

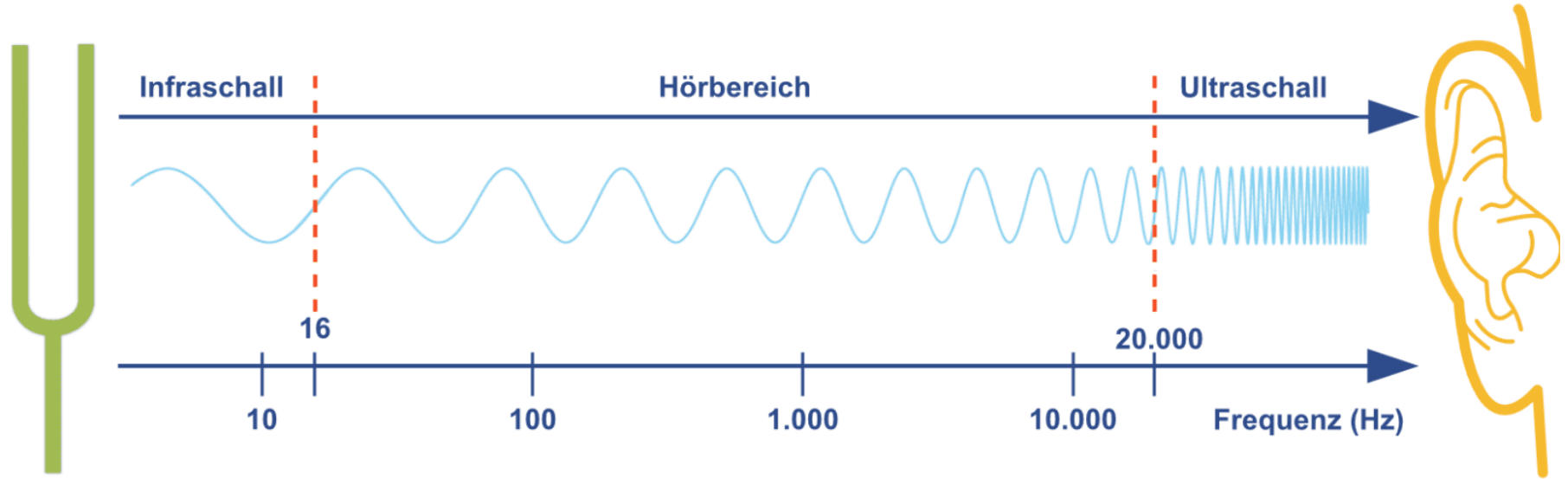
## Test HC-SR04 & VL53LOX

- Daten
- Ultraschall
- HC-SR04
- HC-SR04 Timing Diagramm
- HC-SR04 Leistungstest
- HC-SR04 Entfernung
- VL53LOX
- Elektromagnetisches Spektrum
- VL53LOX
- ToF Sichtfeld
- ESP32 Schaltplan
- RPI pico Schaltplan
- Versuche
- Python Libraries
- Python auf ESP32
- main.py Informationen
- main.py I
- main.py II
- REPL Thonny
- Monitor in Python
- PyInterface\_1.py Informationen
- Python-Monitor
- ESP32 Pinout
- HC-SR04 Schaltplan
- VL53LOX Schaltplan
- Links
- <https://github.com/EKlatt/Experiences>

## Daten

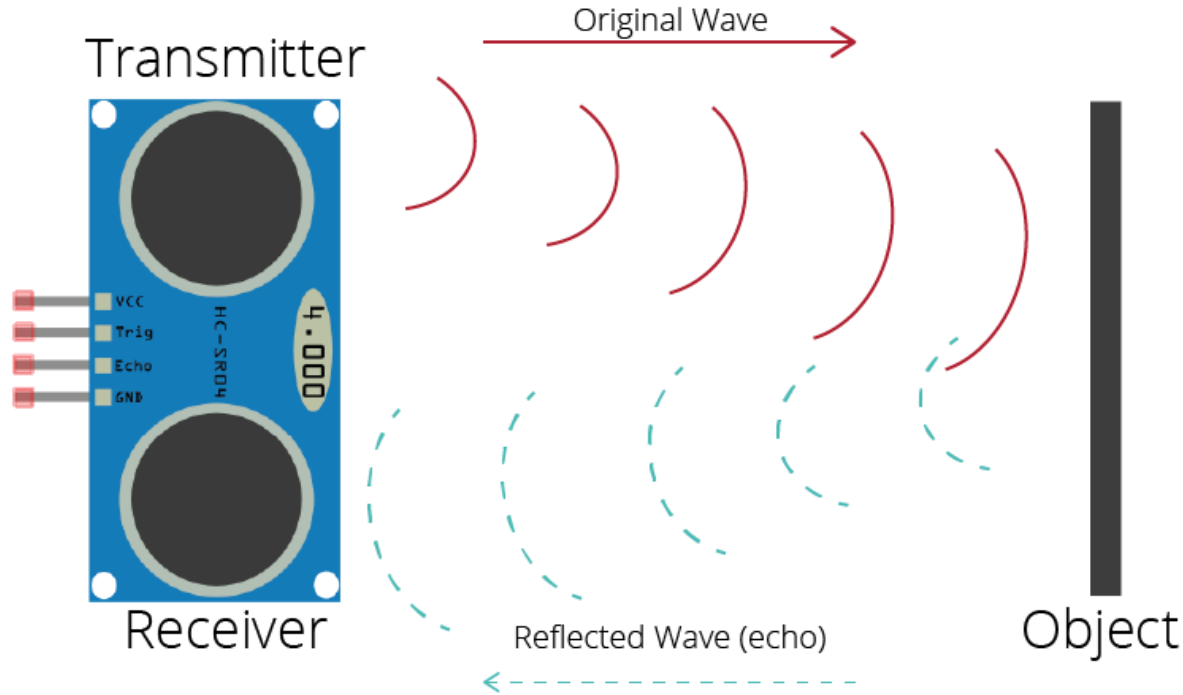
	Ultraschall HC-SR04	Infrarotlaser V53LOX
Maximale Reichweite	4,0 m	Weißes Ziel typisch: 2,0 m (long range)
		Minimum: 1,2 m
		Graues Ziel typisch: 80 cm
		Minimum: 70 cm
Minimale Reichweite	2 cm	> 5 mm
Erkennungswinkel	$\leq 15^\circ$	$25^\circ$
Arbeitsfrequenz	40 kHz	940 nm laser VCXEL
Trigger	10 $\mu$ s TTL Pulse	
Echo	TTL Puls	
		I <sup>2</sup> C
Betriebsarten		default bis 1200 mm long range bis 2200 mm

