Technische Hochschule Georg Agricola

ROBOTIK

ROBOT OPERATING SYSTEM (ROS)

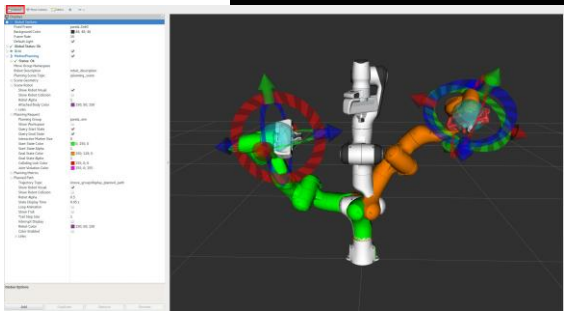
29.10.2025 Bochum

Dr.-Ing. Lars N. Josler

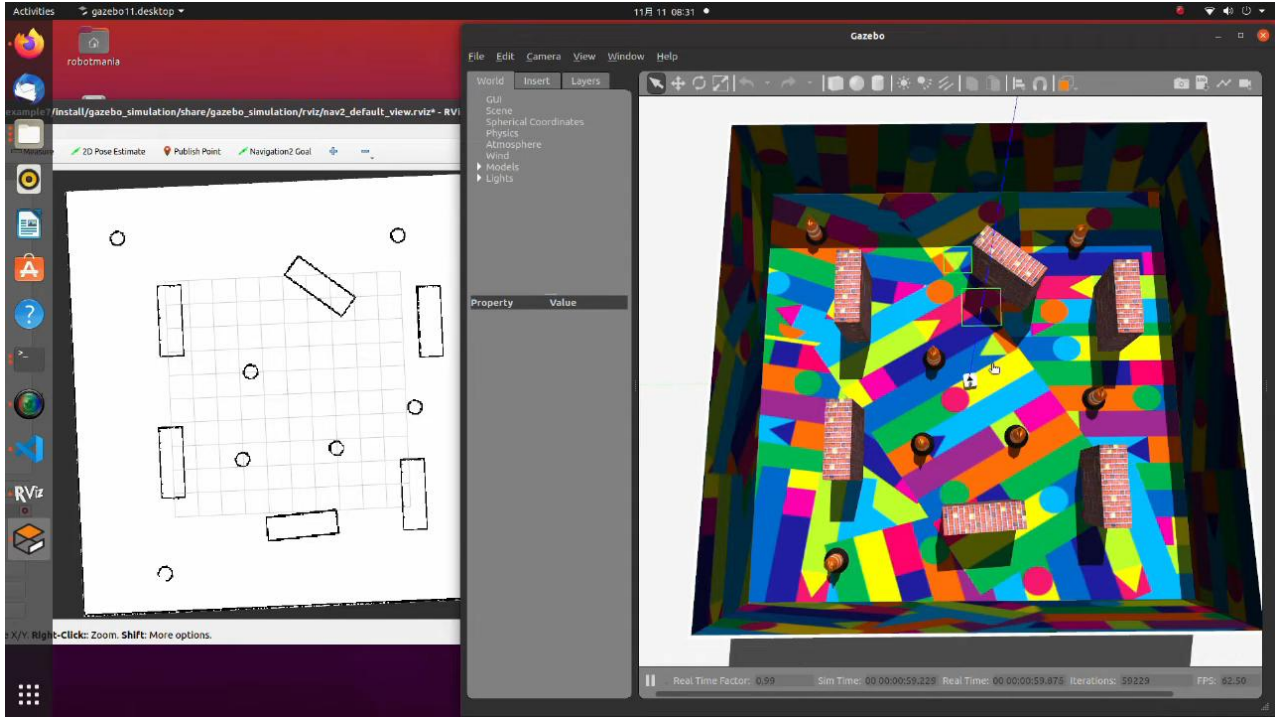
ROBOT OPERATING SYSTEM 2 (ROS2)

- Open-Source-Framework für die Entwicklung autonomer Roboteranwendungen.
- Klare Regeln und Strukturen für den Aufbau von Robotersoftware (Packages, Nodes)
- Kommunikationsinfrastruktur (Middleware) für den Austausch zwischen Nodes.
- modulares Framework
- ROS2 unterstützt mehrere Plattformen und enthält integrierte Sicherheitsfunktionen.
- ermöglicht Aufgaben wie Navigation, Manipulation und Wahrnehmung.

