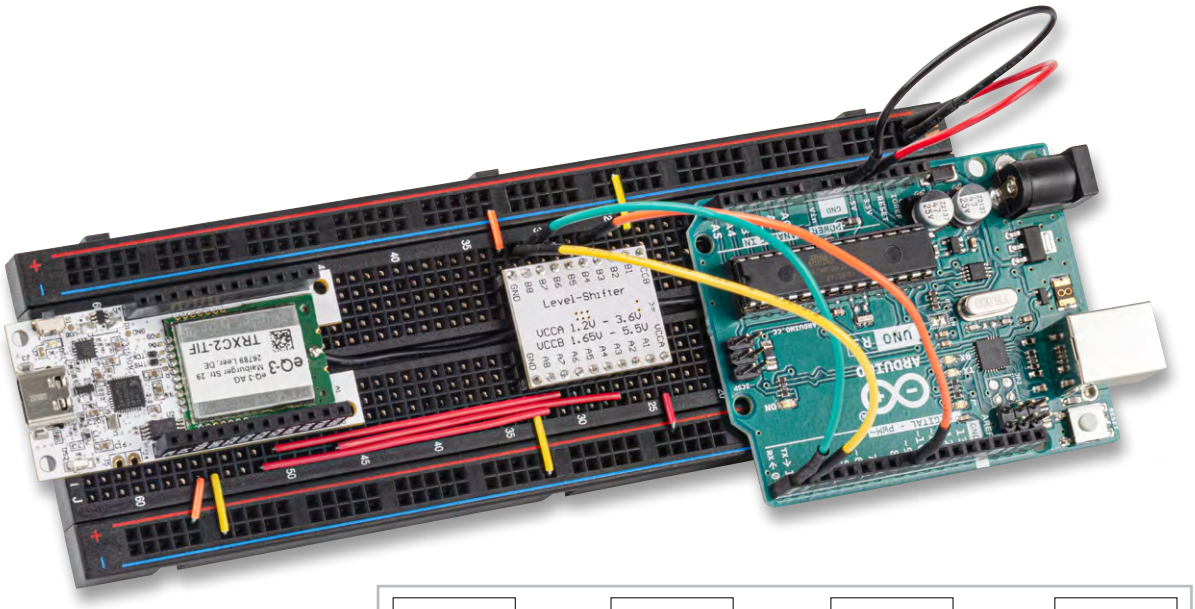
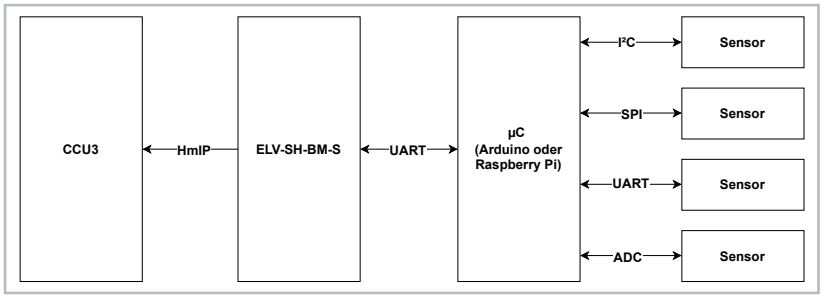
Die eigenen Messwerte, wo und wann Sie wollen

Mit der ELV Smart Home Sensor Base per UART eigene Sensoren in die Homematic IP-Welt bringen

Jeder Tüftler kennt das Problem: Sie haben einen interessanten Sensor in Ihrer Teilekiste liegen und können sich schon ausmalen, wie der Messwert dieses Sensors mit in die Smart-Home-Steuerung eingebunden werden könnte. Für Homematic IP Installationen gibt es dafür jetzt eine einfache Lösung: eine neue Coprozessor-Firmware für die ELV Smart Home Sensor Base, die mithilfe der UART-Schnittstelle Daten von Arduino- sowie Rasberry Pi-kompatiblen Sensoren in das Homematic IP System überträgt. Doch was genau ist UART? Wie funktioniert die serielle Kommunikation, und worauf muss ich achten, wenn ich ein anderes Gerät an das ELV-SH-BM-S anschließe? In diesem Artikel beantworten wir diese und weitere Fragen.