# Ultraschall



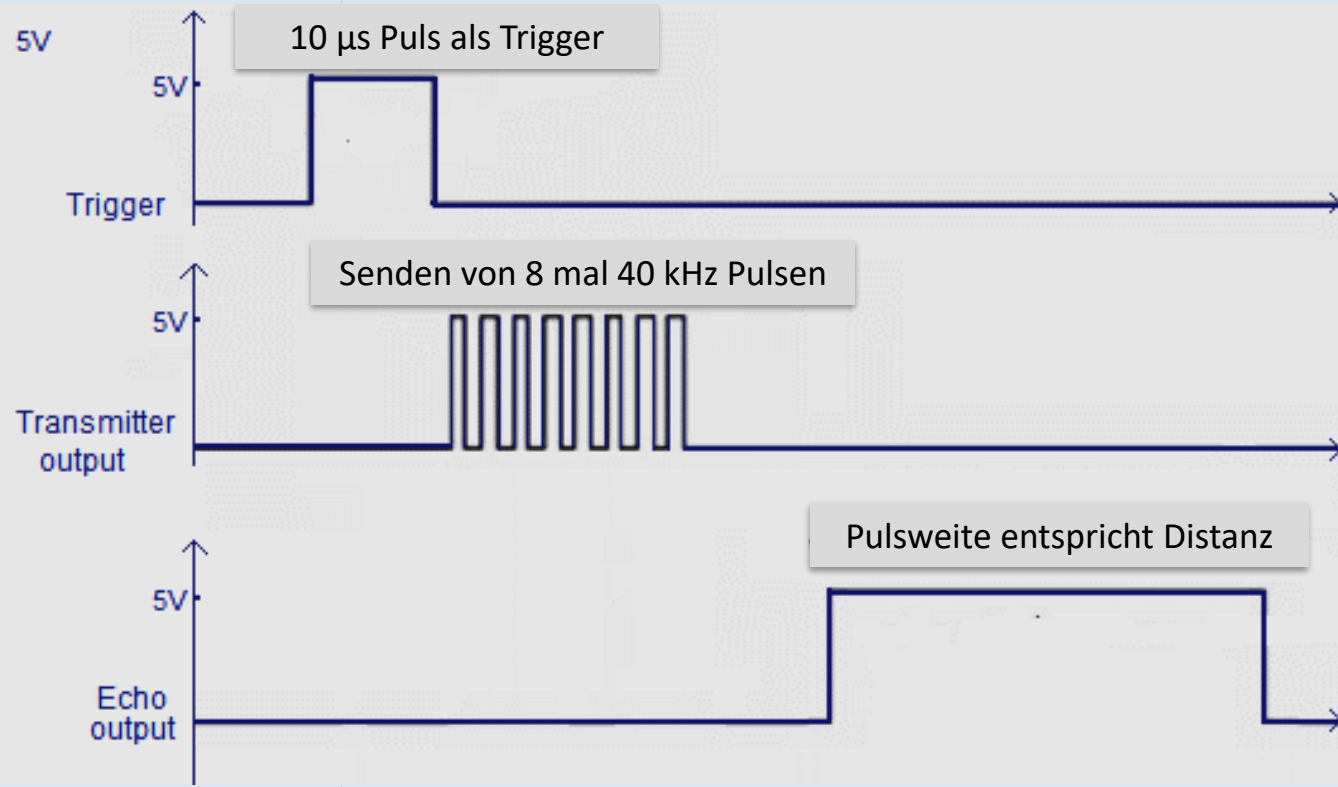
<https://edistechlab.com/ultraschallsensor-hc-sr04-einfach-erklaert/?v=3a52f3c22ed6>

# HC-SR04



<https://randomnerdtutorials.com/micropython-hc-sr04-ultrasonic-esp32-esp8266/>

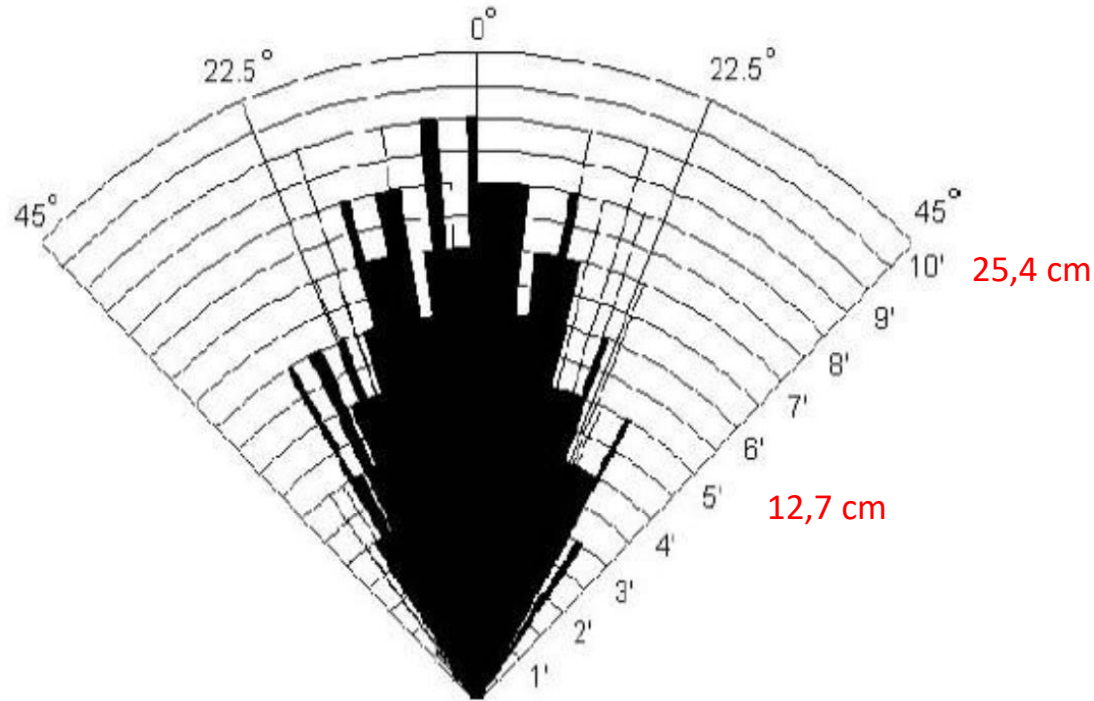
# HC-SR04 Timing Diagram



<https://opencircuit.shop/blog/de-hc-sr04-ultrasonische-afstands-detectie>

# HC-SR04 Leistungstest

Beste Werte bei  $(2 \times 15^\circ) = 30^\circ$



## HC-SR04 Entfernung

Wie wird die Entfernung berechnet?