ROS 2™

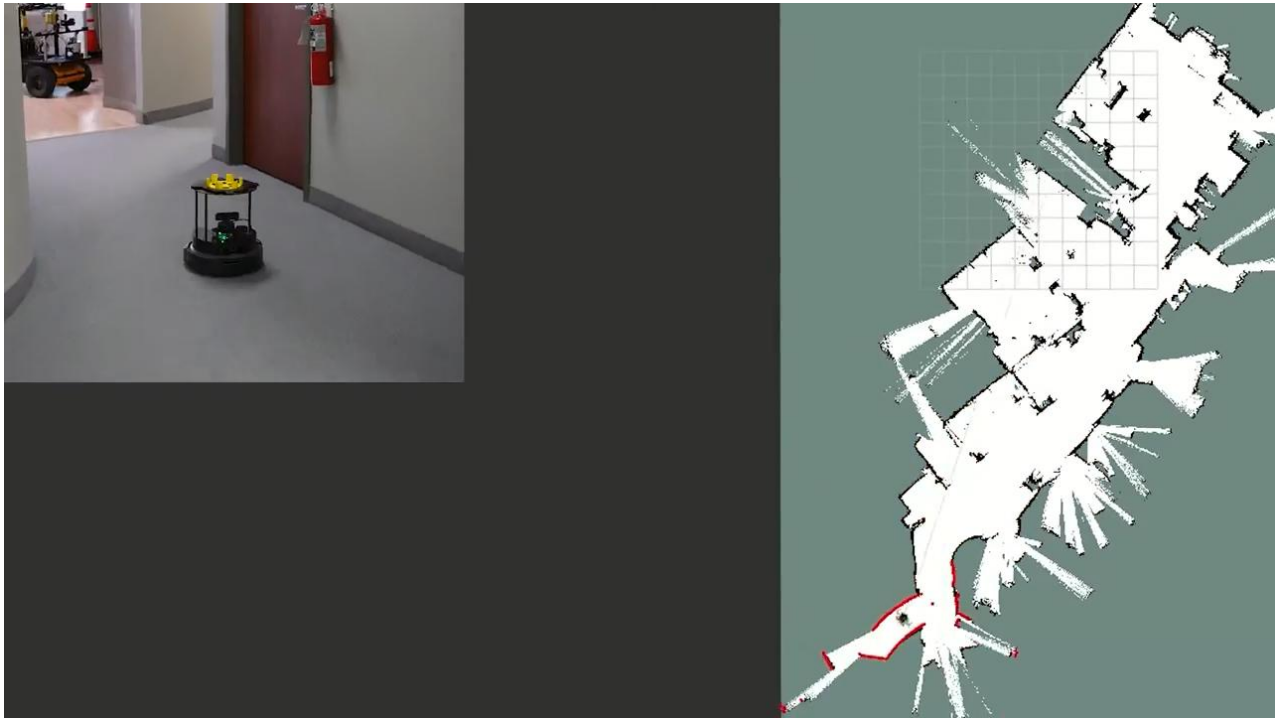


BEISPIEL: GAZEBO UND RGB-KAMERA



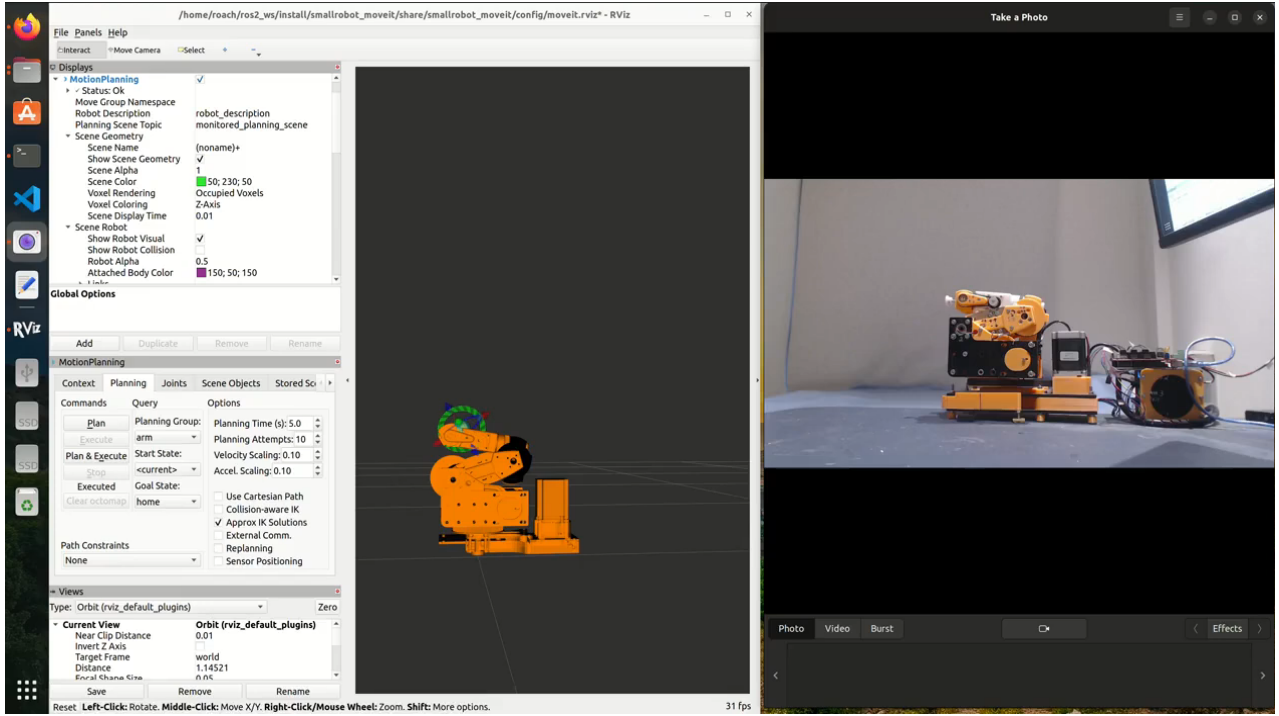
Robot navigation using only an RGBD camera - YouTube

