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

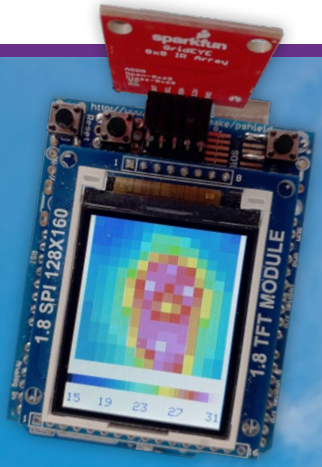
elektor MAG
design > share > earn

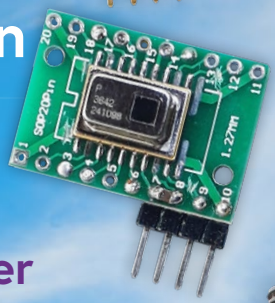
602
* SEIT 1961 *

JULI/AUGUST 2024
ELEKTORMAGAZINE.DE

Realisiert mit einem Arduino UNO

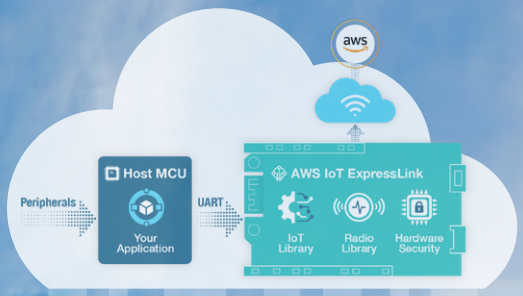
KLEINE WÄRME-BILD-KAMERA





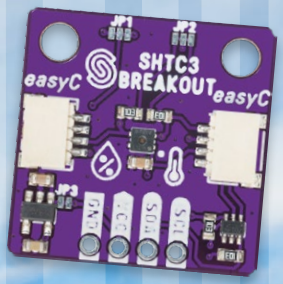
IM FOkUS

IoT und Sensoren



AWS für Arduino und Co.

Teil 1: AWS-IoT-ExpressLink in der Praxis

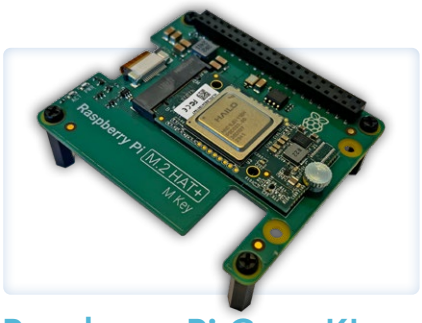


Universeller Garten-Logger

Ein Schritt auf dem Weg zur KI-Gartenarbeit

Sensoren für Wetterstationen

Welchen Sensor sollte man wählen?

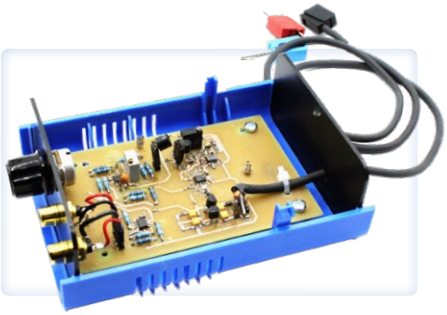


Raspberry Pi Goes KI

KI-Beschleuniger mit 13 TOPS

ESP32-Energiemessgerät

Integration mit Home Assistant



Analoger 1-kHz-Generator

Sinuswellen mit geringen Verzerrungen

40008
4 198630 314904

Treten Sie jetzt der Elektor Community bei!

Jetzt elektor GOLD membership Mitglied werden!

- Zugang zum kompletten Online-Archiv (1970-heute)!
- 8x Elektor Magazin (gedruckt)
- 8x Elektor Magazin (PDF)
- 10% Rabatt im Elektor Store und exklusive Angebote
- Zugriff auf über 5.000 Gerber-Dateien

u.v.m. aus der Projektplattform Elektor Labs

Auch erhältlich

Die digitale elektor GREEN membership Mitgliedschaft!

- Zugang zum kompletten Online-Archiv
- 8x Elektor Magazin (PDF)
- 10% Rabatt im Elektor Store und exklusive Angebote
- Zugriff auf über 5.000 Gerber-Dateien

u.v.m. aus der Projektplattform Elektor Labs

www.elektormagazine.de/Abonnement

elektor design › share › earn

IMPRESSUM

55. Jahrgang, Nr. 602
Juli/August 2024
ISSN 0932-5468

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

Für alle Ihre Fragen
service@elektor.de

Mitglied werden
www.elektormagazine.de/abo

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

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

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

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

EDITORIAL

Jens Nickel

Chefredakteur ElektorMag

Ein Sensor für Innovationen

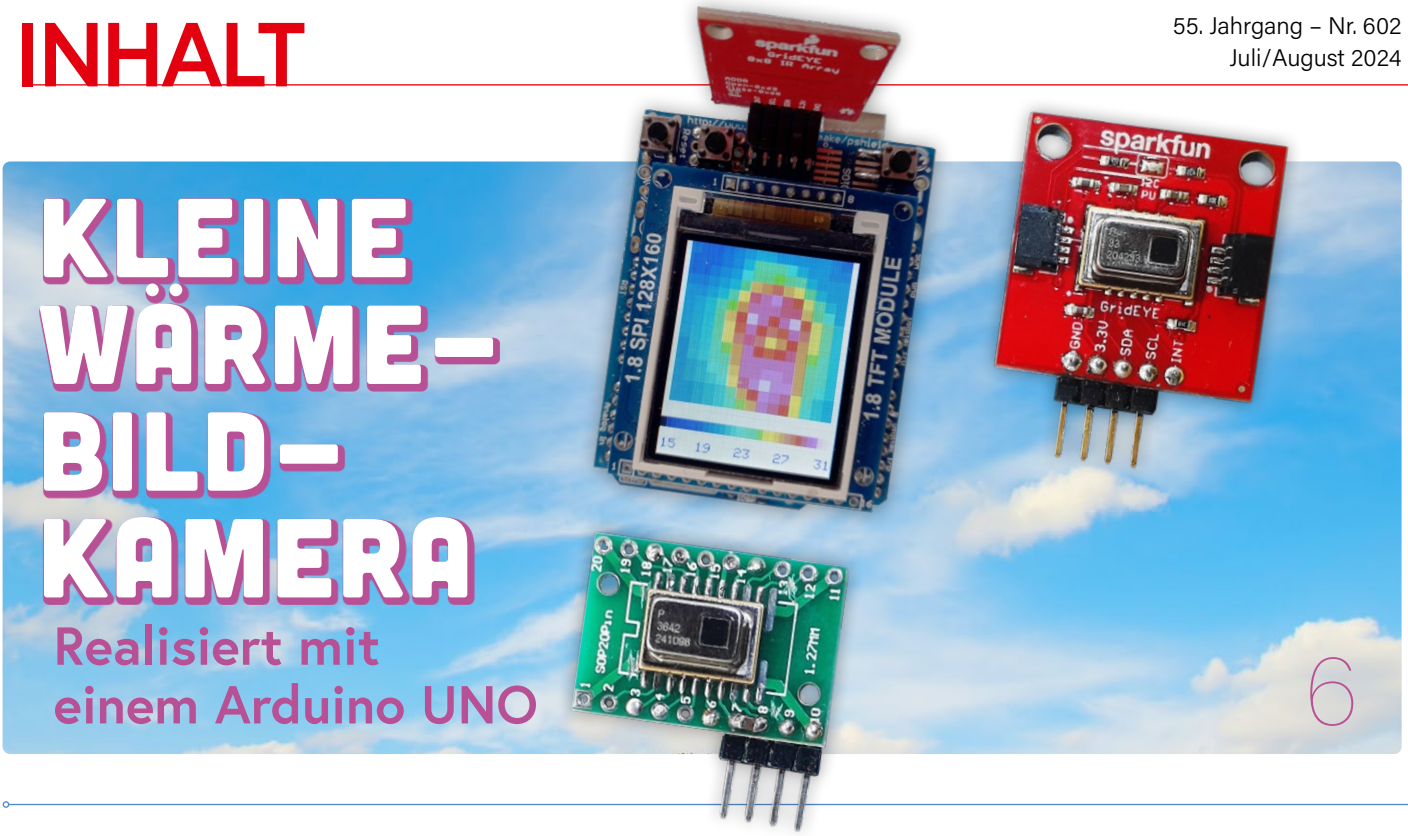
Wenn ich an Elektronikmesse denke, fallen mir zuerst die embedded world und die electronica ein - beide sind für Elektor ein Pflichttermin, und natürlich sind wir dort auch mit einem eigenen Stand vertreten. Schon jetzt kann ich übrigens ankündigen, dass wir zur electronica im November etwas ganz Besonderes in Vorbereitung haben - mehr darüber in den kommenden Ausgaben.
Doch es sind auch die kleineren Messen, die einen Besuch lohnen. Kürzlich bin ich von der Sensor+Test zurückgekehrt, die parallel zur PCIM in Nürnberg stattfand. Mit den vielen kleinen Ständen, den teilweise hochspezialisierten Ausstellern und nicht zuletzt der Unterbringung in den etwas „angestaubten" Messehallen 1 und 2 wird sich wohl manch ein Besucher zuerst ein wenig in der Zeit zurückversetzt fühlen. Millionenteure Stände, mit denen Unternehmen darstellen, wie man sich die auf Künstliche Intelligenz basierte Marktmacht in 10 Jahren vorstellt, sucht man hier vergeblich. Doch gerade die kleineren Firmen sind Antreiber von Innovation. Auf der Sensor+Test habe ich eine Menge interessanter Dinge gesehen, zum Beispiel ein Handheld-Gerät, das eine Art Kamerabild von Geräuschquellen erstellen kann. Kombiniert mit einem Bild einer optischen Kamera und einem Thermographiebild, ergibt sich ein neuartiges Tool zur Maschinenwartung.
Neben der Sensor+Test (passend zum Schwerpunktthema dieser Ausgabe) habe ich auch die Leistungselektronikmesse PCIM besucht. Einen kleinen Bericht finden Sie in der digitalen Bonusausgabe, die Sie kostenlos von der Elektor-Webseite herunterladen können (*www.elektormagazine.de/2407-bonus*).
Es freut mich, dass wir Ihnen solche digitale Bonus-Ausgaben ab jetzt regelmäßig anbieten können. Die kostenlosen PDF-Downloads bieten zusätzliche Projekte und Hintergrundartikel, die unsere regulären Ausgaben ergänzen. Vielleicht kennen Sie dieses Format schon von unserem letzten Circuit Special oder der gemeinsamen Gastausgabe mit Espressif - die letztgenannte Bonus-Ausgabe wurde 114.000 Mal heruntergeladen!
Die Bonus-Ausgaben finden Sie auf den speziellen Themenseiten, die wir das ganze Jahr über aktualisieren. Wenn Sie ein besonderes Interesse an IoT und Sensoren haben, besuchen Sie unsere Themenseite *IoT und Sensoren*, auf der Sie das ganze Jahr über aktuelle Nachrichten und exklusive Artikel finden (*www.elektormagazine.de/iot-sensoren*).
In diesem Heft nimmt mein Kollege Saad Imtiaz sein ESP32-basiertes Energiemessgerät zum ersten Mal in Betrieb, das beeindruckende Home-Assistant-Framework erspart ihm dabei viel Programmierarbeit (Seite 12). IoT-Fans sollten auf jeden Fall auch einen Blick auf Seite 62 werfen. Schritt für Schritt zeigt unser Autor Tam Hanna, wie man kleine Controller-Boards mit den mächtigen Amazon Web Services verbinden kann.

Elektor-Labs: Ideen und Projekte

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

INHALT

55. Jahrgang – Nr. 602
Juli/August 2024



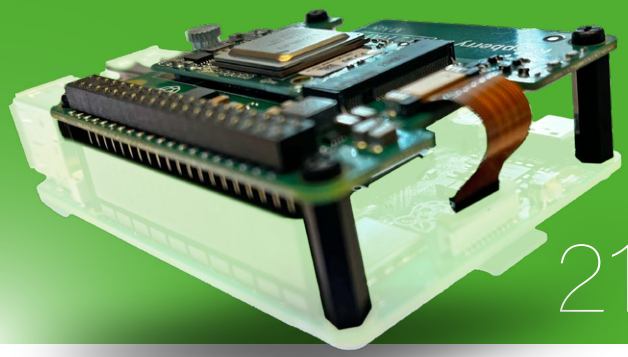
Rubriken

Hintergrund

Projekte

Raspberry Pi Goes KI

Neues Kit enthält M.2-HAT mit KI-Beschleuniger



21

Sensoren für Wetterstationen

Welchen Sensor sollte ich wählen?

24

Tauchen Sie ein in IoT und Sensoren

Besuchen Sie die Elektor-Webseite *IoT und Sensoren* für Projekte, Videos und Tutorials!

www.elektormagazine.de/iot-sensoren

Industry

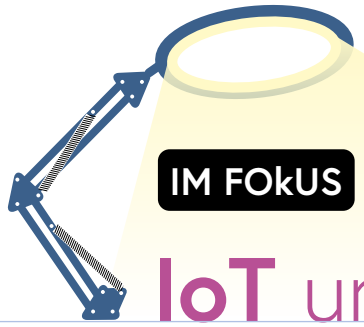
Vorschau

Elektor Circuit Special 2024 (August/September 2024)
Ganz in der Tradition der „Halbleiterhefte" ist die nächste Ausgabe besonders dick und prall gefüllt mit Dutzenden von DIY-Projekten, Retro-Schaltungen, Tipps und Tricks und vielem mehr!
Elektor Circuit Special 2024 erscheint am 14. August 2024.

Elektor September/Oktober 2024
Das Septemberheft ist wie immer randvoll gefüllt mit Schaltungsprojekten, Grundlagen sowie Tipps und Tricks für Elektroniker. Schwerpunkt wird das Thema **Drahtlose Applikationen** sein.

- ESP32-Reichweitenverlängerer
- LoRa-Station mit dem kompakten WIO-E5-Modul
- Erweiterungsplatine für ESP32S3 XIAO
- AWS-Cloud für Arduino und Co: Daten verschicken
- HF-Tastkopf mit LED-Balkenanzeige
- Konfigurierbares Notch-Filter

Elektor September/Oktober 2024
erscheint am 11. September 2024.
Änderungen vorbehalten!



IM FOkUS

IoT und Sensoren

Quelle: Shutterstock

Kleine Wärmebildkamera

Realisiert mit einem Arduino UNO

Von Roland Stiglmayr (Deutschland)

Wer schon mal Hot-Spots auf einer Elektronik-Baugruppe aufspüren, die Temperatur an einem Leistungsbauteil messen oder Menschen in einem Raum detektieren wollte, der hat sich sicher eine Wärmebildkamera gewünscht. In diesem Artikel zeigen wir eine Low-Cost-DIY-Lösung auf Basis eines 8x8-Pixel-Sensors von Panasonic.

Ein ambitionierter Elektroniker wird sich bestimmt fragen, ob und wie man eine Wärmebildkamera selbst bauen kann. Die Antwort lautet, ja, man kann, wenn man die geeigneten Bauteile einsetzt. Der am besten geeignete Wärmesensor ist der AMG88xx „Grid-EYE" von Panasonic. Aufgrund seiner internen Signalverarbeitung, die die Berechnung der Temperaturen der 64 Messstellen vollständig übernimmt, entlastet er den externen Rechner ganz enorm. Deshalb reicht ein UNO mit einem 1,8"-TFT-Screen wie in **Bild 1** völlig aus, um eine einfache Wärmebildkamera zu bauen.

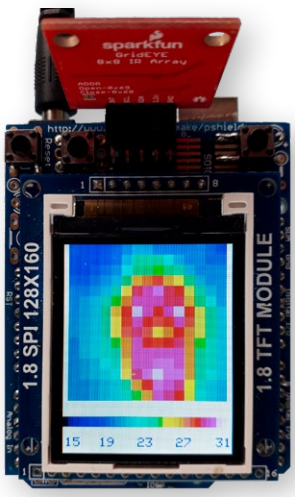


Bild 1. Dahinter steckt immer ein kluger Kopf…

Der „Infrared Array Sensor"

Der Sensor des AMG88xx [1] ist ein auf einem Keramikträger montierter, in MEMS-Technik hergestellter Chip mit 64 infrarot-empfindlichen Messpunkten (Pixeln). Die Pixel sind in einer quadratischen Matrix angeordnet. Daneben sitzt ein ASIC mit der bereits erwähnten Auswerte-Elektronik und ein Thermistor zur Erfassung der Referenztemperatur. Der Keramikträger ist mit einer Metallkappe abgedeckt, in die eine Linse eingelassen ist. Um die Strahlung im Wellenlängenbereich von 5…12 µm nicht zu dämpfen, ist die Linse aus Silizium hergestellt.

Die Linse bildet die Messobjekte auf dem Sensorfeld ab. Man kann sich das so vorstellen, als ob auf jedem Pixel die Spitze einer quadratischen Pyramide stände und er so nur die innerhalb dieser Pyramide geführte Strahlung empfängt. In Summe werden also 64 quadratisch angeordnete Objektflächen erfasst, die sich aus der Aneinanderreihung der Pyramiden ergeben.

Bild 2 zeigt eine vereinfachte und idealisierte Darstellung einer Anordnung mit 4 × 4 Messpunkten. Das Sichtfeld (englisch: Field of View, FOV) ist der für das Array sichtbare Bereich, der ausschließlich über den vertikalen und den horizontalen Öffnungswinkel bestimmt ist. Da das Array quadratisch ist, sind die Winkel gleich und werden mit α bezeichnet. Für jeden Objektabstand a ergibt sich innerhalb des Sichtfelds eine sichtbare Fläche, mit der Kantenlänge *S*. Die Fläche wiederum besteht aus 16 Einzelflächen mit der Kantenlänge s_{pix}, von der jede einem bestimmten Pixel zugeordnet ist. Die Objekttemperatur wird nur dann exakt gemessen, wenn das Objekt wenigstens eine Teilfläche vollständig abdeckt. Mit dieser Forderung lässt sich die kleinste erfassbare Objektgröße für einen bestimmten Abstand a berechnen. Im Falle einer quadratische Matrix mit n×n Pixeln gilt:

$$s_{pix} = 1/n\ (2a \times \tan(\alpha/2))$$
$$A_{pix} = s_{pix}^2$$

Im **Bild 2** erfüllt nur der Anteil des Objekts mit den Koordinaten (2,2) diesen Anspruch. Die Flächen des Sichtfeldes mit den Koordinaten (2,1), (1,2), (2,3) sind nur zur Hälfte abgedeckt. Die gemessene Temperaturänderung der zugehörigen Pixel beträgt deshalb nur die Hälfte gegenüber der der Fläche (2,2).

Einfach gesagt bedeutet das, dass ein Objekt

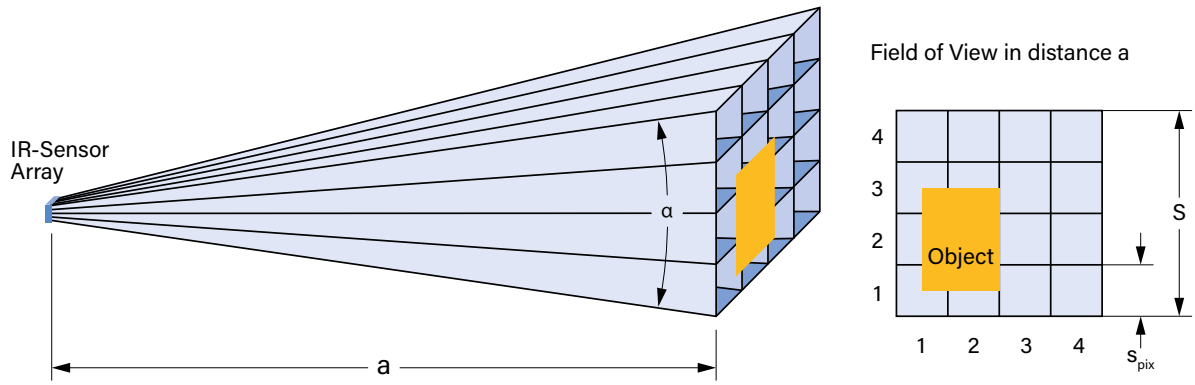


Bild 2. Der sichtbare Bereich eines quadratischen 4×4-Sensors mit Sichtfeld α im Abstand a.

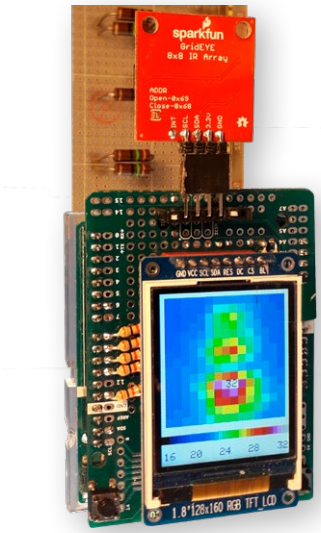


Bild 3. Messobjekt in 50 mm Abstand, α= 60°.

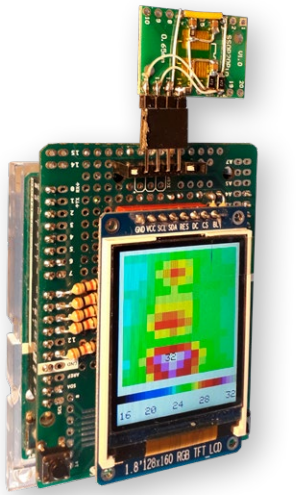


Bild 4. Messobjekt in 100 mm Abstand, α= 30°.

mit zunehmendem Abstand immer größer werden muss, damit die auf einen Pixel projizierte Fläche voll erfasst wird. Dieser Zusammenhang ist immer zu berücksichtigen, um Fehlmessungen und -interpretationen zu vermeiden. **Bild 3** und **Bild 4** sind Beispiele, die den Zusammenhang des Sichtfelds und des Abstands zum Messobjekt zeigen.

Die Genauigkeit der Messung ist auch vom Emmissionsgrad ε des Objekts abhängig. Er resultiert aus der Beschaffenheit der Oberfläche und bestimmt die Fähigkeit, Wärmestrahlung auszusenden. Der Grid-EYE-Sensor nimmt bei seiner internen Berechnung der Temperatur einen festen Emmissionsgrad von 0,93 an.

Thermopile

Jeder einzelne Messpunkt des Sensorfeldes ist eine Thermosäule, ein sogenanntes *Thermopile* [2], das aus vielen elektrisch in Serie und thermisch parallel geschalteten Thermoelementen besteht. Der Keramikträger bildet die Bezugstemperatur, die bestrahlte Oberfläche die zu messende Temperatur des Thermopiles. Thermoelemente generieren eine Spannung, die proportional zur Temperaturdifferenz ist. Um die absolute Temperatur zu ermitteln, muss die Bezugstemperatur sehr exakt gemessen werden und zur Temperaturdifferenz addiert werden. Die Kennlinie eines Thermoelements ist nicht linear, sondern ist eine Funktion der Bezugstemperatur und der gemessenen Spannung. Über im ROM des AMGs gespeicherte Kurven werden die Messwerte linearisiert. Exemplarstreuungen der einzelnen Thermopiles werden durch individuelle, intern gespeicherte Kalibrierdaten berücksichtigt. Man kann sich gut vorstellen, wie aufwendig beziehungsweise rechenintensiv die Linearisierung der Kennlinie wäre, wenn nicht glücklicherweise das interne ASIC diese Aufgabe übernehmen würde. Das ASIC stellt über seine I²C-Schnittstelle die linearisierten und perfekt aufgearbeiteten Messwerte in Grad Celsius zur Verfügung. Je nach Anwendung sind verschiedene Sensor-Typen erhältlich. **Tabelle 1** gibt einen Überblick.

Zum Aufbau der Kamera

Die gesamte Schaltung der Thermobildkamera besteht aus einem Arduino UNO, einem Shield mit einem kleinen TFT-Farbdisplay, zwei

Tabelle 1. Typische Werte der „Infrared Array Sensors" der Baureihe AMG88xx.

Modul	V_{DD}	Messbereich	Erfassungswinkel gesamt	Erfassungswinkel eines Pixels	Objektgröße bezogen auf ein Pixel in 0,3 m Abstand
AMG8832	3,3 V	-20°C ... 100°C	60°	7,5°	4 cm
AMG8833	3,3 V	0°C ... 80°C	60°	7,5°	4 cm
AMG8834	3,3 V	-20°C ... 100°C	60°	7,5°	4 cm
AMG8853	5 V	0°C ... 80°C	60°	7,5°	4 cm
AMG8854	5 V	-20°C ... 100°C	60°	7,5°	4 cm
AMG883642	3,3 V	-20°C ... 100°C	32° ... 34°	4°	2 cm
AMG883543	3,3 V	0°C ... 80°C	90°	11,25°	6 cm

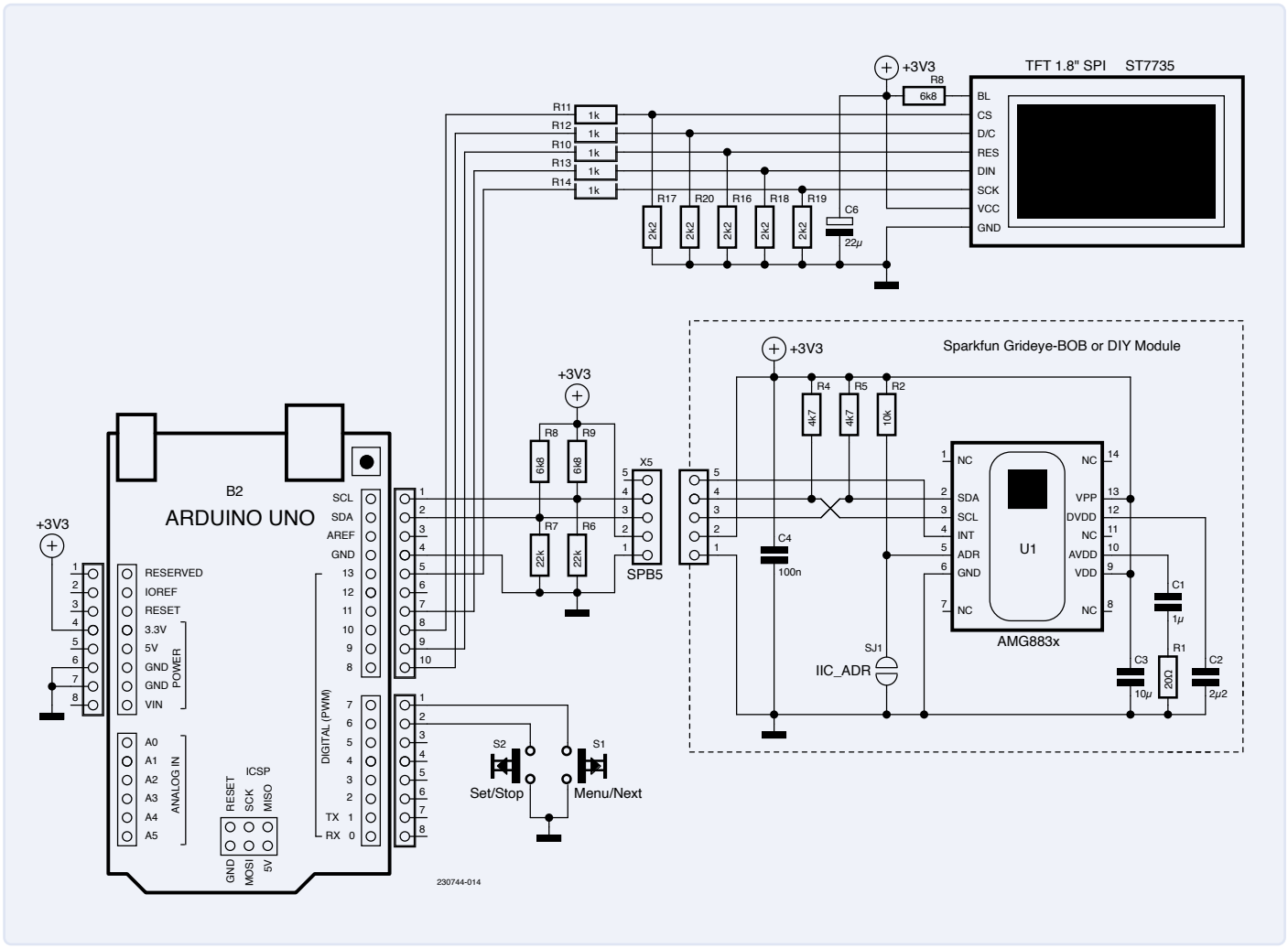


Bild 5. Schaltbild der Kamera.

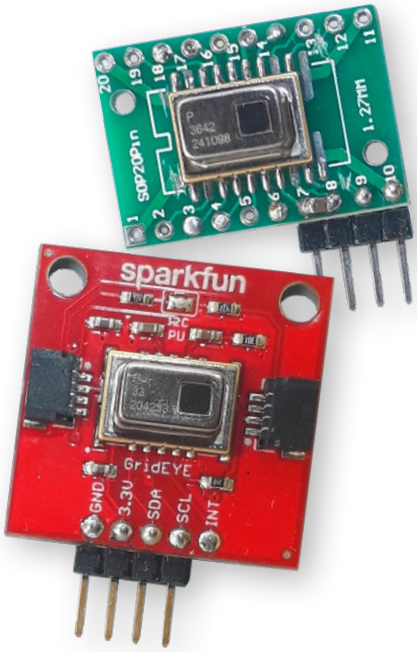


Bild 6. DIY-Sensormodul neben dem Sparkfun-BoB.

Tastern sowie dem Kameramodul. **Bild 5** zeigt das vollständige Schaltbild der Kamera.

Kamera-Modul

Wer sich dafür entscheidet, ein fertig bestücktes Breakout-Board [4] (**Bild 6**) mit einem AMG8833 zu verwenden, ist mit dem Thema auch schon durch. Man muss dann nur überlegen, ob die Verkabelung mit Qwiic- oder Pfostenverbindern realisiert wird.

Man kann das Modul aber auch relativ einfach selbst aufbauen, weil sich der Sensor problemlos mit einem feinen Lötkolben löten lässt. Der Vorteil der selbstgebauten Version liegt darin, dass man sich den geeigneten Sensor aussuchen kann. So ist zum Beispiel der AMG883642 [3] aufgrund des kleinen Sichtfeldes und dem weiten Temperaturbereich für viele Anwendungen sehr interessant.

Der Autor hat den Sensor auf einem SSOP20-zu-DIL-Adaptermodul (1,27 mm auf 2,54 mm) aufgelötet und gleich die wenigen Bauteile in SMD mit auf das Modul gepackt (**Bild 7**). Die I²C-Device-Adresse ist belanglos, da die Software beim Start die richtige Adresse ermittelt, so dass der Pin ADDR auch auf GND gelegt werden darf.

TFT-Shield

Da die Schaltung in Bild 5 so einfach ist, kann sie leicht auf einem Arduino-Mega-Proto-Shield aufgebaut werden. Dort ist genügend Platz, um bedrahtete Bauteile zu

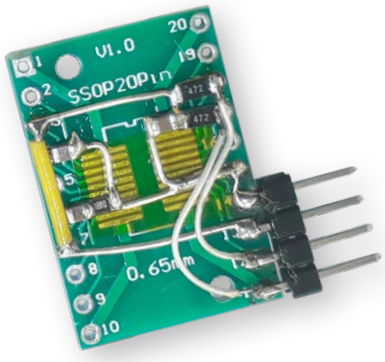


Bild 7. DIY-Sensormodul, Rückseite.

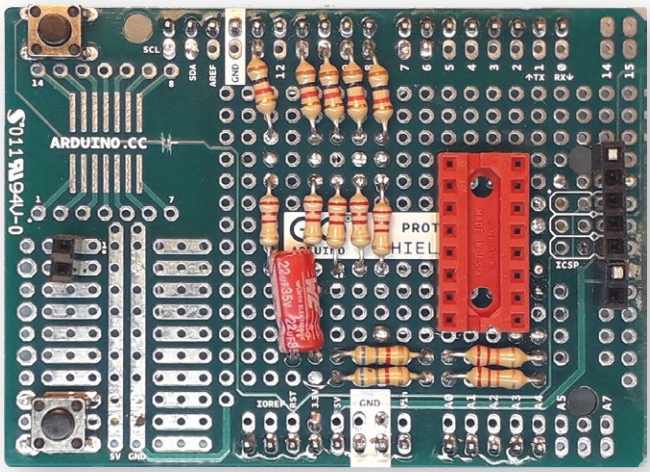


Bild 8. Eigenbau-Shield, Bestückungsseite.

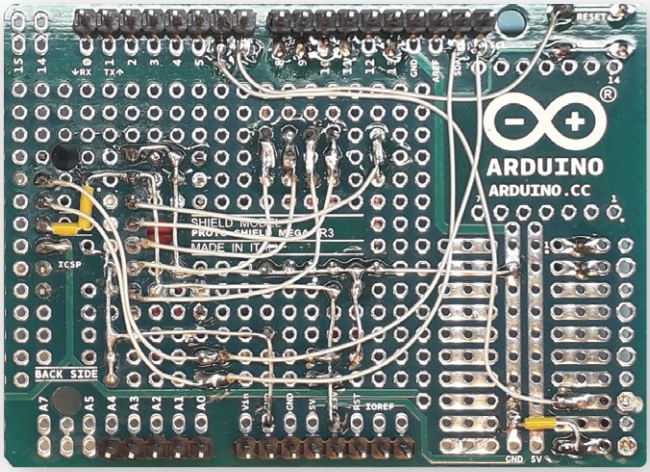


Bild 9. Eigenbau-Shield, Verdrahtungsseite.

verwenden (**Bild 8**, **Bild 9**). Das TFT-Display mit seiner Diagonalen von 1,8 Zoll und 128×160 Bildpunkten muss einen ST7735 Controller besitzen und sollte mit 3,3 V zu betreiben sein. Ein 5-Euro-Teil aus chinesischer Fertigung ist völlig ausreichend. Aber Vorsicht! Diese Boards weisen unterschiedliche Anschlussbelegungen auf. Das stellt zwar kein Problem dar, aber man muss darauf achten. Als Stecker für das Display lässt sich sehr gut eine DIL-Fassung einsetzen.

Versorgung und Busse

Das Kameramodul wird mit der 3,3-V-Versorgung des UNOs betrieben. Deshalb werden die Pullup-Widerstände der I²C-Leitungen auf 3,3 V gelegt. Zusätzlich zu den Widerständen auf dem Kameramodul sind auf dem Shield des Autors noch zwei Pullups auf 3,3 V vorhanden sowie zwei Pulldowns, um den internen Strom von 140 µA aus den Ports zu kompensieren. Obwohl der UNO mit 5 V läuft, liest er die 3,3-V-Signale absolut fehlerfrei.

Das TFT-Display wird über die SPI-Schnittstelle angesteuert. Der eingesetzte Controller ST7735 akzeptiert nur 3,3-V-Signale an seiner Schnittstelle. Die 5-V-Portsignale des UNOs werden deshalb mit den Spannungsteilern R10 und R14 sowie R16 und R20 auf 3,3 V reduziert. Da der SPI-Takt 10 MHz beträgt, dürfen die Widerstände nicht allzu hochohmig gewählt werden, sonst wären die parasitären Einflüsse zu störend. Die angegebenen Werte bilden einen guten Kompromiss aus Stromaufnahme und verzerrungsfreier Signalübertragung. Die Platzierung der beiden Taster ist im **Bild 10** zu sehen.

Inbetriebnahme

Als Erstes installiert man mit dem Bibliotheksmanager der Arduino IDE, die beiden Bibliotheken *Adafruit_ST7735_and_ST7789_Library* und *Adafruit_GFX_Library*. Dann kompiliert man den Sketch *Grideye_V2x.ino* von der Elektor-Projektseite [5] und lädt das Programm auf den UNO.

Bei ausgeschalteter Stromversorgung steckt man das Shield, noch ohne Kameramodul, auf und versorgt den UNO. Das TFT muss jetzt hell werden und die Fehlermeldung *„no valid device found"* anzeigen. Jetzt überprüft man die Polarität der Versorgungsspannung am Stecker zum Kameramodul. Mit einem Oszilloskop lassen sich die I²C-Signale messen. Dann schaltet man wieder aus, verbindet das Kameramodul mit dem Shield und nimmt die Baugruppe wieder in Betrieb.

Als Ergebnis erhält man (hoffentlich) ein sinnvolles Bild. Je nach Version des ST7735-Controllers kann die Position und die Farbdarstellung falsch sein. Dies lässt sich im Programm berichtigen. Hierzu bietet der Sketch vier unterschiedliche Initialisierungs-Routinen an. Dafür sind vier Werte `TFT_TYPE_n` definiert, von denen einer der Konstanten `TFT_TYPE` zugewiesen werden muss. Beim richtigen Wert ist das Bild oben, links und rechts eingepasst. Wenn dann noch die Farben rot und blau vertauscht sind, muss der Wert `TFT_CHANGE_COLR` auf `true` gesetzt werden. Sollte das Bild auf dem Kopf stehen, weist man `TFT_ORIENTATION` den Wert `02` zu. Als Letztes wird der Temperaturbereich des verwendeten Sensors durch Setzen des Wertes `AMG88x3` auf `true` oder `false` ausgewählt.

```
#define AMG88x3 false
//select sensor type, true if AMG88x3
#define PERMANENT_AUTO false
//if true autoranging always active
#define T_OFFSET 0
//offset for correcting T [°C]

#define TFT_ORIENTATION 00
//portrait format
//#define TFT_ORIENTATION 02
//portrait format overhead

#define TFT_TYPE_1 00
//module type 1 (w/o SD)
#define TFT_TYPE_2 01
//module type 2
#define TFT_TYPE_3 02
//module type 3 (with SD)
#define TFT_TYPE_4 03
//module type 4
const byte TFT_TYPE= TFT_TYPE_1;
//set used module type

#define TFT_CHANGE_COLR true
//change red-blue depending on tft
```

Software

Der Sketch wurde in der Arduino-Programmierumgebung erstellt. Für den Grid-EYE-Sensor wird keine externe Bibliothek genutzt,

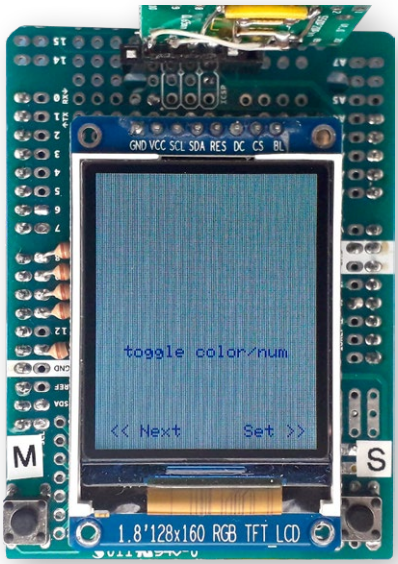


Bild 10. Positionierung der Taster, Umschalten zwischen grafischer und numerischer Anzeige.

Listing 1. Die Funktion read_GridEYE.

```
if (acquire==true)
     (read_GridEYE (&lowestT, &highestT,              //read all pixels and
                    &highestTpix));                   //get lowest and highest T

...

  void read_GridEYE (int *pl, int *ph, int *phnum)
     {
       *ph= -512; *pl= 1024;                          //init values highest, lowest
       int T;                                         //temporary
       byte pixnum;

        //  read data of 64 pixels
        //  ----------------------

       for(byte y = 0; y <8; y++)                     //line address
         {
           for(byte x=0; x<8; x++)                    //column address
             {
               pixnum= (y+56)-(x*8);                  //due to orientation, start with pixel 56
               T= GridEye_getT (pixnum);              //read this pixel
               pixel_Table[y*2][x*2]= T;              //T to each second column, line position

                 if (T>*ph)                           //find highest temperature
                   { *ph=T;
                     *phnum= y*256 + x;               //save position of this pixel
                   }
                 if (T<*pl) *pl=T;                    //find lowest temperature
             }
         }

        //  interpolate between the horizontal pixels of each line
       //  ------------------------------------------------------

       for (byte y=0; y<8; y++)                       //line address counter
         {
           for (byte x=0; x<7; x++)                   //column address counter
             {
               T= (pixel_Table [2*y][2*x] +  pixel_Table [2*y][(2*x)+2])/2;
               pixel_Table [2*y][(2*x)+1]= T;
             }
         }

      //  interpolate between the vertical pixels of each column
      //  ------------------------------------------------------

       for (byte x=0; x<15; x++)                      //column address counter
         {
           for (byte y=0; y<7; y++)                   //line address counter
             {
               T= (pixel_Table [2*y][x] + pixel_Table [(2*y)+2][x])/2;
               pixel_Table [(2*y)+1][x]= T;
             }
         }
     }
```

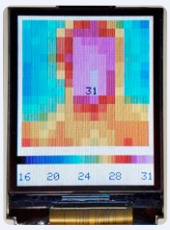
alle erforderlichen Funktionen sind direkt im Sketch realisiert. Damit ist es sehr einfach, Änderungen oder Erweiterungen nach eigenen Vorstellungen umzusetzen.
Das Hauptprogramm ruft die Messroutine `read_GridEYE` (**Listing 1**) zyklisch auf, das heißt, der Ablauf ist deterministisch. Die Messroutine liest über die Funktion `GridEye_getT` die 64 Temperaturen (während des Einlesens werden auch die niedrigste und die höchste Temperatur ermittelt und zusammen mit der Position der höchsten Temperatur abgespeichert).
Die Temperaturen haben eine Auflösung von 0,25 K. Um keine gebrochenen Zahlen zu verwenden, werden sie vom Sensor mit dem Faktor 4 multipliziert. Der Sensor liefert die Werte dann mit 12 bit, wobei das 12. Bit das Vorzeichen ist. Die Leseroutine macht daraus einen int-Typ, also 16 bit im Zweierkomplement.
Die Werte des Sensors werden in das zweidimensionale Array `pixel_Table` geschrieben. Dieses Array speichert 15 × 15 int-Werte und kann deshalb auch die interpolierten Werte aufnehmen. Nach dem Einlesen der Sensorwerte erzeugt die Routine jeweils zwischen zwei benachbarten Zeilenwerten einen neuen Wert durch Interpolation. Diese Interpolation wird dann auch spaltenweise inklusive der bereits interpolierten Werte durchgeführt. Die sich ergebenden 15 × 15 = 225 Werte stehen nun im Array `pixel_Table` bereit.

Die Funktion `showT_as_Color` zeigt die 225 Werte in Falschfarben an. Die Zuordnung der Farben zu den Werten wird über die LUT `color_Scale` vorgenommen. Zur guten Unterscheidbarkeit beschränkt sich die Auflösung auf 32 Farben.
Deshalb müssen die Messwerte so skaliert werden, dass sie in einen Bereich von 32 Werten passen. Dies übernimmt die Funktion `do_Auto_Ranging`. Abhängig von der tiefsten gemessenen Temperatur wählt die Funktion aus sechs vorgegebenen Werten den niedrigsten Wert des Messbereichs aus und übergibt ihn an die Variable `Tcoloffset`. Aus der Differenz der höchsten gemessenen Temperatur und `Tcoloffset` wird die Auflösung `Tcolrange` des Messbereichs aus fünf vorgegebenen Werten bestimmt. Die Messwerte werden durch Subtrahieren von `Tcoloffset` und Multiplizieren mit `Tcolrange` in einen Bereich mit einem Werteumfang von 32 transformiert, so dass sie durch 32 Farben dargestellt werden können.
Die numerische Darstellung der Messwerte übernimmt `showT_as_Numeric`, deren serielle Ausgabe die Funktion `Serial_send_T`.
Die Routine `Set_menu` stellt diverse Einstellungen zur Auswahl und übernimmt sie bei Aktivierung. Die Einstellungen werden im EEPROM gespeichert und bei jedem Neustart wieder hergestellt.

Bedienung

Über die Taste *Menu* gelangt man zu den Einstellungen. Durch weitere Betätigung dieser Taste kann man durch die auswählbaren Punkte blättern. Die gerade aktiven Einstellungen werden grün dargestellt. Mit der Taste *Set* lässt sich der angezeigte Eintrag übernehmen. Der erste Parameter ist *Auto Ranging,* mit dem meistens eine optimale Darstellung erzielt wird. Mit *toggle color/num* (siehe Bild 10) lassen sich die Ergebnisse numerisch oder falschfarbig darstellen. Betätigt man während des normalen Messablaufs die Taste *Set*, wird die Messung gestoppt und eingefroren. Die Darstellung lässt sich im Menü bearbeiten, ohne dass die Messwerte verändert werden. Nach einem nochmaligen Klick wird die Messung fortgesetzt.
Bei der Falschfarben-Darstellung wird die Temperatur des wärmsten Punktes in die Grafik als numerischer Wert eingeblendet. Beim Neustart kann dieses Feature unterdrückt werden, indem während des Resets die *Menü*-Taste gedrückt gehalten wird. Über die serielle Schnittstelle werden die Ergebnisse fortlaufend mit einer Auflösung von 0,25 K ausgegeben.
Abschließend kann man nur mit Erstaunen zusehen, was so ein kleiner ATmega-Prozessor zu leisten vermag!

RG — 230744-02



Über den Autor

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

Sie haben Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zu diesem Projekt? Wenden Sie sich bitte an den Autor unter 1134-715@online.de oder die Elektor-Redaktion unter redaktion@elektor.de.

Passendes Produkt

› **SparkFun Grid-EYE Infrared Array Breakout – AMG8833**
www.elektor.de/19605

WEBLINKS

[1] Grid-EYE Dokumentation: https://industrial.panasonic.com/ww/products/pt/grid-eye
[2] Thermopile (Wikipedia): https://de.wikipedia.org/wiki/Thermosäule
[3] Datenblatt AMG883462: https://industrial.panasonic.com/cdbs/www-data/pdf/ADI8000/ast-ind-139049.pdf
[4] Grid-EYE-BoB: https://sparkfun.com/products/14607
[5] Projektseite: https://elektormagazine.de/230744-02

Projekt-Update #3: Energie-messgerät mit ESP32

Integration und Test mit Home Assistant

Von Saad Imtiaz (Elektor)

Im letzten Projekt-Update haben Sie von den Verbesserungen des Schaltplans und der Platine des ESP32-Energiezählers erfahren. Dieses Projekt-Update konzentriert sich auf die praktische Umsetzung und Integration der neuen Version. Es erläutert Schritt für Schritt die Einrichtung des Messgeräts mit ESPHome und Home Assistant für eine effektive Energieüberwachung. Außerdem befassen wir uns mit der Kalibrierung des Geräts.

In unserem letzten Artikel [1] haben wir uns auf Verbesserungen des Schaltplans und der Platine des ESP32-Energiemessgeräts konzentriert und dabei Modularität und Sicherheitsfunktionen verbessert. Bevor wir uns mit dem nächsten Projekt-Update befassen, gönnen wir uns einen kurzen Überblick.

Bei den letzten Weiterentwicklungen des ESP32-Energiemessgeräts haben wir ein Upgrade des Mikrocontrollers auf einen ESP32-S3 vorgenommen, der eine höhere Verarbeitungsleistung und einen größeren Funktionsumfang bietet. Mit dem neuen Design wurde die Platine verschlankt und ein Stromversorgungssystem mit Transformatoren eingeführt, das die „Hochspannung" von 230 V von der Elektronik fernhält und sowohl für die Spannungsabtastung als auch für den Betrieb des Systems harmlose 12 V verwendet. Dadurch wurde die Sicherheit erheblich verbessert und gleichzeitig die Flexibilität für ein- und dreiphasige Installationen beibehalten.

Bild 1. Das zusammengebaute ESP32-Energiemessgerät mit seinem OLED-Display und den Statusanzeigen.

Weitere Verbesserungen betrafen die Verwendung eines effizienteren Abwärtswandlers des Typs AP63203WU-7, die Modularisierung der Leiterplatte, die Kalibrierung der Stromwandler-Abtastschaltung und vieles mehr. Dadurch wurden nicht nur die Leistung und die Funktionalität des Energiezählers optimiert, sondern auch die Kosten verringert und die Größe reduziert.

In diesem dritten Projekt-Update gehen wir auf die Schritte ein, die unternommen wurden, um dieses Energiemessgerät in Betrieb zu nehmen, und auf den Weg, den er vom Labortisch bis zum Zählerkasten genommen hat. Darüber hinaus wird erläutert, wie das Gerät eingerichtet und kalibriert und wie es mit ESP Home in den Home Assistant integriert wird, um die gesammelten Daten des Energiemessers anzuzeigen und zu überwachen. In **Bild 1** ist ein Schnappschuss des ESP32-Energiemessers in einem 3D-gedruckten Gehäuse mit OLED-Display zu sehen. Das Bild zeigt auch die Statusanzeigen, anhand der sich die Stromaufnahme in Echtzeit verfolgen lässt.

Zusammenbau

Die neue Platine wurde so entworfen, dass sie kompakter und einfacher zu löten ist. Das Layout weist nun angemessene Abstände um jedes Bauteil auf, was den Lötprozess erleichtert. Um die Verbreitung des Projekts in der Community und Änderungen durch Enthusiasten und Fachleute gleichermaßen zu erleichtern, stehen die komplette Stückliste (BOM) im Mouser-Format und die Produktionsdateien auf dem GitHub-Repository von Elektor Labs [2] bereit.

Für die Spannungs- und Stromabnahmeanschlüsse wurden Schraubklemmen von *CUI Devices* verwendet. Die Qualität dieser Klemmen ist viel höher als die der billigen blauen Klemmen, die man bei den meisten Sensormodulen sieht. Da wir es mit Wechselspannungen

und Energiemessung zu tun haben, sind sichere und zuverlässige Verbindungen unerlässlich.
Die Rauschunterdrückung ist ein wichtiger Aspekt des Platinenlayouts. Durch die Aufnahme von Elektrolyt- und Keramikkondensatoren um den Energiemess-Chip ATM90E32S herum wird sowohl nieder- als auch hochfrequentes Rauschen herausgefiltert, was eine genauere und stabilere Energiemessung gewährleistet. Die Platine ist in **Bild 2** abgebildet.
Wie bereits erwähnt, haben wir uns für die Verwendung eines 12-V-Transformators für die Spannungsabtastung und die Energieversorgung des gesamten Systems entschieden. Einen solchen Transformator zu finden ist einfach und billig, aber die meisten dieser Transformatoren benötigen viel Platz, wenn sie in einem kundenspezifischen Gehäuse verwendet werden, wie **Bild 3** beweist. Daher sollten Sie am besten DIN-Schienen-Transformatoren (zum Beispiel Klingeltrafos) verwenden, um den Aufbau sauberer und sicherer zu gestalten. Solche Transformatoren lassen sich leicht im Netz finden. Außerdem hängt die Genauigkeit der Spannungsmessungen von den Eigenschaften der Transformatoren ab, einschließlich der Genauigkeit ihres Spannungsverhältnisses, der Phasenverschiebung und der Linearität.
Vielleicht haben Sie auf den Bildern bemerkt, dass nur ein Transformator an das Energiemessgerät angeschlossen ist. Der Energiemesser wurde nämlich hier für einphasigen Betrieb konfiguriert (Lötjumper JP8 auf der Rückseite der Platine angebracht, wie in **Bild 4** zu sehen). Um den Energiezähler im dreiphasigen Modus zu betreiben und die Spannung jeder Phase in einem dreiphasigen System mit drei Transformatoren abzufragen, müssen Sie die Primärseiten der drei Transformatoren an die jeweiligen Außenleiter (L1, L2, L3) gegen den Neutralleiter N anschließen. Auf der Sekundärseite schließen Sie ein Ende der Wicklung jedes Transformators an einen gemeinsamen Nullpunkt auf der Platine an und bilden so eine Sternkonfiguration (Y). Die freien Enden der Sekundärwicklungen (V1, V2, V3) liefern dann die Spannungsausgänge (UA, UB und UC) auf der Platine für jede Phase. Unbedingt muss sichergestellt sein, dass die Transformatoren für Netzspannung und den Strom des Systems ausgelegt sind und dass Primär- und Sekundärstromkreise strikt voneinander isoliert sein müssen, aus Sicherheitsgründen und um einen stabilen und ausgeglichenen Neutralleiter-Anschluss und damit eine hohe Messgenauigkeit zu gewährleisten.

Einrichten mit ESPHome und Home Assistant

Als Teil der Entwicklung wird eine spezielle Firmware erstellt, um die Fähigkeiten des Energiemess-Chips und die fortschrittlichen KI-Funktionen des ESP32-S3 zu nutzen.
Obwohl die Entwicklung einer solchen maßgeschneiderten Firmware viel Zeit in Anspruch nimmt und zur Zeit noch im Gange ist, schränkt dies die Verwendbarkeit des Energiemessgeräts nicht ein. Das Gerät kann schon jetzt mit bestehenden Plattformen wie Home Assistant voll funktionsfähig sein und bietet eine sofortige Lösung für die Energieüberwachung. Daher liegt der Schwerpunkt in diesem Artikel auf der Integration des ESP32-Energiemessers in das Smart Home mit Home Assistant [3] und ESPHome [4]. Dieser Abschnitt führt Sie zunächst durch die Einrichtung des Energiemessgeräts in der Home-Assistant-Umgebung, damit Sie seine vollständige Funktionalität nutzen können.

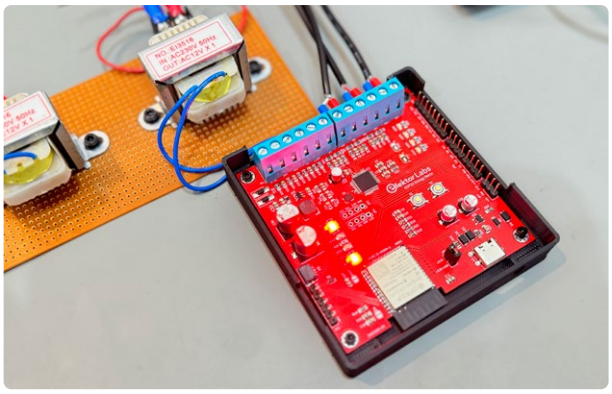


Bild 2. Platine des vollständig montierten ESP32-Energiemessers.

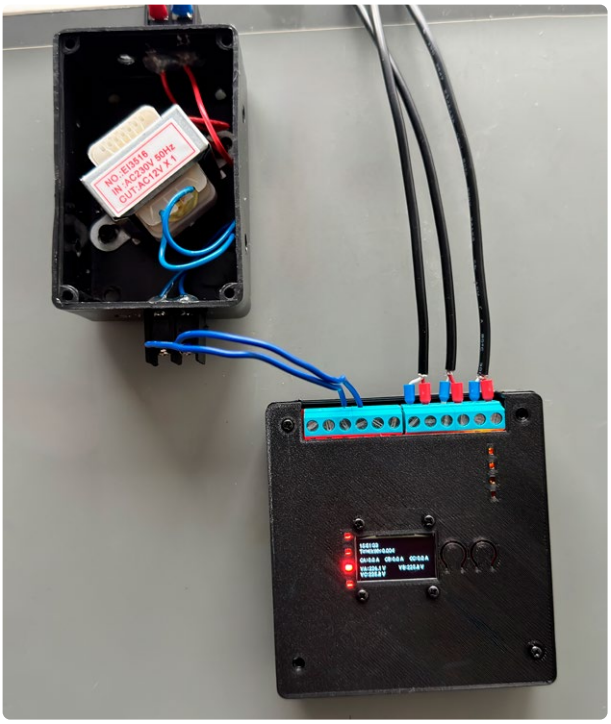


Bild 3. Der Trafo in einem kundenspezifischen Gehäuse verringert 230 V auf 12 V.

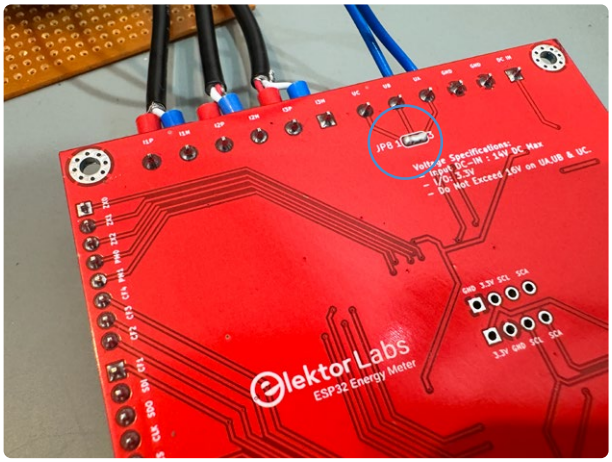


Bild 4. Jumper-Konfiguration auf der Platine für den Betrieb des Energiemessgeräts im Einphasenbetrieb.

Um das ESP32-Energiemessgerät mit der ESPHome-Firmware einzurichten und es in Home Assistant zu integrieren, müssen Sie natürlich zuerst Home Assistant installieren. Eine ausführliche Anleitung dazu finden Sie in einem Artikel meines Elektor-Kollegen Clemens Valens [5]. Ist die Installation gelungen, fügen Sie die ESPHome-Integration aus dem *Add-on Store* hinzu und erstellen ein neues Projekt in ESPHome für Ihr ESP32-Gerät. Dadurch wird automatisch eine grundlegende YAML-Konfigurationsdatei erzeugt, die ein eigenes ESPHome-Projekt mit allen verwendeten Sensoren und vielen anderen Optionen spezifiziert. Sie müssen diese Standarddatei herunterladen, bevor Sie die nächsten Schritte ausführen.

Verbinden Sie Ihren ESP32 mit Ihrem Computer und wählen Sie den richtigen COM-Port für den ESP32-S3 aus. Klicken Sie im ESPHome-Dashboard auf *Install* und wählen Sie die *.bin*-Datei aus, um die Firmware zu flashen.

Sobald die Firmware erfolgreich hochgeladen wurde und Ihr ESP32-Energiemessgerät vom Home Assistant erkannt wird, können Sie die ursprüngliche YAML-Konfiguration bearbeiten. Öffnen Sie dazu das ESPHome-Dashboard in Home Assistant, suchen Sie Ihr Gerät und klicken Sie auf die Option *Edit* auf der Karte des Energiezählers. Ersetzen Sie die vorhandene Konfiguration durch den YAML-Inhalt aus dem GitHub-Repository [2]. Anschließend konfigurieren Sie in dieser neuen YAML-Konfiguration die API-, OTA- und WLAN-Anmeldedaten. Installieren Sie die neue Konfiguration drahtlos auf Ihrem ESP32-Energiemeter. Sobald dies geschehen ist, wird das Gerät aktiv und verbindet sich. Um die Daten des Energiezählers auf Ihrem Home-Assistant-Dashboard anzuzeigen, weisen Sie das ESPHome-Gerät einfach einem bestimmten Bereich in Home Assistant zu. Dies hilft Ihnen, Ihr Dashboard zu organisieren, indem Sie die Geräte nach ihrem

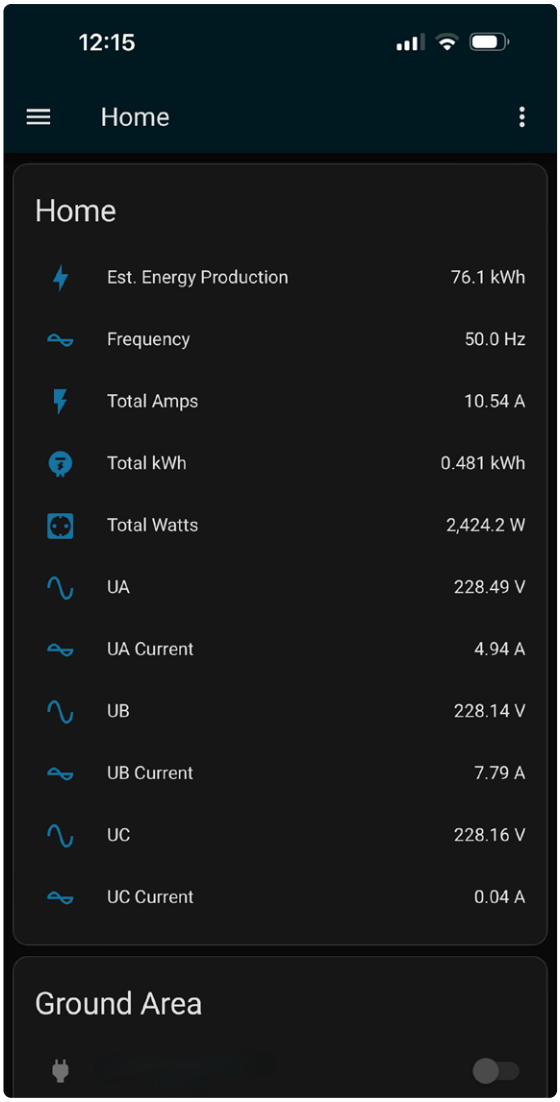


Bild 5. Dashboard des Home Assistant mit Echtzeit-Energiedaten des ESP32-Energiemeters.

Bild 6. Detaillierte Grafiken zum Energieverlauf im Home Assistant.

physischen oder logischen Standort in Ihrem Haus gruppieren. Ein Beispiel, wie eine solche Anzeige der Energiemessdaten auf dem Home-Assistant-Dashboard aussehen könnte, zeigt **Bild 5**.
Die Integration des ESP32-Energiemessgeräts in den Home Assistant vereinfacht nicht nur die Überwachung des Energieverbrauchs, sondern schaltet auch eine Reihe von leistungsstarken Funktionen der Plattform frei. Home Assistant bietet eine intuitive Schnittstelle für die Visualisierung von Daten in Echtzeit, Steuerungsautomatisierung und nahtlose Integration mit anderen intelligenten Geräten in Ihrem Haus.
Diese Integration ermöglicht detaillierte Verlaufsgrafiken und Analysen innerhalb von Home Assistant, die einen detaillierten Einblick in das Muster des Stromverbrauchs im Laufe der Zeit geben, wie in **Bild 6** dargestellt. Dadurch kann man fundierte Entscheidungen über den Energiebedarf treffen, potenzielle Einsparmöglichkeiten ermitteln und die Energieeffizienz des Hauses optimieren.
Mit dieser Einrichtung können die Nutzer alle Vorteile dieser Funktionen nutzen und das ESP32-Energiemeter zu einem zentralen Baustein des Smart-Home-Ökosystems machen. Durch die Integration wird nicht nur die Funktionalität des Energiemessgeräts erweitert, sondern auch das gesamte Smart Home mit umfassenden Energieüberwachungs- und -management-Tools bereichert.

YAML-Konfiguration

Die YAML-Konfiguration richtet das ESP32-Energiemessgerät mit ESPHome ein und ermöglicht die Überwachung wichtiger elektrischer Parameter wie Spannung, Strom und Leistung aller drei Phasen. Sie nutzt die Fähigkeiten des ATM90E32-Sensors, mit detaillierten Definitionen für die SPI-Kommunikation und individuelle Sensoren für jede Phase. Diese Konfiguration misst nicht nur, sondern berechnet auch den Gesamtverbrauch, wobei ein Energiezähler für die Tagesaufnahme in Kilowattstunden und ein OLED-Display zur Visualisierung der Daten in Echtzeit vorhanden sind. Diese Konfigurationen werden gemäß den Anweisungen auf der ESPHome-Seite für den ATM90E32-Sensor [6] vorgenommen.

Bild 7. Aufbau zum Testen und Kalibrieren des ESP32-Energiezählers mit einer variablen Last.

Bild 8. Einsatz des Zangen-Strommessers bei der Kalibrierung.

Damit die vom ESP32-Energiemessgerät gemeldeten Daten ausreichend genau sind, ist eine Kalibrierung unabdingbar. Wie man die Verstärkungseinstellungen für Stromwandler und Spannungseingänge vornimmt, wird im nächsten Abschnitt beschrieben.

Testaufbau und Kalibrierung

Für den Testaufbau wurde ein Haartrockner mit mehreren Heizstufen und Geschwindigkeiten als Last verwendet, der einen Bereich von 0,7...8 A abdeckt. Das Netzkabel einer Verlängerungssteckdose wurde abisoliert, um den klappbaren Stromwandler am stromführenden oder am neutralen Leiter anzubringen, was eine direkte Überwachung unter verschiedenen Bedingungen ermöglichte, wie in **Bild 7** dargestellt.
Ich habe für die Strom- und Spannungskalibrierung des ESP32-Energiemessgeräts mein Digitalmultimeter UT201+ zu Hilfe genommen. Es bietet eine Auflösung von 0,001 A bei einem Fehler von ±4% +10 Digits für den Strom und eine Auflösung von 0,001 V bei einem Fehler von ±1% +5 Digits für die Spannung. Diese Präzision ist für die meisten Projekte ausreichend, aber etwas weniger genau als die meisten professionellen Messgeräte.
Während der Kalibrierung wurde ein Offset der Strommesswerte beobachtet: Der Messwert der Stromzange betrug 1,692 A (siehe **Bild 8**), während die vom Energiemessgerät berechneten Messwerte nach der Kalibrierung 1,70...1,73 A anzeigten (siehe **Bild 9**). In Anbetracht der Spezifikationen des UT201+ und des SCT-013-000, eines Stromwandlers mit geteiltem Kern der Klasse 1 mit einem Fehler von 1% des tatsächlichen Wertes liegt diese kleine Diskrepanz innerhalb der erwarteten Fehlerspanne. Um eine noch höhere Genauigkeit zu erreichen, müsste ein präziseres Zangenmessgerät verwendet werden.
Um eine optimale Messgenauigkeit des ESP32-Energiemeters zu

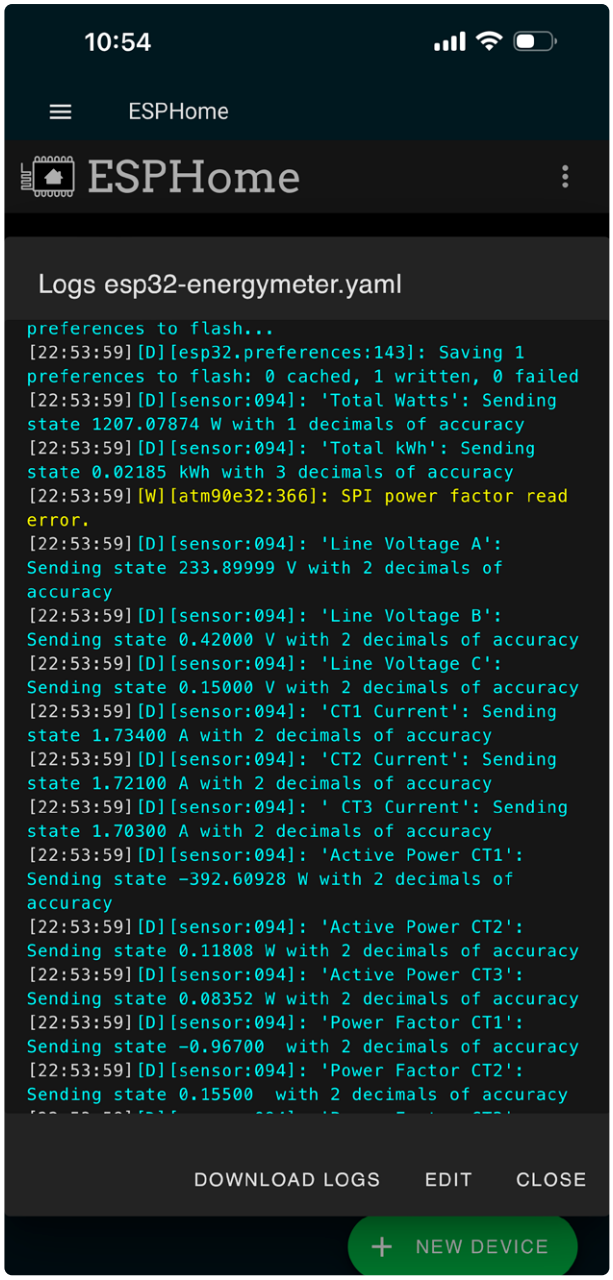


Bild 9. Die endgültigen Ergebnisse der Kalibrierung zeigen eine bessere Genauigkeit der Messung.

WARNUNG: Die Arbeit am und im Zählerkasten birgt Risiken, einschließlich der Gefahr eines elektrischen Schlags oder eines Brandes. Schalten Sie unbedingt den Strom ab, bevor Sie mit der Installation beginnen. In den meisten Ländern darf diese Arbeit ohnehin nur von einem qualifizierten Elektriker durchgeführt werden!

erreichen, wurden die Verstärkungseinstellungen für die Spannungs- und Strommessungen angepasst. Für die Spannung wurde der Sensor mit der folgenden Formel kalibriert:

Neue gain_voltage = (Ihr Spannungsmesswert / ESPHome-Spannungsmesswert) × vorhandener Wert für gain_voltage

Ähnlich verhält es sich mit der Stromeinstellung:

Neue gain_ct = (Ihr aktueller Messwert / ESPHome-Strommesswert) × vorhandener gain_ct-Wert

Diese neuen Verstärkungswerte wurden dann in der YAML-Konfigurationsdatei von ESPHome eingetragen, anschließend neu kompiliert und die Firmware hochgeladen. Dieser Vorgang kann bei Bedarf wiederholt werden, um die Präzision jederzeit „nachzuschärfen". Dies ist für eine genaue Berichterstattung und Analyse in jeder Einrichtung zur Energieüberwachung entscheidend.

Das ESP32-Messgerät im Zählerkasten

Die Installation des ESP32-Energiemessgeräts in meinem Zählerkasten erwies sich zwar als eine überschaubare Aufgabe, die jedoch eine sorgfältige Beachtung der Details erforderte, um sowohl Sicherheit

Bild 10. ESP32-Energiemessgerät, installiert in einer Schalttafel zur Überwachung des Stromverbrauchs in Echtzeit.

als auch Funktionalität zu gewährleisten. Ich begann mit der Auswahl des am niedrigsten abgesicherten Stromkreises, da mir dies einen Sicherheitspuffer bot. Der Schutzschalter würde bei unerwarteten Überspannungen oder Transformatorausfällen auslösen und so das System schützen.

Die Verwendung von klappbaren Stromwandlern war besonders vorteilhaft, weil sie schnell an jede beliebige Last geklemmt werden, aber es war entscheidend, auf die Richtung des Stromflusses zu achten, um genaue Messwerte zu gewährleisten. Wenn die Stromrichtung und der Stromwandler nämlich nicht korrekt ausgerichtet sind, erscheinen die Leistungsmesswerte negativ.

Bild 10 zeigt das in einen Zählerkasten eingebaute ESP32-Energiemessgerät in Aktion mit Anzeige von Strom- und Spannungsmessungen in Echtzeit sowie die entsprechende Last in Kilowatt.

Entwicklung und Aussichten

Während die derzeitige Software auf ESPHome läuft, werden die Möglichkeiten des ESP32-Energiemeters weiter ausgebaut. Die neue Firmware freut sich darauf, in das Projekt integriert zu werden, die speziell dafür entwickelt wurde, das volle Potenzial des ESP32-S3-Chips zu nutzen. Es wird erwartet, dass diese zukünftige Firmware fortschrittliche Funktionen wie detaillierte Energieanalysen und möglicherweise bahnbrechende KI/ML-Funktionen enthält, die Energieverbrauchsmuster vorhersagen und sogar ein Gerät anhand seines Lastprofils identifizieren könnten.

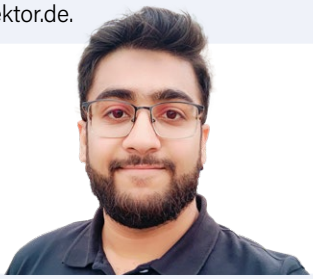
Obwohl die grundlegenden Design- und Betriebsaspekte des Projekts bereits abgeschlossen sind, ist die Entwicklung dieser anspruchsvollen Funktionen ein komplexes und zeitaufwändiges Unterfangen. Ich bin begeistert von den Möglichkeiten und möchte die Grenzen dessen, was dieses Energiemessgeräts erreichen kann, weiter ausreizen.

Das ESP32-Energiemessgerät-Projekt wird kontinuierlich weiterentwickelt und mit jedem Update werden weitere Funktionen hinzugefügt. Maker der Elektor-Community, die sich für die kommenden KI- und ML-Funktionen interessieren oder die zur Entwicklung beitragen möchten, sind herzlich eingeladen, sich zu beteiligen. Die Zusammenarbeit wird dazu beitragen, eine noch robustere und funktionsreichere Energieüberwachungslösung zu entwickeln.

RG — 240244-02

Haben Sie Fragen oder Kommentare?

Wenn Sie Fragen zu diesem Artikel haben, wenden Sie sich bitte an den Autor unter saad.imtiaz@elektor.com oder an die Elektor-Redaktion unter redaktion@elektor.de.



Über den Autor

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

Passende Produkte

- **Stromzange 4350 von PeakTech**
  www.elektor.de/18161
- **Multimeter SDM3045X von Siglent**
  www.elektor.de/17892

WEBLINKS

[1] Saad Imtiaz, „Projekt-Update #2: Energiemeter mit ESP32", Elektor 5-6/2024: https://www.elektormagazine.de/magazine/elektor-342/62845
[2] Github Repository des Energiemessgeräts mit ESP32: https://github.com/ElektorLabs/esp32-energymeter
[3] Home Assistant: https://home-assistant.io/
[4] ESPHome: http://esphome.io
[5] Clemens Valens, „Hausautomatisierung leicht gemacht", Elektor 9-10/2020: https://www.elektormagazine.de/magazine/elektor-154/58936
[6] Power Sensor ATM90E32: https://esphome.io/components/sensor/atm90e32.html

2024 Eine Odyssee in die KI

Verbessern der Objekterkennung: Nutzung verfeinerter Techniken

Von Brian Tristam Williams (Elektor)

Nach dem erfolgreichen Einsatz der grundlegenden Objekterkennung auf einem headless betriebenen Raspberry Pi geht unsere Reise weiter, mit dem Schwerpunkt auf der Verfeinerung dieser Technologie. In dieser Folge geht es um technische Verbesserungen, die Genauigkeit und Effizienz steigern, sowie um die Integration zusätzlicher Sensoren, um die Funktionalität unter verschiedenen Umgebungsbedingungen zu verbessern.

Bild 1. Bei PAL- und NTSC-Videos wurde ein Zeilensprungverfahren eingesetzt, um die Halbbildzeilen darzustellen. Bei (schnellen) Bewegungen war das nicht immer vorteilhaft.

Nachdem ich bei dem Versuch, Objekte und Texte auf alten Archivvideobändern zu erkennen, auf Schwierigkeiten gestoßen war, musste ich nach Möglichkeiten suchen, die Erkennungsalgorithmen zu verbessern. In meinem Fall war die Tatsache, dass das Video, an dem ich gearbeitet habe, ein Zeilensprungverfahren nutzt und zudem eine niedrige Auflösung hat, wahrscheinlich ein Faktor, der Fehler verursacht. Die Unterschiede zwischen sich abwechselnden Halbbildern können das Erkennungsmodell verwirren, das für die Analyse von jeweils einem Bild (statt zwei Halbbildern) eingerichtet ist (**Bild 1**). Ich musste das Old-School-Video zunächst „deinterlacen", bevor ich es durch den Workflow laufen lasse.

