Leibniz Universität Hannover

GIH

Introduction to Robot Operating System (ROS)

Leibniz Universität Hannover
Geodätisches Institut
Winter Semester 2021

M.Sc. Arman Khami

What is ROS?

- The Robot Operating System (ROS) is a set of APIs and tools that help you build robot applications. (www.ros.org)

- First developed by **Stanford Artificial intelligence Laboratory** in 2007 and since 2013 managed by **Open Source Robotics Foundation** (OSRF).

Source: www.ros.org

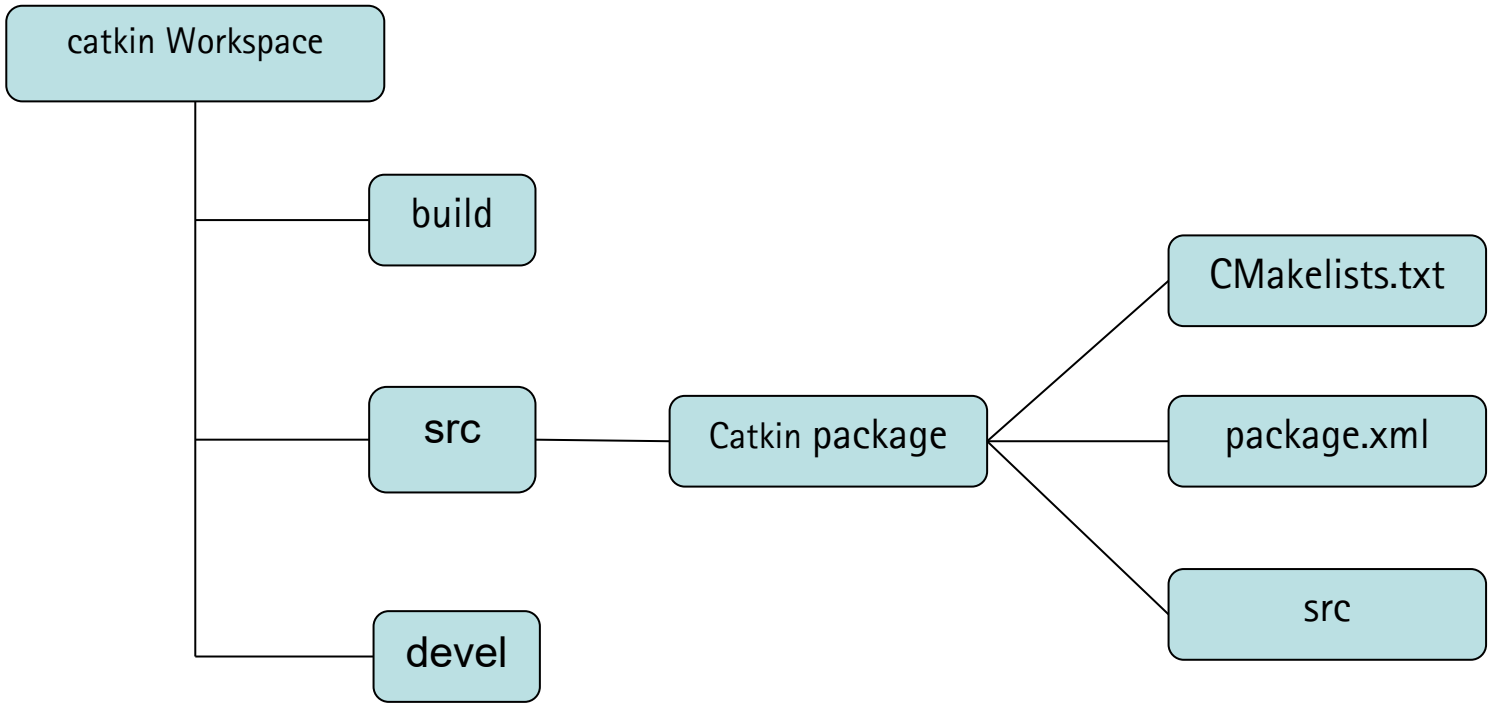
Why ROS?

- ROS is a framework and a middleware which **facilitate the communication** between sensors and use the data for different applications
- It is **open source** which means everyone can also contribute in the development of the libraries.

https://roboticsandautomationnews.com

- Individual programs (nodes) can communicate with each other over defined **API** (Application Programming Interface).
- Programming with different languages is possible. (C++, Python).

