

## Test HC-SR04 & VL53LOX

- Projekt
- Daten
- Ultraschall
- HC-SR04
- HC-SR04 Timing Diagram
- HC-SR04 Leistungstest
- HC-SR04 Berechnung
- VL53LOX
- VL53LOX Anwendungen
- Elektromagnetisches Spektrum
- VL53LOX Block diagram
- VL53LOX Entfernung
- Continuous Wave
- ToF Sichtfeld
- ESP32 Schaltplan
- Python Libraries
- Python auf ESP32
- main.py Informationen
- Versuche
- REPL Thonny
- Termite
- Serielle Kommunikation
- USB-to-TTL
- Python Monitor
- PC-Monitor in Python
- PC Monitor
- PyInterface.py Informationen
- ESP32 Pinout
- Messverfahren
- Links

---

# Projekt

---

- Entfernungsmessung mit Hilfe von Ultraschall und Infrarotlicht.
- Programmiersprache Python/MicroPython.
- Messwerte auf OLED und PC-Monitor ausgeben.
- Vergleich der physikalischen Messmethoden.
- Vor- und Nachteile der Messmethoden.
- Funktionsweisen der Sensoren

GitHub: <https://github.com/EKlatt/Experiences>

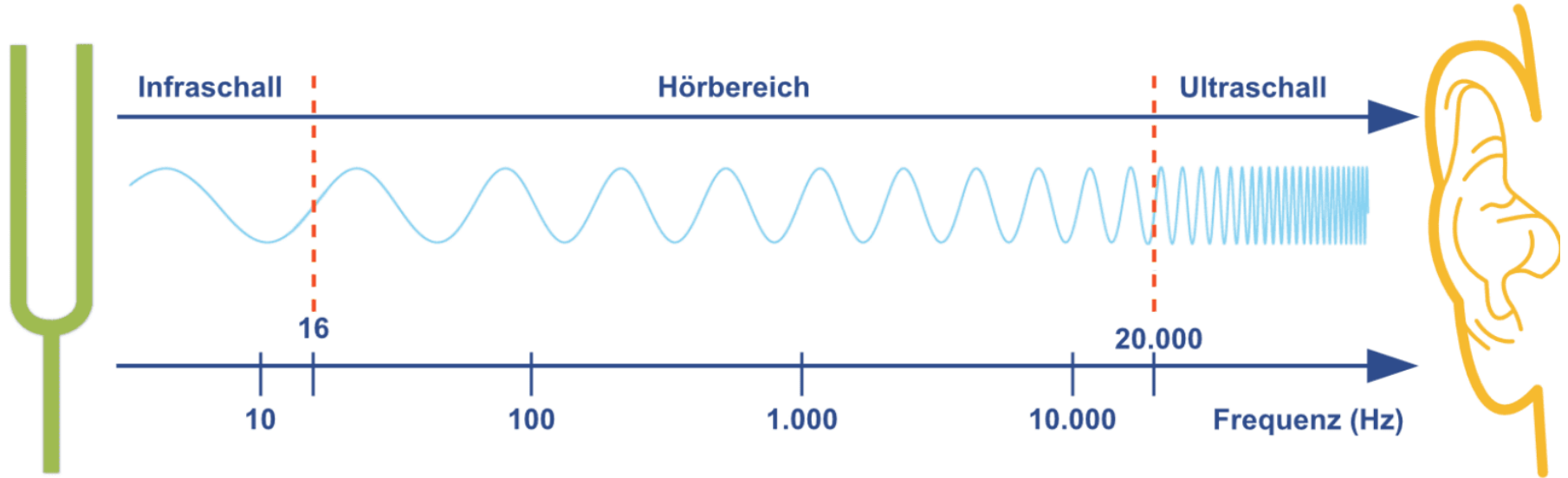
Ordner: HC-SR04 & VL53LOX

---

## Daten

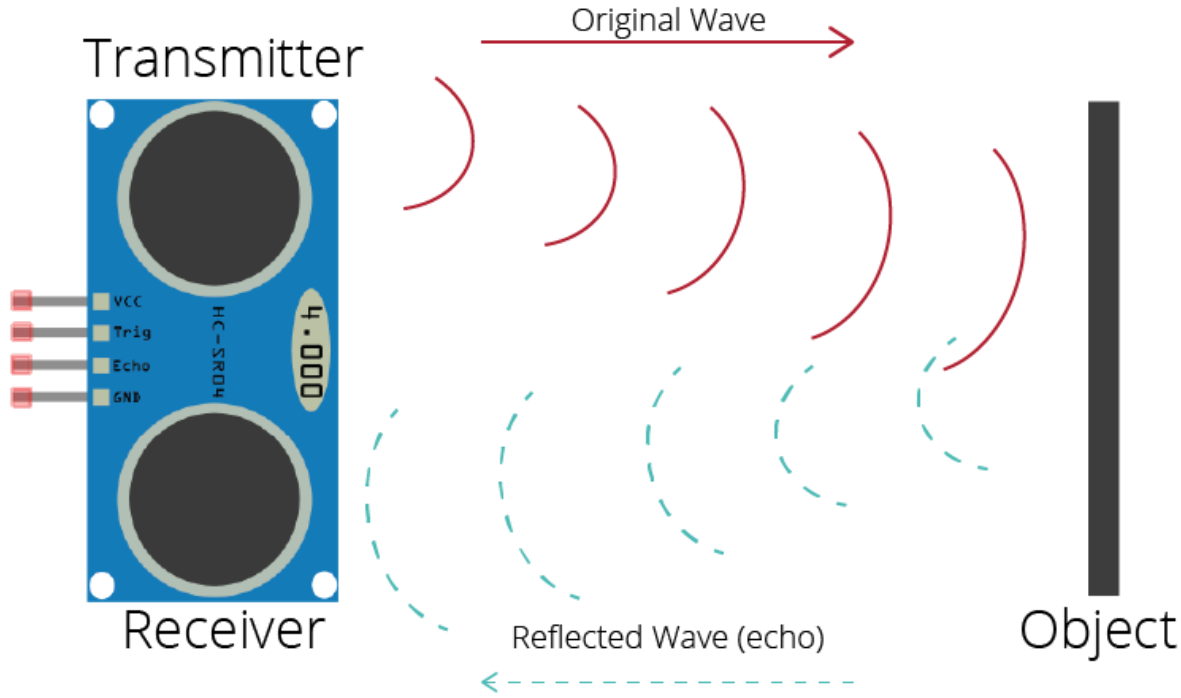
	Ultraschall HC-SR04	Infrarotlaser V53LOX
Maximale Reichweite	4,0 m	Weißes Ziel typisch: 2,0 m (long range)
		Minimum: 1,2 m
		Graues Ziel typisch: 80 cm
		Minimum: 70 cm
Minimale Reichweite	2 cm	> 5 mm
Erkennungswinkel	$\leq 15^\circ$	$25^\circ$
Arbeitsfrequenz	40 kHz	940 nm laser VCXEL
Trigger	10 $\mu$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

# Ultraschall



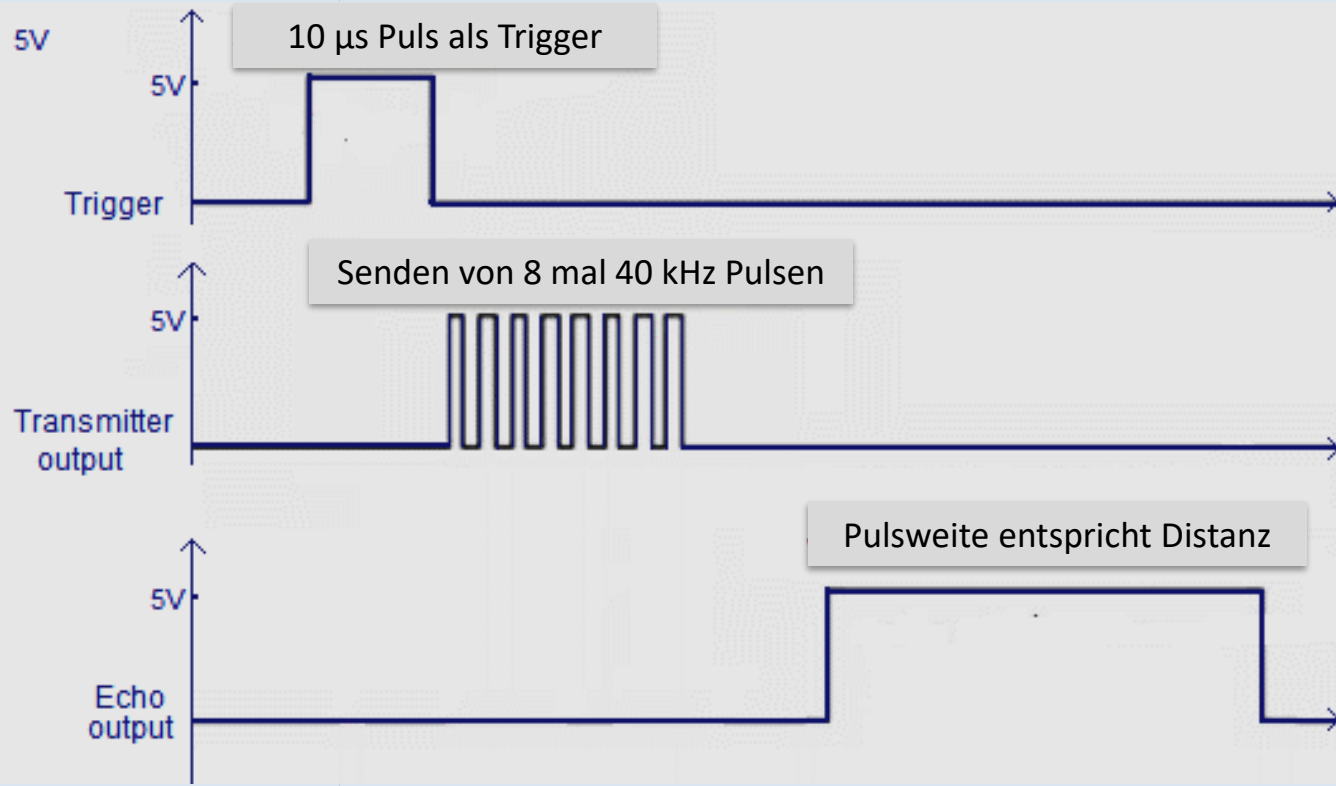
<https://edistechlab.com/ultraschallsensor-hc-sr04-einfach-erklaert/?v=3a52f3c22ed6>

