Handreichung PhC

1. Kurzvorstellung

Entfernungsmessung mit Ultraschallsensor am Arduino

2. Einordnung in die Lehrpläne

→ am besten im Gymnasium Klasse 10 (mit mehr Hilfen) oder Grundkurs (auch BGy) bzw. GTA (4-6 h)
→ außerdem **naturwissenschaftlicher Profilbereich Klasse 9/10 LB 2** – Messen, Steuern, Regeln – fächerübregreifend mit Physik

OS	Gym	BGy
Klasse 8: LB 2 – Informationen verarbeiten: Modell – Algorithmus – Lösung - Programmstrukturen - Kennen des Problemlöseprozesses - Übertragen der Kenntnisse zum Problemlöseprozess	PHYSIK Klasse 7: WB 2 - Elektrische Schaltungen Beherrschen des Aufbaus elektrischer Schaltungen - Bauen einer Anwendung	
Klasse 9/10: LB 2 – Daten darstellen: Informatikprojekte (9) / Arbeit in Projekten (10) - Gestalten eines eigenen Projekts	Klasse 9/10: LB 4 – Algorithmen und Programme - algorithmische Grundstrukturen - Anwenden der Phasen des Problemlöseprozesses WB 1: Messen, Steuern, Regeln mit Informatiksystemen **naturwissenschaftliches Profil LB 2:** Messen, Steuern, Regeln	
	Grundkurs LB 8B: Technische Informatik – Hardware und Prozessdatenverarbeitung - Messen - Steuern - Anwenden der Kenntnisse über die Ansteuerung paralleler und serieller Schnittstellen	Grundkurs LB 3 – Algorithmen und Programme - Grundstrukturen LB 4B – Projekt Technische Informatik - Gestalten eines Projektes zur Steuerung und Regelung von Prozessen

3. Lernziele

Die folgenden Lernziele sind auf das Gymnasium bezogen.

kognitive Lernziele: Die Schüler:innen …

- wenden die algorithmischen Grundstrukturen Sequenz, Selektion und Zyklus an. (Lehrplan S. 15, Bildungsstandards S. 19)

<Autor>

- wenden arithmetische und logische Operatoren an. (Bildungsstandards S. 15)
- verwenden Variablen und Wertzuweisungen. (Bildungsstandards S. 16)
- erweitern bestehende Informatiksysteme mit Soft- und Hardwarekomponenten (Bildungsstandards S. 17)
- zerlegen ein Problem in Teilprobleme. (Lehrplan S. 12)
- kennen die praktische Nutzung von Informatiksystemen auf dem Gebiet der Automatisierung am Beispiel von Sensoren. (Lehrplan S. 15)
- **Grundkurs:** … erhalten Einblick in die Prozessdatenverarbeitung zum Messen und Steuern. (Lehrplan S. 21)
- wenden Kenntnisse über die Ansteuerung serieller Schnittstellen an. (Lehrplan S. 21)

psychomotorische Lernziele: Die Schüler:innen …
- realisieren eine Schaltung mit einfachen Bauelementen.

affektive Lernziele: Die Schüler:innen …
- kommentieren automatisierte Vorgänge und beurteilen deren Umsetzung. (Bildungsstandards S. 18)
- beurteilen ein Modell, die Implementierung und die verwendeten Werkzeuge kritisch. (Bildungsstandards S. 19)
- werden aufmerksam auf die praktische Nutzung von Informatiksystemen auf dem Gebiet der Automatisierung am Beispiel von Sensoren.
- kooperieren in Projektarbeit bei der Bearbeitung eines informatischen Problems. (Lehrplan S. VII, Bildungsstandards S. 21)

4. Voraussetzungen

Fachliche Voraussetzungen
- Die Schüler:innen kennen die algorithmischen Grundstrukturen.
- Die Schüler:innen kennen den Problemlöse-Prozess.
- Physik, Kl. 6: Beherrschen des Aufbaus von Stromkreisen nach Schaltplänen
- Physik Kl. 9: Kennen der physikalischen Größe elektrischer Widerstand
- vorherige Einführung in Arduino, z.B. Vorführen/gemeinsames Bauen und Programmieren einer einfachen Schaltung wie einer blinkenden LED