ROS file system structure (catkin Workspace)



Catkin package

- package.xml: **properties** about the package such as the **package name**, **version numbers**, **authors**, maintainers, and **dependencies** on other catkin packages
- CMakelists.txt:
  - input to the CMake build system for building software packages
  - describe how to build the code and where to install it to.
- src: contain all Nodes (Programms)

Important definitions

- roscore: is a collection of nodes and programs that are pre-requisites of a ROS-based system.
- Nodes: A node is an executable that uses ROS to communicate with other nodes. (www.ros.org)
- Messages: Nodes communicate with each other by publishing messages to topics. A message is a simple data structure, comprising typed fields. Standard primitive types (integer, floating point, boolean, etc.) are supported, as are arrays of primitive types. (www.ros.org)
- Topics: Nodes can publish messages to a topic as well as subscribe to a topic to receive messages. (www.ros.org)

Publisher and Subscriber

- ROS **Publisher** and **Subscriber** are services which allow **nodes** to communicate with each other using **published topics**.

- Each node can be a Subscriber or Publisher or **both** of them.

- A node can Subscribe to a **Topic** published by **itself** or other nodes.

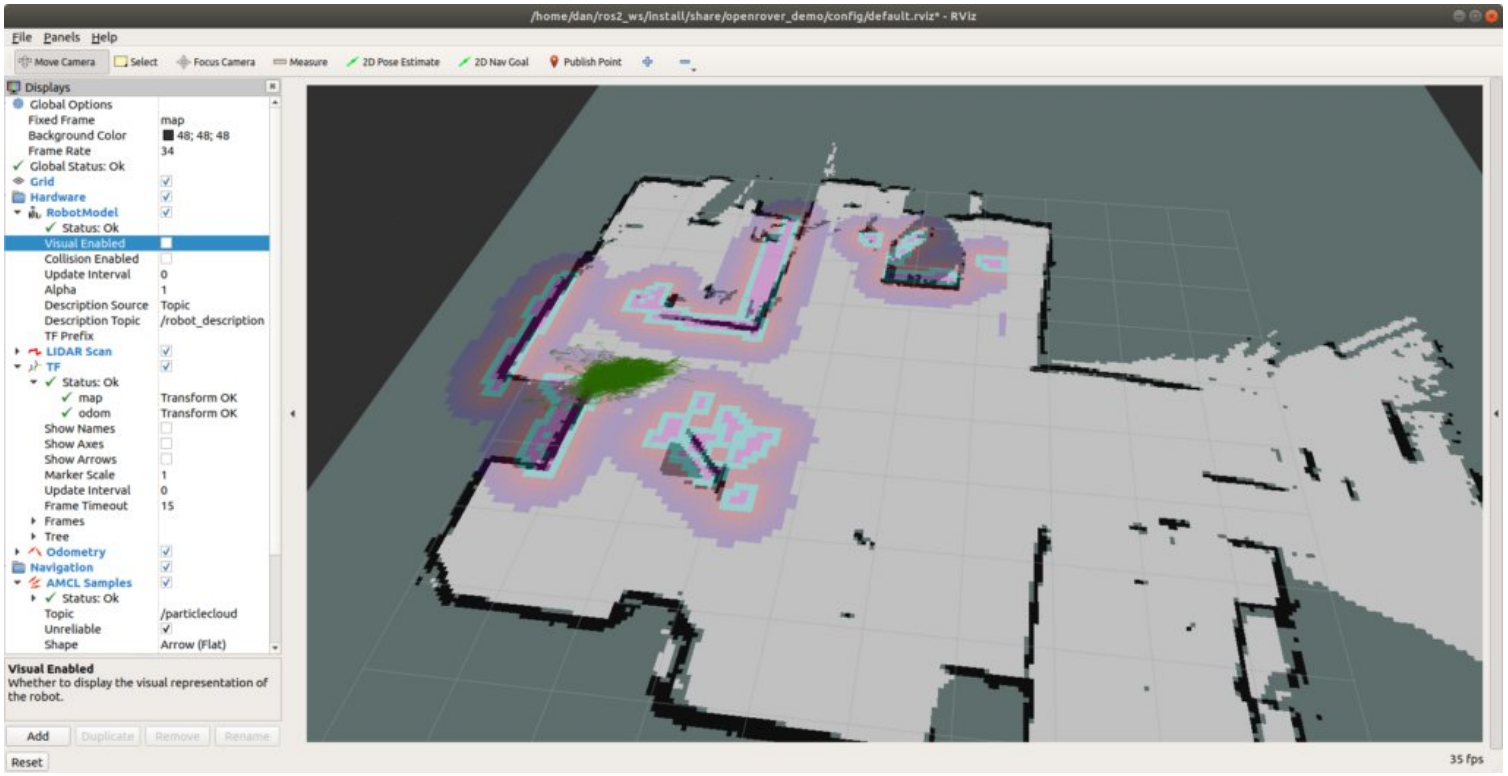
Graphical tools

- rqt_graph: illustrate the interactive diagram of the active nodes and topics their relationship.

rgt_graph (source: M. Teja Krishna 2014)

Graphical tools

- Rviz:



Source: https://blog.roverrobotics.com

ROS Cheat sheet

ROS Cheat Sheet

Filesystem Command-line Tools

rospack/rosstack	A tool inspecting packages/stacks.
roscd	Changes directories to a package or stack.
rosls	Lists package or stack information.
roscreate-pkg	Creates a new ROS package.
roscreate-stack	Creates a new ROS stack.
rosdep	Installs ROS package system dependencies.
rosmake	Builds a ROS package.
roswtf	Displays a errors and warnings about a running ROS system or launch file.
rxdeps	Displays package structure and dependencies.

Usage:

```
$ rospack find [package]
$ roscd [package[/subdir]]
$ rosls [package[/subdir]]
$ roscreate-pkg [package_name]
$ rosmake [package]
$ rosdep install [package]
$ roswtf or roswtf [file]
$ rxdeps [options]
```

Common Command-line Tools

roscore

A collection of nodes and programs that are pre-requisites of a ROS-based system. You must have a roscore running in order for ROS nodes to communicate.

roscore is currently defined as:

```
master
parameter server
rosout
```

Usage:

```
$ roscore
```

