Kurzfassung

Die Entwicklung von mechatronischen Systemen ist eine anspruchsvoller Prozess. Denn die Teilsysteme stammen aus verschiedenen Disziplinen wie Mechanik, Elektronik und Informatik.

Dieser Umstand macht es schwierig die einzelnen Teilsystemen zu entwickeln und testen. Um die Entwicklung zu unterstützen und beschleunigen werden Werkzeuge eingesetzt. In dieser Arbeit sollen die Möglichkeiten vom *ROS*-Ökosystem als unterstützendes Entwickler-Werkzeug evaluiert werden. Im speziellen sollen Lösungen und Vorlagen für die Entwickler von *EEROS*-Applikationen erarbeitet werden.

Dafür sollen zwei Fallbeispiele umgesetzt werden. Die Umsetzung besteht zum eine aus eine Simulation und zum anderen aus einer Visualisierungs-Lösung. Mit der Visualisierung sollen Daten und Zustände gleichermassen vom realen System oder vom simulierten System dargestellt werden.

Das erste Fallbeispiel ist eine einfachen Motor-Baugruppe. Mit diesem Beispiel konnte erfolgreich aufgezeigt werden, wie die Entwicklung einer *EEROS*-Applikation unterstützt werden kann. Ebenfalls wurde eine Entwicklungsumgebung mit Simulation und Visualisierung für den *EEDURO-Delta* Roboter erstellt. Diese kann dann eingesetzt werden bei dem Upgrade der bestehenden *EEROS*-Applikation für den Delta-Roboter auf die neuste *EEROS*-Version. Auch kann mit der Entwicklungsumgebung vom *EEDURO-Delta* Roboter das Framework *EEROS* kennen gelernt werden ohne zuerst Hardware anzuschaffen.

Mit den beider in dieser Arbeit gezeigten Fallbeispielen erhalten *ROS*- und *EEROS*-Entwickler eine Vorlage für das Erstellen von Simulationen und Visualisierungen.

Inhaltsverzeichnis

1 Einleitung

1.1 Anforderungen/Aufgabenstellung

- ros konzepte kurz vorstellen wie:
  - msg
  - topic
  - node
- ros-ökosystem: gazebo, rviz, rqt

1.2 ROS

Im ROS-Ökosystem gibt es für die unterschiedlichen Aufgabe jeweils Komponenten bzw. Programme um diese

1.3 EEROS

kurze Einführung, Verweis

In dieser Arbeit wird EEROS als externe Software verwendet. Jedoch soll hier angemerkt werden dass jede Software verwendet werden kann, insofern die ROS-Library in diese integriert werden kann. Ein Beispiele wie diese Integration aussehen kann ist in der Arbeit

2 Konzept

Stichworte

- Übersicht von Konzept mit Graphik
- 3 Bereiche aufzeigen: EEROS, Gazebo, rqt & rviz
- Schnittstellen zu einander
- Verweis zu Mäsis Arbeit
- urdf, sdf vorstellen und erklären, hier??? oder in Motor

Eine Skizze des Konzeptes ist in der Abbildung zusehen. Die zwei Hauptkomponenten sind *EEROS* und *ROS*. *EEROS* übernimmt in diesem Szenario die Aufgabe des Regelns. Und *ROS* die Aufgaben der Simulation und der Visualisierung.

Damit *EEROS* mit *ROS*-Komponenten kommunizieren kann, wurde einen *ROS*-Node ins *EEROS* integriert. Somit hat *EEROS* die Fähigkeit über das *ROS*-Kommunikationsprinzip *Publish & Subscribe* Daten auszutauschen. Das Konzept sieht vor das *EEROS* wahlweise ein echtes oder simuliertes System regelt. Und in beiden Fällen können die Zustände und Regel-Parameter durch *ROS* visualisiert werden.

Für die Simulation von Systemen wird *Gazebo* eingesetzt. Die Darstellung von den System Zuständen wird mit *rviz* realisiert. Und für die Darstellung von Daten wie der Regel-Parameter in Graph-Plots wird *rqt* eingesetzt.

2.1 Gazebo

- festkörper simulation
- kann mit selbst geschriebenen Plugins erweitert werden.
- benötig sdf für beschreibung des zu simulierenden systems

Gazebo ist eine Simulationsumgebung für Starrkörper. Das zu simulierenden Systems wird mit einem *sdf*-File beschrieben. Das File ist im XML-Format aufgebaut.

