# ROS机械臂开发：从入门到实战

## —— 第2讲：风靡机器人圈的ROS到底是什么

主讲人 胡春旭

机器人博客“古月居”博主
《ROS机器人开发实践》作者
武汉精锋微控科技有限公司 联合创始人
华中科技大学 自动化学院 硕士

## 目录

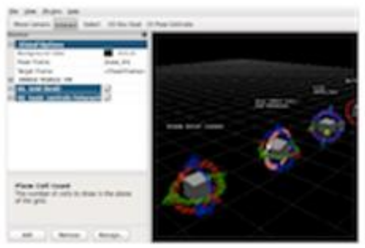
## 1. ROS是什么

## 1. ROS是什么

=

通信机制

+



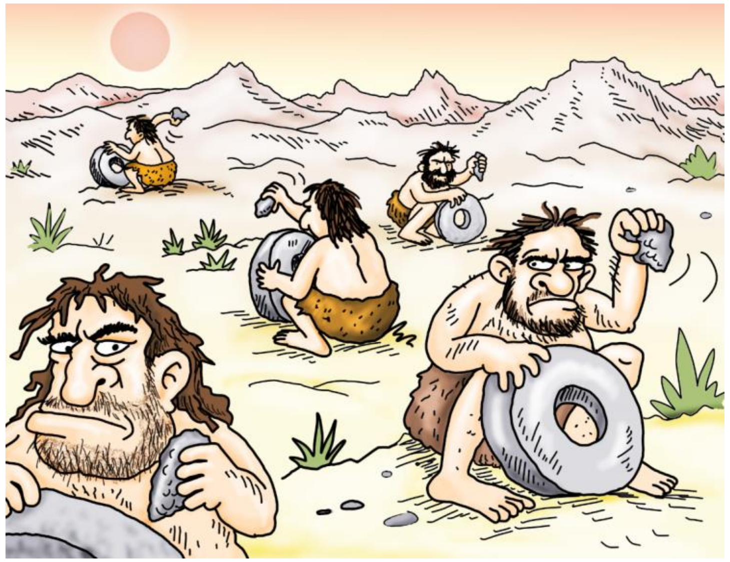
开发工具

+

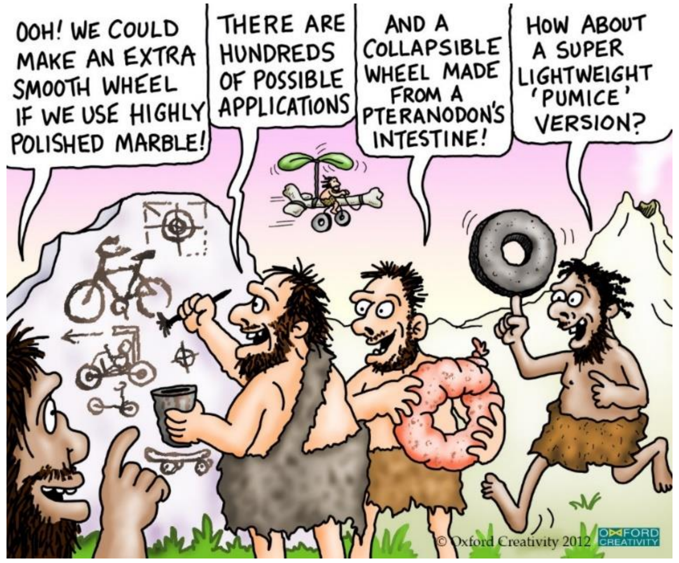
应用功能

+

生态系统



传统模式

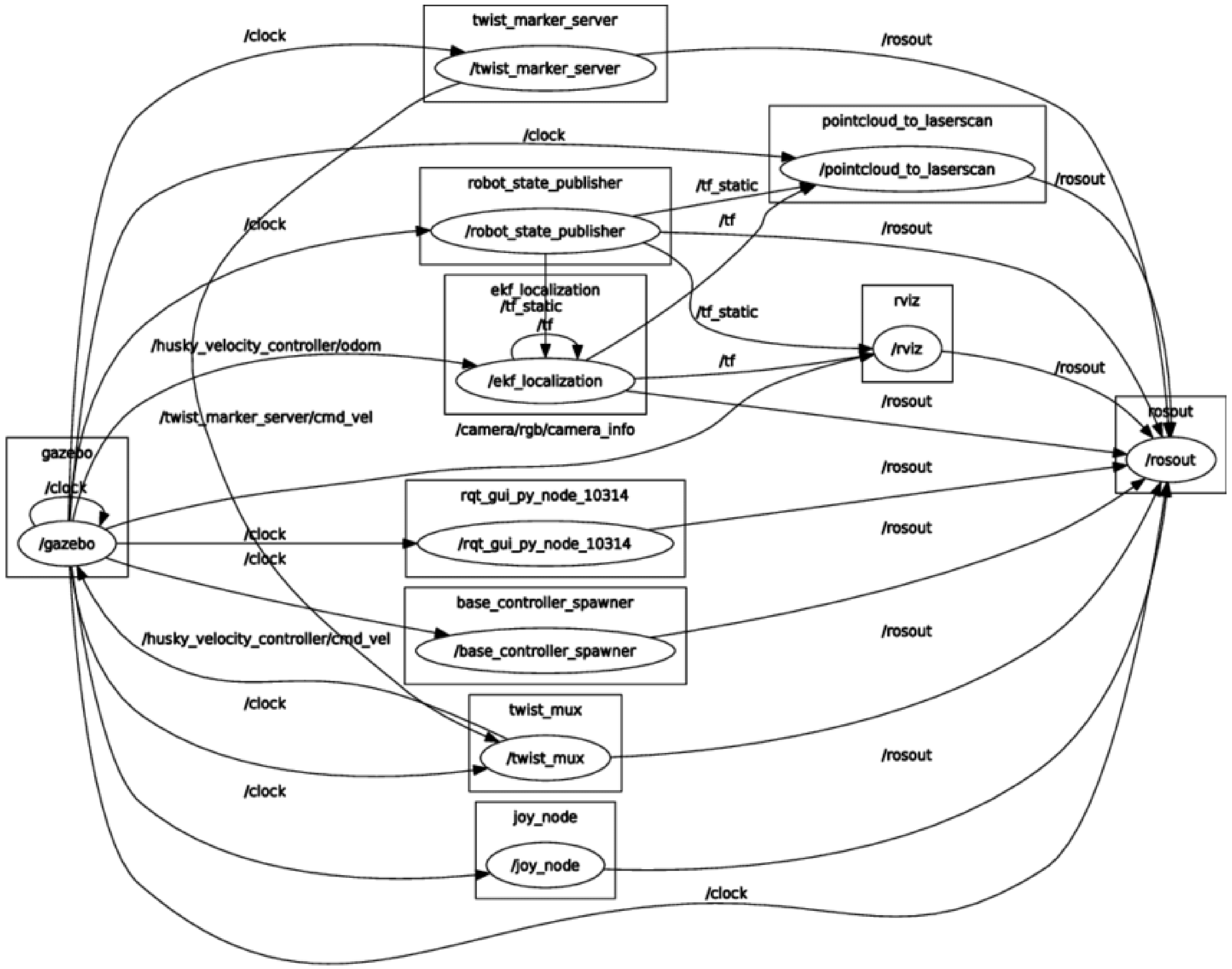

OOH! WE COULD MAKE AN EXTRA SMOOTH WHEEL IF WE USE HIGHLY POLISHED MARBLE!
THERE ARE HUNDREDS OF POSSIBLE APPLICATIONS
AND A COLLAPSIBLE WHEEL MADE FROM A PTERANODON'S INTESTINE!
HOW ABOUT A SUPER LIGHTWEIGHT 'PUMICE' VERSION?
© Oxford Creativity 2012
OXFORD CREATIVITY