Nachbessern

Abgesehen von den Problemen in meinem Spezialfall bedeutet das jedoch nicht, dass wir die Erkennung nicht mit einigen Verfeinerungen noch ein wenig verbessern können. Ich habe herausgefunden, dass dies bei der Arbeit mit digitalem Videomaterial ohne Zeilensprungverfahren, das vom Raspberry Pi Camera Module 3 aufgenommen wurde, hilfreich ist:

Dynamische Schwellenwertbildung: Um schwankenden Lichtverhältnissen und Entfernungen entgegenzuwirken, habe ich eine adaptive Konfidenzschwelle implementiert. Dieser Ansatz passt die Erkennungswahrscheinlichkeit in Echtzeit auf der Grundlage der von der Kamera erkannten durchschnittlichen Helligkeit an, wobei ein einfacher Helligkeitsalgorithmus verwendet wird:

```
def adjust_threshold(lux):
    base_threshold = 0.5  # base confidence level
    if lux < 50:  # low light conditions
        return base_threshold - 0.1
    elif lux > 500:  # very bright conditions
        return base_threshold + 0.1
    return base_threshold
```

Integration eines Lux-Sensors: Apropos Beleuchtung: Ich weiß zwar, dass auch die Kamera selbst eine Art Helligkeitssensor ist, aber nachdem ich mir vergeblich die Haare gerauft habe, als ich versuchte, die zahlreichen Parameter des Kameramoduls so einzustellen, dass meine Bilder und Videos „genau richtig" sind, weiß ich, dass die Helligkeit des ausgegebenen Videosignals in keinem Verhältnis zur tatsächlichen Helligkeit im Freien steht - das ist nur eine weitere verwirrende Variable, mit der ich mich nicht weiter auseinandersetzen möchte. Also habe ich mich kurzerhand dazu entschlossen, einen eigenen Sensor für diesen Parameter zu verwenden.
Ich entschied mich für den Lux-Sensor TLS2561 (**Bild 2**), der in Echtzeit Daten an den Raspberry Pi liefert, der dann die Belichtung und die Verarbeitungsparameter der Kamera dynamisch anpasst. Dies gewährleistet eine optimale Bildqualität für die Erkennungsalgorithmen bei unterschiedlichen Lichtverhältnissen:

```
lux = read_lux_sensor()
camera.set_exposure(calculate_exposure(lux))
```

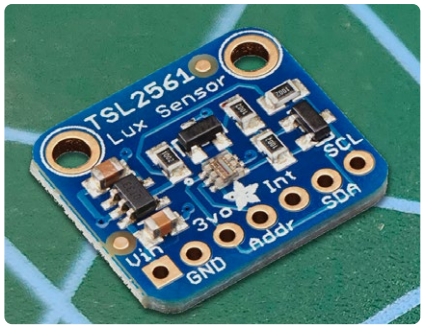


Bild 2. Der Helligkeitssensor TSL2561 auf einem Modul von Adafruit.

Auch hier musste ich keine Funktionen schreiben, da es von Adafruit schon eine großartige TSL2561-Bibliothek gibt [1]. Die Installation erfolgt einfach über das Terminal:

```
sudo apt-get update
sudo apt-get install python3-smbus
pip3 install Adafruit-TSL2561
```

Mit diesem Python-Code können Sie dann die Lux-Werte lesen und ausgeben:

```
import time
from Adafruit_TSL2561 import TSL2561

# Initialize the sensor
tsl = TSL2561()

while True:
  lux = tsl.calculate_lux()
  print(„Current Lux: „, lux)
  time.sleep(1)  # Delay for 1 second
```

Fix und fertige Bibliotheken im Internet machen die Arbeit heutzutage wirklich sehr viel einfacher. Früher musste man für jeden neuen Sensor, den man kaufte, das Datenblatt studieren und seine eigenen Funktionen oder Assembler-Unterprogramme entwickeln.

Verbesserte Frame-Verarbeitung: Durch die Aufnahme von `cv2.GaussianBlur()` vor der Objekterkennung wurden die Auswirkungen von Rauschen und Körnung in schwach beleuchteten Videobildern reduziert. Ich fand dies besonders nützlich, als ich das Raspberry Pi Camera Module 3 NoIR (**Bild 3**, rechts) für die Nachtüberwachung testete. Dieses Kameramodell hat kein Infrarotfilter, so dass es Infrarotlicht recht gut erkennen kann. Es hängt jedoch immer noch von der Entfernung zwischen der Infrarotbeleuchtung und dem Objekt ab, und manchmal stellen entfernte Objekte eine Herausforderung dar.
Durch den Aufruf dieser Funktion konnte das Modell mit deutlich weniger Rauschen arbeiten, auch wenn dies die Schärfe des Eingangsbildes beeinträchtigte. Wenn Sie jedoch Objekte im Makromaßstab erkennen wollen und nicht gerade versuchen, Nummernschilder zu lesen, ist es dem Modell egal, ob Ihr „Mensch“ ein wenig unscharf ist - es obliegt dem Benutzer, den Kamera-Stream auf Details zu überprüfen, sobald ein Alarm ausgelöst wird. Der Funktionsaufruf besteht aus einer einzigen Codezeile:

```
frame = cv2.GaussianBlur(frame, (5, 5), 0)
```

Multi-Model-Unterstützung: Zunächst entschied ich mich für ein einziges Modell, nachdem ich mehrere ausprobiert hatte, aber es war ein wenig enttäuschend, dass es kein Modell gab, das für alle Konfigurationen und Szenarien geeignet war. Dann wurde mir klar, als sich die Zahnräder in meinem Kopf viel zu langsam drehten, dass man tatsächlich einfach zwischen den Modellen hin und her wechseln kann. Die Einbindung eines sekundären Erkennungsmodells, das auf bestimmte Ziele spezialisiert ist, erhöht die Präzision. Beispielsweise ermöglicht die Integration eines vereinfachten YOLO-Modells eine zuverlässigere Fahrzeug- und Fußgängererkennung. Das Umschalten zwischen den Modellen auf der Grundlage des Szenenkontextes wird wie in diesem recht groben Beispiel gehandhabt:

```
if scene == 'urban':
  model.load('yolo_city.tflite')
else:
  model.load('tensorflow_lite_default.tflite')
```

Verringerung der Latenzzeit: Leider stieg mit der Komplexität der Erkennung auch die Latenzzeit. Ich habe versucht, die Modellverarbeitung asynchron zu implementieren, was dazu beigetragen hat, die Dinge in Bezug auf die rohen Zahlen zu beschleunigen, auch wenn der Gewinn für mich nicht sehr auffällig war. Nichtsdestotrotz soll mir jede Hilfe willkommen sein:

```
from concurrent.futures import ThreadPoolExecutor

with ThreadPoolExecutor() as executor:
  future = executor.submit(process_frame, frame)
  result = future.result()
```

Hier sehen wir die Verwendung des `ThreadPoolExecutor()` aus dem Modul `concurrent.futures`, das Teil der Standardbibliothek von Python ist. Dieser Ansatz behandelt die asynchrone Ausführung von Aufgaben in separaten Threads, was besonders nützlich für Anwendungen der Echtzeit-Videoverarbeitung sein kann, bei denen die Reaktionsfähigkeit entscheidend ist.
Nach der ersten Zeile, die den `ThreadPoolExecutor()` importiert, führen die nächsten drei Zeilen jeweils die folgenden Aufgaben aus:

1. Erstellen einer `ThreadPoolExecutor`-Instanz, die einen Pool von Threads zur asynchronen Ausführung von Aufrufen verwaltet. Die `with`-Anweisung sorgt dafür, dass die Ressourcen des Executors ordnungsgemäß verwaltet werden und dieser automatisch herunterfährt, wenn er nicht mehr benötigt wird.
2. Die `submit()`-Methode plant die Ausführung der Funktion `process_frame()` mit `frame` als Argument. Die Funktion wird in einem separaten Thread ausgeführt, der vom Executor verwaltet wird. `submit()`

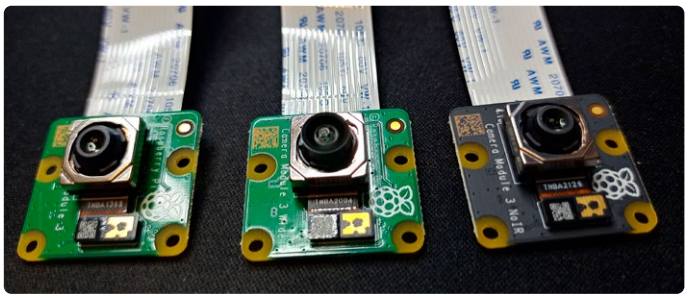


Bild 3. Neben dem Wide- und dem „normalen“ Camera Modul 3 habe ich die NoIR-Variante rechts ausprobiert, weil sie nachts von der Infrarotbeleuchtung profitieren kann.

gibt ein `future`-Objekt zurück, das das endgültige Ergebnis des Funktionsaufrufs darstellt.
3. Blockiert den Hauptthread, bis die Funktion `process_frame()` abgeschlossen ist und ein Ergebnis zurückgibt. Die `result()`-Methode ruft das Ergebnis der Funktion ab, wenn sie beendet ist. Wenn die Funktion eine Ausnahme auslöst, wird sie hier erneut ausgelöst.

Durch die Auslagerung intensiver Aufgaben auf gesonderte Threads kann der Thread des Hauptprogramms reaktionsfähig bleiben. Dies ist besonders wichtig bei grafischen Benutzeroberflächen oder Echtzeitanwendungen, bei denen eine träge Reaktion das Benutzererlebnis negativ beeinflussen würde.
Der Einsatz dieses Systems als Teil einer Gebäudeüberwachung hat gezeigt, dass es in der Lage ist, eine Person von einem Hund oder einem Baum zu unterscheiden, aber es ist nicht so ausgeklügelt wie die Gesichtserkennung, um zwischen verschiedenen Personen zu unterscheiden.

Wohin jetzt?

Abgesehen von all der Zeilenentflechtung (Deinterlacing) und der Vorverarbeitung, die ich machen muss, um mit meinem Projekt für alte Videos voranzukommen, gibt es noch viele andere Möglichkeiten, die ich mit TensorFlow Lite erforschen kann, und ich bin mir nicht sicher, welche davon (für mich und für Sie) am interessantesten sind. Wenn Sie möchten, können Sie gerne mit mir in Kontakt treten und mir mitteilen, ob es noch Möglichkeiten gibt, die Sie gerne erforschen möchten! Einige Dinge, die wir ausprobieren könnten:

Bild- und Videoverarbeitung: Ich arbeite immer noch daran. Fortgeschrittene Bild- und Videoverarbeitung für Anwendungen wie Bildsegmentierung werden unterstützt, bei der das Modell verschiedene Objekte im Bild identifiziert und sie vom Hintergrund trennt, oder Stilübertragung, bei der der Stil eines Bildes auf den Inhalt eines anderen Bildes übertragen wird.
Echtzeit-Objekterkennung und -klassifizierung: TensorFlow Lite kann Modelle ausführen, die Objekte in Echtzeit erkennen und klassifizieren, was für Anwendungen wie Sicherheitskameras, automatisierte Qualitätskontrolle und Wildtierüberwachung nützlich ist. Vortrainierte Modelle wie MobileNet können angepasst und eingesetzt werden, um Objekte mit hoher Effizienz zu identifizieren.
Spracherkennung und Audioverarbeitung: TensorFlow Lite kann Modelle verarbeiten, die für die Spracherkennung entwickelt wurden, was sprachaktivierte Anwendungen, die Umwandlung von Sprache in Text und andere Audioverarbeitungsaufgaben ermöglicht. Für mich bedeutet dies, vergessene Geschichten aus den verstaubten Medienarchiven zu retten, aber es ermöglicht auch die Entwicklung von „freihändigen" Steuerungen oder Hilfsmitteln für Menschen mit Behinderung.
Verarbeitung natürlicher Sprache: Natural Language Processing (NLP) unterstützt leichtgewichtige Sprachverarbeitungsmodelle, die Aufgaben wie Stimmungsanalyse, Spracherkennung oder sogar die Steuerung einfacher Chatbots übernehmen können. Diese Fähigkeit ist besonders nützlich für Anwendungen, die Benutzerinteraktion und Verarbeitung von Feedbacks erfordern.
Gestenerkennung: Mit TensorFlow Lite können Sie Systeme entwickeln, die menschliche Gesten verstehen und als Befehle interpretieren, um interaktive Installationen zu ermöglichen oder die Benutzeroberflächen von Geräten zu verbessern.
Anomalie-Erkennung: TensorFlow Lite kann für die Erkennung von Anomalien oder Ausreißern in Zeitseriendaten verwendet werden, was für die vorausschauende Wartung in der Industrie oder für Überwachungssysteme wie die Erkennung unregelmäßiger Herzschläge durch medizinischen Geräte wertvoll ist.
Erkennung der Körperhaltung: Das Framework ist in der Lage, Modelle zur Einschätzung von Posen auszuführen, die menschliche Figuren und ihre Körperhaltungen in Bildern oder Videos erkennen. Dies kann in der Sportanalytik, bei Fitness-Apps und fortgeschrittenen interaktiven Anwendungen eingesetzt werden, wie wir bei unserem kürzlichen Besuch der embedded world 2024 in Echtzeit beobachten konnten.

Mit den Erweiterungen von TensorFlow Lite auf dem Raspberry Pi machen wir solide Fortschritte bei der Objekt- und Texterkennung, wenn auch nicht bei niedrig aufgelösten und Interlaced-Videos. Diese Verbesserungen könnten Ihnen helfen, Echtzeitanwendungen effektiv zu verwalten und sind besonders nützlich für Projekte wie Gebäudeüberwachung und Medienarchivierung. Es gibt noch viel mehr zu erforschen und zu verbessern, und ich freue mich darauf zu sehen, wie wir gemeinsam diese Tools weiter an unsere Bedürfnisse anpassen.

RG — 230181-G-02

Haben Sie Fragen oder Kommentare?

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

Über den Autor

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

WEBLINK

[1] Adafruit, Python-Bibliothek für den TSL2561 auf dem Raspberry Pi: https://github.com/adafruit/TSL2561-Arduino-Library

▲ *Bild 1. Das steckt im Raspberry Pi AI-Kit.*

Raspberry Pi Goes AI

Neues Kit enthält M.2-HAT mit KI-Beschleuniger

Von Brian Tristam Williams (Elektor)

Das Raspberry Pi AI-Kit, das den M.2-HAT+ und das Hailo-8L AI Accelerator Board enthält, liefert dem Raspberry Pi 5 13 TOPS Leistung. Das macht es perfekt für Echtzeit-KI-Anwendungen, Edge Computing, Robotik und Computer Vision. Der Integrationsprozess ist unkompliziert, und es gibt umfangreiche Softwareunterstützung.

Raspberry Pi hat ein Kit auf den Markt gebracht, das aus seinem M.2 HAT+ und dem KI-Beschleuniger Hailo-8L besteht. Diese neue Kombination bringt ernstzunehmende KI-Power auf den Raspberry Pi 5 und bietet Entwicklern und Ingenieuren eine beeindruckende Inferenzleistung, mit der sie entwickeln können. Werfen wir einen genaueren Blick auf das Raspberry Pi AI-Kit.

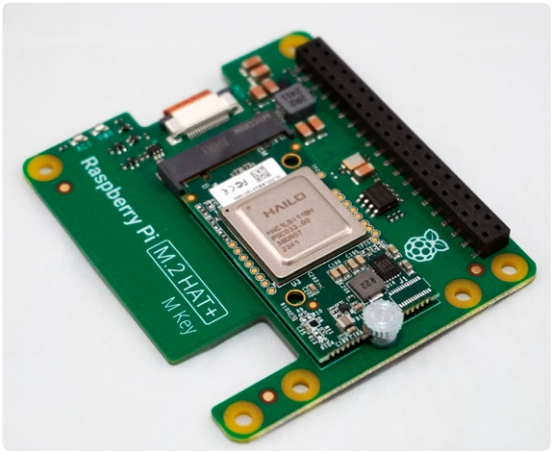
Warum an der Edge leben?

Edge Computing verarbeitet Daten lokal auf den Geräten, anstatt sich auf cloudbasierte Server zu verlassen. Dieser Ansatz reduziert die Latenzzeiten erheblich, sorgt für schnellere Reaktionszeiten und verbessert die Zuverlässigkeit. Dies ist von entscheidender Bedeutung für Anwendungen, die Echtzeit-Entscheidungen erfordern, zum Beispiel in der Robotik, bei autonomen Fahrzeugen und intelligenten Haushaltsgeräten. Durch die Verwaltung von Daten „am Rande“ des Netzwerks kann die Leistung gesteigert, die Bandbreiten-Beanspruchung, die Latenzzeit verringert und die Datensicherheit verbessert werden.

Der Haken an der Sache ist, dass für den Einsatz direkt an der Edge des Systems typischerweise Hardware benötigt wird, die nicht annähernd die Leistung der Cloud-Serverfarmen hat. Das Raspberry Pi AI-Kit zielt darauf ab, dieses Problem zu lösen. Die Integration des Hailo-8L mit dem M.2-HAT+ des Raspberry Pi ist ein Beispiel für diesen Wandel hin zu effizienteren und reaktionsschnelleren KI-gesteuerten Lösungen. In einem kürzlich geführten Chat betonte Eben Upton, Mitbegründer von Raspberry Pi, die wachsende Bedeutung von Edge Computing: „Da immer mehr Geräte intelligent werden, steigt der Bedarf an lokaler KI-Verarbeitung. Unser Ziel mit diesem neuen KI-Kit ist es, Entwicklern, die im Edge-Bereich arbeiten, den Zugang zu High-Performance-Computing zu erleichtern.“

Was steckt in der Box?

In der Verpackung des Kits (**Bild 1**) befinden sich ein Raspberry Pi M.2-HAT+ mit einer fertig ausgestatteten Hailo-8L-M.2-Beschleunigerkarte, ein 16-mm-Stapel-Header, vier Abstandshalter und acht Schrauben. Der Hailo-8L ist bereits auf dem Raspberry Pi M.2-HAT+ montiert (**Bild 2**).



◀ *Bild 2. Das Kit besteht aus dem Hailo-8L-Chip auf seiner Beschleunigerplatine im M.2-Format, die mit dem Raspberry Pi M.2-HAT+ verbunden ist.*

Bild 3. Die Beschleunigerplatine bietet die Möglichkeit, den Hailo-8L-Chip über den M.2-Anschluss mit einer PCIe-Schnittstelle zu verbinden.

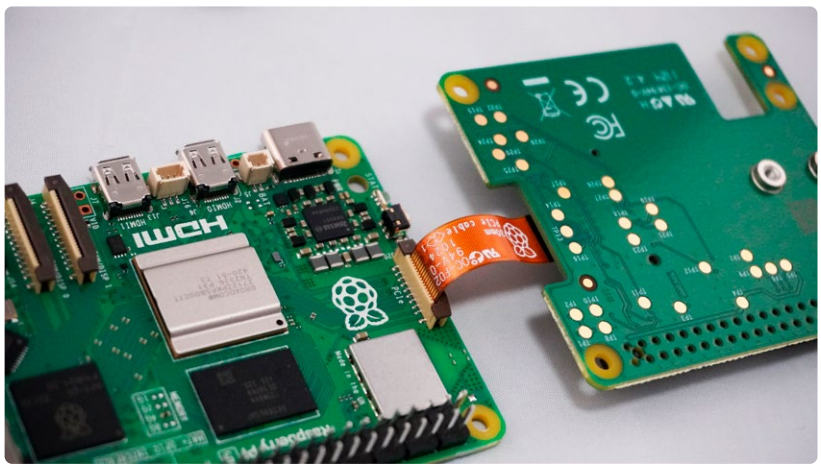
Der Hailo-8L-Chip

Was hat es nun mit dem Hailo-8L auf sich? Zunächst einmal ist dieser KI-Beschleuniger ein echtes Kraftpaket, das sich perfekt für Echtzeit-KI-Aufgaben eignet. Das bedeutet, dass Sie Projekte in den Bereichen Robotik, Computer Vision, Smart-Home-Geräte und Industrieautomatisierung in Angriff nehmen können, ohne dass der Rechner ins Schwitzen gerät. Mit dem M.2-M-2242-Anschluss der Hailo -Beschleunigerkarte (22 mm × 42 mm) in **Bild 3** können Sie diese Leistung ganz einfach nutzen und die Vielseitigkeit des Raspberry Pi 5 voll ausschöpfen. Stecken Sie die Karte in den M.2-HAT+, der wiederum über ein 30 mm langes Flachbandkabel mit dem PCIe-Port des Raspberry Pi 5 verbunden wird (**Bild 4**).

Der KI-Beschleuniger Hailo-8L wurde entwickelt, um Edge-Geräte mit einer Leistung der Rechenzentrumsklasse auszustatten und bietet bis zu 13 TOPS (Billionen von Operationen pro Sekunde). Dies macht ihn zur idealen Lösung für Einsteigerprojekte, die KI-Kapazität im lokalen Bereich benötigen. Der Hailo-8L zeichnet sich durch seine überragende Energieeffizienz auf geringster Fläche aus, die ihn im Vergleich zu anderen Lösungen seiner Kategorie äußerst wettbewerbsfähig macht.

Durch seine hocheffiziente Verarbeitung glänzt der Hailo-8L mit geringer Latenz, die komplexe Pipelines mit mehreren Echtzeit-Streams und die gleichzeitige Verarbeitung verschiedener Modelle und KI-Aufgaben bewältigen kann. Dieser Beschleuniger ist außerdem mit der umfassenden Software-Suite des Hailo-8 kompatibel, was eine nahtlose Aufrüstung auf höhere KI-Kapazitäten in der Zukunft gewährleistet.

Bild 4. Das Raspberry Pi AI-Kit wird mit dem PCIe-Anschluss des Raspberry Pi 5 verbunden.



Der KI-Beschleuniger Hailo-8L enthält eine umfassende Software-Suite mit Hailo-Gerätetreibern, *HailoRT* und *HailoTappas*, die einfach über den apt-Paketmanager installiert werden können. Dies ermöglicht eine nahtlose Einrichtung und Bedienung. Die Unterstützung der KI-Beschleunigung ist vollständig in den Kamera-Software-Stack des Raspberry Pi integriert, einschließlich der Unterstützung für *libcamera*, *rpicam-apps* und *picamera2*. Dies ermöglicht fortgeschrittene Bildverarbeitung und KI-Anwendungen direkt auf dem Raspberry Pi 5.

Haupteigenschaften und Vorteile:

- **Hoher Wirkungsgrad**: Beim Hailo-8L dreht sich alles um hochleistungsfähige KI-Verarbeitung bei minimaler Leistungsaufnahme. Dies ist entscheidend für Edge-Geräte, bei denen ein Kompromiss zwischen Leistung und Energieeffizienz erforderlich ist.
- **KI-Verarbeitung in Echtzeit**: Mit seinen 13 TOPS bewältigt der Hailo-8L komplexe Aufgaben wie Objekterkennung, Bildklassifizierung und Spracherkennung in Echtzeit. Ihre Anwendungen können reaktionsschneller und intelligenter werden.
- **Nahtlose Integration**: Das Kit wird mit dem vormontierten Hailo-8L geliefert. Schließen Sie ihn einfach an Ihren Raspberry Pi 5 an, und schon können Sie loslegen. Ohne Probleme, ohne Stress.
- **Robuste Software-Unterstützung**: Vollständig in die Raspberry-Pi-OS-Umgebung integriert, können Sie die Softwarepakete von Hailo einfach über den apt-Paketmanager installieren. Dazu gehören Gerätetreiber und Bibliotheken für neuronale Netzwerke, sodass Sie mit Ihren KI-Projekten sofort beginnen können.

Dieses Setup vereint die Erschwinglichkeit und Vielseitigkeit des Raspberry Pi mit den fortschrittlichen KI-Fähigkeiten des Hailo-8L. Das Ergebnis? Sie können anspruchsvolle KI-Modelle direkt vor Ort ausführen und dabei Latenzzeiten und Bandbreitenbedarf im Vergleich zu Cloud-basierten Lösungen verringern. Der Hailo-8L unterstützt beliebte KI-Frameworks wie TensorFlow, TensorFlow Lite, Keras, PyTorch und ONNX und ist mit ARM-Host-Architekturen

kompatibel. Das macht es ideal für den Einsatz von anspruchsvollen KI-Modellen auf dem Raspberry Pi 5 und eröffnet uns ganz neue Möglichkeiten, was wir mit unseren Raspberry-Pi-Setups erreichen können.
Upton erläuterte die Entscheidung für die Zusammenarbeit mit Hailo: „Was uns an Hailo begeistert hat, war seine hohe Leistung und Effizienz. Der Hailo-8L bringt 13 TOPS mit, deutlich mehr als bisherige Lösungen. In Kombination mit den hervorragenden Werkzeugen für die Modellkonvertierung war dies eine natürliche Ergänzung für unsere Ambitionen im Bereich der Edge-KI."

Entwicklung und Herausforderungen

Auf die Frage, ob er voraussah, dass der M.2-HAT neben NVMe-SSDs auch für KI-Beschleuniger verwendet werden würde, antwortete er: „Als wir den Raspberry Pi 5 auf den Markt brachten, dachten wir zunächst, dass der M.2-HAT hauptsächlich für Speichermodule verwendet werden würde. Wir haben jedoch schnell festgestellt, dass es ein großes Interesse auch an KI-Beschleunigern, Netzwerkkarten und Grafiklösungen gibt."
Was die unvermeidlichen Hindernisse angeht, die bei der Zusammenstellung einer so anspruchsvollen Kombination auftreten, erinnert er sich: „Eine Herausforderung, der wir kurz vor der Markteinführung gegenüberstanden, war die Optimierung des thermischen Verhaltens. Ursprünglich hatten wir kein Wärmeleitpad zwischen dem Beschleuniger und dem Baseboard, aber wir haben es in der endgültigen Version integriert, um die Wärmeableitung zu verbessern."
Trotz dieser Hürden erwies sich die Zusammenarbeit zwischen Raspberry Pi und Hailo als fruchtbar. Die Stärken beider Unternehmen wurden genutzt, um ein robustes und effizientes Produkt zu entwickeln.
Upton erwähnte auch die Schwierigkeiten bei der Anwendung von KI-Beschleunigung aufgrund der sich schnell ändernden Architekturen der Modelle: „Eine der Herausforderungen, die wir meiner Meinung nach derzeit mit GenAI haben, ist, dass es sehr schwierig ist, Beschleunigung anzuwenden. Die Dinge müssen sich erst einpendeln, bevor man einen Beschleuniger entwickeln kann, der sinnvollerweise leistungsfähiger ist als eine CPU oder GPU, ohne dabei an Flexibilität einzubüßen."

Anwendungsfälle und ideale Anwendungen

Upton zeigte sich begeistert von der Zukunft der KI auf der Raspberry-Pi-Plattform: „Wir werden einige erstaunliche Anwendungen in der Industrierobotik und in intelligenten Heimgeräten sehen. Das Potenzial ist enorm, und dieses KI-Kit wird dazu beitragen, es für unsere Nutzer zu erschließen", sagte Upton. Er wies auch auf die Beliebtheit von Kameras als Zubehör für den Raspberry Pi hin: „Kameras sind ein unglaublich beliebtes Zubehör für den Raspberry Pi, und die Möglichkeit, mehr Intelligenz in diesen Kameraanwendungen anzuwenden, ist eine große Sache. Dieser KI-Beschleuniger ist perfekt für Aufgaben wie Objekterkennung und Bildklassifizierung."
Die Hailo-8L ist für bildverarbeitungsbezogene Aufgaben optimiert und damit ideal für Anwendungen wie Überwachung, automatische Qualitätskontrolle in der Fertigung und Bildgebung im Gesundheitswesen. „Eine der Herausforderungen bei GenAI sind die sich schnell ändernden Architekturen der Modelle. Der Hailo-8L ist jedoch darauf ausgelegt, stabilere, bildverarbeitungsbezogene Aufgaben effizient zu bewältigen."

Partnerschaft zwischen Raspberry Pi und Hailo

Beim Raspberry Pi geht es darum, Hochleistungscomputer zugänglich und erschwinglich zu machen. Hailo hingegen hat sich auf KI-Prozessoren spezialisiert, die die Leistung von Rechenzentren auf Edge-Geräte übertragen. Diese Partnerschaft zeigt, was möglich ist, wenn man modernste Hardware-Innovationen mit KI-Beschleunigung kombiniert.
Upton ging auch auf die edukativen Auswirkungen ihrer Produkte ein: „Unser Ziel war es immer, programmierbare Hardware in die Hände von Kindern zu geben und zu sehen, was passiert. Es geht darum, die Neugierde zu fördern und die nächste Generation von Ingenieuren und Entwicklern zu unterstützen."
Für diejenigen unter uns, die immer auf der Suche nach neuen Möglichkeiten der künstlichen Intelligenz sind, ist das neue Raspberry Pi AI Kit ein echter Wendepunkt. Es eröffnet uns ganz neue Möglichkeiten, was wir mit unseren Raspberry-Pi-Setups erreichen können.

RG – 240298-02

Sie haben Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare? Dann senden Sie bitte eine E-Mail an den Autor unter brian.williams@elektor.com.

Passendes Produkt

> **Raspberry Pi AI Kit**
> www.elektor.de/20879

Sensoren für Wetterstationen

Welchen Sensor sollte man wählen?

Von Saad Imtiaz (Elektor)

Wenn Sie eine Wetterstation bauen, können Sie aus einer Vielzahl von Sensoren wählen. Sie sollten genau, zuverlässig und nicht zu teuer sein und sich leicht mit einem Mikrocontroller Ihrer Wahl steuern lassen. In diesem Artikel werden wir einige Umweltsensoren im Detail vergleichen.

Die Auswahl der Sensoren ist ein entscheidender Schritt beim Aufbau einer eigenen Wetterstation und hat großen Einfluss auf die Genauigkeit und Zuverlässigkeit der gesammelten Daten. Angesichts der vielen verfügbaren Optionen ist es wichtig, die Stärken und Grenzen der verschiedenen Umweltsensoren zu kennen. Dieser Artikel bietet einen detaillierten Vergleich verschiedener Sensoren, die in Wetterstationen verwendet werden können, mit Schwerpunkt auf deren Leistungsfähigkeit, einfache Integration und allgemeine Zuverlässigkeit. Durch die Erläuterung der technischen Aspekte dieser Sensoren können fundierte Entscheidungen getroffen werden, um die Genauigkeit der Umweltmessungen zu verbessern. **Bild 1** zeigt eine vollständig ausgestattete Wetterstation, die in Elektor [1] vorgestellt wurde.

Bild 1. Wetterstationsprojekt des ehemaligen Elektor-Ingenieurs Mathias Claussen [1].

Auswahlkriterien für den Sensorvergleich

Bevor mit dem Vergleich von Umweltsensoren für Wetterstationen begonnen werden kann, sind genaue Richtlinien festzulegen, um sich in der breiten Palette der verfügbaren Sensoren zurechtzufinden. Der Fokus wird anhand der folgenden Kriterien eingegrenzt:

Qualität: Die Analyse beschränkt sich auf Sensoren, die für Consumer-Anwendungen entwickelt wurden. Diese Sensoren bieten ein ausgewogenes Preis-Leistungs-Verhältnis und sind daher ideal für Maker, Bastler und Enthusiasten, die eine eigene Wetterstation bauen oder verbessern wollen.

Verfügbarkeit: Es werden nur Sensoren berücksichtigt, die auf dem Markt leicht erhältlich sind. Die Zugänglichkeit ist wichtig, damit die Menschen die Bauteile nutzen können, ohne warten zu müssen, bis sie verfügbar sind, oder sich auf die Suche nach einem anderen Lieferanten machen zu müssen.

Modulare Lösungen: Der Vergleich konzentriert sich auf Sensoren, die als Module angeboten werden. Dies erleichtert die Integration, da Module in der Regel bereits die erforderliche Signalverarbeitung und Schnittstellenschaltung enthalten, so dass sie für eine Vielzahl von Anwendungen geeignet sind, ohne dass ein komplexes Elektronikdesign erforderlich ist.

Preisspanne: Der Vergleich bezieht sich auf Sensoren mit einem Verkaufspreis zwischen 2 € und 20 €. Diese Preisspanne wurde gewählt, um ein breites Publikum anzusprechen, auch Hobbyisten und Bildungseinrichtungen, damit der Aufbau einer eigenen Wetterstation ein erschwingliches Projekt bleibt.

Interoperabilität und einfache Integration: Ein wichtiger Aspekt unserer Richtlinien ist die einfache Integration in bestehende Systeme. Bevorzugte Sensoren sind solche, die mit gängigen Mikrocontrollern und Entwicklungsplattformen wie Arduino und Raspberry Pi zurechtkommen. Dadurch können Nutzer diese Sensoren einfach in ihre Projekte integrieren, ohne dass ein zusätzlicher Lernprozess oder eine Neukonfiguration des Systems erforderlich ist.

Sensoren

In einer typischen Wetterstation, insbesondere einer, die für den privaten oder schulischen Gebrauch konzipiert ist, wird in der Regel ein grundlegender Satz von Umweltsensoren verwendet,

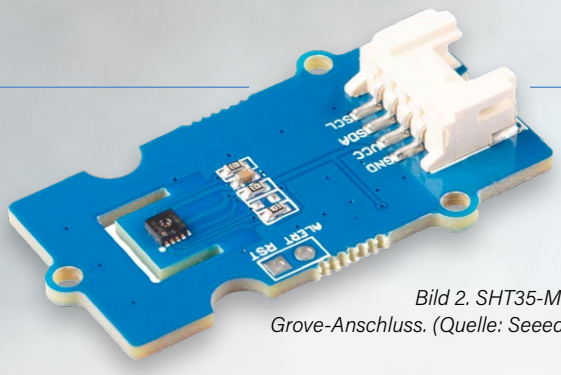


Bild 2. SHT35-Modul mit Grove-Anschluss. (Quelle: Seeed Studio)

um verschiedene atmosphärische Parameter zu überwachen. Der Schwerpunkt liegt auf den folgenden Sensortypen, die alle eine Schlüsselrolle in einer Wetterstation spielen:

Temperatur- und Feuchtesensoren: Diese Sensoren sind von grundlegender Bedeutung für die Messung des Wärme- und Feuchtigkeitsgehalts der Luft. Die Temperatur ist ein wichtiger meteorologischer Parameter, während die Luftfeuchtigkeit für die Beurteilung des Niederschlagsrisikos und des Komforts entscheidend ist.

Barometrische Drucksensoren: Druckmessungen sind der Schlüssel zur Vorhersage von Wetteränderungen. Ein Luftdrucksensor gibt Auskunft über Wettertendenzen, zum Beispiel herannahende Fronten oder Stürme, indem er Schwankungen des Luftdrucks erkennt.

Sensoren für Windgeschwindigkeit und Windrichtung (Anemometer und Windfahnen): Diese Sensoren messen die Windgeschwindigkeit und -richtung, wichtige Daten für die Wettervorhersage und die Untersuchung von charakteristischen Windverhältnissen an einem bestimmten Ort.

Sonnenstrahlungssensoren: Für eine umfassende Wetteranalyse ist die Messung der auf die Erdoberfläche auftreffenden Sonnenenergie unerlässlich. Sonnenstrahlungssensoren oder Pyranometer messen die Intensität des Sonnenlichts und tragen so zu den Daten über Wetterbedingungen und Solarenergiepotenzial bei.

UV-Index-Sensoren: In einigen modernen oder spezialisierten Wetterstationen überwachen UV-Index-Sensoren den Anteil der ultravioletten Strahlung der Sonne. Diese Informationen sind für die Gesundheits- und Umweltüberwachung von entscheidender Bedeutung, da sie das Sonnenbrandrisiko und die Auswirkungen der UV-Strahlung auf die Ökosysteme anzeigen.

Diese Sensoren bilden das Rückgrat einer Wetterstation und bieten einen umfassenden Überblick über die atmosphärischen Bedingungen. Diese Sensoren sind sowohl erschwinglich als auch einfach in eigene Wetterstation (in spe) zu integrieren.

Temperatur und Feuchte

In diesem Abschnitt werden verschiedene Temperatur- und Luftfeuchtigkeitssensoren verglichen, wobei der Schwerpunkt auf Genauigkeit, Betriebsbereich und Stromaufnahme liegt, damit Sie leichter den am besten geeigneten Sensor für eine zuverlässige und effiziente Umweltüberwachung auswählen können.
Die Sensoren **SHT35** und **SHT40** von Sensirion [2] sind für ihre hohe Genauigkeit von bis zu 0,1°C bei Temperaturmessungen und bei Feuchtemessungen bis zu 1,5% RH bekannt. **Bild 2** zeigt das SHT35-Modul von Seeed Studio. Mit diesen Modulen sind sehr präzise Umweltmessungen möglich, was aber im Vergleich zu anderen Optionen ihren (höheren) Preis hat. Der SHT40 ist sinnvoll in Projekten, bei denen die Kosten eine Rolle spielen, da er eine kostengünstigere Lösung mit etwas geringerer Genauigkeit bietet.
Der Sensor **AHT20** von Aosong Electronic [3] arbeitet mit einer Versorgungsspannung von 2,2…5,5 V und misst die Luftfeuchtigkeit von 0…100% RH und die Temperatur von -40…+85 °C. Er bietet eine Genauigkeit (oder besser, einen Fehler) von ±2% RH beziehungsweise ±0,3 °C. Der Sensor ist sehr genau und reagiert innerhalb von 5 s auf Temperatur- und 8 s auf Feuchteschwankungen. Er verwendet das serielle I²C-Protokoll zur Kommunikation mit vielen kompatiblen Mikrocontrollern.
Der Sensor **SHTC3** von Sensirion [4] kann Temperatur und Feuchte mit einer Genauigkeit von 0,2 °C und 2% RH messen. Er arbeitet in einem Temperaturbereich von -40…+125 °C und einem Feuchtebereich von 0…100% RH mit einer Ansprechzeit von circa 5 s. Er wird mit einer Spannung von 1,62…3,6 V betrieben und hat eine extrem niedrige durchschnittliche Stromaufnahme von 0,5 µA. Dieser kompakte Sensor besitzt eine I²C-Schnittstelle, wodurch er sich einfach integrieren lässt. **Bild 3** zeigt ein SHTC3-Modul von Soldered Electronics.
Der **HIH6130** [5] von Honeywell bietet einen guten Kompromiss zwischen Genauigkeit (±0,5 °C für Temperatur und ±3% RH für Feuchte) und Robustheit und verfügt über einen digitalen Ausgang. Sein Strombedarf ist moderat, typischerweise etwa 450 µA während des Betriebs.
Der **MPL3115A2** von NXP Semiconductors [6] ist ein Präzisionshöhensensor, der Druck und Temperatur misst und genaue Höhenberechnungen ermöglicht. Er arbeitet in einem Druckbereich von 20…110 kPa und einem Temperaturbereich von -40…+85°C mit einer Druckgenauigkeit von ±0,4 kPa und einer Temperaturgenauigkeit von ±1,0°C. Der Sensor bietet eine Höhenauflösung

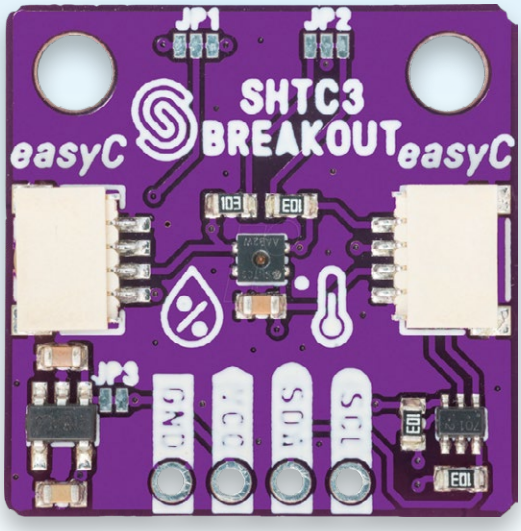


Bild 3. Temperatur- und Feuchtemodul SHTC3. (Quelle: Reichelt Elektronik)

von 30 cm und eine Temperaturauflösung von 0,1°C und unterstützt sowohl I²C- als auch SPI-Schnittstellen zur Kommunikation. Er hat eine geringe Stromaufnahme, typischerweise nur 40 µA im Standardmodus.

Sensoren für Niederschlag, Windgeschwindigkeit und Windrichtung

Um die korrekte Regenmenge zu ermitteln, muss das Gewicht des Regens in einem Behälter bekannter Größe gemessen werden. Hierfür gibt es eine Reihe von Möglichkeiten, aber auch preisgünstige Selbstbaulösungen sind möglich.

Für eine Selbstbaulösung kann ein Behälter mit einem Aufsatz mit einem Lastsensor 3D-gedruckt werden. Der Lastsensor misst das Gewicht des Wassers und durch die Berechnung des Gewichts des Behälters kann das Volumen der Regenmenge ermittelt werden. Schließlich kann die Regenmenge in Milliliter aufgezeichnet werden.

Für die Messung von Windgeschwindigkeit und Windrichtung gibt es verschiedene Lösungen. Es ist empfehlenswert, einen im Handel erhältlichen Bausatz zu kaufen, wie im Bild gezeigt. Es ist aber auch möglich, die Windgeschwindigkeit selbst zu messen. Dazu benötigt man einen 3D-Drucker für die drei Becher, die sich im Wind drehen, und eine Windfahne, die sich nach der Windrichtung ausrichtet. Um die Windgeschwindigkeit zu messen, eignet sich ein magnetischer Encoder. Dazu wird ein Magnet am Ende der Welle mit den drei Bechern befestigt und der magnetische Encoder auf der benachbarten Seite angebracht, um die Drehzahl der Welle zu messen und daraus die Windgeschwindigkeit zu berechnen. Zur Bestimmung der Windrichtung kann ein digitaler Kompasssensor verwendet werden, der einfacher zu implementieren ist als die Widerstandsäquivalentmethode, die in den meisten kommerziellen Optionen verwendet wird.

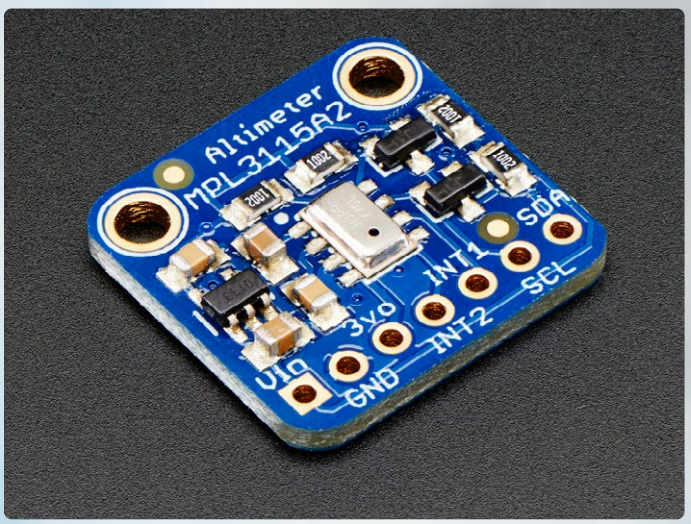


Bild 4. MPL3115A2-Sensormodul von Adafruit. (Quelle: Adafruit)

Vergleichende Analyse

Durch die Bewertung bestimmter herausragender Sensoren auf der Basis wichtiger Leistungsindikatoren kann die optimale Wahl für verschiedene Anwendungsanforderungen getroffen werden:

Messgenauigkeit: Der Sensirion SHT35 ist der genaueste Sensor für Temperatur- und Feuchtemessungen, ideal für kritische Anwendungen, bei denen Präzision entscheidend ist.

Kosteneffizienz: Der Sensirion SHT40 bietet ein ausgezeichnetes Preis-Leistungs-Verhältnis, da er eine angemessene Genauigkeit bei geringeren Kosten bietet. Dies macht ihn besonders attraktiv für kostengünstige Projekte. Der MPL3115A2 von NXP Semiconductors ist ebenfalls eine gute Wahl, da er einen angemessenen Preis und eine gute Genauigkeit bietet. **Bild 4** zeigt das MPL3115A2-Modul von Adafuit.

Stromaufnahme: In Bezug auf die Energieeffizienz ist der Sensirion SHTC3 führend. Mit einer Stromaufnahme von nur 0,5 µA eignet er sich hervorragend für Anwendungen, die eine minimale Stromaufnahme erfordern, zum Beispiel batteriebetriebene Geräte oder ferngesteuerte Messeinheiten.

Jeder Sensor hat seine eigenen Vor- und Nachteile und sollte auf der Grundlage der spezifischen Projektanforderungen, der Umgebungsbedingungen, in denen er eingesetzt werden soll, und der erforderlichen Genauigkeit ausgewählt werden. Bei der Auswahl sollten auch die einfache Integration in bestehende Systeme und die Gesamtkosten des Sensors und der zugehörigen Peripherie berücksichtigt werden.

Für detaillierte Vergleiche ist es unerlässlich, die Datenblätter der einzelnen Sensoren zu lesen. Diese enthalten genaue Angaben zu elektrischen Eigenschaften, Beständigkeit gegen Umwelteinflüsse, Größe, Strombedarf und vieles mehr, so dass eine fundierte Entscheidung auf Basis der projektspezifischen Anforderungen

Tabelle 1: Vergleich von Temperatur- und Feuchtesensoren

Sensor-Bezeichnung	Hersteller	Mess-parameter	Fehler (Temp./ Feuchte/Druck)	Arbeitsbereich (Temp./Feuchte/ Druck)	Strom-aufnahme (gemittelt)	Weitere Eigenschaften
SHT35	Sensirion	Temperatur, Feuchte	±0,1°C / ±1,5% RH	-40 to 125 °C / 0...100% RH	1,5 µA	Hohe Genauigkeit, Werkskalibrierung
DHT22	Aosong Electronics	Temperatur, Feuchte	±0,5°C / ±2,5% RH	-40 to 80 °C / 0...100% RH	~	Budgetfreundlich
HTU21D	TE Connectivity	Temperatur, Feuchte	±0,3°C / ±2% RH	-40 to 125 °C / 0...100% RH	2,7 µA	Schnelle Reaktion, I²C-Schnittstelle
DS18B20	Maxim Integrated	Temperatur	±0,5°C	-55 to 125°C / NA	1 µA	Wasserdichte Modelle verfügbar
MPL3115A2	NXP Semiconductors	Temperatur, Druck	±0,3°C / ±0,4 kPa	-40 to 85°C / 20...110 kPa	2 µA	Höhenmesser-Funktion
LPS22HB	STMicroelectronics	Druck	±0,1 hPa	NA / 260...1260 hPa	3 µA	Ultrakompakter, hochauflösender Sensor
MS5611	TE Connectivity	Druck	±1,5 hPa	NA / 10...1200 hPa	1,2 µA	Hohe Auflösung, geringe Stromaufnahme
SHTC3	Sensirion	Temperatur, Feuchte	±0,2°C / ±2% RH	-40 to 125 °C / 0...100% RH	0,5 µA	I²C-Schnittstelle, 5 s Ansprechzeit

getroffen werden kann. Einen allgemeinen Vergleich dieser Sensoren zeigt **Tabelle 1**.

Multifunktionale Sensoren

Sensoren, die viele Parameter ermitteln können, zum Beispiel die Serien BME280 und BME688 von Bosch Sensortec, bieten aus mehreren Gründen Vorteile. Sie sind sie eine rationelle Lösung für die Umweltüberwachung, da sie Variablen wie Temperatur, Feuchte, Druck und Gaskonzentration in einem einzigen kompakten Bauteil messen.

Der Sensor **BME280** [7] von Bosch Sensortec ist ein weit verbreiteter Sensor zur Messung von Temperatur, Feuchte und Druck. Er erreicht eine Temperaturgenauigkeit von ±1,0 °C, eine Feuchtegenauigkeit von ±3% und eine Druckgenauigkeit von ±1,0 hPa. Er arbeitet in einem Temperaturbereich von -40...+85 °C, einem Feuchtebereich von 0...100% RH, einem Druckbereich von 300...1100 hPa und zeichnet sich durch eine niedrige Stromaufnahme von nur 0,1 µA im Sleep-Modus aus. Aufgrund seiner Präzision und seines mäßigen Stromhungers ist der Sensor ideal für tragbare Wetterstationen und Hausautomatisierungssysteme geeignet.

Der Bosch Sensortec **BME688** [8] erweitert die Fähigkeiten des BME280 um die Detektion von flüchtigen organischen Verbindungen (VOCs) und eignet sich damit für die Überwachung der Luftqualität in Innenräumen. Dieser Sensor hat die gleiche Temperatur- und Feuchtegenauigkeit wie der BME280, bietet aber eine verbesserte Druckgenauigkeit von ±0,6 hPa. Der BME688 verfügt außerdem über Gasdetektions- und KI-Funktionen, die zwar etwas mehr Strom benötigen, aber dennoch einen für batteriebetriebene Geräte geeigneten Wirkungsgrad aufweisen. Der BME688 ist ideal für Innenräume, Sensorknoten und Netzwerkanwendungen.

Die **HPM-Serie** von Honeywell [9] konzentriert sich auf die Detektion von Feinstaub und misst PM2.5 und PM10 mit einer Genauigkeit von ±5 µg/m³ beziehungsweise ±10 µg/m³. Er wurde für die Umwelt- und Gesundheitsüberwachung entwickelt und arbeitet im Bereich von 0...1.000 µg/m³. Dieser Sensor besitzt ein Gebläse, um die Luft für die Probenahme anzusaugen, was zu einer höheren Stromaufnahme von rund 80 mA während des Betriebs führt..

Der kompakte Sensor **SGP40** [10] von Sensirion misst die Luftqualität in Innenräumen, wobei er flüchtige organische Verbindungen (TVOC) und äquivalente CO_2-Werte erfasst. Er bietet eine typische Genauigkeit von 15 ppb für TVOC-Messungen. Der durchschnittliche Strombedarf des SGP40 liegt bei etwa 2,6 mA mit Spitzenströmen von bis zu 3 mA (bei 3,3-V-Versorgung) während der Messung.

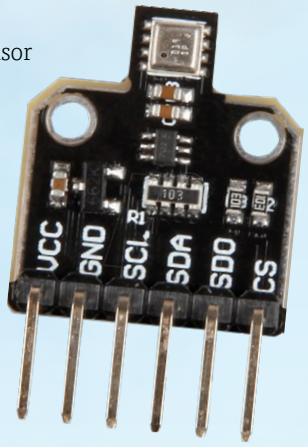


Bild 5. Sensormodul BME680 von Bosch Sensortec. (Quelle: Reichelt Elektronik)

Das **BME680** [11] von Bosch Sensortec führt Messungen von Temperatur, Feuchte, Druck und flüchtigen organischen Verbindungen (VOC) in einem einzigen Bauteil durch. Mit einer Genauigkeit von ±1,0 °C für die Temperatur, ±3 % RH für die Feuchte und ±0,12 hPa für den Druck zeichnet er sich durch seine Präzision aus. Der Sensor verfügt außerdem über einen Gassensor zur Erkennung flüchtiger organischer Verbindungen (VOC), was seine Funktionalität für eine detaillierte Umweltüberwachung erhöht. Der BME680 arbeitet in einem Temperaturbereich von -40...+85°C, einem Feuchtebereich von 0...100% RH und einem Druckbereich von 300...1100 hPa. Er ist energieeffizient, da er im Standardmodus weniger als 0,1 mA benötigt. **Bild 5** zeigt ein BME680-Modul von Joy-IT.

Diese integrierten Sensoren vereinfachen das Design und den Einsatz, sind platzsparend und reduzieren die Komplexität der Schaltungen. Außerdem sind sie in der Regel energie- und kosteneffizienter als separate Sensoren für jede Umgebungsvariable.

Tabelle 2: Vergleich multifunktionaler Sensoren

Sensor-Bezeichnung	Hersteller	Mess-parameter	Fehler (Temp./ Feuchte/Druck)	Betriebsbereich (Temp./Feuchte/ Druck)	Strom-aufnahme (gemittelt)	Besondere Eigenschaften
BME280	Bosch Sensortec	Temp., Feuchte, FDruck	Temp. ±1,0 °C, Feuchte ±3%, Druck ±1,0 hPa	Temp. -40...85 °C, Feuchte 0...100% RH, Druck 300...1100 hPa	<0,1 µA	Stromsparend, tragbare Stationen, Hausautomatisierung
BME688	Bosch Sensortec	Temp., Feuchte, Druck, VOCs	Temp. ±1,0 °C, Feuchte ±3%, Druck ±0,6 hPa	wie BME280, zusätzlich Gasdetektion	3,7 µA	Luftqualität in Innenräumen, VOC-Erkennung, KI-Funktionen
HPM Series	Honeywell	Feinstaub (PM2.5, PM10)	PM2.5 ±5 µg/m³, PM10 ±10 µg/m³	0...1.000 µg/m3	~80 mA	Überwachung der Umwelt und Gesundheit
SGP40	Sensirion	TVOCs, CO_2 -Gehalt (äquivalent)	TVOC ±15 ppb	Temp. -10...50 °C, Feuchte 0...90% RH	2,6 mA	Luftqualität in Innenräumen, kompaktes Design
BME680	Bosch Sensortec	Temp., Feuchte, Druck, VOCs	Temp. ±1,0 °C, Feuchte ±3%, Druck ±0,12 hPa	Temp. -40...85 °C, Feuchte 0...100% RH, Druck 300...1100 hPa	<0,1 mA	Detaillierte Umwelt-überwachung, energieeffizient

Diese Sensoren mit standardisierter Schnittstelle (zum Beispiel Grove-Verbinder), erhöhen die Benutzerfreundlichkeit und ermöglichen eine schnelle Integration in Projekte ohne aufwendige Verkabelung oder technische Einrichtung. Diese Integration ist besonders vorteilhaft in Anwendungen, bei denen Platz und Energie knapp sind, zum Beispiel Wearables, tragbaren Geräten und kompakten Umweltüberwachungsstationen. In **Tabelle 2** werden diese Sensoren miteinander verglichen.

Schlussbemerkungen

Die Auswahl von Sensoren für eine autonome Wetterstation erfordert eine sorgfältige Bewertung der technischen Parameter, der einfachen Integration und der Kosteneffizienz. In diesem Artikel wurden zahlreiche Sensoren gründlich bewertet, wobei der Schwerpunkt auf Sensormodelle gelegt wurde, die leicht erhältlich und kostengünstig sind und zuverlässige Informationen liefern. Der technische Fortschritt wird die Genauigkeit und Nützlichkeit dieser Sensoren wahrscheinlich verbessern und die Möglichkeiten der Wetterüberwachung erweitern. Um genaue und aktuelle Informationen zu erhalten, sollten die neuesten Datenblätter und Veröffentlichungen der Hersteller herangezogen werden.

Diese Strategie stellt sicher, dass Ihre Wetterstation ein zuverlässiges Instrument für die Umweltüberwachung bleibt.

SE — 240002-02

Haben Sie Fragen oder Kommentare?

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

Passende Produkte

- **Wind- und Regensensoren für Wetterstation**
  www.elektor.de/20234
- **Enviro+ Umweltüberwachungsstation**
  www.elektor.de/18975
- **SparkFun Wetter--Shield**
  www.elektor.de/19089

WEBLINKS

[1] Mathias Claussen, „Open-Network-Wetterstation Mark 2", Elektor 5/2020: https://www.elektormagazine.de/magazine/elektor-147/58575
[2] Sensirion SHT3X-DIS-F | Datenblatt: https://sensirion.com/resource/datasheet/sht3x-d
[3] Aosong Electronic AHT20 | Datenblatt: http://www.aosong.com/en/products-32.html
[4] Sensirion SHTC3 | Datenblatt: https://sensirion.com/resource/datasheet/shtc3
[5] Honeywell HIH6130 | Datenblatt: https://eu.mouser.com/datasheet/2/187/HWSC_S_A0012940693_1-3073215.pdf
[6] NXP MPL3115A2S | Datenblatt: https://www.nxp.com/docs/en/data-sheet/MPL3115A2S.pdf
[7] Bosch BME 280 | Datenblatt: https://tinyurl.com/BME-280-Datasheet
[8] Bosch BME 688 | Datenblatt: https://tinyurl.com/BME-688-Datasheet
[9] Honeywell HPM-Reihe: https://tinyurl.com/hpm-series
[10] Sensirion SGP40 | Datenblatt: https://sensirion.com/resource/datasheet/sgp40
[11] Bosch BME680 | Datenblatt: https://tinyurl.com/BME-680-Datasheet

Microchip is...

IoT

Sensor Interface ‹
Microcontrollers ‹
Microprocessors ‹
Security ‹
Connectivity ‹
Cloud Services ‹

- Automotive
- Home Appliance
- Lighting
- Medical
- Smart Energy/Metering
- Wireless Audio

MICROCHIP

microchip.com/IoT

MEC2481A-UK-02-23

KI-gestützter Wasserzähler

Quelle: Adobe Stock

Teil 1: Bringen Sie Ihren alten Zähler ins IoT!

Von Daniel Scaini (Italien)

Elettronica In
WWW.ELETTRONICAIN.IT

Intelligente Wasserzähler sind bereits seit einiger Zeit auf dem Markt, aber der Austausch der alten Zähler ist aus technischen oder bürokratischen Gründen oft nicht so einfach. Mit diesem Projekt kann ein beliebiger analoger Zähler mit Hilfe einer ESP32-CAM-Plattform und Künstlicher Intelligenz in einen digitalen Zähler verwandelt werden. Da wir in diesem Artikel auch viele Hintergründe enthüllen werden, haben wir ihn in zwei Teile aufgeteilt.

Italien ist eines der europäischen Länder, die enorm viel Wasser für den zivilen Gebrauch entnehmen und verbrauchen, gleich nach Griechenland. Dies ist besorgniserregend, vor allem wenn man die Wasserkrisen bedenkt, die unser Land immer wieder heimsuchen. Die durchschnittliche Verfügbarkeit von Wasser ist in den letzten drei Jahren um etwa 19 % zurückgegangen, was durch die immer höheren Temperaturen und den Mangel an Niederschlag noch verschlimmert wurde. Dieses Problem ist nicht nur in Italien anzutreffen, und in etlichen Ländern werden drastische Gegenmaßnahmen ergriffen wie die Einschränkung des Verbrauchs oder die Bereitstellung von Mitteln zur Renovierung privater oder staatlicher Verteilungsanlagen, um Lecks zu eliminieren.

In Anbetracht der ständigen technologischen Entwicklung wird seit einigen Jahren auch versucht, das Ablesen von Zählern für Verbraucher effizienter zu gestalten und die Haus-zu-Haus-Ablesung durch Mitarbeitern der Wasserwerke zu vermeiden. Solche Maßnahmen sollen im Endeffekt ebenfalls dazu beitragen, die Verschwendung der Ressource Trinkwasser zu reduzieren.

In diesem Artikel werden wir uns damit befassen: einen analogen Zähler digital ablesbar zu machen, so dass die Werte an uns oder einen Server übermittelt werden können. Das Ganze basiert auf einem ESP32-CAM, einer Plattform, die vielen bereits bekannt ist, und einem System mit Künstlicher Intelligenz (KI), das in der Lage ist, Bilder zu erkennen, auszuwerten und in entsprechende Messwerte zu übersetzen.

Architektur

KI-Systeme haben in großem Stil Einzug in unser tägliches Leben gehalten - man denke nur an Sprachassistenten oder Bilderkennung. Für die komplexen Berechnungen, die ein KI-System zu bewältigen hat, kann man auf spezielle Online-Plattformen im Cloud-Computing zurückgreifen oder sie direkt auf dem Chip, im so genannten Edge-Computing durchführen. Mit der zunehmenden Verbesserung der Prozessoren wird diese zweite Art der Verarbeitung immer häufiger angewendet und bildet deshalb auch die Grundlage für unser Projekt. Dabei werden ein KI-Netzwerk und ein ESP32-CAM zusammenarbeiten, um dem Nutzer ein digital nutzbares Ergebnis zu liefern, das durch digitales Fotografieren eines klassischen analogen Wasserzählers gewonnen wird. Die Erkennung und Digitalisierung wird von der ESP32-CAM-Plattform mit Hilfe eines neuronalen Faltungsnetzwerks (Convolutional Neural Network, CNN oder ConvNet) durchgeführt, auf das wir später in diesem Artikel eingehen werden.

Der erste Schritt bei der Umsetzung unseres Projekts ist die Installation der Firmware auf unserem ESP32-CAM. Als nächstes brauchen wir eine Kalibrierungsphase, in der wir die Bereiche identifizieren, die für die Erkennung der Zahlen und weiteren Anzeigen auf unserem Zähler vorgesehen sind. Sobald dies abgeschlossen ist, haben wir alle Informationen zur Verfügung, um sie digital an den gewünschten Ort zu senden.

Neuronale Netze

Neuronale Netze sind Berechnungsmodelle, die sich an der Funktionsweise des menschlichen Gehirns orientieren. Sie sind Systeme der künstlichen Intelligenz, die versuchen, die Art und Weise, wie das Gehirn Informationen verarbeitet, zu imitieren. Neuronale Netze werden im Bereich des Maschinellen Lernens eingesetzt, in dem es darum geht, Computer so zu trainieren, dass sie bestimmte Aufgaben ausführen, ohne dass sie explizit für diesen Zweck programmiert wurden.

Neuronale Netze spielen eine entscheidende Rolle bei Deep-Learning-Modellen, einem Zweig des Maschinellen Lernens, der auf die

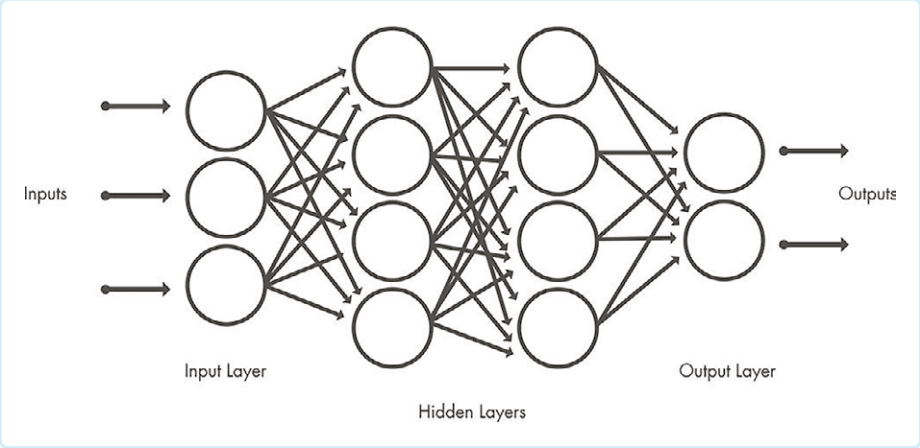


Bild 1. Neuronale Netze bestehen aus einer Eingabeschicht, einer Ausgabeschicht und einigen „verborgenen" Zwischenschichten. (Quelle für alle Bilder in diesem Artikel: Elettronica In - https://futuranet.it)

Verarbeitung komplexer Daten abzielt. Sie bestehen aus Berechnungseinheiten, die als künstliche Neuronen oder Knoten bezeichnet werden. Diese Neuronen sind durch künstliche Verbindungen, sogenannte Gewichte oder Gewichtungen, miteinander verbunden. Jeder Verbindung ist ein numerischer Wert zugeordnet, der die Signifikanz der Verbindung für das Modell angibt.

Neuronale Netze sind in Schichten organisiert, wobei zwischen der Eingabeschicht und der Ausgabeschicht eine oder mehrere Schichten verborgen sind. Die erste Schicht, die so genannte Eingabeschicht, empfängt die Eingabedaten. Die mittleren, so genannten verborgenen Schichten verarbeiten die Informationen durch ihre Neuronen. Die letzte Schicht, die so genannte Ausgabeschicht, erzeugt schließlich die gewünschten Ergebnisse, wie in **Bild 1** schematisch dargestellt.

Neuronale Netze können in einem breiten Spektrum von Anwendungen eingesetzt werden, zum Beispiel bei der maschinellen Übersetzung, der Verarbeitung natürlicher Sprache, der medizinischen Diagnose und in unserem Fall der Bilderkennung. In diesem speziellen Fall werden während des Trainingsprozesses Filter mit unterschiedlichen Auflösungen auf das Bild angewendet, und die Ausgabe jedes verarbeiteten Bildes wird als Input für die nächste Schicht verwendet. Die Filter beginnen mit grundlegenden Merkmalen wie Helligkeit oder Kantenerkennung und werden im weiteren Verlauf immer komplexer und feiner, bis hin zu den Merkmalen, die das Objekt eindeutig definieren.

CNNs sind eine spezielle Art von neuronalen Netzen, die für die effiziente Verarbeitung strukturierter Daten wie Bilder oder Videos entwickelt wurden. Was CNNs von herkömmlichen neuronalen Netzen unterscheidet, ist die Verwendung einer Operation namens Faltung. Bei der Faltung wird ein kleines Fenster, ein so genanntes Filter oder Kernel, über das Eingangsbild geschoben und eine Reihe von mathematischen Operationen auf jeden Teil des Bildes angewendet. Dieser Prozess ermöglicht es dem CNN, automatisch und effizient relevante Merkmale wie Kanten, Texturen oder Muster aus den Bildern zu extrahieren.

Auch diese Netze bestehen aus einer Eingabeschicht, einer Ausgabeschicht und vielen verborgenen Zwischenschichten. Sie enthalten drei Haupttypen von Schichten oder Layern, nämlich:

- Faltungsschicht
- Pooling-Layer
- Vollverknüpfte Schicht (fully-connected oder FC-Layer)

Die Faltungsschicht ist die erste Schicht eines neuronalen Faltungsnetzes. Sie ist für die Extraktion der hervorstechenden Merkmale aus dem Eingangsbild verantwortlich. Auf Faltungsschichten können weitere Faltungsschichten oder Pooling-Schichten folgen. Nachfolgende Faltungsschichten verarbeiten die von den vorherigen Schichten extrahierten Merkmale weiter, so dass das Netz immer komplexere und abstraktere Merkmale lernen kann.

Die voll verknüpfte Schicht, auch Ausgabeschicht genannt, ist die letzte Schicht eines CNN. Auf dieser Ebene werden die extrahierten Merkmale verwendet, um Vorhersagen oder Klassifizierungen zu treffen. Diese Ebene ist für die Bereitstellung der endgültigen Ausgabe des neuronalen Faltungsnetzwerks verantwortlich.

Auf jeder Ebene nimmt die Komplexität des CNN zu, da das Netz immer ausgefeiltere und abstraktere Merkmale des Eingangsbildes erlernt. Außerdem nimmt mit dem Durchlauf der Faltungsebenen die Bedeutung des Teils des Bildes zu, der vom neuronalen Netz identifiziert und detailliert analysiert wird, wie in **Bild 2** dargestellt. Im Gegensatz zu einem herkömmlichen neuronalen Netz verfügt ein CNN über gemeinsame Gewichte, die für alle in einer bestimmten Schicht versteckten Neuronen identisch sind. Nachdem die Merkmale in mehreren Schichten gelernt wurden, geht die Architektur eines CNN zur Klassifizierung über.

Die vorletzte Schicht ist eine vollständig verknüpfte Schicht, die einen Vektor der Größe K erzeugt (wobei K die Anzahl der vorhersagbaren Klassen ist) und die Wahrscheinlichkeiten für jede Klasse eines

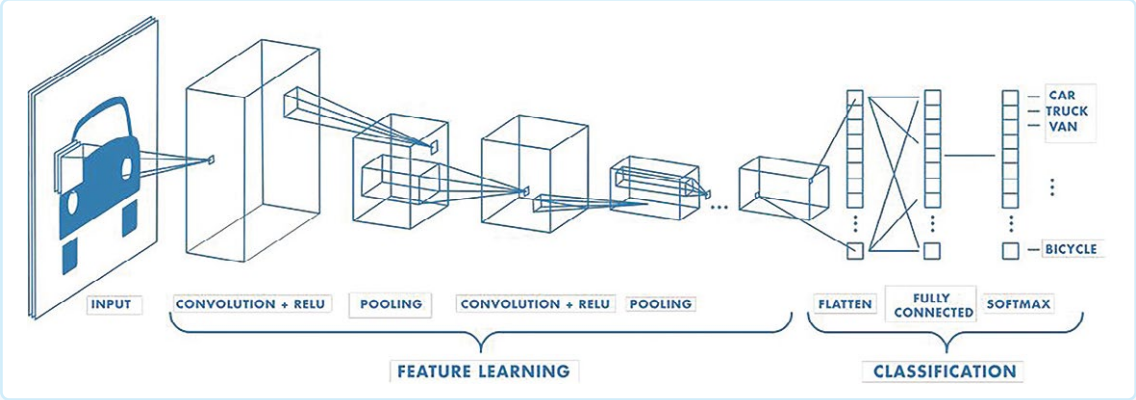


Bild 2. Mit jeder Schicht nimmt die Komplexität des CNN zu.

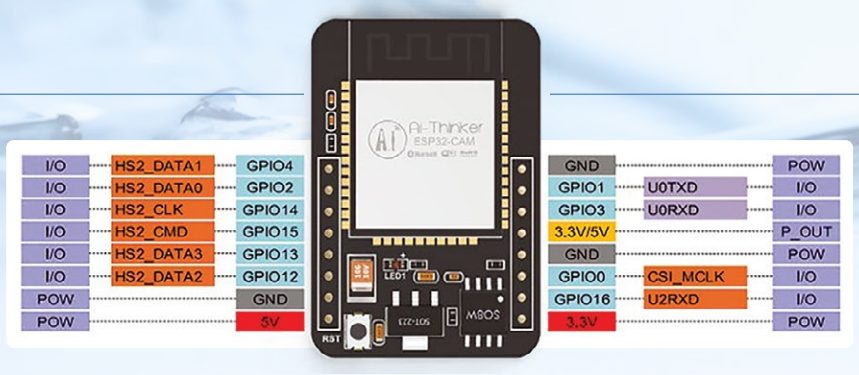


Bild 3. Pinbelegung des ESP32-CAM-Moduls.

Bild 4. Die Plattform ESP32-CAM ist das Herzstück dieses Projekts.

klassifizierten Bildes enthält. Die letzte Schicht der CNN-Architektur verwendet eine Klassifizierungsschicht, um die Ausgabe der endgültigen Klassifizierung zu liefern. Normalerweise gibt es bereits trainierte Modelle dieser Netze, die mehrere Megabyte oder sogar Gigabyte groß sind. In unserem Fall handelt es sich nur um ein paar Megabyte, die dennoch auf einer SD-Karte gespeichert werden, um den internen Speicher des Chips nicht zu belasten.

Konkret gibt es im Konfigurationsordner mehrere Dateien mit der Endung *.tflite*, was für TensorFlow Lite steht. TensorFlow, das im Jahr 2015 aus Google Brain hervorging, hat sich zu einer Referenzbibliothek für die Erstellung von Deep-Learning-Modellen entwickelt. Ursprünglich wurde TensorFlow für den internen Gebrauch bei Google entwickelt, dann wurde es als Open Source veröffentlicht und gewann deshalb schnell an Popularität in der Community für maschinelles Lernen. TensorFlow liefert eine breite Palette von Modellen und Algorithmen für maschinelles Lernen und Deep Learning, die als neuronale Netze bekannt sind, und stellt sie Entwicklern über eine intuitive API zur Verfügung.

TensorFlow nutzt die Leistungsfähigkeit von Python oder JavaScript, um eine einfach zu bedienende Programmierschnittstelle für die Erstellung von Anwendungen bereitzustellen. Die Ausführung solcher Anwendungen findet in C++ statt, was eine hohe Rechenleistung ermöglicht. Dies macht TensorFlow zu einer vielseitigen Wahl für groß angelegte maschinelle Lernprojekte. Modelle, die mit TensorFlow trainiert wurden, können auch auf mobilen oder Edge-Computing-Geräten, wie in unserem Fall, sowie auf iOS- oder Android-Betriebssystemen implementiert werden. Das TensorFlow-Ökosystem bietet Werkzeuge wie TensorFlow Lite an, die TensorFlow-Modelle für die effiziente Ausführung auf solchen Geräten optimieren. Mit TensorFlow Lite kann ein Kompromiss zwischen der Größe des Modells und seiner Genauigkeit geschlossen werden. Ein kleineres Modell kann weniger Speicherplatz beanspruchen, zum Beispiel 12 MB statt 25 MB oder vielleicht auch etwas mehr als 100 MB, aber es kann einen (leichten) Verlust an Genauigkeit erleiden. Dieser Genauigkeitsverlust ist jedoch meist vernachlässigbar, wenn man die Geschwindigkeits- und Energieeffizienzvorteile dagegenhält, die das komprimierte Modell bietet.

In unserem Fall wird dieses mit TensorFlow trainierte Modell verwendet, um Zahlen innerhalb der Zähleranzeige zu unterscheiden und zu erkennen und auch um den Zustand von Indikator-Anzeigen zu bestimmen. Mit Hilfe von TensorFlow sind wir in der Lage, die Leistung des Deep Learning für die Datenanalyse und die künstliche Intelligenz zu nutzen und die Effizienz und Genauigkeit unserer Anwendungen zu verbessern.

Hardware

Die gewählte Plattform ESP32-CAM von AI-Thinker ist nicht zuletzt ihrer kompakten Größe (40,5×27×4,5 mm) wegen ideal für den Einsatz in der von uns geplanten Anwendung. Das Board besteht aus einem ESP32-Modul (Nachfolger des berühmten ESP8266, auf den wir später noch eingehen werden) mit einem Anschluss - dessen Pinbelegung in **Bild 3** zu sehen ist - zur Aufnahme eines separaten Kameramoduls und einem Steckplatz für eine SD-Karte mit einer Größe bis zu 4 GB. Genau betrachtet handelt es sich beim ESP32 um einen programmierbaren Mikrocontroller mit integrierter WLAN- und Bluetooth-Konnektivität sowie einem optionalen externen RAM von bis zu 4 MB.

Darüber hinaus kann in den Kameraanschluss entweder ein OV2640- oder ein OV7670-Modul eingesteckt werden. Erstere Kamera gehört zum Lieferumfang des Moduls und bietet eine Auflösung von 2 Megapixeln. Die SPI-Geschwindigkeit beträgt 8 MHz und die Größe des Bildspeichers 384 KB. Die normale Stromaufnahme liegt bei 70 mA, im Energiesparmodus bei nur 20 mA. Aus diesem Grund ist das Kameramodul zu empfehlen.

Auf der Platine befindet sich auch eine sehr helle LED (**Bild 4**), die als Blitzlicht oder zur Beleuchtung der zu fotografierenden Bereichs verwendet werden kann. Diese Funktion ist für unser Projekt von entscheidender Bedeutung, da das Gerät in Umgebungen eingesetzt werden soll, die normalerweise sehr dunkel sind. Dieses Licht kann moduliert werden, um das gewünschte Ergebnis zu erzielen. Sollte das nicht ausreichen, können wir bei sehr schlechten Lichtverhältnissen dank der zahlreichen freien GPIOs auf der Platine eventuell eine externe Beleuchtung anbringen.

Für kleine Anwendungen wie unserer erweist sich dieser Chip als sehr robust und rechenstark, da er sich auf zwei 32-Bit-Kerne mit einem Takt von 120 MHz stützt. Dies ermöglicht eine ziemlich hohe Bildrate, die natürlich auch vom Format und der Größe abhängt: Als Richtwert können wir einen Durchsatz von bis zu acht JPEGs in SVGA-Qualität (800×600 Pixel) pro Sekunde erreichen.