Mithilfe der neuen Firmware kann über einen Mikrocontroller als Zwischenebene auf die Messwerte von beliebigen Sensoren zugegriffen werden.

Was ist UART?

UART steht für Universal Asynchronous Receiver/Transmitter (universeller asynchroner Empfänger/Sender) und ist der Name eines Bauteils oder eines Peripherieelements innerhalb eines Geräts, das Signale in einer bestimmten Form über zwei Signalleitungen (Rx für Receiver, also Empfangsleitung, und Tx für Transmitter, also Sendeleitung) senden und empfangen kann.

Damit beide Partner kommunizieren können, müssen beide Geräte die gleiche Baudrate haben. Diese entspricht der Anzahl der Symbole, die pro Sekunde übertragen werden. Im Falle des UARTs entspricht ein Symbol einem Datenbit. Die Baudrate hat einige Standardwerte: 9600 oder 115200 sind die gängigsten. Wurde die Baudrate für beide Kommunikationspartner eingestellt, kann bestimmt werden, wie die Kommunikationspakete aussehen:

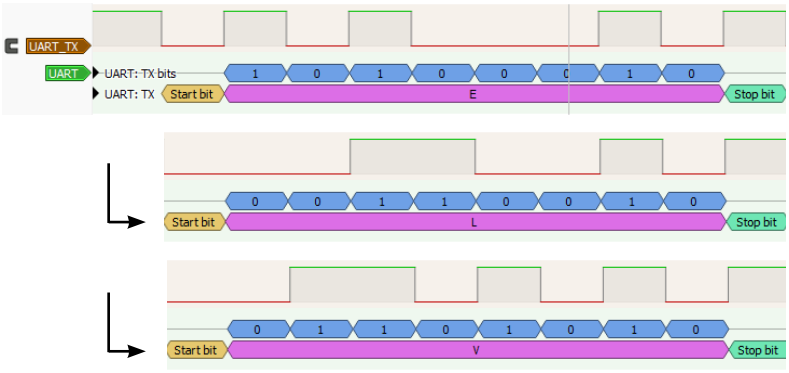


Bild 1: So sieht die Nachricht „ELV" aus, wenn sie über einen UART mit der Konfiguration 9600-8-N-1 verschickt wird.

1. Wenn nicht gesendet wird, hat die Kommunikationsleitung immer einen HIGH-Logikpegel.
2. Ein Datenpaket beginnt immer mit einem Startbit. Dieses signalisiert den Beginn eines Pakets und dient dem Empfänger als Synchronisationspunkt, von dem aus, entsprechend der Baudrate, die Bits aufgelöst werden.
3. Ein Datenpaket hat eine festgelegte Länge an Datenbits, möglich sind fünf bis neun, üblicherweise werden hier acht verwendet, da so genau ein Byte Daten in ein Paket passt.
4. Nach den Datenbits folgt ein optionales Paritätsbit. Es ermöglicht, eventuelle Fehler in der Kommunikation aufzudecken. Dazu werden die Datenbits jeweils entsprechend ihres Logiklevels gezählt. Je nachdem ob die Parität gerade (engl. EVEN) oder ungerade (engl. ODD) eingestellt ist, wird das Paritätsbit gesetzt, um eine entsprechende Anzahl an Bits zu versenden, die auf dem Logiklevel HIGH sind. Ein Beispiel: Es soll der Buchstabe „E" versendet werden, dieser entspricht den Datenbits „01000101". Wenn das Paket mit gerader Parität versendet werden soll, wird gezählt, wie viele Bits der Nachricht 1 sind. In diesem Fall sind es drei. Da drei eine ungerade Zahl ist, muss also das Paritätsbit auf 1 gesetzt werden, um eine gerade Zahl von 1en zu erreichen. So kann eine grundlegende Korrektheit der Nachricht geprüft werden, denn ist ein Bit „gekippt", stimmt die Parität nicht mehr mit dem Paritätsbit überein. Allerdings kann diese Herangehensweise lediglich erkennen, dass eine Kommunikation fehlgeschlagen ist, diese aber nicht korrigieren. Bei einem Fehler in zwei Bits ist sie ebenfalls nicht hilfreich, denn die Parität würde wieder stimmen. In vielen Fällen wird deswegen auf das Paritätsbit verzichtet und die Konfiguration NONE (Deutsch: keine) gewählt.
5. Nach den Paritätsbits folgt eine vorkonfigurierte Anzahl an Stopp bits. Es können 1, 1,5 oder 2 Bits gesendet werden. Die Stoppbits sind immer Logik-Level HIGH und symbolisieren das Ende eines Pakets.

