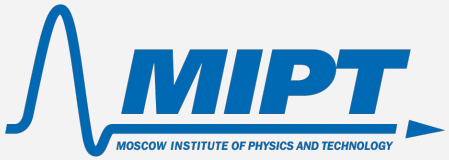
Моделирование колесных роботов

Лекция 11. Имена и пространства имен в ROS, время, инструменты отладки и симуляции

Николай Жердев

ИППИ РАН



Москва, 2023

СОДЕРЖАНИЕ ЛЕКЦИИ

1. Имена и пространства имен в ROS
2. Время
3. Дополнительные инструменты
    a. tf
    b. .bag файлы
    c. Rqt
    d. Rviz
    e. Gazebo

ИМЕНА В ROS

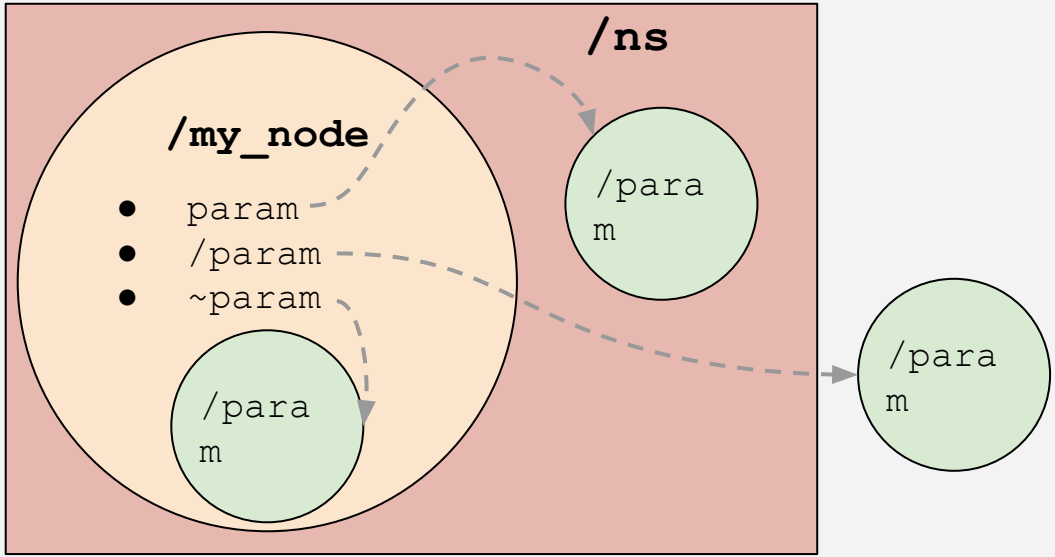
http://wiki.ros.org/Names

- В ROS имена есть у:
  - Нодов
  - Топиков
  - Параметров
  - Сервисов
  - ...
- Имена и пространства имен позволяют инкапсулировать информацию
- Валидные имена отвечают следующим правилам:
  - Первый символ: ([a-z|A-Z]), тильда (~) или прямой слеш (/)
  - Последующие символ: ([0-9|a-z|A-Z]), подчеркивания (_), или прямые слешы (/)

ИМЕНА В ROS

http://wiki.ros.org/Names

- Имя может иметь один из 4 типов:
  - relative/name
    - base
  - /global/name
  - ~private/name
- Правила преобразования имен на примере нода и параметра:



Имя нода	Относительное имя	Глобальное имя	Приватное имя
/my_node	param -> /param	/param -> /param	~param -> /my_node/param
/ns/my_node	param -> /ns/param	/param -> /param	~param -> /ns/my_node/param
	param_ns/param -> /ns/param_ns/param	/param_ns/param -> /param_ns/param	~param_ns/param -> /ns/my_node/param_ns/param

ВРЕМЯ В ROS

http://wiki.ros.org/rospy/Overview/Time

- UNIX-время используется в ROS в качестве временных меток
  - **UNIX-время** — целое число, увеличивающееся каждую секунду и равное количеству секунд, прошедших с 00:00:00 UTC 1 января 1970 года
- Клиентские библиотеки (rospy, roscpp, ...) предоставляют API для работы со временем:
  - Базовые классы `Time` и `Duration, Timer` с поддержкой арифметических операций
  - Функции для получения системного времени
  - Функции `rospy.sleep()` и `rospy.Rate.sleep()`

```
now = rospy.get_rostime() # эквивалентно now = rospy.Time.now()
rospy.loginfo("Current time %i %i", now.secs, now.nsecs)
```

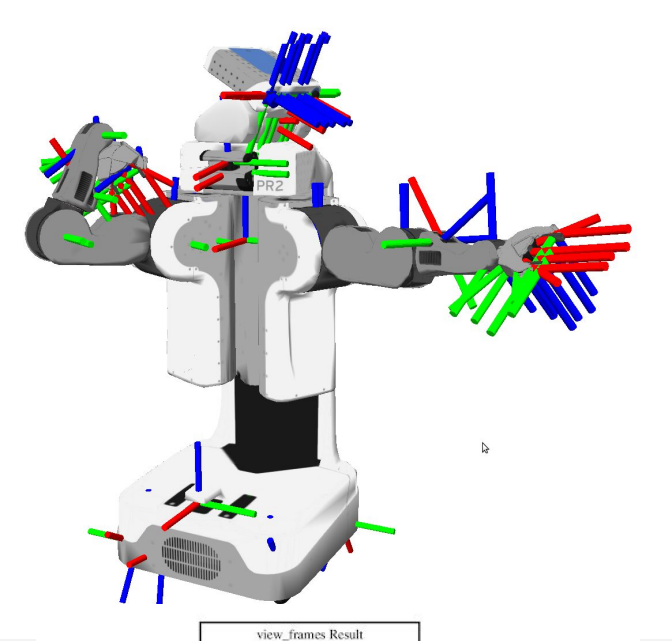
```
two_hours = rospy.Duration(60*60) + rospy.Duration(60*60)
one_hour = rospy.Duration(2*60*60) - rospy.Duration(60*60)
tomorrow = rospy.Time.now() + rospy.Duration(24*60*60)
negative_one_day = rospy.Time.now() - tomorrow
```

