UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TECHNISCHE INFORMATIK

Übungsblatt 4

Einführung in ROS 2

Vorlesung *Mobile Roboter*, Sommersemester 2022

Dozent: Prof. Dr.-Ing. Heiko Hamann
Ein Diskussionsforum zur Übung finden Sie unter: `https://moodle.uni-luebeck.de`.

Das Robot Operating System (ROS)[1] ist ein Set von Softwarebibliotheken und Tools, die die Entwicklung von Applikationen für Roboter vereinfacht. Dies ermöglicht es u.a. durch Hardwareabstraktion, Gerätetreiber, Interprozesskommunikation und Paketmanagement. ROS wird nicht nur in der Forschung und Entwicklung eingesetzt, sondern findet auch in der Industrie Einsatz. ROS 2 ist die Weiterentwicklung von ROS und integriert bisher fehlende Anforderungen, wie beispielsweise Echtzeitfähigkeit und Sicherheit.

In dieser Übung erlernen Sie die Grundlagen von ROS 2 und implementieren einen ersten eigenen ROS 2-Node. Zum Einsatz kommt dabei der mobile Roboter TurtleBot3 Waffle Pi, siehe Abbildung 1.

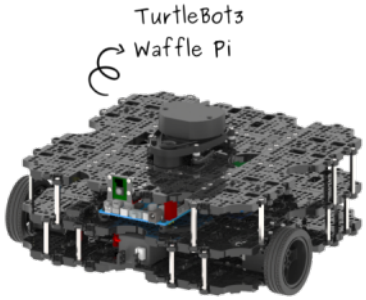

Abbildung 1: TurtleBot3 Waffle Pi [Quelle: `http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/`]

Der TurtleBot3 hat zwei Differentialmotoren, eine Raspberry Pi-Kamera, eine IMU, einen Laser-Entfernungssensor (LIDAR LDS-01), sowie einen Raspberry Pi (RPi) als Bordcomputer. Die Programmierung erfolgt unter Verwendung von Python 3 und ROS 2 Foxy Fitzroy.[2]

In dieser Übung verwenden Sie die Sensorwerte des LIDARs. Der LIDAR deckt 360° ab und liefert in 1°-Schritten Messwerte im Bereich von 0,12 bis 3,5 m. Die Messwerte des LIDARs werden auf dem ROS 2-Topic *scan* publiziert. Die Daten werden in Form einer *LaserScan*-Nachricht[3] zur Verfügung gestellt. Der Beispielnachricht in Abbildung 2 können Sie die oben gegebenen Daten des Sensors (vgl. *angle_min*, *angle_max*, *range_min*, *range_max*) sowie die gemessenen Entfernungen (*ranges*) entnehmen.

Der TurtleBot3 Waffle Pi verfügt über einen Differentialantrieb, den Sie über das Publizieren einer *Twist*-Nachricht[4] auf dem ROS 2-Topic *cmd_vel* steuern können. In der *Twist*-Nachricht müssen Sie die gewünschte Translation (`linear.x`) und Rotation (Bogenmaß, `angular.z`) des Roboters spezifizieren.

[1] `https://docs.ros.org/en/foxy/index.html#ros-2-documentation`
[2] `http://docs.ros.org/en/foxy/Releases/Release-Foxy-Fitzroy.html`
[3] `http://docs.ros.org/en/api/sensor_msgs/html/msg/LaserScan.html`
[4] `http://docs.ros.org/api/geometry_msgs/html/msg/Twist.html`

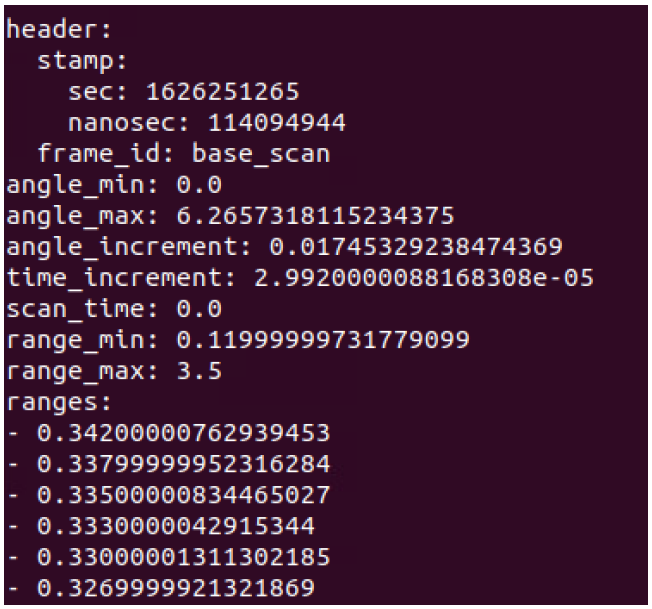

Abbildung 2: Beispiel einer *LaserScan*-Nachricht auf dem Topic *scan*

Aufgabe 1 Vorbereitung zu Hause: ROS 2 Tutorial (Individuell)