Der HC-SR04 nutzt das Prinzip der Ultraschallreflexion. Das Modul sendet acht Ultraschallimpulse (40 kHz) aus und wartet auf dessen Empfang.

Die Zeit zwischen Senden und Empfangen dieser Impulse kann dann zur Berechnung der Entfernung herangezogen werden. Die Schallgeschwindigkeit durch Luft beträgt etwa 343 m/s :

Geschwindigkeit = 0,034 cm/ $\mu$ s.

Die Zeit „t“ zwischen Senden und Empfangen muss durch 2 geteilt werden, da das Schallsignal die 2-fache Distanz zum Objekt (hin und zurück) zurückgelegt hat.

$T_{\max}$  ca. 38 ms (bei nicht reflektiertem Signal)

Mit diesen Daten kann die Entfernung nach folgender Formel berechnet werden:

Entfernung = Geschwindigkeit \* Zeit / 2

Entfernung = (t  $\mu$ s \* 0,034 cm/ $\mu$ s) / 2

---

## VL53L0X

### VCSEL

Vertical Cavity Surface-Emitting Laser

Der VL53L0X funktioniert nach dem Time-of-Flight (ToF) Prinzip.

Er besitzt einen IR Laser dessen reflektierte Strahlen hinsichtlich ihrer Flugzeit ausgewertet werden.

Der VCSEL arbeitet mit einer Wellenlänge von 940 nm.

Das reflektierte Licht wird von einem Photodioden Array detektiert

Um einen Millimeter zurückzulegen, benötigt Licht ca.  $3.3 \times 10^{12}$  Sekunden.

<https://www.neumueller.com/Downloads/News/Article/PDF/Fachartikel-Time-of-Flight-2016.pdf>

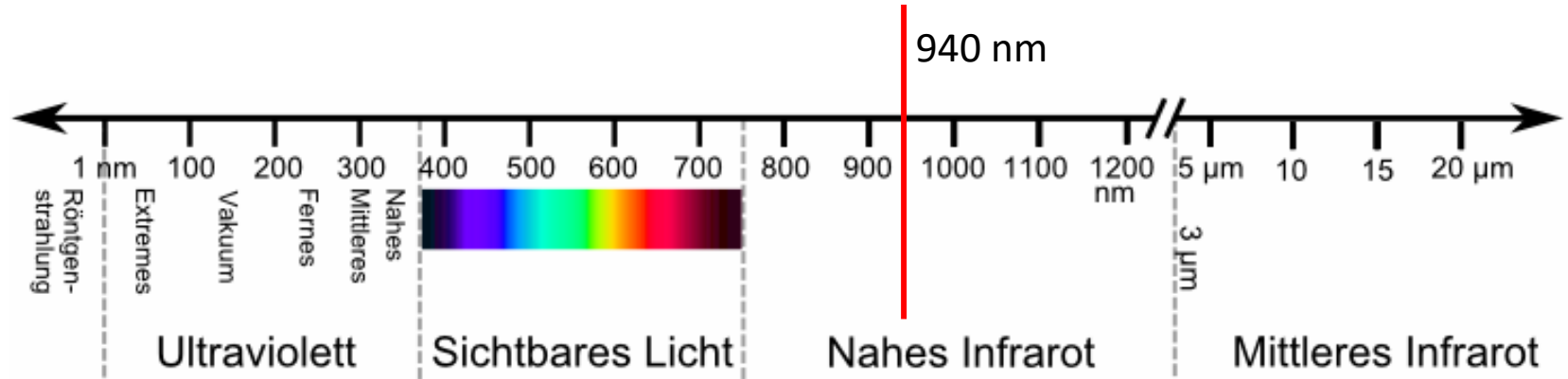
<https://wolles-elektronikkiste.de/vl53l0x-und-vl53l1x-tof-abstandssensoren>

[https://de.wikipedia.org/wiki/Elektrooptische\\_Entfernungsmessung](https://de.wikipedia.org/wiki/Elektrooptische_Entfernungsmessung)