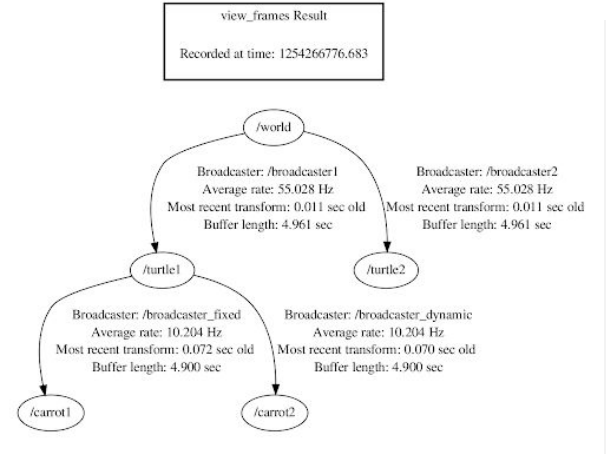
```
# sleep for 10 seconds
rospy.sleep(10.)
# sleep for duration
d = rospy.Duration(10, 0)
rospy.sleep(d)
```

Tf

http://wiki.ros.org/tf
http://wiki.ros.org/tf2

- Пакеты tf и tf2:
  - Хранят преобразования между системами координат робота в виде дерева буферизованного во времени
  - Предоставляют интерфейс для преобразования точек векторов и т.д. между любыми системами координат
  - Позволяют восстановить преобразования между системами координат в прошлом





ROSBAG

http://wiki.ros.org/rosbag
Что у тебя в сумке? Как работать с bag файлами и не утонуть в данных

- `.bag` файлы содержат **сериализованные** ROS-сообщения.
- `.bag` файлы могут быть проиграны в виде топиков, которые в них записаны.
- Формат `.bag` файлов **эффективен** как **для записи**, так и **для проигрывания**, так как сообщения хранятся в том же формате, что и при передаче по сети внутри ROS.

КАК ЗАПИСАТЬ .BAG ФАЙЛ

```
rosbag record <topic-names>
```

`-d, --duration`

Максимальная продолжительность файла.
$ rosbag record --duration=30 /chatter
$ rosbag record --duration=5m /chatter
$ rosbag record --duration=2h /chatter

`--split`

Разделение файла когда достигнута максимальная длина/продолжительность
$ rosbag record --split --size=1024 /chatter
$ rosbag record --split --duration=30 /chatter
$ rosbag record --split --duration=5m /chatter
$ rosbag record --split --duration=2h /chatter

`--max-splits=MAX_SPLITS`

Разделить bag максимум **MAX_SPLITS** раз, после чего начать удалять устаревшие файлы.
$ rosbag record --split --size 1024 --max-splits 3 /chatter
$ rosbag record --split --duration 10m --max-splits 6 /chatter

`-b SIZE, --buffsize=SIZE`

Использовать внутренний буфер размера SIZE MB (Default: 256, 0 = бесокнечный). Создает очередь сообщений объекта recorder, которая заполняется до того как быть записанов в файл. Уменьшения размера буфера приведет к потере сообщений.
$ rosbag record -b 1024 /chatter

`--chunksize=SIZE`

Записывать блоки данных размера SIZE KB (Default: 768). Это размер буфера объекта bag файл.Уменьшение буфера приведет к более частой записи на диск.
$ rosbag record --chunksize=1024 /chatter

`-l NUM, --limit=NUM`

Записать только NUM сообщений из каждого топика.
$ rosbag record -l 1000 /chatter

`--node=NODE`

Записать все топики, на которые подписан NODE.
$ rosbag record --node=/joy_teleop

`-j, --bz2`

Применить компрессию BZ2.
$ rosbag record -j /chatter

`--lz4`

Применить компрессию LZ4.
$ rosbag record --lz4 /chatter

КАК ВОСПРОИЗВЕСТИ .BAG ФАЙЛ

`rosbag play <bag-files>` - считывает содержимое bag файлов и публикует их синхронизируя по времени

`-i, --immediate`
Проиграть все сообщения без задержки.
$ rosbag play -i recorded1.bag

`--pause`
Начать проигрывание в режиме “Пауза”.
$ rosbag play --pause recorded1.bag

`--queue=SIZE`
Размер очереди публикации сообщений SIZE (по умолчанию: 0).
$ rosbag play --queue=1000 recorded1.bag

`--clock`
Публиковать время.
$ rosbag play --clock recorded1.bag

`--hz=HZ`
Публиковать время с частотой HZ (по умолчанию: 100).
$ rosbag play --clock --hz=200 recorded1.bag

`-d SEC, --delay=SEC`
Делать паузу в проигрывании на SEC секунд после каждого нового объявления топика (чтобы подписчики имели время на подписку).
$ rosbag play -d 5 recorded1.bag

`-r FACTOR, --rate=FACTOR`
Умножить скорость воспроизведения на FACTOR.
$ rosbag play -r 10 recorded1.bag

`-s SEC, --start=SEC`
Начать воспроизведение с SEC секунды от начала файла.
$ rosbag play -s 5 recorded1.bag

`-u SEC, --duration=SEC`
Проиграть SEC секунд файла.
$ rosbag play -u 240 recorded1.bag