Die Programmierung des ESP32-CAM-Moduls kann auf verschiedene Arten erfolgen: Die klassische ist die Verwendung eines Adapters von FTDI (**Bild 5**), der dank der FT232RL-Schnittstelle die RS232- und COM-Ports simuliert und somit eine schnelle Plug-and-Play-Funktionalität ermöglicht.

Das im Kasten **Passende Produkte** am Ende dieses Artikels vorgeschlagene Modul ist vollständig pinkompatibel und praktischerweise im Elektor Store erhältlich. Es bindet einen Micro-USB-Anschluss mit der entsprechenden seriellen Schnittstelle in den Chipsatz ein, so dass keine externen seriellen Adapter erforderlich sind.

Falls Sie den oben genannten Adapter nicht zur Verfügung haben, brauchen Sie nicht zu verzweifeln. Sie können dieses Modul nämlich mit jedem anderen Modul programmieren, das über TX- und RX-Pins verfügt, zum Beispiel von verschiedenen Arduino- oder ESP-Modellen.

Verdrahtung zur Programmierung des ESP32-CAM

Aufgrund dieser verschiedenen Möglichkeiten gibt es auch verschiedene Arten der Verkabelung, die sich zwar ähneln, aber

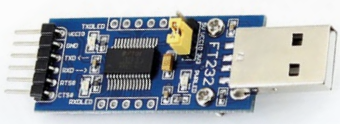


Bild 5. Der USB-TTL-Wandler mit dem FT232RL.

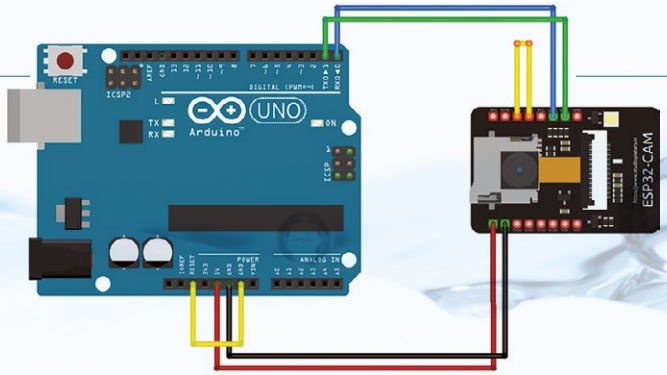


Bild 6. Das ESP32-CAM-Modul kann mit dem Arduino UNO programmiert werden.

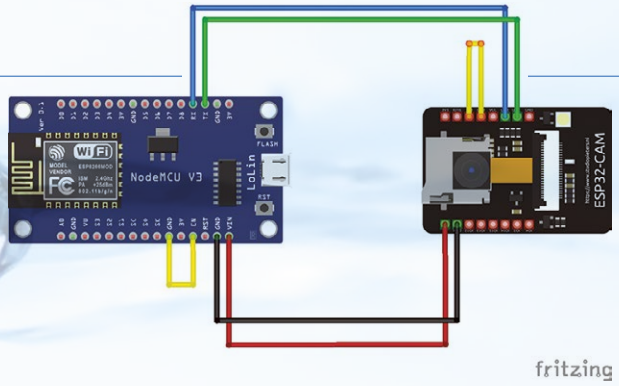


Bild 7. Verdrahtung zur Programmierung des ESP32-CAM mit einem ESP8266.

Tabelle 1. Verdrahtung des Arduino UNO mit dem ESP32-CAM

Arduino ⇒ ESP32-CAM	Arduino	ESP32-CAM
TX ⇒ U0TXD	Reset ⇒ GND	GPIO0 ⇒ GND
RX ⇒ U0RXD		
5V ⇒ 5V		
GND ⇒ GND		

Tabelle 2: Verdrahtung des ESP8266 mit dem ESP32-CAM

ESP8266 ⇒ ESP32-CAM	ESP8266	ESP32-CAM
TX ⇒ U0TXD	EN ⇒ GND	RX ⇒ U0RXD
5V ⇒ 5V		
GND ⇒ GND		

sich auch in einigen Punkten unterscheiden. Normalerweise verfügt ein ESP32-CAM-Modul über keinen USB-Anschluss, um direkt mit einem PC verbunden zu werden, und genau deshalb müssen wir ein externes Modul verwenden.

Beginnen wir mit dem klassischsten aller Module, dem Arduino UNO. In einem ersten Schritt identifizieren wir die Sende- und Empfangs-Pins auf unserem ESP32-CAM, die U0TXD beziehungsweise U0RXD heißen. Diese müssen direkt mit den TX- und RX-Pins an unserem Arduino verbunden werden. Am Kameramodul entsprechen diese Pins auch den Pins GPIO1 und GPIO3. Als nächstes verbinden wir den IO0-Kontakt des ESP32-CAM mit dem GND-Kontakt, um den richtigen Kommunikationsmodus zu wählen. Für die Stromversorgung verbinden wir die korrespondierenden 5V- und GND-Pins zwischen den beiden Plattformen.

Um den Arduino bei Bedarf zurücksetzen zu können, schließen wir den RESET-Pin mit dem GND-Pin kurz. Zusammenfassend ergibt dies den Verdrahtungsplan aus **Bild 6** oder – wenn Sie es in Wort lieber als in Bild haben – in **Tabelle 1**.

Wenn Sie keinen Arduino UNO, sondern einen ESP8266 haben, ändert sich die Verdrahtung nur sehr wenig. Die einzige erwähnenswerte Änderung betrifft die Kontakte, die auf unserem Programmiermodul verbunden werden müssen: Es sind nicht mehr RESET und GND, sondern EN und GND. Die Verdrahtung ist in **Bild 7** dargestellt und auch in **Tabelle 2** angegeben.

Zum Schluss noch die einfachste Verdrahtung von allen. Wenn wir ein USB-TTL-Wandlermodul auf der Basis des FTDI-Chips besitzen, ist das Ganze sogar mit einer Verbindung weniger gelöst. Auf der Seite des Programmierers müssen wir nämlich nichts mehr verbinden, sondern nur noch die TX- und RX-Kontakte miteinander vertauschen, wie in **Bild 8** und in **Tabelle 3** gezeigt.

Die beschriebenen Verbindungen werden nur während der Programmierphase des ESP32-CAM-Moduls benötigt. Sobald die Programmierung abgeschlossen ist, versorgen Sie das ESP32-CAM-Modul einfach über die 5V- und GND-Leitungen mit Strom. Es ist dann nicht mehr notwendig, das Modul direkt an den Arduino anzuschließen oder eine andere Verkabelung vorzunehmen.

Im zweiten und letzten Teil dieses Artikels werden wir erfahren, wie die Firmware installiert, das Kameraobjektiv korrekt für eine optimale Fokussierung eingestellt und das Lesegerät richtig auf den Wasserzähler ausgerichtet wird. Natürlich wird auch der gesamte KI-basierte Prozess für die korrekte Erkennung und Auslesung der Elemente des Zählers behandelt. Freuen sie sich also auf die Fortsetzung des Artikels in der Septemberausgabe von Elektor!

RG — 240213-02

Tabelle 3: Verdrahtung FTDI zum ESP32-CAM

FTDI	ESP32-CAM
RX	U0TXD
TX	U0RXD
5V	5V
GND	GND

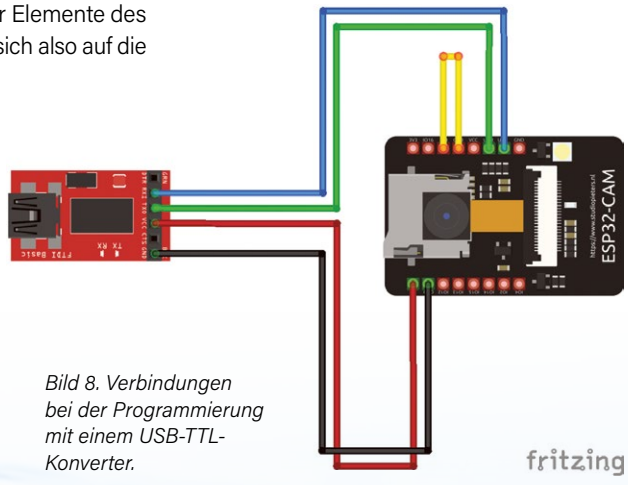


Bild 8. Verbindungen bei der Programmierung mit einem USB-TTL-Konverter.

Haben Sie Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare 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

Ein GSM-Alarm

Nutzung der GSM-Technologie für die Fernüberwachung von Garagen

Von Pascal Rondane (Frankreich)

Der Artikel beschreibt ein kostengünstiges Sicherheits- und Feueralarmsystem mit geringem Strombedarf für einen gemieteten Lagerraum ohne elektrischen Anschluss. Das System nutzt einen GSM-Sender für die Alarmierung und erreicht ohne Aufladen der Batterie eine Betriebsdauer von etwa zwei Jahren.

Bild 1. Das fertig aufgebaute Projekt.

Ich wurde zu diesem Projekt veranlasst, nachdem ich für meine Motorräder und Fahrräder eine Lagerbox gemietet hatte. Der Ort befindet sich einige Kilometer von meinem Haus entfernt, im Keller eines Gebäudes ohne Stromanschluss. Ich wollte aber meine Garage gegen Diebstahl und Feuer schützen, also brauchte ich ein System, das einen Alarm auslöst, wenn die Tür geöffnet wird oder mein Feuermelder anschlägt. Das fertige Projekt ist in **Bild 1** dargestellt. Mein System läuft seit fast zwei Jahren, ohne dass ich die Batterie (12 V, 7 Ah) hätte nachladen müssen. An dieser Stelle und ganz zuvorderst möchte mich bei Vincent Ruggieri für seine unschätzbare Hilfe bei der Verwirklichung dieses Projekts bedanken.

Aufbau des Systems

Die Alarme werden über einen GSM-Sender mit einer bestückten SIM-Karte gesendet. Natürlich braucht man dafür ein Abonnement, aber in Frankreich ist einer der Anbieter bekannt dafür, dass er einen Tarif für nur 2 € pro Monat anbietet. Das ist für dieses Projekt sehr gut geeignet. Das System musste eine sehr niedrige Stromaufnahme aufweisen, da der Einbau eines Solarpanels in diesem Keller nicht möglich war.

Ich habe versucht, ein fertiges Low-Power-GSM-Modem zu finden, da ich nicht die Zeit für die Hardware- und Software-Entwicklung einer vollständig benutzerdefinierten Lösung hatte. Das Ziel war es deshalb, ein einigermaßen erschwingliches Gerät zu finden - in diesem Fall unter 50 €. Ich entschied mich für das GSM-Modem GL90 Plus, das bei verschiedenen Anbietern im Netz leicht zu beschaffen ist. Es bietet die folgenden Funktionen:

- Versand von Textnachrichten und Anrufen im Falle eines Alarms
- Möglichkeit, regelmäßig Textnachrichten zu senden, um den ordnungsgemäßen Betrieb des GSM-Senders sicherzustellen
- Konfigurierbare Batteriestatusmessung, die das Versenden eines SMS-Alarms bei niedrigem Batteriestand ermöglicht
- Eingangsspannung von 5 V bis 16 V
- Sehr niedrige Standby-Stromaufnahme von weniger als 4 mA
- Überwachung von acht konfigurierbaren NO- oder NC-Eingängen
- Konfiguration per SMS, so dass keine PC-Verbindung erforderlich ist
- Version mit einem Ausgang für den Anschluss einer Sirene ist verfügbar
- GSM-Antenne und Koaxialkabel

Aufbau des Alarms

Die Eingänge des GL09 sind über einen zweireihigen Header (2×5 Kontakte im 2,54 mm Raster) zugänglich. Dieser Anschluss befindet sich an der Seite des Geräts und ist für einen aufgepressten Flachbandkabel-Verbinder vorgesehen. Diese Eingänge sind für den

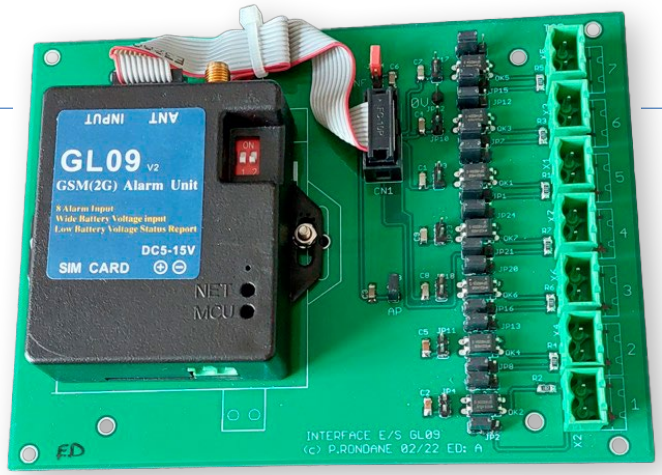


Bild 2. Ein genauerer Blick auf die bestückte Adapterplatine.

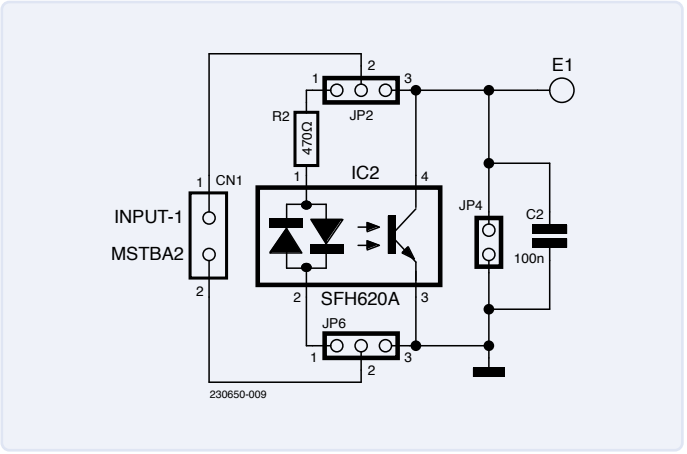


Bild 3. Ein Teil der Optokopplerschaltungen.

Anschluss an Schalter nach Masse gedacht und vertragen keine Eingangsspannung (das heißt, die Elektronik würde dann leicht zerstört). Um jegliches diesbezügliche Risiko zu vermeiden, habe ich eine Adapterplatine entwickelt (**Bild 2**), auf die das GL09-Modul aufgeschraubt werden kann.

Die Adapterplatine ermöglicht es, dass die Eingänge des GL09 sowohl mit passiven, spannungsfreien Schaltern (auch Trockenkontakte genannt) oder mit externen Sensoren, die Spannungsausgänge haben (zum Beispiel 6...12 V für ein logisches High und 0 V für ein logisches Low) belegt werden können. Dies wird durch Optokoppler erreicht, die mit Jumpern aktiviert beziehungsweise deaktiviert werden können. Die Schalter oder Sensoren, die die Alarme auslösen, werden über steckbare Klemmleisten am Rand der Platine angeschlossen. Jeder der sieben Optokoppler kann mit zwei Jumpern auf dreipoligen Stiftleisten so konfiguriert werden, dass entweder ein potentialfreier Kontakt oder ein Spannungssignal angeschlossen werden kann. **Bild 3** zeigt die Schaltung eines einzelnen Optokoppler-Eingangs.

Wenn Sie einen potentialfreien Kontakt anschließen möchten, setzen Sie die Jumper auf Pin 2 und Pin 3 von JP2 und JP6, bei einem Spannungseingang (zwischen 6 V und 12 V) setzen Sie die Jumper jeweils auf Pin 1 und Pin 2. Wenn Sie ausschließlich passive Schalter verwenden wollen, brauchen Sie weder den Optokoppler noch den Strombegrenzungswiderstand R2 zu bestücken. Ich empfehle dennoch, die Stiftleiste für JP4 zu bestücken, weil sie bei Tests nützlich ist, um Alarme während der Konfiguration des GL09-Moduls zu simulieren. Der vollständige Schaltplan in **Bild 4** (nächste Seite) zeigt sieben identische Optokopplerblöcke sowie zwei weitere Jumper, zwei Kondensatoren und die Steckverbinder.

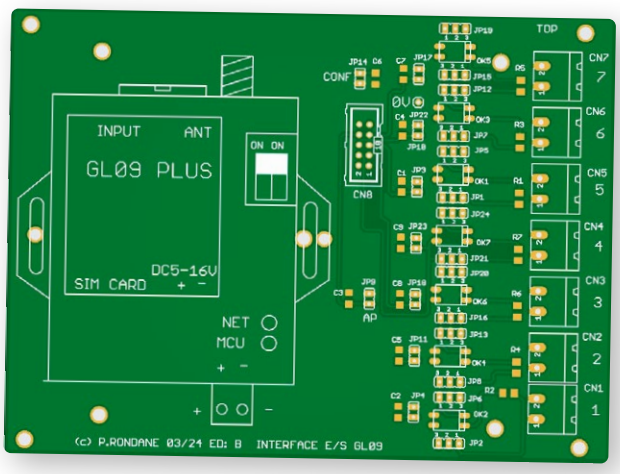


Bild 5. Draufsicht auf die Adapterplatine.

Mit dem Jumper JP14 schaltet man das Modul in den Programmiermodus; an JP9 kann ein Manipulationsschutz angeschlossen werden.

Die Platine in **Bild 5** ist recht einfach gestaltet. Alle SMD-Bauteile stecken in gut handhabbaren 1206-Gehäusen; Klemmen, Verbinder und Pinheader sind Durchsteck-Bauteile. Ich habe die Platine mit Eagle geroutet, und Sie können die Gerber-Datei unter [1] finden. Die gesamte Baugruppe kann in einem spritzwasserdichten Elektro-Anschlussgehäuse montiert werden.

Einrichten des Senders

Die mit dem GL09-Modul gelieferte Anleitung ist ausführlich genug, um den Sender schnell in Betrieb nehmen zu können. Wird ein Alarmeingang ausgelöst, kann eine Alarm-Textnachricht an bis zu sechs vordefinierte Mobiltelefone gesendet werden. Die Alarm-Textnachricht kann mit eigenen Worten (zum Beispiel „Garagentor offen") und die Rückkehr zum Normalzustand mit einer anderen Nachricht wie „Garagentor geschlossen" versehen werden. Es können drei verschiedene Alarmmodi ausgewählt werden: Telefonanruf, Textnachricht oder beides. Beachten Sie, dass der Sender während eines Telefonanrufs keine Sprachnachricht abspielt und die Verbindung trennt, sobald der Anruf entgegengenommen wird.

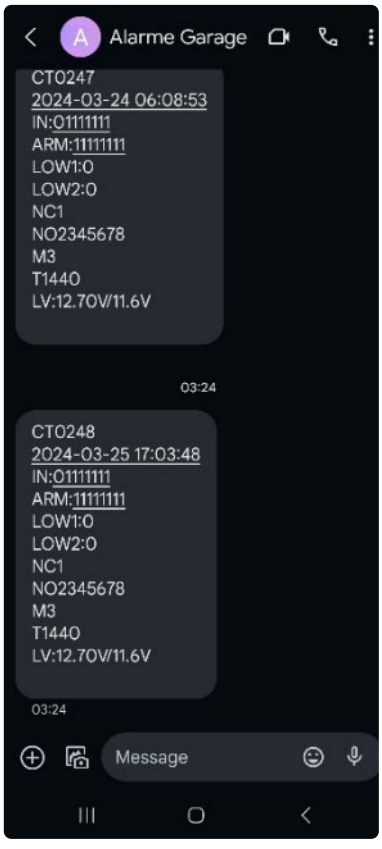


Bild 6. Beispiele für empfangene SMS-Nachrichten. T ist die Periode der Statusmeldungen (hier bedeutet T1440 1440 Minuten oder 24 Stunden), M ist der Arbeitsmodus (hier bedeutet M3, dass ein Alarm sowohl eine SMS als auch einen Telefonanruf auslöst).

Die Versorgungsspannung kann gemessen werden, und wenn sie unter den eingestellten Grenzwert fällt (bei einer 12-V-Batterie liegt der Grenzwert bei 11,6 V), sendet der Sender

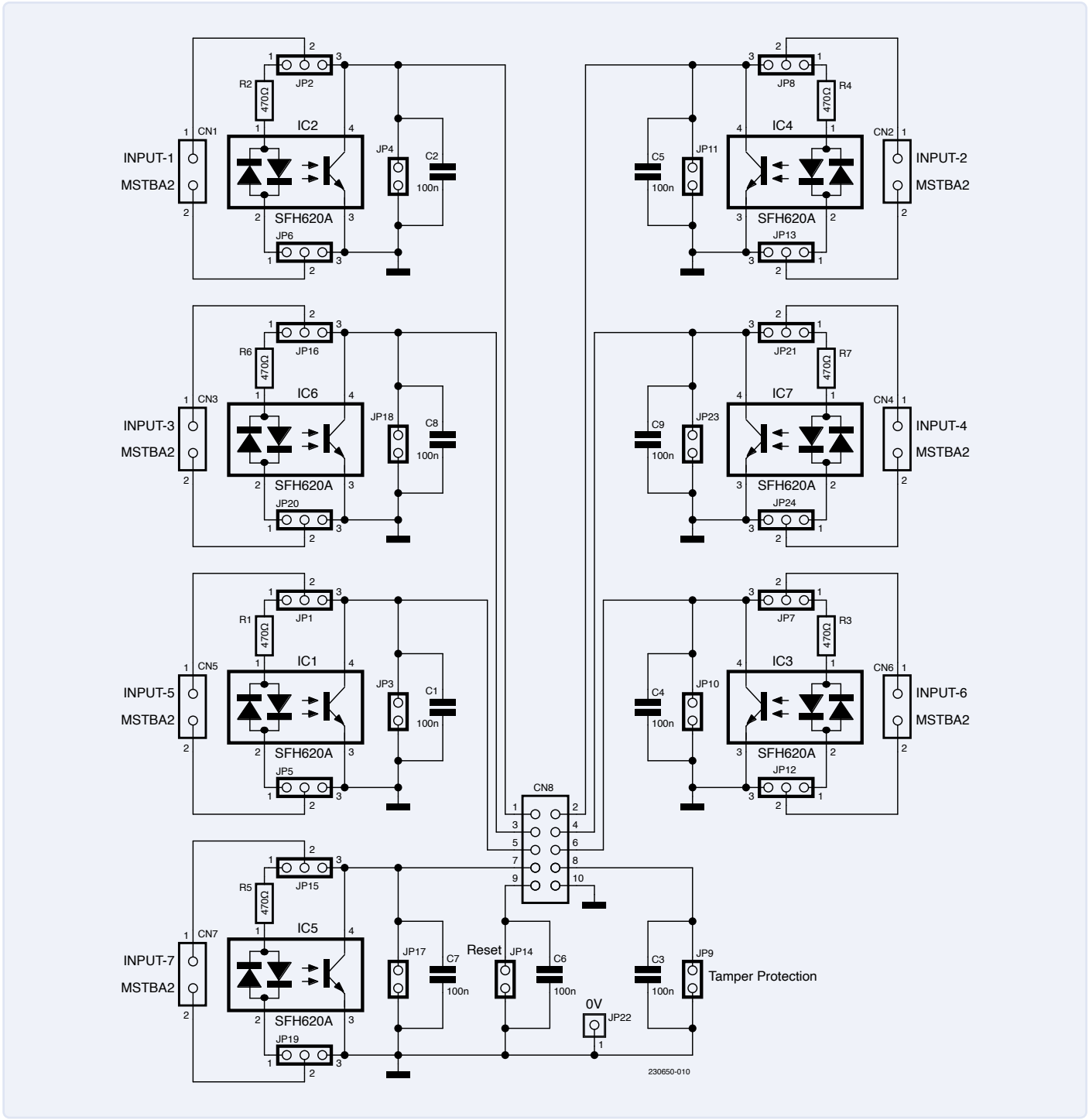


Bild 4. Schaltplan des Projekts mit allen Optokopplern.

einen Alarmtext. Ich habe den Sender so eingestellt, dass er einmal pro Tag eine SMS sendet, um anzuzeigen, dass er ordnungsgemäß funktioniert. Eine dieser täglichen Statusmeldungen per SMS ist in **Bild 6** zu sehen. Sie enthält Datum und Uhrzeit, den aktuellen Zustand der Eingänge, die Bitmaske für die „scharfen" Eingänge sowie die aktuelle Batteriespannung und die programmierte Unterspannungsschwelle. Um die Lebensdauer der Batterie zu verlängern, muss der energiesparende Betriebsmodus des Moduls gewählt werden. In diesem Modus beträgt die Stromaufnahme nach Angaben des Herstellers bei 12 V nur etwa 30 µA.

Feurio!

Ich habe meine Garage mit einem dieser Standard-Rauchmelder ausgestattet, die eine 9-V-Batterie benötigen und in vielen europäischen Ländern in Häusern vorgeschrieben sind. Ich habe das Gerät so modifiziert, dass es das Signal abruft, wenn ein Feuer erkannt wird. Hierfür gibt es zwei Lösungen. Die erste besteht darin, zu erkennen, wann die eingebaute LED des Melders eingeschaltet ist, indem man einen Draht an +9 V von der Batterie und einen anderen Draht an einen der Vorwiderstandsklemmen der LED anschließt. Dieses Signal kann zur Ansteuerung der LED in einem der Optokoppler auf der Platine verwendet werden. Die zweite Möglichkeit steht zur Verfügung, wenn Ihr Brandmelder das IC RE46C181 [2] enthält, was sehr häufig der Fall ist. Sie erhalten eine Spannung von etwa 8 V zwischen Pin 7 (TESTOUT) und Masse, wenn Rauch erkannt wird, und diese Spannung kann ebenfalls zur Ansteuerung eines Optokopplers verwendet werden.

RG — 230650-02

Über den Autor

Pascal Rondanes Hobby ist die Elektronik, seit er ein Teenager war, und er hat jedes Elektor-Heft seit dem Start der französischen Ausgabe im Jahr 1978 verschlungen. Er machte eine Ausbildung zum Elektroniker und arbeitete 20 Jahre lang in einem Unternehmen, das Funkgeräte von Motorola wartete und Elektronik-Platinen für IBM Frankreich reparierte. Danach arbeitete er 22 Jahre lang im Kundendienst und in der Prüfplatz-Konstruktion eines großen französischen Konzerns, der Straßenschilder herstellt. Seit einem Jahr ist er im Ruhestand, was ihm mehr Zeit für sein liebstes Hobby lässt und es ihm ermöglicht, sich an den Aktivitäten der *Association du Centre Historique de la Diffusion Radiophonique* (ACHDR) zu beteiligen, die sich für die Bewahrung des französischen audiovisuellen Erbes einsetzt [3].

Haben Sie Fragen oder Kommentare?

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

Passende Produkte

- **OzzMaker BerryGPS-GSM für Raspberry Pi**
  www.elektor.de/19326
- **D. Ibrahim, *GSM/GPRS Projects*, Elektor 2017**
  E-Buch, PDF, englisch: www.elektor.de/18203
- **Crowtail-4G SIM A7670E Modul GPS Breakout Board**
  www.elektor.de/20542

WEBLINKS

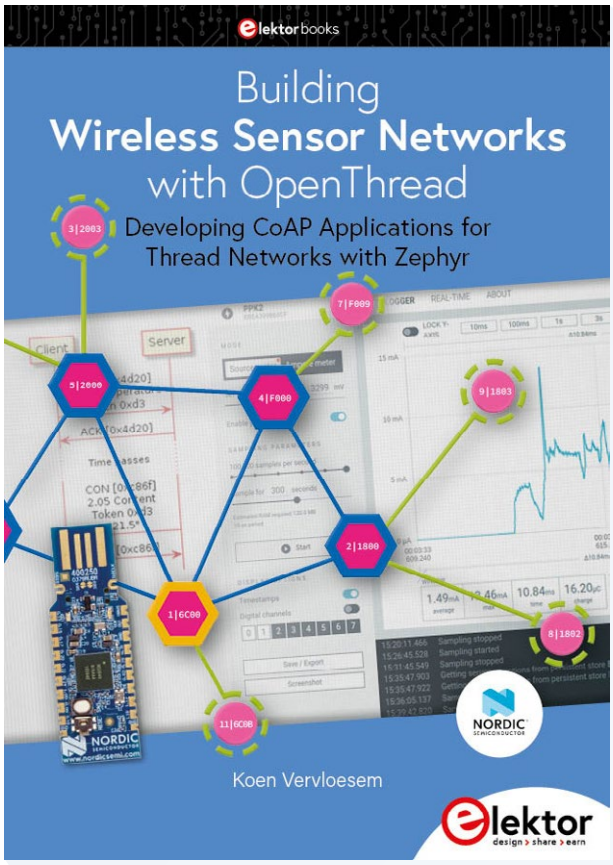
[1] Projekt-Downloads: https://elektormagazine.de/230650-02
[2] Rauchmelder-IC RE46C181: https://microchip.com/en-us/product/RE46C181
[3] ACHDR-Verband (französisch): https://achdr.over-blog.com

Low-Power-Thread-Geräte **optimiert und getestet**

Niedriger Energiebedarf … Niedrige Leistung?

Von Koen Vervloesem (Belgien)

Das neu veröffentlichte Elektor-Buch *Building Wireless Sensor Networks with OpenThread* bietet nicht nur eine Fülle von Hintergrundinformationen zum Aufbau und zur Programmierung eigener Netzwerke zum Anschluss aller Arten von Sensoren, sondern zeigt auch praktische Anwendungen mit einem nRF52840-Dongle von Nordic Semiconductor, der dem Buch beigefügt ist. Schauen wir uns einen solchen Anwendungsfall an, um zu sehen, ob es mit Zephyr, CoAP, Wireshark und Nordic wirklich ein Spaziergang im Low-Power-Park ist.



Anmerkung der Redaktion: *Dieser Artikel ist ein Auszug aus dem Elektor-Buch* Building Wireless Sensor Networks with OpenThread. *Dieser Auszug wurde formatiert und leicht bearbeitet, um den Konventionen und dem Seitenlayout der Zeitschrift Elektor zu entsprechen. Der Autor und die Redaktion helfen Ihnen gerne bei Fragen weiter. Kontaktdaten finden Sie im Kasten* Fragen oder Kommentare.

Die Tatsache, dass Thread oft als ein „energiesparendes drahtloses Mesh-Netzwerk" bezeichnet wird, schreit geradezu nach konkreten Zahlen zum Strombedarf von … (Trommelwirbel) … Thread-Anwendungen! Dieses Versäumnis wird durch diesen Artikel wettgemacht, in dem Sie sich mit diesem Thema beschäftigen werden:

- Messung der Stromaufnahme der grundlegenden Thread-Firmware auf einem nRF52840-Dongle von Nordic Semiconductor
- Senkung des Strombedarfs durch Deaktivierung ungenutzter Hardware
- Senkung des Strombedarfs durch Umwandlung der Anwendung in ein *Sleepy End Device* (SED)

Um zu beginnen, flashen Sie die grundlegende Thread-Anwendung (mehr darüber in Kapitel 5 des Buches) auf einen nRF52840-Dongle (der Code ist unter [4] zu finden). Bestätigen Sie anschließend, dass der Dongle eine Verbindung zu Ihrem Thread-Netzwerk

herstellt, indem Sie zum Beispiel die übertragenen Netzwerkpakete in Wireshark überwachen. Nun können Sie den Strombedarf der Anwendung auf dem Dongle untersuchen und optimieren.

Messung der Stromaufnahme mit dem Power Profiler Kit

Um Ihre Thread-Anwendung für einen geringeren Stromverbrauch zu optimieren, muss ihre Stromaufnahme messbar sein. Ein bemerkenswertes Hardware-Kit, das diesen Zweck erfüllt, ist das Power Profiler Kit II (PPK2) von Nordic Semiconductor. Dieses Gerät kann Ströme im Bereich unter einem Mikroampere bis zu 1 A messen und optional liefern, bei Spannungen unter 5 V. Das Gerät, dessen Stromaufnahme Sie messen (in diesem Fall der nRF52840-Dongle), wird als Device Under Test (DUT) bezeichnet. Auf der Softwareseite verwenden Sie die *Power Profiler*-Anwendung, die Teil von *nRF Connect for Desktop* [1] ist, der plattformübergreifenden Entwicklungssoftware von Nordic Semiconductor. Laden Sie das neuste Paket für Ihre Plattform (Windows/Linux/macOS) herunter und führen Sie es aus. Hinweis: Unter Linux müssen Sie die heruntergeladene AppImage-Datei zunächst ausführbar machen, etwa mit:

```
chmod +x nrfconnect-4.2.1-x86_64.appimage
```

Öffnen Sie dann *nRF Connect for Desktop* und klicken Sie neben *Power Profiler* auf *Install*.
Auf der Hardwareseite müssen Sie den nRF52840-Dongle so vorbereiten, dass er von einer externen Quelle über seinen VDD OUT-Pin mit Strom versorgt werden kann. Dazu müssen Sie die Leiterbahn, die die Lötbrücke SB2 kurzschließt, auftrennen und SB1 anlöten, wie in [2] beschrieben. SB1 und SB2 befinden sich auf der Unterseite der Platine, wie in **Bild 1** dargestellt. Achtung! Wenn Sie Ihren nRF52840-Dongle einmal auf diese Weise modifiziert haben, benötigt er dauerhaft eine externe Stromversorgung, selbst zum Flashen eines neuen Firmware-Images über USB.
Das PPK2 hat zwei Modi zur Messung des Stromverbrauchs:

- **Ampere Meter Mode** — das zu testende Gerät erhält Strom von einer externen Quelle (zum Beispiel USB oder einer Batterie). Sie können diesen Modus verwenden, um die Stromaufnahme des Dongles im Betrieb mit einer 3-V-Batterie zu testen.
- **Source Meter Mode** — das zu testende Gerät wird vom PPK2 mit Strom versorgt.

In beiden Modi müssen Sie wissen, wo sich die VDD OUT- und GND-Pins des Dongles befinden (**Bild 2**). Im Modus Ampere Meter verbinden Sie:

- PPK2 VIN mit Batterie BAT+
- PPK2 VOUT mit Dongle VDD OUT
- PPK2 GND mit Dongle GND
- Batterie BAT- mit Dongle GND

In dieser Konfiguration versorgt die Batterie den nRF52840-Dongle,

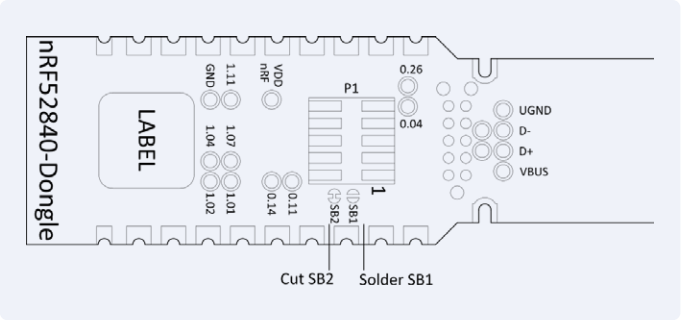


Bild 1. Schneiden Sie SB2 ab und löten Sie SB1 auf den nRF52840-Dongle für den Anschluss einer externen geregelten Spannungsquelle (1,8…3,6 V) über VDD OUT. (Bildquelle: Nordic Semiconductor)

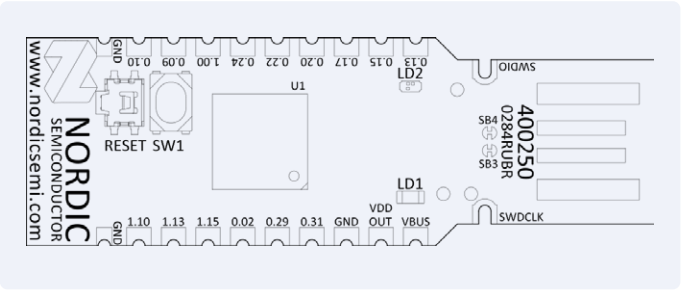


Bild 2. Die Pinbelegung des nRF52840-Dongles. (Bildquelle: Nordic Semiconductor)

aber der Strom fließt durch das Power Profiler Kit II, so dass er gemessen werden kann. Verbinden Sie im Modus Source Meter:

- PPK2 VOUT mit Dongle VDD OUT
- PPK2 GND mit Dongle GND

Die Anschlüsse sind in **Bild 3** dargestellt. In dieser Konfiguration versorgt das Power Profiler Kit II den nRF52840-Dongle mit Strom und misst gleichzeitig dessen Stromaufnahme (**Bild 4**).

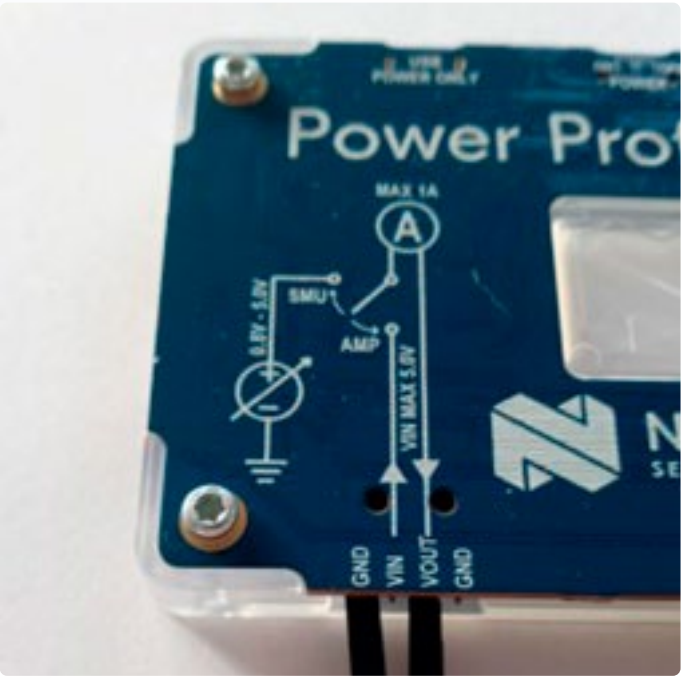


Bild 3. Anschluss des Power Profiler Kit II an den nRF52840-Dongle im Source-Meter-Modus.

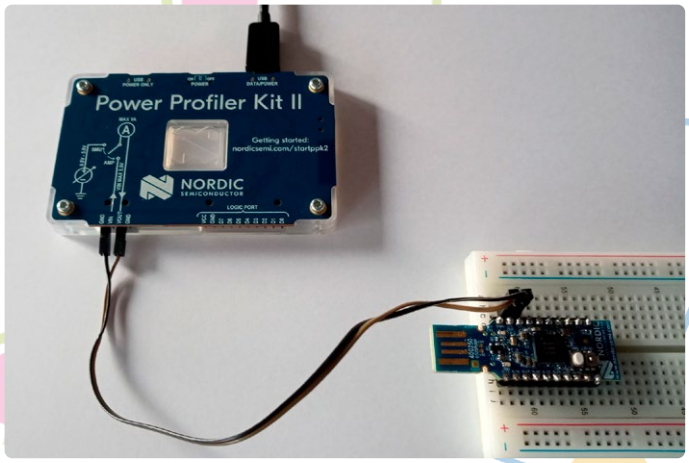


Bild 4. Das Power Profiler Kit II ist bereit, die Stromaufnahme des nRF52840-Dongles im Source-Meter-Modus zu messen.

Bild 5. Verbinden Sie das Power Profiler Kit II mit Ihrem PC.

Stromaufnahme der Thread-Anwendung

Als nächstes verbinden Sie das PPK2 mit dem Computer, indem Sie ein USB-Kabel in den USB-Anschluss *DATA/POWER* stecken und den POWER-Schalter neben dem USB-Anschluss auf *ON* stellen (siehe **Bild 5**). Öffnen Sie die Anwendung *Power Profiler* in *nRF Connect for Desktop* und klicken Sie oben links auf *Select Device*. Wählen Sie *PPK2*. Die Anwendung könnte Sie nun auffordern, das Gerät zu programmieren. Bestätigen Sie, indem Sie auf *Program* klicken.

Wählen Sie den Modus *Source Meter* oder *Ampere Meter*, je nachdem, wie das PPK2 mit Ihrem Dongle verbunden ist. Stellen Sie die Versorgungsspannung auf 3300 mV ein und aktivieren Sie *Enable Power Output* (auch im Ampere-Meter-Modus). Sofort nach dem Einschalten dieses Schalters wird der Dongle mit Spannung versorgt und Sie sehen seine Pakete im Thread-Netzwerk in Wireshark.

Klicken Sie dann im Power Profiler auf Start. Wie im **Bild 6** gezeigt, zeigt die Registerkarte *Data Logger* nun kontinuierlich die Stromaufnahme des Geräts an.

Die Statistik unter der Grafik zeigt einen durchschnittlichen Stromverbrauch von 6,29 mA. Dies ist hoch, aber verständlich, da Sie ja noch keine Maßnahmen zur Stromoptimierung getroffen haben. Wenn Sie den Power Profiler zusammen mit Wireshark eine Weile laufen lassen, werden Sie bei jedem Empfang einer Mesh-Link-Establishment-Message an die Link-Local-All-Nodes-Adresse `ff02::1` einen höheren Peak beobachten. Wenn Sie in diesen Peak hineinzoomen, sehen Sie etwas Ähnliches wie in **Bild 7**.

Senkung des Strombedarfs

Eine Menge Hardware auf dem Dongle, die Sie in der Produktion nicht verwenden oder nicht verwenden wollen, ist standardmäßig aktiviert. Aber jede aktivierte Komponente zehrt an den Energiereserven! Lassen Sie uns daher alle Gerätebaumdefinitionen in:

```
~/zephyrproject/zephyr/boards/arm/
    nrf52840dongle_nrf52840/nrf52840dongle_nrf52840.dts
```

eingehend untersuchen und die nicht benötigten Definitionen in der Overlay-Datei des Gerätebaums des Projekts deaktivieren:

```
boards/nrf52840dongle_nrf52840.overlay:
```

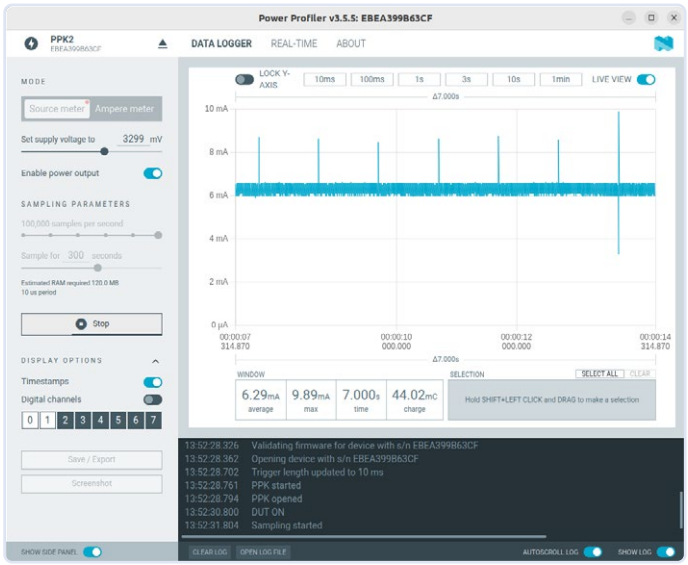


Bild 6. Die grundlegende Thread-Anwendung benötigt im Durchschnitt 6,29 mA.

Bild 7. Beim Empfang einer MLE-Anzeige gibt es eine deutliche Spitze in der Stromaufnahme.

Listing 1: Gerätebaumdefinitionen

```
/*
* Copyright (c) 2024 Koen Vervloesem
  <koen@vervloesem.eu>
*
* SPDX-License-Identifier: MIT
*/

/*
* Disabled unused hardware
*/

&adc {
        status = "disabled";
};
&uart0 {
        status = "disabled";
};
&i2c0 {
        status = "disabled";
};
&i2c1 {
        status = "disabled";
};
&pwm0 {
        status = "disabled";
};
&spi0 {
        status = "disabled";
};
&spi1 {
        status = "disabled";
};
&usbd {
        status = "disabled";
};
```

Listing 2: Kconfig-Konfigurationsdatei (prj.conf)

```
#
# Copyright (c) 2024 Koen Vervloesem
#
# SPDX-License-Identifier: Apache-2.0
#

# Enable networking and OpenThread
CONFIG_NETWORKING=y
CONFIG_NET_IPV6_NBR_CACHE=n
CONFIG_NET_IPV6_MLD=n
CONFIG_NET_L2_OPENTHREAD=y
CONFIG_OPENTHREAD_THREAD_VERSION_1_3=y
CONFIG_OPENTHREAD_SLAAC=y
CONFIG_OPENTHREAD_PING_SENDER=y
CONFIG_OPENTHREAD_DNS_CLIENT=y
CONFIG_OPENTHREAD_MLR=y

# Kernel options
CONFIG_MAIN_STACK_SIZE=2560

# Enable power management
CONFIG_PM_DEVICE=y

# Create Sleepy End Device
CONFIG_OPENTHREAD_MTD=y
CONFIG_OPENTHREAD_MTD_SED=y
CONFIG_OPENTHREAD_POLL_PERIOD=1000

# Disable USB
CONFIG_USB_DEVICE_STACK=n
CONFIG_BOARD_SERIAL_BACKEND_CDC_ACM=n
```

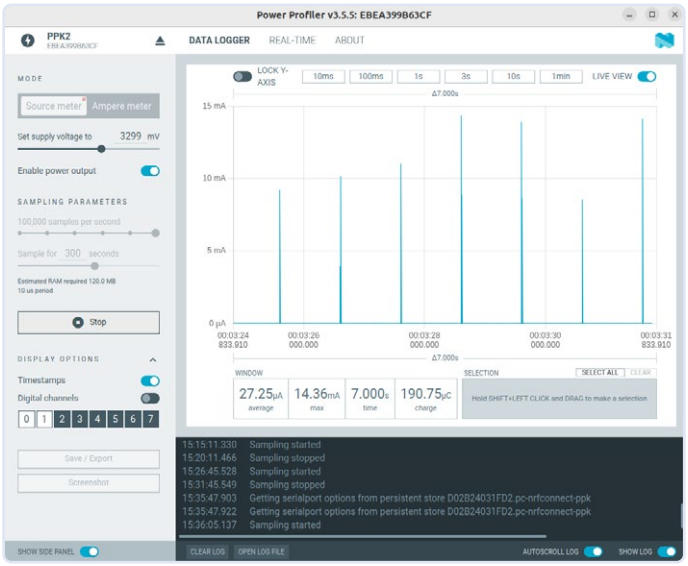


Bild 8. Die grundlegende Thread-Anwendung, die ihren Router jede Sekunde als Sleepy End Device abfragt, benötigt durchschnittlich 27 µA.

Das entsprechende Programm finden Sie in **Listing 1**. Hinweis: Die Zephyr-Shell ist nicht mehr über USB zugänglich, wenn `uart0` deaktiviert wird.

Außerdem sieht die Kconfig-Konfigurationsdatei `prj.conf` wie in **Listing 2** aus. Damit wird im Wesentlichen die Energieverwaltung aktiviert und USB deaktiviert. Es fehlen auch die Optionen zur Aktivierung des Logging und der Zephyr-Shell. Außerdem wird das Gerät als *Sleepy End Device* konfiguriert, so dass der Funk die meiste Zeit schlafen kann. Die Polling-Periode (das Intervall zwischen Anfragen des SED an seinen Router) wird auf 1000 ms festgelegt. Erzeugen Sie nun die Firmware neu

```
$ west build -p auto -b nrf52840dongle_nrf52840
```

erstellen Sie das Paket und flashen Sie die Firmware auf den Dongle. Sie messen nun eine wesentlich geringere durchschnittliche Stromaufnahme von etwa 27 µA (**Bild 8**). Man sieht deutlich eine Spitze von mehr als 10 mA in der Stromaufnahme pro Sekunde, und gleichzeitig zeigt Wireshark eine Datenanfrage vom SED an seinen Router.

Wenn Sie eine Ping-Anfrage an das SED senden, werden Sie feststellen, dass es bis zur Antwort bis zu einer Sekunde dauern kann, was dem Polling-Intervall des Geräts entspricht (**Bild 9**).
Je länger Sie das Abfrageintervall einstellen, desto geringer ist die mittlere Stromaufnahme, aber desto langsamer reagiert natürlich das Gerät. Sie können dies testen, indem Sie die Kconfig-Konfigurationsvariable `CONFIG_OPENTHREAD_POLL_PERIOD` auf 5.000 Millisekunden einstellen. Erstellen Sie die Firmware neu und flashen Sie sie auf das Thread-Gerät. Wenn Sie dann den Stromverbrauch messen, beträgt er im Durchschnitt etwa 7,5 µA, mit Stromspitzen im Abstand von 5 s, wie in **Bild 10** dargestellt. Der Nachteil dieser langen Zeitspanne ist also, dass eine Ping-Anfrage an das SED bis zu fünf Sekunden dauern kann, wodurch es deutlich weniger schnell reagiert.

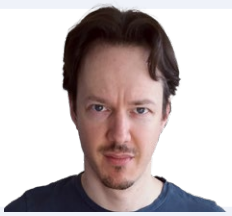
Zusammenfassung und weitere Entdeckungen

In diesem Artikel habe ich gezeigt, wie man die Stromaufnahme einer Thread-Anwendung misst, die auf einem nRF52840-Dongle läuft. Dank des *Power Profiler Kit II* von Nordic Semiconductor und der dazugehörigen Power-Profiler-Anwendung in *nRF Connect for Desktop* können Sie einen umfassenden Einblick in den Strombedarf Ihrer Anwendung erhalten.
Als Beispiel habe ich die Stromaufnahme der grundlegenden Thread-Anwendung in diesem Buch detailliert beschrieben. Sie haben gelernt, wie Sie die Stromaufnahme senken können, indem Sie nicht benötigte Hardware deaktivieren und das Gerät als *Sleepy End Device* konfigurieren. Sie haben auch gelernt, wie sich das Polling-Intervall auf den Stromverbrauch des Geräts auswirkt: Je länger die Polling-Periode, desto geringer der Strombedarf, aber desto weniger reaktionsschnell wird das Gerät.
Dieser Artikel ist nur ein Ausgangspunkt für weitere Optimierungsbemühungen. Wenn Sie die Stromaufnahme Ihrer Thread-Anwendung so gering wie möglich halten wollen, sollten Sie die Vorteile der *Power Management API* von Zephyr [3] nutzen. Damit können Sie die Stromsparfunktionen des SoCs und anderer Bausteine, zum Beispiel angeschlossener Sensoren nutzen. Ihr Code liest dann eine Sensormessung, sendet die Daten über das Thread-Netzwerk, schaltet den Sensor aus, schläft, wacht auf, schaltet den Sensor wieder ein und beginnt den ganzen Zyklus von vorne.

SE — 240225-02

Haben Sie Fragen oder Kommentare?

Haben Sie technische Fragen oder Anmerkungen zu diesem Artikel? Senden Sie bitte eine E-Mail an den Autor unter koen@vervloesem.eu oder an die Elektor-Redaktion unter redaktion@elektor.de.



Über den Autor

Koen Vervloesem schreibt seit über 20 Jahren über Linux, Open-Source-Software, Sicherheit, Heimautomatisierung, künstliche Intelligenz (KI), Programmierung und das Internet der Dinge (IoT). Er hat einen Master-Abschluss in Computer Science Engineering, einen Master-Abschluss in Philosophie und ein LPIC-3 303 Sicherheitszertifikat. Er unterrichtet Linux- und Python-Kurse für Studenten, die einen Associate-Abschluss in Internet der Dinge anstreben.

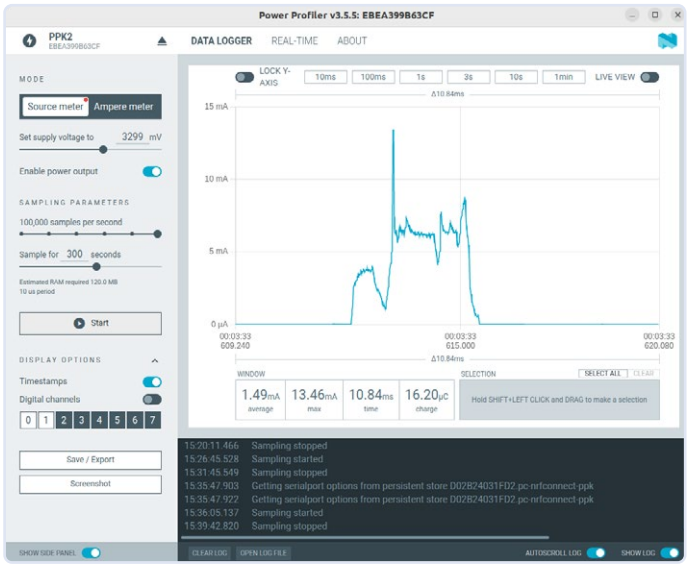


Bild 9. Bei jeder Datenanforderung an seinen Router verzeichnet das Sleepy End Device eine Spitze in seiner Stromaufnahme.

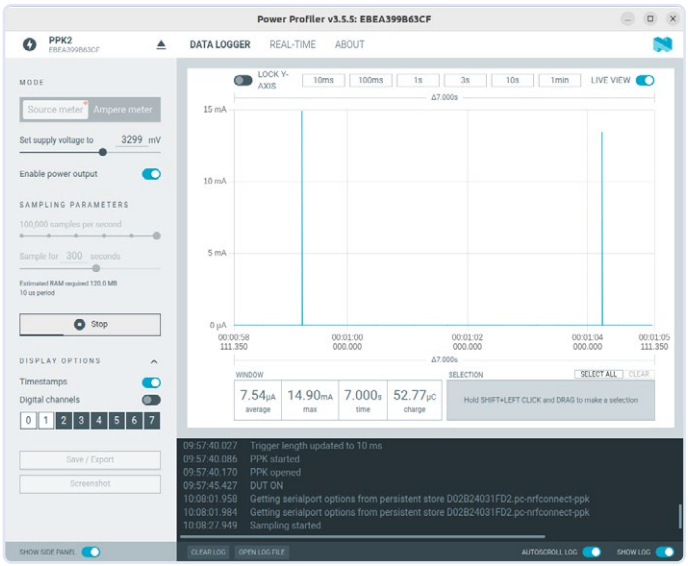


Bild 10. Bei einer Abfragezeit von 5 s benötigt das Sleepy End Device im Durchschnitt nur 7,5 µA.

Passende Produkte

- **Koen Vervloesem, *Building Wireless Sensor Networks with OpenThread*, Elektor 2024**

  Taschenbuch mit nRF52840-Dongle, englisch:
  www.elektor.de/20870

  Taschenbuch, englisch: www.elektor.de/20860
  E-Buch, PDF, englisch: www.elektor.de/20861

WEBLINKS

[1] nRF Connect for Desktop: https://www.nordicsemi.com/Products/Development-tools/nRF-Connect-for-Desktop
[2] nRF52840-Dongle Hardware-Modifikation: https://tinyurl.com/rvj5mp47
[3] Power-Management-API von Zephyr : https://docs.zephyrproject.org/latest/services/pm/index.html
[4] Main.c, CMakeLists.txt und Configuration Overlay für den nRF52840-Dongle:
https://github.com/koenvervloesem/openthread-applications

Aus dem Leben gegriffen

Der Gender-Gap

Marie und Pierre Curie in ihrem Labor.
(Quelle: Shutterstock/Morphart Creation)

Von Ilse Joostens (Belgien)

Frauen und Technik haben aufgrund fehlender Vorbilder, hartnäckiger gesellschaftlicher Vorurteile und stereotyper Geschlechterrollen immer noch ein etwas schwieriges Verhältnis zueinander. Wenn Sie sich Fotos von Funk- und Elektronikmärkten oder von Amateurfunkmessen genau ansehen, werden Sie feststellen, dass der behäbige ältere Mann mit zurückweichendem Haaransatz und einem kleinen (?) Bauch prominent vertreten ist. Frauen hingegen sind kaum zu sehen. Und nicht nur in der Technik, sondern auch beim jährlichen Jamboree für Investoren und Sparer einer bekannten flämischen Wirtschaftszeitung fällt auf, dass der männliche Teil deutlich in der Mehrheit ist.

Glücklicherweise ziehen Tech-Messen, Hackerspaces, Fab Labs und Maker Faires heutzutage ein vielfältigeres Publikum an, aber selbst dort sind Männer im Durchschnitt immer noch in der Mehrheit. Ingenieurwesen und Finanzwissen sind jedoch überhaupt nicht geschlechtsspezifisch, und im Prinzip können sich alle Menschen damit beschäftigen.

Pickelige Nerds

Im Laufe der Geschichte haben Frauen immer wieder gesellschaftlich wichtige Rollen übernommen, von Wikingerkriegerinnen [1][2] bis hin zu Wissenschaftlerinnen und Ingenieurinnen. In einer Welt, in der Männlichkeit die Norm ist und in der Männer regelmäßig die Lorbeeren für die Arbeit von Frauen einheimsen, scheint es, als ob alle großen Entdeckungen und technologischen Revolutionen das Werk von Männern waren. Meine Herren! Hätte sich Pierre Curie bei den Nominierungen für den Nobelpreis 1903 nicht für seine Frau Marie eingesetzt, wäre sie als edle Unbekannte, als Fußnote in der Geschichte gelandet [3].

In der Astronomie, beim Militär während des Zweiten Weltkriegs und auch bei der NASA haben Frauen komplexe Berechnungen von Hand durchgeführt. Später, als diese Arbeit von Maschinen wie dem ENIAC übernommen wurde, wurde das Programmieren von Computern naturgemäß ein Frauenberuf (**Bild 1**) [4][5]. Das änderte sich, als die Arbeitgeber in den späten 1960er Jahren erkannten, dass Programmieren nicht nur eine weitere, weniger wertvolle Verwaltungsarbeit wie Tippen und Ablegen ist. In dem Maße, in dem der Beruf des Programmierers an Prestige gewann, wurden mehr Männer ausgebildet und auch bei Qualifikationstests bei Bewerbungen insgeheim bevorzugt. In der Folge setzte sich der stereotype Typus des ungepflegten Mannes mit begrenzter

Figure 1: Programming was a woman's job. (Source: Shutterstock/emkaplin)

Bild 2. Nerd bei der Arbeit. (Quelle: Shutterstock/ Arsenii Palivoda)

Bild 3. Marthe Douriau (oben Mitte). (Quelle: forum.retrotechnique.org)

Sozialkompetenz, des „pickeligen Nerds" durch. Auch heute noch wird die IT-Branche zu 82 % von Männern vertreten. Offenbar hält sich hartnäckig das Bild des spielsüchtigen, adipösen Nerds, der nachts auf einem kleinen Dachboden, umgeben von leeren Pizzakartons, an Computern bastelt (**Bild 2**). In unserer geliebten Elektronikbranche sind Frauen vor allem in Montageunternehmen tätig, in denen elektronische Produkte gebaut oder Platinen bestückt werden. Glücklicherweise gibt es aber auch immer mehr weibliche Elektronikingenieure, darunter einige mit eigenen YouTube-Kanälen. Selbst in der grauen Vorzeit waren Frauen auf hohem Niveau in der Elektronik tätig, wie etwa Marthe Douriau (1899-1968) [6], um mal eine weniger bekannte Person zu nennen. Diese elegante Dame mit gewelltem, jungenhaftem Haar und Perlenkette war nicht nur Ingenieurin bei Philips und Ferrix, sondern schrieb auch Artikel für die Zeitschrift L'Antenne, wo sie Mitglied des Redaktionsausschusses war (**Bild 3**). Sie veröffentlichte auch eine Reihe von Büchern über allgemeine und Automobilelektronik, letzteres allerdings unter dem Pseudonym Marc Dory, da sie befürchtete, sonst nicht ernst genommen zu werden.

Auch heute noch werden die technischen Fähigkeiten der Durchschnittsfrau durchaus in Frage gestellt, wenn es um Autotechnik geht. In einem überaus peinlichen Angebot für einen Autobatterietester auf eBay [7] wird unverfroren behauptet, das Gerät sei einfach zu bedienen, sogar für weibliche Autofahrer.

A Man's World

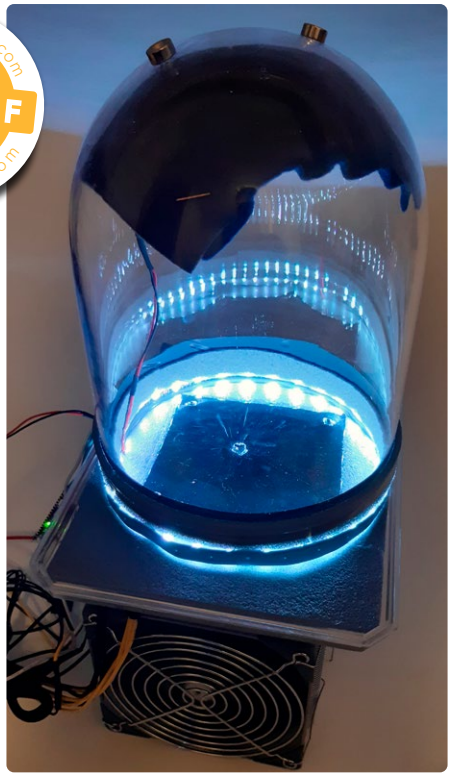
It's A Man's Man's Man's World, sang James Brown schon 1966, und leider ist es wahr. In ihrem Buch „Invisible Women" zeigt Caroline Criado Perez, wie die Welt weitgehend auf Männer zugeschnitten ist und die Hälfte der Weltbevölkerung aufgrund einer (un)bewussten Voreingenommenheit gegenüber dem Mann systematisch ignoriert wird. Sicherheitsgurte zum Beispiel sind nicht für Frauen gemacht und wurden jahrelang nur mit männlichen Dummys getestet. Bei mir schneidet der Sicherheitsgurt immer in den Hals, und obwohl ein Gurtpolster eine gewisse Erleichterung bringt, führt sie zu neuen Ärgernissen wie der Tatsache, dass das Polster dazu neigt, zu verrutschen oder beim Abschnallen das Aufrollen des Gurtes behindert. Den Gurt zwischen den Brüsten zu tragen ist auch nicht gerade bequem, denn ehe man sich versieht, rutscht das verdammte Ding wieder rauf bis zum Hals. Selbst die Tech-Welt ist, vor allem in Westeuropa, immer noch eine Männerbastion. Nehmen Sie den Genter Winter Circus [8], der auf Initiative eines Konsortiums von Genter Unternehmern und Organisationen derzeit in einen Technologietempel für Start-ups und Scale-ups verwandelt wird. Der Ort sieht prächtig und architektonisch imposant aus, aber er fühlt sich nicht sicher, geborgen oder warm an. Die Tatsache, dass man seine Idee erst vor einer (meist männlichen) Jury vorstellen und am besten etwas mit KI machen muss, um aufgenommen zu werden, ist das perfekte Rezept, um Frauen abzuschrecken. Dasselbe gilt für Formulierungen wie „kaufmännisch", „stressresistent", „fanatisch" oder „ergebnisorientiert" in vielen Stellenangeboten für technische Berufe. Obwohl sich viele Frauen für eine technische Ausbildung oder einen technischen Beruf entscheiden, brechen die meisten von ihnen schließlich ab. Unterbewertung, ständiger Kampf gegen Vorurteile und ein unfreundliches, manchmal sogar unsicheres Arbeitsumfeld sind dabei gang und gäbe.

Die Überwindung bestehender Vorurteile und um mehr Frauen für technische Berufe wie das Ingenieurwesen zu gewinnen, sind nach wie vor wichtig, wenn wir uns auf eine integrativere Welt zubewegen wollen, die nicht in erster Linie auf Männer zugeschnitten ist und in der innovative Ideen für und von Frauen entwickelt werden.

SG — 240247-02

WEBLINKS

[1] Wikipedia: Birka Grave Bj 581: https://de.wikipedia.org/wiki/Schildmaid_von_Birka
[2] YouTube: Efin Reality — So you want to be a Warrior...: https://youtu.be/_gfo0peYu6o
[3] Liz Heinecke: When Marie Curie was almost excluded from winning the Nobel Prize: https://lithub.com/when-marie-curie-was-almost-excluded-from-winning-the-nobel-prize
[4] Feminer: Ghislaine Aouragh — Women in tech: are we going back to "the age of the Computer Girls?" (niederländisch): https://feminer.nl/magazine/rolmodellen/vrouwen-in-tech-interview-chantal-schinkels
[5] YouTube: Computer History Archives Project — IBM 701 Rare promo 1953 first of IBM 700 Series Mainframes: https://www.youtube.com/watch?v=fsdLxarwmTk
[6] Forum Retrotechnique (französisch): Marthe Douriau or Marc Dory?: https://forum.retrotechnique.org/t/marthe-douriau-ou-marc-dory/79440
[7] eBay: Car Battery Tester Analyzer for Automobile: https://ebay.co.uk/itm/294919416437?chn=ps&_ul=GB&mkevt=1&mkcid=28
[8] Winterzirkus Ghent: https://wintercircus.be/en



Nebelkammer selbstgebaut

Unsichtbare Strahlung sichtbar machen

Von Matthias Rosezky (Österreich)

Wollten Sie schon immer Radioaktivität und ionisierende Strahlung mit eigenen Augen sehen? In diesem Artikel wird beschrieben, wie man eine Nebelkammer mit ein paar handelsüblichen Komponenten selbst bauen kann. Damit können Sie die natürliche Hintergrundstrahlung buchstäblich sehen und sogar den radioaktiven Zerfall einiger Substanzen sichtbar machen.

Was ist das erste, was Ihnen in den Sinn kommt, wenn Sie die Begriffe „Radioaktivität" oder „ionisierende Strahlung" hören? Wahrscheinlich nicht viel Positives und eher an Gefahr, an die zahllosen Katastrophen in Atomanlagen wie Tschernobyl oder Fukushima. Diese Art von Strahlung gibt es jedoch schon viel länger, als wir sie nutzen und nachweisen können. Tatsächlich ist die natürliche, die so genannte Hintergrundstrahlung immer um uns herum, und zwar schon seit der gesamten Existenz der Menschheit. Sie ist für uns nichts Ungewöhnliches, auch wenn wir uns ihrer vielleicht nicht einmal bewusst sind. In der Tat gibt es zwei Quellen für natürliche Strahlung: Zum einen die terrestrische Strahlung, die aus dem Erdinneren kommt, und zum anderen die kosmische Strahlung, also die gesamte hochenergetische Strahlung, die hauptsächlich dem Weltraum entstammt. Die kosmische Strahlung betrifft meist nur die Luftfahrt, Astronauten und Satelliten, aber die terrestrische ist für uns weitaus wichtiger und interessanter.

Der terrestrische Teil wird durch eine Vielzahl natürlich vorkommender radioaktiver Isotope erzeugt und ist so gut wie überall zu finden: in Gesteinen, Mineralien, in der Luft, im Wasser, in unserer Nahrung und in unserem Körper. Die meisten von ihnen stammen aus dem radioaktiven Zerfall von Uran und Thorium, aber es gibt auch ein natürlich vorkommendes instabiles Isotop von Kalium zum Beispiel. Dadurch sind Sie selbst auch leicht radioaktiv.

Ein Fenster zur Strahlung

Sie fragen sich jetzt vielleicht, ob es eine Möglichkeit gibt, diese Strahlung, die überall um uns herum ist, mit eigenen Augen zu sehen? Natürlich gibt es dazu eine Möglichkeit, die in der Teilchenphysik des frühen 20. Jahrhunderts eine wichtige Rolle spielte: die Nebelkammer. Mit diesem Gerät kann man nicht nur ionisierende Strahlung mit eigenen Augen sehen, sondern auch unterscheiden, welche Art von Strahlung da zu sehen ist. Es gibt zwei Arten von Nebelkammern; diejenige, auf die ich mich hier konzentriere, heißt „Diffusionsnebelkammer" und wurde im Jahr 1936 von einem amerikanischen Physiker erfunden. Sie kann kontinuierlich und praktisch ohne Ausfallzeiten eingesetzt werden und lässt sich mit der heutigen Technologie recht einfach zu Hause nachbauen, was ich auch getan habe.

Die Nebelkammer

Für eine Diffusionsnebelkammer benötigt man im Wesentlichen zwei Dinge: eine sehr kalte Metallplatte am Boden und etwas (Isopropyl-) Alkohol darüber (**Bild 1**). Wenn der Alkohol verdampft, sättigt er die Luft, und wenn sie abkühlt, sinkt er ebenfalls. Wenn dieses gesättigte Luftgemisch dann in die Nähe der Kühlplatte kommt, kondensiert die Luft an

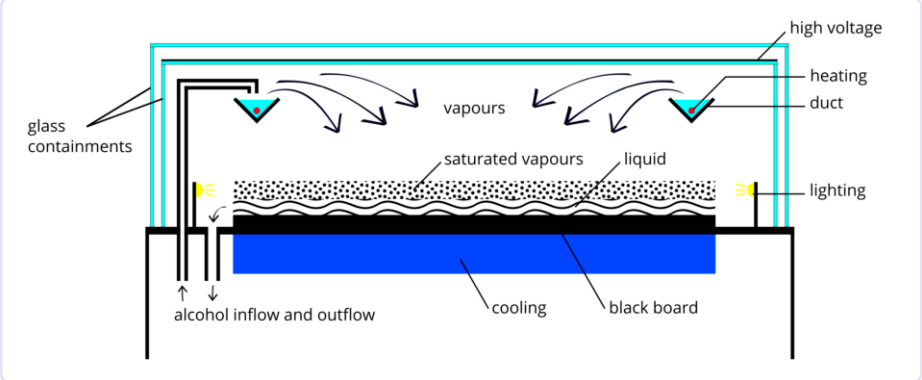


Bild 1. Prinzip einer kontinuierlich arbeitenden Nebelkammer. (Quelle: Nuledo, CC BY-SA 4.0 - https://creativecommons.org/licenses/by-sa/4.0 - über Wikimedia Commons)

der Kühlplatte und der Alkohol wird wieder flüssig. Direkt über der Platte gibt es jedoch einen kleinen Luftabschnitt, in dem dieses Luft-Alkohol-Gemisch übersättigt ist, kurz vor der Kondensation steht und nur darauf wartet, dass irgendetwas es stört.
Und dieser Störenfried ist die ionisierende Strahlung! Wie der Name schon sagt, ionisiert sie die Luftmoleküle beim Durchdringen dieser Schicht, und diese zurückbleibenden Ionen dienen als Kondensationskerne für das Gemisch (**Bild 2**). Sie kondensieren entlang der Ionisationslinie und bilden eine kleine Wolke, die zeigt, wohin das ionisierende Teilchen gereist ist. Je nachdem, welche Art von Teilchen die Nebelkammer durchquert hat, kann man verschiedene Arten von Spuren unterscheiden. Das meiste, was Sie von den Zerfallsprozessen sehen, sind entweder Alpha- oder Beta-Teilchen. Die schweren, voluminösen und langsamen Alphateilchen (es handelt sich um Helium-4-Kerne) hinterlassen kurze, dicke Spuren, die sehr gerade erscheinen. Im Gegensatz dazu hinterlassen schnellere und leichtere Beta-Teilchen (Elektronen oder Positronen) viel längere, dünnere und manchmal ziemlich gezackte Spuren in der Nebelkammer (**Bild 3**). Wenn Sie selbst diverse kleine radioaktive Präparate in die Nebelkammer einbringen, können Sie vielleicht auch andere Zerfallsarten erkennen.

Selber bauen

Wie habe ich es geschafft, eine solche Kammer von Grund auf zu bauen? Wie gesagt, besteht die Nebelkammer aus einem sehr kalten Metallblech und etwas Alkoholdampf. Werfen wir zunächst einen Blick auf das Metallblech. Um eine optimale Wärmeverteilung über die gesamte Fläche zu erreichen, habe ich ein 10×10 cm großes Kupferblech mit einer Stärke von 4 mm gewählt (**Bild 4**). Kupfer ist ein ausgezeichneter Wärmeleiter, so dass die Temperatur über das gesamte Blech sehr gleichmäßig ist. Ich kann die 10×10 cm große Fläche vollständig auf etwa -30°C abkühlen und auf ihrer gesamten Fläche Spuren beobachten, was hilfreich ist, wenn ich verschiedene Proben in die Nebelkammer einfüge. Wenn Sie Geld sparen wollen, können Sie auch viel billigeres Aluminium verwenden, aber es dürfte wahrscheinlich nicht so gut funktionieren.
Zur Kühlung verwende ich einen thermoelektrischen Kühler, ein Peltier-Element, das im Wesentlichen eine kleine Wärmepumpe ohne bewegliche Teile ist. Wenn man eine Spannung anlegt, kühlt sich die eine Seite ab,

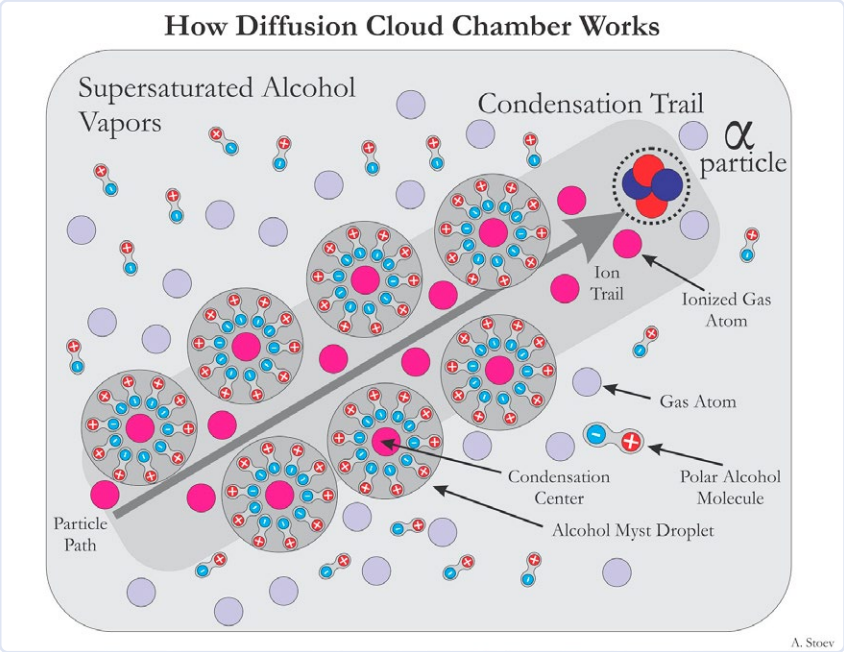


Bild 2. Bildung von Alkoholkondensationsstreifen in der Diffusionsnebelkammer. (Quelle: Kotarak71, CC BY-SA 4.0 - https://creativecommons.org/licenses/by-sa/4.0 - über Wikimedia Commons)

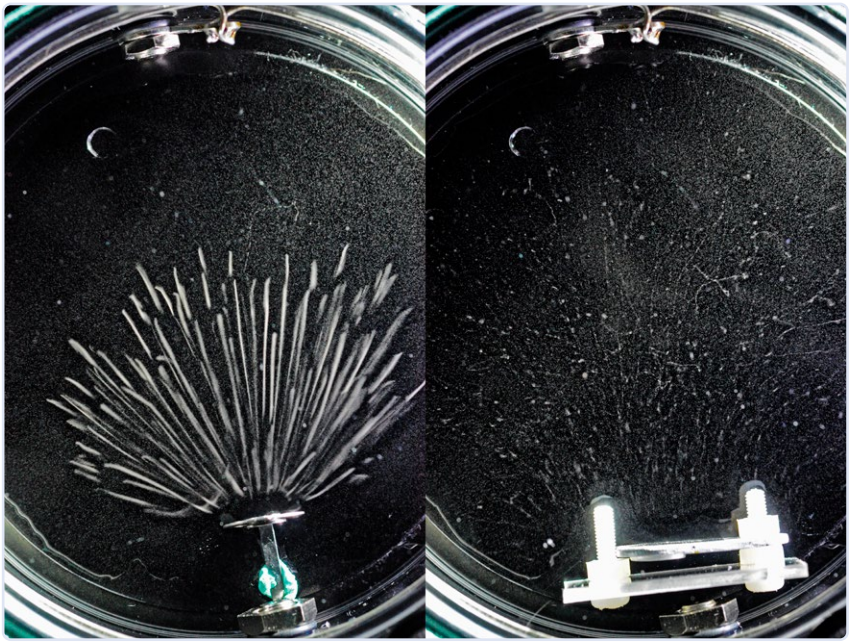


Bild 3. Zum Beispiel: Alphaspuren von einer Am-241-Quelle (links) und Beta-Spuren von einer Sr-90/Y-90-Quelle (rechts). (Quelle: Kebuk awan, CC BY-SA 4.0 - https://creativecommons.org/licenses/by-sa/4.0 - über Wikimedia Commons)

Bild 4. Kupferblech auf der Oberseite des CPU-Lüfters.

während sich die andere erwärmt, so dass ein konstantes Temperaturgefälle zwischen den beiden Seiten entsteht. Die kalte Seite ist (unter Verwendung von Wärmeleitpaste) direkt mit dem Kupferblech verbunden, und die Wärme an der heißen Seite muss so gut wie möglich abgeführt werden, damit sie sich nicht zu stark erwärmt. Das ist notwendig, weil das Temperaturgefälle konstant ist, also die Temperatur der kalten Seite von der an der heißen Seite abhängt. Zum Abtransport der Wärme verwende ich einen kräftigen Dual-Tower-CPU-Lüfter und zwei Ventilatoren, (preiswerte) Bauteile, die in gewöhnlichen PCs verwendet werden. Mit einer herkömmlicher Luft- oder Flüssigkeitskühlung werden Sie nie in der Lage sein, die heiße Seite unter die Umgebungstemperatur zu bekommen. Der erforderliche Temperaturgradient beträgt also mindestens etwa 50°C, um -30°C zu erreichen, wenn Sie den thermoelektrischen Kühler (und später die Nebelkammer) bei Raumtemperatur betreiben. Um dies zu erreichen, müssen Sie mehrere Peltierelemente in Reihe schalten, oder Sie können direkt einen bereits kaskadierten Kühler wie den TEC2-25408 verwenden. Bei diesem Kühler sind zwei einzelne Peltierelemente intern gestapelt, so dass Sie dieses (dickere) einzelne Gerät wie zwei Einzelelemente verwenden können. Der Kühler ist dann in der Lage, einen viel höheren Temperaturgradienten von bis zu 80°C aufzubauen (zugegebenermaßen nur unter den idealen Bedingungen); in der Praxis sind aber 50°C bis 60°C mit diesem einzelnen Modul problemlos möglich, wenn Ihr Kupferblech nicht zu groß ist.

