

Bibliotheken für die Roboter

Micropython Software-Bibliotheken für den Betrieb der Roboter.

- Motorsteuerung
- Ultraschallsensor
- Infrarotsteuerung
- Servosteuerung
- Das komplette Modul "robotlibrary"

Motorsteuerung

Speichere diesen Code auf dem Pico unter dem Namen "motor.py".

Motorbibliothek

```
from machine import Pin, PWM
import utime

MIN_DUTY = 0
MAX_DUTY = 60000
MAX_SPEED = 100
MIN_SPEED = 30

class Motor:
    """This class manages the motor. Don't edit!"""
    def __init__(self, pinNo):
        self.gpio = pinNo
        self.speed=0
        self.forward=True
        self.pwm1=PWM(Pin(pinNo))
        self.pwm1.freq(50000)
        self.pwm1.duty_u16(0)
        self.pwm2=PWM(Pin(pinNo+1))
        self.pwm2.freq(50000)
        self.pwm2.duty_u16(0)
        self.speed_offset = 0

    def set_speed(self,s):
        """Sets the speed of the motor. Checks for sensible input."""
        if s + self.speed_offset <= MIN_SPEED:
            s = 0
            self.reset_offset()
        elif s + self.speed_offset >= MAX_SPEED:
            s = MAX_SPEED
        self.pwm1.duty_u16(int(MAX_DUTY*(s+self.speed_offset)/100))
        self.speed=s
```

```

def change_speed(self,sc):
    """This defines an offset to the speed in motor. It is used with the remote control to turn the robot."""
    if self.speed + sc > MIN_SPEED and self.speed + sc < MAX_SPEED:
        self.speed_offset += sc
        self.set_speed(self.speed)

def reset_offset(self):
    self.speed_offset = 0

def off(self):
    self.pwm1.duty_u16(0)
    self.speed = 0

def set_forward(self,forward):
    """Sets the motor to forward or backward without changing the speed. """
    if self.forward==forward:
        return
    self.pwm1.duty_u16(0)
    self.pwm1,self.pwm2=self.pwm2,self.pwm1
    self.forward=forward
    self.set_speed(self.speed)
    #self.pwm1.duty_u16(int(MAX_DUTY*(self.speed+self.speed_offset)/100)) # uncommenting this causes
problems with the remote control. After changing
    # the direction the robot would drive even if the remote control speed said 0.

```

Beispiel für die Anwendung dieser Bibliothek

Kopiere diesen Code in eine andere Datei auf dem Pico, z. B. „motortest.py“.

```

from motor import Motor
from utime import sleep, sleep_ms
motor = Motor(12)

```

```
motor.set_speed(70)
motor.set_forward(True)
sleep(1)
motor.off()
```

Ultraschallsensor

Ultraschallbibliothek

Speichere diesen Code auf dem Pico unter dem Namen "ultrasonic.py".

```
from machine import Pin
from time import sleep
import utime

class Ultra:
    """This class manages the ultrasonic sensor. It returns the distance to an obstacle in cm. """
    def __init__(self, pinNo):
        self.trigger = Pin(pinNo, Pin.OUT) # to trigger a sound impulse
        self.echo = Pin(pinNo+1, Pin.IN) # records the echo of the trigger pulse

    def get_dist(self):
        """This returns the measured distance in cm. (float)"""
        timepassed = 0
        signalon = 0
        signaloff = 0
        self.trigger.low()
        utime.sleep_us(2)
        self.trigger.high()
        utime.sleep_us(5)
        self.trigger.low()
        while self.echo.value() == 0:
            signaloff = utime.ticks_us()
        while self.echo.value() == 1:
            signalon = utime.ticks_us()
        timepassed = signalon - signaloff
        distance = round((timepassed * 0.0343) / 2, 2)
        # print("The distance from object is ", distance, "cm.") # for debugging purposes uncomment the line.
        utime.sleep_ms(10) # Wait necessary or program halts
        return distance
```

Beispiel für die Anwendung dieser Bibliothek

Kopiere diesen Code in eine andere Datei auf dem Pico, z. B. „ultratest.py“.

```
from ultrasonic import Ultra
from utime import sleep, sleep_ms
us = Ultra(16)

while True:
    print(f"gemessene Entfernung: {us.get_dist()} cm.")
    sleep(1)
```

Zum Fahren siehe [Motorsteuerung](#).