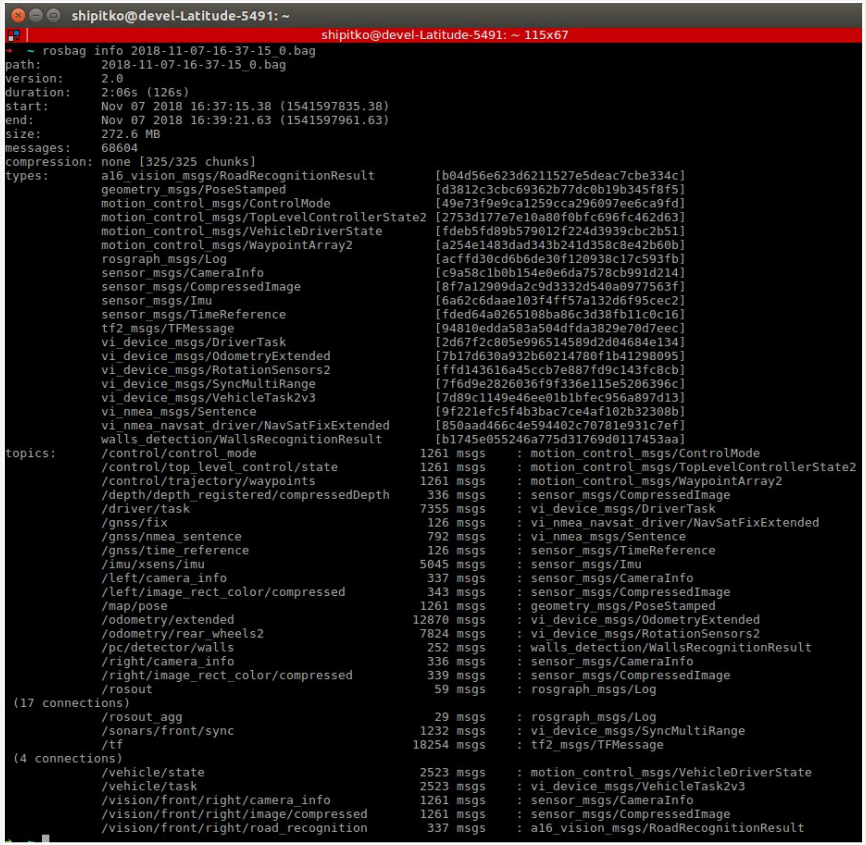
`--skip-empty=SEC`
Пропустить участки bag файла без сообщений длиной более SEC секунд.
$ rosbag play --skip-empty=1 recorded1.bag

`-l, --loop`
Зациклить воспроизведение.
$ rosbag play -l recorded1.bag

`-k, --keep-alive`
Не останавливать воспроизведение после проигрывания всех сообщений (полезно для публикации latched топиков).
$ rosbag play -k recorded1.bag

ПРОСМОТР СОДЕРЖИМОГО

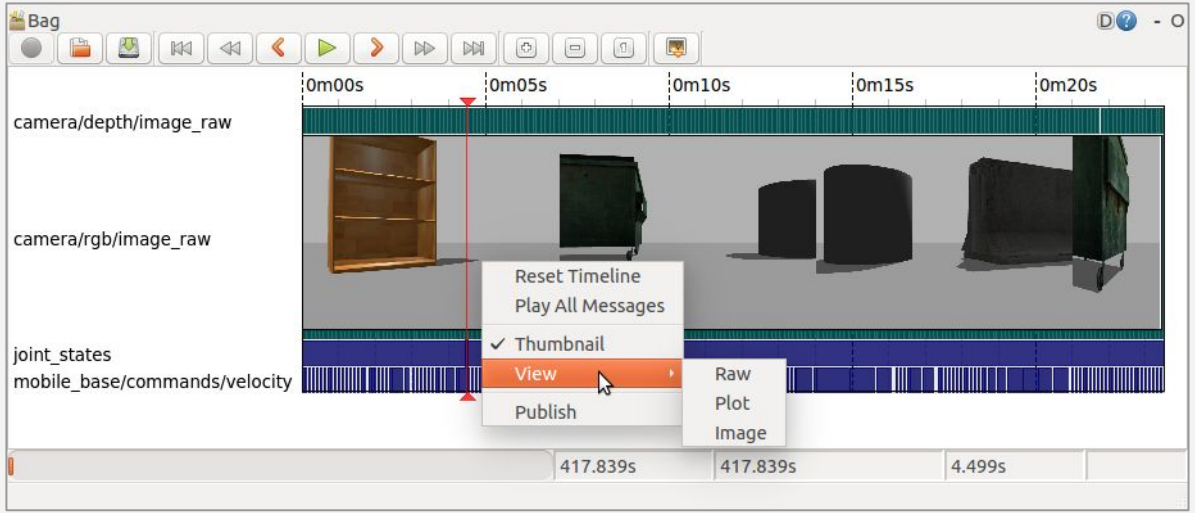
- `rosbag info <bag-files>`
- `rostopic list -b <bag-file>`
- `rostopic echo -b <bag-file>`



ПРОСМОТР СОДЕРЖИМОГО

Пакет `rqt_bag`

- Показывает наличие сообщений в топиках
- Показывает thumbnails изображения на временной шкале
- Позволяет строить графики числовых сообщений
- Публиковать / записывать выбранные топики
- Экспортировать сообщения из выбранного временного промежутка в новый bag



