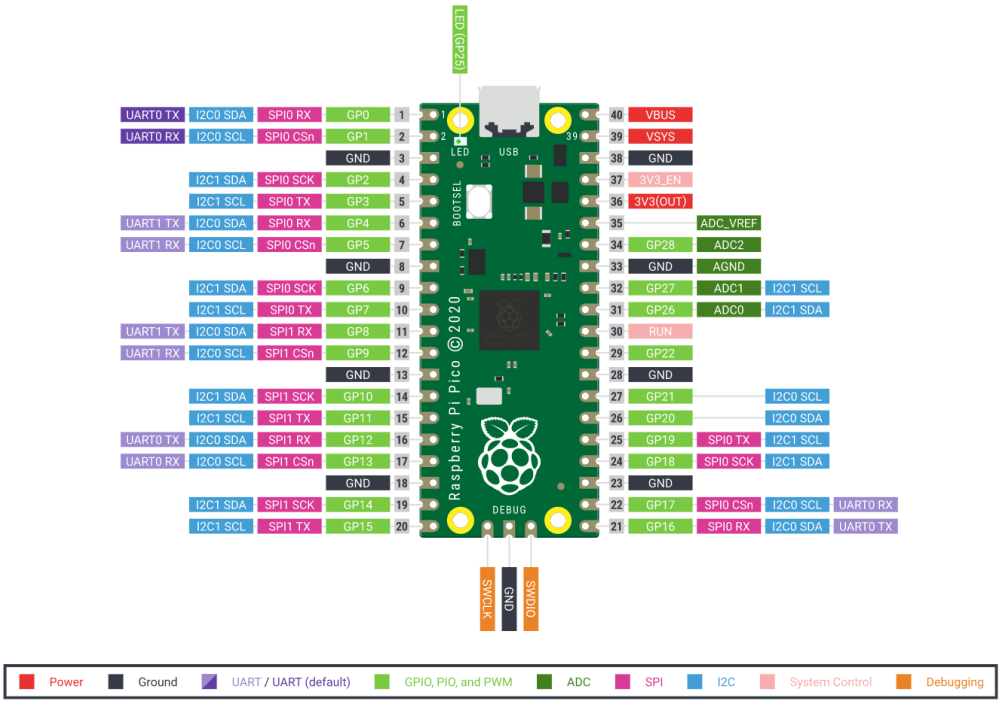
# 3 Afstand meten




## 3.1 Einddoel

Je gaat een apparaat maken dat de omgeving scant en op het scherm toont waar er objecten staan in de omgeving.

Als je klaar bent met de opdracht lever je in It's learning in:

- De definitieve python code.
- Een mp4-filmpje met een demo van je programma.

Als dit gedaan is laat je de opdracht zien aan de docent zodat deze de opdracht kan goedkeuren.

## 3.2 Kennis

Voor deze opdracht heb je kennis nodig van microPython.

## 3.3 Benodigdheden

Voor deze opdracht heb je de onderstaande materialen nodig. Controleer aan het begin of al deze spullen aanwezig zijn. Bij het opruimen dien je weer te controleren of alles aanwezig is. Indien er iets defect is geraakt moet je de docent op de hoogte brengen.

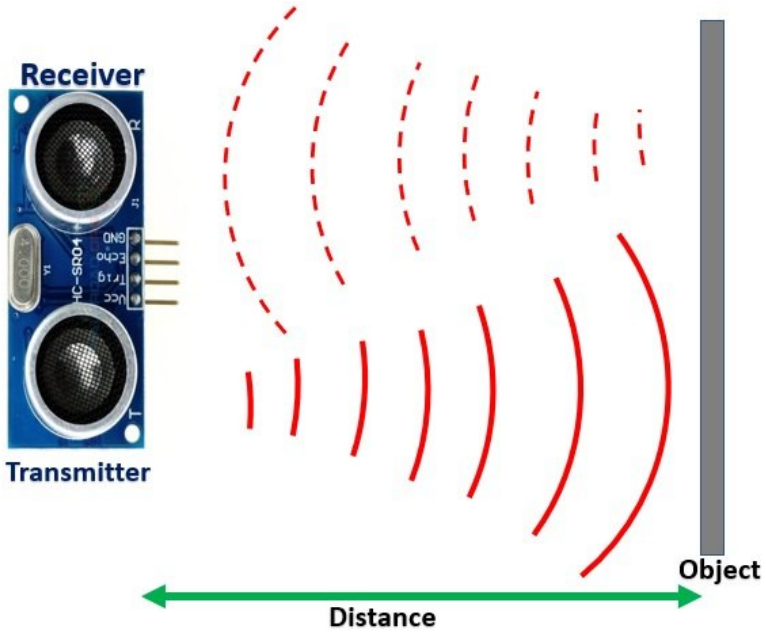
- Raspberry Pi Pico – H
- UBS-usb mini kabel
- Ultrasoon sensor HC-SR04
- Micro servomotor SG90

