Mikrokontroller & I²C Bus

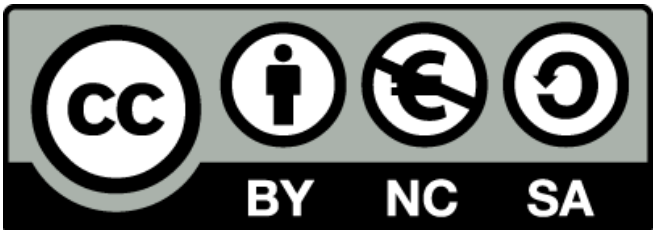
by AS

www.makerconnect.de

https://www.makerconnect.de/resource

Raspberry Pi Pico
Entfernungsmessung mit Ultraschall

Copyright



Sicherheitshinweise

Lesen Sie diese Gebrauchsanleitung, bevor Sie diesen Bausatz in Betrieb nehmen und bewahren Sie diese an einem für alle Benutzer jederzeit zugänglichen Platz auf. Bei Schäden, die durch Nichtbeachtung dieser Bedienungsanleitung verursacht werden, erlischt die Gewährleistung / Garantie. Für Folgeschäden übernehmen wir keine Haftung! Bei allen Geräten, die zu ihrem Betrieb eine elektrische Spannung benötigen, müssen die gültigen VDE-Vorschriften beachtet werden. Besonders relevant sind für diesen Bausatz die VDE-Richtlinien VDE 0100, VDE 0550/0551, VDE 0700, VDE 0711 und VDE 0860. Bitte beachten Sie auch nachfolgende Sicherheitshinweise:

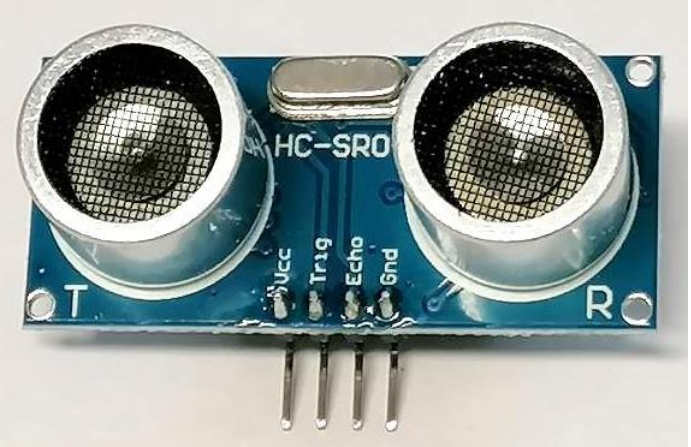
- Nehmen Sie diesen Bausatz nur dann in Betrieb, wenn er zuvor berührungssicher in ein Gehäuse eingebaut wurde. Erst danach darf dieser an eine Spannungsversorgung angeschlossen werden.
- Lassen Sie Geräte, die mit einer Versorgungsspannung größer als 24 V- betrieben werden, nur durch eine fachkundige Person anschließen.
- In Schulen, Ausbildungseinrichtungen, Hobby- und Selbsthilfewerkstätten ist das Betreiben dieser Baugruppe durch geschultes Personal verantwortlich zu überwachen.
- In einer Umgebung in der brennbare Gase, Dämpfe oder Stäube vorhanden sind oder vorhanden sein können, darf diese Baugruppe nicht betrieben werden.
- Im Falle eine Reparatur dieser Baugruppe, dürfen nur Original-Ersatzteile verwendet werden! Die Verwendung abweichender Ersatzteile kann zu ernsthaften Sach- und Personenschäden führen. Eine Reparatur des Gerätes darf nur von fachkundigen Personen durchgeführt werden.
- Spannungsführende Teile an dieser Baugruppe dürfen nur dann berührt werden (gilt auch für Werkzeuge, Messinstrumente o.ä.), wenn sichergestellt ist, dass die Baugruppe von der Versorgungsspannung getrennt wurde und elektrische Ladungen, die in den in der Baugruppe befindlichen Bauteilen gespeichert sind, vorher entladen wurden.
- Sind Messungen bei geöffnetem Gehäuse unumgänglich, muss ein Trenntrafo zur Spannungsversorgung verwendet werden
- Spannungsführende Kabel oder Leitungen, mit denen die Baugruppe verbunden ist, müssen immer auf Isolationsfehler oder Bruchstellen kontrolliert werden. Bei einem Fehler muss das Gerät unverzüglich ausser Betrieb genommen werden, bis die defekte Leitung ausgewechselt worden ist.
- Es ist auf die genaue Einhaltung der genannten Kenndaten der Baugruppe und der in der Baugruppe verwendeten Bauteile zu achten. Gehen diese aus der beiliegenden Beschreibung nicht hervor, so ist eine fachkundige Person hinzuzuziehen