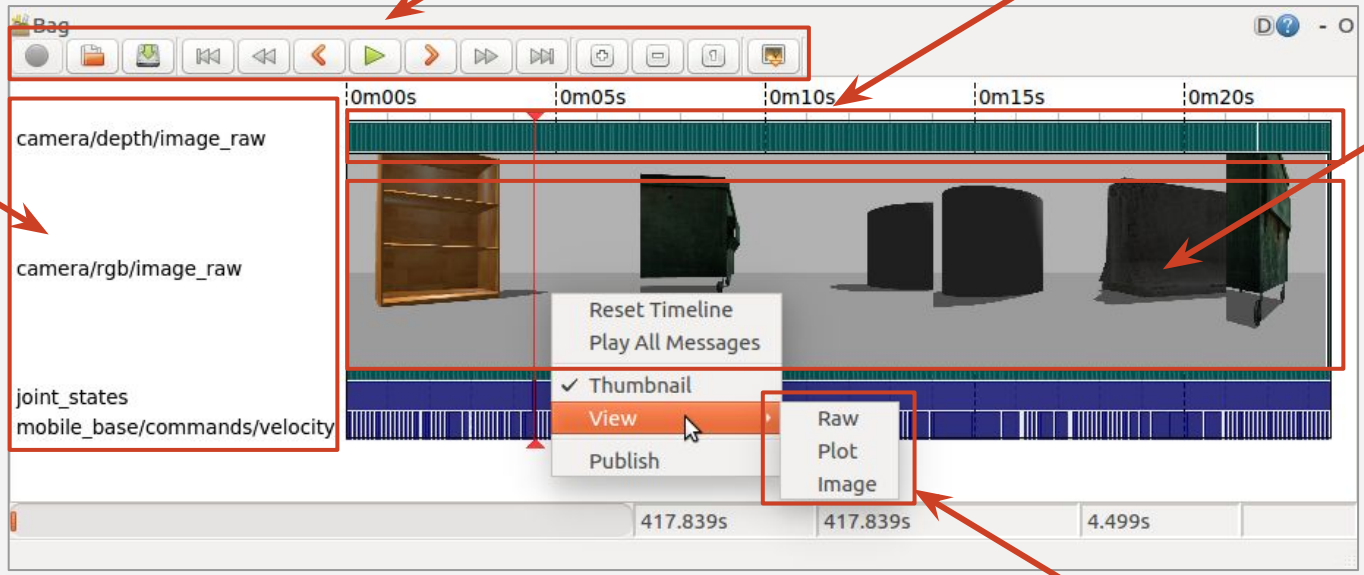
`rqt_bag` имеет API, позволяющий реализовывать свои плагины

Воспроизведение/запись/навигация

Индикация прихода сообщений в топик

Список топиков

Визуализация топиков с изображениями



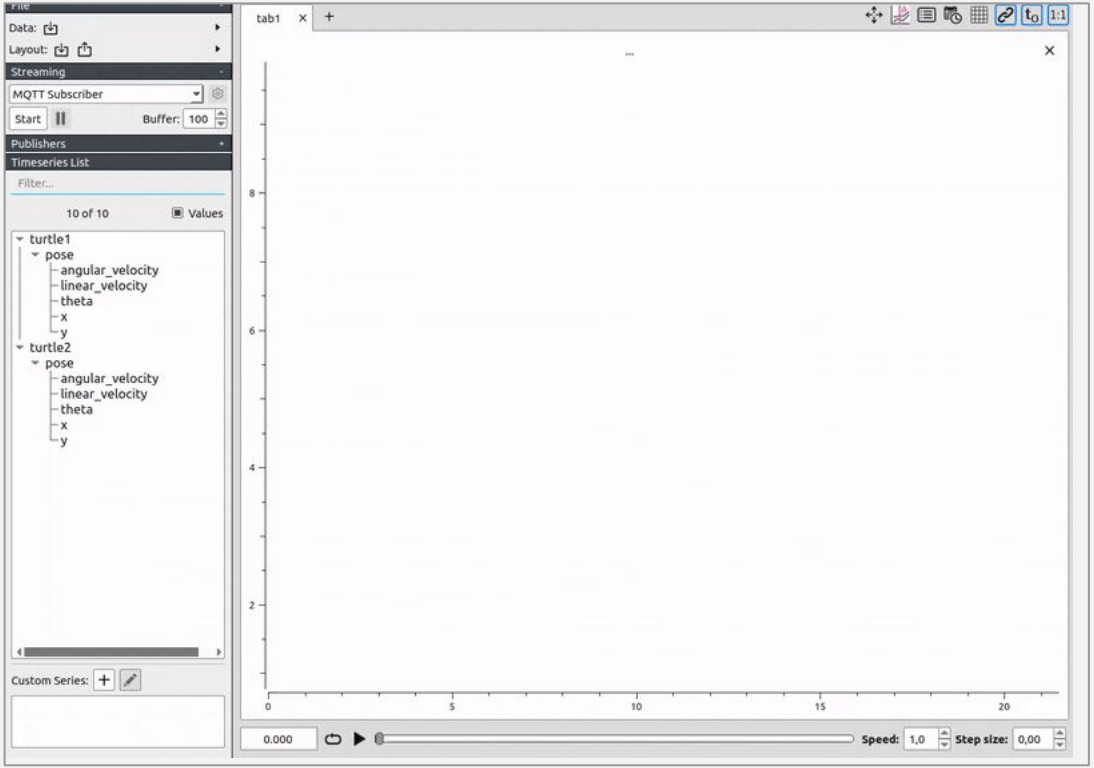
Выбор типа визуализации (сырые данные/график/изображение)