Die konfigurierbaren Parameter der UART-Kommunikation lassen sich durch eine kurze Information zusammenfassen, die dem Format [Baudrate]-[Anzahl Datenbits]-[Parität]-[Anzahl Stoppbits] folgt. Die gängigste Konfiguration ist hier 9600-8-N-1. In dieser Konfiguration arbeitet auch die Schnittstelle der ELV Smart Home Sensor Base (siehe Bild 1).

Was ist beim Auslesen zu beachten?

Häufig werden Mikrocontroller wie Arduino UNOs oder auch Raspberry Pis verwendet, um Daten bzw. Messwerte von speziellen Sensoren auszulesen. Durch entsprechende Bibliotheken können diese Mikrocontroller unterschiedliche Standards auslesen und so z. B. mit I²C- oder auch anderen Bus-Sensoren kommunizieren und die ausgelesenen Daten softwareseitig zur Verfügung stellen.

Über die UART-Schnittstelle zwischen dem eingesetzten Mikrocontroller, wie bspw. dem Arduino UNO, sowie dem Mikrocontroller der Smart Home Sensor Base kann dann wiederum eine entsprechende Datenübertragung in das Homematic IP System erfolgen.

Damit zwei Mikrocontroller miteinander kommunizieren können, müssen die Ausgänge der jeweiligen UART-Schnittstellen verbunden werden. Diese sind im Datenblatt oder in einigen Fällen direkt auf dem Controller-Board mit Rx und Tx gekennzeichnet. Hier müssen die Pins so verbunden werden, dass ein Tx-Pin jeweils an den Rx-Pin der Gegenstelle angeschlossen ist. Auch die beiden Massen (GND) der beiden Teilnehmer müssen verbunden werden.

ACHTUNG! WICHTIG! ZERSTÖRUNGSGEFAHR!

Bevor Sie Geräte miteinander verbinden, prüfen Sie UNBEDINGT die Spannungskompatibilität beider Teilnehmer.
Sollten diese nicht übereinstimmen, müssen Sie einen sogenannten Level-Shifter einsetzen. Die Verwendung wird im Folgenden erklärt.

In der Kombination eines Arduino UNO und der ELV Smart Home Sensor Base ist eine solche Inkompatibilität gegeben. Der Arduino hat als Logikpegel HIGH ein Potential von 5 V, die maximal an den UART-Pins der ELV Smart Home Sensor Base anliegende Spannung beträgt 3,6 V. Eine direkte Verbindung könnte also zur Zerstörung führen. Abhilfe schafft hier ein Level-Shifter (Bild 2). Dieser ist im Bausatz der Prototypenadapter PAD4 enthalten. Bauen Sie diesen wie folgt in das System ein:

Arduino	Level-Shifter		ELV Smart Home Sensor Base
D0(RX)	B7	A7	PC01(TX)
D1(TX)	B8	A8	PC00(RX)
5V	Vccb	Vcca	+VDD
GND	GND	GND	GND
D8	B3	A3	PA00

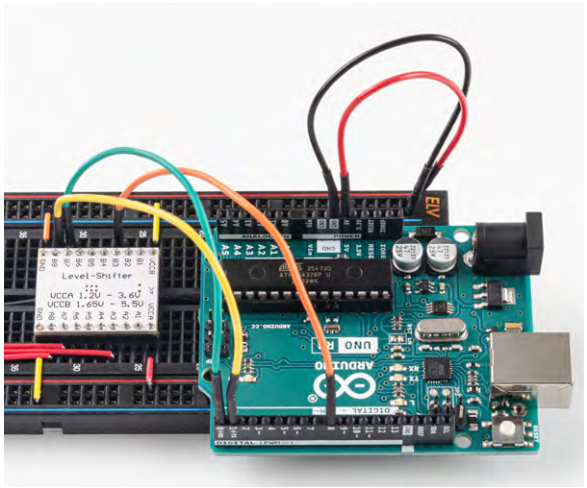


Bild 2: Der Level-Shifter schützt die Smart Home Sensor Base vor Überspannung.