Technische Voraussetzungen
- Arduino IDE oder Internetzugang und create.arduino.cc/editor

Materielle Voraussetzungen
- Computer
- Arduino MEGA
- USB-Kabel zum Arduino
- Steckbrett
- Ultraschallsensor (HC-SR04)
- Taster
- Ampel
- Piezo-Lautsprecher
- mind. 10 Steck-Kabel

5. Kurzdarstellung

praktische Hinweise

- für jedes Team einen Arduino und eine Kiste mit den notwendigen Bauteilen vorbereiten
- bei bestehender USB-Verbindung sofortiges Ausprobieren des Codes möglich
- beliebige Erweiterungen möglich: Tutorials online, billige Bauteile
- interdisziplinär mit Physik zusammenarbeiten, z.B. zum Thema Ultraschall oder zum Thema Schaltungen – ggf. dann auch Schaltplan zeichnen lassen
- Arduino-Schaltungen sind auch mit Tinkercad erstellbar, dort gibt es allerdings keine Ampel (man könnte die RGB-LED als Ersatz nutzen, die auch 4 Anschlüsse hat)
- Schwierigkeiten liegen meist im korrekten Aufbau der Schaltung – alle Elemente am besten immer einzeln testen, bei Schaltungen schnell helfen

Mögliche Aufgabenstellung

Hintergrund: Um das Einparken von Autos zu erleichtern, bieten einige integrierte Einparkhilfe an, die auf Ultraschall basieren. Diese gehen meist automatisch an, wenn das Auto rückwärts fährt und messen über Ultraschall die Entfernung zu anderen Autos, Mauern, Straßenlaternen usw. Einfache Ausführungen piepen umso aufdringlicher, umso näher das Auto einem anderen Gegenstand kommt. Andere geben auf einem Bildschirm mit Farben oder Symbolen Signale.

Zielsetzung: Baut mit einem Arduino, einem Ultraschallsensor und einer Ampel einen eigenen Entfernungsmesser. Dieser Ampel soll auf grün stehen und wenn die Entfernung geringer als 50cm ist auf gelb umschalten, bei weniger als 30cm auf rot.

Weitere Einsatzmöglichkeiten

Der Entfernungsmesser kann mit kleinen Abänderungen z.B. auch

- mit einer größeren LED-Anlage an einer Garagenwand befestigt werden und so beim Einparken in die Garage helfen.
- in einem Parkhaus anzeigen, welche Plätze belegt sind.
- einem Roboter beim Ausweichen helfen.
- als Bewegungsmelder in einer Tür installiert sein.
- mit Display als Maßband-Ersatz dienen.

Projektgliederung: Implementiert die Aufgabe in einzelnen Schritten. Holt euch nach jedem Schritt Feedback, bevor ihr weiterarbeitet.

1. Zerlegt das Projekt in Teilaufgaben, die ihr auf dem Arduino einzeln testen könnt. Schreibt diese einzeln auf und notiert euch Reihenfolge und Einzelheiten zur Umsetzung.
2. Baut die einzelnen Elemente am Steckbrett und Arduino zusammen.
3. Programmiert die Teilprojekte in der Arduino IDE und testet sie nacheinander. *Hinweis: Nutzt die serielle Ausgabe, um den Ultraschallsensor zu testen.*

Zusatzaufgaben zur Auswahl:

A. Baut einen Piezo-Lautsprecher dazu, der umso schneller piept, umso geringer die Entfernung ist.
B. Baut einen Schalter dazu, der erst gedrückt werden muss, bevor das restliche Programm abläuft.
C. Modularisiert das Programm, indem ihr Code-Teile in Unterprogramme auslagert.