现代模式

**提高机器人研发中的软件复用率**

## 2. ROS的通信机制

## 2. ROS的通信机制 —— 松耦合分布式通信



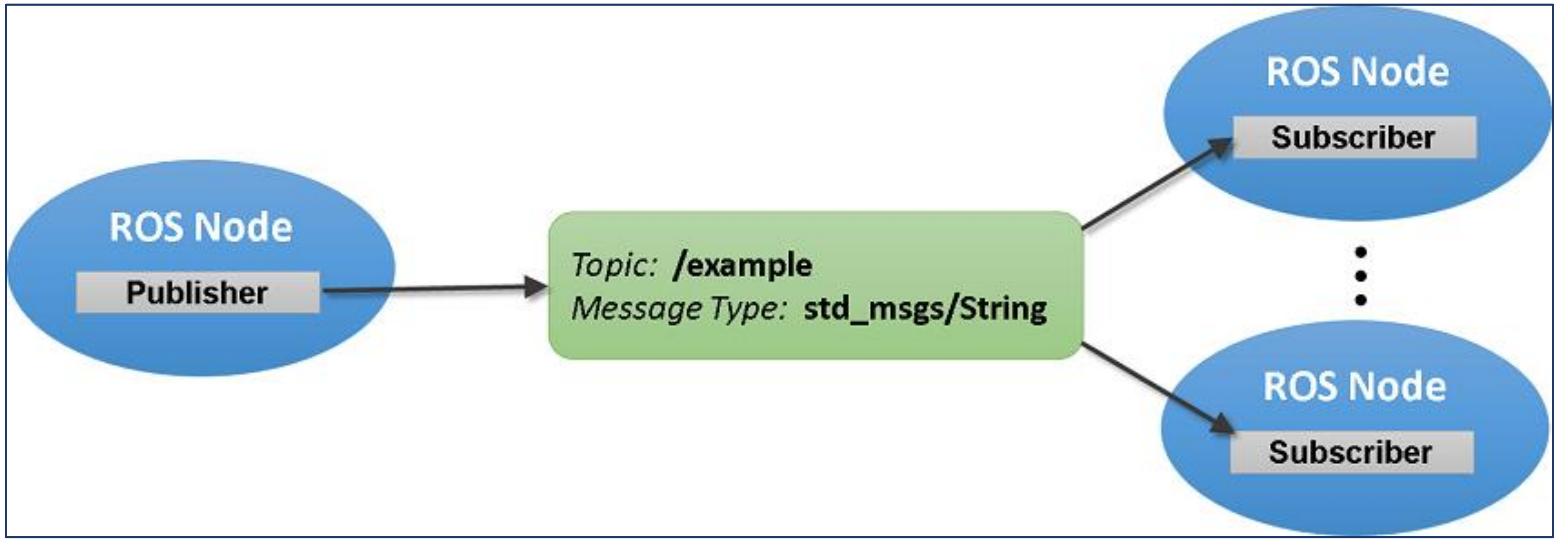
## 2. ROS的通信机制 —— 核心概念

（1）节点（Node）—— 软件模块

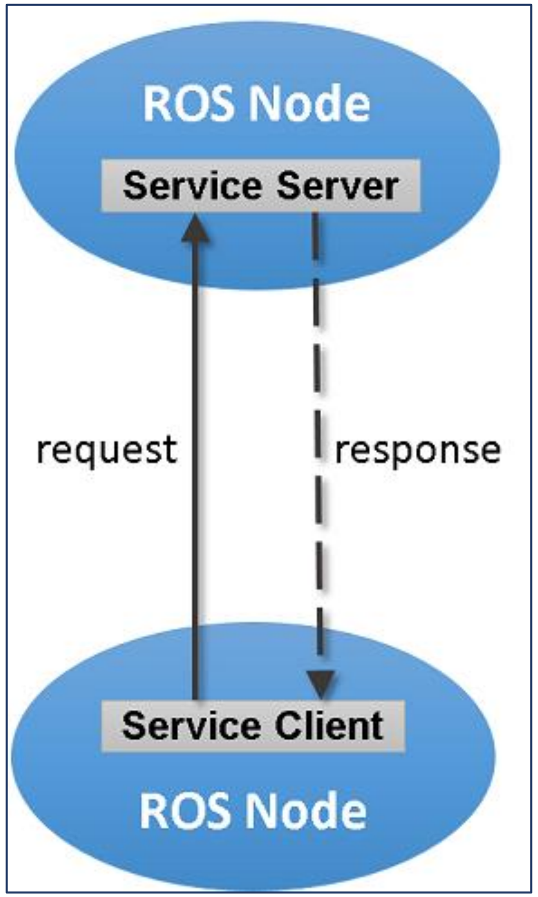
（2）节点管理器 （ROS Master）—— 控制中心，提供参数管理

（3）话题（Topic）—— 异步通信机制，传输消息（Message）

（4）服务（Service）—— 同步通信机制，传输请求/应答数据



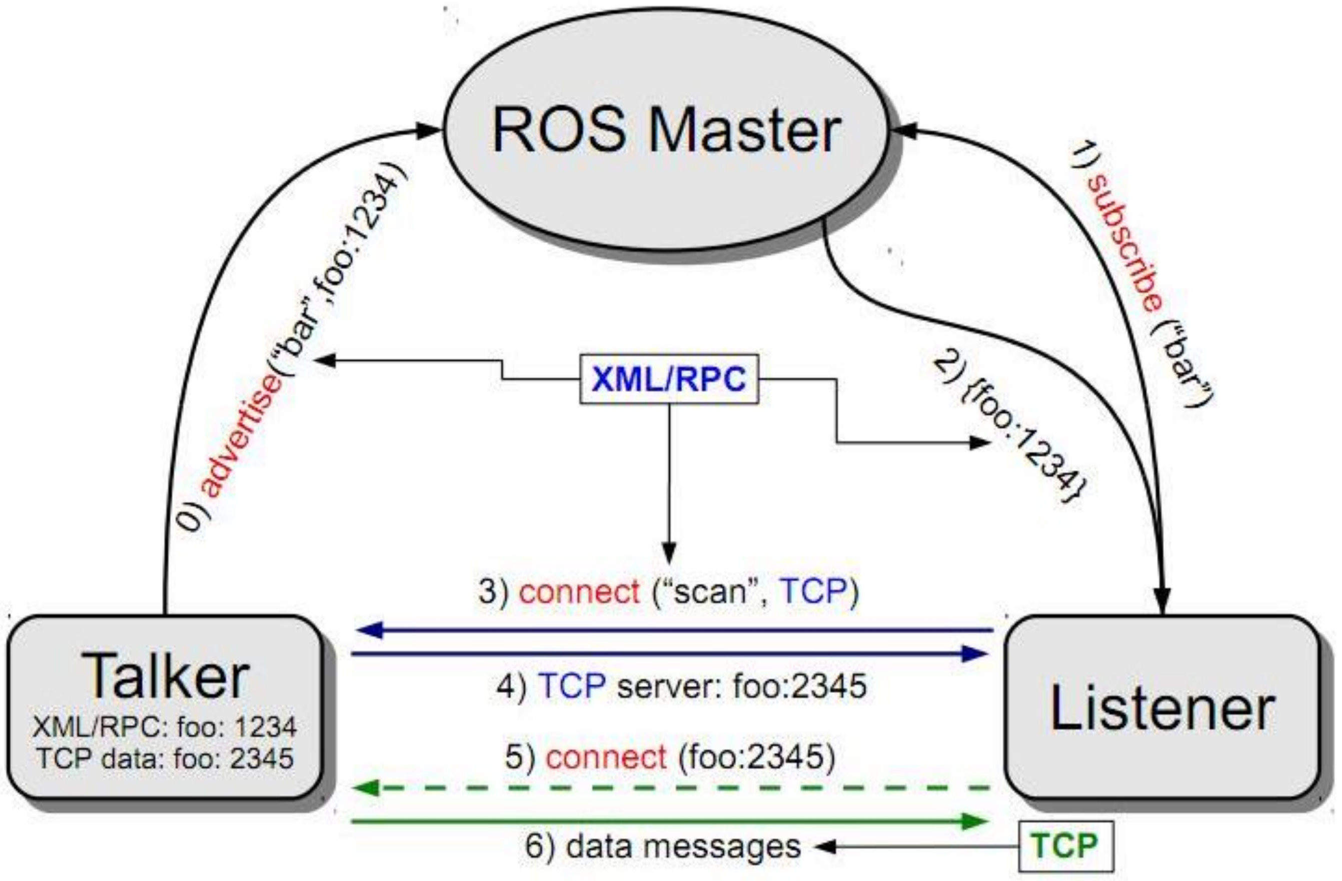
话题模型（发布/订阅）



服务模型
（请求/应答）

## 2. ROS的通信机制 —— 话题通信模型

- Talker注册
- Listener注册
- ROS Master进行信息匹配
- Listener发送连接请求
- Talker确认连接请求
- 建立网络连接
- Talker向Listener发布数据

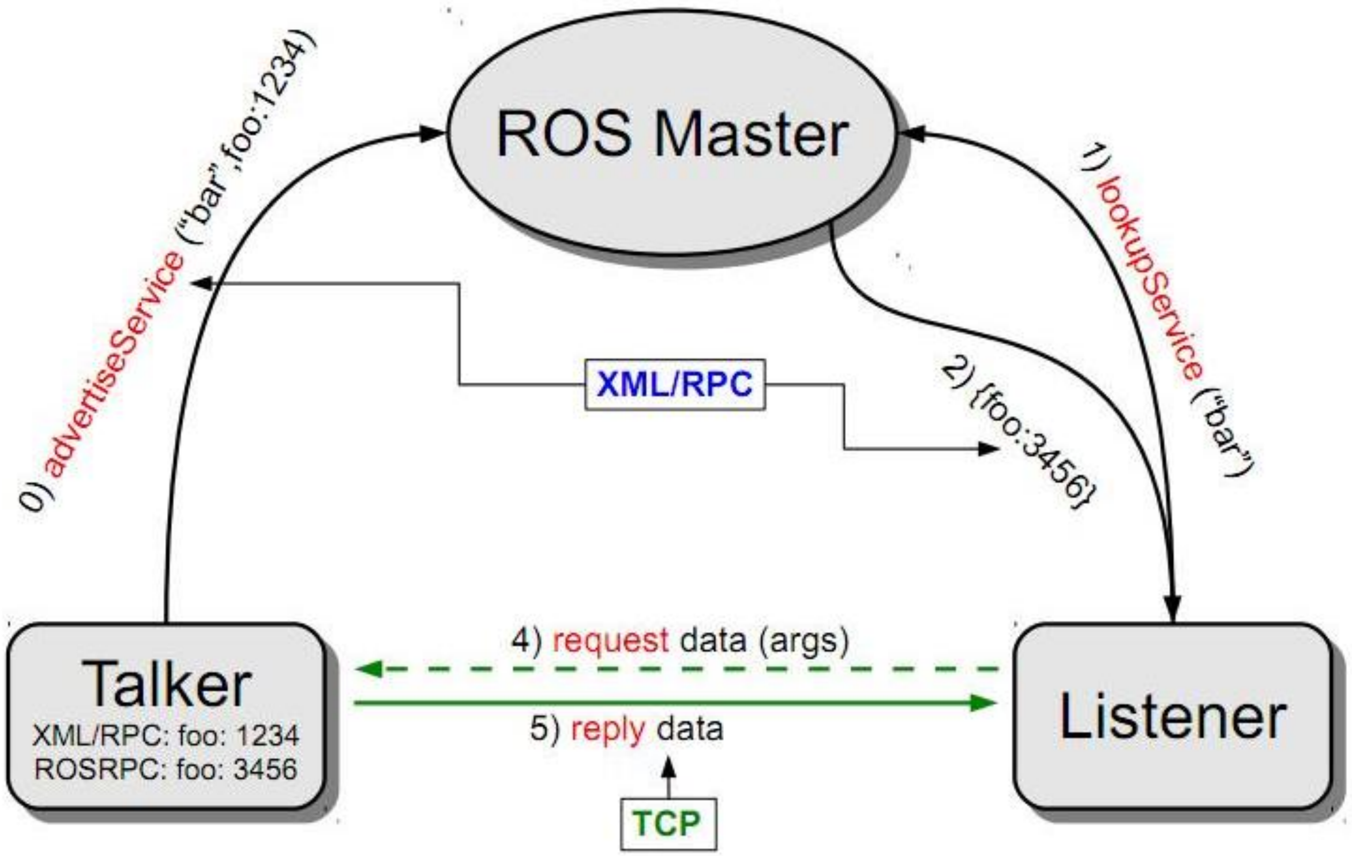


**话题通讯的建立过程**

（前五个步骤：RPC，最后两个步骤：TCP）

# 2. ROS的通信机制 —— 服务通信模型

- Talker注册
- Listener注册
- ROS Master进行信息匹配
- 建立网络连接
- Talker向Listener发布服务应答数据



**服务通讯的建立过程**

（前三个步骤：RPC，最后两个步骤：TCP）

## 2. ROS的通信机制 —— 话题 vs 服务

古月居 GYH.AI

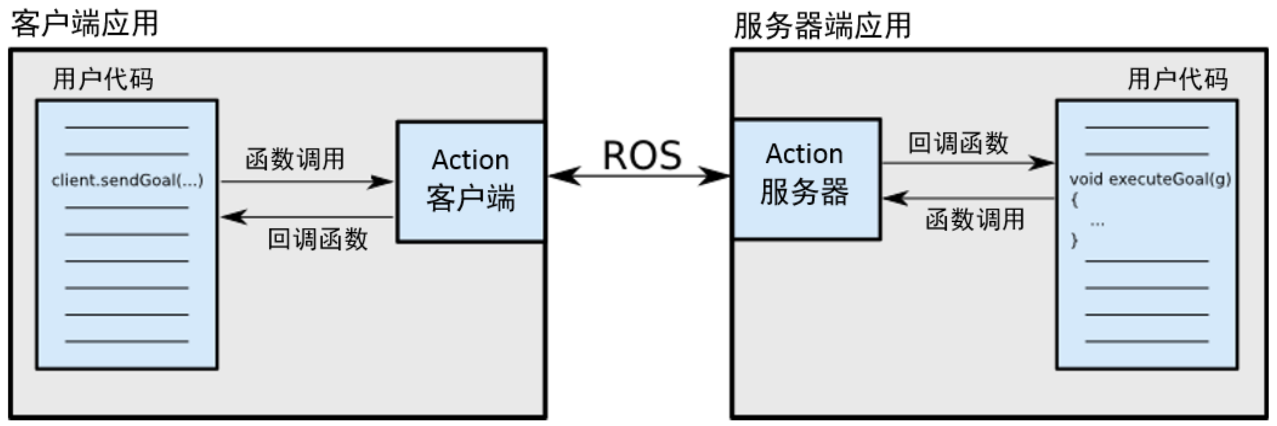
### 话题与服务的区别

| | 话题 | 服务 |
|---|---|---|
| 同步性 | 异步 | 同步 |
| 通信模型 | 发布/订阅 | 服务器/客户端 |
| 底层协议 | ROSTCP/ROSUDP | ROSTCP/ROSUDP |
| 反馈机制 | 无 | 有 |
| 缓冲区 | 有 | 无 |
| 实时性 | 弱 | 强 |
| 节点关系 | 多对多 | 一对多（一个server） |
| 适用场景 | 数据传输 | 逻辑处理 |