Um *Gazebo* mit Funktionen zu erweitern können Plugins verwendet werden. Dabei kann man schon fertige Plugins verwenden oder selber eines programmieren.

2.2 Darstellung

2.2.1 rviz

- darstellen von System und deren Zuständen
- benötigt urdf-file für Darstellung

Das Visualisierungs-Tool *rviz* ist für die Darstellung von Systemen geeignet. Dabei können z.B. Zustände von einem Roboter visualisiert werden oder Sensor-Daten wie von einer 3D-Kamera.

Ein System besteht aus mehren Körpern. Um sie darzustellen braucht es unter anderem Informationen über das Aussehen und Form dieser. Diese Informationen müssen dem *rviz* über eine *urdf*-File zur Verfügung gestellt werden. Für die Darstellung der Körper im Raum benötigt *rviz* für jeden Körper noch die Position und Orientierung von diesem im Raum. Deshalb müssen dem *rviz* stetig Koordinaten-Daten für jeden Körper übermittelt werden.

2.2.2 rqt

rqt ist Framework für die GUI Entwicklung in *ROS*. Diese GUI's werden als Plugins implementiert. Somit können mehre GUI's in einem *rqt*-Fenster verwendet werden. Für fast jedes gängige *ROS*-Command-line-Tool gibt es schon Plugins. Es können aber auch selbst programmiert Plugins verwendet werden. Für die Darstellung von zeit veränderlichen Parametern wird das Plugin *multiplot* verwendet.

2.3 System-Beschreibung

Die Programme *Gazebo* und *rviz* verwenden für die Beschreibung der Systeme unterschiedliche Datei-Formate. In diesem Abschnitt werden diese beiden XML-Formate kurz vorgestellt. Sie können beide kinematische Strukturen repräsentieren. Die zwei Hauptkomponenten für die Beschreibung dieser Struktur sind das Glied (Link) und das Gelenk (Joint). Für jedes gib es ein entsprechendes XML-Element. Die Beziehung zwischen den beiden Komponenten ist in der Abbildung xx zusehen.

Link

Joint

2.3.1 URDF-Unified Robot Description Format

Das *URDF*-Format wird standardmässig in *ROS* verwendet, für die Beschreibung von Systemen. Es kann nur kinematische Strukturen abbilden die einen Baum-Form haben. Somit können keine geschlossenen kinematischen Ketten beschrieben werden.

2.3.2 SDF-Simulation Description Format

Das *SDF*-Format wird bis jetzt ausschliesslich im *Gazebo* verwendet. Es kann kinematische Graph-Strukturen abbilden. Deshalb könne auch geschlossenen kinematische Ketten beschrieben werden.

2.3.3 Vereinfachung

Damit für ein System nicht zwei Dateien erstellt und instand gehalten werden müssen gibt es eine Lösung. Die Lösung besteht darin die *URDF*-Datei in eine *SDF*-Datei zu konvertieren. Das *URDF*-Format ist jedoch nicht so mächtig wie das *SDF*-Format. Um jedoch den ganzen Umfang der Möglichkeiten vom *SDF*-Format zu nutzen kann die *URDF*-Datei mit speziellen XML-Elementen erweitert werden.

2.3.4 TF

2.3.5 xacro

3 Motor

Stichworte

- überblick (zusammen mit EEROS)
- Aufbau urdf
- verwendung von urdf
- gazebo plugins
- joint state publisher
- rqt plugins

In diesem Kapitel wird anhand eines einfachen Fallbeispiels gezeigt, wie eine Simulation in Gazebo erstellt wird. Durch das einfache Beispiel können alle Grundlagen vorgestellt werden, die es braucht um ein Simulationsmodell aufzubauen. Ein Motor-Baugruppe (Abbildung 3.1) soll als einfache Beispiel dienen. Die Baugruppe besteht aus einem Motor, Schwungrad und linearen Dämpfer. Diese drei Komponenten sind mit einander verbunden. Der lineare Dämpfer ist realisiert als zweiter kurzgeschlossener Motor.

