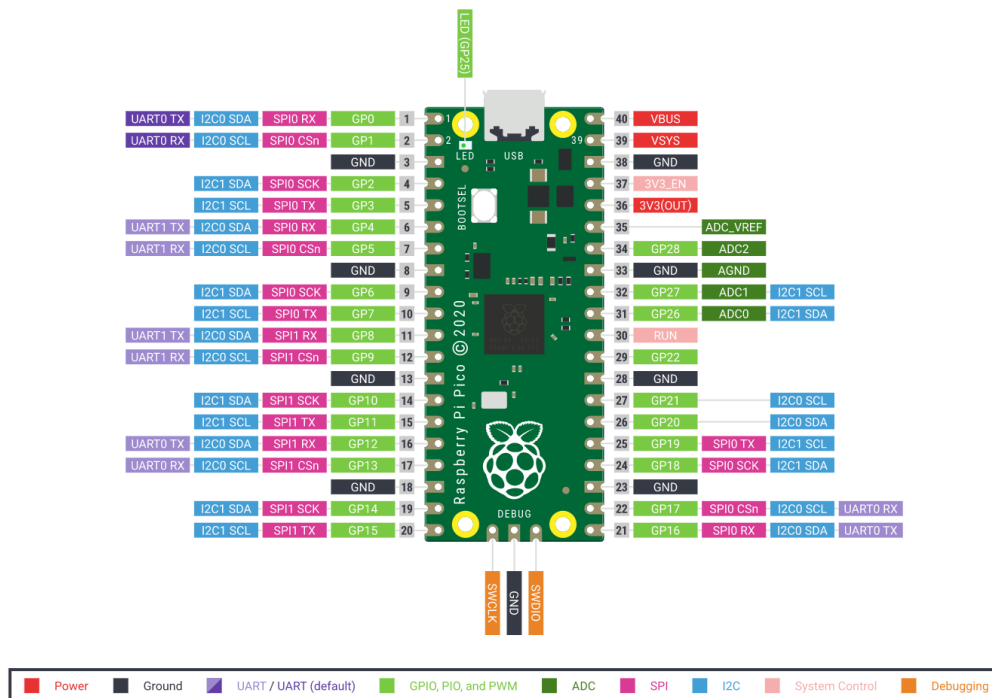


# 3 Afstand meten



## 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.

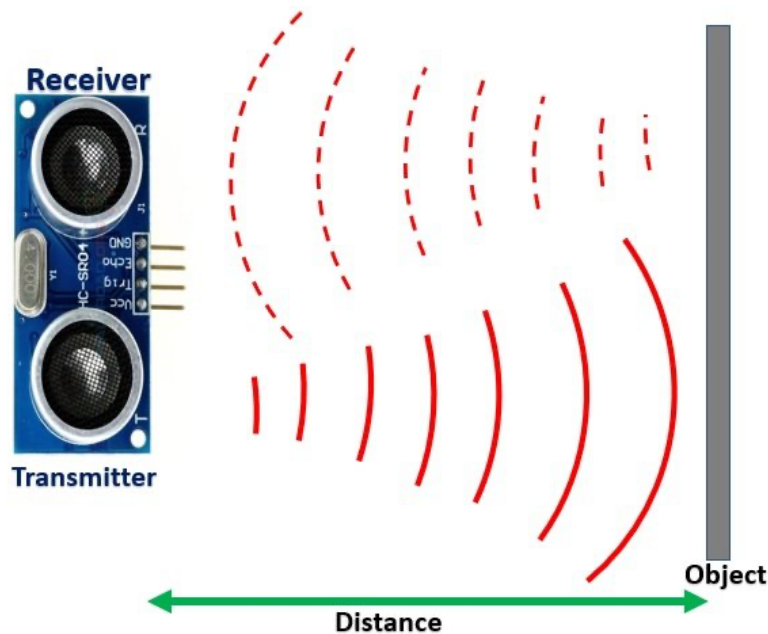
## 3.3 Benodigheden

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 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 20°C is 340 m/s. Met deze informatie kan de afstand berekend worden.

De tijd die we meten is in microseconden, 1 microseconden 1  $\mu s$  = 0.000001 seconden.