Bestimmungsgemäße Verwendung

- Auf keinen Fall darf 230 V~ Netzspannung angeschlossen werden. Es besteht dann Lebensgefahr!
- Dieser Bausatz ist nur zum Einsatz unter Lern- und Laborbedingungen konzipiert worden. Er ist nicht geeignet, reale Steuerungsaufgaben jeglicher Art zu übernehmen. Ein anderer Einsatz als angegeben ist nicht zulässig!
- Der Bausatz ist nur für den Gebrauch in trockenen und sauberen Räumen bestimmt.
- Wird dieser Bausatz nicht bestimmungsgemäß eingesetzt kann er beschädigt werden, was mit Gefahren, wie z.B. Kurzschluss, Brand, elektrischer Schlag etc. verbunden ist. Der Bausatz darf nicht geändert bzw. umgebaut werden!
- Für alle Personen- und Sachschäden, die aus nicht bestimmungsgemäßer Verwendung entstehen, ist nicht der Hersteller, sondern der Betreiber verantwortlich. Bitte beachten Sie, dass Bedien- und /oder Anschlussfehler außerhalb unseres Einflussbereiches liegen. Verständlicherweise können wir für Schäden, die daraus entstehen, keinerlei Haftung übernehmen.
- Der Autor dieses Tutorials übernimmt keine Haftung für Schäden. Die Nutzung der Hard- und Software erfolgt auf eigenes Risiko.

Raspberry Pi Pico – Entfernungsmessung mit Ultraschall

Ein sehr beliebtes Selbstbauprojekt bei Bastlern ist die Entfernungsmessung mit Ultraschall. Dazu wird sehr gern das Modul HC-SR04 verwendet. Es wird von den verschiedensten Herstellern produziert und auf vielen Seiten im Internet angeboten.



HC-SR04

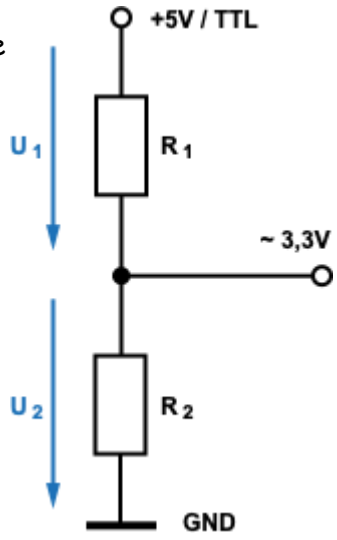
Es ist klein, kompakt und leicht zu verarbeiten. Auf Grund seiner 4 Steckverbinder kann es leicht auf Breadboard verwendet werden.

Die technischen Daten:

- Betriebsspannung 5V
- Betriebsstrom ca. 2mA
- Erfassungsbereich ca. 2 cm bis 450 cm
- Eingangstrigger 10µs TTL
- Sensorwinkel bis ca. 15 Grad

Leider liegt in diesen Daten bereits das Problem. Der Pico arbeitet „nur" mit einer Spannung von 3,3V als Vcc und an den GPIOs. Es gibt auch Sonderformen die sowohl mit 5V und 3,3V arbeiten. Dafür haben diese dann einen entsprechend anderen Preis. Im Internet werden diese oft angeboten bzw. mit diesen Spannungen angegeben. Leider kauft man oft das verkehrte oder man wird betrogen. Oft sind aber Module mit 5V bereits vorhanden und man kann sie weiter verwenden. Der Anschluss der 5V Module an den Pico erfolgt dann über einen Spannungsteiler.

