

## Arbeitsblatt: Parksensor

### Beschreibung

Jeder kennt sie, fast jeder hat sie und niemand will sie mehr missen: Die Einparkhilfe oder der Einparkassistent heutiger Fahrzeuge macht das Einparken zum Kinderspiel bzw. befördert das Fahrzeug sogar autonom in die freie Parklücke. Nahezu alle auf dem Markt befindlichen Parkassistenten basieren dabei auf derselben Technik zur Distanzerfassung durch Ultraschallsensoren. Auch das Arduino Car soll nun um eine einfache Einparkhilfe ergänzt werden, sodass eine Warnmeldung (LED) beim Unterschreiten eines bestimmten Abstandes angezeigt wird.

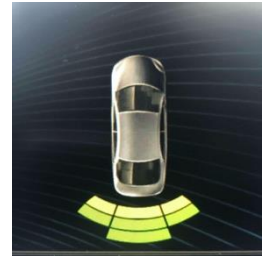


Foto: R. P. Dröge

### 1. Ultraschallsensor zur Abstandsmessung

Damit am Arduino Car ein geeigneter Ultraschallsensor zur Abstandsmessung implementiert und programmiert werden kann, müssen vorab einige physikalische Grundlagen bezüglich der Ausbreitung von Schallwellen bekannt sein.



#### a) Schallgeschwindigkeit und Echo

Schallwellen, Töne und Geräusche breiten sich mit der sogenannten Schallgeschwindigkeit aus (siehe Video im QR-Code). Die Schallgeschwindigkeit ist abhängig vom Medium und der Temperatur, in dem/der sich der Schall ausbreitet. Beispielsweise beträgt die Schallgeschwindigkeit bei 20 °C in Luft ca. 343 m/s und in Wasser 1 484 m/s. Es gilt:

$$v = \frac{s}{t} \quad v - \text{Geschwindigkeit}; s - \text{Strecke}; t - \text{Zeit}$$

Demnach legt der Schall in Luft innerhalb einer Sekunde eine Strecke von 343 m zurück, denn:

$$v = \frac{s}{t} \Rightarrow s = v \cdot t = 343 \frac{\text{m}}{\text{s}} \cdot 1 \text{s} = 343 \text{ m}$$

Trifft die von einer Quelle ausgesendete Schallwelle bei ihrer Ausbreitung in Luft auf ein anderes Objekt, wie zum Beispiel eine Wand, so wird ein Teil der Schallwelle absorbiert und ein Teil reflektiert – also zurückgeworfen. Erreicht der reflektierte Teil der Schallwelle wieder die Quelle, so kann dies als sogenanntes Echo wahrgenommen werden.

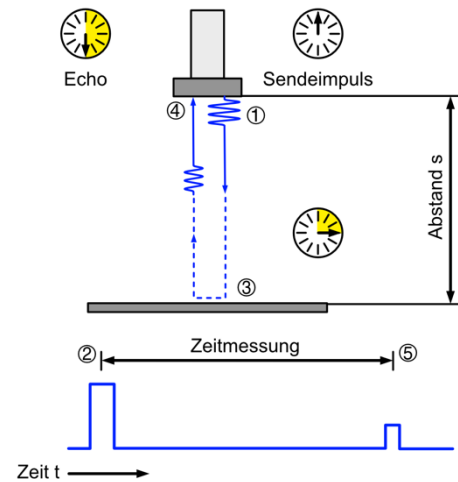


Video: „Schallgeschwindigkeit erklärt“

## b) Funktionsweise eines Ultraschallsensors

Durch den Ultraschallsensor, bestehend aus einer Sende- und einer Empfangseinheit, kann die Laufzeit der Schallwelle gemessen und damit die Entfernung zwischen Sensor und Objekt berechnet werden:

- (1) Der Sensor sendet einen kurzen Impuls aus.
- (2) Die Zeitmessung wird gestartet.
- (3) Die Schallwelle wird reflektiert.
- (4) Das Echo wird vom Sensor erkannt.
- (5) Die Zeitmessung wird gestoppt.
- (6) Aus dem Messwert kann der Abstand errechnet werden (vgl. Berechnungsformel).