# HC-SR04



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

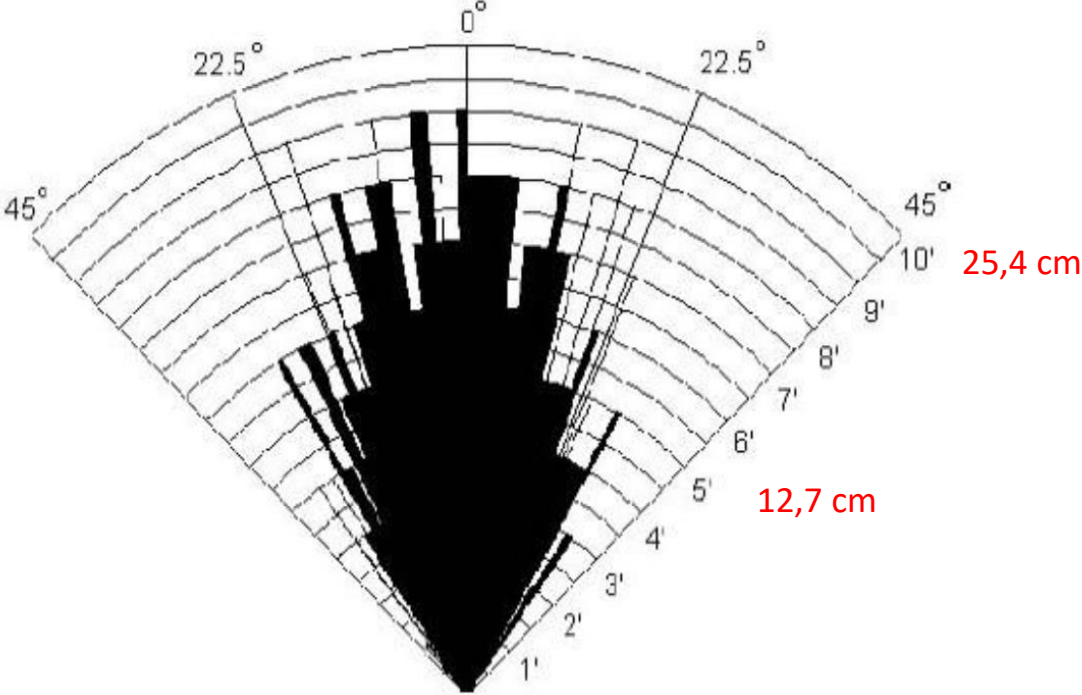
# HC-SR04 Timing Diagram



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

# HC-SR04 Leistungstest

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



## HC-SR04 Berechnung

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, entspricht 13 m)

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)

