch in diesem Artikel diese extrem spannende Technologie vorstellen.

Ich verwende das Ökosystem WisBlock von RAKwireless, um einen solarbetriebenen LoRa-Knoten für den Anschluss von Sensoren zu bauen. Das *RAK19007 WisBlock Base Board 2nd Gen* ist eine gute Basis für das *Controller Board WisBlock Core* und verschiedene Sensormodule. Der RAK4631-R ist ein stromsparender *WisBlock Core*, der auf einer nRF52840-MCU von Nordic Semiconductor und dem LoRa-Chip SX1262 von Semtech basiert und sowohl Bluetooth Low Energy (BLE) als auch LoRaWAN-Konnektivität bietet (siehe **LoRa und LoRaWAN**).

Der RAK4631-R ist mit dem *RAKwireless Unified Interface V3* (RUI3) ausgestattet, was die Softwareerstellung für IoT-Knoten erheblich vereinfacht. Ich verwende die RUI3-API für die Anwendungsprogrammierung. Außerdem demonstriere ich die Konfiguration des IoT-Geräts mit der *WisToolBox-App*. Weitere Informationen über das WisBlock-Ökosystem finden Sie auf der Webseite von RAKwireless [1] und in meinem E-Buch *IoT-Projects for Makers* [2], einschließlich einer Einführung und Anwendungsbeispielen.

Stromversorgung

Das RAK19007 ist ein *WisBlock Base Board* der zweiten Generation, an das verschiedene WisBlock-Module angeschlossen werden können. Es verfügt über einen USB-C-Anschluss, Anschlüsse für die Stromversorgung und die Verbindung zu den angeschlossenen Modulen. **Bild 1** zeigt ein RAK19007:

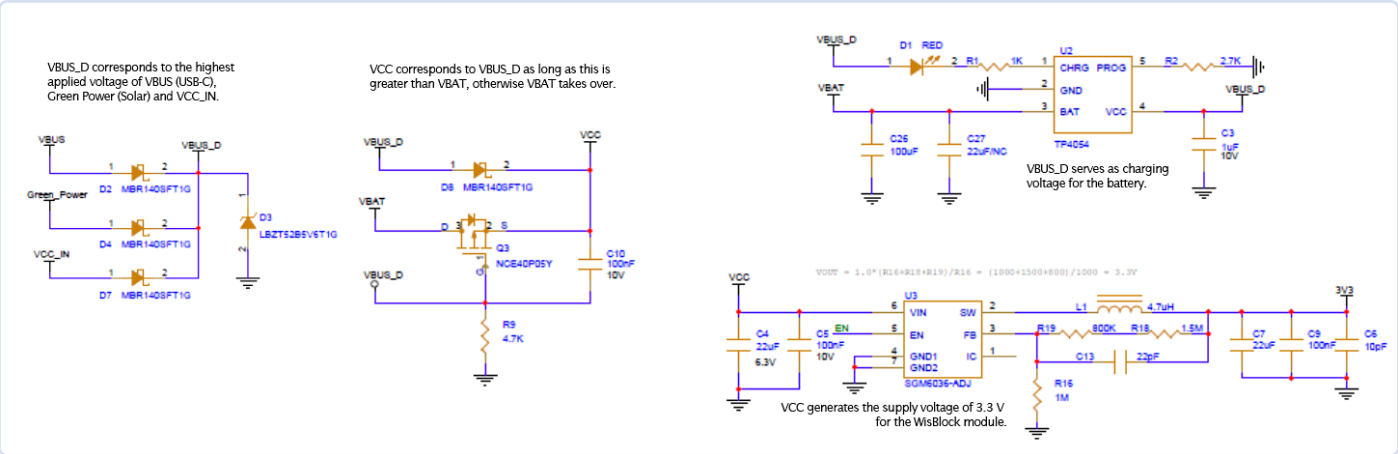


Bild 2. Stromversorgung des RAK19007 (Auszug aus dem Schaltplan der Basisplatine). (Quelle: RAKwireless)

Ganz links sind die verfügbaren Stromversorgungsoptionen mit JST-Anschlüssen für Solar und Batterie und dem USB-C-Anschluss für die Stromversorgung und den Programm-Upload zu sehen. Dieses Modul unterstützt die Stromversorgung durch eine Low-Power-Batterie, das Aufladen eines Lithium-Polymer-Akkus (LiPo) und das Aufladen eines angeschlossenen LiPo-Akkus über ein Solarpanel.

Bild 2 zeigt die Hauptbestandteile der Stromversorgung des RAK19007. Die drei möglichen Versorgungsspannungen für den RAK19007 sind: VBUS von USB-C, Green Power von einer Solarzelle und eine externe Spannung VCC_IN. Die Quelle mit der höchsten Spannung wird automatisch auf VBUS_D geschaltet. Solange VBUS_D größer ist als die Batteriespannung VBAT, bestimmt VBUS_D die VCC-Spannung. Wenn VBUS_D kleiner oder nicht vorhanden ist, sorgt VBAT für die Eingangsspannung des Ultra-Low-Power-Abwärtswandler SGM6036, der die Systemspannung von 3,3 V für die WisBlock-Module einschließlich WisBlock Core ausgibt.

Energieverwaltung

Der RAK19007 besitzt kein echtes Power-Management-System wie der AXP192 oder ähnliche Modelle, so dass nur die verfügbaren Spannungen zur Überprüfung des Ladezustands des LiPo-Akkus herangezogen werden können. Die Batteriespannung VBAT wird über einen Spannungsteiler am Analogeingang AIN0 gemessen. Der Analogeingang AIN1 steht auch am Header J11 zur Messung der Green-Power-Spannung zur Verfügung. Auch hier wird ein Spannungsteiler benötigt. In einem praktischen Test werde ich die beiden Spannungen `Vsol` und `Vbat` aufzeichnen und daraus den Quotienten `State = Vsol/Vbat` bilden. Bei `State > 1` ist die Batterie geladen und die Solarzelle versorgt das Gerät. Wenn `State < 1` ist, muss die Batterie das Gerät versorgen und wird dabei entladen.

Mit dem *RAKwireless Unified Interface V3* (RUI3) ist es einfach, eine IoT-Anwendung mit kundenspezifischer Firmware zu erstellen. Der entsprechende Code, um die Werte der Spannungen Vsol und Vbat zu ermitteln, lautet wie folgt:

```
float Vsol = readADC(AIN1);
float Vbat = api.system.bat.get();
float State = Vsol/Vbat;
```

Setup

Ich habe zunächst mit dem Programm *RAK4631-R_Sensor-Test.ino* den internen ADC sowie die Module RAK1903 (Umgebungslicht) und RAK14001 (RGB-LED) getestet (Blockdiagramm in **Bild 3**). Das Programm finden Sie auf meinem GitHub-Repository [3]. Das Sensormodul *RAK1903* für Umgebungslicht liefert Informationen über die Entwicklung des Umgebungslichts während des Tages. Das RAK14001 fügt mit seiner RGB-LED zu den beiden blauen und grünen LEDs auf der Basisplatine weitere Anzeige-Möglichkeiten hinzu. Beide Module sind allerdings optional und werden für den Zweck des Artikels nicht benötigt.

Ich verwende ein transparentes Acrylgehäuse (RAKBox-B5), um das Testgerät sicher zu betreiben, wobei zu beachten ist, dass dieses Gehäuse nicht wetterfest ist (**Bild 4**). Dieses Gehäuse verfügt bereits über einen Batteriehalter für eine 18650-LiPo-Zelle, meinen bevorzugten LiPo-Akku-Typ, weil er durch die Bauform einfacher zu handhaben ist als andere LiPo-Akku-Formate. Allerdings gibt es LiPo-Akkus dieses Typs mit unterschiedlichen Kapazitäten und leicht unterschiedlichen Größen, über die [4] eine Übersicht gibt. Der hier verwendete Akkuhalter ist für einen „kurzen" LiPo-Akku ausgelegt, wie ihn **Bild 5** ganz links zeigt.

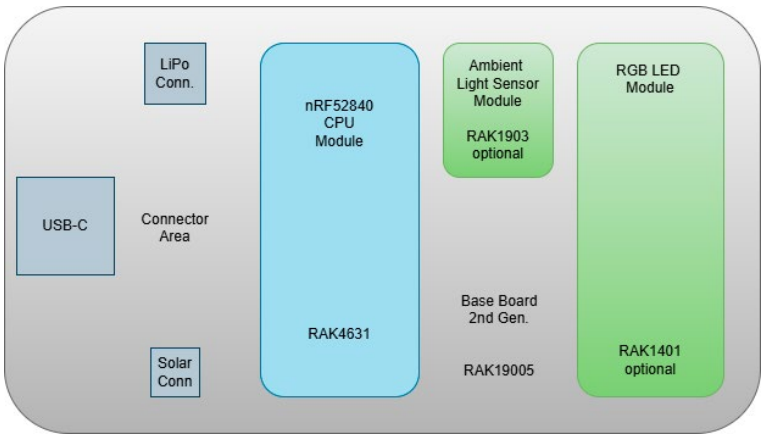


Bild 3. Device under test.

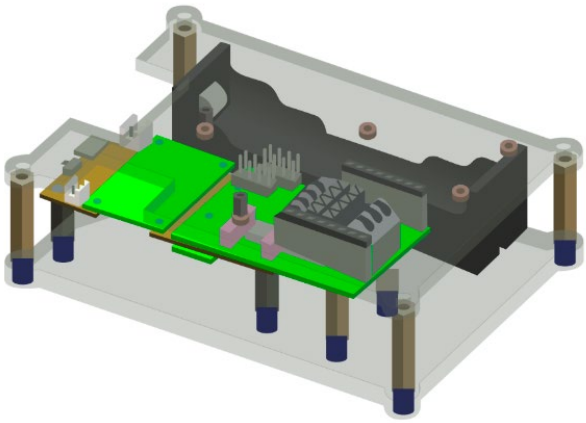


Bild 4. RAKBox-B5 im transparenten Acryl-Gehäuse. (Quelle: RAKwireless)

Bild 5. Varianten des LiPo-Akkus NCR18650. (Quelle: RAKwireless)

Bild 6. Ein Mikro-USB-Solarpanel von Revolt.

Solarpanel

In den unter [4] veröffentlichten Batterietests erreicht ein LiPo-Akku mit einer Kapazität von 3500 mAh bei einem Entladestrom von 1000 mA auf eine Spannung von 2,5 V eine Laufzeit von 200 h. Grob geschätzt könnte ein Entladestrom von 1 mA für 138 Tage aufrechterhalten werden. Zur Reduzierung der Stromaufnahme sollte der IoT-Knoten die meiste Zeit im Schlafmodus verbringen und nur zum Messen und zur Datenübertragung aufwachen. Um den autonomen Betrieb des IoT-Knotens über eine bestimmte Zeit zu gewährleisten, muss die Entladung der Batterie ausgeglichen werden. Dies kann (bei genügendem Sonnenlicht) durch eine Solarzelle geschehen, die an das RAK19007-Baseboard angeschlossen wird. Ein preiswertes MicroUSB-Solarpanel von Revolt (**Bild 6**) ist dazu völlig ausreichend.

Das Revolt-Panel besteht aus monokristallinen Solarzellen mit hohem Wirkungsgrad und langer Lebensdauer. Es liefert eine Ausgangsspannung von 5 V bei einer Leistung (im Idealfall) von 5 W. Die Kabellänge beträgt 3 m. Der flexibel ausrichtbare Montagebügel und die Schutzart IP65 des Gehäuses bieten gute Voraussetzungen für die Außenmontage.

Da die Systemspannung des verwendeten RAK4631-Kernmoduls 3,3 V beträgt, ist ein weiterer Spannungsteiler erforderlich, um die Spannung der Solarzelle zu messen. Zur Montage des Spannungsteilers verwende ich ein microUSB-Breakout-Board wie in **Bild 7**. Vsol und GND sind am JST-Anschluss auf der RAK19007-Grundplatine angeschlossen. Die Spannung VAIN1 geht zum Analogeingang AIN1 an der Header J11. **Bild 8** zeigt die komplette Testanordnung zur Überwachung der Spannungen von Solarzelle und LiPo-Akku.

Anwendungsprogramme

Das Programm muss die Spannungen der Solarzelle und des LiPo-Akkus abfragen und die Ergebnisse drahtlos übertragen. Um die Stromaufnahme des Geräts zu reduzieren, sollte der RAK4631-Core die meiste Zeit schlafen. Mit der RUI3-API ist die Umschaltung in den Ruhezustand nicht allzu kompliziert. Die im Folgenden vorgestellten Anwendungsprogramme unterscheiden sich von üblichen Arduino-Programmen, bei denen die `loop()`-Schleife alle Funktionen endlos wiederholt. Hier schaltet die `loop()`-Funktion nur den RAK4631-Core in den Ruhezustand. Alle Funktionen der aktiven Phase werden durch einen timergesteuerte Handler ausgeführt. Ich habe zwei Programmversionen geschrieben, von denen eine die Messergebnisse über BLE UART, die zweite über LoRaWAN überträgt.

BLE

Nach der Initialisierung durch `setup()` fragt ein timergesteuerter Handler die Spannungen ab und speichert die gesammelten Messwerte in einem Datenpaket, das seriell über BLE ausgegeben wird. Dann versetzt die Funktion `loop()` den RAK4631-Core wieder in den Ruhezustand. Das Programm mit dem Namen *RAK4631-R_Sleep_Test_BLE.ino* ist auf GitHub unter [5] zu finden. Während der Sleep-Periode habe ich eine Stromaufnahme (bei einer Quellspannung von 3,8 V) von 270 µA gemessen. Um die über BLE empfangenen Daten anzuzeigen, verwende ich die Android-App *Serial Bluetooth Terminal 1.46* [6] von Kai

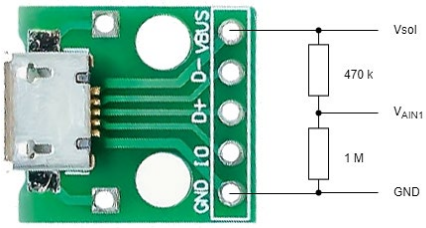


Bild 7. Das Mikro-USB-Breakout Board.

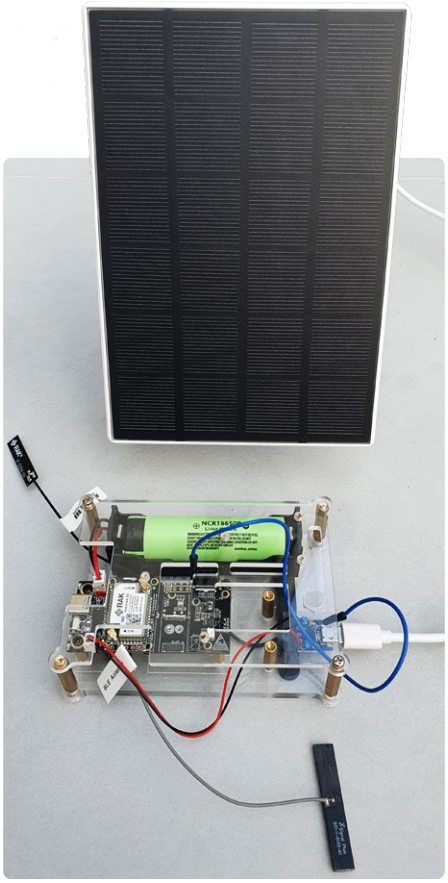


Bild 8. Die gesamte Testausrüstung.

Morich, erhältlich im Google-Play-Store. **Bild 9** zeigt den Spannungszustand nach dem Einschalten am Abend. `State < 1` bedeutet, dass die Batterie das Gerät mit Strom versorgt und entladen wird. Am nächsten Tag war es für Oktober ziemlich sonnig, so dass `State > 1` und die Solarzelle genug Spannung lieferte, um das Gerät zu versorgen und die Batterie aufzuladen (**Bild 10**).

LoRaWAN

Vor der Programmierung der LoRaWAN-Anwendung habe ich die Desktop-App *WisToolBox* verwendet, um den RAK4631-Core zu konfigurieren. Diese App kommuniziert mit den WisBlock-Geräten über USB, und nach Auswahl des richtigen USB-Ports und des RAK4631-Core-Moduls erscheinen alle Geräteinformationen im Dashboard (**Bild 11**). Sie können alle für die LoRaWAN-Konfiguration erforderlichen Parameter einstellen, wie in **Bild 12** und **Bild 13** gezeigt.

Die in WisToolBox eingestellten Parameter werden an den RAK4631-Core gesendet und im Flash-Speicher abgelegt. Im Programm ist ihre Einstellung deshalb nicht mehr erforderlich. In meinem Programm *RAK4631-R_Sleep_Test_LoRaWAN.ino*, das auf GitHub [7] verfügbar ist, habe ich noch den traditionellen Weg durch explizite Deklaration verwendet. Während der Sleep-Phase habe ich eine Stromaufnahme (bei einer Quellspannung von 3,8 V) von 235 µA gemessen.

The Things Stack

Hier werde ich den Cluster für Europa (eu1) von *The Things Stack Sandbox* (TTSS, ehemals The Things Stack) in der Community Edition (TTSCE) als LoRaWAN-Netzwerkserver (LNS) verwenden. Die Cluster sind für nicht-kommerzielle, kleinere Tests und Experimente gedacht und verwalten die Konnektivität von LoRaWAN-Geräten. Es ist frei verfügbar und quelloffen, so dass es von jedermann genutzt und verändert werden kann, und wird von The Things Network [10] unterstützt.