Der Eingangspegel eines GPIOs am Raspberry Pi Pico für "High" liegt bei einer Spannung von +3,3 Volt. Wenn man aber ein Signal mit einem größeren Eingangspegel hat, beispielsweise mit 5 Volt, dann darf man dieses Signal nicht direkt an einen GPIO-Eingang legen. Der GPIO-Eingang wird überlastet und kann zerstört werden. Die Lösung ist ein einfacher Spannungsteiler, der eine Eingangsspannung von 5 Volt auf 3,3 Volt herunterteilt.



Spannungsteiler

Da ein GPIO-Eingang nur maximal +3,3 V verträgt, muss die Eingangsspannung verringert werden. Dazu eignet sich ein Spannungsteiler, der im einfachsten Fall aus zwei Widerständen besteht am besten. Vom Prinzip her ist ein Spannungsteiler eine Reihenschaltung aus 2 Widerständen.
Da der Strom in einer Reihenschaltung überall gleich groß ist, verursachen ungleiche Widerstände in der Reihenschaltung unterschiedliche Spannungsabfälle an den Widerständen. Diese Spannungsabfälle, auch Teilspannungen (U1, U2, ...) genannt, verhalten sich dabei wie die

dazugehörigen Widerstände (R1, R2, ...). Das bedeutet, am größten Widerstand fällt der größte Teil der Gesamtspannung (5 Volt) ab. Am kleinsten Widerstand fällt der kleinste Teil der Gesamtspannung (5 Volt) ab.

Welche Widerstandswerte sind sinnvoll?

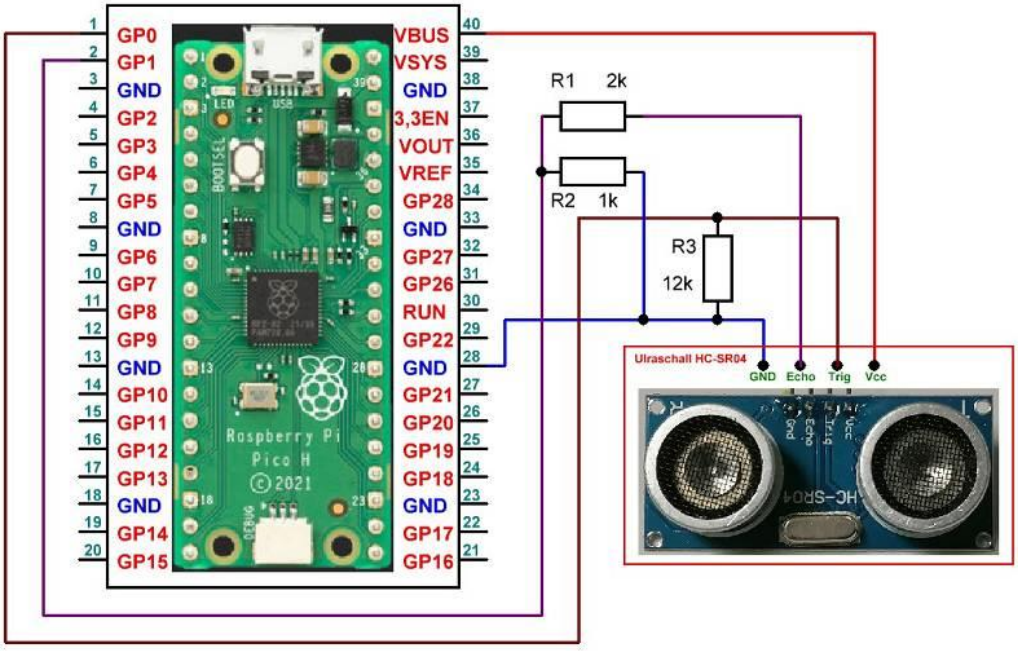
Im Prinzip brauchen wir zwei Widerstände an denen sich die Spannung von 5 Volt in 1,7 Volt und 3,3 Volt aufteilt (5 Volt = 1,7 Volt + 3,3 Volt). Dabei soll auch ein kleiner Strom von ca. 1,5mA fliessen. Am besten wieder mit dem Ohm'schen Gesetz:

$$\frac{U}{R\ x\ I} \rightarrow I = \frac{U}{R} = \frac{5V}{3\ kOhm} = \ 0{,}0016A = 1{,}6\ mA$$

$$R\ 1 = \frac{3{,}3\ V}{1{,}6\ mA} \sim 2\ kOhm \qquad R\ 2 = \frac{1{,}7\ V}{1{,}6\ mA} \sim 1\ kOhm$$

Somit ergibt sich ein Spannungsteiler aus den Widerständen R_1 = 2 kOhm und R_2 = 1 kOhm bei einem Strom von 1,6mA.

Raspberry Pi Pico mit Ultraschall Modul HC-SR04



Spannungsteiler mit R1 und R2. R3 ist ein Last-widerstand für den Trigger Impuls.
Im Internet sind verschiedene Grössen für R3 angegeben. Ist R3 < 1 kOhm kann es zu Problemen beim Messen der Entfernung bzw. Geschwindigkeit kommen. Die 12 kOhm haben sich als guter Kompromiss zwischen Belastung und Geschwindigkeit erwiesen.

Im Programmcode wird ein GPIO-Ausgang für das Trigger-Signal zum Sensor und ein GPIO-Eingang für das Echo-Signal vom Sensor initialisiert. Anschließend wird in einer Endlos-Schleife die Signallaufzeit gemessen. Innerhalb der Schleife wird aus der Startzeit und Endzeit die Differenz berechnet und mit der Schallgeschwindigkeit multipliziert. Am Ende der Schleife wird der Abstand ausgegeben.

Das komplette Programm:

```
# Bibliotheken laden
from machine import Pin
from utime import sleep, sleep_us, ticks_us
```

```
# Initialisierung GPIO-Ausgang für Trigger-Signal
trigger = Pin(0, Pin.OUT) # Angabe des Trigger am Pico

# Initialisierung GPIO-Eingang für Echo-Signal
echo = Pin(1, Pin.IN) # Angabe des Echo Pins am Pico

# Wiederholung (Endlos-Schleife)
while True:
  # Abstand messen
  trigger.low()
  sleep_us(2)
  trigger.high()
  sleep_us(5)
  trigger.low()

  # Zeitmessungen
  while echo.value() == 0:
    signaloff = ticks_us()
  while echo.value() == 1:
    signalon = ticks_us()

  # Vergangene Zeit ermitteln
  timepassed = signalon – signaloff

  # Abstand/Entfernung ermitteln
  # Entfernung über die Schallgeschwindigkeit (34320 cm/s bei 20 °C) berechnen
  # Durch 2 teilen, wegen Hin- und Rückweg
  abstand = timepassed * 0.03432 / 2

  # Ergebnis ausgeben
  print('    Off:', signaloff)
  print('     On:', signalon)
  print('   Zeit:', timepassed)
  print('Abstand:', str("%.2f" % abstand), 'cm')
  print()
  # 2 Sekunde warten
  sleep(2)
```

Die Anzeige der Daten erfolgt auf dem PC.

