

AZ-Delivery

Willkommen!

Vielen Dank, dass sie sich für unser AZ-Delivery Ultraschallmodul vom Typ "HC-SR04" entschieden haben. In den nachfolgenden Seiten werden wir Ihnen erklären wie Sie das Gerät einrichten und nutzen können.

Viel Spaß!



Az-Delivery

Der Ultraschallsensor "HC-SR04" verwendet Ultraschallwellen, um den Abstände zu Objekten zu bestimmen. Das Modul "HC-SR04" kann berührungslose Messungen in einem Bereich von 20 bis 4000 Millimeter mit einer Genauigkeit von 3 Millimetern durchführen. Jedes "HC-SR04"-Modul verfügt über einen Ultraschall-Sender, -Empfänger und eine elektronische Schaltung.

Der Sensor besteht aus einem Ultraschallsender und einem Empfänger. Der Sensor sendet Ultraschallwellen auf einer bestimmten Frequenz aus. Die Ultraschallwelle durchdringt den Raum, trifft auf ein Hindernis, wird von dem Hindernis zurückgeworfen und dann von einem Empfänger erfasst. Die Geschwindigkeit der Ultraschallwellen in der Luft beträgt etwa 343 m/s, so dass man nur das Zeitintervall vom Senden bis zum Empfang messen muss. Zur Berechnung der Entfernung wird die Geschwindigkeit der Ultraschallwelle in der Luft mit der Hälfte des Zeitintervalls multipliziert. Das Zeitintervall muss durch zwei geteilt werden, weil die Ultraschallwelle zum Hindernis und zurück wandert.

Technische Daten:

- » Stromversorgung und Logikspannungsbereich: 5V
- » Betriebsgleichstrom: 15mA
- » Winkel der Messung: 15°
- » Abstandsbereich: 20mm - 4000mm
- » Praktischer Messabstand : 20mm - 800mm