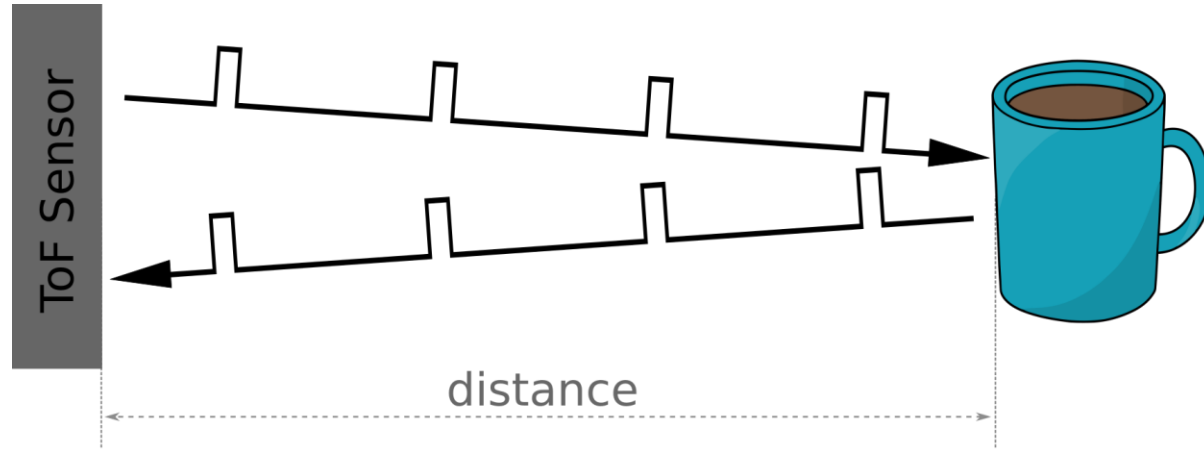
---



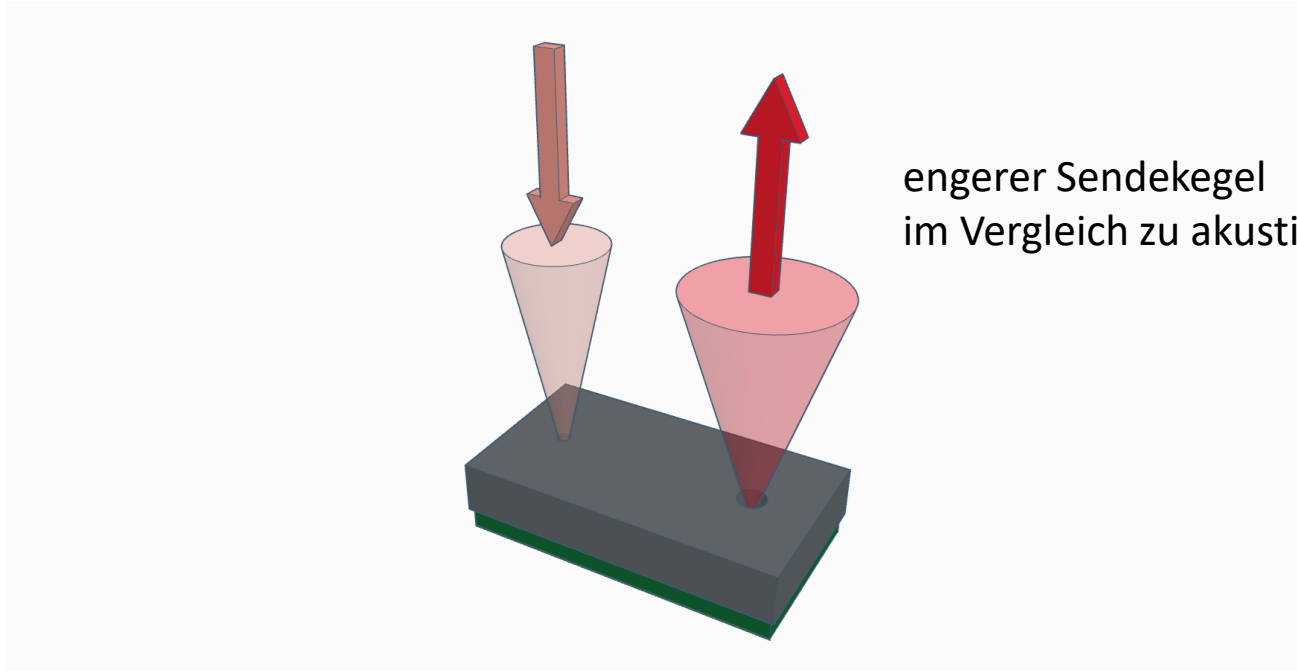
# Elektromagnetisches Spektrum



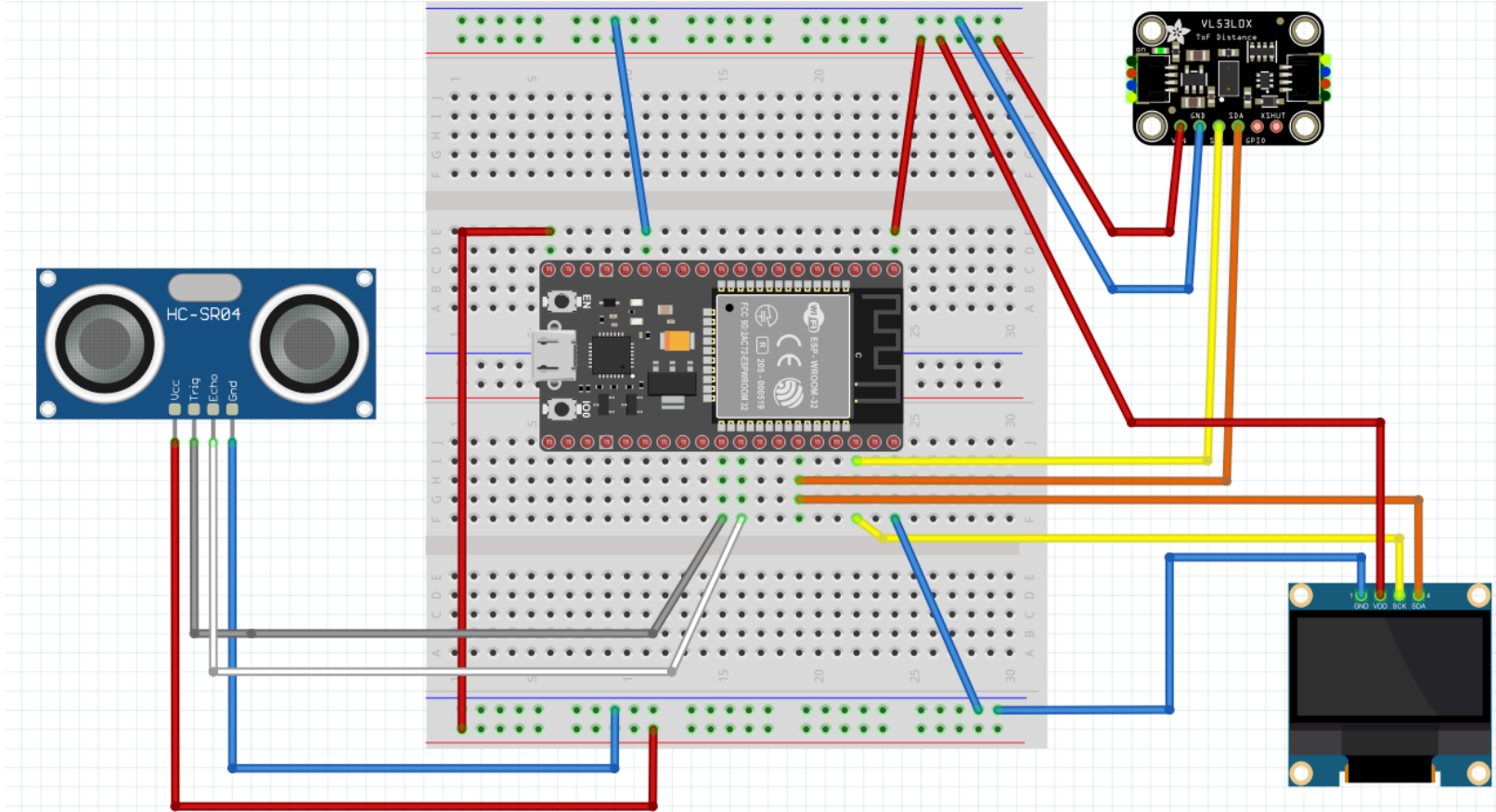
<https://psi.physik.kit.edu/313.php>



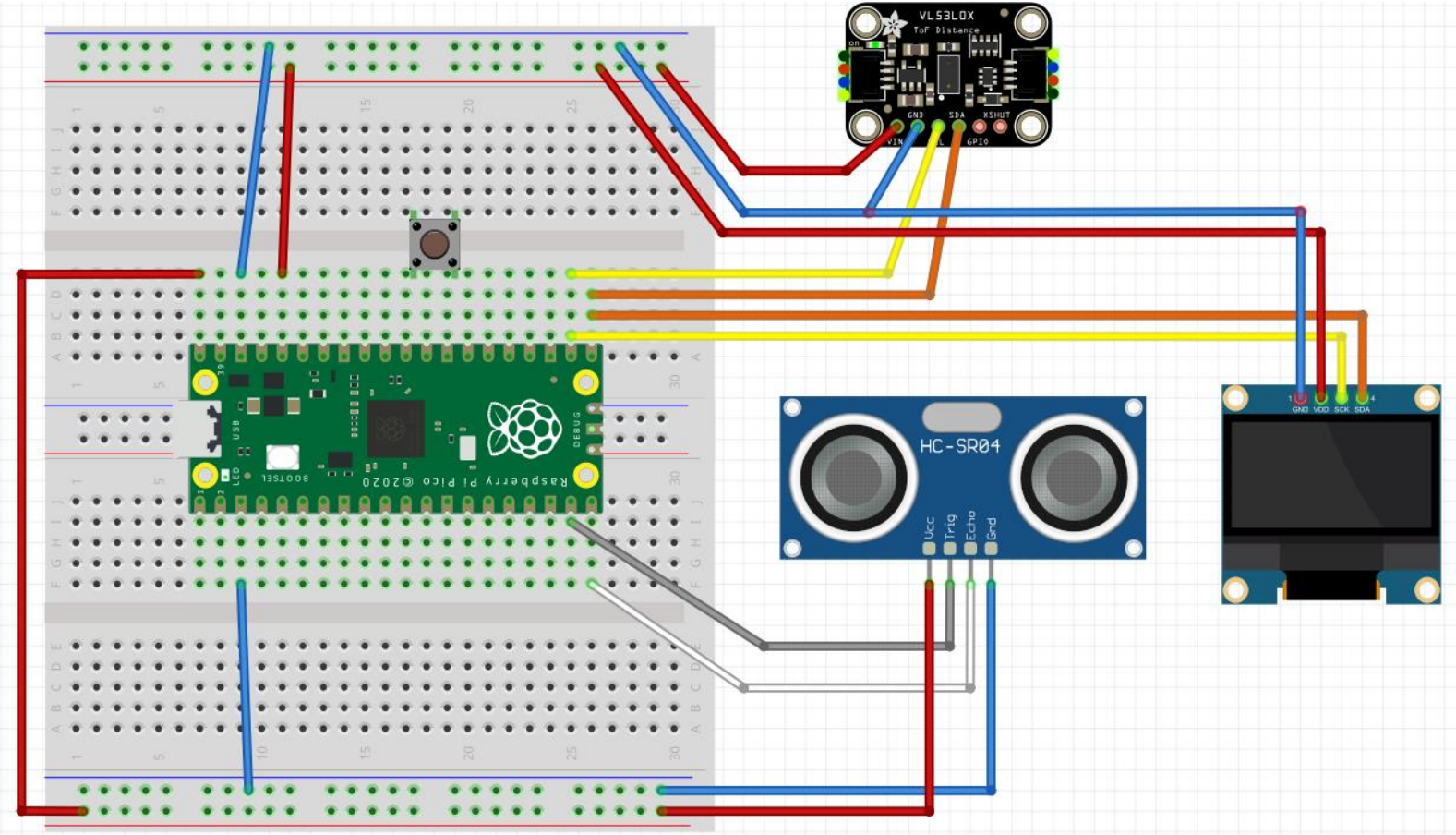
$$\text{distance} = \frac{\text{Speed of Light} \times \text{Time of Flight}}{2}$$



# ESP32 Schaltplan



# RPI pico Schaltplan



---

## Versuche

	Einfluss von
	• farbigen Hintergründen
	• Oberflächenbeschaffenheit
	• absorbierenden Flächen
	• Sensorkegel (Winkel)

## Python Libraries

HC-SR04  
als „hcsr04.py“

<https://github.com/rsc1975/micropython-hcsr04/blob/master/hcsr04.py>

VL53L0X  
Als „vl53l0x.py“

<https://www.grzesina.de/az/theremin/VL53L0X.py>

SSD1306  
als „ssd1306.py“

<https://github.com/adafruit/micropython-adafruit-ssd1306/blob/master/ssd1306.py>

SSD1306  
(neueste Version)

[https://github.com/adafruit/Adafruit\\_CircuitPython\\_SSD1306/blob/main/adafruit\\_ssd1306.py](https://github.com/adafruit/Adafruit_CircuitPython_SSD1306/blob/main/adafruit_ssd1306.py)

# Python auf ESP32

Editor

Portable Thonny 4.1.1

Dateien auf ESP32

boot.py (Standard, nicht benutzt)

hcsr04.py (Library HC-SR04)


vl53l0x.py (Library VL53L0X)


ssd1306.py (Library OLED)


main.py (Hauptprogramm, wird bei RESET gestartet)