TTSS erfordert die Registrierung des Geräts; die in **Bild 12** gezeigten LoRaWAN-Schlüssel und weitere Daten sind für die Registrierung erforderlich. Einzelheiten zu TTSS als LoRaWAN-Netzwerk-Server und Datacake [8] als Visualisierungswerkzeug finden Sie auch in meinem anderen Buch mit dem Titel *LoRaWAN-Knoten im IoT* [9].

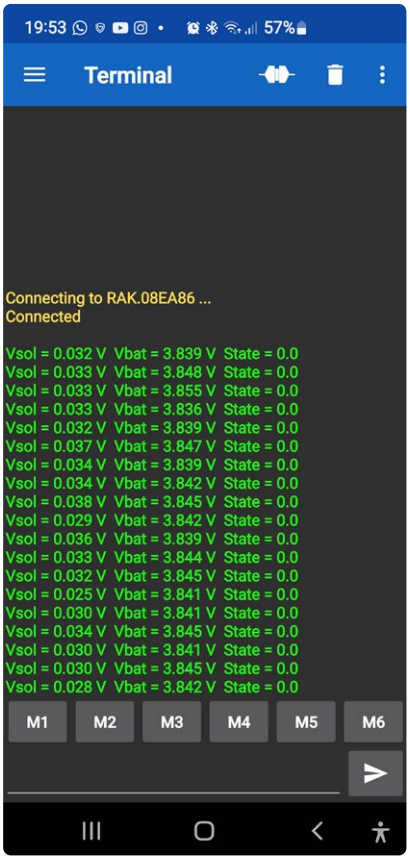


Bild 9. Spannungszustand nach dem Einschalten am Abend.

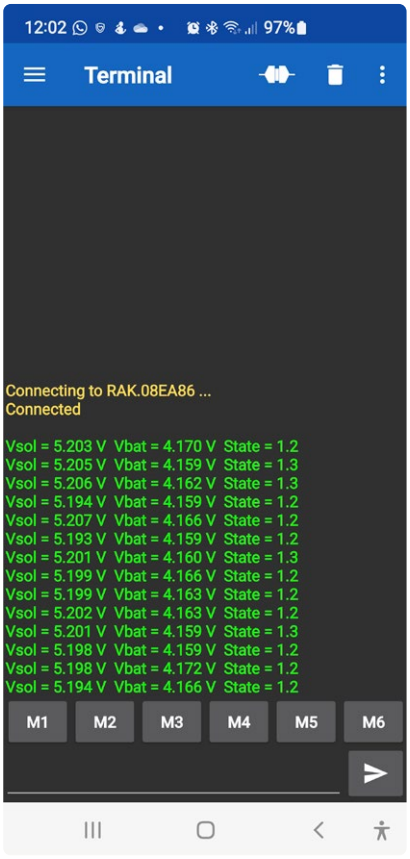


Bild 10. Spannungszustand an einem sonnigen Oktobertag.

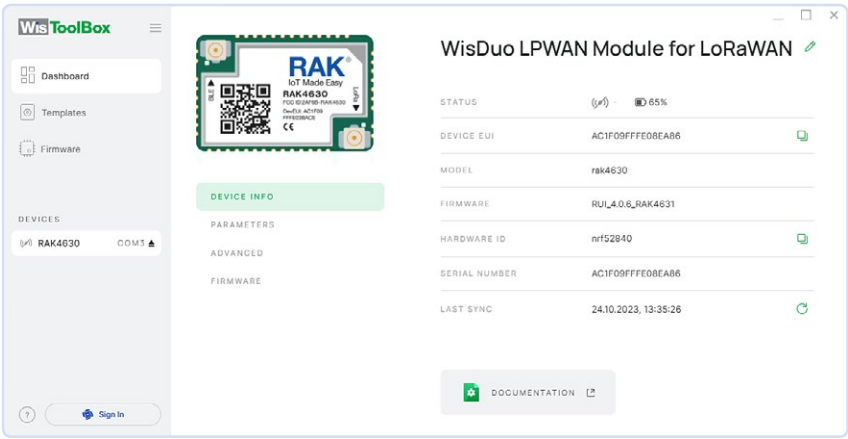


Bild 11. Geräteinfo im WisToolBox-Dashboard.

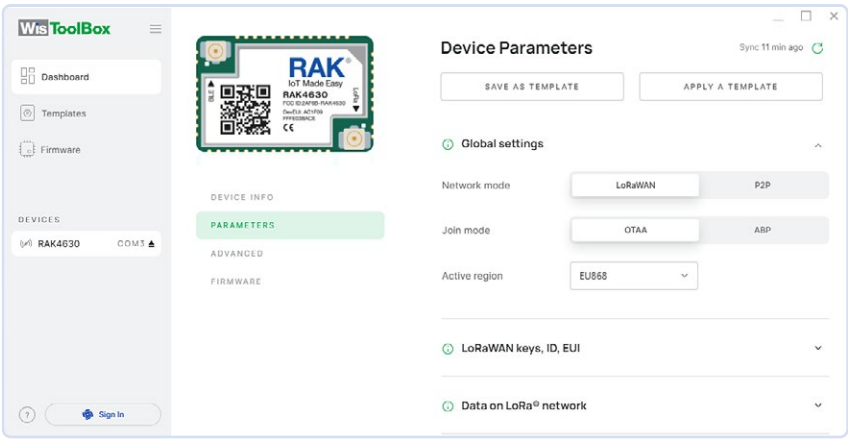


Bild 12. Globale Einstellungen im WisToolBox-Dashboard.

Für TTSS gilt eine Fair-Use-Policy, die die Uplink-Sendezeit auf 30 s pro 24 Stunden pro Knoten und die Downlink-Nachrichten auf zehn Nachrichten pro Tag und Knoten begrenzt. Die Nutzlast und die Häufigkeit des Uploads bestimmen die Uplink-Sendezeit. Mit dem Airtime-Rechner [11] sehen Sie dies auf einen Blick,

Bei einer Nutzlast von fünf Bytes und einer Datenrate DR5 (SF7BW125) errechnet sich eine Airtime von 51,5 ms. Zwischen zwei Uploads muss eine Zeitspanne von mindestens 148,2 s vergehen, um die Fair-Use-Policy einzuhalten. Die Uploadzeit im Anwendungsprogramm beträgt fünf Minuten. Nach erfolgreicher Registrierung und Start der Anwendung zeigt die TTSS-Konsole die empfangenen Uplink-Nachrichten (**Bild 14**). Der verwendete Payload-Dekoder ist ebenfalls auf GitHub verfügbar.

Ergebnisse

Die detaillierten Ereignisse der Konsolenausgabe sind in **Listing 1** aufgeführt, aus dem hervorgeht, dass die Spezifikationen erfüllt werden. Bandbreite, Spreizfaktor und die resultierende Sendezeit sind wie erwartet und fett markiert. Um die vom LoRaWAN-Netzwerk-Server empfangenen Daten anzuzeigen, verwende ich Datacake, das sich leicht in TTSS integrieren lässt. Lesen Sie dazu die Online-Dokumentation [12] oder die Beispiele in dem oben erwähnten Buch. Nach der Integration können Sie ein Datacake-Dashboard ähnlich wie in **Bild 15** erstellen.

Sie können die aktuellen Spannungen der Solarzelle und des LiPo-Akkus sowie den Verlauf dieser Spannungen über einen Tag hinweg sehen. Im Laufe des Tages liefert die Solarzelle Strom, um den LiPo-Akku zu laden. Am frühen Abend fällt die Spannung der Solarzelle unter die des LiPo-Akkus. Dann versorgt die Batterie den Knoten und wird auf „natürliche" Weise entladen. Der Test lief über mehrere Wochen und zeigte, dass der IoT-Knoten auch unter ungünstigeren Wetterbedingungen im Herbst und im Winter zuverlässig funktionierte.

SE – 230668-02

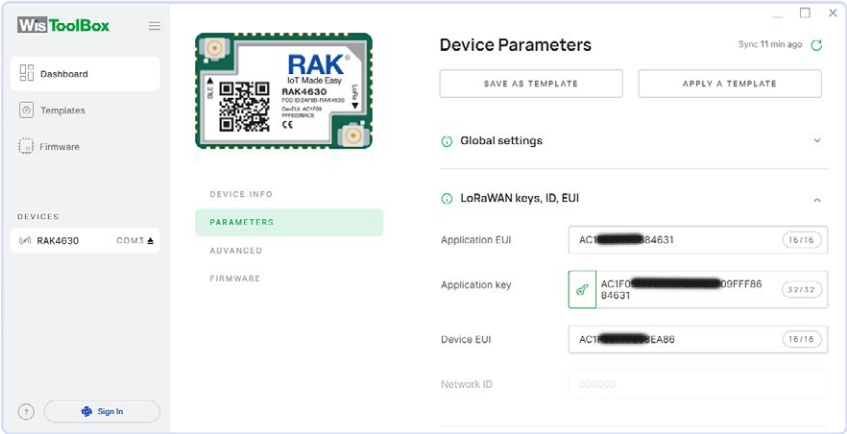


Bild 13. LoRaWAN-Schlüssel, ID und EUI im WisToolBox-Dashboard.

Listing 1: Ausgabe auf der Konsole

```
"settings": {
   "data_rate": {
     "lora": {
 "bandwidth": 125000,
 "spreading_factor": 7,
 "coding_rate": "4/5"
     }
   },
   "frequency": "867100000",
   "timestamp": 4014400019,
   "time": "2023-10-25T13:28:28.098315Z"
 },
 "received_at": "2023-10-25T13:28:28.116904692Z",
 "consumed_airtime": "0.051456s",
 "version_ids": {
   "brand_id": "rakwireless",
   "model_id": "wisblock-4631",
   "hardware_version": "1.0",
   "firmware_version": "1.2.0",
   "band_id": "EU_863_870"
 },
```

Bild 14. Konsolenausgabe TTS (CE).

Bild 15. Datacake-Dashboard für RAK-Sensorknoten.

Sie haben Fragen oder Kommentare?

Haben Sie technische Fragen oder Anmerkungen zu diesem Artikel? Dann wenden Sie sich bitte an den Autor unter claus@ckuehnel.ch oder an die Elektor-Redaktion unter redaktion@elektor.de.

Über den Autor

Dr. Claus Kühnel hat an der Technischen Universität Dresden Informatik studiert und entwickelt seit vielen Jahren erfolgreich eingebettete Systeme, unter anderem für Labordiagnosegeräte. In diesem interdisziplinären Bereich kam er mit der Maker-Szene in Kontakt. Er hat zahlreiche Artikel und Bücher über Mikrocontroller-Hardware und -Software im In- und Ausland veröffentlicht. Er ist ein absoluter Fan neuer Technologien rund um Mikrocontroller.

Passende Produkte

- **C. Kühnel, LoRaWAN-Knoten im IoT (Elektor, 2021)**
  Buch, kartoniert, deutsch: www.elektor.de/products/lorawan-knoten-im-iot
  E-Buch, PDF, deutsch: www.elektor.de/products/lorawan-knoten-im-iot-pdf
- **LR1302 LoRaWAN-HAT für Raspberry Pi (EU868)**
  www.elektor.de/20744
- **RA-08H LoRaWAN-Entwicklungsboard mit integriertem RP2040 und 1,8"-LCD (EU868)**
  www.elektor.de/20541

THEMENFOKUS
Besuchen Sie unsere Seite **Drahtlose Kommunikation** für Artikel, Projekte, Nachrichten und Videos.
www.elektormagazine.de/**drahtlose-kommunikation**

WEBLINKS

[1] WisBlock-System von RAKwireless: https://www.rakwireless.com/en-us/products/wisblock
[2] C. Kühnel, IoT-Projects for Makers (Elektor, 2023): https://www.amazon.de/dp/B0C8VCF4DF
[3] RAK4631-R_Sensor-Test.ino auf GitHub: https://github.com/ckuehnel/WisBlock/tree/main/RAK4631-R_Sensor-Test
[4] 18650-Akkus bei Akkuline: https://www.akkuline.de/test/18650-lithium-ion-zelle-vergleich?projektLineId=11
[5] RAK4631-R_Sleep_Test_BLE.ino auf GitHub: https://github.com/ckuehnel/WisBlock/tree/main/RAK4631-R_Sleep_Test_BLE
[6] Serielles Bluetooth-Terminal: https://www.kai-morich.de/android
[7] RAK4631-R_Sleep_Test_LoRaWAN.ino auf GitHub: https://github.com/ckuehnel/WisBlock/tree/main/RAK4631-R_Sleep_Test_LoRaWAN
[8] Datacake: https://datacake.co/
[9] Claus Kühnel, LoRaWAN-Knoten im IoT (Elektor, 2021): https://www.elektor.de/products/lorawan-knoten-im-iot
[10] The Things Network: https://www.thethingsnetwork.org/
[11] Airtime-Rechner: https://avbentem.github.io/airtime-calculator/ttn/eu868
[12] Online-Dokumentation Datacake: https://docs.datacake.de/device/configuration

AWS für Arduino und Co.

Teil 2: Daten senden mit AWS-IoT-ExpressLink

Von Tam Hanna (Ungarn)

Im ersten Teil dieser zweiteiligen Serie haben wir AWS IoT ExpressLink vorgestellt. Um einem Mikrocontroller das Senden von Daten zu den Amazon Web Services (AWS) zu erleichtern, übernimmt ein zwischengeschaltetes Modul die Kommunikation. Kleine Mikrocontroller (und deren Firmware-Entwickler) können das „AWS IoT ExpressLink powered connectivity module" einfach mit AT-Befehlen ansteuern und müssen sich nicht um Details kümmern. Neben der Konfiguration des ESP32-basierten ExpressLink-Moduls ging es im ersten Teil auch um das Anlegen eines Things auf AWS und die nötigen Zertifikate und Berechtigungen. In diesem Artikel werden wir Daten, die ein Arduino anliefert, zu AWS schicken. Die Cloud-Plattform bietet hier eine Reihe mächtiger Möglichkeiten.

Gleich zum Geleit sei angemerkt, dass die Nutzung von AWS natürlich kein Selbstzweck ist. Wer ein ExpressLink-Modul in seine Applikation einbindet, sollte dies mit der Absicht tun, dass die diversen im AWS-IoT-Produktportfolio zur Verfügung stehenden Services auch genutzt werden.
AWS zeigt sich dabei funktionsgewaltig (**Bild 1**), einen Überblick findet man in [1]. Wer mehr darüber wissen möchte, was der Betrieb von AWS IoT Core in der Praxis kostet, wird von Amazon unter [2] informiert.

Daten senden

Die Nutzung der Funktionen setzt logischerweise voraus, dass sich die zu verarbeitenden Informationen in der AWS-Cloud befinden. Als Methode zum Anliefern - insbesondere im Data-Science-Bereich ist hierfür der Begriff *Ingestion* weit verbreitet - hat sich MQTT etabliert.
Im ersten Teil [3] haben wir bereits einen Test-Sketch von Espressif und AWS vorgestellt, der das ExpressLink-Modul über AT-Befehle in einen Zustand versetzt, mit dem es für AWS „sichtbar" ist (Provisioning) - natürlich muss das Modul dafür die SSID des Routers kennen. Wie wir uns erinnern, lassen sich die Verbindungsdaten dafür bequem auch von einem Smartphone aus eingeben, das über Bluetooth mit dem Modul Kontakt aufnimmt.