BEISPIEL: RVIZ UND 2D-LASER



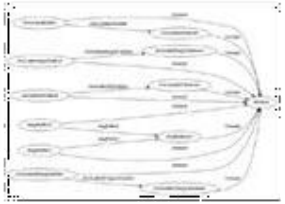
https://www.youtube.com/watch?v=T3if0aPj0Eo

BEISPIEL: BEWEGUNGSPLANUNG ROBOTERARM

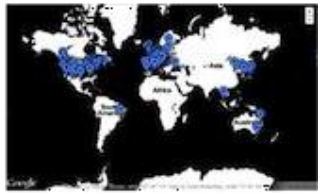


Quelle: ros2 control + moveIt for smallrobotarm - YouTube

ROS-EIGENSCHAFTEN


+

+

+


Plumbing

- Prozessmanagement
- Interprozessüber-greifende Kommunikation
- Gerätetreiber

Tools

- Grafische Benutzeroberfläche
- Simulation
- Visualisierung
- Messwerterfassung

Capabilities

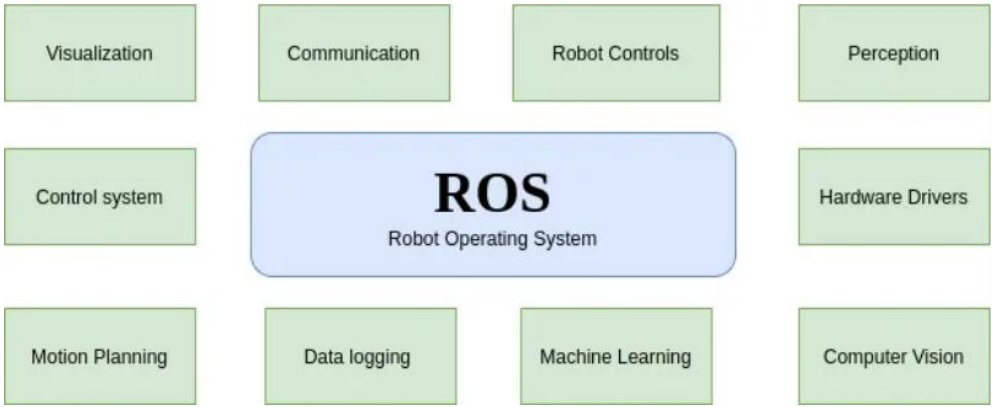
- Steuerung
- Planung
- Wahrnehmung
- Abbildung
- Manipulation

Ecosystem

- Organisation der Pakete
- Software-Verteilung
- Dokumentation
- Tutorials

ROS, EIN MIDDLEWARE

- **Open-Source-Middleware (kein Betriebssystem)**
- **Entwicklungsumgebung**
  - Visualisierungs- und Introspektionswerkzeuge
- **Kommunikationsbibliothek und –tools**
- **ROS-Packages**
  - colcon-Befehl
- **Aktiv Community**