Montageherausforderungen gelöst

Ich wollte das Kupferblech, den thermoelektrischen Kühler und den Luftkühler zu einer stabilen mechanischen Einheit verbinden, die man an- und abschrauben kann, falls etwas modifiziert oder repariert werden muss. Außerdem wollte ich in der Lage sein, den thermoelektrischen Kühler mit ein wenig Wärmeleitpaste mit gutem Druck zu befestigen.
Aus diesem Grund habe ich die mit dem CPU-Lüfter gelieferten Intel-Halterungen verwendet, die passenden Löcher in das Kupferblech gebohrt und alle Teile mit Kunststoffschrauben und Abstandshaltern verbunden. Kunststoffschrauben sind ein offensichtlicher Kompromiss, da sie nicht annähernd so stark und zuverlässig sind wie normale Metallschrauben. Die Verwendung von Metallschrauben in der Nähe des sehr kalten Kupferblechs und des warmen CPU-Kühlers ist jedoch eine

Bild 5. Kupferblech mit Wärmeisolierung und Kunststoffabstandshaltern.

Bild 6. Schwarzes Kupferblech mit der Acrylplatte drum herum. Ich habe das Acrylglas mit einer weiteren Isolierung überzogen, um eine weiche Dichtung für die Glasglocke zu schaffen.

schreckliche Idee, schon allein wegen der enormen Wärmeübertragung, die zwischen der heißen und der kalten Seite stattfinden würde. Kunststoff hingegen hat eine sehr schlechte Wärmeleitfähigkeit, was in diesem Fall sehr wichtig ist. Keramikschrauben wären zwar ideal, sind aber viel zu teuer für diese Anwendung. Um die Wärmebelastung des Kupferblechs weiter zu verringern, habe ich außerdem eine Wärmeisolierung an der Unterseite des Blechs und um das Peltier-Element herum angebracht, wie in **Bild 5** zu sehen.
Um die empfindliche Nebelschicht so wenig wie möglich zu stören und um sicherzustellen, dass das Luftgemisch später mit Alkohol gesättigt ist, wird eine Glasglocke um das Kupferblech herum aufgestellt, so dass es etwas Platz braucht, damit sie darauf „sitzen" kann. Zu diesem Zweck habe ich eine einfache Acrylplatte besorgt, die größer ist als der Durchmesser der Glasglocke, und sie von unten mit einigen Plastikstiften auf dem Kupferblech befestigt (**Bild 6**). Diese Anordnung bietet auch etwas Platz für einige helle LEDs, die tangential auf das Kupferblech strahlen und die feine Nebelschicht beleuchten können. Dieses Licht erhöht den Kontrast und trägt wesentlich dazu bei, dass alle Spuren auf dem Kupferblech deutlich sichtbar werden. Um die Sichtbarkeit weiter zu verbessern, habe ich auch die freiliegende Oberseite des Kupferblechs mit schwarzem Isolierband abgedeckt.

Stromversorgung und mehr

Um das alles mit Strom zu versorgen, habe ich ein Standard-ATX-Netzteil verwendet, das ich noch von einem alten PC übrig hatte. Diese Netzteile können bei ausreichendem Strom 12 V ausgeben, mit denen der thermoelektrische Kühler direkt versorgt werden kann. Über einige 5-V-Leitungen können die Lüfter direkt und ohne Controller betrieben werden. Sie drehen dann leise, aber schnell genug, um den riesigen Kühlkörper mit ausreichend Luftstrom zu versorgen. In meinem Fall habe ich einen Schalter zwischen 5 V und 12 V als groben Wahlschalter für die Lüftergeschwindigkeit eingebaut. Die von mir verwendeten LEDs sind adressierbare 5-V-RGB-LEDs, so dass ich mit verschiedenen Farben und Helligkeitswerten experimentieren konnte. Aus diesem Grund habe ich sie auch an die Stromversorgung angeschlossen und einen Raspberry Pi Pico verwendet, um alle Helligkeits- und Farbwerte einstellen zu können. Am Ende habe ich mich für ein leichtes Kaltweiß bei maximaler Helligkeit entschieden, was für mich beim Betrachten der verschlungenen Ionisationspfade am besten funktionierte.
Sie haben vielleicht bemerkt, dass ich noch nicht über den Alkohol gesprochen habe. Er muss irgendwo über die Kühlplatte laufen, ohne zu tropfen. Dazu habe ich einige Filzstücke genommen (**Bild 7**) und sie mit einigen Magneten an der Oberseite der Glocke befestigt,

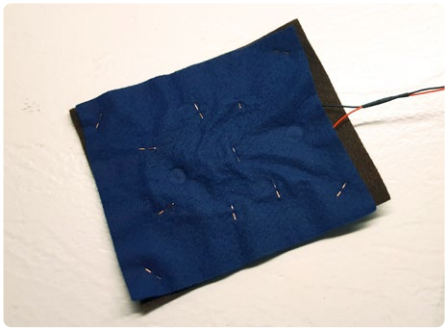


Bild 7. Zusammengeheftete Filzstücke mit Heizwiderstand.

wie auf dem Titelbild dieses Artikels zu sehen ist. Diese halten fest genug, ohne dass eine kompliziertere Art der Befestigung erforderlich wäre, und mit dem zusätzlichen Vorteil, dass die Filzstücke leicht abnehmbar sind. Eine letzte Verbesserung, bevor man das Ganze einbaut, ist eine kleine Heizung für den Alkohol. Um eine noch dickere Nebelschicht über der Kühlplatte zu erhalten, könnte man entweder deren Temperatur senken (was eine höhere Spannung oder mehrere Peltier-Elemente erfordern würde) oder die Temperatur des Alkohols erhöhen, damit mehr davon verdunstet. Aus diesem Grund habe ich die Filzstücke einfach mit ein paar Heizwiderständen versehen, sie festgetackert und an die 12-V-Schiene meines Netzteils angeschlossen. **Aus Sicherheitsgründen muss man unbedingt darauf achten, dass in der Nähe des Alkohols nichts kurzgeschlossen werden kann, denn er ist ja leicht entzündlich!** In meinem Fall sind die Widerstände ziemlich groß, was zu einer Temperaturerhöhung von nur etwa 5...10 °C über der Umgebungstemperatur führt. Aber selbst damit kann man schon eine deutliche Verbesserung feststellen.

Inbetriebnahme und Ergebnisse

Um die Nebelkammer in Betrieb zu nehmen, gebe ich einfach etwas Isopropylalkohol auf den Filz, so dass er gut durchtränkt ist, aber nicht tropft. Dann lege ich die Glasglocke um die obere Platte und setze die Nebelkammer in Betrieb. Es dauert ein paar Minuten, bis sie vollständig auf etwa -30°C abgekühlt ist. Über der Platte bildet sich eine kleine Pfütze aus Alkohol, die verhindert, dass sich auf der kalten Platte Eis bildet. Danach bildet sich nach und nach eine Wolkenschicht über der Platte, und bald darauf beginnen sich überall in dieser Region Spuren zu bilden.

Die Ergebnisse sind in diesem Video [1] zu sehen, in dem ich eine kleine, leicht radioaktive Probe in die Nebelkammer gab. Außerdem ist das Projekt auf der Elektor-Labs-Projektseite [2] sowie auf Hackaday [3] verfügbar.

That's it! Es ist nicht allzu kompliziert und funktioniert gut. Die Tatsache, dass man mit ein paar Standardkomponenten und etwas Bastelei eine so gut funktionierende Nebelkammer bauen kann, erstaunt mich immer wieder. Lassen Sie mich wissen, was Sie von diesem Projekt halten!

RG — 230495-02

Haben Sie Fragen oder Kommentare?

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

Über den Autor

Matthias Rosezky, Diplomand in Wien, ist spezialisiert auf Strahlungsphysik und im tiefsten Herzen ein Tüftler und Bastler. Er liebt es, zu programmieren und mit Elektronik zu arbeiten, um interessante Dinge zu schaffen und bestehende Designs zu verbessern. In einigen seiner anderen Projekte [4] hat Matthias diese Philosophie aufgegriffen und erschwingliche Open-Source-Hardware zur Strahlungsdetektion entwickelt.

Passende Produkte

- **MightyOhm Geigerzähler-Kit (inkl. Gehäuse)**
  www.elektor.de/18509
- **Elektromagnetischer Strahlungstester WT3122**
  www.elektor.de/20521

WEBLINKS

[1] Einsetzen von Thorianit in meine selbstgebaute Peltier-Nebelkammer: https://youtu.be/BRjhhpcfuWA
[2] Elektor-Labs-Seite für dieses Projekt: https://elektormagazine.de/labs/peltier-cloud-chamber
[3] Das Projekt auf Hackaday: https://hackaday.io/project/192146-peltier-cloud-chamber
[4] Weitere Projekte dieses Autors: https://nuclearphoenix.xyz

SparkFun Thing Plus Matter

Ein vielseitiges Matter-basiertes IoT-Entwicklungsboard

Von Saad Imtiaz (Elektor)

Das Board Thing Plus Matter MGM240P von SparkFun betritt die IoT- und Smart-Home-Arena mit robusten Funktionen für Entwickler und Hobbyisten.

Früher bekannt als das Project CHIP (Connected Home over IP) ist Matter [1] ein Protokoll, das die Interoperabilität zwischen Smart-Home- und IoT-Geräten ermöglicht. Dies macht das Board *Thing Plus Matter* von SparkFun [2] zu einer aufregenden Ergänzung der IoT-Entwicklungslandschaft.

Das Board (**Bild 1**) zeichnet sich durch seine kompakten Abmessungen von 5,84 cm × 22,9 cm aus und ist kompatibel mit dem Feather-kompatiblen Thing-Plus-Formfaktor. Sie können Software für das MGM240P-Board mit dem Simplicity Studio Debugging-Tool entwickeln, das unter Windows, Mac OSX und Linux Ubuntu arbeitet. Der Leitfaden [3] von SparkFun enthält detaillierte Hardware-Informationen und eine Schritt-für-Schritt-Anleitung, die Ihnen den Einstieg in die Simplicity-Studio-IDE erleichtert.

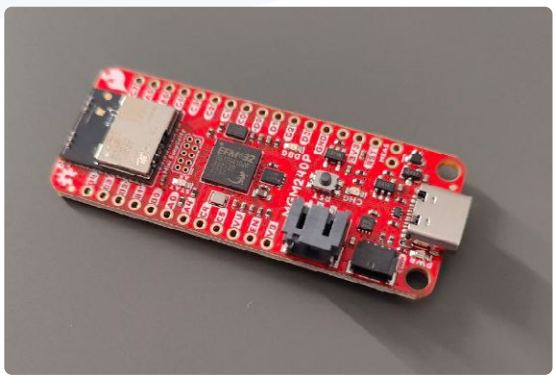


Bild 1. SparkFun-Board Thing Plus Matter MGM240P

Dies sind die Highlights des SparkFun-Boards Thing Plus Matter MGM240P:

- MGM240P-Funkmodul [4]
- Drahtlos: 802.15.4-Wireless-Protokolle (Zigbee und Open Thread) und Bluetooth Low Energy 5.3
- Matter-ready
- Silicon-Labs-SoC EFR32MG24 [5]
- 1.536 KB Flash-Speicher, 256 KB RAM
- Steckplatz für microSD-Karte
- Zwei Headerreihen mit 21 GPIO Anschlüssen
- 4-poliger Qwiic-Verbinder (JST)
- Mikrocontroller EFM32GG12B410F1024GL120-A als J-Link-Programmer und Debugging-IC
- Unbestückter Mini-Simplicity-Anschluss für einen externen Debugger
- 2-poliger JST-Anschluss für einen LiPo-Akku (nicht im Lieferumfang enthalten) mit einem LiPo-Lader MC73831 und einem LiPo-Zellen-Kapazitätsanzeige-IC MAX17048
- Stromaufnahme 15 µA, wenn sich der MGM240P im Low-Power-Modus befindet

Unterstützung des Matter-Protokolls

Das Board wurde zur Unterstützung des Matter-Protokolls konzipiert, was einen erheblichen Vorteil für alle darstellt, die Smart-Home- oder IoT-Geräte entwickeln möchten, die nahtlos mit anderen Matter-kompatiblen Geräten kommunizieren sollen. Die Fähigkeit des Protokolls, verschiedene IoT-Ökosysteme zu verbinden, ist ein Wendepunkt in der IoT-Entwicklung.

Drahtlose Konnektivität und Tests

Dieses Board ist mit einem MGM12P-Funkmodul aus der EFR32MG12-Familie von Silicon Labs ausgestattet. Dieses Modul unterstützt mehrere Drahtlos-Protokolle, darunter Bluetooth Low Energy (BLE) und IEEE 802.15.4, die Grundlage von Thread und Matter (**Bild 2**). Im Simplicity Studio von SparkFun sind

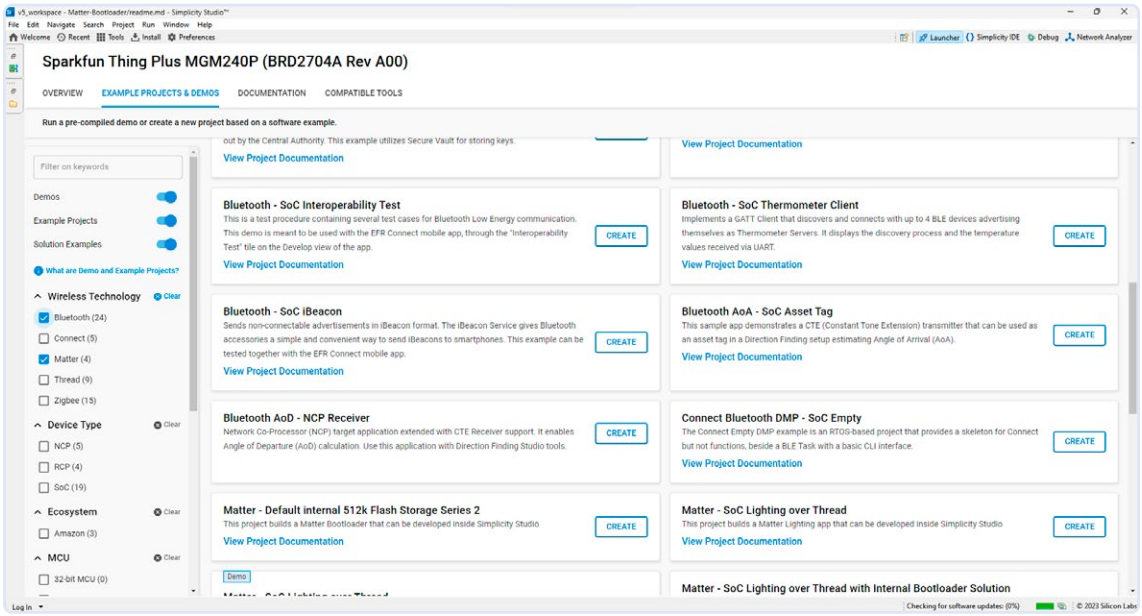


Bild 2. Dieses Modul unterstützt verschiedene Funkprotokolle wie Bluetooth Low Energy (BLE) und IEEE 802.15.4 (die Basis von Thread und Matter), von denen ich einige getestet habe.

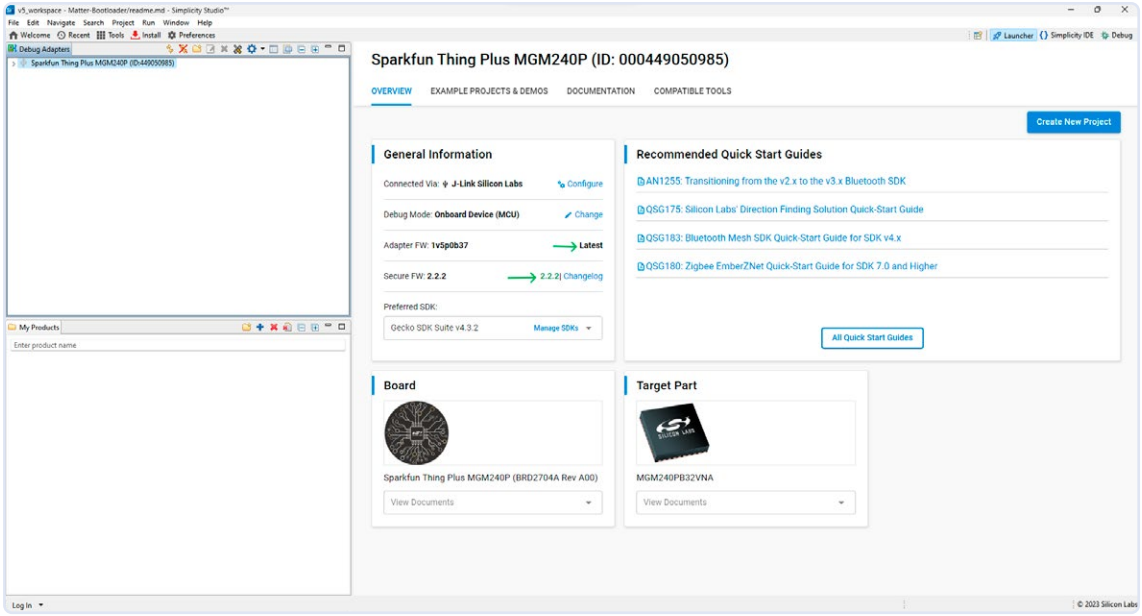


Bild 3. Das Flashen dieses Codes auf das Board ist ganz einfach: Installieren Sie Simplicity Studio, schließen Sie Ihr Board an, und es wird automatisch erkannt.

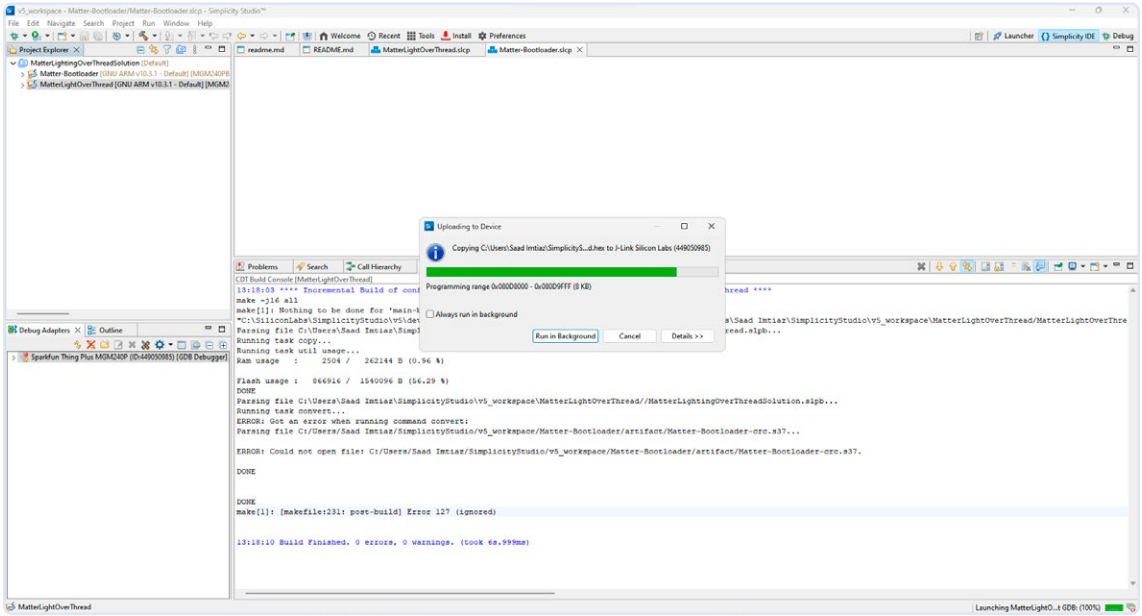


Bild 4. Ich wählte das Beispiel Matter Light Over Thread, kompilierte die Firmware und übertrug sie dann auf die Platine.

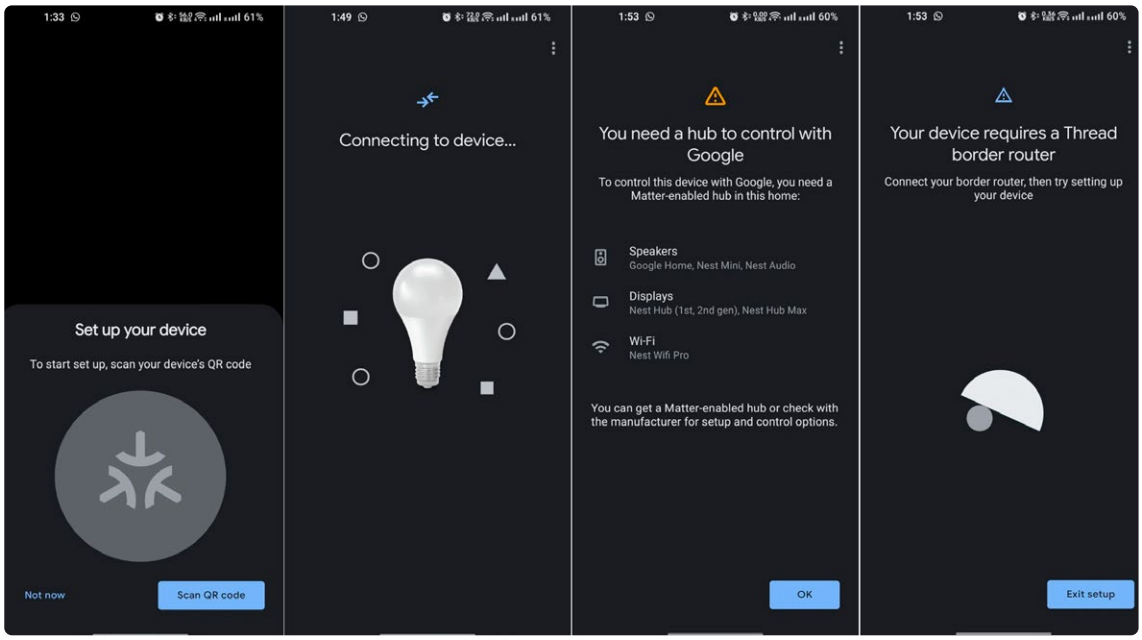


Bild 5. Bei der Einrichtung mit der Google-Home-App erkennt das Smartphone das Board kurz nach dem Flashen des Codes. Für die Steuerung von Matter-Geräten mit Google ist jedoch ein Google Hub erforderlich.

zahlreiche Code-Beispiele für dieses Board enthalten. In meinen Tests habe ich einige dieser Beispiele ausprobiert, darunter auch das Beispiel *Matter Light Over Thread*, das im Grunde eine Matter-Beleuchtungs-App ist.

Das Aufspielen des Codes auf dieses Board ist einfach: Zuerst installieren Sie Simplicity Studio, dann schließen Sie ihr Board an und es wird automatisch erkannt. Sorgen Sie dafür, dass Sie die Firmware des Boards vor der Verwendung aktualisieren, da es immer gut ist, das Board auf dem neuesten Stand zu halten. Damit werden Fehler und Probleme früherer veralteter Firmware-Versionen vermieden (**Bild 3**).

Nach dem Aktualisieren der Firmware klicken Sie einfach auf *Create New Project* und erhalten eine Unmenge von Beispielen, die zum Großteil auch für dieses Board geeignet sind. Ich habe das Beispiel *Matter Light Over Thread* ausgewählt, die Firmware erstellt und auf das Board hochgeladen (**Bild 4**).

Für *Google Home Developer* [6] ist ein Google-Konto erforderlich. Wenn Sie den Code auf Ihr Board hochladen, erkennt Ihr Smartphone es automatisch als Matter-Gerät und fordert sie auf, es zu pairen. Ich konnte den Anweisungen auf der SparkFun-Seite [7] zwar folgen, aber leider benötigt man einen *Google Hub*, um Matter-Geräte mit Google zu steuern. Wie im **Bild 5** gezeigt, steht dort: *Your device requires a Thread border router*. Ich konnte mich deshalb nicht weiter damit beschäftigen, aber ich bin sehr zuversichtlich, dass es mit einem Hub funktioniert hätte.

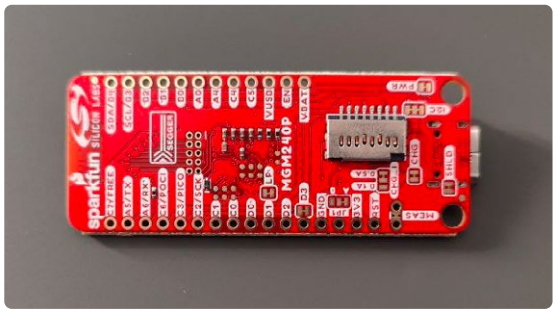


Bild 6. Das Board unterstützt mehrere Optionen zur Stromversorgung. Es enthält Funktionen, die die Mobilität von IoT-Geräten verbessern.

Flexible Stromversorgung

Das Board (siehe **Bild 6**) erlaubt verschiedene Varianten der Stromversorgung; über USB, eine Li-Po-Batterie oder eine externe Stromquelle. Es verfügt über PTH-Pins, die mit den 3,3 V, V_USB- und V_BATT verbunden sind. Das Board ist mit einem 2-poligen JST-Anschluss für eine Einzelzellen-LiPo-Batterie versehen, was ideal für batteriebetriebene mobile Anwendungen ist. Auf dem Board sorgt ein 3,3-V-Spannungsregler für einen stabilen Betrieb der Elektronik. Zusätzlich besitzt das Board zwei wichtige Bauteile, nämlich den Einzelzellen-LiPo-Lader MCP73831, der das Laden des angeschlossenen Akkus über USB-C ermöglicht und die Einzelzellen-Batteriekapazitätsanzeige MAX17048, die kontinuierlich den Ladezustand der Batterie überwacht. In meinen Tests hat das sehr gut funktioniert. Dies macht das Board besonders nützlich, da ja die meisten IoT-Geräte drahtlos sein sollten, um mobil zu sein.

Standardmäßig ist der Ladestrom auf 500 mA konfiguriert. Es gibt jedoch einen Dreifach-Jumper mit der Bezeichnung CHG, der es dem Anwender ermöglicht, zwischen 500 mA oder 100 mA Ladestrom oder der vollständigen Deaktivierung des Laderegler-ICs zu wählen. In meinen Tests funktionierte die Standard-Ladeschaltung ohne Probleme. Diese Funktion ermöglicht es auch, ein Projekt kompakter zu gestalten, da kein zusätzliches Lademodul eingebaut werden muss.

Erweiterung per Qwiic

Eines der hervorragenden Merkmale dieses Boards ist seine Qwiic-Kompatibilität, dem Plug-and-Play-System von SparkFun für den Anschluss verschiedener Sensoren und Peripheriebausteine (siehe **Bild 7**). Es vereinfacht den Hardware-Aufbau und erleichtert das Hinzufügen zusätzlicher Komponenten zu IoT-Projekten.

Debugger

Dieses Board ist mit dem Mikrocontroller EFM32GG12B410F1024GL120-A ausgestattet, der als J-Link Programmer und Debugger dient. Es verfügt über einen Mini-Simplicity-Verbinder für die Verwendung eines externen Debuggers. Standardmäßig ist der Debugger auf den Standardmodus eingestellt, wobei der Debugger-WAKE-Pin über den LP-Jumper mit V_USB verbunden ist. Der Anwender hat jedoch die Möglichkeit, diesen Jumper zu entfernen, um den Debugger in den Low-Level-Modus zu versetzen.
Dieser Debugger bietet leistungsstarke Low-Level-Debugging-Funktionen in Kombination mit dem Debugging-Tool von Simplicity Studio. Sie können eine Vielzahl der üblichen Debugging-Aufgaben durchführen, einschließlich Debugger-Ausgabe, Setzen von Code-Breakpoints und sogar die Analyse von Assembler-Code für detailliertere Untersuchungen.

Interoperabilität

Eines der Hauptziele des Matter-Protokolls ist Herstellung von Interoperabilität zwischen verschiedenen IoT-Geräten, unabhängig von deren Herstellern. Mit dem Board *SparkFun Thing Plus Matter* sind Sie in der ausgezeichneten Lage, um Geräte entwickeln zu können, die sich nahtlos in bestehende und zukünftige Matter-basierte IoT-Ökosysteme integrieren lassen.

Anpassung und Prototyping

Für diejenigen, die ihre IoT-Lösungen den eigenen Vorstellungen anpassen möchten, bietet das Board die Möglichkeit, Header einzulöten und verschiedene Einstellungen zu konfigurieren. Darüber hinaus bietet das Qwiic-Ökosystem von SparkFun eine Vielzahl von Sensoren und Peripheriebausteine, mit denen Sie Ihre Projekte erweitern können.

Ideen für künftige Projeke

Das Board Thing Plus Matter (MGM240P) von SparkFun ist mit seinen umfangreichen Funktionen und der Unterstützung des Matter-Protokolls ideal für die Entwicklung verschiedenster IoT-Geräte. Erforschen Sie Projektideen von Smart-Home-Geräten bis hin zu Umweltsensoren und nutzen Sie die Fähigkeiten des Boards, um Ihre innovativen Ideen zum Leben zu erwecken.
Das Board Thing Plus Matter (MGM240P) von SparkFun ist ein leistungsstarkes IoT-Entwicklungsboard, das speziell für Matter-basierte IoT-Lösungen entwickelt wurde. Seine Kompatibilität mit mehreren drahtlosen Protokollen, die Qwiic-Anschlüsse, die Akkuladefunktion und die umfangreichen Ressourcen von SparkFun und Silicon Labs ermöglichen es Entwicklern, eine Vielzahl von IoT-Herausforderungen zu meistern. Ganz gleich, ob Sie ein neues IoT-Projekt starten oder ein bestehendes aufrüsten möchten, dieses Board bietet eine hervorragende Grundlage für Ihre IoT-Entwicklung.

SE — 240251-02

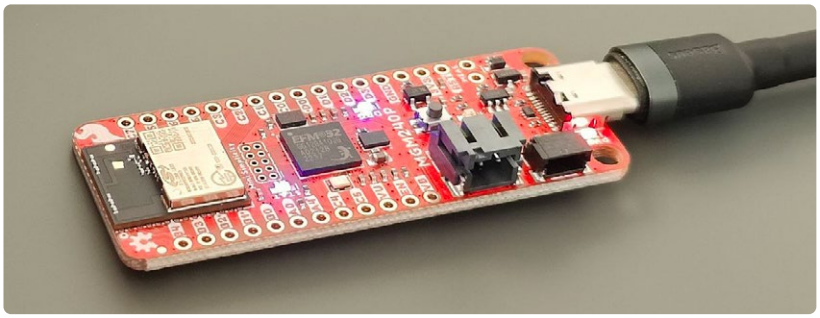


Bild 7. Dieses Board besitzt Qwiic-Verbinder, was die Kompatibilität mit und das Hinzufügen von Komponenten zu Ihren IoT-Projekten vereinfacht.

Haben Sie Fragen oder Kommentare?

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

Passendes Produkt

- **SparkFun Thing Plus Matter (MGM240P)**
  www.elektor.de/20442

WEBLINKS

[1] Was ist Matter?: https://developers.home.google.com/matter/overview
[2] Thing Plus Matter (MGM240P) von SparkFun: https://elektor.com/20442
[3] MGM240P-Anleitung: https://tinyurl.com/mgm240phookup
[4] Wireless-Modul MGM240P: https://tinyurl.com/mgm240modules
[5] Silicon Labs kündigt BG24 und MG24 an: https://tinyurl.com/bg24mg24news
[6] Google-Home-Entwicklerkonsole: https://console.home.google.com/projects
[7] Anleitung zum Anschluss von Thing Plus Matter an Google Nest Hub: https://tinyurl.com/thing-plus-matter-to-google

IoT-Retrofitting

RS232-Geräte fit für Industrie 4.0 machen

Von Matthias Lay (Würth Elektronik eiSos)

Im Industrieumfeld findet man häufig ältere Maschinen, die immer noch zuverlässig funktionieren, aber leider über keine Schnittstelle zu einem industriellen Bussystem verfügen. IoT-Retrofitting kann hier eine elegante Lösung sein, wie das Beispiel eines Schweißautomaten mit RS232-Schnittstelle zeigt.

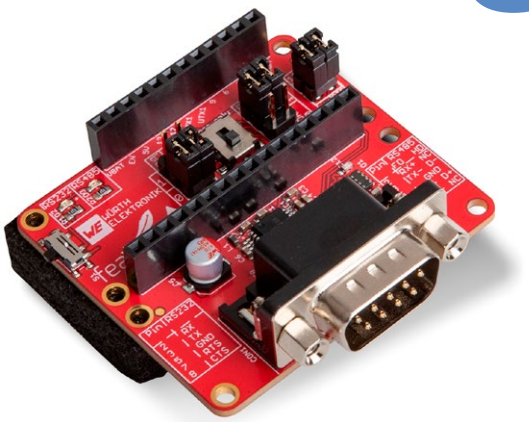


Bild 1. Serial-Bridge-FeatherWing von Würth Elektronik für die physikalischen Kommunikationsstandards RS232, RS422 und RS485.

Die zunehmende Digitalisierung in der Industrie, häufig auch synonym Industrie 4.0 genannt, und die Verbreitung des Industrial Internet of Things (IIoT) erfordert eine immer weiter voranschreitende Vernetzung von Maschinen in Produktionsstätten. Die Vorteile der Vernetzung im industriellen Umfeld sind bereits bekannt [1], und die Bedeutung des Themas für den zukünftigen Erfolg von Unternehmen ist unbestritten. Trotzdem treten bei der konkreten Implementierung immer wieder Schwierigkeiten auf. Einer Umfrage der Vogel Communications Group zufolge ist unter anderem die Schnittstellenproblematik eine Herausforderung bei der Umsetzung von IIoT-Projekten. Sie beschreibt das Problem, dass Produktionsmaschinen durch eine fehlende Standardisierung der Protokolle und Schnittstellen, oder das gänzliche Fehlen eben jener, nicht in der Lage sind, mit anderen Systemen zu kommunizieren. Ein Ansatz dieses Problem zu lösen ist es, der Greenfield-IIoT-Strategie zu folgen und sämtliche Maschinen durch neue IIoT-fähige Maschinen mit einheitlichen Schnittstellen zu ersetzen. Das ist jedoch offensichtlich weder ökologisch noch ökonomisch sinnvoll [2].

Sinnvoller ist es, die Brownfield-IoT-Strategie zu nutzen und bestehende Maschinen mit IoT-Schnittstellen nachzurüsten, also einen sogenannten Retrofit durchzuführen. Die Nachrüstung bietet für Bestandsmaschinen einen ungemeinen Mehrwert. Dieser beginnt mit der Möglichkeit, Kennzahlen der Gesamtanlageneffektivität in Echtzeit zugänglich zu machen, führt darüber hinaus dazu, über einen tieferen Einblick in den Produktionsprozess Optimierungspotentiale der Maschinen erkennen und realisieren zu können und erlaubt endlich mit KI-gestützten Analysemethoden eine vorrausschauende Wartung, welche Ausfälle der Maschine hinreichend genau vorhersagen kann.

Dieser Artikel beschreibt einen Proof of Concept (PoC) für das Retrofitting einer bestehenden Produktionsanlage, die bereits über eine RS232-Schnittstelle verfügt. Hierbei kommt der Ansatz des Rapid-Prototyping zum Einsatz.

Rapid-Prototyping mit Feather-Modulen

Feather- und FeatherWing-Module eignen sich hervorragend für das Rapid-Prototyping in Retrofitting-Projekten. Die Evaluierungsboards sind modular aufgebaut und aufeinander steckbar. Durch das festgelegte Pinning der Feather-Plattform von Adafruit und die Möglichkeit, mehrere Boards zu stapeln, lassen sich mit einem beliebigen Mikrocontroller weitere Funktionen und Schnittstellen hinzufügen. Somit können innerhalb kürzester Zeit funktionsfähige Hardware-Prototypen aufgebaut und verschiedene Konfigurationen ausprobiert werden.

Würth Elektronik hat eine Reihe solcher Evaluierungsboards im Angebot – Open Source und vollständig kompatibel mit dem Feather-Formfaktor. Diese Boards ermöglichen neben der Nutzung von Sensoren und unterschiedlichsten Funkprotokollen (WiFi/Mobilfunk) und dem Betreiben des Prototyps mit unterschiedlichen Versorgungsspannungen auch das Hinzufügen von industriellen Schnittstellen. Es gibt ein GitHub-Repository [3] für alle Open-Source-Boards, einschließlich ihrer Schaltpläne, BoMs, Software und Beschreibungen zur Cloud-Anbindung für Azure und AWS.

Auslesen der RS232-Schnittstelle

Bei Retrofitting-Projekten ist die Nutzung existierender, nicht IoT-fähiger Schnittstellen der Maschine besonders vorteilhaft. Hierdurch können Maschinenparameter erfasst und analysiert werden, ohne dass eine Nachrüstung von externen Sensoren zwangsläufig ist. Eine der am häufigsten anzutreffenden Schnittstellen an Bestandsmaschinen im industriellen Umfeld ist die RS232-Schnittstelle. Diese ist eine sehr

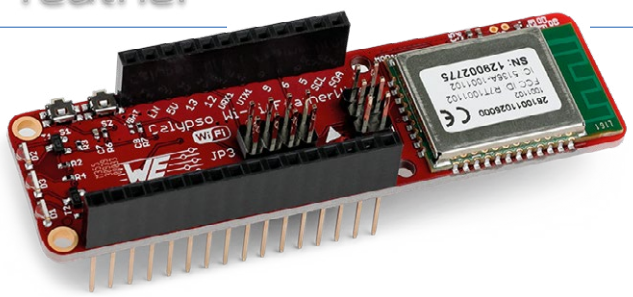


Bild 2. Mit dem Calypso-FeatherWing von Würth Elektronik lässt sich WiFi-Funkkommunikation einfach implementieren.

robuste und trotzdem eine der unkompliziertesten Kommunikationsschnittstellen. Ein grundlegendes Verständnis von Baudrate sowie Daten-, Parity- und Stopbits ermöglicht in der Regel eine erfolgreiche Datenübertragung.

Um eine RS232-Schnittstelle mit dem Mikrocontroller des Retrofit-Prototyps zu verbinden, wird ein Serial-Bridge-FeatherWing von Würth Elektronik (**Bild 1**) verwendet. Dieser wandelt eine serielle in eine UART-Schnittstelle um, welche für Mikrocontroller nutzbar ist. Durch die softwareseitige Konfiguration der UART-Schnittstelle des Mikrocontrollers kann diese gängige Kommunikationsprotokolle wie MODBUS oder ASCII abbilden, und durch die Konfiguration der Serial Bridge können die physikalischen Kommunikationsstandards RS232, RS422 sowie RS485 in den Betriebsarten Half- und Full-Duplex abgebildet werden.

IoT-Anbindung per WiFi/MQTT

Um Retrofit-Prototypen in ein bestehendes IoT-System einzubinden, wird eine geeignete IoT-Schnittstelle benötigt. Als Protokoll für die Schnittstelle lässt sich das MQTT-Protokoll verwenden. Dieses ist eines der am weitesten verbreiteten Protokolle im IoT. Es bietet sich durch seine Einfachheit und geringe Bandbreitenanforderung besonders an, um Daten von vielen unterschiedlichen Maschinen zu empfangen [4]. Vorteilhaft ist, dass es auf jegliche TCP-Kommunikation aufgesetzt werden kann.

Durch das Feather-Ökosystem können, abhängig von den Anforderungen an den Retrofit, unterschiedliche Übertragungsverfahren genutzt werden; eine kabelgebundene Übertragung zum Beispiel, durch die Verwendung eines Ethernet-FeatherWings oder eine kabellose Übertragung mittels eines Funkmoduls aus der FeatherWing-Familie.

Da der Prototyp möglichst schnell realisiert und getestet werden sollte, fiel die Wahl auf den Calypso-WiFi-FeatherWing von Würth Elektronik (**Bild 2**). Dieser kann über bestehende Access Points problemlos in das existierende lokale Netz einer Produktionsstätte eingebunden werden (**Bild 3**) und erspart das zusätzliche Verlegen von Ethernet-Kabeln.

Implementierung

Als konkretes Beispiel für die Realisierung eines Prototypen dient die Digitalisierung der Schweißeinheit einer Produktionslinie (**Bild 4**). Im Vorhinein wurde dieser Teil der Produktionslinie als am sinnvollsten für ein Retrofitting erachtet. Der Grundgedanke ist, dass durch ein Auswerten der Schweißparameter Rückschlüsse auf den aktuellen Zustand der Thermode möglich sind. Diese Informationen können dem Wartungspersonal Hinweise geben, ob eine Thermode Gefahr läuft, während der nächsten Produktionsschicht ihre Verschleißgrenze zu erreichen. Auf diese Weise kann die Thermode bereits vor dem tatsächlichen Ausfall ausgetauscht werden, um einem ungeplanten Stillstand der Produktionslinie vorzubeugen.

Entsprechend der nachzurüstenden Maschine wird der Serial-Bridge-FeatherWing im RS232-Modus konfiguriert und

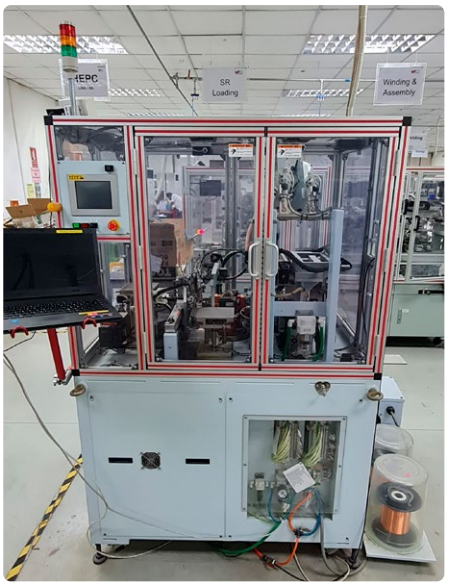


Bild 4. Maschine mit Schweißmodul für den Retrofit.

softwareseitig auf die Baudrate und die Konfiguration des Schweißinverters angepasst. Dieser überträgt nach jeder durchgeführten Schweißung sämtliche erfassten Parameter des Schweißpunktes mittels des ASCII-Protokolls.

Die Schweißdaten werden innerhalb der Maschine bereits mittels RS232 vom Schweißinverter an die SPS übertragen, um ein lokales Qualitätssicherungsprotokoll

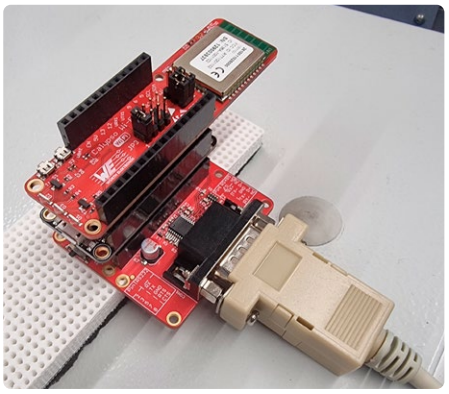


Bild 5. Gestapelte FeatherModule: oben das WiFi-Modul Calypso, in der Mitte der M0-Mikrocontroller und unten die Serial-Bridge.

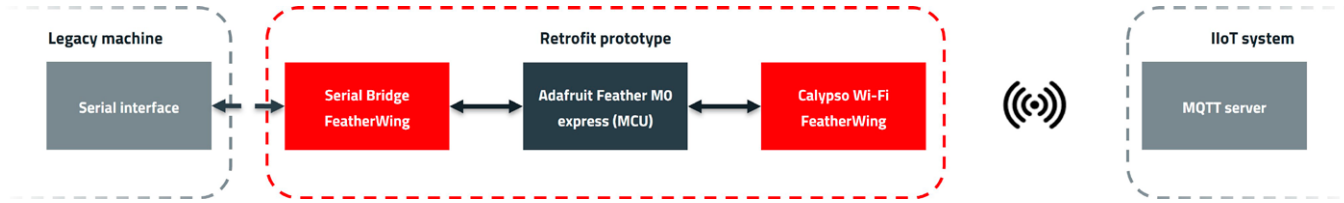


Bild 3. Datenflussdiagramm der Anbindung einer Bestandsmaschine an das bestehende IIoT-System.

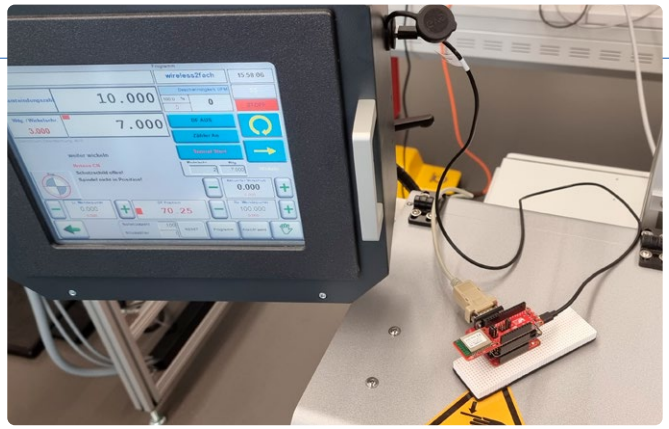


Bild 6: Baugleicher Prototyp als Retrofit an einer Wickelmaschine.

zu erstellen. Um diese Datenübertragung mitzulesen, ist hardwareseitig ein Herausführen des SPS-seitigen RS232-Empfängersignals und des RS232-Referenzpotentials notwendig. Diese herausgeführten Signale und das Referenzpotential werden nun mit dem Serial-Bridge-FeatherWing verbunden. Der Prototyp liest nun kontinuierlich die RS232-Übertragungen mit und bringt die übertragenen Daten bereits in das vom IoT-System geforderte Datenformat.
Anschließend leitet der Mikrocontroller die Daten an das WiFi-FeatherWing Calypso weiter. Dieses ist mit dem lokalen IoT-Netzwerk verbunden und kann die Daten somit direkt in das IoT-System einspeisen.
Die jetzt im IoT-System (**Bild 5**) bereitgestellten Daten können dazu verwendet werden, um in Echtzeit einen Einblick in die Schweißparameter und Auslastung der Produktionslinie zu erhalten und im nächsten Schritt eine Datenanalyse anhand der Schweißparameter durchzuführen.

Prozessverbesserung dank Analyse

Der Prototyp funktioniert und ermöglicht eine kontinuierliche Überwachung der Maschinenauslastung, der Taktzeit und der Schweißparameter. Dadurch ist auch ein Abschätzen des Thermodenzustands möglich.
Durch die Analyse der aufgezeichneten Daten konnte bereits festgestellt werden, dass die Anpassung einiger Schweißparameter vorteilhafte Auswirkungen auf die Lebensdauer der Thermode hat. Beispielsweise wurde ersichtlich, dass durch Absenken der Schweißleistung und Verlängern der Schweißdauer eine deutlich längere Lebensdauer der Thermode erreicht werden kann, und dies bei nur minimaler Erhöhung der Taktzeit.
Ganz ohne Probleme lief die Implementierung des Retrofit-Prototyps allerdings nicht ab. Da der Schweißinverter im Sockel der Maschine untergebracht ist und dieser Sockel vollständig mit Metallpanelen verkleidet ist, wirkt dieser wie ein Faradayscher Käfig. Darum musste der Protoyp außerhalb des Sockels montiert werden, um eine stabile Funkverbindung zum Access Point sicher zu stellen.
Außerdem ist im aktuellen Aufbau der Datenfluss nur in eine Richtung möglich, nämlich vom Schweißinverter in das IoT-System. Der Grund hierfür ist, dass eine RS232-Kommunikation nur zwei Datenleitungen verwendet und es pro Datenleitung nur einen aktiven Transmitter geben darf. Um zukünftig Änderungen der Schweißparameter mittels des IoT-Systems vornehmen zu können, muss die Kommunikation bidirektional möglich sein. Hierfür ist eine Erweiterung des Prototyps um einen weiteren Serial-Bridge-FeatherWing angedacht. Damit würde es möglich, die Kommunikation nicht nur mitzulesen, sondern auch Parameteranpassungen an den Schweißinverter zu senden.

Der anhand eines Schweißinverters vorgestellte Prototyp lässt sich vielfältig einsetzen: Ein Beispiel dafür ist der baugleiche Prototyp als Retrofit an einer Wickelmaschine (**Bild 6**).

FeatherWing als Baukasten für das Prototyping

Abschließend bleibt zu sagen, dass durch die Nutzung des Feather-Formfaktors massiv Entwicklungszeit bei der Erstellung des Hardwareprototypen eingespart werden konnte. Statt eine Platine mit sämtlichen benötigten Komponenten zu entwerfen, diese aufzubauen und in Betrieb zu nehmen, werden die Evaluierungsplatinen im Feather-Format einfach nach benötigter Funktion bestellt und zusammengesteckt. So kann sehr schnell mit der Softwareentwicklung begonnen werden und bei unvorhergesehen Problemen oder Anforderungsänderungen kann der Hardwareprototyp deutlich leichter angepasst werden. Kurzum, die Hardwareentwicklung wird durch das Nutzen des Feather-Formfaktors agiler und beschleunigt so den gesamten Prototyping-Prozess erheblich.

240280-02

Über den Autor

Matthias Lay studierte an der Hochschule Heilbronn Elektrotechnik mit dem Schwerpunkt Automatisierungstechnik und schloss dort mit MSc. ab. Seit 2019 ist er bei Würth Elektronik eiSos angestellt zunächst in den Bereichen Hard- und Softwareentwicklung. Seit 2023 arbeitet er dort als IoT-Systemingenieur mit den Schwerpunkten IIoT und Retrofitting.

WEBLINKS

[1] Sniderman, B. et. al., „Industry 4.0 and manufacturing ecosystems — Exploring the world of connected enterprises", Deloitte: https://tinyurl.com/5745usv8
[2] Pietrangeli, I. et al., „Smart Retrofit: An Innovative and Sustainable Solution", MDPI: https://mdpi.com/2075-1702/11/5/523
[3] GitHub-Repository: https://github.com/WurthElektronik/FeatherWings
[4] Ries, U., „MQTT-Protokoll: IoT-Kommunikation von Reaktoren und Gefängnissen öffentlich einsehbar", Heise Security: https://tinyurl.com/388yrykb

MICROCHIP

IoT mit 8-Bit-Mikrocontrollern

Von Joshua Bowen (Microchip Technology)

Seit ihrer Einführung in den 1970er Jahren sind Mikrocontroller ein wesentlicher Bestandteil bei der Steuerung verschiedener Automotive-, Consumer- und Industrieprodukte. Ihr Einsatzbereich hat sich inzwischen auf tragbare funkbasierte Geräte und tragbare Internet-of-Things-Anwendungen ausgeweitet. Darüber hinaus hat der Gesundheitssektor ein beträchtliches Wachstum erfahren, wobei 8-Bit-Mikrocontroller in zahlreiche Embedded-Designs integriert wurden.

Embedded-Systeme, die 8-Bit-Mikrocontroller (MCUs) verwenden, müssen leistungsfähig und wirtschaftlich tragfähig sein, wobei die Fertigungsmengen oft Hunderttausende oder sogar Millionen von Einheiten pro Anwendung erreichen. Im Fahrzeugbereich steuern 8-Bit-MCUs zahlreiche Subsysteme, von elektrisch verstellbaren Sitzen und Fensterhebern bis hin zu intelligenten Türgriffen und Reifendrucksensoren. Bei dieser großflächigen Nutzung sind selbst geringe Kostenunterschiede entscheidend. Die Wartungskosten für Millionen von Geräten, die bei der Entwicklung oft übersehen werden, lassen sich durch erhöhte Zuverlässigkeit und Langlebigkeit, einfacheren Code sowie durch Hardwareverbesserungen senken. Dies verringert den Bedarf an Software-Redundanzen.

Die anhaltende Beliebtheit von 8-Bit-MCUs ist auf ihre kontinuierlichen Neuerungen in den Bereichen Speicher, Stromaufnahme, Gehäuse und Kern-unabhängige Peripherie (Core-Independent Peripherals, CIPs) zurückzuführen.

Erhebliche Verbesserungen bei 8-Bit-MCUs

Mit dem Aufkommen des IoT und der Umgestaltung der Städte mit intelligenten Systemen ist skalierbare Intelligenz für viele Branchen unverzichtbar geworden. Einrichtungen wie intelligente Straßenlaternen und individuelle Parkplatzdetektoren erfordern MCUs, die Daten sammeln, verarbeiten und kommunizieren. Häufig können diese Aufgaben von einer 8-Bit-MCU mit einem integrierten A/D-Wandler (ADC) bewältigt werden, während der Prozessoror-Kern in einem stromsparenden Zustand verbleibt. Dieser Ansatz eignet sich für intelligente Parkhäuser, vernetzte Straßenlaternen und automatisierte Landschaftspflege in der Stadt, wo Energieeffizienz entscheidend ist.

Die Vorteile kleinerer Bausteine liegen in ihrem geringeren Stromverbrauch und ihrer kompakten Größe, so dass sie in tragbare IoT-Produkte mit begrenztem Platzangebot passen.

Bei der Entwicklung neuerer MCUs wurde auf Kosteneffizienz geachtet. Sie bieten die erforderlichen Funktionen und schonen das Budget des Nutzers. Darüber hinaus haben die Fortschritte in der Speichertechnologie die Fähigkeiten moderner MCUs erheblich gesteigert.

Speicher

Moderne MCUs haben sich deutlich weiterentwickelt, vor allem durch die Fortschritte beim Flash-Speicher. Heutige Anwendungen erfordern komplexere Programme, was einen höheren Speicherbedarf in MCUs mit sich bringt. Die Embedded-Flash-Speicher in diesen Bausteinen sind langlebig und halten strengen Tests im Fahrzeugbereich sowie zahlreichen Schreib- und Löschzyklen stand. Aktuelle 8-Bit-MCUs bieten einen Speicher von 384 Bit bis über 128 KB und eignen sich damit für eine Vielzahl von Anwendungen.

Stromaufnahme

Die Energieeffizienz ist für 8-Bit-MCUs zu einem wichtigen Thema geworden, insbesondere bei batteriebetriebenen Anwendungen. So verfügen die

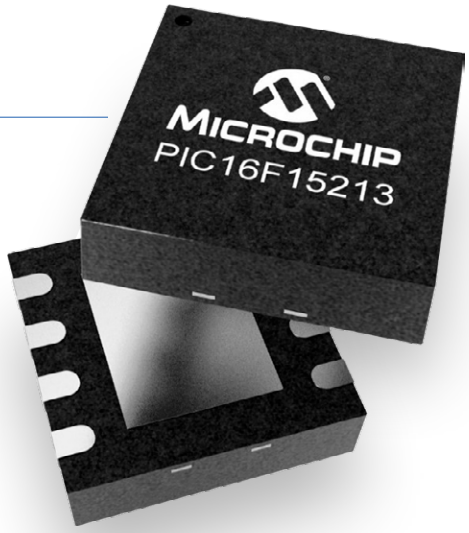


Bild 1. Viele neue PIC- und AVR-MCUs bieten eine Vielzahl von Gehäusebauformen bis hin zu 3 mm×3 mm VQFN für platzbeschränkte Anwendungen.

nanoWatt XLP eXtreme Low Power PIC-MCUs [1] über Systemüberwachungskreise, die auf minimalen Stromverbrauch ausgelegt sind. Dies führt zur branchenweit niedrigsten Stromaufnahme im *Betriebs*- als auch im *Sleep*-Modus, in dem die Bausteine 90...99% ihrer Zeit verbringen. Techniken wie *Peripheral Module Disable* sorgen für weitere Stromeinsparungen, indem Peripherie vollständig von der Stromschiene und dem Taktbaum abgekoppelt wird. Zu den Vorteilen der nanoWatt-XLP-Technologie gehören:

- Sleep-Ströme unter 20 nA
- Brown-out-Reset bis zu 45 nA
- Watchdog-Timer mit nur 220 nA
- Echtzeituhr/Kalender mit weniger als 470 nA
- Betriebsströme bis zu 50 µA/MHz
- Volle Analog- und Selbstschreibfunktion bis zu 1,8 V

Da viele 8-Bit-MCUs in batteriebetriebenen Anwendungen zum Einsatz kommen, sind weitere Stromeinsparungen durch optimierte Peripherie möglich, auf die hier später eingegangen wird.

Gehäuse

Ein wichtiges Unterscheidungsmerkmal von 8-Bit-MCUs ist ihre Fähigkeit, in kleine Gehäuse zu passen, was ideal für platzbeschränkte Anwendungen in funkbasierten, tragbaren Produkten ist. Gehäuse wie das 8-polige SOIC oder 8-polige DFN und das gängige 20-polige VQFN (Very Thin Quad Flat Pack No-Leads) bieten kompakte Lösungen (**Bild 1**). Für komplexere Anwendungen sind größere Gehäuse wie 40-Pin-PDIP und 44-Pin-TQFP erhältlich.

Core-unabhängige Peripherie

CIPs (Core-Independent Peripherals) erweitern die Funktionen von MCUs, indem sie unabhängig vom Prozessor-Kern arbeiten, was für stromsparende und kostengünstige Designs von Vorteil ist. CIPs können verschiedene Aufgaben unabhängig voneinander erledigen, wodurch sich Eingriffe durch die CPU (Central Process Unit) verringern, was die Effizienz und Zuverlässigkeit des Systems erhöht. Dieser modulare Ansatz vereinfacht die Umsetzung von Touch-Schnittstellen, Sensordatenverarbeitung und mehr.

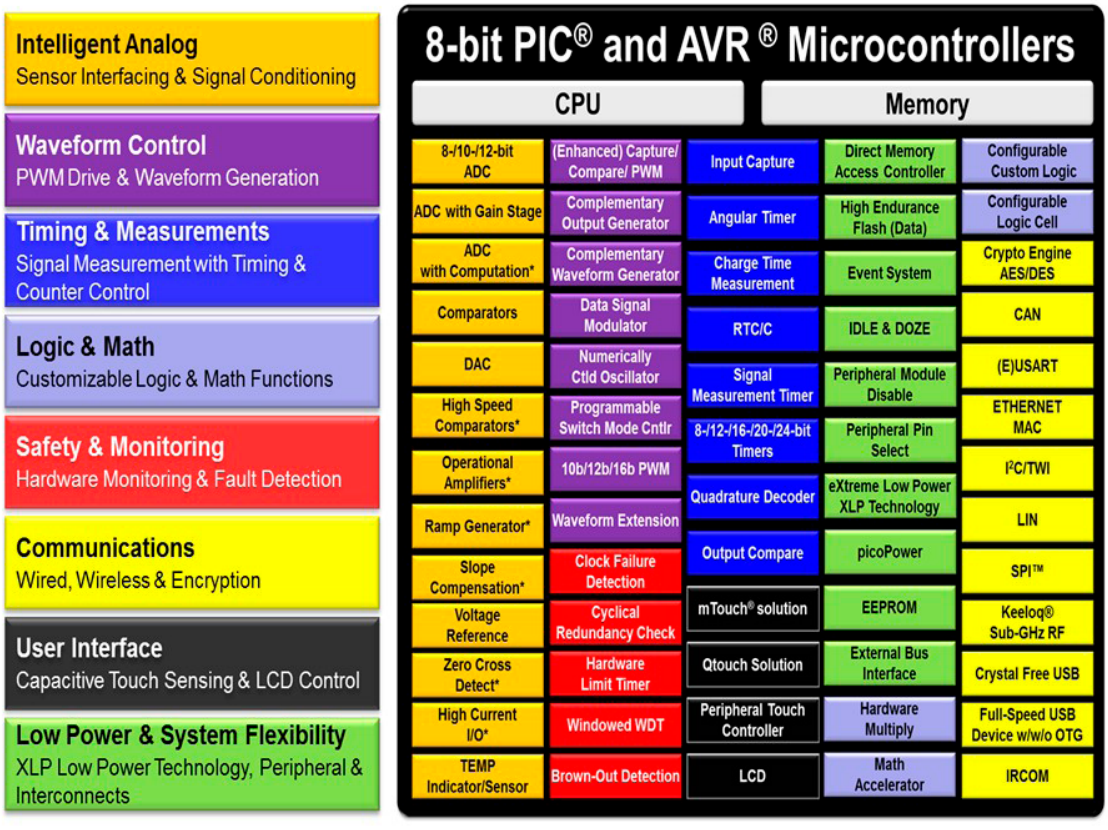


Bild 2. Core-unabhängige Peripherie eignet sich für zahlreiche 8-Bit-MCU-Designs.

Dieser Designansatz bietet ein vorgefertigtes Mittel, Ereignisse auf der Grundlage von Peripherie zu programmieren. Das *Event-System* kann so Ereignisse auf der Grundlage von GPIOs (General Purpose Input/Output) auslösen oder Interrupts auf mehreren Kanälen steuern.

Die derzeit verfügbaren CIPs für 8-Bit-PIC- und AVR-MCUs sind in **Bild 2** nach Peripheriekategorien farblich gekennzeichnet. Die acht Kategorien und ihre Unterkategorien decken die meisten der Funktionen ab, die von einem kostengünstigen Embedded-Controller erwartet werden. Die grünen Elemente enthalten zusätzliche Möglichkeiten zur Energieeinsparung.

CIPs bieten eine höhere Zuverlässigkeit, indem sie den Code-Overhead reduzieren. Mit Hardware umgesetzte Funktionen vermeiden potenzielle Softwarekonflikte. Zudem lassen sich durch das Zusammenschalten von Peripherie in der Hardware externe Verbindungen einsparen, was die Zuverlässigkeit des Endsystems erhöht. Diese erhöhte Zuverlässigkeit der Komponenten senkt die Kosten über die gesamte Lebensdauer des Projekts.

Die neuesten 8-Bit-MCU-Familien bieten umfangreiche Optionen in Bezug auf Speicher und Anzahl der Pins, so dass Entwickler mit größeren Bausteinen beginnen und bei der Optimierung des Codes auf kleinere skalieren können. Die PIC16F152XX-Reihe zum Beispiel ist für kostensensitive Sensor- und Echtzeit-Steuerungsanwendungen konzipiert und verfügt über einen 10-Bit-A/D-Wandler, PPS (Peripheral Pin Select), digitale Kommunikationsperipherie und Timer. Zu den Speicherfunktionen zählt MAP (Memory Access Partition), die Nutzer bei Datenschutz- und Bootloader-Anwendungen unterstützt.

Entwicklungstools für schnellere und vereinfachte Anwendungen

Fortschritte bei den Tools haben den Entwicklungsprozess gestrafft und den Bedarf an umfangreicher Codierung reduziert. Der *MPLAB Code Configurator* (MCC) [2] hilft dabei, kompakten, effizienten Code zu erstellen und verkürzt die Entwicklungszeiten erheblich. Das Curiosity-Nano-Evaluierungskit PIC16F15244 [3] (Teilenummer: EV09Z19A) zählt ebenfalls zu diesem Angebot und unterstützt neue Designs mit umfassenden Programmier- und Debugging-Funktionen (**Bild 3**).

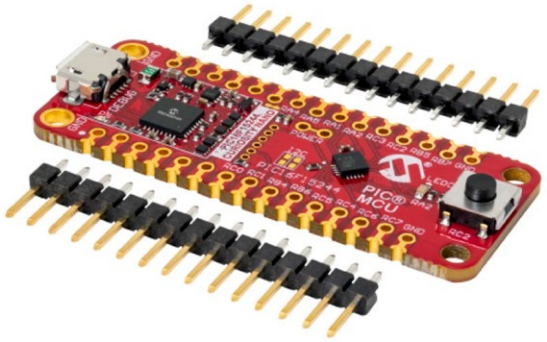


Bild 3. Das Curiosity-Nano-Evaluierungsboard PIC16F15244 und die beiden 1×15-poligen Stiftleisten im 100-mil-Raster im Curiosity-Nano-Evaluierungskit vereinfachen das Design.

Die integrierte Entwicklungsumgebung (IDE) MPLAB X bietet eine vielseitige Plattform für die Entwicklung von Code für 8-, 16- und 32-Bit-MCUs und erleichtert die Hardwaresimulation sowie die Integration mit Tools von Drittanbietern.

Eine vielversprechende (und kosteneffiziente) Zukunft

MCUs haben sich rasant weiterentwickelt, wobei 8-Bit-MCUs in Bezug auf Speicher, Stromverbrauch, Gehäuse und Peripherie führend sind. Sie unterstützen komplexe Anwendungen mit größerem Speicher und vereinfachen das Design, was die Entwicklungs- und Fertigungskosten senkt. Die Anpassungsfähigkeit und Effizienz heutiger 8-Bit-MCUs macht sie zur bevorzugten Wahl für zahlreiche IoT-Anwendungen und sichern eine vielversprechende und kostengünstige Zukunft für Embedded-Systeme.

Heutige 8-Bit-MCUs gehen weit über die reine Datenerfassung hinaus. Sie sind in der Lage, Daten in verschiedenen IoT-Anwendungen zu erfassen, zu verarbeiten und zu übertragen. Neuste 8-Bit-Modelle tragen der zunehmenden Komplexität dieser Anwendungen Rechnung, indem sie deutlich größere Speicherkapazitäten und verbesserte Peripherie bieten. Für kompakte und kosteneffiziente Designs, zum Beispiel Sensoren und grundlegende Echtzeit-Steuerungen, sind die optimierten Funktionen der 8-Bit-MCUs PIC16F152xx jedoch besonders vorteilhaft. Ausgestattet mit Core-unabhängiger Peripherie sind diese MCUs eine bevorzugte Wahl für viele Entwickler.

240306-02

WEBLINKS

[1] nanoWatt XLP eXtreme Low Power PIC MCUs: https://ww1.microchip.com/downloads/en/devicedoc/39941d.pdf
[2] MPLAB Code Configurator: https://microchip.com/en-us/tools-resources/configure/mplab-code-configurator
[3] Curiosity-Nano-Evaluierungskit PIC16F15244: https://microchip.com/en-us/development-tool/EV09Z19A

Technologie als Motor der Nachhaltigkeit

Neuerungen führen zu Energieeffizienz in vielen Applikationen

Von Mark Patrick (Mouser Electronics)

In der Elektronikindustrie ist der Wirkungsgrad ein zentrales Thema. Ganz gleich, ob dies durch neue und innovative Lösungen mit geringerem Energie- oder Rohstoffverbrauch oder durch eine verbesserte Produktentwicklung und -wartung erreicht wird – ein höherer Wirkungsgrad trägt häufig zu nachhaltigeren Designs bei.

Bild 1. Das Otii Ace Pro stellt sicher, dass die Entwickler einen vollständigen Überblick über den Stromverbrauch ihrer Designs haben. (Quelle: Mouser Electronics)

Diese kontinuierliche Dynamik der Elektronikindustrie wird auch deutlich anhand vieler Initiativen von Regierungen und Arbeitsgruppen auf der ganzen Welt. Manche Initiativen zielen darauf ab, Energie nachhaltiger zu erzeugen, während andere die Nutzung der gesamten Energie mit weniger Abfall anstreben, wie in den Zielen für nachhaltige Entwicklung der Vereinten Nationen (Sustainable Development Goals, SDGs) dargelegt [1].

Gleichzeitig wächst unser Streben nach mehr Technologie und Automatisierung stetig. Dieser Sektor ist allerdings auch ein bedeutender Energieverbraucher: Elektromotoren verschlingen 40 % der weltweiten Energie [2]. Doch gleichzeitig leistet die Technologie in vielen Bereichen einen wichtigen Beitrag zu einem nachhaltigen Leben.

Nachhaltigkeit beginnt mit gutem Design

Wenn ein Produkt nicht bereits von Beginn an auf Nachhaltigkeit ausgelegt ist, wird es auch niemals nachhaltig sein. Entwickler müssen dies von Anfang an im Blick haben und dafür sorgen, dass ihre Designs im Betrieb einen hohen Wirkungsgrad haben und im Standby- oder Ruhemodus extrem energiesparend sind.

Deshalb entwickeln viele Halbleiter-Hersteller Lösungen zur Verbesserung des Wirkungsgrads. Bauteile wie beispielsweise Halbleiter, Mikroprozessoren und Kommunikationsmodule (um nur einige zu nennen) leisten einen wichtigen Beitrag zu einem besseren Betriebsverhalten bei gleichzeitig geringerem Stromverbrauch.

Angesichts dieser Ausrichtung auf die Reduzierung des Stromverbrauchs bereits in der Entwurfsphase wurden in jüngster Zeit eine Reihe von Tools entwickelt, die diese Aufgabe erleichtern sollen. Ein Beispiel ist das Netzteil und Messgerät *Otii Ace Pro* von Qoitech [3] (**Bild 1**). Dieses kleine Tischgerät überwacht und speichert Strom- und Spannungswerte und ermöglicht so Analysen in Echtzeit sowie eine Abschätzung der Lebensdauer von Batterien.

Das Tool kann die Gesamtstromaufnahme, die Kriechverluste und den Strom im Ruhezustand überwachen, aufzeichnen und analysieren und gibt den Entwicklern die Informationen an die Hand, die sie zur Verbesserung der Nachhaltigkeit ihrer Designs benötigen.

Nachhaltige Halbleiter als Schlüssel für einen hohen Wirkungsgrad

Die Erzeugung erneuerbarer Energie ist eine der schwierigsten Bereiche für Halbleiter, da Nachhaltigkeitsziele verlangen, dass wirklich die gesamte Energie in elektrische Energie umgewandelt wird. Ein weiterer Bereich ist der Markt für Elektrofahrzeuge, bei dem eine möglichst große Reichweite aus einer Batterie mit fester Kapazität herausgeholt werden muss.

Silizium ist seit vielen Jahrzehnten die Grundlage für Halbleiter-Bauteile. Zwar ist es in den meisten Applikationen nach wie vor das bevorzugte Material, doch die anspruchsvollsten Anwendungen verlangen einen Wirkungsgrad, den herkömmliches Silizium

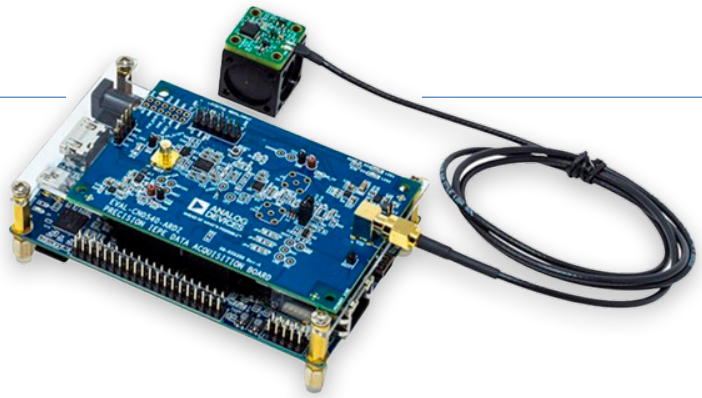


Bild 2. Die Entwicklungsplattform CN0549 von Analog Devices überwacht Maschinen, damit das Personal die Instandhaltung nur bei Bedarf durchführt. (Quelle: Mouser Electronics)

nur schwer erreichen kann. Die Entwickler setzen daher immer mehr auf andere Materialien wie Siliziumkarbid (SiC), die selbst im Hochfrequenzbetrieb eine hohe Zuverlässigkeit bei extrem niedrigen Verlusten bieten.

Ein Beispiel dafür sind Wechselrichter. Wechselrichter sind für die Umwandlung von DC aus Solar-/Photovoltaik-Modulen in AC-Netzstrom unerlässlich. Silizium kann einen Wirkungsgrad von 98 % erreichen, was gut ist. Durch den Wechsel zu einer SiC-Lösung werden die Verluste jedoch halbiert (99 % Wirkungsgrad). Würden in den gesamten USA SiC-Wechselrichter eingesetzt, wo 60 GW Energie durch Solarpanels erzeugt werden, würde die Einsparung 600 MW betragen. In Europa, wo über 200 GW Strom aus Sonnenenergie gewonnen werden, würden die Einsparungen sogar mehr als 2 GW betragen.

SiC-Bauteile sind zwar noch nicht alltäglich, aber viele Hersteller bieten diese Hochleistungsbauteile bereits an. Onsemi [4] hat ein breites Spektrum an SiC-Produkten im Angebot, darunter die M3S EliteSiC MOSFETs [5], die im Vergleich zu früheren Generationen 40 % geringere Gesamtschaltverluste aufweisen.

Nachhaltigkeit in der industriellen Instandhaltung

Fabriken sind für die Herstellung von Gütern des täglichen Bedarfs unerlässlich und werden auch in Zukunft nachhaltige Güter produzieren. In der Vergangenheit haben sie viel Energie verbraucht, aber steigende Energiekosten und die Notwendigkeit der Nachhaltigkeit haben dazu geführt, dass man sich stärker auf die Einsparung von Energie konzentriert.