Die Abstandsmessung erfolgt durch Messen der Signallaufzeit von Ultraschallsignalen und der anschließenden Umrechnung über die Schallgeschwindigkeit.
Leider ist die Schallgeschwindigkeit temperaturabhängig von der Luft. Wenn man die Zeit mit einer festen Schallgeschwindigkeit umrechnet, dann gilt der errechnete Abstand nur bei einer bestimmten Temperatur. Wenn die tatsächliche Temperatur von dieser Temperatur abweicht, dann stimmt der berechnete Abstand nicht. Ob die Abweichung in einem konkreten Anwendungsfall ein Problem darstellt, hängt davon ab, wie viel die aktuelle Temperatur von beispielsweise 20°C abweicht und ob die Abweichung akzeptabel ist.
Im folgenden Programmcode führen wir eine Temperaturkompensation durch. Das heißt, wir ermitteln die aktuelle Temperatur mit Hilfe des Temperatursensors im Raspberry Pi Pico und berechnen dann zuerst die Schallgeschwindigkeit für diese Temperatur. Und erst dann berechnen wir den Abstand.

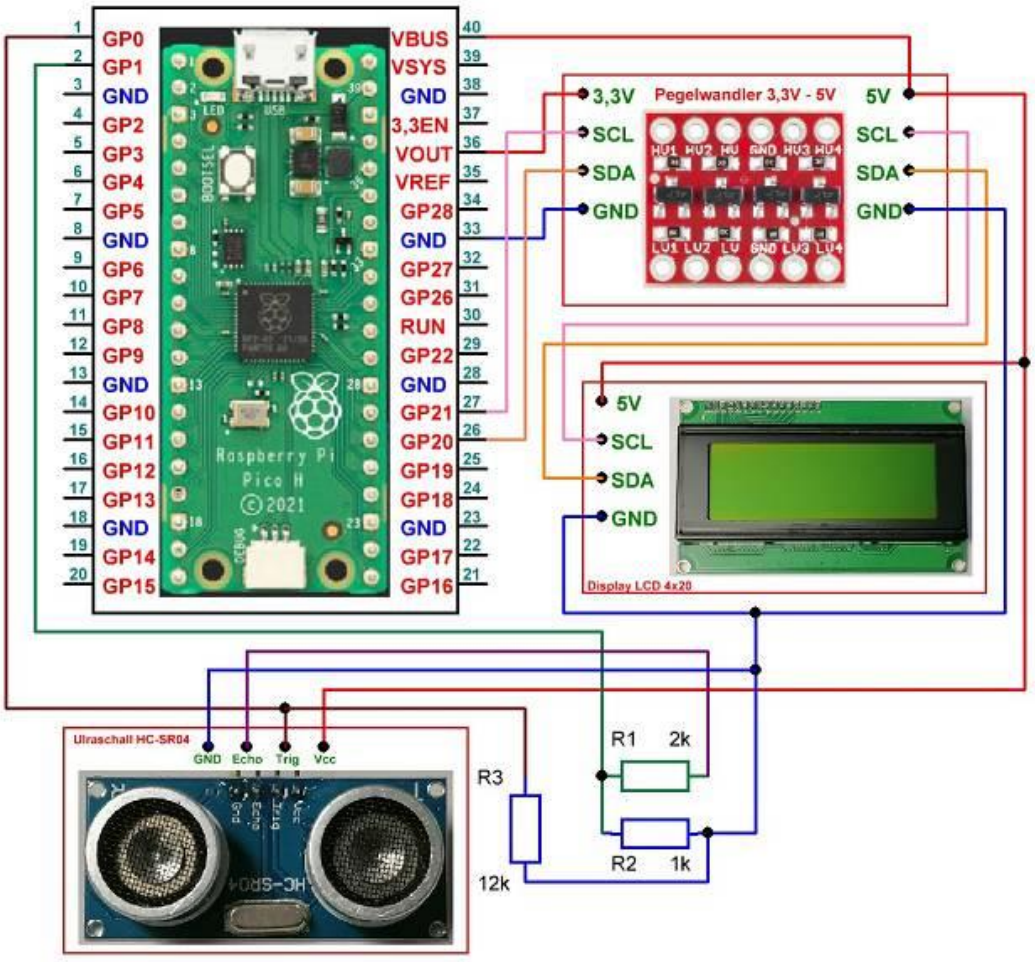
```
# Bibliotheken laden
from machine import Pin, ADC
from utime import sleep, sleep_us, ticks_us
# Initialisierung GPIO-Ausgang für Trigger-Signal
trigger = Pin(0, Pin.OUT)
# Initialisierung GPIO-Eingang für Echo-Signal
echo = Pin(1, Pin.IN)
# Initialisierung des ADC4 (Temperatursensor)
sensor_temp = ADC(4)
conversion_factor = 3.3 / (65535)
# Wiederholung (Endlos-Schleife)
while True:
    # Spannung messen und in Temperatur umrechnen
    read = sensor_temp.read_u16()
    spannung = read * conversion_factor
    temperatur = 27 - (spannung - 0.706) / 0.001721
    # Abstand messen
    trigger.low()
    sleep_us(2)
    trigger.high()
    sleep_us(5)
    trigger.low()
    # Zeitmessungen
    while echo.value() == 0:
      signaloff = ticks_us()
    while echo.value() == 1:
      signalon = ticks_us()
    # Vergangene Zeit ermitteln
    timepassed = signalon - signaloff
    # Abstand/Entfernung ermitteln
    # Entfernung über die Schallgeschwindigkeit (34320 cm/s bei 20 °C) berechnen
    # Durch 2 teilen, wegen Hin- und Rückweg
    abstand = timepassed * 0.03432 / 2
    # Abstand/Entfernung mit Temperatur-Korrektur
    abstand_korr = timepassed * (331.3 + 0.606 * temperatur) / 10000 / 2
    # Ergebnis ausgeben
    print('  Temp.:', str("%.1f" % temperatur), '°C')
    print('    Off:', signaloff)
    print('     On:', signalon)
    print('   Zeit:', timepassed)
    print('Abstand:', str("%.2f" % abstand), 'cm')
    print('  Korr.:', str("%.2f" % abstand_korr), 'cm')
    print()
    # 2 Sekunde warten
    sleep(2)
```