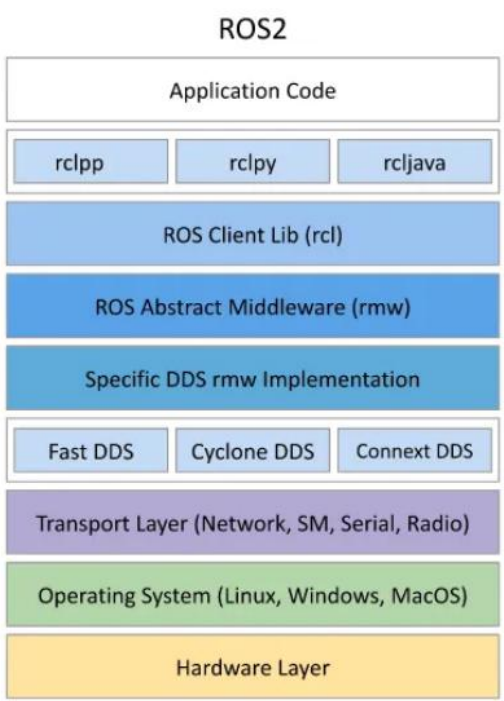


Quelle: Medium

ABSTRAKTE ROS-SCHICHTEN

- **DDS (Data Distribution Service)**
  - Standard, der den Datenaustausch nach einem Publish-Subscribe-Muster ermöglicht.
  - Middleware, die eine leistungsstarke, skalierbare und sichere Möglichkeit für Knoten bietet, Daten auszutauschen und miteinander zu kommunizieren.
  - Wird als Kommunikationsschicht für ROS2 verwendet.

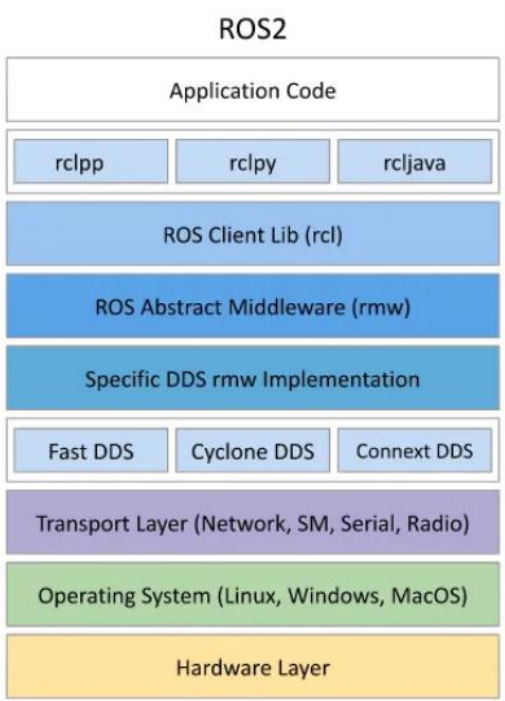
- **rmw (ROS-MiddleWare)**
  - Middleware, die die zugrunde liegende Kommunikationsinfrastruktur für ROS2 unter Verwendung von DDS bereitstellt
  - Abstrahiert die Komplexität von DDS
  - müssen sich → Entwickler keine Gedanken über die Details der Funktionsweise von DDS machen.



Quelle: Medium

ABSTRAKTE ROS-SCHICHTEN 2

- **rcl (ROS-Client-Bibliothek)**
  - Middleware, die eine High-Level-Schnittstelle zum Erstellen und Ausführen von Roboteranwendungen mit rmw bereitstellt.
  - Abstrahiert die Komplexität von rmw
  - müssen sich → Entwickler keine Gedanken über die Details der Funktionsweise von RMW machen.
- **RCLCPP**
  - C++-Implementierung von rcl
  - Bietet eine Reihe von C++-Klassen und -Funktionen zum Erstellen und Ausführen von Roboteranwendungen mithilfe von rcl
  - Bietet eine Reihe von Funktionen und Werkzeugen, die das Entwickeln und Debuggen von Roboteranwendungen erleichtern.
- **rclpy (RCLJAVA Manager)**
  - wie rclpp von in Python (resp. java)



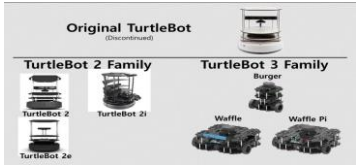
Quelle: Medium

WAS IST ROS NICHT

- **Kein (Computer-)Betriebssystem, sondern ein *modulares Framework***
  - Offiziell verfügbar auf Ubuntu Gnu/Linux
  - Experimentelle Unterstützung für: macOS, MS Windows, Fedora, Gentoo, Debian. . .
- **Keine Programmiersprache**
  - ROS-Programme, die in C++ und Python geschrieben wurden.
  - Experimentell: Java, Lisp, Octave. . .
- **Keine harte Echtzeitumgebung**
  - nur in Kombination mit einem Echtzeitbetriebssystem
- **Keine Entwicklungsumgebung**
  - Kann über IDE, Texteditor und Befehlszeile verwendet werden

