Sensordaten drahtlos zur Smartphone-App phyphox übertragen und grafisch auswerten –

ein einfaches Beispiel mit dem ESP32 und dem Ultraschallsensor HC-SR04

Dominik Dorsel – Sebastian Staacks – Maximilian Loch – Alexander Pusch

Bei vielen Projekten mit Mikrocontrollern möchte man die Messwerte der angeschlossenen Sensoren gerne grafisch darstellen und auswerten. Dies erforderte bisher in der Regel aber einen umständlichen Export der Daten. In diesem Artikel stellen wir eine sehr einfache und universelle Möglichkeit vor, Messwerte von einem Mikrocontroller per Bluetooth in Echtzeit an die App *phyphox* zu übertragen und grafisch darzustellen. An einem einfachen Beispiel wird anschließend gezeigt, wie mit dem Mikrocontroller ESP32 Messwerte des aus physikalischer Sicht sehr interessanten Ultraschallsensors HC-SR04 in einfachen Experimenten aufgenommen und auf dem Smartphone dargestellt werden können.

1 Messdaten von Mikrocontrollern grafisch darstellen

Es gibt nahezu kein Arduino-Projekt, das ohne Sensor auskäme. Jedes Starter Kit enthält neben Kabeln und Tasten eine Reihe von Sensoren. Dies ist nicht verwunderlich, da mit den meisten Projekten ein Ereignis in der Umgebung erkannt und dazu passend etwas ausgelöst werden soll. Lassen wir „Eingabe-Sensoren“ wie Tasten und Potentiometer außen vor, müssen dazu physikalische Größen erfasst werden, was mit einfachen Photowiderständen (Beleuchtungsstärke) beginnt, über Thermometer und Luftfeuchtesensoren geht und beim Ultraschall-Entfernungsmesser keinesfalls endet. Zugehörige Bibliotheken übernehmen die Kommunikation mit den Sensoren, so dass auch Programmieranfänger/innen ohne tiefere Kenntnis über die Funktionsweise oder Kommunikation mit dem Sensor mit wenigen Zeilen Code die Sensordaten auslesen und damit LEDs und Servomotoren steuern können.
Aus Sicht der Physik bleibt hierbei jedoch leider oft der Messwert selbst auf der Strecke. Er wird eventuell zur Fehlersuche in das serielle Terminal des angeschlossenen PCs geschrieben, ist aber andernfalls nur ein Steuersignal für einen Aktuator wie z.B. eine LED oder einen Servomotor. Dabei sehnen sich gerade Physiklehrkräfte in Schulen nach “echten” Messdaten, um mit Schüler/inne/n „echte” Experimente durchführen zu können und theoretische Zusammenhänge mit Beobachtungen aus der realen Welt zu untermauern. Laborausstattung ist allerdings ebenso teuer wie Messgeräte der Lehrmittelhersteller, so dass die Schule oft neidisch auf die Hobby-Bastler/innen schaut, die beispielsweise für wenige Euro einen Luftdrucksensor in ihren Projekten nutzen können.
Solange die Messwerte aber nur als rohe Fließkommazahl in den seriellen Monitor der Arduino-IDE am angeschlossenen Computer geschrieben werden können, ist eine solche Lösung für den Unterricht wenig geeignet, insbesondere wenn Schüler/innen während der Experimente selbst damit arbeiten sollen und somit einen kompletten Rechner als Messgerät bräuchten. Technisch ist es zwar auch anders möglich, diese Daten am Rechner in Empfang zu nehmen, zu plotten oder sie sogar über drahtlose Schnittstellen weiter zu reichen, aber hier endet dann in der Regel die Behauptung, dass solche Projekte für Programmieranfänger/innen in wenigen Zeilen umsetzbar sind.
Mit der *phyphox*-Bibliothek für die Plattform Arduino wurde hierfür eine Alternative veröffentlicht. Der Name *phyphox* bezieht sich hierbei auf die bekannte und verbreitete Smartphone-App *phyphox* (Staacks, Hütz, Heinke & Stampfer, 2018), welche die in Smartphones verbauten Sensoren zunächst ohne Mikrocontroller und weitere Hardware für den Physikunterricht zugänglich macht und deren Messwerte für Experimente in Schule und Hochschule aufbereitet.