## 3.4 Voorbeeldcode en libraries

In It's Learning kan me de onderstaande voorbeeldcode vinden om je op weg te helpen.

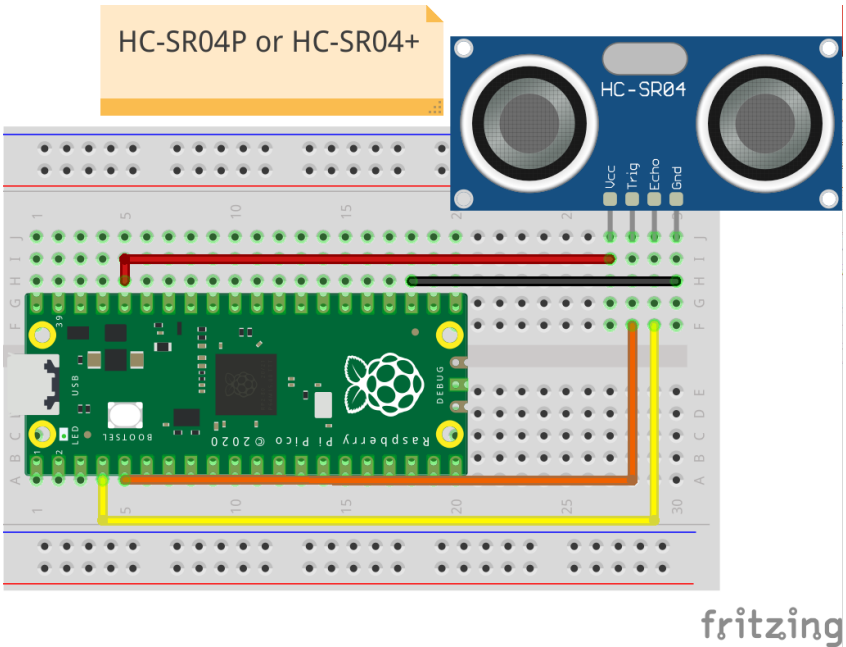

## 3.5 Opdracht

Met de HC-SR04 kunnen we door middel van geluidsgolven de afstand bepalen.




Een ultrasoon sensor meet de tijd dat een signaal is verzonden totdat deze is ontvangen. Dit is dus tweemaal de afstand die we uiteindelijk moeten hebben. De snelheid van geluid door lucht bij 20ºC is 340 m/s. Met deze informatie kan de afstand berekend worden.
De tijd die we meten is in microseconden, 1 microseconden 1 µs = 0.000001 seconden.

$$2s = v * t \rightarrow s = \frac{340\ in\ {}^{m}/_{s}}{2} * t\ in\ s$$
$$= 170 * t$$
$$= 0.17[\frac{mm}{\mu s}] * t\ [\mu s]$$