Über Sinn und Unsinn einer Temperaturkompensation

Grundsätzlich ist eine Temperaturkompensation sinnvoll, wenn man nicht von stabilen Temperaturen ausgehen kann. Vor allem außerhalb von geschlossenen Räumen können Temperaturen erheblich voneinander abweichen. Entsprechend ungenau sind dann auch die Ergebnisse.
Ein Aspekt, bei der Temperaturmessung, muss noch zwingend berücksichtigt werden. Das ist der Ort der Temperaturmessung. In diesem Aufbau ist der Ort grober Unfug. Warum? Weil der integrierte Temperaturfühler die Temperatur des Picos ermittelt. Das ist natürlich Quatsch. Wir brauchen die Temperatur, die innerhalb der Übertragungsstrecke des Schalls herrscht. Dafür brauchen wir einen externen Temperaturfühler. Das Problem dabei ist, dass innerhalb der Übertragungsstrecke kein Temperaturfühler sein darf, weil sonst der Abstand zum Temperaturfühler gemessen wird. Er muss also außerhalb der Übertragungsstrecke des Schalls platziert werden. Die Temperaturmessung wäre dann aber nicht ganz korrekt, weil der Temperaturfühler irgendeine Umgebungstemperatur ermittelt. Eine Temperaturkompensation, egal wie, kann also nur eine grobe Annäherung sein.

Als nächstes wollen wir den Abstand auf einem Display anzeigen lassen. Dazu verwende ich wieder ein LCD Display mit 5x20 Segmenten.

Im Programmcode wird ein GPIO-Ausgang für das Trigger-Signal zum Sensor und ein GPIO-Eingang für das Echo-Signal vom Sensor initialisiert. Außerdem wird die Ansteuerung des Displays mit der I^2C-Schnittstelle initialisiert.
In einer Endlos-Schleife die Signallaufzeit gemessen. Innerhalb der Schleife wird aus der Startzeit und Endzeit die Differenz berechnet und mit der Schallgeschwindigkeit multipliziert. Am Ende der Schleife wird der Abstand auf dem Display ausgegeben.