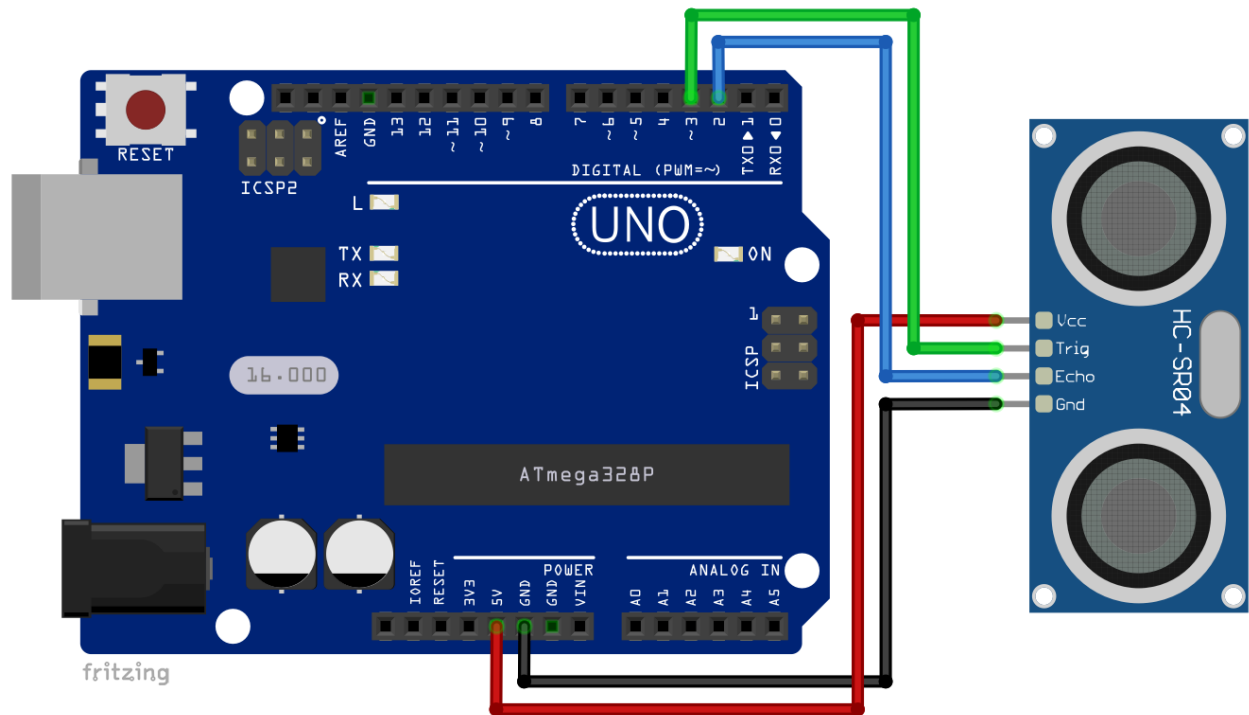
Az-Delivery

Der HC-SR04-Sensor wird sowohl mit Mikrocontroller- als auch mit Mikroprozessor-Plattformen wie dem Uno oder dem Raspberry Pi verwendet. Stromversorgung und Logikspannung betragen 5V. Der Arbeitsstrom beträgt weniger als 15mA und kann direkt über die 5V-Pins versorgt werden. Der Raspberry Pi arbeitet mit 3,3V, so dass wir, um ihn zu verwenden, Spannungen von 5V in 3,3V mit einem "Logic Level Converter" oder "Shifter" umwandeln müssen.

Wenn wir das Modul mit dem Mikrocontroller verbinden, benötigen wir zwei Pins für die Messung: Den Trigger-Pin als Eingangspin und den Echo-Pin als Ausgangspin. Der Trigger-Pin muss für 10us auf den Zustand *HIGH* und dann auf den Zustand *LOW* gesetzt werden. Dadurch wird die Ultraschallwelle übertragen, und der Empfänger wartet darauf, dass die Welle zurückgeworfen wird. Wenn die Welle zurückkehrt, geht der Echo-Pin auf *HIGH*.

Az-Delivery

Verbindung des Moduls mit dem Uno



HC-SR04 Pin	>	Uno Pin
VCC	>	5V
GND	>	GND
ECHO	>	D2
TRIG	>	D3

Roter Draht

Schwarzer Draht

Blauer Draht

Grüner Draht

Az-Delivery

Sketch-Beispiel:

```
const int echoPin = 2;
const int trigPin = 3;
long duration;
int distance;
void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Serial.begin(9600);
}
void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2;
  Serial.print("Distance: ");
  Serial.println(distance);
  delay(1000);
}
```

Az-Delivery

Zu Beginn der Skizze werden zwei Konstanten erstellt, die `echoPin` und `trigPin` genannt werden und deren Werte auf 2 bzw. 3 gesetzt werden. Diese Werte stellen digitale Pins des Uno dar, die mit `echoPin` und `trigPin` des Sensors verbunden sind. Dann werden zwei Variablen erstellt, eine für die Messung der Zeit, genannt `duration`, und die zweite für den berechneten Abstand, genannt `distance`. In der `setup()`-Funktion stellen wir die Pin-Modi der verwendeten Pins ein; der Echo-Pin wird als *INPUT* und der Trig-Pin als *OUTPUT* definiert. Am Ende der `setup()`-Funktion starten wir die serielle Kommunikation mit einer Baudrate von 9600.

In der `loop()`-Funktion setzen wir zuerst den Zustand von `trigPin` auf *LOW*, und warten für 2us. Dann geht der Zustand der Variablen `trigPin` auf *HIGH*, und der Algorithmus wartet für 10us. Danach ändert sich der Zustand von `trigPin` auf *LOW*.

Mit der nächsten Codezeile messen wir das Zeitintervall zwischen der Übertragung und der Erkennung der Ultraschallwelle und speichern es in der Variable `duration`: `duration = pulseIn(echoPin, HIGH);`