GESCHICHTE VON ROS

- **Ursprünglich von der Stanford University (2006)**
  - persönliches Projekt von Keenan Wyrobek und Eric Berger



„Stop Re-Inventing the Wheel“

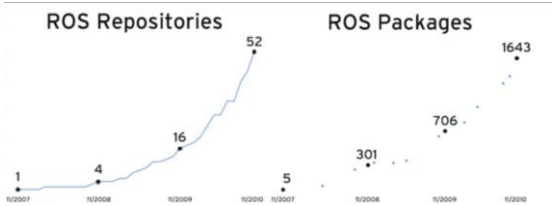
TurtleBot gestartet

ROS2

2006 — 2008 — 2011 — 2014 — 2017

- **Investition von Willow Garage**
  - Erste Version im Jahr 2009 (ROS Mango Tango)
  - ROS Groovy Galapagos, im Jahr 2012

- **Open Source Robotics Stiftung**
  - ROS Hydro Medusa, im Jahr 2013
    - ROS Melodic Morenia, im Jahr 2018



Quelle: Folie vom Eric and Keenan Pitch Deck

WARUM ROS ?

- **Open Source:**
  - Aktive Gemeinschaft, die sich ständig weiterentwickelt und verbessert
- **Wiederverwendbarkeit:**
  - Bietet verschiedene Open-Source-Tools und -Software, die das Beitragen, Anpassen und Teilen erleichtern
- **Unterstützung für Entwicklungswerkzeuge:**
  - Debugging-Tools, 2D- und 3D-Visualisierungstools (Rviz)
- **Offline Simulation:**
  - Vor dem Testen an physischen Robotern stellen ROS-Simulatoren (Gazebo) und eine einfache Möglichkeit zur Aufzeichnung und Wiedergabe von Sensordaten (Rosbags) zur Verfügung

NODES

- **ROS-Anwendungen basieren auf einer Komposition von unabhängigen, ausführbaren Programmen → Nodes (dt. Knoten).**
  - jeder Node ist ein separater Betriebssystemprozess
  - erfüllt in der Regel nur eine **schmale, dedizierte Aufgabe** (z.B. Kamerabild einlesen, Pfadplanung berechnen, Motorsteuerung)
  - sind vom Rest des Systems **vollständig entkoppelt**.
  - kommunizieren miteinander über ein Publish-Subscribe- (über **Topics**), Request-Response-Messaging-Muster (über **Services**) oder Goal–Feedback–Result (über **Actions**)

ROS 2-KOMMUNIKATIONSARTEN

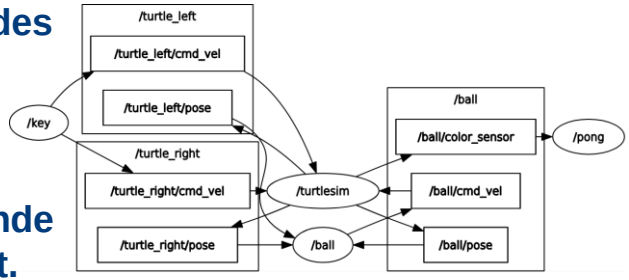
	Kommunikation	Zweck	Verhalten	Anwendung	Vergleichbar mit
Topics	Publish–Subscribe	Kontinuierlicher Datenfluss (Streaming)	Asynchron: Sender sendet ohne auf Bestätigung zu warten	Sensorik (Lidar, Kamera), Zustands-Updates, Geschwindigkeitsbefehle	Radiosender – sendet dauerhauft auf 98,5 FM (Publisher), jeder Empfänger (Subscriber) kann auf der Frequenz zuhören.
Services	Request–Response	Einmalige synchrone Anfrage	Synchron: Client blockiert und wartet auf die finale Antwort	Statusabfragen, Konfigurationsänderungen („Karte speichern“)	Online-Wetterdienst –eine konkrete Anfrage (Request), der Server antwortet direkt mit den aktuellen Daten (Response).
Actions	Goal–Feedback–Result	Länger andauernde, abbrechbare Aufgaben	Asynchron mit Kontrolle: Client kann Fortschritt verfolgen und abbrechen	Komplexe Navigation („Fahre zu Punkt X“), Greifen, Laden der Batterie	Paketverfolgung –Bestellung aufgeben zu Zielposition (Goal), regelmäßig Versandupdates (Feedback), Möglichkeit Vorgang abzubrechen. Paket wird zugestellt (Result).