ПРОСМОТР СОДЕРЖИМОГО

https://github.com/facontidavide/PlotJuggler

Пакет `plotjuggler`

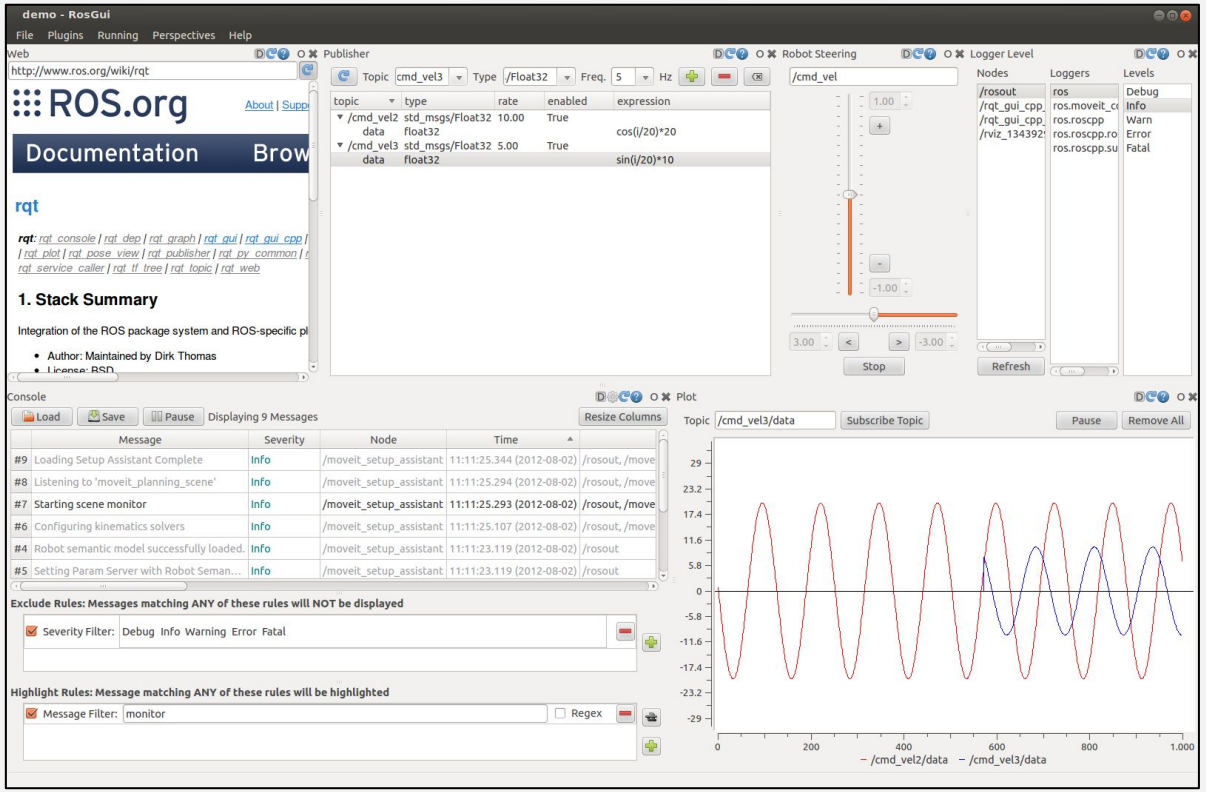
- `rqt_bag` на максималках
- Позволяет



RQT

http://wiki.ros.org/rqt

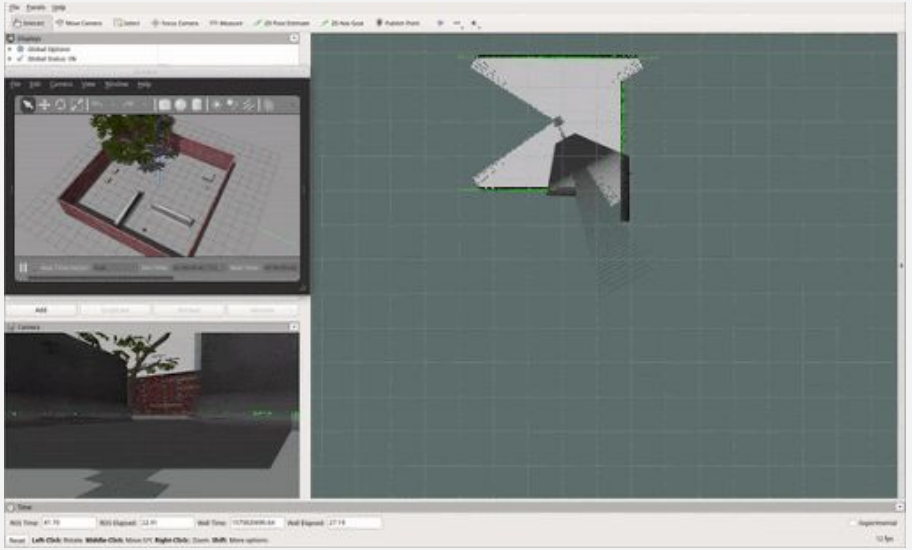
- Фреймворк для создания GUI ROS-приложений на основе Qt
- Содержит набор готовых плагинов
- Предоставляет API для написания своих плагинов

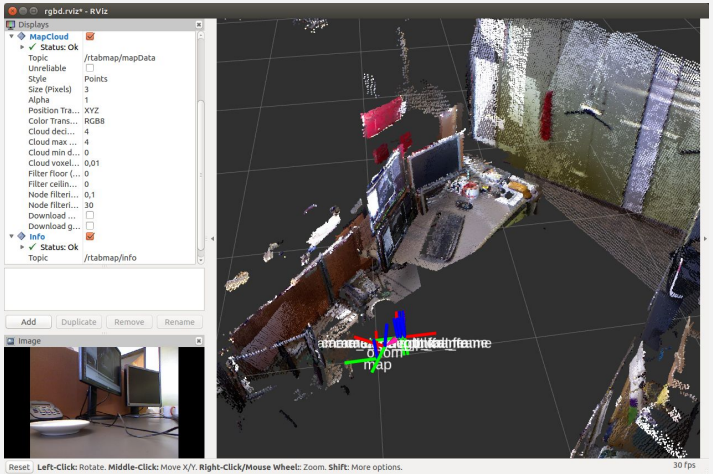


RVIZ

http://wiki.ros.org/rviz

- Инструмент 2D/3D визуализации в ROS
- Поддерживает визуализацию распространенных типов данных (карты занятости, лазерные сканы, облака точек, системы координат, траектория и др.), а также отрисовку простых геометрических примитивов (кубы, цилиндры, точки, линии и др.) и даже полноценных CAD-моделей
- Функционал может быть расширен пользовательскими плагинами