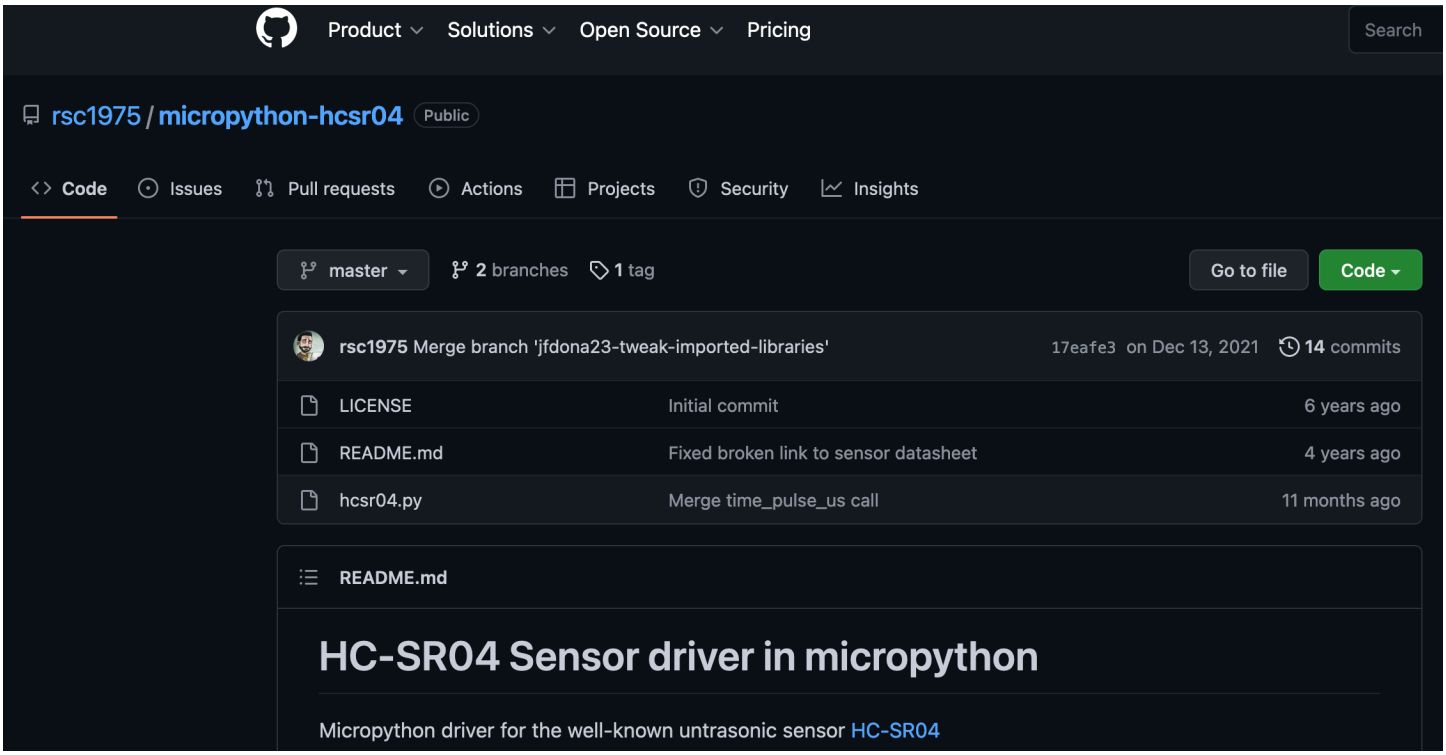



We sluiten de ultrasoon sensor als volgt aan. (De Vcc staat hieronder op 3.3V, dit zou beter op de Vbus aangesloten kunnen worden.)

| HC-SR04 | Pico |
|---|---|
| VCC | 3V3 |
| GND | GND |

| Trig | GP2 |
|---|---|
| Echo | GP3 |

De berekening die we hierboven hebben gezien hoeven we niet zelf te programmeren. We gaan hiervoor een librarie gebruiken. We kunnen deze van GitHub downloaden: https://github.com/rsc1975/micropython-hcsr04




Het bestand moet op je Pico opgeslagen worden met de naam hcsr04.py
Met de onderstaande code kan meet je de afstand. Let op dat de correcte trigger en echo pinnen nog ingevoerd moeten worden.

```
from machine import Pin
from hcrs04 import HCSR04
from time import sleep

SLEEP       = 1
TRIGGER_PIN = 21
PIN_ECHO    = 20

_sensor = HCSR04(trigger_pin=TRIGGER_PIN, echo_pin=PIN_ECHO, echo_timeout_us=10000)

def measure():
    return _sensor.distance_cm()

if __name__ == "__main__":
    while True:
        print('Distance:', measure(), 'cm', '|')
        time.sleep(SLEEP)
```

### 3.5.1 Opdracht 1: Afstand weergeven met Ledjes

Met de leds op de break-out board laten we zien op welke afstand een voorwerp is. Voor iedere 4 cm dat een voorwerp verwijderd is gaat er een ledje uit op het break-out board. Dus als een voorwerp op 4cm wordt gemeten gaan er 15 leds aan. Als het voorwerp op 8 cm wordt gemeten gaan er 14 leds aan. Wordt een voorwerp op 60 cm of meer gemeten dan gaan er geen leds aan. Als een voorwerp binnen de 4cm is zal de sensor deze niet detecteren. Er zullen dan geen leds aangaan.