Arduino-Bibliotheken verwenden

Jetzt sind die beiden Geräte korrekt miteinander verbunden, eine Komponente fehlt jedoch noch: Die aktualisierte Firmware! Um diese auf die ELV Smart Home Sensor Base zu installieren, gehen Sie auf die Produktseite und laden Sie die Firmware unter Downloads herunter. Wie Sie diese mit dem Flasher Tool auf die Base übertragen, finden Sie im Fachbeitrag.

Die Firmware für die Arduino-Seite muss allerdings noch geschrieben werden. Hierzu stellt ELV eine Arduino-Bibliothek zur Verfügung. Eine Bibliothek ist eine Sammlung von bereits geschriebenem Code, die bestimmte Funktionen bereitstellt. Anstatt alles von Grund auf selbst zu programmieren, können Entwickler und Bastler auf diese fertigen Bausteine zurückgreifen.

Ohne eine Bibliothek müsste selbst programmiert werden, wie die Schnittstelle angesteuert wird, welche Signale sie erwartet und welche Befehle nötig sind. Eine Bibliothek nimmt Ihnen diese Arbeit ab: Sie enthält bereits die richtigen Anweisungen und bietet einfache Befehle, um die UART-Schnittstelle der Smart Home Sensor Base anzusteuern.

Warum sind Bibliotheken so nützlich?

Bibliotheken bieten mehrere Vorteile:

- **Sie sind einfach:** Viele komplexe Aufgaben lassen sich mit wenigen Zeilen Code erledigen.
- **Sie minimieren Fehler:** Die meisten Bibliotheken wurden von erfahrenen Entwicklern geschrieben und getestet. Das reduziert Fehler im eigenen Code.
- **Sie sparen Zeit:** Anstatt sich stundenlang mit den technischen Details eines Bauteils auseinanderzusetzen, können Sie sich auf das Wesentliche konzentrieren: Ihr eigentliches Projekt!
- **Sie eröffnen weitere Möglichkeiten:** Dank Bibliotheken können Sie auch Bauteile nutzen, für die Sie selbst keine detaillierten Programmierkenntnisse besitzen.

Wie lässt sich eine Bibliothek nutzen?

Die Verwendung einer Bibliothek ist einfach:

Bibliothek installieren

Laden Sie zunächst die Bibliothek von der ELVshop-Seite der Smart Home Sensor-Base herunter. Öffnen Sie anschließend die Arduino IDE (mindestens Version 2.2.1) und fügen Sie die Bibliothek über das Menü „Sketch“ → „Include Library“ → „Add ZIP-Library“ hinzu, indem Sie das zuvor heruntergeladene ZIP-Archiv auswählen (siehe Bild 3).

Bibliothek in den Code einbinden

In Sketch (dem Arduino-Programmcode) wird eine Bibliothek mit #include <ELV-SH-BM-S.h> eingebunden. Anschließend können Sie die Funktionen der Bibliothek im Programm nutzen.

Beispielsketches nutzen

Die Bibliothek enthält einen Beispielprogrammcode, der zeigt, wie Sie diesen verwenden (Bild 4). Sie finden ihn unter „Datei“ → „Beispiele“ → „ELV-SH-BM-S UART Interface“ in der Arduino IDE.

Hauptbestandteil der Bibliothek ist die Funktion „sendFrame(channel_no_t channel, float value)“. Diese hat als Übergabeparameter als Erstes die Messwertkanalnummer. Die Messwertkanalnummern sind bereits vordefiniert und können einfach als Konstante mit dem Namen ELV_SH_BM_S::CHANNEL_[NR] abgerufen werden.

Als Nächstes wird der Übergabewert erwartet mit dem Datentyp Float oder ein 16-Bit-Integer, in dem der eigentliche Messwert steht.