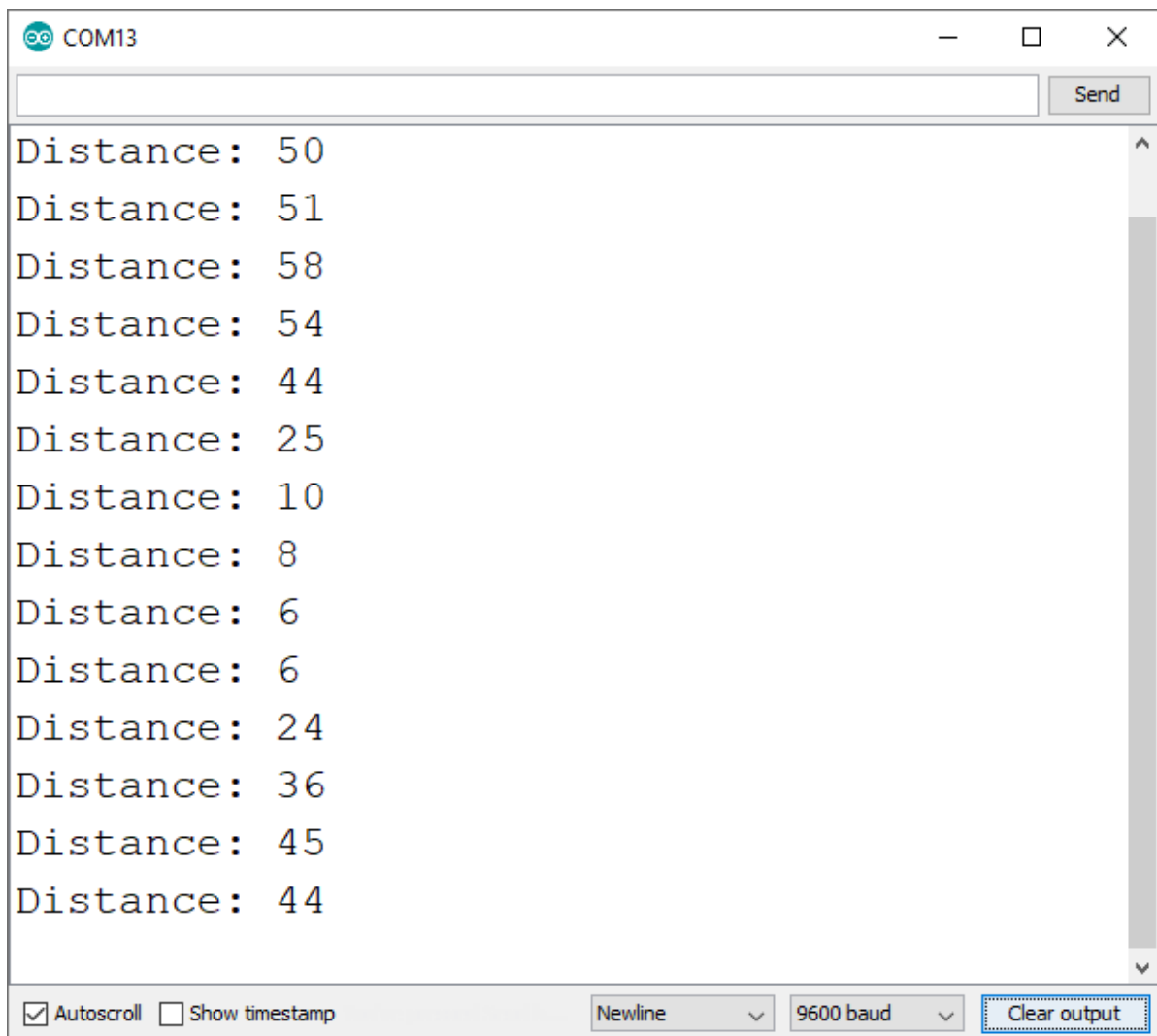
Um den Abstand zu berechnen, benutzen wir diese Zeile Code:

```
distance = duration * 0.034 / 2;
```

Am Ende der Funktion "`loop()`" geben wir die Entfernungsdaten auf dem seriellen Monitor aus. Außerdem gibt es am Ende der `loop()`-Funktion eine Verzögerung von 1000 Millisekunden. Dies ist die Verzögerung zwischen zwei Messungen.

Az-Delivery

Laden Sie die Skizze in den Uno hoch und starten Sie den Serial Monitor (Tools > Serial Monitor). Die Ausgabe sollte wie folgt aussehen:



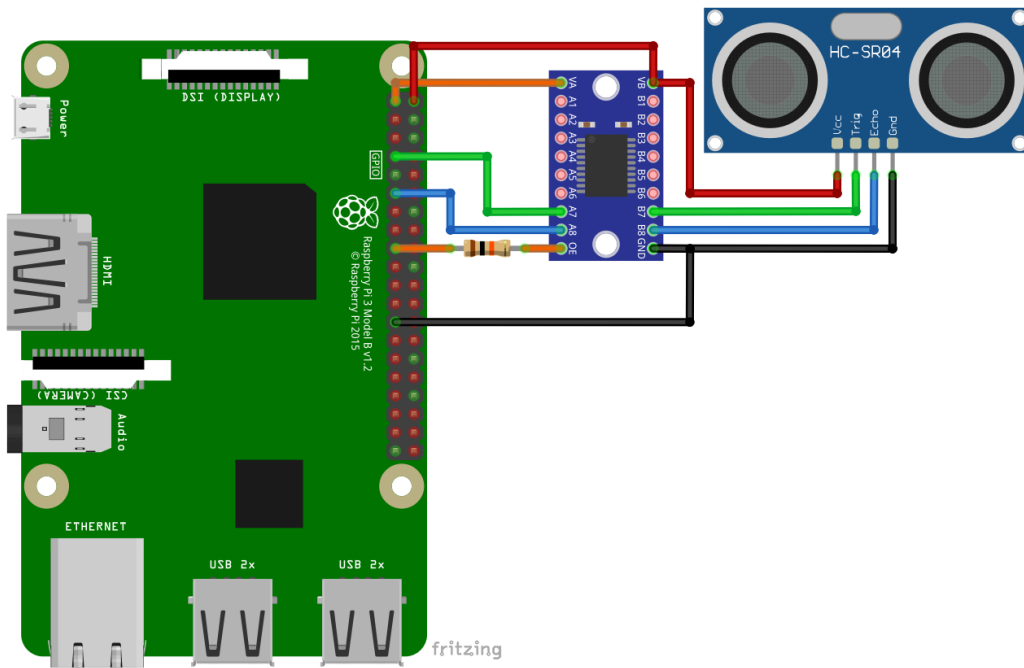
The screenshot shows the Serial Monitor window for COM13. The window title is "COM13" and it has standard window controls (minimize, maximize, close). At the top, there is a text input field and a "Send" button. The main area displays a list of distance measurements in centimeters, each on a new line. The measurements are: 50, 51, 58, 54, 44, 25, 10, 8, 6, 6, 24, 36, 45, and 44. At the bottom, there are checkboxes for "Autoscroll" (checked) and "Show timestamp" (unchecked). To the right of these are two dropdown menus: "Newline" and "9600 baud". A "Clear output" button is located at the bottom right of the window.

```
Distance: 50
Distance: 51
Distance: 58
Distance: 54
Distance: 44
Distance: 25
Distance: 10
Distance: 8
Distance: 6
Distance: 6
Distance: 24
Distance: 36
Distance: 45
Distance: 44
```

AZ-Delivery

Verbindung des Moduls mit dem Raspberry Pi

Verbinden Sie den Sensor mit dem Raspberry Pi, wie unten abgebildet:



HC-SR04 Pin > Raspberry Pi Pin

VCC	>	5V	[pin 2]	Roter Draht
GND	>	GND	[pin 25]	Schwarzer Draht
TRIG	>	GPIO4	[pin 7] via LLC*	Grüner Draht
ECHO	>	GPIO17	[pin 11] via LLC*	Blauer Draht

* LLC – Logic Level Converter

LLC pin > Raspberry Pi pin

VA	>	3V3	[pin 1]	Oranger Draht
VB	>	5V	[pin 2]	Roter Draht
GND	>	GND	[pin 25]	Schwarzer Draht
OE	>	3V3	[pin 17]	Oranger Draht

Bringen Sie zwischen OE pin des LLC und 3V3, einen 10kΩ Pull-Up-Widerstand an.

AZ-Delivery

Die Ausgangssignale des Ultraschallsensors "HC-SR04" liegen im 5V-Bereich, so dass der Modulausgang von 5V in 3,3V umgewandelt werden muss, um eine Beschädigung des Raspberry Pi zu vermeiden. Dies kann mit einem "Logic level converter" oder "Shifter" erreicht werden. AZ-Delivery bietet Solche unter der Bezeichnung "TXS0108E 8-Kanal-Logik-Pegelwandler" an. Dieses Modul ist bidirektional, es kann Spannungen von 3,3V in 5V und umgekehrt konvertieren. Es hat 8 Kanäle, was bedeutet, dass 8 verschiedene digitale Pins für Spannungsumwandlungen verwendet werden können.

Der VA-Pin wird als Referenz für eine Spannung mit niedrigerem Pegel verwendet (z.B. 3,3V der Raspberry Pi).

Der VB-Pin wird als Referenz für eine Spannung mit höherem Pegel verwendet (z.B. 5V des Ultraschallsensors).

Der OE-Pin oder der "Output Enable" muss über einen 10k Ω Pull-Up-Widerstand an die 3,3V angeschlossen werden.

Der echoPin des Ultraschall-Sensors ist mit dem B7-Pin des Logik-Pegelwandlers verbunden, und der trigPin ist mit dem B6-Pin des Logik-Pegelwandlers verbunden.

Der A7-Pin des Logik-Pegelwandlers ist mit dem GPIO-Pin 17 und der A6-Pin des Logik-Pegelwandlers mit dem GPIO-Pin 4 verbunden.