rosmsg/rossrv

rosmsg/rossrv displays Message/Service (msg/srv) data structure definitions.

Commands:

rosmsg show	Display the fields in the msg.
rosmsg users	Search for code using the msg.
rosmsg md5	Display the msg md5 sum.
rosmsg package	List all the messages in a package.
rosnode packages	List all the packages with messages.

Examples:

Display the Pose msg:

```
$ rosmsg show Pose
```

List the messages in nav_msgs:

```
$ rosmsg package nav_msgs
```

List the files using sensor_msgs/CameraInfo:

```
$ rosmsg users sensor_msgs/CameraInfo
```

rosrun

rosrun allows you to run an executable in an arbitrary package without having to cd (or roscd) there first.

Usage:

```
$ rosrun package executable
```

Example:

Run turtlesim:

```
$ rosrun turtlesim turtlesim_node
```

rosnode

Displays debugging information about ROS nodes, including publications, subscriptions and connections.

Commands:

rosnode ping	Test connectivity to node.
rosnode list	List active nodes.
rosnode info	Print information about a node.
rosnode machine	List nodes running on a particular machine.
rosnode kill	Kills a running node.

Examples:

Kill all nodes:

```
$ rosnode kill -a
```

List nodes on a machine:

```
$ rosnode machine aqy.local
```

Ping all nodes:

```
$ rosnode ping --all
```

roslaunch

Starts ROS nodes locally and remotely via SSH, as well as setting parameters on the parameter server.

Examples:

Launch on a different port:

```
$ roslaunch -p 1234 package filename.launch
```

Launch a file in a package:

```
$ roslaunch package filename.launch
```

Launch on the local nodes:

```
$ roslaunch --local package filename.launch
```

rostopic

A tool for displaying debug information about ROS topics, including publishers, subscribers, publishing rate, and messages.

Commands:

rostopic bw	Display bandwidth used by topic.
rostopic echo	Print messages to screen.
rostopic hz	Display publishing rate of topic.
rostopic list	Print information about active topics.
rostopic pub	Publish data to topic.
rostopic type	Print topic type.
rostopic find	Find topics by type.

Examples:

Publish hello at 10 Hz:

```
$ rostopic pub -r 10 /topic_name std_msgs/String hello
```

Clear the screen after each message is published:

```
$ rostopic echo -c /topic_name
```

Display messages that match a given Python expression:

```
$ rostopic echo --filter "m.data=='foo'" /topic_name
```

Pipe the output of rostopic to rosmsg to view the msg type:

```
$ rostopic type /topic_name | rosmsg show
```

rosparam

A tool for getting and setting ROS parameters on the parameter server using YAML-encoded files.