Thonny


MicroPython device

 boot.py

 hcsr04.py

 main.py

 ssd1306.py

 vl53l0x.py



# main.py Informationen

```
# ESP32 quick reference I2C:
# https://docs.micropython.org/en/latest/esp32/quickref.html
# Recommended values for SoftI2C: scl=Pin(22), sda=Pin(21)

# ssd1306 OLED driver, I2C:
# https://randomnerdtutorials.com/micropython-oled-display-esp32-esp8266/
# https://www.engineersgarage.com/micropython-esp8266-esp32-ssd1306/
# Python library for ssd1306:
# https://github.com/adafruit/micropython-adafruit-ssd1306/blob/master/ssd1306.py

# VL53L0X TOF Sensor with ESP32:
# https://www.az-delivery.de/en/blogs/azdelivery-blog-fur-arduino-und-raspberry-pi/digitales-theremin-mit-esp32-in-micropython
# Python library for VL53L0X:
# https://www.grzesina.de/az/theremin/VL53L0X.py
# A def ping(self)-function has been added to vl53l0x.py

# HC-SR04 Ultrasonic Sensor with ESP32:
# https://randomnerdtutorials.com/micropython-hc-sr04-ultrasonic-esp32-esp8266/
# https://github.com/rsc1975/micropython-hcsr04/tree/master
# Python library for HCSR-04:
# https://github.com/rsc1975/micropython-hcsr04/blob/master/hcsr04.py
```

# main.py I

```
from machine import Pin, SoftI2C
import ssd1306
from vl53l0x import VL53L0X
from hcsr04 import HCSR04
from time import sleep

# ESP32 Pin assignment
# Software I2C bus
i2c = SoftI2C(scl=Pin(22), sda=Pin(21))

# Create a ssd1306 "oled" object
oled = ssd1306.SSD1306_I2C(128, 64, i2c)

# Create a VL53L0X "time of flight" object
sensor_VL53L0X = VL53L0X(i2c)
correction_VL = (-70)

# Create a HCSR04 object
sensor_HCSR04 = HCSR04(trigger_pin=5, echo_pin=18, echo_timeout_us=10000)
correction_HC = (0)

print('BootStrapReady')          # send this message in order to get rid of ESP32 bootstrap messages
```

## main.py II

```
# ----- code for oled display -----  
def clearLine(_line):          # for a 128x64 oled-monochrome-display with 6 rows (0 to 5)  
    oled.framebuf.fill_rect(0, 11 * _line, 128, 11, 0)  # width = 128, hight = 11  
  
oled.fill(0)  
oled.invert(0)  
oled.text('EBW', 0, 0)  
# ----- end of code oled display -----  
while True:  
    distance_VL53L0X = sensor_VL53L0X.ping() + correction_VL  
    distance_HCSR04 = (sensor_HCSR04.distance_mm()) + correction_HC  
  
    print('VL:' + str(distance_VL53L0X))  
    print('HC:' + str(distance_HCSR04))  
  
    clearLine(1)  
    clearLine(2)  
    oled.text('VL:' + str(distance_VL53L0X), 0, 11*1)  
    oled.text('HC:' + str(distance_HCSR04), 0, 11*2)  
    oled.show()  
  
    sleep(.5)
```

# REPL Thonny

Shell ×

```
>>> %Run -c $EDITOR_CONTENT
```

```
MPY: soft reboot
```

```
BootStrapReady
```

```
VL:132
```

```
HC:131
```

```
VL:131
```

```
HC:127
```

```
VL:131
```

```
HC:131
```

```
VL:133
```

```
HC:131
```

```
VL:133
```

```
HC:127
```

```
VL:131
```

```
HC:127
```

```
VL:130
```

```
HC:127
```

```
VL:131
```

```
HC:127
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 71, in <module>
```

```
KeyboardInterrupt:
```

```
>>>
```

# Monitor in Python

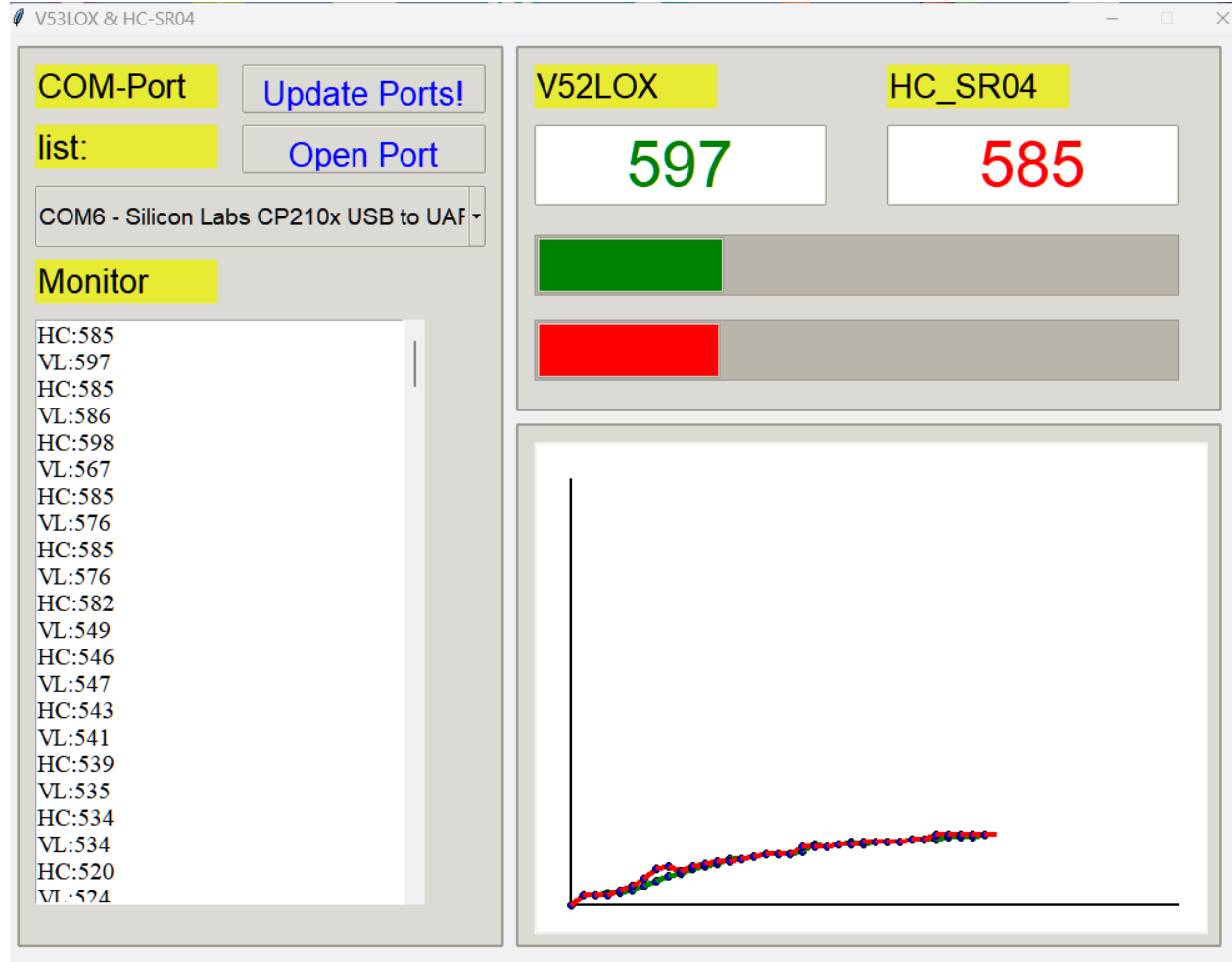
Editor	PyScripter
Python	Python 3.11 vorinstalliert
Projekt	PyInterface
Dateien	PyInterface_1.py
Installierte Libraries	> cmd.exe > pip list
Ausgabe	pip 23.2.1 pyFirmata 1.1.0 pyserial 3.5 (Hier vorhanden, muss evtl. installiert werden) readchar 4.0.5 setuptools 65.5.0
Installation mit pip:	> cmd.exe > pip install pyserial

