VRX – Gazebo – ROS Anleitung

**ROS Version:** ROS Noetic Ninjemys (ältere Version ist ROS Melodic Morenia)

# Ubuntu Befehle:

* sudo apt update (schaut ob es neue Updates gibt)
* sudo apt upgrade (installiert neue Updates)
* pwd (print working directory) (gibt den aktuellen Ordner zurück)
* mkdir (make directory) (erstellt einen neuen Ordner)
* ls (list) (Zeigt alle Unterordner im aktuellen Ordner an)
* > (schreibt das Ergebnis eines Befehls in eine Datei)
* echo „…“ (gibt den Text in den Gänsefüßen zurück)
* echo "This is a test" > test\_1.txt (Schreibt den Text in eine Datei)
* cat (Liest eine Datei aus)
* less (Liest ebenfalls eine Datei aus, allerdings in einem extra Fenster) (q um zu schließen)
* sudo (superuser do) (Vollen Zugriff auf alle Funktionen und Dateien des Computers)
* source (Führt ein Skript aus z.B. um Variablen (neu) zu setzen)

# ROS Befehle

## Einen Workspace erstellen:

* Ordner für den Workspace mit Unterordner src erstellen, dann catkin\_make im Ordner ausführen
* source devel/setup.bash (Neue Umgebungsvariablen aus Development Ordner laden)

## ROS (Robot Operating System!)

* roscore (Startet die Austauschplattform, die für Kommunikation zwischen Skripten genutzt werden kann)
* rosnode list (Zeigt die aktuell aktiven Knoten an)
* rosnode info <Knotenname> (Zeigt mehr Informationen zu einem Knoten an)
* rosrun <Packetname> <Knotenname> (Startet ein Programm und den Knoten)
* rostopic echo <topic> (Zeigt die Informationen an, die über ein Topic zwischen zwei Knoten ausgetauscht werden)
* rostopic type <topic> (Zeigt an in welchem Format das Topic ist)
* rostopic pub [topic] [msg\_type] [args] (Sendet eine Nachricht an ein Topic)(-1 für einmal, -r für kontinuierlich)
* rosrun rqt\_graph rqt\_graph (Stellt alle Topics und Knoten in einer Grafik dar)
* rosrun rqt\_plot rqt\_plot (Plottet Variablen aus beliebigen Topics)
* rosservice list (Gibt die aktuell verfügbaren Services zurück) (rosservice call um auszuführen)
* rosrun rqt\_console rqt\_console (Zeigt log des roscore an)
* rosrun rqt\_logger\_level rqt\_logger\_level (Einstellungen für die logger level (Info, Warnung ..))
* rosed [package\_name] [filename] (Zum Bearbeiten von Dateien)
* rosmsg show [message type] (Zeigt eine erstellte msg an)
* roscp [package\_name] [file\_to\_copy\_path] [copy\_path] (Kopiert eine ROS Datei in einem anderen Package an einen neuen Ort im neuen Package)

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* rospack = ros+pack(age) : provides information related to ROS packages
* roscd = ros+cd : **c**hanges **d**irectory to a ROS package or stack
* rosls = ros+ls : **l**ist**s** files in a ROS package
* roscp = ros+cp : **c**o**p**ies files from/to a ROS package
* rosmsg = ros+msg : provides information related to ROS message definitions
* rossrv = ros+srv : provides information related to ROS service definitions
* catkin\_make : makes (compiles) a ROS package
* rosmake = ros+make : makes (compiles) a ROS package (if you're not using a catkin workspace)
* catkin build: makes (compiles) a ROS package in an isolated manner while maintaining efficiency due to parallelisation
* rosbag record -O subset <Knoten> (die Daten aus einem bestimmten Knoten werden gespeichert
* rosbag record -a (die Daten aus allen Knoten werden gespeichert)
* rosbag play <Datei.bag> (Spielt das aufgezeichnete Szenario nochmal ab)
* roswtf (Kann jederzeit genutzt werden, versucht möglichen Fehler zu finden und zu beschreiben)

## Schildkröte

* rosrun turtlesim turtlesim\_node (Startet die Schildkröte)
* rosrun turtlesim turtle\_teleop\_key (Öffnet Steuerfeld für die Schildkröte)
* rostopic pub -1 /turtle1/cmd\_vel geometry\_msgs/Twist -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, 1.8]' (Lässt Schildkröte einmal einen drittel Kreis laufen)

## VRX

* roslaunch vrx\_gazebo vrx.launch (Startet die Umgebung in Gazebo)
* roslaunch vrx\_gazebo usv\_keydrive.launch (Öffnet Steuerfeld für das Boot)
* rostopic pub --once /wamv/thrusters/right\_thrust\_cmd std\_msgs/Float32 "data: 1.0"
* rosrun vrx\_gazebo kleine\_fahrt\_skript.py (Startet eigene Python Datei)
* roslaunch wamv\_gazebo rviz\_vrx.launch (Startet Programm zum Anzeigen der Sensordaten)
* rosservice call /gazebo/reset\_world (Setzt das Boot an die Anfangsposition zurück)