### 3.5.2 Opdracht 2: Servomotor en afstandsmeting

We laten een servomotor steeds van 0 naar 180 graden in stapjes van 5 graden draaien, en weer terug. Bij iedere stap meten we de afstand naar een eventueel voorwerp. De gemeten afstanden en de hoek worden op het scherm getoond. Op de leds kan je weer zien of er een object gedetecteerd wordt.
Met de onderstaande code kan je een servomotor waarvan de data pin op 28 is aangesloten draaien naar 90 graden. Om te weten wat de draadjes doen kan je met het nummer aan de zijkant van de servomotor op internet de aansluiting opzoeken. Sluit de servomotor aan en experimenteer met het draaien van de hoek van de servomotor.

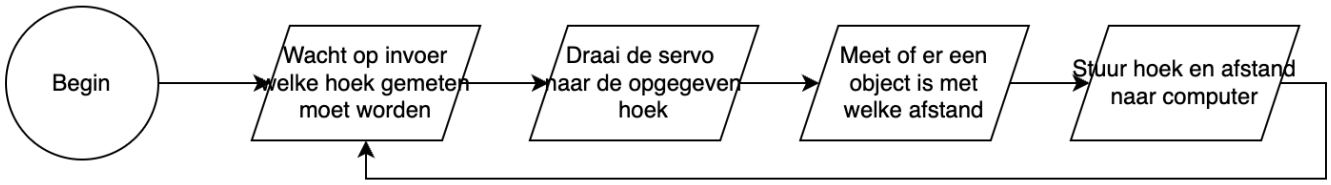
```
import machine
from time import sleep

class Servo:
    def __init__(self, MIN_DUTY=300000, MAX_DUTY=2300000, pin=0, freq=50):
        self.pwm = machine.PWM(machine.Pin(pin))
        self.pwm.freq(freq)
        self.MIN_DUTY = MIN_DUTY
        self.MAX_DUTY = MAX_DUTY

    def rotateDeg(self, deg):
        deg = max(0, min(deg, 180))
        duty_ns = int(self.MAX_DUTY - deg * (self.MAX_DUTY-self.MIN_DUTY)/180)
        self.pwm.duty_ns(duty_ns)

servo = Servo(pin=28)
servo.rotateDeg(90)
```

### 3.5.3 Opdracht 3: Knoppen gebruiken

De hoek van de servomotor is in de vorige opdracht steeds in stappen ingesteld. Verander de code zodat je met 2 drukknoppen de servomotor of maar links of naar rechts kan sturen.

### 3.5.4 Opdracht 4: Tonen op computer

De gegevens die we meten met de pico gaan we sturen naar de computer. Dit doen we via seriële communicatie. We moeten nu we de meting iets veranderen. De computer gaat vragen dat de servomotor op een hoek gaat staan en dan wordt de meting gedaan. Deze waarde wordt dan aan de computer teruggegeven.