2 Die phyphox-Bibliothek für Mikrocontroller

Die *phyphox*-Bibliothek bietet die Möglichkeit, die Messdaten nahezu beliebiger Sensoren bequem über Bluetooth Low Energy (BLE) am Smartphone in *phyphox* darzustellen. Die App kann selbst auf durchschnittlichen Handys mehrere 100.000 Datenpunkte flüssig zoombar darstellen. Sie ist den Schüler/innen/n in der Bedienung oft schon bekannt und unterstützt sowohl Android- als auch iOS-Geräte, so dass die App ein ideales Visualisierungswerkzeug für externe Sensordaten darstellt. Hierzu sind mit der Bibliothek nur drei Zeilen Code nötig. Einzige Voraussetzung ist ein unterstützter Bluetooth-fähiger Mikrocontroller wie der Arduino Nano 33 BLE, Arduino Nano 33 IoT, Arduino Nano Sense oder auch die besonders günstigen Arduino-kompatiblen ESP32-Boards. Wir verwenden die Variante ESP32 Dev Kit C für unser nachfolgendes Beispiel.
Programmcodes für den weit verbreiteten Mikrocontroller Arduino zum Auslesen der Sensoren laufen in der Regel auch auf dem ESP32. Die Programme können ebenfalls mit der verbreiteten Arduino-IDE kompiliert und aufgespielt werden. Der ESP32 ist im Vergleich aber deutlich leistungsfähiger, vergleichsweise günstig, hat Bluetooth und WiFi-Funktionalität und daher für dieses Vorhaben eine gute Wahl. Die Bluetooth-Funktionalität kann beim Arduino UNO bspw. auch mit einem Breakoutboard nachgerüstet werden (Pusch, Ubben, Laumann, Heinicke & Heusler, 2021).

MNU-Journal – Ausgabe 01.2023 – ISSN 0025-5866 – © Verlag Klaus Seeberger, Neuss

```
#include <phyphoxBle.h>                          //phyphox-Bibliothek einbinden

void setup() {
    PhyphoxBLE::start();                         //Bluetooth-Funktion starten
}

void loop() {
    float voltage = analogRead(34)*3.3/4096.0; //Spannung an Pin 34
                                                 // messen
    PhyphoxBLE::write(voltage);                 //Messwert an phyphox schicken
    delay(50);                                   //Kurze Pause
}
```

Programmcode 1. Rudimentärer Programmcode zur Datenübertragung mit der *phyphox*-Bibliothek

Ist ein kompatibler Mikrocontroller vorhanden, kann man die *phyphox*-Bibliothek bequem über die Bibliothek-Verwaltung der Arduino-IDE installieren und sofort mit einem kleinen Code-Schnipsel (Programmcode 1) Messwerte eines Analogeingangs in *phyphox* plotten.
Die eigentliche Messung ist in diesem Beispiel für den ESP32 das Auslesen des Analogeingangs an Pin 34 mit dem Umrechnungsfaktor 3,3V/4096. Der Umrechnungsfaktor ergibt sich aus der Auflösung des Analog-Digital-Wandlers (beim ESP32 4096 Werte) und seiner Referenzspannung (hier 3,3V). Das Ergebnis ist eine Spannung zwischen 0 und 3,3V. Ein kurzes `delay` begrenzt die Rate, mit der Werte generiert werden.
Um die Werte an *phyphox* zu übermitteln und dort zu plotten, sind nur drei Schritte notwendig: Die Bibliothek muss mittels `include` eingebunden werden, der Bluetooth-Betrieb wird dann mit `PhyphoxBLE::start()` aktiviert und anschließend können Messwerte beliebig über `PhyphoxBLE::write(voltage)` verschickt werden.
In der Praxis haben wir hiermit nun schon ein (wenn auch nicht allzu genaues und störanfälliges) Multimeter, dessen Messung in *phyphox* dargestellt werden kann. Verbindet man den Arduino (bzw. in unserem Fall den ESP32) mit der Spannungsversorgung, kann man auf einem Smartphone in der App *phyphox* auf das „+“-Symbol im Hauptmenü tippen und den Eintrag für Bluetooth-Geräte wählen. Der Mikrocontroller erscheint im Suchdialog und nach dessen Auswahl gelangt man schon in die Messansicht.

Abb. 1. Screenshot einer Spannungsmessung im zeitlichen Verlauf. Über ein Potentiometer wurde die Spannung am analogen Eingang variiert, was in Echtzeit in phyphox dargestellt wird.

Ab hier kann man die Datenaufnahme einfach mit einem Tipp auf das Startsymbol (dreieckiger „Play-Button“) beginnen und sieht sofort, wie die gemessene Spannung im Graphen aufgetragen wird. (Abb. 1)
Selbstverständlich kann man auch physikalisch korrekt die Achsenbeschriftung im Arduino-Code anpassen, wozu nur wenige weitere Zeilen notwendig sind. Wie das geht, wird im nächsten Abschnitt am Beispiel des Ultraschallsensors HC-SR04 erklärt.

3 Messdaten des Ultraschallsensors HC-SR04 mit dem ESP32 übertragen

Im Grunde braucht man nur die zuvor genannten drei Zeilen Code, um die Sensordaten von einem (nahezu beliebigen) Sensor mit einem Mikrocontroller wie dem ESP32 zur App *phyphox* zu übertragen. In dem folgenden Beispiel möchten wir mit dem Ultraschallsensor Typ HC-SR04 zeigen, wie dieser Sensor eingebunden wird, die Darstellung in der App *phyphox* beschriftet und einfache Experimente realisiert werden können.
Der Ultraschallsensor HC-SR04 ist vergleichsweise kostengünstig und seine Funktionsweise ist zudem für den (Physik)Unterricht sehr interessant (Kasten 1). Bei der Verwendung des ESP32 statt z.B. des bekannten Arduino UNO ist zu beachten, dass der ESP32 mit 3,3V statt 5V arbeitet. Dieses in der Praxis häufig auftretende Problem der unterschiedlichen Logik-Spannungen hat Auswirkungen auf die Auswahl und den Anschluss von Sensoren und Aktuatoren. Der HC-SR04 ist bspw. auf 5V an seinem Ein- und Ausgang ausgelegt. Um ihn zu verwenden, muss das 5V Signal des Sensors mittels eines Spannungsteilers auf 3,3V reduziert werden (Kasten 2). Die benötigte 5V Versorgungsspannung für den HC-SR04 kann vom ESP32 abgegriffen werden, der diese aus dem USB-Kabel durchschleift.