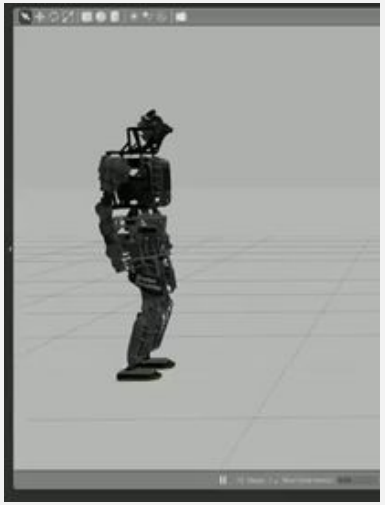


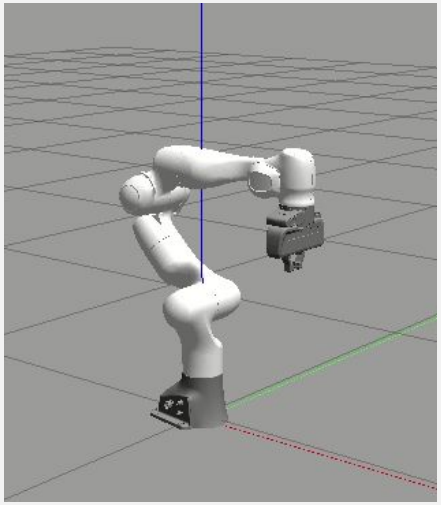


GAZEBO

http://wiki.ros.org/gazebo_ros_pkgs

- **Gazebo** — 3D симулятор твердых тел с открытым исходным кодом.
- Часто применяется в связке с ROS для моделирования роботов и используется для проведения робототехнических соревнований .
- Gazebo может:
  - Использовать различные физические движки: ODE, Bullet, и др.
  - Осуществлять реалистичный рендеринг включая различные источники освещения, тени, текстуры и т.д.
  - Моделировать сенсоры, включая их шумы измерения: LIDAR, камеры, камеры глубины