## 2. ROS的通信机制 —— 动作

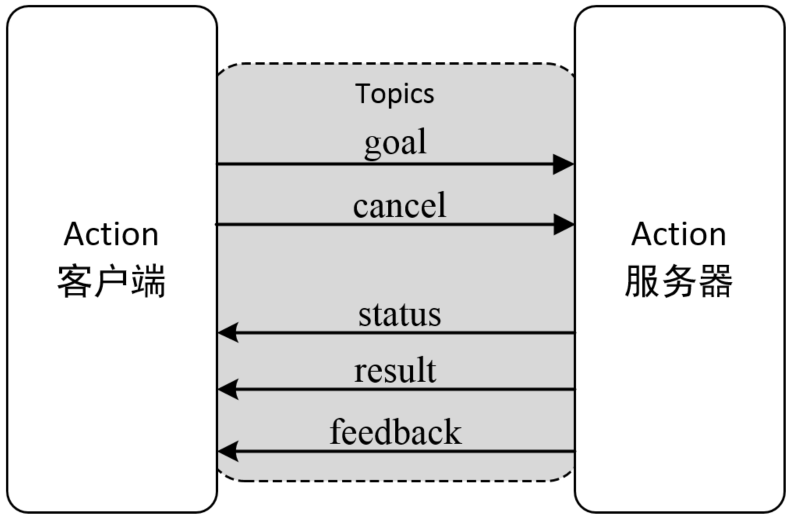
### 什么是动作（action）

- 一种问答通信机制;
- 带有连续反馈;
- 可以在任务过程中止运行;
- 基于ROS的消息机制实现。



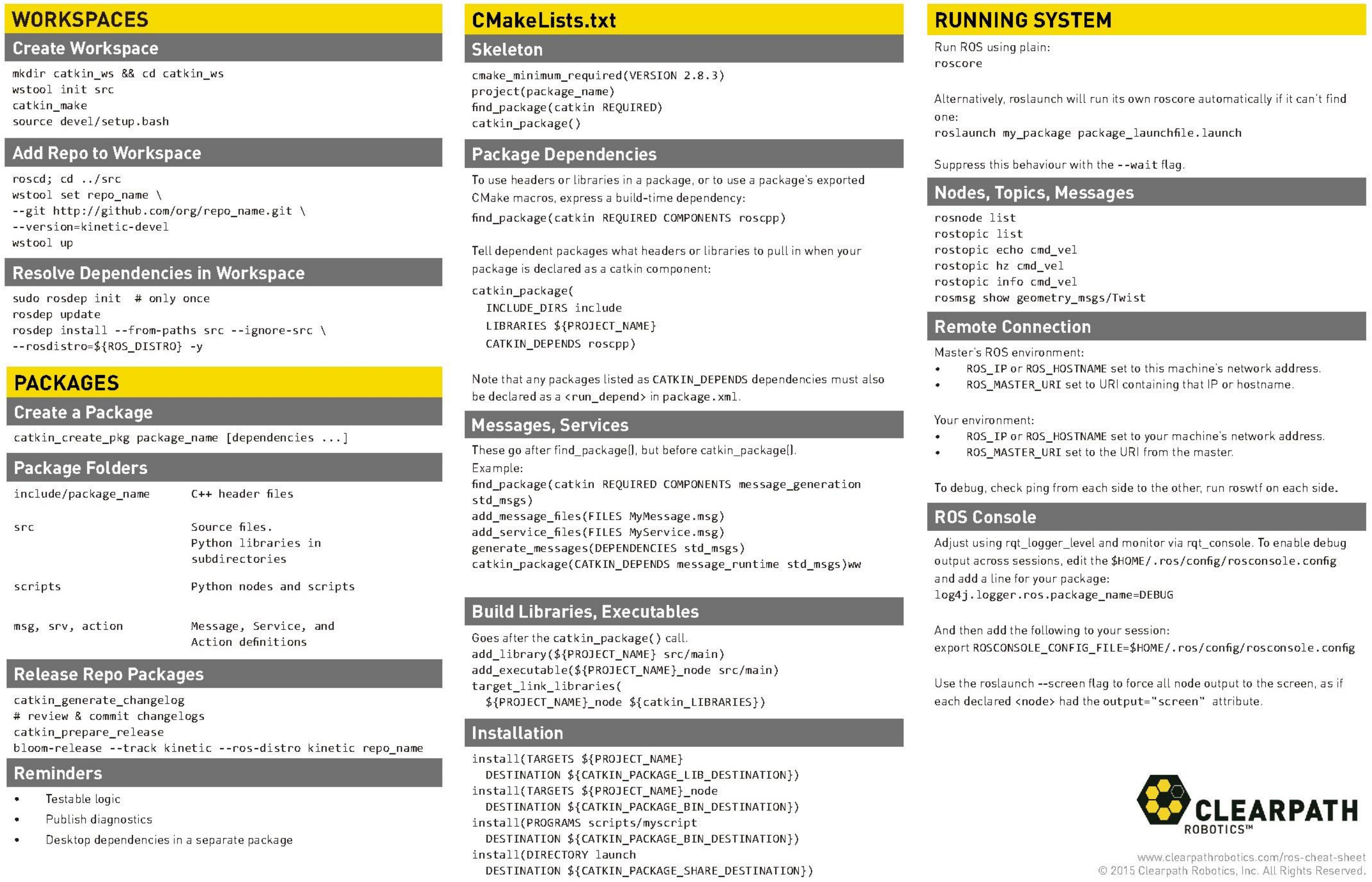
### Action的接口

- goal：发布任务目标;
- cancel：请求取消任务;
- status：通知客户端当前的状态;
- feedback：周期反馈任务运行的监控数据;
- result：向客户端发送任务的执行结果，只发布一次。



## 3. ROS的开发工具

# 3. ROS的开发工具 —— 命令行&编译器



## WORKSPACES

### Create Workspace

```
mkdir catkin_ws && cd catkin_ws
wstool init src
catkin_make
source devel/setup.bash
```

### Add Repo to Workspace

```
roscd; cd ../src
wstool set repo_name \
--git http://github.com/org/repo_name.git \
--version=kinetic-devel
wstool up
```

### Resolve Dependencies in Workspace

```
sudo rosdep init  # only once
rosdep update
rosdep install --from-paths src --ignore-src \
--rosdistro=${ROS_DISTRO} -y
```

## PACKAGES

### Create a Package

```
catkin_create_pkg package_name [dependencies ...]
```

### Package Folders

| | |
|---|---|
| include/package_name | C++ header files |
| src | Source files. Python libraries in subdirectories |
| scripts | Python nodes and scripts |
| msg, srv, action | Message, Service, and Action definitions |

### Release Repo Packages

```
catkin_generate_changelog
# review & commit changelogs
catkin_prepare_release
bloom-release --track kinetic --ros-distro kinetic repo_name
```

### Reminders

- Testable logic
- Publish diagnostics
- Desktop dependencies in a separate package

## CMakeLists.txt

### Skeleton

```
cmake_minimum_required(VERSION 2.8.3)
project(package_name)
find_package(catkin REQUIRED)
catkin_package()
```

### Package Dependencies

To use headers or libraries in a package, or to use a package's exported CMake macros, express a build-time dependency:

```
find_package(catkin REQUIRED COMPONENTS roscpp)
```

Tell dependent packages what headers or libraries to pull in when your package is declared as a catkin component:

```
catkin_package(
  INCLUDE_DIRS include
  LIBRARIES ${PROJECT_NAME}
  CATKIN_DEPENDS roscpp)
```

Note that any packages listed as CATKIN_DEPENDS dependencies must also be declared as a <run_depend> in package.xml.

### Messages, Services

These go after find_package(), but before catkin_package().

Example:

```
find_package(catkin REQUIRED COMPONENTS message_generation
std_msgs)
add_message_files(FILES MyMessage.msg)
add_service_files(FILES MyService.msg)
generate_messages(DEPENDENCIES std_msgs)
catkin_package(CATKIN_DEPENDS message_runtime std_msgs)ww
```

### Build Libraries, Executables

Goes after the catkin_package() call.

```
add_library(${PROJECT_NAME} src/main)
add_executable(${PROJECT_NAME}_node src/main)
target_link_libraries(
  ${PROJECT_NAME}_node ${catkin_LIBRARIES})
```

### Installation

```
install(TARGETS ${PROJECT_NAME}
  DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION})
install(TARGETS ${PROJECT_NAME}_node
  DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION})
install(PROGRAMS scripts/myscript
  DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION})
install(DIRECTORY launch
  DESTINATION ${CATKIN_PACKAGE_SHARE_DESTINATION})
```

## RUNNING SYSTEM

Run ROS using plain:

```
roscore
```

Alternatively, roslaunch will run its own roscore automatically if it can't find one:

```
roslaunch my_package package_launchfile.launch
```

Suppress this behaviour with the --wait flag.

### Nodes, Topics, Messages

```
rosnode list
rostopic list
rostopic echo cmd_vel
rostopic hz cmd_vel
rostopic info cmd_vel
rosmsg show geometry_msgs/Twist
```

### Remote Connection

Master's ROS environment:

- ROS_IP or ROS_HOSTNAME set to this machine's network address.
- ROS_MASTER_URI set to URI containing that IP or hostname.

Your environment:

- ROS_IP or ROS_HOSTNAME set to your machine's network address.
- ROS_MASTER_URI set to the URI from the master.

To debug, check ping from each side to the other, run roswtf on each side.

### ROS Console

Adjust using rqt_logger_level and monitor via rqt_console. To enable debug output across sessions, edit the $HOME/.ros/config/rosconsole.config and add a line for your package:

```
log4j.logger.ros.package_name=DEBUG
```

And then add the following to your session:

```
export ROSCONSOLE_CONFIG_FILE=$HOME/.ros/config/rosconsole.config
```

Use the roslaunch --screen flag to force all node output to the screen, as if each declared <node> had the output="screen" attribute.

CLEARPATH ROBOTICS™

www.clearpathrobotics.com/ros-cheat-sheet

## 3. ROS的开发工具 —— Launch启动文件

```xml
<launch>
  <!-- local machine already has a definition by default.
       This tag overrides the default definition with
       specific ROS_ROOT and ROS_PACKAGE_PATH values -->
  <machine name="local_alt" address="localhost" default="true" ros-root="/u/user/ros/ros/" ros-package-path="/u/user/ros/ros-pkg" />
  <!-- a basic listener node -->
  <node name="listener-1" pkg="rospy_tutorials" type="listener" />
  <!-- pass args to the listener node -->
  <node name="listener-2" pkg="rospy_tutorials" type="listener" args="-foo arg2" />
  <!-- a respawn-able listener node -->
  <node name="listener-3" pkg="rospy_tutorials" type="listener" respawn="true" />
  <!-- start listener node in the 'wg1' namespace -->
  <node ns="wg1" name="listener-wg1" pkg="rospy_tutorials" type="listener" respawn="true" />
  <!-- start a group of nodes in the 'wg2' namespace -->
  <group ns="wg2">
    <!-- remap applies to all future statements in this scope. -->
    <remap from="chatter" to="hello"/>
    <node pkg="rospy_tutorials" type="listener" name="listener" args="--test" respawn="true" />
    <node pkg="rospy_tutorials" type="talker" name="talker">
      <!-- set a private parameter for the node -->
      <param name="talker_1_param" value="a value" />
      <!-- nodes can have their own remap args -->
      <remap from="chatter" to="hello-1"/>
      <!-- you can set environment variables for a node -->
      <env name="ENV_EXAMPLE" value="some value" />
    </node>
  </group>
</launch>
```

Launch文件：通过XML文件实现多节点的配置和启动（**可自动启动ROS Master**）

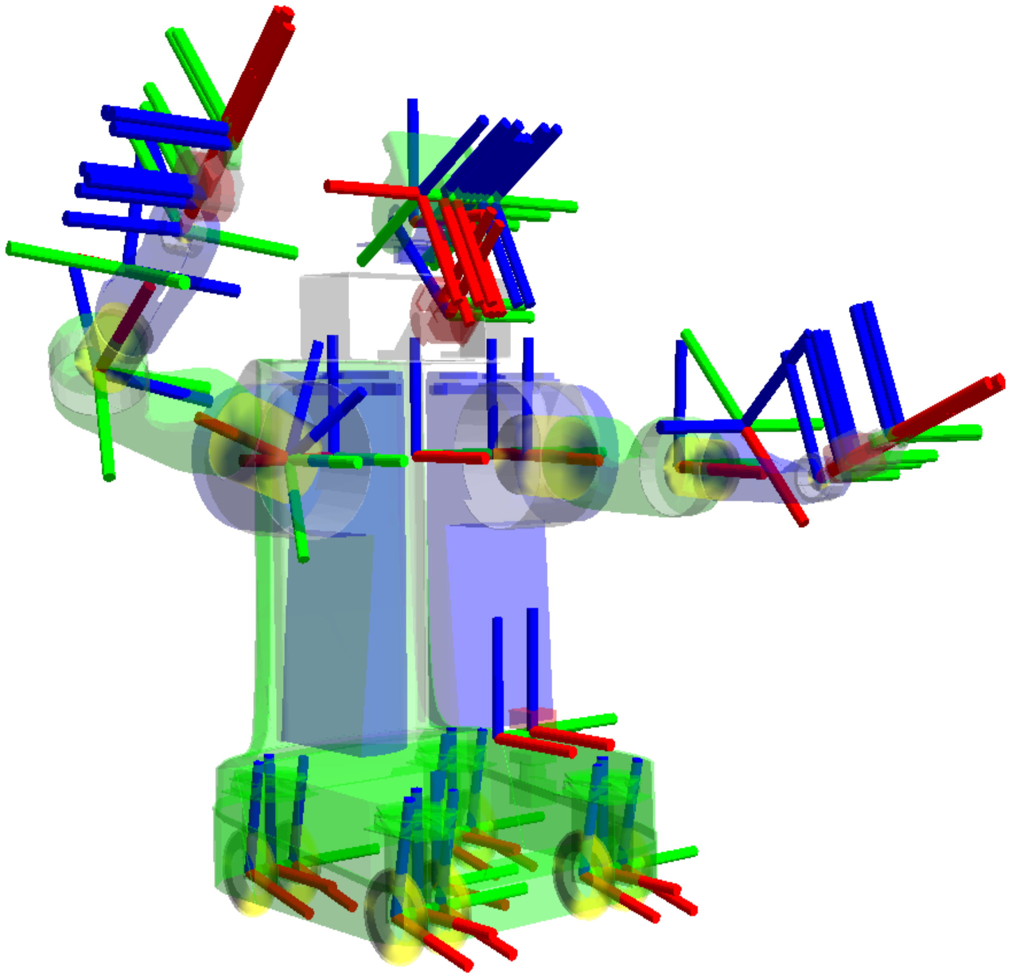
## 3. ROS的开发工具 —— TF坐标变换库

TF功能包能干什么?