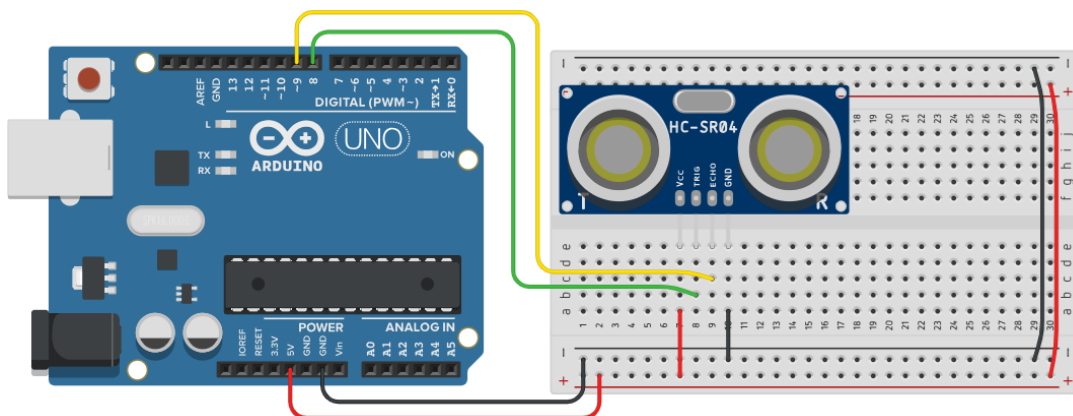


Quelle: BSZ Bietigheim

## 2. Realisierung des Parksensors mit dem Arduino

### a) Hardwarekonfiguration

Starte ein neues Projekt und entnehme zuerst den Abstandssensor (4-Pin/HC-SR04) aus dem Teilekatalog und beschalte es wie nachfolgend abgebildet.



Quelle: BSZ Bietigheim

### b) Einbindung des Ultraschallmoduls in den Programmcode



Zuerst müssen für die Verwendung des Ultraschallmoduls einige spezifische Variablen festgelegt werden:

```
int trigger=8; //Trigger-Pin des Ultraschallsensors an Pin8 des Arduino-Boards. An diesem Pin wird durch einen kurzen Impuls die Schallwelle ausgesandt.
```

```
int echo=9; //Echo-Pin des Ultraschallsensors an Pin9 des Arduino-Boards. Über diesen Pin wird die reflektierte Schallwelle, das Echo, erfasst.
```

```
long dauer=0; // Das Wort „dauer“ ist jetzt eine Variable, unter der die Zeit gespeichert wird, die eine Schallwelle bis zur Reflektion und zurück benötigt. Startwert ist 0.
```

`long entfernung=0;` // Das Wort „entfernung“ ist jetzt die Variable, unter der die berechnete Entfernung gespeichert wird. Info: Anstelle von „int“ steht hier vor den beiden Variablen „long“. Das hat den Vorteil, dass eine größere Zahl gespeichert werden kann. Nachteil: Die Variable benötigt mehr Platz im Speicher.

`int [ ]=10;` // Hier muss auch noch ein geeigneter Pin für die Ansteuerung der „park\_LED\_1“ festgelegt werden

`void setup()`

`{`

`pinMode(trigger, OUTPUT);` // Der Trigger-Pin ist ein Ausgang.

`pinMode(echo, INPUT);` // Der Echo-Pin ist ein Eingang.

`[ ]` // Der park\_LED\_1-Pin ist ein Ausgang.

`}`

### c) Ansteuerung und Auswertung des Ultraschallmoduls im Programmteil loop()

Als Basis für unser Abstandsproblem müssen wir wieder auf die if-else-Anweisung zurückgreifen. Zudem wird der spezielle Befehl `pulseIn(A, B)` benötigt. Mit `pulseIn(pin, B)` wird die Zeit gemessen, welche vergeht, bis am Eingang „pin“ der Signalzustand B (HIGH oder LOW) anliegt.



Vervollständige die nachfolgenden Programmzeilen für den Fall, dass die Warnmelde-LED des Parkassistenten leuchtet, sobald ein Abstand von 30 cm unterschritten wird.



`void loop()`

`{`

`digitalWrite([ ], [ ]);` // Hier nimmt man die Spannung vom Trigger-Pin (Tonsignal ausgeschaltet), damit sichergestellt ist, dass aktuell kein Tonsignal ausgegeben wird.

`delay(5);` // Dauer: 5 Millisekunden

`digitalWrite(trigger, HIGH);` // Jetzt sendet man eine Ultraschallwelle (Tonsignal) los.

`[ ];` // Dieser „Ton“ erklingt für 10 Millisekunden. Hinweis: Ultraschalltöne können Menschen nicht hören.

`digitalWrite([ ], [ ]);` // Dann wird der „Ton“ abgeschaltet.

`dauer = pulseIn(echo, HIGH);` // Jetzt wird mit dem Befehl „pulseIn“ die Zeit in Mikrosekunden erfasst, bis die Schallwelle zum Ultraschallsensor zurückkehrt.

`entfernung = 0.0001*dauer*343/2;` // Berechnung des Abstandes mit der Formel  $s = v \cdot t / 2$

Hinweis: Da die Zeit in Mikrosekunden erfasst und auch so verwendet wird, wäre das Ergebnis der Berechnung normalerweise in Mikrometern. Durch den Faktor 0.0001 wird das Ergebnis angepasst und in cm umgerechnet.

`if ([ ])` // Wenn der Abstand kleiner ist als 60 cm

```
{  
    [ ] //park_LED_1 einschalten  
}  
Else  
{  
    [ ] //park_LED_1 ausschalten  
}}
```

Jetzt kannst Du Dein Programm in TinkerCAD testen!



### Für Profis:

Ändere das Programm so ab, dass der Abstand mithilfe von drei LEDs angezeigt wird.

Stufe 0: Entfernung größer 150 cm	000	X: LED „AN“	0: LED „AUS“
Stufe 1: Entfernung kleiner 150 cm	X00		
Stufe 2: Entfernung kleiner 90 cm	XX0		
Stufe 3: Entfernung kleiner 30 cm	XXX		