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

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-SR04P or HC-SR04+

HC-SR04	Pico
VCC	3V3
GND	GND
Trig	GP2
Echo	GP3

fritzing

De berekening die we hierboven hebben gezien hoeven we niet zelf te programmeren. We gaan hiervoor een bibliotheek gebruiken. We kunnen deze van GitHub downloaden:

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

Product Solutions Open Source Pricing Search

rsc1975 / micropython-hcsr04 Public

Code Issues Pull requests Actions Projects Security Insights

master 2 branches 1 tag Go to file Code

rsc1975 Merge branch 'jfdona23-tweak-imported-libraries' 17eafe3 on Dec 13, 2021 14 commits

LICENSE	Initial commit	6 years ago
README.md	Fixed broken link to sensor datasheet	4 years ago
hcsr04.py	Merge time_pulse_us call	11 months ago

README.md

## HC-SR04 Sensor driver in micropython

Micropython driver for the well-known untrasonic sensor [HC-SR04](#)

rsc1975 / micropython-hcsr04 Public Notifications Fork

Code Issues Pull requests Actions Projects Security Insights

master micropython-hcsr04 / hcsr04.py /<> Jump to Go to file

rsc1975 Merge time\_pulse\_us call Latest commit 481cf40 on Dec 13, 2021 History

3 contributors

82 lines (72 sloc) 3.32 KB

Raw Blame

```
1 from machine import Pin, time_pulse_us
2 from utime import sleep_us
3
4 __version__ = '0.2.1'
5 __author__ = 'Roberto Sánchez'
6 __license__ = "Apache License 2.0. https://www.apache.org/licenses/LICENSE-2.0"
7
8 class HCSR04:
9     """
10     Driver to use the untrasonic sensor HC-SR04.
11     The sensor range is between 2cm and 4m.
12
13     The timeouts received listening to echo pin are converted to OSError('Out of range')
```

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.

```

1 from machine import Pin
2 from hcsr04 import HCSR04
3 from time import sleep
4
5 SLEEP      = 1
6 TRIGGER_PIN = 21
7 PIN_ECHO   = 20
8
9 _sensor = HCSR04(trigger_pin=TRIGGER_PIN, echo_pin=PIN_ECHO, echo_timeout_us=10000)
10
11 def measure():
12     return _sensor.distance_cm()
13
14 if __name__ == "__main__":
15     while True:
16         print('Distance:', measure(), 'cm', '|')
17         time.sleep(SLEEP)
18

```

### 3.4.1 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.

### 3.4.2 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.

```

1 import machine
2 from time import sleep
3
4 class Servo:
5     def __init__(self, MIN_DUTY=300000, MAX_DUTY=2300000, pin=0, freq=50):
6         self.pwm = machine.PWM(machine.Pin(pin))
7         self.pwm.freq(freq)
8         self.MIN_DUTY = MIN_DUTY
9         self.MAX_DUTY = MAX_DUTY
10
11     def rotateDeg(self, deg):
12         deg = max(0, min(deg, 180))
13         duty_ns = int(self.MAX_DUTY - deg * (self.MAX_DUTY-self.MIN_DUTY)/180)
14         self.pwm.duty_ns(duty_ns)
15
16 servo = Servo(pin=28)
17 servo.rotateDeg(90)

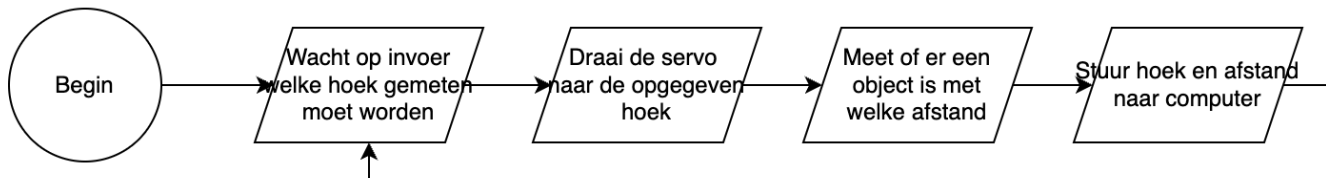
```