Hintergrundinformationen zum Ultraschall-Sensor für die Schüler:innen

Ultraschall-Sensor allgemein

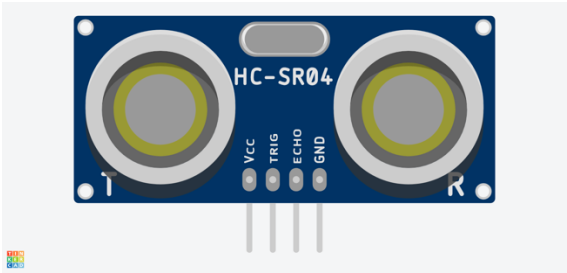
Ultraschall an sich bezeichnet Schallwellen, die eine so hohe Frequenz haben, dass sie von Menschen nicht wahrgenommen werden können.

Der Ultraschallsensor sendet per Lautsprecher ein Signal aus. Dieses wird reflektiert und so vom Sensor wieder empfangen. Mit der Zeit, die zwischen Aussenden und Empfangen vergangen ist, kann die Entfernung ermittelt werden. Dies wird aber nicht vom Sensor zurückgegeben, sondern muss erst umgerechnet werden.

Der Ultraschall-Sensor HC-SR04

Der Sensor HC-SR04 kann Entfernungen von 2cm bis 400cm bis auf 3mm genau messen. Als Spannung benötigt er 5V. Er benötigt vier Anschlüsse:

VCC – die +5 Volt-Stromversorgung
TRIG – der "Trigger"-Pin zum Senden der Ultraschallimpulse
ECHO – der "Echo"-Pin zum Empfangen des reflektierten Signals (beim Empfangen „HIGH")
GND – der Minuspol der Stromversorgung



Der Programm-Code

Ihr benötigt die Variable `dauer`.

```
1 digitalWrite(trigger, LOW);
2 delay(5); //Dauer: 5 Millisekunden
3 digitalWrite(trigger, HIGH);
4 delay(10);
5 digitalWrite(trigger, LOW);
6 dauer = pulseIn(echo, HIGH);
```

Zeile 1: Man nimmt die Spannung kurz vom Trigger-Pin, um Rauschen zu vermeiden.
Zeile 3-5: Für 10 ms wird ein „Ton" abgespielt, also Ultraschall ausgesendet.
Zeilt 6: „pulseIn" gibt die Zeit in ms zurück, die der Schall braucht, um nach einer Reflektion zum Sensor zurückzukehren. Die Variable `dauer` solltet ihr vorher schon deklariert haben.
Die Dauer muss noch in Entfernung umgerechnet werden. Benutzt dafür den Term (dauer/2) * 0.03432. Die Dauer wird durch zweigeteilt, weil nur eine Strecke berechnet werden soll, der Schall aber eine Hin- und Rückstrecke hatte. Diesen halbierten Wert wird mit der Schallgeschwindigkeit in cm/ms multipliziert.

Hintergrundinformationen zur seriellen Ausgabe für die Schüler:innen

Ihr müsst die serielle Ausgabe im setup beginnen:

```
void setup() {
    Serial.begin(9600);
    . . .
}
```

In `void loop()` könnt ihr dann mit der Funktion `Serial.print()` schreiben. `Serial.println()` fügt automatisch einen Zeilenumbruch am Ende ein.
Die Ausgabe findet ihr unter „Werkzeuge" → „Serieller Monitor" in der Menüleiste.

Auf dem Arduino blinkt „RX/TX“, wenn serielle Kommunikation stattfindet.

Hintergrundinformationen zum Piezo-Lautsprecher für die Schüler:innen

Für den Piezo-Lautsprecher müsst ihr erst einen Pin, z.B. mit dem Namen ton als Eingabe festlegen. Der zweite Pin des Lautsprechers muss mit der Masse verbunden werden.

Über tone(ton, 100); könnt ihr dann einen Ton abspielen. Der zweite Wert (100) gibt die Tonhöhe an. Mit noTone(ton); wird der Lautsprecher wieder abgeschaltet.