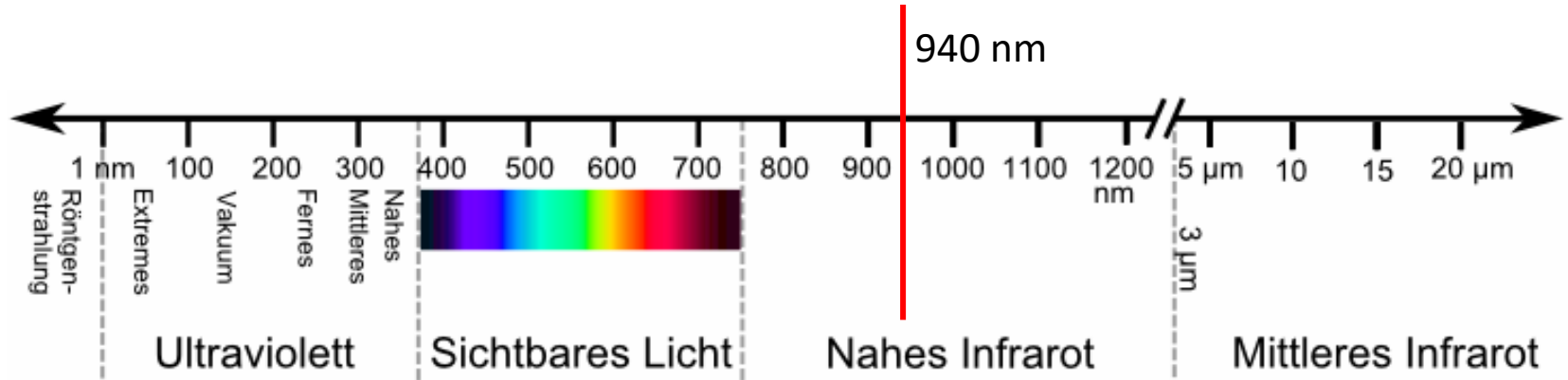
---

## VL53L0X Anwendungen

---

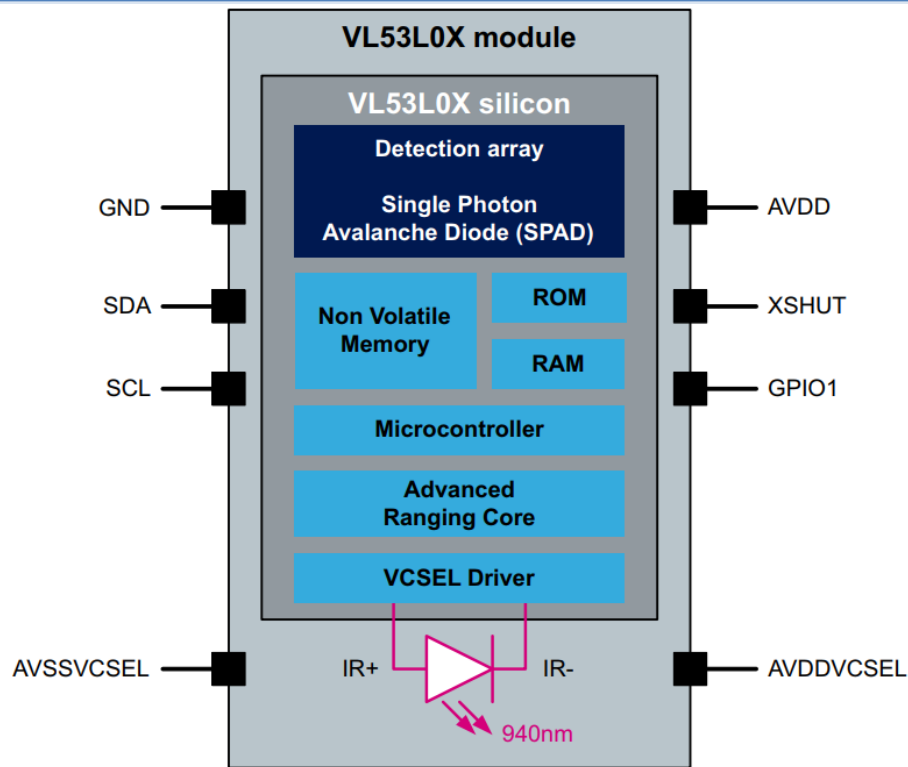
- Hinderniserkennung und -vermeidung in der Robotik.
  - Näherungssensor in Smartphones.
  - Autofokus in Kameras.
  - Gestenerkennung.
  - Erkennung von Handbewegungen (Wasserhähne).
  - Benutzererkennung in Laptops.
-