3.1 Umsetzung der Schaltung

Es gibt verschiedene Möglichkeiten, Sensoren wie den HC-SR04 an einen Mikrocontroller anzuschließen. Nachfolgend stellen wir zwei Varianten zur Umsetzung vor und diskutieren diese aus schuldidaktischer Perspektive.
Die erste Variante ist die Umsetzung der Schaltung auf einem **Steckbrett** (Breadboard), wie in Abbildung 2 dargestellt. Auffällig bei der breiten Bauform vieler gängiger ESP32 Entwicklerboards wie dem Dev Kit C ist, dass nur noch die Anschlüsse einer Seite abgegriffen werden können. Möchte man beide

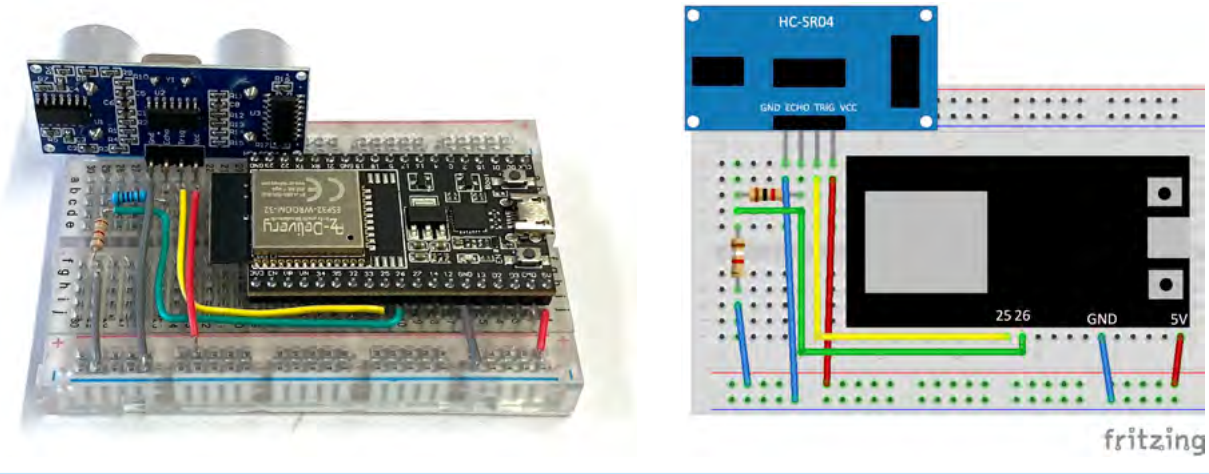

Abb. 2. Schaltungsaufbau mit einem Steckbrett (Breadboard)