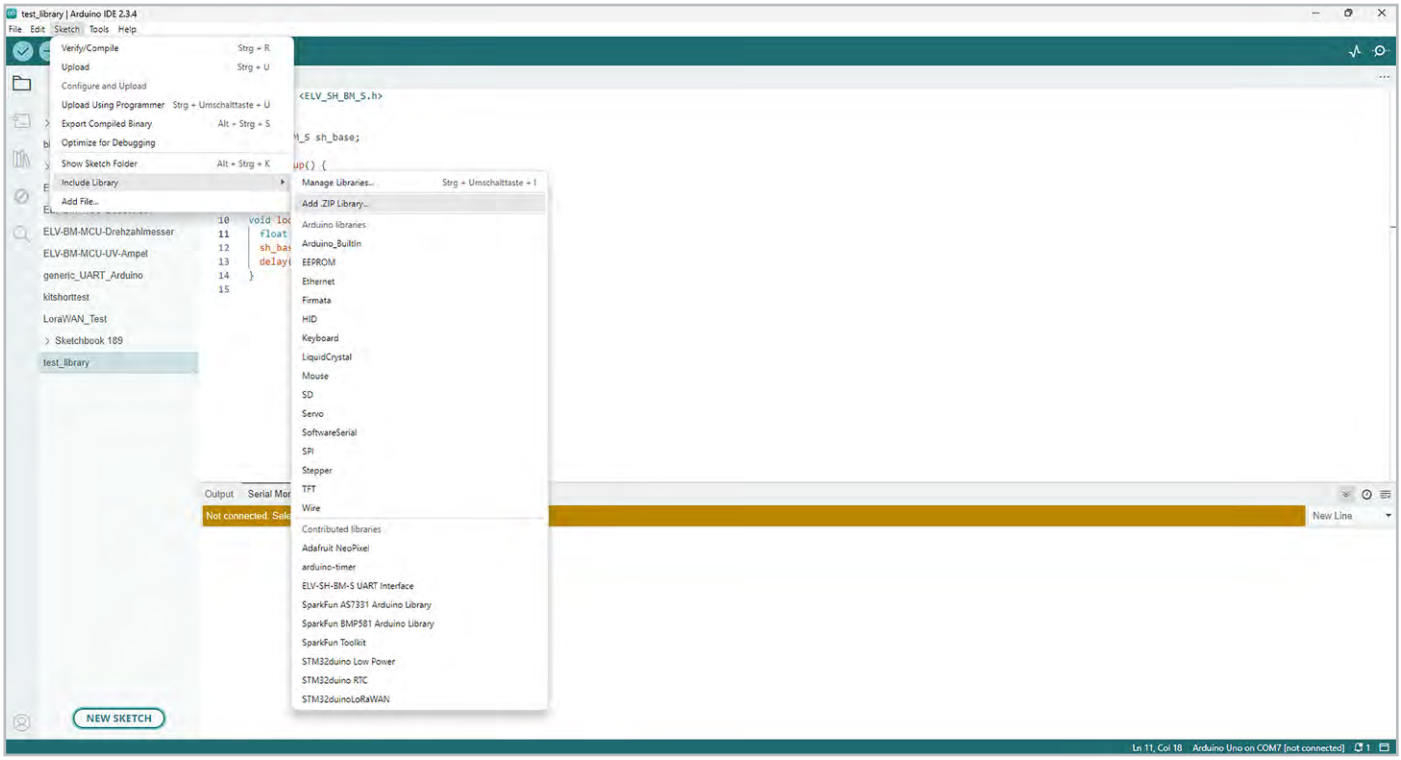


Bild 3: So fügen Sie die Bibliothek hinzu.