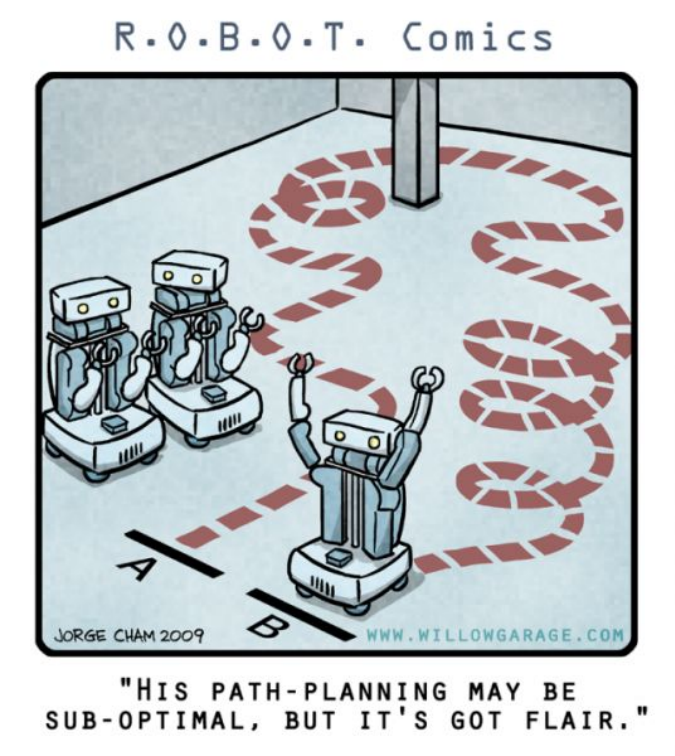


GAZEBO

ROS NAVIGATION STACK

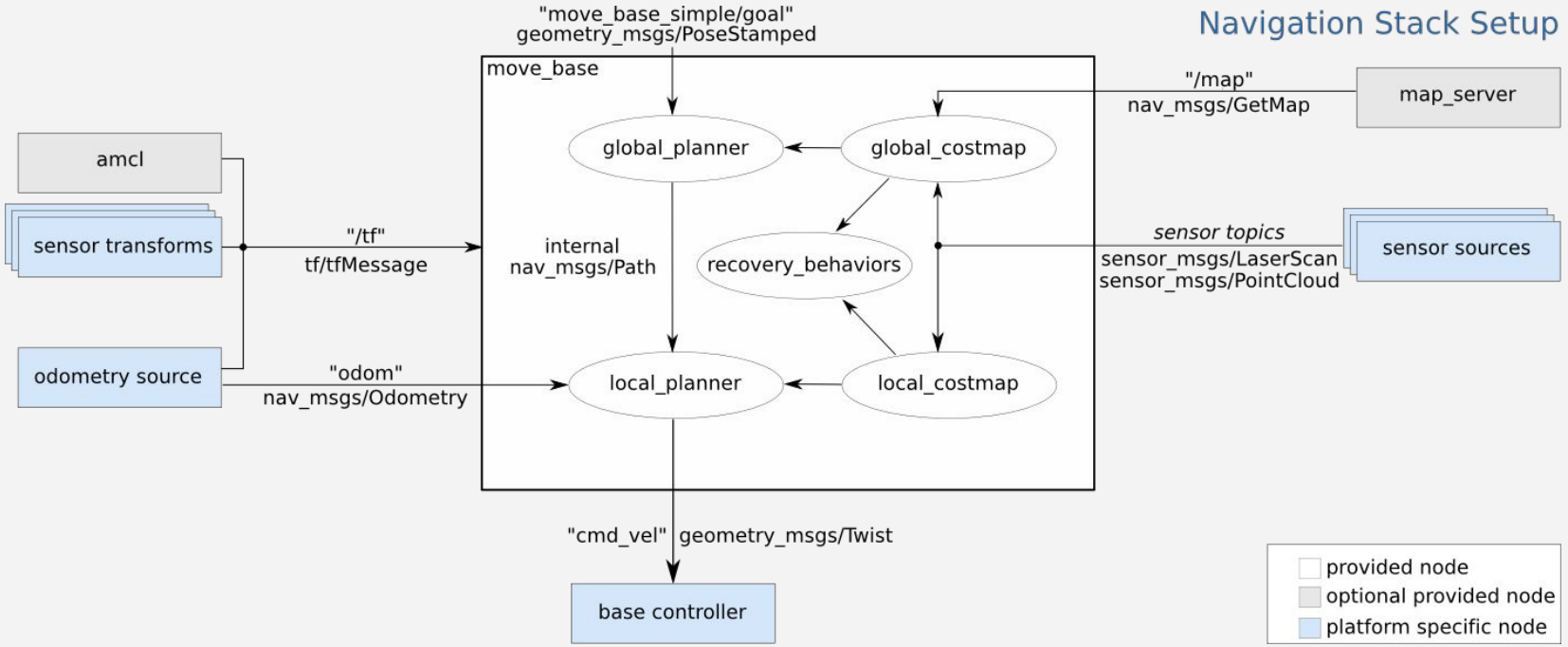
http://wiki.ros.org/navigation

- Стек 2D навигации принимает в качестве входов **одометерию**, **сенсорные данные**, и **целевое положение** и рассчитывает **управление по скорости** (линейной и угловой), позволяющее роботу достигнуть заданной точки.
- Ограничения стека навигации:
  - Предназначен для роботов **с дифференциальным приводом** или **голономных** роботов
  - Робот должен иметь планарный лазерный дальномер (или другой сенсор способный генерировать 2D сканы) для генерации карты и локализации
  - Подходит для роботов с квадратной/круглой базой. Для роботов других форм планирование пути может быть субоптимальным



ROS NAVIGATION STACK

http://wiki.ros.org/navigation



TURTLEBOT

http://wiki.ros.org/Robots/TurtleBot
http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/#turtlebot

Original TurtleBot
(Discontinued)

TurtleBot 2 Family

TurtleBot 2

TurtleBot 2i

TurtleBot 2e

TurtleBot 3 Family

Burger

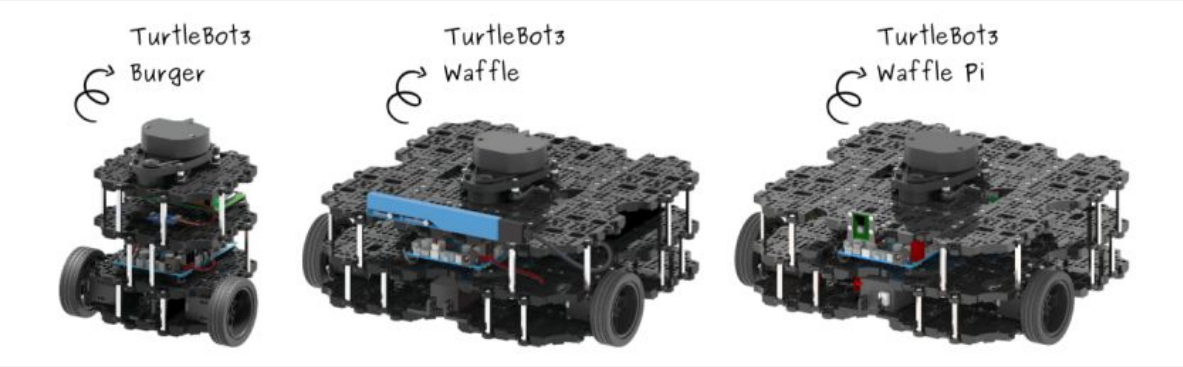
Waffle

Waffle Pi

TURTLEBOT SIMULATION. ПОДГОТОВКА

http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/#turtlebot

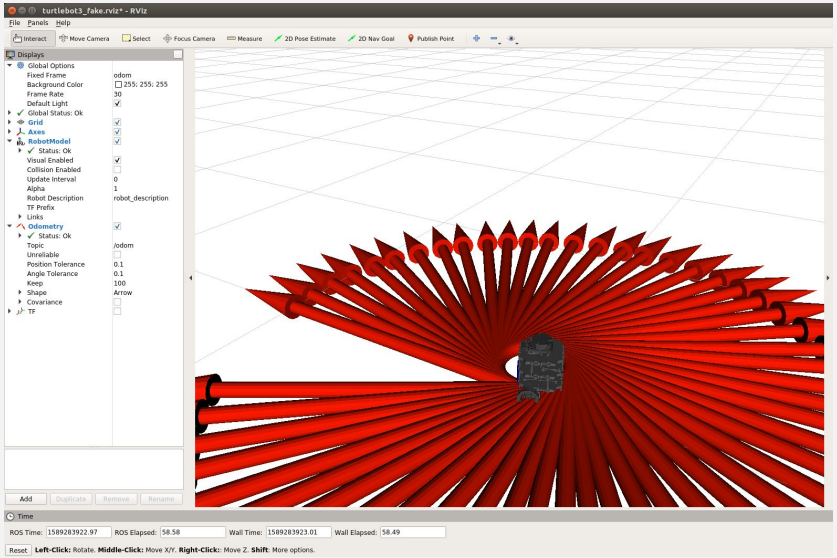
- `sudo apt update`
- `sudo apt install ros-melodic-turtlebot3`
- `cd /root/my_ros_ws/src`
- `git clone https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git`
- `source /opt/ros/melodic/setup.zsh`
- `cd /root/my_ros_ws && catkin_make`
- `source ./devel/setup.zsh`