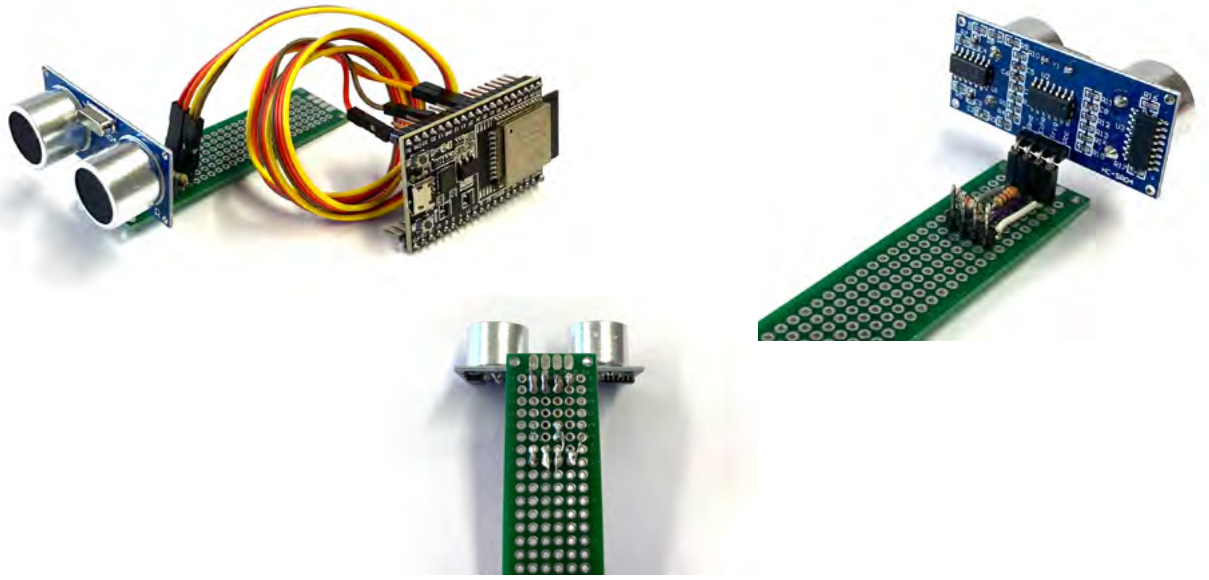

Abb. 3. Schaltungsaufbau mit einer Platine

Seiten abgreifen, kann man auch zwei Steckbretter nebeneinanderlegen und den ESP auf beide Bretter stecken (dies ist in der Praxis aber etwas wackelig). Wenn bereits grundlegende Erfahrungen mit Steckbrettern gesammelt wurden, kann diese Variante auch mit Lernenden nachgebaut werden. Die farbigen Drähte helfen beim Aufbau und Nachvollziehen auf Grundlage des Schaltplanes (verfügbar unter *https://phyphox.org/p/mnu-arduino*). Steckbretter weisen generell eine große Flexibilität auf, die vor allem beim Erstellen von Prototypen wichtig ist. Es können sich allerdings beim Experimentieren die Kabel lösen oder beim Aufbau Fehler in der Schaltung einschleichen (PUSCH, im Druck).

Wesentlich robuster ist die zweite Variante mit einer **Platine** (Abb. 3). Dort wird der Spannungsteiler fest verlötet, der Sensor und die Verbindungskabel sind steckbar. Eine Platine kann gut in verschiedene Experimente (z.B. mit Klemme am Stativmaterial) oder in ein Gehäuse integriert werden. Beim Anschließen der Kabel an die Steckverbindungen muss auf die korrekte Belegung geachtet werden. Die Schaltung kann nachträglich allerdings nicht mehr einfach geändert werden, sodass diese Variante erst nach dem Prototypen-Status sinnvoll ist.

3.2 Erweiterung des Programmcodes

Der zu Beginn in Abschnitt 2 gezeigte Befehl `analogRead()` kann einfach durch die entsprechende Rückgabe eines Sensors ersetzt werden. Der HC-SR04 unseres Beispiels kann durch die Arduino-native Funktion `pulseIn()` ausgelesen werden (Programmcode 2). So wird aus dem bereits vorgestellten Code Programmcode 2. Die Änderung, die vorgenommen wurde, ist die Quelle unserer Messwerte. Die Berechnung, angehängt an die `pulseIn()`–Funktion, dient dabei der Umrechnung von Mikrosekunden in die Distanz in Metern. Mehr dazu im Kasten 2. Um die in der App *phyphox* dargestellten Graphen anzupassen, werden weitere Programmzeilen benötigt (Programmcode 3). Die Einstellungen für die Darstellung können mit den folgenden Befehlen gesetzt werden.

```
#include <phyphoxBle.h>                 //phyphox-Bibliothek einbinden

void setup() {
    PhyphoxBLE::start();                // Bluetooth-Funktion starten
    pinMode(25, OUTPUT);                // Setzt den Pin 25 als Output
    pinMode(26, INPUT);                 // Setzt den Pin 26 als Input
}

void loop() {
    digitalWrite(25, HIGH);             // Starten des Sendeimpulses
    delayMicroseconds(5);               // Pause (5 Microsekunden)
    digitalWrite(25, LOW);              // Stoppen des Sendeimpulses

    // Auslesen der Zeitdauer des HIGH-Signals am Pin 26, Umrechnen
    // des Messwertes, speichern in die Fließkommavariable „d“
    float d = pulseIn(26, HIGH)* 343 / (2 * 1000000.0);

    PhyphoxBLE::write(d); // Messwert „d“ an phyphox schicken
    delay(50);                  // Pause (50 Millisekunden)
}
```

Programmcode 2. Auslesen des Ultraschallsensors HC-SR04 und Übertragung mit der phyphox-Bibliothek

```
PhyphoxBleExperiment::Graph graph;          // Erstelle Graphen
graph.setLabel(„Abstand gegen Zeit“);       // Überschrift
graph.setUnitX(„s“);                        // Einheit der X-Achse
graph.setUnitY(„m“);                        // Einheit der Y-Achse
graph.setLabelX(„Zeit“);                    // Name der X-Achse
graph.setLabelY(„Abstand“);                 // Name der Y-Achse
```

Programmcode 3. Parameter zur Darstellung der Graphen

Dies ist nur der Code-Auszug, der die Anpassungsmöglichkeiten eines Graphen illustriert. Die Darstellung eines Experiments in *phyphox* gliedert sich in sogenannte “Views” (Tabs in der *phyphox*-Ansicht), welche wiederum mehrere Elemente wie den hier deklarierten Graphen `graph` enthalten können. Der vollständige Code legt daher solche Elemente als Variablen an, fügt diese einem “View” hinzu und bündelt einen oder mehrere solcher “Views” zu einem Experiment.
Der vollständige, kommentierte Programmcode, der beliebig für andere Sensoren angepasst und adaptiert werden kann, ist