# Elektromagnetisches Spektrum



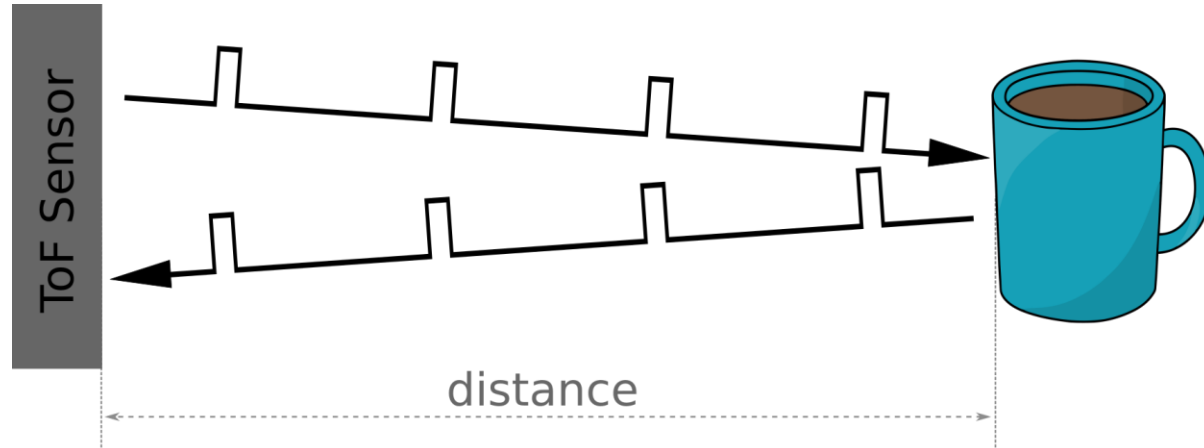
<https://psi.physik.kit.edu/313.php>

# VL53L0X Block diagram



<https://www.mouser.com/pdfdocs/enDM00270461.pdf>

# VL53LOX Entfernung

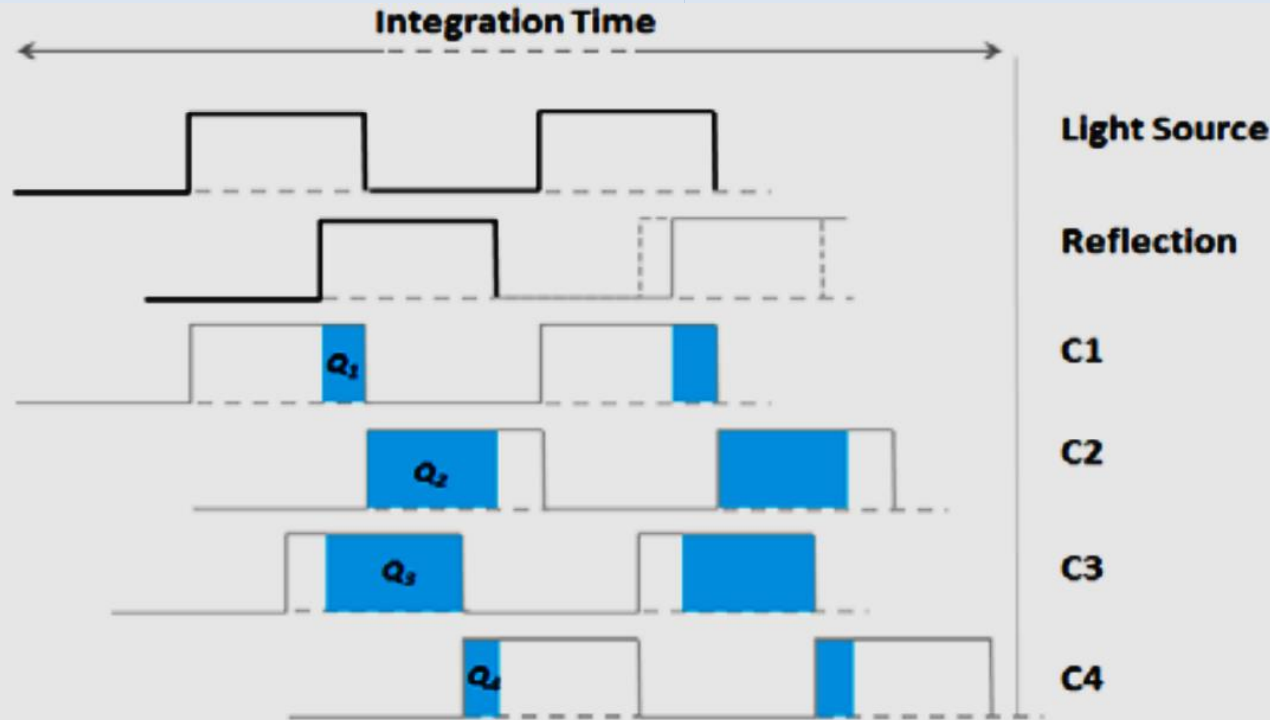


$$\text{distance} = \frac{\text{Speed of Light} \times \text{Time of Flight}}{2}$$

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

# Continuous Wave

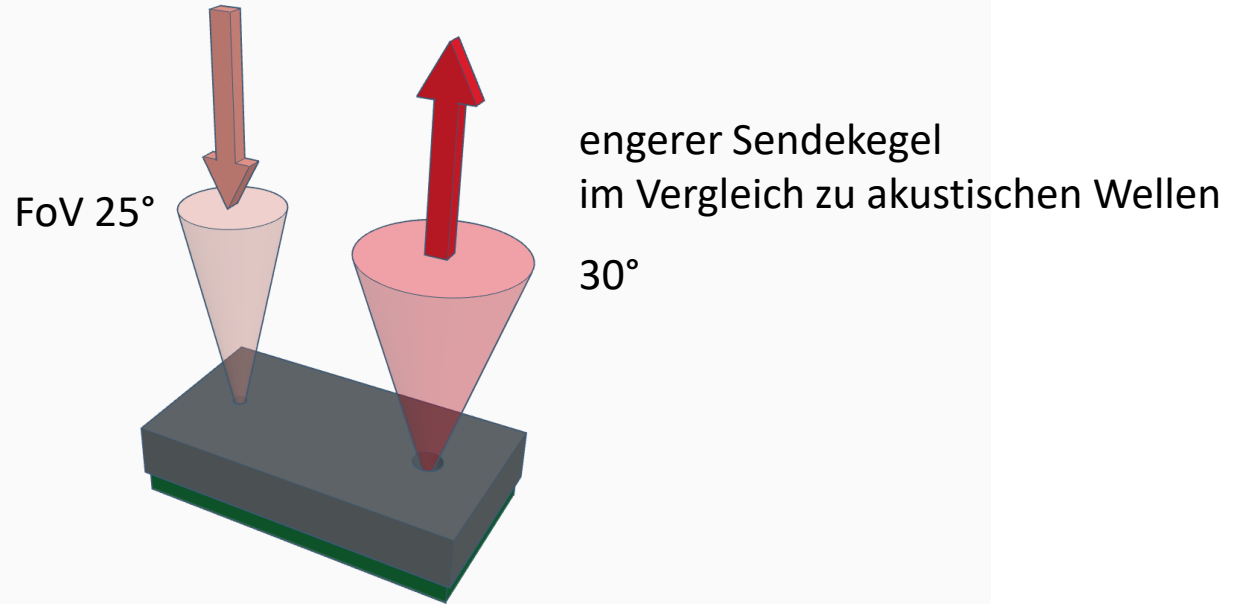
Continuous-wave Method?