Folgender Programmcode spielt einen Ton 1 Sekunde lang ab:

```
tone(ton, 100);        // Ton wird angeschaltet mit Höhe 100
delay(1000);           // mit einer Dauer von 1 Sekunde
noTone(ton);           // Ton wird abgeschaltet
```

Lösungen

Aufgabe 1

Zerlegt das Projekt in Teilaufgaben, die ihr auf dem Arduino einzeln testen könnt. Schreibt diese einzeln auf und notiert euch Einzelheiten zur Umsetzung.

Teilprojekte:
- Ultraschallsensor
- Ampel

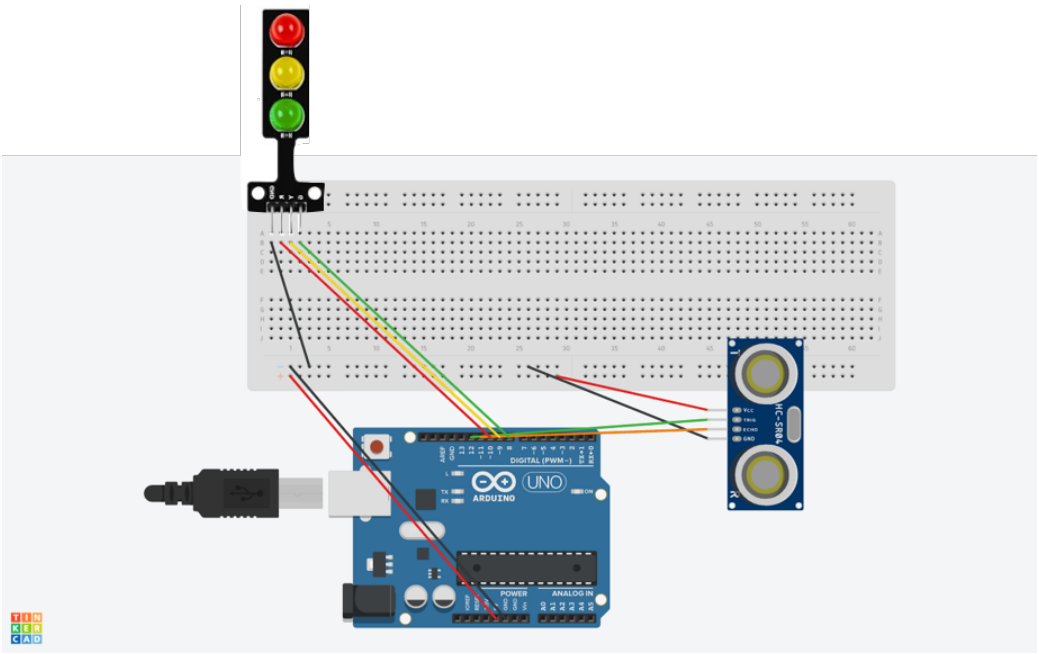
Ultraschallsensor:
- VCC mit Spannung und GND mit Masse verbinden
- TRIG und ECHO an digitale Pins
- TRIG- und ECHO-Pins im Programm in Variablen speichern
- Programmcode ist gegeben

Ampel:
- keine eigene Spannungsversorgung, sondern durch Pins
- GND mit Masse verbinden
- digitale Pins für Rot/Grün/Gelb
- Programmcode mit Verzweigungen, die jeweils verschiedene Farben an-/ausschalten

Aufgabe 2

Baut die einzelnen Elemente am Steckbrett und Arduino zusammen.

Beispiellösung:



Aufgabe 3

Programmiert die Teilprojekte in der Arduino IDE und testet sie nacheinander.
Hinweis: Nutzt die serielle Ausgabe, um den Ultraschallsensor zu testen.