COMPUTATIONAL GRAPH

Der Computational Graph ist Netzwerk von Knoten aller laufenden ROS-Prozesse und ihrer Kommunikationsbeziehungen.

Er besteht aus:

- **Nodes: Ausführende Prozesse, z. B. Sensor-Treiber, Steueralgorithmen oder Visualisierungsmodule.**
- **Topics: Asynchrone Kommunikationskanäle, über die Nodes Nachrichten (Messages) veröffentlichen und abonnieren.**
- **Services: Synchrone Anfrage-Antwort-Schnittstellen zwischen zwei Nodes.**
- **Actions: Erweiterung des Service-Konzepts für langlaufende Aufgaben mit Zwischenfeedback und Abbruchmöglichkeit.**
- **Bags: Datei-Format zur Aufzeichnung und Wiedergabe von Nachrichten auf Topics. Nützlich für Debugging, Simulation und Offline-Analyse.**



PUBLISHER AND SUBSCRIBER

- **Publisher und Subscriber sind Schlüsselkomponenten für die Kommunikation.**

Publisher

- Rolle:
  - Sendet Daten an ein bestimmtes Thema.
- Charaktereigenschaften:
  - Veröffentlicht Nachrichten eines bestimmten Typs.
  - Mehrere Publisher können in dasselbe Thema schreiben.

Subscriber

- Rolle:
  - Empfängt Daten zu einem bestimmten *topic*.
  - Auslösen einer Callback-Funktion
- Charaktereigenschaften:
  - Abonniert Nachrichten eines bestimmten Typs.
  - Mehrere Abonnenten können aus demselben Thema lesen.

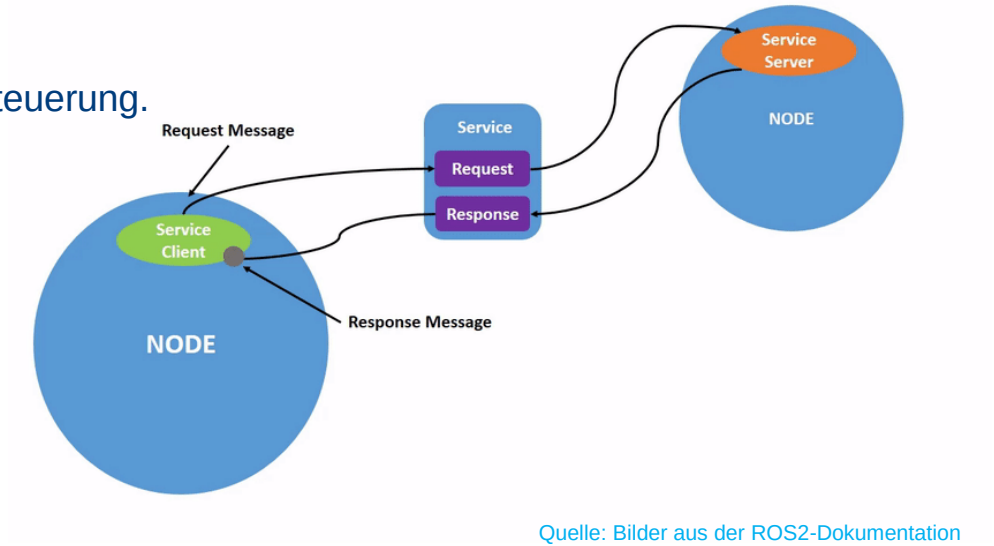
TIMER

Timer

- Rolle:
  - Erstellen Sie Timer, um Callback-Funktionen in bestimmten Intervallen zu planen.

SERVICE

- **Anwendungsfälle**
  - für synchrone Anforderung-Antwort-Interaktionen.
  - Wird häufig für Aufgaben verwendet, die Parameter festlegen, Daten anfordern oder Aktionen auslösen.
- **Nützt**
  - Synchrone Kommunikation für Echtzeitsteuerung.
  - Robuste Fehlerbehandlung durch Antwortmeldungen.