So werden in Fabriken und Lagern beispielsweise häufig stromsparende Sensoren eingesetzt, um die Beleuchtung so zu steuern, dass sie nur dann eingeschaltet wird, wenn Menschen anwesend sind, um so Verschwendung zu vermeiden.

Auch Instandhaltung kann ein heikles Thema sein: Wenn sie zu oft durchgeführt wird, werden Ressourcen verschwendet, und wenn sie zu lange hinausgezögert wird, besteht die Gefahr eines Ausfalls und einer teuren Reparatur. Mit Hilfe der Entwicklungsplattform CN0549 [6] von Analog Devices Inc. [7] für die zustandsbasierte Überwachung (**Bild 2**) – bestehend aus dem IEPE-Datenerfassungsboard CN0540 [8], dem Evaluierungsboard für Schaltungen CN-0532 [9] und dem Montageblock EVAL-XLMOUNT1 [10] – lassen sich Maschinenvibrationen ermitteln, die auf Verschleiß hinweisen und anzeigen, ob eine Instandhaltungsmaßnahme erforderlich ist.

Fazit

Nachhaltigkeit wird ein Schlüsselthema der Zukunft sein, und die Technologiebranche spielt dabei eine wichtige Rolle. Sie wird eine Vorreiterrolle bei der noch effizienteren Erzeugung elektrischer Energie spielen und sicherstellen, dass diese wertvolle Ressource durch optimale Designs, Automatisierung und Überwachung möglichst effizient genutzt wird.

240281-02

Über den Autor

Als Direktor für technische Inhalte bei Mouser Electronics im EMEA-Wirtschaftsraum ist Mark Patrick für die Erstellung und Verbreitung technischer Inhalte verantwortlich - Inhalte, die für die Strategie von Mouser zur Unterstützung, Information und Inspiration des technischen Publikums entscheidend sind. Bevor er die Leitung des Bereichs Technical Content übernahm, war Mark Patrick Teil des *EMEA Supplier Marketing Teams* von Mouser und spielte eine wichtige Rolle beim Aufbau und der Entwicklung von Beziehungen zu wichtigen Fertigungspartnern. Marks frühere Erfahrungen umfassen praktische Ingenieursaufgaben, technischen Support, technischen Halbleitervertrieb und verschiedene Marketingpositionen. Mark ist im Herzen ein „praktischer" Ingenieur und hat einen erstklassigen Abschluss in Elektrotechnik von der Coventry University. Seine Leidenschaft gilt alten Synthesizern und der Wartung und Reparatur britischer Motorräder, wobei er nicht scheut, selber zum Schraubschlüssel zu greifen.

WEBLINKS

[1] United Nations' Sustainable Development Goals (SDGs): https://sdgs.un.org/goals
[2] IEA, „Energy Efficiency Policy Opportunities for Electric Motor-Driven Systems", 2011: https://iea.org/reports/energy-efficiency-policy-opportunities-for-electric-motor-driven-systems
[3] Qoitech Otii Ace Pro Power Supply & Measuring Instrument : https://tinyurl.com/qoitech-otii
[4] Onsemi bei Mouser: https://tinyurl.com/manufacturer-onsemi
[5] M3S-EliteSiC-MOSFETs: https://tinyurl.com/M3S-EliteSiC
[6] Entwicklungsplattform CN0549: https://tinyurl.com/ADCN0549
[7] Analog Devices bei Mouser: https://tinyurl.com/manufacturer-AD
[8] EVAL-CN0540-ARDZ: https://tinyurl.com/ADCN0540
[9] EVAL-CN0532-EBZ: https://tinyurl.com/ADCN-0532
[10] EVAL-XLMOUNT1: https://tinyurl.com/EVAL-XLMOUNT1

AWS für Arduino und Co.

Teil 1: AWS-IoT-ExpressLink in der Praxis

Von Tam Hanna (Ungarn)

Die Kommunikation mit Amazon Web Services (AWS) ist nicht trivial, doch für einen Raspberry Pi eine machbare Aufgabe. Allerdings sind Single-Board-Computer für viele IoT-Anwendungen zu groß. Mit AWS-IoT-ExpressLink bietet Amazon auch Anbietern von Kleinstgeräten einen einfachen Zugang zur Cloud. Das *AWS IoT ExpressLink powered Connectivity Module* ist dabei für die Cloud-Kommunikation zuständig, der Host-Mikrocontroller kann sich auf das Messen und Steuern konzentrieren. Im ersten Teil der zweiteiligen Serie geht es um die Einrichtung eines ExpressLink-Moduls, hier basierend auf einem ESP32.

Clouddienste wie Microsoft Azure oder Amazon Web Services bieten ihren Nutzern mächtige Optionen zur Haltung, Verarbeitung und Darstellung von Daten an. Die Kommunikation mit Cloudservices ist nicht trivial, doch für Single Board Computer wie beispielsweise einen Raspberry Pi oder Orange Pi eine machbare Aufgabe. Allerdings sind derartige Systeme für viele Anwendungsfälle zu groß. Für Mikrocontrollerboards gilt: Das Implementieren eines vollwertigen TCP/IP-Stacks auf einem 32-Bit-Controller ist möglich, aber unbequem. Auf noch kleineren Maschinen artet der Task in eine herkulische Aufgabe aus.

Mit der AWS-IoT-ExpressLink-Technologie möchte Amazon Anbietern von Kleinhardware dabei helfen, diese letzte Meile zu überwinden. In diesem Artikel werfen wir einen Blick auf die Möglichkeiten und Grenzen des Systems, um danach Experimente mit einem Arduino UNO R4 Wifi und einem Espressif ESP32-C3-AWS-ExpressLink-DevKit durchzuführen. Ziel ist die Vorstellung eines „End-to-End-Workflows“, der sich in eigenen Projekten produktiv einsetzen lässt.

Einfaches AT-Protokoll

Das einst von Hayes geschaffene AT-Protokoll hat sich seit langer Zeit von der Steuerung von Modems emanzipiert: Wer mit Funkmodulen interagiert, findet den AT-Standard in vielerlei Implementierungen.

Die beste Zusammenfassung, wie ein ExpressLink-Modul funktioniert, findet sich in der Dokumentation von Espressif - **Bild 1**

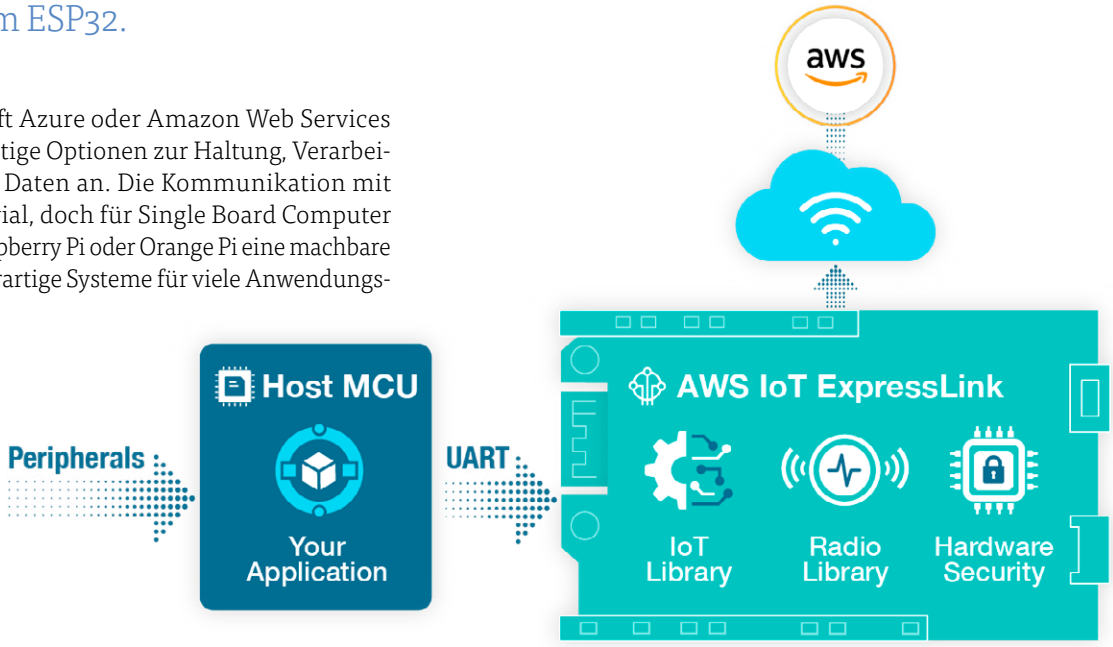


Bild 1. Böse Zungen bezeichnen das ExpressLink-Modul als AWS-Modem. (Bildquelle: [1]).

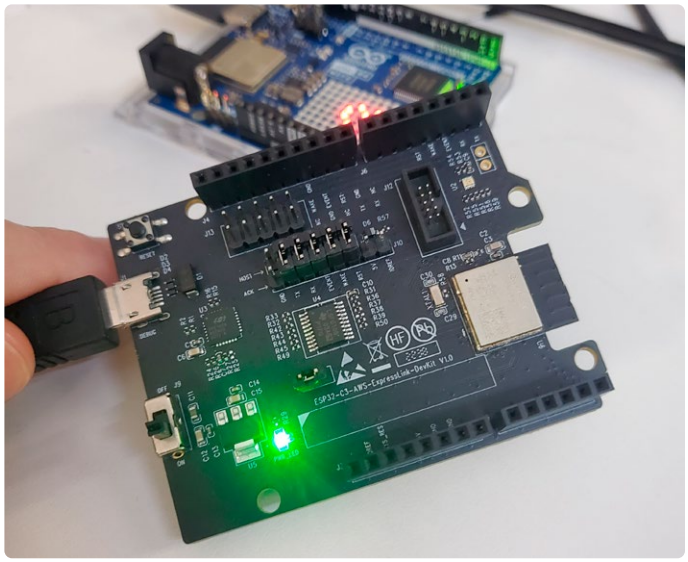


Bild 2. Mit diesem Modul zum Erfolg!

zeigt, wie die einzelnen Elemente zusammenarbeiten. Das ExpressLink-Modul kommuniziert über eine simple Schnittstelle mit dem Host-Mikrocontroller, der seinerseits für die Interaktion mit Sensoren und Aktoren alleinverantwortlich zeichnet.
Die auf dem Host-Mikrocontroller laufende Software hat dann die Aufgabe, wohlformatierte Befehle in Richtung des Moduls zu senden, das sich um die Kommunikation mit der Cloud kümmert. Explizit angemerkt sei, dass sich die ExpressLink-Module normalerweise auch um das Bereitstellen der Luftschnittstelle kümmern. Wer ein derartiges Modul mit einem vollwertigen Funk-SoC wie einem GigaDevice GD32W515 oder einem weiteren ESP32-Board kombiniert, muss darauf achten, dass die Cloud-Kommunikation prinzipiell über den Transceiver des ExpressLink-Moduls erfolgen muss. Ein angenehmer Nebeneffekt dieser Design-Entscheidung ist allerdings, dass der Gutteil der ExpressLink-Module auch eine Antenne mitbringt. Der Kampf mit Antennen-Geometrie, Vektor-Netzwerk-Analysator und Co. lässt sich auf diese Art und Weise zumindest bis zu einem gewissen Grad entschärfen.
Mit dem in **Bild 2** gezeigten ESP32-C3-AWS-ExpressLink-DevKit steht ein durchaus attraktives Starter-Kit zur Verfügung, das sich mit verschiedensten Arduinos koppeln lässt. Ob seines Oemsecrets-Bestpreises von rund 28 Euro (siehe [2]) handelt es sich um eine niederschwellige Möglichkeit, in die Arbeit mit AWS-IoT-ExpressLink einzusteigen. In den folgenden Schritten wird der Autor als Basisplatine einen Arduino UNO R4 Wifi verwenden, dessen WLAN-Modul brachliegt (siehe oben).

Inbetriebnahme der Hardware

In der Theorie könnte das „Online-Nehmen" der Platine einfacher nicht sein - weil Espressif das ExpressLink-Modul in Form eines Arduino-Shields anbietet, reicht es aus, die beiden Platinen aufeinander zu stecken.
Bild 1 lügt an dieser Stelle insofern, als dass das Kommunikations-Interface zwischen Host-MCU und Funkmodul nicht nur aus einem reinen seriellen Link besteht. Stattdessen gibt es, wie in **Bild 3** gezeigt, auch einige zusätzliche Signalleitungen, die das Signalisieren von kritischen oder sonstwie relevanten Betriebszuständen ermöglichen.
Da wir in den folgenden Schritten auf die Kombination aus Arduino und Espressif-Arduino-Shield setzen, sollen uns diese Pins hier

ExpressLink Pin	ESP32-C3 GPIO Pin	ESP32-C3-MINI-1-N4-A Module Pin
TX	IO19	27
RX	IO18	26
EVENT	IO10	16
WAKE	IO3	6
RESET	EN	8

Bild 3. Einige GPIO-Pins sind für das Signalisieren von kritischen oder sonstwie relevanten Betriebszuständen erforderlich (Bildquelle: [14]).

nicht weiter aufhalten. Die Erwähnung erfolgt vor allem deshalb, weil Espressif in der Dokumentation an mehreren Stellen auf die Wichtigkeit dieser Verbindungen hinweist - fehlen sie, so lässt sich nur eine „Hello World"-Kommunikation abwickeln; eine praktische Kommunikation mit den AWS ist nicht möglich.
Vor dem Zusammenstecken von Arduino und Espressif-Modulkarte ist es empfehlenswert, ein Firmware-Update für die auf dem Modul lebende Software durchzuführen. Ursache dafür ist, dass die Module kein sonderlich schnell drehendes Produkt sind und mitunter einige Zeit im Lager des Distributors schlummern, bevor sie von einem Kunden erworben werden.
In der Theorie könnte diese Aktualisierung über die Luftschnittstelle durchgeführt werden, was allerdings schon eine bestehende Verbindung zwischen dem Modul und dem AWS-IoT-Cloud-Services voraussetzen würde.
Als Alternative stellt Espressif ein Python-Skript zur Verfügung, das auf Seiten des PCs Python 3 voraussetzt. Auf der Workstation des Autors, die unter Ubuntu 20.04 LTS läuft, präsentiert sich der Versionsstand folgendermaßen:

```
tamhan@TAMHAN18:~$ python3 --version
Python 3.8.10
```

Für die eigentliche Kommunikation ist dann die Eingabe des Befehls `pip3 install pyserial==3.5` erforderlich, die eine Brückenbibliothek zwischen der Python-Runtime und den seriellen Ports des PCs zur Verfügung stellt.
Im nächsten Schritt ist das als *.py*-Datei vorliegende Tool zum Downloaden der Firmware erforderlich, das sich in GitHub unter [3] befindet. Beachten Sie den in **Bild 4** hervorgehobenen Button, der das direkte Herunterladen des Tools ohne den normalerweise notwendigen Umweg in die Zwischenablage ermöglicht.
Im folgenden Schritt ist das Herunterladen der Firmware notwendig - auch hierzu ist ein Besuch bei GitHub erforderlich [4].

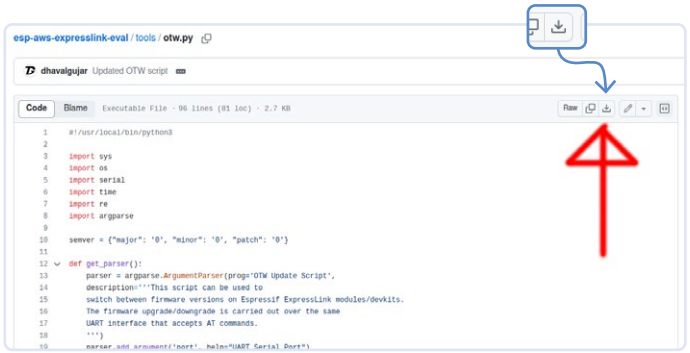


Bild 4. Dieser Knopf spart wertvolle Zeit.

```
[ 1613.074190] usb 1-1.1: new high-speed USB device number 7 using ehci-pci
[ 1613.191003] usb 1-1.1: New USB device found, idVendor=0403, idProduct=6010, bcdDevice= 7.00
[ 1613.191006] usb 1-1.1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 1613.191007] usb 1-1.1: Product: FT2232H-56Q MiniModule
[ 1613.191009] usb 1-1.1: Manufacturer: FTDI
[ 1613.191010] usb 1-1.1: SerialNumber: FTL6Z5PC
[ 1613.235698] usbcore: registered new interface driver ftdi_sio
[ 1613.235776] usbserial: USB Serial support registered for FTDI USB Serial Device
[ 1613.235898] ftdi_sio 1-1.1:1.0: FTDI USB Serial Device converter detected
[ 1613.236007] usb 1-1.1: Detected FT2232H
[ 1613.237935] usb 1-1.1: FTDI USB Serial Device converter now attached to ttyUSB0
[ 1613.238114] ftdi_sio 1-1.1:1.1: FTDI USB Serial Device converter detected
[ 1613.238940] usb 1-1.1: Detected FT2232H
[ 1613.239673] usb 1-1.1: FTDI USB Serial Device converter now attached to ttyUSB1
[ 1676.337202] usb 1-1.4: new full-speed USB device number 8 using ehci-pci
[ 1676.452057] usb 1-1.4: New USB device found, idVendor=10c4, idProduct=ea60, bcdDevice= 1.00
[ 1676.452062] usb 1-1.4: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 1676.452065] usb 1-1.4: Product: CP2102N USB to UART Bridge Controller
[ 1676.452067] usb 1-1.4: Manufacturer: Silicon Labs
[ 1676.452069] usb 1-1.4: SerialNumber: 5a66658d521fed11a7015cb1dcc2a201
[ 1676.452682] cp210x 1-1.4:1.0: cp210x converter detected
[ 1676.455540] usb 1-1.4: cp210x converter now attached to ttyUSB2
tamhan@TAMHAN18:~$
```

Bild 5. Beide USB-Seriell-Wandler melden sich im /dev-Dateisystem der Workstation an.

```
tamhan@TAMHAN18:~/Desktop/Stuff/2024Mar/EAG-ExpressLink$ python3 otw.py /dev/ttyUSB1 v2.5.0.bin
File size 1511424
Uploaded 100.0%
Done...
tamhan@TAMHAN18:~/Desktop/Stuff/2024Mar/EAG-ExpressLink$
```

Bild 6. So OTW.py keine Fehlermeldungen anzeigt, ist das Modul auf dem aktuellen Stand.

Lohn der Mühen war im Test des Autors das Herunterladen der Firmwareversion *v2.5.0.bin*, und achten Sie darauf, nicht versehentlich die Hash-Datei zu erbeuten.

Die Aktualisierung der Betriebs-Software kann nicht über das auf dem Board verbaute USB-Interface erfolgen. Es ist ja mit einem „Applikations-UART" verbunden und ermöglicht nicht die Programmierung des als Funkmodul dienenden ESP32. Zur Umgehung des Problems verwendet der Autor ein FTDI-Minimodul, spezifischerweise das FT2232H MINI MODULE, das im Labor auch sonst gute Dienste bei der Programmierung von ESP32-Mikrocontrollern leistet.

Zu beachten ist, dass die Verbindung etwas eigenwillig ist. Pin BD1 wird mit TX des ESP32-Moduls verbunden, während Pin BD0 mit dem RX-Pin verbunden wird. Die Pins TX und RX befinden sich dabei in der Mitte des Moduls, sie stimmen nicht mit der üblichen Arduino-Pinbelegung überein.

Zu guter Letzt ist dann noch ein Hüpf-Draht erforderlich, der die beiden Massen-Potenziale miteinander verbindet. Für die eigentliche Inbetriebnahme wird dann im ersten Schritt das FTDI-Modul mit dem Rechner verbunden, die Energieversorgung des Express-Link-Boards erfolgt erst danach. In *dmesg* „sequenziert" sich dieser Verbindungsaufbau dann wie in **Bild 5** gezeigt.

Wichtig ist dabei vor allem die Feststellung, dass die Ports ttyUSB0 und ttyUSB1 für die PC-FTDI-Kommunikation verantwortlich sind. Wenn man die genannten Pins benutzt, dann läuft die eigentliche Programmauslieferung nach folgendem Schema:

```
tamhan@TAMHAN18:~/Desktop/Stuff/2024Mar/EAG-ExpressLink
  $ python3 otw.py /dev/ttyUSB1 v2.5.0.bin
```

Der Erfolg zeigt sich wie in **Bild 6**. Nach dem Ausschalten wird das FTDI-Modul nicht mehr benötigt. Das ExpressLink-Modul darf nun mit dem Arduino verbunden werden.

Aufbau einer Cloudverbindung

Nach dem Aktualisieren der auf dem Modul befindlichen Firmware sind wir zu seiner Inbetriebnahme bereit. Der Autor geht in den folgenden Schritten davon aus, dass Sie ein betriebsbereit konfiguriertes AWS-Konto besitzen, und sich unter der URL *https://aws.amazon.com/* mit ihrem root-User einloggen können. Die in der Praxis aus sicherheitstechnischer Sicht empfehlenswerte Nutzung von IAM wollen wir in den folgenden Schritten aus Platzgründen unterlassen.

Wer noch kein AWS-Konto hat, findet unter [5] eine Schnellstart-Anleitung. Zu beachten ist, das die Einrichtung von AWS den Besitz einer Kreditkarte voraussetzt.

Das Modul mit seiner fixen Firmware wird über die Cloud oder über das Senden von AT-Befehlen mit bestimmten Konfigurations-Parametern initialisiert. Für die folgenden Schritte ist auf Workstation-Seite die unter der URL *http://console.aws.amazon.com/iot* bereitstehende AWS-IoT-Konsole erforderlich, die sie in einem Browserfenster geöffnet halten sollten.

Als Entwicklungsumgebung für den Arduino-Sketch dient in den folgenden Schritten die Arduino-IDE 2. Der Start erfolgt unter Linux aus einer AppImage-Datei heraus und soll hier nicht weiter besprochen werden.

Wichtig ist, dass im laufenden Betrieb nur der Arduino mit dem Rechner verbunden werden muss - das Modul bezieht seine Energie aus dem angeschlossenen Arduino. Soll der PC direkt mit dem Modul durch Senden von AT-Kommandos kommunizieren, so ist die Verbindung zum Arduino zu trennen und das Micro-USB-Kabel anzuschließen.

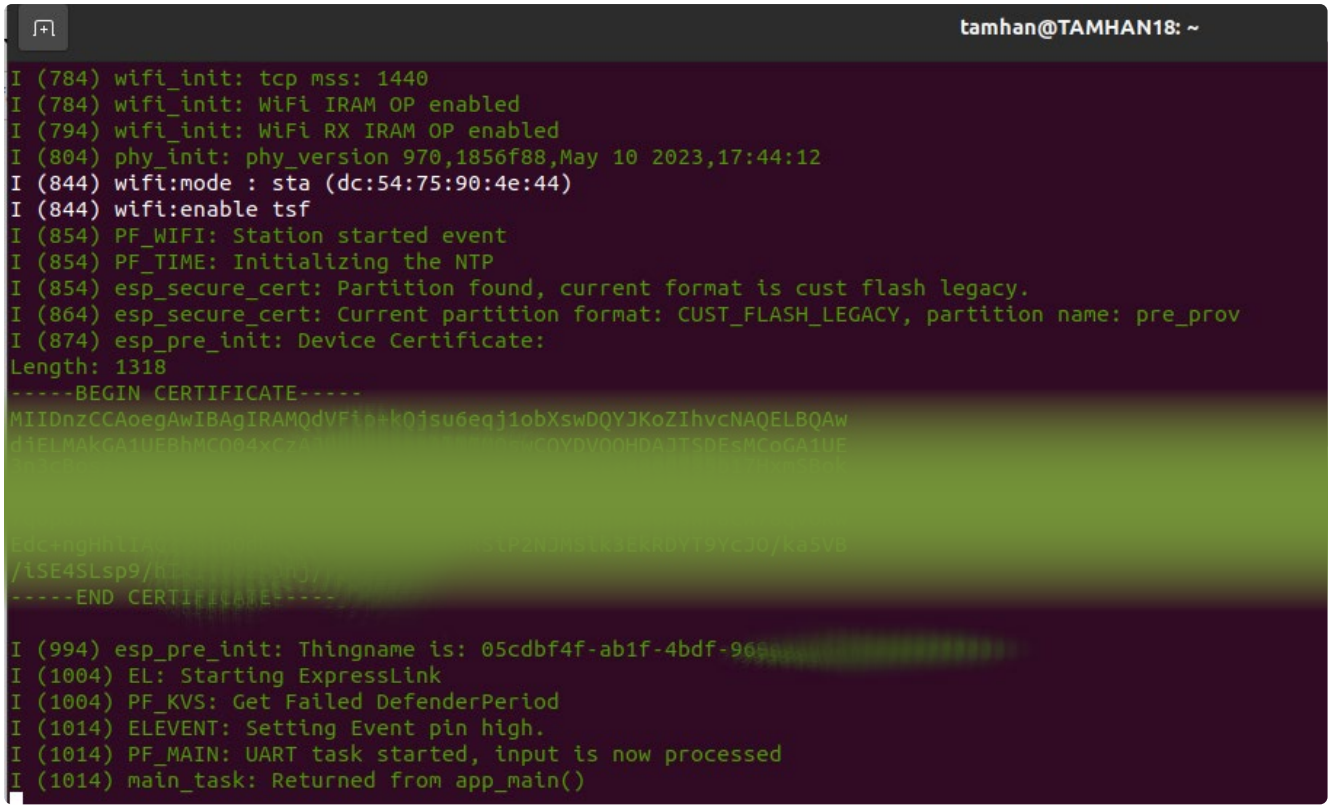


Bild 7. Das ExpressLink-Modul zeigt sich beim Hochfahren von der kommunikativen Seite.

Hardware, die mit den AWS kommuniziert, wird auf dem Server als *Thing* abgebildet. Für die Authentifizierung unseres *Things* beim Server ist einerseits ein *Thing Name* und andererseits ein kryptographisches Zertifikat erforderlich. Beide befinden sich im Speicher des ExpressLink-Moduls und müssen im ersten Schritt ermittelt werden.

In der Dokumentation erwähnt Amazon zwar auch die Befehle `AT+CONF? ThingName` und `AT+CONF? Certificate pem`. In der Praxis ist es aber einfacher, das Modul direkt mit dem Rechner zu verbinden, ein Terminalprogramm zu starten und danach die Resettaste zu drücken. Die von Espressif implementierte Firmware gibt in diesem Fall, wie in **Bild 7** gezeigt, die benötigten Informationen aus, wo sie direkt in die Zwischenablage überführt werden können.

Als Terminal-Programm verwendet der Autor unter Linux gerne *Screen* - obwohl es dabei eigentlich um eine Zweckentfremdung handelt, lässt sich der Verbindungsaufbau nach folgendem Schema befehligen:

```
tamhan@TAMHAN18:~$ screen /dev/ttyUSB0 115200
```

Zum Beenden von *Screen* öffnen Sie dann ein zweites Terminalfenster, in dem sie folgenden Befehl eingeben. Sinn dieser „Umgehung" ist, dass die Inhalte des *Screen*-Dialogs im anderen Terminalfenster stehen bleiben und abgeerntet werden können:

```
tamhan@TAMHAN18:~$ sudo killall screen -s9
```

Als nächste Amtshandlung im Kampf um die Konfiguration müssen wir in die weiter oben genannte AWS-IoT-Konsole zurückkehren. Die genaue URL ist dabei vom geographischen Aufenthaltsort abhängig, den sie als Basis für Ihr IoT-System ausgewählt haben - auf der Workstations des Autors lautet der Pfad *https://eu-west-1.console.aws.amazon.com/iot/home?region=eu-west-1#/home*.

Die Webseite ist insofern kompliziert, als dass die links eingeblendete Navigation-Oberfläche erstens unvollständig und zweitens scrollbar ist. Sei dem wie es sei, als nächstes ist ein Klick auf *Link Manage* erforderlich - die benötigte Option findet sich nicht im Baum. Stattdessen müssen sie in der daraufhin geladenen Ansicht auf den in **Bild 8** gezeigten Knopf *Create Thing* klicken.

Lohn der Mühen ist das Starten des Konfigurationsassistenten, der in drei Schritten zur Einpflegung eines neuen *Things* in den AWS-IoT-Service führt.

Im ersten Schritt ist lediglich das Feld *Thing Name* relevant, wo wir den weiter oben ausgegebenen String eingeben. Danach folgt auch schon der Klick auf *Next Step*, wo uns Amazon zum „Hochladen" der kryptographischen Dateien auffordert.

Hier entscheiden wir uns im ersten Schritt für die Option *Use my certificate* und danach für *CA is not registered with AWS IoT*.

Das .pem-Dateiformat hat sich im Laufe der letzten Jahre als Quasi-Standardcontainer für allerlei Arten kryptographischer

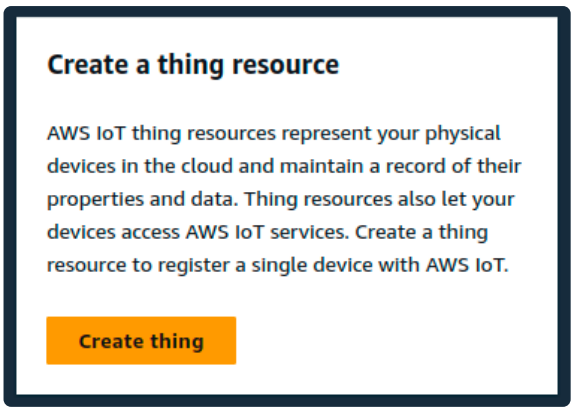


Bild 8. Nur diese Option ermöglicht die Erzeugung neuer Things.

ExpressLink abseits von Espressif

Dieser Artikel nutzt das Espressif-Modul aufgrund seiner bequemen Verfügbarkeit. Amazon hat mittlerweile mit diversen anderen Partnern wie beispielsweise U-blox, Infineon, Realtek und Telit Verträge abgeschlossen - unter [15] findet sich eine aktuelle Liste von Starter-Boards aller Hersteller.

Schlüssel etabliert. Eine detaillierte Beschreibung des Dateiformats finden Interessierte übrigens unter [6]. Für unsere Bedürfnisse reicht es allerdings aus, ein gültiges pem-File zu erzeugen:

```
tamhan@TAMHAN18:~/Desktop/ExpressLink$ touch tamscert.pem
tamhan@TAMHAN18:~/Desktop/ExpressLink$ gedit tamscert.pem
```

Nach der Eingabe des `gedit`-Befehls erscheint die neue Datei in Texteditor, wo wir den weiter oben aus der Zwischenablage übernommenen Zertifikatstext einfügen. Wichtig ist, dass die Präambeln `-----BEGIN CERTIFICATE-----` und `-----END CERTIFICATE-----` ebenfalls in das Textfile wandern, weil der Parser die Zertifikat-Informationen sonst nicht erbeuten kann.
Nach dem Abspeichern der .pem-Datei können Sie diese hochladen und im letzten Schritt in der Rubrik *Certificate status* die Option *Active* auswählen. Danach folgt auch schon ein weiterer Klick auf *Next Step*, um in den dritten Schritt *Attach policies to certificate* zu wechseln.
AWS IoT implementiert ein Zertifikat-berechtigungsbasiertes Schema. Das bedeutet, dass Zertifikate mit Policy-Dokumenten verbunden werden; diese legen dann fest, was die jeweilige Ressource im AWS-Verbund „anstellen“ darf.
In frisch konfigurierten AWS-Clustern sind seit längerer Zeit keine Policies mehr von Haus aus angelegt: Amazon wählt diese Vorgehensweise wohl deshalb, um im Fall von durch unsichere Konfigurationen verursachten Sicherheitsvorfällen nicht haftbar zu sein. Aus diesem Grund klicken wir im ersten Schritt auf den Button *Create Policy*, der ein neues Browser-Fenster mit dem Policy-Editor öffnet. Im ersten Schritt vergeben wir dort einen Namen, unter dem die Policy fortan bekannt werden soll.
Darunter findet sich eine zweite Option, die für das Festlegen der eigentlichen durch die Policy gewährten Berechtigungen verantwortlich ist. Dort klicken wir auf die Option JSON, um die direkte Texteingabe zu aktivieren. Der dort bereits befindliche Code ist im Allgemeinen schon übernahmebereit - achten Sie lediglich darauf, in den Feldern *Action* und *Resource* die im folgenden Schema gezeigten Wildcards einzupflegen:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "*",
      "Resource": "*"
    }
  ]
}
```

Die gezeigte Policy erlaubt dem Gerät den universalen Zugriff auf alles, was im AWS-Backend kreucht und/oder fleucht. Aus der Logik folgt, dass diese Vorgehensweise in praktischen Deployments nicht empfehlenswert ist. Dass Sie diese Policies nach dem Experimentieren so schnell wie möglich eliminieren sollten und Verlag und Autor jegliche Haftung ablehnen, sei schon im Interesse der rechtlichen Vollständigkeit nochmals angemerkt!
Nach dem Anklicken der *Create*-Option öffnet das Policy-Editorfenster eine Liste der Policies - dies ist Anweisung genug, das Fenster zu schließen und zum eigentlichen Thing-Generationsprozess zurückzukehren. Die soeben angelegte neue Policy präsentiert sich dann zur Auswahl bereit.
Nun ist noch ein Klick auf den *Create*-Button erforderlich, um den Einpflege-Prozess im Backend abzuschließen.
Zum Abschluss der Konfigurationshandlungen müssen Sie die links eingeblendete Navigationsoberfläche der AWS-IOT-Konsole ganz nach unten scrollen, um auf die Option *Settings* klicken zu können. In der Rubrik *Device Data Endpoints* findet sich danach ein nach dem Schema *a3gw111abcdefgdl-ats.iot.eu-west-1.amazonaws.com* aufgebauter String, der ebenfalls zu speichern ist. Er legt fest, über welchen Einsprungpunkt das Modul Kontakt zur Cloud aufnehmen wird.

Herstellen der WLAN-Verbindung

Die korrekte Konfiguration auf Seiten der AWS-Cloud ist nur die halbe Miete, als das ESP32-Modul ja noch zur Kommunikation mit Ihrem Netzwerk befähigt werden muss. Dazu ist es erforderlich, WLAN-Verbindungsdaten in das Secure Element zu schreiben.
In der von Amazon spezifizierten AT-Sprache finden sich die Kommandos `AT+CONF SSID=<replace-with-your-router-ssid>` und `AT+CONF Passphrase=<replace-with-your-router-passphrase>`, die das Übertragen der Parameter über einen seriellen Link ermöglichen.
Espressif hat seine hauseigenen ExpressLink-Module allerdings mit einer kleinen Bequemlichkeitsoption ausgestattet: unter [6] beziehungsweise [7] finden sich Android- und iOS-Anwendungen, die eine grafische Konfiguration der Module ermöglichen.
Problematisch ist in diesem Zusammenhang lediglich, dass die Exponierung des notwendigen Bluetooth-Interfaces zuerst das Übertragen des Befehls `AT+CONFMODE` voraussetzt. Hierzu ist der unter [8] von Amazon bereitgestellte Beispiel-Sketch erforderlich.
Normalerweise nutzt der Arduino ja den seriellen Hardware-Port für die Ausgabe von `printf`-Messages und anderen Statusinformationen. Nach Anstecken eines ExpressLink-Moduls steht dieser (logischerweise) nicht mehr zur Verfügung, weshalb das AWS-Entwicklerteam stattdessen auf die *SoftwareSerial* Library setzt. Zum Zeitpunkt der Drucklegung galt allerdings, dass diese Bibliothek mit dem Arduino R4 nicht sonderlich gut funktioniert (siehe [10]) und außerdem nicht erforderlich ist.

Unsere erste Amtshandlung ist deshalb das Bereinigen des Sketches um alle diesbezüglichen Aufrufe. Unter [11] findet sich nämlich die erfreuliche Information, dass der Renesas-Arduino zwei Hardware-UART-Schnittstellen mit unterschiedlichen Pins bietet. `Serial1` sendet dabei Daten an das Funkmodul, während `Serial` den Seriellen Monitor der Arduino-IDE auf dem PC bespielen kann. Das bedeutet, dass die für die Kommunikation mit dem ESP32-Modul verantwortliche Methode nach folgendem Schema anzupassen ist:

```
String execute_command(String command, unsigned long
  timeout_ms)
{
    Serial.print("EXC : ");
    Serial.println(command);

     unsigned long saved_timeout_value_ms = Serial1.
       getTimeout();
    Serial1.setTimeout(timeout_ms);
    Serial1.println(command);
    String s = Serial1.readStringUntil('\n');
    s.trim();
    Serial1.setTimeout(saved_timeout_value_ms);
    return s;
}
```

Achten Sie darauf, vor der Programmausführung nach folgendem Schema Korrekturen an der Konfiguration vorzunehmen:

```
#define EVENT_PIN 2
#define RESET_PIN 4

#define MY_AWS_IOT_ENDPOINT "aenderemich-ats.iot.
eu-west-1.amazonaws.com"
```

Aufgrund einer vom Autor mittlerweile unter [12] gemeldeten Beißerei zwischen dem Renesas- und dem Arduino-Teil der Bibliothek ist dann eine Anpassung der Methode `process_event` erforderlich. Spezifischerweise darf der von `c_str()` retournierte Wert nicht direkt an `scanf` übergeben werden:

```
event_t process_event()
{
  String response;

    int val = 0;
    event_t event_number = EVENT_NONE;
    val = digitalRead(EVENT_PIN);
    if(val)
    {
        response = execute_command("AT+EVENT?", 3000);
        if (response.equals("OK"))
        {
            return EVENT_NONE;
        }
        else {
            char ok_string[3];
            int topic_index;
            int total_read;
            char someResponse[300];
            strcpy(someResponse, response.c_str());
            total_read = sscanf(someResponse, "%s %d %d
                %*s" , ok_string, &event_number,
                &topic_index);
            return event_number;
        }
    }
}
```

Das ESP32-Modul wird mit dem Arduino verbunden; nur dieser bekommt ein USB-Kabel eingesteckt. Die Energieversorgung auf dem ExpressLink-Kit sollte auf Off geschaltet sein. Schnelles Doppelt-Anklicken der RST-Taste des Arduino UNO R4 ist erforderlich, um den Bootloader zum Entgegennehmen von neuen Kompilaten zu befähigen.
Nach der erfolgreichen Anpassung und Flashen des Test-Sketches [13] erhält das Modul den notwendigen Befehl, wenn es noch nicht in einem provisionierten Zustand ist. Die Überprüfung dafür erfolgt übrigens durchaus trickreich durch probeweises Zurücklesen der im Secure Element gespeicherten SSID:

```
void loop()
{
    event_t event = EVENT_NONE;
    String response;
    static state_t state = STATE_INIT;
    if (state != STATE_INIT)
    {
        event = process_event();
    }
    switch (state) {
        . . .
        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);
       break;
       case STATE_UNPROVISIONED:
      response = execute_command("AT+CONFMODE", 5000);
       if (response.equals("OK CONFMODE ENABLED"))
       {
           state = STATE_PROVISIONING;
       }
       break;
```

Amazon realisiert das hier gezeigte Programm als kolossalen Zustandsautomaten. Die für das Verarbeiten der SSID

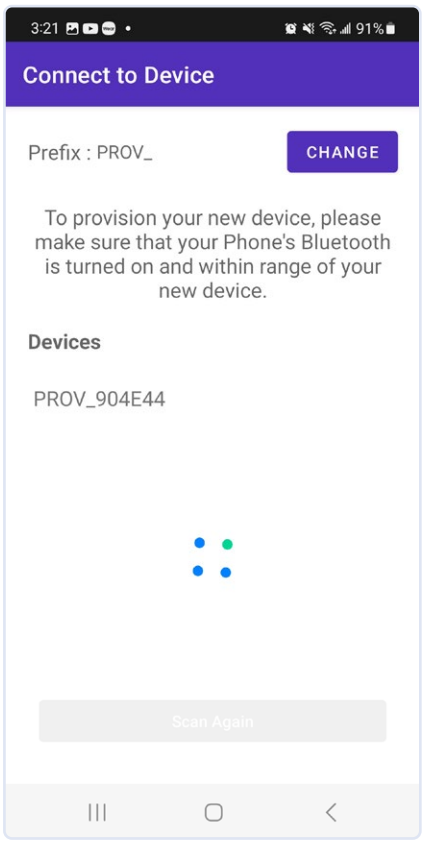


Bild 9. Der Bluetooth-LE-Scanner hat ein neues Opfer gefunden.

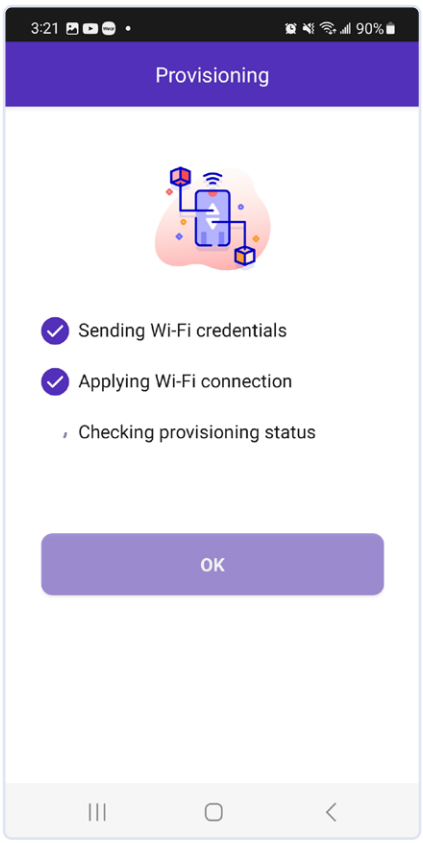


Bild 10. Das Provisioning nimmt einige Zeit (bis zu 2 Minuten) in Anspruch …

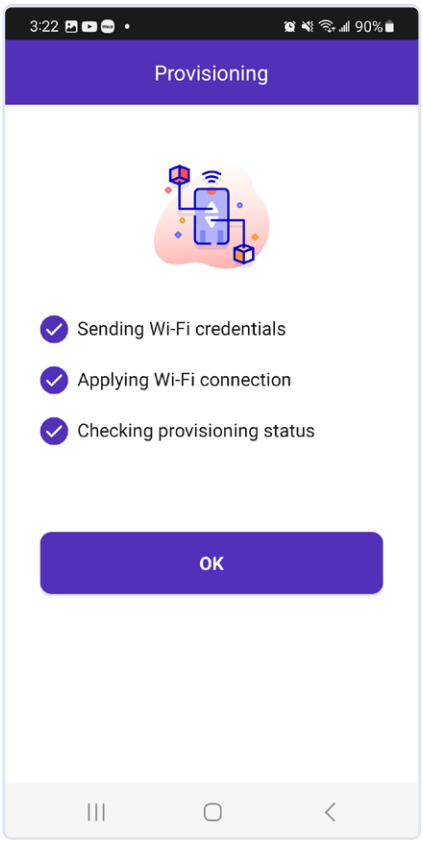


Bild 11. … und ist erst beim Erscheinen des hier gezeigten Fensters abgeschlossen.

verantwortliche Methode setzt dann nach folgendem Schema den korrekten Zustand, wenn die zurückgelieferte SSID unbrauchbar sein sollte:

```
state_t process_ssid(String response)
{
    int event_number = 0;
    char ok_string[3];
    char ssid_string[33] = {0};
    int total_read;

    total_read = sscanf(response.c_str(), "%s %s" ,
      ok_string, ssid_string);

    if(total_read > 1) {
        return STATE_PROVISIONED;
    }
    else {
        return STATE_UNPROVISIONED;
    }
}
```

Sobald das Modul im Provisioning-Status angekommen ist, ist es für einen Bluetooth-LE-Scanner sichtbar. Am zweckmäßigsten ist aber die Nutzung der soeben erwähnten Smartphone-App. Aufgrund von Besonderheiten im Android-Permission-System reagiert das Betriebssystem beim ersten Start mit der bekannten Enchillada von Permission-Anfragen. Wer diese abnickt, findet sich in einem Kamerabildschirm wieder - Espressif nutzt dieselbe Applikation nämlich nicht nur zur Bluetooth-LE-Provisionierung, sondern auch zur Erfassung von ExpressLink-Modulen, die ihren Modul-Namen über einen QR-Code angeben.

In **Bild 9** ist gezeigt, wie das Modul gefunden wird. Die eigentliche Programm-Ausführung erfolgt dann durch einen Wizard, der nach der Auswahl des WLAN-Namens zur Eingabe des Passworts auffordert. Zu beachten ist, dass der Übergang zwischen den in **Bild 10** und **Bild 11** gezeigten Bildschirmen einige Zeit in Anspruch nehmen kann.

Fazit

In diesem Artikel konnten wir bisher nur die Einrichtung des AWS-IoT-ExpressLink-Moduls demonstrieren. Im zweiten Teil dieses Artikels wollen wir uns mit der Übermittlung von Daten beschäftigen. Doch schon jetzt sollten die Vorteile des Systems eindeutig sein. Mit solch geringem Hardware-Einsatz ist es sonst nur schwer möglich, Verbindung zum Cloud-Platzhirsch aufzunehmen. Der Mehrpreis für das Funkmodul amortisiert sich schon deshalb, weil man bei guter Zusammenarbeit von Amazon erfahrungsgemäß immer wieder einen Promotion-Keks zugeworfen bekommt.

240240-02

Über den Autor

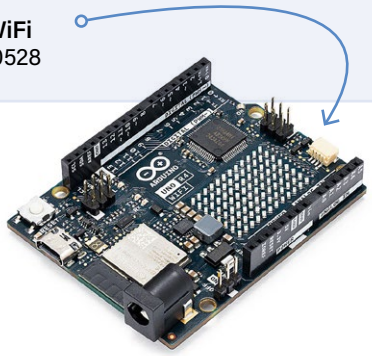
Ingenieur Tam Hanna befasst sich seit mehr als 20 Jahren mit Elektronik, Computern und Software; er ist freiberuflicher Entwickler, Buchautor und Journalist (www.instagram.com/tam.hanna). In seiner Freizeit beschäftigt sich Tam unter anderem mit 3D-Druck und dem Vertrieb von Zigarren.

Sie haben Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare? Dann senden Sie bitte eine E-Mail an den Autor unter tamhan@tamoggemon.com oder die Elektor-Redaktion unter redaktion@elektor.de.

Passendes Produkt

› **Arduino Uno R4 WiFi**
www.elektor.de/20528



WEBLINKS

[1] ExpressLink Dokumentation von Espressif: http://www.espressif.com/en/solutions/device-connectivity/esp-aws-iot-expresslink
[2] ESP32-C3-AWS-ExpressLink-DevKit Distributoren: https://www.oemsecrets.com/compare/ESP32-C3-AWS-ExpressLink-DevKit
[3] Download-Tool: https://github.com/espressif/esp-aws-expresslink-eval/blob/main/tools/otw.py
[4] Firmware des Moduls: https://github.com/espressif/esp-aws-expresslink-eval/releases
[5] AWS Schnellstart-Anleitung: https://docs.aws.amazon.com/iot-expresslink/latest/gettingstartedguide/elgsg-set-up.html
[6] Beschreibung des .pem-Dateiformats:
https://docs.progress.com/bundle/datadirect-hybrid-data-pipeline-installation-46/page/PEM-file-format.html
[7] Config-Tool Android: https://play.google.com/store/apps/details?id=com.espressif.provble
[8] Config-Tool iOS: https://apps.apple.com/app/esp-ble-provisioning/id1473590141
[9] Arduino-Sketch für Bluetooth-Aktivierung:
https://github.com/espressif/esp-aws-expresslink-eval/blob/main/sketches/arduino_sample_sketch.ino
[10] Probleme mit Software-UART: https://github.com/arduino/uno-r4-library-compatibility/issues/12
[11] Zwei Hardware-UARTs: https://forum.arduino.cc/t/uno-r4-and-serial-rx-tx/1146022/2
[12] Probleme mit Bibliotheken: https://github.com/espressif/esp-aws-expresslink-eval/issues/23
[13] Sketch des Autors: http://www.elektormagazine.de/240240-02
[14] Getting Started Guide ExpressLink Evaluation Kit: https://github.com/espressif/esp-aws-expresslink-eval
[15] ExpressLink Starter-Boards: https://partners.amazonaws.com/search/
qualified-devices?page=1&facets=Qualifications%20%3A%20AWS%20IoT%20ExpressLink&redirected=true

Luftstromdetektor (nur) mit Arduino

Keine externen Sensoren erforderlich!

Quelle: Adobe Stock

Von Raymond Schouten (Niederlande)

Diese Detektorschaltung verwendet nur den Arduino Nano selbst, ohne einen zusätzlichen Sensor. Er kann Luftströme aus einer Entfernung von 30 cm erkennen, indem er die interne Temperatur des aktiven Chips mit einer Auflösung von 0,01°C misst und schnelle Temperaturänderungen feststellt. Dieser Entwurf verwendet eine Dithering-Technik, die eine höhere Auflösung als das LSB eines ADCs verspricht.

Was steckt hinter dem Konzept der Luftstromerkennung in diesem Entwurf? Es ist ein recht einfaches Prinzip. Wenn er arbeitet, heizt sich der Controller ATmega328P des Arduino Nano auf etwa 6...10°C über die Umgebungstemperatur auf. Ein auftretender des Luftstrom in Richtung des Chips kühlt ihn dann ab: Dies ist die Grundlage für die Erkennung eines plötzlichen Luftstroms in Richtung des Arduino-Nano-Boards.

Zu diesem Zweck wird zweimal pro Sekunde die aktuelle Temperatur des Chips mit der vorherigen Temperaturmessung verglichen. Temperaturänderungen ergeben sich nicht nur durch den Luftstrom eines Lüfters, sondern auch, wenn man die Platine bewegt oder das Chipgehäuse mit dem Finger berührt (**Bild 1**). Dabei ist zu beachten, dass

- Sie dem Arduino nach dem Kaltstart etwas Zeit zum Aufwärmen geben müssen (5...10 min), damit er als Detektor funktionieren kann. Erst wenn der Chip eine stabile Temperatur erreicht hat, kann man diese kleinen, plötzlichen Luftströme erkennen.
- es sich um einen sehr empfindlichen Detektor mit Ausleseschwankungen von <0,02°C handelt. Ein Luftstrom verursacht aber größere Schwankungen; die Stabilität der Messung kann bewertet werden, indem man den Arduino zum Beispiel mit einer Plastiktüte vom Luftstrom abschirmt (**Bild 2**).

Das Konzept der Temperaturauslesung

Der interne ADC des Chips kann nicht nur seine analogen Eingangspins auslesen, sondern auch so eingestellt werden, dass er den On-Chip-Temperatursensor liest, dessen Spannung immer weniger als 1 V beträgt. Um die Empfindlichkeit des ADC zu erhöhen, wird die Referenzspannung (in der Software) auf eine interne Quelle von 1,1 V eingestellt, anstelle auf die Versorgungsspannung von 5 V. Bei einem 10-Bit-ADC (1.024 Schritte) beträgt der kleinste lesbare Schritt (gerundet) dann 1 mV statt 5 mV.

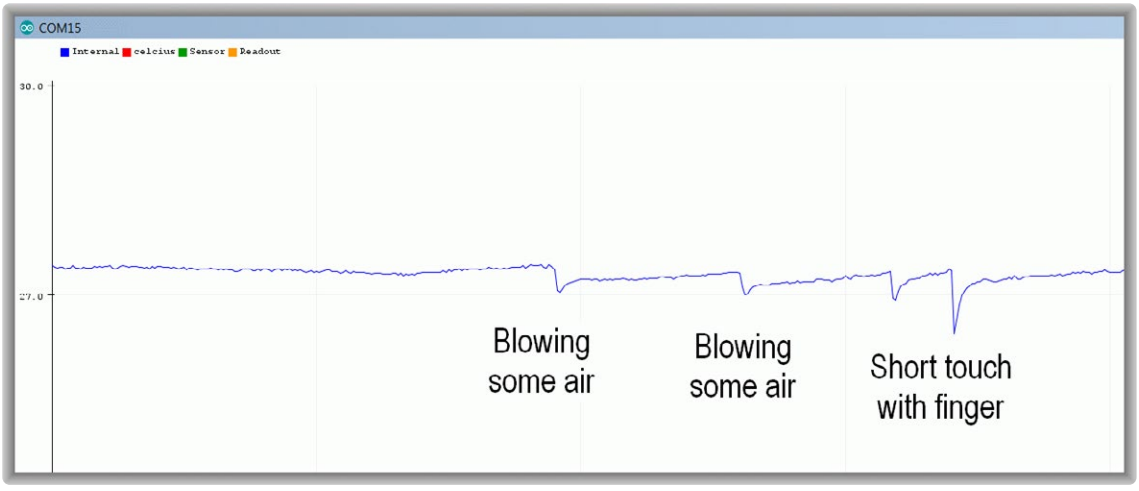


Bild 1. Die schnelle Reaktion der Temperaturmesswerte des Wärmesensors auf dem ATmega328-Controller des Arduino-Nano-Boards.

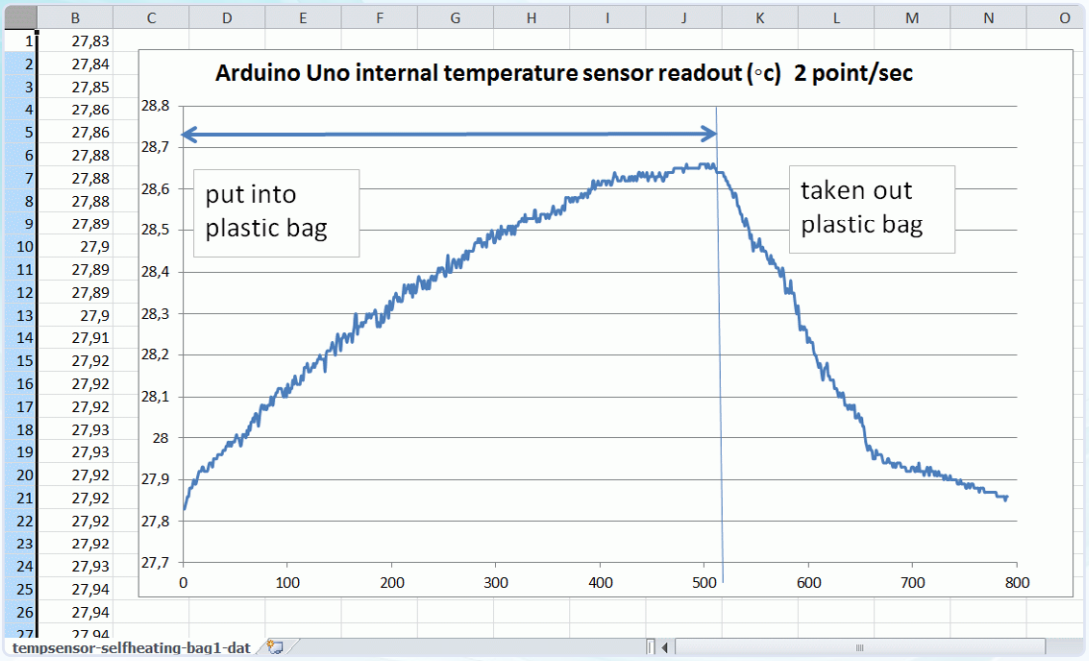


Bild 2. Um die Stabilität der Anzeige zu bewerten, kann man die Platine in eine Plastiktüte stecken und so vor unkontrollierten Luftstromänderungen abschirmen.

Dieser kleinste Schritt wird durch das LSB (Least-significant Bit) ausgedrückt. Laut Datenblatt beträgt der Ausgangshub des Sensors 1 mV/°C, so dass wir mit 1 mV/Schritt eine Gesamtmessauflösung von 1°C erhalten - viel zu grob für unsere Zwecke. Wie lässt sich die Auflösung auf etwa 0,01°C/Schritt verbessern?

Bessere ADC-Ausgangsauflösung durch Dithering

Wie bereits erwähnt, ist ein ADC in der Regel darauf beschränkt, Änderungen (Schritte) in der Größenordnung eines LSB zu erkennen. Nachdem wir die Referenzspannung auf 1,1 V gesenkt haben, haben wir eine Schrittgröße von etwa 1 mV für das LSB.

Der ADC gibt dann stets ein Vielfaches des Wertes für 1 mV aus. Wenn das Eingangssignal beispielsweise zwischen 99,5 mV und 100,5 mV liegt, wird immer „100 mV" angezeigt. Wenn wir also ein Eingangssignal von 100,1 mV haben, lesen wir immer noch „100 mV" ab. Hier kann uns das Konzept des Dithering weiterhelfen. Diese Technik ist eine der seltenen Anwendungen, bei denen das Vorhandensein von Rauschen die Situation verbessert.

Um besser zu verstehen, wie Dithering funktioniert, lassen Sie uns ein Beispiel für unseren speziellen Fall zeichnen, wie in **Bild 3** dargestellt. Wenn wir einem 100,1-mV-Signal ein zufälliges Rauschen hinzufügen (nehmen wir einmal an, dass das Rauschen eine Amplitude von 1 LSB = 1 mV hat) und schnell viele Messungen durchführen, erhalten wir etwa 90% Messwerte von „100 mV" und 10% „101 mV". Wir können den Mittelwert berechnen und daraus schließen, dass wir tatsächlich ein Eingangssignal von 100,1 mV hatten. Bei größeren Rauschamplituden führen die gemittelten Ergebnisse ebenfalls zu diesem Wert.

Aufgrund der durch das Rauschen verursachten Schwankungen ist dieses Ergebnis nicht vollkommen stabil, aber als Faustregel kann man sagen, dass man bei einer Mittelwertbildung über 100 Stichproben eine 10-fache Verbesserung erhält. Die Verbesserung ist die Quadratwurzel aus der Anzahl der Stichproben. In dieser Anwendung werden für jeden Temperaturmesswert 6.400 Stichproben gemittelt. Die Quadratwurzel hieraus ergibt eine 80-fache Verbesserung.

Wir begannen mit einer Auflösung von 1°C, so dass wir theoretisch die Auflösung auf 0,0125°C verbessern könnten. Die Testdaten, die auf der rechten Seite von **Bild 4** zu sehen sind, zeigen einen Trend von mit 0,01...0,02°C schwankenden Schritten bei der Temperatur, was

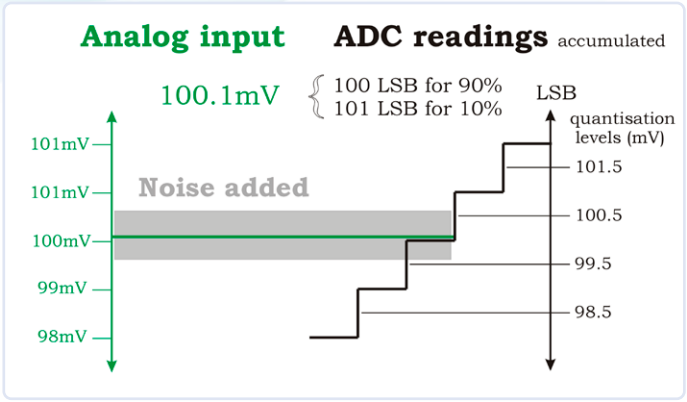


Bild 3. Die Messwerte der Rauschkomponente werden zu denen des Hauptsignals addiert, dann wird aus der Vielzahl dieser Messwerte ein gemitteltes Ergebnis berechnet.

darauf hindeutet, dass wir in der Praxis nahe an dem theoretischen Wert dran sind. Schauen Sie sich die die Daten in der Datei an, die auf der Elektor-Labs-Seite für dieses Projekt [1] herunterladen können und beachten Sie auch die Ergebnisse im Demonstrationsvideo [2]!

Präzision und Genauigkeit

Man beachte, dass der ausgegebene Wert eine hohe Präzision aufweist (Schwankungen von <0,02 °C), aber keineswegs auf diesem Niveau genau ist, was bedeutet, dass die Messwerte mit geringer Schwankung (Präzision) um mehrere Grad vom tatsächlichen Wert (Genauigkeit)

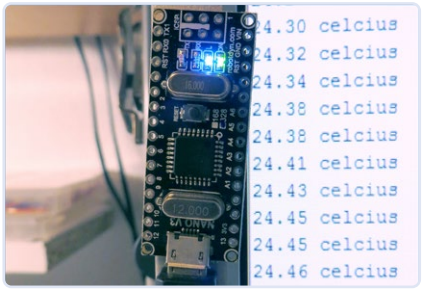


Bild 4. Die in diesem Projekt verwendete Arduino-Nano-Platine. Rechts ist der Verlauf der Messwerte zu sehen.

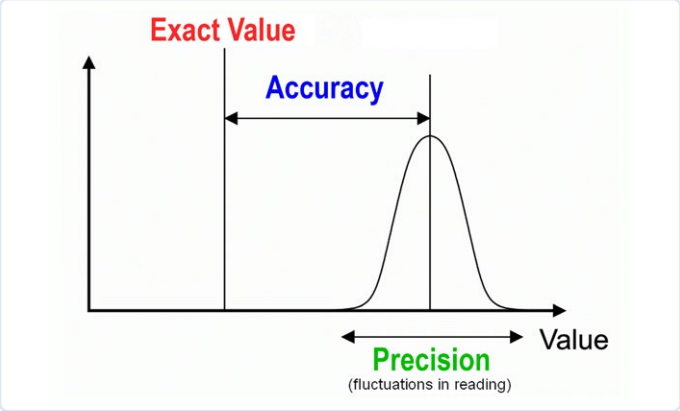


Bild 5. Zusammenhang zwischen Präzision (Precision) und Genauigkeit (Accuracy). (Quelle: Wikimedia Commons, Pekaje / Anthony Cutler, https://commons.wikimedia.org/wiki/File:Accuracy_and_precision.svg)

abweichen können, wie in **Bild 5** dargestellt. Da es bei dieser Anwendung um die Erfassung kleiner relativer Temperatur*änderungen*, nicht aber um absolute Messwerte geht, ist für uns nur die hohe Präzision interessant.

Software

Hier wurde ein Arduino Nano verwendet, aber bei anderen Boards mit demselben ATmega328P-Controller (wie Arduino UNO oder Arduino Pro Mini) sollte es ebenfalls funktionieren. Der Quellcode für dieses Projekt ist unter [1] verfügbar und in seinem Aufbau recht einfach:

1. Initialisierung durch Einstellen der ADC-Referenz und des Eingangsmultiplexers für das Lesen des internen Temperatursensors.
2. Akkumulieren von 6.400 ADC-Messwerten, um die Temperatur mit hoher Präzision zu bestimmen.
3. Das Ergebnis über USB senden
4. Prüfen, ob der Messwert plötzlich abfällt. Wenn ja, wird die Anzeige-LED ausgeschaltet, wenn nicht, wird sie eingeschaltet.
5. Zurück zu Phase 2.

Beschränkungen

- Das Projekt funktioniert am besten auf Arduino-Boards, die mit der SMD-Version des Mikrocontrollers ausgestattet sind. Das größere DIL-Gehäuse ist thermisch besser isoliert und hält Temperaturschwankungen länger stand und ist daher für diese Anwendung nicht geeignet.
- Der Dithering-Prozess verlangsamt die eigentliche Messrate, da wir viele Messwerte erfassen müssen. Hierfür verwenden wir einen 10-kHz-ADC, der durch das Dithering auf etwa zwei Messwerte pro Sekunde verlangsamt wird. Bei Temperaturmessungen ist dies kein Problem.
- Ein ADC ist nicht perfekt, und wenn seine kleinsten Schritte ungleich sind, entstehen zusätzliche Fehler in der Statistik. Einen Einblick in das Thema DNL (differentielle Nichtlinearität) erhalten Sie vielleicht in [3].
- Für Fortgeschrittene: Rauschen mit einer Amplitude von mehr als 1 LSB trägt ebenfalls dazu bei, die DNL-Fehler des ADC etwas auszugleichen.
- Wenn das Rauschen nicht gleichmäßig auf alle Werte verteilt ist, wird auch die Statistik beeinträchtigt.

Rauschquelle für das Dithering

Nun, wir haben keine besondere Quelle für Rauschen hinzugefügt, denn bei diesen kleinen 1-mV-LSB-Pegeln bekommen wir es hier kostenlos von den Schaltungselementen und dem Sensor selbst. Beim Lesen einer Eingangsspannung mit der 5-V-ADC-Referenz stellte ich allerdings fest, dass das „natürliche" Umgebungsrauschen zu gering war, damit ein Dithering ordnungsgemäß funktioniert.

Wie bin ich zu diesem Schluss gekommen? Ich konnte immer noch feststellen, dass die meisten Messwerte um die 5-mV-LSB-Schritte herum lagen. Ich plane, in einem zukünftigen Beitrag zu zeigen, wie dies gelöst werden kann, um ein hochauflösendes Voltmeter zu erstellen.

Einige ungetestete Vorschläge

Verwandeln Sie den Arduino in ein Thermometer

Wenn Sie die durchschnittliche Verlustleistung des Controllers verringern, können Sie diese Schaltung auch als Thermometer verwenden. Dies könnte geschehen, indem man den Controller in den Schlafmodus versetzt und ihn alle paar Sekunden aufweckt, um zu messen.

Externe Sensoren auslesen

Sie könnten externe Dioden an die analogen Eingangspins des ADCs anschließen und einen schwachen Pull-up-Widerstand verwenden, um sie mit einem kleinen Strom vorzuspannen. Auf diese Weise könnten Sie vermutlich mit demselben Softwarekonzept mehrere externe Temperaturen mit hoher Präzision (und geringer Genauigkeit) ablesen.

RG — 220615-02

Über den Autor

Neben seiner beruflichen Tätigkeit, bei der er an rauscharmer Instrumentenelektronik arbeitet, verwirklicht Raymond Schouten Hobbyprojekte, bei denen er winzige Musiksynthesizer und andere kompakte Schaltungen entwickelt. Die meisten seiner Designs zielen darauf ab, mit einfachster Hardware maximale Ergebnisse zu erzielen. Er veröffentlicht auch häufig Projekte auf Elektor Labs, YouTube und auf seiner persönlichen Website rs-elc.nl.

Haben Sie Fragen oder Kommentare?

Haben Sie technische Fragen oder Kommentare zu diesem Artikel? Schreiben Sie bitte an den Autor unter rs.elc.projects@gmail.com oder an die Elektor-Redaktion unter redaktion@elektor.de.

Passende Produkte

- **Arduino Nano**
  www.elektor.de/17002
- **Ashwin Pajankar, *Kickstart to Arduino Nano* (Elektor, 2022)**
  Buch, Taschenbuch, englisch: www.elektor.de/20241
  E-Book, PDF, englisch: www.elektor.de/20242

WEBLINKS

[1] Elektor-Labs-Seite für dieses Projekt: https://t1p.de/ey498
[2] Demo-Video auf YouTube: https://youtu.be/0p6hssAf7Xs
[3] Kurze Wikipedia-Anmerkung über DNL: https://de.wikipedia.org/wiki/Differenzielle_Nichtlinearit%C3%A4t

Wasser-leck-detektor

Verbunden mit der Arduino-Cloud

Source: Adobe Stock

Bild 1. Mein Wasserverbrauchszähler mit angebautem Durchflusssensor.

Von Yves Bourdon (Frankreich)

Als eine Wasserleitung unter meinem Haus unbemerkt brach, verursachte dies einen Schaden von mehreren tausend Euro. Deshalb beschloss ich, ein System zur Überwachung des Wasserverbrauchs zu entwickeln und zu installieren, damit solche kostspieligen Unfälle in Zukunft nicht mehr unbemerkt bleiben. Mit der Arduino Cloud kann ich nun meinen Wasserverbrauch jederzeit und von überall auf der Welt auf meinem Handy überwachen.

Wasser ist kostbar und wird immer teurer. Vor einem Jahr brach eine Wasserleitung unter meinem Haus, ohne dass ich es sofort bemerkte, und es kostete mich mehrere Tausend Euro! Also suchte ich nach einer Möglichkeit, benachrichtigt zu werden, wenn so etwas wieder passiert. Die Wasserwerke installierten einen Zähler mit einem Durchflusssensor (**Bild 1**) und einer kleinen drahtlosen Fernanzeige. Leider war die Zuverlässigkeit nicht ausreichend, und es gab keine Möglichkeit, mich zu alarmieren, außer durch ein „F" auf dem Display, das ein Leck anzeigen sollte...

Nach dem Elektor-Artikel über die Wasserleckanzeiger [1] von Denis Lafourcade habe ich mich entschlossen, sein Projekt (mit seiner freundlichen Genehmigung) zu übernehmen. Allerdings habe ich es ziemlich stark modifiziert, um es in mein bestehendes System zu integrieren. Außerdem wollte ich die Arduino Cloud verwenden, die in der Arduino-Gastausgabe 2022 von Elektor [2] vorgestellt wurde.

Voraussetzungen

Ich wollte von meinem Handy aus auf meine Daten zugreifen können, wofür die Arduino-Cloud-Umgebung besonders geeignet ist. So konnte ich ein Dashboard auf meinem Handy haben, ohne eine spezielle Anwendung entwickeln zu müssen, die ständig aktualisiert werden muss. Sicherheit war wichtig; insbesondere wollte ich keinen Port meines Routers öffnen, um auf meine Daten zugreifen zu können. Und schließlich wollte ich im Falle eines Lecks per Push-Benachrichtigung mit einem akustischen Alarm gewarnt werden.

Spezifikationen

- Nutzung der Arduino-Cloud zur Datensammlung und Anzeige von Statistiken. Ein Diagramm zeigt den täglichen Wasserverbrauch in Echtzeit an. Bis zu drei Monaten an Daten werden in der Cloud gespeichert. Zusätzlich kann eine CSV-Datei (Comma-separated Values) einfach heruntergeladen werden, um die Daten zum Beispiel in einem Tabellenkalkulationsprogramm weiterzuverarbeiten.
- Verwendung eines ESP32-Prozessors und des Entwicklungsboards WiFi Kit 32 von Heltec [3], das unter anderem ein OLED-Display und ein Batterieladegerät enthält.

- Direkte Verbindung (über einen Optokoppler zur Unterdrückung von Störungen) mit dem Reed-Sensor in meinem Wasserzähler, der sich etwa 40 m entfernt an der Hauswand befindet. Für einen Zähler ohne Sensor sind Reed-Relais-Adapter für jeden Zählertyp leicht erhältlich **(Bild 2)**.
- Mit Hilfe von Interrupts misst das System den Wasserdurchfluss meines Zählers in Litern pro Zeiteinheit.
- Jede Stunde wird der Wasserverbrauch in Litern in einer Tabelle gespeichert.
- Jeden Tag um Mitternacht wird der Wert gespeichert, um den Tagesverbrauch zu messen (Verbrauch zum Zeitpunkt t minus Verbrauch um Mitternacht davor).
- Jedes Jahr am 31. Dezember um Mitternacht wird der Verbrauchswert ebenfalls gespeichert. Dies ermöglicht eine einfache Berechnung des Jahresverbrauchs (Verbrauch zum Zeitpunkt t minus Verbrauch am 1. Januar).
- Die Messwerte werden in 24-Stunden-Intervallen gespeichert. Dies ermöglicht die Überprüfung des Wasserverbrauchs auch zu ungewöhnlichen Zeiten. Zum Beispiel reinigt mein Wasserenthärter sein Harz alle 15 Tage um 2:15 Uhr.
- Alle 15 Minuten wird der Verbrauch der letzten 24 Stunden berechnet. Wenn er die Alarmgrenze überschreitet (in meinem Fall 750 Liter/Tag), wird ein akustischer Alarm ausgelöst, eine virtuelle Anzeige auf dem Arduino-Dashboard aktiviert und eine E-Mail oder eine Push-Benachrichtigung über die Arduino-Cloud gesendet, damit rechtzeitig eine Warnung erfolgt!
- Ebenso wird geprüft, ob zwei aufeinanderfolgende Zeiträume ohne Wasserverbrauch (oft nachts) vorliegen. Wenn diese Bedingung nicht erfüllt ist, kann ein Wasserleck vermutet werden und es werden Alarme gesendet.
- Die einzige Wartungsmaßnahme, die erforderlich ist, ist das regelmäßige Ablesen des Wasserzählers. Der Unterschied zwischen der Messung über das Reed-Relais und dem tatsächlichen Wert, den der Zähler anzeigt, ist sehr gering. Ich stelle ihn nur etwa alle zwei Monate nach.

Anbindung an die Arduino Cloud

Bei der Arbeit mit der Arduino-Cloud [4] kann auf die klassische Arduino-IDE auf dem Rechner verzichtet werden. Der Online-Compiler reicht aus, um komfortabel eine Anwendung zu entwickeln und die schrittweisen und transparenten Updates verbessern die Entwicklungsumgebung erheblich. Da OTA-Updates (Over-the-Air) in dieser Umgebung nativ sind, kann man wählen, ob man die Software über einen USB-Port (für den ersten Download obligatorisch) oder aus der Ferne herunterladen möchte, was sehr komfortabel ist.

Es gibt viele Tutorials im Internet, die den Umgang mit der Cloud lehren, aber im Großen und Ganzen ist es nicht so kompliziert. Alles ist kostenlos, wenn man nicht mehr als fünf Variablen in der Anwendung zu aktualisieren hat. Das ist nicht viel, aber es reicht für den Wasserleckanzeiger, wenn man mit begrenzten Anzeigen zufrieden ist: Tagesverbrauch, zwei Tabellen mit Zeitintervallen, Jahresverbrauch und Zählerstand.

Ich selbst habe ein Maker-Abonnement für 5,99 € pro Monat abgeschlossen (es gibt auch Sonderangebote), da ich diese Anwendung für viele Projekte verwende. In der kostenlosen Version ist man auf zwei Anwendungen beschränkt. Mit meinem Maker-Abonnement werden meine Daten drei Monate lang gespeichert und ich kann eine CSV-Datei mit den zeitgestempelten Daten herunterladen.

Wenn Sie alle Funktionen nutzen möchten, die ich entwickelt habe (einschließlich Statistiken über die Betriebszeit und Verbindung mit

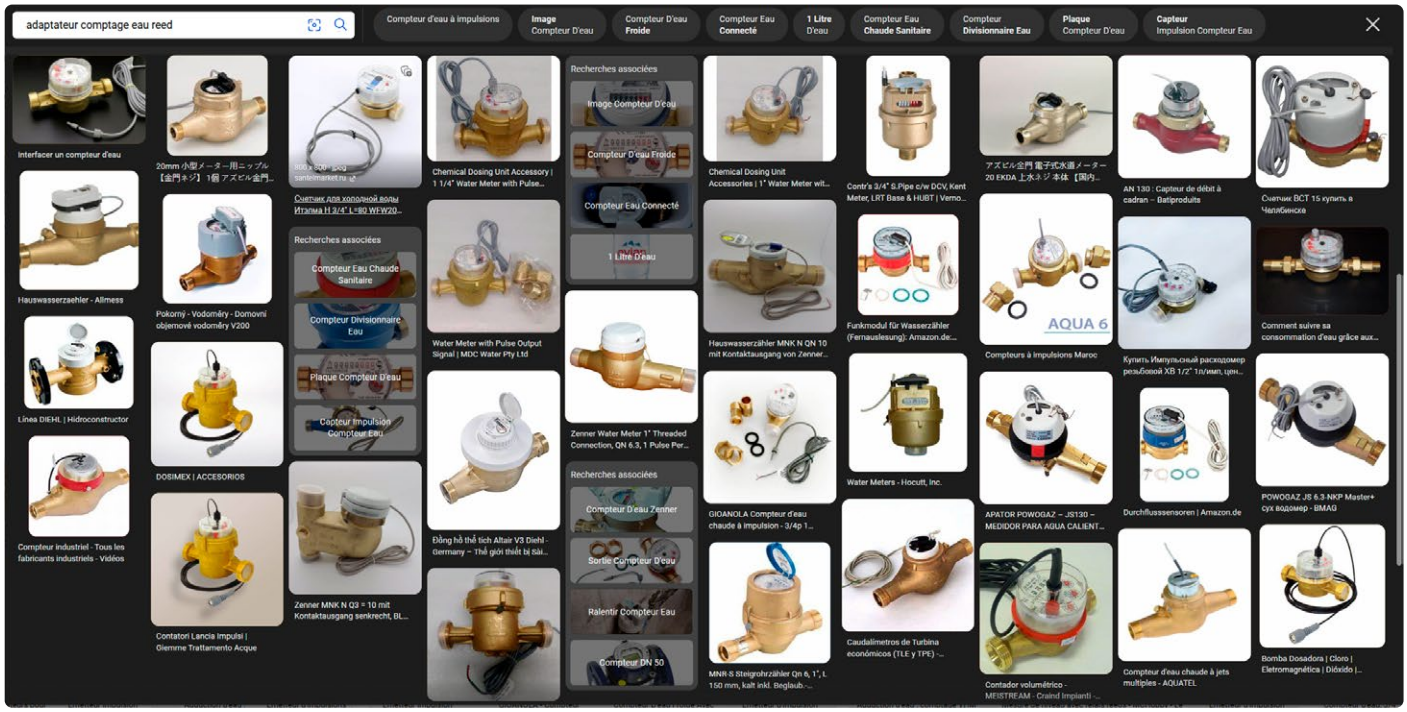


Bild 2. Reed-Sensoren für alle Arten von Zählern sind leicht erhältlich.

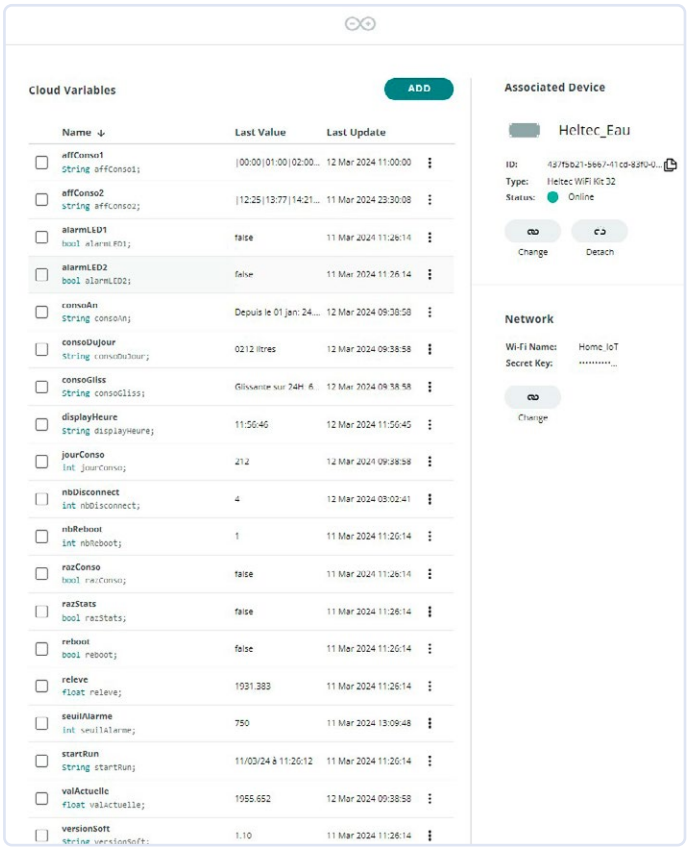


Bild 3. Schritt 2 — Konfiguration eines „Things“.

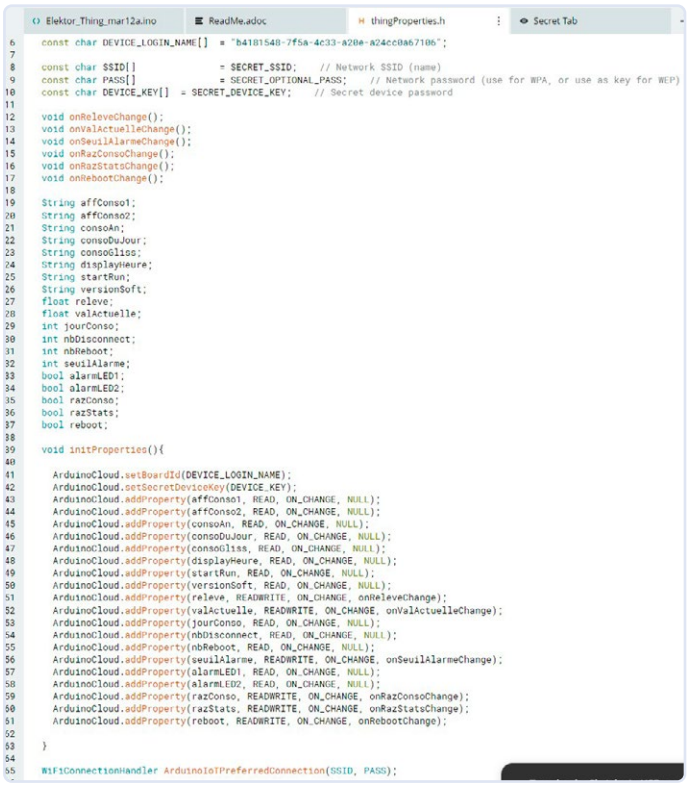


Bild 4. So sieht es aus, wenn alle Variablen definiert sind.

Reset-Befehlen), empfehle ich Ihnen, das Maker-Abonnement für mindestens einen Monat abzuschließen, um es auszuprobieren. Ich habe die Verbindungssicherheit wie folgt optimiert:

- Wenn die WLAN-Verbindung abbricht, erkennt das System dies und versucht, sich erneut zu verbinden.
- Wenn die Verbindung zur Arduino-Cloud unterbrochen wird (was gelegentlich vorkommt), erkennt das Programm dies, und nach mehreren Wiederherstellungsversuchen wird die CPU neu gestartet.

Die lokale Zeit wird wie folgt eingestellt:

- Wenn das System mit der Arduino-Cloud verbunden ist, holt es sich regelmäßig die UTC-Zeit von der Cloud.
- Da ich die lokale Zeit inklusiver der Sommerzeitänderungen benötige, wird die lokale Zeit des ESP32 unter Berücksichtigung dieser Parameter aktualisiert.

Schritt 1: Gerät hinzufügen

Dieser Schritt bereitet Ihr System auf die Cloud-Umgebung vor, indem Sie ihm eine Geräte-ID und einen geheimen Schlüssel zuweisen. Wählen Sie dazu im Menü *Setup Device* die Option *A third-party device* aus, dann *ESP32* und schließlich *Heltec WiFi Kit 32* aus der Dropdown-Liste. Achten Sie darauf, dass die Liste möglicherweise nicht alphabetisch sortiert ist, aber die Auswahlmöglichkeit ist vorhanden. Geben Sie einen Namen für die Anwendung ein. Die Cloud stellt Ihnen dann zwei Variablen zur Verfügung, die Sie mit Hilfe der herunterladbaren PDF-Datei korrekt speichern müssen.

Schritt 2: Ein „Thing“ konfigurieren

Als nächstes müssen Sie die Variablen verknüpfen, die Sie anzeigen oder ändern möchten. Beachten Sie dabei Groß- und Kleinschreibung (siehe **Bild 3**).
Bei der Erstellung müssen Sie die Variablentypen `bool`, `int`, `float`, `string` sowie deren *Read & Write*-Attribute definieren. *Read & Write* ist für eine Variable, die in der mobilen Anwendung geändert werden kann, zum Beispiel für die Alarmgrenze. *Read only* ist für Werte, die wie der Alarmstatus nur angezeigt werden sollen. Lassen Sie die Variable Update *Policy - On change* für alle Variablen aktiviert.
Außerdem müssen Sie die SSID und das Passwort aus dem Netzwerkmenü sowie den zuvor abgerufenen geheimen Schlüssel eingeben. **Bild 4** zeigt das Ergebnis dieses Schrittes.

Schritt 3: Der Sketch

Gehen Sie zum Reiter *Sketch* und wählen Sie *Open full editor*. Sie befinden sich nun in der Entwicklungsumgebung.
Laden Sie die für dieses Projekt benötigen Bibliotheken: *SSD1306.h, OLEDDisplayUi.h, TimeLib.h, math.h und Preferences.h.* Klicken Sie dafür einfach auf den Pfeil im Reiter *Libraries*. Ich habe die Bibliotheken in dem Ordner *Libraries* abgelegt.
Laden Sie den Quellcode von der Projektseite auf Elektor Labs [5] herunter. Öffnen Sie die Datei *Elektor_Thing_mar12a.ino* in der klassischen Arduino-IDE oder einem Texteditor, um das Programm in die Arduino-Cloud-IDE zu kopieren. Alles, was Sie dafür tun müssen, ist, den Quellcode in den ersten Reiter zu kopieren und einzufügen.

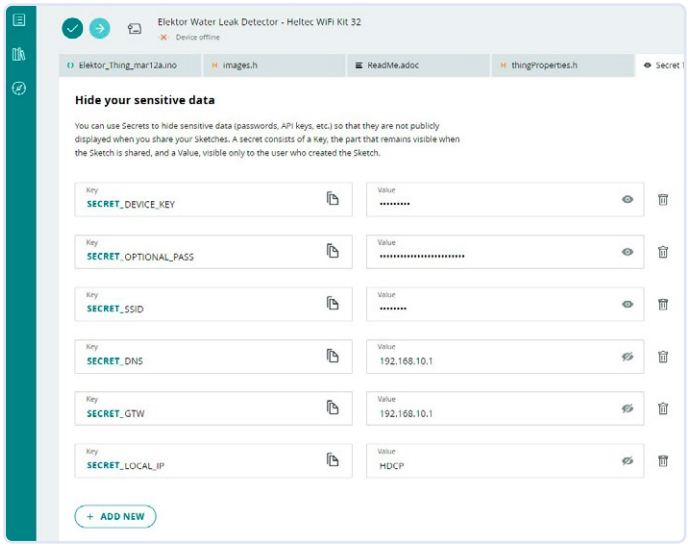


*Bild 5. Füllen Sie die Seite **Secret** wie hier gezeigt aus.*

Ein Bug?

Kürzlich ist mir ein Problem aufgefallen, das auf meinem iPhone, nicht aber auf meinem Tablet auftritt. Im Arduino-Dashboard ist es sehr kompliziert, einen neuen Wert für den Wasserzählerstand, den letzten Stand am 1. Januar oder die Alarmschwelle einzugeben. Aus irgendeinem Grund wird der neue Wert sofort gelöscht, bevor man die Möglichkeit hat, auf den Button *Done* zu tippen.
Um dieses Problem zu umgehen, das nach einem Update von Arduino Cloud auftritt, drücken Sie einfach auf den *Reboot*-Button, geben Sie den neuen Wert ein und warten Sie, bis das System neu gestartet ist. Der neue Wert wird dann übernommen. Dieses Problem könnte aber bereits behoben sein, wenn Sie diesen Artikel lesen.

Erstellen Sie den Tab *images.h*, indem Sie auf das Pluszeichen in der Hauptleiste klicken und den Inhalt meiner Datei kopieren und einfügen. Sie müssen auch die Registerkarte *Secret* vervollständigen, wie in **Bild 5** gezeigt.
Kompilieren Sie das Programm, indem Sie auf die entsprechende Schaltfläche klicken. Wenn alles in Ordnung ist, können Sie das Programm auf Ihr Board laden (nach der Installation des *Cloud Agents*, wie von der IDE vorgeschlagen). Das Kompilieren geht sehr schnell - kein Vergleich zur klassischen IDE.
Hinweis: Ich habe einige Parameter im *Secret*-Tab gruppiert. Neben den üblichen Parameter `SECRET_SSID`, `SECRET_PASSWORD` und `SECRET_DEVICE_KEY` können auch Netzwerkparameter (IP, Gateway, DNS) definiert werden. Ist die IP-Adresse *dhcp oder DHCP*, wird die WLAN-Verbindung über DHCP hergestellt.

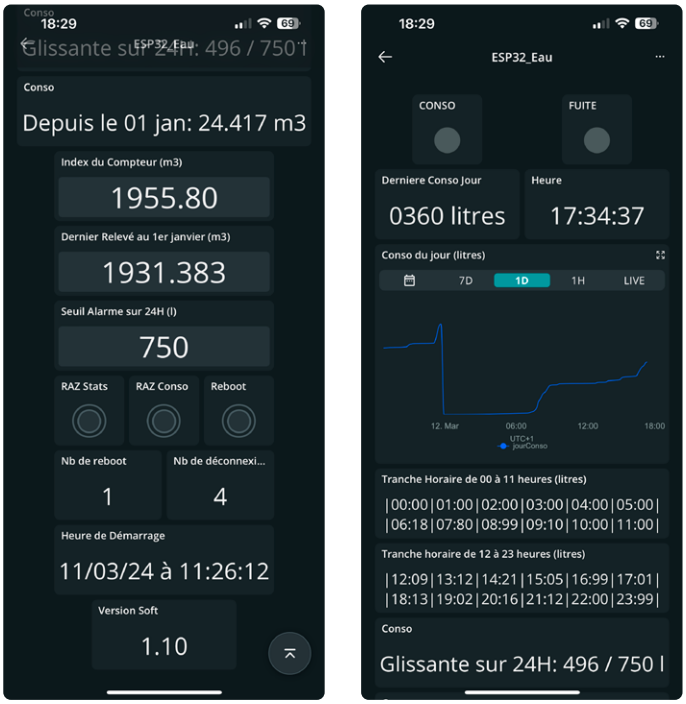


Bild 6. Das sehe ich auf meinem Handy!

Das Dashboard

Ein Dashboard wie meines ist einfach zu erstellen (**Bild 6**). Tippen Sie im Hauptmenü auf + *Dashboard*. Sie können es umbenennen. Drücken Sie dann auf *Add*. Wählen Sie die Art des Widgets, das Sie anzeigen möchten. Anschließend müssen Sie nur noch *Link Variable* wählen, um dem Widget einen Wert zuzuweisen. Bestätigen Sie mit *Done* und wiederholen Sie diesen Vorgang für alle Variablen, die Sie auf dem Dashboard anzeigen möchten..
Ich habe auch die Variable `jourConso` auch einem grafischen Widget zugewiesen, um eine Historie des Wasserverbrauchs zu erhalten.
Wenn Sie alle Variablen zugewiesen haben, wählen Sie die Editierfunktion (zwei gekreuzte Pfeile), um Ihre Widgets zu verschieben und zu vergrößern.
Klicken Sie bitte auf das Telefonsymbol, um ein anderes Layout für Ihr Mobilgerät zu erhalten. Das Erstellen eines Dashboards macht Spaß und es dauert nur wenige Minuten, um ein ziemlich beeindruckendes Ergebnis zu erzielen.

E-Mail- und Push-Nachrichten

Nichts könnte einfacher sein. Gehen Sie einfach im Hauptmenü auf die Seite *Triggers*. Klicken Sie auf + *Trigger*, verknüpfen Sie dann beispielsweise die Variable `alarmLED1` und wählen Sie aus, ob Sie eine E-Mail, eine Push-Nachricht oder beides versenden möchten.

Die Schaltung

Ich habe die gleiche Schaltung (**Bild 7**) wie in einigen anderen Projekten wiederverwendet, die auf meiner Elektor-Labs-Seite [6] zu finden sind (ESP32-Thermostat, VMC-Regler, Kraftstoffmesser, Linky-Analysator und Lastabwurf, NTP-Server).
Ich habe mein Gerät in der Nähe meines Patchpanels platziert, das von einer USV unterstützt wird, aber man kann auch eine wiederaufladbare Lithiumbatterie anschließen (das Heltec-Modul hat einen Anschluss dafür), die sich automatisch auflädt und die Stromversorgung für etwa 20 Minuten aufrechterhält. Da die Daten periodisch gespeichert werden, geht aber nur der Wasserverbrauch der letzten 15 Minuten verloren, wenn Sie keine unterbrechungsfreie Stromversorgung haben.
Der Wasserzähler ist fast 40 Meter von meinem Haus entfernt. Um die Zuverlässigkeit der Impulszählung zu gewährleisten, habe ich statt eines RC-Filters einen Optokoppler verwendet, um Störungen aus der Leitung zu blockieren. Er bietet nicht nur eine galvanische Trennung, sondern filtert auch Störimpulse heraus, die zu schwach sind, um die LED des Optokopplers zum Leuchten zu bringen. Etwa 10 mA bei

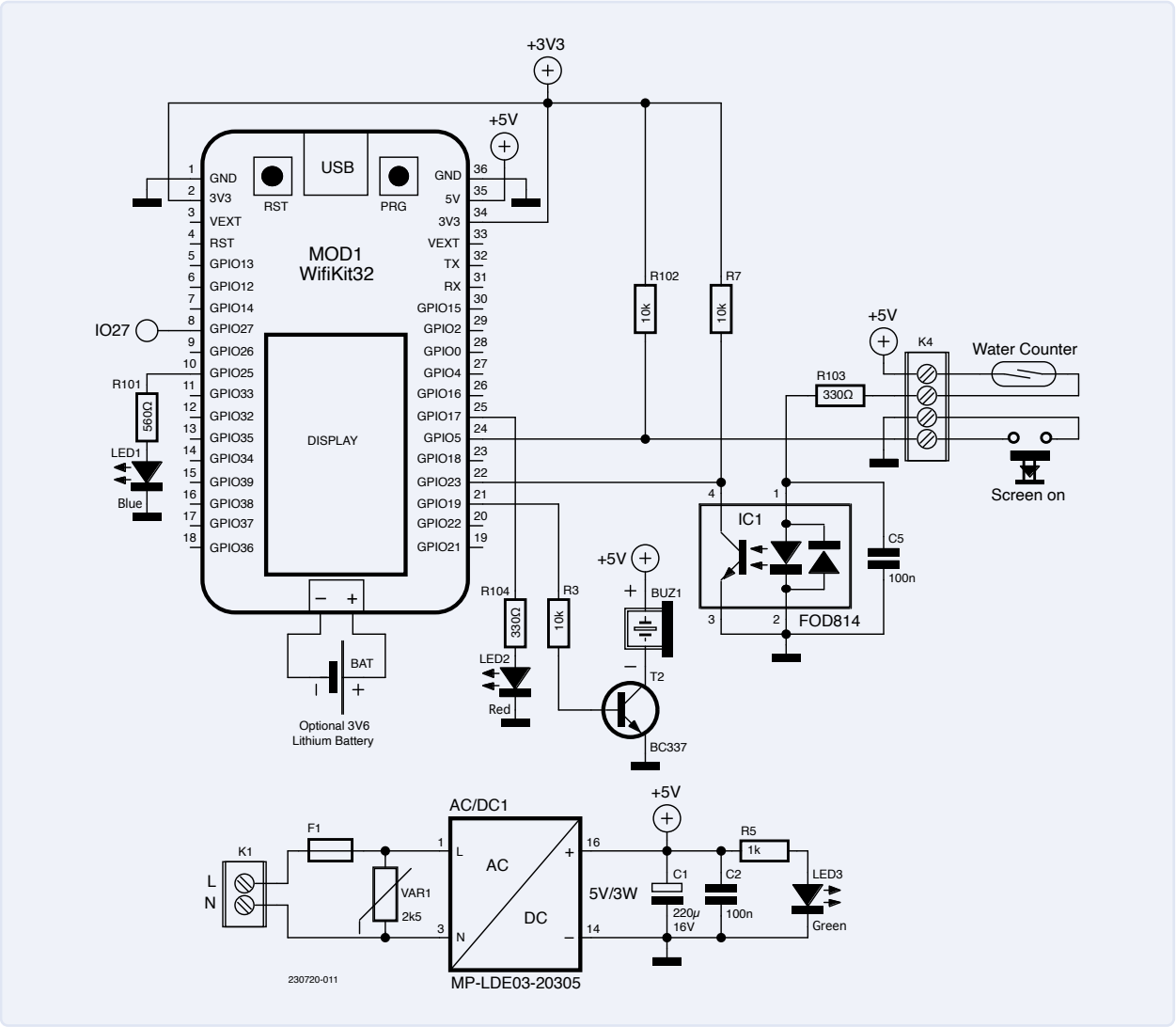


Bild 7. Das Schaltbild des Wasseraustrittsdetektors ist eine angepasste Version des ESP32-Thermostats [7]. Es wurden die Widerstände R101, R102 und R103 hinzugefügt. Der Ausgang des Optokopplers wurde auf GPIO23 umgeleitet.

Listing 1: Die Interrupt-Routine.