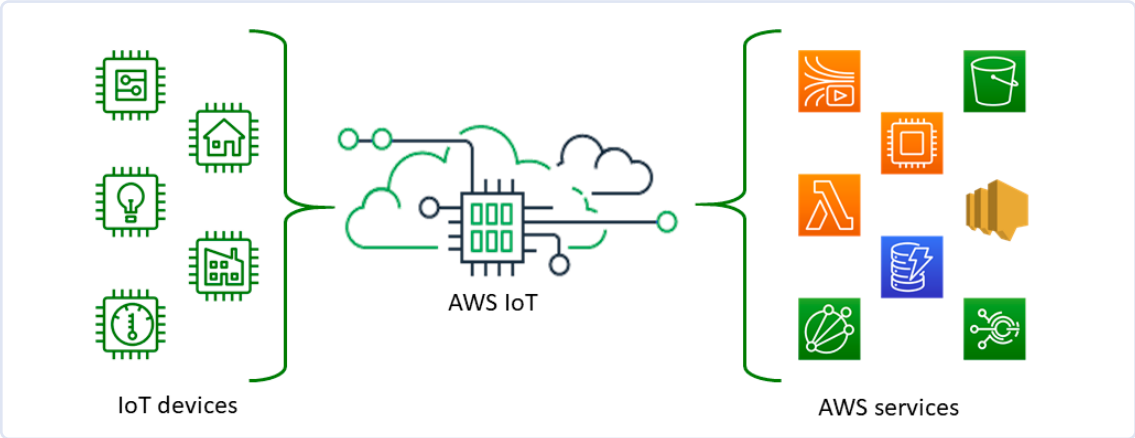


Bild 1. Amazon bietet im AWS-Portfolio umfangreiche Dienstleistungen an. (Quelle: AWS [1])

Falls alles geklappt hat, lassen wir den Test-Sketch nun periodisch Informationen in Richtung der AWS-Server schicken:

```
case STATE_CONNECTED:
if (event == EVENT_CONLOST) {
state = STATE_PROVISIONED;
break;
}
static unsigned long last_send_time;
if (millis() - last_send_time >= 10000) {
response =
     execute_command("AT+SEND1 Hello World", 5000);
last_send_time = millis();
}
break;
```

Der Befehl `execute_command("AT+SEND1...)` ist dabei für das Übertragen einer Message an den AWS IoT-Hub verantwortlich. Diensterfahrene Entwickler fragen sich an dieser Stelle naturgemäß, wie der vom jeweiligen Gerät verwendete MQTT-Topic festgelegt wird.
Die Antwort darauf findet sich im Code etwas weiter oben:

```
case STATE_EL_READY:
    response = execute_command
   ("AT+CONF Endpoint="MY_AWS_IOT_ENDPOINT"", 3000);
    response = execute_command
   ("AT+CONF Topic1=TEST", 3000);
    response = execute_command
   ("AT+CONF? SSID", 3000);
    state = process_ssid(response);
```

Da das mehrfache Festlegen eines MQTT-Topics kostenlos ist, empfiehlt der Autor, eine Modifikation des Beispielprogramms vorzunehmen und den Befehl vor jeder Datenübertragung in Richtung des Moduls zu jagen:

```
if (millis() - last_send_time >= 10000) {
response = execute_command
     ("AT+CONF Topic1=TEST", 3000);
response = execute_command
     ("AT+SEND1 Hello World", 5000);
last_send_time = millis();
}
```

Nach diesen Modifikationen ist das Programm zur Ausführung bereit. Zu beachten ist, dass die Arduino-Firmware bei Vorhandensein einer Stromversorgung permanent Nachrichten in Richtung des Server sendet, und diese verursachen logischerweise Kosten. Daraus folgt, dass der Arduino auf keinen Fall unbeaufsichtigt angesteckt bleiben sollte!
Für eine erste Überprüfung der Funktionsfähigkeit bietet sich dann der Arduino Serial-Monitor an. In der letzten Ausgabe wurde gezeigt, wie man das Test-Programm modifiziert, um den zweiten Hardware-UART des Arduino UNO R4 zu nutzen. In diesem Fall

```
EXC : AT+CONF Endpoint=a3g[illegible]iot.eu-west-1.amazonaws.com
EXC : AT+CONF Topic1=TEST
EXC : AT+CONF? SSID
EXC : AT+CONNECT
EXC : AT+CONF Topic1=TEST
EXC : AT+SEND1 Hello World
EXC : AT+CONF Topic1=TEST
EXC : AT+SEND1 Hello World
EXC : AT+CONF Topic1=TEST
EXC : AT+SEND1 Hello World
```

Bild 2. Die Messages scheinen im Seriellen Monitor der Arduino-IDE auf.

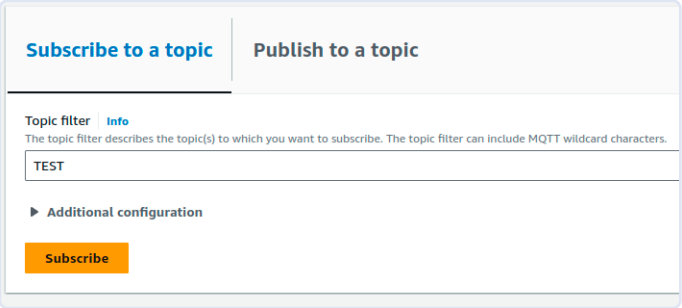


Bild 3. Diese Einstellungen führen zum Ziel.

wird man von der Konsole (wie in **Bild 2** gezeigt) über die erfolgreiche Anlieferung der an die AWS-Cloud zu sendenden Pakete informiert.
Natürlich möchten wir nun auch sehen, ob die Nachrichten auch tatsächlich bei AWS angekommen sind. Wer im Menü im AWS-Backend die Option *Test→MQTT Test Client* anklickt, findet sich im AWS-Test-Client wieder. Wie im letzten Artikel gilt auch hier, dass die URL von der geographischen Lokalisierung der AWS-Primitiva abhängig ist - in der Konfiguration des Autors lautete sie *https://eu-west-1.console.aws.amazon.com/iot/home?region=eu-west-1#/test*. Wichtig ist, dass nach dem Laden der Seite in der Rubrik *Connection Details* grün *connected* eingeblendet wird - der von Amazon zur Verfügung gestellte Client leidet manchmal unter Startschwierigkeiten.
Zur Einrichtung der MQTT-Verbindung ist im nächsten Schritt die Konfiguration erforderlich, die in *Subscribe to a Topic* nach dem in **Bild 3** gezeigten Schema erfolgt.
Nach dem Anklicken von *Subscribe* sehen Sie wie in **Bild 4** gezeigt das Aufscheinen der eingehenden Nachrichten.

Fortgeschrittene Befehlsübertragung

Eine der Stärken von cloudgetriebenen MQTT-Brokern ist ihre Möglichkeit, auf die übertragenen Informationen direkt zu reagieren, also ohne großes Programmieren von Applikationslogik.
Zur Realisierung dieser extrem nützlichen Funktion ist von Seiten des Entwicklers allerdings Mithilfe erforderlich, als die zu übertragenden Nutzdaten (Payload) normalerweise in ein für das Backend verständliches Format gebracht werden müssen.
In den meisten Fällen kommt dabei eine an JSON erinnernde Struktur zum Einsatz. Neben der eigentlichen Payload wie zum Beispiel Messwerten können solche Messages Attribute enthalten, die beispielsweise die kritische Temperatur einer Turbine in ein

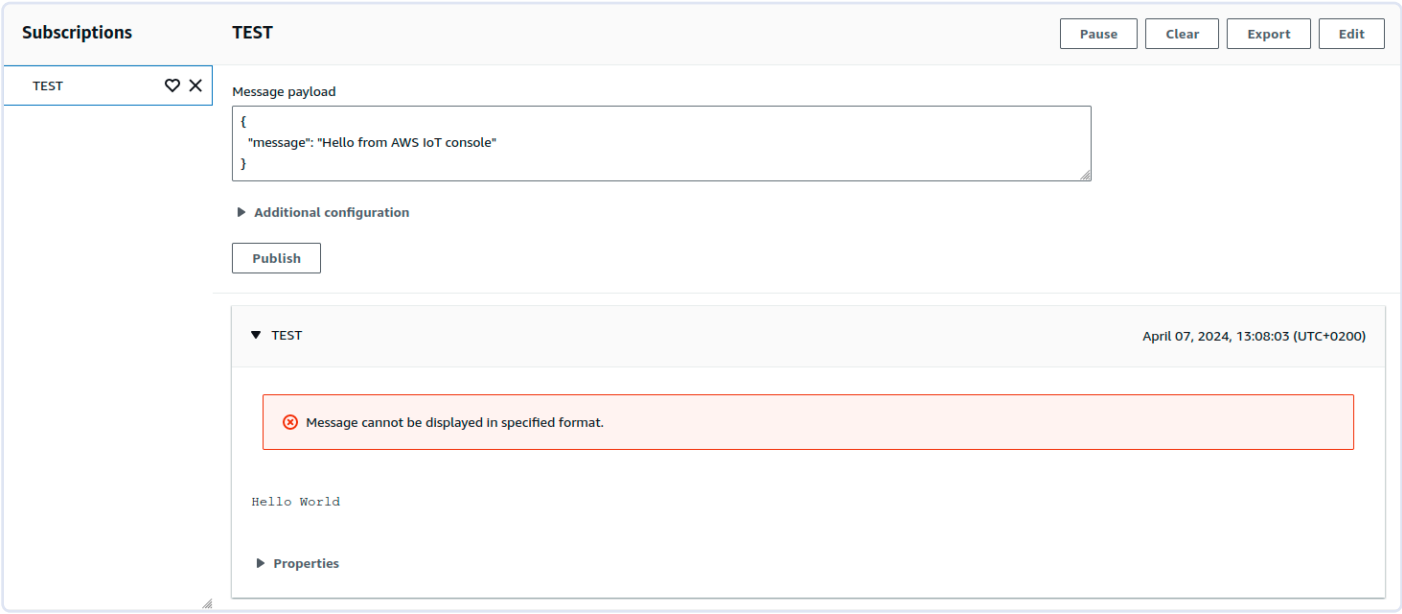


Bild 4. Die vom Arduino gesendeten Nachrichten zeigen sich in der AWS-Konsole.

Table 3 - Configuration dictionary non-persistent keys

Configuration Parameter	Type	Initial Value	Buff Size	Description
QoS	R/W	0	1	QoS level selected for SEND commands
Topic1	R/W	{Empty}	≥128	Custom defined topic 1
Topic2	R/W	{Empty}		Custom defined topic 2
...				
Topic<Max Topic>	R/W	{Empty}		Custom defined topic MaxTopic
EnableShadow	R/W	0	1	0 - disabled, or 1 - enabled
Shadow configuration parameters (required only by modules that support the Shadow feature, see 9.2 AWS IoT Device Shadow)				
Shadow1	R/W	{Empty}	64	Custom defined named shadow
...				
Shadow<MaxShadow>	R/W	{Empty}		Custom defined named shadow
BLE configuration parameters (required only by modules that support BLE host control, see 12 Bluetooth Low Energy)				
BLECentral1	R/W	{Empty}	≥ 128	GAP Central discovery/connect configurations.
BLECentral2	R/W	{Empty}	≥ 128	
...				
BLECentral<MaxBLECentral>	R/W	{Empty}	≥ 128	
BLEGATT1	R/W	{Empty}	≥ 128	GATT Characteristic definitions (JSON).
BLEGATT2	R/W	{Empty}	≥ 128	
...				
BLEGATT<MaxBLEGatt>	R/W	{Empty}	≥ 128	
BLEPeripheral	R/W	{Empty}	≥ 128	GAP Peripheral advertising configuration.

Bild 5. Der Konfigurationsspeicher des Expresslink-Moduls ist durchaus parameterreich. (Quelle: AWS [7])

SQL ist nicht SQL

Wie im Fall von jedem anderen relationalen Datenbanksystem gilt naturgemäß auch im Fall von AWS, dass eine hauseigene SQL-Syntax zum Einsatz kommt. Weitere Informationen zur Amazon-Variante finden sich dabei unter der URL *https://docs.aws.amazon.com/iot/latest/developerguide/iot-sql-reference.html*.

für AWS verständliches Format bringen, so dass ein Alarm ausgelöst werden kann, sollte diese Temperatur überschritten werden. Zur Realisierung derartiger Funktionen ist es meist erforderlich, fortgeschrittene Funktionen des ExpressLink-Moduls zu verwenden. Die unter der [4] bereitstehende Dokumentation bietet dabei einen Überblick des Befehlssatzes.

Bei genauer Analyse des Sendebefehls stellen wir fest, dass er nach dem Schema `SEND[#]` definiert ist (oben wird zum Beispiel `SEND1` verwendet). Es ist also erlaubt, im Konfigurationsspeicher des Moduls mehrere als zu bespielend markierende Kanäle zu speichern. Spezifischerweise gehören zu jedem Kanal einige Dutzend Attribute, die Amazon detailliert dokumentiert - **Bild 5** zeigt einen Auszug der Möglichkeiten.

Das ideale Übertragungsformat für die Nutzdaten ist stark vom jeweiligen Konsumenten abhängig - wir wollen uns anfangs auf „einfach aufbrechbare" Nachrichten beschränken, weshalb wir uns an die unter [5] bereitstehende JSON-Spezifikation halten.

Die Arduino-Umgebung bietet in der Theorie verschiedenste Bibliotheken an, die das dynamische Zusammenbauen von JSON-Payloads ohne große String-Manipulation ermöglichen. Für unsere kleinen Experimente benötigen wir solche Libraries nicht. Wir adaptieren die Sende-Routine unseres Beispielprogramms einfach wie hier gezeigt:

```
if (millis() - last_send_time >= 10000) {
    response = execute_command
      ("AT+CONF Topic1=TEST", 3000);
    response = execute_command
      ("AT+SEND1 { \"messdaten\":
        { \"temperatur\": 22, \"kaefer\": 0 }}", 5000);
    last_send_time = millis();
}
```

Für C-unerfahrene Entwickler sei noch angemerkt, dass die Sequenz `\"` (in C auch als Escapesequenz bezeichnet) dafür sorgt, dass das Anführungszeichen als Teil des Strings und nicht als String-Terminator gesehen wird.

An dieser Stelle ist die erste Version des Programms ausführungsbereit - der weiter oben besprochene MQTT-Analysator wird das Eingehen der Payloads bestätigen.

Regeln mit SQL

Im nächsten Schritt wollen wir ein fiktives Humidor-Überwachungssystem nach folgendem Schema adaptieren, um regelmäßig eine besorgniserregende Nachricht abzusenden, die auf das Auftreten von Tabakkäfern in der geschätzten Zigarren-Sammlung des Autors hinweist:

```
static unsigned long last_send_time;
static int laeufer = 0;
if (millis() - last_send_time >= 10000)
{
  response = execute_command
     ("AT+CONF Topic1=TEST", 3000);
  if(laeufer++<10)
  {
    response = execute_command
       ("AT+SEND1 {  \"messdaten\":
       {    \"temperatur\": 22,
            \"kaefer\": 0  }}", 5000);
  }
  else
  {
    laeufer=0;
    response = execute_command
       ("AT+SEND1 {  \"messdaten\":
       {    \"temperatur\": 22,
            \"kaefer\": 1  }}", 5000);
  }
  last_send_time = millis();
  }
```

Das Aufscheinen der Nachrichten mit unterschiedlichen Käfer-Werten im AWS-Backend informiert uns darüber, dass unser System „gefechtsbereit" ist. Im nächsten Schritt müssen wir das AWS-Backend allerdings darüber informieren, dass das Vorfinden von Käfern im Humidor eine relevante Information darstellt. Dies erfolgt durch Einrichtung einer Regel - der Regel-Manager wartet im AWS-Backend dabei unter der URL *Manage→Message routing Rules*. Wer ein „jungfräuliches" AWS-Konto verwendet, findet hier - wie in **Bild 6** gezeigt - den Assistenten zum Anlegen neue Regeln vor. Im ersten Schritt fragt der Einrichtungsassistent abermals nach dem Namen der Regel - in den folgenden Schritten wird der Autor hierfür *TamsKaeferRegel* verwenden. Eine Beschreibung und Text sind nicht notwendig.

Im zweiten Schritt ist die Eingabe einer SQL-Query erforderlich, die ein Ereignis triggert. Im Interesse der Einfachheit setzt Amazon auf eine SQL-Variante, die das Beschreiben der Regel ermöglicht (siehe dazu auch den Kasten **SQL ist nicht gleich SQL**). Für unser erstes Experiment wollen wir die Query `SELECT messdaten.kaefer FROM 'TEST' WHERE kaefer>0` verwenden, und in den nächsten Schritt wechseln.

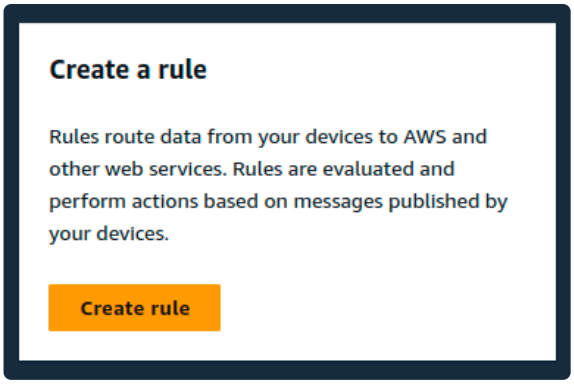


Bild 6. Dieses Werkzeug erlaubt die Generierung von Regeln.

Basic information Info

Blueprint name

Send email on click of an IoT button — nodejs18.x

An AWS Lambda function that sends an email on the click of an IoT button.

Function name

Enter a name that describes the purpose of your function.

tamsEagKaeferFunc

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime

nodejs18.x

Architecture

x86_64

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.

- Create a new role with basic Lambda permissions
- Use an existing role
- Create a new role from AWS policy templates

Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

Role name

Enter a name for your new role.

tamsEagKaeferFunc

Use only letters, numbers, hyphens, or underscores with no spaces.

Policy templates - *optional* Info

Choose one or more policy templates.

AWS IoT Button permissions
SNS

Bild 7. Die Einstellungen sollten den AWS-Regelgenerator befriedigen.

Successfully created the function **tamsEagKaeferFunc**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Lambda > Functions > tamsEagKaeferFunc

tamsEagKaeferFunc

Throttle | Copy ARN | Actions

Function overview Info

Export to Application Composer | Download

Diagram | Template

tamsEagKaeferFunc

Layers (0)

+ Add trigger

+ Add destination

Description

An AWS Lambda function that sends an email on the click of an IoT button.

Last modified

14 seconds ago

Function ARN

arn:aws:lambda:eu-west-1:326959574431:function:tamsEagKaeferFunc

Function URL Info

-

Bild 8. Das grüne Banner auf der Bildschirmoberseite bestätigt die erfolgreiche Generierung der AWS Cloud-Funktion.