Raspberry Pi Pico mit Ultraschall Modul HC-SR04, Pegelwandler und Display

In diesem Beispiel haben wir das Problem unterschiedlicher Pegel zwischen den einzelnen Platinen. Der Ausgangspegel am I^2C Bus des Pico beträgt 3,3V. Der Eingangspegel des LCD-Displays 4x20 beträgt 5V (mit PCF8574). Zwischen diesen beiden Platinen habe ich einen Pegelwandler geschaltet. Damit wird der Ausgangspegel des Pico von 3,3V an den 5V Pegel des Displays angepasst. Dabei ist besonders auf die korrekte Verkabelung der Spannungen zu achten.

```
# Bibliotheken laden
import machine
from machine import I2C, Pin
from lcd_api import LcdApi
from pico_i2c_lcd import I2cLcd
from utime import sleep, sleep_us, ticks_us

# Initialisierung GPIO-Ausgang für Trigger-Signal
trigger = Pin(0, Pin.OUT)

# Initialisierung GPIO-Eingang für Echo-Signal
echo = Pin(1, Pin.IN)

# Vorbereitung Bus, Angabe Bus Nr und Pins
SCL_Pin = 21 # 21 oder 1
SDA_Pin = 20 # 20 oder 0
Bus = 0 # Angabe Busnummer
# Initialisierung I2C, Bus 0, sda-20, scl-21 und Display
i2c_addr     = 0x27 # Angabe I2C Adresse Display
i2c_num_rows = 4 # Angabe der Zeilen entweder 2 oder 4
i2c_num_cols = 20 # Angabe der Zeichen entweder 16 oder 20
i2c = I2C(Bus, scl = Pin(SCL_Pin), sda = Pin(SDA_Pin), freq = 400000)
lcd = I2cLcd(i2c, i2c_addr, i2c_num_rows, i2c_num_cols)

lcd.clear()
lcd.move_to(0,0)  # Spalte , Zeile
lcd.putstr("Abstand Ultraschall") # + "\n" + str("%.2f" % abstand) + ' cm')

# Wiederholung (Endlos-Schleife)
while True:
    # Abstand messen
    trigger.low()
    sleep_us(2)
    trigger.high()
    sleep_us(5)
    trigger.low()
    # Zeiten messen
    while echo.value() == 0:
      signaloff = ticks_us()
    while echo.value() == 1:
      signalon = ticks_us()
    # Vergangene Zeit ermitteln
    timepassed = signalon - signaloff
```

```
# Abstand/Entfernung ermitteln
# Entfernung über die Schallgeschwindigkeit (34320 cm/s bei 20 °C) berechnen
# Durch 2 teilen, wegen Hin- und Rückweg
abstand = timepassed * 0.03432 / 2
lcd.move_to(0,2)  # Spalte , Zeile
lcd.putstr('Abstand:' + "\n" + str("%.2f" % abstand) + ' cm')
# 2 Sekunde warten
sleep(2)
```

Einige Teile des Textes wurden zur besseren Übersicht farblich gestaltet.
Die Nutzung erfolgt auf eigenes Risiko.
Ich wünsche viel Spaß beim Bauen und programmieren
Achim

myroboter@web.de

Quellenangaben:

http://www.elektronik-kompendium.de/sites/raspberry-pi/2612261.htm
https://www.elektronik-kompendium.de/sites/raspberry-pi/2701141.htm
https://www.elektronik-kompendium.de/sites/raspberry-pi/2701151.htm
https://www.elektronik-kompendium.de/sites/raspberry-pi/2701131.htm
Auszüge aus dem Buch „Raspberry Pi Pico“ von Dogan Ibrahim, Projekt 20

Die Artikel wurden dem elektronik-kompendim.de entnommen und meiner Hardware angepasst.
Insbesondere die Ansteuerung zum I^2C Bus und der Anschluss an 5V wurde geändert.
Ich danke den Autoren für ihre sehr gute Arbeit.