```
/* Interrupt counting routine
 masking over 750 ms (the transmitter produces pulses of approximately 200 ms for each 0.51 of water consumed)
 at the fastest, 1 pulse every 725 ms, which corresponds to about 2.5 m3 per hour (physical limit of my counter)
 ------------------------------------------------------------*/
 void IRAM_ATTR handle Interrupt () {
   static unsigned long lastInterrupt = 0;
// debouncing of the pulse - we must wait 750 ms before processing another interrupt
   unsigned long interrupt = millis();  // start of interrupt
   if (interrupt - lastInterrupt >= 750) { // last interrupt more than 750 ms ago?
     portENTER_CRITICAL_ISR(&mux);
     indexCounter++;  // cumulative counter index, 1 pulse = 0.5 liter
     indexJour++;  // daily counter index
     portEXIT_CRITICAL_ISR (&mux);
     flagInterrupt = true;  // an interrupt has been detected
     digitalWrite(SENSOR_LED, HIGH);
// blue LED (sensor) is ON - will be put to OFF 80 ms later in the loop program
   }
   lastInterrupt = interrupt;  // reset and memorize the last interrupt to manage debouncing and blue LED
 }
```

3,3 V sind erforderlich, um die LED korrekt zum Leuchten zu bringen. Dies hat zudem den positiven Nebeneffekt, dass die Kontakte des Reedrelais nicht oxidieren. Ein 100-nF-Kondensator reicht aus, um die Verbindung zum Reed-Sensor zu filtern.
Zwei LEDs geben Statusinformationen. Die blaue LED1, die an IO25 (SENSOR_LED) angeschlossen ist, leuchtet bei jedem an IO23 empfangenen Impuls für circa 100 ms auf. Wenn die rote LED2 an IO17 (CONNECT_LED) leuchtet, bedeutet dies, dass das System nicht mehr mit der Arduino-Cloud verbunden ist.

Die Software

Die größte Schwierigkeit des Projekts bestand darin, keinen Impuls vom Wasserzähler zu verpassen. Darauf wurde beim Schreiben der Interruptroutine (**Listing 1**) besonders geachtet. Bei jeder Unterbrechung leuchtet die blaue LED auf. Die globale Variable `Last_interrupt` speichert den Zeitpunkt der Unterbrechung, so dass die Hauptschleife die LED etwa 100 ms später ausschaltet (blinkt).
Die Variable `indexCounter` vom Typ `volatile int` enthält die Anzahl der Impulse, also die Anzahl der verbrauchten halben Liter Wasser. Diese Variable kann nur innerhalb der Interrupt-Serviceroutine mit den Befehlen `portENTER_CRITICAL_ISR(&mux)` und `portEXIT_CRITICAL_ISR(&mux)` geändert werden.
Mein Sensor gibt zwei Impulse pro Liter ab. Dies kann in der Software als 1-2-4 Impulse pro Liter konfiguriert werden. Abhängig von den Eigenschaften des Sensors wird der Wert im Hauptprogramm mit Hilfe von Bitverschiebungen durch 1, 2, 4, 8 und so weiter geteilt:

```
Total_Counter = indexCounter >> pulsParLitre;
  // 2 pulses / l
```

Die `volatile int`-Variable `indexJour` entspricht der Variable `jourConso` in der Arduino-Cloud. Sie wird nicht aktualisiert, wenn der Wert manuell im Arduino-Dashboard geändert wird. Wie `indexCounter` wird auch sie durch `pulsParLitre` geteilt.

Alle 15 Minuten

- Die Variablen `indexCounter` und `indexJour` werden im nichtflüchtigen Speicher (NVS) abgelegt.
- Ein Array `conso[24]` enthält den Wasserverbrauch für jede Stunde (aus Darstellungsgründen können nur 99 l/h gezählt werden). Dies ermöglicht die Leckageüberwachung: Wenn es keine zwei aufeinanderfolgenden Stunden mit Nullverbrauch gibt, bedeutet dies, dass kontinuierlich Wasser fließt. In diesem Fall wird ein Alarm ausgelöst und die Variable `alarmLED2` auf 1 gesetzt.
- Die Gesamtmenge des seit Mitternacht verbrauchten Wassers wird ebenfalls überprüft. Wenn sie einen vom Benutzer definierten Schwellwert überschreitet, wird von einem Überverbrauch ausgegangen. In diesem Fall wird ein Alarm ausgelöst und die Variable `alarmLED1` auf 1 gesetzt.
- Im Alarmfall versendet die Arduino-Cloud eine Push-Benachrichtigung und eine E-Mail.

Alle 60 Minuten

wird der Verbrauch für die nächste Stunde auf Null gesetzt und der Bildschirm für die Dauer des Bildschirmschoners (fünf Minuten, siehe unten) eingeschaltet. Das kleine OLED-Display ermöglicht die Anzeige

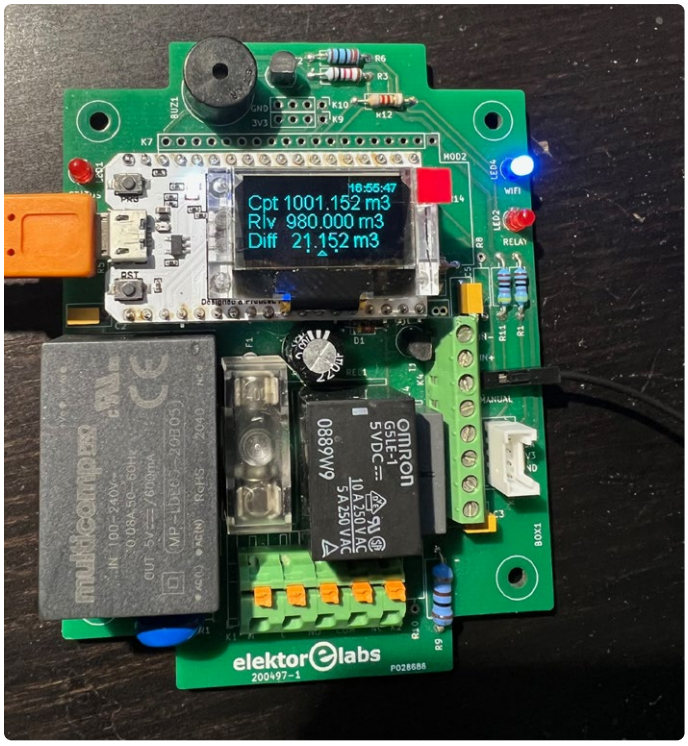


Bild 8. Die Elektronikplatine mit dem Heltec-Modul ESP32.

Bild 9: Die Platinenmontage in einem Gehäuse.

von drei Seiten, die alle fünf Sekunden umgeschaltet werden. Auf diese Weise können Netzeigenschaften, Uhrzeit, verschiedene Verbrauchszähler und gegebenenfalls die Art des ausgelösten Alarms angezeigt werden. Um den 24-Stunden-Verbrauch auf nur zwei Zeilen darzustellen, wird die Routine `formatData()` verwendet. Einige zusätzliche Funktionen ermöglichen

- das Zurücksetzen aller Stundenintervalle auf 0 (normalerweise nur während Tests).

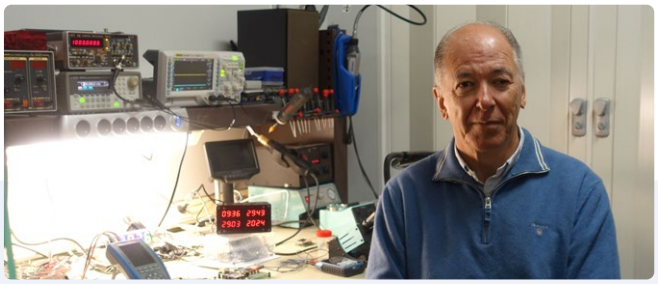
- die Prüfung, ob die Arduino-Cloud ausgeschaltet ist (was natürlich nicht zu Zählfehlern führt). Wenn das System nach 600 s (10 Minuten) immer noch keine Verbindung zur Arduino-Cloud herstellen kann, wird ein Netzwerkproblem angenommen und das System neu gestartet. Die Anzahl der Neustarts kann eingesehen werden, und bei jedem Neustart werden Datum und Uhrzeit des Neustarts angezeigt.
- das Zurücksetzen des jährlichen Wasserverbrauchs auf Null am 31. Dezember um Mitternacht, und der neue Wert des Wasserzählers (`Last_Counter`) wird gespeichert.

Um zu überprüfen, ob alles wie erwartet funktioniert, habe ich einen Impulszähler parallel angeschlossen und konnte keinen Unterschied zwischen meinen Messungen und der Anzeige des Zählers feststellen. Da ich ein OLED-Display verwende, das im Dauerbetrieb schnell altert, habe ich auch eine Bildschirmschoner-Routine eingebaut, die das Display nach fünf Minuten abschaltet. Das Display schaltet sich wieder ein, wenn ein Alarm ausgelöst wird oder eine Taste gedrückt wird, die mit dem System verbunden ist.

Erfolgreiche Überwachung

Für die Hardware habe ich die gleiche Platine wie für das ESP32-Thermostat [7] verwendet (siehe **Bild 8** und **Bild 9**). Der Vorteil dieser Platine ist, dass sich die 230 V-Netzspannungsversorgung auf der Platine befindet und alle Anschlüsse ordnungsgemäß mit 3,81-mm-Klemmen ausgeführt sind.
Der in diesem Artikel beschriebene Wasserleckdetektor ist nun seit über einem Jahr in Betrieb. Bisher sind mir keine Fehler aufgefallen, weil oder obwohl ich das System regelmäßig mit Verbesserungen und Korrekturen aktualisiert habe.
Vor einigen Monaten begann eine Toilette außerhalb des Hauses ganz leise zu lecken. Das System meldete einen kontinuierlichen Wasserverlust von etwa zwanzig Litern pro Stunde von Mitternacht bis 7 Uhr morgens. Ich bemerkte den Alarm, als ich aufwachte. Ohne den Detektor hätte sich dieses Ereignis leicht zu einer ökologischen und finanziellen „Katastrophe" entwickeln können.
Mit dem System kann ich auch sehen, ob mein Wasserenthärter um 2 Uhr morgens sein Harz gereinigt hat und wie viel Wasser dafür verbraucht wurde. Und es kann auch anzeigen, ob er defekt ist.
So ist der Wasserleckanzeiger zu einem wertvollen Bestandteil unseres internen elektrischen und elektronischen Systems geworden.

SE — 230720-02



Über den Autor

Yves Bourdon, der sich schon in jungen Jahren für Elektronik und Technologie begeisterte, studierte Ingenieurwissenschaften an der INSA Lyon. Noch vor Abschluss seines Studiums gründete er 1981 sein erstes Unternehmen ERIM. Im Jahr 1991 entwickelte er den ersten französischen PC mit einem Intel-286-Prozessor, der nur 10 cm × 10 cm groß war. Nach dem Verkauf seiner Unternehmen im Jahr 2009 widmete Yves einen Großteil seiner Zeit der Unterstützung innovativer Unternehmen bei ihrer Entwicklung. Unter anderem war er mehr als zehn Jahre lang ehrenamtlicher Präsident von Cap'Tronic (einer öffentlichen Einrichtung). Mit 66 Jahren ist Yves nun im Ruhestand und verbringt einen Großteil seiner Zeit in seinem Elektroniklabor, wo er an ESP32-basierten Anwendungen forscht.

Haben Sie Fragen oder Kommentare?

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

Passende Produkte

- **LILYGO T-Display ESP32 Entwicklungsboard (16 MB)**
  www.elektor.de/19774
- **D. Ibrahim, *The Complete ESP32 Projects Guide*, Elektor 2019**
  Taschenbuch, englisch: www.elektor.de/18860
  E-Buch, PDF, englisch: www.elektor.de/18869

WEBLINKS

[1] . Lafourcade, „Wasserverbrauchsüberwachung mit ESP32", Elektor 7/2019: https://www.elektormagazine.de/magazine/elektor-99/50884
[2] S. Romero, „Vernetzte Projekte, einfach eingerichtet", Elektor Gastausgabe 12-2022/1-2023: https://www.elektormagazine.de/magazine/elektor-269/61274
[3] Heltec WiFi-Kit 32: https://heltec.org/project/wifi-kit32-v3/
[4] Arduino-Cloud: https://cloud.arduino.cc
[5] Dieses Projekt auf Elektor Labs: https://elektormagazine.de/labs/esp32-water-leak-detector-connected-to-iot-arduino
[6] Die Projekte des Autors auf Elektor Labs: https://elektormagazine.de/labs/13670/ybourdon/projects
[7] Y. Bourdon, „ESP32-verbundenes Thermostat", Elektor 9-10/2021: https://www.elektormagazine.de/magazine/elektor-182/59862

Quarze

Bemerkenswerte Bauteile

Von David Ashton (Australien)

Von natürlichen bis hin zu hochentwickelten synthetischen Quarzen spielen diese Bauteile eine zentrale Rolle in der Elektronik, da sie eine präzise Frequenzsteuerung für Bauteile/gruppen wie Mikrocontroller und Funkgeräte ermöglichen. Werfen wir einen Blick auf ihre praktischen Anwendungen in der Elektronik, von der Computertechnik bis hin zu Funksystemen.

Bild 1. Ein kleiner Scheibenquarz mit einer Resonanzfrequenz von einigen Megahertz aus dem Jahre 1975 für die Platinenmontage, bei dem die Abdeckung entfernt wurde, und ein größerer Quarzblock (100 kHz) in einer Röhre von 1959.

Eines der seltsamsten Bauteile, mit denen ein Elektronikanfänger zurechtkommen muss, ist der Quarz. Im Grunde genommen handelt es sich um ein Quarzplättchen mit beidseitig aufgebrachten Elektroden. Der Quarz ist eines der am häufigsten verwendeten und am besten erkennbaren elektronischen Bauteile. Man findet mindestens einen auf so ziemlich jeder Platine mit einem Mikrocontroller und auch in jedem Funksystem. Beide Geräte benötigen eine stabile Frequenzreferenz - für Mikrocontroller das Taktsignal, mit dem sie arbeiten und ihre Kommunikation mit der Außenwelt synchronisieren, und für Funkgeräte die Frequenz, mit der sie betrieben werden.

Die Entwicklung der Quarze

Vor etwa dem Jahr 1925 verwendeten die meisten Oszillatoren Resonanzkreise - eine Kombination aus (normalerweise) einer festen Spule und einem variablen Kondensator. Doch selbst bei einem guten Design waren sie anfällig für Frequenzabweichungen.

1880 entdeckten Jacques und Pierre Curie den piezoelektrischen Effekt: Ein Quarz mit Elektroden auf beiden Seiten verformt sich, wenn eine Spannung an die Elektroden angelegt wird. Wird dieser Effekt für eine positive Rückkopplung in einem Verstärkerschaltkreis genutzt, entsteht ein Oszillator. Im Ersten Weltkrieg nutzte Paul Langevin diesen Effekt, um Ultraschallfrequenzen zu erzeugen. Im Jahr 1925 setzte Westinghouse in seinem Radiosender KDKA einen Quarzoszillator ein, und die Quarzsteuerung wurde bald zum Normalfall. Bis zum Zweiten Weltkrieg wurden natürliche Quarzkristalle verwendet, die meisten davon aus Brasilien. Die Bell Laboratories entwickelten ein Verfahren zur Züchtung synthetischer Quarzkristalle, und bis 1970 waren die meisten Quarze synthetisch.

Quarzeigenschaften vestehen

Ein Quarz entspricht einem präzisen, sehr hoch abgestimmten Schaltkreis mit hoher Güte (Q-Wert). Quarze haben eine serielle (niedrige Impedanz) und eine parallele (hohe Impedanz) Resonanzfrequenz. Unterhalb von 30 MHz wird in der Regel eine Frequenz zwischen den beiden verwendet, während oberhalb davon die serielle Resonanz die Norm ist. Quarze haben unterschiedliche Eigenschaften, je nachdem, welcher Schnitt des Quarzes verwendet wird, das heißt, wo der Quarzkristall relativ zu seinen Achsen eingeschnitten ist. Es gibt noch viele andere Variablen, und die Quarzphysik ist teuflisch kompliziert. Einige Quarze, insbesondere für hohe Frequenzen, können so hergestellt werden, dass sie mit der dritten, der fünften oder sogar der siebten Harmonischen (Oberton) ihrer natürlichen Frequenz arbeiten.

Quarze sind temperaturabhängig, und es werden Techniken eingesetzt, um diesen Makel zu überwinden - Temperaturkompensation

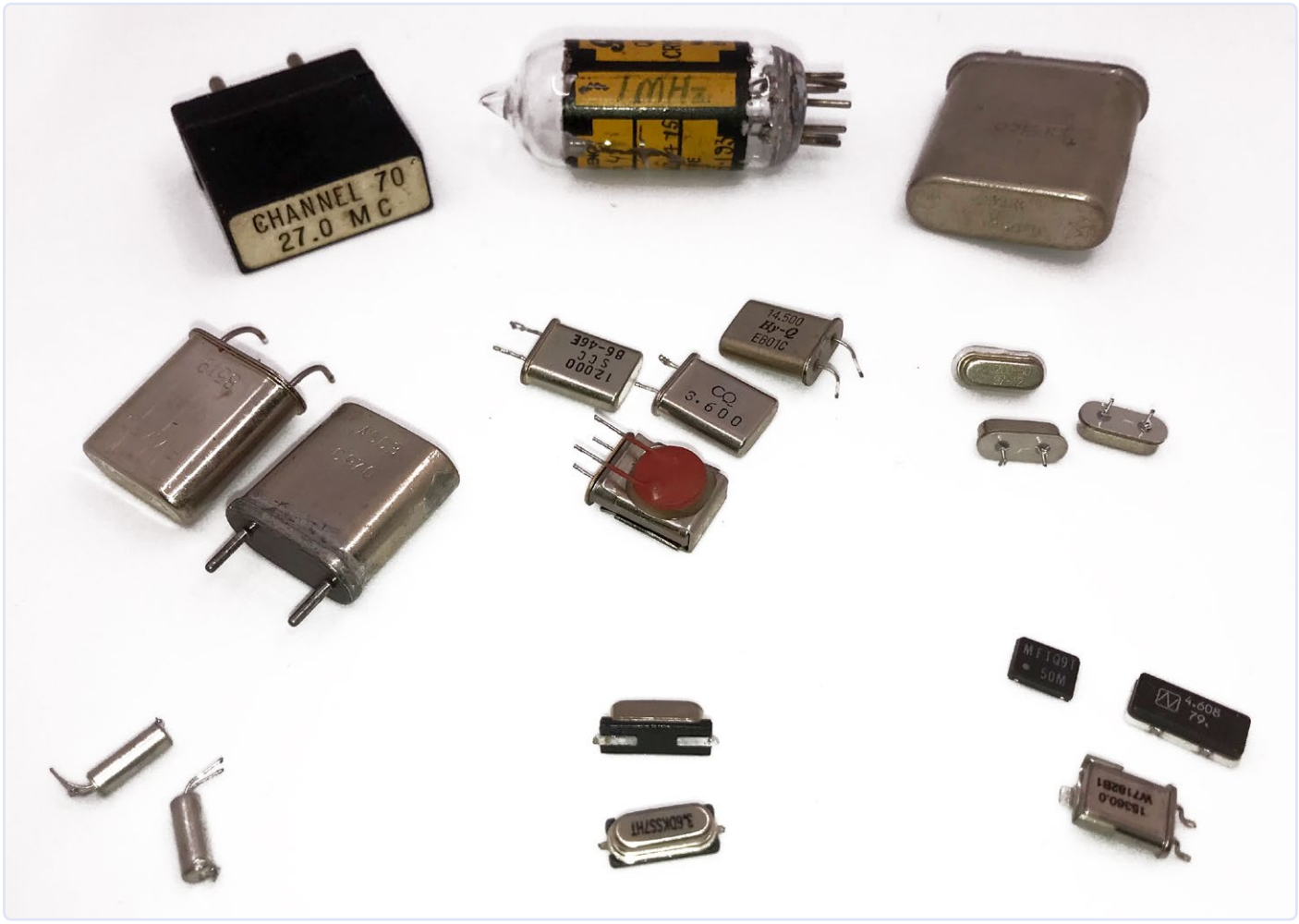


Bild 2. Verschiedene alte und moderne Quarze. Oben: ältere Typen. Mitte (von links nach rechts): HC6/U, HC49/U (einer mit einem roten PTC-Widerstand, der als Ofen dient) und HC49S mit Low-Profile-Typen. Unten (von links nach rechts): 32.768-Hz-Uhrenquarze, HC49SMD und andere oberflächenmontierbare Typen.

im Oszillatorschaltkreis oder sogar ein richtiger Ofen, um den Quarz auf einer bestimmten Temperatur über der Umgebungstemperatur zu halten. Die Frequenz kann durch „Ziehen" präzise eingestellt werden - in der Regel durch eine einstellbare serielle oder parallele Induktivität oder einen Kondensator, der die natürliche Resonanzfrequenz leicht verändert.

Quarze sind ein wesentlicher, wenn auch manchmal missverstandener Bestandteil der meisten modernen elektronischen Geräte. **Bild 1** zeigt den sehr unterschiedlichen inneren Aufbau zweier Quarze und **Bild 2** einige alte und moderne Quarze in Gehäusen.

SG — 240214-02

Haben Sie Fragen oder Kommentare?

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

Über den Autor

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

Universeller Garten-Logger

Ein Schritt auf dem Weg zur KI-gestützten Gartenarbeit

Von Gamal Labib (Ägypten)

Intelligente Bewässerung, die von Sensoren gesteuert wird, spart Wasser und hilft Pflanzen, die durch unzureichende Bewässerung in Stress geraten sind. Der Übergang zu einer intelligenten Bewässerung auf der Grundlage von Maschinellem Lernen wäre ein noch größerer Schritt. In diesem Artikel stelle ich einen Datenlogger vor, der den ersten Schritt in Richtung zum intelligentem Garten markiert.

Die Automatisierung der Gartenbewässerung entlastet den Gärtner von der Arbeit der täglichen manuellen Bewässerung. Doch trotz aller Automatisierung können bei ineffizienter Wasserverteilung immer noch verfärbte Flecken im Rasen auftreten und Pflanzen Anzeichen von Trockenstress zeigen.

Intelligente Bewässerung ist seit Jahrzehnten im Fokus von Herstellern und Forschern. Die gesammelten Daten von Sensoren, die im Garten verteilt sind, stellten den Zauberstab dar, mit dem Bewässerungspläne je nach Bedarf aktiviert oder deaktiviert werden konnten. Der beliebteste Sensor, der zu diesem Zweck eingesetzt wird, misst die Bodenfeuchtigkeit unter den Pflanzenwurzeln. Bei trockenen Bodenverhältnissen wird die Bewässerung aktiviert, bis die Wurzeln gut durchfeuchtet sind [1].

Künstliche Intelligenz bringt die Bewässerungstechnik in eine andere Dimension. Dabei sind mehrere Ansätze möglich. In unserem Projekt [2] erkennt eine simple Farbkamera mit Techniken des Maschinellen Lernens (ML), ob sich eine Pflanze aufgrund von Wassermangel im Trockenstress befindet. In einem anderen Beispiel [3] wird Maschinelles Lernen zur Analyse der Messwerte von Feuchtigkeit, Temperatur und pH-Wert [4] [5] eingesetzt, um den Bewässerungsbedarf des Bodens vorherzusagen.

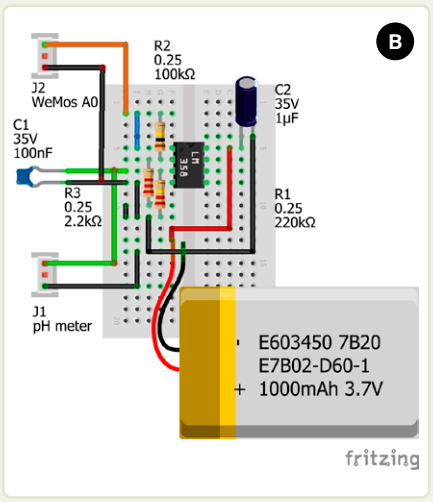
Anwendungen des Maschinellen Lernens sind im Allgemeinen auf eine große Menge von für das System relevanten Datensätzen angewiesen, die zum Trainieren des Modells verwendet werden. Neue Daten, die das Modell noch nie gesehen hat, können dann in das Modell eingespeist werden, das im Gegenzug nützliche Informationen darüber liefert, und es können dann entsprechende Maßnahmen ergriffen werden. Die in ML-Anwendungen verwendeten Daten können unterschiedlich formatiert sein, aber eines der beliebtesten Formate ist das der kommagetrennten Werte (Comma-separated Values, CSV). Bei der Planung der KI-Gartenarbeit müssen wir uns zunächst auf das Sammeln von Daten konzentrieren. Aus diesem Grund habe ich eine Vielzahl von Sensoren aufgenommen, die eine breite Palette von Parametern im Garten abdecken.

Quelle: Adobe Stock

Anschließen eines pH-Messgeräts

Der pH-Wert des Bodens schwankt über lange Zeiträume und braucht Wochen und manchmal Monate, um messbare Veränderungen zu zeigen, vor allem dann, wenn er vom Landwirt neu eingestellt wird. Daher betrachtete ich die tägliche Überwachung des Boden-pH-Wertes als ein sekundäres Ziel, das manuell oder als Zusatz zu diesem Projekt erreicht werden kann. Als ich mir die auf dem Markt erhältlichen digitalen Boden-pH-Sensoren ansah, kam ich zu dem Schluss, dass sie recht (heißt: zu) teuer sind, insbesondere diejenigen, die sich für die Verbindung mit Arduino-Projekten eignen.

Der billigste Sensor für dieses Projekt war mein manueller Sensor, den ich bereits für meinen Garten hatte und der erstaunlicherweise keine Batterien zum Betrieb benötigt (siehe **Bild A**). Der Sensor hat zwei Elektroden aus verschiedenen Metallen, zwischen denen der Boden als Elektrolyt dient. Eine solche Anordnung bildet eine elektrochemische Zelle, die natürlich recht schwach ist, aber stark genug, um das kleine analoge Messgerät zu betreiben. Mit einem Operationsverstärker und einer kleinen Batterie konnte ich dieses Signal verstärken und zusammen mit den anderen Sensoren an den Analogeingang des WeMos leiten (**Bild B**).



Ein „universeller" Datenlogger könnte eine CSV-Datei mit den Umgebungswerten von Temperatur und Luftfeuchtigkeit, Sonneneinstrahlung, Bodenfeuchtigkeit, pH-Wert und NPK (Stickstoff-Phosphor-Kalium-Fertilität und Nährstoffmessungen) und so weiter erstellen. Auch die Wassermenge, die den Boden erreicht, könnte erfasst werden, so dass die Effizienz der zur Bewässerung eingesetzten Sprinkleranlagen ermittelt werden kann. Doch leider sind digitale pH- und NPK-Sensoren zu teuer, um damit zu experimentieren. Es ist mir jedoch gelungen, einen billigen analogen pH-Sensor so anzuschließen, dass er mit Mikrocontrollern zusammenarbeitet, wie im Textkasten **Anschluss eines pH-Messgeräts** berichtet wird.

Aufbau des Projekts

Hirn des Gartenloggers ist ein WeMos D1 Mini, ein Modul, das auf dem ESP8266-Mikrocontroller basiert. Mit seiner Fähigkeit zur WLAN-Kommunikation lässt es sich leicht in das häusliche IoT integrieren. Aktuell arbeiten vier analoge Sensoren für den Logger: ein TEMT6000-Lichtsensor zur Messung der Sonneneinstrahlung auf die Pflanzen, ein Regentropfensensor, der Niederschlag erkennt (was in meiner Heimat nicht oft vorkommt) und den ich auch zur Steuerung der Sprinkleranlage verwenden möchte, ein Wasserstandssensor zur Messung des gesammelten Regen- oder Sprinklerwassers, das im Bereich des Datenloggers auf den Boden trifft und ein Bodenfeuchtesensor, der die Feuchte (oder vielmehr die Trockenheit) an den Pflanzenwurzeln erkennen soll. Zusätzlich gibt es einen einzelnen digitalen Sensor (ein DHT11), der die Temperatur- und die Luftfeuchte im Bereich des Loggers ermittelt. Einige der analogen Sensoren verfügen auch über einen digitalen Ausgang, der je nach dem (mit einem eingebauten Potentiometer) voreingestellten Schwellwert ein- oder ausgeschaltet werden kann. Der Gartenlogger selbst ist in **Bild 1** dargestellt.

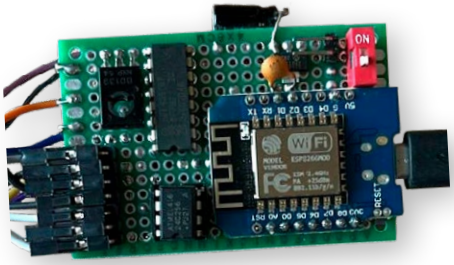


Bild 1. Ansicht des Prototypen mit dem Wemos-Modul.

Da der ESP8266 nur einen einzigen Analogeingang (A0) besitzt und ich viele analoge Sensoren daran anschließen möchte, habe ich einen Analogmultiplexer 4052 vor den Eingang geschaltet. Damit ist der Anschluss von bis zu acht analogen Sensoren an den Logger möglich. Der Analogmultiplexer belegt zwei GPIOs des D1 Mini, nämlich hier Pin 16 (D0) und Pin 14 (D5), um einen der vier analogen Sensoren auszuwählen und ihn mit dem Analogeingang des Mikrocontrollers zu verbinden.

Da der Logger von einer Batterie mit Energie versorgt wird und über längere Zeiträume laufen soll, ist Energieeffizienz von größter Bedeutung. Während der Intervalle, in denen die Aufzeichnung deaktiviert ist, werden die Sensoren abgeschaltet, indem kurzerhand ihre Masseverbindung von einem NPN-Transistors (BD139) unterbrochen wird. Der Transistor kann 1,5 A aufnehmen; seine Basis wird direkt vom GPIO15/D8-Pin des D1 Mini angesteuert. Die Transistorbasis verträgt eine Spannung von bis zu 5 V. Wenn GPIO15 auf High gesetzt wird, geht der Transistor in die Sättigung und aktiviert die Sensoren. Andernfalls wird er auf Low gesetzt, um zu sperren und so Energie zu sparen.

Da der Logger universell einsetzbar sein soll, habe ich mich dafür entschieden, die reinen analogen Messwerte der Sensoren aufzuzeichnen, was dem Benutzer die Freiheit gibt, die Sensoren zu kalibrieren und die Werte möglichst flexibel zu interpretieren. Der Regentropfensensor war eine Ausnahme, bei der ich mich sowohl auf seine analogen als auch auf seine digitalen Ausgänge verließ, wie später beschrieben wird. Zur Überwachung seines digitalen Ausgangs habe ich den Pin GPIO13/D7 verwendet.

Bild 2 veranschaulicht die Verdrahtung der Komponenten des Projekts. Regentropfen- und Bodenfeuchtesensor sind vollständig passiv und benötigen daher Treibermodule, um einen messbaren Analogausgang zu schaffen (die mit den in der Stückliste angegebenen Produkten geliefert werden), während die übrigen Sensoren autark sind. **Bild 3**

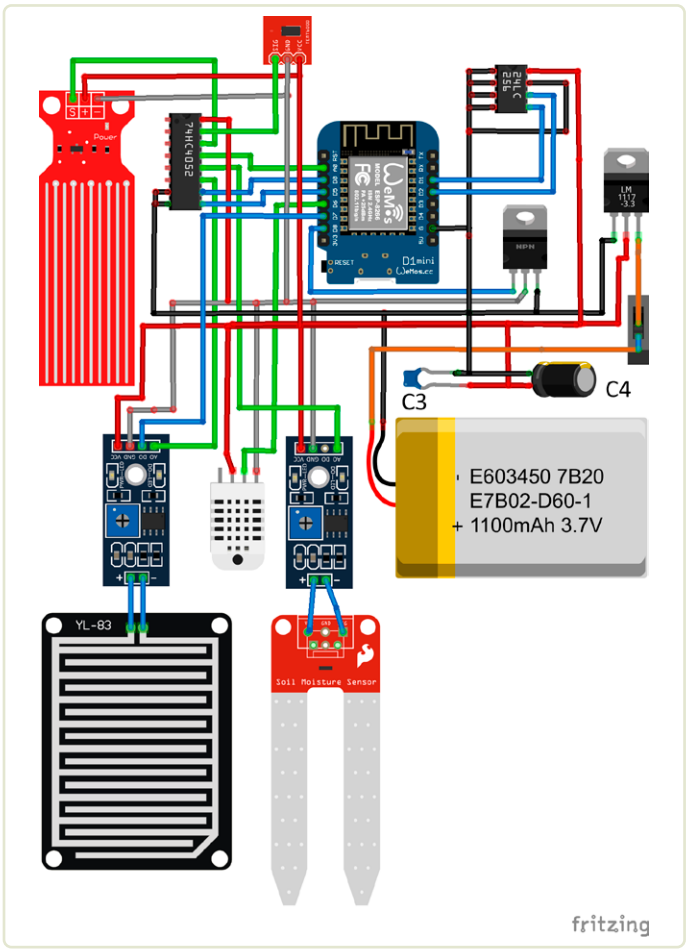


Bild 2. Verdrahtung des Gartenloggers.

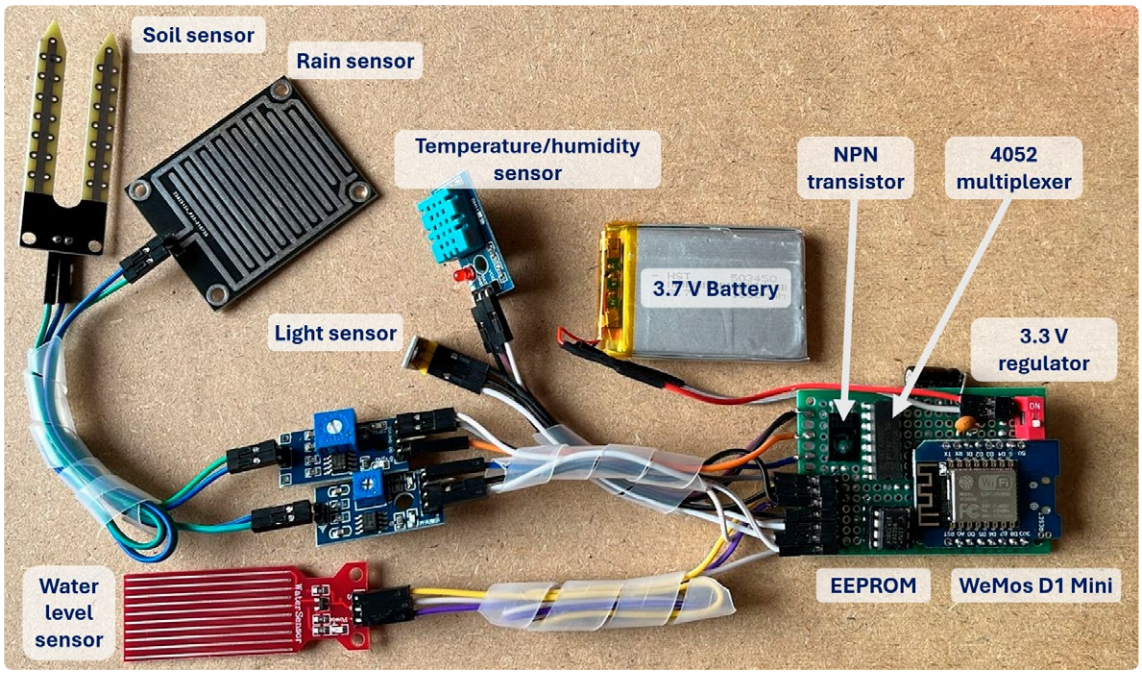


Bild 3. Alle Teile des Projekts, außerhalb des Gehäuses verdrahtet.

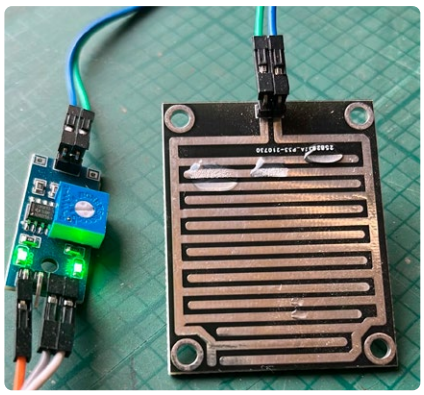


Bild 4. Regentropfen, die an meinen Sensor klopfen.

```
13:55:15.169 -> Analog output: 538   → New reading
13:55:16.187 -> Analog output: 538
13:55:17.186 -> Analog output: 540
13:55:18.190 -> Analog output: 538
13:55:19.169 -> Analog output: 538   → Readings induced by water viscosity
13:55:20.165 -> Analog output: 538
13:55:21.159 -> Analog output: 538
13:55:22.187 -> Analog output: 537
13:55:23.189 -> Analog output: 537
13:55:24.182 -> Analog output: 537
```

Bild 5. Interpretation der Messungen des Regentropfensensors.

zeigt die „physische" Verdrahtung der Komponenten vor dem Einbau in das Gehäuse.

Nach dem Einbau soll der Temperatur- und Feuchtesensor von der Unterseite des Gehäuses aus zugänglich sein, damit er sich so nah wie möglich an der Bodenumgebung befindet, aber dennoch vor Bewässerung oder Regenwasser geschützt ist. Auf der Oberseite des Loggers müssen Licht- und Regentropfensensor der Umgebung ausgesetzt sein, um Sonnenlicht und fallende Wassertropfen optimal zu überwachen. Um den Lichtsensor vor den Naturgewalten zu schützen, wäre ein kleines Glasfenster im Gehäuse sinnvoll. Wasserstands- und Bodenfeuchtigkeitssensor können mit den 20 cm langen Kabeln bequem um den Logger herum positioniert werden.

Der Logger verwendet eine 3,7-V-Lithiumbatterie als Stromquelle. Die verwendete Batterie passt genau in das Projektgehäuse und weist eine moderate Kapazität von 1.100 mAh auf. Da der WeMos D1 mit 3,3 V arbeitet, musste ich mit einem Low-Dropout-Regler (LDO) die Batteriespannung um nur 0,4 V reduzieren. Ich habe mich für einen Regler mit einem maximalen Ausgangsstrom von 800 mA entschieden, was für die vorgesehenen Logger-Komponenten ausreichend ist. Sie können sich natürlich auch für eine kleinere oder größere Batterie entscheiden, um die von Ihnen gewünschte Betriebsdauer des Loggers zu erreichen.

Abschätzen der Wassermenge

Die Menge des Wassers, die an der betreffenden Stelle im Garten ankommt, ist eine wichtige Messgröße, wenn es um die Lösung von Problemen mit Pflanzenstress geht. Ich habe den Wasserstandssensor verwendet, um solche Messungen vorzunehmen. Dabei handelt es sich um eine traditionelle Methode, bei der das fallende Wasser, entweder vom Regen oder von Sprinklern, durch einen Trichter gesammelt und einem definierten Behälter zugeführt wird. Die Höhe des Wasserspiegels L in einem Behälter, zum Beispiel einem Zylinder mit bekannter Querschnittsfläche A ergibt ein Volumen $V = L \times A$. Beim Schreiben des Arduino-Sketches habe ich die Fläche A als eine (hartkodierte) Konstante definiert und zeichne den Verlauf des gesammelten Wasservolumens über die Zeit auf, indem ich die Messwerte des Wasserstandssensors für die Größe L verwende.

Der Regentropfensensor (**Bild 4**) verfolgt dagegen einen anderen Ansatz für die Berechnung einer Wassermenge. Zwar funktioniert der Sensor nach dem gleichen Prinzip wie der Wasserstandssensor, das heißt, je mehr Wasser auf den Sensor fällt, desto geringer ist sein elektrischer Widerstand, aber er hat anders geformte Leiterbahnen, die es ihm ermöglichen, die zu einem bestimmten Zeitpunkt fallenden Wassertropfen zu erkennen. Das Problem dabei ist, dass sich aufgrund der Viskosität beziehungsweise der Oberflächenspannung des Wassers die Tropfen gerne zwischen den Leiterbahnen ansammeln.

Um dieses Problem zu lösen, habe ich den Logger so im Garten installiert, dass das Gehäuse geneigt ist und die Leiterbahnen schräg und zum Boden ausgerichtet verlaufen, damit die Wassertropfen besser „rutschen" und bereits erfasste Tropfen sich entfernen. Die Wassertropfen müssen schnell in dem Augenblick, in dem sie auf den Sensor treffen, erfasst und die überwältigend vielen folgenden „Messwerte" verworfen werden, die verursacht werden, wenn sich ein Wassertropfen über die Leiterbahnen des Sensors gen Boden bewegt und dabei weitere Kontakte verursacht. Ein weiteres Problem besteht darin, dass Wassertropfen an der unteren horizontalen Leiterbahn hängenbleiben und sich dort auf unbestimmte Zeit festsetzen.

Hier kommt der digitale Ausgang der Sensortreiberschaltung zur Hilfe. Die Einstellung des Trimmpotis hilft, die Auswirkungen dieser Situation zu neutralisieren. Der digitale Ausgang zeigt dann keinen Regen an (logisch hoch). Bleibt noch die Auswirkung der Bewegung der Wassertropfen, wenn sie an den Leiterbahnen heruntergleiten. Beim Experimentieren mit dem Regensensor konnte ich leichte Fluktuationen am Analogausgang des Sensors feststellen. Dieses Phänomen habe ich genutzt, indem ich die einem echten Messwert folgenden ähnlichen Messwerte verworfen habe. Dies wird in **Bild 5** veranschaulicht: Oben ist die analoge Ausgabe eines einzelnen Wassertropfens mit „538" anzeigt; die nachfolgenden Messwerte, die entstehen, wenn der Tropfen der Schwerkraft folgt und nach unten gleitet, sind fast gleich (und können eliminiert werden).

Es gibt zwei Arten von Beregnern, nämlich die, die einen festen Radius um sich herum bewässern, und rotierende Regner, die einen Teil der Strömungsenergie des Wassers nutzen, um zu rotieren. Um beide Arten von Rasensprengern zu berücksichtigen, muss die Firmware des Mikrocontrollers in der Lage sein, zwischen einem kontinuierlichen, gleichmäßigen Wasserfluss und intermittierenden Wasserstößen zu unterscheiden.

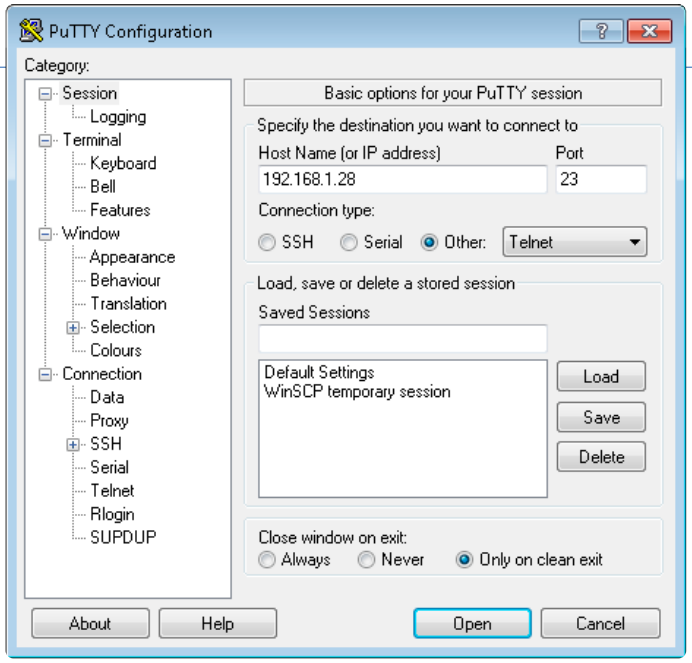


Bild 6. PuTTY-Konfiguration (1/2).

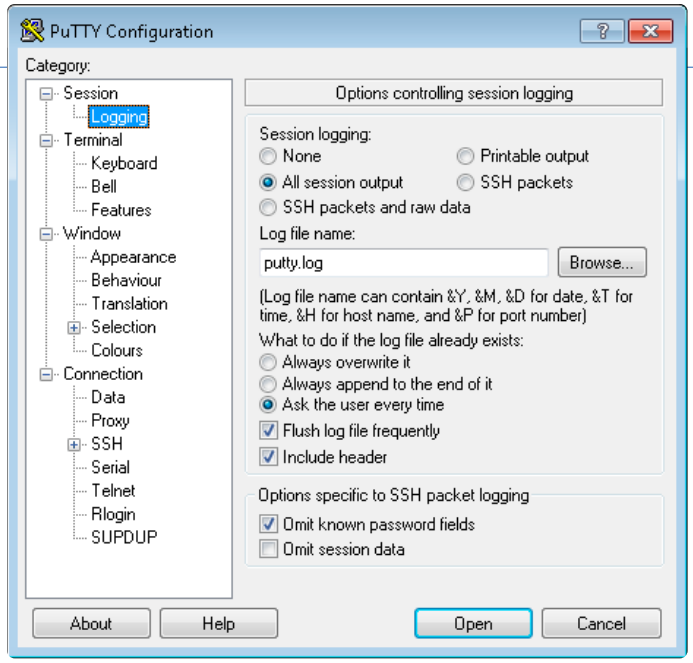


Bild 7. PuTTY-Konfiguration (2/2).

Umgang mit aufgezeichneten Daten

Der Gartenlogger verfügt über ein externes 32-KB-EEPROM, das die erfassten Daten während der gesamten Dauer der Aufzeichnung speichert. Ich habe den Chip in eine DIP-Fassung gesetzt anstatt ihn direkt auf die Prototyp-Platine zu löten, damit ich ihn bei Bedarf problemlos durch ein Exemplar mit höherer Speicherkapazität ersetzen kann. Auch der ESP8266 besitzt ein internes EEPROM, das für den gleichen Zweck verwendet werden könnte, allerdings mit einer begrenzten Speichergröße von 4 KB. Auch hat das häufige Beschreiben des internen EEPROMs den negativen Effekt, dass sich die Lebensdauer des ESP-Controllers verringert, so dass ich empfehle, den Speicher nicht für die Datenaufzeichnung zu verwenden.

Jeder Datensatz ist als durch Komma getrennte Werte formatiert. Ich ging davon aus, dass alle Messungen im Integer-Format vorliegen können, da diese Anwendung nicht die Präzision von Brüchen erfordert. Der Code wandelt daher alle `Float`-Messwerte in den Datentyp `Integer` um und verwendet dann die Funktion `String()`, um sie in Strings (`strVal`) zu konvertieren. Durch die Kaskadierung mit einer `strcat(strVal, ",")`-Funktion wird zwischen den Datenelementen des Datensatzes ein Komma eingebettet. Außerdem wird jeder dieser so entstehenden Zeichenkette ein Zeitstempel vorangestellt.

Das Projekt enthält kein RTC-Modul, um genaue Zeitstempel zu setzen. Stattdessen kontaktiert es es jeden Tag um Mitternacht einen NTP-Server im Internet, um die Zeitangaben zu erhalten, und verwendet dann die Funktion `millis()`, um die aktuelle Messzeit zu berechnen und daraus einen Zeitstempel zu erzeugen. Die Struktur eines Messdatensatzes ist: *Zeitstempel, Temperatur, Luftfeuchtigkeit, Licht, aufgefangenes Volumen, abgegebenes Volumen, Bodenfeuchtigkeit*.

Für die Übermittlung der aufgezeichneten Werte an einen Computer gibt es mehrere Möglichkeiten. Eine wäre, auf dem Datenlogger einen Webserver laufen zu lassen; alternativ kann man auch den Mikrocontroller so programmieren, dass die Aufzeichnungen, sobald sie in das externe EEPROM geschrieben wurden, im Batch-Modus an einen PC gesendet werden. Ein Telnet-Client wie PuTTY kann verwendet werden, um die Aufzeichnungen in einer CSV-Datei zu speichern. Eine dritte Möglichkeit besteht darin, die Datensätze an den PC zu senden,

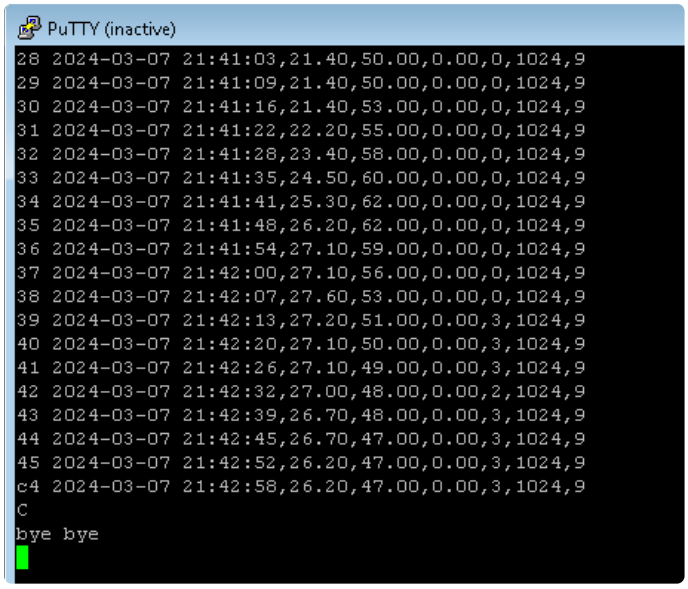


Bild 8. Datenprotokoll im Telnet-Terminalfenster auf dem PC.

sobald eine Zeile der CSV-Daten erstellt wurde. In der aktuellen Version dieses Projekts habe ich mich für die letzte Methode entschieden. **Bild 6** zeigt das Konfigurationsfenster von PuTTY auf dem PC.

Der Umgang mit PuTTY ist recht einfach: Sorgen Sie dafür, dass sich der Gartenlogger und der PC im selben Drahtlos-Netzwerk befinden, geben Sie die IP-Adresse des Loggers in das Feld *Host Name* ein, setzen Sie den Port 23 und wählen Sie aus dem Dropdown-Feld das Telnet-Protokoll aus. Bevor Sie auf *Open* klicken, um die Telnet-Shell zu starten, müssen Sie die Protokollierung der vom Logger übermittelten Daten einrichten. Wählen Sie links unter *Category* den Eintrag *Logging*, so dass das Fenster in **Bild 7** erscheint. Sie müssen oben die Option *All session output* und unten *Ask the user every time* auswählen, damit eine bestehende Protokolldatei aus früheren Sitzungen nicht überschrieben wird. Anschließend muss der gewünschte Speicherort der CSV-Datei über die Schaltfläche *Browse...* ausgewählt werden.

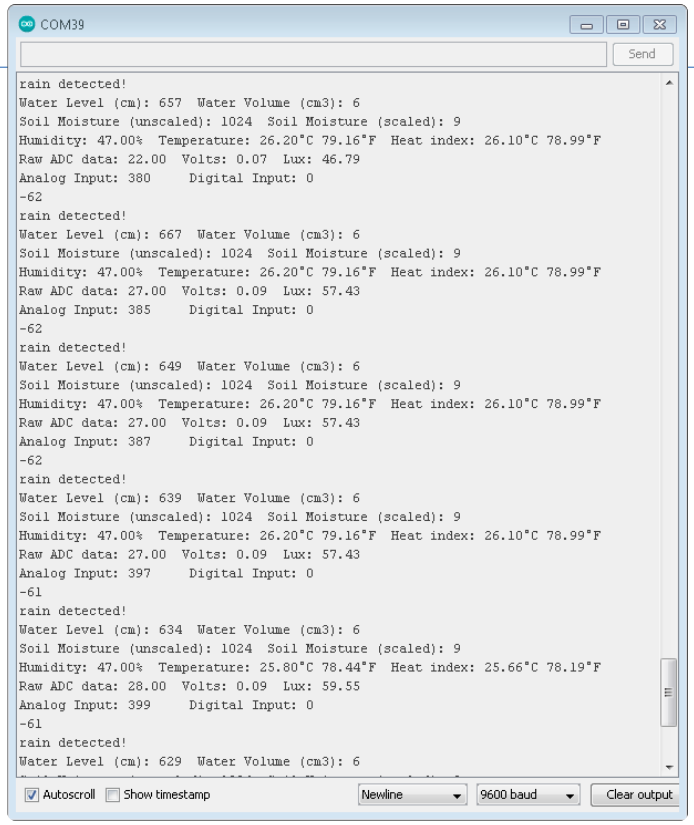


Bild 9. Datenausgabe der einzelnen Sensoren im Seriellen Monitor der Arduino-IDE.

Nun können Sie auf *Open* klicken, um die Telnet-Shell zu starten. Nach dem Start beginnt die Shell mit dem Sammeln der vom Logger stammenden Protokolle, wie in **Bild 8** dargestellt. Es ist möglich, die Protokollierungssitzung durch Eingabe von „C" zu beenden; der Logger antwortet mit der von der Funktion `telnetAction()` generierten Nachricht „bye bye". Die Sensorwerte sind auch an der seriellen Schnittstelle verfügbar, da sie in den Log-Einträgen zusammengestellt sind. Sie können über die serielle Schnittstelle ausgelesen werden, zum Beispiel mit dem Seriellen Monitor der Arduino-IDE (**Bild 9**).

Der Code

Das Projekt wird mit der Arduino-IDE programmiert. Das Programm besteht aus der Hauptdatei *Garden.ino* und fünf Hilfsdateien. Jede der Hilfsdateien fasst die Funktionen zusammen, die ein Peripheriegerät steuern: *Timing.ino* enthält die Funktionen, die mit dem NTP-Server zusammenhängen, *telnet-streaming.ino* befasst sich mit der Formatierung der Aufzeichnungen und dem Senden an den PC, *analog-sensors.ino* enthält vier Funktionen für die analogen Sensoren für Licht, Regen, Wassermenge und Bodenfeuchtigkeit. Die Datei *digital-sensors.ino* schließlich enthält eine einzige Funktion, die die Feuchte und die Temperatur der Umgebungsluft ausliest, während die Datei

Listing 1. Die Hauptschleife

```
void loop() {
  delay(2000);
  digitalWrite(SENSORS, HIGH);  // enable sensors power
  delay(1000);
  senseAmbient(); // read ambient humidity & temperature
  senseLight();   // read ambient light intensity
  senseRain();    // read rain status
  senseWater();   // read irrigation water level
  senseSoil();    // read soil moisture level
  telnetAction(); // assemble logged data and send to telnet app
  digitalWrite(SENSORS, LOW); // disable sensors power
 }