Infrarotsteuerung

Ein Infrarotsensor kann zum Erkennen von Hindernissen oder der Verfolgung einer schwarzen Linie genutzt werden. Die folgende Klasse steuert den Sensor.

```
from machine import Pin,Timer

import micropython

micropython.alloc_emergency_exception_buf(100)

class IR:
    """This class manages the IR-sensor. Write your code in Robot.ir_detected()"""
    def __init__(self, pinNo,callback_func):
        self.out = pinNo
        self.ir_detected = callback_func
        self.ir = Pin(pinNo, Pin.IN, Pin.PULL_UP)
        self.ir.irq(trigger=Pin.IRQ_FALLING | Pin.IRQ_RISING, handler=self.obstacle)
        self.detected=False
        self.timer = Timer()

    def reset_detected(self,t):
        self.detected = False

    def obstacle(self, pin):
        """This is called on any change in the IR-sensor. """
        if not self.detected:
            self.ir_detected(pin,self.out)
            self.detected = True
            self.timer.init(mode=Timer.ONE_SHOT, period=100, callback=self.reset_detected)
```

Der Code muss unter dem Dateinamen „infrared.py“ auf dem Pico gespeichert werden.

Der Infrarotsensor wird folgendermaßen im eigenen Programm eingebunden. Das Beispiel ist für zwei Infrarotsensoren.

```
from infrared import IR
IR_PIN_LEFT=0
IR_PIN_RIGHT=1

def ir_detected(pin, pinno):
    print(f"Pin: {pin}, pin number: {pinno}")
    if pinno == IR_PIN_LEFT:
        print("links")
    elif pinno == IR_PIN_RIGHT:
        print("right")

ir_left = IR(0, ir_detected)
ir_right = IR(1, ir_detected)
```


Servosteuerung

Der Ultraschallsensor kann auch mit einem Servomotor drehbar gemacht werden. Die folgende Klasse steuert den Servomotor:

```
from machine import Pin, PWM
import utime

class Servo:
    """This class manages the servo motor that turns the ultrasonic sensor. You need a servo motor installed to get use out of this.
    Don't use directly or edit."""
    def __init__(self, pin):
        self.pin = PWM(Pin(pin))
        self.pin.freq(50)
        self.min = 1350
        self.max = 8100
        self.angle = 0

    def set_angle(self, a):
        """If installed, the servor motor will set the angle of the ultrasonic sensor. 90° ist straight ahead."""
        if a > self.angle:
            for i in range(self._get_duty(self.angle), self._get_duty(a)):
                self.pin.duty_u16(i)

        elif a < self.angle:
            for i in range(self._get_duty(self.angle), self._get_duty(a), -1):
                self.pin.duty_u16(i)
        self.angle = a
        utime.sleep_ms(4)

    def _get_duty(self, angle):
        """Internal function. Calculates the PWM duty for the given angle."""
        return round((self.max - self.min) / 180 * angle + self.min)
```

Dieser Code muss unter dem Dateinamen „servo.py“ auf dem Pico gespeichert werden.

Das komplette Modul "robotlibrary"

Dieses Modul, das von [Github](#) heruntergeladen werden kann, steuert die Roboter mit allen Peripheriegeräten (Motoren, Sensoren). Dazu muss das Paket heruntergeladen und entpackt werden. Das Verzeichnis „robotlibrary“ muss dann auf den Pico hochgeladen werden.

SMARS Roboter "Theo III"

Um das Modul zu benutzen, muss nur folgender Import gemacht werden: `from robotlibrary.robot import Robot`.

Ein kurzes Codebeispiel, wie der Roboter funktioniert, ist in der Quelldatei zu finden.

Oder hier ein Beispiel für die Benutzung des Servomotors.

```
from robotlibrary.robot import Robot
from time import sleep, sleep_ms
try:
    r = Robot(False)
    r.set_angle(0)
    sleep_ms(500)
    r.set_angle(180)
    sleep_ms(500)
    r.set_angle(90)
    r.set_speed(80)
    while True:
        while r.get_dist() > 15:
            pass
        r.emergency_stop()
        sleep_ms(400)
        r.set_speed_instantly(80)
        r.spin_before_obstacle(20)
        r.set_forward(True)
```

```
r.set_speed(80)
except:
    r.emergency_stop()
    print("Robot stopped")
```

Under construction:

Crawly

Crawly ist ein Roboter, der, ähnlich wie eine Schildkröte auf vier Beinen kriechen kann. Um Crawly zu steuern, muss nur folgender Import gemacht werden: `from robotlibrary.crawly import Crawly`

Ein kurzes Codebeispiel, wie der Roboter funktioniert, ist in der Quelldatei zu finden.

Walky

Walky ist ein Roboter, der, ähnlich wie ein Hund, auf vier Beinen laufen kann. Um Walky zu steuern, muss nur folgender Import gemacht werden: `from robotlibrary.walky import Walky`

Ein kurzes Codebeispiel, wie der Roboter funktioniert, ist in der Quelldatei zu finden.

Die Dokumentation für die Bibliothek ist unter `docs` zu finden oder kann hier heruntergeladen werden: [robotlibrary.pdf](#)