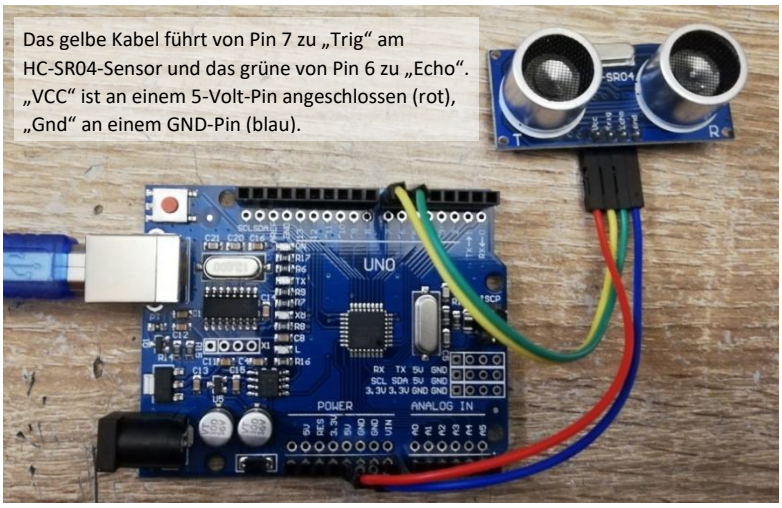
Ein Arduino-Projekt für das Gruselkabinett oder Halloween

Da steht ein bunter Ding auf dem Tisch mit einem Gesicht und der Aufschrift: „Schau mir in die Augen!“ Du gehst näher dran , ein rotes Licht geht an und je dichter du dem Gesicht kommst, desto tiefer wird der Ton. Das könntest du als Halloween-Gag nutzen …

Wir kümmern uns hier um das Innenleben: Ein Arduino-Board, ein Ultraschall-Entfernungssensor eine LED mit Widerstand, ein kleiner Lautsprecher. Das ist alles für knapp zehn Euro erhältlich. Außerdem brauchst du einen Computer, auf dem die Arduino-Entwicklungsumgebung installiert ist. Und du musst wissen, wie man damit arbeitet. Es gibt im Netz viele Anleitungen dazu.

I. Der Abstandssensor

Zuerst bringen wir den Abstandssensor „HC-SR04“ am Arduino zum Laufen. (Eigentlich sollte man „Abstandssensor –Modul“ reden, weil es sich um eine komplette komplizierte Schaltung handelt.) Die beiden scheinbaren „Augen“ des Moduls sind in Wirklichkeit ein Ultraschall-Lautsprecher und ein Mikrofon. Der Lautsprecher sendet kurze Ultraschallsignale aus, Gegenstände reflektieren diesen Schall, das Mikrofon fängt das Echo auf. Der Arduino berechnet dann mit dem Programm, das wir gleich sehen werden, aus dem Zeitunterschied zwischen Trigger-Output und „pulse_In“-Input und der Schallgeschwindigkeit den Abstand. Das Ergebnis teilt er uns über den seriellen Monitor mit, der ein Teil der Arduino-Entwicklungsumgebung ist.



Oben siehst du, wie der Abstandssensor (Modell HC-SR04) an den Arduino angeschlossen werden muss, rechts den Sketch (das Programm), das über die Arduino-Entwicklungsumgebung hochgeladen werden muss.

Die Zeile „abstand = (dauer/2) * 0.034;“ dient der Berechnung des Abstands : Die Dauer zwischen dem Aussenden des Trigger-Signals und dem Erfassen („pulseIn“) des Echos wird mit dem Wert für die Schallgeschwindigkeit (in cm pro Sekunde) multipliziert.

ArduinoUltraSchall | Arduino 1.8.13

Datei Bearbeiten Sketch Werkzeuge H

ArduinoUltraSchall

```
int trigger=7;
int echo=6;
long dauer=0;
long abstand=0;

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);
abstand = (dauer/2) * 0.03432;
Serial.println("Abstand:");
delay(500);
```

Der trigger-Pin, hier Arduino-Pin 7, bringt das Signal zum Sensor-Modul.
Das echo-Signal wird als Puls mit dem „pulsIn“ -Befehl an Pin 6 des Arduinos erfasst.
Zeitangaben werden beim Arduino als Millisekunden gemacht: „delay(5)“ bedeutet eine Pause von 5 Millisekunden.
Der Trigger-Pin wird also zunächst mit „LOW“ auf 0 Volt gesetzt, bleibt 5 Millisekunden so, dann wird er 10 Millisekunden mit „HIGH“ auf 5 Volt gesetzt.

II. Das Dimmen des roten Lichts

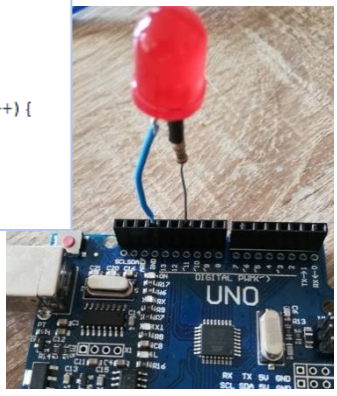
Die digitalen Pins 3, 5, 6, 9, 10 und 11 erlauben es, eine LED zu dimmen. Der Trick: Die LED wird an diesen digitalen Pins, die nur 5 Volt tatsächlich nur ein- und ausschalten können, so zu takten, dass sie dem Augen mehr oder weniger hell erscheinen. Das geht in den Stufen von 0 bis 255, siehe den Sketch „ArduinoLedNurDimmen“. Für die endgültige „Gruselprogrammierung“ werden wir die Helligkeit über die Entfernung steuern, die vom Ultraschallsensor gemessen wird. In beiden Fällen wird der Plusanschluss der Led über einen 150-Ohm-Widerstand an Pin 11 angeschlossen und er Minusanschluss der LED an GND. Rechts die Schaltung und der Sketch zum Dimmen einer roten LED.