Commands:

rosparam set	Set a parameter.
rosparam get	Get a parameter.
rosparam load	Load parameters from a file.
rosparam dump	Dump parameters to a file.
rosparam delete	Delete a parameter.
rosparam list	List parameter names.

Examples:

List all the parameters in a namespace:

```
$ rosparam list /namespace
```

Setting a list with one as a string, integer, and float:

```
$ rosparam set /foo "['1', 1, 1.0]"
```

Dump only the parameters in a specific namespace to file:

```
$ rosparam dump dump.yaml /namespace
```

rosservice

A tool for listing and querying ROS services.

Commands:

rosservice list	Print information about active services.
rosservice node	Print the name of the node providing a service.
rosservice call	Call the service with the given args.
rosservice args	List the arguments of a service.
rosservice type	Print the service type.
rosservice uri	Print the service ROSRPC uri.
rosservice find	Find services by service type.

Examples:

Call a service from the command-line:

```
$ rosservice call /add_two_ints 1 2
```

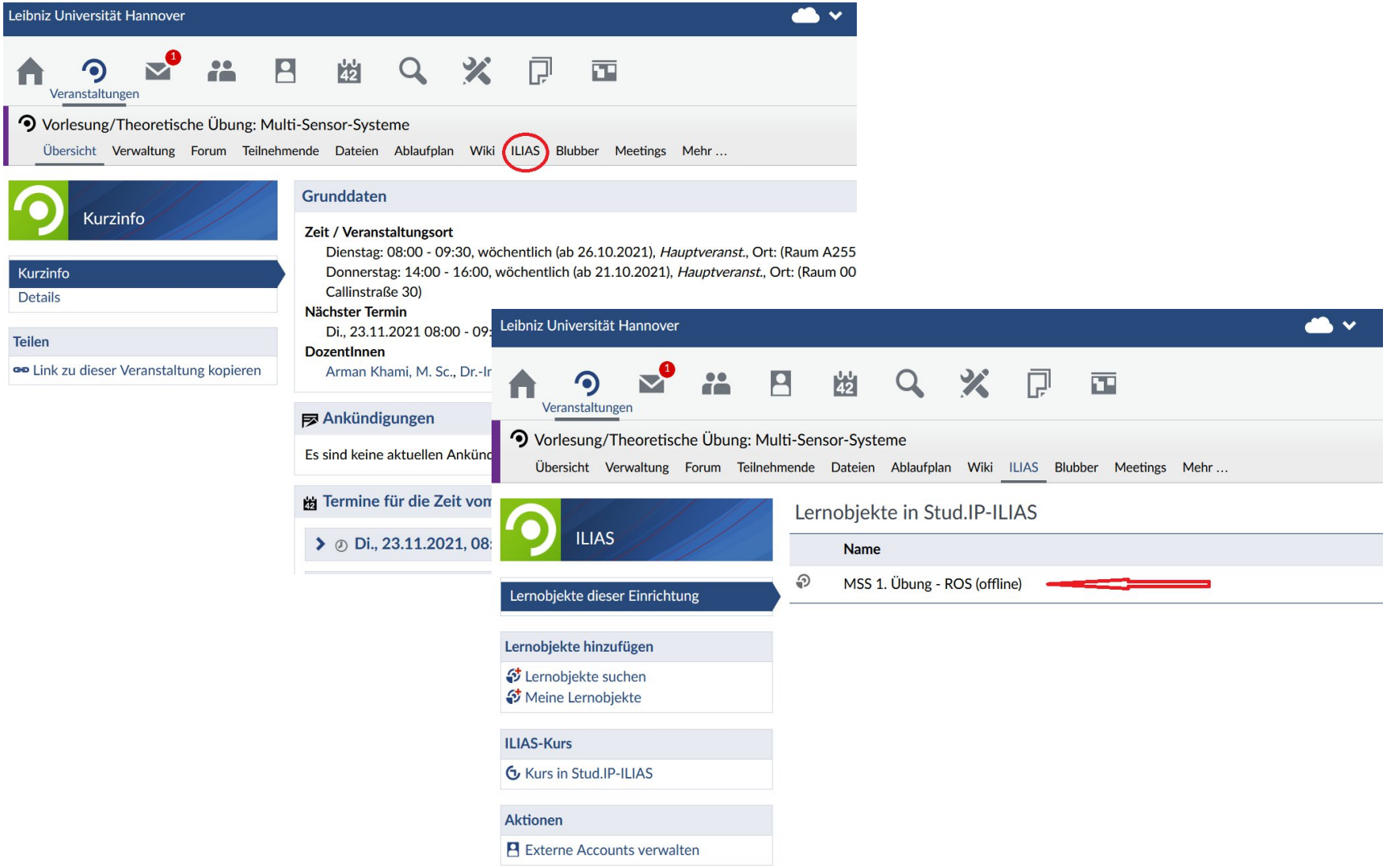
Pipe the output of rosservice to rossrv to view the srv type:

```
$ rosservice type add_two_ints | rossrv show
```