# PyInterface\_1.py Informationen

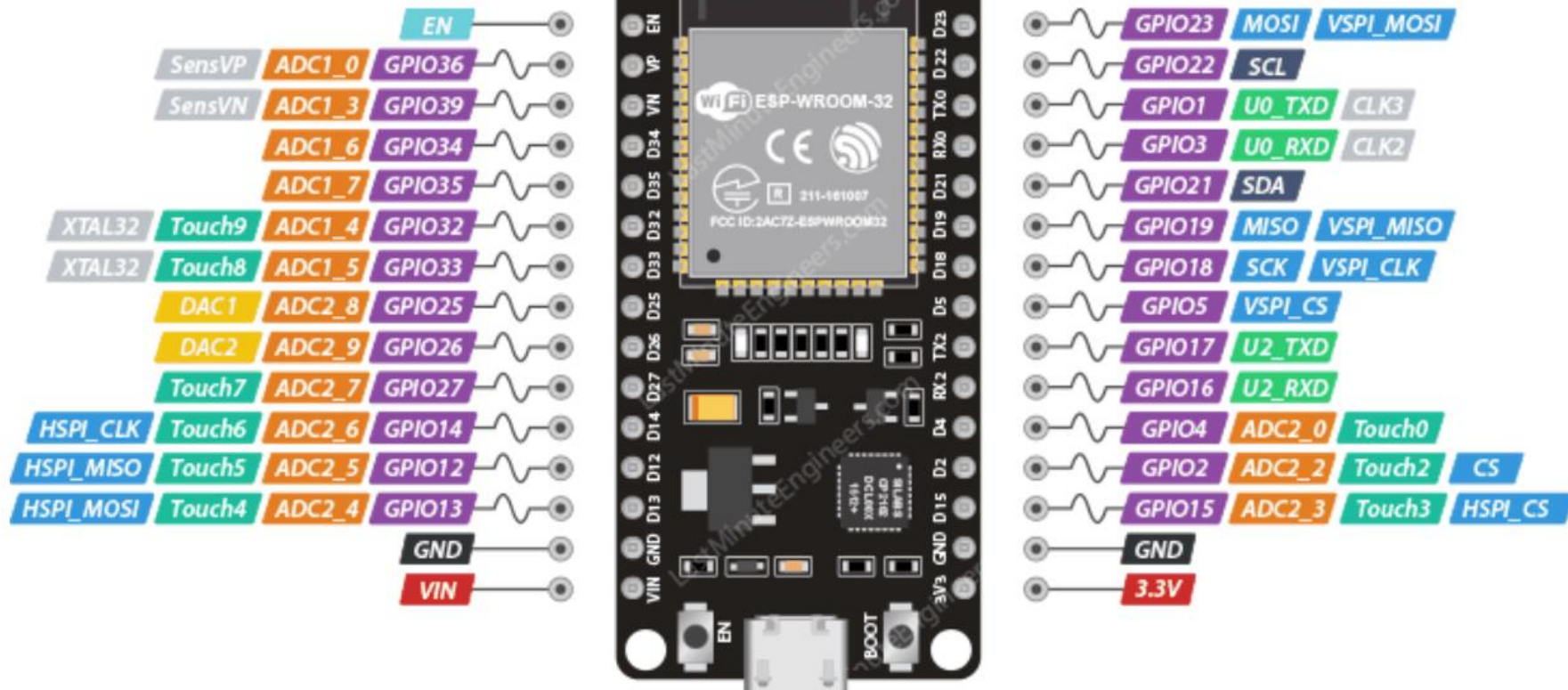
```
# https://tmml.sourceforge.net/doc/tk/keyword-index.html#KW-container
# https://docs.python.org/3/library/tkinter.ttk.html
# https://anzelg.github.io/rin2/book2/2405/docs/tkinter/index.html
# https://www.pythontutorial.net/tkinter/tkinter-button/
# https://www.pythontutorial.net/tkinter/tkinter-combobox/
# https://www.plus2net.com/python/tkinter-Combobox.php
# https://www.inf-schule.de/software/gui/entwicklung_tkinter/auswahl/listbox
# https://pyserial.readthedocs.io/en/latest/shortintro.html
# https://github.com/pyserial/pyserial/issues/655
# https://docs.huihoo.com/tkinter/tkinter-reference-a-gui-for-python/grid.html
# https://www.w3resource.com/python-exercises/tkinter/python-tkinter-canvas-and-graphics-exercise-4.php
# https://tkdocs.com/shipman/ttk-style-layer.html
```