Für die Programmierung des TurtleBot3 benötigen Sie **zwingend** ROS 2 Foxy Fitzroy. ROS 2 Foxy Fitzroy läuft prinzipiell unter Ubuntu 20.04, Windows 10, and macOS 10.14. Allerdings sind die Anleitungen für den TurtleBot 3 auf Ubuntu ausgelegt, weshalb wir die Nutzung dieses Betriebssystems empfehlen. Eine Anleitung zur Installation auf dem eigenen Rechner oder in einer virtuellen Maschine ist hier verfügbar: `https://emanual.robotis.com/docs/en/platform/turtlebot3/quick-start/` (oben Foxy auswählen!).

Bearbeiten Sie individuell (**nicht** im Team!) folgende Tutorials im Abschnitt “Beginner” auf der Webseite `https://docs.ros.org/en/foxy/Tutorials.html`:

- Configuring your ROS 2 environment (**nur lesen!**)
- Introducing turtlesim and rqt
- Understanding ROS 2 nodes
- Understanding ROS 2 topics
- Understanding ROS 2 services
- Understanding ROS 2 parameters
- Creating your first ROS 2 package
- Writing a simple publisher and subscriber (Python)

Machen Sie nach dem Tutorial “Writing a simple publisher and subscriber (Python)” einen Screenshot Ihres kompletten Bildschirms. Laden Sie diesen bis **Mittwoch, 15.06.2022** in Moodle hoch. Die Abgabe **muss** einzeln erfolgen, d.h. jede:r Studierende muss einen individuellen Screenshot abgeben!

Hinweis: ROS 2 nutzt Quality of Service (QoS)[5] Einstellungen für die Kommunikation zwischen ROS 2 Nodes. Damit kann beispielsweise festgelegt werden, dass bestimmte Nachrichten nur in einem bestimmten Zeitraum nach dem Versand empfangen werden können. Für Sensordaten wird das QoS Profil `qos_profile_sensor_data` verwendet. Einen Subscriber für das scan-Topic des TurtleBots können Sie folgendermaßen anlegen:

```
subscriber = self.create_subscription(LaserScan,'scan',self.callback, qos_profile=qos_profile_sensor_data)
```

Ohne die Angabe des korrekten QoS Profils ist nicht garantiert, dass die Sensornachrichten empfangen werden.

[5] `https://docs.ros.org/en/foxy/Concepts/About-Quality-of-Service-Settings.html`

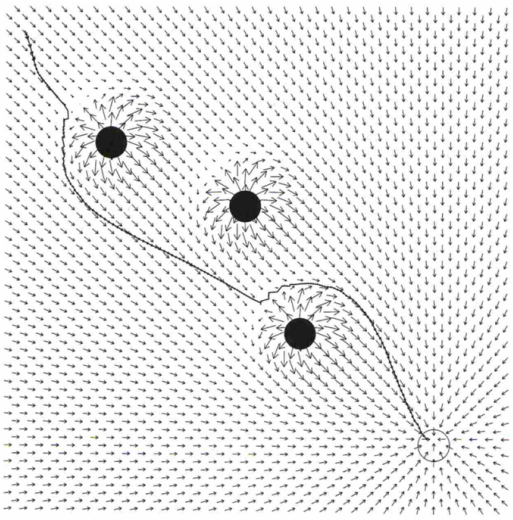

Abbildung 3: Beispiel eines Potentialfelds mit Hindernissen (schwarze Kreise) und Ziel.

Aufgabe 2 Vorbereitung zu Hause (Team)

Implementieren Sie ein Hindernisvermeidungsverhalten auf Basis eines Potentialfelds (vgl. Folien 2-68 bis 2-78). Dabei sollen ein abstoßendes und ein anziehendes Feld so kombiniert werden, dass der Roboter durchgängig fährt, vgl. auch Abbildung 3.

a) Laden Sie das Template *potentialfield.zip* aus Moodle herunter. Extrahieren Sie das ROS 2-Package in den Ordner */colcon_ws/src*. Implementieren Sie Ihre Lösung in der Datei *potentialfield/potentialfield_member_function.py* des ROS 2-Packages.

b) Implementieren Sie ein abstoßendes Feld basierend auf den Messungen des LIDAR. Dabei sollen Hindernisse nur bis zu einem Abstand R eine abstoßende Kraft erzeugen.
Tipp: Mit Hilfe von Sinus und Cosinus können Sie eine Distanz/Magnitude und einen Winkel in einen Richtungsvektor transformieren. Zur Berechnung können Sie z.B. das Paket NumPy in Python nutzen.

