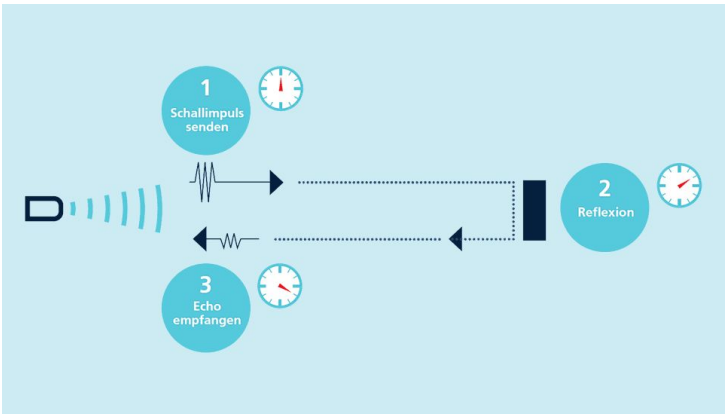
Einparkhilfe mit einem Ultraschallsensor

Bewegt sich ein Objekt auf den Entfernungsmesser zu, soll je nach dem noch verfügbaren Abstand zum Entfernungsmesser ein akustisches und optisches Signal gegeben werden.

Der Ultraschallsensor arbeitet nach einem einfachen Prinzip:



https://www.microsonic.de/de/service/ultraschallsensoren/prinzip.htm

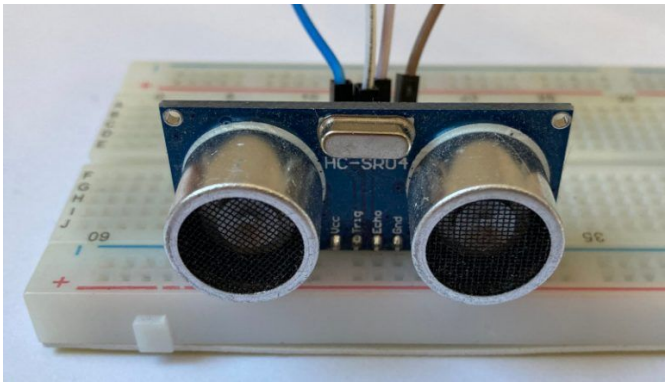
In regelmäßigen Abständen wird eine Schallwelle gesendet. Trifft sie auf einen Gegenstand wird die Schallwelle reflektiert und gelangt als Echo zurück. Mithilfe der Zeitspanne zwischen dem Aussenden des Signals und dem Wiedereintreffen lässt sich die Entfernung berechnen.

Der Sender schickt das Signal und es wird vom Echo empfangen.
Die Geschwindigkeit des Schalls in der Luft beträgt 343,2 m/s.
Der Arduino bestimmt die Zeit in Millisekunden (1 Sekunde = 1000 Millisekunden), deshalb muss der Wert in cm pro Millisekunde berechnet werden:

Umrechnung in cm:	343,2 m • 100 = 34.320 cm
Strecke pro Sekunde:	34.320 : 1.000 = 34,32 cm
Strecke pro Millisekunde:	34,3 : 1.000 = 0,03432 cm

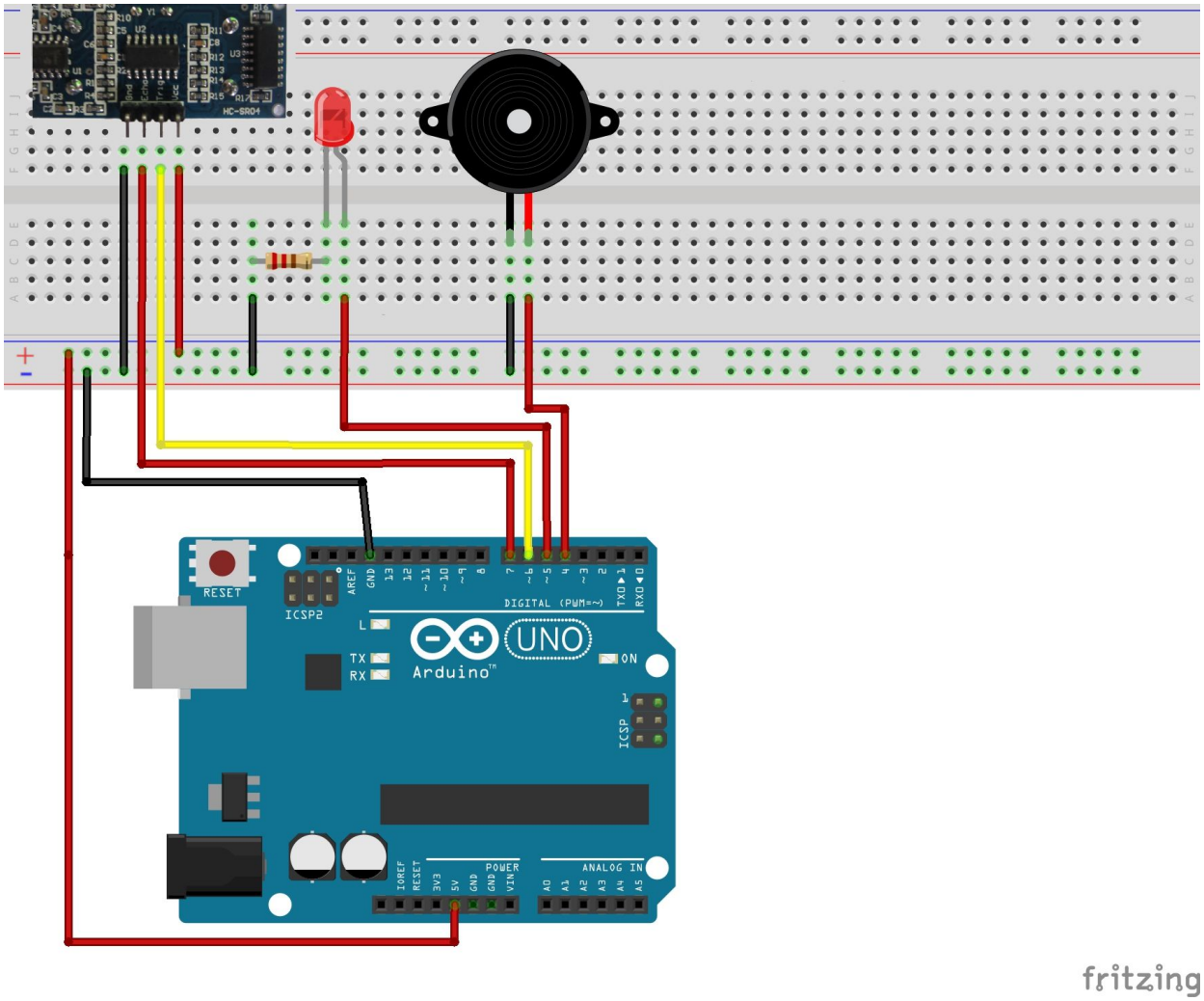
Benötigte Bauteile:

- rote LED
- Widerstand > 100 Ω
- Ultraschallsensor HC-SR04
- Leitungsdrähte



Damit der Ultraschallsensor ordnungsgemäß arbeiten kann, musst du ihn so einstecken, dass Sender und Empfänger nach vorn zeigen, damit das Signal ungehindert gesendet und empfangen werden kann. Die Leitungsdrähte werden auf der Rückseite eingesteckt.

Baue die Schaltung auf.



Der Ultraschallsensor misst die Entfernung zum Hindernis.
Je kleiner der Abstand wird, desto schneller blinkt die LED und die Abstände zwischen den Tonsignalen werden kleiner.

Abstand zum Entfernungsmesser in cm	Abstand zwischen den Signalen in Millisekunden
13 bis 15	700
11 bis 12	500
9 bis 10	300
7 bis 8	200
5 bis 6	100
2 bis 4	0

Kleinere Abstände als 2 cm können nicht gemessen werden.

Im Kopf werden die Variablen definiert. Da die switch-Abfrage sich nicht auf einzelne Werte, sondern auf einen Wertebereich bezieht, muss die Bibliothek `stdio.h` eingebunden werden.

Außerdem soll die Zuweisung der Variablen mit `define` erfolgen.

```
// wird für switch ... case bei der Abfrage von Wertebereichen benötigt
# include <stdio.h>

# define LAUTSPRECHER 4
# define LED 5
# define SENDEN 6
# define ECHO 7
```

Der setup-Teil definiert die Bauteile als Eingabe-/Ausgabegeräte.

```
void setup()
{
  pinMode(SENDEN, OUTPUT);
  pinMode(ECHO, INPUT);
  pinMode(LAUTSPRECHER, OUTPUT);
  pinMode(LED, OUTPUT);
}
```

Die Funktion `EntfernungMessen()` misst die Entfernung des Objekts zum Entfernungsmesser.

```
int EntfernungMessen()
{
  long Entfernung = 0;

  // Sender kurz ausschalten um Störungen des Signal zu vermeiden
  digitalWrite(SENDEN, LOW);
  delay(5);

  // Signal senden
  digitalWrite(SENDEN, HIGH);
  delayMicroseconds(10);
  digitalWrite(SENDEN, LOW);
```

```
  // pulseIn -> Zeit messen, bis das Signal zurückkommt
  long Zeit = pulseIn(ECHO, HIGH);

  // Entfernung in cm berechnen
  Entfernung = (Zeit / 2) * 0.03432;
  return Entfernung;
}
```

Der loop-Teil sorgt dafür, dass je nach gemessener Zeit der Lautsprecher im Intervall piepst und die LED im gleichen Intervall kurz aufleuchtet.

```
void loop()
{
  // Funktion aufrufen
  long Entfernung = EntfernungMessen();

  // versuchen Messfehler auszuschließen
  if (Entfernung < 100)
  {
    switch (Entfernung)
    {
      case 13 ... 15:
        tone(LAUTSPRECHER, 1000, 2);
        digitalWrite(LED, HIGH);
        delay(50);
        digitalWrite(LED, LOW);
        delay(500);
        break;

      case 11 ... 12:
        tone(LAUTSPRECHER, 1000, 2);
        digitalWrite(LED, HIGH);
        delay(50);
        digitalWrite(LED, LOW);
        delay(400);
        break;

      case 9 ... 10:
        tone(LAUTSPRECHER, 1000, 2);
        digitalWrite(LED, HIGH);
        delay(50);
        digitalWrite(LED, LOW);
        delay(300);
        break;

       case 7 ... 8:
         tone(LAUTSPRECHER, 1000, 2);
         digitalWrite(LED, HIGH);
         delay(50);
         digitalWrite(LED, LOW);
         delay(200);
         break;
```

```
      case 5 ... 6:
        tone(LAUTSPRECHER, 1000, 2);
        digitalWrite(LED, HIGH);
        delay(50);
        digitalWrite(LED, LOW);
        delay(100);
        break;

      case 2 ... 4:
        tone(LAUTSPRECHER, 1000);
        digitalWrite(LED, HIGH);
        break;
    }
  }
}
```