$$\varphi = \arctan \left( \frac{Q_3 - Q_4}{Q_1 - Q_2} \right)$$

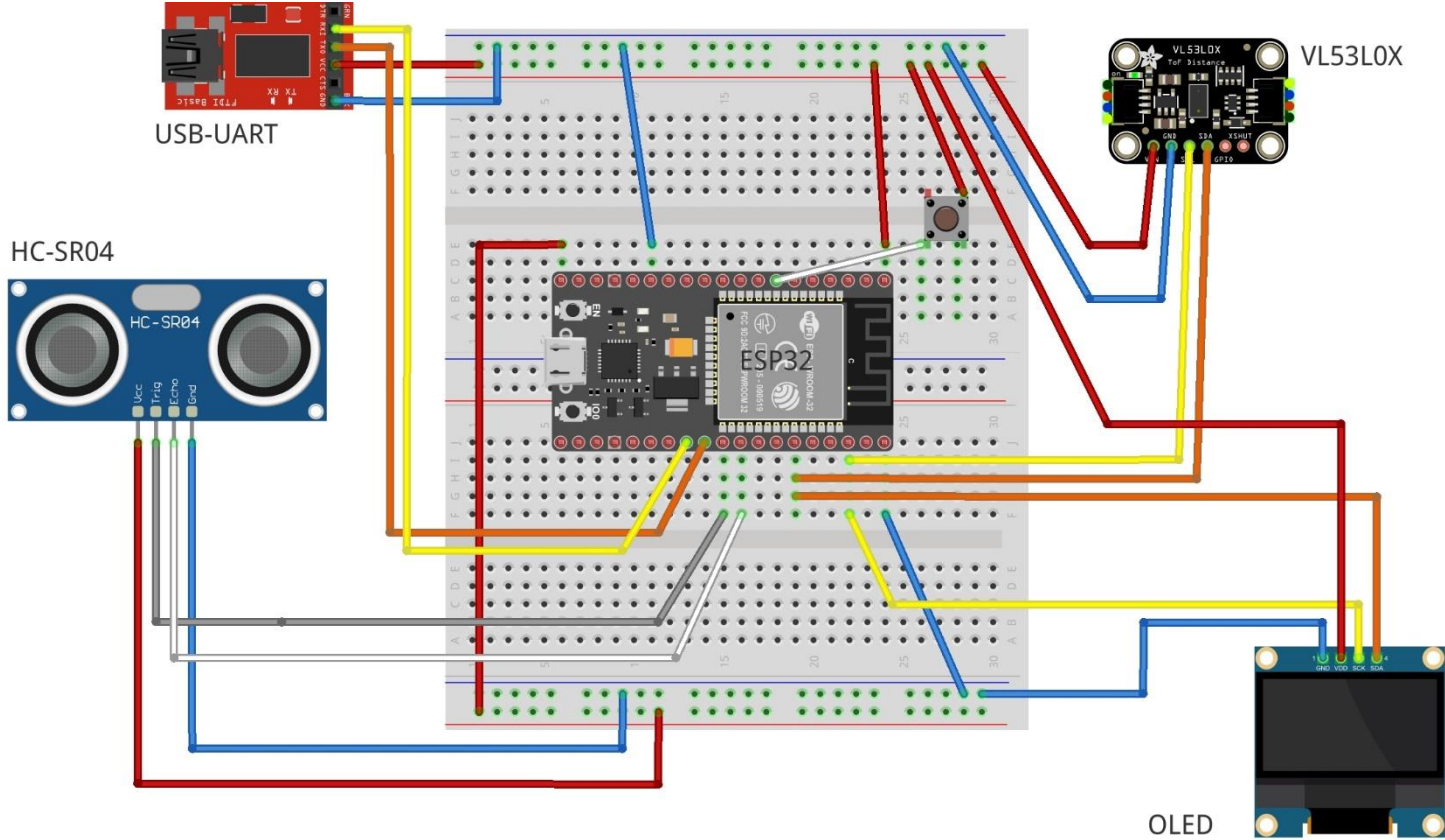
$$d = \frac{c}{4\pi f} \varphi.$$

<https://www.ti.com/lit/wp/sloa190b/sloa190b.pdf>



FoV: full field of view

# ESP32 Schaltplan



OLED



---

## 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  
(neuere 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)


Microcontroller  
(device)


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
# https://www.electronicwings.com/esp32/vl53l0x-sensor-interfacing-with-esp32
# 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://esp32io.com/tutorials/esp32-ultrasonic-sensor
# https://github.com/rsc1975/micropython-hcsr04/tree/master
# Python library for HCSR-04:
# https://github.com/rsc1975/micropython-hcsr04/blob/master/hcsr04.py