TURTLEBOT SIMULATION. ЗАПУСК

http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/#turtlebot

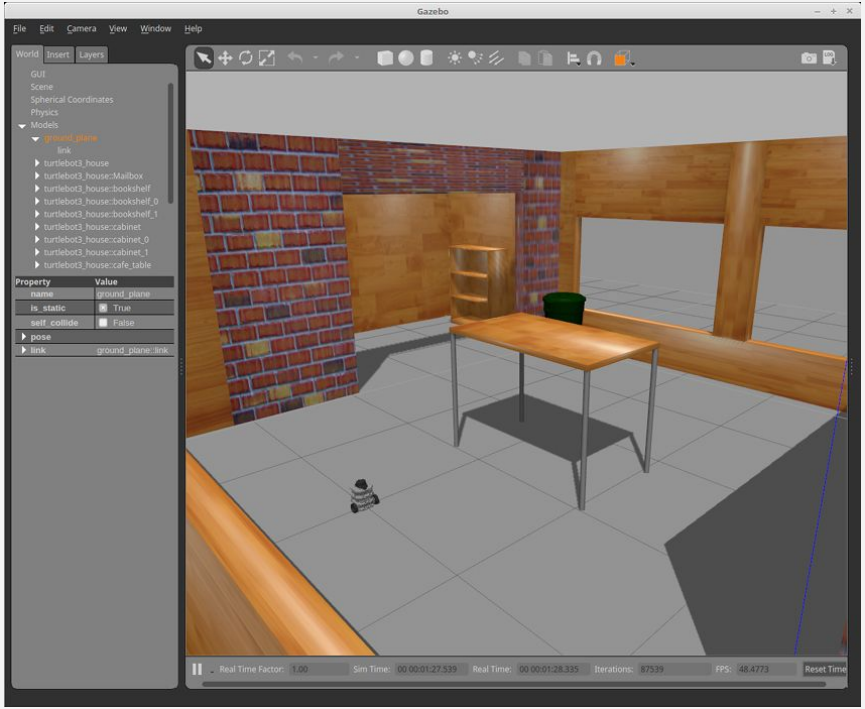
- `sudo apt update`
- В первом терминале:
  - `export TURTLEBOT3_MODEL=${TB3_MODEL}`
    - `${TB3_MODEL}`: **burger, waffle, waffle_pi**
  - `roslaunch turtlebot3_fake turtlebot3_fake.launch`
- В другом терминале:
  - `export TURTLEBOT3_MODEL=${TB3_MODEL}`
    - `${TB3_MODEL}`: **burger, waffle, waffle_pi**
  - `roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch`



TURTLEBOT SIMULATION. COLLISION AVOIDANCE

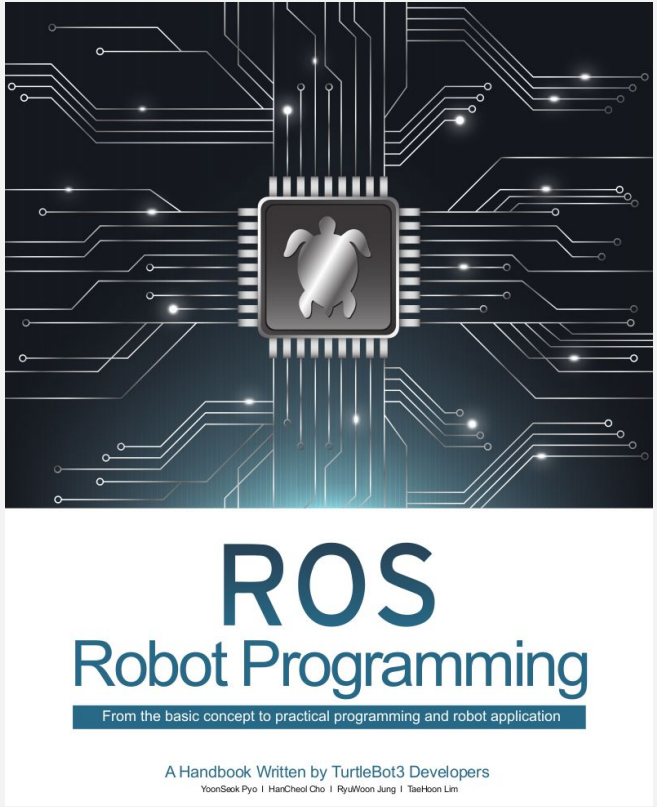
http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/#turtlebot

- В первом терминале:
  - `export TURTLEBOT3_MODEL=waffle_pi`
  - `roslaunch turtlebot3_gazebo turtlebot3_house.launch`
- В другом терминале:
  - `export TURTLEBOT3_MODEL=waffle_pi`
  - `roslaunch turtlebot3_gazebo turtlebot3_simulation.launch`
- В третьем:
  - `export TURTLEBOT3_MODEL=waffle_pi`
  - `roslaunch turtlebot3_gazebo turtlebot3_gazebo_rviz.launch`



ДОПОЛНИТЕЛЬНЫЕ ИСТОЧНИКИ

1. Книга: ROS Robot Programming.
   YoonSeok Pyo, HanCheol Cho, RyuWoon Jung, TaeHoon Lim (Eng)
2. Обучающие инструкции ROS (Eng)
3. Введение в ROS от Voltbro (Rus)
4. Clearpath Robotics ROS Tutorial (Eng)



ИНФОРМАЦИЯ О ПРЕЗЕНТАЦИИ

Эта презентация была подготовлена Олегом Шипитько в рамках курса “Моделирование колесных роботов” кафедры когнитивных технологий Московского физико-технического института (МФТИ). Автор выражает благодарность, авторам, чьи материалы были использованы в презентации. В случае, если вы обнаружили в презентации свои материалы, свяжитесь со мной, для включения в список авторов заимствованных материалов.

This presentation was prepared by Oleg Shipitko as part of the “Mobile Robotics” course at the Department of Cognitive Technologies, Moscow Institute of Physics and Technology. The author is grateful to the authors whose materials were used in the presentation. If you find your materials in a presentation, contact me to be included in the list of contributing authors.