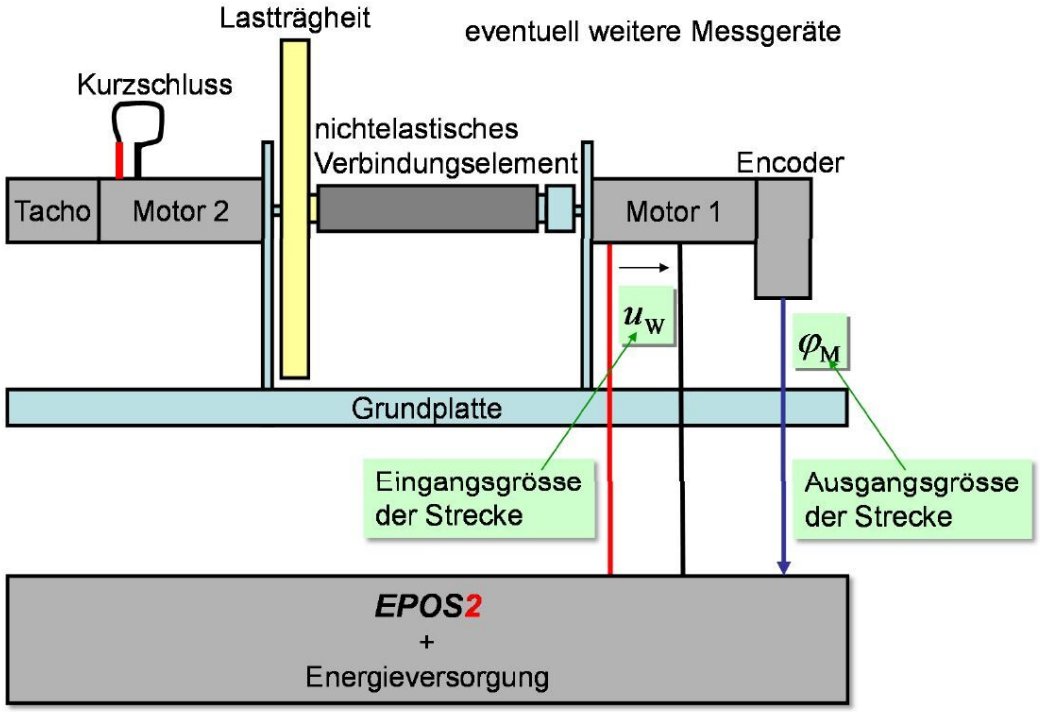

Abbildung 3.1: Motor-Baugruppe

3.1 Aufbau/Überblick

Was brauchen wir?

- beschreibung von Simulationsmodell
- beschreibung für Darstellung von Modell

erwähnen man braucht auch beschreibung wenn arbeiten ohne sim

Parameter für Model aus Datenblättern für die einzelnen Komponenten entnommen. Berechnung dämpfung Für Parameter die nicht im Datenblatt stehen oder nicht genügend Informationen für eine Berechnung vorhanden ist, wurden geschätzt. Jedoch haben diese Für jeden Link müssen immer die Masse und der Trägheits-tensor definiter sein. Dabei darf die Masse nicht 0 sein und

3.2 Gazebo

3.2.1 Gazebo Plugins

beschreiben für was die Plugins sind, wie sie eingesetzt werden

3.3 rviz

3.3.1 Joint State Publisher

erklären für was gebraucht wird

3.4 rqt

keine speziellen anpassungen an urdf

4 Delta

5 Ergebnisse, Fazit und Ausblick

5.1 Ergebnisse

- zeigen von vollständigem Zusammenspiel eeros <-> ros
- zeigen erstellen Simulation komplexer Systeme
- Vorlagen und einzeln Komponenten die wiederverwendet werden können

5.2 Fazit

- möglichkeit EEROS auszuprobieren
- ausprobieren ohne realtime kernel
- ausprobieren mit eeduro-Delta

5.3 Ausblick

- wiederverwenden von einzelnen Komponenten wie plugins
- EEROS-Applikation für Delta-roboter auf neuste EEROS-Version updaten
- erweitern von Delta-Model mit Rotation Werkzeug
-

Quellenverzeichnis

[1] *Web: ROS, https://ros.org/*
Stand vom 20.08.2017

A Anhang

- Datenblätter Motor-Baugruppe
- Datenblätter Bauteile von EEDUO-Delta
- Source code