# UART duplex serial communication
# https://docs.micropython.org/en/latest/library/machine.UART.html#machine.UART
# https://docs.micropython.org/en/latest/esp32/quickref.html#uart-serial-bus
```

---

## Versuche

Einfluss von

- farbigen Hintergründen
  - Oberflächenbeschaffenheit
  - absorbierenden Flächen
  - Sensorkegel (Winkel)
  - Umgebungslicht
-

# REPL Thonny

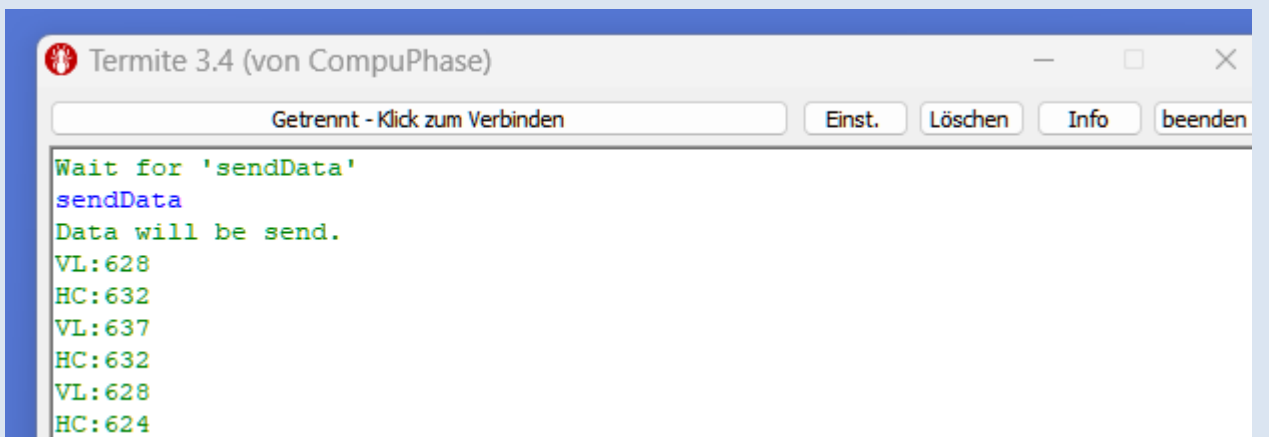
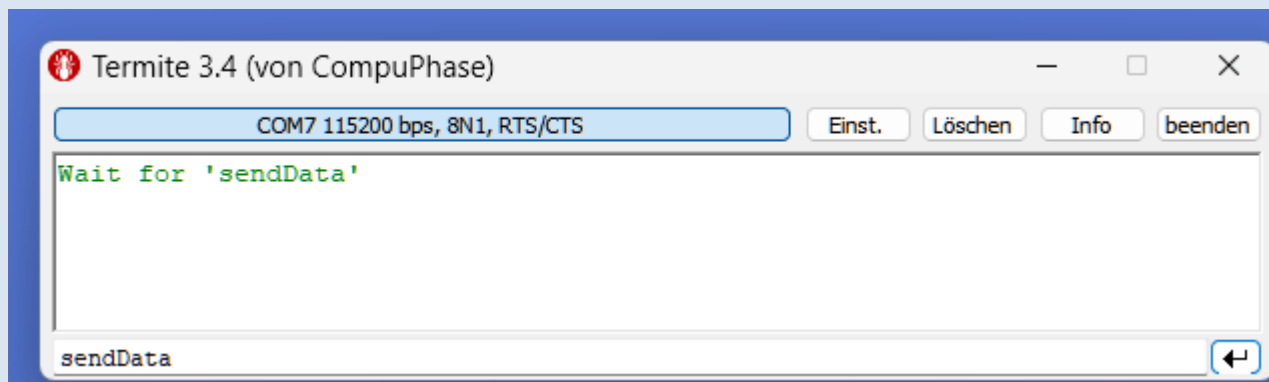
Besonderheit:  
Taster zum Start  
drücken.

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

MPY: soft reboot
Press Button or sendData
Data will be send.
VL:254
HC:250
VL:261
HC:249
VL:267
HC:249
VL:261
HC:249
Traceback (most recent call last):
  File "<stdin>", line 105, in <module>
KeyboardInterrupt:
>>>
```

# Termite

Besonderheit:  
Eingabe des Befehls  
„sendData“.



## Serielle Kommunikation

### Problem

- MicroPython läuft als Interpreter auf dem Mikrokontroller ( $\mu\text{C}$ ).
- Der Thonny-Editor ist über USB mit dem  $\mu\text{C}$  verbunden.
- Der Thonny-Editor greift auf das Dateisystem des  $\mu\text{C}$  zu.
- Hier liegt die auszuführende „main.py“.
- **Die USB-Schnittstelle kann nicht zur Kommunikation genutzt werden.**

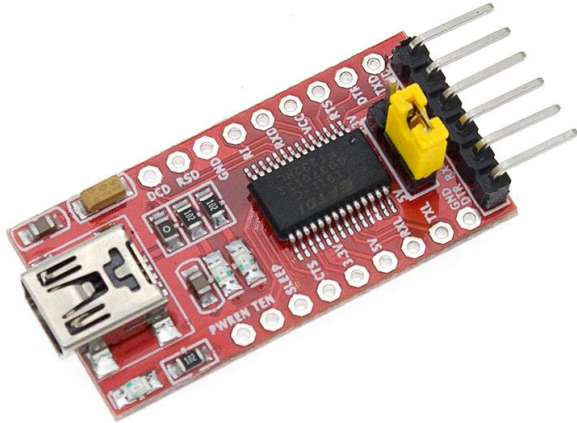
### Abhilfe

- UART-TTL Adapter
- Zusätzliche serielle Schnittstelle.