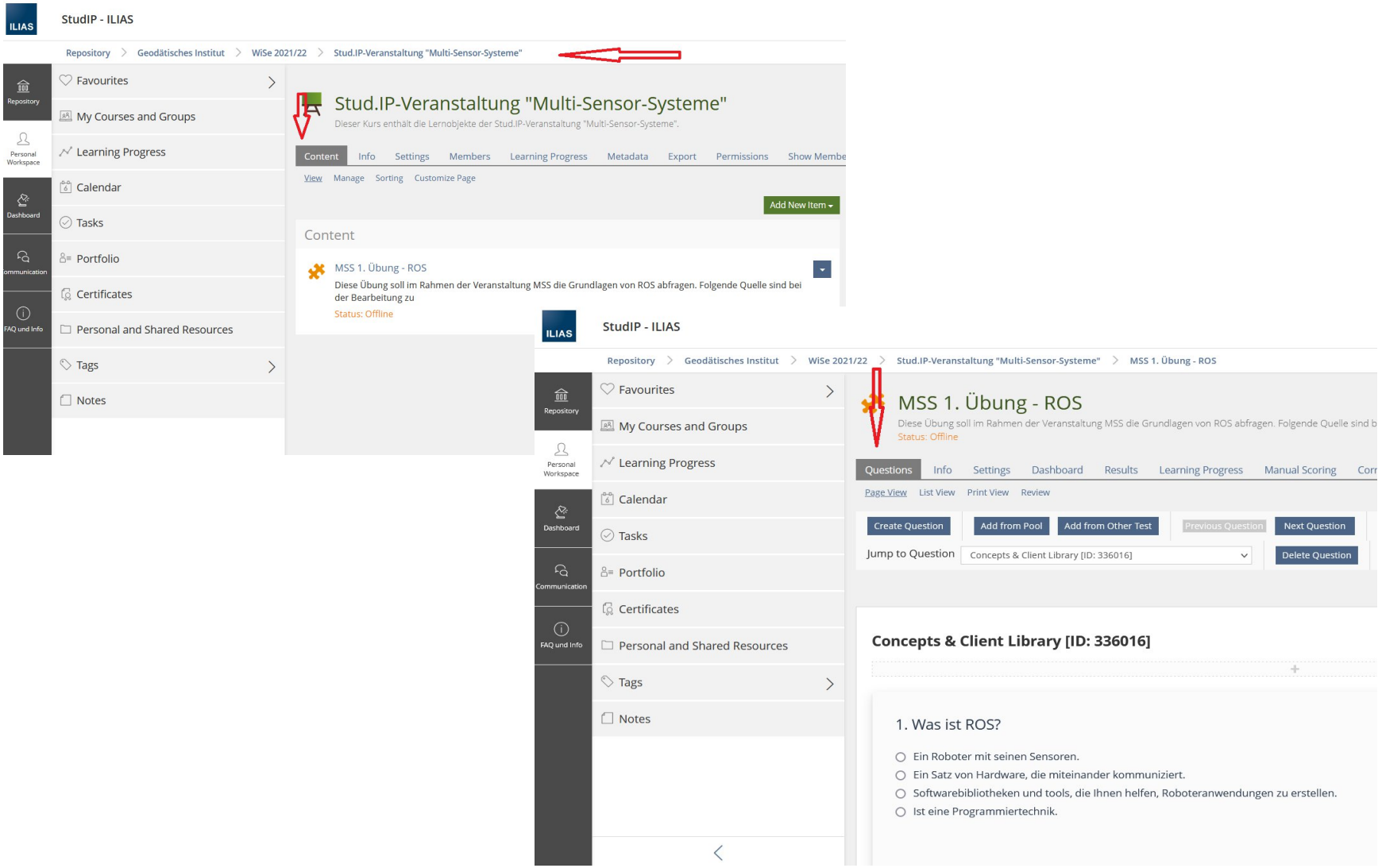
Display all services of a particular type:

```
$ rosservice find rospy_tutorials/AddTwoInts
```