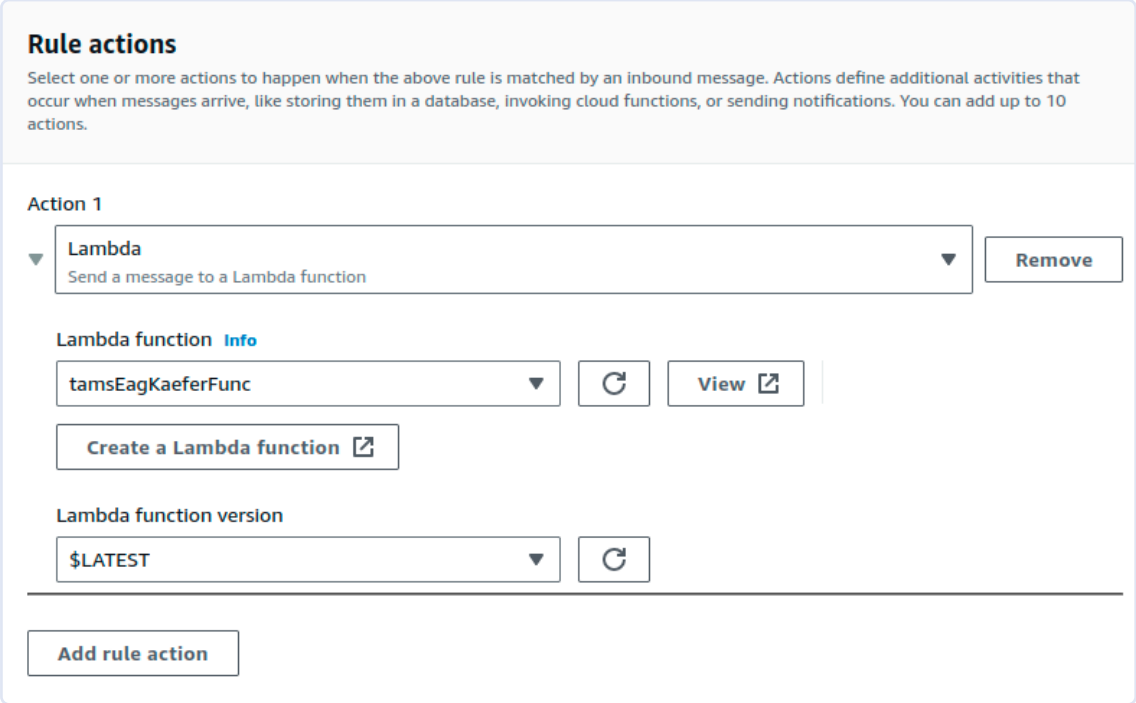


Bild 9. Die Regel-Konfiguration ist in Arbeit.

Im dritten Schritt fordert Amazon zum Festlegen der *Rule Actions* auf. Dabei handelt es sich um Aktionen, die das AWS-Backend beim Antreffen eines Trigger-Ereignisses auszulösen hat.
Wie wollen uns im folgenden Schritt für die Aktion *Lambda* entscheiden: Es handelt sich dabei um in der Cloud lebende Codestücke, die das Amazon-Backend (ohne Bedarf für eine vom Entwickler einzurichtende virtuelle Maschine) zur Ausführung bringen kann. Im Bereich der Lambda-Funktionen unterstützt Amazon dabei ein gutes Dutzend verschiedener Programmiersprachen - unter [6] findet sich Dokumentation zum Thema.
Im nächsten Schritt klicken wir auf den Knopf *Create a Lambda Function*, der einen Assistenten zur Erzeugung der eigentlichen in der Cloud anzulegenden Funktion in einem neuen Browserfenster öffnet.
Im nächsten Schritt entscheiden wir uns dort für die Option *Use a Blueprint*, um den Funktionsgenerator zum Anbieten verschiedener Projektvorlagen zu animieren. Die für uns benötigte heißt dann *IT Automation→Send E-Mail on Click of IoT Button*.
Nach dem Auswählen der gewünschten Projektvorlage können Sie nach unten scrollen und im Feld *Function Name* einen Namen vergeben. Da unsere Funktion eine aktive Rolle spielen wird, muss sie natürlich noch Rechte zugewiesen bekommen. In der Rubrik *Execution Role* entscheiden wir uns für die Option *Create a new Role from AWS Policy Templates*, und vergeben außerdem auch einen Namen für die neu zu generierende Rolle.
Bild 7 zeigt die vom Autor für die folgenden Schritte avisierte Konfiguration. Im Feld *AWS IoT Trigger* wählen wir die Option *Custom IoT Rule*, um danach im Feld *Environment Variables* die E-Mail-Adresse des Autors einzugeben. Danach folgt ein Klick auf *Create Rules*.
Normalerweise tritt an dieser Stelle ein Henne-Ei-Problem auf: Im Feld *Existing Rules* wird eine Eingabe erwartet; falls hier nichts ausgefüllt ist, führt dies zum Aufscheinen der Fehlermeldung `This field is required`.
Zur Lösung des Problems klicken wir in der Rubrik *Triggers* auf den Knopf *Remove* und kehren danach in die Sektion *Environment Variables* zurück. Nach dem Anklicken von *Create Function* beginnt die Erzeugung der Cloud-Funktion, die ob der Langsamkeit von des Backends mitunter etwas Zeit in Anspruch nimmt.
Lohn der Mühen ist jedenfalls das Aufscheinen des in **Bild 8** gezeigten Assistenten.
Im nächsten Schritt kehren wir in das andere Browser-Fenster zurück, wo wir die Regel wie in **Bild 9** gezeigt konfigurieren.

Bild 10. Die Test-Nachrichten treffen erfolgreich ein.

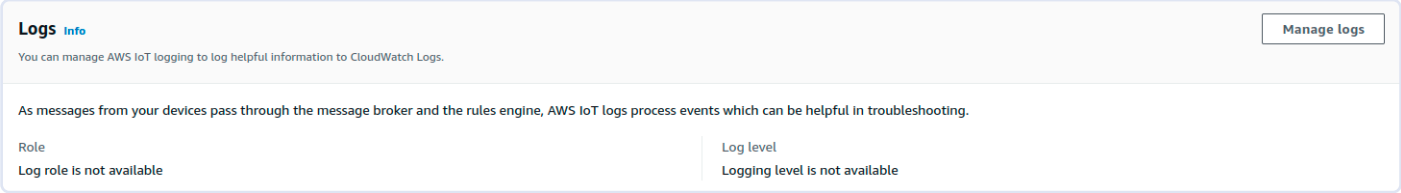


Bild 11. Hier fehlt es an Logging-Rollen.

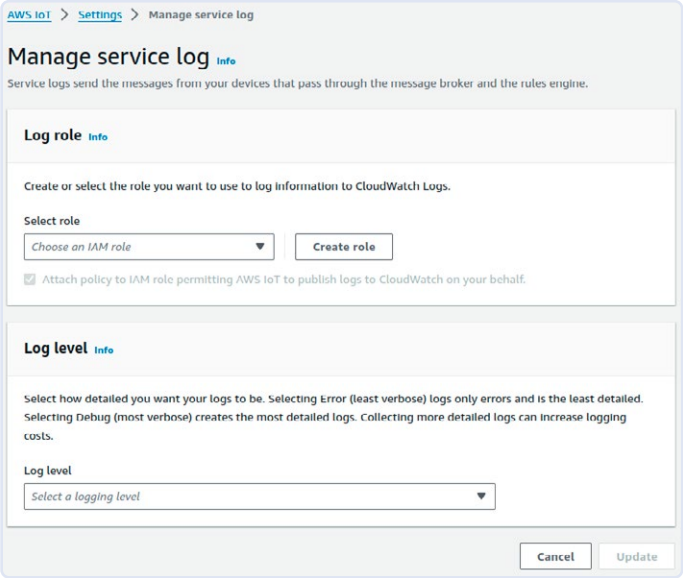


Bild 12. Zwei Einstellungen stehen zwischen Entwickler und erfolgreicher Logging-Konfiguration.

Danach klicken wir auf *Next*, um eine Liste der Einstellungen zu bekommen. Zu guter Letzt fehlt dann noch das Anklicken von *Create*, um die neue Regel im Amazon-Backend anzulegen.
Nun sollte im Backend alles an seinem Platz sein - von besonderer Wichtigkeit ist die in der Rubrik *Message Routing→Rules* angelegte neue Regel, die die „Verarbeitung" beschreibt. Sodann bietet sich das probeweise Anwerfen der Arduino-Software an - **Bild 10** zeigt, wie die vom Arduino gesendeten Nachrichten im MQTT-Test-Client zur Abwertung aufscheinen.
Problematisch ist, dass in der eingestellten E-Mail-Adresse keine Messages eingehen. Offensichtlich ist etwas Fehlersuche erforderlich.

Einrichtung eines Log-Watchers

Bei Cloud-Systemen gilt, dass Provider in einem Spannungsfeld stehen: Einerseits möchten sie so viele Leistungen wie möglich verkaufen. Andererseits gilt, dass eine zu aggressiv eingestellte Default-Parametrierung zu hohen Kosten, unzufriedenen Kunden und schlechter Presse führt. Aus diesem Grund deaktivieren Anbieter viele Funktionen. Ein häufig deaktiviertes Feature im Fall von AWS ist das Logging.
Zu seiner Freischaltung kehren wir in die AWS-IoT-Konsole zurück, klicken auf *Settings* und scrollen danach in Richtung der *Logs*-Pane nach unten.
Eine frisch konfigurierte AWS-Konsole begrüßt uns dort mit der in **Bild 11** gezeigten Warnung, die auf das Fehlen konfigurierter Rollen hinweist.
Nachdem diese Aufgaben erledigt sind, scheint die in **Bild 12** gezeigte Logging-Hauptkonfiguration auf, die neben der Konfiguration der für den Zugriff auf die Informationen notwendigen IAM-Rolle auch den Grad der Granularität festlegt.
Zunächst klicken wir dabei im Feld *Log Role* auf den Knopf *Create Role*, um den Assistenten zur Generierung einer neuen IAM-Regel anzuwerfen. Im Bereich der Namensvergabe für ebendiese haben Sie dann mehr oder weniger freie Wahl. Der Autor wird sich in den folgenden Schritten für den String *tamsiamlogrole* entscheiden.
Nach dem Bestätigen der Namenseingabe erscheint eines der von weiter oben bekannten grünen Bannern am oberen Bildschirmrand, außerdem wird unter der Combobox die grüne Meldung *Policy attached* sichtbar. Dies weist auf die erfolgreiche Erzeugung der Regel und die Einschreibung der für den Zugriff notwendigen Primitiva hin.
Im nächsten Schritt wechseln wir in die Rubrik *Log Level*, wo wir uns für die Option *Debug* entscheiden. Sie sammelt die maximale Menge an Logging-Informationen.
Nach der Auswahl von *Debug* folgt das Anklicken von *Update*, um die Logging-Regeln zu aktualisieren. Interessant ist, dass das Backend manchmal - wie in **Bild 13** gezeigt - auf das Fehlen der Logging-Regel hinweist.
In diesem Fall handelt es sich häufig um ein Problem mit der angemeldeten Kreditkarte. Öffnen Sie in diesem Fall die allgemeine AWS-Konsole und überprüfen Sie, ob die Kreditkarten-Informationen valide sind. Falls Sie Probleme feststellen, kann es erforderlich sein, eine weitere Rolle anzulegen.
Eine erfolgreich konfigurierte Regel präsentiert sich dann wie in **Bild 14** gezeigt.

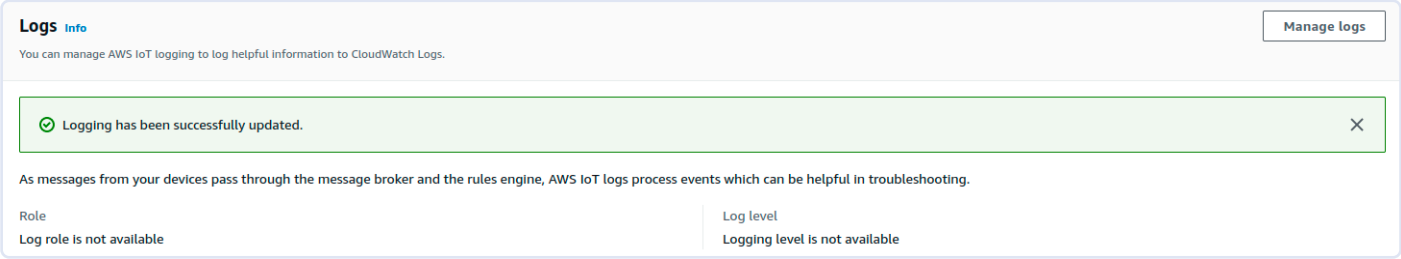


Bild 13. Fehlermeldung „Log role is not available".

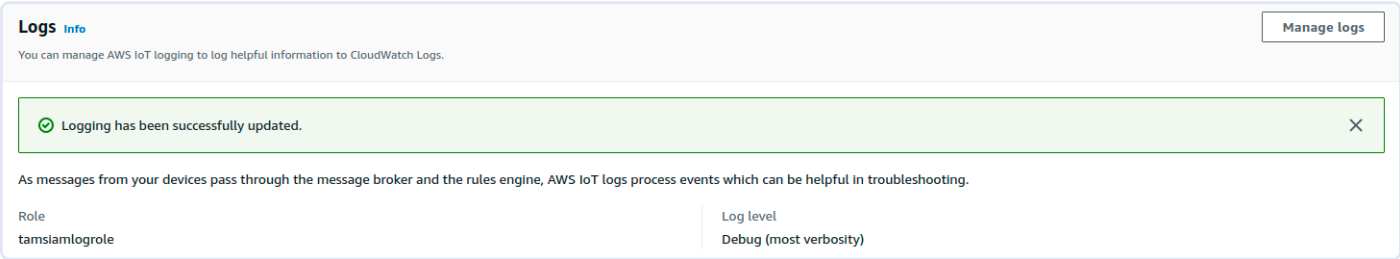


Bild 14. Fehlt die Fehlermeldung, so sind wir zum nächsten Schritt bereit.

Bild 15. Das Lesen der CloudWatch-Logs will gelernt sein.

Im nächsten Schritt ist es empfehlenswert, abermals in den MQTT-Test-Client zurückzukehren und den Arduino wieder einzuschalten. Warten Sie danach einige Zeit, bis ein gutes Dutzend oder mehr Messages eingetroffen sind - die Logging-Funktionen von Cloud-Anbietern (AWS gehörte hier explizit dazu) zeichnen sich im Allgemeinen nicht durch übermäßige Hektik bei der Arbeit aus. Vor der Durchführung der folgenden Schritte kann es notwendig sein, bis zu eine halbe Stunde Totzeit einzuplanen - die Aktualisierung der in *Logs→Log Groups* vorgehaltenen Themen-Liste erfolgt nur sehr langsam.

Sei es wie es sei, im nächsten Schritt ist ein neues Browserfenster erforderlich, in dem Sie *https://console.aws.amazon.com/cloudwatch* eingeben - das Amazon-Backend leitet sie dann automatisch in Richtung der Konsole weiter, wo die für ihren Account vorgesehenen CloudWatch-Logginginformationen zur Aberntung bereitstehen.

Entscheiden Sie sich danach für die Logging-Gruppe *AWSIotLogsV2*, die von AWS IoT Core zum Ablegen von IoT-bezogenen Logging-Informationen verwendet wird. Klicken Sie danach auf einen der Datenströme, um die in ihm enthaltenen Logging-Informationen auf den Bildschirm zu holen.

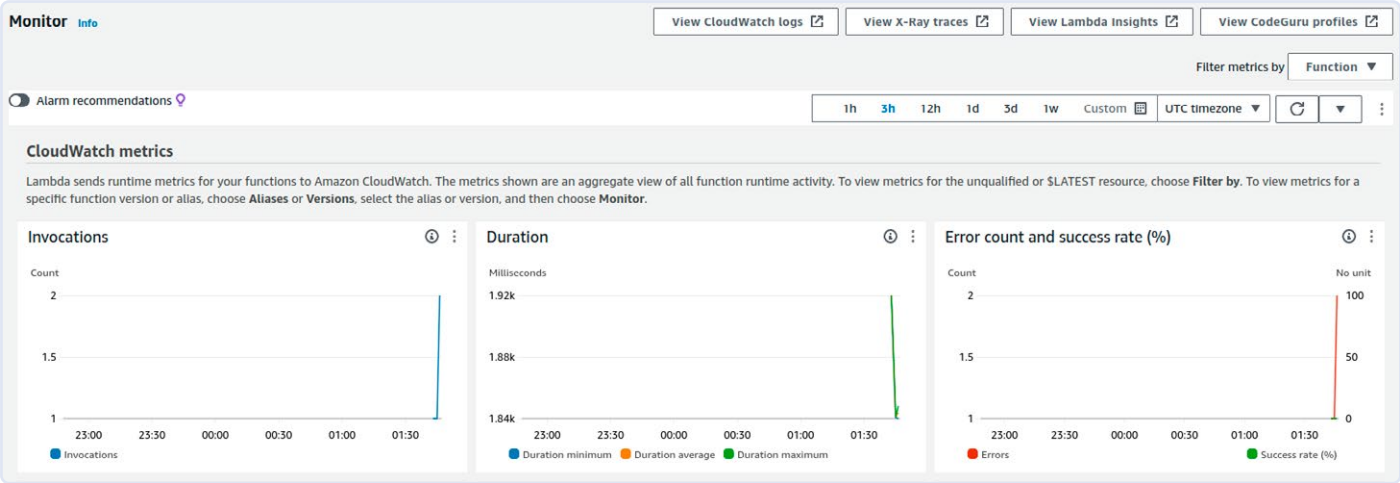


Bild 16. Die Lambda-Funktion startet, scheitert aber bei der Ausführung.