Quellcode auf: [Experiences/HC-SR04 & VL53LOX at main · EKlatt/Experiences · GitHub](#)

# Python-Monitor

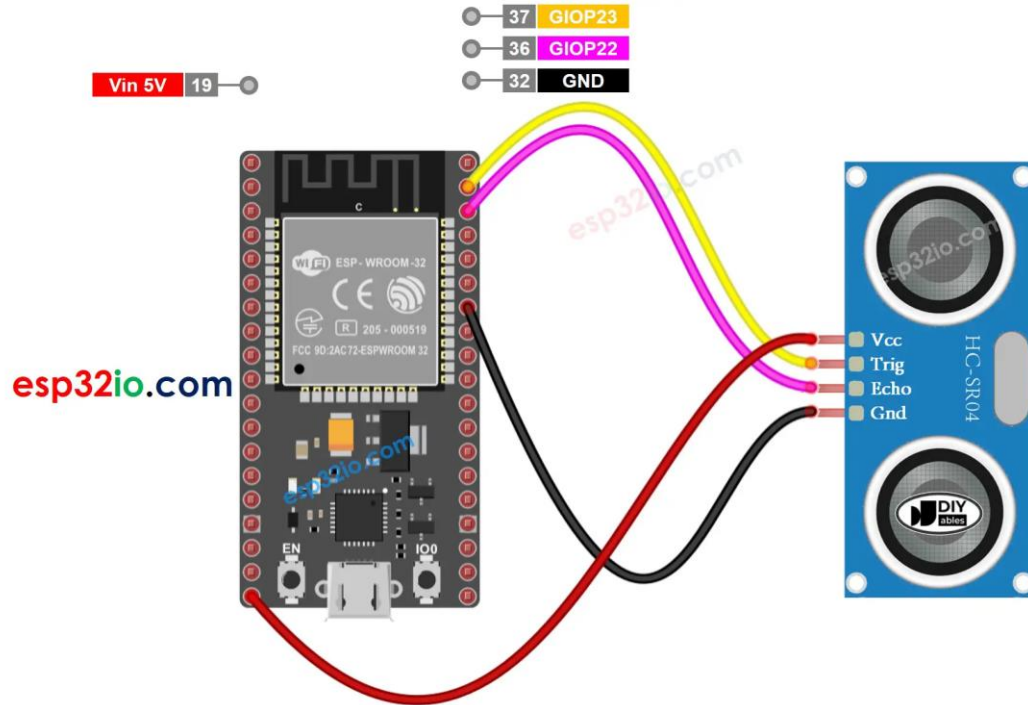


# ESP32 Pinout



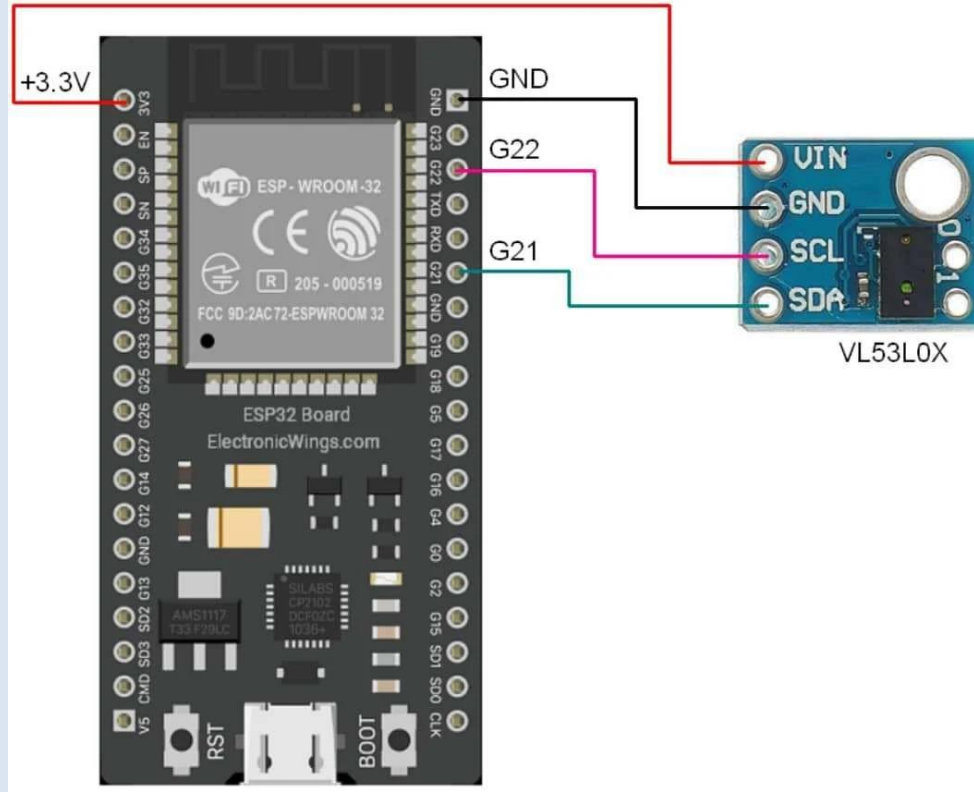


# HC-SR04 Schaltplan



<https://esp32io.com/tutorials/esp32-ultrasonic-sensor>

# VL53L0X Schaltplan



<https://www.electronicwings.com/esp32/vl53l0x-sensor-interfacing-with-esp32>

---

## Links

<https://learn.adafruit.com/adafruit-vl53l0x-micro-lidar-distance-sensor-breakout/downloads>

[https://github.com/adafruit/Adafruit\\_VL53L0X](https://github.com/adafruit/Adafruit_VL53L0X)

<https://learn.adafruit.com/adafruit-vl53l0x-micro-lidar-distance-sensor-breakout/arduino-code>

[adafruit-vl53l0x-micro-lidar-distance-sensor-breakout.pdf](#)

<http://www.d3noob.org/2022/10/connecting-time-of-flight-sensor-to.html>

[https://wiki.hshl.de/wiki/index.php/Ultraschallsensor\\_HC-SR04](https://wiki.hshl.de/wiki/index.php/Ultraschallsensor_HC-SR04)

---