Az-Delivery

Python Skript:

```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
TRIG = 17
ECHO = 4
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)
GPIO.output(TRIG, False)
time.sleep(2)
print('[press ctrl+c to end the script]')
try: # Main program loop
    while True:
        GPIO.output(TRIG, True)
        time.sleep(0.00001)
        GPIO.output(TRIG, False)
        while GPIO.input(ECHO) == 0:
            pulse_start = time.time()
        while GPIO.input(ECHO) == 1:
            pulse_end = time.time()
        pulse_duration = pulse_end - pulse_start
        distance = pulse_duration * 17150
        distance = round(distance, 2)
        print('Distance is {} cm'.format(distance))
        time.sleep(2)

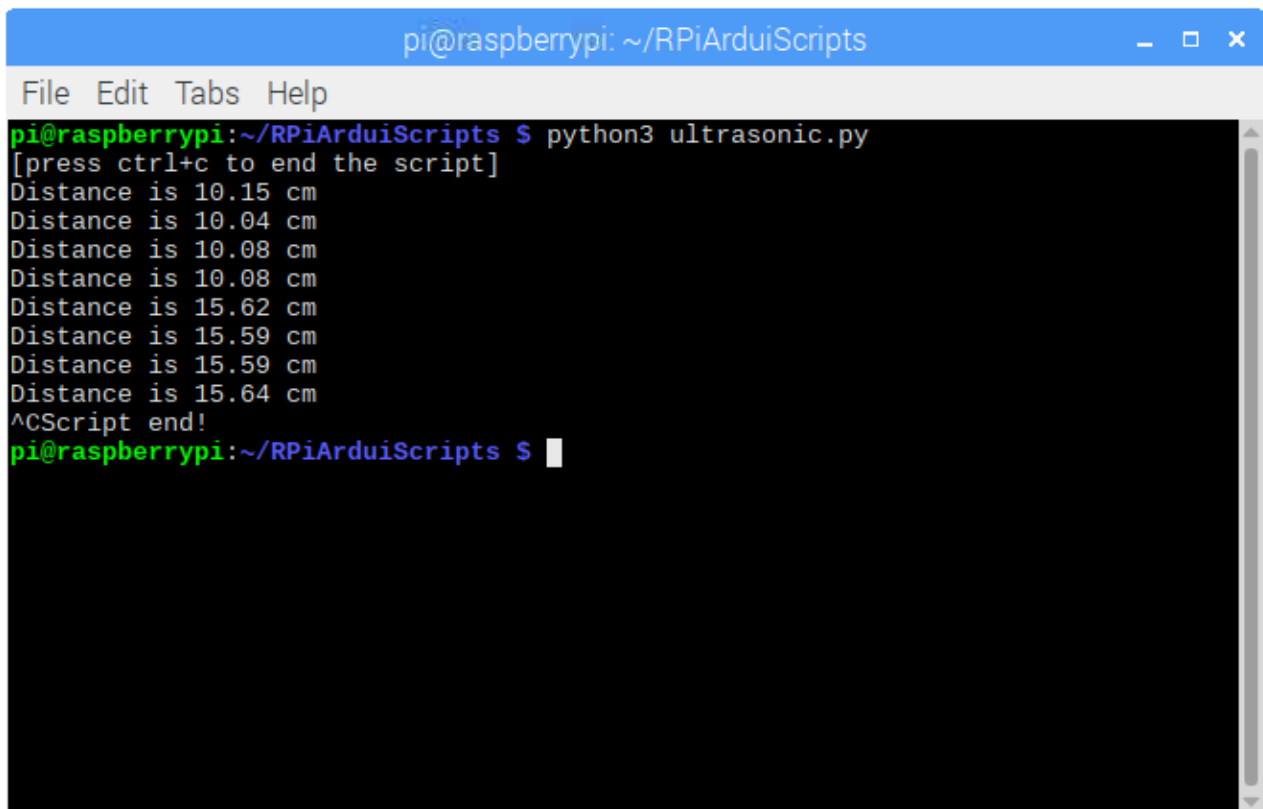
# Scavenging work after the end of the program
except KeyboardInterrupt:
    print('Script end!')

finally:
    GPIO.cleanup()
```

Az-Delivery

Speichern Sie das Skript als "*ultrasonic.py*", öffnen Sie das Terminal und führen Sie diesen Befehl aus: **python3 ultrasonic.py**

Die Ausgabe sollte wie folgt aussehen:



```
pi@raspberrypi: ~/RPiArduiScripts
File Edit Tabs Help
pi@raspberrypi:~/RPiArduiScripts $ python3 ultrasonic.py
[press ctrl+c to end the script]
Distance is 10.15 cm
Distance is 10.04 cm
Distance is 10.08 cm
Distance is 10.08 cm
Distance is 15.62 cm
Distance is 15.59 cm
Distance is 15.59 cm
Distance is 15.64 cm
^CScript end!
pi@raspberrypi:~/RPiArduiScripts $
```

Um das Skript zu beenden, drücken Sie STRG + C.