c) Implementieren Sie zusätzlich ein virtuelles anziehendes (attraktives) Feld, das eine konstante Vorwärtsbewegung abbildet.

d) Kombinieren Sie beide Potentialfelder und berechnen Sie aus dem resultierenden Vektor die geforderte lineare Geschwindigkeit und die Rotationsgeschwindigkeit des Roboters. Stellen Sie sicher, dass die Werte die Maximalwerte nicht überschreiten (max. lineare Geschwindigkeit: 0,26 $\frac{m}{s}$, max. Rotationsgeschwindigkeit: 1,82 $\frac{\text{rad}}{s}$).

Aufgabe 3 Test in Simulation

Testen Sie Ihren Code im Simulator *Gazebo*, indem Sie die folgenden Schritte ausführen.

a) Laden Sie die Simulation *Gazebo Welt: ITI Gang* aus Moodle herunter. Extrahieren Sie die Zip-Datei in den Ordner */colcon_ws/src*. Sollten Sie nicht die zur Verfügung gestellte virtuelle Maschine verwenden, müssen Sie noch folgende Pakete installieren:

```
sudo apt install ros-foxy-dynamixel-sdk ros-foxy-turtlebot3-msgs ros-foxy-turtlebot3
cd ~/colcon_ws/src
git clone -b foxy-devel https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
```

b) Bearbeiten Sie die Datei *.bashrc* und setzen Sie die ROS Domain ID:

```
nano ~/.bashrc
export ROS_DOMAIN_ID=X
```

Setzen Sie die Domain ID X dabei basierend auf Ihrer Team Nummer. Team 2A nutzt bitte Domain ID 21, Team 2B Domain ID 22, ... , Team 3A Domain ID 31, Team 3B Domain ID 32, usw.

Speichern Sie die Änderungen und machen Sie diese Ihrem System durch Ausführen des folgenden Befehls bekannt.

```
source ~/.bashrc
```

c) Navigieren Sie in Ihren colcon-Workspace und rufen Sie `colcon build` auf. Sourcen Sie die *setup*-Datei.

```
cd ~/colcon_ws
colcon build --symlink-install
. install/setup.bash
```

d) Starten Sie die *Gazebo* Simulation, um den TurtleBot3 zu simulieren:

```
cd ~/colcon_ws/src/ITI Gang Simulation
./start_simulation.sh
```

e) Führen Sie Ihren Code in einem neuem Terminal aus:

```
ros2 run potentialfield potentialfield
```

f) Verbessern Sie Ihre Lösung solange bis das geforderte Verhalten ausgeführt wird.

Aufgabe 4 Durchführung in der Übung

Lassen Sie Ihren Code auf dem echten TurtleBot3 laufen, indem Sie die folgenden Schritte ausführen. Mit **[VM]** markierte Schritte sind in Ihrer Virtuellen Maschine im TGI Labor durchzuführen, mit **[TB]** markierte Schritte sollen auf dem TurtleBot3 vorgenommen werden.

a) **[VM]** Bearbeiten Sie die Datei *.bashrc* der virtuellen Maschine.

```
nano ~/.bashrc
```

Ändern Sie die ROS Domain ID zur Nummer X Ihres TurtleBots ab. *X* ist die sich auf Ihrem Roboter befindliche Nummer.

```
export ROS_DOMAIN_ID=X
```

Speichern Sie die Änderungen und machen Sie diese Ihrem System durch Ausführen des folgenden Befehls bekannt.

```
source ~/.bashrc
```

b) **[VM]** Verbinden Sie sich mit dem TurtleBot3 via ssh.

```
ssh ubuntu@turtleX.local
```

X ist die sich auf Ihrem Roboter befindliche Nummer. (Passwort des TurtleBots: *turtlebot*)

c) **[VM]** Kopieren Sie Ihr ROS 2-Package aus Aufgabe 2 in den Ordner */colcon_ws/src*.

d) **[VM]** Navigieren Sie in Ihren colcon-Workspace und rufen Sie *colcon build* auf. Sourcen Sie die *setup*-Datei im Anschluss.

```
cd ~/colcon_ws
colcon build --symlink-install
. install/setup.bash
```

e) **[TB]** Starten Sie den ROS-Node des TurtleBots.

```
ros2 launch turtlebot3_bringup robot.launch.py
```

f) **[VM]** Führen Sie Ihren Code aus.

```
ros2 run potentialfield potentialfield
```

Testen Sie Ihren Code und verbessern Sie auftretende Fehler. Optimieren Sie das Fahrverhalten.