- 五秒钟之前，机器人头部坐标系相对于全局坐标系的关系是什么样的?
- 机器人夹取的物体相对于机器人中心坐标系的位置在哪里?
- 机器人中心坐标系相对于全局坐标系的位置在哪里?

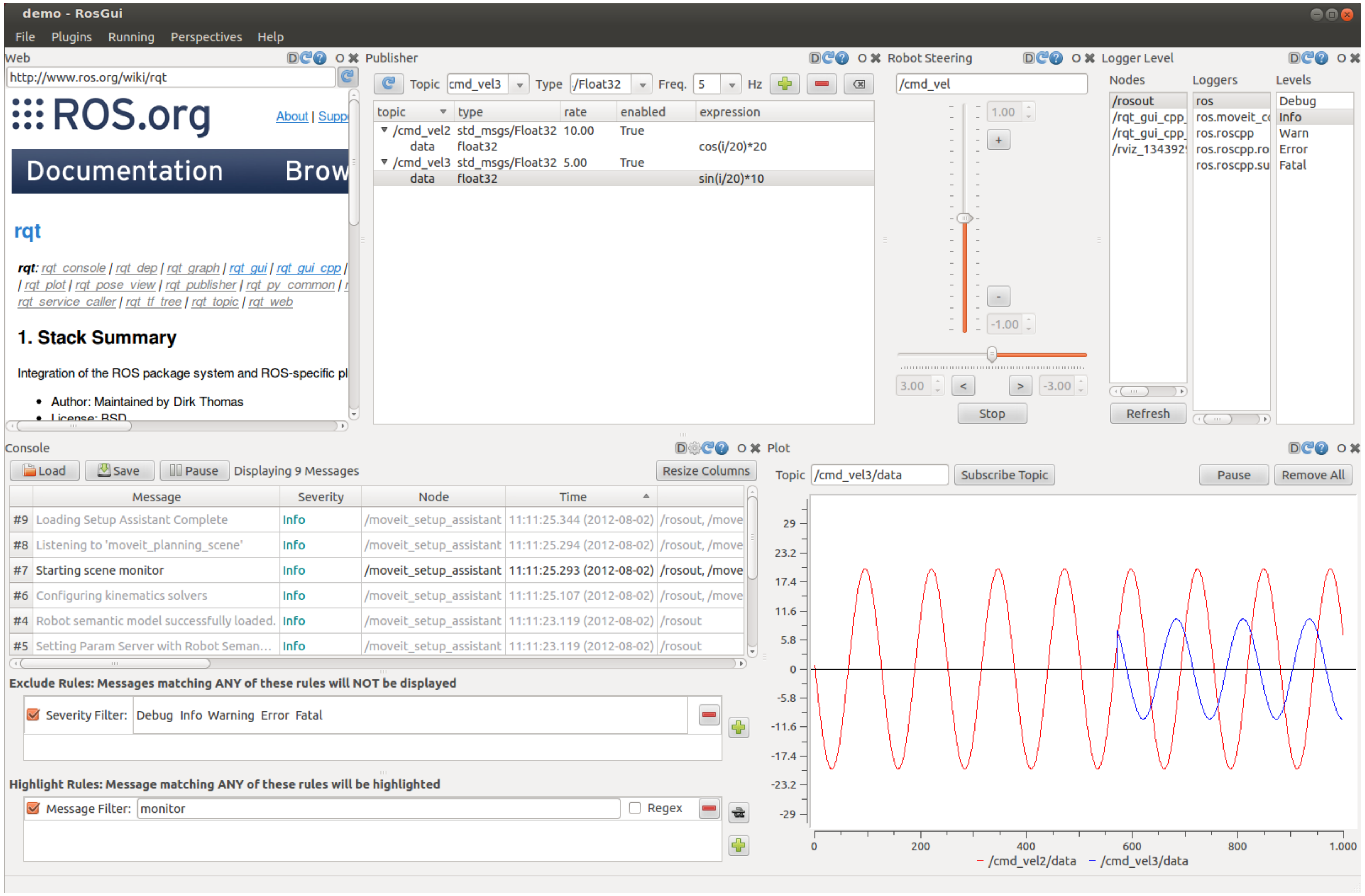
TF坐标变换如何实现?

- 广播TF变换
- 监听TF变换


机器人系统中繁杂的坐标系

# 3. ROS的开发工具 —— Qt工具箱

古月居 GYH.AI



http://www.ros.org/wiki/rqt

ROS.org About | Supp

Documentation Brow

**rqt**

*rqt*: rqt_console | rqt_dep | rqt_graph | rqt_gui | rqt_gui_cpp | | rqt_plot | rqt_pose_view | rqt_publisher | rqt_py_common | r rqt_service_caller | rqt_tf_tree | rqt_topic | rqt_web

**1. Stack Summary**

Integration of the ROS package system and ROS-specific pl

- Author: Maintained by Dirk Thomas
- License: BSD

Topic: cmd_vel3 Type: /Float32 Freq.: 5 Hz

| topic | type | rate | enabled | expression |
|---|---|---|---|---|
| ▼ /cmd_vel2 | std_msgs/Float32 | 10.00 | True | |
| data | float32 | | | cos(i/20)*20 |
| ▼ /cmd_vel3 | std_msgs/Float32 | 5.00 | True | |
| data | float32 | | | sin(i/20)*10 |

/cmd_vel

1.00 -1.00 3.00 -3.00

| Nodes | Loggers | Levels |
|---|---|---|
| /rosout | ros | Debug |
| /rqt_gui_cpp_ | ros.moveit_co | Info |
| /rqt_gui_cpp_ | ros.roscpp | Warn |
| /rviz_134392 | ros.roscpp.ro | Error |
| | ros.roscpp.su | Fatal |

Displaying 9 Messages

| | Message | Severity | Node | Time | |
|---|---|---|---|---|---|
| #9 | Loading Setup Assistant Complete | Info | /moveit_setup_assistant | 11:11:25.344 (2012-08-02) | /rosout, /move |
| #8 | Listening to 'moveit_planning_scene' | Info | /moveit_setup_assistant | 11:11:25.294 (2012-08-02) | /rosout, /move |
| #7 | Starting scene monitor | Info | /moveit_setup_assistant | 11:11:25.293 (2012-08-02) | /rosout, /move |
| #6 | Configuring kinematics solvers | Info | /moveit_setup_assistant | 11:11:25.107 (2012-08-02) | /rosout, /move |
| #4 | Robot semantic model successfully loaded. | Info | /moveit_setup_assistant | 11:11:23.119 (2012-08-02) | /rosout |
| #5 | Setting Param Server with Robot Seman... | Info | /moveit_setup_assistant | 11:11:23.119 (2012-08-02) | /rosout |

**Exclude Rules: Messages matching ANY of these rules will NOT be displayed**

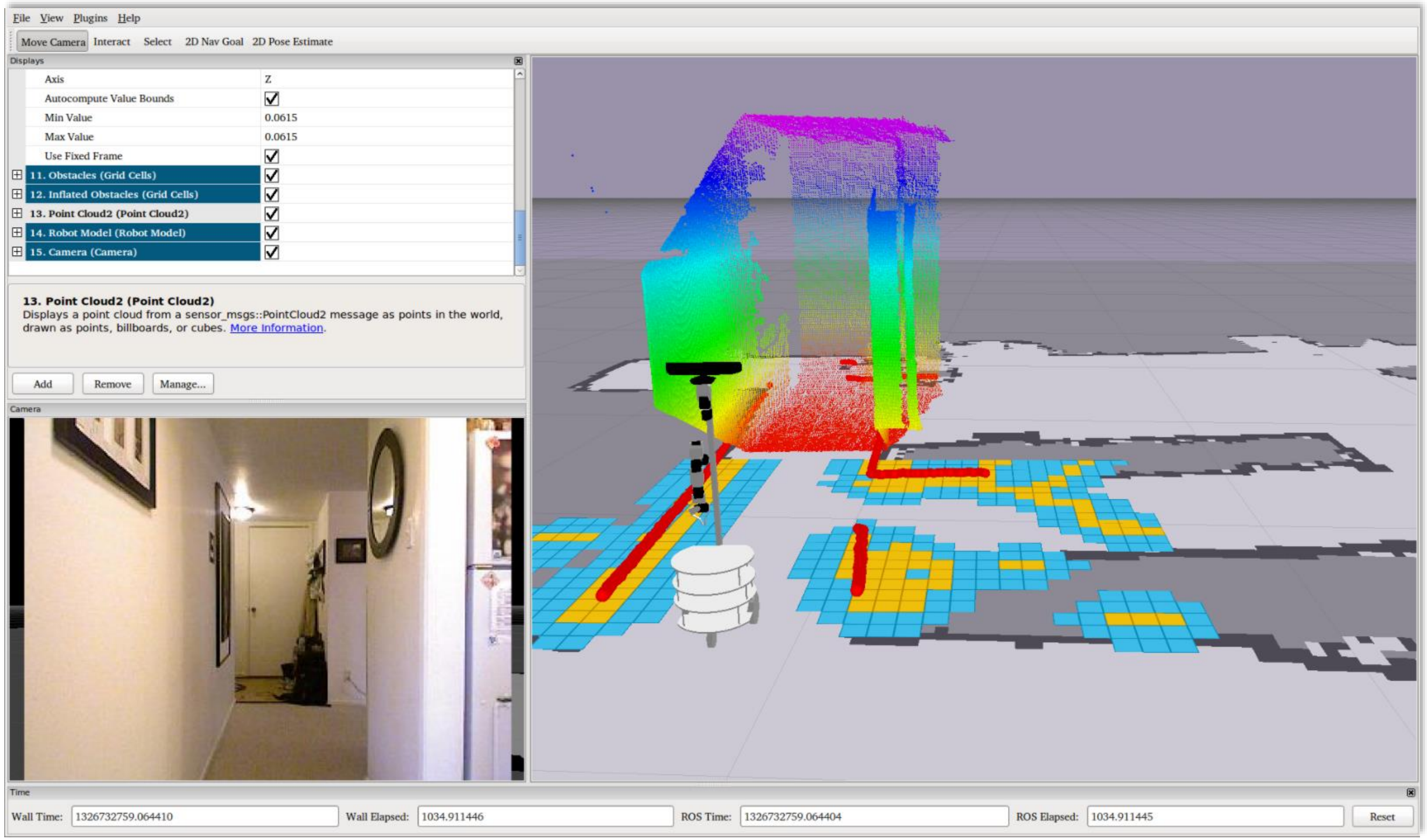
Severity Filter: Debug Info Warning Error Fatal

**Highlight Rules: Message matching ANY of these rules will be highlighted**

Message Filter: monitor

Topic /cmd_vel3/data

# 3. ROS的开发工具 —— Rviz



| | |
|---|---|
| Axis | Z |
| Autocompute Value Bounds | ✓ |
| Min Value | 0.0615 |
| Max Value | 0.0615 |
| Use Fixed Frame | ✓ |
| 11. Obstacles (Grid Cells) | ✓ |
| 12. Inflated Obstacles (Grid Cells) | ✓ |
| 13. Point Cloud2 (Point Cloud2) | ✓ |
| 14. Robot Model (Robot Model) | ✓ |
| 15. Camera (Camera) | ✓ |

**13. Point Cloud2 (Point Cloud2)**
Displays a point cloud from a sensor_msgs::PointCloud2 message as points in the world, drawn as points, billboards, or cubes. More Information.