Programmcode für Ultraschallsensor

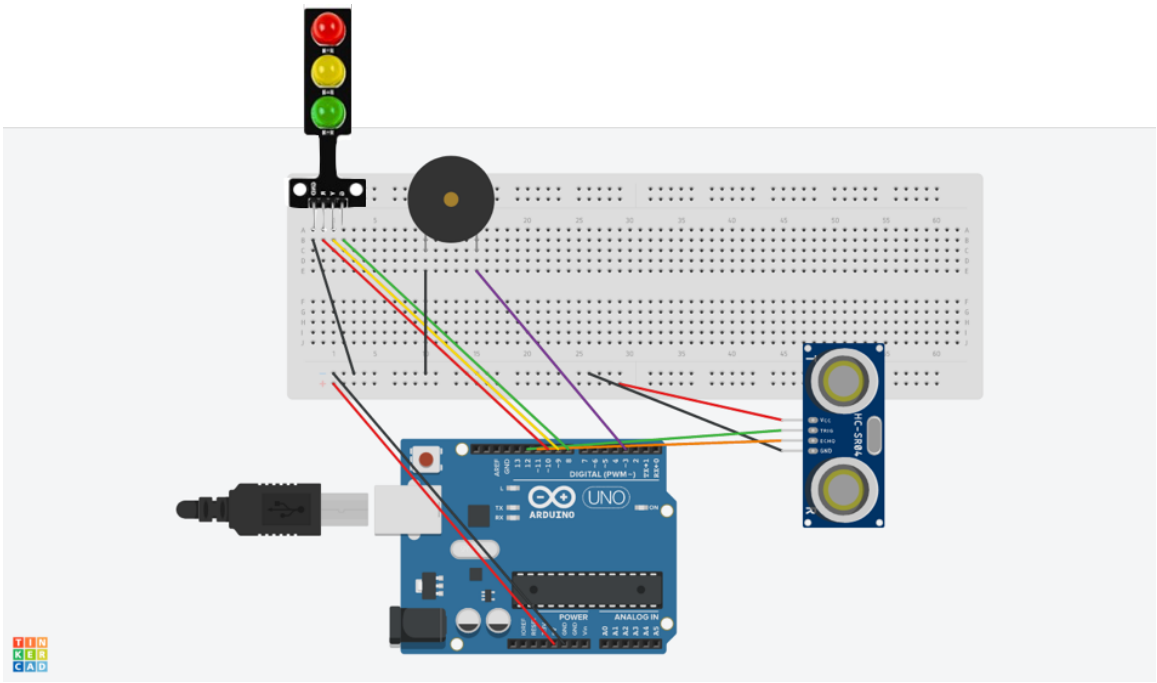
```
int trigger=12;
int echo=11;
long dauer=0;
long entfernung=0;
int rot = 10;
int gelb = 9;
int gruen = 8;

void setup() {
    Serial.begin(9600);
    pinMode(trigger, OUTPUT);
    pinMode(echo, INPUT);
}

void loop() {
  digitalWrite(trigger, LOW);
  delay(5);
  digitalWrite(trigger, HIGH);
  delay(10);
  digitalWrite(trigger, LOW);
  dauer = pulseIn(echo, HIGH);
  entfernung = (dauer/2) * 0.03432
  if (entfernung >= 400 || entfernung <= 2) {
    Serial.println("Kein Messwert");
  }
  else {
    Serial.print(entfernung);
    Serial.println(" cm");
    delay(1000);
  }
}
```

Programmcode mit Ampel

```
int trigger=12;
int echo=11;
long dauer=0;
long entfernung=0;
int rot = 10;
int gelb = 9;
int gruen = 8;

void setup() {
    Serial.begin(9600);
    pinMode(trigger, OUTPUT;
    pinMode(echo, INPUT);
    pinMode(rot, OUTPUT);
    pinMode(gelb, OUTPUT);
    pinMode(gruen, OUTPUT);
}

void loop() {
  digitalWrite(trigger, LOW);
  delay(5);
  digitalWrite(trigger, HIGH);
  delay(10);
  digitalWrite(trigger, LOW);
  dauer = pulseIn(echo, HIGH);
  entfernung = (dauer/2) * 0.03432
  if (entfernung >= 400 || entfernung <= 2) {
    Serial.println("Kein Messwert");
  }
  else {
    Serial.print(entfernung);
    Serial.println(" cm");
    if (entfernung > 50) {
      digitalWrite(rot, LOW);
      digitalWrite(gelb, LOW);
      digitalWrite(gruen, HIGH);
    }
    else if (entfernung > 30) {
      digitalWrite(rot, LOW);
      digitalWrite(gelb, HIGH);
      digitalWrite(gruen, LOW);
    }
    else {
      digitalWrite(rot, HIGH);
      digitalWrite(gelb, LOW);
      digitalWrite(gruen, LOW);
    }
  }
  delay(1000);
}
```