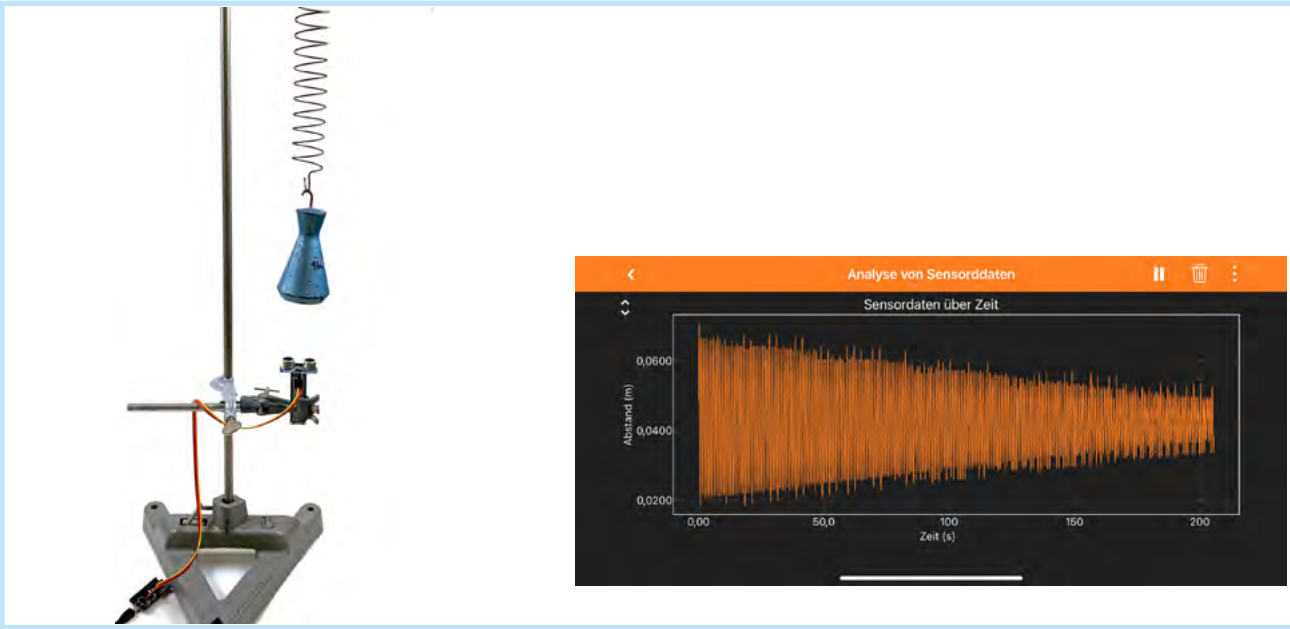

Abb. 4. Aufbau Federpendel sowie Messergebnisse. Erkennbar ist die Dämpfung

unter *https://phyphox.org/p/mnu-arduino* zum Download verfügbar. Die Anpassung ist denkbar einfach, nahezu selbsterklärend und ermöglicht eine beliebige Beschriftung, die auch von den Schüler/innen/n vorgenommen werden kann.

3.3 Experimente mit dem Ultraschallabstandssensor

Mit einem Abstandssensor wie dem HC-SR04 können einfache Demo- oder Schülerexperimente samt grafischer Auswertung mit der App *phyphox* durchgeführt werden. Dazu gehören z.B. die Aufnahme der Auslenkung eines Federpendels (Abb. 4), die beschleunigte Bewegung eines Rollwagens an der geraden und schiefen Ebene (Abb. 5 und Abb. 6) sowie Rotationsbewegungen (Abb. 7). Die technische Anwendung des im Kasten 2 beschriebenen Prinzips des Sensors kann am Beispiel Einparkhilfe oder Sonar thematisiert werden. Mögliche Kontexte sind z.B. auch die Echoortung von Fledermäusen.

Hinsichtlich der Güte der Messungen ist der HC-SR04 nicht der „beste“ Sensor. Es gibt gerade bei größeren Abständen und nicht parallel zum Sender stehenden Oberflächen vereinzelte Ausreißer. Diese Faktoren geben aber Lernenden auch eine

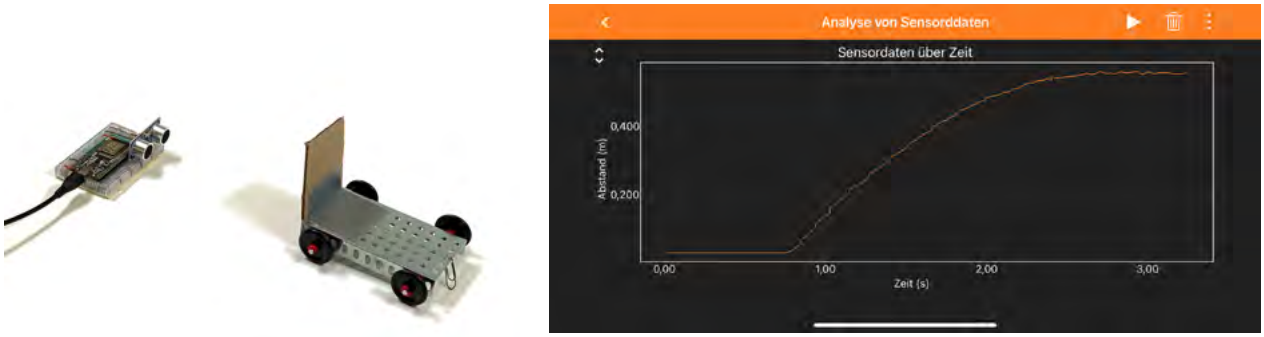

Abb. 5. Aufbau beschleunigte Bewegung auf Ebene sowie Messergebnisse. Damit der Sensor den Wagen besser erfassen kann, wurde eine Pappscheibe montiert.

