Codebeispiele

Theo III

Best Practise

Ausschalten der Motoren bei Unterbrechung des Programms

Anfangs wird man sehr viel an dem Roboterprogramm testen müssen. Dabei wird das Programm dann häufig abstürzen. Damit die Motoren nicht weiter in dem Zustand laufen, in dem sie dabei geschaltet waren, kann man mit einem try/excepts arbeiten:

```
try:
  # Hier läuft das Programm
except Exception as err:
    print(err) # Nötig, um Fehlermeldungen angezeigt zu bekommen.
    r.emergency_stop() # Roboter anhalten, hier ein Beispiel mit der robotlibrary.
    print("Robot stopped") # Damit es ganz deutlich ist.
except KeyboardInterrupt:
    r.emergency_stop()
    print("Keyboard interrupt")
```

Effizienter Code

Auch wenn die Rechenleistung der Picos ausreichen sollte, ergibt es Sinn, sich über Effizienz Gedanken zu machen, da schwer zu erkennen ist, ob manche Probleme durch Überlastung des Prozessors hervorgerufen werden.

```
while True:
    robot.drive()
    if us.get_dist() > min_distance:
        # stop or turn
        robot.stop()
```

In diesem Beispiel wird in der Schleife der Befehl `drive()` mit jedem Durchlauf aufgerufen, was nicht sonderlich effizient ist, da die Motoren weiterfahren, auch wenn das Programm gerade andere Befehle ausführt. Eine bessere Variante wäre diese:

```
robot.drive()
while us.get_dist() > min_distance:
    pass
robot.turn()
```

Hier wird nur die Entfernung zum nächsten Hindernis überprüft. Sobald der Roboter zu nahe gekommen ist, wird die Schleife beendet und der Code wird weiter ausgeführt.

Fehlertoleranter Code

Die Sensoren, die wir benutzen, liefern nicht immer zuverlässige und korrekte Ergebnisse. Daher kann man sich nicht darauf verlassen, dass eine Messung ausreicht. Ist man auf genauere Ergebnisse angewiesen, kann es sinnvoll sein, die Ergebnisse von Sensormessungen (insbesondere des Ultraschallsensors) zu filtern. Dazu kann gehören, Extremwerte, die im vorliegenden Fall unwahrscheinlich sind, zu ignorieren oder Mittelwerte von mehreren Messungen zu bilden.

Beschleunigung mit Entfernungsmessung

```
obstacle_detected = False
new_speed = 100
speed_now = 0
min_distance = 15
while speed_now <= new_speed and not obstacle_detected:
    #Set the speed for the motors, f. ex. motor.set_speed(speed_now)
    utime.sleep_ms(10+int(speed_now/2))
    speed_now += 1
    if us.get_dist() < min_distance: # Adjust the code to your needs.
        obstacle_detected = True
if obstacle_detected:
    # Stop or turn or whatever
    obstacle_detected = False
else:
    # keep going
    pass
```

Revision #7
Created 17 December 2024 11:25:11 by Marcus Jacobs
Updated 11 February 2025 12:21:31 by Marcus Jacobs