Wall Time: 1326732759.064410 Wall Elapsed: 1034.911446 ROS Time: 1326732759.064404 ROS Elapsed: 1034.911445

**Rviz**是一款三维可视化工具，可以很好的兼容基于ROS软件框架的机器人平台。

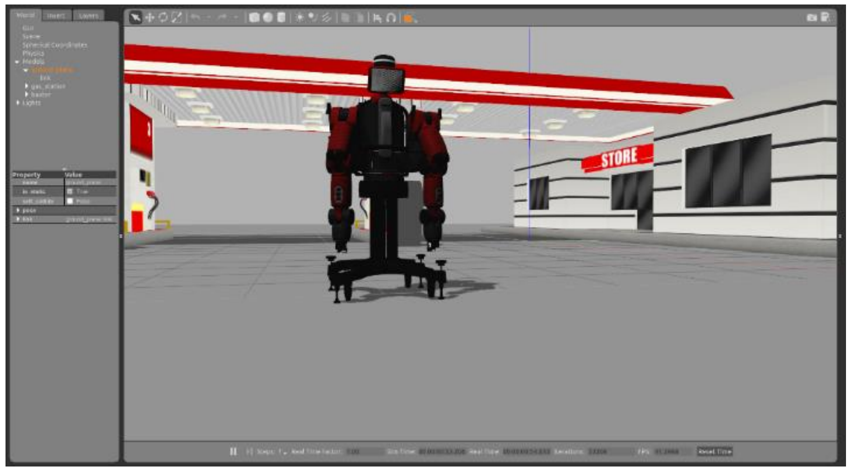
## 3. ROS的开发工具 —— Gazebo

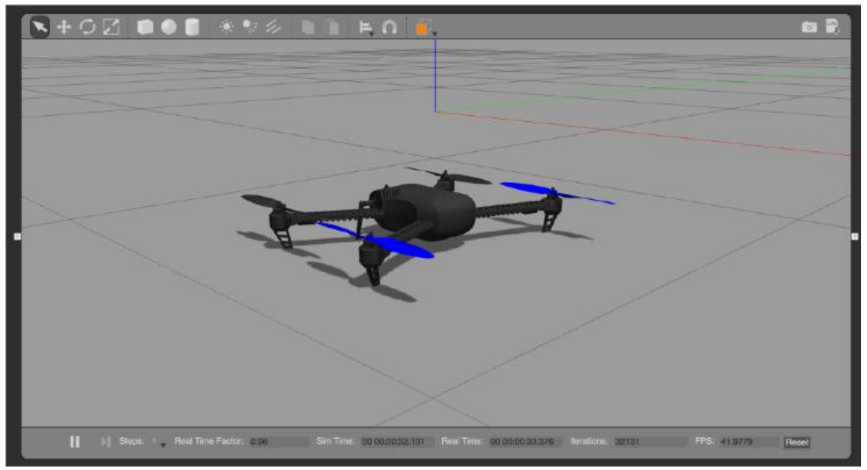
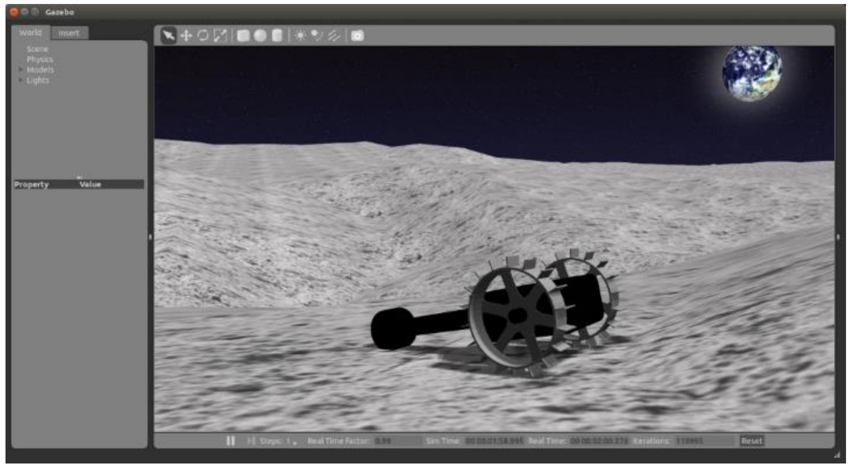
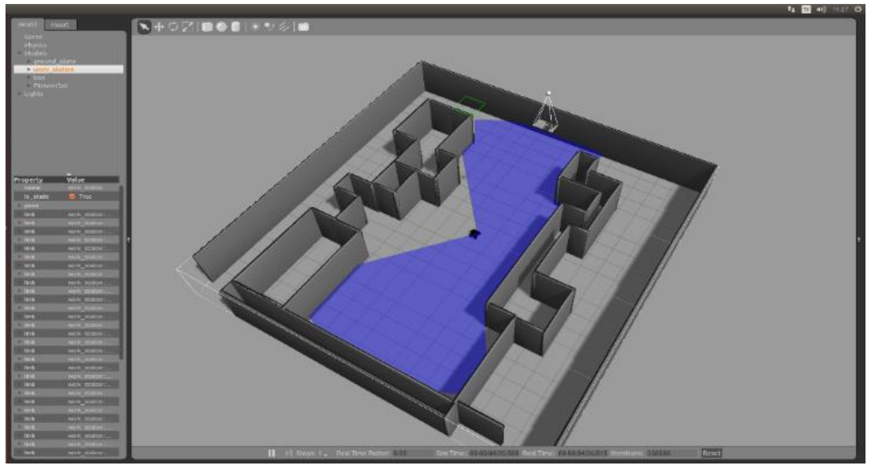
**Gazebo是一款功能强大的三维物理仿真平台**