Exercise


Leibniz Universität Hannover
Veranstaltungen
Vorlesung/Theoretische Übung: Multi-Sensor-Systeme
Übersicht Verwaltung Forum Teilnehmende Dateien Ablaufplan Wiki ILIAS Blubber Meetings Mehr …
Kurzinfo
Kurzinfo
Details
Teilen
Link zu dieser Veranstaltung kopieren
Grunddaten
Zeit / Veranstaltungsort
Dienstag: 08:00 - 09:30, wöchentlich (ab 26.10.2021), Hauptveranst., Ort: (Raum A255
Donnerstag: 14:00 - 16:00, wöchentlich (ab 21.10.2021), Hauptveranst., Ort: (Raum 00
Callinstraße 30)
Nächster Termin
Di., 23.11.2021 08:00 - 09:
DozentInnen
Arman Khami, M. Sc., Dr.-Ir
Ankündigungen
Es sind keine aktuellen Ankünd
Termine für die Zeit vom
Di., 23.11.2021, 08:
Leibniz Universität Hannover
Veranstaltungen
Vorlesung/Theoretische Übung: Multi-Sensor-Systeme
Übersicht Verwaltung Forum Teilnehmende Dateien Ablaufplan Wiki ILIAS Blubber Meetings Mehr …
ILIAS
Lernobjekte dieser Einrichtung
Lernobjekte hinzufügen
Lernobjekte suchen
Meine Lernobjekte
ILIAS-Kurs
Kurs in Stud.IP-ILIAS
Aktionen
Externe Accounts verwalten
Lernobjekte in Stud.IP-ILIAS
Name
MSS 1. Übung - ROS (offline)

Excercise


StudIP - ILIAS
Repository > Geodätisches Institut > WiSe 2021/22 > Stud.IP-Veranstaltung "Multi-Sensor-Systeme"
Favourites
My Courses and Groups
Learning Progress
Calendar
Tasks
Portfolio
Certificates
Personal and Shared Resources
Tags
Notes
Stud.IP-Veranstaltung "Multi-Sensor-Systeme"
Dieser Kurs enthält die Lernobjekte der Stud.IP-Veranstaltung "Multi-Sensor-Systeme".
Content
Info
Settings
Members
Learning Progress
Metadata
Export
Permissions
View
Manage
Sorting
Customize Page
Add New Item
Content
MSS 1. Übung - ROS
Diese Übung soll im Rahmen der Veranstaltung MSS die Grundlagen von ROS abfragen. Folgende Quelle sind bei der Bearbeitung zu
Status: Offline
StudIP - ILIAS
Repository > Geodätisches Institut > WiSe 2021/22 > Stud.IP-Veranstaltung "Multi-Sensor-Systeme" > MSS 1. Übung - ROS
MSS 1. Übung - ROS
Diese Übung soll im Rahmen der Veranstaltung MSS die Grundlagen von ROS abfragen. Folgende Quelle sind b
Status: Offline
Questions
Info
Settings
Dashboard
Results
Learning Progress
Manual Scoring
Page View
List View
Print View
Review
Create Question
Add from Pool
Add from Other Test
Previous Question
Next Question
Jump to Question
Concepts & Client Library [ID: 336016]
Delete Question
Concepts & Client Library [ID: 336016]
1. Was ist ROS?
Ein Roboter mit seinen Sensoren.
Ein Satz von Hardware, die miteinander kommuniziert.
Softwarebibliotheken und tools, die Ihnen helfen, Roboteranwendungen zu erstellen.
Ist eine Programmiertechnik.

References

- www.ros.org
- www.roboticsandautomationnews.com
- amidi, Teja Krishna & Kumar, Gangula. (2014). Control of IIT Kharagpur Humanoid using Robotic operating system.. 10.13140/RG.2.1.4318.6645.