### ESP32

Vordefinierte UART2 gewählt.

# USB-to-TTL



	UART0	UART1	UART2
tx	1	10	17
rx	3	9	16

## Code:

```
# Create a UART object for EPS32 with UART2
# GPIO17: tx=17,
# GPIO16: rx=16
uart = UART(2, 115200)
uart.write("Wait for 'sendData'\n")
```

[FT232RL \(kompatibel\) | USB-to-TTL | 3,3V / 5V | Serial UART FTDI Arduino ESP32 | eBay](#)  
<https://docs.micropython.org/en/latest/esp32/quickref.html>



---

# Python Monitor

Idee	<ul style="list-style-type: none"><li>• Eine graphische Oberfläche (GUI) zur Darstellung von Daten.</li></ul>
------	---

Alternativen	<ul style="list-style-type: none"><li>• Python-GUI</li><li>• MicroPython mit Webinterface</li><li>• MicroPython mit MQTT und Node-Red</li></ul>
--------------	---

Python Tools	<ul style="list-style-type: none"><li>• Klassen „tkinter“ und „ttkinter“</li></ul>
--------------	--

Nachteil	Deutlich größerer Aufwand als z.B. SmallBasic.
----------	--

Vorteil	Python als gemeinsame Programmiersprache!
---------	---

# PC-Monitor in Python

Editor	PyScripter
Python	Python 3.11 vorinstalliert (Pfad im Environment bekannt)
Projekt	PyInterface
Dateien	PyInterface.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

# PC Monitor

## Funktionen:

- Portliste erneuern
- Port öffnen
- Port schließen
  
- Anzeige Ports
- Ausgabe der Daten
- Einzelwertanzeige
- Balkenanzeige
- xy-Diagramm

Monitor V53LOX & HC-SR04

COM-Port

COM7 - USB Serial Port (COM7)

Monitor

HC:482  
VL:528  
HC:477  
VL:519  
HC:478  
VL:527  
HC:478  
VL:528  
HC:477  
VL:524  
HC:477  
VL:526  
HC:477  
VL:526  
HC:482  
VL:501  
HC:478  
VL:485  
HC:460  
VL:481

V52LOX **528**

HC-SR04 **482**

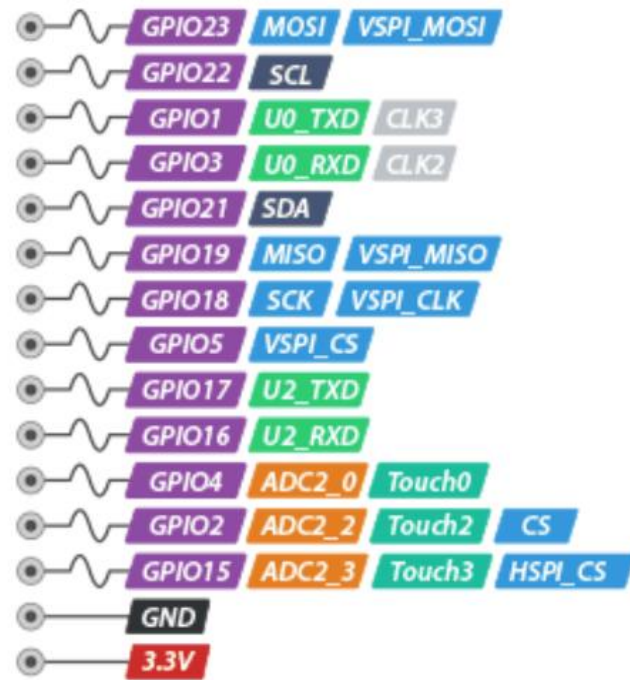
xy-Diagramm

## PyInterface.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://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](#)

# ESP32 Pinout



---

## Messverfahren

<https://wolles-elektronikkiste.de/en/vl6180x-tof-proximity-and-ambient-light-sensor>

<https://wolles-elektronikkiste.de/en/vl53l0x-and-vl53l1x-tof-distance-sensors>

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

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

<https://www.ti.com/lit/wp/sloa190b/sloa190b.pdf>

---

## 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)

<https://www.az-delivery.de/blogs/azdelivery-blog-fur-arduino-und-raspberry-pi/wie-gross-bin-ich-koerpergroesse-messen-mit-vl53l0x-tof-sensor-und-sprachausgabe-in-micropython>