- 具备强大的物理引擎
- 高质量的图形渲染
- 方便的编程与图形接口
- 开源免费

其典型应用场景包括

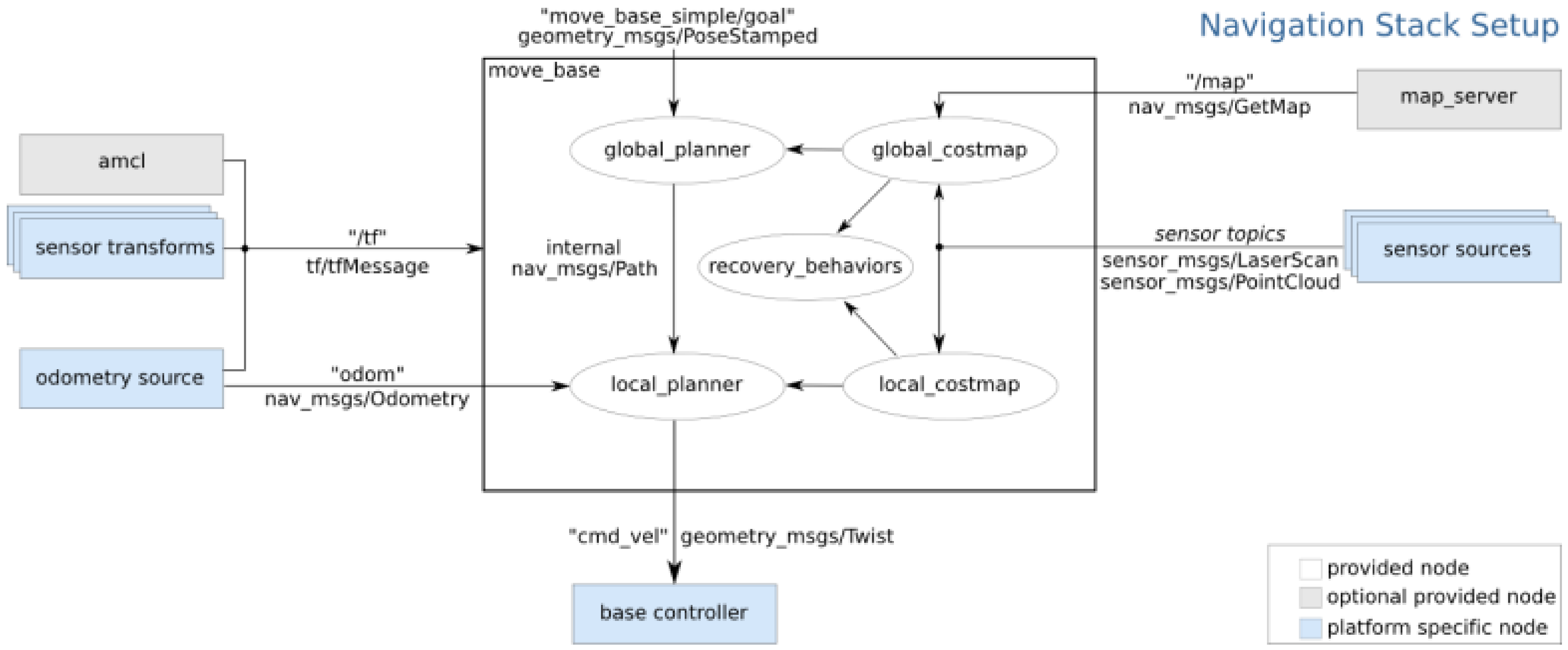
- 测试机器人算法
- 机器人的设计
- 现实情景下的回溯测试



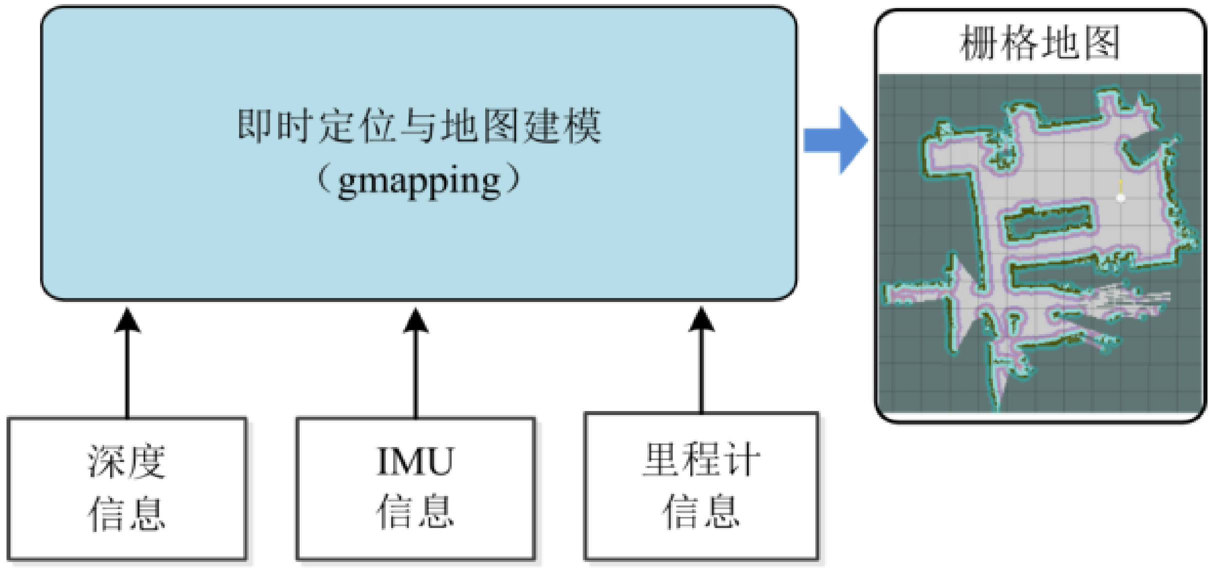




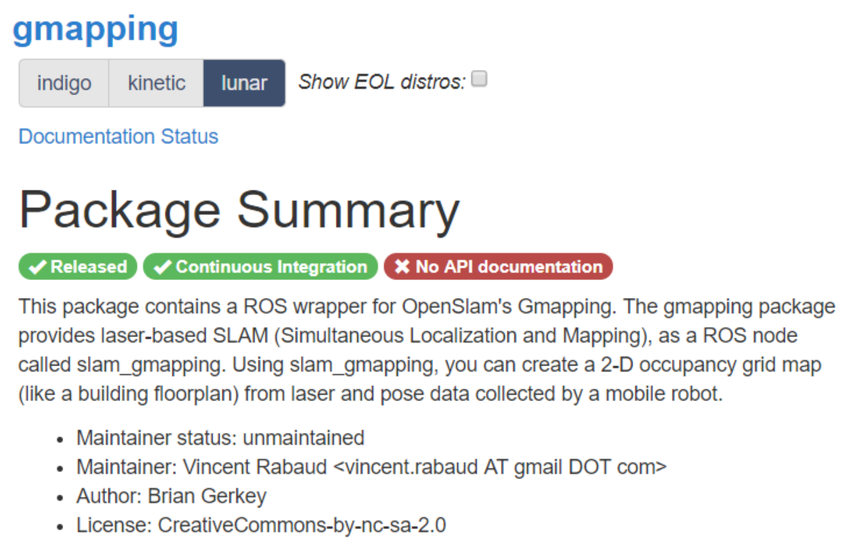
## 4. ROS的应用功能

# 4. ROS的应用功能 —— 导航框架



# 4. ROS的应用功能 —— SLAM





## gmapping

indigo | kinetic | lunar — *Show EOL distros:* ☐

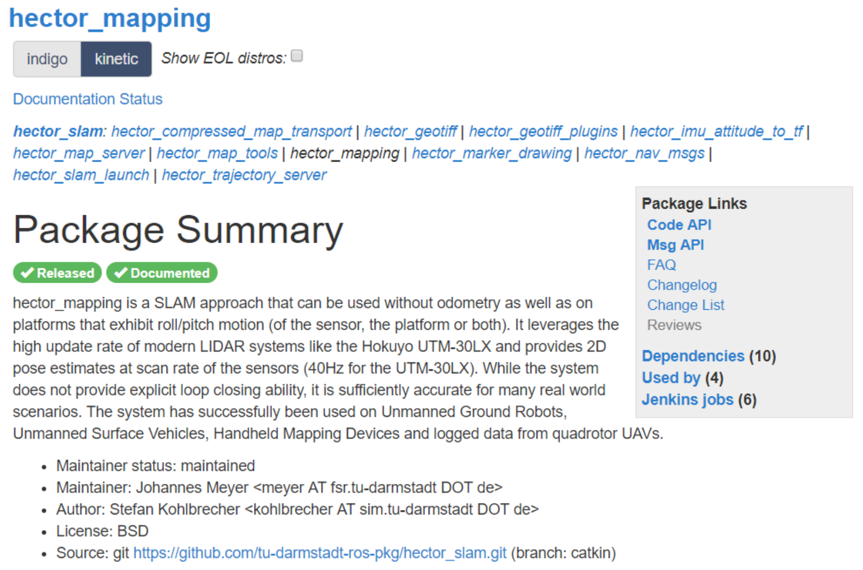
Documentation Status

### Package Summary

✔ Released ✔ Continuous Integration ✖ No API documentation

This package contains a ROS wrapper for OpenSlam's Gmapping. The gmapping package provides laser-based SLAM (Simultaneous Localization and Mapping), as a ROS node called slam_gmapping. Using slam_gmapping, you can create a 2-D occupancy grid map (like a building floorplan) from laser and pose data collected by a mobile robot.

- Maintainer status: unmaintained
- Maintainer: Vincent Rabaud <vincent.rabaud AT gmail DOT com>
- Author: Brian Gerkey
- License: CreativeCommons-by-nc-sa-2.0



## hector_mapping

indigo | kinetic — *Show EOL distros:* ☐

Documentation Status

***hector_slam**: hector_compressed_map_transport | hector_geotiff | hector_geotiff_plugins | hector_imu_attitude_to_tf | hector_map_server | hector_map_tools | hector_mapping | hector_marker_drawing | hector_nav_msgs | hector_slam_launch | hector_trajectory_server*

### Package Summary

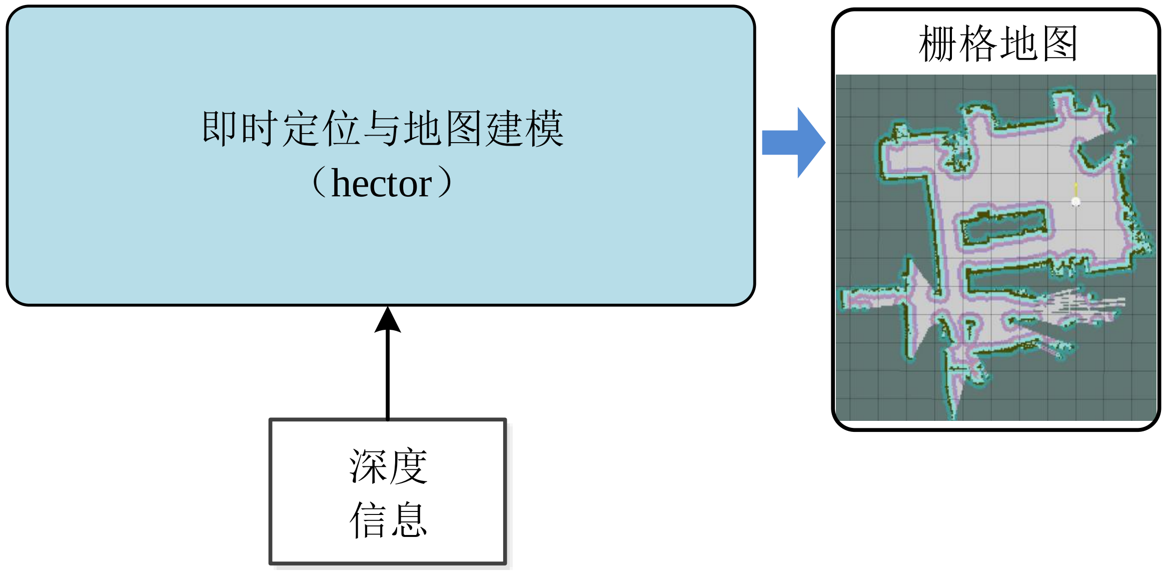
✔ Released ✔ Documented

hector_mapping is a SLAM approach that can be used without odometry as well as on platforms that exhibit roll/pitch motion (of the sensor, the platform or both). It leverages the high update rate of modern LIDAR systems like the Hokuyo UTM-30LX and provides 2D pose estimates at scan rate of the sensors (40Hz for the UTM-30LX). While the system does not provide explicit loop closing ability, it is sufficiently accurate for many real world scenarios. The system has successfully been used on Unmanned Ground Robots, Unmanned Surface Vehicles, Handheld Mapping Devices and logged data from quadrotor UAVs.

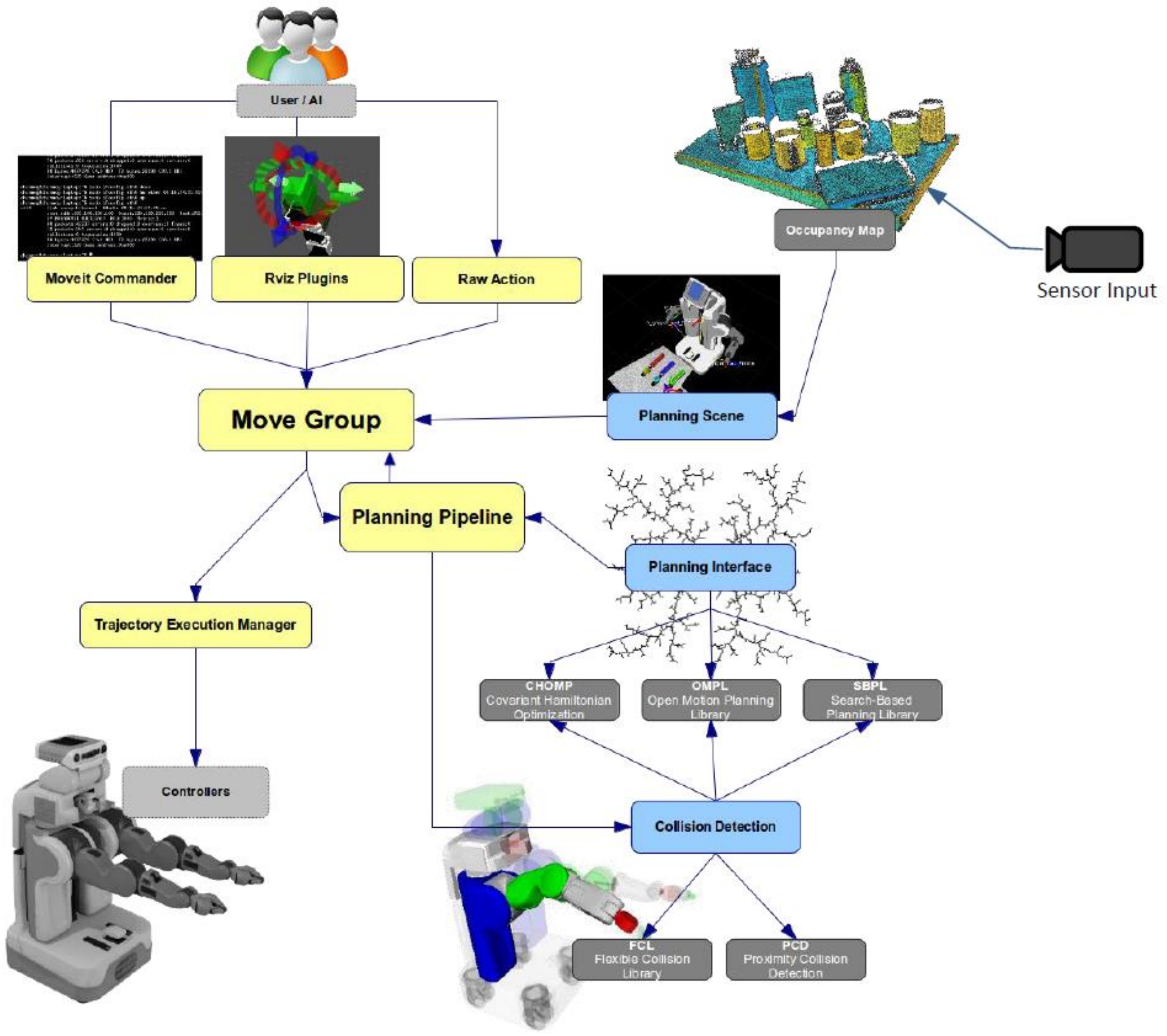
- Maintainer status: maintained
- Maintainer: Johannes Meyer <meyer AT fsr.tu-darmstadt DOT de>
- Author: Stefan Kohlbrecher <kohlbrecher AT sim.tu-darmstadt DOT de>
- License: BSD
- Source: git https://github.com/tu-darmstadt-ros-pkg/hector_slam.git (branch: catkin)

**Package Links**
- Code API
- Msg API
- FAQ
- Changelog
- Change List
- Reviews

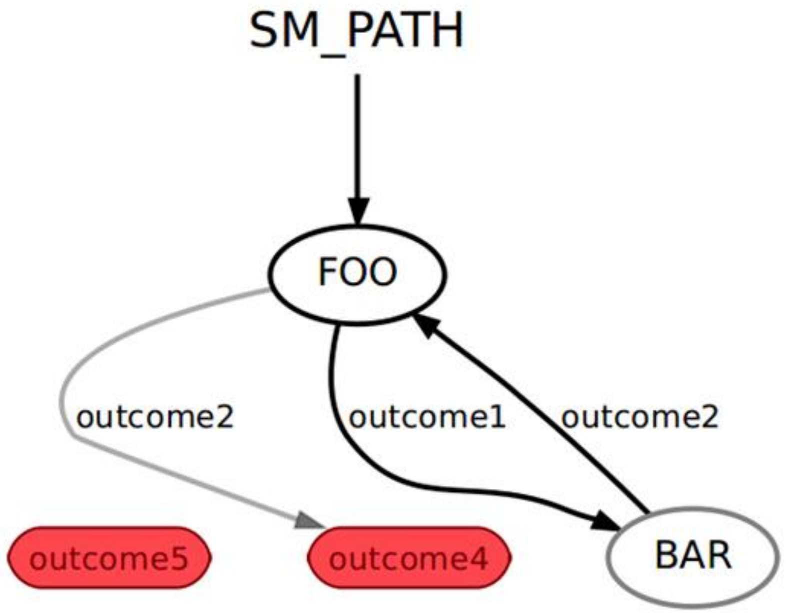
**Dependencies** (10)
**Used by** (4)
**Jenkins jobs** (6)