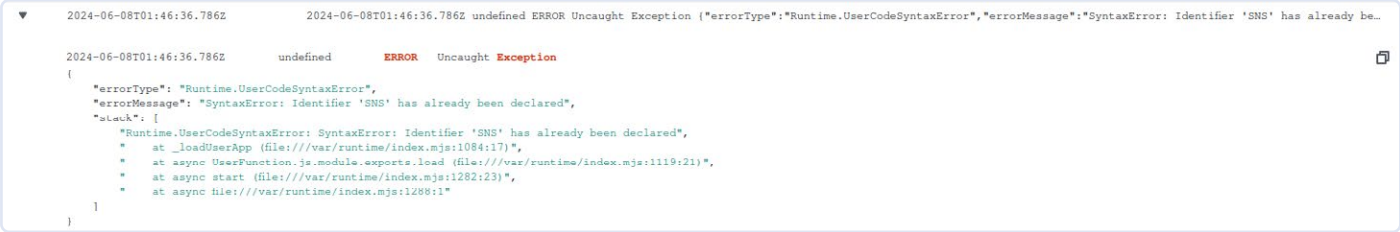


Bild 17. Ein offensichtlicher Fehler der Funktionsdefinition.

THEMENFOKUS
Besuchen Sie unsere Webseite **IoT & Sensoren** für mehr Artikel, Projekte, Nachrichten und Videos.
www.elektormagazine.de/iot-sensoren

Wer die vom Autor übergebene Abfrage verwendet hat, sieht wie in **Bild 15** Error-Meldungen, die auf die Verarbeitung des Operators hinweisen.
Spezifisch sehen wir hier eine Fehlermeldung vom Typ *OperatorEval*, die auf das Eintreffen eines undefinierten Ausdrucks bei der Auswertung des Operators in der Regel-Bedingung hinweist. Dass dies zu einem Abbruch der Regel-Verarbeitung führt, ist logisch, zusätzliche Warnungen, die auf die Nicht-Erfüllung der *Where*-Bedingung hinweisen, sind tautologisch und zu ignorieren.
Zur Behebung des Problems reicht es aus, in die Eigenschaften der Regel zurückzukehren und als SQL-String nun `SELECT messdaten FROM 'TEST' WHERE messdaten.kaefer>0 zu vergeben`.
Nach dem Speichern der Regel empfiehlt sich eine abermalige mehrminütige Ausführung samt Wartezeit, danach das Öffnen der im Lambda-Backend bereitstehenden Metriken für die Ausführung. **Bild 16** zeigt, dass nun gelegentliche Invokationen der Lambda-Funktion sichtbar werden.
Zur weiteren Analyse bietet sich dann das Anklicken des Knopfes *View Cloud Watch-Logs* an - im Log-Datenstrom zeigen sich die in **Bild 17** gezeigten Fehler, die auf die Ungültigkeit des angelegten Codes hinweisen.
Spezifischerweise findet sich der Fehler in der Inklusion des SNS-Objekts, das für das Versenden der E-Mails verantwortlich ist:

```
import { SNS } from '@aws-sdk/client-sns';

const EMAIL = 'tamhan@tamoggemon.com';
const SNS = new SNS();
```

Ein Weg zur Behebung wäre die Umbenennung der Instanz `SNS` in `mySNS` – ein Thema, das allerdings den Rahmen dieses Artikels sprengen würde.

Hier noch ein wichtiger Hinweis:
Der Betrieb von AWS Logging kann durchaus erhebliche Kosten verursachen. Es ist dringend empfehlenswert, die Logging-Funktion durch Auswahl eines hohen Loglevels zu deaktivieren, bevor Sie sich für den Abend ausloggen.

Fülle von Möglichkeiten

Schon aus Platzgründen konnte dieser Artikel nur einen kleinen Teil der Möglichkeiten von AWS IoT vorstellen: Sowohl das Senden von Nachrichten aus der Cloud an das Endgerät als auch die Nutzung von reichen Visualisierungs- oder ML-Payloads konnte wir hier leider nicht behandeln. Angemerkt sei außerdem, dass viele ExpressLink-Module seit einiger Zeit auch Unterstützung für Bluetooth LE anbieten, was die Realisierung von per Bluetooth ansprechbaren Geräten erleichtert. Trotzdem haben wir in diesem Artikel die Grundlagen der Interaktion mit AWS IoT illustriert. Autor und Redaktion hoffen, dass Sie dies zu weiteren eigenen Experimenten animiert!

240240-B-02

Sie haben Fragen oder Kommentare?

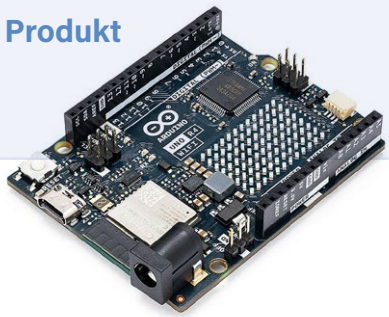
Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Dann senden Sie bitte eine E-Mail an den Autor unter tamhan@tamoggemon.com oder die Elektor-Redaktion unter redaktion@elektor.de.

Über den Autor

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

Passendes Produkt

› **Arduino Uno R4 WiFi**
www.elektor.de/20528



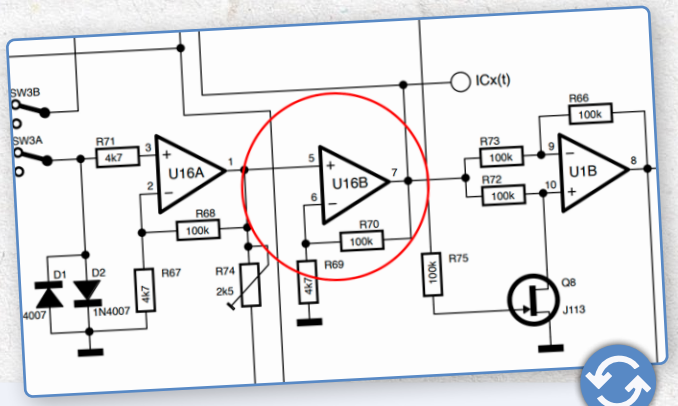
WEBLINKS

[1] AWS IoT Core: Developer Guide (PDF): https://docs.aws.amazon.com/pdfs/iot/latest/developerguide/iot-dg.pdf
[2] AWS IoT Core pricing: https://aws.amazon.com/iot-core/pricing/
[3] Tam Hanna, „AWS für Arduino und Co.", Teil 1, Elektor 7-8/2024: http://www.elektormagazine.de/240240-02
[4] AWS-IoT-ExpressLink Dokumentation: https://docs.aws.amazon.com/iot-expresslink/latest/programmersguide/elpg.html
[5] Spezifikation der JSON-Extensions: https://docs.aws.amazon.com/iot/latest/developerguide/iot-sql-json.html
[6] Dokumentation der Lambda-Funktionen: https://docs.aws.amazon.com/lambda/
[7] AWS-IoT-ExpressLink Dokumentation: Configuration dictionary non-persistent keys: https://docs.aws.amazon.com/iot-expresslink/latest/programmersguide/elpg-configuration-dictionary.html#elpg-table3
[8] Software-Download: http://www.elektormagazine.de/240240-B-02

Projekt 2.0

Korrekturen, Updates und Leserbriefe

Zusammengestellt von Jean-François Simon (Elektor)



In-Circuit LC Meter

Elektor 5-6/2024, S. 8 (230581)

Entschuldigung, es hat sich ein Fehler in den Schaltplan eingeschlichen: Die Eingänge des Operationsverstärkers U16B sind vertauscht. Die korrigierte Version ist oben abgebildet.

Universeller Garten-Logger

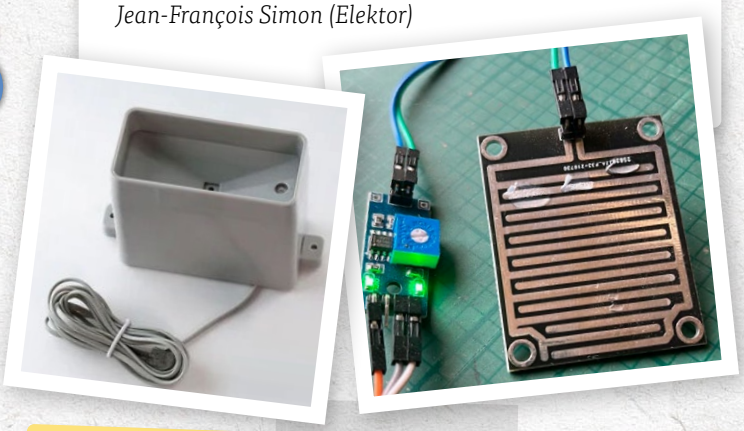
Elektor 7-8/2024, S. 82 (230629)

Sensoren für Wetterstationen

Elektor 7-8/2024, S. 24 (240002)

n dem Artikel *Universeller Garten-Logger* bauen Sie einen Regensensor, der erkennen soll, ob es regnet und versuchen soll, die Regenmenge zu messen. Ich denke, dass dies ein sehr ungenaues Ergebnis liefert, während in der gleichen Elektor-Ausgabe auf Seite 26 ein Regensensor abgebildet ist, der gut funktioniert (wie ich selber feststellen durfte). Dieser Sensor ist bei AliExpress erhältlich, wenn man nach „rain sensor“ sucht. Ich hoffe, dies kann anderen Lesern helfen!
Wim Sanders (Niederlande)

Vielen Dank für den Hinweis! In der Tat ein interessanter Zufall. Die beiden Artikel stammen von verschiedenen Autoren, die unterschiedliche Lösungen verwenden, was auch ein Teil des Reichtums unserer Zeitschrift ist! Ich habe noch nie einen der beiden Sensoren verwendet, kann mich also nicht persönlich dazu äußern. Wie Sie sagen, ist es wahrscheinlich, dass das kommerzielle Modell eine bessere Genauigkeit aufweist.
Jean-François Simon (Elektor)

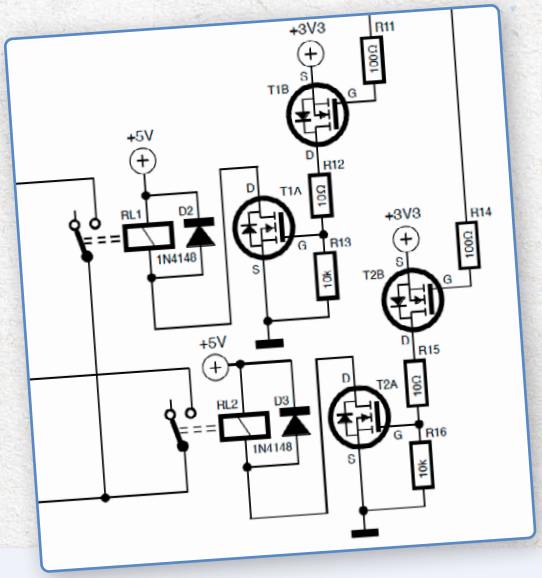


DAB+-Radios

Ich bin seit mehreren Jahren Elektor-Abonnent und habe schon viele FM-Radio-Projekte gesehen. Leider wird der UKW-Dienst hierzulande sehr bald abgeschaltet werden. Wäre es möglich, ein Projekt für ein DAB+-Radio zu entwickeln?
Pierre-René Beljean (Schweiz)

Vielen Dank für die Anregung. Es sind einige Dinge in der Pipeline, aber sie sind noch nicht bereit für die Veröffentlichung in einer bestimmten Ausgabe. Haben Sie auch das Projekt FM/DAB+-Empfänger [1] gesehen, das wir im November 2022 veröffentlicht haben?
Jens Nickel und Clemens Valens (Elektor)

Sie haben eine gute Idee oder wertvolles Feedback für Elektor? Bitte melden Sie sich bitte bei uns unter redaktion@elektor.de. Wir sind gespannt auf Ihre Meinung!



Smarte Rollladensteuerung

Elektor 7-8/2024 Bonusausgabe, S. 6 (230276)

Die Ansteuerung der Relais für die Rollladensteuerung funktioniert nicht. Auch die Verriegelung, die dafür sorgt, dass man nicht beide Relais gleichzeitig einschalten kann, ist nur in der Software gelöst. Dies hätte in der Hardware durch die Verwendung eines freien Wechselkontakts an den Relais gelöst werden können. Außerdem sind die Relaisausgänge beim Allegro ACS712 nicht abgesichert. Ob man den Aufwand braucht, die Laufzeit mit dem ESP-Controller zu messen, ist fraglich. Dies kann auch über die Konfiguration gelöst werden. Ich freue mich auf eine Antwort, danke!

Siggi Lehmann (Deutschland)

Ja, Sie haben Recht! Vielen Dank für den Hinweis auf diesen Fehler, den wir beim Layout übersehen haben. Bitte sehen Sie sich den korrigierten Schaltplanausschnitt an. Vielen Dank auch für Ihre anderen technischen Anmerkungen. Natürlich gibt es oft viele Möglichkeiten, ein bestimmtes Problem zu lösen, und jeder Autor bevorzugt seine eigenen Techniken. Was das Fehlen von Sicherungen betrifft, so haben Sie Recht; aber im Prinzip sollte der Stromkreis für die elektrischen Rollläden auf jeden Fall durch einen eigenen Schutzschalter im Zählerkasten abgesichert werden, zumindest in Frankreich und ich denke, auch in anderen Ländern.

Jean-François Simon (Elektor)

Der Doppler-Bewegungssensor HB100

Elektor 7-8/2023, S. 90 (230205)

Ein interessanter Artikel zum Thema Geschwindigkeitsmessung mit Hilfe des Dopplereffekts. Schnell kam ich auf die Idee, die Geschwindigkeit einer Spur-0-Modellbahn zu messen, um sie maßstabsgetreu fahren zu können. Dazu musste ich erst einmal Material besorgen und Versuche durchführen. Ich habe das Datenblatt der HB100 gelesen und folgendes herausgefunden. Für Deutschland soll eine Sendefrequenz von 9,35 GHz verfügbar sein. Allerdings habe ich keine HB100 mit dieser Frequenz gefunden. Alle Anbieter geben eine Frequenz von 10,525 GHz an, auch die hier in Deutschland ansässigen. Normalerweise würde ich davon ausgehen, dass die HB100 mit 10,525 GHz hier in Deutschland nicht angeboten oder verkauft werden darf. Können Sie mir eventuell eine Bezugsquelle und vergleichbare Radarsysteme mit für Deutschland zugelassenen Frequenzbändern nennen? Vielen Dank, ich lese Elektor seit 1976!

Arno Pelzer (Deutschland)

Vielen Dank für Ihre E-Mail, und auch dafür, dass Sie ein solch treuer Leser sind! Ihre Mitteilung hat mich dazu veranlasst, mich über die HB100 zu informieren; ein in der Tat sehr interessantes Bauteil. Wie Sie habe ich gesehen, dass es mehrere deutsche Anbieter dafür gibt, also gibt es anscheinend einige Leute, die das Risiko eingehen und es benutzen, obwohl es nicht zugelassen ist. Im zugelassenen 9,35-GHz-Band gäbe es den HB410 vom gleichen Hersteller (AgilSense) oder den NJR4181 von Nisshinbo, aber ich habe keinen Anbieter gefunden, der sie auf Lager hatte. Vielleicht können Sie einige zugelassene Modelle im 60-GHz-Band finden? Ich beschäftige mich nicht mit Modelleisenbahnen, aber ich habe gesehen, dass eine beliebte Methode zur Geschwindigkeitsmessung darin zu bestehen scheint, zwei IR-Strahlsensoren in einem bestimmten Abstand (von beispielsweise 200 mm) zu verwenden und die Zeitdifferenz zwischen den Momenten zu messen, in denen der Zug den ersten Strahl unterbricht und dann den zweiten Strahl. Die Messung der Drehgeschwindigkeit der Räder würde wahrscheinlich auch funktionieren, entweder mit einer IR-LED und einem Fototransistor oder einem vorgefertigten Sensor. Das würde funktionieren, solange die Räder nicht zu sehr durchrutschen. Viel Erfolg!

Jean-François Simon (Elektor)

Einstellbare Präzisionsstromquelle

Elektor 4/2014, S. 42 (130287)

Ich bin auf der Suche nach dem Artikel über die Schaltung 130287-1, kann ihn aber in Ihrem Archiv nicht finden. Ich habe jetzt die fertigen Platinen und möchte gerne weiterbauen.

Werner Ehrhardt (Deutschland)

Vielen Dank für Ihre E-Mail. Es stimmt, dass die integrierte Suchmaschine auf der Elektor-Website manchmal nicht sofort das liefert, was Sie suchen. In solchen Fällen empfehle ich Ihnen, eine andere Suchmaschine wie Google mit den Stichworten „130287“ und „Elektor“ zu füttern. In diesem Fall können Sie den Artikel unter [2] finden. Viel Spaß beim Bauen!