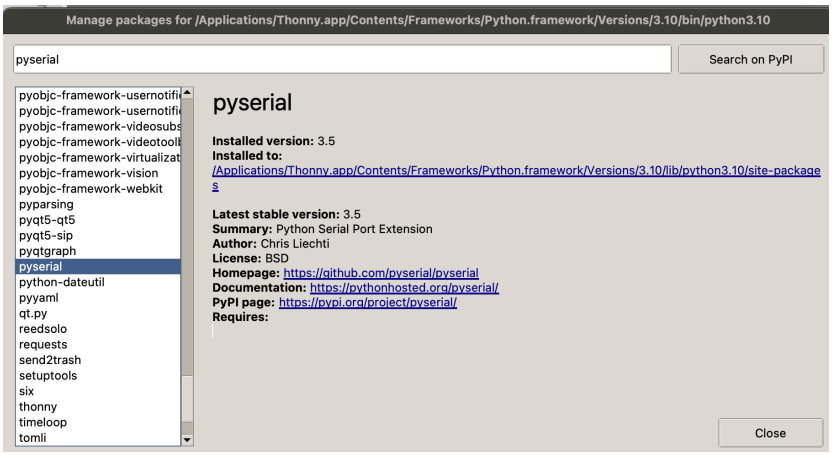



Vanaf de Pico worden de berichten verstuurd met de hoek en de afstand dat een object eventueel gemeten wordt. Er is alleen een groot probleem. De seriële communicatie wordt door Thonny gebruikt om te code naar de pico te sturen, en weer te ontvangen om op het scherm te plaatsen. Dit omzeilen we door onderstaande code met de naam main.py op de pico op te slaan. Het programma main.py wordt altijd uitgevoerd vanaf het moment dat er spanning op de pico komt. We kunnen vanaf dat moment op de computer de gegevens ontvangen. We kunnen alleen niet meer kijken wat er op de pico gebeurd.

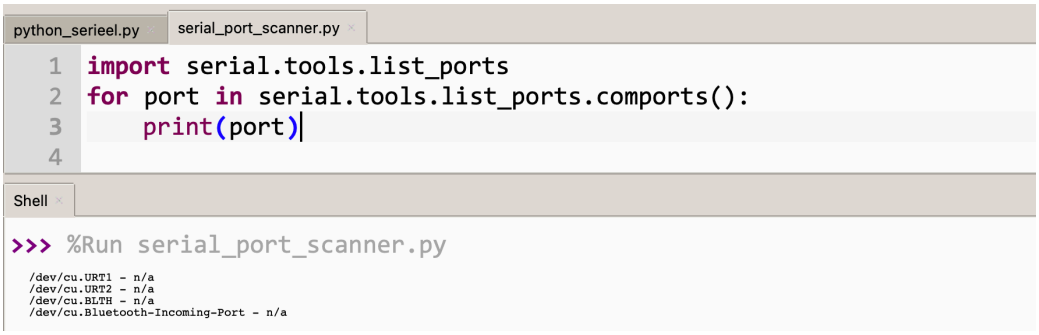
```
#save on pico with main.py
#only works when NOT connected to IDE
import machine
from machine import I2C, Pin
import time
import uos

# need this UART to read from BME and be able to send data to local computer
uart = machine.UART(0, baudrate=115200)
uart.init(115200, bits=8, parity=None, stop=1, tx=Pin(0), rx=Pin(1))
uos.dupterm(uart)

while True:
    print('{:.1f} C,{:.1f} %,{:.1f} hPa'.format(42,69,1234))
    time.sleep(1)

```

De volgende stap is de seriële communicatie voor de computer in orde maken.

Op de computer installeren we eerst pyserial. Dit kan door pip install pyserial in de shell in te typen, of via de install manager




Verder moeten we weten op welke seriële poort de Raspberry pico aangesloten zit. Dat doen we met onderstaande code.




Vervolgens vullen we de seriële poort in, in onderstaand programma. Dit programma gaat luisteren naar de seriële poort of daar een bericht ontvangen wordt.

```
import serial

# Define the serial port and baud rate
serial_port = '/dev/ttyUSB0'  # Update this with your actual serial port
baud_rate   = 115200

ser = serial.Serial(serial_port, baud_rate)             # Create a serial object
try:
    while True:
        line = ser.readline().decode('utf-8').strip()   # Read a line from the serial port
        print(line)                                     # Print the received data
except KeyboardInterrupt:
    # Close the serial port when the program is interrupted (Ctrl+C)
    ser.close()
    print("Serial port closed.")

```

Kijken we nu naar de output in de shell dan zien we dat deze gegevens ontvangen worden door het programma.

In it's learning staat het programma pygame_scanner.py. Dit programma laat een radarbeeld zien. In dit programma staat een functie measure, Deze functie laat nu nog default code zien. Deze functie bouwen we om dat de data die we net binnen zagen komen wordt getoond op het radarbeeld.

```
30 _angle = 0
31 def measure():
32     global _angle
33     #data = ultrasoon.readDistance()
34     hoek, afstand = (_angle, random.randint(5, 75) * DISTANCE_MAX / 75)
35     print(hoek, afstand)
36     _angle += 5
37     if _angle > 360:
38         _angle = 0
39     return hoek, afstand * DISTANCE_MAX / 75
```