Zusatzaufgabe A

Baut einen Piezo-Lautsprecher dazu, der umso schneller piept, umso geringer die Entfernung ist.



Möglichkeit 1 ist, dass du die Lautsprecher mit der Ampel koppelst (am besten erst ab gelb beginnend) und stufenweise die Zeitabstände zwischen den Tönen verkürzt.

Möglichkeit 2 ist, dass der Lautsprecher den Ton abspielt und das delay-dazwischen mit der Entfernung multipliziert wird - umso weiter die Entfernung, umso länger ist die Dauer zwischen den Tönen. Die beiden Größen sind dadurch linear abhängig.

Hinweis: Mit der Länge der Töne variiert auch die Abtastrate des Ultraschallsensors!

<u>Möglichkeit 1</u>

```
int ton = 3;
void setup()
{
    . . .
    pinMode(ton, OUTPUT);
}


void loop()
{
  . . .
  else {
    . . .
    if (entfernung > 50) {
      . . .
   }
    else if (entfernung > 30) {
      . . .
      tone(ton, 100);
      delay(500);
```

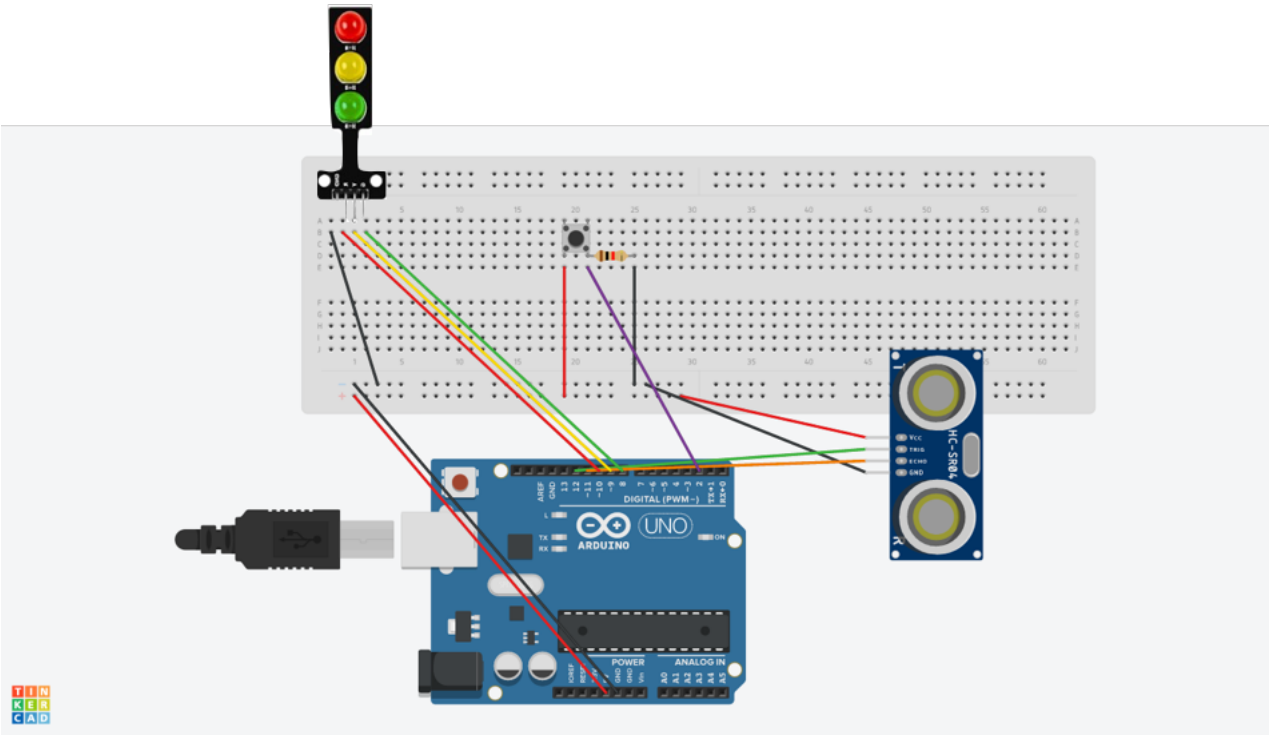
```
      noTone(ton);
       delay(500);

    }
    else {
      tone(ton, 100);
      delay(200);
      noTone(ton);
      delay(200);

    }
  }
  delay(1000);
}
```