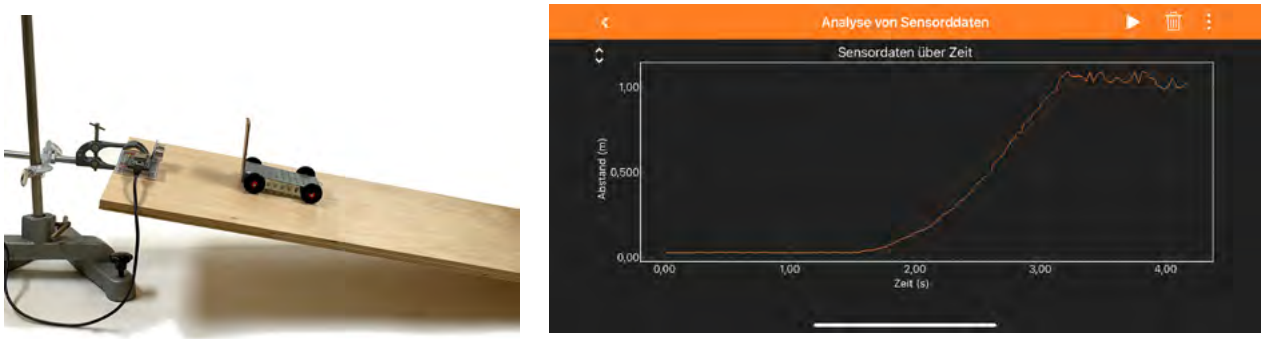

Abb. 6. Aufbau beschleunigte Bewegung auf schiefer Ebene sowie Messergebnisse. Damit der Sensor den Wagen besser erfassen kann, wurde eine Pappscheibe montiert.

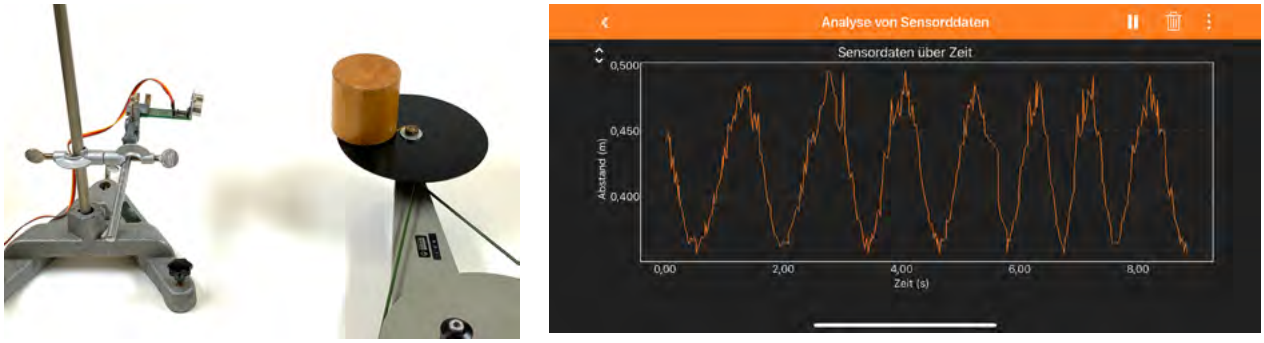

Abb. 7. Aufbau periodische Rotationsbewegung sowie Messergebnisse. Alternativ kann auch ein Drehteller oder Bürostuhl verwendet werden.

Der Unterschied zwischen physikalischer und technischer Funktionsweise
Zu unterscheiden ist die physikalische Funktionsweise, die im Rahmen des Physikunterrichts in den Themen Akustik und Kinematik thematisiert werden kann, sowie die technische Funktionsweise, die für den Anschluss und Betrieb des Sensors relevant ist. Im Folgenden werden beide dargestellt:

Physikalische Funktionsweise
Ein Ultraschallabstandssensor wie der HC-SR04 sendet aus einem Lautsprecher ein Ultraschallsignal von etwa 40 kHz aus, welches an Oberflächen reflektiert wird. Die Reflektion des Signals wird vom Sensor empfangen (Abb. 8). Aus der Laufzeit t des Signals (hin und zurück) lässt sich die Distanz d bis zum Objekt durch $d = 1/2 \cdot v \cdot t$ berechnen. Die Schallgeschwindigkeit v in Luft beträgt bei 20 °C und Normaldruck $v = 343$ m/s. Der Einfluss der Temperatur auf die Messung ist gering, aber vorhanden: bei 0 °C liegt die Schallgeschwindigkeit bei $v = 331{,}5$ m/s und bei 30 °C bei $v = 349{,}3$ m/s.
Die Schallwellen breiten sich kegelförmig mit einem Öffnungswinkel von etwa 15° aus (s. Datenblatt HC-SR04). Dies hat die Konsequenz, dass auf größere Entfernung auch unbeabsichtigt andere Objekte im Kegel detektiert werden können und gleichzeitig die Reichweite auch durch die Stärke des reflektierten Signals begrenzt ist (beim HC-SR04 in der Praxis grob 3m). Kleinere Objekte lassen sich nur im Nahbereich erfassen.