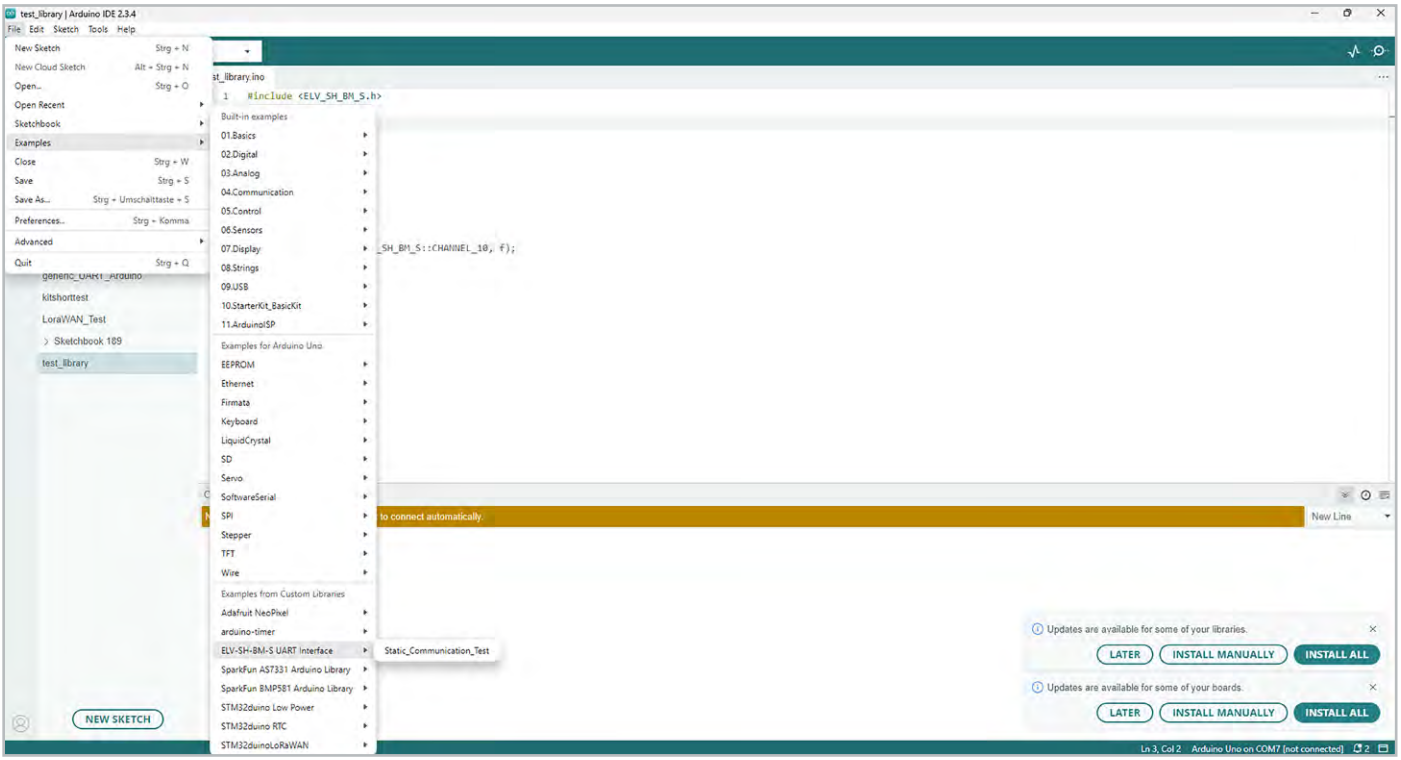


Bild 4: Beispielaufbau zwischen einer Smart Home Sensor Base und einem über den Arduino angeschlossenen Staubsensor