Möglichkeit 2

```
int ton = 3;
void setup()
{
    . . .
    pinMode(ton, OUTPUT);
}


void loop()
{
  . . .
  else {
    . . .
    if (entfernung > 50) {
      . . .
    }
    else if (entfernung > 30) {
      . . .
    }
    else {
      . . .
    }
    if (entfernung < 30) {
      tone(ton, 100);
      delay(entfernung*10); // mit einer Dauer von 1 Sekunde
      noTone(ton); // Der Ton wird abgeschaltet
      delay(entfernung*10); // Der Lautsprecher bleibt eine Sekunde aus
    }
     else {
       delay(1000);
     }
  }
}
```

Zusatzaufgabe B

Baut einen Schalter dazu, der erst gedrückt werden muss, bevor das restliche Programm abläuft.



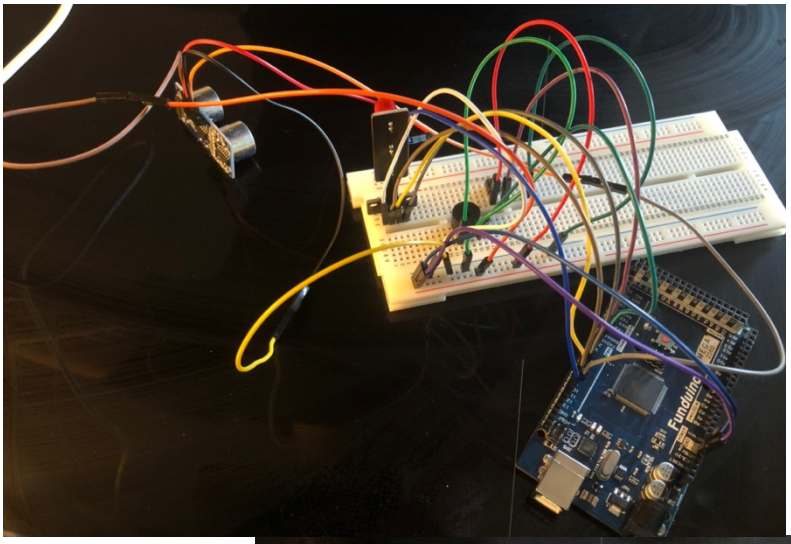

```
. . .

int taster = 2;
int tasterstatus = 0;

void setup()
{
    . . .
    pinMode(taster, INPUT);
}


void loop()
{
  tasterstatus = digitalRead(taster);
  if (tasterstatus == HIGH) {
    while (true) {
      . . .
      if (entfernung >= 400 || entfernung <= 0) {
        . . .
      }
      else {
        . . .
      }
    }
  }
}
```