Jean-François Simon (Elektor)

262.144 Wege, das Spiel des Lebens zu spielen

Elektor 3-4/2024, S. 56 (230258)

Tolles Projekt. Ich habe einige der Teile bestellt und werde die Liste bald abschließen. Im Schaltplan sieht es so aus, als ob alle Daumenrad-Schalterpositionen mit Dioden belegt sind (was Sinn hat); in der Stückliste sind Dioden jedoch nicht aufgeführt. Ich danke Ihnen für Ihren Rat.

Henry J. Becker

Hallo Henry. Vielen Dank für Ihre Wertschätzung für dieses Projekt und dafür, dass Sie diesen Fehler entdeckt haben. Sie haben Recht, Dioden sind in dieser Schaltung unerlässlich, aber leider sie nicht in der Stückliste aufgeführt. Sie können 36 Dioden des Typs 1N4148 für diesen Zweck verwenden.

Roberto Armani (Elektor)

Reparatur von elektronischen Geräten

Elektor 5-6/2024, S. 26 (240069)

Bei der Fehlersuche in Schaltkreisen, die direkt an das Stromnetz angeschlossen sind, wie zum Beispiel die „heiße" Eingangsseite eines Schaltnetzteils oder ein Gerät, das ohne eigenen Trenntransformator direkt an das Stromnetz angeschlossen ist, ist ein 1:1-Netztrenntransformator unerlässlich. Dies gilt insbesondere bei der Verwendung eines Oszilloskops oder eines anderen Prüfgeräts, das über ein eigenes Netzkabel mit Schutzleiter und entsprechendem Schuko-Stecker verfügt. Diese Transformatoren sind recht teuer und sperrig, da sie die Netzfrequenz von 50 Hz oder 60 Hz übertragen und die Leistungsaufnahme des Geräts, an dem gearbeitet wird, gewährleisten müssen. Ich verwende dazu immer noch einen altehrwürdigen Transformator aus dem Jahr 1964, als viele Radios, Plattenspieler und Fernsehgeräte mit dem Chassis direkt an das Netz angeschlossen waren, wobei die Röhrenheizungen über das Netz in Reihe geschaltet waren und die B+- und Gitterspannungen gleichgerichtet oder sogar vervielfacht wurden.

Michael Kiley

Vielen Dank für Ihre E-Mail. Sie haben recht, Trenntransformatoren können nützlich sein, und ich besitze und benutze tatsächlich einen. Allerdings wollte ich sie in dem Artikel nicht besonders hervorheben, da dies ein weiteres Wespennest ist, das einen eigenen Artikel verdienen würde. Je nach Hersteller können einige Transformatoren unterschiedlich verdrahtet sein. Manchmal kann eine Klemme der Sekundärseite mit der Erde verbunden sein, was den Zweck verfehlt; manchmal ist der Schutzleiter mit der PE-Klemme des Ausgangssteckers/der Ausgangssteckdose verbunden, manchmal nicht; und so weiter. Außerdem kann ihre Verwendung bei unsachgemäßer Anwendung sogar gefährlich sein. In vielen Haushalten sind FI-Schutzschalter installiert. Wenn der Benutzer beim Prüfen von Stromkreisen ohne Transformator versehentlich ein stromführendes Teil berührt, löst der FI-Schutzschalter in vielen Fällen aus und verhindert einen Stromschlag. Mit einem Trenntansformator kann der Benutzer ein „stromführendes" Teil mit einer Hand berühren, ohne dass er dies bemerkt. Wenn er jedoch dann mit der anderen Hand einen anderen Teil des Stromkreises berührt (was ohnehin schlecht ist, aber Unfälle passieren nun einmal), kann er ernsthaften Schaden erleiden, weil aufgrund des Trenntransformators der FI-Schutzschalter nicht auslöst. Das ist ein heikles Thema, und jeder Elektroniker hat seine eigenen Vorlieben und Gewohnheiten. Arbeiten Sie sicher!

Jean-François Simon (Elektor)

10-MHz-Referenzgenerator

Elektor 5-6/2024, S. 96 (180385)

Hallo, der Artikel von Herrn Rosenkränzer ist wieder einmal genau nach meinem Geschmack, tolle Ideen und Umsetzung. Elektor hat immer relevante Projekte in Sachen Messtechnik oder Amateurfunktechnik. Super!

Jens Lemkamp, DL1LEP (Deutschland)

Hallo, Herr Lemkamp. Vielen Dank für Ihr Feedback. Es freut uns sehr, dass es Ihnen gefallen hat!

Jens Nickel (Elektor)

RG – 240397-02

WEBLINKS

[1] Yves Bourdon, „UKW/DAB+-Empfänger", Elektor 11-12/2022: http://www.elektormagazine.de/220249-02

[2] Henry Morizot, „Einstellbare Präzisionsstromquelle", Elektor 4/2014: https://www.elektormagazine.de/magazine/elektor-201404/26200

2024: Eine Odyssee in die KI

Beschleuniger: Desktop vs. Embedded, ein Blick auf einige Optionen

Von Brian Tristam Williams (Elektor)

Künstliche Intelligenz (KI) entwickelt sich rasant weiter, und die Wahl des richtigen KI-Beschleunigers ist entscheidend. Im Folgenden werden einige wichtige Beschleuniger vorgestellt und ihre Funktionen und Leistung verglichen, um eine fundierte Entscheidung für Ihre speziellen KI-Anforderungen zu treffen.

Bild 1. Raspberry Pi AI Kit.

Warum brauchen wir KI-Beschleuniger? Sicherlich würde jede CPU oder jeder Mikrocontroller genau die Berechnungen durchführen, die auch ein KI-Beschleuniger durchführen könnte. Allzweck-CPUs und Mikrocontroller sind jedoch für eine breite Palette von Aufgaben ausgelegt, aber meist nicht sehr effizient bei spezifischen KI-bezogenen Berechnungen.

Ein KI-Beschleuniger ist eine spezialisierte Hardwarekomponente zur Beschleunigung von Aufgaben der künstlichen Intelligenz, die sich in erster Linie auf die Inferenzphase konzentriert, in der KI-Modelle auf der Grundlage neuer Daten Vorhersagen machen. Im Gegensatz zu Allzweckprozessoren sind KI-Beschleuniger für die Durchführung umfangreicher Matrixoperationen und Parallelverarbeitung optimiert, die für Aufgaben wie Bilderkennung, Verarbeitung natürlicher Sprache und andere Anwendungen des maschinellen Lernens unerlässlich sind. Dadurch sind sie im Vergleich zu Standard-CPUs wesentlich schneller und effizienter bei der Durchführung dieser speziellen Berechnungen.

Im Fortschreiten der KI gewinnt Edge Computing - die lokale Verarbeitung von Daten auf kleinen Geräten statt auf zentralen Cloud-Servern - zunehmend an Bedeutung. Dieser Wandel verbessert die Echtzeit-Reaktionsfähigkeit, den Datenschutz und die Gesamteffizienz. Um die Anforderungen zu erfüllen, wurden verschiedene Edge-KI-Beschleuniger sowohl für eingebettete Systeme als auch für PCs entwickelt.

Ob Sie leistungsstarke Lösungen für komplexe Aufgaben oder leichtgewichtige Geräte für Echtzeit-Außeneinsätze benötigen, die Wahl des richtigen Tools für Ihre spezifischen Anforderungen kann schwierig sein. Deshalb geben wir einen Überblick über einige der führenden Edge-KI-Beschleuniger, wobei wir ihre Funktionen, typische Anwendungen und Leistungsfähigkeit hervorheben.

Warum dieser Vergleich?

Für Anwendungen wie die Transkription von Sprache in Text (zum Beispiel mit Whisper [1]) und die automatische Videobearbeitung und die Klassifizierung von archivierten Videoaufnahmen muss ich oft große Datenmengen verarbeiten. Für diese Aufgaben sind Lösungen auf PCs mit leistungsstarken GPUs wie meine NVIDIA RTX 4070 aufgrund ihrer Rechenleistung und Flexibilität unverzichtbar. Für Echtzeiterkennung im Feld oder in industriellen Umgebungen sind dagegen eingebettete Lösungen wie das Raspberry Pi AI-Kit oder das Coral-Dev-Board aufgrund ihrer Kompaktheit und ihres geringeren Energiebedarfs praktischer. Dieser Vergleich soll helfen, die besten Tools für verschiedene Szenarien zu bestimmen.

Eingebettete Edge-AI-Beschleuniger

Raspberry Pi AI-Kit mit integriertem Hailo-8L

Das kürzlich vorgestellte Raspberry Pi AI-Kit [2] verbindet den KI-Beschleuniger Hailo-8L mit dem Raspberry Pi 5 und ermöglicht damit leistungsstarke KI-Funktionen in einem kostengünstigen, kompakten Formfaktor. Es ist eine erschwingliche Lösung, um KI verschiedenen Raspberry-Pi-Projekten hinzuzufügen. Der KI-Beschleuniger Hailo-8L wird in den M.2-Steckplatz des mitgelieferten Raspberry-Pi-M.2-HAT+ gesteckt (**Bild 1**). Dort bietet er 13 Tera-Operationen pro Sekunde (TOPS) und eignet sich damit für Aufgaben wie Objekt-

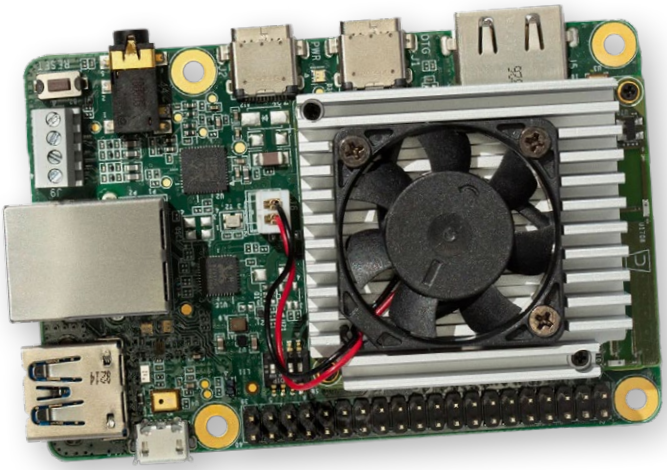


Bild 2. Dev-Board von Coral. Quelle: coral.ai.

erkennung in Echtzeit, semantische Segmentierung, Posenabschätzung zur Erkennung von Gesten und Mimik. Der niedrige Strombedarf des Kits ist ideal für batteriebetriebene Anwendungen, und seine kompakten Abmessungen lassen sich gut mit dem offiziellen Raspberry-Pi-Kamerazubehör kombinieren. Obwohl das Kit nicht so leistungsfähig ist wie einige High-End-NPUs, eignet es sich perfekt für leichte KI-Aufgaben, Bildungsprojekte und Hobbyanwendungen.

Coral Dev-Board

Das 2019 von Coral vorgestellte Dev-Board [3] wurde entwickelt, um TensorFlow-Lite-Modelle effizient als eigenständige Lösung auszuführen. In einem Raspberry-Pi-ähnlichen Formfaktor (**Bild 2**) verfügt es über einen Edge-TPU-Coprozessor, der 4 TOPS liefert. Das Board eignet sich hervorragend zur Bildklassifizierung, Objekt- und Spracherkennung und ist damit ideal für IoT-Anwendungen. Die geringe Stromaufnahme ist ein großer Vorteil, insbesondere für eingebettete Systeme. Allerdings ist das Dev-Board in erster Linie auf TensorFlow-Lite-Modelle beschränkt und bietet im Vergleich zu einigen anderen Beschleunigern weniger Rechenleistung. Trotzdem lässt es sich nahtlos in die Cloud-Dienste von Google integrieren und bietet eine effiziente Leistung für grundlegende KI-Anwendungen in eingebetteten Systemen.

Coral USB-Beschleuniger

Wenn Sie sich bereits für eine Computing-Plattform entschieden haben und noch etwas externe KI-Leistung benötigen, bietet sich der USB-Accelerator [4] von Coral an. Dieses kompakte Gerät, das ungefähr die Größe eines Standard-USB-Sticks hat (**Bild 3**), bringt die Leistung der Edge-TPU von Google über eine USB-3.0-Verbindung auf jeden kompatiblen Computer. Der Beschleuniger bietet eine Leistung von 4 TOPS und nimmt dabei eine Leistung von nur 2 W auf. Der USB-Beschleuniger ist ideal für Entwickler, die bestehende Systeme mit KI-Funktionen ausstatten oder Prototypen von Edge-KI-Anwendungen entwickeln möchten, ohne sich auf ein komplettes Entwicklungsboard festlegen zu müssen. Er eignet sich hervorragend für Aufgaben wie Bildklassifizierung und Objekterkennung und ist damit ein vielseitiges Werkzeug für KI-Experimente und kleine Implementierungen.

Bild 3. USB-Beschleuniger von Coral.

NVIDIA Jetson

Bei der Jetson-Reihe von NVIDIA handelt es sich um eine Reihe von Embedded-Computing-Boards für KI-Anwendungen im Edge-Bereich. Dieses Ökosystem umfasst mehrere Produkte für unterschiedliche Leistungsanforderungen und Anwendungsfälle:

- Jetson Nano: Das Einsteigermodell, geeignet für kleinere KI-Projekte und Prototyping
- Jetson Xavier NX: Eine Mittelklasse-Option mit deutlich mehr Leistung als der Nano
- Jetson AGX Xavier: Ein Hochleistungsmodul für anspruchsvolle KI-Anwendungen
- Jetson AGX Orin: Das neueste und leistungsstärkste Modul der Reihe für erweiterte Robotik und autonome Maschinen.

Diese Module variieren in ihrer Rechenleistung und reichen von 0,5 TOPS beim Nano bis über 200 TOPS beim AGX Orin. Alle Jetson-Geräte laufen auf dem JetPack-SDK von NVIDIA, das Bibliotheken für Deep Learning, Computer Vision, Accelerated Computing und Multimedia-Verarbeitung enthält. Diese gemeinsame Software-Umgebung ermöglicht eine einfachere Skalierung und Bereitstellung für die gesamte Jetson-Familie.

Die Jetson-Familie eignet sich für eine Vielzahl von Anwendungen wie Robotik, Drohnen, intelligente Videoanalyse und tragbare medizinische Geräte. Ihre Skalierbarkeit macht sie zu einer vielseitigen Wahl für Unternehmen, die KI-Anwendungen über verschiedene Leistungsstufen hinweg entwickeln und einsetzen möchten.

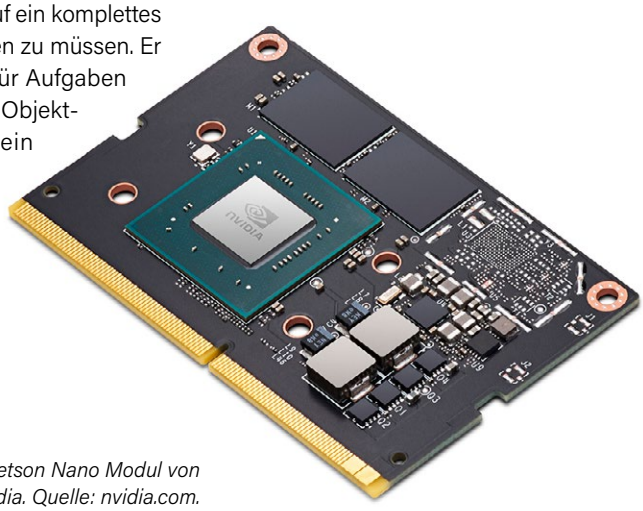


Bild 4. Jetson Nano Modul von Nvidia. Quelle: nvidia.com.

Bild 5. Neural Compute Stick 2 von Intel (Intel NCS2).

Das 2019 eingeführte Einsteigermodell Jetson Nano (**Bild 4**) brachte KI-Rechenfähigkeiten in kleinen und preiswerten Formfaktoren. Es verfügt über einen Maxwell-Grafikprozessor mit 128 Kernen, eine Quad-Core-ARM-Cortex-A57-CPU und 4 GB RAM und liefert 472 GFLOPS (0,47 TOPS). Der Jetson Nano ist für kleine, aber anspruchsvolle KI-Aufgaben in eingebetteten Systemen gut geeignet. Er kann mehrere neuronale Netze gleichzeitig verarbeiten, was ihn zu einer flexiblen Option für verschiedene KI-Anwendungen macht. Allerdings ist seine Stromaufnahme höher als bei anderen eingebetteten Optionen, und er hat einen größeren Formfaktor.

Der Nano ist ein KI-Modul mit einem 260-poligen Edge-Verbinder, so dass es nicht möglich ist, ihn eigenständig zu verwenden, ohne ihn in etwas hineinzustecken. Als Einsteiger sollten Sie mit einem Dev-Kit beginnen, das Sie unter **Passende Produkte** finden. Weitere Informationen zu den anderen Boards der Jetson-Reihe finden Sie auf der Vergleichsseite von NVIDIA unter [6].

Intel Neural Compute Stick 2

Der Neural Compute Stick 2 (NCS2), der 2018 von Intel vorgestellt wurde, ist ein weiterer „portabler" USB-basierter KI-Beschleuniger [7]. Er ist mit der hauseigenen Movidius-Myriad-X-VPU ausgestattet und bietet eine Leistung von etwa 1 TOPS. Er ist ideal für KI-Prototyping, Forschung und die Erweiterung bestehender PCs mit KI-Funktionen. Seine Portabilität und Benutzerfreundlichkeit machen ihn zu einer beliebten Wahl für Entwickler und Forscher. Allerdings ist er für die Stromversorgung und Datenverarbeitung auf einen Host-PC angewiesen und bietet im Vergleich zu High-End-GPUs eine geringere Rechenleistung.