ArduinoNurLedDimmen | Arduino 1.8.13

Datei Bearbeiten Sketch Werkzeuge Hilfe

ArduinoNurLedDimmen

```
int ledPin = 11;
int helligkeit =0;
void setup(){
}
void loop(){
 for (int i=0; i <= 255; i++){
    analogWrite(ledPin, i);
    delay(10);
  }
}
```



III. Sound

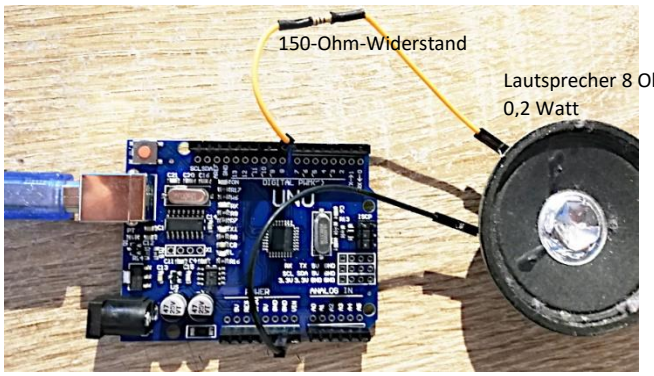
Nun fehlen noch die Töne: Mit dem „tone“-Befehl können über einen digitalen Pin Frequenzen für einen Lautsprecher bereitgestellt werden. Im einfachsten Fall kann das so aussehen:

ArduinoNurSound | Arduino 1.8.13

Datei Bearbeiten Sketch Werkzeuge Hilfe

ArduinoNurSound §

```
int frequenz=0;
void setup(){
}
void loop(){
 for (int i=1; i <= 50; i++){
      frequenz=i*80;
      tone(soundPin,frequenz);delay(100);
      noTone(soundPin);
      }
}
```



Über den digitalen Pin 8 werden - gesteuert durch den Sketch „ArduinoNurSound“ - Frequenzen von 50 bis 4000 Hertz an den einen Anschluss des Lautsprecher gegeben. Ein 150-Ohm-Widerstand in der Zuleitung begrenzt die Stromstärke und stellt die Lautstärke ein. Der zweite Anschluss des Lautsprechers wird an einen GND-Pin angeschlossen. Zur Soundausgabe kann auch ein passiver Piezo-Lautsprecher („Buzzer“)gewählt werden.

IV. Die gesamte „Gruselschaltung“

Abstand, Licht, Sound – die drei Zutaten sind vorbereitet. Nun geht es an eine wirkungsvolle Kombination. Hier wird ein Vorschlag gemacht, den du für deine Zwecke abwandeln und verbessern kannst.

ArduinoGrusel | Arduino 1.8.13

Datei Bearbeiten Sketch Werkzeuge Hilfe

ArduinoGrusel

```
int ledPin = 11;
int soundPin = 8;
int trigger=7;
int echo=6;
long dauer=0;
long abstand=0;
int helligkeit =0;
int frequenz=0;
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);
abstand = (dauer/2)*0.034;
  if (abstand >= 100 || abstand <= 0)
  {Serial.println("Keine Aktion!");delay(500);}
  else
  {Serial.print("Abstand: ");Serial.print(abstand);Serial.println(" cm");
   helligkeit = int(abstand*2.55);
   Serial.print("Helligkeit: ");Serial.println(helligkeit);
   analogWrite(ledPin,helligkeit);delay(800);
   frequenz = abstand*40;
   Serial.print("Frequenz: ");Serial.println(frequenz);
   tone(soundPin,frequenz);delay(200);
   noTone(soundPin);}
}
```

Code der verwendeten Sketche:

ArduinoUltraSchall:

```
int trigger=7;
int echo=6;
long dauer=0;
long abstand=0;
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);
abstand = (dauer/2) * 0.03432;
Serial.println("Abstand:");
delay(500);
}
```

ArduinoNurLedDimmen:

```
int ledPin = 11;
int helligkeit =0;
void setup(){
}
void loop(){
 for (int i=0; i <= 255; i++){
   analogWrite(ledPin, i);
   delay(10);
  }
}
```

ArduinoNurSound:

```
int soundPin = 8;
int frequenz=0;
void setup(){
}
void loop(){
 for (int i=1; i <= 50; i++){
     frequenz=i*80;
     tone(soundPin,frequenz);delay(100);
     noTone(soundPin);
     }
}
```

ArduinoGrusel.ino:

```
int ledPin = 11;
int soundPin = 8;
int trigger=7;
int echo=6;
long dauer=0;
long abstand=0;
int helligkeit =0;
int frequenz=0;
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);
abstand = (dauer/2)*0.034;
  if (abstand >= 100 || abstand <= 0)
  {Serial.println("Keine Aktion!");delay(500);}
  else
  {Serial.print("Abstand: ");Serial.print(abstand);Serial.println(" cm");
   helligkeit = int(abstand*2.55);
   Serial.print("Helligkeit: ");Serial.println(helligkeit);
   analogWrite(ledPin,helligkeit);delay(800);
   frequenz = abstand*40;
   Serial.print("Frequenz: ");Serial.println(frequenz);
   tone(soundPin,frequenz);delay(200);
   noTone(soundPin);}
 }
```