// this is part of setup()
  pinMode(MUXA, OUTPUT);          // part of 2-bit analog sensor selection address
  pinMode(MUXB, OUTPUT);          // part of 2-bit analog sensor selection address
  pinMode(SENSORS, OUTPUT);       // sensors GND control pin
  pinMode(DHTPIN, INPUT_PULLUP);  // ambient DHT sensor feed
  pinMode(RAIN, INPUT);           // rain status sensor digital feed
  digitalWrite(SENSORS, LOW);     // disable sensors power

 // see [5] for the complete code
```

Stückliste

Mikrocontroller-Modul:
WeMos D1 Mini (ESP8266)

Sensoren:
Wasserstandsensor (z.B. https://t1p.de/jtm5t)
Regensensor LM393 (z.B. https://t1p.de/zde01)
Bodenfeuchtesensor YL-69/HC-38 (z.B. https://t1p.de/3qbd1)
Temperatur- und Feuchtesensor DHT11
Lichtsensor TEMT6000

Außerdem:
C3 = 100 n, keramisch
C4 = 470 µ, 10 V
EEPROM 24LC256
Spannungsregler LM1117MP-3.3-NOPB 3.3V
Transistor BD139
Analog-Multiplexer CD4052BE
Lithium-Batterie 3,7 V (z.B. 1.100 mAh)
ABS-Gehäuse 85×50×21 mm
Doppelseitige Lochrasterplatine 40×60 mm

external-eeprom.ino die Funktionen zum Lesen und Schreiben des EEPROM-Chips enthält.
Diese strukturierte Kodierung erleichtert die Nachverfolgung und Fehlersuche in jedem funktionalen Teil des Projekts und erleichtert die Wiederverwendbarkeit. **Listing 1** zeigt den `loop()`-Abschnitt des Sketches, der nur zehn Befehle umfasst, von denen sechs die zuvor besprochenen Funktionsaufrufe sind. Das Listing zeigt auch den Teil des `setup()`-Abschnitts, mit dem ich das Multiplexing der vier analogen Sensorausgänge auf den analogen Eingangspin des D1 Mini steuere. Außerdem wird in diesem Abschnitt ein GPIO zur Steuerung des Transistors zugewiesen, der zum stromsparenden Ein- und Ausschalten des Sensors verwendet wird. Den vollständigen Code finden Sie unter [6].
Es sollte erwähnt werden, dass der ESP8266 über einen Analog-Digital-Wandler mit einer Auflösung von 10 Bit verfügt, wodurch er analoge Eingangssignale in einen Wert im Bereich von 0...1.023 umzuwandeln kann. Viele bestehende Projekte wandeln diesen Wert mit Mapping-Funktionen wie `map(analogVal, 0, 1023, 0, 255)` in eine 8-Bit-Darstellung um und verkleinern so den Wertebereich auf 0...255. Ich habe auf ein solches Mapping verzichtet, um die Sensormesswerte so genau wie möglich zu erhalten.

Schlussbemerkungen

Intelligente Bewässerung auf der Grundlage Maschinellen Lernens erfordert eine große Menge an Sensordaten, um ein genaues Bewässerungsmodell zu erstellen, das die beste Bewässerungseinstellung unter verschiedenen Umgebungsbedingungen vorhersagt. Dieses Projekt erleichtert die Aufzeichnung der Umweltbedingungen, denen ein Garten ausgesetzt ist. Eine Erweiterung des Projekts könnte die Aufzeichnung des Zustands von Pflanzen von Interesse mit einer Kamera oder einer anderen Art von Sensor in Betracht ziehen und diese Informationen zu den Protokollen dieses Projekts hinzufügen. Dies würde es uns ermöglichen, ein effizientes Bewässerungsmodell mit überwachten maschinellen Lerntechniken zu erstellen.

RG — 230629-02

Über den Autor

Gamal Labib begeistert sich seit zwei Jahrzehnten für eingebettete Systeme und ist derzeit als Mentor tätig (bei *codementor.io*). Er hat einen MEng und einen Doktortitel in IT. Er schreibt nicht nur für Fachzeitschriften, sondern ist auch Gastprofessor an ägyptischen Universitäten und zertifizierter IT-Berater.

haben Sie Fragen oder Kommentare?

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

WEBLINKS

[1] Beispiel für ein Bewässerungsprojekt: https://circuitdigest.com/microcontroller-projects/automatic-irrigation-system-using-arduino-uno
[2] KI-gestützte Pflanzenüberwachung: https://smellslike.ml/posts/tf-microcontroller-challenge-droopthereitis/
[3] Arduino-basiertes Projekt zur intelligenten Bewässerung: https://tinyurl.com/2kkbc2zx
[4] pH-Wert im Boden in Europa (EU): https://t1p.de/7bsf7
[5] Deutschlandkarte zum Boden-pH: https://t1p.de/v4klf
[6] Downloads zu diesem Projekt: https://elektormagazine.de/230629-02

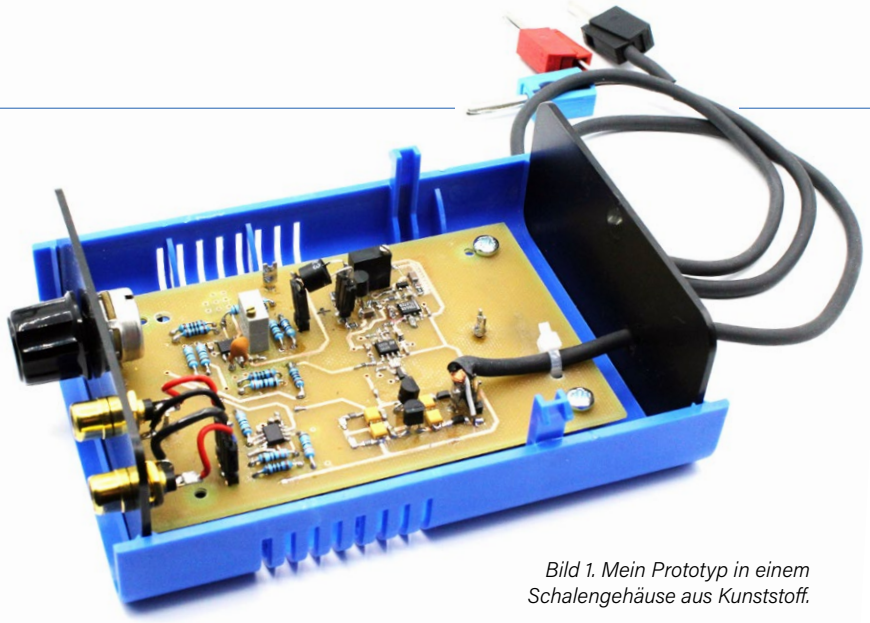


Bild 1. Mein Prototyp in einem Schalengehäuse aus Kunststoff.

Analoger 1-kHz-Generator

Sinuswellen mit geringen Verzerrungen

Von Alfred Rosenkränzer (Deutschland)

Für Messungen an Audio-Elektronik ist ein hochqualitatives Sinussignal mit 1 kHz unerlässlich. An analoge Sinusgeneratoren kommt digitales Equipment immer noch nicht ganz heran. Fertiggeräte sind wie immer im High-End-Bereich nicht ganz preiswert. Mit der hier vorgeschlagenen Schaltung bietet der Selbstbau eines solchen Generators deswegen deutliche Kostenvorteile, ohne dafür allzu große Kompromisse bei der Signalqualität eingehen zu müssen.

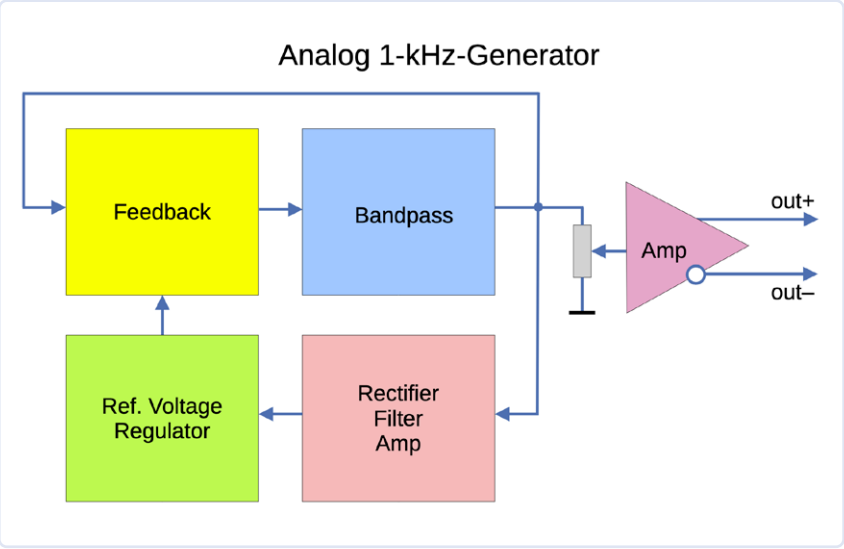


Bild 2. Die Blockschaltung des 1-kHz-Generators.

Mein Review des QA403 von QuantAsylum [1] zeigt, dass digitale Sinusgeneratoren mittlerweile ganz brauchbar sind. Diese Firma hat leider ihr analoges Modell QA480 - eine Ergänzung zum QA403 - aufs Altenteil geschickt (siehe Blog unter [2]). Dieses Gerät kombinierte einen analogen 1-kHz-Generator mit einem passenden Notch-Filter und einem hochwertigen 12-dB-Verstärker. Amplitude und Notch-Filter konnten dabei via USB vom einem PC gesteuert werden. Aber das ist Geschichte. Die Auslistung erfolgt laut Blog unter anderem wegen eines zu hohen Aufwands für den Abgleich.

Die Schaltung basierte auf dem Entwurf einer anderen Firma [3] und wurde glücklicherweise in diesem Blog veröffentlicht. Da es im Moment kein äquivalentes Gerät zu kaufen gibt, habe ich den Schaltplan leicht überarbeitet und eine passende Platine entwickelt. Der Prototyp ist in **Bild 1** zu sehen. Die Stromversorgung erfolgt über externes, symmetrisches ±15-V-Netzteil. Der Einfachheit halber habe ich die Amplitudeneinstellung mit einem Poti realisiert. Es gibt also weder USB noch Software und daher auch keine Störsignale durch digitale Elektronik.

Schaltung

Im Blockschaltbild von **Bild 2** sieht man fünf Funktionseinheiten. Der eigentliche Oszillator besteht aus einem Bandpassfilter, welches durch Mitkopplung zum Schwingen gebracht wird. Die Amplitude wird durch eine Regelung stabilisiert, deren Istwert per Gleichrichtung und Filterung aus dem Ausgangssignal gewonnen wird. Für ein stabiles Signal braucht es noch eine Referenzspannung. Der Ausgangsverstärker bietet ein Differenzsignal mit einstellbarer Amplitude.

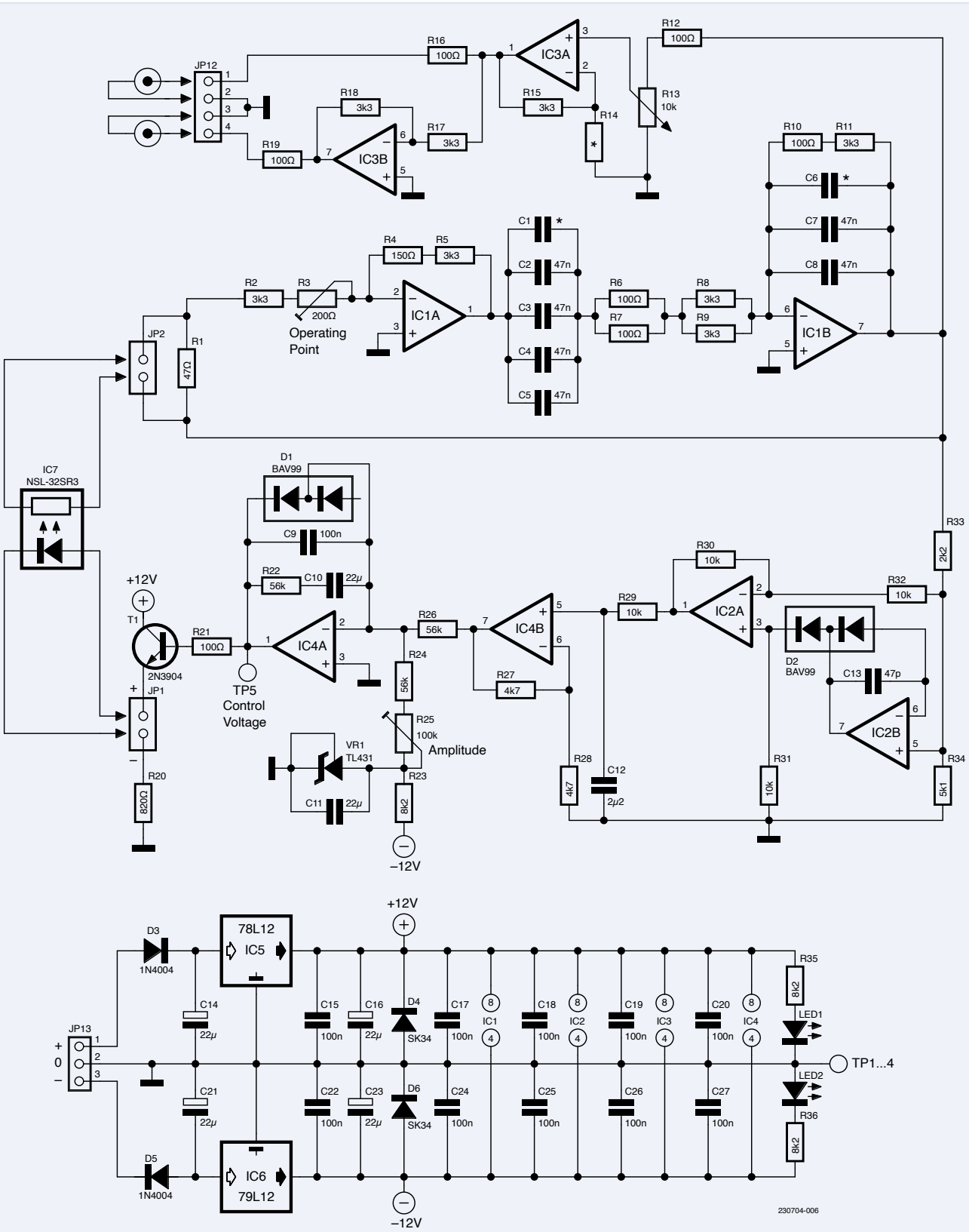


Bild 3. Die Schaltung kommt mit vier Dual-Opamps, zwei Spannungsreglern und einem Optokoppler aus.

Die Funktionsblöcke werden in der Schaltung in **Bild 3** in konkrete Elektronik umgesetzt.

Bandpass

Der Bandpass wird durch die Beschaltung von IC1B mit R6...R11 und C1...C8 realisiert. Seine Mittenfrequenz bestimmt die Oszillationsfrequenz. Die Qualität der hier verwendeten Bauteile beeinflusst die Verzerrungen des erzeugten Sinussignals. Als Kondensatoren können entweder keramische COG-Kondensatoren im SMD-Gehäuse oder bedrahtete Folienkondensatoren bestückt werden. In der Schaltung ist daher die doppelte Anzahl der Kondensatoren zu sehen, von denen nur die Hälfte, also SMDs ***oder*** bedrahtete Folienkondensatoren, bestückt wird. Auf der Platine kann man gut sehen, dass am Eingang von IC1B faktisch nur zwei Kondensatoren parallel geschaltet und bestückt sind (plus C1, falls man die

Frequenz anpassen will). Im Rückkopplungszweig ist es sogar nur ein Kondensator (plus eventuell C6 zur Frequenzanpassung). Die Pads der Widerstände haben die leicht von Hand lötbare Bauform SMD1206. Auf diese Pads passen auch MELF204-Widerstände, die ähnlich gut sind wie bedrahtete Metallschicht-Versionen. Statt MELF- kann man auch Dünnschicht-Widerstände verwenden. Die Kapazität im Eingangszweig ist doppelt so groß wie im Rückkopplungsweg. Dafür sind am Eingang die Widerstände nur halb so groß wie bei der Rückkopplung. Diese Maßnahme soll zumindest rechnerisch niedrigeres Rauschen ermöglichen. Es empfiehlt sich, die Kondensatoren auszumessen und möglichst gleiche Exemplare parallel zu schalten. Dadurch wird die spätere Feinabstimmung der Frequenz durch C1 und C6 beziehungsweise R6/R7 und R10 erleichtert. Der Bandpass hat bei der Mittenfrequenz eine Verstärkung von etwa -1.

Mitkopplung

Die (steuerbare) Mitkopplung der Schaltung um IC1A (R1...R5) muss für stabile Schwingungen ebenfalls eine Verstärkung von etwa -1 haben. Bei zu geringer Mitkopplung schwingt die Schaltung nicht und bei zu großer werden die Signale so groß, dass sie durch den möglichen Spannungshub der Opamps begrenzt werden. Eine konstante Amplitude mit unbegrenzter, sinusförmiger Signalform ist nur durch eine Regelung der Mitkopplung zu erreichen.

Als Regelelement dient der gut erhältliche „Optowiderstand" NSL-32SR3 von Advanced Photonix [4]. Der Strom durch die LED auf der Eingangsseite beeinflusst den Widerstand des integrierten LDRs auf der Ausgangsseite. Dieser Optokoppler bietet die übliche galvanische Trennung von Ein- und Ausgang.

Amplitudenmessung

Über den Spannungsteiler R33/R34 gelangen circa 70% des Ausgangssignals von IC1B an den Eingang von IC2B. Diese Abschwächung verhindert ein merkwürdiges Einschwingverhalten nach dem Einschalten, wenn der Oszillator kurzfristig bis in die Amplitudenbegrenzung getrieben wird.

Die Schaltung des mit IC2 aufgebauten Präzisionsgleichrichters ist sehr gut in einem *Reference Design Paper* von TI [5] beschrieben. Das gleichgerichtete Signal wird vom Tiefpass aus R29 und C12 integriert beziehungsweise „gemittelt" und dann von IC4B zweifach verstärkt. Falls man die entsprechenden Größen berechnen will, ist ein passender Online-Kalkulator [6] hilfreich. IC4A arbeitet dank seiner Beschaltung als Regler.

Amplitudenstabilisierung

Als Referenzspannungsquelle VR1 fungiert das handelsübliche IC TL431. An ihm fällt eine stabilisierte Spannung von -2,5 V gegen Masse ab. C11 reduziert dabei das Rauschen. Mit R24 und R25 wird der Strom von IC4B und R26 kompensiert und mit R25 die Amplitude eingestellt. D1 verhindert, dass die Regelspannung am Ausgang von IC4A negativ wird. Über R21 wird T1 angesteuert. Sein Emitter treibt über R20 die LED des Optokopplers, dessen Ausgangswiderstand die Mitkopplung verändert. Damit ist der Regelkreis geschlossen. R26, R22, C9 und C10 bilden dessen Zeitkonstante.

Ausgangsstufe

Über R12 gelangt das Sinussignal an Poti R13, mit dem die Ausgangsamplitude zwischen 0 und dem durch R25 festgelegten Höchstwert eingestellt werden kann. IC3A dient als Ausgangsverstärker. Mit R14 kann man die Verstärkung bei Bedarf etwas höher einstellen. IC3B verstärkt invertierend mit dem Faktor 1 und bildet so das negative Ausgangssignal. Als Signalbuchsen können je nach Wunsch Cinch- (RCA), BNC-, Klinken- oder XLR-Ausführungen verwendet werden.

Stromversorgung

Die Schaltung wird mit ±15 V versorgt. Zwei Spannungsregler erzeugen daraus stabilisierte ±12 V. D3/D5 verhindern Schäden durch Verpolung und D4/D6 schützen IC5 und IC6 vor einem sogenannten Latchup beim Einschalten. Die beiden LEDs leuchten bei vorhandener Stromversorgung.

Auswahl der Bauteile

Die Qualität der Opamps von IC1 und IC3 haben großen Einfluss auf die Signalqualität. Aus diesem Grund habe ich hier die etwas teureren Typen OPA2211 eingesetzt. Die Schaltung funktioniert natürlich auch mit anderen, pinkompatiblen Dual-Opamps. Für IC2 und IC4 kann man fast jeden passenden Dual-Opamp nehmen. In meinem Prototyp stecken TL072.

Die Kondensatoren C1 bis C8 beeinflussen

Stückliste

Widerstände:

(Wenn nicht anders angegeben:
MELF 0204 oder Dünnfilm SMD 1206, 1%)
R1 = 47 Ω
R2, R5, R8, R9, R11, R15, R17, R18 = 3k3
R3 = 200 Ω, Mehrgang-Trimmpoti, stehend, RM 1/10"
R4 = 150 Ω
R6, R7, R10, R12, R16, R19 = 100 Ω*
R12, R16, R19 = 100 Ω
R13 = 10 k, Poti, linear
R14 = optional*
R20 = 820 Ω, SMD 0603
R21 = 100 Ω, SMD 0603
R22, R24, R26 = 56k, SMD 0603
R23, R35, R36 = 8k2, SMD 0603
R25 = 100 k, Mehrgang-Trimmpoti, stehend, RM 1/10"
R27, R28 = 4k7, SMD 0603
R29, R30, R31, R32 = 10 k, SMD 0603
R33 = 2k2, SMD 0603
R34 = 5k1, SMD 0603

Kondensatoren:

C1, C6 = optional*, COG, SMD 0603
C2, C3, C8= 47 n, Folie, RM 2/10"
(Alternative zu C4, C5, C7)*
C4, C5, C7 = 47 n, COG, SMD0805
(Alternative zu C2, C3, C8)*
C9, C15, C17..C20, C22, C24...C27 = 100 n, X7R, SMD 0603
C10, C11 = 22 µ / 16 V, SMD 1206
C12 = 2µ2 / 16V, SMD 1206
C13 = 47 p, COG, SMD 0603
C14, C16, C21, C23 = 22 µ / 25 V, Tantal-Elko, SMC-B

Halbleiter:

LED1, LED2 = LED, SMD 0805
D1, D2 = BAV199, SOT23
D3, D5 = 1N4004
D4, D6 = SK34, DO214AC
T1 = 2N3904, SOT23
IC1, IC3 = OPA2211*, SOIC8
IC2, IC4 = TL072, SOIC8
IC5 = 78L12, TO92
IC6 = 79L12, TO92
IC7 = NSL-32SR3*
VR1 = TL431, TO92

Außerdem:

JP1, JP2 = 1×2-pol. Stiftleiste, RM 1/10"
JP12 = 1×4-pol. Stiftleiste, RM 1/10"
JP13 = 1×3-pol. Stiftleiste, RM 1/10"
Platine 230704-01
Knopf für Poti R13
Audiobuchsen*

* siehe Text

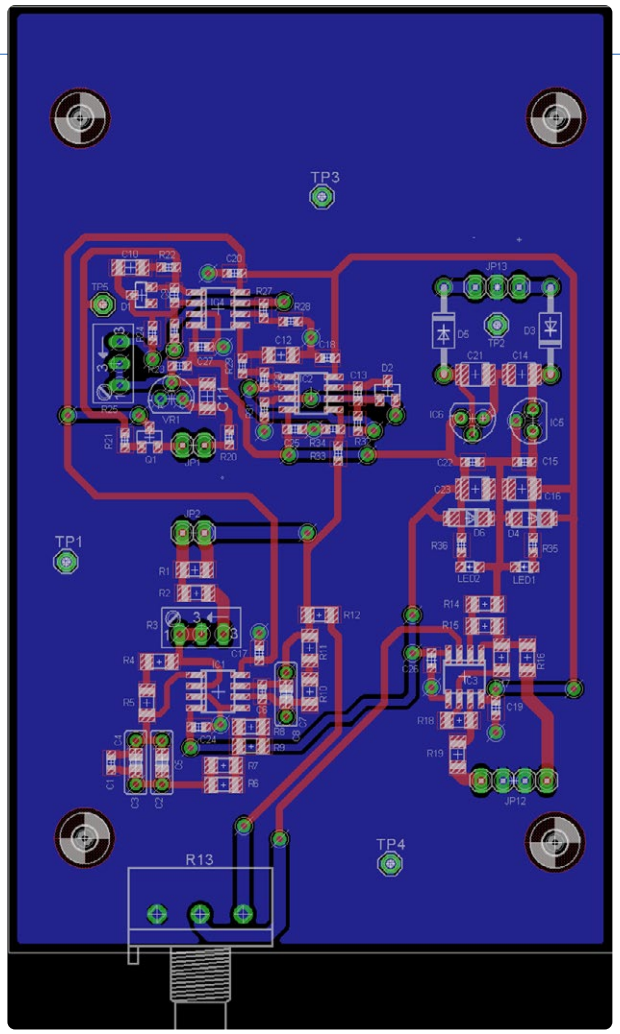


Bild 4. Layout der Platine. Die zugehörigen Dateien können im Eagle-Format unter [7] heruntergeladen werden.

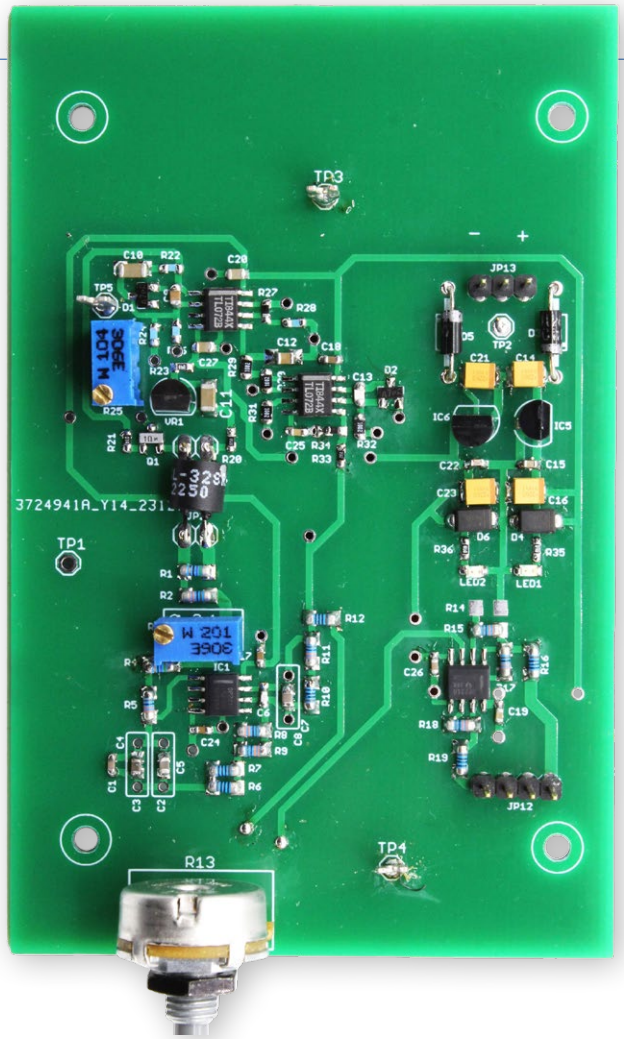


Bild 5. Die fertig bestückte Platine meines Prototypen.

ebenfalls die Verzerrungen. Wie schon erwähnt sollte man hier COG-Typen im SMD-Gehäuse oder bedrahtete Folienkondensatoren verwenden. Merke: Ein einziger, nach audiophilen Kriterien „schlechter" Kondensator mit XR7-Dielektrikum oder Ähnlichem verdirbt alles! Die Widerstände im Signalweg sollten als Dünnfilmwiderstände der Bauform 1206 oder MiniMELF204 ausgeführt werden.

Bestückung und Inbetriebnahme

Den Optokoppler IC7 sollte man zunächst nicht bestücken. In Bild 1 ist zu sehen, dass ich ihn mit zwei zweipoligen Stift-/Buchsenleisten steckbar ausgeführt habe. Das erleichtert den Test der Schaltung: Schließt man nämlich die Pins für den integrierten LDR kurz und dreht an Poti R3, sollte der Oszillator bei einer bestimmten Einstellung schwingen und ein auf 20 V_{SS} begrenztes Signal generieren.

Damit kann man die Funktion von Ausgangsstufe, Gleichrichter und Filter inklusive Verstärker überprüfen. Die Frequenz sollte in der Nähe von 1 kHz liegen. Der Feinabgleich erfolgt später bei geregelter, stabiler Amplitude. Die Frequenz verändert sich leicht, wenn die Opamps in die Sättigung geraten. Bei begrenztem Signal sollte der Ausgang von IC4A 0 V liefern. Stoppt man den Oszillator, so geht er langsam hoch bis circa +10 V. Die Transfereigenschaften des Optokopplers können sehr schwanken. Ich habe vier Exemplare gekauft, und jeder verhielt sich anders. Mit einem Labornetzteil und einem Vorwiderstand lässt sich der Strom durch die LED einstellen und dann mit einem Multimeter der Widerstand messen. Als guter Arbeitspunkt hat sich ein LED-Strom von 3…5 mA bewährt.

Für die weitere Inbetriebnahme sollte man einen Widerstand mit dem gemessenen Wert parallel zu R1 schalten und dann R3 so einstellen, dass der Oszillator gerade schwingt. Notfalls kann man R4 anpassen. Entfernt man den Widerstand und setzt den Optokoppler ein, sollte die Schaltung ein Sinussignal mit konstanter und stabiler Amplitude liefern. Mit R25 kann man die gewünschte Amplitude auf Werte von 1…3 V_{SS} einstellen. Größere Amplituden korrespondieren mit höheren Verzerrungen. Die Regelzeitkonstante wurde bewusst recht groß gewählt. Dadurch dauert zwar das Einschwingen lange, aber danach ist der Pegel sehr konstant. Mit R3 wird der Arbeitspunkt des Reglers eingestellt.

Frequenz und mehr

Nun kann die Feineinstellung der Frequenz erfolgen. Man kann die Kapazitäten durch Bestückung mit C1 und C6 (C1 = 2 × C6) ändern oder aber die Werte von R6/R7 und R10 anpassen. Die Frequenz berechnet sich nach der üblichen Formel $f = 1/(2 \times \pi \times R \times C)$. Nach dem Frequenzabgleich ist eventuell nochmals eine Korrektur des Arbeitspunktes nötig. Besitzt man ein Notch-Filter, so sollten die Frequenzen von Generator und Notch gleich sein.

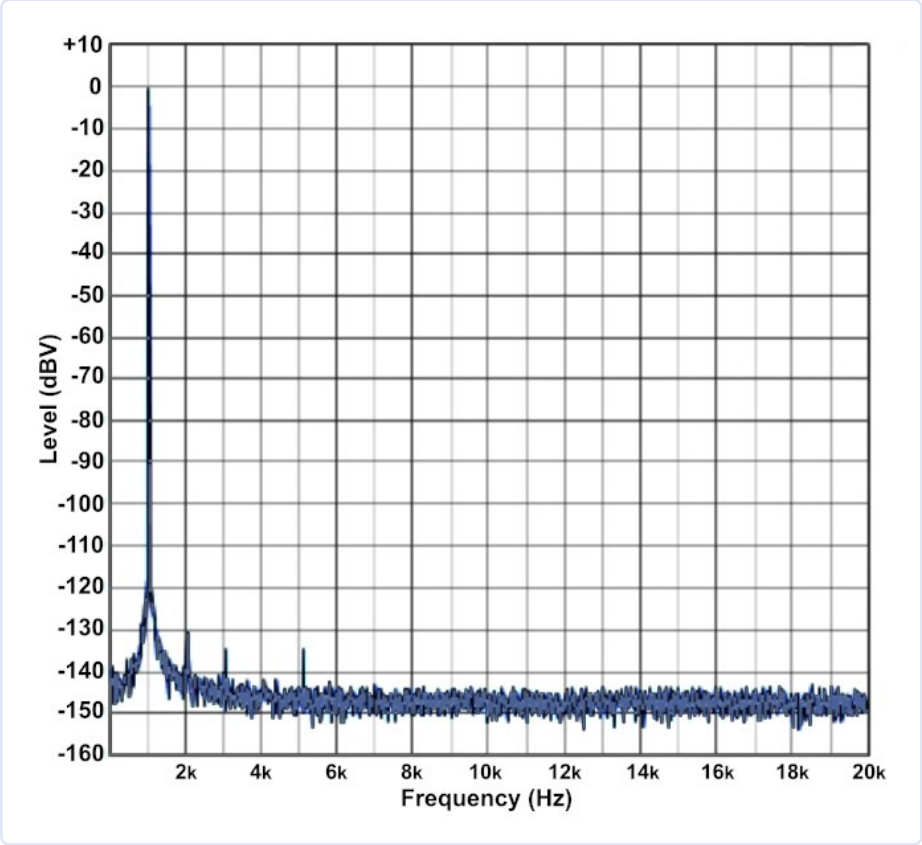


Bild 6. Das Spektrum des Ausgangssignals bei einem Pegel von 0 dBV.

Für die Schaltung habe ich eine passende Platine entwickelt. **Bild 4** zeigt deren Layout und **Bild 5** die bestückte Platine. **Bild 6** zeigt das Spektrum beim Nominalpegel von 0 dBV. Alle Oberwellen haben einen Pegel von weniger als -130 dBV: Das Ausgangssignal hat also extrem niedrige Verzerrungen. Dieses Spektrum wurde mit dem Audio Analyzer APx555 von Audio Precision erfasst.

Die Qualität des Selbstbau-Signalgenerators kann sich also sehen lassen!

Die in der Schaltung angegebenen beschriebene Bauteilewerte funktionieren bei meinem Prototypen recht gut, lassen aber noch Spielraum für Optimierungen. Ich habe auch noch ein paar Leerplatinen übrig. Bei Interesse einfach direkt bei mir melden!

230704-02

Über den Autor

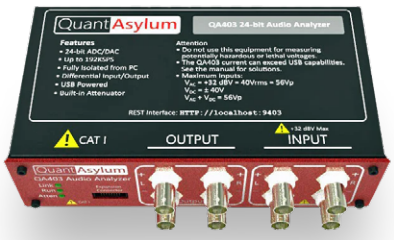
Alfred Rosenkränzer arbeitete viele Jahre als Entwicklungsingenieur, zu Anfang im Bereich der professionellen Fernsehtechnik. Seit Ende der 1990er Jahre entwickelt er digitale High-Speed-und Analogschaltungen für IC-Tester. Das Thema Audio ist sein privates Steckenpferd.

Sie haben Fragen oder Kommentare?

Haben Sie technische Fragen oder Anmerkungen zu diesem Artikel? Schicken Sie bitte eine E-Mail an den Autor unter alfred_rosenkraenzer@gmx.de oder kontaktieren Sie Elektor unter redaktion@elektor.de.

Passende Produkte

- **QuantAsylum QA403 24-bit Audio Analyzer**
  www.elektor.de/20530
- **FNIRSI 2C23T (3-in-1) 2-Kanal Oszilloskop (10 MHz) + Multimeter + Signalgenerator**
  www.elektor.de/20717



WEBLINKS

[1] A. Rosenkränzer, „Comparing the QuantAsylum QA403 to the Gold Standard“, 2023: https://tinyurl.com/yudmy5sk
[2] Blog QA480 (inaktiv): https://tinyurl.com/yckdwe8e
[3] Webseite JanasCard: http://www.janascard.cz/aHome.html
[4] Optoresistor NSL-32SR3 (Mouser): https://tinyurl.com/4c5c8nss
[5] Präzisionsgleichrichter (TI): http://www.ti.com/lit/ug/tidu030/tidu030.pdf
[6] Online-Rechner Sinuskenngrößen: https://tinyurl.com/26pk8bye
[7] Artikel-Webseite: https://www.elektormagazine.de/230704-02

Miletus:
Web-Apps offline nutzen

System- und Gerätezugriff inklusive

Von Dr. Veikko Krypczyk (Deutschland)

Web-Anwendungen haben sich heute in vielen Bereichen als Standard durchgesetzt. Sie werden im Browser ausgeführt und sind daher auf nahezu allen Geräten lauffähig, auch auf einem Raspberry Pi. Mit dem Framework Miletus kann man diese Web-Apps zu nativen Applikationen verpacken, so dass sie sich offline ausführen lassen. Darüber hinaus hat man Zugriff auf die Systemschnittstellen, um beispielsweise Signale über GPIO-Pins einzulesen und auszugeben.

Anwendungen für die Steuerung von Hardwarekomponenten, welche an den PC oder den Raspberry Pi angeschlossen sind, müssen an das jeweilige Betriebssystem angepasst werden und benötigen einen Zugriff auf die Schnittstellen des Systems. Dabei handelt es sich um so genannte native Applikationen, welche explizit für das Zielsystem erstellt und auf diesem ausgeführt werden. Demgegenüber stehen Web-Applikationen. Eine Web-Applikation kann auf nahezu allen Systemen gestartet werden, denn diese läuft im Browser. Sie hat jedoch gegenüber einer nativen App auch einige Einschränkungen. Eine Offline-Nutzung und ein Zugriff auf die Systemschnittstellen sind nicht ohne weiteres möglich. **Tabelle 1** enthält eine Gegenüberstellung der wesentlichen Merkmale von Desktop- und Web-Applikationen. Bei Anwendungen, welche den Elektroniker besonders interessieren, dürften die Argumente Offline-Nutzung, Zugriff auf die

Tabelle 1. Wichtige Eigenschaften von Desktop- und Web-Applikationen.

Merkmal	Desktop-Applikation	Web-Applikation
User Interface	nativ	auf der Basis von HTML, CSS und JavaScript
Installation	notwendig, d.h. ausführbare Dateien müssen auf das System kopiert werden	nicht notwendig, die Dateien werden vom Browser über das Internet geladen
Updates	müssen lokal auf jedem Gerät installiert werden	zentral verwaltet auf dem Server
Ausführung	direkt auf dem System	im Browser
Cross Plattform	nein, für jedes System muss eine eigene App erstellt werden	ja
Offline-Nutzung	ja	nein
Zugriff auf Geräte- und Systemfunktionen	ja	nein
Zugriff auf lokal installierte Datenbanken, zum Beispiel SQLite	ja	nein
Ansteuerung von spezifischer Hardware über Treiber	ja	nein
Zugriff auf Dateisystem	ja	mit Einschränkungen

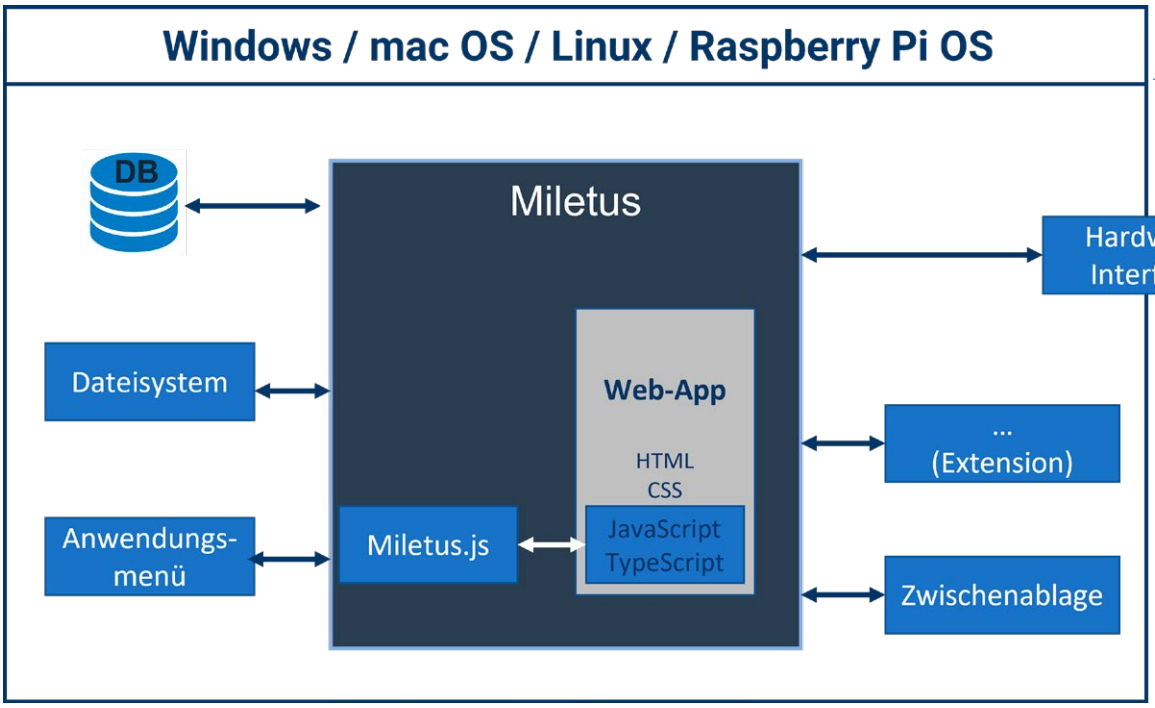


Bild 1. Architektur einer Miletus-Anwendung.

Systemschnittstellen und die Einbindung von individuellen Treibern am meisten für eine native Desktop-Applikation sprechen. Möchte man diese Anwendung auf unterschiedlichen Systemen ausführen, zum Beispiel unter Windows, macOS und einem Raspberry Pi, dann wird es schnell sehr aufwändig. Jedes System hat seine eigenen Spielregeln, was beispielsweise das Erstellen der Benutzeroberfläche betrifft. Mit anderen Worten: Drei Zielsysteme, drei Anwendungen und damit ein dreifacher Entwicklungsaufwand. Neben dem Aufwand scheitert es meist auch an den dafür notwendigen Kenntnissen, denn eine native Windows-App programmiert man vollständig anders als eine Anwendung für den Raspberry Pi oder für macOS. Die Web-App wird dagegen mit HTML, CSS und JavaScript erstellt und läuft im Browser; darüber hinaus vereinfachen leistungsfähige Bibliotheken wie Bootstrap die Gestaltung der Oberfläche.

In diesem Artikel stellen wir ein interessantes Framework namens *Miletus* [1] vor. Mit Miletus kann man eine neue oder bestehende Web-App zu einem Anwendungs-Package für unterschiedliche Zielsysteme (unter anderem für den Raspberry Pi) verpacken und dieses wie eine native Anwendung ausführen. Die Anwendung lässt sich dann offline nutzen und hat Zugriff auf die Systemfunktionen.

Das Web-Framework Miletus

Bevor wir zur praktischen Verwendung kommen, stellen wir die wichtigsten Eigenschaften von Miletus vor. Die erstellten Anwendungen funktionieren auf den Betriebssystemen Windows, macOS und Linux, inklusive Raspberry Pi OS. Man hat über die Miletus-API einen Zugriff auf die System- und Gerätefunktionen, das Dateisystem und über Treiber auf externe am Gerät angeschlossene Hardware.

Aus Sicht des Elektronikers ist es ein besonderes Feature, dass man auf die Hardwareschnittstellen des Raspberry Pi, also die Ports GPIO, I²C, SPI und UART, und auf den Memory-Buffer zugreifen kann. Auf diese Weise werden fast alle Einschränkungen der Web-Applikation aufgehoben und man profitiert dennoch von deren Cross-Plattform-Fähigkeit. Zum besseren Verständnis zeigt **Bild 1** eine Skizze der Architektur einer auf dem Miletus-Framework basierenden Anwendung.

Durch die Einbettung der Web-Applikation in ein natives Anwendungspaket für das jeweilige Zielsystem ergeben sich neue Anwendungsfelder. Dazu gehören beispielsweise Applikationen zur Maschinensteuerung oder Anwendungen aus dem Bereich des Internet of Things (IoT). In beiden Anwendungsfällen muss man Signale über die Hardwareschnittstellen des Raspberry Pi senden und empfangen. Eine weitere Besonderheit des Frameworks ist die Erweiterbarkeit der APIs. Das erfolgt über so genannte *Shared Libraries*, unter Windows zum Beispiel in Form einer *dll-Datei*. Existiert eine solche Systembibliothek für das Zielsystem, zum Beispiel ein Treiber für individuelle Hardware, dann kann man diese Bibliothek in Miletus einbinden und dann die Funktionen aus der Web-App verwenden. Miletus nutzt die jeweils vom Zielsystem bereitgestellte Standardbrowser-Engine, das heißt, auf:

- *Windows*: WebView2
- *macOS*: WebKit (Safari)
- *Linux/ Raspberry Pi OS*: WebKitGTK

Man kann im Regelfall davon ausgehen, dass auf den Zielsystemen die genannten Browser-Engines installiert sind, so dass eine zusätzliche Auslieferung durch die Applikation nicht notwendig ist. Dadurch wird das zu verteilende Anwendungs-Package nicht sehr groß, das heißt die Verteilung der App wird beschleunigt und sie geht schonend mit den Systemressourcen um. Das ist besonders für eine Ausführung auf dem Raspberry Pi wichtig. Dadurch, dass der bereits vorhandene Browser des Systems genutzt wird, profitieren die Anwendungen auch automatisch von Funktions- und Sicherheitsupdates des Browsers.

Einschränkend muss man ergänzen, dass auf dem Zielsystem einige Systemvoraussetzungen zu prüfen sind und gegebenenfalls Systembibliotheken nachinstalliert werden müssen. Auf den Zielsystemen müssen folgende Bibliotheken vorhanden sein:

- **Windows 32-/64-Bit:** Es muss die aktuelle Version des Edge Chromium-Browsers installiert sein. Zusätzlich müssen die Dateien *WebView2Loader_x86.dll* (32-Bit) beziehungsweise *WebView2Loader_x64.dll* (64-Bit) in die Ordner *System32* respektive *SysWow64* kopiert werden. Diese Dateien werden mit der Miletus-Installation bereitgestellt.
- **Linux/ Raspberry Pi OS:** Es wird die Grafikschnittstelle GTK3

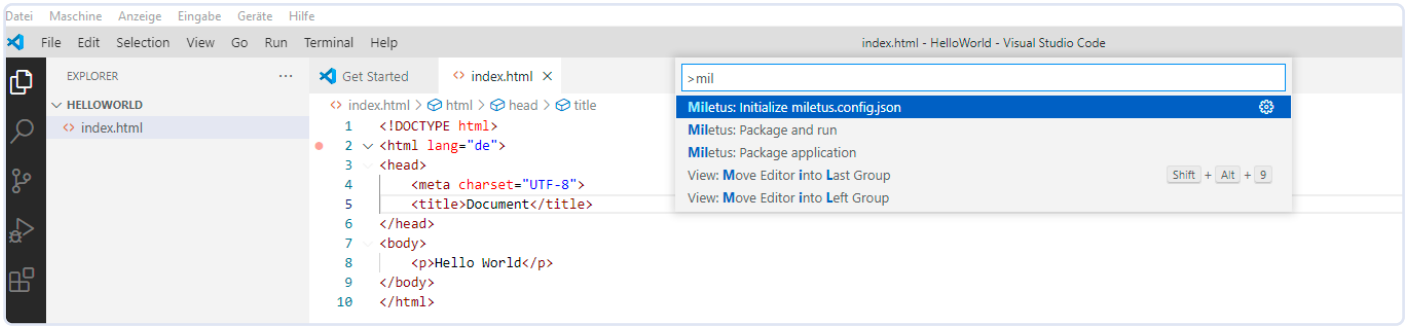


Bild 2. Befehle für den Miletus-Packager in Visual Studio Code.

genutzt, welche mittels des Befehls `sudo apt install libwebkit2gtk-4.0-dev` über die Kommandozeile installiert werden kann.

- **macOS:** Es sind keine weiteren Voraussetzungen notwendig.

Alternative: Electron

Das Framework Electron [2] wird ebenso dazu genutzt, um eine Web-App zu einer Desktop-Anwendung zu verpacken. Beispielsweise ist damit der hier genutzte Editor Visual Studio Code erstellt (das heißt, dieser ist eigentlich eine Web-App). Im Gegensatz zu Miletus liefert Electron eine eigene Browser-Engine aus, die Anwendungspackages sind dadurch deutlich größer. Ebenso können Electron-Apps nicht auf dem Raspberry Pi gestartet werden. Auch ist das API nicht erweiterbar.

Installation und erste Experimente

Dazu brauchen wir:

- *Eine* **Web-Applikation:** Diese kann für erste Experimente lediglich aus einer HTML-Datei (*index.html*) bestehen und nur einen „Hello World"-Text ausgeben. Wir platzieren noch einen Button, um die Funktionsweise der Miletus-API zu demonstrieren (siehe unten).
- **Visual Studio Code:** Diesen Editor lädt man von [3] für das eigene Betriebssystem herunter.
- **Extension für Visual Studio Code:** Diese laden Sie auch von der Miletus-Seite [1] herunter und installieren sie manuell in Visual Studio Code (*vsix-Datei*).
- **Packager (optional):** Diesen laden Sie für Windows, macOS oder Linux von [4] herunter und installieren ihn. Damit kann der Packager auch über die Kommandozeile genutzt werden.

Ein Hinweis für erfahrene Web-Entwickler: Sofern auf Ihrem System der Paketmanager npm installiert ist, können Sie Miletus auch über `npm install miletus` installieren. Danach ist die API auf dem Entwicklungssystem installiert und kann in eigene Web-Projekte eingebunden werden.
Hier die *index.html* unseres „Hello World"-Beispiels. Den Quellcode zu den hier vorgestellten Beispielen findet man auf der Webseite des Autors unter [5].

```
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script src="miletus.js"></script>
    <script>
        function save() {
            miletus.dialogs.showSaveDialog();
        }
    </script>
</head>

<body>
    <p>Hello World</p>
   <button onclick="save()">Save to File</button>
</body>
</html>
```

Damit haben wir schon alles zusammen. Sofern Sie die Extension für *Visual Studio Code* installiert haben, ist der Packager automatisch eingerichtet. In Visual Studio Code stehen dann folgende zusätzliche Befehle zur Verfügung (**Bild 2**):

- *Miletus: Package application*: Erstellt die Packages der Anwendung für die Zielsysteme Windows, macOS und Linux/ Raspberry Pi OS.
- *Miletus: Initialize config*: Es wird ein Template für die Konfigurationsdatei *miletus.config.json* angelegt.
- *Miletus: Package and run*: Es werden die Packages für die gewählten Zielsysteme erstellt und die Anwendung auf dem Entwicklungssystem gestartet. Ein Raspberry Pi kann nicht als Entwicklungsumgebung genutzt werden, auf diesem lässt sich die Anwendung nach dem Packen lediglich ausführen.

Öffnen Sie den Ordner mit der *index.html*-Datei. Sie können diese HTML-Datei zum Testen in Ihrem Browser anzeigen (**Bild 3**).
Jetzt wollen wir diese „Web-App" in eine Desktop-Anwendung verpacken. Die Steuerung des Packagers erfolgt über die Konfigurationsdatei *miltetus.config.json*. Hier werden der Name und Version der Anwendung,

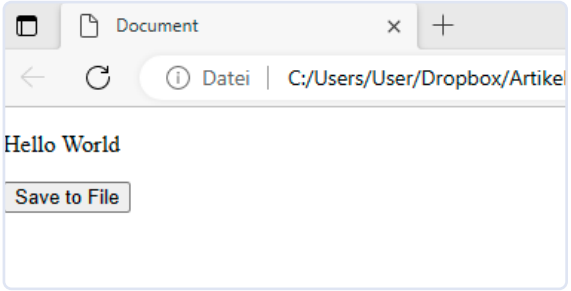


Bild 3. Eine Web-App mit Button.

die Zielsysteme, der Einstiegspunkt für die Applikation und noch einige Dinge mehr definiert. Die genaue Beschreibung der Konfigurationsvariablen findet man in der Dokumentation [6]. Sie können die Datei manuell anlegen oder automatisch mit Visual Studio Code generieren. Letzteres geschieht über die Befehlsschaltfläche (*[Strg + Umschalt + P]*) und den Befehl: *Miletus: Initialize config*. Durchlaufen Sie die Abfragen: Vergeben Sie einen Namen für die Applikation, ordnen Sie die Startdatei zu (*index.html*) und wählen Sie die gewünschten Zielsysteme aus. Die generierte *miltetus.config.json*-Datei sieht folgendermaßen aus:

```
{
    "name": "Dialogs",
    "output": "output",
    "debug": true,
    "main": {
        "html": "index.html"
    },
    "target": [
        "win_ia32"
    ],
    "include": [
        "index.html",
        "miletus.js"
    ]
}
```

Aus den Angaben in der Konfigurationsdatei werden die Anwendungspakete für die Zielsysteme erstellt, zum Beispiel *win_ia32* für Windows 32-Bit. Das geschieht über den Befehl *Miletus: Package application*. Das sind für Windows eine *exe*-Datei, für macOS ein Ordner *.app* und die zugehörige *entitlements*-Datei für die Berechtigungen sowie für Linux eine ausführbare Datei und eine *.desktop*-Datei (**Bild 4**).
Kommen wir zu einem ersten Start der Web-App als Desktop-Applikation auf dem Zielsystem. Dazu müssen die ausführbaren Dateien auf das Zielsystem kopiert werden, das heißt im Einzelnen:

- **Windows:** Starten Sie die *exe*-Datei. Arbeiten Sie unter Visual Studio Code unter Windows, dann können Sie die Anwendung mittels des Befehls *Miletus: Package and run* direkt ausführen.
- **Linux:** Kopieren Sie den Ordner *Output* auf das Linux-System. Vergeben Sie die Rechte an den Dateien über den Befehl: `sudo chmod -R 755 /[Pfad der Anwendung]`. Starten Sie jetzt die Anwendung.
- **macOS:** Sie müssen die Applikations-Dateien aus dem Order *Output* kopieren, die Rechte zuweisen und dann kann die Anwendung gestartet werden. Beachten Sie: Auf einem ARM-basierten macOS-System ist eine Codesignierung der App notwendig.

Die gestartete „Hello World"-Anwendung sehen Sie in **Bild 5**.

Hinweis: Kommt es zu einer Fehlermeldung oder einem leeren Fenster, dann prüfen Sie die oben genannten Voraussetzungen auf Ihrem System.

Miletus-API verwenden

Eine „Hello World"-Applikation zeigt noch nicht die Verwendung des Frameworks. Das demonstrieren wir an einem einfachen Beispiel. Die Miletus-API bietet unter anderem die folgenden Funktionen:

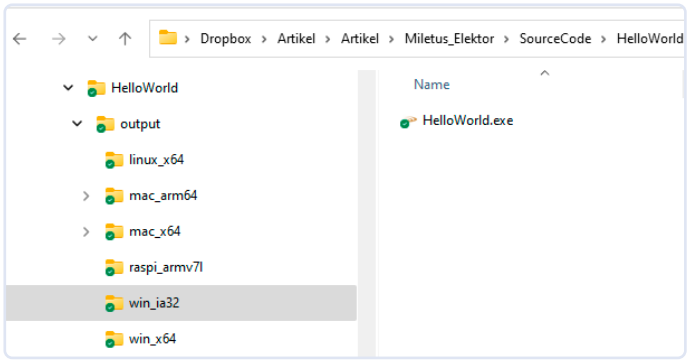


Bild 4. Anwendungspakete für die unterschiedlichen Zielsysteme.

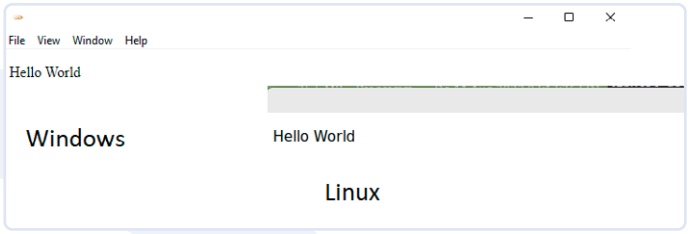


Bild 5. „Hello World"-App unter Windows und Linux.

- **Dialoge:** Das Miletus-API stellt hierfür Funktionen zur Anzeige von Datei-Dialogen (`showOpenDialog(…)`, `showSaveDialog(…)`), einen Fehlerdialog (`showErrorDialog(…)`) und eine Messagebox (`showMessageBox(…)`) bereit. Die Funktionen werden im JavaScript- oder alternativ im TypeScript-Quelltext aufgerufen. Zunächst muss man die zugehörige Bibliothek *miletus.js* einbinden. Danach kann beispielsweise über:

  ```
  miletus.dialogs.showSaveDialog()
  ```

  das *Speichern unter...*-Dialogfeld aufgerufen werden. Diese Quelltextzeile wird dazu in eine Funktion mit der Bezeichnung `save()` eingebunden, welche wiederum über das `onclick()`-Event eines Buttons aufgerufen wird. **Bild 6** zeigt den Aufruf des Datei-Dialogs.
- **Dateizugriff:** Miletus bietet Methoden zum Laden und Schreiben von Text- und Binärdateien über die Methoden `loadTextFile(…)`, `saveTextFile(…)`, `loadBinaryFile(…)` und `saveBinaryFile(…)`. Es gibt Methoden zum Überwachen von Dateienänderungen, nämlich `watchFile(…)`, `removeWatch(…)` und `removeAllWatches(…)` und für Drag & Drop-Operationen die Methode `startDrag(…)`. Ebenso stehen Funktionen zur Anzeige aller Elemente eines Dateiordners (`openFile(…)`) und zum Löschen von Dateien (`moveToBin(…)`) zur Verfügung.
- **Anwendungsmenüs:** Man kann Menüeinträge zum Anwendungsfenster hinzufügen. Ein Menü kann Untermenüpunkte enthalten. Auf das Klicken eines Menüpunktes kann reagiert werden. Über `application.setMenu(menu)` wird ein zuvor definiertes Menü der Anwendung zugewiesen.
- **Systemfunktionen:** Beispielsweise kann man eine externe URL über `shell.openURL(…)` aufrufen. Diese wird im Browser angezeigt. Über `execute(…)` kann ein Systemkommando ausgeführt werden.

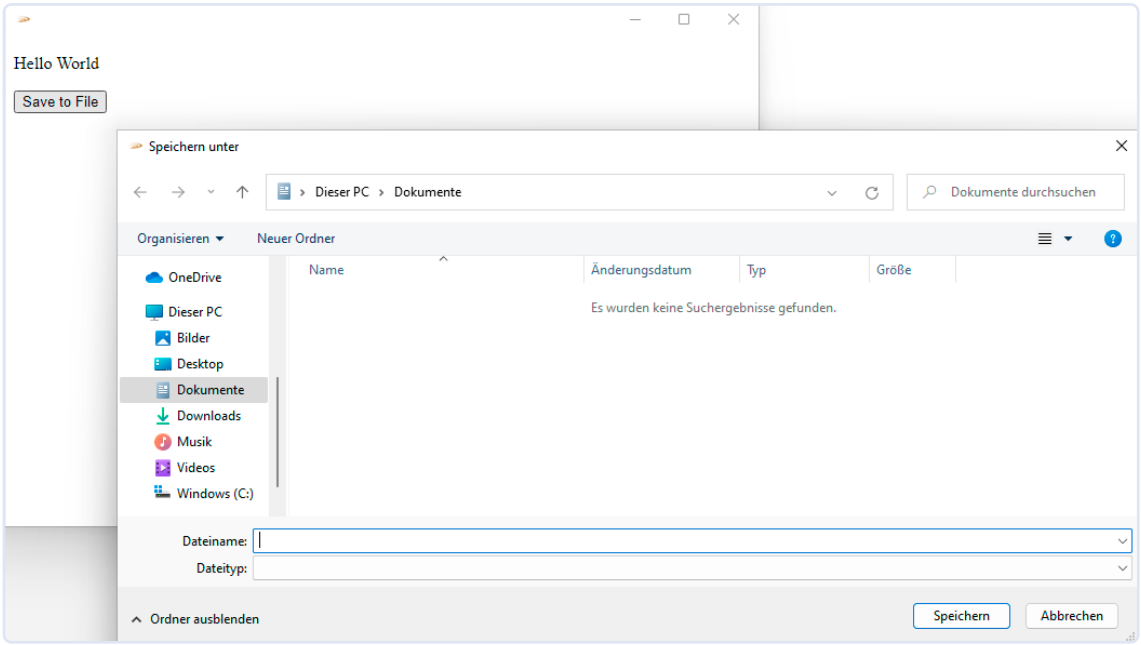


Bild 6. Dateidialog unter Windows – aufgerufen mittels Miletus-API.

- **Zugriff auf Datenbank:** Es werden die folgenden Datenbanken unterstützt: SQLite, MySQL, MSSQL, PostgreSQL, MS Access (Windows), Firebird und Interbase. Es muss lediglich die Verbindung zur Datenbank über einen sogenannten Connection-String in der Form:

```
let db = new DataBase('sqlite', {database: 'path_to_my/sqlitedb.db'})
```

konfiguriert werden.

- **Windows-Registry:** Auf Windows kann man auf die Registry zugreifen, also Schlüssel lesen, schreiben und überprüfen.

Weitere Funktionen (Anwendungsfenster, Lesen und Schreiben von ini-Dateien, Systemnachrichten) des Miletus-API sind in der Dokumentation beschrieben. Eine Besonderheit ist der Zugriff auf die Hardwareschnittstellen des Minicomputers Raspberry Pi. Wie das funktioniert, das sehen wir uns im kommenden Abschnitt an.

Eine App für den Raspberry Pi

Wir wollen jetzt eine App erstellen, welche am Raspberry Pi angeschlossene Sensoren ausliest. Die App wird als Web-App erstellt und dann mittels Miletus zu einem App-Package für den Raspberry Pi verpackt. Wir verwenden folgenden Versuchsaufbau.

- Raspberry Pi ab Version 2
- BME280 Environmental Sensor von Bosch Sensortec (technische Daten siehe **Tabelle 2**), in diesem Artikel wird ein Breakout-Board von Waveshare verwendet
- Anschluss des Sensors über das I²C-Interface
- Anschluss von Monitor, Maus und Tastatur am Raspberry Pi

Der BME280 ermittelt Werte für Temperatur, Luftfeuchtigkeit und Luftdruck. Wir möchten diese Werte über JavaScript auslesen und auf der Oberfläche anzeigen. Der Sensor wird direkt über die Steckverbinder mit den Anschlüssen auf der Platine des Raspberry Pi verbunden. Hierzu sind lediglich vier Kabel notwendig. VCC (rot) und GND (schwarz) des BME280-Moduls werden mit Pin 2 beziehungsweise Pin 39 des Raspberry-Pi-Erweiterungssteckers verbunden. SDA (Blau) und SCL (gelb) kommen an Pin 3 und Pin 5 des Raspberry Pi. **Bild 7** zeigt den Versuchsaufbau.

Auf dem Raspberry Pi installieren wir die aktuelle Version von Raspberry Pi OS, die wir mit Hilfe des Installers auf eine SD-Karte schreiben.

Tabelle 2: Technische Daten des Sensors BME280 [7]

Sensoreigenschaft	Werte
Arbeitsspannung	5 V/3 V
Interface	I2C/ SPI
Temperaturbereich	-40 bis 85°C, Auflösung 0,01 °C, Genauigkeit ±1 °C
Luftfeuchtigkeit	0 bis 100 RH, Auflösung 0,008% RH, Genauigkeit ±3% RH, Antwortzeit 1 s, Verzögerung <=2% RH
Luftdruck	300 bis 1.100 hPa, Auflösung 0,18 Pa, Genauigkeit ±1 hPa
Abmessungen	27 mm × 20 mm × 2 mm

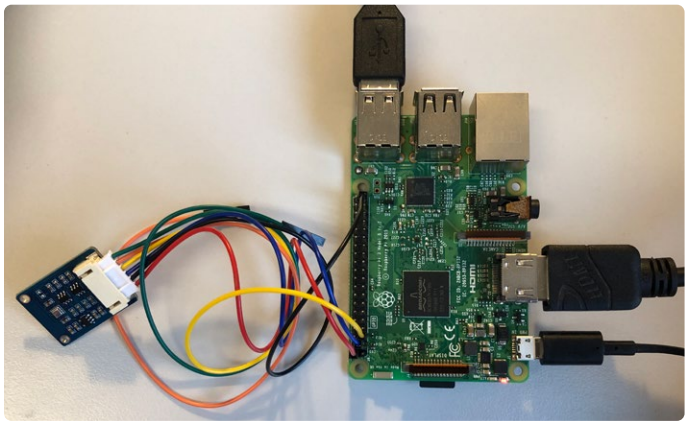


Bild 7. Versuchsaufbau von Raspberry Pi und Sensorplatine BME280.

Listing 1. index.html der Sensor-App

```
<!DOCTYPE html>
<html>
    <head>
        <title></title>
        <script src="../js/miletus.js"></script>
        <script src="../js/bme280.js" defer></script>
        <link rel="stylesheet" type="text/css" href="../css/style.css" />
    </head>
    <body>
        <h1>Weather Station</h1>
        <div class="container">
            <div class="child">
                <img class="image" width="100" src="../img/temp.png">
                <p class="label">Temperature</p>
                <div class="value" id="temperature"></div>

            </div>
            <div class="child">
                <img class="image" width="100" src="../img/air.png">
                <p class="label">Pressure</p>
                <p class="value" id="pressure"></p>

            </div>
            <div class="child">
                <img class="image" width="100" src="../img/humidity.png">
                <p class="label">Humidity  </p>
                <p class="value" id="humidity"></p>

            </div>
           <div class="buttonscontainer">
               <button id="start-reading" class="btn">Start reading</reading></button>
               <button id="stop-reading" disabled class="btn">Stopp reading </button>
           </div>
        </div>
        <img class="logo" src="../img/logo.png">
    </body>
</html>
```

Die Kommunikation zwischen beiden Bausteinen erfolgt über das I²C-Interface, welches man in Raspberry Pi OS aktivieren muss. Rufen Sie dazu ein Terminal auf dem Raspberry Pi auf und geben den Befehl:

```
sudo raspi-config
```

ein. Im nächsten Schritt wählt man die Option:

```
Interfacing Options -> I2C
```

und aktiviert so den I²C-Kerneltreiber. Nach dem Speichern der Einstellungen und einem Reboot des Raspberry Pi

```
sudo reboot
```

kann man Daten über das I²C-Interface austauschen.
Damit sind die Vorbereitungen abgeschlossen. Kommen wir zur Programmierung. Das Projekt besteht aus den folgenden Dateien:

- *index.html*: Diese Datei ist der Startpunkt der Web-Applikation (**Listing 1**). Die *index.html* verweist auf die CSS-Datei (*style.css*) und die beiden JavaScript-Dateien *miletus.js* und *bme280.js*.
- *style.css*: Das Layout und das Design der Web-Applikation werden über CSS definiert.
- *miletus.js*: Enthält die Miletus-API.
- *bme280.js*: Es werden alle Funktionen für die Kommunikation mit dem Sensor definiert.

Als Erstes haben wir einen minimalen Designentwurf gemacht (**Bild 8**). Hierzu können Sie ein beliebiges Prototyping-Tool nehmen. Hilfsweise genügt auch eine handgezeichnete Skizze. Die Web-App haben wir dann im nächsten Schritt mit HTML (Struktur), CSS (Layout, Design) und JavaScript (Funktionen) realisiert.
Die App kann man zunächst wieder in einem Browser auf jedem beliebigen System starten (**Bild 9**). Hier lässt sich das Design der Web-App begutachten. Die eigentliche Funktion, das Auslesen der Sensordaten, funktioniert jedoch nur auf dem Raspberry Pi.

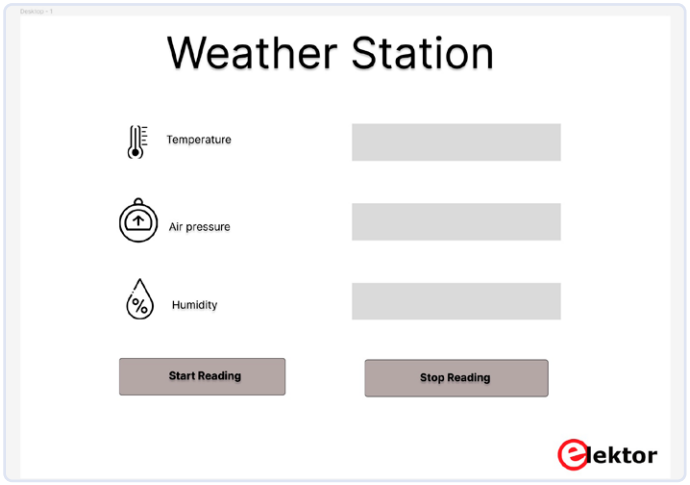


Bild 8. Entwurf der App (Layout und Design) als Prototyp.

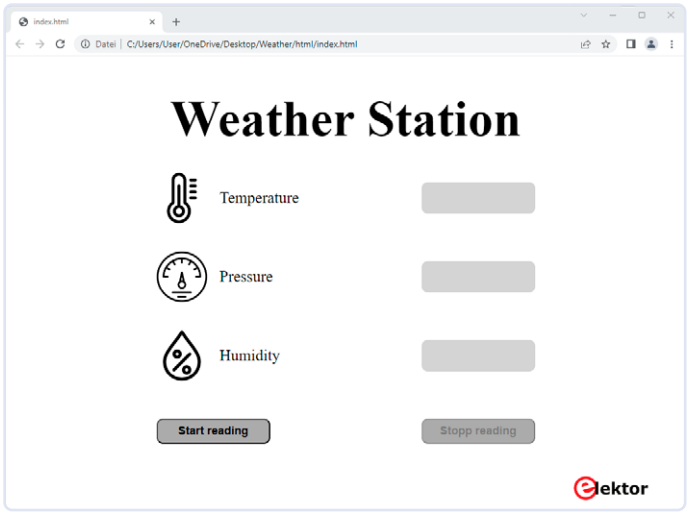


Bild 9. Wetter-App – ausgeführt im Browser.

Sehen wir uns den Quellcode noch etwas genauer an. In der HTML-Datei wird ein Button mit der Aufschrift „*Start reading*“ definiert:

```
<button class="start" id="start-reading">Start reading</
button>
```

Klickt man auf diesen Button, dann sollen die Sensordaten ausgelesen werden. In der *bme280.js*-Datei wird dazu das `onclick`-Event gebunden:

```
document.getElementById("start-reading").onclick =
startReading;
```

Dieses verweist auf eine JavaScript-Funktion namens *startReading* mit dem Inhalt von **Listing 2**.
Es wird eine Kalibrierung der Sensordaten durchgeführt, der *Start*-Button wird deaktiviert, der *Stopp*-Button aktiviert, die Daten werden in einer Schleife ausgelesen und so weiter. Diese Vorgehensweise wird durch die Konventionen des Sensors BME280 bestimmt und kann in der Dokumentation und in Beispielen nachgelesen werden [7][8]. Blicken wir zum Beispiel in die Funktion `readSensorData()`, dann finden wir folgende Quellcodezeile:

```
let rawValues = await i2c.readBuffer(0xFA, 3)
```

Darüber werden die Temperaturdaten ausgelesen. Das Objekt `i2c` ist wiederum wie folgt definiert:

```
i2c = new miletus.I2C()
```

Über die Miletus-API hat man einen Zugriff auf das I²C-Interface des Raspberry Pi. Auf diese Weise können wir in der Web-Applikation mittels JavaScript die Daten vom Sensor auslesen. Erstellen wir jetzt die Konfigurationsdatei *miltetus.config.json* für das Projekt in Visual Studio Code wie oben beschrieben (Befehl: *Miletus: Initialize config*). Tragen Sie hier auch die Verweise auf die Bilddateien ein, die auf der Benutzeroberfläche erscheinen sollen (zum Beispiel Icons).
Danach können wir das Anwendungs-Package für den Raspberry Pi (Befehl: *Miletus: Package application*) generieren. Die erzeugten Anwendungsdateien kopieren wir auf den Raspberry Pi, zum Beispiel mit Hilfe eines USB-Sticks. Für die Anwendungsdatei ist das Attribut für eine ausführbare Datei zu setzen. Starten Sie dann die App auf dem Raspberry Pi. Diese sollte in einem eigenen Fenster ausgeführt werden. Mit Klick auf den Button werden die Sensordaten ausgelesen und angezeigt (**Bild 10**).

Weitere Möglichkeiten mit dem Raspberry Pi

Wir haben die Kommunikation über das I²C-Interface betrachtet. Mit Miletus erreichen wir folgende weitere Schnittstellen auf den Raspberry Pi [9]:

- *GPIO*: Wir können Daten über die 26 GPIO-Pins lesen und schreiben.
- *SPI*: Dieser serielle Bus benötigt drei Leitungen für die Kommunikation und basiert auf dem Master/ Slave-Prinzip.
- *UART*: Dieser Bus dient unter anderem zur Kommunikation mit RS232- oder RS485-Schnittstellen, zum Beispiel zur Datenübertragung mit Mikrocontrollern.

Ebenso kann auf den *MemoryBuffer* zugegriffen werden. Man verwendet diesen, um auf Shell-Anwendungsebene den Speicherpuffer zu lesen und zu schreiben.

Betrachten wir noch ein Beispiel zum Datenaustausch über GPIO. Importieren Sie die Bibliothek *gpio* mittels:

```
import { gpio } from 'miletus'
```

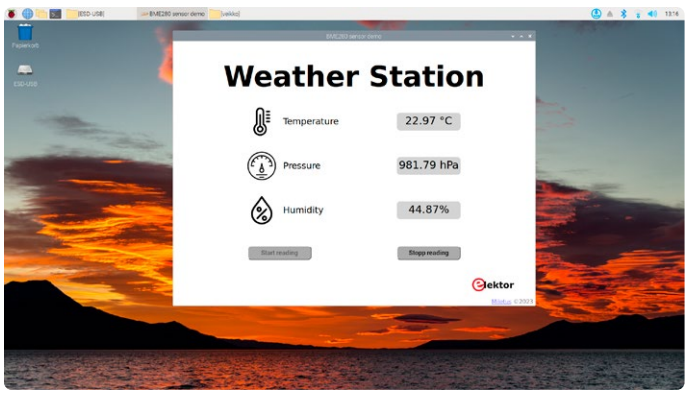


Bild 10. Wetter-App auf dem Raspberry Pi.

Listing 2. Funktion zum Auslesen der Sensor-Daten

```
async function startReading() {
    await readCalibrationData();
    document.getElementById("start-reading").disabled = true;
    document.getElementById("stop-reading").disabled = false;
    doRead = true;
    while (doRead) {
        await readSensorData();
        await new Promise(resolve => setTimeout(resolve, 1000));
    }
}
```

Passende Produkte

- **Raspberry Pi 4 B (2 GB RAM)**
  www.elektor.de/18965
- **Raspberry Pi 5 (4 GB RAM)**
  www.elektor.de/20598

Jetzt können wir über die Methoden `setPin(…)` den Modus des Pins setzen (Eingabe oder Ausgabe), mit `write(…)` einen Pin aktiv oder inaktiv schalten (High- oder Low-Signal) und mit Hilfe von `read(…)` einen Pin auslesen. Damit die Oberfläche der Web-Applikation während des Datenaustausches mit der Schnittstelle reaktionsfähig bleibt, muss der Zugriff asynchron erfolgen, also zum Beispiel:

```
await gpio.setPin(1, 'read')
```

Auf diese Weise wird Pin 1 des GPIO-Ports auf den lesenden Modus programmiert.

Erweiterbarkeit

Das Framework kann sehr leicht erweitert werden. Man kann aus Miletus eine beliebige Bibliothek laden und dann aus der Web-Anwendung heraus auf die exportierten Funktionen dieser Library zugreifen. Das betrifft den Zugriff auf *dll*- (Windows), *dylib*- (macOS) und *so*-Dateien (Linux/ Raspberry). Auf diese Weise kann man spezielle Systemfunktionen der Zielsysteme in einer mit Miletus verpackten Web-Applikation nutzen, beispielsweise zur Steuerung von an den PC angeschlossene Hardware über Treiber des Systems. Informationen zu diesem Thema findet man unter [10].

Weniger Programmieraufwand

Der zunächst ungewöhnliche erscheinende Schritt, eine Web-App zu einer nativen Applikation zu verpacken, bietet einige Vorteile. Die App kann auf allen Systemen ausgeführt werden (Cross Plattform), sie verhält sich wie eine native Applikation, kann offline genutzt werden und hat Zugriff auf die Systemfunktionen. Auf dem Raspberry Pi kann sie Daten über die Schnittstellen senden und empfangen. Damit kann man auch eine Web-App zur Steuerung von elektronischen Schaltungen verwenden. Diese Vorgehensweise kann für viele Anwendungsfälle eine deutliche Vereinfachung und eine Reduzierung des Programmieraufwandes bedeuten. Das gilt besonders dann, wenn man mehrere Systeme simultan erreichen möchte.

220616-02

Über den Autor

Dr. Veikko Krypczyk ist Softwareentwickler, Trainer und Fachautor. Seine Leidenschaft und sein Wissen gibt er in Coachings, Seminaren, Trainings und Workshops weiter. Workshops und Unterstützung können Sie unter *https://larinet.com* anfragen und die Agenda vorab einsehen.

Sie haben Fragen oder Kommentare?

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

WEBLINKS

[1] Miletus: https://miletus.org/
[2] Electron: https://www.electronjs.org/
[3] Visual Studio Code: https://code.visualstudio.com/
[4] Miletus: Download des Packagers: https://miletus.org/download.html
[5] Download des Beispielcodes: https://wp.larinet.com/?page_id=663
[6] Miletus: Konfigurationsvariablen des Packagers: https://miletus.org/doc/gettingstarted/packager/
[7] Datenblatt des BME280: https://www.waveshare.com/w/upload/9/91/BME280_datasheet.pdf
[8] BME280 Treiberbibliothek: https://github.com/BoschSensortec/BME280_driver/blob/master/bme280.c
[9] Miletus: GPIO-Zugriff: https://miletus.org/doc/reference/gpio/
[10] Miletus: Erweiterungen: https://miletus.org/doc/gettingstarted/extensibility/

Q&A mit Nemo Galletti

Von 4G zu 5G

Ist es wirklich so einfach?

Fragen von Roberto Armani (Elektor)

Täglich werden wir mit stolzen Werbebotschaften über die Wunder der neuesten 5G-Lizenzen überschwemmt. Zweifellos tragen sie wesentlich zur Netzwerkgeschwindigkeit bei, aber ist alles Gold, was glänzt? In diesem Interview mit Nemo Galletti, einem italienischen Manager eines Unternehmens, das sich auf die Produktion von Gigahertz-Antennen für Telekommunikationsnetze spezialisiert hat, betrachten wir die Unterschiede zwischen 5G und der älteren 4G-Technologie. Wir prüfen auch, ob möglicherweise etwas von den Anbietern „unter den Teppich gekehrt" wurde.

Roberto Armani: Nemo, würdest du uns etwas über deine Erfahrungen in der Telekommunikationsbrabche erzählen?
Nemo Galletti (Radio Frequency Systems): 2024 werde ich 40 Jahre in der Telekommunikationsbranche tätig sein. Ich arbeite derzeit bei Radio Frequency Systems, einem von Nokia kontrollierten Unternehmen, das sich auf passive Komponenten spezialisiert hat, die hauptsächlich in Mobilfunknetzen eingesetzt werden, wie spezielle Feeder, Antennen, Wellenleiter und spezielle Koaxialkabel (radiating cables).

Roberto: Wie würdest du 5G in wenigen Worten beschreiben?
Nemo: 5G steht für die fünfte Generation der Mobilfunktechnologie. Im Vergleich zu früheren Generationen von Mobilfunknetzen lassen sich die Hauptvorteile von 5G in drei Hauptmerkmalen zusammenfassen: Geschwindigkeit, Kapazität und geringere Latenzzeiten.

Roberto: Wie unterscheidet sich die 5G-Technologie im Detail von früheren Generationen, etwa 4G?
Nemo: Es gibt relevante technologische Aspekte, die durch 5G-Netze eingeführt wurden und sich von den vorherigen Generationen von Mobilfunknetzen unterscheiden:

- **Datenübertragungsgeschwindigkeit**: Im Vergleich zu 4G bietet 5G deutlich schnellere Datenübertragungsgeschwindigkeiten. Während 4G-Netze typischerweise Download-Geschwindigkeiten von bis zu mehreren hundert Megabit pro Sekunde bieten, können 5G-Netze Geschwindigkeiten von mehreren Gigabit pro Sekunde erreichen. Allerdings sollten wir besser „könnten erreichen" sagen, wie im Folgenden erläutert wird.
- **Latenz**: Latenz bezieht sich auf die Zeit, die Daten benötigen, um über ein Netzwerk zwischen Geräten zu reisen. Während 4G-Netze typischerweise Latenzen im Bereich von einigen zehn Millisekunden haben, können 5G-Netze ultra-niedrige Latenzen im Bereich von nur wenigen Millisekunden erreichen.
- **Netzkapazität**: 5G-Netze bieten eine deutlich höhere Netzkapazität als 4G. Das bedeutet, dass 5G-Netze eine viel größere Anzahl von verbundenen Geräten gleichzeitig ohne Leistungseinbußen bewältigen können.
- **Spektrale Effizienz**: Die 5G-Technologie ist spektral effizienter als die Vorgängertechnologien. Das bedeutet, dass sie mit der gleichen Menge an Funkfrequenzen mehr Daten übertragen kann. 5G erreicht dies durch hoch entwickelten Modulationstechniken, eine effizientere Nutzung der verfügbaren Frequenzbänder und die Verwendung höherer Frequenzbänder.
- **Network Slicing**: 5G führt das Konzept des Network Slicing ein, das es Netzbetreibern ermöglicht, innerhalb einer einzigen physischen Netzinfrastruktur mehrere virtuelle Netze aufzubauen. Jedes Netzwerk-Slice kann für bestimmte Anwendungsfälle oder Anwendungen optimiert werden, zum Beispiel für verbessertes mobiles Breitband, ultrazuverlässige Kommunikation mit geringer Latenz und massive IoT-Bereitstellungen.

Bild 1. Weltweite Nutzung der 5G-Frequenzen. (Quelle aller Bilder, wenn nicht anders angegeben: Radio Frequency Systems)

- **Edge Computing**: 5G-Netze ermöglichen Edge-Computing-Funktionen, indem Rechenressourcen näher „an den Rand" des Netzes gebracht werden. Dies verringert die Entfernung, die Daten zurücklegen müssen, verbessert die Latenz weiter und ermöglicht eine schnellere Verarbeitung von Anwendungen, die Echtzeitanalysen oder Antworten mit geringer Latenz erfordern.
- **Verbesserte Konnektivität**: 5G unterstützt fortschrittliche Konnektivitätsfunktionen wie Beamforming und massive MIMO-Antennensysteme (Multiple Input Multiple Output). Diese Technologien verbessern die Signalqualität, erhöhen die Netzabdeckung und steigern die Gesamtleistung und Kapazität des Netzes. Beamforming ermöglicht es dem Netzwerk, sein Signal auf bestimmte Geräte zu fokussieren, während MIMO mehrere Antennen verwendet, um die Signalqualität und Kapazität zu verbessern.
- **Millimeterwellen-Frequenzen**: 5G führt die Nutzung höherfrequenter Bänder ein, darunter Millimeterwellen-Frequenzen. Diese Hochfrequenzbänder bieten wesentlich größere Bandbreiten und ermöglichen schnellere Datenraten.

Roberto: Wir verbinden 5G oft mit dem Einsatz neuer Funkfrequenzen. Welche Frequenzbänder werden oder sollen für 5G-Netze eingesetzt werden? Sind sie je nach Land unterschiedlich?

Nemo: Die 5G-Standardisierungsgremien haben spezifische Bänder definiert, die für 5G verwendet werden sollen. Sie sind mit dem Buchstaben „n" gefolgt von einer Zahl gekennzeichnet. Die wichtigsten nXX-Frequenzbänder, die für 5G zugewiesen sind oder zugewiesen werden sollen, sind in **Tabelle 1** aufgeführt (aber es können viele weitere Bänder verwendet werden). Für das am häufigsten genutzte Band, n78, sind in derselben Tabelle genauere Werte angegeben. Das Diagramm im **Bild 1** ist eine starke Vereinfachung. Europa nutzt nicht in jedem Land die gleichen Frequenzen: Zum Beispiel plant Deutschland nicht, n41 für 5G zu verwenden, während die genutzten

Tabelle 1: Weltweite Bandcodes und Frequenzallokation

Band Code	Frequenz (GHz)	Europa	USA/Kanada	China	Japan	Korea	Australien
n28	0,7	x	x		x		x
n40	2,3						x
n41	2,5		x	x			
n78	3,5	3,4-3,8	3,7-4,3	3,3-4,9	3,6-4,9	3,4-3,7	3,5
n257	28	x	x	x	x	x	x
n258	26	x	x				x

Bild 2. Ein typischer 4G/5G-Turm für die Bänder 700 MHz und 1.800 MHz, mit einer detaillierten Ansicht des Inneren eines Reflektors.

Unterbänder des am häufigsten verwendeten Bandes, n78, von Land zu Land unterschiedlich sein werden. Die Tendenz in Europa besteht darin, in diesem Band unter 3,8 GHz zu bleiben. Da die verschiedenen von 5G genutzten Frequenzen sehr unterschiedlich sind, könnte die Frage eher lauten, welche besser ist.
Wir haben gesehen, dass 5G niedrige, mittlere und hohe Frequenzen verwendet. Eine niedrige Frequenz (definiert als elektromagnetische Emission mit einer Frequenz von mehreren hundert Megahertz) hat die Fähigkeit, Hindernisse zu durchdringen und eine viel größere Entfernung zu überbrücken als eine hohe Frequenz, kann jedoch weniger Daten pro Zeiteinheit übertragen (typischerweise in Bits pro Sekunde oder Bit/s ausgedrückt). Eine hohe Frequenz (im Gigahertz-Bereich) hat dagegen eine viel geringere Reichweite, kann jedoch viele Daten pro Zeiteinheit transportieren. Aus diesem Grund kann eine unzuverlässige (gehinderte) Verbindung zu einem Hochfrequenzband manchmal langsamer sein als eine gute Verbindung zu einem Niedrigfrequenzband.
Daher können wir nicht sagen, dass ein Frequenzband besser ist als ein anderes: Es gibt verschiedene Frequenzen für verschiedene Zwecke, und das beste 5G-Netz ist das, das alle drei Frequenzbänder basierend auf dem spezifischen Bedarf des Augenblicks gut nutzt. Ein Video-Stream, der Zugriff auf E-Mails oder ein intelligentes Thermostat, das mit einem Smart Home verbunden ist, haben unterschiedliche Anforderungen in Bezug auf die Bandbreite!
Aus diesem Grund verwenden neue 5G-Smartphones eine Technologie namens „adaptive beam switching“, die es ihnen ermöglicht, von einem Frequenzband zum anderen zu wechseln, um eine stabile Verbindung aufrechtzuerhalten. Diese Funktion maximiert die Vorteile einer Verbindung über mehrere Frequenzen gleichzeitig. Besonders bei Millimeterfrequenzen, wie dem Hochband n257/n258, ermöglicht eine sehr hohe Frequenz extrem hohe Datenraten, jedoch können bewölktes oder nebliges Wetter die Millimeterwellen beeinträchtigen. Ohne adaptives Beam Switching könnten wir unsere Verbindung nicht auf die Millimeterübertragung stützen.
Die am häufigsten genutzten 5G-Frequenzen werden zumindest anfangs die mittleren Frequenzen oder „sub 6“ zwischen 1 GHz und 6 GHz sein. Im Vergleich zu Millimeterwellen ermöglichen diese Frequenzen eine gute Datenrate, ausreichende Kapazität zur Durchdringung von Hindernissen und sind weniger anfällig für Störungen. Niedrige Frequenzen sind dagegen ideal für Geräte, Sensoren, IoT, Controller und Smart Homes, bei denen eine niedrige Datenrate ausreicht, aber eine hohe Durchdringung und niedrige Latenz erforderlich sind.

Roberto: Zum Thema Frequenz gehören auch die 5G-Antennen. Unterscheiden sich diese von 4G-Antennen?
Nemo: Vom Konzept her sind 4G- und 5G-Antennen nicht unterschiedlich. Allerdings haben einige innovative Technologien, die mit 5G eingeführt wurden, auch Auswirkungen darauf. Hier konzentrieren wir uns auf zwei Elemente: Eines ist physisch, das heißt die Größe des Dipols sollte so nah wie möglich an der Hälfte der Wellenlänge λ liegen. Die vereinfachte Formel zur Berechnung der optimalen Länge (in Millimeter) eines λ/4-Dipols lautet: Lichtgeschwindigkeit [m/s] / Frequenz [kHz] / 2.
Dipole können die gleiche Größe wie die Wellenlänge oder ein Vielfaches davon haben, ihre Form kann linear, V-förmig oder gebogen sein, aber ihre Größe ist immer proportional zur Wellenlänge. Wir haben gesehen, dass die niedrigste Frequenz für 5G bei 700 MHz liegt, daher sind diese Dipole länger als die bei 800 MHz für 4G verwendeten, während für Millimeterwellen 5G eine Matrix aus mehreren sehr kleinen Dipolanordnungen einsetzt. Um eine Vorstellung davon zu geben: Bei 700 MHz beträgt die typische λ/4-Dipollänge 214 mm, während wir bei 28 GHz von 5,35 mm sprechen.

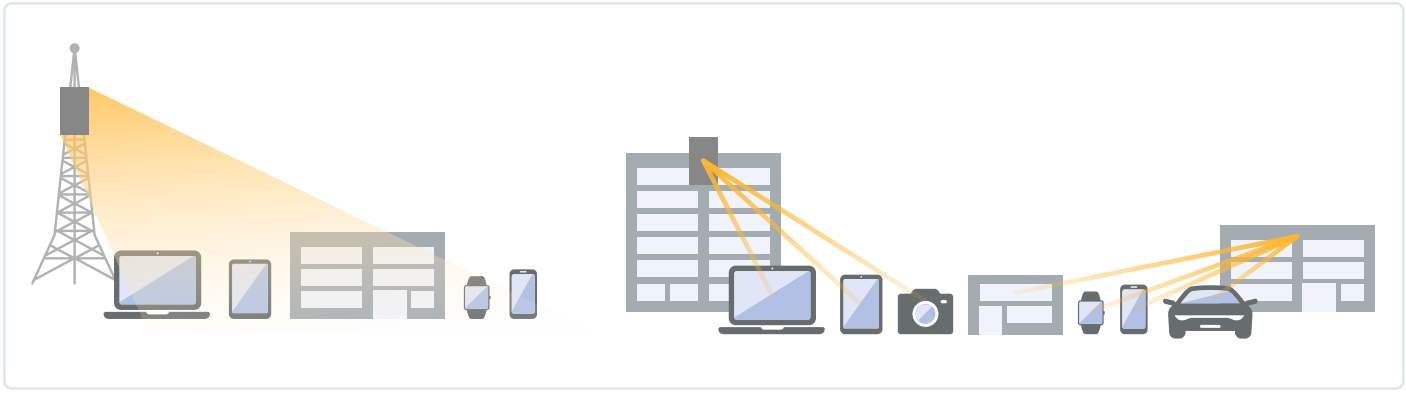


Bild 3. Unterschied zwischen dem standardmäßigen Sendegebiet eines 4G-Turms (links) und einem modernen, optimierten 5G-Turm mit Beamformungsfähigkeiten (rechts).

Das zweite Element bezieht sich auf die Technologie: Die „traditionelle“ 4G-Antenne besteht für jede Frequenz aus einer vertikalen Anordnung von Dipolen, die auf einem Reflektor montiert sind. Im **Bild 2** sehen Sie einen typischen Sendemast mit dem Satz von Reflektoren, die im klassischen 120°-Emissionsschema montiert sind, während das Bild in der unteren rechten Ecke die innere Struktur eines Reflektors zeigt. In diesem Fall sehen wir im Inneren die bei 700 MHz kreuzpolarisierten Dipole (goldene Elemente) und darin die kleineren, weißen Dipole für das 1,8-GHz-Band. Ab dem mittleren Band aufwärts bestehen 5G-Antennen oft aus einer Matrix von Arrays: mehrere vertikale Arrays parallel, zunächst 6×6 oder 8×8, bis zu 32×32 in Release 15 und mehr in zukünftigen Releases. Die Anzahl und Größe der Arrays steigt mit der Frequenz, die im aktiven Modus verwaltet werden, um Massive MIMO und Beamforming zu realisieren.
Daraus ergibt sich, dass sich die Antennen für niedrige Frequenzen nicht wesentlich von denen unterscheiden, die wir bei 4G sehen, aber die typischen 5G-Antennen für höhere Frequenzen werden rechteckige oder quadratische Felder sein, die mit zunehmender Frequenz immer kleiner werden.

Roberto: Du hast Massive MIMO, Beamforming und Dipol-Matrizen erwähnt. Sind das neue Technologien?
Nemo: Massive MIMO und Beamforming durch Matrixarrays von Dipolen sind keine neuen Technologien. Wenn wir jedoch über den kommerziellen Mobilfunkeinsatz sprechen, war das erste Auftreten dieser Technologien vor etwa 20 Jahren mit den Wimax-Netzen. Aus verschiedenen Gründen entwickelten sich die Wimax-Netze nicht wie erwartet, sodass dieser Ansatz auf spezifische Anwendungen beschränkt blieb. Erst mit 5G sehen wir den umfangreichen Einsatz dieser Lösungen.
MIMO (Multiple Input, Multiple Output) nutzt die räumliche Diversität und die Technik der räumlichen Multiplexierung, um über verschiedene Antennen unabhängig und separat codierte Datensignale, so genannte „Streams“, unter Wiederverwendung der gleichen Zeitperiode und Frequenzressource zu übertragen.
Bei Multi-User MIMO (MU-MIMO) sendet der Sender unterschiedliche Streams an verschiedene Benutzer gleichzeitig, wobei dieselben Zeit- und Frequenzressourcen genutzt werden. Die spektrale Effizienz und Kapazität kann durch Hinzufügen zusätzlicher Antennen zur Unterstützung weiterer Streams verbessert werden. Der MIMO-Effekt beruht darauf, dass ein Funksignal zwischen Sender und Empfänger von seiner Umgebung beeinflusst wird, wobei Reflexionen von Gebäuden und anderen Hindernissen zu mehreren Signalpfaden führen. Die verschiedenen reflektierten Signale erreichen die Empfangsantenne mit unterschiedlichen Zeitverzögerungen, Dämpfungen und Ausbreitungsrichtungen.
Wenn mehrere Empfangsantennen eingesetzt werden, empfängt jede Antenne eine leicht unterschiedliche Version des Signals, die mathematisch kombiniert werden können, um die Qualität des übertragenen Signals zu verbessern. Wir sprechen von „räumlicher Diversität“. Dies wird auch erreicht, indem das Funksignal über mehrere Antennen gesendet wird, wobei jede Antenne modifizierte Versionen des Signals sendet.
Neben der räumlichen Diversität erhöht die räumliche Multiplexierung die Kapazität der Funkverbindung, indem die mehreren Übertragungspfade als zusätzliche Kanäle zur Datenübertragung genutzt werden. Die räumliche Multiplexierung ermöglicht es, mehrere einzigartige Datenströme zwischen Sender und Empfänger zu senden, was den Durchsatz erheblich erhöht und es auch ermöglicht, mehrere Netzwerknutzer von einem einzigen Sender zu unterstützen.
Wie im **Bild 3** zu sehen, verwendet Beamforming fortschrittliche Antennentechnologien, um ein Funksignal in eine bestimmte Richtung zu fokussieren, anstatt es in einem weiten Bereich zu senden.

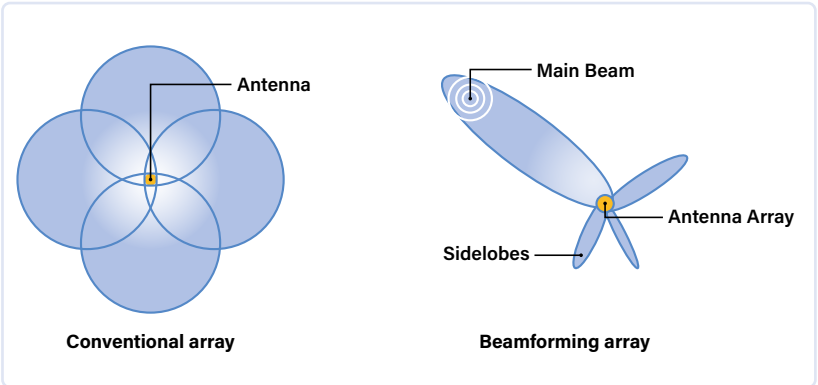


Bild 4. Richtcharakteristiken eines Standard-Dipol-Antennenarrays (links) und eines 5G-Arrays der neuesten Generation (rechts) mit Beamforming-Fähigkeit.

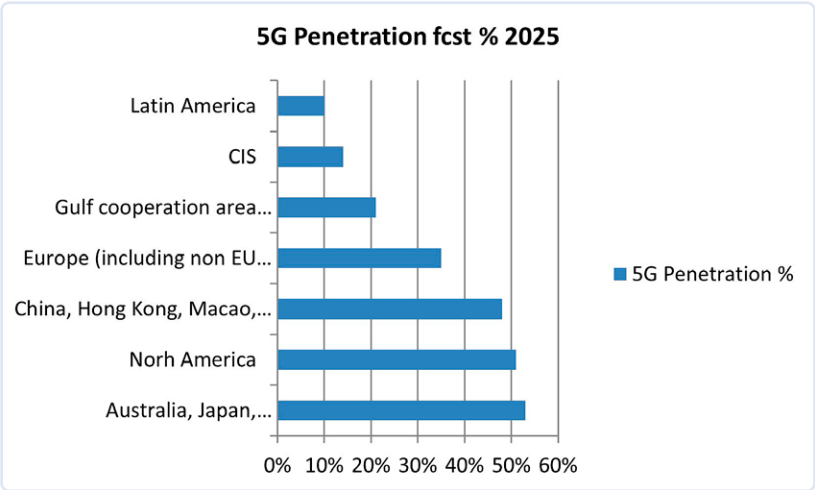
Roberto: Könntest du das Funktionsprinzip näher erläutern?
Nemo: Im **Bild 4** sehen Sie die Strahlungsmuster einer Standard-Dipolarray-Antenne (links) und einem 5G-Array der neuesten Generation (rechts) mit Beamforming-Fähigkeiten. Die Beamforming-Technologie funktioniert durch Fokussierung des Funksignals in eine bestimmte Richtung, anstatt es in einem weiten Bereich zu senden. Diese Technik reduziert die Interferenzen zwischen Signalen, die in verschiedene Richtungen gerichtet sind, und ermöglicht den Einsatz größerer Antennenarrays, die daher mit der Massiven MIMO-Technologie verbunden sind. 3D-Beamforming, das durch die große Anzahl von Antennen in einem Massiven MIMO-System erreicht wird, erzeugt sowohl horizontale als auch vertikale Strahlen auf die Benutzer zu, um die Datenraten bei Bedarf zu erhöhen und „folgt" der Anfrage von einem mobilen Nutzer. Komplexe Algorithmen wurden entwickelt, um die räumlichen Informationen, die aus einem Kanalzustandsinformationen-Referenzsignal (CSI-RS) gewonnen werden, zu koordinieren, um die Basisstation zu befähigen, mit mehreren Geräten unabhängig und gleichzeitig zu kommunizieren. CSI-RS ist ein Signal, das von der Basisstation an das Benutzerelement (UE) gesendet wird, um dem UE zu ermöglichen, die Kanalzustandsinformationen (CSI) zu berechnen und sie an die Basisstation zurückzumelden. Diese Informationen werden vom MIMO-System verwendet, um eine erhebliche Signalverarbeitung durchzuführen, wobei das CSI gebraucht wird, um die Kanalübertragungsfunktion in Matrixform darzustellen.

Roberto: Wie steht Europa im Vergleich zu anderen Ländern beim 5G-Ausbau?
Nemo: Ab 2016 hat die EU einen allgemeinen 5G-Ausbauplan festgelegt, der folgende Meilensteine vorsieht:

- Erste Testnetze bis Ende 2018.
- Vollständige kommerzielle 5G-Dienste in mindestens einer großen Stadt bis Ende 2020.
- Vollständige 5G-Abdeckung in städtischen Gebieten und kontinuierliche Abdeckung auf den Hauptverkehrsrouten (Straßen, Eisenbahnen) bis Ende 2025.
- Im März 2021 wurde ein weiterer Meilenstein gesetzt: 5G-Abdeckung aller bewohnten Gebiete bis Ende 2030.

Bis Ende 2020 haben 23 EU-Länder das Ziel erreicht, dass mindestens eine große Stadt vollständig von 5G abgedeckt ist. Nur vier Länder konnten dieses Ziel nicht erreichen.
Laut den neuesten Umfragen geht die EU-Kommission davon aus, dass bis Ende 2025 nur elf Länder die volle 5G-Abdeckung aller städtischen Gebiete und Hauptverkehrsrouten erreichen werden. Mit anderen Worten, der 5G-Ausbau in Europa erfolgt langsam, und die Liste der 13 Länder, die hinterherhinken werden, umfasst auch einige unerwartete Namen (es liegt an Ihnen, die „Überraschungen" zu identifizieren): Österreich, Tschechien, Estland, Deutschland, Irland, Polen, Litauen, Slowenien, Belgien, Bulgarien, Kroatien, Zypern und Griechenland.
Gemäß der Prognose der GSM Association wird der Prozentsatz der 5G-Mobilverbindungen gegenüber allen Verbindungen bis Ende 2025 so sein wie im **Bild 5** dargestellt.
Gut? Definitiv nicht, würde ich sagen. Europa wird mickrige 35% erreichen. Außerdem ist es wichtig zu beachten, dass die Definition von „5G-Abdeckung" in Bezug auf die Anzahl der 5G-Verbindungen ein generisches Konzept ist, da 5G in mehreren Phasen (oder Schritten) entlang einer Zeitachse implementiert wird. Daher bedeutet „Anzahl der 5G-Verbindungen" nicht unbedingt, dass die 5G-Dienste tatsächlich auf einer 5G-Verbindung verfügbar sind und genutzt werden.

Bild 5. Globale Prognose zur 5G-Abdeckung bis 2025. (Quelle: GSMA)



Roberto: Was sind diese 5G-Implementierungsschritte?
Nemo: Sie werden von 3GPP entworfen, dem wichtigsten weltweiten Standardisierungsorgan für Mobilfunktechnologien. Es vereint sieben Telekommunikations-Standardisierungsorganisationen — ARIB, ATIS, CCSA, ETSI, TSDSI, TTA und TTC — und stellt die Spezifikationen für zellulare Telekommunikationstechnologien auf, einschließlich Funkzugang, Kernnetz und Dienstfähigkeiten.
3GPP hat den 5G-Ausbau in Phasen eingeteilt, die nebeneinander existieren können und die die „endgültige" 5G-Implementierung in einer Zeit einführen werden, die von der Geschwindigkeit der Infrastrukturinvestitionen abhängt. Heute befindet sich jeder Betreiber in jedem Land auf einem Zwischenschritt

auf diesem Weg. Leider muss ich sagen, dass Europa im Vergleich zu anderen Ländern wie den USA, China und Südkorea, die weit fortgeschritten sind mit dem 5G-Ausbau, im Rückstand ist.
Zu den 5G-Evolutionsphasen 15, 16, 17 und 18, deren Standardisierung von 3GPP abgeschlossen wurde, sollte Europa bis Ende 2023 mit der Einführung von Phase 18 begonnen haben (siehe **Tabelle 2**).

Roberto: Wo steht 5G also jetzt?
Nemo: In den meisten Fällen, abgesehen von spezifischen Testzonen, befindet sich 5G zwischen Phase 15 und Phase 16.
Der erste Schritt von 5G wird 5G NR (New Radio) genannt. Dies ist tatsächlich ein komplexes Upgrade, aber nur der erste Schritt. Vereinfacht ausgedrückt bezieht sich 5G NR auf den Funkzugang und alle damit verbundenen Protokolle und Steuerungsschnittstellen. Wie im **Bild 6** gezeigt, verbindet sich das 5G-Benutzerelement über die NR-Uu-Schnittstelle mit dem 5G-Funkzugangsnetzwerk. Das 5G-RAN (Radio Access Network), das von gNBs (5G Node B, der Basisbandeinheit) gebildet wird, wurde so konzipiert, dass es seine eigenen Kernsteuerungsfunktionen hat, aber nur wenige Netzwerke implementieren diese Steuerungsfunktion in nativer Weise.
Da die vollständige Bereitstellung einer 5G-Kernsteuerungsfunktion Zeit und massive Investitionen erfordern wird, implementieren die meisten Mobilfunkbetreiber die erste Phase der 5G-Steuerungsfunktion auf der 4G-Kernsteuerungsfunktion. Wir sprechen von nicht eigenständiger 5G-NSA (Non StandAlone), um es von zukünftiger eigenständiger 5G-SA (StandAlone) zu unterscheiden. Im Fall von 5G-NSA wird die Benutzererfahrung nur von einer 15…50% Verbesserung der Datenrate und einer reduzierten Latenz profitieren, immer noch weit entfernt von den versprochenen dramatischen Verbesserungen, die mit dem endgültigen 5G-SA erreichbar sind.
Aber: Es gibt ein „aber". Auch für 5G-NSA benötigen Sie eine Voraussetzung, nämlich eine vollständig effiziente 4G-Netzabdeckung. Einige Länder sind jedoch weit davon entfernt, selbst eine vollständige 4G-Abdeckung in ländlichen Gebieten oder entlang der Hauptkommunikationswege innerhalb dieser Länder (Autobahnen, Eisenbahnen) bereitzustellen.

Roberto: Aus deinen Worten entnehme ich, dass der Übergang nicht so reibungslos verläuft. Gibt es vielleicht eine übermäßige Vorsicht seitens der Anbieter, die Verzögerungen bei der Implementierung von 5G verursacht?
Nemo: 5G ist sehr ambitioniert und erfordert enorme

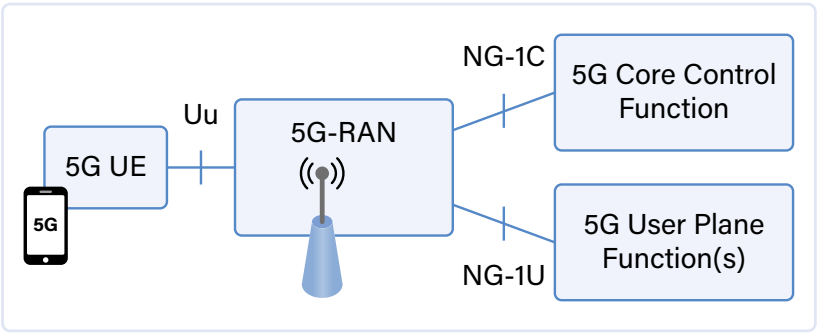


Bild 6. Das 5G-UE (User Element), das mit dem 5G-RAN (Radio Access Network) mit seinen eigenen Kernfunktionen (-1C) und Userfunktionen (-1U) interagiert.

Tabelle 2: 5G-Implementierungsphasen (Releases)

Phase (Release)	Jahr der Einführung oder Definition	Zweck	Status (März 2024)
15	2018	Behandlung der Konzepte für den Zugang zum neuen Funk (New Radio, NR)	Abgeschlossen
16	2020	Vervollständigung von NR, einschließlich unlizenzierter Frequenzen und Satellitenzugang	Abgeschlossen
17	2022	Definition zusätzlicher Dienste wie erweitertes MIMO, verbesserte gemeinsame Nutzung des Spektrums, verbesserte Abdeckung, Einbeziehung von Frequenzbändern bis 71 GHz, verbesserte Unterstützung privater Netze, industrielles IoT, NR-Geräte mit geringer Komplexität, Edge Computing, Steuerung des Zugangsverkehrs, Unterstützung von Switching und Splitting, Netzautomatisierung für 5G, Network Slicing, fortgeschrittener V2X-Dienst, Unterstützung mehrerer USIM und andere Dienste	Nicht abgeschlossen. Die Aktivierung ist noch im Gang und wird durch mehrere Verzögerungen beeinträchtigt
18	2023	Auch definiert als Advanced 5G. Die neuen Spezifikationen umfassen Edge Computing, intelligente Energie und Infrastruktur, fahrzeugmontierte Relais, Weiterentwicklung des IMS-Multimedia-Telefondienstes, hochpräzise Positionierung mit geringem Stromverbrauch für industrielle IoT-Geräte, Verbesserung des Network Slicing	Definition der Spezifikationen abgeschlossen. Muss noch implementiert werden
19	2024	3GPP arbeitet derzeit an der Definition der Funktionen dieser Advanced-5G-Version	Weder vollständig definiert noch umgesetzt

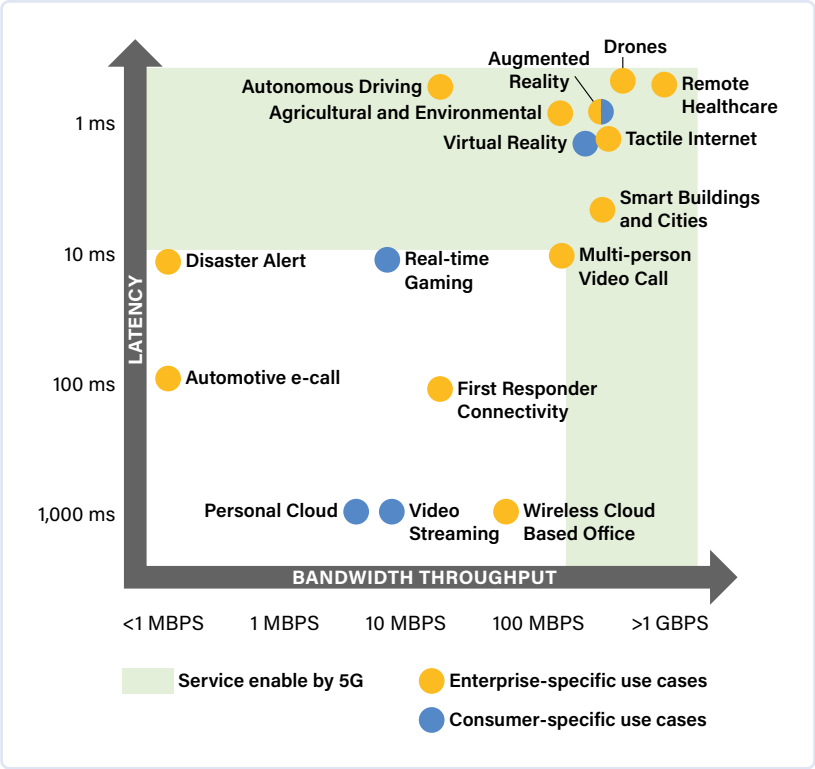


Bild 7. Anwendungsgebiete von 4G (weiß) und 5G (hellgrün) basierend auf der Kombination von Latenz und Durchsatz der Verbindung. (Quelle: GSMA)

Investitionen. Für private Unternehmen sind diese Investitionen direkt mit den erwarteten Einnahmen verbunden. Und hier liegt das Problem, zumindest in Europa: Die Kapitalrentabilität (Return on Investment, ROI) ist nicht zufriedenstellend, da in mehreren Ländern die Menschen mit der Leistung von 4G zufrieden und nicht bereit sind, mehr auszugeben, um die wenigen Prozentpunkte an Serviceverbesserung von 5G-NSA zu erhalten. Andererseits werden die erwarteten Einnahmen hauptsächlich von Unternehmen und neuen Sektoren kommen, die Dienste benötigen, die nur mit 5G-SA entwickelt werden.

Das Diagramm im **Bild 7**, das aus einer Analyse der GSMA-Gruppe abgeleitet wurde, zeigt, welche echten Erwartungen 5G hinsichtlich der „Dienste, die ermöglicht werden" hat. Der weiße Bereich ist das, was wir mit einem effizienten 4G-Netzwerk erreichen können. Der hellgrüne Bereich bezieht sich auf die Dienste, die 5G ermöglichen wird.

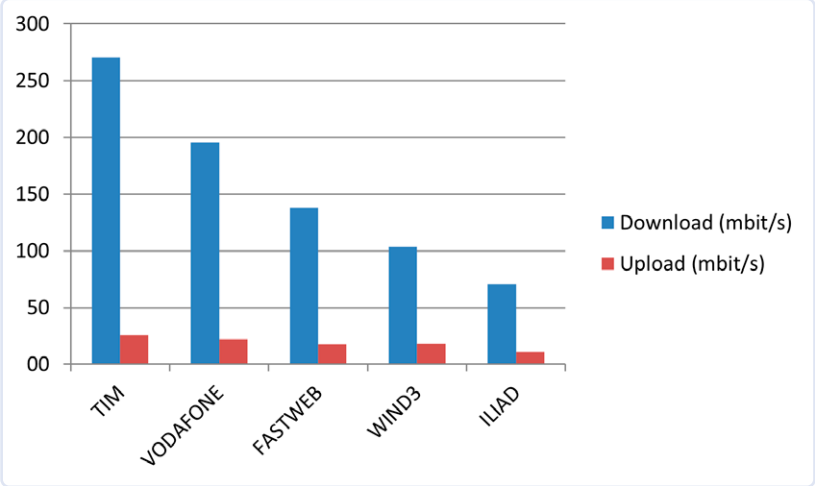


Bild 8. Durchschnittliche Download- und Upload-Geschwindigkeit von 5G-Netzwerken in Italien nach Netzwerkbetreibern, Stand März 2023. (Quelle: Opensignal, Mai 2023)

Wir leben heute mit dem Paradox von Henne und Ei: Massive Einnahmen kommen aus verbraucherspezifischen Anwendungsfällen, und wir können sehen, dass es derzeit keine massive Nachfrage nach spezifischen 5G-Diensten von diesem Publikum gibt. Gleichzeitig führt ein Preiskampf in mehreren europäischen Ländern (hauptsächlich Frankreich und Italien) zu einem Rückgang des ARPU (durchschnittlicher Umsatz pro Benutzer). Die Folge ist, dass normale Verbraucher nicht bereit sind, einen Premiumbetrag für eine Verbesserung von Funktionen zu zahlen, die sie nicht wirklich benötigen. Wenn wir die typischen Bedürfnisse von Unternehmen (oder Regierungen) betrachten, sprechen wir von Diensten, die noch nicht von 5G-NSA implementiert sind und die hauptsächlich mit 5G-SA verfügbar sein werden. Dies erfordert weitere Entwicklung und Investitionen, die aus den aktuellen Umsatzniveaus von mehreren Netzwerkbetreibern schwer zu finanzieren sind.

Nun, um zum ursprünglichen Thema der Unterschiede zwischen 4G und 5G zurückzukehren, würde ich die aktuelle Situation heute mindestens als unklar definieren. 4G-Netzwerke haben weltweit sehr unterschiedliche Leistungen, während die Leistungen (Datenrate pro Stunde) von 5G-Netzwerken, die größtenteils noch auf dem 4G-Kern basieren, mit zunehmender Nutzerzahl abnehmen und enorme Unterschiede weltweit aufweisen: von 500 Mbit/s in Korea über 300 Mbit/s in China bis zum derzeitigen Durchschnitt von rund 200 Mbit/s in Italien, aber mit erheblichen Unterschieden zwischen den Betreibern, wie in der Grafik im **Bild 8** gezeigt.

Generell kann 5G drei bis fünfmal schneller als 4G sein oder zweimal so schnell wie 4.5G in der Trägeraggregation, aber bei dem Versuch, einen effektiven Vergleich anzustellen, ist sowohl die Referenzgeschwindigkeit von 4G als auch die von 5G ein Problem. Die Vorteile von 5G werden tatsächlich durch die Kombination der drei Faktoren gemessen: Geschwindigkeit, Latenz und Netzwerkkapazität. Die separate Betrachtung der Faktoren vermittelt kein Bild davon, welchen Fortschritt es im Vergleich zu 4G darstellt.

Ein weiteres Beispiel für den Stand der 5G-Abdeckung, Stand März 2024, ist in der 5G-Abdeckungskarte von Vodafone Deutschland zwischen Hannover, Hamburg und Berlin (violett steht für 5G, orange für 4G und rot für 4.5G) in **Bild 9** dargestellt, was zeigt, dass die Ziele noch weit entfernt sind.

Roberto: Sollte ich also sagen: „Lasst uns vorerst die Bereitstellung von 6G vergessen?"

Nemo: Nicht wirklich. Vielleicht sieht das Bild, das ich gemalt habe, zu pessimistisch aus, aber ich glaube, dass wir nur eine vorübergehende Verlangsamung beim Ausbau von 5G erleben. 5G bietet prinzipiell fantastische Funktionen und Dienste. Was müssen

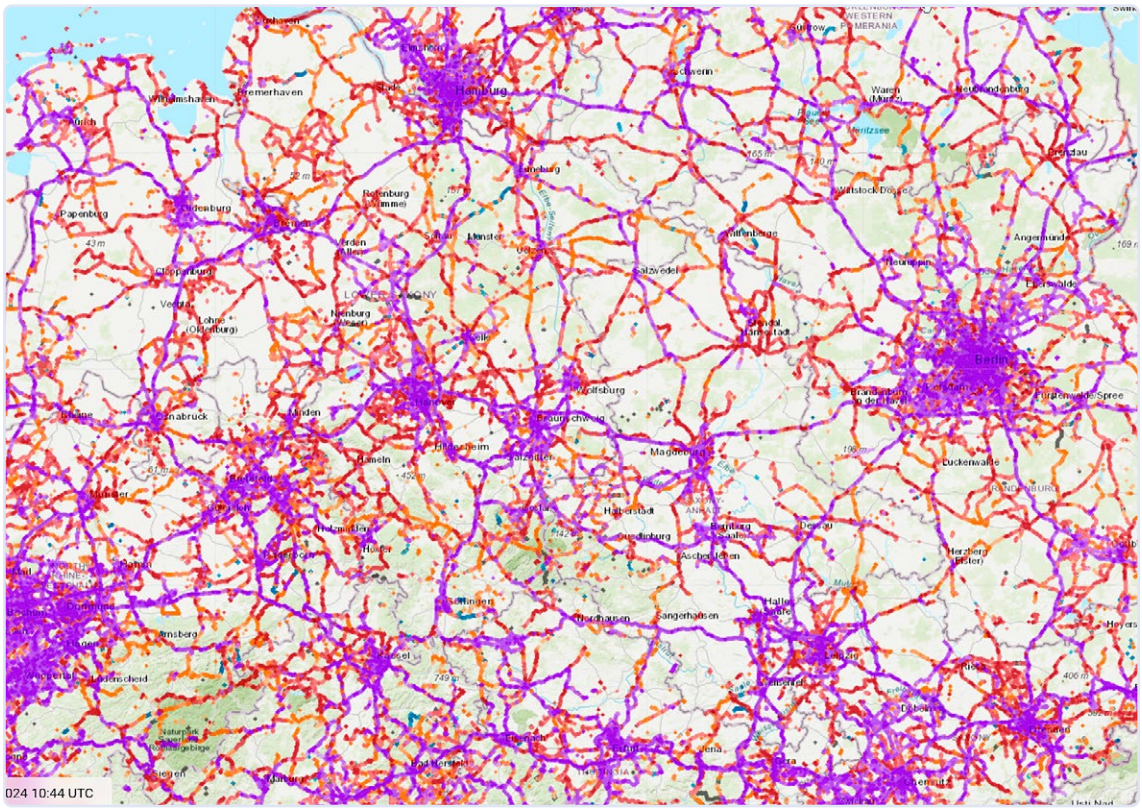


Bild 9. 4G- und 5G-Netzwerkbereitstellung von Vodafone im Bereich Hannover–Berlin–Hamburg (Quelle: NPERF, nperf.com/en/map/DE/)

jetzt auf das unvermeidliche Wachstum der Nachfrage nach solchen Diensten warten, um die notwendige Auslastung zu haben, die es den 5G-Netzen ermöglicht, wieder profitabel zu werden. Die Nachfrage nach neuen 5G-Diensten ist bisher nicht wie erwartet vorangeschritten, entwickelt sich jedoch und wird bald den erforderlichen Druck ausüben, damit 5G seinen Wettlauf wieder aufnehmen kann.

In der Zwischenzeit hat die 3GPP im Dezember 2023 angekündigt, dass sie mit der Ausarbeitung der Spezifikationen für die sechste Generation (6G) beginnen wird, mit dem Ziel, die ersten Tests vor 2030 zu starten. Es ist deshalb zwar noch zu früh, um die genaue Leistungsfähigkeit und Architektur von 6G vorherzusagen, aber wir können dennoch voraussehen, dass wir in der Entwicklung einer Art von 5.5G unterstützt werden, indem einige der Funktionen, die Teil von 6G sein sollen, vorweggenommen und übernommen werden, indem sie als Industriestandard angenommen werden.

Zum Beispiel werden mehrere Ankündigungen im Zusammenhang mit den 5G-Netzwerken wie die Unterstützung von vollautomatischen Fahrsystemen wahrscheinlich nur dann einen realen massiven Schub erleben, wenn bessere Geschwindigkeit, flächendeckende Abdeckung und Integration verschiedener Netzwerkbaukomponenten, die konzeptionell Teil der 6G-Netzwerke sein werden, verfügbar sind.

Stand heute sind die Hauptziele von 6G ein weiterer Sprung bei der Datenrate von bis zu 1 Tbit/s, eine weitere Reduktion der Latenz auf die Größenordnung von 100 µs, die vollständige Abdeckung des Territoriums durch die Integration von zellularen und nicht-zellularen Netzwerken einschließlich Konstellationen von Telekommunikationssatelliten, neue Arten von Hotspots und neue Geräte mit Unterstützung von KI-Systemen zur Optimierung aller Netzwerkressourcen, die ein echtes personalisiertes Serviceangebot bieten und — dies ist eine weitere wichtige Anforderung — die elektromagnetische Verschmutzung und den Energieverbrauch reduzieren. Wenn von 5G erwartet wird, dass es die Explosion des IoT unterstützt, wird 6G Künstliche Intelligenz in das Netzwerk integrieren und eine engere Interaktion zwischen Realität und ihrer Erweiterung im Metaverse ermöglichen.

SE — 240233-02

Über Nemo Galletti

Nemo Galletti ist der Standortleiter von RFS Italia, der italienischen Niederlassung von Radio Frequency Systems GmbH, einem multinationalen Unternehmen, das zum Zeitpunkt dieses Interviews von Nokia kontrolliert wird. Er studierte Elektrotechnik am Politecnico di Milano und sammelte 39 Jahre lang Erfahrungen im Telekommunikationsbereich, zunächst in der Forschung und Entwicklung des Alcatel-Konzerns, dann im kommerziellen Sektor in Übersee, wo er Telekommunikationsnetze verkaufte, bevor er bei RFS landete.

Über Roberto Armani

Roberto Armani ist Elektronikingenieur. Nach seinem Studium am Politecnico di Milano hat er über 35 Jahre Erfahrungen in verschiedenen Bereichen gesammelt. Bevor er sich dem Elektor-Team als leitender Redakteur anschloss, arbeitete er in der Computerindustrie, der elektronischen Bildverarbeitung, der Telekommunikation, der Materialprüfung und dem Web-Publishing. Neben der Elektronik hört (und singt) er gerne klassische Musik und unternimmt Höhenwanderungen in den Bergen.

elektorstore

www.elektor.de

Der Elektor Store

Nie teuer, immer überraschend!

Der Elektor Store hat sich vom Community-Store für Elektor-eigene Produkte wie Bücher, Zeitschriften, Bausätze und Module zu einem umfassenden Webshop entwickelt, der einen großen Wert auf überraschende Elektronik legt.

Wir bieten die Produkte an, von denen wir selbst begeistert sind oder die wir einfach ausprobieren wollen. Wenn Sie einen Produktvorschlag haben, sind wir hier erreichbar (sale@elektor.de).

Andonstar AD210 10,1” Digital-Mikroskop

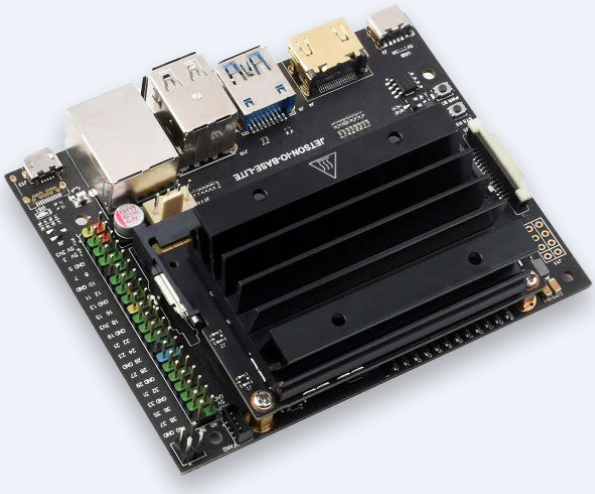
Das Andonstar AD210 ist ein Digital-Mikroskop mit einem großen 10,1” IPS-Display, das einen Betrachtungswinkel von 178° bietet und 1080P-Video- und 12-MP-Fotoaufnahmen unterstützt. Mit 260-facher Vergrößerung ermöglicht das Mikroskop einen klaren Blick auf die Seiten von Bauteilen auf Leiterplatten. Im Lieferumfang des Mikroskops sind außerdem eine 32-GB-Speicherkarte und eine Fernbedienung enthalten.

Preis: ~~179,95 €~~

Sonderpreis: 129,95 €

www.elektor.de/20802

Waveshare Jetson Nano Development Kit Lite



Das Waveshare Jetson Nano Developer Kit basiert auf den KI-Computern Jetson Nano (mit 16 GB eMMC) und Jetson Xavier NX. Es bietet nahezu die gleichen I/O-Anschlüsse, die gleiche Größe und Höhe wie das Jetson Nano Developer Kit (B01), wodurch ein Upgrade des Kernmoduls besonders einfach wird.

Preis: 269,00 €

Mitgliederpreis: 242,10 €

www.elektor.de/20761

UNI-T UT512D Isolations-widerstandstester (2,5 kV)

Preis: 289,00 €

Mitgliederpreis: 260,10 €

www.elektor.de/20786

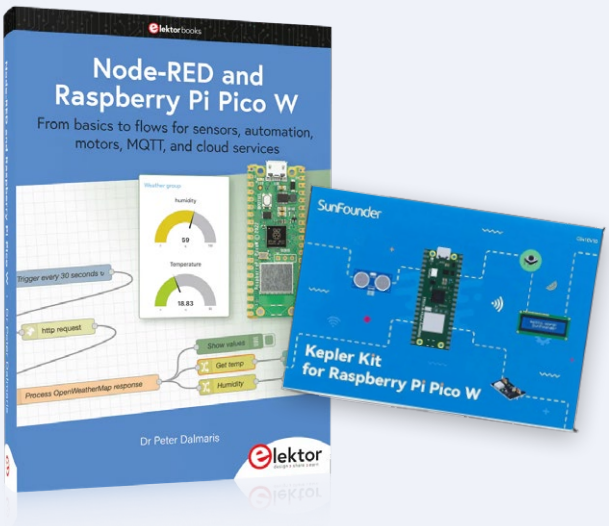
CrowVision 11,6” IPS kapazitives Touch-Display (1366 x 768)

Preis: 139,95 €

Mitgliederpreis: 125,96 €

www.elektor.de/20792

Node-Red Development Bundle



Preis: ~~104,95 €~~

Sonderpreis: 84,95 €

www.elektor.de/20849

Raspberry Pi 5 RTL-SDR V4 (Bundle)

Preis: ~~94,95 €~~

Sonderpreis: 79,95 €

www.elektor.de/20878

Aller Anfang...

muss nicht schwer sein: Vom Spannungsfolger zum Instrumentenverstärker

Von Eric Bogers (Elektor)

In dieser Folge des Basiskurses beschäftigen wir uns mit weiteren wichtigen Operationsverstärker-Schaltungen, darunter den Differenzverstärker und den Instrumentenverstärker. Außerdem werden wir uns mit einem Thema befassen, das vor allem bei Audioanwendungen von größter Bedeutung ist: symmetrische Verbindungen.

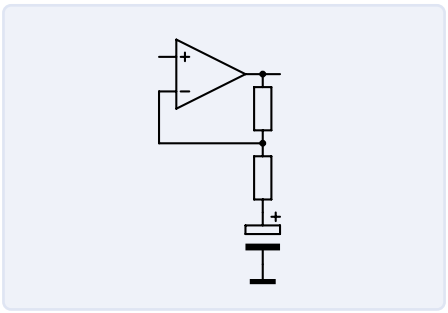


Bild 3. Verringerung der Gleichstromverstärkung.

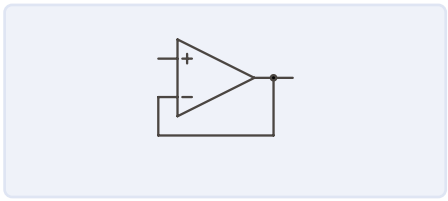


Bild 1. Der Spannungsfolger.

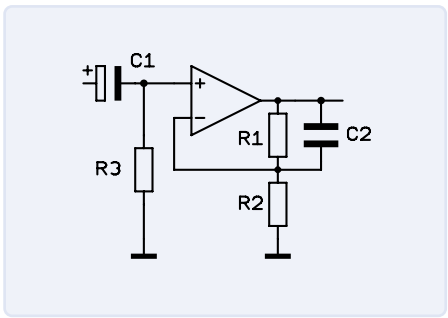


Bild 2. Begrenzung der Bandbreite.

Der Spannungsfolger

Eine häufige Anwendung des nicht-invertierenden Verstärkers ist der Spannungsfolger (**Bild 1**). Wenn wir für R1 eine Null und für R2 unendlich in die Formel für die Verstärkung einsetzen würden, wäre das Ergebnis genau 1. Wie der vor einiger Zeit besprochene Emitterfolger hat auch der Spannungsfolger einen hohen Eingangswiderstand und einen niedrigen Ausgangswiderstand - er übertrifft den Emitterfolger in dieser Hinsicht um mehrere Zehnerpotenzen. Der Spannungsfolger wird deshalb auch als Impedanzwandler bezeichnet.

Bandbreitenbegrenzung

Beim nicht-invertierenden Verstärker muss die Bandbreite begrenzt werden, und auch hier werden zu diesem Zweck Kondensatoren verwendet (**Bild 2**). Im Gegensatz zum invertierenden Verstärker kann der Eingangswiderstand unabhängig von der Verstärkung eingestellt werden, was die Verwendung kleinerer Kondensatoren ermöglicht. Ansonsten sind die Kondensatoren genauso dimensioniert wie beim invertierenden Verstärker.
Eine häufig anzutreffende Variante dieses Themas besteht darin, einen Kondensator in Reihe mit R2 zu schalten. In diesem Fall ist die Verstärkung für Gleichspannungen gleich 1, während die Wechselspannungsverstärkung (jenseits der Grenzfrequenz) wie üblich durch R1 und R2 bestimmt wird (**Bild 3**).

Der Summierverstärker

Eine Variation des invertierenden Verstärkers ist der Summen- oder Summierverstärker (**Bild 4**), der die Spannungen mehrerer Eingänge addiert.
Normalerweise werden gleiche Werte für die Eingangswiderstände verwendet, aber die Schaltung funktioniert auch mit ungleichen Eingangswiderständen - die einzelnen zu addierenden Spannungen müssen dann mit dem jeweiligen Faktor multipliziert werden. Summierverstärker werden häufig in

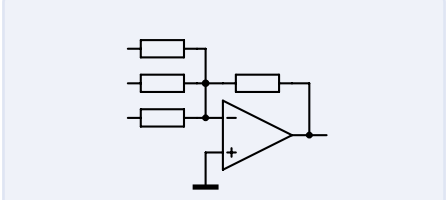


Bild 4. Der Summierverstärker.

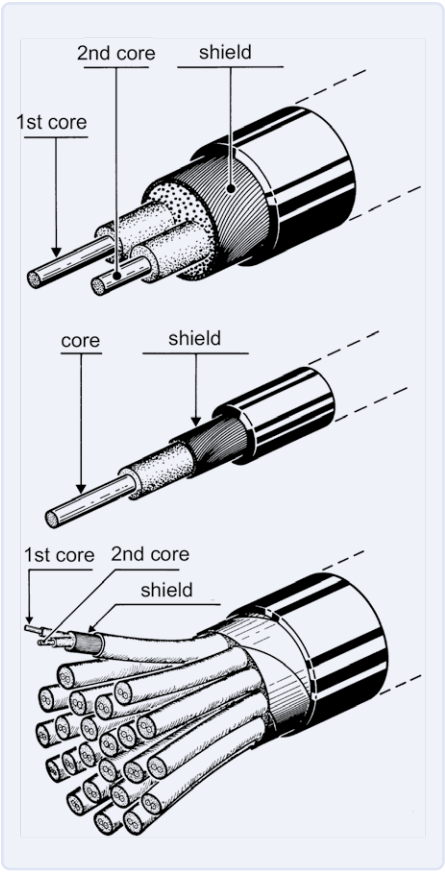


Bild 5. Symmetrische und asymmetrische Leitungen und ein mehradriges Kabel (unten).

Mischpulten eingesetzt, um die einzelnen Kanäle zu summieren. Da der Knotenpunkt (Summierpunkt) am invertierenden Eingang eine virtuelle Masse darstellt, können die einzelnen Kanäle frei gemischt werden, ohne dass es zu Störungen kommt. Die niedrige Impedanz des Summenpunktes macht die Verbindung des Summensignals zudem relativ unempfindlich gegenüber Störungen.
Natürlich liegt diese Verbindung nicht exakt auf Nullpotential, und die Impedanz ist auch nicht genau Null, so dass die Störungsunempfindlichkeit begrenzt ist. Um Störungen weiter zu minimieren, werden die einzelnen Signalleitungen vorzugsweise zwischen zwei Masseverbindungen gelegt.

Symmetrische Verbindungen

Es ist auch möglich, das Signal symmetrisch zu führen, aber das erfordert zwei Signalverbindungen. Um diese Schaltung zu untersuchen, müssen wir uns zunächst mit den Unterschieden zwischen symmetrischen und asymmetrischen Verbindungen befassen.
Wie wir zu Beginn dieser Artikelserie gesehen haben, benötigen wir eine Hin- und Rückleitung, damit ein elektrischer Strom fließen kann. In der Gleichstromtechnik sprechen wir von einer Plus- und Minusleitung, in der Wechselstromtechnik von Phase und Masse (oder Null).
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.
Lautsprecherkabel sind für das Auffangen von Störsignalen nicht besonders kritisch: Sie führen in der Regel hohe Signalpegel und sind mit der sehr niedrigen Impedanz des Lautsprechers abgeschlossen.
Signalführende Leitungen führen jedoch einen deutlich niedrigeren Pegel und sind mit viel höheren Impedanzen abgeschlossen, so dass für diesen Zweck ein abgeschirmtes Kabel verwendet werden muss (**Bild 5**). Die Abschirmung kann aus einer leitfähigen Folie oder einem Geflechtmantel bestehen. Dies wirkt wie ein Faradayscher Käfig - Störsignale können dann (zumindest theoretisch) nicht zu den signalführenden Adern innerhalb der Abschirmung vordringen.
Im einfachsten Fall besteht ein solches geschirmtes Kabel aus einem Innenleiter mit einer Abschirmung drum herum; der Innenleiter führt das Signal, während die Abschirmung mit der Masse verbunden ist. Bei nicht zu langen Kabeln mit hohem Signalpegel ist eine solche unsymmetrische Verbindung völlig ausreichend, so dass sie in der Praxis oft verwendet wird. Alle Kabel, die mit Cinch- oder RCA-Steckern oder Mono-Klinkensteckern ausgestattet sind, sind unsymmetrisch. Stereo-Klinkenstecker könnten zwar auch ein symmetrisches Monosignal übertragen, werden aber meist für ein asymmetrisches Stereosignal verwendet. In **Bild 5** sehen Sie in der Mitte ein typisches Beispiel für ein einadriges asymmetrisches Kabel.
Für lange Kabel und/oder bei sehr niedrigen Signalpegeln ist es besser, ein symmetrisches Kabel (oben in Bild 5) zu verwenden. Hier wird das Signal durch die eine Ader in der richtigen Phase und durch die andere in der entgegengesetzten Phase übertragen, wobei beide Adern von einer gemeinsamen Abschirmung umhüllt sind. Aber was ist der eigentliche Vorteil dieses „doppelten" Signaltransports?
Werfen Sie einen Blick auf **Bild 6a**. Dort ist eine asymmetrische Signalübertragung zwischen zwei Geräten zu sehen. Treten Störungen auf, so werden diese zwar durch die Abschirmung zur Erde abgeleitet, aber im wirklichen Leben dringt ein kleiner Teil der Störung zum Signalleiter vor. Für die folgende Diskussion ist es nicht relevant, ob dies daran liegt, dass die Abschirmung

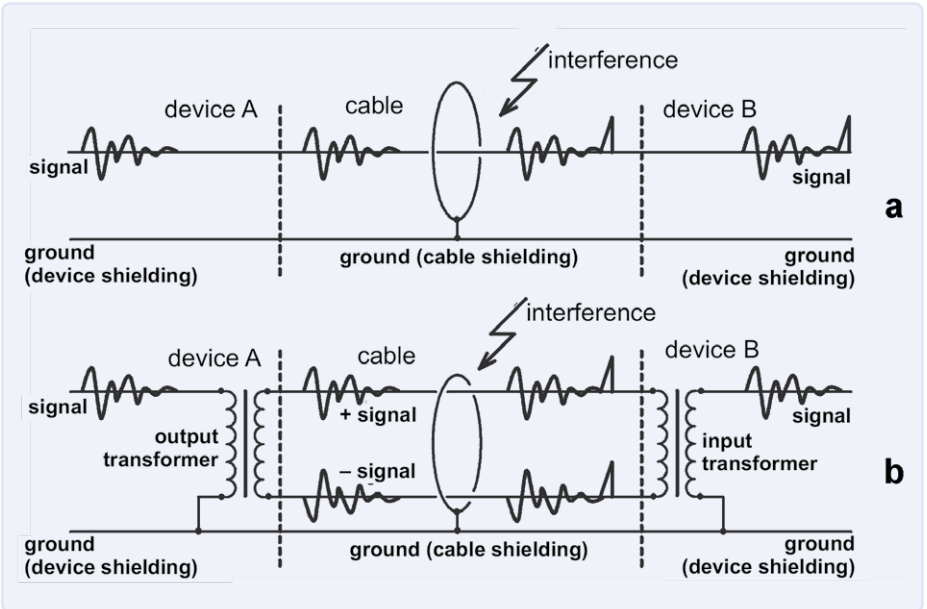


Bild 6. Asymmetrische (a) und symmetrische (b) Signalübertragung.

nicht perfekt ist, oder daran, dass die Störung durch den ohmschen Widerstand der Abschirmung eintritt. Wir gehen hier einfach davon aus, dass die Abschirmung nicht zu 100 % wirksam ist.

Bei einer symmetrischen Verbindung (**Bild 6b**) führt eine Ader das Signal in der richtigen Phase und die andere das gleiche Signal in der entgegengesetzten Phase. Ein Störsignal kann zwar in ähnlicher Weise zu den signalführenden Adern durchdringen, aber dieses Störsignal hat (im Gegensatz zum Nutzsignal) auf beiden Leitern die gleiche Phase. Beim Empfangsgerät werden die Signale auf beiden Adern voneinander subtrahiert (und so ein Differenzsignal gebildet), und da das Störsignal auf beiden Adern die gleiche Phase hat, wird es (zumindest theoretisch) vollständig ausgelöscht.

Sie fragen sich vielleicht, warum wir überhaupt noch eine Abschirmung benötigen, wenn bei einer symmetrischen Verbindung die Störsignale vollständig unterdrückt werden. Nun, erstens nehmen die beiden signalführenden Adern das Störsignal nie völlig identisch auf, selbst sie verdrillt sind. Zweitens ist die Bildung des Differenzsignals nie ganz perfekt. Differenzeingangsstufen, die mit 1%igen Widerständen ausgestattet sind, erreichen eine Gleichtaktunterdrückung (die Unterdrückung gleichphasiger Signale) von etwa 40 dB, professionelle Signalübertrager (je nach Frequenz) erreichen etwa 60 dB. In der Praxis ist das oft nicht ausreichend. Kombiniert man jedoch eine symmetrische Signalübertragung mit einer guten Abschirmung, so erhält man ein deutlich besseres Signal-Rausch-Verhältnis.

Der Differenzverstärker

Der Differenzverstärker (fälschlicherweise oft als Differentialverstärker bezeichnet) bildet die Differenz von zwei Eingangssignalen (**Bild 7**). Die Verstärkung wird wie beim invertierenden Verstärker berechnet, also für die Ausgangsspannung:

$$U_{out} = (U_{in+} - U_{in-}) \cdot \frac{R_2}{R_1}$$

Ein Problem des Differenzverstärkers ist die extrem ungleiche Eingangsimpedanz beider Eingänge. Dieses Problem kann umgangen werden, indem man die Werte der Widerstände nicht genau gleich wählt, sondern ihr Verhältnis gleich hält. In der Praxis wirkt sich dies jedoch wiederum nachteilig auf die oben beschriebene Gleichtaktunterdrückung aus: Wenn wir gleiche Widerstandswerte verwenden, stammen sie in vielen Fällen aus der gleichen Produktionsserie - das heißt, sie weichen zwar geringfügig vom aufgedruckten Wert ab, sind aber untereinander praktisch gleich groß.

Bei unterschiedlichen Widerstandswerten dürften die beiden Verhältnisse R2/R1 nie ganz gleich sein, was bedeutet, dass ein Offset-Signal mitverstärkt wird. Bei der Verwendung von 1%igen Metallfilmwiderständen ergibt sich in der Regel eine maximale Gleichtaktunterdrückung von 40 dB.

Der Instrumentenverstärker

Je höher die Verstärkung eines Differenzverstärkers ist, desto größer sind die Probleme mit ungleichen Eingangsimpedanzen. Bei Verstärkungen von 20 dB oder mehr - also dem 10-fachen oder mehr – sollte man sogenannte Instrumentenverstärker verwenden (**Bild 8**). Der erste Vorteil dieses Aufbaus besteht darin, dass die beiden Eingänge eine gleiche Eingangsimpedanz haben, die in der Regel durch einen Widerstand gegen Masse auf einen nutzbaren (niedrigeren) Wert eingestellt wird.

Bitte beachten Sie die besondere Beschaltung der Widerstände an den Eingangsverstärkern: Bei symmetrischen Signalen befindet sich in der Mitte von R4 ein virtueller Massepunkt, so dass in der Formel für die Verstärkung die Hälfte (!) des Wertes von R4 eingesetzt werden muss. Bei Gleichtaktsignalen hingegen kann R4 als nicht vorhanden betrachtet werden, so dass die Verstärkung der Gleichtaktsignalkomponenten gleich eins ist.

Die Zunahme der asymmetrischen Dämpfung liegt in der Größenordnung der Verstärkung der Eingangsstufe. Da es uns in erster Linie darum geht, die asymmetrische Dämpfung zu maximieren, werden wir in den meisten Fällen R1 und R2 den gleichen Wert zuweisen - sagen wir 10 kΩ - und die Eingangsstufe die Verstärkung übernehmen lassen. Wenn wir sie auf 20 dB einstellen, lässt sich für die gesamte Schaltung eine Gleichtaktdämpfung von 60 dB erreichen.

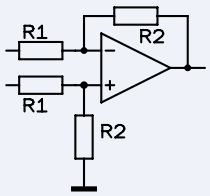


Bild 7. Der Differenzverstärker.

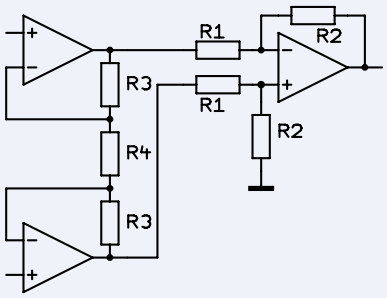


Bild 8. Der Instrumentenverstärker.

Wir belassen es vorerst dabei; die nächste Folge des Basiskurses wird viele weitere interessante und praktische Schaltungen mit Operationsverstärkern präsentieren.

RG — 240127-02

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

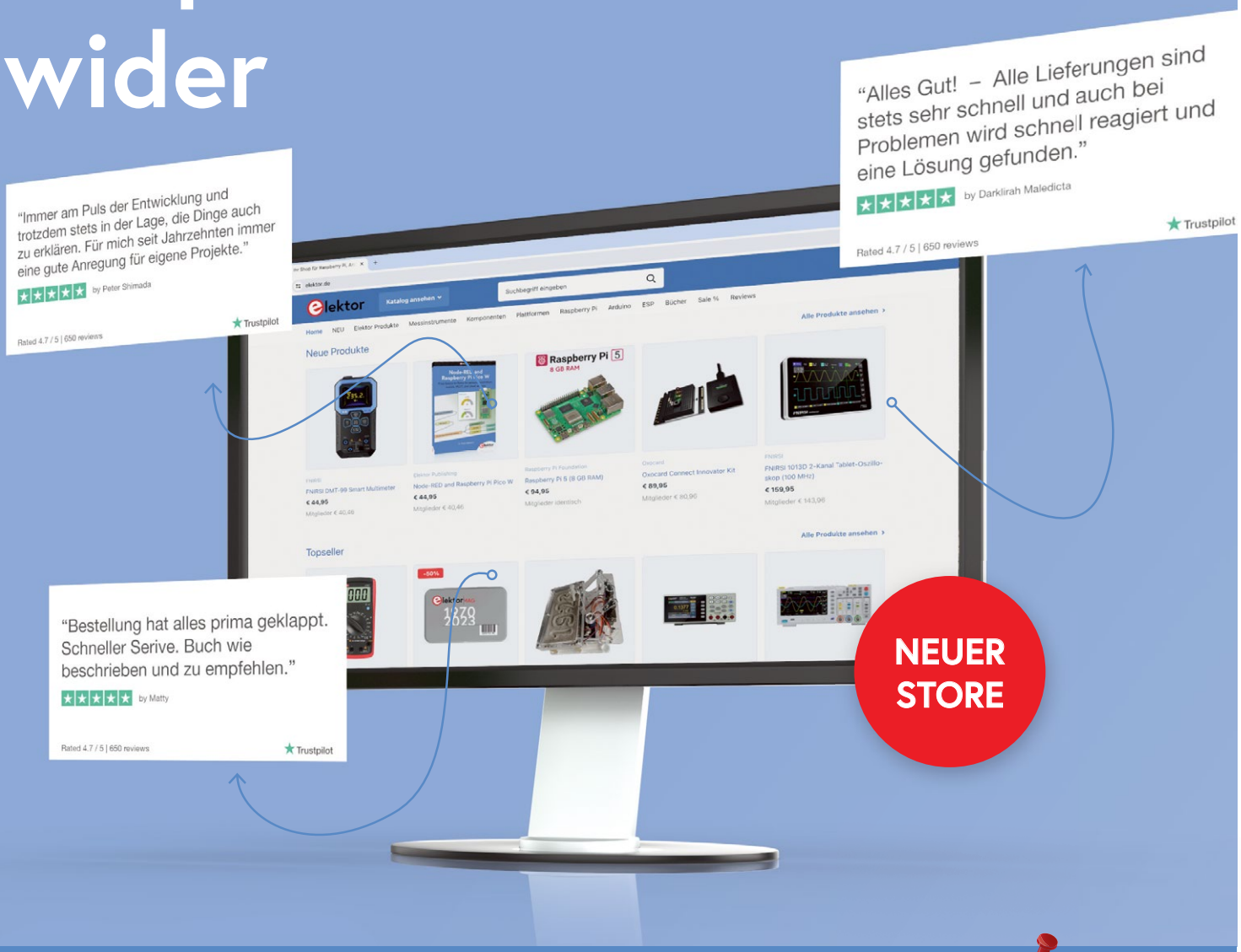
Haben Sie Fragen oder Kommentare?

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

Passendes Produkt

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

Jede Bewertung spiegelt ein persönliches Erlebnis wider



Wir lieben Elektronik und Projekte, und wir setzen alles daran, die Bedürfnisse unserer Kunden zu erfüllen

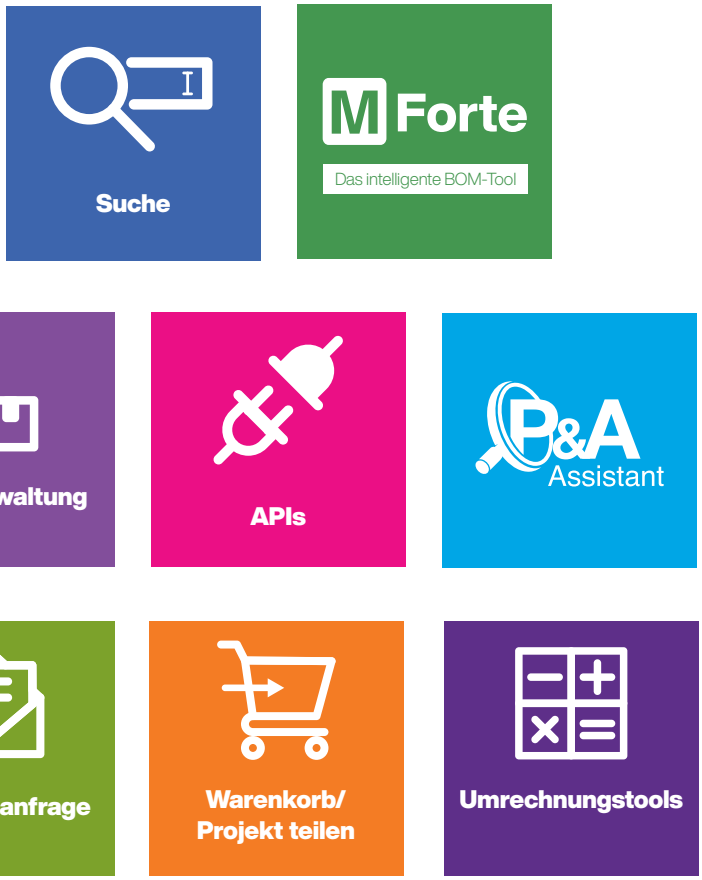
Der Elektor-Store: '**Never expensive, always surprising**'

"Toller Service und schnelle Lieferung – Sehr guter Service und sehr schnell geliefert. Top Unternehmen!"
by Lorandt Foelkel
Rated 4.7 / 5 | 1,049 reviews
Trustpilot

Sehen Sie sich weitere Bewertungen auf unserer Trustpilot-Seite an: www.elektor.com/TP/de

Oder bilden Sie sich selbst eine Meinung und besuchen Sie unseren Elektor Store, www.elektor.de

elektor
design > share > earn



Bestellen leicht gemacht

Tools für Suche, Bestandsabfrage und Einkauf

mouser.de/servicesandtools

MOUSER ELECTRONICS