PC-basierte Edge-KI-Beschleuniger

NVIDIA-GPUs

Die oben erwähnte GPU RTX 4070 [8] von NVIDIA gehört zur RTX 40-Serie, die 2022 auf den Markt kam, und ist ein Kraftpaket sowohl für Spiele als auch für professionelle KI-Anwendungen. Dieser High-End-Grafikprozessor, der auf Grafikkarten verschiedener Anbieter erhältlich ist, verfügt über 29,8 TFLOPS (entsprechend 29.800 GFLOPS) und 12 GB GDDR6-Speicher. Die GPU eignet sich perfekt für das Training und die Inferenz großer neuronaler Netze, Hochleistungsspiele und Aufgaben zur Erstellung von Inhalten wie Videobearbeitung und 3D-Rendering. Die umfangreiche Softwareunterstützung und die außergewöhnliche Rechenleistung prädestinieren die GPU für anspruchsvolle KI-Anwendungen, einschließlich Sprache-zu-Text-Transkription und Videobearbeitung. Wenn Ihr Budget es zulässt, können Sie auf die RTX 4090 aufsteigen, um noch mehr Leistung zu erhalten, oder mit einer RTX 3080 sparen, die immer noch eine beachtliche Leistung bietet. Aufgrund der hohen Stromaufnahme und der hohen Kosten eignen sich diese NVIDIA-GPUs jedoch eher für größere Desktop-Workstations.

Mit der RTX-Serie, die NVIDIA mit dem Slogan „the ultimate in ray tracing and AI" bewirbt, ist im oberen Preissegment angesiedelt. Nachteilig kann sein, dass, wenn Sie sich kurz von Ihrer Workstation entfernen, das eine oder andere eifrige Kind im Haushalt Ihren Platz kapert, um den neuesten Ego-Shooter mit hohen FPS-Raten zu spielen, der auf diesen Biestern verfügbar ist.

Apple Chips der M-Serie

Diejenigen, die in das Apple-Ökosystem investiert haben, sollten wissen, dass der M1-Chip, der als Apples erster speziell für den Mac entwickelter Chip angepriesen wird, im Jahr 2020 eingeführt wurde. Der nachfolgende M2-Chip wurde 2022 auf den Markt gebracht, und beide verbinden KI-Beschleunigung direkt mit der CPU. Der M1-Chip verfügt über eine Neural Engine mit acht Kernen und eine Leistung von 11 TOPS, während der M2-Chip eine Neural Engine mit 16 Kernen und 15,8 TOPS bietet. Diese Chips sind in Geräten wie dem MacBook Pro und dem Mac Mini zu finden und bieten geräteinterne KI-Verarbeitung für Anwendungen wie Siri, Bildverarbeitung und Augmented Reality. Die nahtlose Integration in macOS und iOS macht diese Chips äußerst effizient für KI-Anwendungen im Consumer-Bereich, insbesondere für diejenigen, die im Apple-Ökosystem zu Hause sind.

M3, M3 Pro und M3 Max wurden 2023 angekündigt und sollen Verbesserungen auf der ganzen Linie erbringen. Die M3-Chips waren die ersten 3-Nanometer-Chips und verfügten über eine neue GPU-Architektur, die Geschwindigkeitsverbesserungen von bis zu 30 % beziehungsweise 50 % gegenüber dem M2 und M1 ermöglicht. Mit schnellerer CPU-Leistung, einer effizienteren Neural Engine und einer GPU-Architektur der nächsten Generation, die fortschrittliche Funktionen wie dynamisches Caching und hardwarebeschleunigtes Raytracing (Strahlenverfolgung) unterstützt, eignet sich die M3-Serie besonders gut für intensive Aufgaben wie High-End-Grafikverarbeitung, KI/ML-Workloads und komplexe Rechenaufgaben.

Die Tinte der M3-Ankündigung war kaum getrocknet, als Apple im Mai 2024 den M4-Chip ankündigte, aber dieser ist derzeit nur im iPad Pro des Unternehmens verfügbar. Es wäre interessant zu sehen, ob ein Drittentwickler einen Anwendungsfall findet, um ein teures iPad Pro für eine kommerzielle oder industrielle Aufgabe im Bereich des maschinellen Lernens einzusetzen.

Wenn Sie Ihre schweren Aufgaben in der Apple-Welt erledigen, werden Sie von einem MacBook mit einem M2- oder M3-Chip [9] für Ihre KI-Anwendungen nicht enttäuscht sein. Welchen Chip Sie unter der Haube haben wollen und werden, hängt von Ihrem Budget und Ihrer Anwendung ab.

Eingebettet vs. PC-basiert: Welche Wahl ist die richtige?

Die Wahl zwischen eingebetteten und PC-basierten Edge-Computing-Lösungen hängt weitgehend von den spezifischen Anforderungen Ihrer Anwendung ab.

Eingebettete Edge-KI

- Am besten geeignet für: Echtzeitanwendungen mit geringem Strombedarf, bei denen Größe und Stromverbrauch kritisch sind, beispielsweise Smart Home-Geräte, Wearables und autonome Sensoren
- Vorteile: Geringere Stromaufnahme, kompakte Abmessungen, kostengünstig.

PC-basierte Edge-KI

- Ideal für: Anwendungen, die eine hohe Rechenleistung und Flexibilität erfordern, zum Beispiel Desktop-KI-Anwendungen, komplexe Datenanalysen sowie die Entwicklung und Prüfung von KI-Modellen
- Vorteile: Hohe Leistung, größere Flexibilität, Fähigkeit, komplexere Aufgaben zu bewältigen

Die Wahl ist keine Qual

Eingebettete und PC-basierte Edge-KI-Lösungen haben ihre spezifischen Vor- und Nachteile. Eingebettete Beschleuniger eignen sich perfekt für stromsparende Echtzeitanwendungen, während auf Workstation/Laptop basierenden Lösungen wie RTX-GPUs von NVIDIA und die M-Chipreihe von Apple eine hohe Leistung für anspruchsvollere Aufgaben bieten. Wenn Sie die Stärken und Grenzen der einzelnen Lösungen kennen, können Sie die richtige Lösung für Ihre spezifischen Anforderungen herauspicken und die Leistung der KI nutzen, um intelligentere, effizientere Systeme zu schaffen.

Dieser Vergleich ist keineswegs erschöpfend, aber er betrachtet einige der beliebtesten Edge-KI-Beschleuniger, die ich ausprobiert habe. Jeder hat seine Vor- und Nachteile - entscheidend ist, dass Sie den für Ihre Anwendung am besten geeigneten auswählen. Umfassendere Spezifikationen, Vergleiche und weiteres Lesematerial finden unter **Weblinks**.

RG – 230181-H-02

Über den Autor

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

THEMENFOKUS

Besuchen Sie unsere Seite **Embedding & KI** für Artikel, Projekte, Nachrichten und Videos.

www.elektormagazine.de/**embedded-ki**

Sie haben Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Schreiben Sie bitte dem Autor unter brian.williams@elektor.com.

Passende Produkte

- **Raspberry Pi AI-Kit**
  www.elektor.de/20879
- **Google Coral USB-Beschleuniger**
  www.elektor.de/19366
- **Waveshare Jetson Nano Development-Kit Lite**
  www.elektor.de/20761

WEBLINKS

[1] Whisper (Sprache-zu-Text von OpenAI): https://openai.com/blog/whisper
[2] Raspberry Pi AI-Kit: https://raspberrypi.com/products/ai-kit
[3] Coral Dev Board: https://coral.ai/products/dev-board
[4] Coral USB-Accelerator: https://coral.ai/products/accelerator
[5] Apple M3-Chipserie: https://t1p.de/3q0ja
[6] Jetson-Serie eingebetteter Compute-Module von Nvidia: https://developer.nvidia.com/embedded/jetson-modules
[7] Neural Compute Stick 2 von Intel (Intel NCS2): https://t1p.de/d6zrp
[8] GPU-Serie GeForce RTX 40 von NVIDIA: https://nvidia.com/en-us/geforce/graphics-cards/40-series
[9] Apple Mac Modelle, die derzeit auf M2- und M3-Chips basieren: https://apple.com/mac/compare

Der Elektor Store

Nie teuer, immer überraschend!

Der Elektor Store hat sich vom Community-Store für Elektor-eigene Produkte wie Bücher, Zeitschriften, Bausätze und Module zu einem umfassenden Webshop entwickelt, der einen großen Wert auf überraschende Elektronik legt.

Wir bieten die Produkte an, von denen wir selbst begeistert sind oder die wir einfach ausprobieren wollen. Wenn Sie einen Produktvorschlag haben, sind wir hier erreichbar (sale@elektor.de).

FNIRSI HRM-10 Innenwiderstandstester

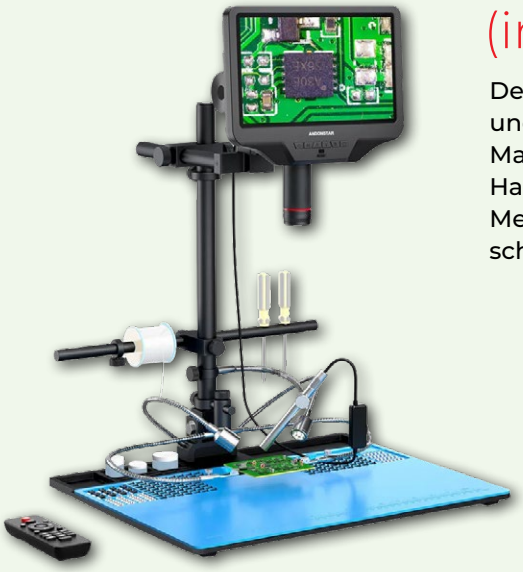
Das FNIRSI HRM-10 ist ein tragbares, hochpräzises Innenwiderstandsmessgerät. Dieses Gerät bietet echte Vierleitermessung und ist sowohl auf Genauigkeit als auch auf Benutzerfreundlichkeit ausgelegt. Es misst automatisch gleichzeitig Innenwiderstands- und Spannungswerte und zeigt die Ergebnisse deutlich auf dem HD-Farbdisplay an. Benutzer haben die Möglichkeit, Spannungs- und Widerstandsbereiche manuell an ihre Bedürfnisse anzupassen. Das Gerät verfügt außerdem über einen Sortiermodus, der anhand vom Benutzer festgelegter Schwellenwerte automatisch die guten und schlechten Batterien filtert. Darüber hinaus unterstützt es die Speicherung historischer Daten und ermöglicht den Export von Messdatensätzen im Tabellenformat.

Preis: ~~59,95 €~~

Sonderpreis: 49,95 €

www.elektor.de/20947

Andonstar AD409 Max-ES 10,1” Digital-Mikroskop (inkl. Endoskop)



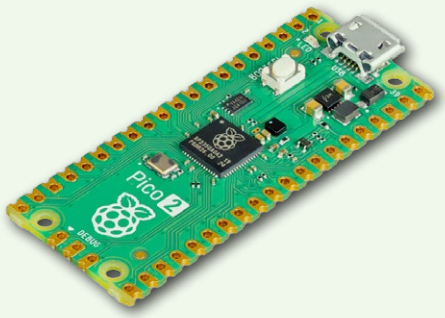
Der Andonstar AD409 Max-ES verfügt über eine hochwertige Metalllinse und ein einzigartiges UV-Filterdesign. Er besteht aus erstklassigen Materialien in Industriequalität und bietet unübertroffene Präzision und Haltbarkeit und sorgt so für ein zuverlässiges Produkterlebnis. Der vor der Metalllinse positionierte UV-Filter blockiert Löthitze, Rauch und Staub, schützt die Linse und macht sie perfekt für Löt- und Wartungsprofis.

Preis: ~~499,00 €~~

Sonderpreis: 349,00 €

www.elektor.de/20852

Raspberry Pi Pico 2



Preis: 5,95 €

www.elektor.de/20950

Raspberry Pi 5 (2 GB RAM)

Preis: 54,95 €

www.elektor.de/20951

KiCad Like A Pro (Bundle)

Preis: ~~94,95 €~~

Sonderpreis: 74,95 €

www.elektor.de/20942

The Book of 555 Timer Projects

Preis: 29,95 €

Mitgliederpreis: 26,96 €

www.elektor.de/20948

ESP32-Reichweiten-verlängerer

Eine einfache Antennenmodifikation

Von Peter Neufeld (Deutschland)

ESP32-Module haben oft „integrierte“ (auf der Platine aufgedruckte) Antennen, was manchmal zu einer schlechten WLAN-Reichweite und anderen Problemen führt. Es gibt eine Lösung: Verwenden Sie eine externe Antenne! Leider hat nicht jedes Modul einen Anschluss für eine externe Antenne, aber lassen Sie sich davon nicht abschrecken! Eine einfache Lösung wird in diesem Artikel beschrieben: Hack the Board!

Als ich mit ESP32-Modulen, insbesondere den ESP32-CAMs experimentierte, habe ich mir eine größere WLAN-Reichweite gewünscht. Eine externe Antenne könnte helfen, aber nur wenige Module sind damit ausgestattet. Die Kosten und der mechanische Aufwand stehen vielleicht nicht im Verhältnis zum Nutzen.

Die ursprüngliche gedruckte Antenne auf einer Standard-ESP32-Platine ist eine mäanderförmige, invertierte F-Antenne mit einer Wellenlänge von λ/4 (siehe **Bild 1**). Dieser Antennentyp bietet ein ausgewogenes Verhältnis zwischen Größe, nahezu omnidirektionaler Strahlungscharakteristik, akzeptabler Leistungseffizienz und WLAN-Reichweite innerhalb der Grenzen der gegebenen HF-Leistung und Empfangsempfindlichkeit des ESP32. Die Links [1]...[5] bieten nützliche Informationen über solche Antennen.

Bild 1. Die F-förmige λ/4-Antenne „gedruckt“ auf einem ESP32-BoB.

Der Hack

Meine Idee war, die WLAN-Reichweite des Moduls zu erhöhen, indem ich die integrierte Viertelwellenantenne mit einer neuen, externen Halbwellenantenne koppelte, die größer und nicht, um sie besser platzieren zu können, durch Mäander verkürzt ist,. Durch das Aufkleben des zusätzlichen Antennendrahtes auf die Platine wurden zudem keine irreversiblen Änderungen vorgenommen. Das Ergebnis ist in **Bild 2** dargestellt.

Unter der Annahme, dass die gedruckte Antenne ein 1/4-Wellenlängen-Monopol für 2,4 GHz ist, habe ich ihre Länge ausgemessen und in erster Näherung verdoppelt. Dies ergab eine Halbwellenantenne aus isoliertem Schaltdraht (Durchmesser 0,5 mm) mit einer Länge von 62 mm. Die Hälfte des zusätzlichen Antennendrahtes folgt der Mäanderform der gedruckten F-Antenne in einem Abstand, der durch die Isolierung des Drahtes vorgegeben wird. Die andere Hälfte ist „frei“ montiert. Beides führt zu einer starken induktiven und kapazitiven Kopplung der beiden Antennen und bildet ein größeres, optimiertes Strahlerelement. Der beste Klebepunkt im HF-Sinne ist die niederohmige Mitte der neuen Halbwellenantenne, genau in der rechten Ecke des ESP32-Moduls. Dadurch wird verhindert, dass der hochohmige Teil durch zusätzliches HF-dämpfendes Klebematerial beeinträchtigt wird.

Bild 2. Ein zusätzlicher Antennendraht ist auf die Platine geklebt.

Die Messung

Um den Effekt zu messen, habe ich ein modifiziertes ESP32-Board neben einem unmodifizierten Board als Referenz platziert, damit beide Module ungefähr die gleichen Empfangs- und Sendebedingungen in Bezug auf den Access Point haben und sich gleichzeitig nur minimal gegenseitig beeinflussen. Diese Anordnung war sehr einfach überall zu platzieren und half dabei, die WLAN-Bedingungen für beide Module gleichzeitig zu untersuchen.

Ich hatte die Idee, dass das zweite ESP32-Modul seine Daten an das erste Modul sendet und das erste Modul eine Webseite mit beiden Datenleitungen grafisch darstellt, die in einem

Bild 3. Messaufbau mit modifiziertem und unmodifiziertem ESP32 und einem Tablet, das die empfangenen RSSI-Werte anzeigt.

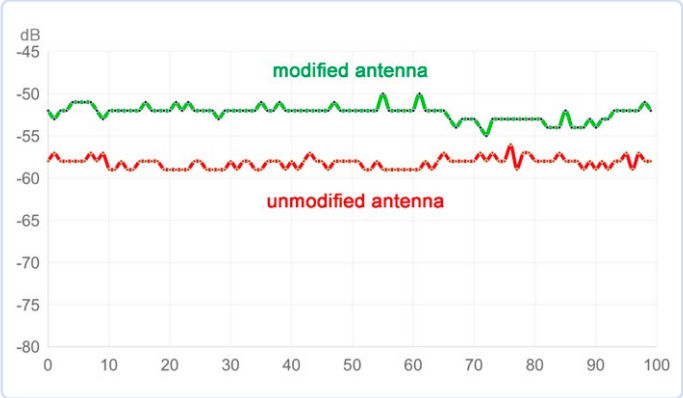


Bild 4. Messergebnis mit Gewinn: Durch die Modifikation beträgt die mittlere Verstärkung etwa +6 dB.