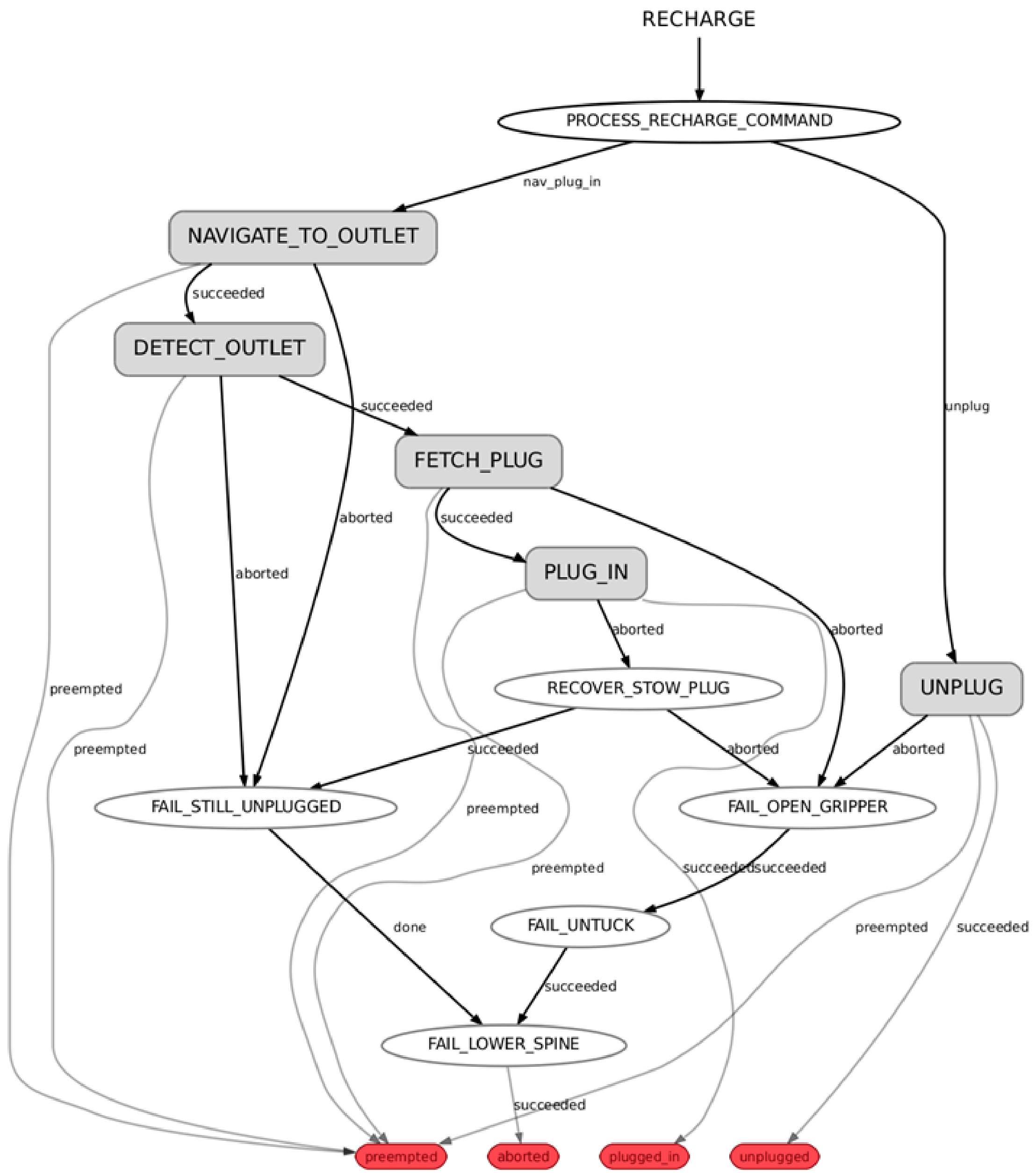


## 4. ROS的应用功能 —— MoveIt!



## 4. ROS的应用功能 —— SMACH任务级状态机





## 5. ROS的生态系统

## 5. ROS的生态系统

1. **发行版（Distribution）**：ROS发行版包括一系列带有版本号、可以直接安装的功能包。

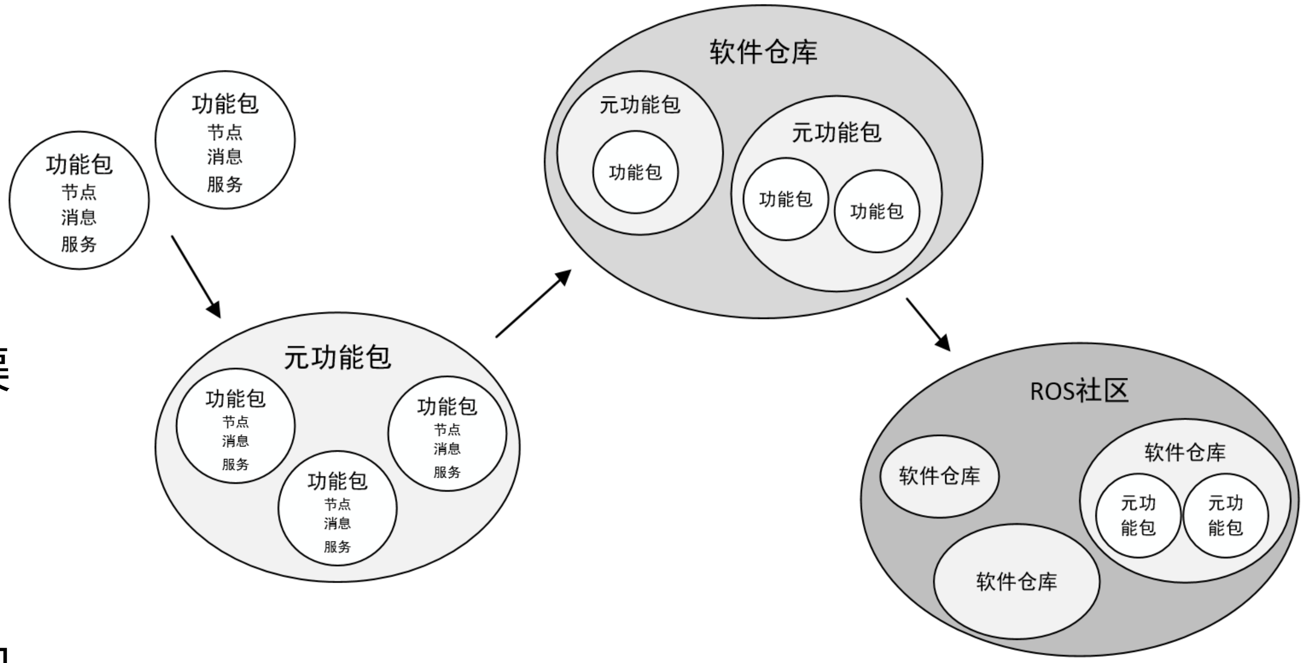
2. **软件源（Repository）**：ROS依赖于共享网络上的开源代码，不同的组织机构可以开发或者共享自己的机器人软件。

3. **ROS wiki**：记录ROS信息文档的主要论坛。

4. **邮件列表（Mailing list）**：交流ROS更新的主要渠道，同时也可以交流ROS开发的各种疑问。

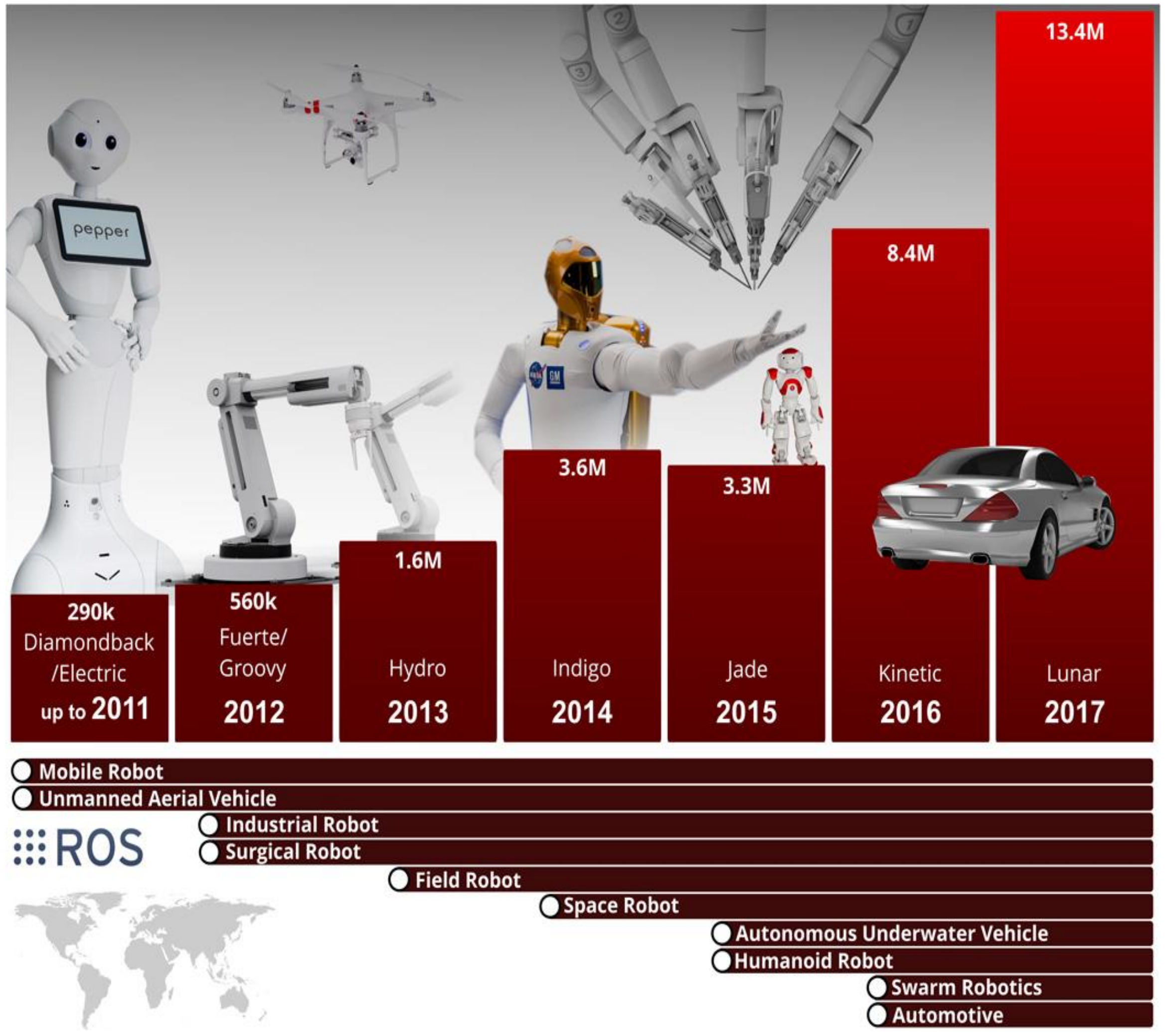
5. **ROS Answers**：咨询ROS相关问题的网站。

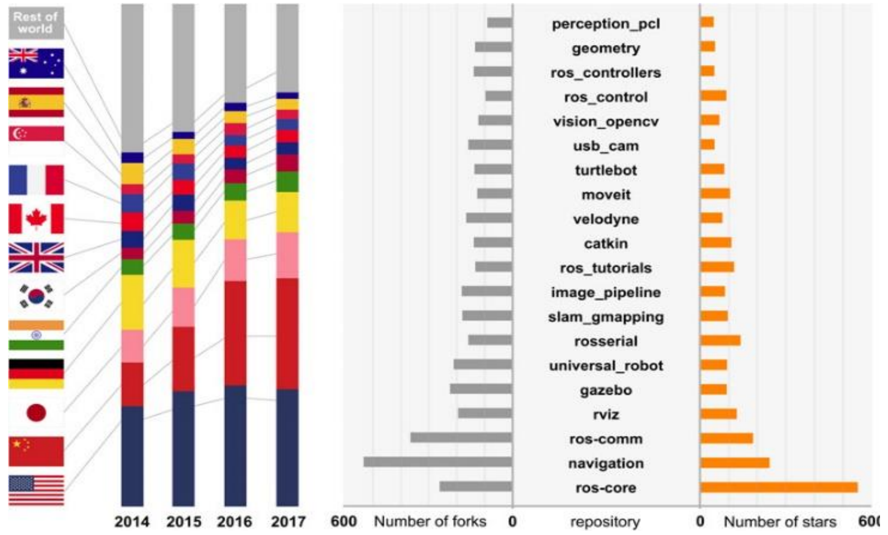
6. **博客（Blog）**：发布ROS社区中的新闻、图片、视频（http://www.ros.org/news）