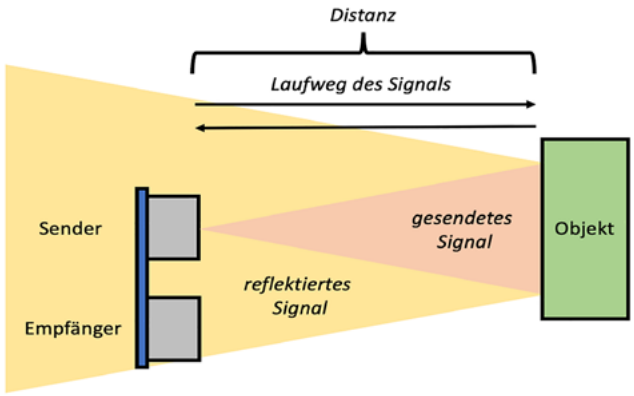

Abb. 8. Physikalisches Funktionsprinzip einer Ultraschallabstandsmessung (stark vereinfacht).

Technische Funktionsweise
Der HC-SR04 hat vier Anschlüsse: zwei zur Spannungsversorgung (5V-VCC und GND) sowie einen Trigger- und einen Echo-Pin. Um die Messung zu starten, muss auf den Trigger-Pin ein kurzes High-Signal gelegt werden. Eine Spannung wird als HIGH bzw. LOW gewertet, wenn sie eine gewisse Schwelle über- bzw. unterschreitet. Obwohl der Sensor eigentlich auf 5V ausgelegt ist, registriert er aber – wie viele andere auch – ein Signal von 3,3V ebenfalls als HIGH bzw. logische 1. Der Sensor sendet anschließend eine Signalfolge im Ultraschallbereich, deren Reflektion wieder detektiert wird. Die Zeitdauer, die die Signalfolge "unterwegs" war, wird anschließend über den Echo-Pin wiedergegeben. Dieser wird für die entsprechende Zeitdauer auf HIGH (5V) gesetzt.
Abbildung 9 zeigt den Signalverlauf einer Messung mit Hilfe eines Logic-Analysers. Ein Logic-Analyser ist vergleichbar mit einem digitalen Oszilloskop, welches die zeitlichen Zustände von Signalen (HIGH und LOW) misst. Mit dem Mikrocontroller kann die Zeitdauer des HIGH auf dem Echo-Pin über die Funktion `pulseIn()` gemessen werden und anschließend unter Einbezug der mittleren Schallgeschwindigkeit in Luft in die Gesamtdistanz (Laufweg durch zwei dividieren) umgerechnet werden. In dem dargestellten Fall beträgt die auf dem Echo-Kanal rückgemeldete Laufzeit $t = 1{,}26$ ms. Eingesetzt in die Formel $d = 1/2 \cdot v \cdot t$ ergibt sich für den Abstand $d = 1/2 \cdot 343$ m/s $\cdot$ 0,00126 $s = 0{,}22$ m.

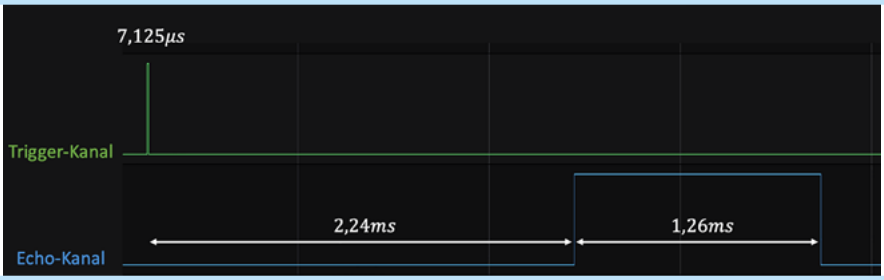

Abb. 9. Aufnahme mit einem Logic-Analyser. Der Abstand zwischen Sender und Hindernis beträgt 22 cm.