Az-Delivery

Zunächst wird die GPIO-Suite eingeführt, die eine grundlegende GPIO-Steuerung darstellt. Wir schalten alle Warnungen in Verbindung mit den GPIO-Pins aus. Danach erstellen und initialisieren wir zwei Variablen, die wir *TRIG* und *ECHO* nennen. In diesen Variablen speichern wir die Nummern 17 und 4. Diese Nummern stellen die GPIO-Pins dar, an welche die Sensor-Pins angeschlossen werden.

Nachdem die Pin-Modi eingestellt wurden (TRIG-Pin als OUTPUT und ECHO-Pin als INPUT), wird der Zustand des TRIG-Pins auf LOW gesetzt.

Im "infinite loop block" (while True:) wurde eine Ultraschallwelle übertragen und der Zeitpunkt der Übertragung aufgezeichnet. Dann warten wir, bis die Ultraschallwelle zurückgeworfen wird und vom Ultraschallempfänger erkannt wird. Dieser Zeitpunkt wurde ebenfalls aufgezeichnet.

Jetzt senden wir den Ultraschallimpuls mit dieser Codezeile `GPIO.output(TRIG, True)`,
und warten auf die zurückgeworfene Ultraschallwelle.

"While" blocks werden verwendet, um sicherzustellen, dass jeder Zeitstempel des Signals in der richtigen Reihenfolge aufgezeichnet wird. Die Funktion "time.time()" zeichnet den jeweiligen aktuellsten Zeitstempel auf. Die Aufzeichnung dieser Zeitstempel erfolgt in zwei "while" blocks:

```
while GPIO.input(ECHO) == 0:  
    pulse_start = time.time()
```

```
while GPIO.input(ECHO) == 1:  
    pulse_end = time.time()
```

Az-Delivery

Mit dieser Codezeile berechnen wir das Zeitintervall zwischen der Übertragung und der Erfassung der Ultraschallwelle:

```
pulse_duration = pulse_end - pulse_start
```

Da die Ultraschallwelle vom Sender bis zum Hindernis und zurück zum Empfänger wandert, muss das Zeitintervall in der Variablen `pulse_duaration` durch 2 geteilt werden, wenn wir die Entfernung zum Hindernis berechnen.

Zur Berechnung der Entfernung verwenden wir diese Zeile des Codes:

```
distance = pulse_duration * 17150
```

Am Ende des "infinite loop blocks" runden wir den Distanzwert auf zwei Dezimalstellen und geben die Daten aus. Dann stellen wir das Zeitintervall zwischen zwei Messungen auf 2 Sekunden ein in der `time.sleep(2)`.

Um das Skript zu stoppen, drücken Sie die Tastenkombination STRG + C. Dies wird als Tastaturunterbrechung bezeichnet, und wir warten darauf im Block `except KeyboardInterrupt`

Sie haben es geschafft. Sie können jetzt unser Modul für Ihre Projekte nutzen.

Az-Delivery

Jetzt sind Sie dran! Entwickeln Sie Ihre eigenen Projekte und Smart-Home Installationen. Wie Sie das bewerkstelligen können, zeigen wir Ihnen unkompliziert und verständlich auf unserem Blog. Dort bieten wir Ihnen Beispielskripte und Tutorials mit interessanten kleinen Projekten an, um schnell in die Welt der Mikroelektronik einzusteigen. Zusätzlich bietet Ihnen auch das Internet unzählige Möglichkeiten, um sich in Sachen Mikroelektronik weiterzubilden.

Falls Sie nach noch weiteren hochwertigen Produkten für Arduino und Raspberry Pi suchen, sind Sie bei AZ-Delivery Vertriebs GmbH goldrichtig. Wir bieten Ihnen zahlreiche Anwendungsbeispiele, ausführliche Installationsanleitungen, E-Books, Bibliotheken und natürlich die Unterstützung unserer technischen Experten.

<https://az-delivery.de>

Viel Spaß!

Impressum

<https://az-delivery.de/pages/about-us>