Quelle: Bilder aus der ROS2-Dokumentation

SERVICE - .SRV-SCHNITTSTELLE

- Definiert die Struktur von Anforderungen und Antworten.
- Ermöglicht Knoten eine nahtlose Kommunikation.
- Standard-SRV-Schnittstellen verfügbar in
  - https:github.com/ros2/common_interfaces/tree/rolling/std_srvs/srv

AddTwoInts.srv

```
int64 a
int64 b
---
int64 Summe
```

SERVICE – SERVER/CLIENT

Server:

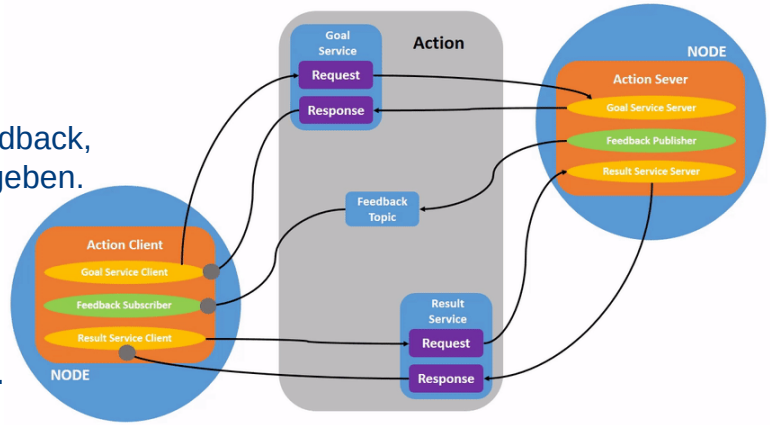
- **Rolle:**
  - Lauscht auf eingehende Anforderungen.
  - Bietet auf Anfrage eine Dienstleistung an.
- **Charaktereigenschaften:**
  - Bietet einen spezifischen Dienst mit einer klar definierten Schnittstelle.

Client:

- **Rolle:**
  - Sendet Anfragen an einen Service-Server.
  - Empfängt Antworten vom Server.
- **Charaktereigenschaften:**
  - Ruft einen bestimmten Dienst mit derselben klar definierten Schnittstelle wie der Server auf

ACTIONS

- **Anwendungsfälle**
  - Aktionen eignen sich für asynchrone Aufgaben mit langer Laufzeit.
    - Bsp.: Bahnplanung, Roboternavigation, . . .
- **Nützt**
  - Überwachung des Zielstatus und Feedback.
    - Ermöglichen es Clients, Ziele zu senden, und Servern, die ausgeführt werden sollen, und geben dieses stetiges Feedback, im Gegensatz zu Diensten, die eine einzelne Antwort zurückgeben.
  - Ähnlich wie bei Diensten (Client-Server), mit der Ausnahme, dass Aktionen präempbar sind (Sie können sie während der Ausführung abbrechen).
  - Robuste Fehlerbehandlung durch Ergebnismeldungen.

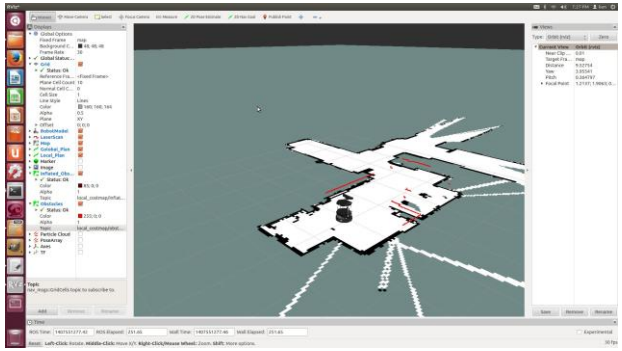


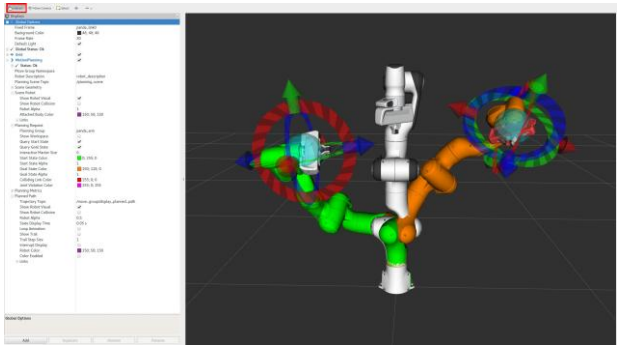
Quelle: Bilder aus der ROS2-Dokumentation

TOOLS VON DRITTANBIETERN - RVIZ

RViz (ROS-Visualisierung)

- 3D-Visualisierungssoftware-Tool für Roboter, Sensoren und
- Algorithmen Ermöglicht es, die Wahrnehmung des Roboters seiner Welt (real oder simuliert) zu sehen.