Kasten 1. Funktionsweise des Ultraschallsensors HC-SR04

Ein Spannungsteiler ist eine Reihenschaltung aus (z.B.) zwei Widerständen (vgl. Abb. 10). Er kann u.a. dafür genutzt werden, um die abzugreifende Spannung in einer Schaltung auf einen bestimmten Wert einzustellen oder diese mit Hilfe eines verstellbaren Widerstandes (Potentiometer) zu variieren.
Nach den Kirchhoffschen Regeln teilt sich die Spannung auf die beiden Widerstände auf, während durch beide der gleiche Strom fließen muss. Aus dem ohmschen Gesetz folgt damit, dass das Verhältnis der beiden Spannungen dem Verhältnis der Widerstände entspricht. Um aus den 5V des Sensors also die Logikspannung des ESP32 von 3,3V zu erhalten, müssen wir die Spannung in 3,3V + 1,7V = 5V aufteilen. Mit 3,3V/1,7V ≈ 2/1 erreichen wir das durch einen Spannungsteiler mit Widerständen im Verhältnis von $R_2/-R_1 \approx 2{:}1$ (Abb.10).
Exakt passende Verhältnisse zwischen zwei Widerständen sind in der Praxis wegen der Widerstandsreihen und den zur Verfügung stehenden Widerstandssortimenten oft nicht möglich. Ähnliche Verhältnisse funktionieren aber meist ausreichend gut, wie in diesem Fall z.B. 1,0 *k*Ω und 1,94 *k*Ω oder 3,3 *k*Ω zu 6,41 *k*Ω . Zu Bedenken ist natürlich, dass höhere Widerstände, bei gleichem Verhältnis, eine geringere Stromstärke zur Folge haben. Um die angelegte Spannung als Logik-Zustand zu detektieren, muss durch den Mikrocontroller aber kein großer Strom fließen.

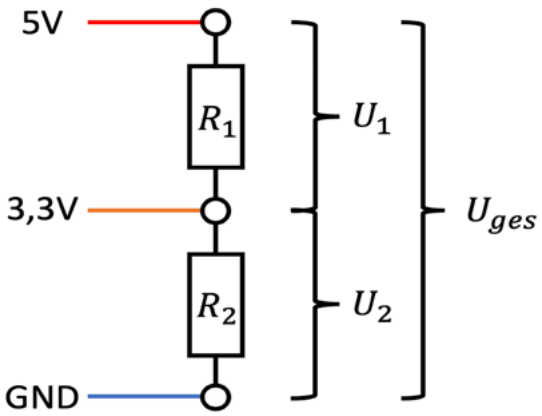

Abb. 10. Schaltbild eines Spannungsteilers

Kasten 2. Spannungsteiler

Möglichkeit, ein realistisches Bild über Messprozesse zu erhalten. Es lassen sich in den Daten deutlich sichtbar die Bewegungsformen von periodischen und auch beschleunigten Bewegungen sowie Reibungseffekte erkennen. Der Bau der Schaltung, die Anwendung des Messverfahrens mit seinen Besonderheiten (z.B. Oberflächeneinfluss, Messkegel, Temperatureinfluss), die Optimierung des Programmcodes (bspw. „Abfangen“ von Fehlmessungen, Mittelungen um Messwerte zu “glätten” etc.) und die Festlegung der Beschriftungen der Darstellung können neben den Experimenten selbst auch sehr sinnvolle Lerngelegenheiten darstellen.

4 Ausblick

In diesem Artikel wurde ein einfaches Beispiel gezeigt, wie Messwerte zwischen Mikrocontrollern wie dem ESP32 und der Smartphone-App *phyphox* übertragen werden können. Alternative Methoden wie das Nutzen des seriellen Monitors und Plotters oder die Anbindung eines Displays sind oftmals ein vergleichsweise größerer Mehraufwand mit geringerem Praxisnutzen durch die fehlenden Möglichkeiten bei der Diagrammdarstellung.

In einem nächsten Vorhaben werden wir zeigen, wie man zwei Sensordaten gleichzeitig an *phyphox* schickt und gegeneinander aufträgt, um so komplexere Experimente realisieren zu können.

Der ausführlich kommentierte Programmcode und eine erprobte Schritt-für-Schritt-Anleitung zur Installation der *phyphox*-Bibliothek und zur Einbindung der Board-Treiber des ESP32 in die Arduino-IDE ist unter *https://phyphox.org/p/mnu-arduino* zu finden.

Literatur

Datenblatt HC-SR04: *https://www.mikrocontroller.net/attachment/218122/HC-SR04_ultraschallmodul_beschreibung_3.pdf* (letzter Aufruf 15.08.22)

Pusch, A., Ubben, M., Laumann, D., Heinicke, S. & Heusler, S. (2021). Real-time data acquisition using Arduino and phyphox: measuring the electrical power of solar panels in contexts of exposure to light in physics classroom. *Physics Education, 56 (4)*, 045001. doi: 10.1088/1361-6552/abe993

Pusch, A. (im Druck). Wie beginne ich mit dem Arduino? *MNU-Journal*.

Staacks S., Hütz, S.,Heinke, H. & Stampfer, C. (2018). Advanced tools for smartphone-based experiments: phyphox. *Physics Education, 53 (4)*, 045009. doi: 10.1088/1361-6552/aac05e

Dominik Dorsel ist Doktorand am 2. Physikalischen Institut A der RWTH Aachen.

Dr. Sebastian Staacks entwickelt die App „phyphox“ am 2. Physikalischen Institut A der RWTH Aachen.

Maximilian Loch ist wissenschaftlicher Mitarbeiter am Institut für Didaktik der Physik an der Westfälischen Wilhelms-Universität Münster.

Dr. Alexander Pusch lehrt und forscht am Institut für Didaktik der Physik an der Westfälischen Wilhelms-Universität Münster.

Kontakt über
staacks@physik.rwth-aachen.de und alexander.pusch@wwu.de