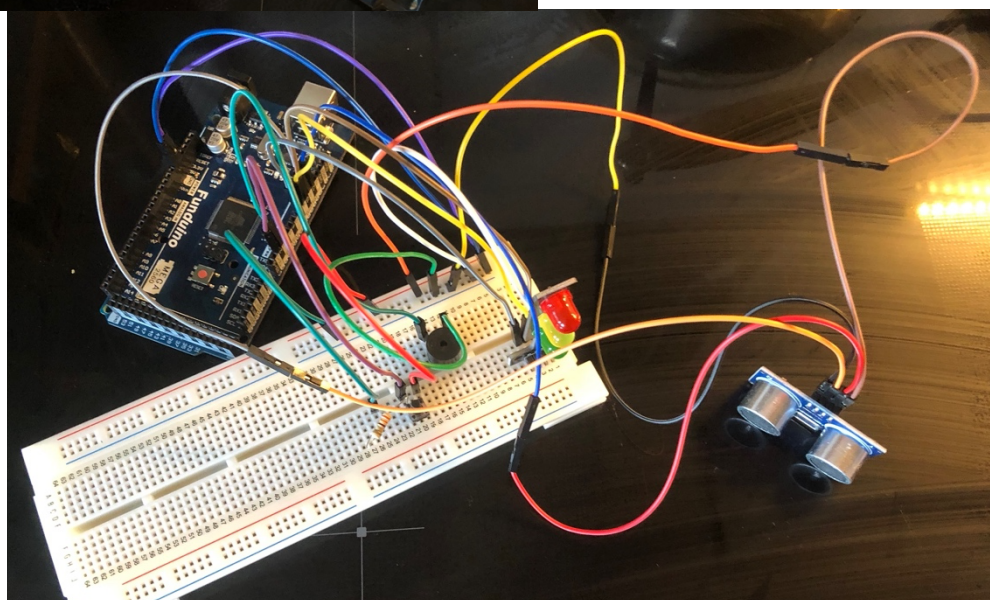
Zusatzaufgabe C

Modularisiert das Programm, indem ihr Code-Teile in Unterprogramme auslagert.

Ihr könnt die Ampel-Schaltung oder das Messen der Entfernung als eigene Funktionen definieren. Diese Implementationen sind nur Beispiele - vielleicht fallen euch besser Möglichkeiten der Modularisierung ein!

```
void setup() { . . . }

void roteAmpel() {
  digitalWrite(rot, HIGH);
  digitalWrite(gelb, LOW);
  digitalWrite(gruen, LOW);
}

void gelbeAmpel() {
  digitalWrite(rot, LOW);
  digitalWrite(gelb, HIGH);
  digitalWrite(gruen, LOW);
}
void grueneAmpel() {
  digitalWrite(rot, LOW);
  digitalWrite(gelb, LOW);
  digitalWrite(gruen, HIGH);
}

long getEntfernung() {
  digitalWrite(trigger, LOW);
  delay(5);
  digitalWrite(trigger, HIGH);
  delay(10);
  digitalWrite(trigger, LOW);
  dauer = pulseIn(echo, HIGH);
  return (dauer/2) * 0.03432;
}

void loop()
{
  entfernung = getEntfernung();
  if (entfernung >= 400 || entfernung <= 0) {
    . . .
  }
  else {
    . . .
    if (entfernung > 50) {
      grueneAmpel();
    }
    else if (entfernung > 30) {
      gelbeAmpel();
    }
    else {
      roteAmpel();
    }
  }
  delay(1000);
}
```





Quellen

Das Projekt wurde anhand https://funduino.de/nr-10-entfernung-messen erstellt.

Hilfreich waren auch die Seiten

- https://www.kreativekiste.de/ultraschall-abstandsmessung-einparkhilfe-arduino
- https://www.mymakerstuff.de/2016/05/24/arduino-tutorial-der-ultraschallsensor/

Zuletzt abgerufen wurden die Seiten am 28.01.2022.

Gesellschaft für Informatik (GI) e.V. (2008): Bildungsstandards Informatik für die Sekundarstufe I. Beilage zu LOG IN 2008, 28(150/151).

Sächsisches Staatsministerium für Kultus (2019): Lehrplan Gymnasium. Informatik. Dresden: Comenius-Institut.