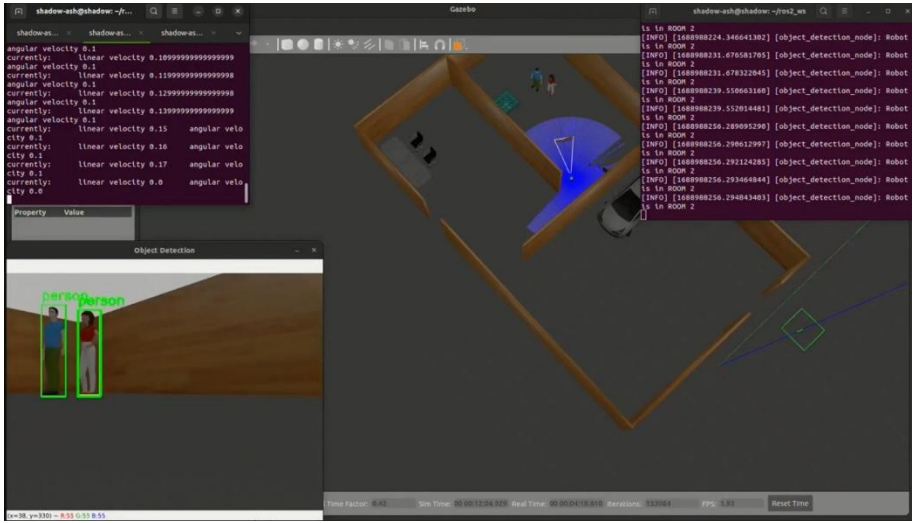


- Rviz ist KEIN Simulationswerkzeug

TOOLS VON DRITTANBIETERN - GAZEBO

- **Open-Source-Physiksimulation für Roboter mit realistischer Dynamik, Sensorik und 3D-Umgebung.**
  - sichere, reproduzierbare Testumgebung für reale Robotikalgorithmen
  - Austausch von Topics, Services, Actions
- **Typische Einsatzbereiche:**
  - Test von Navigation, Manipulation, SLAM
  - Sensor-Emulation (Kamera, LiDAR, IMU)
  - Integration mit RViz und MoveIt 2 möglich

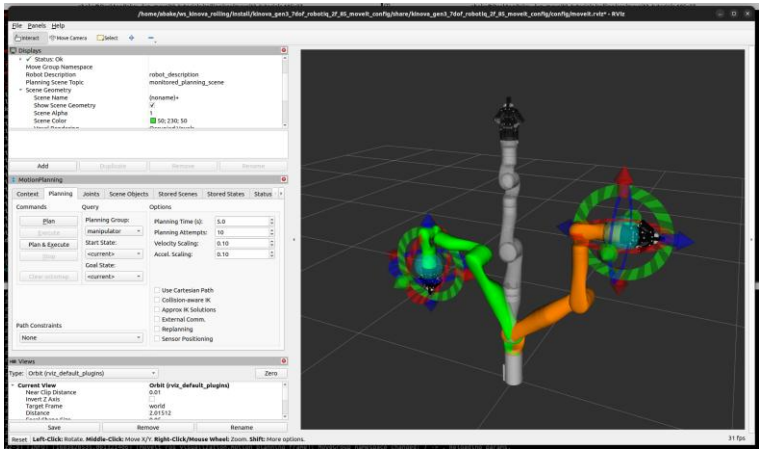
GAZEBO



TOOLS VON DRITTANBIETERN - MOVEIT 2

- **Planungs- und Steuerungsframework für Bewegungsplanung, Manipulation und Steuerung von Roboterarmen in ROS 2.**
  - Nutzt Kinematik, Kollisionsprüfung und Trajektorienerzeugung
  - Speichern und Laden von Planungs-Setups und Roboterkonfigurationen
  - Integration mit RViz zur interaktiven Planung und Visualisierung
  - Schnittstellen zu realen Robotern und Simulatoren (z. B. Gazebo)
  - Unterstützt inverse und direkte Kinematik
  - Modulare Architektur mit Plugins für Planer, Controller und Sensorik

MoveIt



Quelle: Tutorials — MoveIt Documentation: Rolling documentation

ROS DOKUMENTATIONEN

- **Online-Community:**
  - ROS2 lebt von einer aktiven, internationalen Community, die kontinuierlich neue Pakete entwickelt, Fehler behebt und Wissen teilt.
- **ROS2 ist frei verfügbar und unter einer Open-Source-Lizenz veröffentlicht.**
- **Wichtige Community-Plattform für technische Fragen und Antworten:**
  - Robotics Stack Exchange (https://robotics.stackexchange.com/)
- **Wichtige Plattform für Tutorials, How-to-guides u.v.m.:**
  - ROS Developer Documentation: https://docs.ros.org/