









ROS Übersicht: Einführung (/de/ROS/Introduction) | Konzepte (/de/ROS/Concepts) | Higher-Level Concepts (/ROS/Higher-Level%20Concepts) | Client Libraries (/Client%20Libraries) | Technical Overview (/ROS/Technical%20Overview)

Inhaltsverzeichnis

1. Was ist ROS?
2. Ziele
3. Betriebssysteme
4. Releases
5. Mithilfe
6. Next


1. Was ist ROS?

ROS ist ein Open Source, Meta-Betriebssystem für deinen Roboter. Es stellt Dienste zur Verfügung, welche du von einem Betriebssystem erwartest: Hardwareabstraktion, Gerätetreiber, Utilityfunktionen, Interprozesskommunikation und Paketmanagement. Des Weiteren sind Werkzeuge und Bibliotheken für das Beziehen, Builden, Schreiben und Ausführen von Code über mehrere Computer vorhanden. ROS kann in einigen Aspekten mit anderen Roboterframeworks verglichen werden. Dazu gehören:  Player (<http://playerstage.sf.net>),  YARP (<http://eris.liralab.it/yarp/>),  Orocos (<http://www.orocos.org/>),  CARMEN (<http://carmen.sourceforge.net>),  Orca (<http://orca-robotics.sourceforge.net>),  MOOS (<http://www.robots.ox.ac.uk/~pnewman/TheMOOS/index.html>) sowie  Microsoft Robotics Studio (<http://msdn.microsoft.com/en-us/robotics/default.aspx>).

Der ROS Laufzeitgraph ist ein  Peer-to-Peer Netzwerk (<http://de.wikipedia.org/wiki/Peer-to-Peer>) von Prozessen, welche via die ROS Kommunikationsinfrastruktur lose gekoppelt sind. ROS stellt verschiedene Kommunikationsarten zur Verfügung.

- Die synchrone Kommunikation über Services (/Services) (ähnlich einem Remote Procedure Call)
- Asynchrones Streamen über Topics (/Topics)
- Datenspeicher auf dem Parameter Server (/Parameter%20Server)

Details sind im Konzept (/de/ROS/Concepts) zu finden.

ROS ist kein Echtzeit Framework obwohl es möglich ist ROS mit Echtzeitkomponenten zu vereinen. Der PR2 Roboter von Willow Garage verwendet pr2_etherCAT (/pr2_etherCAT), welches Nachrichten zwischen ROS und Echtzeitprozessen austauscht. Des Weiteren besteht eine nahtlose  Integration zum Orocos Framework (<http://www.willowgarage.com/blog/2009/06/10/orocos-rtt-and-ros-integrated>)

2. Ziele

Viele fragen sich worin sich ROS von anderen Roboter Frameworks unterscheidet. Diese Frage ist schwer zu beantworten, da ROS nicht zum Ziel hat ein Framework mit den meisten Funktionen zu sein. Stattdessen ist der Hauptzweck von ROS die Wiederverwendung von Code in der Roboterforschung und -entwicklung. ROS ist ein verteiltes System von Prozessen (Nodes), welches die lose Kopplung von individuellen Komponenten ermöglicht. Für die einfache Handhabung und Verteilung werden diese in Paketen und Stacks organisiert. ROS unterstützt zudem den Zusammenschluss von Code

Repositories, wodurch auch die Zusammenarbeit über verteilte Infrastrukturen ermöglicht wird. Dieses Design, vom Dateisystem bis zur Community, ermöglicht unabhängige Entscheidungen bezüglich Entwicklung und Implementierung, welche mit Hilfe der ROS Infrastrukturwerkzeuge vereint werden können.

Weitere Zielsetzungen:

- Thin: Für ROS geschriebener Code soll auch mit anderen Roboter Frameworks verwendet werden können. ROS wurde bereits erfolgreich mit anderen Systemen kombiniert (darunter OpenRAVE, Orocos, Player).
- ROS-agnostic libraries: Das bevorzugte Entwicklungsvorgehen ist Bibliotheken mit klaren Schnittstellen zu schreiben, welche nicht an ROS gebunden sind.
- Language independence: Das ROS Framework kann einfach in jeder modernen Programmiersprache implementiert werden. Es bestehen Implementierungen in Python (/rospy), C++ (/roscpp) und Lisp (/roslisp), für Java und Lua existieren experimentelle Bibliotheken.
- Easy testing: ROS verfügt über ein eingebautes Unit- und Integrationstest Framework rostest (/roctest).
- Scaling: ROS funktioniert in grossen Laufzeitumgebungen sowie auch für grosse Entwicklungsprozesse.

3. Betriebssysteme

Zurzeit läuft ROS nur auf Unix-basierten Plattformen. Die Software wird hauptsächlich für Ubuntu und Mac OS X getestet, die Community hat jedoch auch Hilfestellung für Fedora, Gentoo und andere Linux Systeme geleistet.

Eine Portierung für Microsoft Windows ist möglich, jedoch noch im experimentellen Stadium.

4. Releases

Das ROS Kernsystem wird zusammen mit nützlichen Werkzeugen und Bibliotheken regelmässig als ROS Distribution (/Distributions) ausgeliefert. Diese ähnelt einer Linuxdistribution und stellt ein Set kompatibler Software zur Verfügung.

5. Mithilfe

Weil ROS unter einer Open Source Lizenz steht, hoffen wir, dass du dir Gedanken darüber machst an ROS mitzuwirken. Mehr Informationen dazu findest du unter Contributing (/Contributing).

6. Next

ROS Konzepte (/de/ROS/Concepts)

Except where otherwise noted,
the ROS wiki is licensed under
the

Wiki: de/ROS/Introduction (zuletzt geändert am 2013-06-12 16:40:16 durch TullyFoote (/TullyFoote))

Creative Commons Attribution 3.0 (<http://creativecommons.org/licenses/by/3.0/>) | Find us on Google+ (<https://plus.google.com/113789706402978299308>)

Brought to you by:  Open Source Robotics Foundation

(<http://www.osrfoundation.org>)