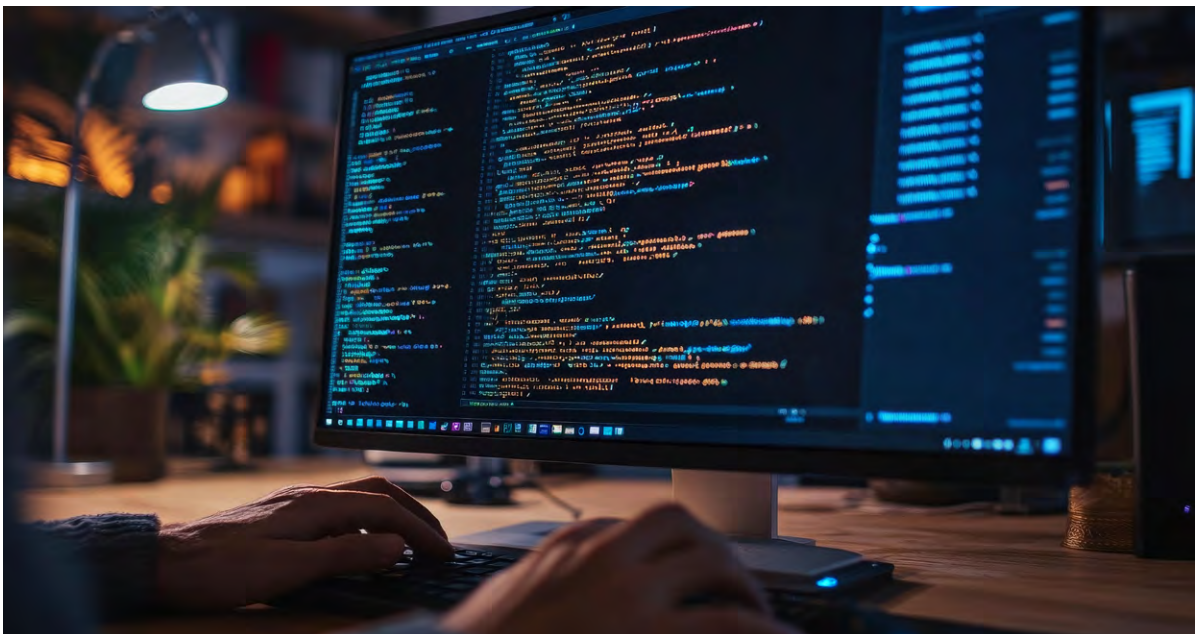
CODEBLOCK 1: Beispiel, in dem ein fester Wert in Messwertkanal 10 geschrieben wird:

```
#include <ELV_SH_BM_S.h>


ELV_SH_BM_S sh_base;

void setup() {
  sh_base.begin();
}

void loop() {
  float f = 20000;
  sh_base.sendFrame(ELV_SH_BM_S::CHANNEL_10, f);
  delay(5000);
}
```



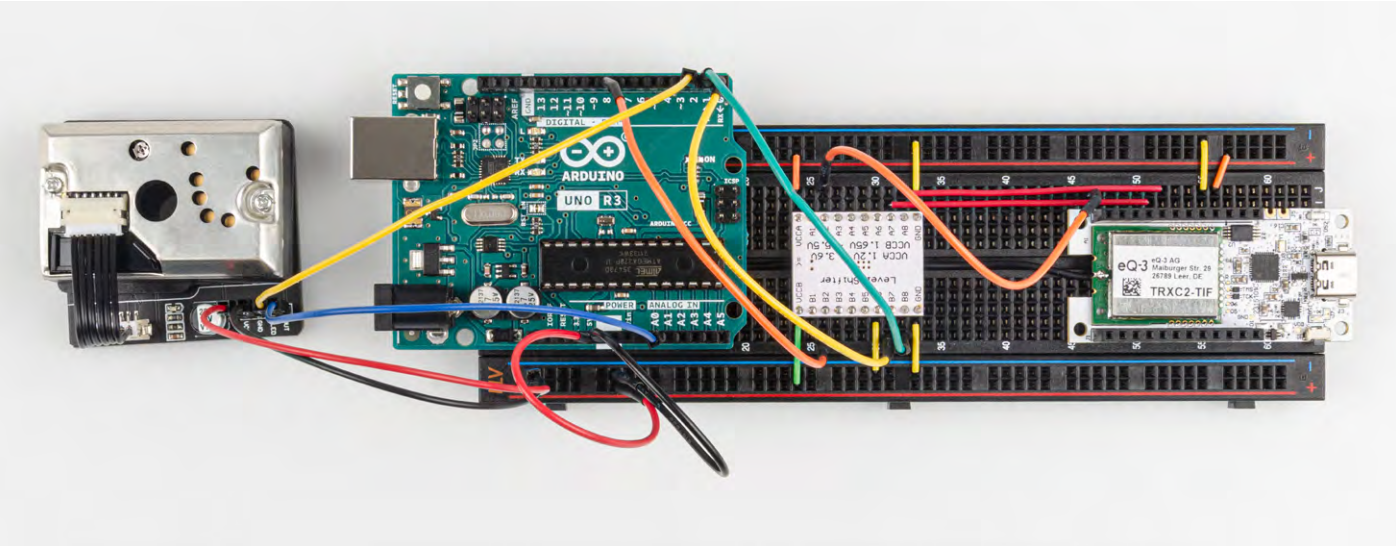


Bild 5: Ein Beispielaufbau der Smart Home Sensor Base mit einem Staubsensor

Anwendungsbeispiel

Neben der theoretischen Beschreibung zur Einbindung unterschiedlichster Sensoren wollen wir uns nachfolgend auch mit einer praxisorientierten Beispielkombination (Bild 5) mit einem Arduino UNO sowie einem optischen Staubpartikelsensor von Joy-IT beschäftigen. Dieser Sensor ist in der Lage, die Staubkonzentration zu messen und als Messwert im Homematic IP System zur Verfügung zu stellen.