Webbrowser betrachtet werden kann.
Dazu habe ich einen Code geschrieben, der für beide ESP32-Module verwendet werden kann (siehe **Listing 1**, verfügbar unter [6]). Durch Auskommentieren einiger Zeilen wird der Code von einer Version für das erste Modul in eine Version für das zweite Modul umgewandelt. Der Code zeigt die von beiden Geräten gemessene WLAN-Feldstärke des Access Points auf einer Webseite an, die auch in einem Browserfenster dargestellt werden kann.
Natürlich müssen sich beide ESP32 am selben Access Point (Router) anmelden. Der von mir verwendete Annex32-BASIC-Interpreter besitzt einen eingebauten Mechanismus, um einen ESP32 an einem Router-Netzwerk anzumelden. Zuerst wird ein Access Point vom ESP32 selbst eingerichtet, dann wird eine Konfigurationswebseite angeboten, auf der der Benutzer die SSID und das Passwort des Routers eingeben kann - diese sensiblen Daten sind also nicht im Listing zu finden.
Bild 3 zeigt den kompletten Messaufbau. Modul 1 mit modifizierter Antenne zeichnet seinen eigenen WLAN-RSSI-Wert auf und fordert einen String mit dem aufgezeichneten RSSI-Wert vom zweiten, nicht modifizierten Modul an. Das Browser-Fenster eines Tablet-PCs zeigt zwei durchlaufende Kurven mit den jeweils letzten 100 Messungen, die einen Zeitraum von etwa 50 s abdecken. **Bild 4** zeigt eine typische Messreihe. Hier beträgt der Gewinn der modifizierten Antenne etwa +6 dB gegenüber der Referenz. Nicht schlecht, oder?
Annex32 [7] ist übrigens ein eigenständiger BASIC-Interpreter und eine IDE, die komplett auf dem ESP32 läuft. Damit ist es einfach, Webseiten zu erstellen, die im Browser eines externen Gerätes angezeigt werden können, auch mit Unterstützung von JavaScript-Funktionen. Der Grund für die Verwendung von Annex32 sind spezielle grafische Anzeigefunktionen, die hier benötigt werden. Für Annex32 gibt es auch ein aktives Forum [8] und ein ausgezeichnetes Online-Handbuch [9].
Ich habe den Effekt der modifizierten Antenne getestet und die WLAN-Signalstärke des Access Points unter verschiedenen Bedingungen aufgezeichnet. Es stellte sich heraus, dass die modifizierten Module mit verlängerten, nicht abgewinkelten Antennendrähten das Signal insgesamt um mindestens +3 dB verbesserten. Unter stark reflektierenden Umgebungsbedingungen erreichten die Verbesserungen fast +10 dB.
Diese groben Messungen zeigen aber auch deutlich, dass sich die verbesserten Bedingungen in den Nullstellen des Antennendiagramms widerspiegeln, die nun erwartungsgemäß stärker ausgeprägt sind. Bei dieser Antenne ist zu berücksichtigen, dass die Richtwirkung im Guten wie im Schlechten größer ist. Ein Teil des verbesserten Gewinns ist sicherlich auf das geänderte Antennendiagramm und die starke Kopplung mit einer teilweise freistehenden, effektiveren Antenne zurückzuführen.

Untersuchungen und Schlussfolgerungen

Ich habe mit dem freien Ende der Antenne experimentiert, das in Bezug zur Platine um 90 Grad nach oben zeigt (siehe **Bild 5**). Dies änderte das resultierende Antennendiagramm und sollte weiter untersucht werden, um optimale Ergebnisse zu erzielen. Diese Variation kann in bestimmten Umgebungen hilfreich sein, zum Beispiel wenn das Modul auf einer stark bevölkerten Platine mit einer freistehenden Antenne platziert wird.

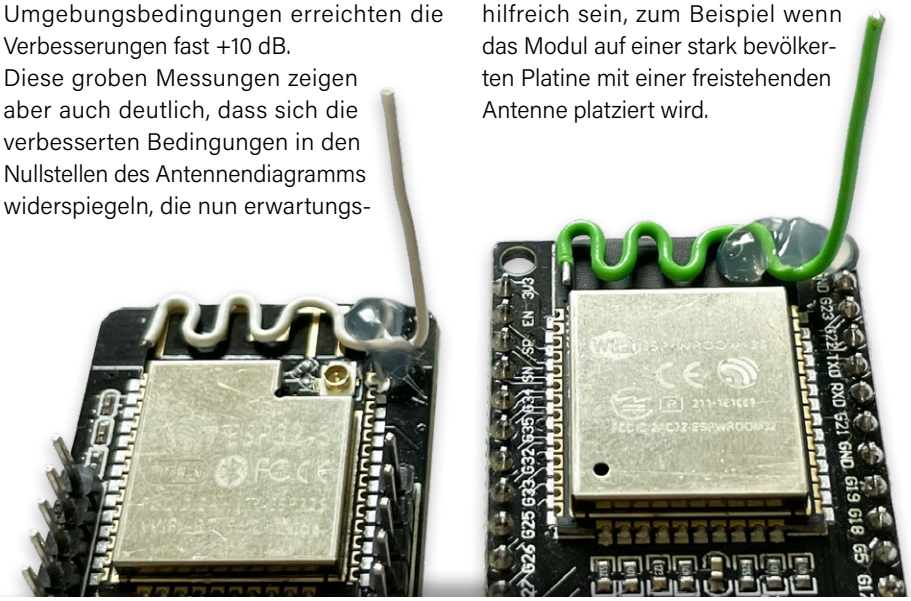


Bild 5. Der gerade Teil der modifizierten Antenne, um 90° nach oben geneigt.

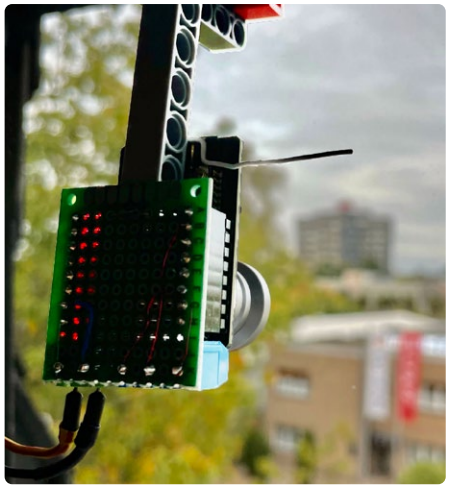


Bild 6. ESP32-CAM mit modifizierter Antenne.

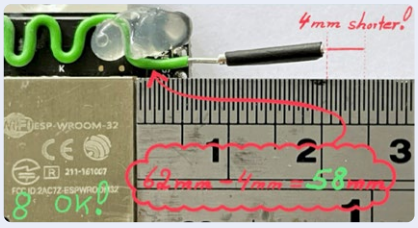
Unter ungünstigen WLAN-Bedingungen mag diese einfache Antennenmodifikation keine Wunder bewirken, aber sie kann die Reichweite des ESP32-Moduls geringfügig erhöhen oder instabile Verbindungen verbessern. Der Preis für diese Verbesserung ist allerdings eine stärkere Richtwirkung. Es funktioniert also am besten, wenn das Modul einen festen Standort hat und eine feste Ausrichtung der Antenne in Bezug auf den Access Point gegeben ist. Das bedeutet, dass die „breite" Seite der Antenne zum Access Point zeigt und die Antenne(n) des Access Points gleich polarisiert sind, das heißt, dass die beiden Antennen möglichst parallel zueinander ausgerichtet sind.

Unter **Optimierung der Antennenlänge** wird eine Methode zur Anpassung der Antennenlänge für optimale HF-Ergebnisse beschrieben. Eine solchermaßen optimierte Antenne führt zu einer zusätzlichen Verbesserung von +1,5 dB gegenüber der gedruckten Antenne, was unter normalen Bedingungen zu Signalen von mindestens +6,5 dB führt.

Bild 6 zeigt eine praktische Anwendung. Die beschriebene Modifikation führte zu einer deutlichen Signalverbesserung und einer sehr stabilen Videoverbindung mit einem ESP32-CAM-Modul an einem zuvor WLAN-kritischen Standort.

SE – 210604-02

Optimierung der Antennenlänge



Da eine einfache Verdoppelung der λ/4-Wellenlänge der invertierten F-Antenne die verschiedenen physikalischen Einflüsse auf den nur teilweise freistehenden Draht nicht berücksichtigt, musste ich die optimale Länge herausfinden. Nach dem Entfernen der Isolierung auf einer Länge von 15 mm und dem anschließenden Kürzen des Drahtes um 6 mm fügte ich eine variable Verlängerung aus einem 10 mm langen silberbeschichteten Draht (Durchmesser 1 mm) hinzu. Ein Stück Schrumpfschlauch diente als provisorischer Schieberhalter.

Die so entstandene Konstruktion (siehe Bild) ermöglichte eine einfache Anpassung der Drahtlänge meines Versuchsaufbaus, um die höchstmögliche Empfangsfeldstärke zu erzielen. Bei dieser Methode wurde der Einfluss des Dielektrikums, der Induktivität und der Kapazität auf die Drahtlänge berücksichtigt. Abschließend fixierte ich die Verbindung mit etwas Lötzinn.

Die experimentell ermittelte optimale Länge für den gesamten Halbwellen-Antennendraht betrug 58 mm, 4 mm weniger als der zuvor geschätzte Wert, was zu einer zusätzlichen Signalverbesserung von +1,5 dB führte!

Über den Autor

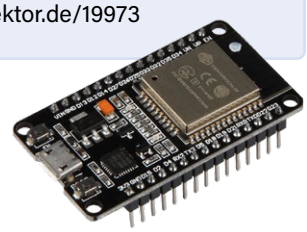
Mehr als 40 Jahre lang durfte sich Peter Neufeld um die Datenverarbeitung, Elektroinstallation, Gebäudeleittechnik sowie Kommunikations- und Medientechnik anderer Leute kümmern. Jetzt findet er wieder Zeit, sich der praktischen Anwendung von klassischer Elektronik und modernen Mikrocontrollern in Hobbyprojekten zu widmen. Einige dieser Projekte hat er auf der Plattform Elektor Labs und in seinem Blog beschrieben.

Sie haben Fragen oder Kommentare?

Haben Sie technische Fragen oder Anmerkungen zu diesem Artikel? Dann wenden Sie sich bitte per E-Mail an den Autor unter peter.neufeld@gmx.de oder an die Elektor-Redaktion unter redaktion@elektor.de.

Passendes Produkt

- **Joy-IT NodeMCU ESP32-Entwicklungsboard**
  www.elektor.de/19973



WEBLINKS

[1] Espressif: Platinenlayout-Prinzipien (Antennenlayout): https://tinyurl.com/esppcblayout
[2] Espressif: ESP32-Spezifikationen (WLAN-Empfang): https://tinyurl.com/esp32wifirec
[3] Texas Instruments: „Kleine 2,4-GHz-Platinenantenne" von Audun Andersen (engl., PDF): https://ti.com/lit/an/swra117d/swra117d.pdf
[4] NXP: „Kompakte Planarantennen für 2,4 GHz" (engl., PDF): https://nxp.com/docs/en/application-note/AN2731.pdf
[5] Invertierte F-Antenne (engl.): https://antenna-theory.com/antennas/aperture/ifa.php
[6] Code zum Herunterladen: https://elektormagazine.de/210604-02
[7] BASIC-Interpreter für ESP32 von ciccioCB: https://flasher.cicciocb.com
[8] Annex-RDS-Forum: https://cicciocb.com/forum/viewforum.php?f=5
[9] Annex32-Dokumentation: https://cicciocb.com/annex32help

Listing 1: ANNEX32-BASIC-Code.

```
'######## WIFI-GRAPH-LOGGER ##################################
'  Allows to compair the two ESP-Modules which have different antennas
'  (peter.neufeld@gmx.de 06/2024)
' Displays  a graphical logging of two WIFI-signals in dB on module_1
' -  the strength of own WIFI-connection to the WIFI access point
' -  the strength of WIFI-connection of a remote ESP32 module_2 to the same AP

' RX = 1 =>  this ESP-Module regularly requests the WIFI-LOG-RESULT-STRING
'            from a remote module with http://REMOTE_IP/msg?x=1
' TX = 1  => this ESP-Module  returns the string with its
'            WIFI-LOG-RESULT-STRING  on request at http://MY_IP/msg?x=1

' uncomment this in code for the ESP32 module_1 as the main module
RX        = 1   'This module sends requests
TX        = 0   'This module is quiet

' uncomment this in the code for the ESP32 module_2 as the 2nd module
'RX         = 0  'This module does not send requests
'TX         = 1  'This module responds to requests

REMOTE_IP$ = "192.168.0.141" 'Set here YOUR IP-Address of module2

X_Num      = 100 'Number of WIFI-MEASURES to display in the graph
WIFI_REMOTE$  = ""
onhtmlreload  WEBPAGE
gosub         WEBPAGE
if TX = 1 onurlmessage RETURN_WIFI_STRING
IF RX = 1 onwgetasync RECEIVE_REMOTE_STRING
timer0 500, LOG_MY_WIFI_CONNECTION
if RX = 1 timer1 1000, GET_REMOTE_WIFI_LOG_STRING
wait
'##############################################################
LOG_MY_WIFI_CONNECTION:
  w=0
  for i  = 1 to 50
  w = wifi.rssi + w
  next i
  w=W/(i-1)   ' Thanks for the hint Fernando !
  WIFI_LOCAL$= trim$(WIFI_LOCAL$ + " " +str$(wifi.rssi,"%2.1f"))
  c = word.count(WIFI_LOCAL$," ")
  p = instr(1, WIFI_LOCAL$, " ")
  P = len(WIFI_LOCAL$) - p
  If c > X_NUM then WIFI_LOCAL$ = right$(WIFI_LOCAL$, p )
  'wlog WIFI_LOCAL$ , c
  jscall |traceme(0,"| + WIFI_LOCAL$ + |");|
 if WIFI_REMOTE$ <> "" jscall |traceme(1,"| + WIFI_REMOTE$ + |");|
return
'##############################################################
RETURN_WIFI_STRING:
URLMSGRETURN WIFI_LOCAL$
return
'##############################################################
```

(weiter auf der nächsten Seite)

```
GET_REMOTE_WIFI_LOG_STRING:
wgetasync ("http://" + REMOTE_IP$+ "/msg?x=1")
return
'#############################################################
RECEIVE_REMOTE_STRING:
WIFI_REMOTE$ = WGETRESULT$
return
'#############################################################
WEBPAGE:
cls
jsexternal "/xy.min.js"
cnt = 0
a$ = ""
a$ = a$ + |<p>WIFI Graph for two ESP32-modules .. <br> |
a$ = a$ + |GREEN = WITH additional antenna <br> RED === with original antenna |
a$ = a$ + |</p><canvas id="canvas1" width="800" height="400"></canvas>|
html a$
pause 500
A$ = ""
A$ = A$ + |var datasets = [|
A$ = A$ + |  {|
A$ = A$ + |    lineColor : 'rgba(20,100,100,1)',|
A$ = A$ + |    pointColor : 'rgba(20,20,20,1)',|
A$ = A$ + |    pointStrokeColor : '#fff',|
A$ = A$ + |    data : []|
A$ = A$ + |  },|
A$ = A$ + |  {|
A$ = A$ + |    lineColor : 'rgba(151,30,0,1)',|
A$ = A$ + |    pointColor : 'rgba(151,80,0,1)',|
A$ = A$ + |    pointStrokeColor : '#fff',|
A$ = A$ + |    data : []|
A$ = A$ + |  }|
A$ = A$ + |];|
A$ = A$ + |var ctx2 = document.getElementById('canvas1').getContext('2d');|
A$ = A$ + ||
A$ = A$ + |var xy = new Xy(ctx2, );|
A$ = A$ + ||
A$ = A$ + |function traceme(set, data){|
A$ = A$ + |  var s = data.split(" ");|
A$ = A$ + |  for (var i=0; i<s.length; i++) {|
A$ = A$ + |    datasets[set].data[i] = [i, s[i]];|
A$ = A$ + |  }|
A$ = A$ + |  xy.draw(datasets);|
A$ = A$ + |}|
jscript a$
A$ = "" ' clean memory
return
```

Messe Frankfurt Group

Einzigartig. Praxisnah. Innovativ.

Das ist die SPS – Smart Production Solutions.
Eine Fachmesse, die sich durch Erfolgsgeschichten, geballte Expertise und wegweisende Lösungen auszeichnet. Als Highlight für die Automatisierung bietet sie auch dieses Jahr wieder eine einzigartige Plattform für alle, die ihr Unternehmen mit smarter und digitaler Automation voranbringen wollen.

Tauchen Sie ein in eine Welt voller Innovationskraft!
Infos und Tickets: sps-messe.de

mesago

sps

12. – 14.11.2024
NÜRNBERG

Bringing Automation to Life

33. Internationale Fachmesse der industriellen Automation

Sie designen. Wir liefern.

Die neuesten Produkte für Ihre neuesten Designs.™

mouser.de/new

MOUSER ELECTRONICS