### 3.4.3 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.

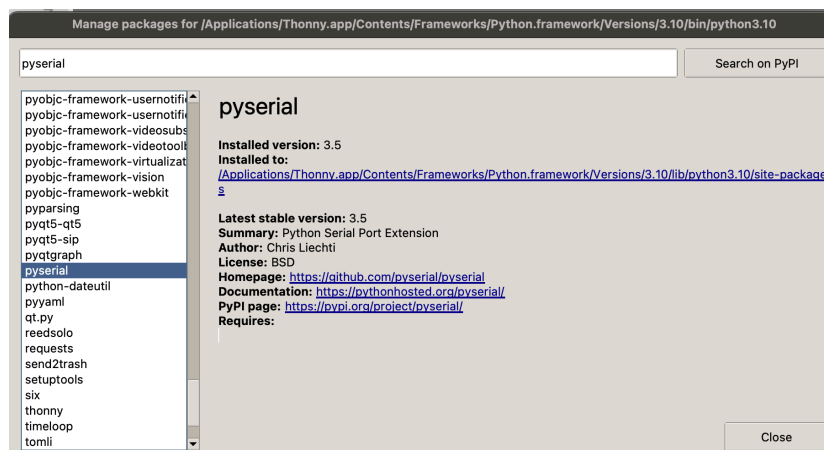
### 3.4.4 Opdracht 4: Tonen op computer

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.



We beginnen eerst met het kijken of we de seriële communicatie in orde kunnen maken.

Op de computer installeren we eerst pyserial. Dit kan door **pip install pyserial**, of via de install manager



We moeten eerst weten op welke seriële poort de Raspberry pico aangesloten zit.

```
python_serieel.py  serial_port_scanner.py
1 import serial.tools.list_ports
2 for port in serial.tools.list_ports.comports():
3     print(port)
4

Shell
>>> %Run serial_port_scanner.py
/dev/cu.URT1 - n/a
/dev/cu.URT2 - n/a
/dev/cu.BTMM - n/a
/dev/cu.Bluetooth-Incoming-Port - n/a
```

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

```

1 import serial
2
3 # Configure the serial connection
4 port = "/dev/cu.URT1"
5 baudrate = 115200
6 serial_connection = serial.Serial(port, baudrate)
7
8 # Read and write data until the transfer is complete
9 while True:
10     data = serial_connection.read(128)
11     if data == b"EOF":
12         break
13     print(data)
14

```

Op dit moment wordt er natuurlijk niets ontvangen, want we hebben nog niets verstuurd. We laten de Pico dit bericht versturen. 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.

```

1 #save on pico with main.py
2 #only works when NOT connected to IDE
3 import machine
4 from machine import I2C, Pin
5 import time
6 import uos
7
8 # need this UART to read from BME and be able to send data to local computer
9 uart = machine.UART(0, baudrate=115200)
10 uart.init(115200, bits=8, parity=None, stop=1, tx=Pin(0), rx=Pin(1))
11 uos.dupterm(uart)
12
13 while True:
14     print('{:.1f} C,{:.1f} %, {:.1f} hPa'.format(42,69,1234))
15     time.sleep(1)
16

```

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

```

Shell x
b'0.0 hPa\r\n0.4 C,1.2 %,0.0 hPa\r\n0.4 C,1.2 %,0.0 hPa\r\n0.4 C,
1.2 %,0.0 hPa\r\n0.4 C,1.2 %,0.0 hPa\r\n0.4 C,1.2 %,0.0 hPa\r\n0.
4 C,1.2 %,0.
b'0 hPa\r\n0.4 C,1.2 %,0.0 hPa\r\n0.4 C,1.2 %,0.0 hPa\r\n0.4 C,1.
2 %,0.0 hPa\r\n0.4 C,1.2 %,0.0 hPa\r\n0.4 C,1.2 %,0.0 hPa\r\n0.4
C,1.2 %,0.0
b'hPa\r\n0.4 C,1.2 %,0.0 hPa\r\n0.4 C,1.2 %,0.0 hPa\r\n0.4 C,1.2
%,0.0 hPa\r\n0.4 C,1.2 %,0.0 hPa\r\n0.4 C,1.2 %,0.0 hPa\r\n0.4 C,
1.2 %,0.0 hP'

```

We hebben nu een opstelling gemaakt om dat de communicatie werkt op deze manier. Nu moeten we de code ombouwen dat we de gewenste situatie krijgen.