1 Zunächst wird hierfür die Smart Home Sensor Base wie in der eingangs zu sehenden Tabelle beschrieben über einen Level-Shifter mit dem Arduino verbunden.

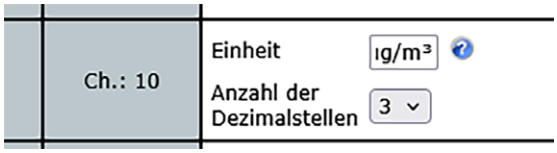
2 Verbinden Sie den Staubsensor entsprechend der nachfolgenden Kontaktzuweisung:

Arduino	Staubsensor
5V	VCC
GND	GND
2	LED
A0	OUT

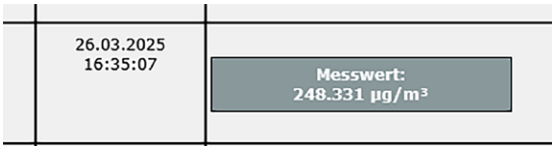
3 Flashen bzw. beschreiben Sie die Smart Home Sensor Base mit der Firmware für die generische UART-Schnittstelle.

4 Der Arduino wird mit dem Code aus Codeblock 2 beschrieben:
- Zunächst wird in Zeile 1 die notwendige Bibliothek hinzugefügt.
- Es folgen einige Pin-Definitionen für den hier verwendeten Staubsensor. Sollten andere als die in der Tabelle aufgeführten Anschlusspins gewählt werden, so müssen diese im Code angepasst werden.
- Die setup()-Methode initialisiert die Kommunikation mit der Base sowie die beiden Anschlusspins des Staubsensors.
- Abschließend wird mit der Loop-Methode der Output des Sensors eingelesen und in Zeile 25 mit der Methode sendFrame() als Wert in den Messwertkanal 10 der SH-Base geschrieben.

5 In der WebUI der CCU3 kann nun unter „Einstellungen" → „Geräte" → [Verwendete SH-Base] → „Einstellen" der Kanal 10 so konfiguriert werden, dass er drei Nachkommastellen liefert und der Einheit „µg/m³" entspricht.



6 Unter „Status und Bedienung" → „Geräte" → [Verwendete SH-Base] kann nun der ausgelesene Wert des Staubsensors im Messwertkanal 10 eingesehen werden.



Ausblick

Dank der Vielzahl an Sensoren und den damit verbundenen Einsatzzwecken sind Ihrer Fantasie keine Grenzen gesetzt. Eine große Auswahl an Mikrocontroller-kompatiblen Sensoren finden Sie unter anderem in unserem Shop in der Kategorie Sensoren.

Auch die kreative Verwendung verschiedener Messwertkanäle, z. B. zur Speicherung von Extremwerten, ist durchaus denkbar.

Senden Sie uns daher gerne besonders innovative oder interessante Kombinationen an: redaktion@elv.com.

Wir freuen uns auf Ihre Einsendungen!

CODEBLOCK 2: Arduino-Anwendung mit Staubsensor

```
#include <ELV_SH_BM_S.h>

int dustPin = 0;
float dustVal = 0;
int ledPower = 2;
int delayTime = 280;
int delayTime2 = 40;
float offTime = 9680;
ELV_SH_BM_S base;

void setup() {
  base.begin();
  pinMode(ledPower, OUTPUT);
  pinMode(dustPin, INPUT);

 }
 void loop() {
   digitalWrite(ledPower, LOW);
   delayMicroseconds(delayTime);
   dustVal = analogRead(dustPin);
   delayMicroseconds(delayTime2);
   digitalWrite(ledPower, HIGH);
   delayMicroseconds(offTime);
   delay(1000);
   if (dustVal > 36.455)
     base.sendFrame(ELV_SH_BM_S::CHANNEL_10,(float(dustVal / 1024) - 0.0356) *
120000 * 0.035);
 }
```