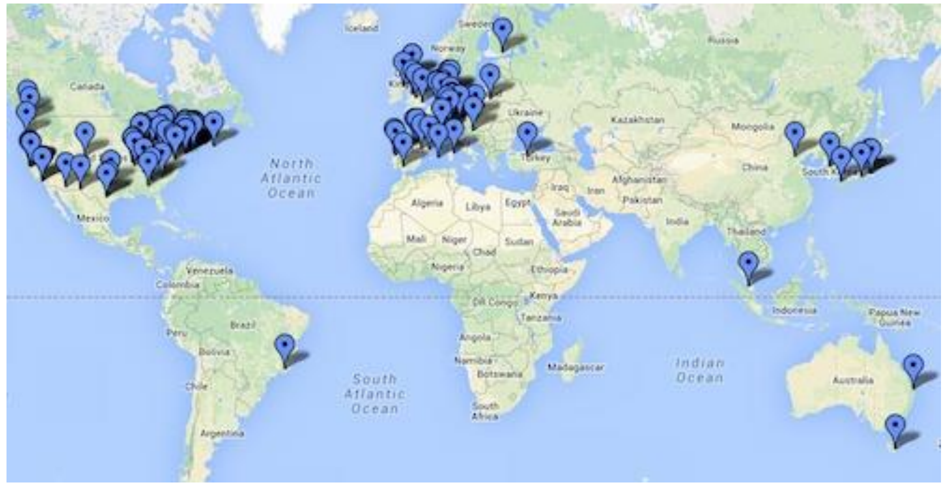


**ROS社区资源的组织形式**

# 5. ROS的生态系统



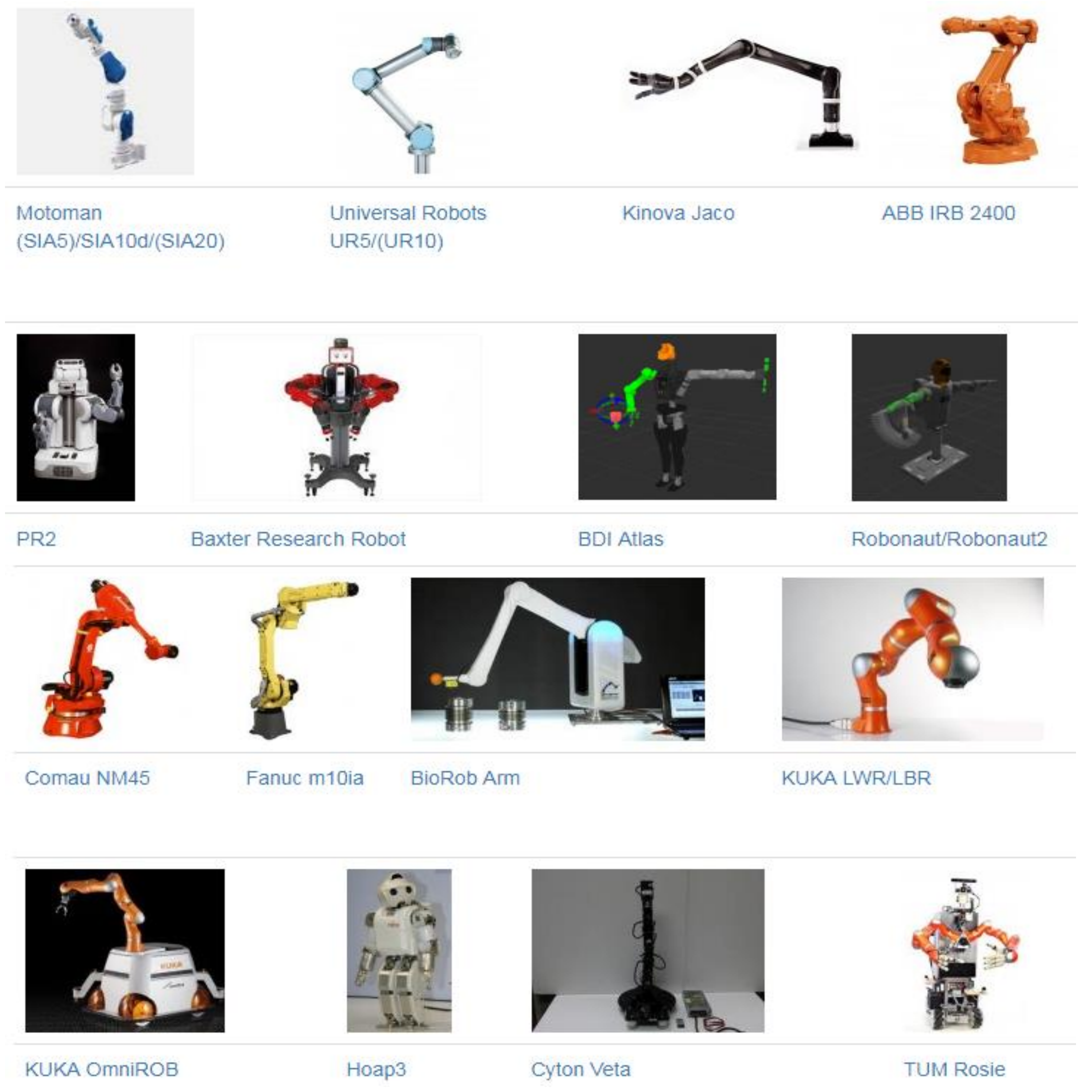


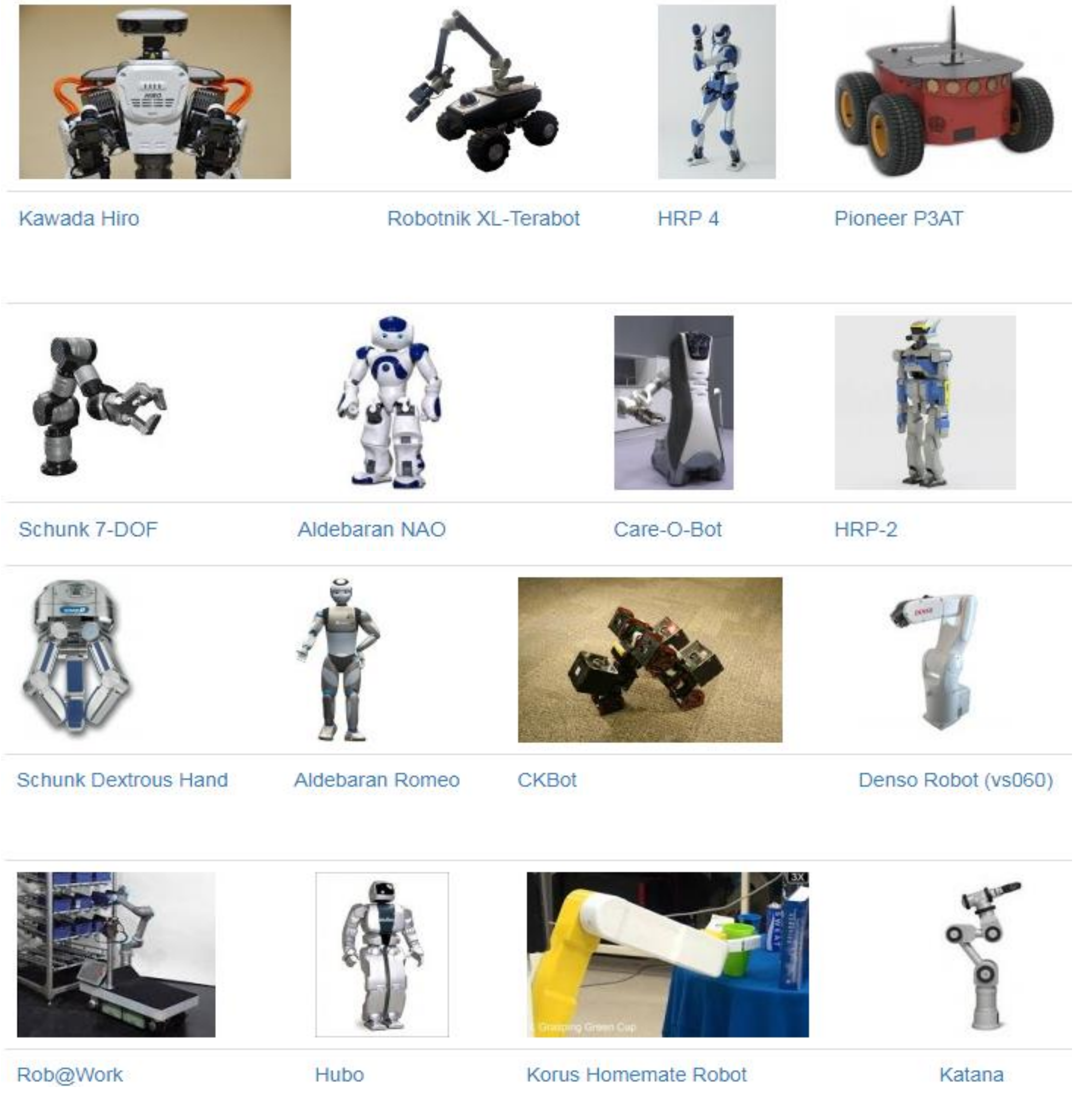


**ROS社区内的功能包数量、关注度、相关文章均呈指数级上涨**

（来源：http://robotics.sciencemag.org/content/2/11/eaar1868）

## 5. ROS的生态系统





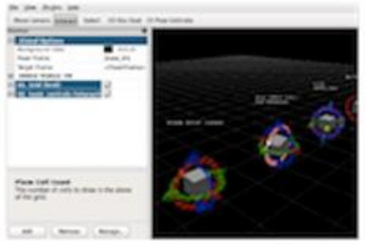
**Powering the world's robots —— 机器人领域的事实标准**

## 本讲小结

=

通信机制

+



开发工具

+

应用功能

+

生态系统

**ROS 是什么**

| | |
|---|---|
| **通信机制** | 松耦合分布式通信：节点、管理器、话题、服务 … |
| **开发工具** | 命令行、Launch、TF、Qt工具箱、Rviz、Gazebo … |
| **应用功能** | Navigation、SLAM、MoveIt! … |
| **生态系统** | 发行版、软件源、wiki、ROS Answers … |

## 本讲作业

1. 练习ROS Tutorials教程
2. 建立工作空间，并编译本讲代码
3. 运行本讲代码中的话题、服务、动作、TF例程
4. 对照ROS wiki，熟悉代码实现原理

## 扩展阅读

- "Powering the world’ s robots" 的ROS是什么?
  https://mp.weixin.qq.com/s/f9QZLfMWD3TbxRH85xAqXA
- 如何学习ROS：
  https://mp.weixin.qq.com/s/Yuku2YGlDFKnFzLki3f7Wg
- ROS Concepts：
  http://wiki.ros.org/ROS/Concepts
- ROS Tutorials
  http://wiki.ros.org/ROS/Tutorials

# Thank You

更多精彩，欢迎关注

古月居

古月春旭